Help

```
#include <stdlib.h>
#include <
href../../../common/math/cdo/cdo_math_h_src.pdfmath.h>
#include "
href../../../common/math/cdo/copulas_h_src.pdfcopulas.h"

typedef struct
{
  double              rho;
  double              g_rho;
  double              u_rho;
  double              factor;
} gaussian_params;

static double gaussian_density(const copula *cop, const double x)
{
  return pnl_normal_density(x);
}

static double *gaussian_compute_prob(const copula *cop, const double f_t)
{
  double              *result;
  gaussian_params     *p;
  double a;
  int i;
  p = cop->parameters;
  result = malloc(cop->size * sizeof(double));
  a = pnl_inv_cdfnor(f_t) / p->g_rho;
  for (i = 0; i < cop->size; i++)
    {
      result[i] = cdf_nor(a - p->u_rho * cop->points[i]);
    }

  return (result);
}

static void gaussian_generate(copula *cop)
{
  ((gaussian_params *)cop->parameters)->factor = pnl_rand_normal(0);
```

```
}

static int gaussian_compute_dt(const copula *cop, const step_fun *H, double *tim
{
  gaussian_params *p;
  double X;
  double zi;
  p = cop->parameters;
  X = p->rho * p->factor + p->g_rho * pnl_rand_normal(0);
  zi = -log(1. - cdf_nor(X));
  if (zi >= H->data[H->size - 1].y2) return (0);
  else
    {
      *time = inverse_sf(H, zi);
      return (1);
    }
}

copula *init_gaussian_copula(const double rho)
{
  copula              *cop;
  gaussian_params     *p;
  double              h;
  double              v0;
  int                 jv;

  cop = malloc(sizeof(copula));
  cop->name = "One-factor Gaussian Copula";
  cop->nfactor = 1;
  p = malloc(sizeof(gaussian_params));
  p->rho = rho;
  p->g_rho = sqrt(1.0 - rho * rho);
  p->u_rho = rho / p->g_rho;
  cop->parameters = p;
  cop->size = 200;
  cop->points = malloc(cop->size * sizeof(double));
  cop->weights = malloc(cop->size * sizeof(double));
  h = 24. / (cop->size - 1);
  for (jv = 0, v0 = -12.; jv < cop->size; jv++, v0 += h)
    {
      cop->points[jv] = v0;
```

```
      cop->weights[jv] = gaussian_density(cop, v0) * h;
    }
  cop->density = gaussian_density;
  cop->generate = gaussian_generate;
  cop->compute_default_time = gaussian_compute_dt;
  cop->compute_cond_prob = gaussian_compute_prob;

  return (cop);
}
```