```
   Help
#include <stdlib.h>
#include <math.h>
#include "copulas.h"


typedef struct
{
  double              rho;
  double              g_rho;
  double              u_rho;
  double              factor1;
  double              factor2;
  double              t1;

} Student_params;

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

double simulate_student (double t)
{
  double s = 0;
  int j;
  double u;
  for (j = 0; j < t; j++)
    {
      u = pnl_rand_normal (PNL_RNG_KNUTH);
      s += u * u;
    }
  return pnl_rand_normal (PNL_RNG_KNUTH) * sqrt (t / s);
}

static double simulate (double t)
{
  double s = 0;
  int j;
  double u;
```

```
  for (j = 0; j < t; j++)
    {
      u = pnl_rand_normal (PNL_RNG_KNUTH);
      s += u * u;
    }
  return t / s;
}

double student_cdf (double t, double x)
{
  int which = 1, status;
  double bound, p, q;
  pnl_cdf_t (&which, &p, &q, &x, &t, &status, &bound);
  if (status != 0)
    {
      printf ("error in pnl_cdf_t");
      abort();
    }
  return p;
}

double student_inv_cdf (double t, double p)
{
  int which = 2, status;
  double bound, q, x;
  if ( p <= 0. ) p = 0.001;
  if ( p >= 1. ) p = 0.999;
  q = 1. - p;
  pnl_cdf_t (&which, &p, &q, &x, &t, &status, &bound);
  if (status != 0)
    {
      printf ("error in pnl_cdf_t");
      abort();
    }
  return x;
}

static void Student_generate (copula *cop)
{
  Student_params *p;
  p = cop->parameters;
```

```
  ( (Student_params *) cop->parameters)->factor1 = pnl_rand
    _normal (0);
  ( (Student_params *) cop->parameters)->factor2 = sqrt (si
    mulate (p->t1) );
}

static double  Student_density (const copula *cop, const
    double x)
{
  Student_params *p;
  p = cop->parameters;
  return (tgamma ((p->t1 + 1) * 0.5) / ((tgamma ((p->t1) *
    0.5)) * sqrt (M_PI * (p->t1))
                                        * exp ((((p->t1) +
    1) * 0.5) * log (1 + x * x / (p->t1)))));
}

static double *Student_compute_prob (const copula *cop,
    const double f_t)
{
  double              *result;
  Student_params      *p;
  double              a;
  int                 i, j;
  int h = (cop->size) * (cop->size);
  p = cop->parameters;
  result = malloc (h * sizeof (double) );

  a = student_inv_cdf (p->t1, f_t) / (p->g_rho);

  for (i = 0; i < cop->size; i++)
    {
      for ( j = 0; j < cop->size; j++)
        {
          result[j + i * cop->size] = cdf_nor (a * sqrt (
    cop->points[i + cop->size] / p->t1)
                                               - (p->u_rho
    * cop->points[j]) );
        }
    }
  return (result);
```

```
}

static int Student_compute_dt (const copula *cop, const
    step_fun  *H, double *time)
{
  Student_params *p;
  double X;
  double zi;
  p = cop->parameters;
  X = (p->rho * p->factor1 + p->g_rho * pnl_rand_normal (0)
     ) * p->factor2;
  zi = -log (1. - student_cdf (p->t1, X) );
  if (zi >= H->data[H->size - 1].y2) return ( 0 );
  else
    {
      *time = inverse_sf (H, zi);
      return ( 1 );
    }
}

static double density_chi2 (double t1, double x)
{
  double a = 1 / (exp ((t1 * 0.5) * log (2.)) * tgamma (t1
    * 0.5));
  if (x > 0)
    {
      return a * exp ((t1 * 0.5 - 1) * log (x)) * exp (-x *
     0.5);
    }
  else return 0;
}

copula *init_student_copula (const double rho, const
    double t1)
{

/* Gauss quadrature weights and kronrod quadrature absciss
    ae and
   weights as evaluated with 80 decimal digit arithmetic by
   L. W. Fullerton, Bell Labs, Nov. 1981. */
```

```
/* abscissae of the 21-point kronrod rule */
 double xgk[12] =
 {
   0.0,
   0.995657163025808080735527280 70,
   0.973906528517171720079640121 0,
   0.930157491355708226001207180 10,
   0.865063366688984510732096688 40,
   0.780817726586416897063717578 30,
   0.679409568299024406234327365 10,
   0.562757134668604683339000099 30,
   0.433395394129247190799265943 20,
   0.294392862701460198131126603 10,
   0.148874338981631210884826001 10,
   0.0
 };

 /* weights of the 21-point gauss rule */
 double wgk[12] =
 {
   0.0,
   0.011694638867371874278064396060,
   0.032558162307964727478818972460,
   0.054755896574351996031381300240,
   0.075039674810919952767043140920,
   0.093125454583697605535065465080,
   0.109387158802297641899210590 30,
   0.123491976262065851077958109 80,
   0.134709217311473325928054001 80,
   0.142775938577060080797094273 10,
   0.147739104901338491374841516 00,
   0.149445554002916905664936468 40
 };

 copula             *cop;
 Student_params     *p;
 int                 jv;
 int                 b;
 double              a1 = -6.0;
 double              b1 = 6.0;
 double              a2 = 0.0;
```

```
double               b2 = 12.0;

cop = malloc (sizeof (copula));
cop->name = "Two-factor Student Copula";
cop->nfactor = 2;
p = malloc (sizeof (Student_params));
cop->parameters = p;
p->rho = rho;
p->g_rho = sqrt (1.0 - rho * rho);
p->u_rho = rho / p->g_rho;
p->t1 = t1;
cop->size = 22;
b = 2 * (cop->size);
cop->points = malloc (b  * sizeof (double));
cop->weights = malloc (b  * sizeof (double));

cop->points[0] = xgk[0];
cop->points[0 + cop->size] = xgk[0];
cop->weights[0] = wgk[0];
cop->weights[0 + cop->size] = wgk[0];
for (jv = 1; jv < 11; jv++)
  {

    cop->points[jv] = (a1 + b1) * 0.5 + (b1 - a1) * 0.5 *
   xgk[jv];
    cop->points[jv + 10] = (a1 + b1) * 0.5 - (b1 - a1) *
  0.5 * xgk[jv];
    cop->weights[jv] = wgk[jv] * gaussian_density (cop,
  cop->points[jv]) * (b1 - a1) * 0.5;
    cop->weights[jv + 10] = wgk[jv] * gaussian_density (
  cop, cop->points[jv + 10]) * (b1 - a1) * 0.5;

    cop->points[jv + cop->size] = (a2 + b2) * 0.5 + (b2 -
   a2) * 0.5 * xgk[jv];
    cop->points[jv + 10 + cop->size] = (a2 + b2) * 0.5 -
  (b2 - a2) * 0.5 * xgk[jv];

    cop->weights[jv + cop->size] = wgk[jv] * density_chi2
   (p->t1, cop->points[jv + cop->size] ) * (b2 - a2) * 0.5;
    cop->weights[jv + 10 + cop->size] = wgk[jv] * density
  _chi2 (p->t1, cop->points[jv + 10 + cop->size]) * (b2 - a2
```

```
    ) * 0.5;
    }

  cop->points[21] = (a1 + b1) * 0.5;
  cop->weights[21] = (b1 - a1) * 0.5 * wgk[11] * gaussian_d
    ensity (cop, cop->points[21]);

  cop->points[43] = (a2 + b2) * 0.5;
  cop->weights[43] = (b2 - a2) * 0.5 * wgk[11] * density_
    chi2 (p->t1, cop->points[43]);
  cop->density = Student_density;
  cop->generate = Student_generate;
  cop->compute_default_time = Student_compute_dt;
  cop->compute_cond_prob = Student_compute_prob;

  return (cop);
}
```

References