Help

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#include "pnl/pnl_random.h"
#include "company.h"


/* generation of constant recovery */
static double      generate_delta_cst(const void       *p_
    recovery)
{
  return ( *((double *) p_recovery) );
}


static dcomplex     phi_recov_cst(const double        u,
                                  const void          *p_
    recovery)
{
  double      *delta = (double *) p_recovery;
  double tmp = u * (1. - *delta);
  return Complex (cos ( tmp ), sin ( tmp ) );
}

static double      generate_delta_unif(const void      *p_
    recovery)
{
  params_recov_unif   *p = (params_recov_unif *) p_recovery
    ;
  double u = pnl_rand_uni_ab (p->a, p->b, 0);
  return u;
}

static dcomplex     phi_recov_unif(const double     u, cons
    t void         *p_recovery)
{
  params_recov_unif   *p = (params_recov_unif *) p_recovery
    ;
  dcomplex     z1;
```

```
  dcomplex     z2;
  dcomplex         result;

  if (u == 0) return Complex ( 1., 0. );
  z1 = Csub ( CIexp(u*(1-p->a)), CIexp(u*(1-p->b)) );
  z2 = Complex ( 0.0, u*(p->b-p->a) );
  result = Cdiv ( z1, z2 );
  return ( result );
}

static double      generate_delta_gauss(const void     *p_
    recovery)
{
  params_recov_gauss  *p = (params_recov_gauss *) p_reco
    very;

  return ( p->m + p->s * pnl_rand_normal(0) );
}

static dcomplex     phi_recov_gauss(const double        u,
                                    const void        *p_
    recovery)
{
  params_recov_gauss  *p = (params_recov_gauss *) p_reco
    very;

  return Cexp( Complex ( - 0.5 * pow(p->s*u,2), u*(1-p->m)
    ) );
}

company         *init_company(const double      nominal,
                              const int         intensity_
    n_step,
                              const double      *intensity_
    x,
                              const double      *intensity_
    h_x,
                              const double      mean_delta,
                              pfun_void_R       *generate_
    delta,
                              pfun_R_complex        *phi_
```

```
    recov,
                                void          *p_recovery)
{
  company     *comp;

  comp = malloc(sizeof(company));
  comp->nominal = nominal;
  comp->h = init_constant_sf(intensity_n_step, intensity_x,
     intensity_h_x);
  comp->H = integrate_sf(comp->h);
  comp->mean_delta = mean_delta;
  comp->generate_delta = generate_delta;
  comp->phi_recov = phi_recov;
  comp->p_recovery = p_recovery;

  return (comp);
}

company         *homogenize_company(const company   *comp,
                                    const double    new_nom
    inal,
                                    const double    new_de
    lta)
{
  company     *hcomp;
  double      *p_delta = malloc(sizeof(double));

  hcomp = malloc(sizeof(company));
  hcomp->nominal = new_nominal;
  hcomp->h = copy_sf(comp->h);
  hcomp->H = copy_sf(comp->H);
  hcomp->mean_delta = new_delta;
  hcomp->generate_delta = generate_delta_cst;
  hcomp->phi_recov = phi_recov_cst;
  *p_delta = new_delta;
  hcomp->p_recovery = p_delta;

  return (hcomp);
}

company         *init_company_cov_cst(const double      nom
```

```
    inal,
                                        const int
    intensity_n_step,
                                        const double      *
    intensity_x,
                                        const double      *
    intensity_h_x,
                                        const double      de
    lta)
{
  double      *p_delta = malloc(sizeof(double));

  *p_delta = delta;
  return (init_company(nominal, intensity_n_step,
    intensity_x, intensity_h_x, delta, generate_delta_cst, phi_recov_cs
    t, p_delta));
}

company         *init_company_cov_unif(const double     nom
    inal,
                                         const int
    intensity_n_step,
                                         const double      *
    intensity_x,
                                         const double      *
    intensity_h_x,
                                         const double      a,
                                         const double      b)
{
  params_recov_unif   *p = malloc(sizeof(params_recov_unif)
    );

  p->a = a;
  p->b = b;
  return (init_company(nominal, intensity_n_step,
    intensity_x, intensity_h_x, (a+b)*0.5, generate_delta_unif, phi_
    recov_unif, p));
}

company         *init_company_cov_gauss(const double
     nominal,
```

```
                                        const int
    intensity_n_step,
                                        const double
     *intensity_x,
                                        const double
     *intensity_h_x,
                                        const double
     m,
                                        const double
     s)
{
  params_recov_gauss  *p = malloc(sizeof(params_recov_
    gauss));

  p->m = m;
  p->s = s;
  return (init_company(nominal, intensity_n_step,
    intensity_x, intensity_h_x, m, generate_delta_gauss, phi_recov_
    gauss, p));
}

void            free_company(company            *co)
{
  free_step_fun(&(co->h));
  free_step_fun(&(co->H));
  free (co->p_recovery);
  free(co);

  return;
}
```

References