Help

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

#include "pnl/pnl_fft.h"
#include "pnl/pnl_specfun.h"
#include "pnl/pnl_finance.h"
#include "pnl/pnl_band_matrix.h"
#include "pnl/pnl_complex.h"
#include "pnl/pnl_matrix.h"
#include "pnl/pnl_mathtools.h"
#include "pde_tools.h"
#include "cgmy.h"
#include "time_change_levy.h"
#include "pnl/pnl_integration.h"

const double sinus_cardinal(double x)
{
  if (abs(x)>1e-8) return sin(x)/x;
  else
    {
      double x2,x4,x6,res=1;
      x2=x*x;
      res-=x2/6;
      x4=x2*x2;
      res+=x4/120;
      x6=x4*x2;
      res-=x6/5040;
      x6=x4*x4;
      res+=x6/362880;
      x6=x6*x2;
      res-=x6/39916800;
      return res;
      // return 1-pow(x,2)/6+pow(x,4)/120-pow(x,6)/5040+po
    w(x,8)/362880;
    }
}
```

```
// Use results on trigonometric function.
// Compute int_R sinc(u)^4 psi(u) exp( i u k ) du

void Levy_fourier_stiffness(PnlVectComplex *Levy_sinus,
    double hx,int bnd,int Nw,double hw,int kmin,int kmax,PnlVect *ro
    w_stiffness)
{
  PnlVectComplex *cos_sin_vect;
  int i,k,m;
  double tmp;
  cos_sin_vect=pnl_vect_complex_create(Nw);
  pnl_vect_resize(row_stiffness,kmax-kmin+1);
  tmp=-bnd*M_PI;
  for (i=0;i<Nw;i++)
    {
      pnl_vect_complex_set(cos_sin_vect,i,CIexp(tmp));
      tmp+=hw;
    }
  for(k=-1;k>=kmin;k--)
    {
      tmp=0;
      m=0;
      for (i=0;i<Nw;i++)
        {

          tmp+=GET_REAL(Levy_sinus,i)*GET_REAL(cos_sin_vec
    t,m)+GET_IMAG(Levy_sinus,i)*GET_IMAG(cos_sin_vect,m);
          m-=k;
          m=m%(Nw);
        }
      LET(row_stiffness,k-kmin)=tmp*hw*1/(M_2PI);
    }
  for(k=0;k<=kmax;k++)
    {
      tmp=0;
      m=0;
      for (i=0;i<Nw;i++)
        {
          tmp+=GET_REAL(Levy_sinus,i)*GET_REAL(cos_sin_vec
    t,m)-GET_IMAG(Levy_sinus,i)*GET_IMAG(cos_sin_vect,m);
```

```
          m+=k;
          m=m%Nw;

        }
      LET(row_stiffness,k-kmin)=tmp*hw*1/(M_2PI);
    }
  if(bnd%2==1)
    for(k=kmin;k<=kmax;k++)
      if(k-kmin%2==0)
        LET(row_stiffness,k-kmin)*=-1;

  pnl_vect_complex_free(&cos_sin_vect);
}


// ---------------  Levy_process --------------------

const dcomplex Levy_process_times_sinus_card(double u,
    Levy_process * mod,double hx,int Dupire)
{
  if(Dupire)
    return RCmul(pow(sinus_cardinal(u/2),4)*hx,
    Levy_process_characteristic_exponent(Complex(-u/hx,-1.),mod));

  return RCmul(pow(sinus_cardinal(u/2),4)*hx,
    Levy_process_characteristic_exponent(Complex(u/hx,0),mod));
}


// Use results on trigonometric function.
void Levy_process_fourier_stiffness(Levy_process * mod,
    double hx,double bnd_fourier,int Nw,int kmin,int kmax,int Dupi
    re,PnlVect *row_stiffness)
{
  PnlVectComplex *Levy_sinus;
  int i,bnd = ceil(bnd_fourier/M_PI);
  double tmp,hw;
  //printf("boundary % f {n",bnd*M_PI);
  Levy_sinus=pnl_vect_complex_create(Nw);
  hw=bnd*M_2PI/(Nw);
  tmp=-bnd*M_PI;
  for (i=0;i<Nw;i++)
    {
      pnl_vect_complex_set(Levy_sinus,i,Levy_process_times_
    sinus_card(tmp,mod,hx,Dupire));
      tmp+=hw;
    }
  Levy_fourier_stiffness(Levy_sinus,hx,bnd,Nw,hw,kmin,kmax,
    row_stiffness);
  //printf("sum of Row stiffness %e {n",pnl_vect_sum(row_
    stiffness));
  //pnl_vect_print(row_stiffness);
  pnl_vect_complex_free(&Levy_sinus);
}


// Test to compute fastly integral operator, not good resul
    t.

typedef struct {
  int k;
  int Dupire;
  double hx;
  Levy_process * Model;
} RFourierFunc ;

double RFourierFuncEvaluation_Obj(double w,
                                  RFourierFunc *Obj)
{
  dcomplex psi= Levy_process_times_sinus_card(w,Obj->Model,
    Obj->hx,Obj->Dupire);
  return psi.r*cos(w*Obj->k)-psi.i*sin(w*Obj->k);
}

double RFourierFuncEvaluation_Void(double w,
                                   void *Obj)
{
  return RFourierFuncEvaluation_Obj(w,(RFourierFunc*)Obj);
}


// Use results on trigonometric function.
```

```
void Levy_process_fourier_stiffness_0(Levy_process * mod,
    double hx,double bnd_fourier,int Nw,int kmin,int kmax,int Dupi
    re,PnlVect *row_stiffness)
{
  double abserr;
  int k,neval;
  RFourierFunc RF;
  PnlFunc Func;
  double A=12.56;
  double epsabs=1e-15;
  double epsrel=1e-15;
  RF.Dupire=Dupire;
  RF.hx=hx;
  RF.Model=mod;
  Func.params=&RF;
  Func.function=&RFourierFuncEvaluation_Void;
  pnl_vect_resize(row_stiffness,kmax-kmin+1);
  for(k=kmin;k<=kmax;k++)
    {
      RF.k=k;
      pnl_integration_GK(&Func,-A,A,epsabs,epsrel,&LET(row_
    stiffness,k-kmin),&abserr,&neval);
      LET(row_stiffness,k-kmin)/=M_2PI;
    }
  printf("sum of Row stiffness %e {n",pnl_vect_sum(row_sti
    ffness));
  pnl_vect_print(row_stiffness);
}

// ---------------  Levy_diffusion --------------------

const dcomplex Levy_diffusion_times_sinus_card(double u,
    double t,Levy_diffusion * mod,double hx,int Dupire)
{
  if(Dupire)
    return RCmul(pow(sinus_cardinal(u/2),4)*hx,
    Levy_diffusion_characteristic_exponent(Complex(-u/hx,-1.),t,mod));

  return RCmul(pow(sinus_cardinal(u/2),4)*hx,
    Levy_diffusion_characteristic_exponent(Complex(u/hx,0),t,mod));
}
```

```
// Use results on trigonometric function.
void Levy_diffusion_fourier_stiffness(Levy_diffusion * mod,
    double t,double hx,double bnd_fourier,int Nw,int kmin,int kmax,
    int Dupire,PnlVect *row_stiffness)
{
  PnlVectComplex *Levy_sinus;
  int i,bnd = ceil(bnd_fourier/M_PI);
  double tmp,hw;
  printf("boundary % f {n",bnd*M_PI);
  Levy_sinus=pnl_vect_complex_create(Nw);
  hw=bnd*M_2PI/(Nw);
  tmp=-bnd*M_PI;
  for (i=0;i<Nw;i++)
    {
      pnl_vect_complex_set(Levy_sinus,i,Levy_diffusion_time
    s_sinus_card(tmp,t,mod,hx,Dupire));
      tmp+=hw;
    }
  Levy_fourier_stiffness(Levy_sinus,hx,bnd,Nw,hw,kmin,kmax,
    row_stiffness);
  pnl_vect_complex_free(&Levy_sinus);
}

//----------------------------------------------------------
    ------------------
//                       Levy Gradient
//----------------------------------------------------------
    ------------------

// Use results on trigonometric function.
// Compute int_R sinc(u)^4 psi(u) exp( i u k ) du
void Levy_fourier_stiffness_gradient(PnlVectComplex *Levy_s
    inus,
                                     double hx,
                                     int bnd,
                                     int Nw,
                                     int grad_size,
                                     double hw,
                                     int kmin,
                                     int kmax,
                                     PnlVect *row_stiffnes
    s)
{
  PnlVectComplex *cos_sin_vect;
  int i,k,m,j;
  double tmp1;
  int bound=kmax-kmin+1;
  cos_sin_vect=pnl_vect_complex_create(Nw);
  pnl_vect_resize(row_stiffness,grad_size*(kmax-kmin+1));
  tmp1=-bnd*M_PI;
  for (i=0;i<Nw;i++)
    {
      pnl_vect_complex_set(cos_sin_vect,i,CIexp(tmp1));
      tmp1+=hw;
    }
  pnl_vect_set_double(row_stiffness,0.0);
  for(k=-1;k>=kmin;k--)
    {
      m=0;
      for (i=0;i<Nw;i++)
        {

          for(j=0;j<grad_size;j++)
            {
              //LET(row_stiffness,j+grad_size*(k-kmin))
                LET(row_stiffness,bound*j+(k-kmin))+=GET_
    REAL(Levy_sinus,j+i*grad_size)*GET_REAL(cos_sin_vect,m)+
                  GET_IMAG(Levy_sinus,j+i*grad_size)*GET_
    IMAG(cos_sin_vect,m);
            }
          m-=k;
          m=m%(Nw);
        }
      //for(j=0;j<grad_size;j++)
      //  LET(row_stiffness,j+grad_size*(k-kmin))*=hw*1/(M_
    2PI);
    }
  for(k=0;k<=kmax;k++)
    {
      m=0;
      for (i=0;i<Nw;i++)
        {
          for(j=0;j<grad_size;j++)
            {
              //LET(row_stiffness,j+grad_size*(k-kmin))
              LET(row_stiffness,bound*j+(k-kmin))+=GET_REA
    L(Levy_sinus,j+i*grad_size)*GET_REAL(cos_sin_vect,m)
                -GET_IMAG(Levy_sinus,j+i*grad_size)*GET_IMA
    G(cos_sin_vect,m);
            }
          m+=k;
          m=m%Nw;

        }
      //for(j=0;j<grad_size;j++)
      //      LET(row_stiffness,j+grad_size*(k-kmin))*=hw*1
    /(M_2PI);
    }

  if(bnd%2==1)
    for(j=0;j<grad_size;j++)
      for(k=kmin;k<=kmax;k++)
        LET(row_stiffness,bound*j+(k-kmin))*=((k-kmin%2==0)
    ?-1:1)*hw*1/(M_2PI);
  else
    for(j=0;j<grad_size;j++)
      for(k=kmin;k<=kmax;k++)
        LET(row_stiffness,bound*j+(k-kmin))*=hw*1/(M_2PI);
  /*
    for(k=kmin;k<=kmax;k++)
      if(k-kmin%2==0)
      for(j=0;j<grad_size;j++)
        LET(row_stiffness,j+grad_size*(k-kmin))*=-1;
  */
  pnl_vect_complex_free(&cos_sin_vect);
}
```

```
void Levy_process_times_sinus_card_gradient(PnlVectComplex
    * Gradient,double u,Levy_process_gradient * mod,double hx,
    int Dupire)
{

  if(Dupire)
    Levy_process_gradient_characteristic_exponent(Gradient,
    Complex(-u/hx,-1.),mod);
  else Levy_process_gradient_characteristic_exponent(Gradie
    nt,Complex(u/hx,0),mod);
  pnl_vect_complex_mult_double(Gradient,pow(sinus_cardinal(
    u/2),4)*hx);
}

// Use results on trigonometric function.
void Levy_process_fourier_stiffness_gradient(Levy_process_g
    radient * mod,
                                          double hx,
                                          double bnd_fo
    urier,
                                          int Nw,
                                          int kmin,
                                          int kmax,
                                          int Dupire,
                                          PnlVect *row_
    stiffness)
{
  PnlVectComplex gradient;
  PnlVectComplex *Levy_sinus;
  int i,bnd = ceil(bnd_fourier/M_PI);
  double tmp,hw;
  //printf("boundary % f {n",bnd*M_PI);
  Levy_sinus=pnl_vect_complex_create(mod->grad_size*Nw);
  hw=bnd*M_2PI/(Nw);
  tmp=-bnd*M_PI;
  for (i=0;i<Nw;i++)
    {
      gradient=pnl_vect_complex_wrap_subvect(Levy_sinus,i*
    mod->grad_size,mod->grad_size);
      Levy_process_times_sinus_card_gradient(&gradient,tmp,
    mod,hx,Dupire);
      tmp+=hw;
    }
  Levy_fourier_stiffness_gradient(Levy_sinus,hx,bnd,Nw,mod-
    >grad_size,hw,kmin,kmax,row_stiffness);
  //printf("sum of Row stiffness %e {n",pnl_vect_sum(row_
    stiffness));
  //pnl_vect_print(row_stiffness);
  pnl_vect_complex_free(&Levy_sinus);
}
```

References