```
   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 "levy_process.h"
#include "levy_diffusion.h"

#define GETPROCESSPARAMETER(v,i){              {
  if (i>=v->nb_parameters || i<0){           {
      perror("index out of range"); abort();}{
  else{return ((double *)v)[i];}}

#define SETPROCESSPARAMETER(v,i,a){   {
  if (i>=v->nb_parameters || i<0){           {
   perror("index out of range"); abort();}   {
  else{((double *)v)[i]=a;}}

#define GETLEVYPARAMETER(v,i){           {
  if (i>=v->nb_parameters || i<0){           {
      perror("index out of range"); abort();}{
  else{return ((double *)v->process)[i];}}

#define SETLEVYPARAMETER(v,i,a){  {
  if (i>=v->nb_parameters || i<0){           {
   perror("index out of range"); abort();}   {
  else{((double *)v->process)[i]=a;}}


#define IMPLICIT_VOL 0.01
#define NB_PARAM_HESTON 5
#define NB_PARAM_BATES 8
```

```
#define NB_PARAM_BNS 5
#define NB_PARAM_DPS 15
#define NB_PARAM_CIR 4
#define NB_PARAM_GAMMA 4
#define NB_PARAM_CIRVG 7
#define NB_PARAM_GAMMAVG 7



// ------------------------------ Heston_diffusion  --------
    ---------------

static dcomplex Heston_diffusion_characteristic_exponent_
    without_cast(dcomplex u,double t,Heston_diffusion * mod)
{
  dcomplex u_sqr_plus_i_u,kmrho,d,g,emdt,gemdt,demdt,onemg,
    onemgemdt,psi;
  u_sqr_plus_i_u=Cmul(u,(Cadd(u,CI)));
  kmrho=Complex(mod->Kappa+mod->rho_theta*u.i,-mod->rho_th
    eta*u.r);
  d=Cadd(Cmul(kmrho,kmrho),RCmul(mod->theta_sqr,u_sqr_plus_
    i_u));
  d=Csqrt(d);
  g=Cdiv(Csub(kmrho,d),Cadd(kmrho,d));
  kmrho=Csub(kmrho,d);
  emdt=Cexp(RCmul(-t,d));
  gemdt=Cmul(g,emdt);
  demdt=Cmul(d,emdt);
  onemg=Complex(1-g.r,-g.i);
  onemgemdt=Complex(1-gemdt.r,-gemdt.i);
  psi=Cdiv(Cmul(onemg,demdt),Cmul(onemgemdt,onemgemdt));
  psi=Complex(1+mod->sigma_sqr_d_eta_kappa*psi.r,+mod->si
    gma_sqr_d_eta_kappa*psi.i);
  psi=Cmul(kmrho,psi);
  psi=Csub(psi,RCmul(2,Cdiv(Cmul(g,demdt),onemgemdt)));
  psi= RCmul(-mod->etakappathetam2,psi);
  return psi;
}
dcomplex Heston_diffusion_characteristic_exponent(dcomplex
    u,double t,void * mod)
{
```

```
  return Heston_diffusion_characteristic_exponent_without_
    cast(u,t,(Heston_diffusion *) mod);
}

dcomplex Heston_diffusion_ln_characteristic_function_withou
    t_cast(dcomplex u,double t,Heston_diffusion * mod)
{
  dcomplex u_sqr_plus_i_u,kmrho,d,g,emdt,gemdt,onemg,onem
    gemdt,psi;
  u_sqr_plus_i_u=Cmul(u,(Cadd(u,CI)));
  kmrho=Complex(mod->Kappa+mod->rho_theta*u.i,-mod->rho_th
    eta*u.r);
  d=Cadd(Cmul(kmrho,kmrho),RCmul(mod->theta_sqr,u_sqr_plus_
    i_u));
  d=Csqrt(d);
  g=Cdiv(Csub(kmrho,d),Cadd(kmrho,d));
  kmrho=Csub(kmrho,d);
  emdt=Cexp(RCmul(-t,d));
  gemdt=Cmul(g,emdt);
  onemg=Complex(1-g.r,-g.i);
  onemgemdt=Complex(1-gemdt.r,-gemdt.i);
  psi=Cdiv(Complex(1-emdt.r,-emdt.i),onemgemdt);
  psi=RCmul(mod->sigma_sqr_d_eta_kappa,psi);
  psi=RCadd(t,psi);
  psi=Cmul(kmrho,psi);
  psi=Csub(psi,RCmul(2,Clog(Cdiv(onemgemdt,onemg))));
  psi= RCmul(mod->etakappathetam2,psi);
  return psi;
}

dcomplex Heston_diffusion_ln_characteristic_function(dcompl
    ex u,double t,void * mod)
{
  return Heston_diffusion_ln_characteristic_function_withou
    t_cast(u,t,(Heston_diffusion *) mod);
}

Heston_diffusion * Heston_diffusion_create(double Eta_,
    double Kappa_,double Rho_,
                                           double Theta_,
    double Sigma_,
```

```
                                                double *jump_dr
    ift)
{
  Heston_diffusion * process = malloc(sizeof(Heston_diffus
    ion));
  process->nb_parameters=NB_PARAM_HESTON;
  process->Eta=Eta_;
  process->Kappa=Kappa_;
  process->Rho=Rho_;
  process->Theta=Theta_;
  process->Sigma=Sigma_;
  process->sigma_sqr=Sigma_*Sigma_;
  process->theta_sqr=Theta_*Theta_;
  process->sigma_sqr_d_eta_kappa=process->sigma_sqr/(Eta_*
    Kappa_);
  process->etakappathetam2=(Eta_*Kappa_)/process->theta_sq
    r;
  process->rho_theta=Rho_*Theta_;
  process->Drift=0;
  //>> Two way to compute drift term due to jump,
  //>> Put on Band matrix
  (*jump_drift)= 0;
  //>> Or Put in FD scheme (comment previous line and un
    comment to next line)
  //  (*jump_drift)= -process->C_Gamma_minus_Alpha_Minus*
    process->Lambdap1powAlphaMinus;
  //process->Lambdap1powAlphaMinus=0.0;
  return process;
};

void Heston_diffusion_update(Heston_diffusion * process)
{
  process->sigma_sqr=process->Sigma*process->Sigma;
  process->theta_sqr=process->Theta*process->Theta;
  process->sigma_sqr_d_eta_kappa=process->sigma_sqr/(proces
    s->Eta*process->Kappa);
  process->etakappathetam2=(process->Eta*process->Kappa)/
    process->theta_sqr;
  process->rho_theta=process->Rho*process->Theta;
  process->Drift=0;
};
```

```
Heston_diffusion * Heston_diffusion_create_from_vect(const
    PnlVect * input)
{
  int i;
  Heston_diffusion * process = malloc(sizeof(Heston_diffus
    ion));
  process->nb_parameters=NB_PARAM_HESTON;
  for(i=0;i<process->nb_parameters;i++)
    SETPROCESSPARAMETER(process,i,GET(input,i));
  Heston_diffusion_update(process);
  return process;
};

void Heston_diffusion_list(const Heston_diffusion * proces
    s)
{
  printf(" Eta= %7.4f {n kappa = %7.4f {n rho = %7.4f {n th
    eta =  %7.4f {n sigma_0 =  %7.4f {n ",process->Eta,process-
    >Kappa,process->Rho,process->Theta,process->Sigma);
}


// ----------------------------- Bates_diffusion  ---------
    --------------

dcomplex Bates_diffusion_characteristic_exponent_without_
    cast(dcomplex u,double t,Bates_diffusion * mod)
{
  dcomplex u_sqr_plus_i_u,kmrho,d,g,emdt,gemdt,demdt,onemg,
    onemgemdt,psi,psi_J;
  u_sqr_plus_i_u=Cmul(u,(Cadd(u,CI)));
  kmrho=Complex(mod->Kappa+mod->rho_theta*u.i,-mod->rho_th
    eta*u.r);
  d=Cadd(Cmul(kmrho,kmrho),RCmul(mod->theta_sqr,u_sqr_plus_
    i_u));
  d=Csqrt(d);
  g=Cdiv(Csub(kmrho,d),Cadd(kmrho,d));
  kmrho=Csub(kmrho,d);
  emdt=Cexp(RCmul(-t,d));
```

```
  gemdt=Cmul(g,emdt);
  demdt=Cmul(d,emdt);
  onemg=Complex(1-g.r,-g.i);
  onemgemdt=Complex(1-gemdt.r,-gemdt.i);
  psi=Cdiv(Cmul(onemg,demdt),Cmul(onemgemdt,onemgemdt));
  psi=Complex(1+mod->sigma_sqr_d_eta_kappa*psi.r,+mod->si
    gma_sqr_d_eta_kappa*psi.i);
  psi=Cmul(kmrho,psi);
  psi=Csub(psi,RCmul(2,Cdiv(Cmul(g,demdt),onemgemdt)));
  psi= RCmul(-mod->etakappathetam2,psi);
  // Jump part
  psi_J=RCmul(-mod->sigmaj_sqr_demi,u_sqr_plus_i_u);
  psi_J=C_op_apib(psi_J,RCmul(mod->lnonepmuj,u));
  psi_J=RCadd(-1,Cexp(psi_J));
  psi =Csub(psi,RCmul(mod->Lambda_J,psi_J));
  psi=C_op_apib(psi,CRmul(u,mod->Drift));
  return psi;
}
dcomplex Bates_diffusion_characteristic_exponent(dcomplex
    u,double t,void * mod)
{
  return Bates_diffusion_characteristic_exponent_without_
    cast(u,t,(Bates_diffusion *) mod);
}

dcomplex Bates_diffusion_ln_characteristic_function_withou
    t_cast(dcomplex u,double t,Bates_diffusion * mod)
{
  dcomplex u_sqr_plus_i_u,kmrho,d,g,emdt,gemdt,onemg,onem
    gemdt,psi,psi_J;
  u_sqr_plus_i_u=Cmul(u,(Cadd(u,CI)));
  kmrho=Complex(mod->Kappa+mod->rho_theta*u.i,-mod->rho_th
    eta*u.r);
  d=Cadd(Cmul(kmrho,kmrho),RCmul(mod->theta_sqr,u_sqr_plus_
    i_u));
  d=Csqrt(d);
  g=Cdiv(Csub(kmrho,d),Cadd(kmrho,d));
  kmrho=Csub(kmrho,d);
  emdt=Cexp(RCmul(-t,d));
  gemdt=Cmul(g,emdt);
  onemg=Complex(1-g.r,-g.i);
```

```
  onemgemdt=Complex(1-gemdt.r,-gemdt.i);
  psi=Cdiv(Complex(1-emdt.r,-emdt.i),onemgemdt);
  psi=RCmul(mod->sigma_sqr_d_eta_kappa,psi);
  psi=RCadd(t,psi);
  psi=Cmul(kmrho,psi);
  psi=Csub(psi,RCmul(2,Clog(Cdiv(onemgemdt,onemg))));
  psi= RCmul(mod->etakappathetam2,psi);
  // Jump part
  psi_J=RCmul(-mod->sigmaj_sqr_demi,Cmul(u,u));
  psi_J=C_op_apib(psi_J,RCmul(mod->mu_J,u));
  psi_J=RCadd(-1,Cexp(psi_J));
  psi =Cadd(psi,RCmul(mod->Lambda_J*t,psi_J));
  psi=C_op_amib(psi,CRmul(u,mod->Drift*t));
  return psi;
}



dcomplex Bates_diffusion_ln_characteristic_function(dcompl
    ex u,double t,void * mod)
{
  return Bates_diffusion_ln_characteristic_function_withou
    t_cast(u,t,(Bates_diffusion *) mod);
}



Bates_diffusion * Bates_diffusion_create(double Eta_,
    double Kappa_,double Rho_,
                                          double Theta_,
    double Sigma_,
                                          double mu_J_,
                                          double Sigma_J_,
    double Lambda_J_,double *jump_drift)
{
  Bates_diffusion * process = malloc(sizeof(Bates_diffusio
    n));
  process->nb_parameters=NB_PARAM_BATES;
  process->Eta=Eta_;
  process->Kappa=Kappa_;
  process->Rho=Rho_;
  process->Theta=Theta_;
```

```
  process->Sigma=Sigma_;
  process->sigma_sqr=Sigma_*Sigma_;
  process->theta_sqr=Theta_*Theta_;
  process->sigma_sqr_d_eta_kappa=process->sigma_sqr/(Eta_*
    Kappa_);
  process->etakappathetam2=(Eta_*Kappa_)/process->theta_sq
    r;
  process->rho_theta=Rho_*Theta_;
  process->mu_J=mu_J_;
  process->Sigma_J=Sigma_J_;
  process->Lambda_J=Lambda_J_;
  process->sigmaj_sqr_demi=0.5*Sigma_J_*Sigma_J_;
  process->Drift=Lambda_J_*(exp(mu_J_+process->sigmaj_sqr_
    demi)-1.0);
  //>> Two way to compute drift term due to jump,
  //>> Put on Band matrix
  (*jump_drift)= 0;
  //>> Or Put in FD scheme (comment previous line and un
    comment to next line)
  //  (*jump_drift)= -process->C_Gamma_minus_Alpha_Minus*
    process->Lambdap1powAlphaMinus;
  //process->Lambdap1powAlphaMinus=0.0;
  return process;
}

void Bates_diffusion_update(Bates_diffusion * process)
{
  process->sigma_sqr=process->Sigma*process->Sigma;
  process->theta_sqr=process->Theta*process->Theta;
  process->sigma_sqr_d_eta_kappa=process->sigma_sqr/(proces
    s->Eta*process->Kappa);
  process->etakappathetam2=(process->Eta*process->Kappa)/
    process->theta_sqr;
  process->rho_theta=process->Rho*process->Theta;
  process->sigmaj_sqr_demi=0.5*process->Sigma_J*process->Si
    gma_J;
  process->Drift=process->Lambda_J*(exp(process->mu_J+proc
    ess->sigmaj_sqr_demi)-1.0);
  //>> Two way to compute drift term due to jump,
  //>> Put on Band matrix
};
```

```
Bates_diffusion * Bates_diffusion_create_from_vect(const Pn
    lVect * input)
{
  int i;
  Bates_diffusion * process = malloc(sizeof(Bates_diffusio
    n));
  process->nb_parameters=NB_PARAM_BATES;
  for(i=0;i<process->nb_parameters;i++)
    SETPROCESSPARAMETER(process,i,GET(input,i));
  Bates_diffusion_update(process);
  return process;
};
void Bates_diffusion_list(const Bates_diffusion * process)
{
  printf(" Eta= %7.4f {n kappa = %7.4f {n rho = %7.4f {n th
    eta =  %7.4f {n sigma_0 =  %7.4f {n , mu_J = %7.4f {n , Si
    gma_J= %7.4f {n , Lambda_J = %7.4f{n",process->Eta,process->
    Kappa,process->Rho,process->Theta,process->Sigma,process->
    mu_J,process->Sigma_J,process->Lambda_J);
}


// ------------------------------ BNS_diffusion  -----------
    ------------

dcomplex BNS_diffusion_characteristic_exponent_without_cas
    t(dcomplex u,double t,BNS_diffusion * mod)
{
  //>> Case 1 code infinitesimal generator of backward k=i-j
  // Result is not correct compare to differenciation of ln
    _phi,
  // formula to be check ...
  dcomplex u_sqr_plus_i_u,f1,f2,f2mb,f1mb,df1;
  PNL_ERROR(" fonction BNS_diffusion_characteristic_expon
    ent_without_cast does not return good result","time_change_levy.c
    ");
  u_sqr_plus_i_u=Cmul(u,(Cadd(u,CI)));
  f2=RCmul(-0.5*mod->Lambda_m1,u_sqr_plus_i_u);
  f1= RCmul(1-exp(-mod->Lambda*t),f2);
  f2=C_op_apib(f2,RCmul(mod->Rho,u));
```

```
  df1=RCmul(0.5*exp(-mod->Lambda*t),u_sqr_plus_i_u);
  f1=C_op_apib(f1,RCmul(mod->Rho,u));
  f2mb=RCadd(-mod->Beta,f2);
  f1mb=RCadd(-mod->Beta,f1);
  df1=Cmul(df1,RCadd(mod->Sigma0_sqr,RCdiv(mod->Alpha*mod->
    Beta,Cmul(f1mb,f2mb))));
  df1=Cadd(df1,RCmul(mod->Alpha*mod->Lambda,Cdiv(f2,f2mb)))
    ;
  f1=C_op_apib(df1,CRmul(u,t*mod->Drift));
  return f1;
}

dcomplex BNS_diffusion_characteristic_exponent(dcomplex u,
    double t,void * mod)
{
  return BNS_diffusion_characteristic_exponent_without_cas
    t(u,t,(BNS_diffusion *) mod);
}

dcomplex BNS_diffusion_ln_characteristic_function_without_
    cast(dcomplex u,double t,BNS_diffusion * mod)
{
  dcomplex u_sqr_plus_i_u,f1,f2,f2mb,f1mb,iurhomb,tf1;
  u_sqr_plus_i_u=Cmul(u,(Cadd(u,CI)));
  f2=RCmul(-0.5*mod->Lambda_m1,u_sqr_plus_i_u);
  f1= RCmul(1-exp(-mod->Lambda*t),f2);
  f2=C_op_apib(f2,RCmul(mod->Rho,u));
  tf1=f1;
  f1=C_op_apib(f1,RCmul(mod->Rho,u));
  f2mb=RCadd(-mod->Beta,f2);
  f1mb=RCadd(-mod->Beta,f1);
  iurhomb=Complex(-mod->Beta-mod->Rho*u.i,mod->Rho*u.r);

  f1mb=RCmul(mod->Beta,Clog(Cdiv(f1mb,iurhomb)));
  f2=Cadd(RCmul(mod->Lambda*t,f2),f1mb);
  f2=RCmul(-mod->Alpha,Cdiv(f2,f2mb));
  f1=RCmul(mod->Sigma0_sqr,tf1);
  f1=Cadd(f1,f2);
  f1=C_op_amib(f1,CRmul(u,t*mod->Drift));
  return f1;
}
```

```
dcomplex BNS_diffusion_ln_characteristic_function(dcomplex
    u,double t,void * mod)
{return BNS_diffusion_ln_characteristic_function_without_
    cast(u,t,(BNS_diffusion *) mod);}

BNS_diffusion * BNS_diffusion_create(double Lambda_,double
    Rho_,
                                     double Beta_,double
    Alpha_,
                                     double Sigma0_,double
    *jump_drift)
{
  BNS_diffusion * process = malloc(sizeof(BNS_diffusion));
  process->nb_parameters=NB_PARAM_BNS;
  process->Lambda=Lambda_;
  process->Rho=Rho_;
  process->Beta=Beta_;
  process->Alpha=Alpha_;
  process->Sigma0=Sigma0_;
  process->Sigma0_sqr=Sigma0_*Sigma0_;
  process->Lambda_m1=1./Lambda_;
  process->Drift=Alpha_*Lambda_*Rho_/(Beta_-Rho_);
  //>> Two way to compute drift term due to jump,
  //>> Put on Band matrix
  (*jump_drift)= 0;
  //>> Or Put in FD scheme (comment previous line and un
    comment to next line)
  //  (*jump_drift)= -process->C_Gamma_minus_Alpha_Minus*
    process->Lambdap1powAlphaMinus;
  //process->Lambdap1powAlphaMinus=0.0;
  return process;
};

void BNS_diffusion_update(BNS_diffusion * process)
{
  process->Sigma0_sqr=process->Sigma0*process->Sigma0;
  process->Lambda_m1=1./process->Lambda;
  process->Drift=process->Alpha*process->Lambda*process->Rh
    o/(process->Beta-process->Rho);
  //>> Two way to compute drift term due to jump,
```

```
  //>> Put on Band matrix
};

BNS_diffusion * BNS_diffusion_create_from_vect(const PnlVec
    t * input)
{
  int i;
  BNS_diffusion * process = malloc(sizeof(BNS_diffusion));
  process->nb_parameters=NB_PARAM_BNS;
  for(i=0;i<process->nb_parameters;i++)
    SETPROCESSPARAMETER(process,i,GET(input,i));
  BNS_diffusion_update(process);
  return process;
};
void BNS_diffusion_list(const BNS_diffusion * process)
{printf(" Lambda= %7.4f {n Rho = %7.4f {n Beta = %7.4f {n
    Alpha =  %7.4f {n sigma_0 =  %7.4f {n ",process->Lambda,proc
    ess->Rho,process->Beta,process->Alpha,process->Sigma0);}

// ----------------------------- DPS_diffusion  -----------
    ------------

dcomplex DPS_diffusion_characteristic_exponent_without_cas
    t(dcomplex u,double t,DPS_diffusion * mod)
{
  return CZERO;
}
dcomplex DPS_diffusion_characteristic_exponent(dcomplex u,
    double t,void * mod)
{
  return DPS_diffusion_characteristic_exponent_without_cas
    t(u,t,(DPS_diffusion *) mod);
}

dcomplex function_jump_variance(dcomplex a, dcomplex b,dcom
    plex gam, dcomplex onememdt,double t)

{
  dcomplex psi_v,diff;
  psi_v=Csub(b,a);
  psi_v=Cdiv(RCmul(-2.0,a),Csub(Cmul(gam,gam),Cmul(psi_v,ps
```

```
    i_v)));
  psi_v=Cmul(psi_v,Clog(RCsub(1,Cmul(Cdiv(Csub(Cadd(gam,b),
    a),RCmul(2.0,gam)),onememdt))));
  diff=Csub(gam,b);
  diff=Cdiv(diff,Cadd(diff,a));
  return Cadd(RCmul(t,diff),psi_v);
}

dcomplex DPS_diffusion_ln_characteristic_function_without_
    cast(dcomplex u,double t,DPS_diffusion * mod)
{

  dcomplex a,b,c,d,gam,onememdt,beta,alpha_0,psi_y,psi_v,ps
    i_c,psi;
  a=Cmul(u,(Cadd(u,CI)));
  b=Complex(-mod->Kappa-mod->rho_theta*u.i,mod->rho_theta*
    u.r);
  c=Complex(1+mod->rho_j*mod->mu_cv*u.i,-mod->rho_j*mod->mu
    _cv*u.r);
  gam=Csqrt(Cadd(Cmul(b,b),RCmul(mod->theta_sqr,a)));
  onememdt=Cexp(RCmul(-t,gam));
  onememdt=Complex(1.0-onememdt.r,-onememdt.i);

  beta=Cdiv(Cmul(a,onememdt),Csub(RCmul(2.,gam),Cmul(Cadd(
    gam,b),onememdt)));

  alpha_0=RCmul(-2.0*mod->etakappathetam2,Cadd(RCmul(0.5*t,
    Cadd(gam,b)),Clog(RCsub(1,Cmul(Cdiv(Cadd(gam,b),RCmul(2.0,
    gam)),onememdt)))));

  psi_y=RCmul(-mod->Sigma_y_sqr_demi,Cmul(u,u));
  psi_y=C_op_apib(psi_y,RCmul(mod->mu_y,u));
  psi_y=RCadd(-1,Cexp(psi_y));
  psi_y=RCmul(mod->Lambda_y*t,psi_y);

  psi_v=CRsub(function_jump_variance(RCmul(mod->mu_v,a),b,
    gam,onememdt,t),t);
  psi_v=RCmul(mod->Lambda_v,psi_v);
  // Seems to be error in calcul of 'd' page 24 of the paper
  // f^c (u,{tau) = exp(...) d
  // d = ({gamma-b)/((gamma-b)c-mu_{cv,a}) + ...
```

```
  // ->
  // d = (({gamma-b)c)/((gamma-b)c-mu_{cv,a}) + ...

  d=function_jump_variance(RCmul(mod->mu_cv,a),Cmul(b,c),Cm
    ul(gam,c),onememdt,t);
  psi_c=RCmul(-mod->sigma_cy_sqr_demi,Cmul(u,u));
  psi_c=C_op_apib(psi_c,RCmul(mod->mu_cy,u));
  psi_c=Cmul(d,Cexp(psi_c));
  psi_c=RCmul(mod->Lambda_c,Complex(psi_c.r-t,psi_c.i));

  alpha_0=Cadd(psi_y,alpha_0);
  alpha_0=Cadd(psi_v,alpha_0);
  alpha_0=Cadd(psi_c,alpha_0);

  psi=Cadd(alpha_0,RCmul(-mod->sigma_sqr,beta));
  psi=C_op_amib(psi,CRmul(u,mod->Drift*t));
  return psi;
}



dcomplex DPS_diffusion_ln_characteristic_function(dcomplex
    u,double t,void * mod)
{
  return DPS_diffusion_ln_characteristic_function_without_
    cast(u,t,(DPS_diffusion *) mod);
}


DPS_diffusion * DPS_diffusion_create(double Eta_,double Ka
    ppa_,double Rho_,
                                     double Theta_,double
    Sigma_,
                                     double mu_y_,
                                     double Sigma_y_,
                                     double Lambda_y_,
                                     double mu_v_,
                                     double Lambda_v_,
                                     double mu_cy_,
                                     double Sigma_cy_,
                                     double mu_cv_,
```

```
                                    double Lambda_c_,
                                    double rho_j_,
                                    double *jump_drift)
{
  DPS_diffusion * process = malloc(sizeof(DPS_diffusion));
  process->nb_parameters=NB_PARAM_DPS;
  process->Eta=Eta_;
  process->Kappa=Kappa_;
  process->Rho=Rho_;
  process->Theta=Theta_;
  process->Sigma=Sigma_;
  process->sigma_sqr=Sigma_*Sigma_;
  process->theta_sqr=Theta_*Theta_;
  process->sigma_sqr_d_eta_kappa=process->sigma_sqr/(Eta_*
    Kappa_);
  process->etakappathetam2=(Eta_*Kappa_)/process->theta_sq
    r;
  process->rho_theta=Rho_*Theta_;

  process->mu_y=mu_y_;
  process->Sigma_y_sqr_demi=0.5*Sigma_y_*Sigma_y_;
  process->Lambda_y=Lambda_y_;

  process->mu_v=mu_v_;
  process->Lambda_v=Lambda_v_;

  process->sigma_cy_sqr_demi=0.5*Sigma_cy_*Sigma_cy_;
  process->mu_cy=mu_cy_;
  process->mu_cv=mu_cv_;
  process->Lambda_c=Lambda_c_;
  process->rho_j=rho_j_;

  process->s_lambda=Lambda_y_+Lambda_v_+Lambda_c_;
  process->Drift=Lambda_y_*(exp(mu_y_+process->Sigma_y_sqr_
    demi)-1)
    +Lambda_c_*(exp(mu_cy_+process->sigma_cy_sqr_demi)-1);
  //>> Two way to compute drift term due to jump,
  //>> Put on Band matrix
  (*jump_drift)= 0;
  //>> Or Put in FD scheme (comment previous line and un
    comment to next line)
```

```
  //  (*jump_drift)= -process->C_Gamma_minus_Alpha_Minus*
    process->Lambdap1powAlphaMinus;
  //process->Lambdap1powAlphaMinus=0.0;
  return process;
}


void DPS_diffusion_update(DPS_diffusion * process)
{
  process->sigma_sqr=process->Sigma*process->Sigma;
  process->theta_sqr=process->Theta*process->Theta;
  process->sigma_sqr_d_eta_kappa=process->sigma_sqr/(proces
    s->Eta*process->Kappa);
  process->etakappathetam2=(process->Eta*process->Kappa)/
    process->theta_sqr;
  process->rho_theta=process->Rho*process->Theta;
  process->s_lambda=process->Lambda_y+process->Lambda_v+
    process->Lambda_c;
  process->Drift=process->Lambda_y*(exp(process->mu_y+proc
    ess->Sigma_y_sqr_demi)-1)
    +process->Lambda_c*(exp(process->mu_cy+process->sigma_
    cy_sqr_demi)-1);

};

DPS_diffusion * DPS_diffusion_create_from_vect(const PnlVec
    t * input)
{
  int i;
  DPS_diffusion * process = malloc(sizeof(DPS_diffusion));
  process->nb_parameters=NB_PARAM_DPS;
  for(i=0;i<process->nb_parameters;i++)
    SETPROCESSPARAMETER(process,i,GET(input,i));
  DPS_diffusion_update(process);
  return process;
};

void DPS_diffusion_list(const DPS_diffusion * process)
{
  printf("Eta    = %7.4f {n Kappa = %7.4f {n Rho = %7.4f {n
    Theta = %7.4f {n Sigma = %7.4f {n mu_y = %7.4f {n Sigma_y =
```

```
     %7.4f {n Lambda_y = %7.4f {n mu_v = %7.4f {n Lambda_v = %
    7.4f {n mu_cy = %7.4f {n Sigma_cy = %7.4f {n  mu_cv = %7.4
    f {n Lambda_c = %7.4f {n rho_j = %7.4f {n",
         process->Eta,  process->Kappa,  process->Rho,
    process->Theta,  process->Sigma,process->mu_y, 2.*sqrt(proces
    s->Sigma_y_sqr_demi),process->Lambda_y,  process->mu_v,proc
    ess->Lambda_v,
         2.0*sqrt(process->sigma_cy_sqr_demi),process->mu_
    cy,process->mu_cv,process->Lambda_c,process->rho_j);
};

// ---------------    CIR --------------------
/*
  2y0 i u {gamma^2 sinh({gamma t /2) /({kappa+{gamma coth({
    gammat/2))^2
  + {kappa{2 {eta / {lambda^2
  {paren{{frac{{gamma}{{kappa}-{frac{{kappa}{{gamma}} {
    frac{{sinh {gamma
  t/2}{cosh {gamma t /2 + {kappa/gamma {sinh {gamma t/2}

*/

dcomplex CIR_diffusion_characteristic_exponent_no_time_
    levy(dcomplex u,double t,CIR_diffusion * mod)
{
  //>> Case 1 code infinitesimal generator of backward k=i-j
  dcomplex psi,N0,D0;
  dcomplex gamma  = Csqrt(Complex(mod->Kappa_sqr+2*mod->Lam
    bda_sqr*u.i,-2*mod->Lambda_sqr*u.r));
  dcomplex exp_gamma_t=Cexp(RCmul(t*0.5,gamma));
  dcomplex exp_gamma_mt=Cexp(RCmul(-t*0.5,gamma));
  dcomplex cosh=RCmul(0.5,Cadd(exp_gamma_t,exp_gamma_mt));
  dcomplex gcosh=Cmul(gamma,cosh);
  dcomplex sinh=RCmul(0.5,Csub(exp_gamma_t,exp_gamma_mt));
  dcomplex sinhoverg=RCmul(mod->Kappa,Cdiv(sinh,gamma));
  dcomplex g_sqr_sinh=Cmul(RCmul(0.5,Cmul(gamma,gamma)),si
    nh);
  gcosh=RCadd(mod->Kappa,gcosh);
  gcosh=Cmul(gcosh,gcosh);

  psi=RCmul(2*mod->y0,Cdiv(Cmul(Complex(-u.i,u.r),g_sqr_si
```

```
    nh),gcosh));
  N0=Cmul(Csub(Complex(gamma.r/mod->Kappa,gamma.i/mod->Kapp
    a),
                RCmul(mod->Kappa,Cinv(gamma))),sinh);
  D0=Cadd(cosh,sinhoverg);
  psi=Cadd(psi,RCmul(mod->Kappa_sqr_eta_div_lambda_sqr,Cdi
    v(N0,D0)));
  return psi;
}


dcomplex CIR_diffusion_ln_characteristic_function_no_time_l
    evy(dcomplex u,double t,CIR_diffusion * mod)
{
  dcomplex gamma  = Csqrt(Complex(mod->Kappa_sqr+2*mod->Lam
    bda_sqr*u.i,-2*mod->Lambda_sqr*u.r));
  dcomplex exp_gamma_t=Cexp(RCmul(t*0.5,gamma));
  dcomplex exp_gamma_mt=Cexp(RCmul(-t*0.5,gamma));
  dcomplex cosh=RCmul(0.5,Cadd(exp_gamma_t,exp_gamma_mt));
  dcomplex gcosh=Cmul(gamma,cosh);
  dcomplex sinhoverg=RCmul(mod->Kappa,Cdiv(RCmul(0.5,Csub(
    exp_gamma_t,exp_gamma_mt)),gamma));
  dcomplex psi=RCadd(mod->Kappa_sqr_eta_div_lambda_sqr*t,
                     RCmul(2*mod->y0,Cdiv(Complex(-u.i,u.r)
    ,RCadd(mod->Kappa,gcosh))));
  psi=Cadd(psi,RCmul(-mod->Two_kappa_eta_div_lambda_sqr,Clo
    g(Cadd(cosh,sinhoverg))));
  //printf(" %7.4f +i %7.4f -> %7.4f +i %7.4f & {n",u.r,u.
    i,psi.r,psi.i);
  return psi;
}

dcomplex CIR_diffusion_characteristic_exponent_without_cas
    t(dcomplex u,double t,CIR_diffusion * mod)
{
  dcomplex miu,i_psi_u;
  CIR_diffusion_update_time(mod,t);
  miu=Complex(u.i,-u.r);
  i_psi_u=mod->characteristic_exponent(u,mod->Levy);
  i_psi_u=Complex(-i_psi_u.i,i_psi_u.r);
  return
```

```
    Cadd(CIR_diffusion_characteristic_exponent_no_time_
    levy(i_psi_u,t,mod),
         RCmul(mod->Jump_drift_psi,miu));
}

dcomplex CIR_diffusion_ln_characteristic_function_without_
    cast(dcomplex u,double t,CIR_diffusion * mod)
{
  dcomplex miu,i_psi_u,phi;
  CIR_diffusion_update_time(mod,t);
  miu=Complex(u.i,-u.r);
  i_psi_u=mod->characteristic_exponent(u,mod->Levy);
  i_psi_u=Complex(-i_psi_u.i,i_psi_u.r);
  phi= Cadd(CIR_diffusion_ln_characteristic_function_no_
    time_levy(i_psi_u,t,mod),
            RCmul(mod->Jump_drift,miu));
  return phi;
}

dcomplex CIR_diffusion_characteristic_exponent(dcomplex u,
    double t,void * mod)
{
  return  CIR_diffusion_characteristic_exponent_without_cas
    t(u,t,(CIR_diffusion *)mod);
}
dcomplex CIR_diffusion_ln_characteristic_function(dcomplex
    u,double t,void * mod)
{
  return CIR_diffusion_ln_characteristic_function_without_
    cast(u,t,(CIR_diffusion *)mod);
}

CIR_diffusion * CIR_diffusion_create(double Kappa,double Et
    a,
                                     double Lambda,double
    y0,
                                     void * Levy_,
                                     dcomplex (*characteri
    stic_exponent_)(dcomplex,void *),
                                     double *jump_drift)
{
```

```
  CIR_diffusion * process = malloc(sizeof(CIR_diffusion));
  process->nb_parameters=NB_PARAM_CIR;
  process->Kappa=Kappa;
  process->Eta=Eta;
  process->Lambda=Lambda;
  process->y0=y0;
  process->Kappa_sqr=Kappa*Kappa;
  process->Lambda_sqr=Lambda*Lambda;
  process->Kappa_sqr_eta_div_lambda_sqr=process->Kappa_sqr*
    Eta/process->Lambda_sqr;
  process->Two_kappa_eta_div_lambda_sqr=2*Kappa*Eta/proces
    s->Lambda_sqr;


  process->time=0.0;


  process->Levy=Levy_;
  process->characteristic_exponent=characteristic_exponent_
    ;

  (*jump_drift)= 0;
  return process;
};

void CIR_diffusion_update(CIR_diffusion * process)
{
  process->Kappa_sqr=process->Kappa*process->Kappa;
  process->Lambda_sqr=process->Lambda*process->Lambda;
  process->Kappa_sqr_eta_div_lambda_sqr=process->Kappa_sqr*
    process->Eta/process->Lambda_sqr;
  process->Two_kappa_eta_div_lambda_sqr=2.*process->Kappa*
    process->Eta/process->Lambda_sqr;
  process->time=0.0;
}
CIR_diffusion * CIR_VG_diffusion_create_from_vect(const Pn
    lVect * input)
{
  int i;
  CIR_diffusion * process = malloc(sizeof(CIR_diffusion));
  PnlVect Levy_param=pnl_vect_wrap_subvect(input,NB_PARAM_
```

```
    CIR,NB_PARAM_CIRVG-NB_PARAM_CIR);
  process->nb_parameters=NB_PARAM_CIR;
  for(i=0;i<NB_PARAM_CIR;i++)
    SETPROCESSPARAMETER(process,i,GET(input,i));
  CIR_diffusion_update(process);
  process->Levy=VG_process_create_from_vect(&Levy_param);
  process->characteristic_exponent=&VG_process_characteri
    stic_exponent;
  return process;
};


void CIR_diffusion_list(const CIR_diffusion * process)
{
  printf(" Kappa= %7.4f {n  Eta  =  %7.4f {n Lambda = %7.4
    f {n  y0 =  %7.4f {n ",process->Kappa,process->Eta,process-
    >Lambda,process->y0);
}


void CIR_diffusion_update_time(CIR_diffusion *  process,
    double t)
{
  if(process->time!=t)
    {
      dcomplex i_psi_u=process->characteristic_exponent(
    Complex(0,-1),process->Levy);
      i_psi_u=Complex(0.0,i_psi_u.r);
      process->time=t;
      process->Jump_drift=Creal(CIR_diffusion_ln_characteri
    stic_function_no_time_levy(i_psi_u,t,process));
      process->Jump_drift_psi=Creal(CIR_diffusion_charact
    eristic_exponent_no_time_levy(i_psi_u,t,process));
    }
};

// ---------------    GammaOU --------------------
dcomplex GammaOU_diffusion_characteristic_exponent_no_time_
    levy(dcomplex u,double t,GammaOU_diffusion * mod)
{
  //>> Case 1 code infinitesimal generator of backward k=i-j
```

```
  dcomplex iu=Complex(-u.i,u.r);
  dcomplex psi=RCmul(-mod->y0_el,iu);
  dcomplex F1=RCmul(-mod->Lambda_a,Cinv(RCadd(-mod->Lambda_
    b,iu)));
  dcomplex F2=RCmul(-mod->beta_el,iu );
  F2=Cdiv(F2,RCadd(-mod->Beta,RCmul(mod->one_m_el_div_lambd
    a,iu)));
  F2=Csub(F2,iu);
  psi=Cadd(psi,Cmul(F1,F2));
  return psi;
}

dcomplex GammaOU_diffusion_ln_characteristic_function_no_
    time_levy(dcomplex u,double t,GammaOU_diffusion * mod)
{
  dcomplex iu=Complex(-u.i,u.r);
  dcomplex psi=RCmul(mod->y0_one_m_el_div_lambda,iu);
  dcomplex F1=RCmul(mod->Lambda_a,Cinv(RCadd(-mod->Lambda_
    b,iu)));
  dcomplex F2=RCmul(-mod->Beta,Cinv(RCadd(-mod->Beta,RCmul(
    mod->one_m_el_div_lambda,iu))));
  psi=Cadd(psi,Cmul(F1,Cadd(RCmul(mod->Beta,Clog(F2)),RCmu
    l(-t,iu))));
  return psi;
}

dcomplex GammaOU_diffusion_characteristic_exponent_without_
    cast(dcomplex u,double t,GammaOU_diffusion * mod)
{

  dcomplex miu,i_psi_u;
  GammaOU_diffusion_update_time(mod,t);
  miu=Complex(u.i,-u.r);
  i_psi_u=mod->characteristic_exponent(u,mod->Levy);
  i_psi_u=Complex(i_psi_u.i,-i_psi_u.r);
  return
    Cadd(GammaOU_diffusion_characteristic_exponent_no_time_
    levy(i_psi_u,t,mod),
         RCmul(mod->Jump_drift_psi,miu));
}
```

```
dcomplex GammaOU_diffusion_ln_characteristic_function_with
    out_cast(dcomplex u,double t,GammaOU_diffusion * mod)
{
  dcomplex miu,i_psi_u;
  miu=Complex(u.i,-u.r);
  GammaOU_diffusion_update_time(mod,t);
  //i_psi_u=Cmul(u,Complex(0.5*u.r,0.5*(u.i+1)));;
  i_psi_u=mod->characteristic_exponent(u,mod->Levy);
  i_psi_u=Complex(-i_psi_u.i,i_psi_u.r);
  return
    Cadd(GammaOU_diffusion_ln_characteristic_function_no_
    time_levy(i_psi_u,t,mod),
         RCmul(mod->Jump_drift,miu));
}

dcomplex GammaOU_diffusion_characteristic_exponent(dcompl
    ex u,double t,void * mod)
{
  return GammaOU_diffusion_characteristic_exponent_without_
    cast(u,t,(GammaOU_diffusion *)mod);
}
dcomplex GammaOU_diffusion_ln_characteristic_function(dcom
    plex u,double t,void * mod)
{
  return GammaOU_diffusion_ln_characteristic_function_with
    out_cast(u,t,(GammaOU_diffusion *)mod);
}

GammaOU_diffusion * GammaOU_diffusion_create(double Lambda,
    double Alpha,double Beta,double y0,
                                                 void * Levy_,
                                                 dcomplex (*cha
    racteristic_exponent_)(dcomplex,void *),
                                                 double *jump_
    drift)
{
  GammaOU_diffusion * process = malloc(sizeof(GammaOU_dif
    fusion));
  process->nb_parameters=NB_PARAM_GAMMA;
  process->Lambda=Lambda;
  process->Alpha=Alpha;
```

```
  process->Beta=Beta;
  process->y0=y0;
  process->Lambda_a=Lambda*Alpha;
  process->Lambda_b=Lambda*Beta;
  process->beta_el=Beta;
  process->one_m_el_div_lambda=0;
  process->y0_one_m_el_div_lambda=0;
  process->y0_el=y0;
  process->time=0.0;
  process->Levy=Levy_;
  process->characteristic_exponent=characteristic_exponent_
    ;

  (*jump_drift)= 0;
  return process;
};
void GammaOU_diffusion_update(GammaOU_diffusion * process)
{
  process->Lambda_a=process->Lambda*process->Alpha;
  process->Lambda_b=process->Lambda*process->Beta;
  process->beta_el=process->Beta;
}


GammaOU_diffusion * GammaOU_VG_diffusion_create_from_vect(
    const PnlVect * input)
{
  int i;
  GammaOU_diffusion * process = malloc(sizeof(GammaOU_dif
    fusion));
  PnlVect Levy_param=pnl_vect_wrap_subvect(input,NB_PARAM_
    CIR,NB_PARAM_GAMMAVG-NB_PARAM_GAMMA);
  process->nb_parameters=NB_PARAM_GAMMA;
  for(i=0;i<NB_PARAM_GAMMA;i++)
    SETPROCESSPARAMETER(process,i,GET(input,i));
  GammaOU_diffusion_update(process);
  process->Levy=VG_process_create_from_vect(&Levy_param);
  process->characteristic_exponent=&VG_process_characteri
    stic_exponent;
  return process;
};
```

```
void GammaOU_diffusion_list(const GammaOU_diffusion * proc
    ess)
{
  printf(" Lambda= %7.4f {n  Alpha =  %7.4f {n Beta = %7.4
    f {n  y0 =  %7.4f {n ",process->Lambda,process->Alpha,proc
    ess->Beta,process->y0);
}

void GammaOU_diffusion_update_time(GammaOU_diffusion * proc
    ess,double t)
{
  if(process->time!=t)
    {
      dcomplex i_psi_u;
      double one_m_el=(1.-exp(-process->Lambda*t));
      process->one_m_el_div_lambda=one_m_el/process->Lambd
    a;
      process->y0_one_m_el_div_lambda=process->y0*process->
    one_m_el_div_lambda;
      process->y0_el=process->y0*exp(process->Lambda*t);
      process->beta_el=process->Beta*exp(-process->Lambda*
    t);
      process->time=t;

      i_psi_u=process->characteristic_exponent(Complex(0,-1
    .0),process->Levy);
      i_psi_u=Complex(0.0,i_psi_u.r);
      process->Jump_drift=Creal(GammaOU_diffusion_ln_chara
    cteristic_function_no_time_levy(i_psi_u,t,process));
      process->Jump_drift_psi=Creal(GammaOU_diffusion_chara
    cteristic_exponent_no_time_levy(i_psi_u,t,process));
    }
};



// ---------------  Levy_diffusion --------------------
dcomplex Levy_diffusion_characteristic_exponent(dcomplex u,
    double t,Levy_diffusion * mod)
{
```

```
  //>> To debug test characteristic exponent by euler schem
    e on ln_phi.

  dcomplex alpha=mod->characteristic_exponent(u,t,mod->proc
    ess);
  dcomplex alphap=mod->ln_characteristic_function(u,t+1e-8,
    mod->process);
  dcomplex alpham=mod->ln_characteristic_function(u,t-1e-8,
    mod->process);
  alphap=RCmul(- 0.5*1e8,Csub(alphap,alpham));
  printf(" psi = %e +i %e  and %e +i %e  {n",alpha.r,alpha.
    i,alphap.r,alphap.i);
  return Csub(mod->characteristic_exponent(u,t,mod->proces
    s),
              Complex(mod->vol_square*(u.r*u.r-u.i*u.i+u.i)
    ,(mod->vol_square)*(2*u.i*u.r-u.r)));
}

dcomplex Levy_diffusion_ln_characteristic_function(dcompl
    ex u,double t,Levy_diffusion * mod)
{
  //dcomplex phi=Cadd(mod->ln_characteristic_function(u,t,
    mod->process),Complex(t*mod->vol_square*(u.r*u.r-u.i*u.i+u.i)
    ,(t*mod->vol_square)*(2*u.i*u.r-u.r)));
  dcomplex phi=mod->ln_characteristic_function(u,t,mod->
    process);
  //printf(">>> phi(%7.4f + i%7.4f )  = %7.4f + i %7.4f {n"
    ,u.r,u.i,phi.r,phi.i);
  return phi;
}

dcomplex Levy_diffusion_ln_characteristic_function_with_cas
    t(dcomplex u,double t,void * mod)
{return Levy_diffusion_ln_characteristic_function(u,t,(
    Levy_diffusion*) mod);}

/*dcomplex Levy_diffusion_characteristic_function(dcomplex
    u,double t,Levy_diffusion * mod)
{return Cexp(Levy_diffusion_ln_characteristic_function(u,t,
    mod));}
*/
```

```
double Levy_diffusion_get_sigma_square(Levy_diffusion *
    Levy)
{return Levy->vol_square;};


Levy_diffusion * Levy_diffusion_create(void * process_,
                                       dcomplex (*characret
    ristic_exponent_)(dcomplex u,double t,void * mod),
                                       dcomplex (*ln_chara
    cteristic_function_)(dcomplex u,double t,void * mod))
{
  Levy_diffusion * Levy = malloc(sizeof(Levy_diffusion));
  Levy->process=process_;
  Levy->characteristic_exponent=characretristic_exponent_;
  Levy->ln_characteristic_function=ln_characteristic_
    function_;
  Levy->vol_square=IMPLICIT_VOL;
  return Levy;
};

Levy_diffusion * Levy_diffusion_create_from_vect(int model,
    const double * input)
{

  Levy_diffusion * Levy = malloc(sizeof(Levy_diffusion));
  PnlVect input_v;
  Levy->type_model=model;
  switch (model)
    {
    case 1: // Heston
      input_v=pnl_vect_wrap_array(input,NB_PARAM_HESTON);
      Levy->process =(void*)Heston_diffusion_create_from_
    vect(&input_v);
      Levy->nb_parameters=((Heston_diffusion*) Levy->proces
    s)->nb_parameters;
      Levy->characteristic_exponent=Heston_diffusion_chara
    cteristic_exponent;
      Levy->ln_characteristic_function=Heston_diffusion_ln_
    characteristic_function;
      break;
    case 2: //Bates
```

```
  input_v=pnl_vect_wrap_array(input,NB_PARAM_BATES);
  Levy->process =(void*)Bates_diffusion_create_from_vec
t(&input_v);
  Levy->nb_parameters=((Bates_diffusion*) Levy->proces
s)->nb_parameters;
  Levy->characteristic_exponent=Bates_diffusion_charact
eristic_exponent;
  Levy->ln_characteristic_function=Bates_diffusion_ln_
characteristic_function;
  break;
case 3: //BNS
  input_v=pnl_vect_wrap_array(input,NB_PARAM_BNS);
  Levy->process =(void*)BNS_diffusion_create_from_vect(
&input_v);
  Levy->nb_parameters=((BNS_diffusion*) Levy->process)-
>nb_parameters;
  Levy->characteristic_exponent=BNS_diffusion_charact
eristic_exponent;
  Levy->ln_characteristic_function=BNS_diffusion_ln_cha
racteristic_function;
  break;
case 4://DPS
  input_v=pnl_vect_wrap_array(input,NB_PARAM_DPS);
  Levy->process =(void*)DPS_diffusion_create_from_vect(
&input_v);
  Levy->nb_parameters=((DPS_diffusion*) Levy->process)-
>nb_parameters;
  Levy->characteristic_exponent=DPS_diffusion_charact
eristic_exponent;
  Levy->ln_characteristic_function=DPS_diffusion_ln_cha
racteristic_function;
  break;
case 5:// CIRVG
  input_v=pnl_vect_wrap_array(input,NB_PARAM_CIRVG);
  Levy->process =(void*)CIR_VG_diffusion_create_from_
vect(&input_v);
  Levy->nb_parameters=((CIR_diffusion*) Levy->process)-
>nb_parameters;
  Levy->characteristic_exponent=CIR_diffusion_charact
eristic_exponent;
  Levy->ln_characteristic_function=CIR_diffusion_ln_cha
```

```
    racteristic_function;
      break;
    case 6: //GammaOUVG
      input_v=pnl_vect_wrap_array(input,NB_PARAM_GAMMAVG);
      Levy->process =(void*)GammaOU_VG_diffusion_create_fro
    m_vect(&input_v);
      Levy->nb_parameters=((GammaOU_diffusion*) Levy->proc
    ess)->nb_parameters;
      Levy->characteristic_exponent=GammaOU_diffusion_chara
    cteristic_exponent;
      Levy->ln_characteristic_function=GammaOU_diffusion_ln
    _characteristic_function;
      break;
      default:
      return NULL;
    }
  Levy->vol_square=IMPLICIT_VOL;
  return Levy;
}

void Levy_diffusion_free(Levy_diffusion ** Levy)
{
  switch ((*Levy)->type_model)
    {
    case 1:
      free((Heston_diffusion*)((*Levy)->process));
      break;
    case 2:
      free((Bates_diffusion*)((*Levy)->process));
      break;
    case 3:
      free((BNS_diffusion*)((*Levy)->process));
      break;
    case 4:
      free((DPS_diffusion*)((*Levy)->process));
        break;
    case 5:
      free((CIR_diffusion*)((*Levy)->process));
        break;
    case 6:
      free((GammaOU_diffusion*)((*Levy)->process));
```

```
        break;
    default:
      {;}
    }
  free(*Levy);
  *Levy=NULL;
};
```

```
static dcomplex null_function(dcomplex u,void * mod)
{return CZERO;}

// Test for debug;

void test_CIR_diffusion(void )
{
  double Lambda=1.7864;
  double Kappa=1.2101;
```

```
  double Eta=-0.5501;
  double jump_drift;
  double Y0=1.0;
  int i=0;

  CIR_diffusion *Process= CIR_diffusion_create(Kappa,Eta,
    Lambda,Y0,NULL,&null_function,&jump_drift);
  printf(" ------- CIR  --------- {n");
  for(i=-10;i<=10;i++)
    {
      dcomplex u,res1,res2,res,res0;
      u=Complex(i/10.,300.0);
      CIR_diffusion_update_time(Process,0.99);
      res1=CIR_diffusion_ln_characteristic_function_no_
    time_levy(u,0.99,Process);
      CIR_diffusion_update_time(Process,1.01);
      res2=CIR_diffusion_ln_characteristic_function_no_
    time_levy(u,1.01,Process);
      CIR_diffusion_update_time(Process,1.0);
      res=RCmul(50.,Csub(res2,res1));
      res0=CIR_diffusion_characteristic_exponent_no_time_
    levy(u,1,Process);
      printf("> %7.4f +i %7.4f = %7.4f +i %7.4f && %7.4f +
    i %7.4f {n",u.r,u.i,res1.r,res1.i,res2.r,res2.i);
      printf("> %f +i %f = %f +i %f && %f +i %f {n",u.r,u.
    i,res.r,res.i,res0.r,res0.i);
    }

  free(Process);
}
void test_GammaOU_diffusion(void )
{
  double OU_Lambda=1.6790;
  double OU_Alpha=0.3484;
  double OU_Beta=0.7664;
  double jump_drift;
  double Y0=1.0;
  GammaOU_diffusion *Process= GammaOU_diffusion_create(OU_
    Lambda,OU_Alpha,OU_Beta,Y0,NULL,&null_function,&jump_drift);
  int i;
  printf(" ------- GammaOU  --------- {n");
```

```
  for(i=-10;i<=10;i++)
    {
      dcomplex u,res1,res2,res,res0;
        u=Complex(i/10.,0.2);
       GammaOU_diffusion_update_time(Process,0.99);
       res1=GammaOU_diffusion_ln_characteristic_function_n
    o_time_levy(u,0.99,Process);
       GammaOU_diffusion_update_time(Process,1.01);
       res2=GammaOU_diffusion_ln_characteristic_function_n
    o_time_levy(u,1.01,Process);
       GammaOU_diffusion_update_time(Process,1.0);
       res=RCmul(50.,Csub(res2,res1));
       res0=GammaOU_diffusion_characteristic_exponent_no_t
    ime_levy(u,1,Process);
       printf("> %7.4f +i %7.4f = %7.4f +i %7.4f && %7.4f +
    i %7.4f {n",u.r,u.i,res1.r,res1.i,res2.r,res2.i);
       printf("> %f +i %f = %f +i %f && %f +i %f {n",u.r,u.
    i,res.r,res.i,res0.r,res0.i);
    }
  free(Process);
}


#undef NB_PARAM_HESTON
#undef NB_PARAM_BATES
#undef NB_PARAM_BNS
#undef NB_PARAM_DPS
#undef NB_PARAM_CIR
#undef NB_PARAM_GAMMA
#undef GETPROCESSPARAMETER
#undef SETPROCESSPARAMETER
#undef GETLEVYPARAMETER
#undef SETLEVYPARAMETER
```

References