

## Help

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <
href../../../../common/math/cdo/cdo_math_h_src.pdfmath.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 "
href../../../../common/math/equity_pricer/pde_tools_h_src.pdfpde_tools.h"
#include "
href../../../../common/math/equity_pricer/time_change_levy_h_src.pdftime_change_levy

#define IMPLICIT_VOL 0.

// ----- 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 \ gam
    t/2}{cosh \ gamma t /2 + \ kappa/gamma \ sinh \ gamma t/2}

*/

dcomplex CIR_diffusion_characteristic_exponent_no_time_levy(dcomplex u, double t)
{
    //>> Case 1 code infinitesimal generator of backward k=i-j
    dcomplex psi, N0, D0;
    dcomplex gamma = Csqrt(Complex(mod->Kappa_sqr + 2 * mod->Lambda_sqr * u.i, -2
    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)), sinh);
gcosh = RCadd(mod->Kappa, gcosh);
gcosh = Cmul(gcosh, gcosh);

psi = RCmul(2 * mod->y0, Cdiv(Cmul(Complex(-u.i, u.r), g_sqr_sinh), gcosh));
N0 = Cmul(Csub(Complex(gamma.r / mod->Kappa, gamma.i / mod->Kappa),
                RCmul(mod->Kappa, Cinv(gamma))), sinh);
D0 = Cadd(cosh, sinhoverg);
psi = Cadd(psi, RCmul(mod->Kappa_sqr_eta_div_lambda_sqr, Cdiv(N0, D0)));
return psi;
}

dcomplex CIR_diffusion_ln_characteristic_function_no_time_levy(dcomplex u, double t)
{
    dcomplex gamma = Csqrt(Complex(mod->Kappa_sqr + 2 * mod->Lambda_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)), gcosh));
    dcomplex psi = RCadd(mod->Kappa_sqr_eta_div_lambda_sqr * t,
                        RCmul(2 * mod->y0, Cdiv(Complex(-u.i, u.r), RCadd(mod->Kappa, sinhoverg))));
    psi = Cadd(psi, RCmul(-mod->Two_kappa_eta_div_lambda_sqr, Clog(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_cast(dcomplex u, double t)
{
    dcomplex miu, i_psi_u;
    CIR_diffusion_update(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, void *mod)
{
    dcomplex miu, i_psi_u, phi;
    CIR_diffusion_update(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_cast(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 Eta,
                                     double Lambda, double y0,
                                     void *Levy_,
                                     dcomplex(*characteristic_exponent_)(dcomplex u, double t, void *mod),
                                     double *jump_drift)
{
    CIR_diffusion *process = malloc(sizeof(CIR_diffusion));
    process->Kappa = Kappa;
    process->Eta = Eta;
    process->Lambda = Lambda;
    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 / process->Lambda_sqr;
    process->y0 = y0;

    process->time = 0.0;
}

```

```

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

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

void CIR_diffusion_update(CIR_diffusion *process, double t)
{
    if (process->time != t)
    {
        dcomplex i_psi_u = process->characteristic_exponent(Complex(0, -1), process->time);
        i_psi_u = Complex(0.0, i_psi_u.r);
        process->time = t;
        process->Jump_drift = Creal(CIR_diffusion_ln_characteristic_function_no_time_levy(process, t));
        process->Jump_drift_psi = Creal(CIR_diffusion_characteristic_exponent_no_time_levy(process, t));
    }
};

// ----- GammaOU -----
dcomplex GammaOU_diffusion_characteristic_exponent_no_time_levy(dcomplex u, double t)
{
    //>> 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_lambda, 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)
{
    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)), RCmul(-t, iu))));
    return psi;
}

```

```

}

dcomplex GammaOU_diffusion_characteristic_exponent_without_cast(dcomplex u, double t, void *mod)
{
    dcomplex miu, i_psi_u;
    GammaOU_diffusion_update(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_without_cast(dcomplex u, double t, void *mod)
{
    dcomplex miu, i_psi_u;
    miu = Complex(u.i, -u.r);
    GammaOU_diffusion_update(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(dcomplex u, double t, void *mod)
{
    return GammaOU_diffusion_characteristic_exponent_without_cast(u, t, (GammaOU_diffusion *)mod);
}

dcomplex GammaOU_diffusion_ln_characteristic_function(dcomplex u, double t, void *mod)
{
    return GammaOU_diffusion_ln_characteristic_function_without_cast(u, t, (GammaOU_diffusion *)mod);
}

GammaOU_diffusion *GammaOU_diffusion_create(double Lambda, double Alpha, double t,
    void *Levy_,
    dcomplex(*characteristic_exponent_)(dcomplex, void *),
    double *jump_drift)

```

```

{
    GammaOU_diffusion *process = malloc(sizeof(GammaOU_diffusion));
    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, 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->Lambda;
        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_characteristic_function_n);
        process->Jump_drift_psi = Creal(GammaOU_diffusion_characteristic_exponent_n);
    }
};

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

```

```

dcomplex BNS_diffusion_characteristic_exponent_without_cast(dcomplex u, double t)
{
    //> 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_exponent_without_cast does not work");
    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, u_sqr_plus_i_u))));
    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_cast(u, t, (BNS_diffusion_mod *)mod);
}

dcomplex BNS_diffusion_ln_characteristic_function_without_cast(dcomplex u, double t, void *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 *mo
{
    return BNS_diffusion_ln_characteristic_function_without_cast(u, t, (BNS_diffus
}

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->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 uncomment to next line)
    // (*jump_drift)= -process->C_Gamma_minus_Alpha_Minus*process->Lambdap1powAlp
    //process->Lambdap1powAlphaMinus=0.0;
    return process;
};

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

static dcomplex Heston_diffusion_characteristic_exponent_without_cast(dcomplex u
{
    dcomplex u_sqr_plus_i_u, kmrho, d, g, emdt, gemdt, demdt, onemg, onemgemdt, ps
    u_sqr_plus_i_u = Cmul(u, (Cadd(u, CI)));

```



```

kmrho = Complex(mod->Kappa + mod->rho_theta * u.i, -mod->rho_theta * 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->sigma_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 *mo
{
    return Heston_diffusion_characteristic_exponent_without_cast(u, t, (Heston_dif
}

dcomplex Heston_diffusion_ln_characteristic_function_without_cast(dcomplex u, do
{
    dcomplex u_sqr_plus_i_u, kmrho, d, g, emdt, gemdt, demdt, onemg, onemgemdt, ps
    u_sqr_plus_i_u = Cmul(u, (Cadd(u, CI)));
    kmrho = Complex(mod->Kappa + mod->rho_theta * u.i, -mod->rho_theta * 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(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(dcomplex u, double t, void
{
    return Heston_diffusion_ln_characteristic_function_without_cast(u, t, (Heston_
}

Heston_diffusion *Heston_diffusion_create(double Eta_, double Kappa_, double Rho_
    double Theta_, double Sigma_,
    double *jump_drift)
{
    Heston_diffusion *process = malloc(sizeof(Heston_diffusion));
    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_sqr;
    process->rho_theta = Rho_ * Theta_;
    process->Drift = 0;
    printf(" eta= %7.4f kappa = %7.4f rho = %7.4f theta = %7.4f sigma_0 = %7.4f\n",
        process->Sigma);

    //>> 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 uncomment to next line)
    // (*jump_drift)= -process->C_Gamma_minus_Alpha_Minus*process->Lambdap1powAlp
    //process->Lambdap1powAlphaMinus=0.0;
    return process;
};

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

dcomplex Bates_diffusion_characteristic_exponent_without_cast(dcomplex u, double
{

```

```

dcomplex u_sqr_plus_i_u, kmrho, d, g, emdt, gemdt, demdt, onemg, onemgemdt, ps
u_sqr_plus_i_u = Cmul(u, (Cadd(u, CI)));
kmrho = Complex(mod->Kappa + mod->rho_theta * u.i, -mod->rho_theta * 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->sigma_sqr_d_eta_kappa
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_diffu
}

dcomplex Bates_diffusion_ln_characteristic_function_without_cast(dcomplex u, dou
{
    dcomplex u_sqr_plus_i_u, kmrho, d, g, emdt, gemdt, demdt, onemg, onemgemdt, ps
u_sqr_plus_i_u = Cmul(u, (Cadd(u, CI)));
kmrho = Complex(mod->Kappa + mod->rho_theta * u.i, -mod->rho_theta * 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(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(dcomplex u, double t, void *
{
    return Bates_diffusion_ln_characteristic_function_without_cast(u, t, (Bates_di
}

```

```

Bates_diffusion *Bates_diffusion_create(double Eta_, double Kappa_, double Rho_,
                                         double Theta_, double Sigma_,
                                         double mu_J_,
                                         double Sigma_J_, double Lambda_J_, doubl
{
    Bates_diffusion *process = malloc(sizeof(Bates_diffusion));
    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_sqr;
}

```

```

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 uncomment to next line)
// (*jump_drift)= -process->C_Gamma_minus_Alpha_Minus*process->LambdapowAlphaMinus;
//process->LambdapowAlphaMinus=0.0;
printf(" eta= %7.4f kappa = %7.4f rho = %7.4f theta = %7.4f sigma_0 = %7.4f\n",
        process->Eta, process->Kappa, process->Rho, process->Theta,
        process->Sigma, process->mu_J, process->Sigma_J, process->Lambda_J);
return process;
}

```

```

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

dcomplex DPS_diffusion_characteristic_exponent_without_cast(dcomplex u, double t)
{
    return CZERO;
}

dcomplex DPS_diffusion_characteristic_exponent(dcomplex u, double t, void *mod)
{
    return DPS_diffusion_characteristic_exponent_without_cast(u, t, (DPS_diffusion
}

dcomplex function_jump_variance(dcomplex a, dcomplex b, dcomplex gam, dcomplex o

{
    dcomplex psi_v, diff;
    psi_v = Csub(b, a);
    psi_v = Cdiv(RCmul(-2.0, a), Csub(Cmul(gam, gam), Cmul(psi_v, psi_v)));
    psi_v = Cmul(psi_v, Clog(RCsub(1, Cmul(Cdiv(Csub(Cadd(gam, b), a), RCmul(2.0,
    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)
{

```

```
    dcomplex a, b, c, d, gam, onememdt, beta, alpha_0, psi_y, psi_v, psi_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))));

```

```
    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));
    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), Cmul(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 *mo
{
    return DPS_diffusion_ln_characteristic_function_without_cast(u, t, (DPS_diffus
}

```

```

DPS_diffusion *DPS_diffusion_create(double Eta_, double Kappa_, 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->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_sqr;
    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 uncomment to next line)
// (*jump_drift)= -process->C_Gamma_minus_Alpha_Minus*process->Lambdap1powAlp
//process->Lambdap1powAlphaMinus=0.0;
printf(" eta= %7.4f kappa = %7.4f rho = %7.4f theta = %7.4f sigma_0 = %7.4f\n",
       process->Eta, process->Kappa, process->Rho, process->Theta,
       process->Sigma, process->mu_y, process->Sigma_y_sqr_demi, process->Lambda_v);
return process;
}

// ----- Levy_diffusion -----
dcomplex Levy_diffusion_characteristic_exponent(dcomplex u, double t, Levy_diffusion mod)
{
    //>> To debug test characteristic exponent by euler scheme on ln_phi.

    dcomplex alpha = mod->characteristic_exponent(u, t, mod->process);
    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->process),
               Complex(mod->vol_square * (u.r * u.r - u.i * u.i + u.i), (mod->vol_square * u.i)));
}

```



```

dcomplex Levy_diffusion_ln_characteristic_function(dcomplex u, double t, void *mod)
{
    Levy_diffusion *ptmod = (Levy_diffusion *) mod;
    //dcomplex phi=Cadd(mod->ln_characteristic_function(u,t,mod->process),Complex(
    dcomplex phi = ptmod->ln_characteristic_function(u, t, ptmod->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_characteristic_function(dcomplex u, double t, Levy_diffu
{
    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(*characetristic_
                                dcomplex(*ln_characteristic_function_)(dco
{
    Levy_diffusion *Levy = malloc(sizeof(Levy_diffusion));
    Levy->process = process_;
    Levy->characteristic_exponent = characetristic_exponent_;
    Levy->ln_characteristic_function = ln_characteristic_function_;
    Levy->vol_square = IMPLICIT_VOL;
    return Levy;
};

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, &n);
printf(" ----- CIR ----- \ n");
for (i = -10; i <= 10; i++)
{
    dcomplex u, res1, res2, res, res0;
    u = Complex(i / 10., 300.0);
    CIR_diffusion_update(Process, 0.99);
    res1 = CIR_diffusion_ln_characteristic_function_no_time_levy(u, 0.99, Process);
    CIR_diffusion_update(Process, 1.01);
    res2 = CIR_diffusion_ln_characteristic_function_no_time_levy(u, 1.01, Process);
    CIR_diffusion_update(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, res0.r, res0.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, &n);
    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(Process, 0.99);
        res1 = GammaOU_diffusion_ln_characteristic_function_no_time_levy(u, 0.99, Process);
    }
}

```

```

    GammaOU_diffusion_update(Process, 1.01);
    res2 = GammaOU_diffusion_ln_characteristic_function_no_time_levy(u, 1.01,
    GammaOU_diffusion_update(Process, 1.0);
    res = RCmul(50., Csub(res2, res1));
    res0 = GammaOU_diffusion_characteristic_exponent_no_time_levy(u, 1, Proces
    printf("> %7.4f +i %7.4f = %7.4f +i %7.4f && %7.4f +i %7.4f \ n", u.r, u.i
    printf("> %f +i %f = %f +i %f && %f +i %f \ n", u.r, u.i, res.r, res.i, re
    }
    free(Process);
}

```