

[Help](#)

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

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

#define IMPLICIT_VOL 0.0000
#define EPSILON_CALIBRATION 1e-2

#define GETPROCESSPARAMETER(v,i) ((double *)v)[i]

void Heston_diffusion_constraints(PnlVect *res , const Heston_diffusion *mod)
{
    double sigma0_min, ka_min, eta_min, theta_min, rhow_min;
    double sigma0_max, ka_max, eta_max, theta_max, rhow_max;

    sigma0_min = 0;
    ka_min = 0;
    eta_min = 0;
    theta_min = 0;
    rhow_min = -0.99;
    sigma0_max = 1;
    ka_max = 5;
    eta_max = 5;
    theta_max = 1;
    rhow_max = 0.99;

    pnl_vect_resize(res, 11);

    LET(res, 0) = eta_max - GETPROCESSPARAMETER(mod, 0);
    LET(res, 1) = -eta_min + GETPROCESSPARAMETER(mod, 0);
    LET(res, 2) = ka_max - GETPROCESSPARAMETER(mod, 1);
    LET(res, 3) = -ka_min + GETPROCESSPARAMETER(mod, 1);
    LET(res, 4) = rhow_max - GETPROCESSPARAMETER(mod, 2);
    LET(res, 5) = -rhow_min + GETPROCESSPARAMETER(mod, 2);
    LET(res, 6) = theta_max - GETPROCESSPARAMETER(mod, 3);
```

```

LET(res, 7) = -theta_min + GETPROCESSPARAMETER(mod, 3);
LET(res, 8) = sigma0_max - GETPROCESSPARAMETER(mod, 4);
LET(res, 9) = -sigma0_min + GETPROCESSPARAMETER(mod, 4);

LET(res, 10) = 2 * GETPROCESSPARAMETER(mod, 0) * GETPROCESSPARAMETER(mod, 1) -
// 2 Kappa * Eta - theta*theta
// Condition de Feller.

}
void Bates_diffusion_constraints(PnlVect *res , const Bates_diffusion *mod)
{
    double sigma0_min, ka_min, eta_min, theta_min, rhow_min, mu_J_min, Sigma_J_min;
    double sigma0_max, ka_max, eta_max, theta_max, rhow_max, mu_J_max, Sigma_J_max;
    sigma0_min = 0;
    ka_min = 0;
    eta_min = 0;
    theta_min = 0;
    rhow_min = -0.99;
    mu_J_min = -5;
    Sigma_J_min = -0.001;
    Lambda_J_min = -0.0001;
    sigma0_max = 1;
    ka_max = 5;
    eta_max = 5;
    theta_max = 1;
    rhow_max = 0.99;
    mu_J_max = 5;
    Sigma_J_max = 5;
    Lambda_J_max = 10;

    pnl_vect_resize(res, 17);

    LET(res, 0) = sigma0_max - GETPROCESSPARAMETER(mod, 0);
    LET(res, 1) = -sigma0_min + GETPROCESSPARAMETER(mod, 0);
    LET(res, 2) = ka_max - GETPROCESSPARAMETER(mod, 1);
    LET(res, 3) = -ka_min + GETPROCESSPARAMETER(mod, 1);
    LET(res, 4) = eta_max - GETPROCESSPARAMETER(mod, 2);
    LET(res, 5) = -eta_min + GETPROCESSPARAMETER(mod, 2);
    LET(res, 6) = theta_max - GETPROCESSPARAMETER(mod, 3);
    LET(res, 7) = -theta_min + GETPROCESSPARAMETER(mod, 3);
    LET(res, 8) = rhow_max - GETPROCESSPARAMETER(mod, 4);

```

```

    LET(res, 9) = -rhow_min + GETPROCESSPARAMETER(mod, 4);
    LET(res, 10) = -mu_J_min + GETPROCESSPARAMETER(mod, 5);
    LET(res, 11) = mu_J_max - GETPROCESSPARAMETER(mod, 5);
    LET(res, 12) = -Sigma_J_min + GETPROCESSPARAMETER(mod, 6);
    LET(res, 13) = Sigma_J_max - GETPROCESSPARAMETER(mod, 6);
    LET(res, 14) = -Lambda_J_min + GETPROCESSPARAMETER(mod, 7);
    LET(res, 15) = Lambda_J_max - GETPROCESSPARAMETER(mod, 7);
    LET(res, 16) = 2 * GETPROCESSPARAMETER(mod, 1) * GETPROCESSPARAMETER(mod, 2) -
    // Condition de Feller.
}
// ----- BNS -----
void BNS_diffusion_constraints(PnlVect *res , const BNS_diffusion *mod)
{
    pnl_vect_resize(res, 10);
    LET(res, 0) = GETPROCESSPARAMETER(mod, 0) - 0.05;
    LET(res, 8) = 1. - GETPROCESSPARAMETER(mod, 0);
    LET(res, 9) = 5.0 + GETPROCESSPARAMETER(mod, 1);
    LET(res, 1) = -GETPROCESSPARAMETER(mod, 1);
    LET(res, 2) = GETPROCESSPARAMETER(mod, 2);
    LET(res, 3) = 50.0 - fabs(GETPROCESSPARAMETER(mod, 2));
    LET(res, 4) = GETPROCESSPARAMETER(mod, 3) - 0.1;
    LET(res, 5) = 5.0 - fabs(GETPROCESSPARAMETER(mod, 3));
    LET(res, 6) = GETPROCESSPARAMETER(mod, 4);
    LET(res, 7) = 1.0 - fabs(GETPROCESSPARAMETER(mod, 4));

}
// ----- DPS -----
void DPS_diffusion_constraints(PnlVect *res , const DPS_diffusion *mod)
{
    double sigma0_min, ka_min, eta_min, theta_min, rhow_min, mu_J_min, Sigma_J_min;
    double sigma0_max, ka_max, eta_max, theta_max, rhow_max, mu_J_max, Sigma_J_max;
    sigma0_min = 0;
    ka_min = 0;
    eta_min = 0;
    theta_min = 0;
    rhow_min = -0.99;
    mu_J_min = -5;
    Sigma_J_min = -0.001;
    Lambda_J_min = -0.0001;
    sigma0_max = 1;

```

```

ka_max = 5;
eta_max = 5;
theta_max = 1;
rhow_max = 0.99;
mu_J_max = 5;
Sigma_J_max = 5;
Lambda_J_max = 10;
pnl_vect_resize(res, 30);
LET(res, 0) = sigma0_max - GETPROCESSPARAMETER(mod, 0);
LET(res, 1) = -sigma0_min + GETPROCESSPARAMETER(mod, 0);
LET(res, 2) = ka_max - GETPROCESSPARAMETER(mod, 1);
LET(res, 3) = -ka_min + GETPROCESSPARAMETER(mod, 1);
LET(res, 4) = eta_max - GETPROCESSPARAMETER(mod, 2);
LET(res, 5) = -eta_min + GETPROCESSPARAMETER(mod, 2);
LET(res, 6) = theta_max - GETPROCESSPARAMETER(mod, 3);
LET(res, 7) = -theta_min + GETPROCESSPARAMETER(mod, 3);
LET(res, 8) = rhow_max - GETPROCESSPARAMETER(mod, 4);
LET(res, 9) = -rhow_min + GETPROCESSPARAMETER(mod, 4);
LET(res, 10) = -mu_J_min + GETPROCESSPARAMETER(mod, 5);
LET(res, 11) = mu_J_max - GETPROCESSPARAMETER(mod, 5);
LET(res, 12) = -Sigma_J_min + GETPROCESSPARAMETER(mod, 6);
LET(res, 13) = Sigma_J_max - GETPROCESSPARAMETER(mod, 6);
LET(res, 14) = -Lambda_J_min + GETPROCESSPARAMETER(mod, 7);
LET(res, 15) = Lambda_J_max - GETPROCESSPARAMETER(mod, 7);
LET(res, 16) = 2 * GETPROCESSPARAMETER(mod, 1) * GETPROCESSPARAMETER(mod, 2) -
LET(res, 17) = -mu_J_min + GETPROCESSPARAMETER(mod, 8);
LET(res, 18) = mu_J_max - GETPROCESSPARAMETER(mod, 8);
LET(res, 19) = -Lambda_J_min + GETPROCESSPARAMETER(mod, 9);
LET(res, 20) = Lambda_J_max - GETPROCESSPARAMETER(mod, 9);
LET(res, 21) = -Sigma_J_min + GETPROCESSPARAMETER(mod, 10);
LET(res, 22) = Sigma_J_max - GETPROCESSPARAMETER(mod, 10);
LET(res, 23) = -mu_J_min + GETPROCESSPARAMETER(mod, 11);
LET(res, 24) = mu_J_max - GETPROCESSPARAMETER(mod, 11);
LET(res, 25) = -mu_J_min + GETPROCESSPARAMETER(mod, 12);
LET(res, 26) = mu_J_max - GETPROCESSPARAMETER(mod, 12);
LET(res, 27) = -Lambda_J_min + GETPROCESSPARAMETER(mod, 13);
LET(res, 28) = Lambda_J_max - GETPROCESSPARAMETER(mod, 13);
LET(res, 29) = 1. - fabs(GETPROCESSPARAMETER(mod, 14));
}

void CIR_diffusion_constraints(PnlVect *res , const CIR_diffusion *mod)

```

```

{
    double ka_min, eta_min, theta_min;
    double ka_max, eta_max, theta_max;

    ka_min = 0;
    eta_min = 0;
    theta_min = 0;
    ka_max = 5;
    eta_max = 1;
    theta_max = 5.;
    pnl_vect_resize(res, 10);

    LET(res, 0) = eta_max - GETPROCESSPARAMETER(mod, 1);
    LET(res, 1) = -eta_min + GETPROCESSPARAMETER(mod, 1);
    LET(res, 2) = ka_max - GETPROCESSPARAMETER(mod, 0);
    LET(res, 3) = -ka_min + GETPROCESSPARAMETER(mod, 0);
    LET(res, 4) = theta_max - GETPROCESSPARAMETER(mod, 2);
    LET(res, 5) = -theta_min + GETPROCESSPARAMETER(mod, 2);
    LET(res, 6) = 2 * GETPROCESSPARAMETER(mod, 0) * GETPROCESSPARAMETER(mod, 1) -

    LET(res, 7) = GETPROCESSPARAMETER((VG_process *)(mod->Levy), 0);
    LET(res, 8) = 2 - fabs(GETPROCESSPARAMETER((VG_process *)(mod->Levy), 1));
    LET(res, 9) = GETPROCESSPARAMETER((VG_process *)(mod->Levy), 2);
    /*
        LET(res, 7) = 20-fabs(GETPROCESSPARAMETER((NIG_process *)(mod->Levy), 0));
        LET(res, 8) = 20.-fabs(GETPROCESSPARAMETER((NIG_process *)(mod->Levy), 1));
        LET(res, 9) = 5.-fabs(GETPROCESSPARAMETER((NIG_process *)(mod->Levy), 2));
        LET(res, 10) = GETPROCESSPARAMETER((NIG_process *)(mod->Levy), 2);
        LET(res, 11) = GETPROCESSPARAMETER((NIG_process *)(mod->Levy), 0)*GETPROCES
    */
}

void GammaOU_diffusion_constraints(PnlVect *res , const GammaOU_diffusion *mod)
{
    // NIG GammaOU
    double Lambda_max, Lambda_min, OU_Alpha_max, OU_Alpha_min, OU_Beta_min, OU_Bet

    pnl_vect_resize(res, 9);
    Lambda_max = 50.0;
    Lambda_min = -0.0;

```

```

OU_Alpha_max = 10.;
OU_Alpha_min = 0.;
OU_Beta_min = 0.;
OU_Beta_max = 10.;
LET(res, 0) = Lambda_max - GETPROCESSPARAMETER(mod, 0);
LET(res, 1) = -Lambda_min + GETPROCESSPARAMETER(mod, 0);
LET(res, 2) = OU_Alpha_max - GETPROCESSPARAMETER(mod, 1);
LET(res, 3) = -OU_Alpha_min + GETPROCESSPARAMETER(mod, 1);
LET(res, 4) = -OU_Beta_min + GETPROCESSPARAMETER(mod, 2);
LET(res, 5) = OU_Beta_max - GETPROCESSPARAMETER(mod, 2);

LET(res, 6) = GETPROCESSPARAMETER((VG_process *)(mod->Levy), 0);
LET(res, 7) = 2 - fabs(GETPROCESSPARAMETER((VG_process *)(mod->Levy), 1));
LET(res, 8) = GETPROCESSPARAMETER((VG_process *)(mod->Levy), 2);
/*
    LET(res, 6) = 20-fabs(GETPROCESSPARAMETER((NIG_process *)(mod->Levy), 0));
    LET(res, 7) = 20.-fabs(GETPROCESSPARAMETER((NIG_process *)(mod->Levy), 1));
    LET(res, 8) = 5.-fabs(GETPROCESSPARAMETER((NIG_process *)(mod->Levy), 2));
    LET(res, 9) = GETPROCESSPARAMETER((NIG_process *)(mod->Levy), 2);
    LET(res, 10) = GETPROCESSPARAMETER((NIG_process *)(mod->Levy),
    0)*GETPROCESSPARAMETER((NIG_process *)(mod->Levy),
    0)-GETPROCESSPARAMETER((NIG_process *)(mod->Levy),
    1)*GETPROCESSPARAMETER((NIG_process *)(mod->Levy), 1);
*/
}

void Levy_diffusion_constraints(PnlVect *res, const Levy_diffusion *Levy)
{
    switch (Levy->type_model)
    {
        case 1:
            Heston_diffusion_constraints(res, Levy->process);
            break;
        case 2:
            Bates_diffusion_constraints(res, Levy->process);
            break;
        case 3:
            BNS_diffusion_constraints(res, Levy->process);
            break;
        case 4:

```

```
        DPS_diffusion_constraints(res, Levy->process);
        break;
    case 5:
        CIR_diffusion_constraints(res, Levy->process);
        break;
    case 6:
        GammaOU_diffusion_constraints(res, Levy->process);
        break;
    default:
    {
        printf(" constraints do no exists for thhis kind of process \ n");
        abort();
    };
}

}

#undef GETPROCESSPARAMETER
```