

[Help](#)

```

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

#include "levy_process.h"

#define IMPLICIT_VOL 0.0000
#define EPSILON_CALIBRATION 1e-2

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

void BS_process_constraints(PnlVect *res , const BS_process *mod)
{
    printf("to do \n");
}

void Merton_process_constraints(PnlVect *res , const Merton_process *mod)
{
    printf("to do \n");
}

// ----- CGMY -----
void CGMY_process_constraints(PnlVect *res , const CGMY_process *mod)
{
    pnl_vect_resize(res, 5);
    LET(res, 0) = GETPROCESSPARAMETER(mod, 0) - 0.0001;
    LET(res, 1) = GETPROCESSPARAMETER(mod, 1) - 0.1;
    LET(res, 2) = 1.9 - GETPROCESSPARAMETER(mod, 1);
    LET(res, 3) = GETPROCESSPARAMETER(mod, 2) - 0.0001;
    LET(res, 4) = GETPROCESSPARAMETER(mod, 3) - GETPROCESSPARAMETER(mod, 2);
}

// ----- Temperedstable -----

void Temperedstable_process_constraints(PnlVect *res , const Temperedstable_proc
{
    double eps = 1e-2;
    pnl_vect_resize(res, 10);
    LET(res, 0) = GETPROCESSPARAMETER(mod, 0) - 0.1;
    LET(res, 1) = GETPROCESSPARAMETER(mod, 1) - 0.1;

```

```

    LET(res, 2) = GETPROCESSPARAMETER(mod, 2) - 0.2;
    LET(res, 3) = GETPROCESSPARAMETER(mod, 3) - 0.2;
    LET(res, 4) = 10 - GETPROCESSPARAMETER(mod, 2);
    LET(res, 5) = 10 - GETPROCESSPARAMETER(mod, 3);
    LET(res, 6) = fabs(GETPROCESSPARAMETER(mod, 4) - 1.0) - eps;
    //Cplus != 1.0
    LET(res, 7) = GETPROCESSPARAMETER(mod, 4) - eps;
    //Cplus > 0.0
    LET(res, 8) = fabs(GETPROCESSPARAMETER(mod, 5) - 1.0) - eps;
    //Cminus != 1.0
    LET(res, 9) = GETPROCESSPARAMETER(mod, 5) - eps;
    //Cminus > 0.0
}

void NIG_process_constraints(PnlVect *res , const NIG_process *mod)
{
    pnl_vect_resize(res, 6);
    LET(res, 0) = GETPROCESSPARAMETER(mod, 0) + 20;
    LET(res, 1) = 20. - GETPROCESSPARAMETER(mod, 0);
    LET(res, 2) = GETPROCESSPARAMETER(mod, 1) + 5.;
    LET(res, 3) = 5. - GETPROCESSPARAMETER(mod, 1);
    LET(res, 4) = 1. - GETPROCESSPARAMETER(mod, 1) + 1.;
    LET(res, 5) = GETPROCESSPARAMETER(mod, 2) + 5;
}

void VG_process_constraints(PnlVect *res , const VG_process *mod)
{
    pnl_vect_resize(res, 4);
    LET(res, 0) = GETPROCESSPARAMETER(mod, 0) - 0.0001;
    LET(res, 1) = 2. - GETPROCESSPARAMETER(mod, 1);
    LET(res, 2) = GETPROCESSPARAMETER(mod, 1) + 2.;
    LET(res, 3) = GETPROCESSPARAMETER(mod, 2) - 0.0001;
}

void Meixner_process_constraints(PnlVect *res , const Meixner_process *mod)
{
    pnl_vect_resize(res, 3);
    LET(res, 0) = GETPROCESSPARAMETER(mod, 0) - 0.0001;
    LET(res, 1) = GETPROCESSPARAMETER(mod, 1) - 0.0001;

```

```

    LET(res, 2) = GETPROCESSPARAMETER(mod, 2) - 0.0001;
}

// ----- z_distribution -----

void z_distribution_process_constraints(PnlVect *res , const z_distribution_proc
{
    pnl_vect_resize(res, 4);
    LET(res, 0) = GETPROCESSPARAMETER(mod, 0) - 0.0001;
    LET(res, 1) = GETPROCESSPARAMETER(mod, 1) - 0.0001;
    LET(res, 2) = GETPROCESSPARAMETER(mod, 2) - 0.0001;
    LET(res, 3) = GETPROCESSPARAMETER(mod, 2) - 0.0001;
}

void Levy_process_constraints(PnlVect *res, const Levy_process *Levy)
{
    switch (Levy->type_model)
    {
        case 1:
            BS_process_constraints(res, Levy->process);
            break;
        case 2:
            Merton_process_constraints(res, Levy->process);
            break;
        case 3:
            CGMY_process_constraints(res, Levy->process);
            break;
        case 4:
            Temperedstable_process_constraints(res, Levy->process);
            break;
        case 5:
            VG_process_constraints(res, Levy->process);
            break;
        case 6:
            NIG_process_constraints(res, Levy->process);
            break;
        case 7:
            Meixner_process_constraints(res, Levy->process);
            break;
        case 8:

```

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

}

#undef GETPROCESSPARAMETER
```