Help

```
extern "C" {
#include  "
href../../../mod/guyon2d/guyon2d_std2d/guyon2d_std2d_h_src.pdfguyon2d_std2d.h"
#include "pnl/pnl_matrix.h"
#include "pnl/pnl_mathtools.h"
#include "pnl/pnl_vector.h"
#include "pnl/pnl_random.h"
#include "pnl/pnl_finance.h"
#include "
href../../../common/enums_h_src.pdfenums.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2017+2) //The "#els
static int CHK_OPT(MC_Guyon2D)(void *Opt, void *Mod)
{
  return NONACTIVE;
}

int CALC(MC_Guyon2D)(void *Opt, void *Mod, PricingMethod *Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else

static void  Algo1(PnlVect *Ei, PnlVect *Si, PnlVect *Vi, int L, int j) // Compu
{
    double T1,T2;
    int M = Si->size;
    int size_bin = floor((double)(M-1) / L); // taille d'un bin


    PnlVectInt *index = pnl_vect_int_create(M);

    pnl_vect_qsort_index(Si, index, 'i');
    pnl_vect_permute_inplace(Vi, index); // on range Vi comme Si

    for (int l = 0; l < L; l++) // pour chaque bin
    {
        T1 = 0.0;
        T2 = 0.0;
```

```
        for (int k = 1; k < size_bin + 1; k++) // pour chaque S_i,j contenu dans
        {
            T1 += pnl_vect_get(Vi, l * size_bin + k);
            T2 += 1.0;
        }

        pnl_vect_set(Ei, l, (double) T1 / T2);
    }

    pnl_vect_int_free(&index);
}

//Local volatility function
static double computeSigma(double sigma,double delta,double sij){ // Compute Sig
    return sigma + delta*sij/(1+sij);
}

  int MCGuyon2D(double S01,double S02, double K, double TT,int nmonit,int L, int
  {
    double T = nmonit*TT;  // maturity (months)
    int N_step = 1; // number of time steps per month
double sigma_1 = 0.2;//Local volatility parameter function for S1 and S1/S2
    double sigma_2 = 0.3;//Local volatility parameter function for S2
    double delta = 0.05;//Local volatility parameter
    int M = L*multipl + 1; // number of Monte Carlo samples
    int N = N_step*T; // number of time steps

PnlMat *Z, *Z1, *Z2, *Z3, *S1, *S2, *V1, *V2; // Z1 echantillon loi normale, S m
    PnlVect *V = pnl_vect_create_from_double(N, 0.0);
    PnlRng *rng = pnl_rng_create(PNL_RNG_KNUTH);
    PnlVectInt *index = pnl_vect_int_create(M);
    // Initialisation matrice, generateur aleatoire

    Z = pnl_mat_new();
    Z1 = pnl_mat_new();
    Z2 = pnl_mat_new();
    Z3 = pnl_mat_new();
    V1 = pnl_mat_new();
    V2 = pnl_mat_new();
    S1 = pnl_mat_new();
    S2 = pnl_mat_new();
```

```
pnl_rng_sseed(rng, 15646);

pnl_mat_resize(Z, 3*M, N);
pnl_mat_set_zero(Z);
pnl_mat_rng_normal(Z, 3*M, N, rng);

pnl_mat_resize(Z1, M, N);
pnl_mat_set_zero(Z1);
pnl_mat_resize(Z2, M, N);
pnl_mat_set_zero(Z2);
pnl_mat_resize(Z3, M, N);
pnl_mat_set_zero(Z3);

// Define 3 independent brownian movement : Z1,Z2,Z3

for (int i=0;i<M;i++){

    pnl_mat_get_row(V, Z, i);
    pnl_mat_set_row (Z1, V, i);
    pnl_mat_get_row (V, Z, M + i);
    pnl_mat_set_row (Z2, V, i);
    pnl_mat_get_row (V, Z, 2*M + i);
    pnl_mat_set_row (Z3, V, i);
}

pnl_mat_free(&Z);

pnl_mat_resize(V1, M, N + 1);
pnl_mat_set_all(V1, sigma_1+delta*S01/(1+S01));
pnl_mat_resize(V2, M, N + 1);
pnl_mat_set_all(V2, sigma_2+delta*S02/(1+S02));

pnl_mat_resize(S1, M, N + 1);
pnl_mat_set_all(S1, S01);
pnl_mat_resize(S2, M, N + 1);
pnl_mat_set_all(S2, S02);

PnlVect *E1_j = pnl_vect_create_from_double(L, 0.0); // Ei : vecteur de tail
PnlVect *E2_j = pnl_vect_create_from_double(L, 0.0); // Ei : vecteur de tail
PnlVect *E3_j = pnl_vect_create_from_double(L, 0.0); // Ei : vecteur de tail
```

```
PnlVect *V1_j = pnl_vect_create_from_double(M, 0.0); // V1_j : vecteur de ta
PnlVect *V2_j = pnl_vect_create_from_double(M, 0.0); // V2_j : vecteur de ta
PnlVect *V3_j = pnl_vect_create_from_double(M, 0.0); // V3_j : vecteur de ta

PnlVect *S12_j = pnl_vect_create_from_double(M, 0.0); // S12_j : vecteur de
PnlVect *S12_j_copy = pnl_vect_new();

double v1_ij, v2_ij,s1_ij, s2_ij;
double EC_1, EC_2, EC_3, rho, sig_12; // esp cond : S2 (sigma1^2 + sigma2^2)

int a = 0; // number of times rho becomes greater or less than 1 or -1

for (int j = 0; j < N; j++) // sur tous les pas de temps
{
    for (int k = 0; k < M; k ++){
        pnl_vect_set(S12_j,k,pnl_mat_get(S1, k, j)/pnl_mat_get(S2, k, j));
        pnl_vect_set(V1_j,k,pnl_mat_get(S2, k, j)*(computeSigma(sigma_1,delt
                                                    +computeSigma(sigma_2,del
        pnl_vect_set(V2_j,k,pnl_mat_get(S2, k, j));
        pnl_vect_set(V3_j,k,pnl_mat_get(S2, k, j)*computeSigma(sigma_1,delta
    }


    pnl_vect_clone(S12_j_copy, S12_j);
    Algo1(E1_j, S12_j_copy, V1_j, L, j);
    pnl_vect_clone(S12_j_copy, S12_j);
    Algo1(E2_j, S12_j_copy, V2_j, L, j);
    pnl_vect_clone(S12_j_copy, S12_j);
    Algo1(E3_j, S12_j_copy, V3_j, L, j);


    pnl_vect_qsort_index(S12_j, index, 'i');

    for (int i = 0; i < M; i++) //tous les tirages MC
    {
        v1_ij = pnl_mat_get(V1, GET_INT(index,i), j);
        s1_ij = pnl_mat_get(S1, GET_INT(index,i), j);
        v2_ij = pnl_mat_get(V2, GET_INT(index,i), j);
        s2_ij = pnl_mat_get(S2, GET_INT(index,i), j);
```

```
            // Prendre les EC sur le bon bin
            if (i == 0 ) EC_1 = pnl_vect_get(E1_j,0);
            else EC_1 = pnl_vect_get(E1_j, floor((double)(i - 1) * L / (M - 1)))

            if (i == 0) EC_2 = pnl_vect_get(E2_j,0);
            else EC_2 = pnl_vect_get(E2_j, floor((double)(i - 1) * L / (M - 1)))

            if (i == 0 ) EC_3 = pnl_vect_get(E3_j,0);
            else EC_3 = pnl_vect_get(E3_j, floor((double)(i- 1) * L / (M - 1)));

            sig_12 = computeSigma(sigma_1,delta,s1_ij/s2_ij);

            rho = MAX(MIN((EC_1-sig_12*sig_12*EC_2)/(2*EC_3),1.0),-1.0); // Comp

            if ((EC_1-sig_12*sig_12*EC_2)/(2*EC_3)>1.0 || (EC_1-sig_12*sig_12*EC
                a += 1;

            // Calculer S1_t+1,i et S2_t+1,i a partir de S1_t,i et S2_t,i

            pnl_mat_set(S1, i, j + 1, s1_ij*exp( v1_ij/sqrt((double)12*N_step)*p
            pnl_mat_set(S2, i, j + 1, s2_ij*exp( v2_ij/sqrt((double)12*N_step)*(

            // Calculer sigma_t+1,i

            pnl_mat_set(V1, i, j + 1, computeSigma(sigma_1,delta,pnl_mat_get(S1,
            pnl_mat_set(V2, i, j + 1, computeSigma(sigma_2,delta,pnl_mat_get(S2,

        }
    }

    pnl_vect_free(&E1_j);
    pnl_vect_free(&E2_j);
    pnl_vect_free(&E3_j);
    pnl_vect_free(&V1_j);
    pnl_vect_free(&V2_j);
    pnl_vect_free(&V3_j);
    pnl_vect_free(&S12_j);
pnl_vect_free(&V);
    pnl_vect_int_free(&index);
    pnl_rng_free(&rng);
    pnl_vect_free(&S12_j_copy);
```

```
double putW = 0.0;

for (int i=0; i<M;i++)
        putW += MAX(K - MIN(pnl_mat_get(S1, i, N),pnl_mat_get(S2, i, N)),0.0);

    putW /= M;

*ptprice=putW;

pnl_mat_free(&Z);
    pnl_mat_free(&Z1);
    pnl_mat_free(&Z2);
    pnl_mat_free(&Z3);
    pnl_mat_free(&V1);
    pnl_mat_free(&V2);
    pnl_mat_free(&S1);
    pnl_mat_free(&S2);

  return OK;
}

int CALC(MC_Guyon2D)(void *Opt, void *Mod, PricingMethod *Met)
{
  TYPEOPT *ptOpt = (TYPEOPT *)Opt;
  TYPEMOD *ptMod = (TYPEMOD *)Mod;


  return MCGuyon2D(ptMod->S01.Val.V_PDOUBLE,ptMod->S02.Val.V_PDOUBLE,(ptOpt->Pay
  ptOpt->Maturity.Val.V_DATE - ptMod->T.Val.V_DATE,
  ptMod->nmonit.Val.V_PINT,
 Met->Par[0].Val.V_INT,Met->Par[1].Val.V_INT, Met->Par[2].Val.V_ENUM.value,
  &(Met->Res[0].Val.V_DOUBLE)
 );

}

static int CHK_OPT(MC_Guyon2D)(void *Opt, void *Mod)
{

  if ((strcmp(((Option *)Opt)->Name, "PutMinimumEuro") == 0))
```

```
    return OK;

  return  WRONG;
}
#endif //PremiaCurrentVersion

static int MET(Init)(PricingMethod *Met, Option *Opt)
{
  //int type_generator;
  if (Met->init == 0)
    {
      Met->init = 1;

      Met->Par[0].Val.V_INT =100;
  Met->Par[1].Val.V_INT =100;
      Met->Par[2].Val.V_ENUM.value = 0;
      Met->Par[2].Val.V_ENUM.members = &PremiaEnumMCRNGs;
      Met->HelpFilenameHint = "mc_guyon2d";

    }
  return OK;
}

PricingMethod MET(MC_Guyon2D) =
{
  "MC_Guyon2D",
  { {"N Bins", INT2, {100}, ALLOW},
{"Multiplier", INT2, {100}, ALLOW},
    {"RandomGenerator", ENUM, {100}, ALLOW},
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CALC(MC_Guyon2D),
  { {"Price", DOUBLE, {100}, FORBID},
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CHK_OPT(MC_Guyon2D),
  CHK_mc,
  MET(Init)
};
}
```