Help

```
#include  "qtsm2d_stdi.h"
#include "pnl/pnl_vector.h"
#include "pnl/pnl_matrix.h"
#include "pnl/pnl_integration.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2009+2) //The "#els
static int CHK_OPT(AP_QTSM2D)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(AP_QTSM2D)(void *Opt, void *Mod, PricingMethod *Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else


static PnlMat *La_inv, *D, *A;
static PnlVect *tB, *b;
static int pow1, pow2;
static double bound;


//Calculates the eigenvalues and eigenvectors of a two-by-two matrix X; eigenval

static void eig(PnlMat *X, PnlVect *la, PnlMat *D)
{


  double det, tr;

  //calculate the trace and the determinant of the matrix
  tr = MGET(X, 0, 0) + MGET(X, 1, 1);
  det = MGET(X, 0, 0) * MGET(X, 1, 1) - MGET(X, 1, 0) * MGET(X, 0, 1);


  LET(la, 0) = 0.5 * (tr - sqrt(tr * tr - 4 * det));
  LET(la, 1) = 0.5 * (tr + sqrt(tr * tr - 4 * det));
```

```
  //find the eigenvectors
  MLET(D, 0, 0) = 1;
  MLET(D, 0, 1) = 1;
  MLET(D, 1, 0) = -MGET(X, 1, 0) / (MGET(X, 1, 1) - GET(la, 0));
  MLET(D, 1, 1) = -MGET(X, 1, 0) / (MGET(X, 1, 1) - GET(la, 1));

}

//Calculates the Riccati coefficients for a bond with time to maturity T
//rA: quadratic coefficient; rB: linear coefficient; rC: constant coefficient
//Bond price formula: exp(1/2*x'*rA*x+rB'*x+rC)
static void Riccati(PnlMat *rA, PnlVect *rB, double *rC, PnlMat *Kappa, PnlMat *
{


  PnlMat *k1a, *k2a, *k3a, *k4a, *rA1, *rA2, *rA3, *TKappa;
  PnlVect *k1b, *k2b, *k3b, *k4b, *rB1, *rB2, *rB3;
  double k1c, k2c, k3c, k4c;
  int step_num = (int)T / step_size, i;


  k1b = pnl_vect_create(d->size);
  k2b = pnl_vect_create(d->size);
  k3b = pnl_vect_create(d->size);
  k4b = pnl_vect_create(d->size);

  rB1 = pnl_vect_copy(rB);
  rB2 = pnl_vect_copy(rB);
  rB3 = pnl_vect_copy(rB);

  rA1 = pnl_mat_copy(rA);
  rA2 = pnl_mat_copy(rA);
  rA3 = pnl_mat_copy(rA);
  k1a = pnl_mat_create(Gamma->m, Gamma->n);
  k2a = pnl_mat_create(Gamma->m, Gamma->n);
  k3a = pnl_mat_create(Gamma->m, Gamma->n);
  k4a = pnl_mat_create(Gamma->m, Gamma->n);
  TKappa = pnl_mat_transpose(Kappa);
  for (i = 0; i < step_num; i++)
    {
      pnl_mat_clone(k1a, Gamma);
```

```
//pnl_mat_plus_mat(k1a,pnl_mat_mult_mat(TKappa,rA));
pnl_mat_dgemm('T', 'N', 1.0, Kappa, rA, 1.0, k1a);
//pnl_mat_plus_mat(k1a,pnl_mat_mult_mat(rA,Kappa));
pnl_mat_dgemm('N', 'N', 1.0, rA, Kappa, 1.0, k1a);
//pnl_mat_mult_double(k1a,-1.0);
//pnl_mat_plus_mat(k1a,pnl_mat_mult_mat(rA,rA));
pnl_mat_dgemm('N', 'N', 1.0, rA, rA, -1.0, k1a);
pnl_mat_clone(rA1, k1a);
pnl_mat_mult_double(rA1, 0.5 * step_size);
pnl_mat_plus_mat(rA1, rA);
pnl_mat_clone(k2a, Gamma);
//pnl_mat_plus_mat(k2a,pnl_mat_mult_mat(TKappa,rA1));
pnl_mat_dgemm('T', 'N', 1.0, Kappa, rA1, 1.0, k2a);
//pnl_mat_plus_mat(k2a,pnl_mat_mult_mat(rA1,Kappa));
pnl_mat_dgemm('N', 'N', 1.0, rA1, Kappa, 1.0, k2a);
//pnl_mat_mult_double(k2a,-1.0);
//pnl_mat_plus_mat(k2a,pnl_mat_mult_mat(rA1,rA1));
pnl_mat_dgemm('N', 'N', 1.0, rA1, rA1, -1.0, k2a);
pnl_mat_clone(rA2, k2a);
pnl_mat_mult_double(rA2, 0.5 * step_size);
pnl_mat_plus_mat(rA2, rA);
pnl_mat_clone(k3a, Gamma);
//pnl_mat_plus_mat(k3a,pnl_mat_mult_mat(TKappa,rA2));
pnl_mat_dgemm('T', 'N', 1.0, Kappa, rA2, 1.0, k3a);
//pnl_mat_plus_mat(k3a,pnl_mat_mult_mat(rA2,Kappa));
pnl_mat_dgemm('N', 'N', 1.0, rA2, Kappa, 1.0, k3a);
//pnl_mat_mult_double(k3a,-1.0);
//pnl_mat_plus_mat(k3a,pnl_mat_mult_mat(rA2,rA2));
pnl_mat_dgemm('N', 'N', 1.0, rA2, rA2, -1.0, k3a);
pnl_mat_clone(rA3, k3a);
pnl_mat_mult_double(rA3, 0.5 * step_size);
pnl_mat_plus_mat(rA3, rA);
pnl_mat_clone(k4a, Gamma);
//pnl_mat_plus_mat(k4a,pnl_mat_mult_mat(TKappa,rA3));
pnl_mat_dgemm('T', 'N', 1.0, Kappa, rA3, 1.0, k4a);
//pnl_mat_plus_mat(k4a,pnl_mat_mult_mat(rA3,Kappa));
pnl_mat_dgemm('N', 'N', 1.0, rA3, Kappa, 1.0, k4a);
//pnl_mat_mult_double(k4a,-1.0);
//pnl_mat_plus_mat(k4a,pnl_mat_mult_mat(rA3,rA3));
pnl_mat_dgemm('N', 'N', 1.0, rA3, rA3, -1.0, k4a);
```

```
pnl_vect_clone(k1b, d);
//pnl_vect_plus_vect(k1b,pnl_mat_mult_vect(TKappa,rB));
//pnl_vect_mult_double(k1b,-1.0);
pnl_mat_lAxpby(-1.0, TKappa, rB, -1.0, k1b);
//pnl_vect_plus_vect(k1b,pnl_mat_mult_vect(rA, theta));
pnl_mat_lAxpby(1.0, rA, theta, 1.0, k1b);
//pnl_vect_plus_vect(k1b,pnl_mat_mult_vect(rA, rB));
pnl_mat_lAxpby(1.0, rA, rB, 1.0, k1b);
pnl_vect_clone(rB1, k1b);
pnl_vect_mult_double(rB1, step_size);
pnl_vect_plus_vect(rB1, rB);
pnl_vect_clone(k2b, d);
//pnl_vect_plus_vect(k2b,pnl_mat_mult_vect(TKappa,rB1));
//pnl_vect_mult_double(k2b,-1.0);
pnl_mat_lAxpby(-1.0, TKappa, rB1, -1.0, k2b);
//pnl_vect_plus_vect(k2b,pnl_mat_mult_vect(rA1, theta));
pnl_mat_lAxpby(1.0, rA1, theta, 1.0, k2b);
//pnl_vect_plus_vect(k2b,pnl_mat_mult_vect(rA1, rB1));
pnl_mat_lAxpby(1.0, rA1, rB1, 1.0, k2b);
pnl_vect_clone(rB2, k2b);
pnl_vect_mult_double(rB2, step_size);
pnl_vect_plus_vect(rB2, rB);
pnl_vect_clone(k3b, d);
//pnl_vect_plus_vect(k3b,pnl_mat_mult_vect(TKappa,rB2));
//pnl_vect_mult_double(k3b,-1.0);
pnl_mat_lAxpby(-1.0, TKappa, rB2, -1.0, k3b);
//pnl_vect_plus_vect(k3b,pnl_mat_mult_vect(rA2, theta));
pnl_mat_lAxpby(1.0, rA2, theta, 1.0, k3b);
//pnl_vect_plus_vect(k3b,pnl_mat_mult_vect(rA2, rB2));
pnl_mat_lAxpby(1.0, rA2, rB2, 1.0, k3b);
pnl_vect_clone(rB3, k3b);
pnl_vect_mult_double(rB3, step_size);
pnl_vect_plus_vect(rB3, rB);
pnl_vect_clone(k4b, d);
//pnl_vect_plus_vect(k4b,pnl_mat_mult_vect(TKappa,rB3));
//pnl_vect_mult_double(k4b,-1.0);
pnl_mat_lAxpby(-1.0, TKappa, rB3, -1.0, k4b);
//pnl_vect_plus_vect(k4b,pnl_mat_mult_vect(rA3, theta));
pnl_mat_lAxpby(1.0, rA3, theta, 1.0, k4b);
//pnl_vect_plus_vect(k4b,pnl_mat_mult_vect(rA3, rB3));
pnl_mat_lAxpby(1.0, rA3, rB3, 1.0, k4b);
```

```
    k1c = 0.5 * pnl_vect_scalar_prod(rB, rB) + pnl_vect_scalar_prod(theta, rB)
    k2c = 0.5 * pnl_vect_scalar_prod(rB1, rB1) + pnl_vect_scalar_prod(theta, r
    k3c = 0.5 * pnl_vect_scalar_prod(rB2, rB2) + pnl_vect_scalar_prod(theta, r
    k4c = 0.5 * pnl_vect_scalar_prod(rB3, rB3) + pnl_vect_scalar_prod(theta, r


    pnl_mat_mult_double(k2a, 2.0);
    pnl_mat_mult_double(k3a, 2.0);
    pnl_mat_plus_mat(k1a, k2a);
    pnl_mat_plus_mat(k1a, k3a);
    pnl_mat_plus_mat(k1a, k4a);
    pnl_mat_mult_double(k1a, step_size / 6.0);
    pnl_mat_plus_mat(rA, k1a);

    pnl_vect_mult_double(k2b, 2.0);
    pnl_vect_mult_double(k3b, 2.0);
    pnl_vect_plus_vect(k1b, k2b);
    pnl_vect_plus_vect(k1b, k3b);
    pnl_vect_plus_vect(k1b, k4b);
    pnl_vect_mult_double(k1b, step_size / 6.0);
    pnl_vect_plus_vect(rB, k1b);

    *rC = (*rC) + step_size / 6.0 * (k1c + 2 * k2c + 2 * k3c + k4c);

  }
pnl_mat_free(&TKappa);
pnl_mat_free(&k1a);
pnl_mat_free(&k2a);
pnl_mat_free(&k3a);
pnl_mat_free(&k4a);
pnl_mat_free(&rA1);
pnl_mat_free(&rA2);
pnl_mat_free(&rA3);
pnl_vect_free(&k1b);
pnl_vect_free(&k2b);
pnl_vect_free(&k3b);
pnl_vect_free(&k4b);
pnl_vect_free(&rB1);
pnl_vect_free(&rB2);
```

```
  pnl_vect_free(&rB3);

}

//Purpose: compute the value of the integrand for a given value of \ rho_1 and \
static double Integrand(double rho, double phi, void *params)
{
  PnlVect *Xpp, *Xp, *Xt;
  double result, r;
  //compute r
  r = sqrt(2 * rho);
  //compute \ tilde{x}
  Xt = pnl_vect_create(2);
  pnl_vect_set(Xt, 0, r * cos(phi));
  pnl_vect_set(Xt, 1, r * sin(phi));
  //compute x'=La_inv (X_t- tB)
  Xpp = pnl_vect_copy(tB);
  pnl_vect_axpby(1.0, Xt, -1.0, Xpp);
  Xp = pnl_mat_mult_vect(La_inv, Xpp);

  //compute x''= D x'
  pnl_mat_mult_vect_inplace(Xpp, D, Xp);
  //compute the value of the integrand
  result = pow(GET(Xpp, 0), pow1) * pow(GET(Xpp, 1), pow2)
           * exp(-0.5 * pnl_mat_scalar_prod(A, Xp, Xp) - pnl_vect_scalar_prod(b,
  result *= exp(-r * r * 0.5) - bound;

  pnl_vect_free(&Xpp);
  pnl_vect_free(&Xp);
  pnl_vect_free(&Xt);
  return result;



}

static void Price(double T0, double TB, double K, int N, PnlVect *theta, PnlMat
{
  PnlMat *Y, *AA, *Acr, *R, *delY;
  PnlVect *vecR, *vecX;
  PnlMat *TKappa;
```

```
PnlMat *TSigma, *CSigma;
PnlVect *a, *sup;
double d0h, actu;
double e2;
int i, j;
double e1;
double eps = pow(10.0, -15.0);
PnlMat *errY;
PnlMat *Kappa1, *Kappacr, *X, *Z;
PnlVect *vecQ;
double d0t;
PnlVect *mu;
PnlMat *C, *Mu, *B, *Mu_inv, *Z1;
double norm1;
double norm2;
PnlVect *y, *xp, *yp, *xpp;
double alpha;
double beta;
double la;
PnlMat *H;
PnlVect *cm;
double cmm;
PnlMat *phi0;
int size;
PnlMat *lap;
PnlMat *lam;
PnlVect *la0;
int dims[3], tab[3], tab2[3], tab3[3];
PnlHmat *phip;
PnlHmat *phim;
int sss, k;
double r1;
double smu;
double t_bond;
double int_step;
PnlMat *rA, *rA1, *rA2;
PnlVect *rB, *rB1, *rB2, *tmp_vector;
double rC = 0.0, rC1 = 0.0, rC2 = 0.0;
PnlMat *tA, *gc_ij;
double q0;
double epsabs, epsrel, abserr, I1, coef;
```

```
int neval;
PnlMat *Err1, *Neval1;
double up;
PnlMat *gc_alpha;
int l;
double ss;
PnlVect *Hx1, *Hx2, *Hxx1, *Hxx2, *Hxxx1, *Hxxx2;
double hh1, hh2, hh3, hh4, hh5, hh6;
PnlMat *fff, *fff1, *fff2;
PnlHmat *fff3;
int dims2[4], tab4[4];
PnlMat *ph,  *LA, *LLmat, *ffmx, *deltam, *supmat, *supmat2, *Cmu, *Sginv, *TK
PnlVect *LL, *ffm, *g2, *vm, *thetam, *sup2, *sup3;
PnlHmat *Gammam, *ffmxx;
int m;
PnlMat *supmat3, *supmat4, *tmp_matrix;
double bond, bond1;
PnlVect *LAsup, *phsup, *ffsup2;
PnlFunc2D F;
F.F = &Integrand;
F.params = NULL;


*call_price = T0;

a = pnl_vect_create(2);
sup = pnl_vect_create(0);
sup2 = pnl_vect_create(0);
sup3 = pnl_vect_create(0);
b = pnl_vect_create(0);
tmp_vector = pnl_vect_create(0);
tmp_matrix = pnl_mat_create(0, 0);
TKappa = pnl_mat_create(0, 0);
ffsup2 = pnl_vect_create(0);

TSigma = pnl_mat_transpose(Sigma);
TKappaSigma = pnl_mat_create(0, 0);
pnl_mat_dgemm('T', 'T', 1.0, Kappa, Sigma, 0.0, TKappaSigma);
CSigma = pnl_mat_copy(Sigma);
pnl_mat_chol(CSigma);
pnl_mat_chol_syslin_inplace(CSigma, theta);
```

```
pnl_mat_get_row(sup, TKappaSigma, 0);
pnl_mat_chol_syslin_inplace(CSigma, sup);
pnl_mat_set_col(Kappa, sup, 0);
pnl_mat_get_row(sup, TKappaSigma, 1);
pnl_mat_chol_syslin_inplace(CSigma, sup);
pnl_mat_set_col(Kappa, sup, 1);
pnl_vect_clone(tmp_vector, d);
pnl_mat_mult_vect_transpose_inplace(d, Sigma, tmp_vector);
pnl_mat_dgemm('N', 'N', 1.0, Gamma, Sigma, 0.0, tmp_matrix);
pnl_mat_dgemm('T', 'N', 1.0, Sigma, tmp_matrix, 0.0, Gamma);


pnl_mat_dgemm('T', 'N', 1.0, Kappa, Kappa, 0.0, TKappa);
pnl_mat_plus_mat(TKappa, Gamma);
pnl_mat_chol(TKappa);
pnl_mat_chol_syslin(a, TKappa, d);
pnl_vect_clone(b, theta);
pnl_mat_lAxpby(-1.0, Kappa, a, -1, b);

//d0h=pnl_vect_scalar_prod(pnl_mat_mult_vect(Gamma,a),a);
d0h = pnl_mat_scalar_prod(Gamma, a, a);
pnl_mat_mult_vect_inplace(tmp_vector, Kappa, a);
d0h += pow(pnl_vect_norm_two(tmp_vector), 2);
d0h *= 0.5;
d0h += -pnl_vect_scalar_prod(d, a) + d0;

//Step 3: Calculation of $Y$: Get rid of \ Gamma; conjugation with \ exp(\ Phi
//Approximation by Newton's method of the solution to Y^2+Y\ kappa+\ kappa^TY-

//Y=pnl_mat_create_from_double(2,2,0.0);
//!!! note necessary Y=0
//AA=pnl_mat_create(2,2);
//pnl_mat_clone(AA,Y);
//pnl_mat_plus_mat(AA,Kappa);
AA = pnl_mat_copy(Kappa);
pnl_mat_mult_double(AA, -1);

Acr = pnl_mat_create_from_double(4, 4, 0.0);
MLET(Acr, 0, 0) = 2 * MGET(AA, 0, 0);
MLET(Acr, 0, 1) = MGET(AA, 1, 0);
```

```
MLET(Acr, 0, 2) = MGET(AA, 1, 0);
MLET(Acr, 1, 0) = MGET(AA, 0, 1);
MLET(Acr, 1, 1) = MGET(AA, 0, 0) + MGET(AA, 1, 1);
MLET(Acr, 1, 3) = MGET(AA, 1, 0);
MLET(Acr, 2, 0) = MGET(AA, 0, 1);
MLET(Acr, 2, 2) = MGET(AA, 0, 0) + MGET(AA, 1, 1);
MLET(Acr, 2, 3) = MGET(AA, 1, 0);
MLET(Acr, 3, 1) = MGET(AA, 0, 1);
MLET(Acr, 3, 2) = MGET(AA, 0, 1);
MLET(Acr, 3, 3) = 2 * MGET(AA, 1, 1);

R = pnl_mat_copy(Gamma);
pnl_mat_mult_double(R, -1);
//!!! note necessary Y=0
//pnl_mat_plus_mat(R,pnl_mat_mult_mat(Y,Y));
//pnl_mat_plus_mat(R,pnl_mat_mult_mat(Y,Kappa));
//pnl_mat_plus_mat(R,pnl_mat_transpose(pnl_mat_mult_mat(Y,Kappa)));

vecR = pnl_vect_create(4);
vecX = pnl_vect_create(4);
LET(vecR, 0) = MGET(R, 0, 0);
LET(vecR, 1) = MGET(R, 1, 0);
LET(vecR, 2) = MGET(R, 0, 1);
LET(vecR, 3) = MGET(R, 1, 1);
pnl_mat_syslin(vecX, Acr, vecR);
delY = pnl_mat_create(2, 2);
MLET(delY, 0, 0) = GET(vecX, 0);
MLET(delY, 0, 1) = GET(vecX, 2);
MLET(delY, 1, 0) = GET(vecX, 1);
MLET(delY, 1, 1) = GET(vecX, 3);
e2 = 0;
for (i = 0; i <= 1; i++)
  {
    for (j = 0; j <= 1; j++)
      {
        e2 += pow(MGET(delY, i, j), 2);
      }
  }
e1 = sqrt(e2);
Y = pnl_mat_create_from_double(2, 2, 0.0);
while (eps < e1)
```

```
{
  pnl_mat_plus_mat(Y, delY);
  pnl_mat_clone(AA, Y);
  pnl_mat_plus_mat(AA, Kappa);
  pnl_mat_mult_double(AA, -1);

  MLET(Acr, 0, 0) = 2 * MGET(AA, 0, 0);
  MLET(Acr, 0, 1) = MGET(AA, 1, 0);
  MLET(Acr, 0, 2) = MGET(AA, 1, 0);
  MLET(Acr, 1, 0) = MGET(AA, 0, 1);
  MLET(Acr, 1, 1) = MGET(AA, 0, 0) + MGET(AA, 1, 1);
  MLET(Acr, 1, 3) = MGET(AA, 1, 0);
  MLET(Acr, 2, 0) = MGET(AA, 0, 1);
  MLET(Acr, 2, 2) = MGET(AA, 0, 0) + MGET(AA, 1, 1);
  MLET(Acr, 2, 3) = MGET(AA, 1, 0);
  MLET(Acr, 3, 1) = MGET(AA, 0, 1);
  MLET(Acr, 3, 2) = MGET(AA, 0, 1);
  MLET(Acr, 3, 3) = 2 * MGET(AA, 1, 1);

  pnl_mat_clone(R, Gamma);
  //pnl_mat_mult_double(R,-1);
  //pnl_mat_plus_mat(R,pnl_mat_mult_mat(Y,Y));
  //pnl_mat_plus_mat(R,pnl_mat_mult_mat(Y,Kappa));
  //pnl_mat_plus_mat(R,pnl_mat_transpose(pnl_mat_mult_mat(Y,Kappa)));
  pnl_mat_dgemm('N', 'N', 1.0, Y, Y, -1.0, R);
  pnl_mat_dgemm('N', 'N', 1.0, Y, Kappa, 1.0, R);
  //(Y Kappa)^T = Kappa^T Y^T
  pnl_mat_dgemm('T', 'T', 1.0, Kappa, Y, 1.0, R);

  LET(vecR, 0) = MGET(R, 0, 0);
  LET(vecR, 1) = MGET(R, 1, 0);
  LET(vecR, 2) = MGET(R, 0, 1);
  LET(vecR, 3) = MGET(R, 1, 1);
  pnl_mat_syslin(vecX, Acr, vecR);


  MLET(delY, 0, 0) = GET(vecX, 0);
  MLET(delY, 0, 1) = GET(vecX, 2);
  MLET(delY, 1, 0) = GET(vecX, 1);
  MLET(delY, 1, 1) = GET(vecX, 3);
```

```
    e2 = 0;
    for (i = 0; i <= 1; i++)
      {
        for (j = 0; j <= 1; j++)
          {
            e2 += pow(MGET(delY, i, j), 2);
          }
      }
    e1 = sqrt(e2);

  }

pnl_mat_plus_mat(Y, delY);

pnl_mat_free(&Acr);
pnl_mat_free(&AA);
pnl_mat_free(&delY);
pnl_mat_free(&R);
pnl_vect_free(&vecR);



errY = pnl_mat_copy(Gamma);
//pnl_mat_mult_double(errY,-1);
//pnl_mat_plus_mat(errY, pnl_mat_mult_mat(pnl_mat_transpose(Kappa),Y));
//pnl_mat_plus_mat(errY, pnl_mat_mult_mat(Y,Kappa));
//pnl_mat_plus_mat(errY, pnl_mat_mult_mat(Y,Y));
pnl_mat_dgemm('T', 'N', 1.0, Kappa, Y, -1.0, errY);
pnl_mat_dgemm('N', 'N', 1.0, Y, Kappa, 1.0, errY);
pnl_mat_dgemm('N', 'N', 1.0, Y, Y, 1.0, errY);


Kappa1 = pnl_mat_copy(Kappa);
pnl_mat_plus_mat(Kappa1, Y);
Kappacr = pnl_mat_create_from_double(4, 4, 0.);
MLET(Kappacr, 0, 0) = 2 * MGET(Kappa1, 0, 0);
MLET(Kappacr, 0, 1) = MGET(Kappa1, 0, 1);
MLET(Kappacr, 0, 2) = MGET(Kappa1, 0, 1);
MLET(Kappacr, 1, 0) = MGET(Kappa1, 1, 0);
MLET(Kappacr, 1, 1) = MGET(Kappa1, 0, 0) + MGET(Kappa1, 1, 1);
```

```
MLET(Kappacr, 1, 3) = MGET(Kappa1, 0, 1);
MLET(Kappacr, 2, 0) = MGET(Kappa1, 1, 0);
MLET(Kappacr, 2, 2) = MGET(Kappa1, 0, 0) + MGET(Kappa1, 1, 1);
MLET(Kappacr, 2, 3) = MGET(Kappa1, 0, 1);
MLET(Kappacr, 3, 1) = MGET(Kappa1, 1, 0);
MLET(Kappacr, 3, 2) = MGET(Kappa1, 1, 0);
MLET(Kappacr, 3, 3) = 2 * MGET(Kappa1, 1, 1);

vecQ = pnl_vect_create_from_double(4, 0.0);
LET(vecQ, 0) = 2;
LET(vecQ, 3) = 2;

pnl_mat_syslin(vecX, Kappacr, vecQ);

X = pnl_mat_create(2, 2);
MLET(X, 0, 0) = GET(vecX, 0);
MLET(X, 0, 1) = GET(vecX, 2);
MLET(X, 1, 0) = GET(vecX, 1);
MLET(X, 1, 1) = GET(vecX, 3);
LET(sup, 0) = 1;
LET(sup, 1) = 0;
pnl_mat_chol(X);
Z = pnl_mat_create(2, 2);
pnl_mat_chol_syslin_inplace(X, sup);
pnl_mat_set_col(Z, sup, 0);
LET(sup, 0) = 0;
LET(sup, 1) = 1;
pnl_mat_chol_syslin_inplace(X, sup);
pnl_mat_set_col(Z, sup, 1);
pnl_mat_free(&Kappacr);
pnl_mat_free(&X);
pnl_vect_free(&vecX);
pnl_vect_free(&vecQ);


d0t = d0h + (MGET(Y, 0, 0) + MGET(Y, 1, 1) - MGET(Z, 0, 0) - MGET(Z, 1, 1)) /
Z1 = pnl_mat_copy(Z);
mu = pnl_vect_create(2);
D = pnl_mat_create_from_double(2, 2, 0.0);
C = pnl_mat_copy(D);
B = pnl_mat_copy(D);
```

```
A = pnl_mat_copy(D);
Mu = pnl_mat_create_from_double(2, 2, 0.0);
Mu_inv = pnl_mat_copy(Mu);

eig(Z, mu, D);
norm1 = sqrt(MGET(D, 0, 0) * MGET(D, 0, 0) + MGET(D, 1, 0) * MGET(D, 1, 0));
norm2 = sqrt(MGET(D, 1, 1) * MGET(D, 1, 1) + MGET(D, 0, 1) * MGET(D, 0, 1));


MLET(C, 0, 0) = -MGET(D, 0, 0) / norm1;
MLET(C, 1, 0) = -MGET(D, 1, 0) / norm1;
MLET(C, 0, 1) = MGET(D, 0, 1) / norm2;
MLET(C, 1, 1) = MGET(D, 1, 1) / norm2;

MLET(Mu, 0, 0) = sqrt(GET(mu, 0));
MLET(Mu, 1, 1) = sqrt(GET(mu, 1));
MLET(Mu_inv, 0, 0) = 1.0 / MGET(Mu, 0, 0);
MLET(Mu_inv, 1, 1) = 1.0 / MGET(Mu, 1, 1);

pnl_mat_mult_mat_inplace(tmp_matrix, C, Mu_inv);
pnl_mat_minus_mat(Kappa1, Z);
pnl_mat_dgemm('N', 'N', 1.0, Kappa1, tmp_matrix, 0.0, B);
pnl_mat_dgemm('T', 'N', 1.0, C, B, 0.0, tmp_matrix);
pnl_mat_dgemm('N', 'N', 1.0, Mu, tmp_matrix, 0.0, B);
pnl_mat_clone(A, Y);
pnl_mat_mult_double(Z1, 2.0);
pnl_mat_minus_mat(A, Z1);
pnl_mat_mult_double(A, -1);

pnl_mat_free(&Z1);
pnl_mat_free(&Kappa1);


//Step 6. List of all changes of variables needed
y = pnl_vect_create(2);
xp = pnl_vect_create(2);
yp = pnl_vect_create(2);
xpp = pnl_vect_create(2);
pnl_mat_chol_syslin(y, CSigma, x);
pnl_vect_clone(xp, y);
pnl_vect_plus_vect(xp, a);
```

```
//yp=pnl_mat_mult_vect(pnl_mat_transpose(C),xp);
pnl_mat_mult_vect_transpose_inplace(yp, C, xp);
pnl_mat_mult_vect_inplace(xpp, Mu, yp);
pnl_vect_free(&yp);


//Step 7. Calculation of \ alpha, \ beta; calculation of la=\ Lambda
alpha = GET(mu, 1) - GET(mu, 0);
beta = 2 * MGET(B, 0, 1);
la = sqrt(alpha * alpha - beta * beta);
pnl_mat_free(&B);



//Step 8. calculation of the coefficients of Hermite polynomials up to order N
H = pnl_mat_create_from_double(N + 1, N + 1, 0.0);
MLET(H, 0, 0) = 1;
MLET(H, 1, 1) = 2;
for (i = 2; i <= N; i++)
  {
    MLET(H, i, 0) = -MGET(H, i - 1, 1);
    for (j = 1; j < i; j++)
      {
        MLET(H, i, j) = 2 * MGET(H, i - 1, j - 1) - MGET(H, i - 1, j + 1) * (j
      }
    MLET(H, i, i) = 2 * MGET(H, i - 1, i - 1);
  }


//Step 9. Calculation of norms c_m of w_m;
cm = pnl_vect_create(N + 1);
cmm = sqrt(sqrt(M_PI));
LET(cm, 0) = cmm;
for (i = 1; i <= N; i++)
  {
    cmm = cmm * sqrt(i);
    LET(cm, i) = cmm;
  }
```

```
//Step 10: coefficients of eigenfunction expansion of phi_{r,0} in the basis w
size = (int)((0.5 * N) + 1);
phi0 = pnl_mat_create_from_double(size, 2 * (size - 1) + 1, 0.0);
MLET(phi0, 0, 0) = 1;
MLET(phi0, 1, 0) = beta;
MLET(phi0, 1, 1) = 2 * alpha;
MLET(phi0, 1, 2) = beta;
for (i = 2; i < size; i++)
  {
    MLET(phi0, i, 0) = beta * MGET(phi0, i - 1, 0);
    MLET(phi0, i, 1) = 2 * alpha * MGET(phi0, i - 1, 0) + beta * MGET(phi0, i
    MLET(phi0, i, 2 * i - 1) = 2 * alpha * MGET(phi0, (i - 1), (2 * i - 2)) +
    MLET(phi0, i, 2 * i) = beta * MGET(phi0, i - 1, 2 * i - 2);
    for (j = 2; j <= 2 * (i - 1); j++)
      {
        MLET(phi0, i, j) = beta * MGET(phi0, i - 1, j - 2) + 2 * alpha * MGET(
      }
  }


// Step 11. calculation of the bound ss(r) for |s|; calculations of all eigenv
smu = (GET(mu, 0) + GET(mu, 1));
la0 = pnl_vect_create(size);
dims[0] = (int)(ceil(0.5 * N));
dims[1] = N;
dims[2] = N + 1;
lap = pnl_mat_create_from_double((int)(ceil(0.5 * N)), N, 0.0);
lam = pnl_mat_create_from_double((int)(ceil(0.5 * N)), N, 0.0);
phip = pnl_hmat_create_from_double(3, dims, 0.0);
phim = pnl_hmat_create_from_double(3, dims, 0.0);


for (i = 0; i < size; i++)
  {

    sss = N - 2 * i;

    r1 = (i + 0.5) * smu + d0t;
    LET(la0, i) = r1;
    tab[0] = i;
    tab2[0] = i;
```

```
tab3[0] = i;

if (sss > 0)
  {
    for (j = 0; j < sss; j++)
      {
        MLET(lap, i, j) = r1 + (j + 1) * (la + smu) * 0.5;
        MLET(lam, i, j) = r1 + (j + 1) * (-la + smu) * 0.5;

        tab[1] = j;

        if (j == 0)
          {
            tab[2] = 0;
            pnl_hmat_set(phip, tab, (alpha - la)*MGET(phi0, i, 0));
            pnl_hmat_set(phim, tab, (alpha + la)*MGET(phi0, i, 0));
            tab[2] = 2 * i + j + 1;
            pnl_hmat_set(phip, tab, beta * MGET(phi0, i, 2 * i + j));
            pnl_hmat_set(phim, tab, beta * MGET(phi0, i, 2 * i + j));
            if (i > 0)
              {
                for (k = 1; k <= 2 * i + j; k++)
                  {
                    tab[2] = k;
                    pnl_hmat_set(phip, tab, beta * MGET(phi0, i, k - 1) +
                    pnl_hmat_set(phim, tab, beta * MGET(phi0, i, k - 1) +
                  }
              }
          }
        else
          {
            tab[2] = 0;
            tab2[1] = j - 1;
            tab3[1] = j - 1;
            tab2[2] = 0;

            pnl_hmat_set(phip, tab, (alpha - la)*pnl_hmat_get(phip, tab2))
            pnl_hmat_set(phim, tab, (alpha + la)*pnl_hmat_get(phim, tab2))

            tab[2] = 2 * i + j + 1;
            tab2[2] = 2 * i + j;
```

```
                pnl_hmat_set(phip, tab, beta * pnl_hmat_get(phip, tab2));
                pnl_hmat_set(phim, tab, beta * pnl_hmat_get(phim, tab2));
                for (k = 1; k <= 2 * i + j; k++)
                  {
                    tab[2] = k;
                    tab2[2] = k;
                    tab3[2] = k - 1;
                    pnl_hmat_set(phip, tab, beta * pnl_hmat_get(phip, tab3) +
                    pnl_hmat_set(phim, tab, beta * pnl_hmat_get(phim, tab3) +
                  }
              }
          }
      }
  }


//Step 12: Calculate the coefficient matrices for a bond with maturity T_B-T_0
t_bond = TB - T0;
int_step = 5 * pow(10.0, -4);

//rA: quadratic coefficient; rB: linear coefficient; rC: constant coefficient
//Bond price formula: exp(1/2*x'*rA*x+rB'*x+rC)
rA = pnl_mat_create_from_double(2, 2, 0.0);
rB = pnl_vect_create_from_double(2, 0);
Riccati(rA, rB, &rC, Kappa, Gamma, theta, d, d0, t_bond, int_step);

rA1 = pnl_mat_create_from_double(2, 2, 0.0);
rB1 = pnl_vect_create_from_double(2, 0);
Riccati(rA1, rB1, &rC1, Kappa, Gamma, theta, d, d0, T0, int_step);

rA2 = pnl_mat_create_from_double(2, 2, 0.0);
rB2 = pnl_vect_create_from_double(2, 0);
Riccati(rA2, rB2, &rC2, Kappa, Gamma, theta, d, d0, TB, int_step);

//Step 13: Calculate the coefficients of the expansion of g=max(exp(x'rAx+rB'x
//D=pnl_mat_create(2,2);
//D=pnl_mat_mult_mat(Mu, pnl_mat_transpose(C));
pnl_mat_dgemm('N', 'T', 1.0, Mu, C, 0, D);

tA = pnl_mat_copy(rA);
pnl_mat_mult_double(tA, -2);
```

```
pnl_mat_chol(tA);
La_inv = pnl_mat_create_from_double(2, 2, 0.0);
pnl_mat_set(La_inv, 0, 0, 1.0 / MGET(tA, 0, 0));
pnl_mat_set(La_inv, 0, 1, -MGET(tA, 1, 0) / (MGET(tA, 0, 0)*MGET(tA, 1, 1)));
pnl_mat_set(La_inv, 1, 1, 1.0 / MGET(tA, 1, 1));



//tB=pnl_vect_create(2);
//tB=pnl_mat_mult_vect(rA,a);
//pnl_vect_mult_double(tB,-2);
//pnl_vect_plus_vect(tB,rB);
//tB=pnl_mat_mult_vect(pnl_mat_transpose(La_inv), tB);

pnl_mat_mult_vect_inplace(tmp_vector, rA, a);
pnl_vect_mult_double(tmp_vector, -2);
pnl_vect_plus_vect(tmp_vector, rB);
tB = pnl_mat_mult_vect_transpose(La_inv, tmp_vector);


//q0=-rC-pnl_vect_scalar_prod(a,pnl_mat_mult_vect(rA,a))+pnl_vect_scalar_prod(
q0 = -rC - pnl_mat_scalar_prod(rA, a, a) + pnl_vect_scalar_prod(rB, a)
     - 0.5 * pnl_vect_scalar_prod(tB, tB);

//13b: calculate the coefficients c_ij
Err1 = pnl_mat_create(N + 1, N + 1);
Neval1 = pnl_mat_create(N + 1, N + 1);


epsabs = pow(10.0, -3);
epsrel = 1.0;
bound = K * exp(q0);

up = -log(bound);
coef = fabs(exp(-q0) * (MGET(D, 0, 0) * MGET(D, 1, 1) - MGET(D, 1, 0) * MGET(D
gc_ij = pnl_mat_create_from_double(N + 1, N + 1, 0.0);

for (i = 0; i <= N; i++)
  {
    for (j = 0; j <= N; j++)
      {
```

```
          pow1 = i;
          pow2 = j;
          I1 = 0.0;
          abserr = 0.0;
          neval = 0;
          if (up < 0.)
            {
              I1 = 0.;
              abserr = 0.;
              neval = 0;
            }
          else
            {
              pnl_integration_GK2D(&F, 0, up, 0, 2 * M_PI, epsabs, epsrel, &I1,
            }
          MLET(Err1, i, j) = abserr;
          MLET(Neval1, i, j) = neval;
          MLET(gc_ij, i, j) = coef * I1;
        }
    }


pnl_mat_free(&tA);
pnl_mat_free(&Err1);
pnl_mat_free(&Neval1);
pnl_mat_free(&La_inv);
pnl_vect_free(&tB);

//13c: calculate the coefficients c_\ alpha
gc_alpha = pnl_mat_create_from_double(N + 1, N + 1, 0.0);

for (i = 0; i <= N; i++)
  {
    for (j = 0; j <= i; j++)
      {
        ss = 0.0;
        for (k = 0; k <= j; k++)
          {
            for (l = 0; l <= (i - j); l++)
              {
                ss += MGET(gc_ij, k, l) * MGET(H, j, k) * MGET(H, (i - j), l);
```

```
            }
          }
        MLET(gc_alpha, i, j) = ss / pow(2, 0.5 * i) / pow(GET(cm, j) * GET(cm,

      }
  }

pnl_mat_free(&gc_ij);



Hx1 = pnl_vect_create(N + 1);
Hx2 = pnl_vect_copy(Hx1);
Hxx1 = pnl_vect_copy(Hx1);
Hxx2 = pnl_vect_copy(Hx1);
Hxxx1 = pnl_vect_copy(Hx1);
Hxxx2 = pnl_vect_copy(Hx1);
for (i = 0; i <= N; i++)
  {
    hh1 = 0.0;
    hh2 = 0.0;
    hh3 = 0.0;
    hh4 = 0.0;
    hh5 = 0.0;
    hh6 = 0.0;
    for (j = 0; j <= i; j++)
      {
        hh1 += MGET(H, i, j) * pow(GET(xpp, 0), j);
        hh2 += MGET(H, i, j) * pow(GET(xpp, 1), j);
        hh3 += MGET(H, i, j) * pow(GET(xpp, 0), j - 1) * j;
        hh4 += MGET(H, i, j) * pow(GET(xpp, 1), j - 1) * j;
        hh5 += MGET(H, i, j) * pow(GET(xpp, 0), j - 2) * j * (j - 1);
        hh6 += MGET(H, i, j) * pow(GET(xpp, 1), j - 2) * j * (j - 1);
      }
    LET(Hx1, i) = hh1;
    LET(Hx2, i) = hh2;
    LET(Hxx1, i) = hh3;
    LET(Hxx2, i) = hh4;
    LET(Hxxx1, i) = hh5;
    LET(Hxxx2, i) = hh6;
```

```
  }



dims2[0] = 2;
dims2[1] = 2;
dims2[2] = N + 1;
dims2[3] = N + 1;

fff3 = pnl_hmat_create(4, dims2);

fff = pnl_mat_create(N + 1, N + 1);
fff1 = pnl_mat_copy(fff);
fff2 = pnl_mat_copy(fff);
for (i = 0; i <= N; i++)
  {
    tab4[2] = i;
    for (j = 0; j <= N; j++)
      {
        tab4[3] = j;
        MLET(fff, i, j) = GET(Hx1, i) * GET(Hx2, j) / pow(2, (i + j) * 0.5);
        MLET(fff1, i, j) = GET(Hxx1, i) * GET(Hx2, j) / pow(2, (i + j) * 0.5);
        MLET(fff2, i, j) = GET(Hx1, i) * GET(Hxx2, j) / pow(2, (i + j) * 0.5);
        tab4[0] = 0;
        tab4[1] = 0;
        pnl_hmat_set(fff3, tab4, GET(Hxxx1, i)*GET(Hx2, j) / pow(2, (i + j) *
        tab4[1] = 1;
        pnl_hmat_set(fff3, tab4, GET(Hxx1, i)*GET(Hxx2, j) / pow(2, (i + j) *
        tab4[0] = 1;
        pnl_hmat_set(fff3, tab4, GET(Hx1, i)*GET(Hxxx2, j) / pow(2, (i + j) *
        tab4[1] = 0;
        pnl_hmat_set(fff3, tab4, GET(Hxx1, i)*GET(Hxx2, j) / pow(2, (i + j) *
      }
  }



dims[0] = N + 1;
dims[1] = 2;
dims[2] = 2;
```

```
Gammam = pnl_hmat_create_from_double(3, dims, 0.0);

dims[0] = 2;
dims[1] = 2;
dims[2] = N + 1;
ffmxx = pnl_hmat_create_from_double(3, dims, 0.0);

vm = pnl_vect_create_from_double(N + 1, 0.0);
thetam = pnl_vect_copy(vm);
ffmx = pnl_mat_create_from_double(2, N + 1, 0.0);
deltam = pnl_mat_create_from_double(2, N + 1, 0.0);
ph = pnl_mat_create_from_double(N + 1, N + 1, 0.0);
LA = pnl_mat_create_from_double(N + 1, N + 1, 0.0);
LLmat = pnl_mat_copy(LA);
LL = pnl_vect_create(N + 1);
ffm = pnl_vect_create(N + 1);
g2 = pnl_vect_create(N + 1);
pnl_vect_resize(sup, N + 1);
supmat = pnl_mat_create(2, 2);
supmat2 = pnl_mat_copy(supmat);

Sginv = pnl_mat_copy(Sigma);
MLET(Sginv, 0, 0) = MGET(Sigma, 1, 1);
MLET(Sginv, 1, 1) = MGET(Sigma, 0, 0);
MLET(Sginv, 1, 0) = -MGET(Sigma, 1, 0);
MLET(Sginv, 0, 1) = -MGET(Sigma, 0, 1);
pnl_mat_mult_double(Sginv, 1.0 / (MGET(Sigma, 0, 0)*MGET(Sigma, 1, 1) - MGET(S

Cmu = pnl_mat_mult_mat(Mu, C);
pnl_mat_dgemm('N', 'N', 1.0, Sginv, Cmu, 0.0, tmp_matrix);
pnl_mat_dgemm('T', 'N', 1.0, tmp_matrix, tmp_matrix, 0.0, Cmu);


for (i = 0; i <= N; i++)
  {

    dims[2] = i + 1;

    pnl_mat_resize(ph, (i + 1), (i + 1));
    pnl_mat_resize(LA, (i + 1), (i + 1));
    pnl_mat_resize(ffmx, 2, (i + 1));
```

```
pnl_vect_resize(LL, (i + 1));
pnl_vect_resize(ffm, (i + 1));
pnl_vect_resize(g2, (i + 1));
pnl_vect_resize(sup, (i + 1));
pnl_mat_resize(LLmat, (i + 1), (i + 1));
pnl_hmat_resize(ffmxx, 3, dims);

tab2[0] = i;

if ((int)fmod(i, 2) == 0)
  {
    m = (int)(0.5 * i);
    LET(LL, m) = GET(la0, m);
    for (j = 0; j <= i; j++)
      {
        MLET(ph, j, m) = MGET(phi0, m, j) * pow((2 * beta * beta + 4 * alp
      }
    if (m > 0)
      {
        for (j = 0; j < m; j++)
          {
            tab[0] = m - j - 1;
            tab[1] = 2 * j + 1;
            LET(LL, j) = MGET(lam, m - j - 1, 2 * j + 1);
            LET(LL, j + m + 1) = MGET(lap, m - j - 1, 2 * j + 1);
            for (k = 0; k <= i; k++)
              {
                tab[2] = k;
                MLET(ph, k, j) = pnl_hmat_get(phim, tab) *
                                 pow(((alpha + la) * (alpha + la) + beta *
                                     -(2 * j + 1) * 0.5) * pow((2 * beta *
                MLET(ph, k, j + m + 1) = pnl_hmat_get(phip, tab) *
                                         pow(((alpha - la) * (alpha - la)
                                             -(2 * j + 1) * 0.5) * pow((2
              }
          }
      }
  }
else
  {
    m = (int)((i + 1) * 0.5);
```

```
        for (j = 0; j < m; j++)
          {
            tab[0] = m - j - 1;
            tab[1] = 2 * j;
            LET(LL, j) = MGET(lam, m - j - 1, 2 * j);
            LET(LL, j + m) = MGET(lap, m - j - 1, 2 * j);
            for (k = 0; k <= i; k++)
              {
                tab[2] = k;
                MLET(ph, k, j) = pnl_hmat_get(phim, tab) * pow(((alpha + la) *
                MLET(ph, k, j + m) = pnl_hmat_get(phip, tab) * pow(((alpha - l
              }
          }

      }
    for (j = 0; j <= i; j++)
      {
        LET(ffm, j) = MGET(fff, i - j, j);
        LET(g2, j) = MGET(gc_alpha, i, j);
        MLET(ffmx, 0, j) = MGET(fff1, i - j, j);
        MLET(ffmx, 1, j) = MGET(fff2, i - j, j);

        tab4[2] = i - j;
        tab4[3] = j;
        tab3[2] = j;

        tab3[0] = 0;
        tab3[1] = 0;
        tab4[0] = 0;
        tab4[1] = 0;
        pnl_hmat_set(ffmxx, tab3, pnl_hmat_get(fff3, tab4));

        tab3[0] = 1;
        tab3[1] = 0;
        tab4[0] = 1;
        tab4[1] = 0;
        pnl_hmat_set(ffmxx, tab3, pnl_hmat_get(fff3, tab4));

        tab3[0] = 0;
        tab3[1] = 1;
        tab4[0] = 0;
```

```
    tab4[1] = 1;
    pnl_hmat_set(ffmxx, tab3, pnl_hmat_get(fff3, tab4));

    tab3[0] = 1;
    tab3[1] = 1;
    tab4[0] = 1;
    tab4[1] = 1;
    pnl_hmat_set(ffmxx, tab3, pnl_hmat_get(fff3, tab4));

    for (k = 0; k <= i; k++)
      {
        MLET(LA, j, k) = 0.0;
        MLET(LLmat, j, k) = 0.0;
      }
    MLET(LA, j, j) = exp(-GET(LL, j) * T0);
    MLET(LLmat, j, j) = -GET(LL, j);
  }


pnl_mat_syslin(sup, ph, g2);
LAsup = pnl_mat_mult_vect(LA, sup);
phsup = pnl_mat_mult_vect(ph, LAsup);
pnl_vect_clone(sup2, phsup);
LET(vm, i) = pnl_vect_scalar_prod(ffm, phsup);
pnl_mat_mult_vect_inplace(phsup, LLmat, LAsup);
pnl_mat_mult_vect_inplace(sup3, ph, phsup);
LET(thetam, i) = pnl_vect_scalar_prod(ffm, sup3);

pnl_mat_syslin(sup, ph, g2);

pnl_mat_mult_vect_inplace(LAsup, LA, sup);
pnl_mat_mult_vect_inplace(sup, ph, LAsup);

pnl_vect_free(&LAsup);
pnl_vect_free(&phsup);

pnl_mat_set_double(supmat, 0.0);
for (j = 0; j <= i; j++)
  {
    tab3[2] = j;
```

```
    tab3[0] = 0;
    tab3[1] = 0;
    MLET(supmat2, 0, 0) = pnl_hmat_get(ffmxx, tab3);
    tab3[0] = 1;
    tab3[1] = 0;
    MLET(supmat2, 1, 0) = pnl_hmat_get(ffmxx, tab3);
    tab3[0] = 0;
    tab3[1] = 1;
    MLET(supmat2, 0, 1) = pnl_hmat_get(ffmxx, tab3);
    tab3[0] = 1;
    tab3[1] = 1;
    MLET(supmat2, 1, 1) = pnl_hmat_get(ffmxx, tab3);

    pnl_mat_mult_double(supmat2, GET(sup, j));
    pnl_mat_plus_mat(supmat, supmat2);
  }


tab3[0] = i;

tab3[1] = 0;
tab3[2] = 0;
pnl_hmat_set(Gammam, tab3, MGET(supmat, 0, 0));
tab3[1] = 1;
tab3[2] = 0;
pnl_hmat_set(Gammam, tab3, MGET(supmat, 1, 0));
tab3[1] = 0;
tab3[2] = 1;
pnl_hmat_set(Gammam, tab3, MGET(supmat, 0, 1));
tab3[1] = 1;
tab3[2] = 1;
pnl_hmat_set(Gammam, tab3, MGET(supmat, 1, 1));

pnl_mat_mult_vect_inplace(ffsup2, ffmx, sup2);
pnl_mat_syslin(sup2, Sigma, ffsup2);
pnl_mat_mult_vect_inplace(ffsup2, Mu, sup2);
pnl_mat_mult_vect_transpose_inplace(sup2, C, ffsup2);

MLET(deltam, 0, i) = GET(sup2, 0);
MLET(deltam, 1, i) = GET(sup2, 1);
```

```
  }

*call_price = 0.0;
LET(delta, 0) = 0.0;
LET(delta, 1) = 0.0;
pnl_mat_set_double(Ga, 0.0);
*Theta = 0.0;

pnl_vect_clone(sup2, b);
pnl_mat_lAxpby(1.0, Y, xp, -1.0, sup2);
pnl_mat_syslin_inplace(Sigma, sup2);
pnl_vect_mult_double(sup2, -1);


pnl_mat_resize(supmat, 1, 2);
pnl_mat_set_double(supmat, 0.0);
pnl_mat_resize(supmat2, 2, 1);
pnl_mat_set_double(supmat2, 0.0);
supmat3 = pnl_mat_create_from_double(2, 2, 0.0);
supmat4 = pnl_mat_create_from_double(1, 2, 0.0);

MLET(supmat2, 0, 0) = GET(sup2, 0);
MLET(supmat2, 1, 0) = GET(sup2, 1);

for (i = 0; i <= N; i++)
  {
    *call_price += GET(vm, i);
    *Theta += GET(thetam, i);


    pnl_mat_get_col(sup, deltam, i);
    pnl_vect_plus_vect(delta, sup);

    pnl_vect_axpby(*call_price, sup2, 1.0, delta);


    pnl_mat_dgemm('N', 'N', 1.0, Y, Sginv, 0.0, tmp_matrix);
    pnl_mat_mult_mat_inplace(supmat3, Sginv, tmp_matrix);
    pnl_mat_mult_double(supmat3, -(*call_price));
    pnl_mat_minus_mat(Ga, supmat3);
```

```
    tab3[0] = i;

    tab3[1] = 0;
    tab3[2] = 0;
    MLET(supmat3, 0, 0) = pnl_hmat_get(Gammam, tab3);
    tab3[1] = 1;
    tab3[2] = 0;
    MLET(supmat3, 1, 0) = pnl_hmat_get(Gammam, tab3);
    tab3[1] = 0;
    tab3[2] = 1;
    MLET(supmat3, 0, 1) = pnl_hmat_get(Gammam, tab3);
    tab3[1] = 1;
    tab3[2] = 1;
    MLET(supmat3, 1, 1) = pnl_hmat_get(Gammam, tab3);
    pnl_mat_plus_mat(Ga, supmat3);

    MLET(supmat, 0, 0) = GET(sup, 0);
    MLET(supmat, 0, 1) = GET(sup, 1);
    MLET(supmat4, 0, 0) = GET(delta, 0);
    MLET(supmat4, 0, 1) = GET(delta, 1);


    pnl_mat_dgemm('N', 'N', 1.0, supmat2, supmat, 1.0, Ga);
    pnl_mat_dgemm('N', 'N', 1.0, supmat2, supmat4, 1.0, Ga);
  }

actu = exp(-0.5 * pnl_mat_scalar_prod(Y, xp, xp)
           + pnl_vect_scalar_prod(b, xp));
*call_price *= actu;
*Theta *= actu;
pnl_vect_mult_double(delta, actu);
pnl_mat_mult_double(Ga, actu);


bond = exp(rC2 + pnl_vect_scalar_prod(rB2, y)
           + pnl_mat_scalar_prod(rA2, y, y));

bond1 = exp(rC1 + pnl_vect_scalar_prod(rB1, y)
            + pnl_mat_scalar_prod(rA1, y, y));
*put_price = (K * bond1 - bond) + (*call_price);
```

```
pnl_mat_mult_mat_inplace(tmp_matrix, Sginv, rA2);
pnl_mat_mult_vect_inplace(sup, tmp_matrix, y);

pnl_mat_lAxpby(1.0, Sginv, rB2, 2.0, sup);


pnl_mat_mult_mat_inplace(tmp_matrix, Sginv, rA1);
pnl_mat_mult_vect_inplace(sup2, tmp_matrix, sup);

pnl_mat_lAxpby(1.0, Sginv, rB1, 2.0, sup2);

MLET(supmat, 0, 0) = GET(sup, 0);
MLET(supmat, 0, 1) = GET(sup, 1);

pnl_mat_clone(tmp_matrix, supmat);
pnl_mat_dgemm('T', 'N', 1.0, tmp_matrix, tmp_matrix, 0.0, supmat);


pnl_mat_dgemm('N', 'N', 1.0, rA2, Sginv, 0.0, tmp_matrix);
pnl_mat_dgemm('N', 'N', 1.0, Sginv, tmp_matrix, 0.0, supmat2);
pnl_mat_mult_double(supmat2, 2);
pnl_mat_plus_mat(supmat, supmat2);

MLET(supmat3, 0, 0) = GET(sup2, 0);
MLET(supmat3, 0, 1) = GET(sup2, 1);

pnl_mat_clone(tmp_matrix, supmat3);
pnl_mat_dgemm('T', 'N', 1.0, tmp_matrix, tmp_matrix, 0.0, supmat3);


pnl_mat_mult_mat_inplace(tmp_matrix, rA1, Sginv);

pnl_mat_mult_mat_inplace(supmat2, Sginv, tmp_matrix);
pnl_mat_mult_double(supmat2, 2);
pnl_mat_plus_mat(supmat3, supmat2);

pnl_vect_plus_vect(sup, delta);
pnl_vect_mult_double(sup, -1);
pnl_vect_clone(delta_put, sup2);
```

```
pnl_vect_mult_double(delta_put, K);
pnl_vect_plus_vect(delta_put, sup);


pnl_mat_plus_mat(supmat, Ga);
pnl_mat_mult_double(supmat, -1);
pnl_mat_clone(Ga_put, supmat3);
pnl_mat_mult_double(Ga_put, K);
pnl_mat_plus_mat(Ga_put, supmat);

pnl_mat_free(&Z);
pnl_mat_free(&errY);
pnl_vect_free(&ffsup2);
pnl_mat_free(&Y);
pnl_mat_free(&tmp_matrix);
pnl_vect_free(&tmp_vector);
pnl_mat_free(&TKappa);
pnl_mat_free(&TKappaSigma);
pnl_mat_free(&TSigma);
pnl_vect_free(&a);
pnl_vect_free(&b);
pnl_vect_free(&sup);
pnl_vect_free(&sup2);
pnl_vect_free(&sup3);
pnl_vect_free(&y);
pnl_vect_free(&xp);
pnl_vect_free(&xpp);
pnl_vect_free(&cm);
pnl_mat_free(&H);
pnl_mat_free(&phi0);
pnl_mat_free(&lap);
pnl_mat_free(&lam);
pnl_vect_free(&la0);
pnl_hmat_free(&phip);
pnl_hmat_free(&phim);
pnl_vect_free(&mu);
pnl_mat_free(&C);
pnl_mat_free(&D);
pnl_mat_free(&A);
pnl_mat_free(&Mu);
pnl_mat_free(&Mu_inv);
```

```
  pnl_mat_free(&rA);
  pnl_vect_free(&rB);
  pnl_mat_free(&rA1);
  pnl_vect_free(&rB1);
  pnl_mat_free(&rA2);
  pnl_vect_free(&rB2);
  pnl_mat_free(&gc_alpha);
  pnl_vect_free(&Hx1);
  pnl_vect_free(&Hx2);
  pnl_mat_free(&fff);
  pnl_vect_free(&vm);
  pnl_mat_free(&CSigma);
  pnl_mat_free(&TSigma);
  pnl_mat_free(&deltam);
  pnl_vect_free(&thetam);
  pnl_mat_free(&LLmat);
  pnl_mat_free(&ffmx);
  pnl_hmat_free(&ffmxx);
  pnl_hmat_free(&Gammam);
  pnl_mat_free(&LA);
  pnl_mat_free(&ph);
  pnl_vect_free(&ffm);
  pnl_vect_free(&g2);
  pnl_vect_free(&LL);
  pnl_vect_free(&Hxx1);
  pnl_vect_free(&Hxx2);
  pnl_vect_free(&Hxxx1);
  pnl_vect_free(&Hxxx2);
  pnl_mat_free(&fff1);
  pnl_mat_free(&fff2);
  pnl_hmat_free(&fff3);
  pnl_mat_free(&Sginv);
  pnl_mat_free(&Cmu);
  pnl_mat_free(&supmat);
  pnl_mat_free(&supmat2);
  pnl_mat_free(&supmat3);
  pnl_mat_free(&supmat4);

}
```

```
/*Option on Bond in the QTSM2D Model*/
static int zbc_qtsm2d(PnlVect *x, double d0, PnlVect *d, PnlVect *theta, PnlVect
{
  double Strike;
  PnlMat *Sigma, * Kappa, *Gamma, *Ga, *Ga_p;
  PnlVect  *delta, *delta_p;
  int size = 2;
  double Call_Price, Put_Price, Theta;

  Strike = p->Par[0].Val.V_DOUBLE;

  Sigma = pnl_mat_create_from_double(size, size, 0.0);
  Kappa = pnl_mat_create_from_double(size, size, 0.0);
  Gamma = pnl_mat_create_from_double(size, size, 0.0);

  Ga = pnl_mat_create_from_double(size, size, 0.0);
  delta = pnl_vect_create_from_double(size, 0.0);
  Ga_p = pnl_mat_create_from_double(size, size, 0.0);
  delta_p = pnl_vect_create_from_double(size, 0.0);

  MLET(Kappa, 0, 0) = GET(KappaV, 0);
  MLET(Kappa, 0, 1) = GET(KappaV, 1);
  MLET(Kappa, 1, 0) = GET(KappaV, 2);
  MLET(Kappa, 1, 1) = GET(KappaV, 3);

  MLET(Sigma, 0, 0) = GET(SigmaV, 0);
  MLET(Sigma, 0, 1) = GET(SigmaV, 1);
  MLET(Sigma, 1, 0) = GET(SigmaV, 1);
  MLET(Sigma, 1, 1) = GET(SigmaV, 2);

  MLET(Gamma, 0, 0) = GET(GammaV, 0);
  MLET(Gamma, 0, 1) = GET(GammaV, 1);
  MLET(Gamma, 1, 0) = GET(GammaV, 1);
  MLET(Gamma, 1, 1) = GET(GammaV, 2);

  Price(Option_Maturity, Bond_Maturity, Strike, N, theta, Sigma, Kappa, d0, d,
        Gamma, x, &Call_Price, &Put_Price, &Theta, delta, Ga, delta_p, Ga_p);

  //Print Deltas Values
  //pnl_vect_print(delta);
```

```
  //Print Gamma Values
  //pnl_mat_print(Ga);

  /*Price*/
  if ((p->Compute) == &Call)
    *price = Call_Price;
  else
    *price = Put_Price;

  pnl_mat_free(&Sigma);
  pnl_mat_free(&Kappa);
  pnl_mat_free(&Gamma);
  pnl_vect_free(&delta);
  pnl_mat_free(&Ga);
  pnl_vect_free(&delta_p);
  pnl_mat_free(&Ga_p);

  return OK;
}

int CALC(AP_QTSM2D)(void *Opt, void *Mod, PricingMethod *Met)
{
  TYPEOPT *ptOpt = (TYPEOPT *)Opt;
  TYPEMOD *ptMod = (TYPEMOD *)Mod;

  return zbc_qtsm2d(ptMod->x.Val.V_PNLVECT,
                    ptMod->d0.Val.V_PDOUBLE,
                    ptMod->d.Val.V_PNLVECT,
                    ptMod->theta.Val.V_PNLVECT,
                    ptMod->SigmaV.Val.V_PNLVECT,
                    ptMod->GammaV.Val.V_PNLVECT,
                    ptMod->KappaV.Val.V_PNLVECT,
                    ptOpt->BMaturity.Val.V_DATE,
                    ptOpt->OMaturity.Val.V_DATE,
                    ptOpt->PayOff.Val.V_NUMFUNC_1,
                    Met->Par[0].Val.V_INT2,
                    &(Met->Res[0].Val.V_DOUBLE));
}

static int CHK_OPT(AP_QTSM2D)(void *Opt, void *Mod)
{
```

```
  if ((strcmp(((Option *)Opt)->Name, "ZeroCouponCallBondEuro") == 0)
      || (strcmp(((Option *)Opt)->Name, "ZeroCouponPutBondEuro") == 0))
    return OK;
  else
    return WRONG;
}

#endif //PremiaCurrentVersion
static int MET(Init)(PricingMethod *Met, Option *Opt)
{
  if (Met->init == 0)
    {
      Met->HelpFilenameHint = "AP_EigenfunctionExpansion_ZBO";
      Met->init = 1;
      Met->Par[0].Val.V_INT = 30;
    }

  return OK;
}

PricingMethod MET(AP_QTSM2D) =
{
  "AP_EigenfunctionExpansion_ZBO",
  {{"TimeStepNumber", INT, {100}, ALLOW}, {" ", PREMIA_NULLTYPE, {0}, FORBID}},
  CALC(AP_QTSM2D),
  { {"Price", DOUBLE, {100}, FORBID},
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CHK_OPT(AP_QTSM2D),
  CHK_ok,
  MET(Init)
} ;
```