```
   Help
extern "C"{
#include  "affine3d_stdi.h"

#include "math/highdim_solver/laspack/highdim_vector.h"
#include "math/highdim_solver/laspack/qmatrix.h"
#include "math/highdim_solver/laspack/highdim_matrix.h"
#include "math/highdim_solver/laspack/operats.h"

#include "math/highdim_solver/fd_solver.h"
#include "math/highdim_solver/fd_operators.h"
#include "math/highdim_solver/fd_operators_easy.h"
#include "math/highdim_solver/error.h"
}
#include <cmath>
using namespace std;

typedef double (*paramf_t)(double, double);


typedef struct _Model
{
  // Independent model parameters

  double rho12, rho13, rho23;
  double *kappa, *x0, *sigma;
  double delta;
  double T,dT,N,C,K;
  unsigned N1,N2,N3;      // Number of grid points per dim
    ension

  // Dependent model parameters
  // Solution domain definition
  double xL,xR;
  double yL,yR;
  double zL,zR;

  // Offset
  int offx, offy, offz;

} Affine3DModel;
```

```
Affine3DModel MAffine3D;

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
     (2008+2) //The "#else" part of the code will be freely av
    ailable after the (year of creation of this file + 2)
#else

static void setup(Affine3DModel *m)
{
  double h,foff;

  m->xL = m->x0[0]-8.0*fabs(m->x0[0]);
  m->xR = m->x0[0]+8.0*fabs(m->x0[0]);
  m->yL = m->x0[1]-8.0*fabs(m->x0[1]);
  m->yR = m->x0[1]+8.0*fabs(m->x0[1]);
  m->zL = m->x0[2]-8.0*fabs(m->x0[2]);
  m->zR = m->x0[2]+8.0*fabs(m->x0[2]);

  h = (m->xR-m->xL)/(m->N1-1);
  foff = (m->x0[0]-m->xL)/h;
  m->offx = (int)ceil(foff);
  m->xL -= (ceil(foff)-foff)*h;
  m->xR -= (ceil(foff)-foff)*h;

  h = (m->yR-m->yL)/(m->N2-1);
  foff = (m->x0[1]-m->yL)/h;
  m->offy = (int)ceil(foff);
  m->yL -= (ceil(foff)-foff)*h;
  m->yR -= (ceil(foff)-foff)*h;

  h = (m->zR-m->zL)/(m->N3-1);
  foff = (m->x0[2]-m->zL)/h;
  m->offz = (int)ceil(foff);
  m->zL -= (ceil(foff)-foff)*h;
  m->zR -= (ceil(foff)-foff)*h;
}

// // Model to solution space
// //
// static void c_m2s(Affine3DModel *m, double v1, double v2
```

```
    , double v3,
//                        double *w1, double *w2, double *w3)
// {
//   if(w1) *w1 = (v1-m->xL)/(m->xR-m->xL);
//   if(w2) *w2 = (v2-m->yL)/(m->yR-m->yL);
//   if(w3) *w3 = (v3-m->zL)/(m->zR-m->zL);
// }

static void v_m2s(Affine3DModel *m, double t, double V,
    double *W)
{
  *W = V;
}

// static void m2s(Affine3DModel *m, double t, double v1,
    double v2, double v3, double V,
//                      double *w1, double *w2, double *w3,
    double *W)
// {
//   c_m2s(m,v1,v2,v3,w1,w2,w3);
//   v_m2s(m,t,V,W);
// }

// Solution to model space

static void c_s2m(Affine3DModel *m, double w1, double w2,
    double w3,
                      double *v1, double *v2, double *v3)
{
  if(v1) *v1 = (m->xR-m->xL)*w1+m->xL;
  if(v2) *v2 = (m->yR-m->yL)*w2+m->yL;
  if(v3) *v3 = (m->zR-m->zL)*w3+m->zL;
}

static void v_s2m(Affine3DModel *m, double tau, double W,
    double *V)
{
  *V = W;
}

// static void s2m(Affine3DModel *m, double tau, double w1,
```

```
     double w2, double w3, double W,
//                    double *v1, double *v2, double *v3,
    double *V)
// {
//   c_s2m(m,w1,w2,w3,v1,v2,v3);
//   v_s2m(m,tau,W,V);
// }

// Option functions

static double f_phi(double s, double k)
{
  return (1-exp(-1.0*k*s))/k;
}

static double f_psi(double s, int i, int j)
{
  return MAffine3D.sigma[i]*MAffine3D.sigma[j]/(MAffine3D.
    kappa[i]*MAffine3D.kappa[j])*
           (s-f_phi(s,MAffine3D.kappa[i])-f_phi(s,MAffine3
    D.kappa[j])+
            f_phi(s,MAffine3D.kappa[i]+MAffine3D.kappa[j]))
    ;
}

static double f_B(double s, int j)
{
  if(!j)
  {
    return -1.0*MAffine3D.delta*s+0.5*(f_psi(s,0,0)+f_psi(
    s,1,1)+f_psi(s,2,2))+
              MAffine3D.rho12*f_psi(s,0,1)+MAffine3D.rho13*
    f_psi(s,0,2)+MAffine3D.rho23*f_psi(s,1,2);
  }

  return f_phi(s,MAffine3D.kappa[j-1]);
}

static double f_P(double x[], double t, double T)
{
  return exp(f_B(T-t,0)-f_B(T-t,1)*x[0]-f_B(T-t,2)*x[1]-f_
```

```
    B(T-t,3)*x[2]);
}

static double f_CB(double T, double x[])
{
  double CB = 0.0, t = MAffine3D.T+MAffine3D.dT;
  int i;

  for(i=0; i<MAffine3D.N; i++)
  {
    CB += MAffine3D.C*f_P(x,T,t);
    t += MAffine3D.dT;
  }

  return CB;
}

static double boundary(double x[], double t)
{
  double val = f_CB(t,x) - MAffine3D.K*f_P(x,t,MAffine3D.T)
    ;

  return val > 0.0 ? val : 0;
}

double payoff(double x[])
{
  return boundary(x,MAffine3D.T);
}

////////////////////////////////////////////////////
// Initial & boundary conditions
//
static int ic_f_next_elem(struct _FDSolver *s, FDSolverVec
    torFiller *f,
                    unsigned *c, double *v)
{
  double x[3];

  x[0] = 0.0;
  x[1] = 0.0;
```

```
  x[2] = 0.0;

  c_s2m(&MAffine3D,(double)c[0]/(s->size[0]-1),(double)c[1]
    /(s->size[1]-1),
          (double)c[2]/(s->size[2]-1),x,x+1,x+2);
  v_m2s(&MAffine3D,MAffine3D.T-s->t,payoff(x),v);

  return 0;
}

static int b_f_next_elem(struct _FDSolver *s, FDSolverVec
    torFiller *f,
                  unsigned *c, double *v)
{
  double x[3];

  c_s2m(&MAffine3D,(double)c[0]/(s->size[0]-1),(double)c[1]
    /(s->size[1]-1),
          (double)c[2]/(s->size[2]-1),x,x+1,x+2);
  v_m2s(&MAffine3D,MAffine3D.T-s->t,boundary(x,MAffine3D.T-
    s->t),v);

  return 0;
}

////////////////////////////////////////////////
// Equation definition
//

static void eq_first_def(FDOperatorJam *j)
{
  unsigned k;

  for(k=0; k < j->dim; k++)
    FIRST_SPATIAL_DERIVATIVE_CENTERED_MASK(j,k);
}

static int eq_first_apply(FDSolver *s, FDOperatorJam *j, un
    signed *c, void *d,
                        double factor)
{
```

```
  double x,y,z;
  double wx;

  wx = MAffine3D.xR - MAffine3D.xL;

  c_s2m(&MAffine3D,(double)c[0]/(s->size[0]-1),(double)c[1]
    /(s->size[1]-1),
           (double)c[2]/(s->size[2]-1), &x, &y, &z);

  FIRST_SPATIAL_DERIVATIVE_CENTERED_SET(j,0,-1.0*factor*x*
    MAffine3D.kappa[0]/wx*(MAffine3D.N1-1)*s->deltaT);
  FIRST_SPATIAL_DERIVATIVE_CENTERED_SET(j,1,-1.0*factor*y*
    MAffine3D.kappa[1]/wx*(MAffine3D.N2-1)*s->deltaT);
  FIRST_SPATIAL_DERIVATIVE_CENTERED_SET(j,2,-1.0*factor*z*
    MAffine3D.kappa[2]/wx*(MAffine3D.N3-1)*s->deltaT);

  return 0;
}

static void eq_second_def(FDOperatorJam *j)
{
  unsigned k,h;

  ZERO_ORDER_MASK(j);

  for(k=0; k < j->dim; k++)
    UNIFORM_SECOND_SPATIAL_DERIVATIVE_CENTERED_MASK(j,k);

  for(h=1; h<j->dim; h++)
    for(k=0; k<h; k++)
      MIXED_SECOND_SPATIAL_DERIVATIVE_CENTERED_BOUCHUT_MAS
    K(j,h,k);
}

static int eq_second_apply(FDSolver *s, FDOperatorJam *j,
    unsigned *c, void *d,
                    double factor)
{
  double x,y,z;
  double wx,wy,wz;
```

```
  wx = MAffine3D.xR - MAffine3D.xL;
  wy = MAffine3D.yR - MAffine3D.yL;
  wz = MAffine3D.zR - MAffine3D.zL;

  c_s2m(&MAffine3D,(double)c[0]/(s->size[0]-1),(double)c[1]
    /(s->size[1]-1),
           (double)c[2]/(s->size[2]-1), &x, &y, &z);

  ZERO_ORDER_SET(j,-1.0*factor*(MAffine3D.delta+x+y+z)*s->
    deltaT);

  UNIFORM_SECOND_SPATIAL_DERIVATIVE_CENTERED_SET(j,0,
    factor*0.5*pow(MAffine3D.sigma[0]/wx*(MAffine3D.N1-1),2
    )*s->deltaT);
  UNIFORM_SECOND_SPATIAL_DERIVATIVE_CENTERED_SET(j,1,
    factor*0.5*pow(MAffine3D.sigma[1]/wy*(MAffine3D.N2-1),2
    )*s->deltaT);
  UNIFORM_SECOND_SPATIAL_DERIVATIVE_CENTERED_SET(j,2,
    factor*0.5*pow(MAffine3D.sigma[2]/wz*(MAffine3D.N3-1),2
    )*s->deltaT);

  MIXED_SECOND_SPATIAL_DERIVATIVE_CENTERED_BOUCHUT_SET(j,1,
    0,
    factor*(MAffine3D.rho12*MAffine3D.sigma[0]*MAffine3D.si
    gma[1])/(wx*wy)*(MAffine3D.N1-1)*(MAffine3D.N2-1)*s->deltaT)
    ;
  MIXED_SECOND_SPATIAL_DERIVATIVE_CENTERED_BOUCHUT_SET(j,2,
    0,
    factor*(MAffine3D.rho13*MAffine3D.sigma[0]*MAffine3D.si
    gma[2])/(wx*wz)*(MAffine3D.N1-1)*(MAffine3D.N3-1)*s->deltaT)
    ;
  MIXED_SECOND_SPATIAL_DERIVATIVE_CENTERED_BOUCHUT_SET(j,2,
    1,
    factor*(MAffine3D.rho23*MAffine3D.sigma[1]*MAffine3D.si
    gma[2])/(wy*wz)*(MAffine3D.N2-1)*(MAffine3D.N3-1)*s->deltaT)
    ;

  return 0;
}

//////////////////////////////////////////////////
```

```
// Explicit scheme
//
static int ex_eq_def_c(FDOperatorJam *j, void *d)
{
  FIRST_TIME_DERIVATIVE_FORWARD_MASK(j);

  eq_first_def(j);
  eq_second_def(j);

  return 0;
}

static int ex_eq_apply_c(FDSolver *s, FDOperatorJam *j, un
    signed *c, void *d)
{
  FIRST_TIME_DERIVATIVE_FORWARD_SET(j,1.);

  eq_first_apply(s,j,c,d,1.0);
  eq_second_apply(s,j,c,d,1.0);

  return 0;
}

static int ex_eq_def_n(FDOperatorJam *j, void *d)
{
  FIRST_TIME_DERIVATIVE_FORWARD_MASK(j);

  return 0;
}

static int ex_eq_apply_n(FDSolver *s, FDOperatorJam *j, un
    signed *c, void *d)
{
  FIRST_TIME_DERIVATIVE_FORWARD_SET(j,1.);

  return 0;
}

////////////////////////////////////////////////
// Crank-Nicolson scheme
//
```

```
static int cn_eq_def_c(FDOperatorJam *j, void *d)
{
  FIRST_TIME_DERIVATIVE_FORWARD_MASK(j);

  eq_first_def(j);
  eq_second_def(j);

  return 0;
}

static int cn_eq_apply_c(FDSolver *s, FDOperatorJam *j, un
    signed *c, void *d)
{
  FIRST_TIME_DERIVATIVE_FORWARD_SET(j,1.);

  eq_first_apply(s,j,c,d,1.0);
  eq_second_apply(s,j,c,d,0.5);

  return 0;
}

static int cn_eq_def_n(FDOperatorJam *j, void *d)
{
  FIRST_TIME_DERIVATIVE_FORWARD_MASK(j);

  eq_second_def(j);

  return 0;
}

static int cn_eq_apply_n(FDSolver *s, FDOperatorJam *j, un
    signed *c, void *d)
{
  FIRST_TIME_DERIVATIVE_FORWARD_SET(j,1.);

  eq_second_apply(s,j,c,d,-0.5);

  return 0;
}

static int FD_AFFINE3D(double T_swap,double T_option,
```

```
    double coupon,double tenor,double x01,double x02,double x03,
    double t,NumFunc_1  *p,double sigma1,double sigma2,double sigma3
    ,double shift,double k1,double k2,double k3,double rho12,
    double rho13,double rho23,int N1,int N2,int N3,double *ptprice)
{
  double K;
  double T,dt;
  int N;
  double kappa[3], x0[3], sigma[3];
  int k,h,offset;
  FDSolver s;
  FDSolverVectorFiller ic_f, b_f;
  FDSolverCoMatricesFiller AcBcf, AnBnf;
  FDOperatorJamCoMatricesFillerData jfdc_ex,jfdn_ex;
  FDOperatorJamCoMatricesFillerData jfdc_cn,jfdn_cn;

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

  kappa[0]=k1;
  kappa[1]=k2;
  kappa[2]=k3;
  sigma[0]=sigma1;
  sigma[1]=sigma2;
  sigma[2]=sigma3;
  x0[0]=x01;
  x0[1]=x02;
  x0[2]=x03;

  T=T_option;
  dt=tenor;
  N=(int)((T_swap-T_option)/dt);

  MAffine3D.kappa = kappa;
  MAffine3D.x0 = x0;
  MAffine3D.sigma = sigma;

  MAffine3D.rho12 = rho12;
  MAffine3D.rho13 = rho13;
  MAffine3D.rho23 = rho23;
  MAffine3D.delta = shift;
```

```
MAffine3D.T = T;
MAffine3D.dT = dt;
MAffine3D.N = N;
MAffine3D.C = coupon;
MAffine3D.K = K;

MAffine3D.N1 = N1 % 2 ? N1 : N1 + 1;
MAffine3D.N2 = N2 % 2 ? N2 : N2 + 1;
MAffine3D.N3 = N3 % 2 ? N3 : N3 + 1;

setup(&MAffine3D);

offset = (N1-2)*(N2-2)*(MAffine3D.offz-1)+(N1-2)*(MAffine
  3D.offy-1)+MAffine3D.offx;

s.dim = 3;
s.is_A_symmetric = FALSE;

// Evaluate CFL for explicit method
// Assumption: f(y,z) is increasing in its arguments
s.deltaT = pow(MAffine3D.xR-MAffine3D.xL,2)/(0.5*pow(((
  MAffine3D.N1-1)*MAffine3D.sigma[0]),2));

if (pow(MAffine3D.yR-MAffine3D.yL,2)/(0.5*pow(((MAffine3
  D.N2-1)*MAffine3D.sigma[1]),2)) < s.deltaT)
  s.deltaT = pow(MAffine3D.xR-MAffine3D.xL,2)/(0.5*pow(((
  MAffine3D.N2-1)*MAffine3D.sigma[1]),2));

if (pow(MAffine3D.zR-MAffine3D.zL,2)/(0.5*pow(((MAffine3
  D.N3-1)*MAffine3D.sigma[2]),2)) < s.deltaT)
 s.deltaT = pow(MAffine3D.zR-MAffine3D.zL,2)/(0.5*pow(((
  MAffine3D.N3-1)*MAffine3D.sigma[2]),2));

s.deltaT = 0.0001*s.deltaT;

s.size[0] = MAffine3D.N1;
s.size[1] = MAffine3D.N2;
s.size[2] = MAffine3D.N3;

ic_f.init = NULL;
ic_f.next_elem = ic_f_next_elem;
```

```
ic_f.finish = NULL;
ic_f.free = NULL;

b_f.init = NULL;
b_f.next_elem = b_f_next_elem;
b_f.finish = NULL;
b_f.free = NULL;

s.b_filler = &b_f;

// Explicit

s.is_fully_explicit = TRUE;
s.is_fully_implicit = FALSE;

FDOperatorJamCoMatricesFillerSet(&AcBcf,&jfdc_ex,ex_eq_de
  f_c,
                                  ex_eq_apply_c,NULL);
FDOperatorJamCoMatricesFillerSet(&AnBnf,&jfdn_ex,ex_eq_de
  f_n,
                                  ex_eq_apply_n,NULL);

if(FDSolverInit(&s, &ic_f, &AcBcf, &AnBnf)) return 1;

k = 1;

for(;k<=20;k++) FDSolverStep(&s);

// Crank-Nicolson

s.is_fully_explicit = FALSE;
s.is_fully_implicit = FALSE;

h = (int)ceil((MAffine3D.T-s.t)/(sqrt(s.deltaT)));
s.deltaT = (MAffine3D.T-s.t)/h;

FDOperatorJamCoMatricesFillerSet(&AcBcf,&jfdc_cn,cn_eq_de
  f_c,
                                  cn_eq_apply_c,NULL);
FDOperatorJamCoMatricesFillerSet(&AnBnf,&jfdn_cn,cn_eq_de
  f_n,
```

```
                                    cn_eq_apply_n,NULL);

  if(FDSolverResetMatrices(&s, &AcBcf, &AnBnf)) return 1;

  for(;k<=h+20;k++) FDSolverStep(&s);

  /*Price*/
  v_s2m(&MAffine3D,s.t,V_GetCmp(s.xc,offset),ptprice);

  return OK;
}
#endif //PremiaCurrentVersion

extern "C"{
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
     (2008+2) //The "#else" part of the code will be freely av
    ailable after the (year of creation of this file + 2)
static int CHK_OPT(  FD_NataliniBrianiAFFINE3D)(void *Opt,
    void *Mod)
{
  return NONACTIVE;
}
  int CALC(FD_NataliniBrianiAFFINE3D)(void *Opt, void *Mod,
     PricingMethod *Met)
{
return AVAILABLE_IN_FULL_PREMIA;
}
#else
  int CALC(FD_NataliniBrianiAFFINE3D)(void *Opt, void *Mod,
     PricingMethod *Met)
  {
    TYPEOPT* ptOpt=(TYPEOPT*)Opt;
    TYPEMOD* ptMod=(TYPEMOD*)Mod;


    return FD_AFFINE3D(ptOpt->BMaturity.Val.V_DATE,ptOpt->
    OMaturity.Val.V_DATE,
                       ptOpt->FixedRate.Val.V_PDOUBLE,pt
    Opt->ResetPeriod.Val.V_DATE,
                       ptMod->x01.Val.V_DOUBLE,
                       ptMod->x02.Val.V_DOUBLE,ptMod->x03.
```

```
    Val.V_DOUBLE,
                        ptMod->T.Val.V_DATE,
                        ptOpt->PayOff.Val.V_NUMFUNC_1,
                        ptMod->Sigma1.Val.V_PDOUBLE,
                        ptMod->Sigma2.Val.V_PDOUBLE,
                        ptMod->Sigma3.Val.V_PDOUBLE,
                        ptMod->shift.Val.V_PDOUBLE,
                        ptMod->k1.Val.V_PDOUBLE,
                        ptMod->k2.Val.V_PDOUBLE,
                        ptMod->k3.Val.V_PDOUBLE,
                        ptMod->Rho12.Val.V_DOUBLE,
                        ptMod->Rho13.Val.V_DOUBLE,
                        ptMod->Rho23.Val.V_DOUBLE,
                        Met->Par[0].Val.V_INT,Met->Par[1].
    Val.V_INT,
                        Met->Par[2].Val.V_INT,
                        &(Met->Res[0].Val.V_DOUBLE));
  }

static int CHK_OPT(FD_NataliniBrianiAFFINE3D)(void *Opt,
    void *Mod)
  {
    if ( (strcmp( ((Option*)Opt)->Name,"    CouponBearingCallEuro")==0) )
      return OK;

    return WRONG;
  }


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

        Met->Par[0].Val.V_INT2=21;
        Met->Par[1].Val.V_INT2=21;
        Met->Par[2].Val.V_INT2=21;
      }
```

```
    return OK;
  }

  PricingMethod MET(FD_NataliniBrianiAFFINE3D)=
  {
    "FD_NataliniBriani_Affine3D_Swaption",
    {{"SpaceStepNumber 1 ",INT2,{100},ALLOW},
     {"SpaceStepNumber 2",INT2,{100},ALLOW},
     {"SpaceStepNumber 3",INT2,{100},ALLOW}
     ,{" ",PREMIA_NULLTYPE,{0},FORBID}},
    CALC(FD_NataliniBrianiAFFINE3D),
    {{"Price",DOUBLE,{100},FORBID},
     {" ",PREMIA_NULLTYPE,{0},FORBID}},
    CHK_OPT(FD_NataliniBrianiAFFINE3D),
    CHK_ok,
    MET(Init)
  };
}
```

References