```
   Help
#include  "mrc30d_stdnd.h"
#include "enums.h"
#include "pnl/pnl_random.h"
#include "pnl/pnl_vector.h"
#include "pnl/pnl_matrix.h"
#include "pnl/pnl_finance.h"
#include <string.h>

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
     (2012+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(MC_WISHART)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(MC_BASKET30D)(void*Opt,void *Mod,PricingMethod *
    Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else

//------------------Random variable DATA
static PnlMat*  Starting;
static PnlVect*** GaussEuler;
static PnlVect*** GaussML1;
static PnlVect*** GaussML2;
static PnlMat***  CorreLL;
//------------------Initialization parameter
static PnlMat* CorrelationDax;
static PnlMat* lvol_value[30];
static PnlMat* lvol_der_t[30];
static PnlMat* lvol_der_s[30];
static PnlMat* lvol_der_c[30];

static PnlVect* S_value[30];
static PnlVect* I_compo;
static PnlVect* t_value;
static PnlVect* Initial_S;
static double Index_Value;
static PnlVect* Strikeindex;
static PnlVect* MatIndex;
//-------------------------------------------------------------
    ------------------
//------------------Function to get in memory all basket
    local volatilties
//-------------------------------------------------------------
    ------------------




//-------------------------------------------------------------
    ------------------
//------------------Locate a number from a table-----------
    -----------------
//-------------------------------------------------------------
    ------------------
static void locate_fast(PnlVect* x, int size, double y,int
    *rank)
{
  int jl,ju,jm;
  jl=0;
  ju=size-1;
  while(ju-jl >1)
  {
    jm = (int) (jl+ju)*0.5;
    if( y >= pnl_vect_get(x,jm))
    jl=jm;
    else
    ju=jm;
  }
  if(y == pnl_vect_get(x,size-1)) *rank=size-1;
  else if(y == pnl_vect_get(x,0)) *rank=0;
  else *rank= jl;

}



//-------------------------------------------------------------
    ------------------
```

```
//------------------Free all Memories----------------------
    ------------------
//---------------------------------------------------------
    ------------------
static void  Free_vol_local_par()
{

  int i=0;
   for(i=0;i<30;i++)
    {

      pnl_vect_free(&S_value[i]);
      pnl_mat_free(&lvol_value[i]);
      pnl_mat_free(&lvol_der_t[i]);
      pnl_mat_free(&lvol_der_s[i]);
      pnl_mat_free(&lvol_der_c[i]);
    }

   pnl_vect_free(&t_value);
   pnl_vect_free(&Initial_S);

   pnl_mat_free(&CorrelationDax);
   pnl_vect_free(&I_compo);

   pnl_vect_free(&MatIndex);
   pnl_vect_free(&Strikeindex);
 }



//---------------------------------------------------------
    ------------------
//-------------Compute weights and correlation from files--
    -----------------
//---------------------------------------------------------
    ----------------
static void Fill_repo_And_Composition(char *InitialStocksWe
    ights,char *BasketLocalVolatility,
              char *Basket_Correlation)
{
```

```
int nc;
int i,j;
//---------------initialzation of parameter
//--------------------Read files
char*  titreC;
char* titreI;
char* titre;
PnlMat* C;
PnlMat* CTT;
PnlMat* tmp;
nc    = 30;

titreC = InitialStocksWeights;
titreI = BasketLocalVolatility;
titre = Basket_Correlation;

tmp =  pnl_mat_create_from_file(titre);
CorrelationDax= pnl_mat_create_from_double(30,30,0.);


for(i=0;i<30;i++)
  for(j=0;j<30;j++)
    pnl_mat_set(CorrelationDax,i,j,pnl_mat_get(tmp,i,j));

pnl_mat_mult_double(CorrelationDax,0.01);


C= pnl_mat_create_from_file(titreC);

//------------------Store weight Composition
I_compo = pnl_vect_create_from_double(nc,0.);

//pnl_mat_print(C);
for(i=0;i<nc;i++)
  {
    pnl_vect_set(I_compo,i,pnl_mat_get(C,i,0));
  }
Index_Value = pnl_mat_get(C,nc,0);
```

```
  CTT = pnl_mat_create_from_file(titreI);
  Strikeindex = pnl_vect_create_from_double(CTT->n-1,0.);
  MatIndex    = pnl_vect_create_from_double(CTT->m-1,0.);

  for(i=1;i<CTT->n;i++)
    pnl_vect_set(Strikeindex,i-1,pnl_mat_get(CTT,0,i));

  for(i=1;i<CTT->m;i++)
    pnl_vect_set(MatIndex,i-1,pnl_mat_get(CTT,i,0));
  //-------------------Free memory
  pnl_mat_free(&C);
  pnl_mat_free(&CTT);
  pnl_mat_free(&tmp);
}



//---------------------------------------------------------
    ----------------
//--Compute weights,correlation and basket local volatilie
    s from files-----
//---------------------------------------------------------
    ----------------
static void  Fill_vol_local_par(char *InitialStocksWeights,
    char *LocalVolatilities,
       char *Basket_Correlation, char *BasketLocalVolati
    lity)
{
  int ns,nt,nbrs;
  int t,s,k;
  int h;
  double tmp1;
  double dt;
  double ds;

  //---------------initialzation of parameter
  //--------------------Read files
  char*  titreC;
  PnlMat* C;

  //---------------------------------
```

```
Fill_repo_And_Composition(InitialStocksWeights,Basket
  LocalVolatility,Basket_Correlation );
//-----------------------------------


titreC = LocalVolatilities;
C=pnl_mat_create_from_file(titreC);
nt = 14;
ns = 11;
nbrs=30;

h=0;



Initial_S = pnl_vect_create_from_double(nbrs,0.);
t_value = pnl_vect_create_from_double(nt,0.);
k=0;
for(k=0;k<nbrs;k++)
  {

    //printf("Starting with stock number %d{n",k);
    S_value[k]    = pnl_vect_create_from_double(ns,0.);

    lvol_der_t[k] = pnl_mat_create_from_double(nt,ns,0.);
    lvol_der_c[k] = pnl_mat_create_from_double(nt,ns,0.);
    lvol_der_s[k] = pnl_mat_create_from_double(nt,ns,0.);

    lvol_value[k] = pnl_mat_create_from_double(nt,ns,0.);
    pnl_vect_set(Initial_S,k,pnl_mat_get(C,h,0));



    for(t=0;t<nt;t++ )
{

  pnl_vect_set(t_value,t,pnl_mat_get(C,h+t+1,0));
  for(s=0;s<ns;s++)
    {

      pnl_vect_set(S_value[k],s,pnl_mat_get(C,h,1+s));
      pnl_mat_set(lvol_value[k],t,s,pnl_mat_get(C,h+1+t,
  1+s));
    }
}

    h= h + nt +2;
   }

for(k=0;k<nbrs;k++)
  {
     for(t=0;t<nt;t++ )
  {
    for(s=0;s<ns;s++)
    {
      if(0<t&&t<nt-1&&s>0&&s<ns-1)
      {
        //------------The calculus of the derivative
  with respect to the time
        dt = pnl_vect_get(t_value,t+1)-pnl_vect_get(
  t_value,t);
        dt = dt/255.;
        tmp1 =  pnl_mat_get(lvol_value[k],t+1,s)- pn
  l_mat_get(lvol_value[k],t,s);
        tmp1 = tmp1/dt;
        dt =  pnl_vect_get(t_value,t)-pnl_vect_get(t_
  value,t-1);
        dt = dt/255.;
        tmp1 = 0.5*tmp1+ 0.5*(pnl_mat_get(lvol_value[
  k],t,s)- pnl_mat_get(lvol_value[k],t-1,s))/dt;
        pnl_mat_set(lvol_der_t[k],t,s,tmp1);

        //------------The calculus of the derivative
  with respect to the space
        ds = pnl_vect_get(S_value[k],s+1)- pnl_vect_
  get(S_value[k],s);
        tmp1 =  pnl_mat_get(lvol_value[k],t,s+1)- pn
  l_mat_get(lvol_value[k],t,s);
        tmp1 = tmp1/ds;
        ds =  pnl_vect_get(S_value[k],s)-pnl_vect_get
  (S_value[k],s-1);
        tmp1 = 0.5*tmp1+ 0.5*(pnl_mat_get(lvol_value[
  k],t,s)- pnl_mat_get(lvol_value[k],t,s-1))/ds;
        pnl_mat_set(lvol_der_s[k],t,s,tmp1);

        //-------------The calculus of the cross de
  rivative

        ds =  pnl_vect_get(S_value[k],s+1)- pnl_vect_
  get(S_value[k],s);
        dt =  pnl_vect_get(t_value,t+1)-pnl_vect_get(
  t_value,t);
        dt = dt/255.;
        tmp1 = (pnl_mat_get(lvol_value[k],t+1,s+1)+pn
  l_mat_get(lvol_value[k],t,s)-pnl_mat_get(lvol_value[k],t+1,
  s)-pnl_mat_get(lvol_value[k],t,s+1))/(ds*dt);
        pnl_mat_set(lvol_der_c[k],t,s,tmp1);
      }
      if(t== 0&&s>0&&s<ns-1 )
      {
        //------------The calculus of the derivative
  with respect to the time
        dt = pnl_vect_get(t_value,t+1)-pnl_vect_get(
  t_value,t);
        dt = dt/255.;
        tmp1 =  pnl_mat_get(lvol_value[k],t+1,s)- pn
  l_mat_get(lvol_value[k],t,s);
        tmp1 = tmp1/dt;
        pnl_mat_set(lvol_der_t[k],t,s,tmp1);

        //------------The calculus of the derivative
  with respect to the space
        ds = pnl_vect_get(S_value[k],s+1)- pnl_vect_
  get(S_value[k],s);
        tmp1 =  pnl_mat_get(lvol_value[k],t,s+1)- pn
  l_mat_get(lvol_value[k],t,s);
        tmp1 = tmp1/ds;
        ds =  pnl_vect_get(S_value[k],s)-pnl_vect_get
  (S_value[k],s-1);
        tmp1 = 0.5*tmp1+ 0.5*(pnl_mat_get(lvol_value[
  k],t,s)- pnl_mat_get(lvol_value[k],t,s-1))/ds;
        pnl_mat_set(lvol_der_s[k],t,s,tmp1);
        //-------------The calculus of the cross de
  rivative
        ds =  pnl_vect_get(S_value[k],s+1)- pnl_vect_
  get(S_value[k],s);
        dt =  pnl_vect_get(t_value,t+1)-pnl_vect_get(
  t_value,t);
        dt = dt/255.;
        tmp1 = (pnl_mat_get(lvol_value[k],t+1,s+1)+pn
  l_mat_get(lvol_value[k],t,s)-pnl_mat_get(lvol_value[k],t+1,
  s)-pnl_mat_get(lvol_value[k],t,s+1))/(ds*dt);
        pnl_mat_set(lvol_der_c[k],t,s,tmp1);
      }

      if(t== nt-1&&s>0&&s<ns-1 )
      {
        //------------The calculus of the derivative
  with respect to the time
        dt = pnl_vect_get(t_value,t)-pnl_vect_get(t_
  value,t-1);
        dt = dt/255.;
        tmp1 =  pnl_mat_get(lvol_value[k],t,s)- pnl_
  mat_get(lvol_value[k],t-1,s);
        tmp1 = tmp1/dt;
        pnl_mat_set(lvol_der_t[k],t,s,tmp1);

        //------------The calculus of the derivative
  with respect to the space
        ds = pnl_vect_get(S_value[k],s+1)- pnl_vect_
  get(S_value[k],s);
        tmp1 =  pnl_mat_get(lvol_value[k],t,s+1)- pn
  l_mat_get(lvol_value[k],t,s);
        tmp1 = tmp1/ds;
        ds =  pnl_vect_get(S_value[k],s)-pnl_vect_get
  (S_value[k],s-1);
        tmp1 = 0.5*tmp1+ 0.5*(pnl_mat_get(lvol_value[
  k],t,s)- pnl_mat_get(lvol_value[k],t,s-1))/ds;
        pnl_mat_set(lvol_der_s[k],t,s,tmp1);

        //-------------The calculus of the cross de
  rivative
        ds =  pnl_vect_get(S_value[k],s)- pnl_vect_
```

```
get(S_value[k],s-1);
      dt =  pnl_vect_get(t_value,t)-pnl_vect_get(t_
value,t-1);
      dt = dt/255.;
      tmp1 = (pnl_mat_get(lvol_value[k],t-1,s-1)+pn
l_mat_get(lvol_value[k],t,s)-pnl_mat_get(lvol_value[k],t-1,
s)-pnl_mat_get(lvol_value[k],t,s-1))/(ds*dt);
      pnl_mat_set(lvol_der_c[k],t,s,tmp1);
    }
    if(0<t&&t<nt-1&&s==0)
    {
      //------------The calculus of the derivative
with respect to the time
      dt = pnl_vect_get(t_value,t+1)-pnl_vect_get(
t_value,t);
      dt = dt/255.;
      tmp1 =  pnl_mat_get(lvol_value[k],t+1,s)- pn
l_mat_get(lvol_value[k],t,s);
      tmp1 = tmp1/dt;
      dt =  pnl_vect_get(t_value,t)-pnl_vect_get(t_
value,t-1);
      tmp1 = 0.5*tmp1+ 0.5*(pnl_mat_get(lvol_value[
k],t,s)- pnl_mat_get(lvol_value[k],t-1,s))/dt;
      pnl_mat_set(lvol_der_t[k],t,s,tmp1);

      //------------The calculus of the derivative
with respect to the space
      ds = pnl_vect_get(S_value[k],s+1)- pnl_vect_
get(S_value[k],s);
      tmp1 =  pnl_mat_get(lvol_value[k],t,s+1)- pn
l_mat_get(lvol_value[k],t,s);
      tmp1 = tmp1/ds;
      pnl_mat_set(lvol_der_s[k],t,s,tmp1);

      //-------------The calculus of the cross de
rivative

      ds =  pnl_vect_get(S_value[k],s+1)- pnl_vect_
get(S_value[k],s);
      dt =  pnl_vect_get(t_value,t+1)-pnl_vect_get(
t_value,t);
```

```
      dt = dt/255.;
      tmp1 = (pnl_mat_get(lvol_value[k],t+1,s+1)+pn
l_mat_get(lvol_value[k],t,s)-pnl_mat_get(lvol_value[k],t+1,
s)-pnl_mat_get(lvol_value[k],t,s+1))/(ds*dt);
      pnl_mat_set(lvol_der_c[k],t,s,tmp1);
    }

    if(0<t&&t<nt-1&&s==ns-1)
    {
      //------------The calculus of the derivative
with respect to the time
      dt = pnl_vect_get(t_value,t+1)-pnl_vect_get(
t_value,t);
      dt = dt/255.;
      tmp1 =  pnl_mat_get(lvol_value[k],t+1,s)- pn
l_mat_get(lvol_value[k],t,s);
      tmp1 = tmp1/dt;
      dt =  pnl_vect_get(t_value,t)-pnl_vect_get(t_
value,t-1);
      dt = dt/255.;
      tmp1 = 0.5*tmp1+ 0.5*(pnl_mat_get(lvol_value[
k],t,s)- pnl_mat_get(lvol_value[k],t-1,s))/dt;
      pnl_mat_set(lvol_der_t[k],t,s,tmp1);

      //------------The calculus of the derivative
with respect to the space
      ds = pnl_vect_get(S_value[k],s)- pnl_vect_get
(S_value[k],s-1);
      tmp1 =  pnl_mat_get(lvol_value[k],t,s)- pnl_
mat_get(lvol_value[k],t,s-1);
      tmp1 = tmp1/ds;
      pnl_mat_set(lvol_der_s[k],t,s,tmp1);

      //-------------The calculus of the cross de
rivative

      ds =  pnl_vect_get(S_value[k],s)- pnl_vect_
get(S_value[k],s-1);
      dt =  pnl_vect_get(t_value,t)-pnl_vect_get(t_
value,t-1);
      dt = dt/255.;
      tmp1 = (pnl_mat_get(lvol_value[k],t-1,s-1)+pn
l_mat_get(lvol_value[k],t,s)-pnl_mat_get(lvol_value[k],t-1,
s)-pnl_mat_get(lvol_value[k],t,s-1))/(ds*dt);
      pnl_mat_set(lvol_der_c[k],t,s,tmp1);
    }

    if(0==t&&s==ns-1)
    {
      //------------The calculus of the derivative
with respect to the time
      dt = pnl_vect_get(t_value,t+1)-pnl_vect_get(
t_value,t);
      dt = dt/255.;
      tmp1 =  pnl_mat_get(lvol_value[k],t+1,s)- pn
l_mat_get(lvol_value[k],t,s);
      tmp1 = tmp1/dt;
      pnl_mat_set(lvol_der_t[k],t,s,tmp1);

      //------------The calculus of the derivative
with respect to the space
      ds = pnl_vect_get(S_value[k],s)- pnl_vect_get
(S_value[k],s-1);
      tmp1 =  pnl_mat_get(lvol_value[k],t,s)- pnl_
mat_get(lvol_value[k],t,s-1);
      tmp1 = tmp1/ds;
      pnl_mat_set(lvol_der_s[k],t,s,tmp1);

      //-------------The calculus of the cross de
rivative

      ds =  pnl_vect_get(S_value[k],s)- pnl_vect_
get(S_value[k],s-1);
      dt =  pnl_vect_get(t_value,t+1)-pnl_vect_get(
t_value,t);
      dt = dt/255.;
      tmp1 = -(pnl_mat_get(lvol_value[k],t+1,s-1)+
pnl_mat_get(lvol_value[k],t,s)-pnl_mat_get(lvol_value[k],t+
1,s)-pnl_mat_get(lvol_value[k],t,s-1))/(ds*dt);
      pnl_mat_set(lvol_der_c[k],t,s,tmp1);
    }
    if(0==t&&s==0)
    {
      //------------The calculus of the derivative
with respect to the time
      dt = pnl_vect_get(t_value,t+1)-pnl_vect_get(
t_value,t);
      dt = dt/255.;
      tmp1 =  pnl_mat_get(lvol_value[k],t+1,s)- pn
l_mat_get(lvol_value[k],t,s);
      tmp1 = tmp1/dt;
      pnl_mat_set(lvol_der_t[k],t,s,tmp1);

      //------------The calculus of the derivative
with respect to the space
      ds = pnl_vect_get(S_value[k],s+1)- pnl_vect_
get(S_value[k],s);
      tmp1 =  pnl_mat_get(lvol_value[k],t,s+1)- pn
l_mat_get(lvol_value[k],t,s);
      tmp1 = tmp1/ds;
      pnl_mat_set(lvol_der_s[k],t,s,tmp1);

      //-------------The calculus of the cross de
rivative

      ds =  pnl_vect_get(S_value[k],s+1)- pnl_vect_
get(S_value[k],s);
      dt =  pnl_vect_get(t_value,t+1)-pnl_vect_get(
t_value,t);
      dt = dt/255.;
      tmp1 = (pnl_mat_get(lvol_value[k],t+1,s+1)+pn
l_mat_get(lvol_value[k],t,s)-pnl_mat_get(lvol_value[k],t+1,
s)-pnl_mat_get(lvol_value[k],t,s+1))/(ds*dt);
      pnl_mat_set(lvol_der_c[k],t,s,tmp1);
    }
    if(nt-1==t&&s==0)
    {
      //------------The calculus of the derivative
with respect to the time
      dt = pnl_vect_get(t_value,t)-pnl_vect_get(t_
value,t-1);
      dt = dt/255.;
      tmp1 =  pnl_mat_get(lvol_value[k],t,s)- pnl_
mat_get(lvol_value[k],t-1,s);
      tmp1 = tmp1/dt;
      pnl_mat_set(lvol_der_t[k],t,s,tmp1);

      //------------The calculus of the derivative
with respect to the space
      ds = pnl_vect_get(S_value[k],s+1)- pnl_vect_
get(S_value[k],s);
      tmp1 =  pnl_mat_get(lvol_value[k],t,s+1)- pn
l_mat_get(lvol_value[k],t,s);
      tmp1 = tmp1/ds;
      pnl_mat_set(lvol_der_s[k],t,s,tmp1);

      //-------------The calculus of the cross de
rivative

      ds =  pnl_vect_get(S_value[k],s+1)- pnl_vect_
get(S_value[k],s);
      dt =  pnl_vect_get(t_value,t)-pnl_vect_get(t_
value,t-1);
      dt = dt/255.;
      tmp1 = -(pnl_mat_get(lvol_value[k],t-1,s+1)+
pnl_mat_get(lvol_value[k],t,s)-pnl_mat_get(lvol_value[k],t-
1,s)-pnl_mat_get(lvol_value[k],t,s+1))/(ds*dt);
      pnl_mat_set(lvol_der_c[k],t,s,tmp1);
    }

    if(nt-1==t&&s==ns-1)
    {
      //------------The calculus of the derivative
with respect to the time


      dt = pnl_vect_get(t_value,t)-pnl_vect_get(t_
value,t-1);
      dt = dt/255.;
      tmp1 =  pnl_mat_get(lvol_value[k],t,s)- pnl_
mat_get(lvol_value[k],t-1,s);
      tmp1 = tmp1/dt;
      pnl_mat_set(lvol_der_t[k],t,s,tmp1);
```

```
          //------------The calculus of the derivative
    with respect to the space
          ds = pnl_vect_get(S_value[k],s)- pnl_vect_get
    (S_value[k],s-1);
          tmp1 =  pnl_mat_get(lvol_value[k],t,s)- pnl_
    mat_get(lvol_value[k],t,s-1);
          tmp1 = tmp1/ds;
          pnl_mat_set(lvol_der_s[k],t,s,tmp1);

          //-------------The calculus of the cross de
    rivative

          ds =  pnl_vect_get(S_value[k],s)- pnl_vect_
    get(S_value[k],s-1);
          dt =  pnl_vect_get(t_value,t)-pnl_vect_get(t_
    value,t-1);

          dt = dt/255.;
          tmp1 = (pnl_mat_get(lvol_value[k],t-1,s-1)+pn
    l_mat_get(lvol_value[k],t,s)-pnl_mat_get(lvol_value[k],t-1,
    s)-pnl_mat_get(lvol_value[k],t,s-1))/(ds*dt);

          pnl_mat_set(lvol_der_c[k],t,s,tmp1);

        }

      }
    }
      h=h+nt+2;
    }

  pnl_mat_free(&C);

}

//-----------------------------------------------------------
    ----------------
//-------------Get the local volatily for a given time ----
    ----------------
//----------------------and stock by linear interpolation--
```

```
    ----------------
//----------------------------------------------------------
    ----------------

static void  get_local_vol(int i_c,double t, double s, PnlV
    ect* v/* it returns the value, the time derivative and the
    space one*/)
{

  int j_s,j_t;
  double a,b,c;
  double tk,tk1;
  double y[5];
  double sk,sk1;

  tk=0.;
  tk1=0.;
  a=0.;
  if( t*255. <= pnl_vect_get(t_value,0) )
    {
      if(s <= pnl_vect_get(S_value[i_c],0))
  {

    sk =(pnl_mat_get(lvol_value[i_c],0,1)-pnl_mat_get(lvol
    _value[i_c],0,0))/( pnl_vect_get(S_value[i_c],1)-pnl_vect_
    get(S_value[i_c],0));
    sk1 = pnl_mat_get(lvol_value[i_c],0,0)-sk*pnl_vect_get
    (S_value[i_c],0);
    pnl_vect_set(v,0,s*sk+sk1);

    pnl_vect_set(v,1,0.);
    pnl_vect_set(v,2,0.);

    return;
  }
      if(s >= pnl_vect_get(S_value[i_c],S_value[i_c]->size-
    1))
  {
    sk =(pnl_mat_get(lvol_value[i_c],0,S_value[i_c]->size-
    1)-pnl_mat_get(lvol_value[i_c],0,S_value[i_c]->size-2))/(
    pnl_vect_get(S_value[i_c],S_value[i_c]->size-1)-pnl_vect_
    get(S_value[i_c],S_value[i_c]->size-2));
    sk1 = pnl_mat_get(lvol_value[i_c],0,S_value[i_c]->size
    -1)-sk*pnl_vect_get(S_value[i_c],S_value[i_c]->size-1);
    pnl_vect_set(v,0,s*sk+sk1);

    pnl_vect_set(v,1,0.);
    pnl_vect_set(v,2,0.);
    return;
  }

      locate_fast(S_value[i_c], S_value[i_c]->size,s,&j_s);
      sk =(pnl_mat_get(lvol_value[i_c],0,j_s+1)-pnl_mat_get
    (lvol_value[i_c],0,j_s))/( pnl_vect_get(S_value[i_c],j_s+1
    )-pnl_vect_get(S_value[i_c],j_s));
      sk1 = pnl_mat_get(lvol_value[i_c],0,j_s)-sk*pnl_vect_
    get(S_value[i_c],j_s);

      pnl_vect_set(v,0,s*sk+sk1);
      pnl_vect_set(v,1,0.);
      pnl_vect_set(v,2,0.);
      return;

    }

  if( t*255. >= pnl_vect_get(t_value,t_value->size-1) )
    {


      if(s <= pnl_vect_get(S_value[i_c],0))
  {
    pnl_vect_set(v,0,pnl_mat_get(lvol_value[i_c],t_value->
    size-1,0));
    pnl_vect_set(v,1,0.);
    pnl_vect_set(v,2,0.);
    return;
  }
      if(s >= pnl_vect_get(S_value[i_c],S_value[i_c]->size-
    1))
  {
    pnl_vect_set(v,0,pnl_mat_get(lvol_value[i_c],t_value->
    size-1,S_value[i_c]->size-1));
```

```
    pnl_vect_set(v,1,0.);
    pnl_vect_set(v,2,0.);
    return;
  }


      locate_fast(S_value[i_c], S_value[i_c]->size, s,&j_s)
    ;

      sk =(pnl_mat_get(lvol_value[i_c],t_value->size-1,j_s+
    1)-pnl_mat_get(lvol_value[i_c],t_value->size-1,j_s))/( pn
    l_vect_get(S_value[i_c],j_s+1)-pnl_vect_get(S_value[i_c],j_
    s));
      sk1 = pnl_mat_get(lvol_value[i_c],t_value->size-1,j_
    s)-sk*pnl_vect_get(S_value[i_c],j_s);

      pnl_vect_set(v,0,s*sk+sk1);
      pnl_vect_set(v,1,0.);
      pnl_vect_set(v,2,0.);
      return;


    }

  if(  s<=pnl_vect_get(S_value[i_c],0) )
    {

      locate_fast(t_value, t_value->size, t*255.,&j_t);

      tk  = pnl_vect_get(t_value,j_t)/255.;
      tk1 = pnl_vect_get(t_value,j_t+1)/255.;


      b = (pnl_mat_get(lvol_value[i_c],j_t+1,0)- pnl_mat_
    get(lvol_value[i_c],j_t,0))/(tk1-tk);
      a =  pnl_mat_get(lvol_value[i_c],j_t,0)-tk*b;

      pnl_vect_set(v,0,a+t*b);
      pnl_vect_set(v,1, b );
      pnl_vect_set(v,2,0.);
      return;
    }

if( s>=pnl_vect_get(S_value[i_c],S_value[i_c]->size-1) )
    {
      locate_fast(t_value, t_value->size, t*255., &j_t);
      tk  = pnl_vect_get(t_value,j_t)/255.;
      tk1 = pnl_vect_get(t_value,j_t+1)/255.;


      b = (pnl_mat_get(lvol_value[i_c],j_t+1,S_value[i_c]->
    size-1)- pnl_mat_get(lvol_value[i_c],j_t,S_value[i_c]->size
    -1))/(tk1-tk);
      a =  pnl_mat_get(lvol_value[i_c],j_t,S_value[i_c]->si
    ze-1)-tk*b;

      pnl_vect_set(v,0,a+t*b);
      pnl_vect_set(v,1, b);
      pnl_vect_set(v,2,0.);
      return;
    }



 locate_fast(S_value[i_c], S_value[i_c]->size, s,&j_s);
 locate_fast(t_value, t_value->size, t*255.,&j_t);


 y[0]=0.;
 y[1]= pnl_mat_get(lvol_value[i_c],j_t,j_s);
 y[2]= pnl_mat_get(lvol_value[i_c],j_t+1,j_s);
 y[3]= pnl_mat_get(lvol_value[i_c],j_t+1,j_s+1);
 y[4]= pnl_mat_get(lvol_value[i_c],j_t,j_s+1);

 a= (t- pnl_vect_get(t_value,j_t)/255.)/(pnl_vect_get(t_val
    ue,j_t+1)/255.-pnl_vect_get(t_value,j_t)/255.);
 b= (s-pnl_vect_get(S_value[i_c],j_s))/( pnl_vect_get(S_val
    ue[i_c],j_s+1)- pnl_vect_get(S_value[i_c],j_s));


 c = (1.-a) *(1-b)*y[1]+a*(1.-b)*y[2]+a*b*y[3]+(1.-a)*b*y[4
```

```
    ];
 pnl_vect_set(v,0,c);

 tk1 = pnl_vect_get(S_value[i_c],j_s)- pnl_vect_get(S_value
    [i_c],j_s+1);
 tk  = pnl_vect_get(t_value,j_t)/255.-pnl_vect_get(t_value,
    j_t+1)/255.;

 c =  tk1 * ((y[2]-y[1])*(1.-s)+s*(y[3]-y[4]));
 pnl_vect_set(v,1,c);
 c =  tk*( (1.-t)*(y[4]-y[1])+t*(y[3]-y[2]));
 pnl_vect_set(v,2,c);



}


//------------------------------------------------------------
    -----------------
//---------------------------Weak approximation of the
    Gauss----------------
//------------------------------------------------------------
    ----------------


static double DiscLawMatch7(int generator)
{
  double u=2.*pnl_rand_uni(generator)-1.;
  double res=sqrt(6);
  if (fabs(u)<((res-2)/(2*res))) res=sqrt(3+res);
  else res=sqrt(3-res);
  if (u<0) return -res;
  return res;
}

//------------------------------------------------------------
    -----------------
//---------------------------Weak approximation For the dr
    ift part----------
//------------------------------------------------------------
```

```
    ----------------

static void ODE_Compute(double t, double kappa, double a,
    int dim, double eta, double gamma, double I_t,PnlMat* xt)
{

  //-----Declaration variable
  int i,j;

  if(kappa != 0)
    {

      for(i=0;i<dim;i++)
  {
    for(j=0;j<dim;j++)
      {
        if(i!=j)
    pnl_mat_set(xt,i,j, pnl_mat_get(xt,i,j)*exp(-2.*kapp
    a*t)+((1.-exp(-2.*kappa*t))/(1+eta*exp(gamma*log(I_t)))));
      }
  }
    }
}


//-------------------------------------------------------
    ------------------
//------Generate all variable with repect to Number MC and
    time Disc---------
//-------------------------------------------------------
    ------------------
static void Generate_Random_And_Time(int IS_Euler, int NbrM
    c,  int dim, int NbrT, int generator )
{

  //int NbrT;
  int i,j;
  int m,n;


  pnl_rand_init(generator,1,(long) NbrT*NbrMc*dim*dim);
```

```
//NbrT = TimeDisc->size -1 ;
if (IS_Euler ==1)
 {
  GaussEuler  =  malloc(NbrMc*sizeof(PnlVect*) );

  if (GaussEuler==NULL)
{
 printf ("Error allocating requested memory GaussEuler"
 );
 exit (1);
}
  for(i = 0; i < NbrMc; i++)
{
 GaussEuler[i] = malloc((NbrT)*sizeof(PnlVect));
 if(GaussEuler[i] == NULL)
  {
   fprintf(stderr, "out of memory GaussEuler{n");
   //exit or return;
  }
}


  for(i=0;i<NbrMc;i++)
{
 for(j=0;j<NbrT;j++)
  {

   GaussEuler[i][j] = pnl_vect_create_from_double(dim
 ,0.);
   pnl_vect_rand_normal( GaussEuler[i][j],dim,   generator);
  }
}
 }
else
 {

  GaussML1  =  malloc(NbrMc*sizeof(PnlVect*) );
  GaussML2  =  malloc(NbrMc*sizeof(PnlVect*) );
  if (GaussML1==NULL||GaussML2==NULL )
{
 printf ("Error allocating requested memory GaussML" );
```

```
  exit (1);
}
    for(i = 0; i < NbrMc; i++)
{
  GaussML1[i] = malloc((NbrT)*sizeof(PnlVect));
  GaussML2[i] = malloc((NbrT)*sizeof(PnlVect));

  if(GaussML1[i] == NULL || GaussML2[i] == NULL)
    {
      fprintf(stderr, "out of memory GaussML  {n");
      //exit or return;
    }
}


    for(i=0;i<NbrMc;i++)
{
  for(j=0;j<NbrT;j++)
    {

      GaussML1[i][j] = pnl_vect_create_from_double(dim,0
  .);
      GaussML2[i][j] = pnl_vect_create_from_double(dim,0
  .);
      pnl_vect_rand_normal( GaussML1[i][j],dim,    generator);
      pnl_vect_rand_normal( GaussML2[i][j],dim,    generator);
    }
}
  }

CorreLL  =  malloc(NbrMc*sizeof(PnlMat*) );

    if (CorreLL==NULL)
{
  printf ("Error allocating requested memory
  Correlation");
  exit (1);
}
    for(i = 0; i < NbrMc; i++)
{
  CorreLL[i] = malloc((NbrT)*sizeof(PnlMat));
```

```
    if(CorreLL[i] == NULL)
      {
        fprintf(stderr, "out of memory Correlation {n");
        //exit or return;
      }
  }


      for(i=0;i<NbrMc;i++)
  {
    for(j=0;j<NbrT;j++)
      {

        CorreLL[i][j] = pnl_mat_create_from_double(dim,dim
    ,0.);

        for(m=0;m<dim;m++)
    for(n=0;n<dim;n++)
      pnl_mat_set(CorreLL[i][j],m,n,DiscLawMatch7(     generator));

      }
  }
      Starting= pnl_mat_create_from_double(NbrMc,NbrT,0.);
      pnl_mat_rand_uni2(Starting ,NbrMc,NbrT,0.,1.,    generator);

  return;

}

//-----------------------------------------------------------
    ------------------
//------Free all variable with repect to Number MC and
    time Disc------------
//-----------------------------------------------------------
    ------------------
static void Free_random_And_Time(int IS_EULER, int NbrMc,
    int NbrT)
{

  int i;
  int j;
```

```
//int NbrT;
//  NbrT=TimeDisc->size-1;
if(IS_EULER==1)
  {

    for(i=0;i<NbrMc;i++)
for(j=0;j<NbrT;j++)
  pnl_vect_free(&GaussEuler[i][j]);
    for(i = 0; i < NbrMc; i++)
free(GaussEuler[i]);
    free(GaussEuler);
  }
else
  {
    for(i=0;i<NbrMc;i++)
{
for(j=0;j<NbrT;j++)
  {
    pnl_vect_free(&GaussML1[i][j]);
    pnl_vect_free(&GaussML2[i][j]);
  }
}
    for(i = 0; i < NbrMc; i++)
{
  free(GaussML1[i]);
  free(GaussML2[i]);
}
    free(GaussML1);
    free(GaussML2);
  }


for(i=0;i<NbrMc;i++)
  for(j=0;j<NbrT;j++)
    pnl_mat_free(&CorreLL[i][j]);

for(i = 0; i < NbrMc; i++)
  free(CorreLL[i]);
free(CorreLL);

pnl_mat_free(&Starting);
```

```
  return;

}

//---------------------------------------------------------
    ------------------
//-------------------Discretization of the stock part with
    Euler scheme------
//---------------------------------------------------------
    ------------------


static void Scheme_Basic_Stock_EL_Fast(PnlVect* St, const
    PnlMat* xt,  int dim, double t, double DT, PnlVect* RandV,
    double r)
 {
   //-------------Declaration of variable
   int i;
   PnlMat* sqr;
   PnlMat* permute;
   PnlVect* tmp;
   PnlVect* tmp1;
   double S1;


   sqr  = pnl_mat_copy(xt);
   permute= pnl_mat_create_from_double(dim,dim,0.);
   pnl_mat_set_id(permute);

   pnl_mat_chol(sqr);
   tmp = pnl_vect_create_from_double(dim,0.);
   tmp1 = pnl_vect_create_from_double(dim,0.);

   pnl_vect_clone(tmp,RandV);

   pnl_vect_mult_double(tmp,sqrt(DT));

   pnl_mat_mult_vect_inplace(tmp1,sqr,tmp);
   pnl_vect_resize(tmp,3);

   //------------Begin operation
   for(i=0;i<dim;i++)
     {

       S1 = pnl_vect_get(St,i);
       get_local_vol(i, t,  S1, tmp);
       S1 = log(S1)+ DT* ( r-0.5* pnl_vect_get(tmp,0)*pnl_
    vect_get(tmp,0)  ) + pnl_vect_get(tmp,0)*pnl_vect_get(tmp1,
    i);
       pnl_vect_set(St,i, exp(S1));

     }
   //-------------Free Memory
   pnl_vect_free(&tmp1);
   pnl_vect_free(&tmp);
   pnl_mat_free(&sqr);
   pnl_mat_free(&permute);

 }




//---------------------------------------------------------
    ------------------
//-------------------Discretization of the Wishart process
    ------------------
//---------------------------------------------------------
    ------------------


void Wishart_Disc_high_speed_dim_d_weak(PnlMat *F,int dim,
    double t, PnlMat* RdM)
{

  PnlMat* x;
  PnlMat* g;
  int i,j;

  x = pnl_mat_copy(F);
  g = pnl_mat_create_from_double(dim,dim,0.);


  pnl_mat_chol(x);
  for(i=0;i<dim;i++)
  {
    for(j=0;j<dim;j++)
      pnl_mat_set(g,i,j,sqrt(t)*pnl_mat_get(RdM,i,j));//
    DiscLawMatch7( generator));//pnl_rand_normal(generator));
  }
  pnl_mat_plus_mat(x,g);
  pnl_mat_clone(g,x);
  pnl_mat_sq_transpose(x);
  pnl_mat_mult_mat_inplace(F,g,x);



  pnl_mat_free(&x);
  pnl_mat_free(&g);
}



static int mc_basket30d(NumFunc_nd *p,double maturity,
    double r,double kappa0,double eta0,double gamma0,double a0,char
    *InitialStocksWeights,char *LocalVolatilities,char *Basket
    _Correlation,char *BasketLocalVolatility, long NbrMC,int
    NbrT,int generator,double *ptprice,double *pterror_price)
{
  double strike;


  //------------------------------Declaration of variable
  //------------Stock parmater
  int dim;

  //------------Stock Memory
  double Ess;
  double Varr;
  //------------Basket Parameter
  PnlVect* St;
  double It,I0;
  //-------------------------temporary variable
  int i,j,l,m;
  double tmmp=0.;
  PnlMat* xt;
  //------------Monte Carlo Parameter
  double DT;
  double DTe;

  //Dimnesion of the problem
  dim =30;

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

  Fill_vol_local_par(InitialStocksWeights,LocalVolatilitie
    s, Basket_Correlation,BasketLocalVolatility);

  Generate_Random_And_Time(1, NbrMC,  dim, NbrT,generator);

  I0= pnl_vect_scalar_prod(Initial_S,I_compo);
  St = pnl_vect_create_from_double(dim,0.);
  xt = pnl_mat_copy(CorrelationDax);

  Ess = 0.;
  Varr = 0.;
  tmmp=0.;
  DTe=0.;
  i=0;
  j=0;

  DT =(double)( maturity/((double)NbrT));

  for(i=1;i<=NbrMC;i++)
    {

      pnl_mat_clone(xt,CorrelationDax);
      pnl_vect_clone(St,Initial_S);
      It =I0;
      for(j=1;j<= NbrT;j++)
  {
    if(dim ==4)
      DTe= a0*a0*DT;
    else
      {
        if(dim >4)
    {
      DTe = DT*a0*a0;
      DTe = DTe + ((double) dim - 4)*0.5*DTe*DTe;

    }
        else
    {
      DTe = sqrt(1.-2.*( ((double) dim) - 4.)*DT*a0*a0);
      DTe = (-1.+DTe)/(4.-((double) dim));

    }
      }

    if(pnl_mat_get(Starting,i-1,j-1)<=0.5)
      {

        Scheme_Basic_Stock_EL_Fast( St, xt,dim,(double) (
    j-1)*DT,DT,GaussEuler[i-1][j-1],r);
        It = pnl_vect_scalar_prod(St,I_compo);


        ODE_Compute(DT,kappa0, a0, dim,  eta0,  gamma0,
    It/I0, xt);

        Wishart_Disc_high_speed_dim_d_weak(xt,dim,DTe,
    CorreLL[i-1][j-1]);
        for(l=0;l<dim;l++)
    for(m=0;m<dim;m++)
      {
        if(l!=m)
          pnl_mat_set(xt,l,m,pnl_mat_get(xt,l,m)/sqrt(pn
    l_mat_get(xt,l,l)*pnl_mat_get(xt,m,m)));
      }
```

```
        for(l=0;l<dim;l++)
    pnl_mat_set(xt,l,l, 1.);


      }
    else
      {
        It = pnl_vect_scalar_prod(St,I_compo);

        Wishart_Disc_high_speed_dim_d_weak(xt,dim,DTe,
    CorreLL[i-1][j-1]);
        for(l=0;l<dim;l++)
    for(m=0;m<dim;m++)
      {
        if(l!=m)
          pnl_mat_set(xt,l,m,pnl_mat_get(xt,l,m)/sqrt(pn
    l_mat_get(xt,l,l)*pnl_mat_get(xt,m,m)));
      }

        for(l=0;l<dim;l++)
        pnl_mat_set(xt,l,l, 1.);

        ODE_Compute(DT,kappa0, a0, dim,  eta0,  gamma0,
    It/I0, xt);

        Scheme_Basic_Stock_EL_Fast( St, xt,dim,(double) (
    j-1)*DT,DT,GaussEuler[i-1][j-1],r);
      }

  }

      It     = pnl_vect_scalar_prod(St,I_compo);
      tmmp   = strike - It;
      Ess    = Ess + tmmp;
      Varr   = Varr + tmmp*tmmp;

    }
  Ess              = (double)(Ess*exp(-r*maturity)/(double)Nb
    rMC);
  Varr             =  (double)(Varr*exp(-2.*r*maturity)/(
    double)NbrMC) - Ess*Ess;
  Varr             =  fabs(Varr);

  *(ptprice)         = Ess;
  *(pterror_price) = 2.*sqrt(Varr/((double)(NbrMC)));


  //-------------------------Free Local Memory
  Free_random_And_Time(1, NbrMC,NbrT);
  Free_vol_local_par();
  pnl_mat_free(&xt);
  pnl_vect_free(&St);

  return OK;
}

int CALC(MC_BASKET30D)(void *Opt,void *Mod,PricingMethod *
    Met)
{
  TYPEOPT* ptOpt=(TYPEOPT*)Opt;
  TYPEMOD* ptMod=(TYPEMOD*)Mod;

  double r;

  r=log(1.+ptMod->R.Val.V_DOUBLE/100.);

  return mc_basket30d( ptOpt->PayOff.Val.V_NUMFUNC_ND,
                       ptOpt->Maturity.Val.V_DATE-ptMod->T.
    Val.V_DATE,
                       r,
                       ptMod->kappa.Val.V_PDOUBLE,
                       ptMod->eta.Val.V_PDOUBLE,
                       ptMod->gama.Val.V_PDOUBLE,
                       ptMod->a.Val.V_PDOUBLE,
                       ptMod->InitialStocksWeights.Val.V_
    FILENAME,
                       ptMod->LocalVolatilities.Val.V_FIL
    ENAME,
                        ptMod->Basket_Correlation.Val.V_FIL
    ENAME,
                        ptMod->BasketLocalVolatility.Val.V_
    FILENAME,
                        Met->Par[0].Val.V_LONG,
                        Met->Par[1].Val.V_INT,
                        Met->Par[2].Val.V_ENUM.value,
                        &(Met->Res[0].Val.V_DOUBLE),
                        &(Met->Res[1].Val.V_DOUBLE)
    );
}

static int CHK_OPT(MC_BASKET30D)(void *Opt, void *Mod)
{
  if ((strcmp( ((Option*)Opt)->Name,"PutBasketEuro_nd")==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_LONG=15000;
      Met->Par[1].Val.V_INT=10;
      Met->Par[2].Val.V_ENUM.value=0;
      Met->Par[2].Val.V_ENUM.members=&PremiaEnumMCRNGs;
    }

  return OK;
}

PricingMethod MET(MC_BASKET30D)=
{
  "MC_BASKET30D",
  {{"N iterations",LONG,{100},ALLOW},
   {"TimeStepNumber",LONG,{100},ALLOW},
   {"RandomGenerator",ENUM,{100},ALLOW},
   {" ",PREMIA_NULLTYPE,{0},FORBID}},
  CALC(MC_BASKET30D),
  {{"Price",DOUBLE,{100},FORBID},
   {"Error Price",DOUBLE,{100},FORBID},
   {" ",PREMIA_NULLTYPE,{0},FORBID}},
  CHK_OPT(MC_BASKET30D),
  CHK_mc,
  MET(Init)
};
```

References