```
   Help
#include <stdlib.h>
#include "bsdisdiv1d_std.h"
#include "error_msg.h"
#include "pnl/pnl_mathtools.h"
#include "pnl/pnl_cdf.h"
#include "pnl/pnl_finance.h"

#define MALLOC_DOUBLE(n) malloc(n * sizeof(double))

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

}
#else

// calcul des d0 et d1 de la formule de Black-Scholes
static double  d0_d1(int Indice, double S0, double STRIKE,
              double TAUX, double DIVIDENDE,
              double VOL, double MATURITE)
{
  double valeur;
  if (MATURITE>0.0) // formule valable si maturitÈ non nulle
    valeur = (log(S0 / STRIKE))/ (VOL * sqrt(MATURITE)) + (
    (TAUX-DIVIDENDE) * sqrt (MATURITE)/ VOL) + VOL * sqrt(MATU
    RITE) * ((double)Indice-0.5);
  else
    if (S0>STRIKE) valeur=100.0;
    else valeur=-100.0;
  return valeur;
}
```

```
//calcul du prix d'un Call europeen
static double prix_call(double S0, double STRIKE,
                 double TAUX, double DIVIDENDE,
                 double VOL, double MATURITE)
{
  return pnl_bs_call (S0, STRIKE, MATURITE, TAUX, DIVIDEND
    E, VOL);
}

// calcul du deta d'un Call europeen
static double delta_call(double S0, double STRIKE,
                 double TAUX, double DIVIDENDE,
                 double VOL, double MATURITE)
{
  double ptprice, ptdelta;
  pnl_cf_call_bs (S0, STRIKE, MATURITE, TAUX, DIVIDENDE,     VOL, &ptprice, &ptd
  return ptdelta;
}

//dérivées par rapport au strike ordre 1
static double CallK1(double S0, double K, double r, double
    q, double vol, double T)
{
  return (-exp(-r*T)*pnl_cdfnor(d0_d1(0,S0,K,r,q,vol,T)));
}

//dérivées par rapport au strike ordre 1
static double CallKX(double S0, double K, double r, double
    q, double vol, double T)
{
  return (-exp(-r*T)*pnl_normal_density(d0_d1(0,S0,K,r,q,    vol,T)) / (S0 * vol
}



//dérivées par rapport au strike ordre 2
static double CallK2(double S0, double K, double r, double
    q, double vol, double T)
{
  return (exp(-r*T)/(sqrt(2*M_PI)*K*vol*sqrt(T))*exp(-0.5*
```

```
    d0_d1(0,S0,K,r,q,vol,T)*d0_d1(0,S0,K,r,q,vol,T)));
}

//dérivées par rapport au strike ordre 3
static double CallK3(double S0, double K, double r, double
    q, double vol, double T)
{
  return ( exp(-r*T)/(sqrt(2*M_PI)*K*K*vol*sqrt(T))*exp(-0.
    5*d0_d1(0,S0,K,r,q,vol,T)*d0_d1(0,S0,K,r,q,vol,T))*(1/vol/
    sqrt(T)*d0_d1(0,S0,K,r,q,vol,T) -1 ) );
}



//calcul des pi i n et pi i n delta
static void calcul_pin(double* pi0,double* pin, double *pi
    n_delta,
                       double* dates, double* y, double* de
    lta,
                       double vol, double T, int n)
{
  int i;
  pin[n-1]=1;
  for ( i=0 ; i<n-1 ; i++ ) { pin[n-2-i]=pin[n-1-i]*(1-y[n-
    1-i]); }
  if ( n==0 ) *pi0=1; else *pi0=pin[0]*(1-y[0]);
  for ( i=0 ; i<n ; i++ )
    {
      pin_delta[i] = pin[i] * exp (vol*vol * (T - dates[i])
    );
    }
}

//calcul des delta chapeau i
static void calcul_dci(double* dci, double* pin, double *dc
    i_delta, double *pin_delta,
                       double* dates, double* delta,
    double r, double q, double T, int n)
{
  int i;
  for (i=0;i<n;i++)
```

```
    {
      dci[i]=delta[i]*pin[i]*exp((r-q)*(T-dates[i]));
      dci_delta[i]=delta[i]*pin_delta[i]*exp((r-q)*(T-da
    tes[i]));
    }
}

static double delta_das1(double S0, double K, double r,
    double q, double vol,
                         double T, double* dates, double pi
    0, double* dci_delta, int n)
{
  int i;
  double K_delta;
  double res;

  K_delta = K;
  for ( i=0 ; i<n ; i++ ) { K_delta += dci_delta[i]; }

  res = delta_call(pi0*S0,K_delta,r,q,vol,T);

  for ( i=0 ; i<n ; i++ )
    {
      res += dci_delta[i] * ( CallKX(pi0*S0*exp(vol*vol*(T-
    dates[i])), K_delta, r, q, vol, T)
                              - CallKX( pi0*S0, K_delta, r,
     q, vol, T) );
    }
  return res * pi0;
}

/* //calcul par DAS ordre 1
 * static double prix_das1(double S0, double K, double r,
    double q, double vol,
 *                          double T, double* dates, double
    pi0, double* dci, int n)
 * {
 *   int i;
 *   double res=0;
 *   double Kt=K;
 *
```

```
*   //calcul de Ktilde
*   for (i=0;i<n;i++) { Kt+=dci[i]; }
*   //fin calcul de Kt
*
*   res=prix_call( pi0*S0, Kt, r, q, vol, T);
*
*   for (i=0;i<n;i++)
*     {
*       res+=dci[i]*(CallK1( pi0*S0*exp(vol*vol*(T-dates[
   i])), Kt, r, q, vol, T)
*                    - CallK1( pi0*S0, Kt, r, q, vol, T)
   );
*     }
*   return res;
* }
*
*
*
*
* //calcul par DAS ordre 2
* static double prix_das2(double S0, double K, double r,
   double q, double vol,
*                         double T, double* dates, double
   pi0, double* dci, int n)
* {
*   int i,j;
*   double res=0;
*   double SumS=0;
*   double Kt=K;
*
*   //calcul de Ktilde
*   for (i=0;i<n;i++){ SumS+=dci[i]; }
*   Kt+=SumS;
*   //fin calcul de Kt
*
*   //ordre 0
*   res=prix_call( pi0*S0, Kt, r, q, vol, T);
*
*
*   for (i=0;i<n;i++)
*     {
```

```
 *     //ordre 1
 *     res+=dci[i]*(CallK1( pi0*S0*exp(vol*vol*(T-dates[i])
    ), Kt, r, q, vol, T)
 *                  - CallK1( pi0*S0, Kt, r, q, vol, T)  );
 *
 *     //ordre 2
 *     for (j=0;j<n;j++)
 *       {
 *         res+=0.5*dci[i]*dci[j]*exp(vol*vol*(T-MAX(dates[
    i],dates[j])))*
 *           CallK2(pi0*S0*exp(vol*vol*(2*T-dates[i]-dates[
    j])),Kt, r, q, vol, T);
 *       }
 *
 *     res-=SumS*dci[i]*CallK2(pi0*S0*exp(vol*vol*(T-dates[
    i])),Kt, r, q, vol, T);
 *     }
 *
 *   //dernier terme ordre 2
 *   res+=0.5*SumS*SumS*CallK2(pi0*S0,Kt, r, q, vol, T);
 *
 *   return res;
 * } */


//calcul par DAS ordre 3
static double prix_das3(double S0, double K, double r,
    double q, double vol,
                        double T, double* dates, double pi0
    , double* dci, int n)
{
  int i,j,l;
  double res=0;
  double SumS=0;
  double Kt=K;


  //calcul de Ktilde
  for (i=0;i<n;i++){ SumS+=dci[i]; }
  Kt+=SumS;
  //fin calcul de Kt
```

```
//ordre 0
res=prix_call( pi0*S0, Kt, r, q, vol, T);


for (i=0;i<n;i++)
  {
  //ordre 1
  res+=dci[i]*(CallK1( pi0*S0*exp(vol*vol*(T-dates[i])),
  Kt, r, q, vol, T)
               - CallK1( pi0*S0, Kt, r, q, vol, T)  );

  for (j=0;j<n;j++)
    {
    //ordre 2
    res+=0.5*dci[i]*dci[j]*exp(vol*vol*(T-MAX(dates[i],da
  tes[j])))
      *CallK2(pi0*S0*exp(vol*vol*(2*T-dates[i]-dates[j]))
  ,Kt, r, q, vol, T);

    //ordre 3
    for (l=0;l<n;l++)
      {
        res+=1/6.0*dci[i]*dci[j]*dci[l]*
          exp(vol*vol*(3*T-MAX(dates[i],dates[j])-MAX(da
  tes[i],dates[l])-
                       MAX(dates[j],dates[l])))
          *CallK3(pi0*S0*exp(vol*vol*(3*T-dates[i]-dates[
  j]-dates[l])),Kt, r, q, vol, T);
      }

    res-=0.5*SumS*dci[i]*dci[j]*exp(vol*vol*(T-MAX(dates[
  i],dates[j])))
      *CallK3(pi0*S0*exp(vol*vol*(2*T-dates[i]-dates[j]))
  ,Kt, r, q, vol, T);
    }

  //ordre 2
  res-=SumS*dci[i]*CallK2(pi0*S0*exp(vol*vol*(T-dates[i])
  ),Kt, r, q, vol, T);
```

```
    //ordre 3
    res+=0.5*SumS*SumS*dci[i]*CallK3(pi0*S0*exp(vol*vol*(T-
    dates[i])),Kt, r, q, vol, T);
  }

  //dernier terme ordre 2
  res+=0.5*SumS*SumS*CallK2(pi0*S0,Kt, r, q, vol, T);

  //dernier terme ordre 3
  res-=1/6.0*SumS*SumS*SumS*CallK3(pi0*S0,Kt, r, q, vol, T)
    ;

  return res;
}

int CALC(AP_EtoreGobet)(void *Opt,void *Mod,PricingMethod *
    Met)
{
  TYPEOPT* ptOpt=(TYPEOPT*)Opt;
  TYPEMOD* ptMod=(TYPEMOD*)Mod;
  int i, n_dates;
  double S0, K, T;
  double r, q, vol;
  double *dates, *delta, *y;
  double pi0;
  double *pin, *pin_delta, *dci, *dci_delta;

  S0 = ptMod->S0.Val.V_PDOUBLE;
  K = ptOpt->PayOff.Val.V_NUMFUNC_1->Par[0].Val.V_DOUBLE;
  r=log(1.+ptMod->R.Val.V_DOUBLE/100.);
  q = 0.;
  vol = ptMod->Sigma.Val.V_PDOUBLE;
  T = ptOpt->Maturity.Val.V_DATE-ptMod->T.Val.V_DATE;
  dates = ptMod->Dates.Val.V_PNLVECT->array;
  n_dates = ptMod->Dates.Val.V_PNLVECT->size;
  delta = ptMod->Amounts.Val.V_PNLVECT->array;


  y=MALLOC_DOUBLE(n_dates);
  pin=MALLOC_DOUBLE(n_dates);
  pin_delta=MALLOC_DOUBLE(n_dates);
```

```
  dci=MALLOC_DOUBLE(n_dates);
  dci_delta=MALLOC_DOUBLE(n_dates);
  for (i=0;i<n_dates;i++){ y[i] = 0.;; }

  calcul_pin(&pi0, pin, pin_delta, dates, y, delta, vol, T,
     n_dates);
  calcul_dci(dci, pin, dci_delta, pin_delta, dates, delta,
    r, q, T, n_dates);

  Met->Res[0].Val.V_DOUBLE=prix_das3( S0,  K,  r, q, vol,
    T, dates,  pi0, dci, n_dates);
  Met->Res[1].Val.V_DOUBLE=delta_das1( S0,  K,  r, q, vol,
    T, dates,  pi0, dci_delta, n_dates);

  if ( ptOpt->PayOff.Val.V_NUMFUNC_1->Compute == &Put )
    {
      double sum_discount = 0.;
      for ( i=0 ; i<n_dates ; i++ ) { sum_discount += delt
    a[i] * exp(-r * dates[i]); }
      Met->Res[0].Val.V_DOUBLE += exp( -r*T) * K - S0 + su
    m_discount;
      Met->Res[1].Val.V_DOUBLE -= 1.;
    }

  free (pin);
  free (pin_delta);
  free (dci);
  free (dci_delta);
  free (y);

  return OK;
}

static int CHK_OPT(AP_EtoreGobet)(void *Opt, void *Mod)
{
  if ( (strcmp( ((Option*)Opt)->Name,"CallEuro")==0) ||
       (strcmp( ((Option*)Opt)->Name,"PutEuro")==0))
    return OK;
  return WRONG;
}
```

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

PricingMethod MET(AP_EtoreGobet)=
{
  "AP_EtoreGobet",
  {{" ",PREMIA_NULLTYPE,{0},FORBID}},
  CALC(AP_EtoreGobet),
  {{"Price",DOUBLE,{100},FORBID},{"Delta",DOUBLE,{100},FORB
    ID},{" ",PREMIA_NULLTYPE,{0},FORBID}},
  CHK_OPT(AP_EtoreGobet),
  CHK_ok,
  MET(Init)
};
```

References