Help

```
#include <stdlib.h>
#include  "bsdisdiv1d_std.h"
#include "error_msg.h"
#include "pnl/pnl_matrix.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
     (2009+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(TR_Vellekoop)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(TR_Vellekoop)(void*Opt,void *Mod,PricingMethod *
    Met)
{
return AVAILABLE_IN_FULL_PREMIA;
}
#else
static int Vellekoop(int am,double s,NumFunc_1  *p,double
    t,double r,double sigma,PnlVect *divid_dates,PnlVect *divid
    _amounts,int N, double *ptprice,double *ptdelta)
{
  int i,j,k,Nb_div,index;
  double u,d,h,pu,pd,a1,stock,lowerstock,dist1,dist2;
  double *P,*Q,*iv,*S,*vect_t,stock_div=0.;
  int *divid_steps;

  /*Number of Dividends Dates*/
  Nb_div=divid_dates->size;

  /*Compute steps of the tree*/
  N=N*Nb_div;

  /*Price, intrisic value arrays*/
  P= malloc((N+1)*sizeof(double));
  if (P==NULL)
    return MEMORY_ALLOCATION_FAILURE;
  Q= malloc((N+1)*sizeof(double));
  if (Q==NULL)
  return MEMORY_ALLOCATION_FAILURE;
iv= malloc((2*N+1)*sizeof(double));
if (iv==NULL)
  return MEMORY_ALLOCATION_FAILURE;
S= malloc((2*N+1)*sizeof(double));
if (S==NULL)
  return MEMORY_ALLOCATION_FAILURE;
vect_t= malloc((N+1)*sizeof(double));
 if (vect_t==NULL)
   return MEMORY_ALLOCATION_FAILURE;
 divid_steps= malloc((N+1)*sizeof(int));
 if ( divid_steps==NULL)
  return MEMORY_ALLOCATION_FAILURE;

 //for(i=0;i<Nb_div;i++) printf("%d %f
 //%f{n",i,divid_dates->array[i],divid_amounts->array[i])
  ;

 /*Up and Down factors*/
h=t/(double)N;
a1= exp(h*r);
u = exp(sigma*sqrt(h));
d= 1./u;

/*Risk-Neutral Probability*/
pu=(a1-d)/(u-d);
pd=1.-pu;

if ((pd>=1.) || (pd<=0.))
  return NEGATIVE_PROBABILITY;

pu*=exp(-r*h);
pd*=exp(-r*h);

for(i=0;i<=N;i++)
  vect_t[i]=h*(double)i;

//Compute steps related to the dividend dates
for(k=0;k<Nb_div;k++)
  {
    i=0;
```

```
      while(vect_t[i]<pnl_vect_get(divid_dates,k)) i++;
      if(fabs(pnl_vect_get(divid_dates,k)-vect_t[i])<1.e-10
    )
        divid_steps[k]=i;
      else
        {
          dist1=vect_t[i]-pnl_vect_get(divid_dates,k);
          dist2=pnl_vect_get(divid_dates,k)-vect_t[i-1];
          if (dist1<dist2)
            divid_steps[k]=i;
          else
            divid_steps[k]=i-1;
        }
    }

  /*Intrisic value initialisation*/
  lowerstock=s;
  for (i=0;i<N;i++)
    lowerstock*=d;
  stock=lowerstock;
  for(i=0;i<2*N+1;i++)
    {
      iv[i]=(p->Compute)(p->Par,stock);
      stock*=u;
    }

  /*Terminal Values*/
  for(j=0;j<=N;j++)
    P[j]=iv[2*j];

  /*Backward Resolution*/
  for(i=1;i<=N;i++)
    {

      for (j=0;j<=N-i;j++)
        {
          P[j]=pd*P[j]+pu*P[j+1];
          S[j]=s*pow(u,(double)(-(N-i)+2*j));
          Q[j]=P[j];
        }
      /*Dividends*/
      for(k=0;k<Nb_div;k++)
        if(i==divid_steps[k])
            for (j=0;j<=N-i;j++)
              {
                stock_div=S[j]-pnl_vect_get(divid_amounts,
    Nb_div-k-1);
                index=0;
                while(S[index]<stock_div) index++;
                if(index==0)
                 P[j]=Q[0];
                else//linear interpolation
                P[j]=((stock_div-S[index-1])*Q[index]
                      +(S[index]-stock_div)*Q[index-1])/(S[
    index]-S[index-1]);
              }

      if (am)
        for (j=0;j<=N-i;j++)
          P[j]=MAX(iv[i+2*j],P[j]);

      //Delta
      if(i==N-1)
          *ptdelta=(P[1]-P[0])/(s*u-s*d);
    }

  /*Price*/
  *ptprice=P[0];

  free(P);
  free(iv);
  free(Q);
  free(S);
  free(vect_t);
  free(divid_steps);

  return OK;
}

int CALC(TR_Vellekoop)(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 Vellekoop(ptOpt->EuOrAm.Val.V_BOOL,ptMod->S0.Val.
    V_PDOUBLE,ptOpt->PayOff.Val.V_NUMFUNC_1,ptOpt->Maturity.Val
    .V_DATE-ptMod->T.Val.V_DATE,r,ptMod->Sigma.Val.V_PDOUBLE,
    ptMod->Dates.Val.V_PNLVECT,ptMod->Amounts.Val.V_PNLVECT,
    Met->Par[0].Val.V_INT,&(Met->Res[0].Val.V_DOUBLE),&(Met->Res[
    1].Val.V_DOUBLE));
}
static int CHK_OPT(TR_Vellekoop)(void *Opt, void *Mod)
{
  return  OK;
}

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

      Met->Par[0].Val.V_INT2=100;
    }

  return OK;
}

PricingMethod MET(TR_Vellekoop)=
{
  "TR_Vellekoop",
  {{"StepNumbers between dividends dates",INT2,{100},ALLOW}
    ,{" ",PREMIA_NULLTYPE,{0},FORBID}},
  CALC(TR_Vellekoop),
  {{"Price",DOUBLE,{100},FORBID},{"Delta",DOUBLE,{100},FORB
    ID} ,{" ",PREMIA_NULLTYPE,{0},FORBID}},
  CHK_OPT(TR_Vellekoop),
  CHK_tree,
  MET(Init)
};
```

References