```
   Help
#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
     (2007+2) //The "#else" part of the code will be freely av
    ailable after the (year of creation of this file + 2)
#else

#ifndef CIRPP1DINCLUDES_H
#define CIRPP1DINCLUDES_H

#include<stdio.h>
#include<stdlib.h>
#include <math.h>
#include "cirpp1d_stdi.h"

#define pi 3.14159265358979
#define r3 1.73205

/*////////////////////////////////////////////////////
    ////////////////////////////////////////////////////
    ///////////////
//////////////////////////////////// Methods shared by sh
    ifted models (cir & hw) ////////////////////////////////
    ///////////////
////////////////////////////////////////////////////////
    ////////////////////////////////////////////////////
    ///////////////

static int lecture();                              Read the
    Z-coupon data of the market in a file
static double VarTree(double s);                   r=x+Shitf
    and x=VarTree(y), where y is variable compute in the tree,
    here x=y
static double Var(double s);                       Return
    variance of x
static double Var_y(double s);                     Return
    variance of y at time s, must be independent of y for CV (used in
    the tree)
static double Expect(double s);                    Return th
    e Expectation of r at time s.
static double ExpectCond(double x0, double s);     Return th
    e Expectation of r at time t+s, knowing x0 et time t (indep
```

```
    . of t)
static double ExpectCond_y(double x0, double s); Return th
    e Expectation of y at time t+s, knowing x0 et time t (used
    in the tree)

static double mu_r( double s, double r);         Return th
    e rate drift under neutral risk (not used here, used for th
    e EDP methode)
static double sigma_r(double s, double r);       Return th
    e volatility of r under neutral risk, (used for the EDP
    methode)

static double bond(double T);                    Return th
    e ZC bond value P(0,T) of the model, that is to say here th
    e market value
static double Shift(double s);                   Return th
    e Shift of the model (r=x+Shift)

static int DeleteMod(void);                      Free the
    memory allocated for all the tree varaibles


////////////////////////////////////////////////////////
    ////////////////////////////////////////////////////////
    /////////////////
//////////////////////////////////////////  Structure
    PDE and TREE for CIR++  ////////////////////////////////////
    /////////////////
////////////////////////////////////////////////////////////
    ////////////////////////////////////////////////////////
    ////////////////*/

struct Tree
{
  double *t;          /*Time step grid, from t[0] to T[Ng
    rid].*/
  double Tf;          /*Final time time of the tree, dt=Tf/
    Ngrid*/
  int Ngrid;          /*Numter of time step in the Tree, R[
    0][] is the first and R[Ngrid][] the last*/
  double **Payoffunc; /*Vector Payoff for the tree (see th
```

```
    e functions initPayoff--() for more explanations*/

  double P_T;          /*The value of the Z-C bond P(0,T)*/
  double **pLRij;      /*The value of the short rates in th
    e tree*/
  double **pLQij;      /*The value of the Options or other
    things (depend on Payoffunc) in the tree*/
  double **pLPDo;      /*Transition proba. in the trinomial
    tree for the down point*/
  double **pLPMi;      /*Transition proba. in the trinomial
    tree for the midle point*/
  double **pLPUp;      /*Transition proba. in the trinomial
    tree for the uper point*/
  int   **pLRef;       /*Reference index for the midle po
    int of the next time step scale rate*/
  int    *TSize;       /*Size of the scale rate at given
    time step.*/
};

struct EDP
{
  double *t;          /*Time step grid, from t[0] to T[Ngrid
    ].*/
  double Tf;          /*Final time time of the tree, dt=Tf/
    Ngrid*/
  int Ngrid;          /*Numter of time step in the Tree, R[0
    ] is the first and R[Ngrid][] the last*/

  double dx;          /*Pas d'espace (rate axis)*/
  double dt;          /*Time step*/
  int nx;             /*nombre de point d'espace*/
  double   Rm;

  double  **M1;       /*Matrice de l'EDP a inverser*/
  double  **M2;       /*Matrice du second membre de d'EDP*/
  double **Payoffunc;/*Matrix Payoff in space and time  or
    Matrix solution*/
};

int SetTimegrid_EDP(struct EDP *Meth, int n, double T);
double  OPTIONr_EDP(struct EDP *Meth, double r, double s,
```

```
    double T0);
void initPayoff1_EDP(struct EDP *Meth, double T0);

double OPTIONr_tr(struct Tree *Meth, double r, double s);
void initPayoff1_tr(struct Tree *Meth, double T0);
void freePayoff1_tr(struct Tree *Meth, double T0);

#endif
#endif //PremiaCurrentVersion
```

References