```
   Help
#include <stdlib.h>
#include  "hes1d_std.h"
#include "pnl/pnl_basis.h"
#include  "math/alfonsi.h"
#include "enums.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion <
     (2010+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_AM_Alfonsi_LongstaffSchwartz)(void *
    Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(MC_AM_Alfonsi_LongstaffSchwartz)(void *Opt,void *
    Mod,PricingMethod *Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else


/** Price of american put/call option using Longstaff-Schwa
    rtz algorithm **/
/** Heston model is simulated using the method proposed by
    Alfonsi **/
// Exercice dates are : T(0), T(1), ..., T(NbrExerciseDate
    s-1).
// with T(0)=0 and T(NbrExerciseDates-1)=Maturity.
static int MC_Am_Alfonsi_LoSc(NumFunc_1 *p, double S0,
    double Maturity, double r, double divid, double V0, double k,
    double theta, double sigma, double rho, long NbrMCsimulation,
    int NbrExerciseDates, int NbrStepPerPeriod, int generator,
    int basis_name, int DimApprox, int flag_cir, double *ptPriceA
    m)
{
  int m_in_money, j, m, nbr_var_explicatives, init_mc;
  int flag_SpotPaths, flag_VarPaths, flag_AveragePaths;
  double continuation_value, discounted_payoff, S_t, V_t;
  double discount_step, discount, time_step, exercise_date,
```

```
   mean_price;
double *VariablesExplicatives;

PnlMat *SpotPaths, *VarPaths, *AveragePaths, *Explicati
  veVariables;
PnlVect *VectToRegress, *DiscountedOptimalPayoff, *Regres
  sionCoeffVect;
PnlBasis *basis;

nbr_var_explicatives = 2;
init_mc=pnl_rand_init(generator, NbrExerciseDates*NbrStep
  PerPeriod, NbrMCsimulation);
if (init_mc != OK) return init_mc;

basis = pnl_basis_create(basis_name, DimApprox, nbr_var_e
  xplicatives);

VariablesExplicatives = malloc(nbr_var_explicatives*size
  of(double));

ExplicativeVariables = pnl_mat_create(NbrMCsimulation, nb
  r_var_explicatives);
DiscountedOptimalPayoff = pnl_vect_create(NbrMCsimulatio
  n);  // Payoff if following optimal strategy.
VectToRegress = pnl_vect_create(NbrMCsimulation);

RegressionCoeffVect = pnl_vect_new(); // Regression coe
  fficient.
SpotPaths = pnl_mat_new(); // Matrix of the whole trajec
  tories of the spot
VarPaths = pnl_mat_new(); // Matrix of the whole trajec
  tories of the variance
AveragePaths = pnl_mat_new();

time_step = Maturity / (double)(NbrExerciseDates-1);
discount_step = exp(-r*time_step);
discount = exp(-r*Maturity);

/* We store Spot and Variance*/
flag_SpotPaths = 1;
flag_VarPaths  = 1;
```

```
flag_AveragePaths = 0;

// Simulation of the whole paths
HestonSimulation_Alfonsi(flag_SpotPaths, SpotPaths, flag_
  VarPaths, VarPaths, flag_AveragePaths, AveragePaths, S0, Matu
  rity, r, divid, V0, k, theta, sigma, rho, NbrMCsimulation,
  NbrExerciseDates, NbrStepPerPeriod, generator, flag_cir);

// At maturity, the price of the option = discounted_payoff
exercise_date = Maturity;
for (m=0; m<NbrMCsimulation; m++)
  {
    S_t = MGET(SpotPaths, NbrExerciseDates-1, m); // Simu
  lated value of the spot at the maturity T
    LET(DiscountedOptimalPayoff, m) = discount * (p->
  Compute)(p->Par, S_t)/S0; // Discounted payoff
  }

for (j=NbrExerciseDates-2; j>=1; j--)
  {
    /** Least square fitting **/
    exercise_date -= time_step;
    discount /= discount_step;

    m_in_money=0;
    pnl_mat_resize(ExplicativeVariables, NbrMCsimulation,
   nbr_var_explicatives);
    pnl_vect_resize(VectToRegress, NbrMCsimulation);
    for (m=0; m<NbrMCsimulation; m++)
      {
        V_t = MGET(VarPaths, j, m); // Simulated value of
   the variance at t=exercise_date
        S_t = MGET(SpotPaths, j, m); // Simulated value
  of the spot at t=exercise_date

        discounted_payoff = discount * (p->Compute)(p->
  Par, S_t)/S0;

        if(discounted_payoff>0)
          {
            MLET(ExplicativeVariables, m_in_money, 0) =
```

```
S_t/S0;
          MLET(ExplicativeVariables, m_in_money, 1) =
V_t/V0;

          LET(VectToRegress, m_in_money) = GET(Discoun
tedOptimalPayoff, m);
          m_in_money++;
        }
    }

  pnl_mat_resize(ExplicativeVariables, m_in_money, nbr_
var_explicatives);
  pnl_vect_resize(VectToRegress, m_in_money);

  pnl_basis_fit_ls(basis, RegressionCoeffVect, Explic
ativeVariables, VectToRegress);

  /** Dynamical programming equation **/
  for (m=0; m<NbrMCsimulation; m++)
    {
      V_t = MGET(VarPaths, j, m);
      S_t = MGET(SpotPaths, j, m);
      discounted_payoff = discount * (p->Compute)(p->
Par, S_t)/S0; // Discounted payoff

      if (discounted_payoff>0.) // If the payoff is nul
l, the OptimalPayoff doesnt change.
        {
          VariablesExplicatives[0] = S_t/S0;
          VariablesExplicatives[1] = V_t/V0;

          continuation_value = pnl_basis_eval(basis,Reg
ressionCoeffVect, VariablesExplicatives);

          if (discounted_payoff > continuation_value)
            {
              LET(DiscountedOptimalPayoff, m) = discoun
ted_payoff;
            }
        }
    }
```

```
    }
  discount /= discount_step;

  // At initial date, no need for regression, continuation
    value is just a plain expectation estimated with empirical
    mean.
  continuation_value = pnl_vect_sum(DiscountedOptimalPayof
    f)/(double)NbrMCsimulation;
  discounted_payoff = discount*(p->Compute)(p->Par, S0)/S0;

  /* Price */
  mean_price = MAX(discounted_payoff, continuation_value);

  /* Price estimator */
  *ptPriceAm = S0 * mean_price;

  free(VariablesExplicatives);
  pnl_basis_free (&basis);
  pnl_mat_free(&SpotPaths);
  pnl_mat_free(&VarPaths);
  pnl_mat_free(&AveragePaths);
  pnl_mat_free(&ExplicativeVariables);

  pnl_vect_free(&DiscountedOptimalPayoff);
  pnl_vect_free(&RegressionCoeffVect);
  pnl_vect_free(&VectToRegress);

  return OK;
}

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

  double r,divid;

  r=log(1.+ptMod->R.Val.V_DOUBLE/100.);
  divid=log(1.+ptMod->Divid.Val.V_DOUBLE/100.);
  Met->Par[1].Val.V_INT = MAX(2, Met->Par[1].Val.V_INT); //
```

```
     At least two exercise dates.


  return MC_Am_Alfonsi_LoSc(ptOpt->PayOff.Val.V_NUMFUNC_1,
                            ptMod->S0.Val.V_PDOUBLE,
                            ptOpt->Maturity.Val.V_DATE-pt
    Mod->T.Val.V_DATE,
                            r,
                            divid,
                            ptMod->Sigma0.Val.V_PDOUBLE,
                            ptMod->MeanReversion.hal.V_PDO
    UBLE,
                            ptMod->LongRunVariance.Val.V_
    PDOUBLE,
                            ptMod->Sigma.Val.V_PDOUBLE,
                            ptMod->Rho.Val.V_PDOUBLE,
                            Met->Par[0].Val.V_LONG,
                            Met->Par[1].Val.V_INT,
                            Met->Par[2].Val.V_INT,
                            Met->Par[3].Val.V_ENUM.value,
                            Met->Par[4].Val.V_ENUM.value,
                            Met->Par[5].Val.V_INT,
                            Met->Par[6].Val.V_ENUM.value,
                            &(Met->Res[0].Val.V_DOUBLE));
}

static int CHK_OPT(MC_AM_Alfonsi_LongstaffSchwartz)(void *
    Opt, void *Mod)
{
  Option* ptOpt=(Option*)Opt;
  TYPEOPT* opt=(TYPEOPT*)(ptOpt->TypeOpt);

  if ((opt->EuOrAm).Val.V_BOOL==AMER)
    return OK;
  else
    return WRONG;
}
#endif //PremiaCurrentVersion

static int MET(Init)(PricingMethod *Met,Option *Opt)
{
```

```
  if ( Met->init == 0)
    {
      Met->init=1;

      Met->Par[0].Val.V_LONG=50000;
      Met->Par[1].Val.V_INT=20;
      Met->Par[2].Val.V_INT=1;
      Met->Par[3].Val.V_ENUM.value=0;
      Met->Par[3].Val.V_ENUM.members=&PremiaEnumRNGs;
      Met->Par[4].Val.V_ENUM.value=0;
      Met->Par[4].Val.V_ENUM.members=&PremiaEnumBasis;
      Met->Par[5].Val.V_INT=10;
      Met->Par[6].Val.V_ENUM.value=2;
      Met->Par[6].Val.V_ENUM.members=&PremiaEnumCirOrder;
    }

  return OK;
}


PricingMethod MET(MC_AM_Alfonsi_LongstaffSchwartz)=
{
  "MC_AM_Alfonsi_LongstaffSchwartz",
  {
    {"N Simulations",LONG,{100},ALLOW},
    {"N Exercise Dates",INT,{100},ALLOW},
    {"N Steps per Period",INT,{100},ALLOW},
    {"RandomGenerator",ENUM,{100},ALLOW},
    {"Basis",ENUM,{100},ALLOW},
    {"Dimension Approximation",INT,{100},ALLOW},
    {"Cir Order",ENUM,{100},ALLOW},
    {" ",PREMIA_NULLTYPE,{0},FORBID}},
  CALC(MC_AM_Alfonsi_LongstaffSchwartz),
  {   {"Price",DOUBLE,{100},FORBID},
      {" ",PREMIA_NULLTYPE,{0},FORBID}},
  CHK_OPT(MC_AM_Alfonsi_LongstaffSchwartz),
  CHK_ok,
  MET(Init)
};
```

# References