```
   Help
#include <stdlib.h>
#include  "merhes1d_std.h"
#include  "math/alfonsi.h"
#include "pnl/pnl_basis.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_AndersenBroadie_Bates)(voi
    d *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(MC_AM_Alfonsi_AndersenBroadie_Bates)(void *Opt,voi
    d *Mod,PricingMethod *Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
#else

/** Lower bound for american option using Longstaff-Schwa
    rtz algorithm **/
// 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_Bates(NumFunc_1 *p, double S0
    , double Maturity, double r, double divid, double V0,
    double k, double theta, double sigma, double rho, double mu_
    jump,double gamma2,double lambda, long NbrMCsimulation, int Nb
    rExerciseDates, int NbrStepPerPeriod, int generator,  int
    basis_name, int DimApprox, int flag_cir, PnlMat* Regression
    CoeffMat, double *ContinuationValue_0)
{
  int j, m, nbr_var_explicatives;
  int flag_SpotPaths, flag_VarPaths, flag_AveragePaths;
  double regressed_value, discounted_payoff, S_t, V_t, dis
    count, discount_step, step, exercise_date, european_price, eu
    ropean_delta;
  double *VariablesExplicatives;
```

```
PnlMat *SpotPaths, *VarPaths, *AveragePaths, *Explicati
  veVariables;
PnlVect *DiscountedOptimalPayoff, *RegressionCoeffVect;
PnlBasis *basis;

pnl_mat_resize(RegressionCoeffMat, NbrExerciseDates-2,
  DimApprox);

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

nbr_var_explicatives = 2;

/* We store Spot and Variance*/
flag_SpotPaths = 1;
flag_VarPaths  = 1;
flag_AveragePaths = 0;

european_price = 0.;
european_delta = 0.;

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); // Continuation Value

RegressionCoeffVect = pnl_vect_create(0);
SpotPaths = pnl_mat_create(0, 0); // Matrix of the whole
  trajectories of the spot
VarPaths = pnl_mat_create(0, 0); // Matrix of the whole
  trajectories of the variance
AveragePaths = pnl_mat_create(0, 0);
```

```
// Simulation of the whole paths
BatesSimulation_Alfonsi(flag_SpotPaths, SpotPaths, flag_
  VarPaths, VarPaths, flag_AveragePaths, AveragePaths, S0, Matu
  rity, r, divid, V0, k, theta, sigma, rho, mu_jump, gamma2,
  lambda, 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->Compu
  te)(p->Par, S_t); // Price of the option = discounted_payoff
  }

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

    for (m=0; m<NbrMCsimulation; m++)
      {
        V_t = MGET(VarPaths, j, m); // Simulated value of
   the variance
        S_t = MGET(SpotPaths, j, m); // Simulated value
  of the spot
        ApAlosHeston(S_t, p, Maturity-exercise_date, r,
  divid, V_t, k, theta, sigma,rho, &european_price, &european_
  delta);

        MLET(ExplicativeVariables, m, 0) = discount*euro
  pean_price/S0;
        MLET(ExplicativeVariables, m, 1) = discount*euro
  pean_delta*S_t*sqrt(V_t)/S0;
      }

    pnl_basis_fit_ls(basis,RegressionCoeffVect, Explicati
  veVariables, DiscountedOptimalPayoff);
```

```
    pnl_mat_set_row(RegressionCoeffMat, RegressionCoeffV
  ect, j-1); // Save regression coefficients in RegressionCoe
  ffMat.

    /** Dynamical programming equation **/
    for (m=0; m<NbrMCsimulation; m++)
      {
        V_t = MGET(VarPaths, j, m); // Simulated value of
   the variance
        S_t = MGET(SpotPaths, j, m); // Simulated value
  of the spot
        discounted_payoff = discount*(p->Compute)(p->Par,
   S_t);  // Payoff pour la m ieme simulation

        if (discounted_payoff>0) // If the discounted_
  payoff is null, the OptimalPayoff doesnt change.
          {
            ApAlosHeston(S_t, p, Maturity-exercise_date,
  r, divid, V_t, k, theta, sigma,rho, &european_price, &euro
  pean_delta);

            VariablesExplicatives[0] = discount*european_
  price/S0;
            VariablesExplicatives[1] = discount*european_
  delta*S_t*sqrt(V_t)/S0;

            regressed_value = pnl_basis_eval(basis,Regres
  sionCoeffVect, VariablesExplicatives);

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

// At initial date, no need for regression, conditional
  expectation is just a plain expectation, estimated with empi
```

```
    rical mean.
  *ContinuationValue_0 = pnl_vect_sum(DiscountedOptimalPay
    off)/NbrMCsimulation;

  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);

  return OK;
}

/** Upper bound for american option using Andersen and Broa
    die algorithm.
 * @param AmOptionUpperPrice upper bound for the price on
    exit.
 * @param NbrMCsimulationDual number of outer simulation
    in Andersen and Broadie algorithm.
 * @param NbrMCsimulationDualInternal  number of inner simu
    lation in Andersen and Broadie algorithm.
 * @param NbrMCsimulationPrimal number of simulation in Lon
    gstaff-Schwartz algorithm.
 */
static int MC_Am_Alfonsi_AnBr_Bates(double S0, double Matu
    rity, double r, double divid, double V0, double k, double th
    eta, double sigma, double rho, double mu_jump,double gamma2
    ,double lambda, long NbrMCsimulationPrimal, long NbrMCsimu
    lationDual, long NbrMCsimulationDualInternal, int NbrExercis
    eDates, int NbrStepPerPeriod, int generator,  int basis_na
    me, int DimApprox, int flag_cir, NumFunc_1 *p, double *Am
    OptionUpperPrice)
{
  int m, m_i, i, nbr_var_explicatives, ExerciceOrContinua
    tion, init_mc;
  int flag_SpotPaths, flag_VarPaths, flag_AveragePaths;
  double discounted_payoff, discounted_payoff_inner, Conti
```

```
    nuationValue, LowerPriceOld, LowerPrice, LowerPrice_0, Conti
    nuationValue_0;

  double DoobMeyerMartingale, MaxVariable, S_t, V_t, S_t_
    inner, V_t_inner, ContinuationValue_inner;
  double discount_step, discount, step, exercise_date, Cond
    Expec_inner, Delta_0, european_price, european_delta;
  double *VariablesExplicatives;

  PnlMat *RegressionCoeffMat;
  PnlMat *SpotPaths, *SpotPaths_inner;
  PnlMat *VarPaths, *VarPaths_inner, *AveragePaths;
  PnlVect *RegressionCoeffVect;
  PnlBasis *basis;

  SpotPaths = pnl_mat_create(0, 0); /* Matrix of the whole
    trajectories of the spot */
  VarPaths = pnl_mat_create(0, 0); /* Matrix of the whole
    trajectories of the variance */
  SpotPaths_inner = pnl_mat_create(0, 0);
  VarPaths_inner = pnl_mat_create(0, 0);
  RegressionCoeffVect = pnl_vect_create(0);
  RegressionCoeffMat = pnl_mat_create(0, 0);
  AveragePaths = pnl_mat_create(0, 0);

  /* We store Spot and Variance*/
  flag_SpotPaths = 1;
  flag_VarPaths  = 1;
  flag_AveragePaths = 0;

  european_price = 0.;
  european_delta = 0.;

  ContinuationValue_0 = 0.;
  CondExpec_inner = 0;

  step = Maturity / (NbrExerciseDates-1);
  discount_step = exp(-r*step);
  discount = 1.;

  nbr_var_explicatives = 2;
```

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

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

/* Compute the lower price with Longstaff-Schwartz algor
  ithm and save the regression coefficient in RegressionCoeffM
  at. */
MC_Am_Alfonsi_LoSc_Bates(p, S0, Maturity, r, divid, V0,
  k, theta, sigma, rho, mu_jump, gamma2, lambda, NbrMCsimulat
  ionPrimal, NbrExerciseDates, NbrStepPerPeriod, generator,
  basis_name, DimApprox, flag_cir, RegressionCoeffMat, &Conti
  nuationValue_0);

discounted_payoff = discount*(p->Compute)(p->Par, S0);
LowerPrice_0 = MAX(discounted_payoff, ContinuationValue_0
  ); // Price of am.option at initial date t=0.

/* Simulation of the whole paths. These paths are indep
  endants of those used in Longstaff-Schwartz algorithm. */
BatesSimulation_Alfonsi (flag_SpotPaths, SpotPaths, flag_
  VarPaths, VarPaths, flag_AveragePaths, AveragePaths, S0, Matu
  rity, r, divid, V0, k, theta, sigma, rho, mu_jump, gamma2,
  lambda, NbrMCsimulationDual, NbrExerciseDates, NbrStepPerP
  eriod, generator, flag_cir);

basis = pnl_basis_create(basis_name, DimApprox, nbr_var_e
  xplicatives);
Delta_0 = 0;

for (m=0; m<NbrMCsimulationDual; m++)
  {
    exercise_date = 0.;
    MaxVariable = 0.;
    discount = 1.;
    S_t = S0;
    V_t = V0;

    ContinuationValue = ContinuationValue_0;
    discounted_payoff = discount*(p->Compute)(p->Par, S_
  t);

    LowerPrice = MAX(discounted_payoff, ContinuationValu
  e);
    LowerPriceOld = LowerPrice;
    DoobMeyerMartingale = LowerPrice;

    /* Initialization of the duale variable. */
    MaxVariable = MAX(MaxVariable, discounted_payoff-Doo
  bMeyerMartingale);

    for (i=1; i<=NbrExerciseDates-2; i++)
      {
        discount *= discount_step;
        exercise_date += step;

        pnl_mat_get_row(RegressionCoeffVect, Regression
  CoeffMat, i-1);

        ExerciceOrContinuation = (discounted_payoff >
  ContinuationValue);
        // If ExerciceOrContinuation=Exercice, we estima
  te the conditionnal expectation of the lower price.
        if (ExerciceOrContinuation)
          {
            CondExpec_inner = 0;

            BatesSimulation_Alfonsi(flag_SpotPaths, SpotP
  aths_inner, flag_VarPaths, VarPaths_inner, flag_AveragePath
  s, AveragePaths, S_t, step, r, divid, V_t, k, theta, sigma,
   rho, mu_jump, gamma2, lambda, NbrMCsimulationDualIntern
  al, 2, NbrStepPerPeriod, generator, flag_cir);

            for (m_i=0; m_i<NbrMCsimulationDualInternal;
  m_i++)
              {
                S_t_inner = MGET(SpotPaths_inner, 1, m_i)
  ;
                V_t_inner = MGET(VarPaths_inner, 1, m_i);
                discounted_payoff_inner = discount*(p->
  Compute)(p->Par, S_t_inner);

                ApAlosHeston(S_t_inner, p, Maturity-exerc
  ise_date, r, divid, V_t_inner, k, theta, sigma,rho, &europe
  an_price, &european_delta);

                VariablesExplicatives[0] = discount*euro
  pean_price/S0;
                VariablesExplicatives[1] = discount*euro
  pean_delta*S_t*sqrt(V_t)/S0;

                ContinuationValue_inner = pnl_basis_eval(
  basis,RegressionCoeffVect, VariablesExplicatives);

                CondExpec_inner += MAX(discounted_payoff_
  inner, ContinuationValue_inner);

              }

            CondExpec_inner /= (double)NbrMCsimulationDua
  lInternal;
          }

        S_t = MGET(SpotPaths, i, m);
        V_t = MGET(VarPaths, i, m);
        discounted_payoff = discount*(p->Compute)(p->Par,
   S_t);

        ApAlosHeston(S_t, p, Maturity-exercise_date, r,
  divid, V_t, k, theta, sigma,rho, &european_price, &european_
  delta);

        VariablesExplicatives[0] = discount*european_
  price/S0;
        VariablesExplicatives[1] = discount*european_delt
  a*S_t*sqrt(V_t)/S0;

        ContinuationValue = pnl_basis_eval(basis,Regressi
  onCoeffVect, VariablesExplicatives);

        LowerPrice = MAX(discounted_payoff, ContinuationV
  alue);

        /* Compute the martingale part in Doob Meyer de
  composition of the lower price process. */
        if (ExerciceOrContinuation)
          {
            DoobMeyerMartingale = DoobMeyerMartingale +
  LowerPrice - CondExpec_inner;

          }
        else
          {
            DoobMeyerMartingale = DoobMeyerMartingale +
  LowerPrice - LowerPriceOld;
          }

        MaxVariable = MAX(MaxVariable, discounted_payoff-
  DoobMeyerMartingale);

        LowerPriceOld = LowerPrice;
      }

    /** Last Exercice Date. The price of the option here
  is equal to the discounted_payoff.**/
    discount *= discount_step;

    ExerciceOrContinuation = (discounted_payoff > Conti
  nuationValue); // Decision to exerice or not before the
  last exercice date.
    if (ExerciceOrContinuation)
      {
        BatesSimulation_Alfonsi(flag_SpotPaths, SpotPath
  s_inner, flag_VarPaths, VarPaths_inner, flag_AveragePaths,
  AveragePaths, S_t, step, r, divid, V_t, k, theta, sigma, rh
  o, mu_jump, gamma2, lambda, NbrMCsimulationDualInternal, 2,
   NbrStepPerPeriod, generator, flag_cir);

        CondExpec_inner = 0;
        for (m_i=0; m_i<NbrMCsimulationDualInternal; m_i+
  +)
          {
```

```
      S_t_inner = MGET(SpotPaths_inner, 1, m_i);
      discounted_payoff_inner = discount*(p->Compu
 te)(p->Par, S_t_inner);
      CondExpec_inner += discounted_payoff_inner;
     }
    CondExpec_inner /= (double) NbrMCsimulationDua
 lInternal;
   }

  S_t = MGET(SpotPaths, NbrExerciseDates-1, m);
  discounted_payoff = discount*(p->Compute)(p->Par, S_
 t);
  LowerPrice = discounted_payoff;

  if (ExerciceOrContinuation)
   {
    DoobMeyerMartingale = DoobMeyerMartingale + Low
 erPrice - CondExpec_inner;

   }
  else
   {
    DoobMeyerMartingale = DoobMeyerMartingale + Low
 erPrice - LowerPriceOld;
   }

  MaxVariable = MAX(MaxVariable, discounted_payoff-Doo
 bMeyerMartingale);

  Delta_0 += MaxVariable;
 }

Delta_0 /= NbrMCsimulationDual;
*AmOptionUpperPrice = LowerPrice_0 + 0.5*Delta_0;

free(VariablesExplicatives);
pnl_mat_free(&SpotPaths);
pnl_mat_free(&VarPaths);
pnl_mat_free(&SpotPaths_inner);
pnl_mat_free(&VarPaths_inner);
pnl_mat_free(&RegressionCoeffMat);
```

```
  pnl_mat_free(&AveragePaths);
  pnl_vect_free(&RegressionCoeffVect);

  return init_mc;

}

int CALC(MC_AM_Alfonsi_AndersenBroadie_Bates)(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.);

  return MC_Am_Alfonsi_AnBr_Bates(ptMod->S0.Val.V_PDOUBLE,
                                  ptOpt->Maturity.Val.V_DA
    TE-ptMod->T.Val.V_DATE,
                                  r,
                                  divid,
                                  ptMod->Sigma0.Val.V_PDOUB
    LE,
                                  ptMod->MeanReversion.hal.
    V_PDOUBLE,
                                  ptMod->LongRunVariance.
    Val.V_PDOUBLE,
                                  ptMod->Sigma.Val.V_PDOUB
    LE,
                                  ptMod->Rho.Val.V_PDOUBLE,
                                  ptMod->Mean.Val.V_PDOUB
    LE,
                                  ptMod->Variance.Val.V_PDO
    UBLE,
                                  ptMod->Lambda.Val.V_PDOUB
    LE,
                                  Met->Par[0].Val.V_LONG,
                                  Met->Par[1].Val.V_LONG,
                                  Met->Par[2].Val.V_LONG,
                              Met->Par[3].Val.V_INT,
                              Met->Par[4].Val.V_INT,
                              Met->Par[5].Val.V_ENUM.
    value,
                              Met->Par[6].Val.V_ENUM.
    value,
                              Met->Par[7].Val.V_INT,
                              Met->Par[8].Val.V_ENUM.
    value,
                              ptOpt->PayOff.Val.V_
    NUMFUNC_1,
                              &(Met->Res[0].Val.V_
    DOUBLE));
}

static int CHK_OPT(MC_AM_Alfonsi_AndersenBroadie_Bates)(voi
    d *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=100000;
      Met->Par[1].Val.V_LONG=500;
      Met->Par[2].Val.V_LONG=500;
      Met->Par[3].Val.V_INT=10;
      Met->Par[4].Val.V_INT=1;
      Met->Par[5].Val.V_ENUM.value=0;
      Met->Par[5].Val.V_ENUM.members=&PremiaEnumRNGs;
      Met->Par[6].Val.V_ENUM.value=0;
      Met->Par[6].Val.V_ENUM.members=&PremiaEnumBasis;
      Met->Par[7].Val.V_INT=10;
      Met->Par[8].Val.V_ENUM.value=2;
      Met->Par[8].Val.V_ENUM.members=&PremiaEnumCirOrder;
    }

  return OK;
}


PricingMethod MET(MC_AM_Alfonsi_AndersenBroadie_Bates)=
{
  "MC_AM_Alfonsi_AndersenBroadie_MerHes",
  {
    {"N Sim.Primal",LONG,{100},ALLOW},
    {"N Sim.Dual",LONG,{100},ALLOW},
    {"N Sim.Dual Internal",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_AndersenBroadie_Bates),
  {{"Price",DOUBLE,{100},FORBID}, {" ",PREMIA_NULLTYPE,{0},
    FORBID}},
  CHK_OPT(MC_AM_Alfonsi_AndersenBroadie_Bates),
  CHK_ok,
  MET(Init)
};
```

References