Help

```
#include <stdlib.h>
#include  "merhes1d_pad.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_Asian_Alfonsi_AndersenBroadie_Bate
    s)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
int CALC(MC_Am_Asian_Alfonsi_AndersenBroadie_Bates)(void *
    Opt,void *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(NumFunc_2  *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 NbrExercis
    eDates, int NbrStepPerPeriod, int generator,  int basis_na
    me, int DimApprox, int flag_cir, PnlMat* RegressionCoeffMat,
     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, A_
    t, discount, discount_step, step;
    double exercise_date, european_price, european_delta,
    V_mean;
```

```
double *VariablesExplicatives;

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

european_price = 0.;
european_delta = 0.;

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 = 1;

basis = pnl_basis_create(basis_name, DimApprox, nbr_
var_explicatives);

pnl_mat_resize(RegressionCoeffMat, NbrExerciseDates-2,
DimApprox);

VariablesExplicatives = malloc(nbr_var_explicatives*si
zeof(double));

ExplicativeVariables = pnl_mat_create(NbrMCsimulation,
nbr_var_explicatives);
DiscountedOptimalPayoff = pnl_vect_create(NbrMCsimulat
ion); // Payoff if following optimal strategy.

RegressionCoeffVect = pnl_vect_create(0); // Regression
 coefficient.
SpotPaths = pnl_mat_create(0, 0); // Matrix of the whol
e trajectories of the spot.
VarPaths = pnl_mat_create(0, 0); // Matrix of the whol
e trajectories of the variance.
```

```
AveragePaths = pnl_mat_create(0, 0); // Matrix of the
whole trajectories of the average.

// Simulation of the whole paths
BatesSimulation_Alfonsi(flag_SpotPaths, SpotPaths, fla
g_VarPaths, VarPaths, flag_AveragePaths, AveragePaths, S0,
Maturity, r, divid, V0, k, theta, sigma, rho, mu_jump, gamma2
, lambda, NbrMCsimulation, NbrExerciseDates, NbrStepPerP
eriod, 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); // Si
mulated Value of the spot at the maturity T
    A_t = MGET(AveragePaths, NbrExerciseDates-1, m); //
 Simulated Value of the average at the maturity T
    LET(DiscountedOptimalPayoff, m) = discount * (p->
Compute)(p->Par, S_t, A_t);
}

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
        A_t = MGET(AveragePaths, j, m);

        // Regression basis contains price and delta of
 european asian option (under Black-Scholes model) and th
eirs power.
        // As BS volatility, we take sqrt of expectatio
```

```
n of V(Maturity) knowing that V(exercise_date)=V_t.
        V_mean = theta + (V_t-theta)*exp(-k*(Maturity-
exercise_date));
        Ap_FixedAsian_BlackScholes(S_t, A_t, exercise_
date, p, Maturity, r, divid, sqrt(V_mean), &european_price,
 &european_delta);

        MLET(ExplicativeVariables, m, 0) = discount*eu
ropean_price/S0;
        MLET(ExplicativeVariables, m, 1) = discount*eu
ropean_delta*S_t*sqrt(V_t)/S0;
    }

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

    pnl_mat_set_row(RegressionCoeffMat, RegressionCoe
ffVect, j-1); // Save regression coefficients in Regression
CoeffMat.

    /** 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
        A_t = MGET(AveragePaths, j, m);

        discounted_payoff = discount * (p->Compute)(p->
Par, S_t, A_t);  // Payoff pour la m ieme simulation

        if (discounted_payoff>0) // If the discounted_
payoff is null, the OptimalPayoff doesnt change.
        {
            V_mean = theta + (V_t-theta)*exp(-k*(Matu
rity-exercise_date));
            Ap_FixedAsian_BlackScholes(S_t, A_t, exerc
ise_date, p, Maturity, r, divid, sqrt(V_mean), &european_
price, &european_delta);
```

```
            VariablesExplicatives[0] = discount*europe
an_price/S0;
            VariablesExplicatives[1] = discount*europe
an_delta*S_t*sqrt(V_t)/S0;

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

            if (discounted_payoff > regressed_value)
            {
                LET(DiscountedOptimalPayoff, m) = dis
counted_payoff;
            }
        }
    }
}

// At initial date, no need for regression, condition
al expectation is just a plain expectation, estimated with
empirical mean.
*ContinuationValue_0 = pnl_vect_sum(DiscountedOptimalP
ayoff)/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(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 NbrMCsimulationPrimal, long NbrMCsimulationDua
    l, long NbrMCsimulationDualInternal, int NbrExerciseDates,
    int NbrStepPerPeriod, int generator,  int basis_name, int Dim
    Approx, int flag_cir, NumFunc_2 *p, double *AmOptionUpperP
    rice)
{
    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,
    ContinuationValue, LowerPriceOld, LowerPrice, LowerPrice_0,
    ContinuationValue_0;

    double DoobMeyerMartingale, MaxVariable, S_t, V_t, A_t,
     S_t_inner, V_t_inner, A_t_inner, ContinuationValue_inner;
    double discount_step, discount, step, exercise_date,
    CondExpec_inner, Delta_0, european_price, european_delta, V_
    mean;
    double *VariablesExplicatives;

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

    SpotPaths = pnl_mat_create(0, 0); /* Matrix of the whol
    e trajectories of the spot */
    VarPaths = pnl_mat_create(0, 0); /* Matrix of the whol
    e trajectories of the variance */
    AveragePaths = pnl_mat_create(0, 0);
```

```
AveragePaths_inner = pnl_mat_create(0, 0);
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);

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

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*si
zeof(double));

init_mc=pnl_rand_init(generator, NbrExerciseDates*Nb
rStepPerPeriod, NbrMCsimulationPrimal);
if (init_mc != OK) return init_mc;

/* Compute the lower price with Longstaff-Schwartz alg
orithm and save the regression coefficient in RegressionCoe
ffMat. */
MC_Am_Alfonsi_LoSc(p, S0, Maturity, r, divid, V0, k, th
eta, sigma, rho, mu_jump, gamma2, lambda, NbrMCsimulationP
rimal, NbrExerciseDates, NbrStepPerPeriod, generator, basis_
name, DimApprox, flag_cir, RegressionCoeffMat, &Continuatio
nValue_0);

discounted_payoff = discount*(p->Compute)(p->Par, S0,
S0);
LowerPrice_0 = MAX(discounted_payoff, ContinuationValu
e_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, fla
g_VarPaths, VarPaths, flag_AveragePaths, AveragePaths, S0,
Maturity, r, divid, V0, k, theta, sigma, rho, mu_jump, gamma2
, lambda, NbrMCsimulationDual, NbrExerciseDates, NbrStepPe
rPeriod, generator, flag_cir);

basis = pnl_basis_create(basis_name, DimApprox, nbr_
var_explicatives);
Delta_0 = 0;

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

    ContinuationValue = ContinuationValue_0;
    discounted_payoff = discount*(p->Compute)(p->Par,
S_t, A_t);

    LowerPrice = MAX(discounted_payoff, ContinuationV
alue);
    LowerPriceOld = LowerPrice;
    DoobMeyerMartingale = LowerPrice;

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

    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 es
timate the conditionnal expectation of the lower price.
        if (ExerciceOrContinuation)
        {
            CondExpec_inner = 0;

            BatesSimulation_Alfonsi(flag_SpotPaths, Spo
tPaths_inner, flag_VarPaths, VarPaths_inner, flag_AverageP
aths, AveragePaths_inner, S_t, step, r, divid, V_t, k, thet
a, sigma, rho, mu_jump, gamma2, lambda, NbrMCsimulationDua
lInternal, 2, NbrStepPerPeriod, generator, flag_cir);

            for (m_i=0; m_i<NbrMCsimulationDualIntern
al; m_i++)
            {
                S_t_inner = MGET(SpotPaths_inner, 1, m_
i);
                V_t_inner = MGET(VarPaths_inner, 1, m_
i);
                A_t_inner = MGET(AveragePaths_inner, 1,
 m_i);

                discounted_payoff_inner = discount*(p->
Compute)(p->Par, S_t_inner, A_t_inner);

                V_mean = theta + (V_t_inner-theta)*exp(
-k*(Maturity-exercise_date));
                Ap_FixedAsian_BlackScholes(S_t_inner,
A_t_inner, exercise_date, p, Maturity, r, divid, sqrt(V_mea
n), &european_price, &european_delta);

                VariablesExplicatives[0] = discount*eu
ropean_price/S0;
                VariablesExplicatives[1] = discount*eu
ropean_delta*S_t*sqrt(V_t)/S0;

                ContinuationValue_inner = pnl_basis_ev
al(basis,RegressionCoeffVect, VariablesExplicatives);
```

```
                CondExpec_inner += MAX(discounted_payo
ff_inner, ContinuationValue_inner);

            }

            CondExpec_inner /= (double)NbrMCsimulationD
ualInternal;
        }

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

        V_mean = theta + (V_t-theta)*exp(-k*(Maturity-
exercise_date));
        Ap_FixedAsian_BlackScholes(S_t, A_t, exercise_
date, p, Maturity, r, divid, sqrt(V_mean), &european_price,
 &european_delta);

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

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

        LowerPrice = MAX(discounted_payoff, Continuatio
nValue);

        /* 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_payo
ff-DoobMeyerMartingale);

        LowerPriceOld = LowerPrice;
    }

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

    // Decision to exerice or not before the last exerc
ice date.
    ExerciceOrContinuation = (discounted_payoff > Conti
nuationValue);

    if (ExerciceOrContinuation)
    {
        BatesSimulation_Alfonsi(flag_SpotPaths, SpotP
aths_inner, flag_VarPaths, VarPaths_inner, flag_AveragePath
s, AveragePaths_inner, S_t, step, r, divid, V_t, k, theta,
sigma, rho, mu_jump, gamma2, lambda, NbrMCsimulationDualInt
ernal, 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);
            A_t_inner = MGET(AveragePaths, 1, m_i);
            discounted_payoff_inner = discount*(p->
Compute)(p->Par, S_t_inner, A_t_inner);
            CondExpec_inner += discounted_payoff_inner;
        }
        CondExpec_inner /= (double) NbrMCsimulationDua
lInternal;
```

```
        }

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

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

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

        MaxVariable = MAX(MaxVariable, discounted_payoff-
    DoobMeyerMartingale);

        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(&AveragePaths);
    pnl_mat_free(&SpotPaths_inner);
    pnl_mat_free(&VarPaths_inner);
    pnl_mat_free(&RegressionCoeffMat);
    pnl_vect_free(&RegressionCoeffVect);

    return init_mc;

}
```

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

    double T, t_0, T_0;
    double r, divid, time_spent, pseudo_strike, true_strike
    , pseudo_spot;
    int return_value;

    Met->Par[3].Val.V_INT = MAX(2, Met->Par[3].Val.V_INT);
    // At least two exercise dates.

    r=log(1.+ptMod->R.Val.V_DOUBLE/100.);
    divid=log(1.+ptMod->Divid.Val.V_DOUBLE/100.);

    T= ptOpt->Maturity.Val.V_DATE;
    T_0 = ptMod->T.Val.V_DATE;
    t_0= (ptOpt->PathDep.Val.V_NUMFUNC_2)->Par[0].Val.V_PDO
    UBLE;
    time_spent= (T_0-t_0)/(T-t_0);

    if (T_0 < t_0)
    {
        Fprintf(TOSCREEN,"T_0 < t_0, untreated case{n{n{n")
    ;
        return_value = WRONG;
    }

    /* Case t_0 <= T_0 */
    else
    {
        pseudo_spot= (1.-time_spent)*ptMod->S0.Val.V_PDOUB
    LE;
        pseudo_strike= (ptOpt->PayOff.Val.V_NUMFUNC_2)->
    Par[0].Val.V_PDOUBLE-time_spent*(ptOpt->PathDep.Val.V_
    NUMFUNC_2)->Par[4].Val.V_PDOUBLE;
```

```
    true_strike= (ptOpt->PayOff.Val.V_NUMFUNC_2)->Par[0
].Val.V_PDOUBLE;

    (ptOpt->PayOff.Val.V_NUMFUNC_2)->Par[0].Val.V_PDOUB
LE= pseudo_strike;

    return_value =  MC_Am_Alfonsi_AnBr(pseudo_spot,
                                       T-T_0,
                                       r,
                                       divid,
                                       ptMod->Sigma0.
Val.V_PDOUBLE,
                                       ptMod->MeanReversion.h
al.V_PDOUBLE,
                                       ptMod->LongRunV
ariance.Val.V_PDOUBLE,
                                       ptMod->Sigma.Val
.V_PDOUBLE,
                                       ptMod->Rho.Val.
V_PDOUBLE,
                                       ptMod->Mean.Val.
V_PDOUBLE,
                                       ptMod->Variance.
Val.V_PDOUBLE,
                                       ptMod->Lambda.
Val.V_PDOUBLE,
                                       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_2,
                                        &(Met->Res[0].
    Val.V_DOUBLE));

        (ptOpt->PayOff.Val.V_NUMFUNC_2)->Par[0].Val.V_PDOUB
    LE=true_strike;
    }
    return return_value;
}

static int CHK_OPT(MC_Am_Asian_Alfonsi_AndersenBroadie_Bate
    s)(void *Opt, void *Mod)
{
    if ( (strcmp( ((Option*)Opt)->Name,"    AsianCallFixedAmer")==0) || (strcmp(
        return OK;

    return WRONG;
}
#endif //PremiaCurrentVersion

static int MET(Init)(PricingMethod *Met,Option *Opt)
{
    if ( Met->init == 0)
    {
        Met->init=1;
        Met->HelpFilenameHint = "    mc_am_asian_alfonsi_andersenbroadie_merhes"
        Met->Par[0].Val.V_LONG=50000;
        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_Asian_Alfonsi_AndersenBroadie_Bate
    s)=
{
    "MC_Am_Asian_Alfonsi_AndersenBroadie_Bates",
    {
        {"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_Asian_Alfonsi_AndersenBroadie_Bates),
    {{"Price",DOUBLE,{100},FORBID}, {" ",PREMIA_NULLTYPE,{0
    },FORBID}},
    CHK_OPT(MC_Am_Asian_Alfonsi_AndersenBroadie_Bates),
    CHK_ok,
    MET(Init)
};
```

References