

[Help](#)

```

#include "pnl/pnl_finance.h"
#include "math/equity_pricer/IMPLIED_BS.h"
#include "gridsparse_functions_varswap.h"
#include "varswap3d_std.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2009+2) //The "#els
static int CHK_OPT(FD_AchdouPommier)(void *Opt, void *Mod)
{
    return NONACTIVE;
}
int CALC(FD_AchdouPommier)(void *Opt, void *Mod, PricingMethod *Met)
{
    return AVAILABLE_IN_FULL_PREMIA;
}
#else

static int fd_achdoupommier_sparsegrid(VARSWAP3D_MOD *M, PricingMethod *Met)
{
    double price, delta;
    PnlVect *V;
    int N_T = Met->Par[0].Val.V_INT;
    int lev = Met->Par[1].Val.V_INT;
    SVSSparseOp *Op = svsparse_operator_create(lev, N_T, M);
    V = pnl_vect_create_from_zero(Op->G->size);
    GridSparse_Solve_svs(Op, V);
    svsparse_operator_free(&Op);
    price = M->Bond * GET(V, 1);
    price += pnl_bs_impli_call_put(M->is_call, M->V0, M->Bond, M->F0, M->Strike, M
    Met->Res[0].Val.V_DOUBLE = price; //price
    delta = GET(V, 0) / M->F0; // Normalisation due to change of variable spot/log
    delta += pnl_bs_impli_call_put_delta_forward(M->is_call, M->V0, M->Bond, M->F0
    delta *= exp((M->R - M->Divid) * M->T);
    Met->Res[1].Val.V_DOUBLE = delta;
    pnl_vect_free(&V);
    return OK;
}

```

```

int CALC(FD_AchdouPommier)(void *Opt, void *Mod, PricingMethod *Met)
{
    TYPEOPT *ptOpt = (TYPEOPT *)Opt;
    TYPEMOD *ptMod = (TYPEMOD *)Mod;
    double res;
    VARSWAP3D_MOD *M = svb_model_create_from_Model(ptMod);
    svb_model_initialise_from_Option(M, ptOpt);
    res = fd_achdoupommier_sparsegrid(M, Met);
    svb_model_free(&M);
    return res;
}

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

    return WRONG;
}

#endif //PremiaCurrentVersion
static int MET(Init)(PricingMethod *Met, Option *Opt)
{
    if (Met->init == 0)
    {
        Met->init = 1;
        Met->HelpFilenameHint = "fd_achdoupommier";
        Met->Par[0].Val.V_INT2 = 100;
        Met->Par[1].Val.V_INT2 = 7;
    }

    return OK;
}

PricingMethod MET(FD_AchdouPommier) =
{
    "FD_AchdouPommier",
    {"TimeStepNumber", INT2, {100}, ALLOW}, {"Level Grid (<10) ", INT2, {100}, AL
    CALC(FD_AchdouPommier),
    { {"Price", DOUBLE, {100}, FORBID}, {"Delta", DOUBLE, {100}, FORBID},

```

```
    {" ", PREMIA_NULLTYPE, {0}, FORBID}  
  },  
  CHK_OPT(FD_AchdouPommier),  
  CHK_ok,  
  MET(Init)  
};
```