

Help

```

#include <stdlib.h>
#include "rstemperedstable1d_std.h"
#include "math/wienerhopf_rs.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(AP_fastwhamerdig_rstemperedstable)(void
    *Opt, void *Mod)
{
    return NONACTIVE;
}
int CALC(AP_fastwhamerdig_rstemperedstable)(void*Opt,void *
    Mod,PricingMethod *Met)
{
    return AVAILABLE_IN_FULL_PREMIA;
}
#else

static char *infilename;

/*////////////////////////////////////////*/
static int wh_rstemperedstable_amerdigital(double Spot,
    double T, double h, double Strike1,
    double rebate,
    double er, long int step, int n_state,
    double *ptprice, double *ptdelta)
{
    PnlVect *divi, *rr, *lambdap, *num, *nup, *lambdam, *cm,
        *cp, *strike, *rebates, *mu, *qu;
    PnlVect *prices, *deltas;
    double eps;
    PnlMat *lam;
    int res, i, nstates;
    double tomega, omegas, lmnu, lpnu;
    int upordown=1;
    int ifCall=2;

    eps= 1.0e-7; // accuracy of iterations

```

```

res=readparamstsl_rs(&nstates, &rr, &divi, &num, &nup, &
    lambdam, &lambdap, &cm, &cp, &lam, infilename);

if(!res)
{
    printf("An error occurred while reading file!\n");
    *ptprice=0.;
    *ptdelta=0.;
    return OK;
}

mu= pnl_vect_create(nstates+1);
qu= pnl_vect_create(nstates+1);
strike= pnl_vect_create(nstates+1);
rebates= pnl_vect_create(nstates+1);
prices= pnl_vect_create(nstates+1);
deltas= pnl_vect_create(nstates+1);

for(i=0;i<nstates; i++) LET(strike,i)=Strike1;

if(ifCall==0) {omegas=2.0; }
else {omegas=-1.0;}

for(i=0;i<nstates;i++)
{
    LET(rr,i)=log(1.+GET(rr,i)/100.);
    LET(divi,i)=log(1.+GET(divi,i)/100.);
    LET(rebates,i)= rebate;

    if(ifCall==0)
    {
        tomega = GET(lambdam,i)<-2. ? 2. : (-GET(lambdam,
i)+1.)/2.;
        omegas = omegas>tomega ? tomega :omegas;
    }
    else
    {
        tomega=GET(lambdap,i)>1. ? -1. : -GET(lambdap,i)/2
.;
        omegas = omegas<tomega ? tomega :omegas;
    }
}

```

```

    LET(cp,i) = GET(cp,i) * tgamma(-GET(nup,i));
    LET(cm,i) = GET(cm,i) * tgamma(-GET(num,i));
    lpnu=exp(GET(nup,i)*log(GET(lambdap,i)));
    lmnu=exp(GET(num,i)*log(-GET(lambdam,i)));

    LET(mu,i)= GET(rr,i) - GET(divi,i) + GET(cp,i)*(lpnu-
    exp(GET(nup,i)*log(GET(lambdap,i)+1.0))) + GET(cm,i)*(lmnu-
    exp(GET(num,i)*log(-GET(lambdam,i)-1.0)));

    LET(qu,i) = GET(rr,i) + (pow(GET(lambdap,i),GET(nup,
    i)) - pow(GET(lambdap,i)+omegas,GET(nup,i)))*GET(cp,i) + (
    pow(-GET(lambdam,i),GET(num,i))-pow(-GET(lambdam,i)-omegas,
    GET(num,i)))*GET(cm,i);
}

res= fastwienerhopf_rs(1, nstates, mu, qu, omegas, 1, up
    ordown, ifCall, Spot, lambdam, lambdap,num, nup, cm, cp, rr
    , divi, lam,
    T, h, strike, Strike1, rebates, er, step, eps, prices,
    deltas);

//Price
*ptprice =GET(prices,n_state-1);
//Delta
*ptdelta =GET(deltas,n_state-1);

// Memory deallocation
pnl_vect_free(&mu);
pnl_vect_free(&qu);
pnl_vect_free(&prices);
pnl_vect_free(&deltas);
pnl_vect_free(&rr);
pnl_vect_free(&divi);
pnl_vect_free(&lambdap);
pnl_vect_free(&lambdam);
pnl_vect_free(&cp);
pnl_vect_free(&cm);
pnl_vect_free(&num);
pnl_vect_free(&nup);
pnl_vect_free(&strike);

```

```

    pnl_vect_free(&rebates);

    pnl_mat_free(&lam);

return OK;
}

//=====
=====
int CALC(AP_fastwhamerdig_rstemperedstable)(void *Opt,void
    *Mod,PricingMethod *Met)
{
    TYPEOPT* ptOpt=( TYPEOPT*)Opt;
    TYPEMOD* ptMod=( TYPEMOD*)Mod;
    double strike, spot,rebate;

    NumFunc_1 *p;
    int res;

    p=ptOpt->PayOff.Val.V_NUMFUNC_1;
    strike=p->Par[0].Val.V_DOUBLE;
    spot=ptMod->S0.Val.V_DOUBLE;

    rebate=p->Par[1].Val.V_DOUBLE;

    infilename= ptMod->Transition_probabilities.Val.V_FILENA
        ME;

    res = wh_rstemperedstable_amerdigital(spot,
        ptOpt->Maturity.Val.V_DATE-ptMod->T.Val.V_DATE,
        Met->Par[1].Val.V_DOUBLE, strike,rebate,
        Met->Par[0].Val.V_DOUBLE, Met->Par[2].Val.V_INT2
        , Met->Par[3].Val.V_INT,
        &(Met->Res[0].Val.V_DOUBLE), &(
        Met->Res[1].Val.V_DOUBLE));

    return res;
}

```

```

static int CHK_OPT(AP_fastwhamerdig_rstemperedstable)(void
    *Opt, void *Mod)
{
    // Option* ptOpt=(Option*)Opt;
    // TYPEOPT* opt=(TYPEOPT*)(ptOpt->TypeOpt);
    //return NONACTIVE;
    if ((strcmp( ((Option*)Opt)->Name,"DigitAmer")==0))
        return OK;

    return WRONG;
}

```

```

#endif //PremiaCurrentVersion
static int MET(Init)(PricingMethod *Met,Option *Opt)
{
    static int first=1;

    if (first)
    {
        Met->Par[0].Val.V_PDOUBLE=2.0;
        Met->Par[1].Val.V_PDOUBLE=0.01;
        Met->Par[2].Val.V_INT2=10;
        Met->Par[3].Val.V_INT=1;
        first=0;
    }

    return OK;
}

```

```

PricingMethod MET(AP_fastwhamerdig_rstemperedstable)=
{
    "AP_FastWHDig_RSTS",
    { {"Scale of logprice range", DOUBLE, {100}, ALLOW},
      {"Space Discretization Step",DOUBLE,{500},ALLOW},
      {"TimeStepNumber",INT2,{100},ALLOW},
      {"Output state number",INT,{100},ALLOW},
      {" ",PREMIA_NULLTYPE,{0},FORBID}},
    CALC(AP_fastwhamerdig_rstemperedstable),
    {"Price of chosen state",DOUBLE,{100},FORBID},
    {"Delta of chosen state",DOUBLE,{100},FORBID},

```

```
    {" ",PREMIA_NULLTYPE,{0},FORBID}},  
    CHK_OPT(AP_fastwhamerdig_rstemperedstable),  
    CHK_split,  
    MET(Init)  
};
```

References