

[Help](#)

```
#include "mer1d_std.h"

static int Put_Carr(double x, NumFunc_1 *p,
    double T, double r, double divid, double sigma, double lambda,
    double m, double v, double Nlimit, double step,
    double *ptprice, double *ptdelta)
{
    double u, v2, mv2, exmv2, mv, lambdaT, k, K, sigma2T, A,
        B, E, s1, s2, s;
    int i;

    K = p->Par[0].Val.V_DOUBLE;

    v2 = v/2.;
    mv2 = m + v2;
    exmv2 = exp(mv2);
    mv = m + v;
    lambdaT = lambda * T;
    k = log(K);
    sigma2T = sigma * sigma * T / 2;
    E = (r - divid) * T - sigma2T - lambdaT * (exmv2 - 1);
    A = (r - divid) * T + sigma2T - lambdaT * (exmv2 - 1);
    B = 1 / exp(lambdaT * exmv2);
    s = log(x);
    s1 = 0;
    s2 = 0;
    /*Computes  $\pi_1$  and  $\pi_2$  by descretizing
       the above integrals by
       the trapezoid method*/
    i = 0;
    while((double)i < ((double)Nlimit/step))
    {
        i++;
        u = (double)i * step;
        s1 = s1 + B * exp(lambdaT * exmv2 * exp(-u * u * v2)) * cos(
            u * mv2) - u * u * sigma2T * sin(u * A + lambdaT * exmv2 * exp(-u
            * u * v2)) * sin(u * mv) - u * (k - s)) / u;
        s2 = s2 + exp(-u * u * sigma2T + lambdaT * (cos(u * m) *
            exp(-u * u * v2) - 1)) * sin(u * (-k + E + s) + lambdaT * sin(u * m) *
            exp(-u * u * v2)) / u;
    }
}
```

```

    }

    *ptprice=-x*exp(-divid*T)*(0.5-0.5*step*(2*s1+
        A+lambdaT*exmv2*mv-k+s+B*exp(lambdaT*exmv2*exp(-
        Nlimit*Nlimit*v2)*cos(Nlimit*mv2)-Nlimit*Nlimit*
        sigma2T)*sin(Nlimit*A+lambdaT*exmv2*exp(-Nlimit*
        Nlimit*v2)*sin(Nlimit*mv)-Nlimit*k)/Nlimit)/PI)+K
        *exp(-r*T)*(0.5-0.5*step*(2*s2+E-k+s+lambdaT*m+
        exp(-Nlimit*Nlimit*sigma2T+lambdaT*(cos(Nlimit*m)*
        exp(-Nlimit*Nlimit*v2)-1))*sin(Nlimit*(-k+E)+lamb
        daT*sin(Nlimit*m)*exp(-Nlimit*Nlimit*v2))/Nlimit)
        /PI);

    *ptdelta =-exp(-divid*T)*(0.5-0.5*step*(2*s1+A+
        lambdaT*exmv2*mv-k+s+B*exp(lambdaT*exmv2*exp(-Nl
        imit*Nlimit*v2)*cos(Nlimit*mv2)-Nlimit*Nlimit*sig
        ma2T)*sin(Nlimit*A+lambdaT*exmv2*exp(-Nlimit*Nlim
        it*v2)*sin(Nlimit*mv)-Nlimit*k)/Nlimit)/PI);

    return OK;
}

int CALC(AP_Put_Carr)(void*Opt,void *Mod,Pricing
    Method *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 Put_Carr(ptMod->S0.Val.V_PDOUBLE,ptOpt->
        PayOff.Val.V_NUMFUNC_1,ptOpt->Maturity.Val.V_DA
        TE-ptMod->T.Val.V_DATE,r,divid,ptMod->Sigma.Val.V_
        PDOUBLE,ptMod->Lambda.Val.V_PDOUBLE,ptMod->Mean.
        Val.V_PDOUBLE,ptMod->Variance.Val.V_PDOUBLE, Met->
        Par[0].Val.V_PDOUBLE,Met->Par[1].Val.V_DOUBLE,&(
        Met->Res[0].Val.V_DOUBLE),&(Met->Res[1].Val.V_
        DOUBLE));
}

```

```

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

  return strcmp( ((Option*)Opt)->Name,"PutEuro");
}

static int MET(Init)(PricingMethod *Met)
{
  static int first=1;

  if (first)
  {
    Met->Par[0].Val.V_PDOUBLE=1000.;
    Met->Par[1].Val.V_DOUBLE= 0.01;
    first=0;
  }
  return OK;
}

PricingMethod MET(AP_Put_Carr)=
{
  "AP_Carr",
  {"N",DOUBLE,100,ALLOW},
  {"Step",DOUBLE,100,ALLOW},
  {" ",END,0,FORBID}},
  CALC(AP_Put_Carr),
  {"Price",DOUBLE,100,FORBID},{"Delta",DOUBLE,10
    0,FORBID},{" ",END,0,FORBID}},
  CHK_OPT(AP_Put_Carr),
  CHK_ok,
  MET(Init)
} ;

```

## References