```
   Help
#include "mer1d_std.h"

static int Put_Merton(double x,NumFunc_1  *p,
    double T,double r,double divid,double sigma,double la
    mbda,double m,double v,double *ptprice,double *pt
    delta)
{

double lambdaT,mv2,exmv2,EU,mu,M,sigma02,sigmasq
    rt,price,delta,test,test2,puissancen1,factorieln,
    n,d1,d2,sigma2,rT,muT,ex_r_lT,K;


  lambdaT=lambda*T;
  K=p->Par[0].Val.V_DOUBLE;
  mv2=m+v/2.;
  exmv2=exp(mv2);
  EU=exmv2-1;
  mu=r-divid-lambda*EU;
  rT=r*T;
  muT=mu*T;
  M=exp(T*(-divid-lambda*exmv2));
  sigma02=sigma*sigma;
  sigmasqrt=sigma*sqrt(T);
  d1=(log(x/K)+sigma02*T/2+muT)/sigmasqrt;
  d2=d1-sigmasqrt;
  price=K*exp(-rT-lambdaT)*N(-d2)-x*M*N(-d1);
  puissancen1=1.;
  factorieln=1.;
  delta=-M*N(-d1);
  ex_r_lT=exp(-rT-lambdaT);
  test=exp(-lambdaT);
  puissancen1=1.;
  factorieln=1.;
  n=0;
  test2=1;

  while (test<0.99999)
  {n++;
```

```
  factorieln*=n;/*  n!   */
  puissancen1*=lambdaT;/* (lambda*T)^n */
  sigma2=sigma02+v*(double)n/T;
  sigmasqrt=sqrt(sigma2*T);
  d1=(log(x/K)+sigma2*T/2+n*(mv2)+muT)/sigmasqrt
    ;
  d2=d1-sigmasqrt;
  price+=(puissancen1/factorieln)*(K*ex_r_lT*N(-
    d2)-(x*exp(n*mv2)*M*N(-d1)));
  test+=exp(-lambdaT)*puissancen1/factorieln;
  delta+=-(puissancen1/factorieln)*exp(n*mv2)*M*
    N(-d1);
  }

   *ptprice=price;
   *ptdelta=delta;

   return OK;

}

int CALC(CF_Put_Merton)(void*Opt,void *Mod,Pric
    ingMethod *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_Merton(ptMod->S0.Val.V_PDOUBLE,pt
    Opt->PayOff.Val.V_NUMFUNC_1,ptOpt->Maturity.Val.V_
    DATE-ptMod->T.Val.V_DATE,
          r,divid,ptMod->Sigma.Val.V_PDOUBLE,pt
    Mod->Lambda.Val.V_PDOUBLE,ptMod->Mean.Val.V_PDOUBL
    E,ptMod->Variance.Val.V_PDOUBLE,&(Met->Res[0].Val
    .V_DOUBLE),&(Met->Res[1].Val.V_DOUBLE));
}

int CHK_OPT(CF_Put_Merton)(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)
{
  return OK;
}

PricingMethod MET(CF_Put_Merton)=
{
  "CF_Merton",
  {{" ",END,0,FORBID}},
  CALC(CF_Put_Merton),
  {{"Price",DOUBLE,100,FORBID},{"Delta",DOUBLE,10
    0,FORBID},{" ",END,0,FORBID}},
  CHK_OPT(CF_Put_Merton),
  CHK_ok,
  MET(Init)
} ;
```

References