

## Help

```

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2008+2) //The "#els
#else

#ifndef functions_h
#define functions_h

#include "common.h"
#include<stdlib.h>
#include<iostream>
#include <cmath>
#include <vector>

using std::cout;
using std::endl;

//psi=P[ZT>=a,maxZs>=b s dans[0 T]] ou Z levy poisson compose long double expone
long double psiB(long double *x, const long double &T);
//fonction psi=P[ZT>=a] ou Z levy poisson compose long double exponentielle
long double psiVN(long double *, int nb = 20);

//derivi&de psi=P[ZT>=a] % i&a
long double dpsiVN(long double *, int nb = 20);
std::vector<long double> psiVNB(long double *, int nb = 20);
//prix du put loockback floating strike
long double PLB(long double *, const long double &);
//prix du call loockback floating strike
long double CLB(long double *, const long double &);
//delta du put loockback floating strike
long double dPLB(long double *, const long double &);
//delta du call loockback floating strike
long double dCLB(long double *, const long double &);

long double psiM(long double *, const long double &);
long double rebateproba(long double *, const long double &, const long double &);
long double psiMA(long double *, const long double &);
long double psiMB(long double *, const long double &);
class function
{
public:

```

```

    long double *param;
    function();
    function(long double *);
    virtual ~function() {};
    virtual long double f(const long double &)const
    {
        return 0;
    };
    virtual long double Df(const long double &)const
    {
        return 0;
    };
};
inline function::function()
{
    param = NULL;
};
inline function::function(long double *v)
{
    param = v;
};
//algorithme d'inversion de la transform  e de Laplace
class inverselaplace
{
public:
    function *LF;
    inverselaplace() {};
    inverselaplace(function &F)
    {
        LF = &F;
    };
    long double InvLF(const long double &T)const;
};
class solver
{
public:
    function *F;
    solver() {};
    solver(function &G)
    {
        F = &G;
    };
};

```

```
};
virtual ~ solver() {};
virtual std::vector<long double> racine()const
{
    return std::vector<long double>();
};
};

class newton : public solver
{
public:
    long double start;
    long double accuracy;
    newton(function &G, long double x0, long double eps): solver(G)
    {
        accuracy = eps;
        start = x0;
    };
    virtual std::vector<long double> racine()const;
};

class secante : public solver
{
public:
    long double start0;
    long double start1;
    long double accuracy;
    secante(function &G, long double x0, long double x1, long double eps): solver(G)
    {
        accuracy = eps;
        start0 = x0;
        start1 = x1;
    };
    virtual std::vector<long double> racine()const;
};

class dichotomie : public solver
{
public:
    long double a;
    long double b;
    long double accuracy;
    dichotomie(function &G, long double x0, long double x1, long double eps): solver(G)
```

```

{
    accuracy = eps;
    a = x0;
    b = x1;
};
virtual std::vector<long double> racine()const;
};
//fonction exposant caracteristique d'un levy poisson composé;double exponentiel
class KCE : public function
{
public:
    KCE(long double *param): function(param) {};
    virtual long double f(const long double &x)const;
    virtual long double Df(const long double &x)const;
};
class LPLB : public function
{
public:
    LPLB(long double *param): function(param) {};
    //Transformé de Laplace du prix du put lookback floating strike
    virtual long double f(const long double &alpha) const;
};
class DLPLB : public function
{
public:
    DLPLB(long double *param): function(param) {};
    //Transformé de Laplace de la dérivée % i de S0 du prix du put lookback floating strike
    virtual long double f(const long double &alpha) const;
};

class LCLB : public function
{
public:
    LCLB(long double *param): function(param) {};
    //Transformé de Laplace du prix du call lookback floating strike
    virtual long double f(const long double &alpha) const;
};
class DLCLB : public function
{
public:
    DLCLB(long double *param): function(param) {};
};

```

```

//Transformi de Laplace de la derivi de % i de S0 du prix du call loockback float
    virtual long double f(const long double &alpha) const;
};

class LPsiM : public function
{
public:
    LPsiM(long double *param): function(param) {};
//transformi de Laplace de psi=P[maxZs>=b s dans[0 T]] ou Z levy poisson compo
    virtual long double f(const long double &alpha)const;
};
class LdPsiM : public function
{
public:
    LdPsiM(long double *param): function(param) {};
    virtual long double f(const long double &alpha)const;
};
class LPsiMA : public function
{
public:
    LPsiMA(long double *param): function(param) {};
    virtual long double f(const long double &alpha)const;
};
class LPsiMB : public function
{
public:
    LPsiMB(long double *param): function(param) {};
    virtual long double f(const long double &alpha)const;
};
class amer_eq : public function
{
public:
    amer_eq(long double *param): function(param) {};
//fonction dont la racine correspond au prix critique du put americain a horizon
    virtual long double f(const long double &v) const;
};

#endif

#endif //PremiaCurrentVersion

```