Help

```
extern "C" {
#include  "
href../../../mod/bs1d/bs1d_stdz/bs1d_stdz_h_src.pdfbs1d_stdz.h"
#include "
href../../../common/enums_h_src.pdfenums.h"
#include "
href../../../common/error_msg_h_src.pdferror_msg.h"
}
#include "pnl/pnl_integration.h"
#include "pnl/pnl_cdf.h"
#include "pnl/pnl_root.h"

#if defined(PremiaCurrentVersion) && PremiaCurrentVersion < (2018+2) //The "#els
extern "C" {
static int CHK_OPT(AP_LOGNORMAL_RISK_GMMB_BS)(void *Opt, void *Mod)
{
  return NONACTIVE;
}
int CALC(AP_LOGNORMAL_RISK_GMMB_BS)(void *Opt, void *Mod, PricingMethod *Met)
{
  return AVAILABLE_IN_FULL_PREMIA;
}
}
#else

static double sigma, maturity, A0, alpha, r,divid,me, mu, m, risk_level, rho, C,


static double Normal(double d)
{

  int which;
  double p;
  double q;
  double mean;
  double sd;
  int status;
  double bound;
```

```
  which=1;
  mean=0.0;
  sd=1.0;

  pnl_cdf_nor(&which,&p,&q,&d,&mean,&sd,&status,&bound);

  return p;
}

static double p(double z)
{
return exp(-pow((pow(sigma, 2)*maturity / 2 + log(z)), 2) / (2 * pow(sigma, 2)*m
}

static double q(double z)
{
return exp(-pow((pow(sigma, 2)*maturity + log(z)), 2) / (2 * pow(sigma, 2)*matur
}

static double quantile1(double z)
{
return log(z) / sqrt(pow(sigma, 2)*maturity) + 0.5*sqrt(pow(sigma, 2)*maturity);
}

static double quantile2(double z)
{
return log(z) / sqrt(pow(sigma, 2)*maturity) - 0.5*sqrt(pow(sigma, 2)*maturity);
}

static double quantile3(double z)
{
return log(z) / sqrt(pow(sigma, 2)*maturity) + sqrt(pow(sigma, 2)*maturity);
}

static double quantile4(double z)
{
return log(z) / sqrt(pow(sigma, 2)*maturity) - sqrt(pow(sigma, 2)*maturity);
}

static double aT(double z)
{
```

```
return (Normal(quantile1(z)) - Normal(quantile2(z))) / (p(z)*pow(sigma, 2));
}

static double bT(double z)
{
return (Normal(quantile3(z)) - Normal(quantile4(z))) / (q(z)*pow(sigma, 2));
}

static double square_sigma(double z)
{
return log(2 * (bT(z) / aT(z) - 1 - z) / (aT(z)*pow(sigma, 2))) / maturity;
}

static double mean_mu(double z)
{
return 1 - 2 * log(aT(z)) / (maturity*square_sigma(z));
}

static double B(double V)
{
  double res = (exp(-r*maturity)*(alpha*A0) - V) / (A0);
return res;
}

static double inte_f(double V, double z)
{
double percentile = (log((B(V) - z) / me) + mean_mu(z)*square_sigma(z)*maturity

return Normal(percentile);
}

static double inte_f2(double w, double z)
{
double percentile = (log((w - z) / me) + mean_mu(z)*square_sigma(z)*maturity / 2

return Normal(percentile);
}

static double DPhi(double z)
{
        double temp = log(z) - (mu - m - r)*maturity;
```

```
double s = -pow(temp, 2) / (2 * pow(sigma, 2) * maturity);
        return exp(s) / (sqrt(2 * M_PI*maturity)*sigma*z);
}

static double Q(double V, double z)
{
return inte_f(V, z)*DPhi(z);
}

static double Q2(double w, double z)
{
return inte_f2(w, z)*DPhi(z);
}

static double integral_Q2(double w)
{
double sum = 0, z;
double end = w;
double res;
for (z = 0; z<= end; z = z + 0.005)
{
res = Q2(w, z);
if (isnan(res)) continue;
sum = sum + res*0.005;
}
return sum;
}

static double inte_f_(double V, double z)
{
        double c1=(mean_mu(z)-1)*square_sigma(z)*maturity;
        double multi=exp(-c1/2);
        double percentile1 = (log((B(V) - z) / me) + (mean_mu(z)-2)*square_sigma
        return multi*Normal(percentile1);
}

static double Q_(double V,double z)
{
        return (z*inte_f(V,z)+me*inte_f_(V,z))*DPhi(z);
}
```

```
static double  xi(double w)
{
   return 1-tPx*integral_Q2(w);
}

static double c1(double z)
{
    return mean_mu(z)*square_sigma(z)*maturity / 2;
}

static double c2(double z)
{
    return sqrt(square_sigma(z)*maturity);
}

static double Int_f1(double V,double z)
{
    double percentile1=(log(((exp(-r*maturity)*(A0*alpha)-V)/(A0) - z) / me) + c
    if (square_sigma(z)>0)
    {
        return Normal(percentile1);
    }
    else return 0;
}

static double DP(double V, double z)
{
    double temp = log(z) - (mu - m - r)*maturity;
    double s = -pow(temp, 2) / (2 * pow(sigma, 2) * maturity);
    return exp(s) / (sqrt(2 * M_PI*maturity)*sigma*z);
}

static double function1(double u, void *p)
{
    double *tmp;
    tmp = (double *)p;
    return Int_f1(*tmp, u)*DP(*tmp, u);
}

static double Int_Q1(double V)
{
```

```
    double h, result,abserr;
    int neval;
    PnlFunc func;
    func.F = function1;
    func.params = &V;
    h=(exp(-r*maturity)*(A0*alpha)-V)/(A0);
    if (h<=0)
    {
        return 0;
    }
    else
    {
        pnl_integration_qag(&func,0.0,h,0.00001,0.0001,0,&result,&abserr,&neval)
        return result;
    }
}

static double Int_f2(double V,double z)
{
    double percentile2=(log((rho*z-(exp(-r*maturity)*rho*(A0*alpha)+V)/(A0)) / m
    return Normal(percentile2);
}

static double function2(double u, void *p)
{
    double *tmp;
    tmp = (double *)p;
    return Int_f2(*tmp, u)*DP(*tmp, u);
}

static double Int_Q2(double V)
{
    double l, h, result,abserr;
    int neval;
    PnlFunc func;
    func.F = function2;
    func.params = &V;
    l= exp(-r*maturity)*(A0*alpha)/(A0)+V/(rho*(A0));
    h= exp(-r*maturity)*(rho*(A0*alpha)+C)/(rho*(A0));
    pnl_integration_qag(&func,l,h,0.0000001,0.0001,0,&result,&abserr,&neval);
    return result;
```

```
}

static double Int_f3(double V,double z)
{
    double percentile3=(log((exp(-r*maturity)*C-V)/(A0)/me) + c1(z)) / c2(z);
    return Normal(percentile3);
}

static double function3(double u, void *p)
{
    double *tmp;
    tmp = (double *)p;
    if (u>=1200) {
        return 0;
    }
    else
    {
        return Int_f3(*tmp, u)*DP(*tmp, u);
    }
}

static double Int_Q3(double V)
{
    double l, result,abserr;
    int neval;
    PnlFunc func;
    func.F = function3;
    func.params = &V;
    l= exp(-r*maturity)*(rho*(A0*alpha)+C)/(rho*(A0));
    pnl_integration_qag(&func,l, PNL_POSINF, 0.001,0.01,0,&result,&abserr,&neval
    return result;
}

static double Int_Q(double V)
{
  if (rho>0) {return Int_Q1(V)+Int_Q2(V)+Int_Q3(V);}
  else {return Int_Q1(V);}
}

static double integral_Q(double x, void *p)
{
```

```
    return Int_Q(x) - (1 - alpha2) / tPx;
}

static void bisection(double *var)
{
    double x1, x2, r, tol;
    PnlFunc func;

    x1 = -1.;
    x2 = 100.;
    tol= 0.00001;
    func.F = integral_Q;
    func.params = NULL;
    r=pnl_root_brent(&func, x1, x2, &tol);
    *var =r;
}

static double g1(double V, double z)
{
    double percentile1;
    double multi=exp(-(mean_mu(z)-1)*square_sigma(z)*maturity/2.);
    if (square_sigma(z)>0)
    {
        percentile1 = (log((B(V) - z) / me) + (mean_mu(z)-2)*square_sigma(z)*mat
        return multi*Normal(percentile1);
    }
    else return 0;
}

static double z1(double u, void *p)
{
    double *tmp;
    tmp = (double *)p;
    return ((exp(-r*maturity)*(A0*alpha)-(A0)*u)*Int_f1(*tmp, u)- (A0)*me*g1(*tm
}

static double Int_Z1(double V)
{
    double h, result,abserr;
    int neval;
    PnlFunc func;
```

```
    func.F = z1;
    func.params = &V;
    h= (exp(-r*maturity)*(A0*alpha)-V)/(A0);
    if (h<=0)
    {
        return 0;
    }
    else
    {
        pnl_integration_qag(&func,0.0,h,0.00001,0.0001,0,&result,&abserr,&neval)
        return result;
    }
    return result;
}

static double g2(double V, double z)
{
    double multi=exp(-(mean_mu(z)-1)*square_sigma(z)*maturity/2);
    double percentile1 = (log((rho*z-(exp(-r*maturity)*rho*(A0*alpha)+V)/(A0)) /
    return multi*Normal(percentile1);
}

static double z2(double u, void *p)
{
    double *tmp;
    tmp = (double *)p;
    return (rho*((A0)*u-exp(-r*maturity)*(A0*alpha))*Int_f2(*tmp, u)- (A0)*me*g2
}

static double Int_Z2(double V)
{
    double l, h, result,abserr;
    int neval;
    PnlFunc func;
    func.F = z2;
    func.params = &V;
    l = exp(-r*maturity)*(A0*alpha)/(A0)+V/(rho*(A0));
    h = exp(-r*maturity)*(rho*(A0*alpha)+C)/(rho*(A0));
    pnl_integration_qag(&func,l, h, 0.001,0.01,0,&result,&abserr,&neval);
    return result;
}
```

```
double g3(double V, double z)
{
    double multi=exp(-(mean_mu(z)-1)*square_sigma(z)*maturity/2);
    double percentile1 = (log((exp(-r*maturity)*C-V)/(A0)/me) + (mean_mu(z)-2)*s
    return multi*Normal(percentile1);
}

static double z3(double u, void *p)
{
    double *tmp;
    tmp = (double *)p;
    if (u>=1200) {
        return 0;
    }
    else
    {
        return ((exp(-r*maturity)*C)*Int_f3(*tmp, u)- (A0)*me*g3(*tmp, u))*DP(*t
    }
}

static double Int_Z3(double V)
{
    double l, result,abserr;
    int neval;
    PnlFunc func;
    func.F = z3;
    func.params = &V;
    l=exp(-r*maturity)*(rho*(A0*alpha)+C)/(rho*(A0));
    pnl_integration_qag(&func,l, PNL_POSINF, 0.001,0.01,0,&result,&abserr,&neval
    return result;
}

static double CTE(double V)
{
  if (rho>0) {return tPx*(Int_Z1(V)+ Int_Z2(V)+Int_Z3(V))/ (1 - alpha2);}
  else {return tPx*Int_Z1(V)/ (1 - alpha2);}
}

static double CTE2(double V)
{
        return CTE(V)*(1-alpha2)/(1-risk_level);
```

```
}

int AP_GMMB_AE_Lognormal_VaR_CTE(double A0, double alpha, double maturity, doubl
{

 tPx = 0.757;

 alpha2 = xi(exp(-r*maturity)*(alpha*A0)/(A0));

 if (risk_level > alpha2) alpha2=risk_level;

double VaR;

        bisection(&VaR);

        *ptvar=VaR;

        if(risk_level == alpha2) *ptcte=CTE(VaR);

        if(risk_level < alpha2) *ptcte=CTE2(VaR);

return 0;

}

extern "C" {

int  AP_LOGNORMAL_RISK_GMMB_BS(double A0_main, double maturity_main, double r_ma
{
  double var,cte;


 A0=A0_main;
 maturity= maturity_main;
 r=r_main;
 divid=divid_main;
 sigma=sigma_main;
 mu=mu_main-divid;
 me=me_main;
 m=m_main;
 rho=rho_main;
```

```
 risk_level=risk_level_main;
 C_main*=A0_main;
 C=C_main;
 alpha=alpha_main;

  AP_GMMB_AE_Lognormal_VaR_CTE(A0_main,alpha_main,maturity_main,r_main,sigma_mai

  *ptvar=var;
  *ptcte=cte;

  return OK;
}

int CALC(AP_LOGNORMAL_RISK_GMMB_BS)(void *Opt, void *Mod, PricingMethod *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 AP_LOGNORMAL_RISK_GMMB_BS(ptMod->S0.Val.V_PDOUBLE,ptOpt->Maturity.Val.V
}

static int CHK_OPT(AP_LOGNORMAL_RISK_GMMB_BS)(void *Opt, void *Mod)
{
  if ((strcmp(((Option *)Opt)->Name, "GMMB_RISK") == 0))
    return OK;
  else
    return WRONG;
}
}
#endif //PremiaCurrentVersion
extern "C" {
static int MET(Init)(PricingMethod *Met, Option *Opt)
{
  if (Met->init == 0)
    {
      Met->init = 1;
```

```
    }

  return OK;
}

PricingMethod MET(AP_LOGNORMAL_RISK_GMMB_BS) =
{
  "AP_LOGNORMAL_RISK_GMMB_BS",
  {
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CALC(AP_LOGNORMAL_RISK_GMMB_BS),
  { {"VaR", DOUBLE, {100}, FORBID},
  {"CTE", DOUBLE, {100}, FORBID},
    {" ", PREMIA_NULLTYPE, {0}, FORBID}
  },
  CHK_OPT(AP_LOGNORMAL_RISK_GMMB_BS),
  CHK_ok,
  MET(Init)
};
}
```