```
   Help
#ifndef _BLACK_H
#define _BLACK_H

int Init_BS(int BS_Dimension, double *BS_Volatility,
            double *BS_Correlation, double BS_Interest_Ra
    te,
            double *BS_Dividend_Rate);

void End_BS();

void Backward_Path(double *Paths, double *Brownian_Bridge,
                   double *BS_Spot,
                   double Time, long MonteCarlo_Iterations,
                   int BS_Dimension);

void BS_Forward_Path(double *Paths, double *Brownian_Paths,
     double *BS_Spot, double Time,
                    long MonteCarlo_Iterations, int BS_Dim
    ension);


void ForwardPath(double *Path, double *Initial_Stock, int
    Initial_Time,
                 int Number_Dates, int BS_Dimension,
    double Step, double Sqrt_Step,
                 int generator);

void Init_Brownian_Bridge(double *Brownian_Bridge,
                          long MonteCarlo_Iteration,
                          int BS_Dimension, double BS_Matu
    rity, int generator);

void Compute_Brownian_Bridge(double *Brownian_Bridge,
    double Time,
                             double step, int BS_Dimension,
     long MonteCarlo_Iterations,
                             int generator);

void Init_Brownian_Bridge_A(double *Brownian_Bridge, long
    MonteCarlo_Iterations,
```

```
                              int BS_Dimension, double BS_
    Maturity, int generator);

void Compute_Brownian_Bridge_A(double *Brownian_Bridge,
    double Time, double Step,
                                int BS_Dimension, long
    MonteCarlo_Iterations,
                                int generator);

void Compute_Inv_Sqrt_BS_Dispersion(double time, int BS_Dim
    ension, const PnlVect *BS_Spot,
                                    double BS_Interest_Ra
    te, const PnlVect *BS_Dividend_Rate);

void NormalisedPaths(double *Paths, double *PathsN, long
    MonteCarlo_Iterations,
                     int BS_Dimension);

double Discount(double Time, double BS_Interest_Rate);

int BS_Transition_Allocation(int BS_Dimension, double Step)
    ;

void BS_Transition_Liberation();

double BS_TD(double *X, double *Z, int BS_Dimension,
    double Step);

void BS_Forward_Step(double *Stock, double *Initial_Stock,
    int BS_Dimension,
                     double Step, double Sqrt_Step,   int     generator);

void BlackScholes_Transformation(double Instant, double *
    BS, double *B,
                                 int BS_Dimension, double *
    BS_Spot);

void gauss_stock(double *normalvect, int N, int generator);

void RMsigma(double *sigma, int BS_Dimension);
void InitThetasigma(double *theta, double *thetasigma, int
```

```
    BS_Dimension);
void ThetaDriftedPaths(double *Paths, double *thetasigma,
    double Time,
                       long AL_MonteCarlo_Iterations, int
    BS_Dimension);

void BS_Forward_Path(double *Paths, double *Brownian_Paths,
     double *BS_Spot, double Time,
                     long MonteCarlo_Iterations, int BS_Dim
    ension);
double European_call_price_average(PnlVect *BS_Spot,
    double Time, double OP_Maturity,
                                    double Strike, int BS_
    Dimension, double BS_Interest_Rate,
                                    PnlVect *BS_Dividend_Ra
    te);
double European_call_put_geometric_mean(PnlVect *BS_Spot,
    double Time, double OP_Maturity,
                                         double Strike, int
    BS_Dimension,
                                         double BS_Interest_
    Rate, PnlVect *BS_Dividend_Rate,
                                         double *BS_
    Volatility, double *BS_Correlation,
                                         int iscall);
void Compute_Brownian_Paths(double *Brownian_Paths, double
    Sqrt_Time,
                            int BS_Dimension, long
    MonteCarlo_Iterations,
                            int generator);
void Compute_Brownian_Paths_A(double *Brownian_Paths,
    double Sqrt_Time,
                              int BS_Dimension, long
    MonteCarlo_Iterations,
                              int generator);

#endif
```

References