Computational Finance - Mini Task 2

10134621

March 8th 2021

1 Calculate h

$R_{r,t,T}$ is normally distributed with mean $f(r,t,T)$ and variance $v^2(t,T)$,

$$R_{r,t,T} \sim N(f(r,t,T), v^2(t,T)). \tag{1}$$

$N(h)$ is the Cumulative Normal Distribution, or the probability that $x < h$ if $x \sim N(0,1)$, i.e

$$P(x < h). \tag{2}$$

Require $N(h)$ to be equivalent to $P(R_{r,t,T} < X_r)$ which we transform to the standard Normal Distribution and thus

$$P(z < \frac{X_r - f(r,t,T)}{\sqrt{v(r,T)}}), \tag{3}$$

where $z \sim N(0,1)$.

Comparing equation (2) and equation (3) we obtain an expression for h,

$$h = \frac{X_r - f(r,t,T)}{\sqrt{v^2(t,T)}} \tag{4}$$

2 Option value for $\mathbf{r}_0$

Defining the following parameters,

$$\begin{aligned} r_0 &= 0.0263, \\ T &= 3, \\ X_r &= 0.05, \\ \kappa &= 0.0957, \\ \theta &= 0.051, \\ \sigma &= 0.0221, \end{aligned}$$

we calculate the value of the financial contract at time $t = 0$ to be

$$V(r_0, t = 0, T) = 0.819304. \tag{5}$$

3 Option value for multiple r

Taking approximately 100 different values of r in the range $r \in [0, 0.2]$ we make plots of $P(r, t = 0, T)$ and $V(r, t = 0, T)$ as shown in Figure 1.

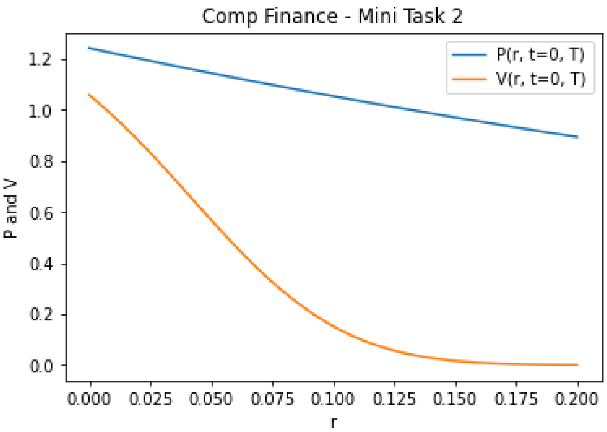

Figure 1: Plots of $P(r, t = 0, T)$ and $V(r, t = 0, T)$ with r for our interest rate derivative contract using a non-standard model.

Listing 1: C++ code for calculating put option value for an interest rate derivative contract using a non-standard model. The data is written to a file called "data.csv" in the working directory.

```
// Header
// Title: Comp finance - Mini task 2
// Student ID: 10134621
// Date Created: 03/03/21
// Last Edited: 03/03/21


#define _USE_MATH_DEFINES_


// Includes
#include <iostream>
#include <iomanip>
#include <cmath>
#include <math.h>
#include <fstream>
#include <vector>
#include <constants.h>  // header file for constants


```

```
// Decalre Functions

// calculate cummulative normal distribution
double norm_cum(const double& x);

// calculate f
double f(const double& r, const double& t, const double& T);

// calculate m
double m(const double& r, const double& t, const double& T);

// calculate q
double q(const double& t, const double& T);

// calculate v^2
double v2(const double& t, const double& T);

// calculate P
double P(const double& r, const double& t, const double& T);

// calculate n
double n(const double& r, const double& t, const double& T);

// calculate k
double k2(const double& t, const double& T);

// calculate V for put
double V_put(const double& r, const double& t, const double& T, const double& h);



// Begin main program
int main()
{
    // define variables
    const double t{ 0 };
    const double T{ 3 };
    double b = 0.2;  // lower r limit
    double a = 0;  // upper r limit
    double n = 100;  // number of calculations

    // open a file stream for writing
    std::ofstream output;

    // open the csv file
    output.open("data.csv");

    // if the file is open
    if (output.is_open()) {

        // for loop over r values
        for (double r{ 0 }; r <= b + 0.002; r += (b - a) / n) {

            // calculate h
            double h_val = (constants::X_r - f(r, t, T)) / pow(v2(t, T), 0.5);

            // calculate P
            double P_val = P(r, t, T);

            // calculate V(r, t=0, T) for a put option
            double V_val = V_put(r, t, T, h_val);

            // write data to file
            output << r << "," << P_val << "," << V_val << std::endl;

        }

        // close the file
        std::cout << "File write successful" << std::endl;
        output.close();

    }
    // if file could not be opened
    else {
        std::cout << "Error: could not open file" << std::endl;
        return 1;
    }

    return 0;
}  // End main program


```

```
// Define functions

// calculate V for put
double V_put(const double& r, const double& t, const double& T, const double& h)
{
    return P(r, t, T) * norm_cum(h);
}

// calculate cummulative normal distribution
double norm_cum(const double& x)
{
    return 0.5 * erfc(-x / pow(2, 0.5));
}

// calculate f
double f(const double& r, const double& t, const double& T)
{
    return m(r, t, T) - 0.5 * q(t, T);
}

// calculate m
double m(const double& r, const double& t, const double& T)
{
    return exp(-constants::kappa*(T-t))*r+(1-exp(-constants::kappa*(T-t)))*constants::theta;
}

// calculate q
double q(const double& t, const double& T)
{
    return (pow(constants::sigma, 2) / (3 * pow(constants::kappa, 2)))* pow(1 - exp(-constants::kappa * (T - t)), 5);
}

// calculate v^2
double v2(const double& t, const double& T)
{
    return (pow(constants::sigma, 2) / constants::kappa) * (1 - exp(-constants::kappa * (T - t)));
}

// calculate P
```

```cpp
double P(const double& r, const double& t, const double& T)
{
    return exp((2. / 3.) * k2(t, T) - (1. / 4.) * n(r, t, T));
}

// calculate n
double n(const double& r, const double& t, const double& T)
{
    return r * (T - t) - ((constants::theta - r) / (2 * constants::kappa)) * (1 - exp(-4 * constants::kappa * (T - t)));
}

// calculate k
double k2(const double& t, const double& T)
{
    return ((pow(constants::sigma, 2)) / (2 * pow(constants::kappa, 3))) * (5 * exp(-constants::kappa * (T - t)) - 3 * exp(-2 * constants::kappa * (T - t))
        + 3 * constants::kappa * (T - t) - 2);
}
```

Listing 2: Header file for constants.

```cpp
#pragma once
// Header file for constants

namespace constants
{
    // define variables
    const double r_0{ 0.0263 };
    const double X_r{ 0.05 };
    const double kappa{ 0.0957 };
    const double theta{ 0.051 };
    const double sigma{ 0.0221 };
}
```