MAP 5611 Intro to Computational Finance HW4

Student: Sen Zhang

I. Executive Summary

In this report, the routine of adaptive quadrature works correctly through testing four kinds of integrals. The value of Gaussian probability function at 1.0 is 0.84134475, which has at least 6 significant digits. Using Black-Scholes model for a European call, the value of the call is 98.567766, which has at least 3 significant digits. And the implied volatility is 0.076088285, which has at least 4 significant digits.

II. Statement of Problem

The Gaussian probability function

$$P(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} dt$$

has no analytical form by which it can be evaluated exactly.

Show that how to compute this function at the desired values of x.

1. Write an adaptive quadrature function that uses Simpson's rule. Test the routine on some intervals to show it working correctly.
2. Find the value of P for x = 1.0, accurate to at least 6 significant digits.
3. The Black-Scholes model for a European call under the assumption of constant volatility and interest rates is

$$C(S,t) = SP(d_1) - Ke^{-r(T-t)}P(d_2)$$

where

$$d_1 = \frac{\log(S/K) + (r + \sigma^2/2)(T-t)}{\sigma\sqrt{T-t}}$$

$$d_2 = \frac{\log(S/K) + (r - \sigma^2/2)(T-t)}{\sigma\sqrt{T-t}}$$

   a) Compute the value of the European call to at least 3 significant digits. Given

T	t	r	σ	K	S
54/365	0	0.0675	0.135	3425.0	3441.0

b) Compute the implied volatility to at least 4 significant digits.
Given

C	T	t	r	K	S
94.0	54/365	0	0.0675	3425.0	3441.0

III. Description of The Mathematics

The idea of Simpson's rule is to approximate the function using Newton Interpolation and integrate the interplant.

For a integral

$$I = \int_a^b f(x)dx$$

Set

$$\Delta x = \frac{b-a}{2}$$

Approximate the f(x) as

$$P_2(x) = f(x_0) + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1)$$

Then integrate $P_2(x)$

$$\tilde{I} = 2\Delta x f(x_0) + 2 f[x_0, x_1]\Delta x^2 + \frac{2}{3} f[x_0, x_1, x_2]\Delta x^3$$
$$= \frac{\Delta x}{3}\left[f(x_0) + 4 f(x_1) + f(x_2) \right]$$

The idea of Adaptive Quadrature:

Firstly, compute the error

$$E_{\frac{\Delta x}{2}} = \frac{1}{2^\gamma - 1}\left(\tilde{I}_{\frac{\Delta x}{2}} - \tilde{I}_{\Delta x} \right)$$

If the error is bigger than the absolute tolerance, divide the interval into two equal parts and employ Simpson's rule on each half. This procedure will be repeated to obtain an approximation to the integral with needed absolute tolerance.

For the Gaussian probability function, the substitution can be made for infinite.

$$\begin{aligned} I &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} dt \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x_L} e^{-t^2/2} dt + \frac{1}{\sqrt{2\pi}} \int_{x_L}^{x} e^{-t^2/2} dt \\ &= I_1 + I_2 \end{aligned}$$

If I_1 is small enough, i.e.

$$I_1 \le \varepsilon I_2 \sim \frac{\varepsilon}{2}$$

ε is the machine epsilon

it can be ignored.

Then

$$\begin{aligned} \int_{-\infty}^{x_L} e^{-x^2/2} dx &\le \int_{-\infty}^{x_L} e^{-{x_L}^2/2} e^{x - x_L} dx \\ &\le \frac{\varepsilon}{2}, \text{ for } x_L < -1 \\ \int_{-\infty}^{x_L} e^{-{x_L}^2/2} e^{x - x_L} dx &= e^{-{x_L}^2/2} \le \frac{\varepsilon}{2} \\ \Rightarrow x_L &= \sqrt{-2 \ln \frac{\varepsilon}{2}} \end{aligned}$$

In single precision: $x_L = -5.8$

In double precision: $x_L = -8.6$, (1)

IV. Description of The Algorithm

Algorithm for Simpson's Rule:

$$Function\ Q(f,a,b)$$

$$\Delta x = \frac{b-a}{2}$$

$$I = \frac{\Delta x}{3}(f(a) + 4f(\frac{a+b}{2}) + f(b))$$

Algorithm for Adaptive Quadrature Function:

Function $Adaptive\ Q(f,a,b,tol)$

$$I_C = Q(f,a,b)$$

$$I_f = Q(f,a,\frac{a+b}{2}) + Q(f,\frac{a+b}{2},b)$$

$$E = \frac{1}{2^\gamma - 1}\left(I_f - I_C\right)$$

$$if\ |E| > tol$$

$$I_L = Adaptive\ Q(f,a,\frac{a+b}{2},\frac{tol}{2})$$

$$I_R = Adaptive\ Q(f,\frac{a+b}{2},b,\frac{tol}{2})$$

$$return\ I_C = I_L + I_R$$

$$else$$

$$return\ I_f + E$$

$$end\ if$$

V. Results

1.

In double precision and $\mathrm{AbsTol}=0.5\times10^{-7}$, the algorithm of adaptive quadrature function is tested by 2 steps.

Firstly, four different kinds of integrals were chosen randomly.

Table 1, Testing Routine Randomly

Integral	Approximation	Real Value	Relative Error	Significant Digits
$\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{0} e^{-t^2/2}dt = 0.5$	0.5	0.5	5.9536464e-10	8

$\frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} e^{-t^2/2}dt = 1$	1	1	7.0626549e-10	8
$\int_0^1 \frac{x^2}{(x+1)^2}dx = \frac{x^2+x-1}{x+1} - 2\ln\lvert x+1\rvert\Big\rvert_0^1$	0.11370564	0.11370564	7.9994754e-10	8
$\int_{0.5}^1 \frac{1}{\sin x \cos x} = \ln\lvert \tan x \rVert_{0.5}^1$	1.0476052	1.0476052	1.1056415e-10	9
$\int_0^2 \frac{1}{1+e^x} = x - \ln(1+e^x)\Big\rvert_0^2$	0.56621917	0.56621917	7.5553982e-11	9

It can be observed from table 1 that the routine of adaptive quadrature works correctly at least on these integrals. Since the Gaussian probability function $P(0.0)$ $P(\infty)$

Secondly, consider the integral of polynomials.

If the interval [a, b] is divided into $\frac{N}{2}$ parts, i.e. $\Delta x = \frac{b-a}{N}$, using Simpson's rule we have

$$I = \int_a^b f(x)dx = \tilde{I} + E$$
$$E = -\frac{b-a}{180}\Delta x^4 f^{(4)}(\mu),\ for\ some\ \mu \in [a,b] \qquad (2)$$

From formula (2), the polynomial under 4 degree can be exactly approximate by Simpson's rule.

Table 2, Testing Routine Using Polynomial

Integral	Approximation	Real Value	Relative Error	Significant Digits
$\int_0^{5.5} x^2 dx$	55.458333	55.458333	1.2812191e-16	15
$\int_0^{5.5} x^3 dx$	156.25	156.25	0	15
$\int_0^{5.5} x^4 dx$	1006.5687	1006.5687	2.2588986e-16	15
$\int_0^{5.5} x^5 dx$	4613.4401	4613.4401	0	15

From table 2, we can see that the relative error for $\int_0^{5.5} x^2 dx$ is not 0 while the error should be 0 with respect to formula (2). It is known that there are at most 15 significant digits in double precision. So it can be accepted since $\int_0^{5.5} x^2 dx$ has 15 significant digits. We can conclude that this routine works correctly for polynomials.

In general, the routine works correctly in double precision.

2.

To accurate the value $P(1.0)$ to at least 6 significant digits, we need to set

$$E_{rel} \le 0.5 \times 10^{-6}$$

Since

$$E_{rel} = \frac{E_{abs}}{P(1.0)} < \frac{\text{AbsTol}}{P(1.0)}$$
$$0.5 = P(0) < P(1.0) < 1$$

Then

$$\frac{\text{AbsTol}}{P(1.0)} < \frac{\text{AbsTol}}{1} \le 0.5 \times 10^{-6}$$
$$\Rightarrow \text{AbsTol} \le 0.5 \times 10^{-6}$$

Let $\text{AbsTol}=0.5\times10^{-7}$, $a = -8.6$ in double precision (gained from formula (1))

We have $P(1.0) = 0.84134475$

$$\text{RelTol} = \frac{\text{AbsTol}}{P(1.0)} \le 0.5 \times 10^{-7} / 0.84 < 0.5 \times 10^{-6}$$

So the integral approximation has 6 significant digits. And the result returned is integral approximation + error, which means it has at least 6 significant digits.

3.

Condition number:

$$\kappa = \frac{r}{C}\frac{dC}{dr}$$
$$= SP'(d_1)\frac{\sqrt{T-t}}{\sqrt{\sigma}} - Ke^{-r(T-t)}P'(d_2)\frac{\sqrt{T-t}}{\sqrt{\sigma}} + K(T-t)e^{-r(T-t)}P(d_2)$$
$$\approx 1$$

$$\kappa = \frac{S}{C}\frac{dC}{dS}$$
$$= \frac{S}{C}\left\{P(d_1) + \frac{1}{\sqrt{2\pi}}\frac{1}{\sigma\sqrt{T-t}}e^{-\frac{d_1^2}{2}} + \frac{1}{\sqrt{2\pi}}\frac{1}{\sigma\sqrt{T-t}}\frac{K}{S}e^{-r(T-t)}e^{-\frac{d_2^2}{2}}\right\}$$
$$\approx 500$$

It means the value of the call will lose at most 3 significant digits.

Since P(x) has 6 significant digits, the result has at least 3 significant digits.

a) The value of the call: $C = 98.567766$

$$\kappa = \frac{\sigma}{C}\frac{dC}{d\sigma}$$
$$= SP'(d_1)(-\frac{\log S/K}{\sigma^2\sqrt{T-t}} + \frac{\sqrt{T-t}}{2}) - Ke^{-r(T-t)}P'(d_2)(-\frac{\log S/K}{\sigma^2\sqrt{T-t}} - \frac{\sqrt{T-t}}{2})$$
$$\approx 1$$

It means the value of volatility won't lose significant digit. Then the volatility have at least 4 significant digits.

b) Using Bisection method, the implied volatility: $sigma = 0.076088285$

VI. Conclusions

The routine of adaptive quadrature works correctly through testing.

The value of P(1.0) = 0.84134475, and the result has 6 significant digits.

The value of the call: C = 98.567766, and the result has at least 3 significant digits.

The implied volatility: sigma = 0.076088285, and the result has at least 4 significant digits.

VII. Program Listing

adaptQuad.h

```
#define FM double
#pragma once

class Quadrature
{public:
	FM  adaptQuad(FM F(FM x), FM a, FM b, FM tol);
		FM  Simpson(FM F(FM x), FM a, FM b);
		FM  Bisect(FM f(FM p), FM a, FM b, FM absTol, FM relTol);

};
```

adaptQuad.cpp

```
#include <iostream>
#include "math.h"
#include "adaptQuad.h"
using namespace std;

//Simpson's rule
FM Quadrature::Simpson(FM F(FM x), FM a, FM b)
{
	FM dx = (b-a)/2;
	FM I = dx/3 * (F(a) + F(b) + 4*F(a+dx));

	return I;
}

FM Quadrature::adaptQuad(FM F(FM x), FM a, FM b, FM tol)
{
	FM Ic = Simpson(F, a, b);
	FM If = Simpson(F, a, (a+b)/2) + Simpson(F, (a+b)/2, b);
	FM E = (If - Ic)/15;//For simpson, gamma = 4, i.e. 2^gama-1=15
	FM IL, IR;

	if (abs(E)>tol)
	{
		IL = adaptQuad(F, a, (a+b)/2, tol/2);
		IR = adaptQuad(F, (a+b)/2, b, tol/2);
		return Ic = IL+IR;
	}
	else
	{
		return If + E;
	}
}


// to return the sigh of a value
int sign(FM s)
{
```

```
        int h=0;
        if (s>0.0)
        {
                h = 1;
        }
        if (s<0.0)
        {
                h = -1;
        }
        return h;
}


//Bisection Method
FM Quadrature::Bisect(FM f(FM p), FM a, FM b, FM absTol, FM relTol)
{
        FM p;
        int n, k;
        n= int( log( (b-a)/(absTol + relTol*min( abs(a), abs(b) )) ) / log(2.0));
        for (int i = 1; i <= n; i++)
        {
                p = a + (b - a)/2;
                if ( abs((b-a))/2 <= absTol + relTol* abs(p) )
                {
                        break;
                }
                else
                {
                        if ( sign(f(a)) * sign(f(p)) < 0 )
                    {
                                b = p;
                    }
                    else
                    {
                            a = p;
                    }
                }
                k=i;
        }

    return p;
}
```

test.cpp

```
#include <iostream>
#include <iomanip>
#include "math.h"
#include "adaptQuad.h"
#include "float.h"
#define pi 3.1415926535898;
using namespace std;

// the function of problem 2
FM P(FM x)
```

```
{
        FM g;
        g= 2.0*pi;
        FM f;
        f = 1.0/pow(g, 0.5) * exp(-x*x/2);
        return f;
}

// the function of problem 3 b)
FM f(FM sigma)
{
        Quadrature Q;
        FM C = 94.0;
        FM T = 117.0/365.0;
        FM t = 0.0;
        FM r = 0.0675;
        FM K = 3475.0;
        FM S = 3461.0;

        FM d1, d2;
        FM temp = T-t;
        d1 = (log(S/K) + (r + sigma*sigma/2.0)*(T - t))/(sigma*pow(temp, 0.5));
        d2 = (log(S/K) + (r - sigma*sigma/2.0)*(T - t))/(sigma*pow(temp, 0.5));

        FM f = S*Q.adaptQuad(P, -8.6, d1, 5.0e-8) - K*exp(-r*(T-t))*Q.adaptQuad(P, -8.6, d2,
5.0e-8) - C;
        return f;
}

FM tempf(FM x)
{
        return 1.0/(1.0+exp(x));
        //return 1.0/(sin(x)*cos(x));
        //return pow(x, 2)/pow((1+x), 2);
}

FM tempF(FM x)
{
        return x-log(1+exp(x));
        //return log(abs(tan(x)));
        //return x-1.0/(1.0+x)-2.0*log(1.0+x);
}

int main()
{
        Quadrature Q;

        FM s = Q.adaptQuad(P, -8.6, 1.0, 5.0e-8);

        //FM real = tempF(2.0)-tempF(0.0);

        cout<< "The P(1.0) is: "<<setprecision(8)<<s<<endl;  //<< " The real value is:
"<<setprecision(8)<<real<< " The abstol is: "<<setprecision(8)<<abs(s-real)<<endl;

        FM T = 54.0/365.0;
```

```
        FM t = 0.0;
        FM r = 0.0675;
        FM sigma = 0.135;
        FM K = 3425.0;
        FM S = 3441.0;
        FM d1, d2, C;
        FM temp = T-t;
        d1 = (log(S/K) + (r + sigma*sigma/2.0)*(T - t))/(sigma*pow(temp, 0.5));
        d2 = (log(S/K) + (r - sigma*sigma/2.0)*(T - t))/(sigma*pow(temp, 0.5));

        C = S*(0.5+Q.adaptQuad(P, 0.0, d1, 5.0e-8)) - K*exp(-r*(T-t))*(0.5+Q.adaptQuad(P, 0.0,
d2, 5.0e-8));
        cout<< "The call value is: "<<setprecision(8)<<C<<endl;

        FM relTol =  5e-11;
        FM absTol = DBL_EPSILON;

        FM rt = Q.Bisect(f, 0.01, 1.0, absTol, relTol);
        cout << "The sigma is: " <<setprecision(8)<< rt << endl<<endl;

        return 0;
}
```