MAP 5611 Intro to Computational Finance HW1

Student: Sen Zhang

Problem 1

I. Executive Summary

After comparing the computed machine epsilon, the largest and smallest floating point numbers with the IEEE standard, we checked that the computer is using the IEEE Standard. And from the basic and strong tests, it can be tested whether two floating point numbers are almost equal.

II. Statement of Problem

To check whether my computer uses the IEEE floating point numbers by writing a re-useable class defining a new floating point number type, FMfloat, that can be changed at compile time to either float or double, and can return the machine epsilon, the largest and smallest floating point numbers.

III. Description of The Mathematics

Today the computers use IEEE floating point numbers:

- For Single Precision: $FL(x) = (-1)^s 2^{E-127} 1.M,\ where\ 0 < E < 255$

The digits of M (Mantissa) are 23, then the

$single\ machine\ epsilon\ \epsilon = 2^{-23} \approx 1.192093 * 10^{-7}$

$sinlge\ smallest\ number = 2^{-126} * 1.00 \ldots 0 \approx 1.175494 * 10^{-38}$

$single\ largest\ number = 2^{127} * 1.11 \ldots 1 = (2 - 2^{-23})2^{127} \approx 3.402823 * 10^{38}$

- For Double Precision: $FL(x) = (-1)^s 2^{E-1023} 1.M,\ where\ 0 < E < 2047$

The digits of M (Mantissa) are 52, then the

$double\ machine\ epsilon\ \epsilon = 2^{-52} \approx 2.220446 * 10^{-16}$

$double\ smallest\ number = 2^{-1022} * 1.00 \ldots 0 = 2^{-1022} \approx 2.225073 * 10^{-308}$

$double\ largest\ number = 2^{1023} * 1.11 \ldots 1 = (2 - 2^{-52})2^{1023} \approx 1.797693 * 10^{308}$

IV. Description of The Algorithm

To return the machine epsilon, the largest and smallest floating numbers, the functions in "float.h" is needed.

For the "bool almostequal(float, float)" and bool almostequal(double,double)", the algorithm is as following:

```
bool falmostequal(float a, float b)
{
	const float epsilon = fepsilon();
	if (fabs(b-a) <= fabs(a)*epsilon && fabs(b-a) <= fabs(b)*epsilon)
		return 1;
	else
		return 0;

}

bool dalmostequal(double a, double b)
{
	const double epsilon = depsilon();
	if (fabs(b-a) <= fabs(a)*epsilon && fabs(b-a) <= fabs(b)*epsilon)
		return 1;
	else
		return 0;
}
```

V. Results

Table 1 Comparison of IEEE Standard and Computed Results for Single Precision

Single Precision	IEEE Standard	Computed	Error
Machine Epsilon	$1.19209 * 10^{-7}$	1.19209e-7	0
Smallest	$1.17549 * 10^{-38}$	1.17549e-38	0
Largest	$3.40282 * 10^{38}$	3.40282e38	0

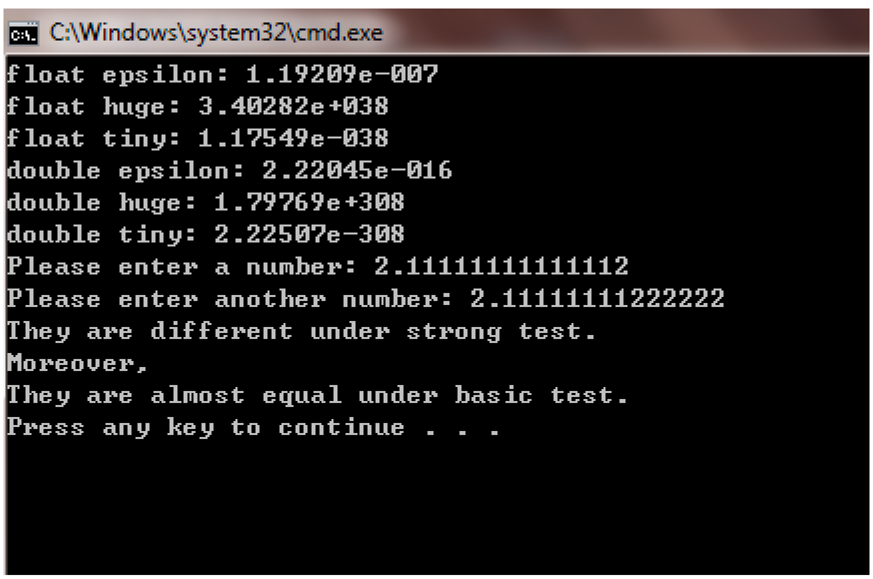
Table 2 Comparison of IEEE Standard and Computed Results for Double Precision

Double Precision	IEEE Standard	Computed	Error
Machine Epsilon	$2.22045 * 10^{-16}$	2.22045e-16	0
Smallest	$2.22507 * 10^{-308}$	2.22507e-308	0
Largest	$1.79769 * 10^{308}$	1.79769e308	0

Table 3 Example of the "bool almostequal( )"

Example for "almostequal"	
a	2.11111111111112
b	2.11111111222222
Result under strong test	Different
Result under basic test	Almost equal

Graph 1 Results of running codes



VI. Conclusions

As predicted, the computer is using the IEEE Standard after comparing it with the computed results for both single and double precisions. To return the machine epsilon, the largest and smallest floating numbers, the functions in "float.h" is used. For the "bool almostequal(float, float)" and bool almostequal(double, double)", the basic test is under single precision, and the strong test is under double precision. When two numbers satisfied the formulas $|b - a| \leq \epsilon|a|\ and\ \ |b - a| \leq \epsilon|b|$, it means that they are almost equal, if not, they are different.

VII. Program Listing

```
//FMfloat.h
class FMfloat
{
public:
	float fepsilon(float);
	double depsilon(double);
	float fhuge(float);
	double dhuge(double);
	float ftiny(float);
	double dtiny(double);
	bool falmostequal(float, float);
	bool dalmostequal(double, double);
};


//FMfloat.cpp
#include <iostream>
#include <cmath>
#include "float.h"
#include "FMfloat.h"
using namespace std;

float fepsilon()
{
	return FLT_EPSILON;
}

float fhuge()
{
	return FLT_MAX;
}

float ftiny()
{
	return FLT_MIN;
}

double depsilon()
{
	return DBL_EPSILON;
}

double dhuge()
{
	return DBL_MAX;
}

double dtiny()
{
	return DBL_MIN;
}
```

```
bool falmostequal(float a, float b)
{
        const float epsilon = fepsilon();
        if (fabs(b-a) <= fabs(a)*epsilon && fabs(b-a) <= fabs(b)*epsilon)
                return 1;
        else
                return 0;
}

bool dalmostequal(double a, double b)
{
        const double epsilon = depsilon();
        if (fabs(b-a) <= fabs(a)*epsilon && fabs(b-a) <= fabs(b)*epsilon)
                return 1;
        else
                return 0;
}

int main()
{
        cout << "float epsilon: " << fepsilon() << endl;
        cout << "float huge: " << fhuge() << endl;
        cout << "float tiny: " << ftiny() << endl;
        cout << "double epsilon: " << depsilon() << endl;
        cout << "double huge: " << dhuge() << endl;
        cout << "double tiny: " << dtiny() << endl;

        double a,b;
        cout<<"Please enter a number: ";
        cin>>a;
        cout<<"Please enter another number: ";
        cin>>b;

        if (dalmostequal(a,b))
                cout<<"They are almost equal under strong test.\n";
        else
                cout<<"They are different under strong test.\n";

        if (falmostequal(a,b))
                cout<< "Moreover, \n" <<"They are almost equal under basic test.\n";
        else
                cout<< "Moreover, \n" <<"They are different under basic test.\n";

        return 0;
}
```

Problem 2

The IEEE floating point number for Single Precision: $FL(x) = (-1)^s 2^{E-127} 1.M$, $where\ 0 < E < 255$

The digits of M (Mantissa) are 23, then

$$single\ machine\ epsilon\ \epsilon = 2^{-23}$$

$$sinlge\ smallest\ number = 2^{-126} * 1.00 \ldots 0 = 2^{-126}$$

$$sinlge\ 2nd\ smallest\ number = 2^{-126} * 1.00 \ldots 1 = (1 + 2^{23}) * 2^{-126}$$

$$single\ largest\ number = 2^{127} * 1.11 \ldots 1 = (2 - 2^{-23})2^{127}$$

$$single\ 2nd\ largest\ number = 2^{127} * 1.11 \ldots 0 = (2 - 2^{-23} - 2^{-23})2^{127}$$

1. $The\ smallest\ spacing = the\ 2nd\ smallest\ number - the\ smallest\ number = 2^{-23}2^{-126} = 1.40129 * 10^{-45}$

2. $The\ largest\ spacing = the\ lagest\ number - the\ 2nd\ largest\ number$
$$= 2^{-23}2^{127} = 2.02824 * 10^{31}$$

3. *The smallest relative spacing is between*

$$\left[\frac{smallest\ spacing}{2nd\ smallest\ number}, \quad \frac{smallest\ spacing}{smallest\ number}\right]$$
$$= \left[\frac{2^{-149}}{2^{-126}(1 + \ 2^{-23})}, \quad \frac{2^{-149}}{2^{-126}}\right] = \left[\frac{2^{-23}}{1 + \ 2^{-23}}, \ 2^{-23}\right]$$

The largest relative spacing is between

$$\left[\frac{largest\ spacing}{largest\ number}, \quad \frac{largest\ spacing}{2nd\ largest\ number}\right]$$
$$= \left[\frac{2^{104}}{2^{127}(2 - \ 2^{-23})}, \quad \frac{2^{104}}{2^{127}(2 - \ 2^{-22})}\right] = \left[\frac{2^{-23}}{2 - \ 2^{-23}}, \frac{2^{-23}}{2 - \ 2^{-22}}\right]$$

The smallest relative spacing and the largest relative spacing are different.

4. The largest and smallest (non-zero) absolute errors due to rounding in the IEEE number system are half of the largest and smallest spacing.

$$largest\ absolute\ errors = \frac{2.02824 * 10^{31}}{2} = 1.01412 * 10^{31}$$

$$smallest\ absolute\ errors = \frac{1.40129 * 10^{-45}}{2} = 7.00649 * 10^{-46}$$

5. $\epsilon * the\ smallest\ number = 2 * smallest\ absolute\ errors$

$$\epsilon * the\ largest\ number = 2 * largest\ absolute\ errors$$

Problem 3

Since the floating point numbers are approximations to the real numbers, then there exists some errors between them.

1. A false positive example in single precision:

Assume real numbers a=0.12345673, b=0.12345671

$$|a-b| = 0.00000002 \geq \epsilon * |a| = 1.47e-8, and\ |a-b| \geq \epsilon * |b|$$

However, the floating numbers fl(a) and fl(b)

$$|fl(a) - fl(b)| = 0 \leq \epsilon * |fl(a)|, and\ |fl(a) - fl(b)| \leq \epsilon * |fl(b)|$$

It's a contradiction.

2. A false negative example in single precision:

Assume real numbers a=6.12345678, b=6.12345641

$$|a-b| = 0.00000036 = 3.6e-7 \leq \epsilon * |a| = 7.3e-7, and\ |a-b| \leq \epsilon * |b|$$

However, the floating numbers fl(a) and fl(b)

$$|fl(a) - fl(b)| = 10e-6 \geq \epsilon * |fl(a)|, and\ |fl(a) - fl(b)| \geq \epsilon * |fl(b)|$$

It's also a contradiction.

Problem 4

No, floating point is not arithmetic distributive.

For an instance $\beta = 10, t = 2, L = -1, U = 2$:

a=9.9, b=7.5, c=0.1

$$\big(fl(a) + fl(b)\big) + fl(c) = 17 + 0.1 = 17,$$

$$fl(a) + \big(fl(b) + fl(c)\big) = 9.9 + 7.6 = 18$$

We can see that $\big(fl(a) + fl(b)\big) + fl(c) \neq fl(a) + \big(fl(b) + fl(c)\big)$