# Homework 1
# CS 210
# Niloufar Hosseini Pour

| Question | Points | Score |
|---|---|---|
| 1 | 10 | |
| 2 | 5 | |
| 3 | 5 | |
| 4 | 5 | |
| 5 | 10 | |
| 6 | 10 | |
| 7 | 10 | |
| 8 | 15 | |
| 9 | 15 | |
| 10 | 15 | |
| Total | 100 | |

## Errors and sources of error

1. (Heath 1.2) What are the approximate absolute and relative errors in approximating $\pi$ by each of the following quantities?
   Absolute error = approximate value − true value
   Relative error = absolute error /true value

   (a) 3
   Absolute error = 3 $-\pi$ = -0.1415926536
   Relative error = -0.1415926536 / $\pi$ = -0.0450703415

   (b) 3.14
   Absolute error = 3.14 $-\pi$ = -0.0015926536
   Relative error = -0.0015926536 / $\pi$ = -0.0005069574

   (c) 22/7
   Absolute error = 22/7 $-\pi$ = 0.0012644893
   Relative error = 0.0012644893 / $\pi$ = -0.0004024994

2. (Heath 1.3) If $a$ is an approximate value for a quantity whose true value is $t$, and $a$ has a relative error $r$, prove from the definitions of these terms that $a = t(1 + r)$.

   $Relative\ error = \frac{absolute\ error}{true\ value}$

   $Relative\ error = \frac{approximate\ value - true\ value}{true\ value}$

   $r = \frac{a-t}{t} \rightarrow a - t = rt \rightarrow a = t + rt \rightarrow a = t(1 + r)$

3. For each of the following statements, indicate whether the statement is true or false.

**T/F**✓ The use of a floating point number system results in so-called truncation error in numerical methods.

✓**T/F** It is possible for small rounding errors to catastrophically destroy the accuracy of an algorithm.

**T/F**✓ A large absolute error implies a large relative error.

## Conditioning and stability

4. For each of the following statements, indicate whether the statement is true or false.

✓**T/F** A problem is ill-conditioned if its solution is highly sensitive to changes in its data.

**T/F**✓ We can improve conditioning of a problem by switching from single to double precision arithmetic.

✓**T/F** In order to numerically solve a problem accurately, it is necessary to have both a well-conditioned problem and a stable algorithm.

✓**T/F** A condition number of 1 means the problem is well-conditioned.

5. (Heath 1.5) Consider the function $f : \mathbb{R} \to \mathbb{R}$ defined by $f(x, y) = x - y$. Measuring the size of the input $(x, y)$ by $|x|+|y|$, and assuming that $|x|+|y| \approx 1$ and $x - y \approx \epsilon$, show that $\text{cond}(f) \approx 1/\epsilon$. What can you conclude about the sensitivity of subtraction?

$Absolute\ forward\ error = f(x + \Delta x, y + \Delta y) - f(x, y) \approx f_x(x, y)\Delta x + f_y(x, y)\Delta y$

so that

$Relative\ forward\ error = \frac{f(x+\Delta x,y+\Delta y)-f(x,y)}{f(x,y)} \approx \frac{f_x(x,y)\Delta x+f_y(x,y)\Delta y}{f(x,y)}$

Since $f(x, y) = x - y \approx \epsilon$

relative forward error is :$\frac{f(x+\Delta x,y+\Delta y)-f(x,y)}{\epsilon} \approx \frac{f_x(x,y)\Delta x+f_y(x,y)\Delta y}{\epsilon}$

and relative backward error is : $\frac{\Delta x+\Delta y}{|x|+|y|}$

Since $|x| + |y| \approx 1$
so relative backward error would be : $\Delta x + \Delta y$

and condition number would be

$condition\ number \approx \left|\frac{f_x(x,y)\Delta x+f_y(x,y)\Delta y}{\epsilon(\Delta x+\Delta y)}\right| = \left|\frac{\Delta x-\Delta y}{\epsilon(\Delta x+\Delta y)}\right| \approx \frac{1}{\epsilon}$

What I can conclude about the sensitivity of subtraction is that since condition number is $\text{cond}(f) \approx 1/\epsilon$, this means that subtraction is very very sensitive to little changes in input data.

6. Show that for a differentiable function $f(x)$, the sensitivity of evaluating the function is described by the condition number

$$\kappa_f \approx \left|\frac{xf'(x)}{f(x)}\right|,$$

$Absolute\ forward\ error = f(x + \Delta x) - f(x) \rightarrow multiply\ and\ divide\ by\ \Delta x \rightarrow$

$Absolute\ forward\ error = \frac{f(x+\Delta x)-f(x)}{\Delta x}\Delta x \approx f'(x)\Delta x$

so that

$Relative\ forward\ error = \frac{f(x+\Delta x)-f(x)}{f(x)} \approx \frac{f'(x)\Delta x}{f(x)}$

and hence by dividing relative forward error by relative backward error we would have condition number.

$condition\ number \approx \left|\frac{\frac{f'(x)\Delta x}{f(x)}}{\frac{\Delta x}{x}}\right| = \left|\frac{xf'(x)}{f(x)}\right|$

and that the sensitivity of evaluating the inverse function $g(y)$ is given by

$$\kappa_g = \frac{1}{\kappa_f}.$$

In the above, I proved that condition number for $f(x)$ is :$\left|\frac{xf'(x)}{f(x)}\right|$
for $y = f(x)$, inverse function is $g(y) = f^-1(y) = x$ ,then $g'(y) = \frac{1}{f'(x)}$

hence according to condition number formula, for $g(y)$ condition number would be: $condition\ number \approx \left|\frac{yg'(y)}{g(y)}\right|$

Since we have $g'(y) = \frac{1}{f'(x)}$

so, $condition\ number \approx \left|\frac{yg'(y)}{g(y)}\right| = \left|\frac{f(x)(1/f'(x))}{x}\right| = \left|\frac{f(x)}{xf'(x)}\right| = \frac{1}{\kappa_f}$

## Floating point

7. (Heath 1.7) A floating point number system is characterized by four integers: the base $\beta$, the precision $p$, and the lower and upper limits $L$ and $U$ of the exponent range.

   (a) If $\beta = 10$, what are the smallest values of $p$ and $U$, and the largest value of $L$, such that both 2365.27 and 0.0000512 can be represented *exactly* in a *normalized* floating-point system?

   $2.36527 \times 10^3 \rightarrow p = 6, U = 3$
   $5.12 \times 10^{-5} \rightarrow L = -5$

   (b) How would your answer change if the system is not normalized, i.e., if gradual underflow is allowed?
   $2.36527 \times 10^3 \rightarrow U = 3, P = 6$
   $0.000512 \times 10^{-1} \rightarrow L = -1$

8. (Trefethen & Bau 13.1) Between an adjacent pair of nonzero IEEE single precision real numbers, how many IEEE doule precision numbers are there?

   $In\ single\ precision\ p = 24\ and\ in\ double\ precision\ p = 53$
   So the number of double precision numbers between two adjacent single precision numbers would be:
   $N = \frac{2^{-23}}{2^{-52}} = 2^{29} \rightarrow 2^{29} - 1$ numbers are there

9. For each of the following statements, indicate whether the statement is true or false.

**T/F**✓ If two numbers are exactly representable in floating point, then the result of an arithmetic operation on them is also an exactly representable floating point number.

✓**T/F** Floating point arithmetic is commutative, but not associative.

**T/F**✓ Floating point numbers are distributed uniformly througout their range.

**T/F**✓ In a denormalized floating point system, the representation of a number is unique.

**T/F**✓ Addition of two positive floating point numbers may cause underflow.

✓**T/F** Division of two positive floating point numbers may cause overflow.

**T/F**✓ Denormalization is used to mitigate arithmetic overflow.

**T/F**✓ In a normalized floating point system, the representation of a machine number is not unique.

10. *Computer problem* (Heath 1.10)

(a) Write a program to solve the quadratic equation $ax^2 + bx + c = 0$ using the standard quadratic formula

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

or the alternative formula

$$x = \frac{2c}{-b \mp \sqrt{b^2 - 4ac}}.$$

Your program should accept values for the coefficients $a$, $b$, and $c$ as input and produce the two roots of the equation as output. Your program should detect when the roots are not real, but need not use complex arithmetic explicitly (for example, you could return the real and imaginary parts of the complex conjugate roots in this case). You should guard against unnecessary overflow, underflow, and cancellation. Try to make your program robust when given unusual input values, such as $a = 0$ or $c = 0$, which otherwise would make one of the formulas fail. Any root that is within the range of the floating-point system should be computed accurately, even if the other is out of range. Submit a copy of your code. You may use the language of your choice.

(b) When should you use each of the two formulas?

when "b" is negative $-b - sqrt(4ac)$ may result in cancellation and when "b" is positive,

$-b + sqrt(4ac)$ may result in cancellation. In order to avoid catastrophic cancellation, when

b is negative, I would use $x = \frac{-b+\sqrt{b^2-4ac}}{2a}$ and $x = \frac{2c}{-b+\sqrt{b^2-4ac}}$. and when b is positive, I would

use: $x = \frac{-b-\sqrt{b^2-4ac}}{2a}$ and $x = \frac{2c}{-b-\sqrt{b^2-4ac}}$. and if $a = 0$ then if $b < 0$ I would use $x = \frac{2c}{-b+\sqrt{b^2-4ac}}$.

and if $b > 0$ I would use $x = \frac{2c}{-b-\sqrt{b^2-4ac}}$.

(c) Test your program using the following values for the coefficients and give your results:

| $a$ | $b$ | $c$ |
|---|---|---|
| 6 | 5 | $-4$ |
| $6 \times 10^{154}$ | $5 \times 10^{154}$ | $-4 \times 10^{154}$ |
| 0 | 1 | 1 |
| 1 | $-10^5$ | 1 |
| 1 | $-4$ | 3.999999 |
| $10^{-155}$ | $-10^{155}$ | $10^{155}$ |

| $a$ | $b$ | $c$ | *First root* | *second root* |
|---|---|---|---|---|
| 6 | 5 | $-4$ | 0.5 | $-1.33333$ |
| $6 \times 10^{154}$ | $5 \times 10^{154}$ | $-4 \times 10^{154}$ | 0.5 | $-1.33333$ |
| 0 | 1 | 1 | -1 | |
| 1 | $-10^5$ | 1 | $10^5$ | $1e-05$ |
| 1 | $-4$ | 3.999999 | 2.00098 | 1.99902 |
| $10^{-155}$ | $-10^{155}$ | $10^{155}$ | 1 | NAN |

C++ Code:

```
#include <iostream>
#include <cmath>
using namespace std;

float max(float a,float b, float c);

int main()
{
    float a,b,c,x1,x2,epsilon, maximum;
    epsilon= 1e-20;
    char answer;
    do{

    cout << "Enter a, b, c" << endl;
    cin >> a >> b >> c;

    maximum= max(a,b,c);

     a = a / maximum;
     b = b / maximum;
     c = c / maximum;

    if(abs(a) < epsilon){
         if(b < 0){
              x1 = 2*c / (-b + sqrt( b*b - 4 * a * c));
              cout<<"x is "<<x1;
         }
         else{
              x2 = 2*c / (-b - sqrt( b*b - 4 * a * c));
             cout<<"x is "<<x2;
         }
    }
    else{
    if(b < 0){

    x1 = (-b + sqrt( b*b - 4 * a * c))/(2*a);
    x2 = 2*c / (-b + sqrt( b*b - 4 * a * c));

    }
    else{

    x1 = 2*c / (-b - sqrt( b*b - 4 * a * c));
    x2 = (-b - sqrt( b*b - 4 * a * c))/(2*a);

    }

    cout<<"First root is "<< x1 <<" second root is: "<< x2<<endl;
    cout<<"Do you want to compute another equation? y/n"<<endl;
    cin >> answer;
    }}while(answer=='y');
    return 0;
}

float max(float a,float b, float c){

float maximum = abs(a);

if(abs(b)> maximum){
                maximum = abs(b);
                    }
if(abs(c)> maximum){
                maximum = abs(c);
                    }
return maximum;
}
```