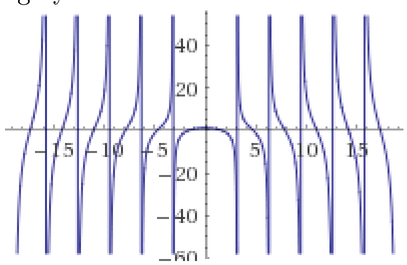
1.4

(a)$\triangle y = sin(x+h) - sin(x) \approx sin'(x) \times h = h \cdot cos(x)$

(b)$\triangle y/y = h \cdot cos(x)/sin(x) = h \cdot cot(x)$, $x \neq \pm n\pi, n = 0,1,2,3....$ When $x = \pm n\pi, n = 0,1,2,3...$, there is no real relative error.

(c)$cond = \frac{\triangle y/y}{\triangle x/x} = \frac{h \cdot cot(x)}{h/x} = x \cdot cot(x)$,$x \neq \pm n\pi, n = 0,1,2,3....$When $x = \pm n\pi, n = 0,1,2,3...$, there is no rcondition number.

(d)The curve of cond is like below with $x \in (-15, 15)$. When $x = \pm n\pi, n = 0,1,2,3...$, the absolute value of cond is extremely large. So the estimation will be highly sensitive.



1.12

(a)$x^2 - y^2$has a higher accuracy in floating-point arithmetic.

$fl\left(x^2 - y^2\right) = \left[\left(x^2\right)(1+\delta_1) - \left(y^2\right)(1+\delta_2)\right](1+\delta_3) = x^2 + x^2\delta_1 - y^2 - y^2\delta_2 + x^2\delta_3 + x^2\delta_1\delta_3 - y^2\delta_3 - y^2\delta_2\delta_3$

$\approx x^2(1+\delta_1+\delta_3) - y^2(1+\delta_2+\delta_3) = (x^2-y^2)(1+2\epsilon_{mach})$

$fl\left((x-y)(x+y)\right) = \left[(x-y)(1+\delta_1)\right]\left[(x+y)(1+\delta_2)\right](1+\delta_3) = (x^2-y^2)(1+2\epsilon_{mach})(1+\epsilon_{mach})$

(b)When x and y are one big with the other very small, $(x-y)(x+y)$is substantially more accurate.

When x and y are very different. For an example, x is 10^{10} bigger than y, x^2will be 10^{20}bigger than y^2, which makes x^2cease to change as $y^2$is relatively negligible.

Programming part

1.4

As k goes from 1 to 8, the estimation continues to increase at a slowing-down speed. So the error continues to approach 0.

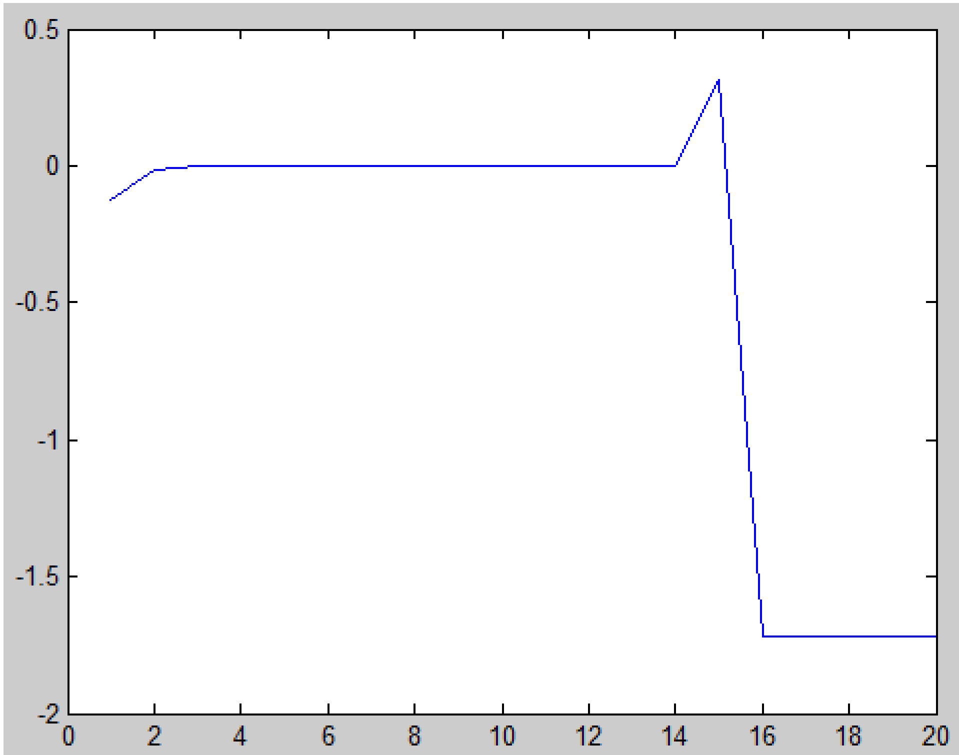
From 9 to 12, the value of $fl\left(1+1/n\right)^n$is greater than e. Because the rounding error introduced by multiplying a value close to $e \times (1+1/10^k), k = [9, 10, 11, 12]$

is influentially relatively big.

From 13 to 14, the rounding error by multiplying the $(1+1/n)$take away the value in the estimation funtction influentially.

At k=15, 1/n is the last position in mantissa, which means rounding error becomes relatively even bigger. And the rounding error makes the value bigger than e.

From 16 to 20, 1/n is smaller than ϵ, so 1+1/n=1 and the error becomes 1-e.




```
K=1:1:20;
for k=K
    n=10^k;
    result=(1+1/n)^n;
    RES(k)=result;
    DIFF(k)=result-exp(1);
end
plot(DIFF);
```

1.6 (a)$fl(x_n) = fl(x_0 + \underbrace{h + \cdots + h}_{n}) = a + nh + (n \cdot a + \frac{(1+n)\times n}{2} \cdot h)\epsilon_{mach}$

$fl(x_n) = fl(x_0 + nh) = [x_0 + nh(1+\delta)]\,(1+\delta) = x_0 \times (1+\delta) + nh \times (1+\delta)^2 \approx a + nh + (a + 2nh)\epsilon_{mach}$

The second one has a smaller rounding error. So the second one is better.

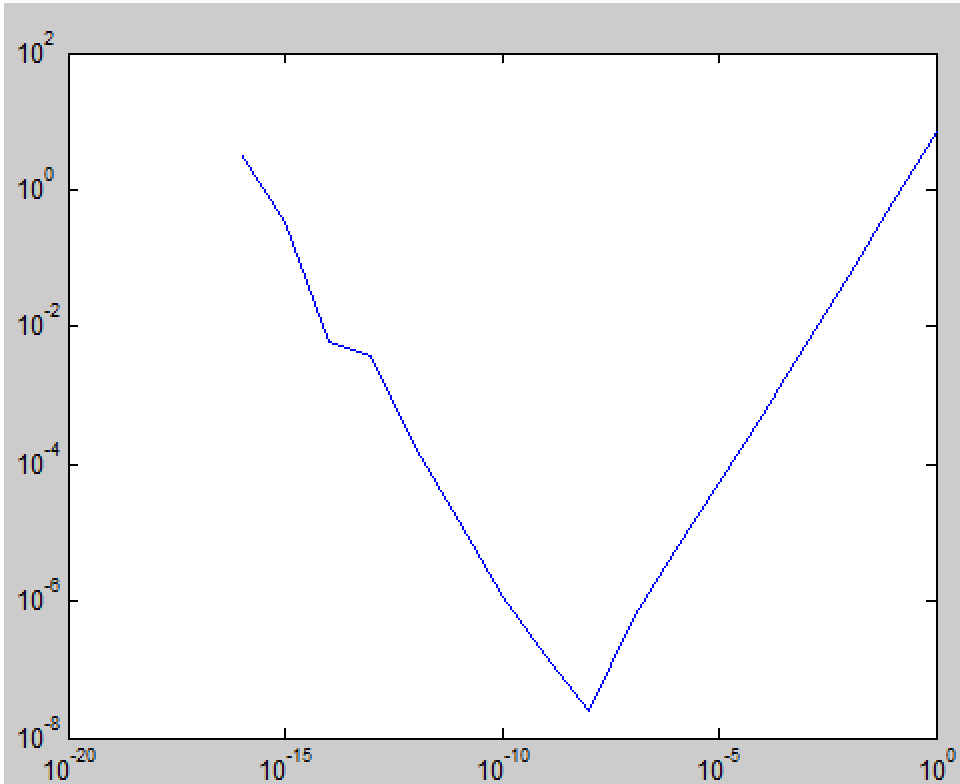
```
format long;
a=0;
b=1;
n=13^8;
xk=a;
for k=1:1:10^7
    xk=xk+1/n;
end

xk2=a+10^7*1/n;
```

The first formula is bigger than the second by 4.456487262549658e-13.

1.7(a)There is a minimum value for the magnitude of error. h=10^{-8}. $\epsilon_{mach} = \frac{1}{2}\beta^{1-p} = \frac{1}{2}10^{1-17} = 0.5 \times 10^{-16}$. $\sqrt{\epsilon_{mach}} = \sqrt{0.5 \times 10^{-16}} \approx 7.1 \times 10^{-9} = 0.71 \times 10^{-8}$. It matches well.

```
function[Derivative]=Appr(x,h,F)
%x, h, F
f=inline(F); %repalce the funciton to be test in the parentheses;
Derivative=(f(x+h)-f(x))/h;
```

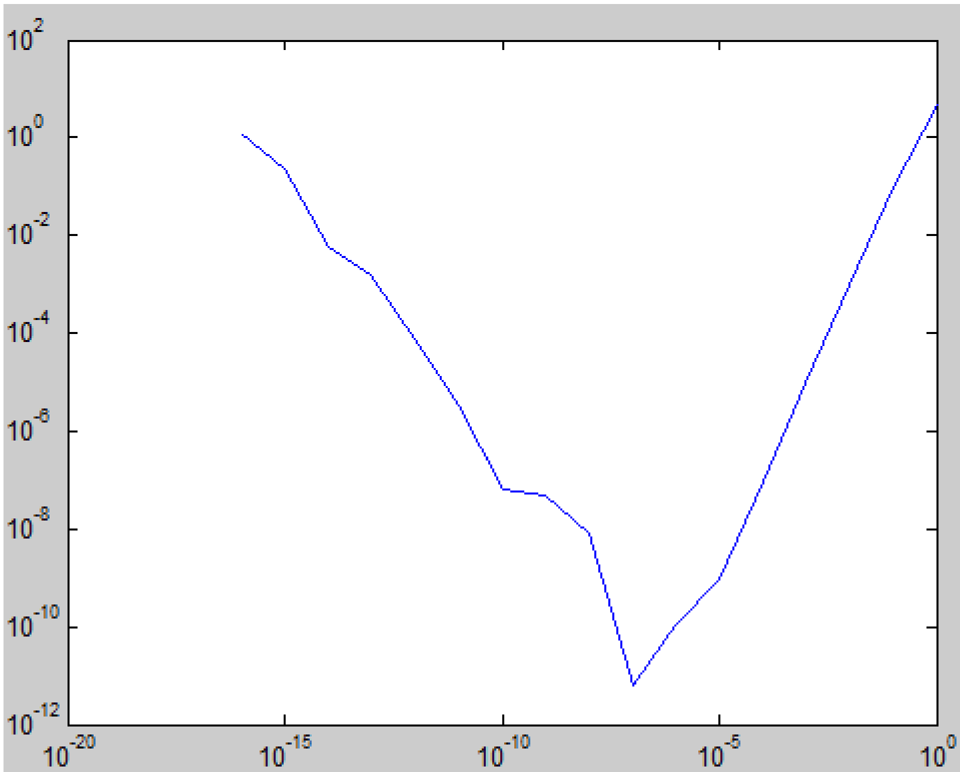
```
format long;
x=1; K=0:1:16; F='tan(x)';
for k=K,
    h(k+1)=10^(-k);
    APPR(k+1)=Appr(x, h(k+1), F);
    ERR(k+1)=APPR(k+1)-sec(x)^2;
end
loglog(h,abs(ERR));
min(abs(ERR))
```



(b)There is a minimum value for error. The corresponding value for h is 10^{-7}.It is 10 times of $\sqrt{\epsilon_{mach}}$. Because this estimation method reduces the truncation error (as you see, the minimum error from the estimation is much smaller than the one from the previous).

```
function[Derivative]=Appr2(x,h,F)
%x, h, F
f=inline(F); %repalce the funciton to be test in the parentheses;
Derivative=(f(x+h)-f(x-h))/2/h;
```

```
format long;
x=1; K=0:1:16; F='tan(x)';
for k=K,
    h(k+1)=10^(-k);
    APPR(k+1)=Appr2(x, h(k+1), F);
    ERR(k+1)=APPR(k+1)-sec(x)^2;
end
loglog(h,abs(ERR));
min(abs(ERR))
```



10^{2}
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
10^{-12}
10^{-20}
10^{-15}
10^{-10}
10^{-5}
10^{0}