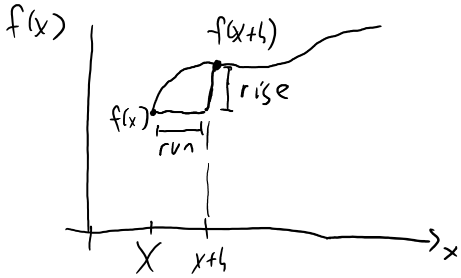
Hw 5 pg 2

Thursday, October 12, 2017 7:41 PM

5.1) derive $f'_{fwd}(x)$, $f'_{back}(x)$, $f'_{cent}(x)$



$$f'_{fwd}(x) = \text{slope} = \frac{\text{rise}}{\text{run}} \approx \frac{f(x+h) - f(x)}{h}$$

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \ldots \ldots$$

let $a = x$
$x = x_{i-1}$

$$f(x_{i-1}) = f(x_i) + \frac{f'(x_i)}{1!}(x_{i-1} - x_i) + \frac{f''(x_i)}{2!}(x_{i-1} - x_i)^2 + \frac{f'''(x_i)}{3!}(x_{i-1} - x_i)^3 \ldots$$

$\boxed{\text{let } h = x_i - x_{i-1} \quad (x_{i-1} = x_i - h)}$ backward

$$f(x_{i-1}) = f(x_i) - \frac{f'(x_i)}{1!}h + \frac{f''(x_i)}{2!}h^2 - \frac{f'''(x_i)}{3!}h^3 + \ldots$$

we want this term

$$f'(x_i)\cancel{h} = \frac{1}{h}\left[f(x_i) - f(x_{i-1}) + \frac{f''(x_i)h^2}{2!} - \frac{f'''(x_i)h^3}{3!} + \ldots\right]$$

$$f'(x_i) = \frac{f(x_i)}{h} - \frac{f(x_{i-1})}{h} + \underbrace{\left[\frac{f''(x_i)h}{2!} - \frac{f'''(x_i)h^2}{3!} + \ldots\right]}_{\text{error}}$$

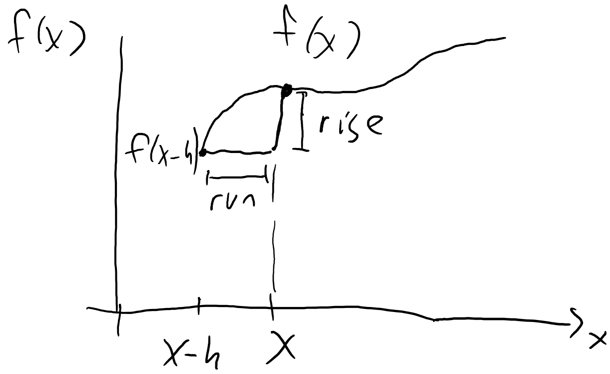
$$f'(x_i) = \frac{f(x_i) - f(x_i - h)}{h} + O(h)$$

backward difference

---

backward difference h $+O(h)$

Forward difference



$$f'_{back}(x) = slope = \frac{rise}{run} \approx \frac{f(x) - f(x-h)}{h}$$

using x_{i+1}, $h = x_{i+1} - x_i$

$$f(x_{i+1}) = f(x_i) + \frac{f'(x_i)}{1!}h + \frac{f''(x_i)}{2!}h^2 + \frac{f'''(x_i)}{3!}h^3 + \dots$$

$$f'(x_i)\cancel{h} = \frac{1}{h}\left[-f(x_i) + f(x_{i+1}) - \frac{f''(x_i)}{2!}h^2 - \frac{f'''(x_i)}{3!}h^3 - \dots\right]$$

$$f'(x_i) = \frac{-f(x_i)}{h} + \frac{f(x_{i+1})}{h} \underbrace{- \frac{f''(x_i)h}{2!} - \frac{f'''(x_i)h^2}{3!} - \dots}_{\text{error term}}$$

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h}$$

Centered difference

$$f(x_{i-1}) = f(x_i) - \frac{f'(x_i)h}{1!} + \frac{f''(x_i)}{2!}h^2 - \frac{f'''(x_i)}{3!}h^3 - \dots$$

$$f(x_{i+1}) = f(x_i) + \frac{f'(x_i)h}{1!} + \frac{f''(x_i)h^2}{2!} + \frac{f'''(x_i)}{3!}h^3 + \dots$$

$$-f(x_i) = -f(x_{i-1}) - \frac{f'(x_i)h}{1!} + \frac{f''(x_i)}{2!}h^2 - \frac{f'''(x_i)}{3!}h^3 - \dots$$

$$- \quad -f(x_i) = f(x_{i+1}) + \frac{f'(x_i)h}{1!} + \frac{f''(x_i)h^2}{2!} + \frac{f'''(x_i)}{3!}h^3 + \dots$$

$$0 = -f(x_{i+1}) + f(x_{i-1}) + 2\frac{f'(x_i)h}{1!} + 2\frac{f'''(x_i)}{3!}h^3 + \dots$$

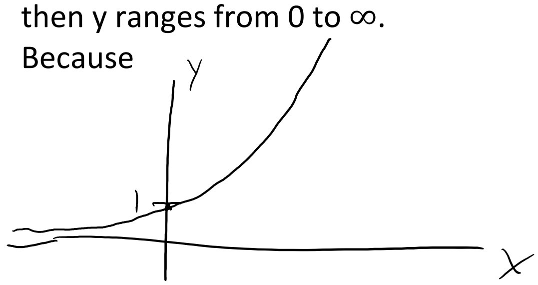
$$-f'(x_i) = \frac{1}{2h}\left[f(x_{i-1}) - f(x_{i+1}) + 2\frac{f'(x_i)}{3!}h^3 + \dots\right]$$

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - \frac{f'''(x_i)h^2}{3!} + \dots$$

error term $O(h^2)$

5.2 Finite difference: logarithmic derivative

Yes, we know that if the domain of e^x is [-∞,∞]
then y ranges from 0 to ∞.
Because



5.2.1 example 1

Question: calculate the value of $y = e^x$, for x = 0.5
y=1.64872

~~~~~~~~~~~~~~~~~~~~~~~~

```
#include<iostream>
#include<math.h>

doubleF(doublex,doubleh){
return(pow(x+h,2)-pow(x-h,2))/(2*h);
}

doubleG(doubley,doublek){
return(pow(log(y+k),2)-pow(log(y-k),2))/(2*k);
}

intmain(){
doublex=0.5;
doubley_05=exp(.5);
std::cout<<y_05<<'\n';
doublemaxError=pow(10,-32);
std::cout<<.1<<":"<<F(x,.1)<<"|"<<":"<<G(y_05,.1)<<'\n';
for(doubleh=.01;h>=maxError;h*=.01){
std::cout<<h<<":"<<F(x,h)<<"|"<<":"<<G(y_05,h)<<'\n';
}//for
}
```

Analysis:
For h = 2, F() returns 1.
For k = 2: G() returns an error.
A: for k = 2, using the center difference at .5 requires
    Ln(.5-2) and ln(.5+2), ln(1.5) is defined, but ln(-1.5) is not.
    So the center difference can't be computed.

```
#include<iostream>
#include<math.h>
//5.2Logarithmicdifference

doubleF(doublex,doubleh){
return(pow(x+h,2)-pow(x-h,2))/(2*h);
}

doubleG(doubley,doublek){
return(pow(log(y+k),2)-pow(log(y-k),2))/(2*k);
}

intmain(){

doublex=0.5;
doubley_05=exp(.5);
doublemaxError=pow(10,-32);
doubleh=2.0;

std::cout<<"h,k="<<h<<":"<<F(x,h)<<"|"<<G(y_05,h)<<'\n';
h=.1;
std::cout<<"h,k="<<h<<":"<<F(x,h)<<"|"<<G(y_05,h)<<'\n';

for(h=.01;h>=maxError;h*=.01){
std::cout<<"h,k="<<h<<":"<<F(x,h)<<"|"<<G(y_05,h)<<'\n';
}//for
}
```

F() | G()
h,k = 0.1:   1 | 0.605038
h,k = 0.01:   1 | 0.606516
h,k = 0.0001:   1 | 0.606531
h,k = 1e-006:   1 | 0.606531
h,k = 1e-008:   1 | 0.606531
h,k = 1e-010:   1 | 0.606531
h,k = 1e-012:   0.999978 | 0.606598
h,k = 1e-014:   0.999201 | 0.607847
h,k = 1e-016:   1.11029 | 0
h,k = 1e-018:   0 | 0
h,k = 1e-020:   0 | 0
h,k = 1e-022:   0 | 0
~~~~~~~~~~~~~~~~~~~~~~~~

h,k = 1e-024:   0 | 0
h,k = 1e-026:   0 | 0
h,k = 1e-028:   0 | 0
h,k = 1e-030:   0 | 0
h,k = 1e-032:   0 | 0

Q8-Q9.
A: as h and k decreases, the calculation starts to overflow.
For both F() and G() the approximation became worse at around 1e-16.

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

```
#include<iostream>
#include<math.h>
//5.2Logarithmicdifference

doublelnF(doublex,doubleh){
return(log(x+h)-log(x-h))/(2*h);
}

doublelnG(doubley,doublek){
return(log(log(y+k))-log(log(y-k)))/(2*k);
}

intmain(){

/doublex=1.0;
doubley_01=exp(1.0);
std::cout<<y_01<<'\n';
doublemaxError=pow(10,-32);


doubleh=.1;
std::cout<<"h,k="<<h<<":"<<lnF(x,h)<<"|"<<lnG(y_01,h)<<'\n';


for(h=.01;h>=maxError;h*=.01){
std::cout<<"h,k="<<h<<":"<<lnF(x,h)<<"|"<<lnG(y_01,h)<<'\n';
}//for


}
```

5.2.2 example 2
lnF() | lnG()
h,k = 0.1:   1.00335 | 0.368462
h,k = 0.01:   1.00003 | 0.367885
h,k = 0.0001:   1 | 0.367879
h,k = 1e-006:   1 | 0.367879
h,k = 1e-008:   1 | 0.367879
h,k = 1e-010:   1 | 0.36788
h,k = 1e-012:   1.00003 | 0.367928
h,k = 1e-014:   0.999201 | 0.377476
h,k = 1e-016:   0.555112 | 0
h,k = 1e-018:   0 | 0
h,k = 1e-020:   0 | 0
h,k = 1e-022:   0 | 0
h,k = 1e-024:   0 | 0
h,k = 1e-026:   0 | 0
h,k = 1e-028:   0 | 0
h,k = 1e-030:   0 | 0
h,k = 1e-032:   0 | 0

Q6-Q7:
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

As h and k decrease, the approximation became bad starting at around 1e-014.

5.6

A. S=0.9, <sigma> =0.5, call: 0.1416, delta: 0.515666

h, A, B, A - delta, B - delta
0.1, 0.519071, 0.513915, 0.00340544, -0.00175105
0.01, 0.5157, 0.515648, 3.41255e-005, -1.76916e-005
0.001, 0.515666, 0.515666, 3.41262e-007, -1.76934e-007
0.0001, 0.515666, 0.515666, 3.41233e-009, -1.76922e-009
1e-005, 0.515666, 0.515666, 3.44773e-011, -2.38094e-011
1e-006, 0.515666, 0.515666, 5.5389e-013, 3.44773e-011
1e-007, 0.515666, 0.515666, -5.23718e-010, -2.15323e-010
1e-008, 0.515666, 0.515666, 4.719e-009, 5.33579e-009

The numerical approximation is about the same.