Numerical analysis and applications HW1 B063012054 林祐安

note: I updated Latex symbol in figure titles after finishing the report, and if you have yet installed jupyter notebook, the website shows the code and outcome.

1. Proof the centered finite-difference approx. of the first derivative.

By the forward and backward Taylor series, we know that

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

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

subtracting the second equation from the first,

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

finally dividing the above equation by h,

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

or

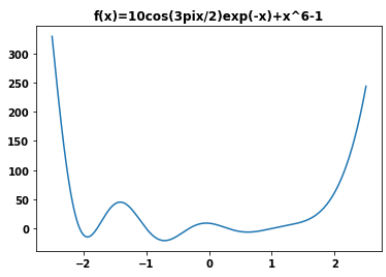
$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} + \mathcal{O}(h^2) \quad \text{where } O(h^2) = \frac{f'''(x_i)}{3!}h^2 + \dots$$

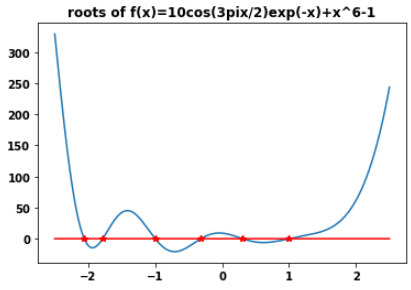
Q.E.D

2.

(a)See fig.1-function, obviously, there are 6 roots involved in the range. Using graphical method evaluates where the sign of f(x) changes.

Approximately, $x \approx$ -2.0597,-1.7838,-1,-0.3179,0.3045,1.(fig.1-roots) Moreover, the highest power of the function is six, which implies that there does NOT any other root exist outside the interval.





b

(function) (roots)

	0	1	2	3	4	5	6	\
x	-2.059701	-2.059700	-1.783822	-1.783821	-1.000001	-1.00000	-0.317895	
f(x)	0.000211	-0.000039	-0.000049	0.000112	0.000094	-0.00004	-0.000048	

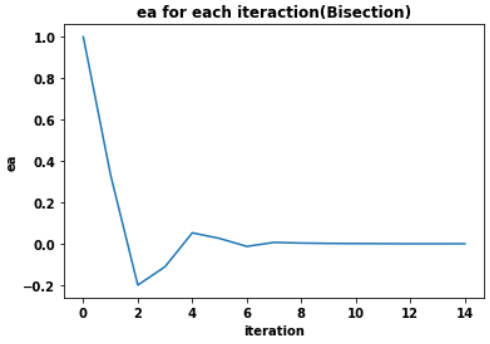
	7	8	9	10	11
x	-0.317894	0.304494	0.304495	1.000000	1.000001
f(x)	0.000015	0.000004	-0.000032	-0.000007	0.000016

(fig 1)

(b) first, we use bisection method to find roots, replacing x_l or x_u with x_r , depending on the sign of $f(x_r)$. if it equals to $f(x_l)$, x_l is substituted for x_r ,otherwise, x_u is substituted for x_r until satisfying the condition $|\varepsilon_a| < \varepsilon_s$.

If there is a root in the range, ε_a converges to zero(fig.2).

	xl	f(xl)	xr	f(xr)	xu	f(xu)	ea
0	0.000000	9.000005	0.250000	1.980602	0.500000	-5.273188	1.000000
1	0.250000	9.000005	0.375000	1.980602	0.500000	-5.273188	0.333333
2	0.250000	1.980602	0.312500	-2.338041	0.375000	-5.273188	-0.200000
3	0.250000	1.980602	0.281250	-0.281945	0.312500	-2.338041	-0.111111
4	0.281250	1.980602	0.296875	0.834622	0.312500	-0.281945	0.052632
5	0.296875	0.834622	0.304688	0.271181	0.312500	-0.281945	0.025641
6	0.296875	0.271181	0.300781	-0.006866	0.304688	-0.281945	-0.012987
7	0.300781	0.271181	0.302734	0.131795	0.304688	-0.006866	0.006452
8	0.302734	0.131795	0.303711	0.062375	0.304688	-0.006866	0.003215
9	0.303711	0.062375	0.304199	0.027750	0.304688	-0.006866	0.001605
10	0.304199	0.027750	0.304443	0.010418	0.304688	-0.006866	0.000802
11	0.304443	0.010418	0.304565	0.001775	0.304688	-0.006866	0.000401
12	0.304443	0.001775	0.304504	-0.002546	0.304565	-0.006866	-0.000200
13	0.304443	0.001775	0.304474	-0.000386	0.304504	-0.002546	-0.000100
14	0.304474	0.001775	0.304489	0.000712	0.304504	-0.000386	0.000050



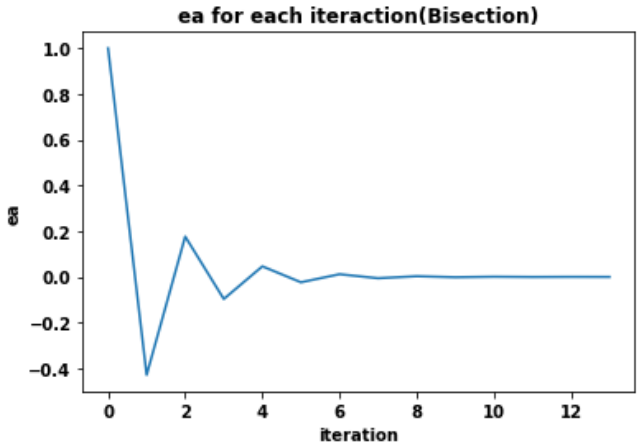
(fig. 2 $root \approx 0.3045$)

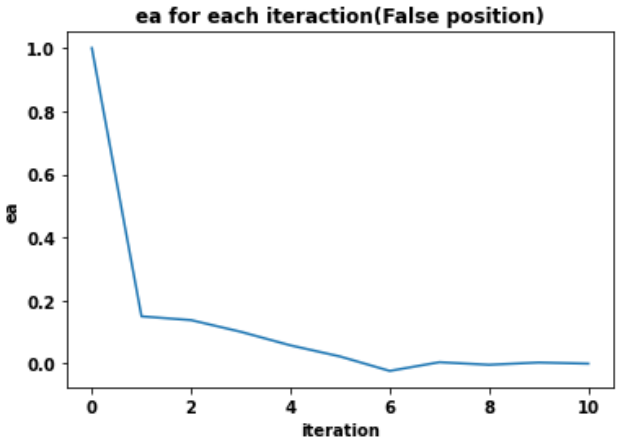
In another way, we are required to use false position method to find roots. For the same reason, replacing x_l or x_u with x_r depends on the sign of $f(x_r)$. We do that until satisfying the condition based on the problem. If there is a root in the range, ε_a converges to zero(fig.3).

	xl	f(xl)	xr	f(xr)	xu	f(xu)	ea
0	0.000000	9.000005	0.315277	-0.378957	0.500000	-5.273188	1.000000
1	0.000000	9.000005	0.198799	3.855695	0.315277	-5.273188	-0.585909
2	0.198799	9.000005	0.310570	-0.214312	0.315277	-0.378957	0.359892
3	0.198799	3.855695	0.300568	0.139412	0.310570	-0.378957	-0.033278
4	0.300568	3.855695	0.310044	-0.195800	0.310570	-0.214312	0.030563
5	0.300568	0.139412	0.304302	0.006769	0.310044	-0.214312	-0.018866
6	0.304302	0.139412	0.306690	-0.077690	0.310044	-0.195800	0.007785
7	0.304302	0.006769	0.304382	0.003935	0.306690	-0.195800	-0.007582
8	0.304382	0.006769	0.304567	-0.002617	0.306690	-0.077690	0.000607
9	0.304382	0.003935	0.304391	0.003617	0.304567	-0.077690	-0.000578
10	0.304391	0.003935	0.304497	-0.000102	0.304567	-0.002617	0.000347
11	0.304391	0.003617	0.304452	0.001456	0.304497	-0.002617	-0.000146
12	0.304452	0.003617	0.304496	-0.000067	0.304497	-0.000102	0.000142
13	0.304452	0.001456	0.304493	0.000039	0.304496	-0.000102	-0.000009

(fig. 3 $root \approx 0.3045$))

(c) Redo problem 2-(b),we find an exact root = 1.

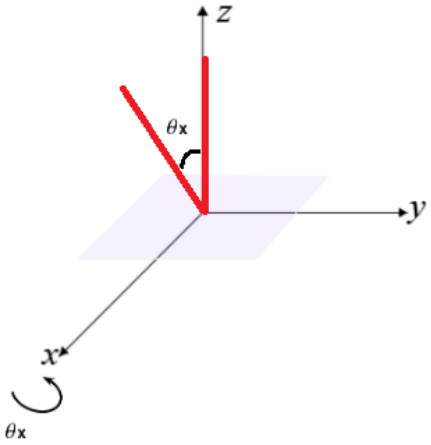




3.

(a)

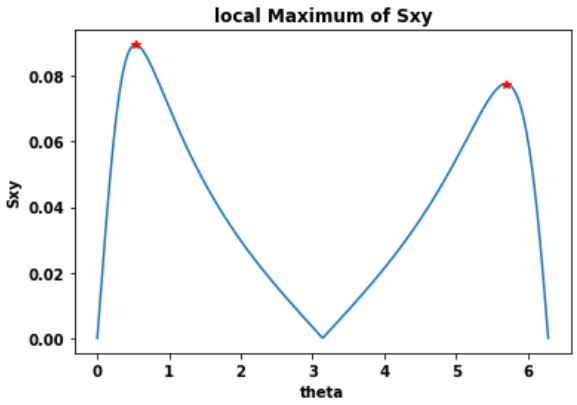
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$



(b)(C)(d)

```
class Simulation:
    #set the initial condition according to 3-(d)
    def __init__(self,theta,S_minus,t,T1,T2):
        self.S=S_minus
        self.theta=theta
        self.t=t
        self.T1=T1
        self.T2=T2
    #3-(a) create a matrix such that rotate a vector around the x-axis by an angle.
    def rotate_x(self):
        R=np.array([[1,0,0],[0,np.cos(self.theta),-np.sin(self.theta)],[0,np.sin(self.theta),np.cos(self.theta)]])
        self.S=R.dot(self.S)
    #3-(b) modulation
    def modulate(self):
        D1=np.array([[np.exp(-self.t/self.T2),0,0],[0,np.exp(-self.t/self.T2),0],[0,0,np.exp(-self.t/self.T1)]])
        D2=np.array([[0],[0],[1-np.exp(-self.t/self.T1)]])
        self.S=D1.dot(self.S)+D2
    #3-(c) clean up the x- and y- components
    def cleanup(self):
        self.S=np.array([[0,0,0],[0,0,0],[0,0,1]]).dot(self.S)
    #3-(d)
    def projection(self):
        Sxy=np.sqrt(np.square(self.S[0])+np.square(self.S[1]))
        return Sxy
```

(e) See fig.4, we know that there are two local maximum .Use graphical method to determine the global maximum.



	0	1
x	0.528845	5.691382
f(x)	0.089464	0.077551

(fig 4.)

In detail, we conclude that local maximum is the largest one among 3-nearest neighbors.

```
for i in range(1,N-1,1):
    # find where does the max value among 3-nearest neighbors.
    if Sxy[i-1]<Sxy[i] and Sxy[i]>Sxy[i+1]:
        root_x.append(theta_space[i])
        root_y.append(Sxy[i])
```

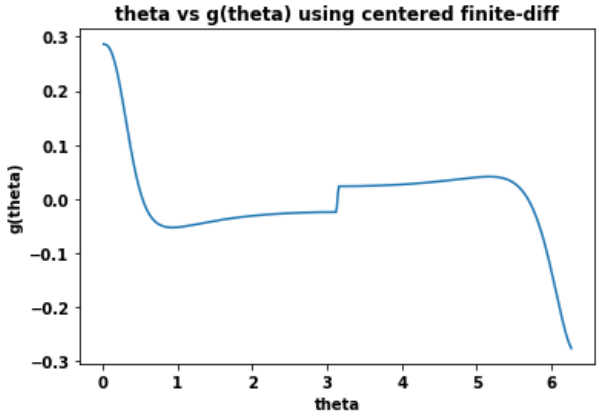
Hence, Two points are selected after searching where we are intrested in(fig 4. right-side).

Obviously, f(x) at x=0.53 is larger than one at x=5.69,

$$\theta_{opt} \approx 0.53(rad) \approx 30.37(deg)$$

To my surprise, if N is growing to larger and larger,in other words,increasing precision of sample, f(x) of each one becomes more and more similar. It means that θ_{opt} happens at 0.53 and 5.69 (rad), however, professor said the rotation degree is only in the range of [0,90] in practical on March 31.

(f) Roots are approximately 0.534, 3.141, 5.687(the resolution is too low because N isn't large enough.)



	0	1	2	3	4	5
theta	0.528845	0.541437	3.135297	3.147888	5.678791	5.691382
g(theta)	0.001952	-0.003100	-0.011912	0.011909	0.002894	-0.000028

(g) Graphical method can help us to find bracket,hence,we decide to search for roots in the range of [0.1,1.5].

	xl	x2	f(x2)	x1	f(x1)	xu	ea
0	0.100000	0.634752	0.087915	0.965248	0.072017	1.500000	1.000000
1	0.100000	0.430495	0.086751	0.634752	0.087915	0.965248	0.520668
2	0.430495	0.634752	0.087915	0.760990	0.082937	0.965248	0.321790
3	0.430495	0.556733	0.089386	0.634752	0.087915	0.760990	0.198877
4	0.430495	0.508514	0.089278	0.556733	0.089386	0.634752	0.140138
5	0.508514	0.556733	0.089386	0.586534	0.088852	0.634752	0.086610
6	0.508514	0.538315	0.089456	0.556733	0.089386	0.586534	0.053528
7	0.508514	0.526932	0.089464	0.538315	0.089456	0.556733	0.034214
8	0.508514	0.519897	0.089407	0.526932	0.089464	0.538315	0.021602
9	0.519897	0.526932	0.089464	0.531280	0.089464	0.538315	0.013351
10	0.526932	0.531280	0.089464	0.533967	0.089464	0.538315	0.008184
11	0.531280	0.533967	0.089464	0.535628	0.089456	0.538315	0.005032
12	0.531280	0.532941	0.089464	0.533967	0.089464	0.535628	0.003110
13	0.532941	0.533967	0.089464	0.534602	0.089464	0.535628	0.001922
14	0.533967	0.534602	0.089464	0.534994	0.089464	0.535628	0.001187
15	0.534602	0.534994	0.089464	0.535236	0.089456	0.535628	0.000733

$$root \approx 0.5354$$

(h)

```
#3-(h)
print(np.arccos(np.exp(-t/T1)))
```

```
0.5614542743438808
```

How to improve? Increase N,i.e. sampling resolution.

N	Graphical method	Centered finite	Golden-seciton	θ_{opt}
500	0.5288(5.8%)	0.534(4.9%)	0.5354(4.6%)	0.5615
5000	0.5581(0.61%)	0.5585(0.53%)	0.5585(0.53%)	0.5615
50000	0.5611(0.07%)	0.5611(0.07%)	0.5612(0.05%)	0.5615
500000	0.5614(0.02%)	0.5614(0.02%)	0.5619(0.07%)	0.5615
*( ):$error\ ratio = \frac{(value-\theta_{opt})}{\theta_{opt}}$				

Because we have more points to approch a continuous function, x and f(x) can be more close to the exact values.