Jamison Lahman (lahmanja@gmail.com)
Computational Physics: FORTRAN Version
by Steven E. Koonin and Dawn C. Meredith
Chapter 01 Exercises — July 7th, 2018

Exercise 1.1: Using any function for which you can evaluate the derivatives analytically, investigate the accuracy of the formulas in the Table 1[1] for various values of h.

Summary: To approximate the derivative numerically, we will utilize five equations based off the Maclaurin Series. The Maclaurin Series of function, *f(x)*, is given by the equation,

$$f(x) = f_0 + xf_0' + \frac{x^2 f_0''}{2!} + \frac{x^n f_0^n}{n!}. \tag{1}$$

Solving the above equation for *f'* yields,

$$f_0' = \frac{f(x) - f_0}{x} + \frac{-1}{x}\left(\frac{x^2 f_0''}{2!}\right) + \frac{-1}{x}\left(\frac{x^n f_0^n}{n!}\right) = \frac{f(x) - f_0}{x} + \mathcal{O}(h) \tag{2}$$

This gives the first of the '2-point' methods as this assumes the function is accurately described by a linear function over the interval $(0, h)$

$$f'(x) \approx \frac{f_{(x+h)} - f_x}{h}. \tag{3}$$

$$f'(x) \approx \frac{f_x - f_{(x-h)}}{h} \tag{4}$$

is found similarly and assumes the function is accurately described by a linear function over the interval $(-h, 0)$. The third equation, the symmetric "3-point" formula is the quadratic polynomial interpolation of the function over the two previous regions. The 3-point equation is found by taking the average of Eq. 3 and 4,

$$f' = \frac{f_{(x+h)} - f_{(x-h)}}{2h} + \mathcal{O}(2^4) \approx \frac{f_1 - f_{-1}}{2h}. \tag{5}$$

The final two equations, the "4-point" and "5-point" formulas, are similarly found and include more interactions fowards and backwards to create higher-order approximations of the function over the given interval. They, along with their higher-order derivatives, are given in Table 1. The 5-point formula is the five-point stencil in one-dimension.

Table 1: 4- and 5-point difference formulas for derivatives

	4-point	5-point
hf'	$\pm\frac{1}{6}(-2f_{\mp1} - 3f_0 + 6f_{\pm1} - f_{\pm2})$	$\frac{1}{12}(f_{-2} - 8f_{-1} + 8f_1 - f_2)$
$h^2 f''$	$f_{-1} - 2f_0 + f_1$	$\frac{1}{12}(-f_{-2} + 16f_{-1} - 30f_0 + 16f_1 - f_2)$
$h^3 f'''$	$\pm(-f_{\mp1} + 3f_0 - 3f_{\pm1} + f_{\pm2})$	$\frac{1}{2}(-f_{-2} + 2f_{-1} - 2f_1 + f_2)$
$h^4 f^{iv)}$	...	$f_{-2} - 4f_{-1} + 6f_0 - 4f_1 + f_2$

Solution: The function we will approximating the derivative numerically is,

$$F(x) = \sin(x).$$

The analytical derivative of the sine function is known to be,

$$\frac{d\sin(x)}{dx} = \cos(x).$$

For a formal proof of the derivation, see Wikipedia - derivative of the sine function. The exact value of the derivative at $x = 1$ (up to six decimal points) is 0.540302. Table 2 shows the accuracy of the formulas above for various values of h.

[1]Table 1.2 from Computational Physics: FORTRAN Version

Table 2: Error in evaluating the $d\sin/dx|_{x=1} = 0.540302$

h	Fwd 2-point Eq. 3	Bkwd 2-point Eq. 4	3-point Eq. 5	4-point Tab. 1	5-point Tab. 1
0.50000	-0.228254	0.183789	-0.022233	-0.009499	-0.001093
0.20000	-0.087462	0.080272	-0.003595	-0.000586	-0.000029
0.10000	-0.042939	0.041138	-0.000900	-0.000072	-0.000002
0.05000	-0.021257	0.020807	-0.000225	-0.000009	-0.000000
0.02000	-0.008450	0.008378	-0.000036	-0.000001	-0.000000
0.01000	-0.004216	0.004198	-0.000009	-0.000000	-0.000000
0.00500	-0.002106	0.002101	-0.000002	-0.000000	-0.000000
0.00200	-0.000842	0.000841	-0.000000	-0.000000	-0.000000
0.00100	-0.000421	0.000421	-0.000000	-0.000000	-0.000000
0.00050	-0.000210	0.000210	-0.000000	-0.000000	-0.000000
0.00020	-0.000084	0.000084	-0.000000	-0.000000	-0.000000
0.00010	-0.000042	0.000042	-0.000000	0.000000	0.000000
0.00005	-0.000021	0.000021	-0.000000	0.000000	0.000000
0.00002	-0.000008	0.000008	-0.000000	-0.000000	0.000000
0.00001	-0.000004	0.000004	-0.000000	-0.000000	-0.000000

Python interpretations of the formulas from above[2],

```
def forward2(x,h):
#Performs the forward 2-point method on the function defined by myFunc
#Input:  x -- independent variable
#        h -- step-size
#Output: ans -- dependent variable
    ans = (myFunc(x)-myFunc(x-h))/h
    return(ans)

def backward2(x,h):
#Performs the backward 2-point method on the function defined by myFunc
#Input:  x -- independent variable
#        h -- step-size
#Output: ans -- dependent variable
    ans = (myFunc(x+h)-myFunc(x))/h
    return(ans)

def symmetric3(x,h):
#Performs the symmetric 3-point method on the function defined by myFunc
#Input:  x -- independent variable
#        h -- step-size
#Output: ans -- dependent variable
    ans = (myFunc(x+h)-myFunc(x-h))/(2*h)
    return(ans)

def symmetric4(x,h):
#Performs the symmetric 4-point method on the function defined by myFunc
#Input:  x -- independent variable
#        h -- step-size
#Output: ans -- dependent variable
    ans = (-2*myFunc(x-h)-3*myFunc(x)+6*myFunc(x+h)-myFunc(x+2*h))/(6*h)
    return(ans)

def symmetric5(x,h):
#Performs the symmetric 5-point method on the function defined by myFunc
#Input:  x -- independent variable
#        h -- step-size
#Output: ans -- dependent variable
    ans = (myFunc(x-2*h)-8*myFunc(x-h)+8*myFunc(x+h)-myFunc(x+2*h))/(12*h)
    return(ans)
```

[2]For the full python script responsible for Table 2, see my GitHub

Exercise 1.2: Using any function whose definite integral you can compute analytically, investigate the accuracy of various quadrature methods for various values of h.

Summary: