Homework 5

MAD4401 - Numerical Analysis

Devere Anthony Weaver

October 27, 2022

Question 1

Consider the function $f(x) = \sin x$.

(a) Compute the *forward difference* approximation of $f'(0.5)$ using $h = 0.1$.

Recall, the *forward difference* approximation is given by,

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h} - \frac{h}{2} f''(\xi).$$

[1]:
```
import numpy as np
import math
```

[2]:
```
# implement forward difference formula
def forwardDifference(f,x0,h):
    """
    forwardDifference - computes the forward difference given by
    the forward difference formula for numerical approximation
    of a function's derivative; takes a function object,
    evaluation point, and value for h in the difference
    quotient.
    """
    return (f(x0+h)-f(x0))/h
```

[3]:
```
# implement given function
def f(x):
    return math.sin(x)
```

[4]:
```
# compute forward difference approximation at x0=0.5, h=0.1
forwardDifference(f,0.5,0.1)
```

[4]:
```
0.8521693479083237
```

Thus the forward approximation $\approx 0.8521693479083237$.

(b) Compute the *backward difference* approximation of $f'(0.7)$ using $h = -0.1$.

Recall, the *backward difference* approximation is given by,

$$f'(x_0) = \frac{f(x_0) - f(x_0 - h)}{h} + \frac{h}{2} f''(\xi).$$

[5]:
```
# implement backward difference formula
def backwardDifference(f,x0,h):
    """
    backwardDifference - computes the backward difference given by
    the backward difference formula for numerical approximation
    of a function's derivative; takes a function object,
    evaluation point, and value for h in the difference
    quotient.
    """
    return (f(x0)-f(x0-h))/h
```

[6]:
```
# compute backward difference approximation at x0=0.7, h=0.1
# (h=-0.1 if using the forward difference version)
backwardDifference(f,0.7,0.1)
```

[6]:
```
0.7957521384265565
```

Thus the backward approximation $\approx 0.7957521384265565$.

Question 2

Compute the actual error produced by the approximations in problem 1, and find the error bounds using the error formula.

To compute the actual error, we need the exact value of the derivative at 0.5 and 0.7, then find the difference from our approximated values. The first derivative of $f(x) = \sin x$ is $f'(x) = \cos x$.

[7]:
```
# implement first derivative (cos(x))
def fp(x):
    """
    Simply evaluates cosine function at x
    """
    return math.cos(x)
```

[8]:
```
# vectorize first derivative function to allow for vector arguments
fpv = np.vectorize(fp)
```

[9]:
```
# compute exact values
vals = np.array([0.5,0.7])
fpv(vals)
```

[9]:
```
array([0.87758256, 0.76484219])
```

Hence our exact value for $f'(0.5) = 0.87758256$ and the exact value for $f'(0.7) = 0.76484219$

[10]:
```
# implement error function to compute both at same time
def error(exact,approx):
    """
    Simply computes the actual error using a numeric vector
    of approximated values and a numeric vector of exact values
    """
    return math.fabs(exact-approx)
```

[11]:
```
# compute actual error for x0=0.5 (used forward error to approximate)
error(fp(0.5),forwardDifference(f,0.5,0.1))
```

[11]:
```
0.0254132139820491
```

Thus the actual error for our forward difference approximation of $f'(0.5)$ using $h = 0.1$ is $\approx 0.0254132139820491$.

[12]:
```
# compute actual error for x0=0.7 (used backward error to approximate)
error(fp(0.7),backwardDifference(f,0.7,0.1))
```

[12]:
```
0.03090995114206796
```

Thus the actual error for our backward difference approximation is $\approx 0.03090995114206796$.

Note that for both of these values when computed by hand and using four decimals differ slightly at about 0.0255 and 0.0312. These slight differences are possibly due to some rounding or truncation error in conjunction with having to divide by small values in the difference quotient.

To compute the error bounds for these approximations, we'll use the last term in both the forward and back difference formulas. The error is bounded by $\frac{M|h|}{2}$ where M is a bound on $|f''(x)|$ for $x_0 < x < x_0 + h$ for the forward difference and $x_0 - h < x < x_0$ for the backward difference.

For this particular problem, our second derivative is given by,

$$f''(x) = -\sin x.$$

Then for the approximation of $f'(0.5)$ using $h = 0.1$, the error bound is estimated by,

$$\Rightarrow \frac{|hf''(\xi)|}{2} = \frac{|h|}{2}|f''(\xi)| \leq \frac{0.1}{2}\left|\sup_{0.5<\xi<0.6}(-\sin\xi)\right| = \frac{0.1}{2}|-\sin 0.6| \approx 0.02823.$$

For the approximation of $f'(0.7)$ using $h = 0.1$, the error bound is estimated by,

$$\Rightarrow \frac{|hf''(\xi)|}{2} = \frac{|h|}{2}|f''(\xi)| \leq \frac{0.1}{2}\left|\sup_{0.6<\xi<0.7}(-\sin\xi)\right| = \frac{0.1}{2}|-\sin 0.7| \approx 0.03221.$$

Question 3

Use the most accurate three-point formula to determine each missing entry in the given table.

For this problem we're give a set of evaluation points and their function values as the following set of ordered pairs, $\{(8.1, 16.94410), (8.3, 17.56492), (8.5, 18.19056), (8.7, 18.82091)\}$ where the first value is the evaluation point and the second is its output.

To determine the approximate derivative value for $f'(8.1)$ and $f'(8.7)$, we should opt to use the three-point endpoint formula. This is given by,

$$f'(x_0) = \frac{1}{2h}\left[-3f(x_0) + 4f(x_0 + h) - f(x_0 + 2h)\right] + \frac{h^2}{3}f^{(3)}(\xi_0),$$

where ξ_0 lies between x_0 and $x_0 + 2h$.

[13]:
```
# implement three-point endpoint formula for example
def threeEndPt(x0,x1,x2,h):
    """
    Implements three-point endpoint formula for the values in
    this given problem where x0=f(x0), x1=f(x0+h), and x2=f(0x+2h)
    """
    return (1/(2*h))*(-3*x0+4*x1-x2)
```

[14]:
```
# compute for endpoint for x0=8.1 and h=0.2
threeEndPt(16.94410, 17.56492, 18.19056, 0.2)
```

[14]:
```
3.092050000000013
```

Thus for $x = 8.1$, $f'(x) \approx 3.092050000000013$

[15]:
```
# compute for endpoint for x0=8.7 and h=-0.2)
threeEndPt(18.82091, 18.19056, 17.56492, -0.2)
```

[15]:
```
3.163525000000007
```

Thus for $x = 8.7$, $f'(x) \approx 3.163525000000007$.

To determine the approximate derivative value for $f'(8.3)$ and $f'(8.5)$, we should opt to use the three-point midpoint formula. This is given by,

$$f'(x_0) = \frac{1}{2h}\left[f(x_0 + h) - f(x_0 - h)\right] + \frac{h^2}{6}f^{(3)}(\xi_1),$$

where ξ_1 lies between $x_0 - h$ and $x_0 + h$.

[16]:
```
# implement three-point midpoint formula
def threeMidPt(x1,x2,h):
    """
    Implements three-point midpoint formula for the values
    in this given problem where x1=f(x0+h) and x2=f(x0-h)
    """
    return (1/(2*h))*(x2-x1)
```

[17]:
```
# compute for midpoint for x0=8.3 and h=0.2
threeMidPt(16.94410,18.19056,0.2)
```

[17]:
```
3.1161500000000064
```

Thus for $x = 8.3$, $f'(x) \approx 3.1161500000000064$.

[18]:
```
# compute for midpoint for x0=8.5 and h=0.2
threeMidPt(17.56492,18.82091,0.2)
```

[18]:
```
3.1399750000000015
```

Thus for $x = 8.5$, $f'(x) \approx 3.1399750000000015$.