Numerical Differentiation

October 26, 2022

1 4.1 - Numerical Differentiation

This section covers different methods used for numerical differentiation of possibly complicated functions.

NOTE: The derivations of these formulae can be found in the typed up pdf notes or in Burden's Numerical Analysis text.

The *forward-difference formula* is given by,

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

This formula's error is bounded by $\frac{M|h|}{2}$ where M is a bound on $|f''(x)|$ for $x_0 < x < x_0 + h$.

The *backward-difference formula* is given by,

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

This formula's error is bounded by $\frac{M|h|}{2}$ where M is a bound on $|f''(x)|$ for $x_0 - h < x < x_0$.

The *centered-difference formula* is given by,

$$f'(x_0) = \frac{f(x_0+h) - f(x_0-h)}{2h} - \frac{h^2}{6} f'''(\xi).$$

Example 1 Use the *forward-difference formula* to approximate the derivative of $f(x) = \ln x$ at $x_0 = 1.8$ using (i) $h = 0.1$, (ii) $h = 0.05$, and (iii) $h = 0.01$ and determine the bounds for the approximation errors.

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

[3]:
```python
# implement forward-difference formula for this example
def fd(x,h):
    return ((math.log(x+h)-math.log(x))/h)
```

[4]:
```python
# evaluation point
x = 1.8
```

[5]:
```
# h=0.1
fd(x,0.1)
```

[5]: 0.5406722127027574

[6]:
```
# h=0.05
fd(x,0.05)
```

[6]: 0.5479794837622887

[7]:
```
# h=0.01
fd(x,0.01)
```

[7]: 0.5540180375615322

To compute the error bound, we must use the second derivative of the given function,

$$f(x) = \ln x \Rightarrow f'(x) = \frac{-1}{x^2}.$$

Since we are using the *forward-difference formula*, we know that for the error term must be $x_0 < x < x_0 + h$. Here $x_0 = 1.8$ thus $x_0 + h = 1.9$.

So, to compute a bound for this approximation error,

$$\Rightarrow \frac{|hf''(\xi)|}{2} = \frac{|h|}{2\xi^2} < \frac{0.1}{2(1.8)^2} \approx 0.0154.$$

NOTE: Some steps are skipped to compute this error bound, fill in the details as need. Recall, we wanted to maximize our second derivative.

[8]:
```
# implement function to compute error bound
def fderr(x,h):
    return math.fabs(((h/2)*(-1/x**2)))
```

[11]:
```
# x=1.8, h=0.1
fderr(x,0.1)
```

[11]: 0.015432098765432098

[12]:
```
# x=1.8, h=0.05
fderr(x,0.05)
```

[12]: 0.007716049382716049

[13]:
```
# x=1.8, h=0.01
fderr(x,0.01)
```

[13]: 0.0015432098765432098

```
[ ]:
```