BICK-hw6

October 18, 2022

[12]:
```
import math
import pandas as pd
pd.options.display.float_format = '{:.2f}'.format
```

0.1 Nathan Bick HW6

1. Apply a fixed point method to find the root of $cosx = sinx$ on $0, \pi/2$, by converting the equation into a fixed point equation

$$x = g(x) = x + cosx - sinx$$

given $x_0 = 0$. Note that $cos(\pi/4) = sin(\pi/4)$, so $x^* = \pi/4$.

[33]:
```
# we check the value of pi/4
true_root = math.pi/4
print("True root is " + str(true_root))

# Our g(x) from x = g(x)
def g(x):
    y = x+math.cos(x)-math.sin(x)
    return y

# Fixed Point Iteration
x = 0
e = abs(x - true_root)
print(x)
for i in range(0,32):
    x_tmp = x
    x = g(x_tmp)
    e_tmp = e
    e = abs(x - true_root)
    e_ratio = e/e_tmp
    if i in [1,10,20,30]:
        print("Iteration Table")
        print("k: " + str(i))
        print("x_k: " + str(x_tmp))
        print("g(x_k): " + str(x))
```

```python
        print("e_k: " + str(e))
        print("e_k/e_{k-1}: " + str(e_ratio))
        print("\n")
```

```
True root is 0.7853981633974483
0
Iteration Table
k: 1
x_k: 1.0
g(x_k): 0.6988313210602433
e_k: 0.08656684233720502
e_k/e_{k-1}: 0.4033835110998099


Iteration Table
k: 10
x_k: 0.785323533884696
g(x_k): 0.7854290759536855
e_k: 3.0912556237217004e-05
e_k/e_{k-1}: 0.41421356105897267


Iteration Table
k: 20
x_k: 0.7853981523017728
g(x_k): 0.7853981679934277
e_k: 4.595979374855119e-09
e_k/e_{k-1}: 0.4142135716927274


Iteration Table
k: 30
x_k: 0.7853981633957987
g(x_k): 0.7853981633981315
e_k: 6.832312493543213e-13
e_k/e_{k-1}: 0.41418764302059496
```

The above results are summarized in the table below.

k	x_k	$g(x_k)$	e_k	$\frac{e_k}{e_{k-1}}$
1	1.0	0.6988	0.0865	0.4033
10	0.7853	0.7854	3.0912e-05	0.4142
20	0.7853	0.7853	4.595e-09	0.4142
30	0.7853	0.7853	6.8323e-13	0.4141

2. Let $f(x) = x^6 - x - 1$.

(a) Use 4 iterations of the Newton's method with $x_0 = 2$ to get an approximate root for this equation.
(b) Use 4 iterations of the Secant method with $x_0 = 2$ and $x_1 = 1$ to get an approximate root for this equation.

Newton's Method uses the Taylor series approximation of $f(x)$ at x_k rather than f. We recall the Taylor approximation is given as $f(x) \approx l(x) = f(x_k) + f'(x_k)(x - x_k)$.

We solve for $l(x) = 0$ rather than $f(x) = 0$. This is equivalent to $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$. We use this to get the kth iteration of Newton's method.

Firstly, we have $x_0 = 2$. Our function is $f(x) = x^6 - x - 1$, so $f'(x) = 6x^5 - 1$. Evaluating at $x_0 = 2$, $f(2) = 2^6 - 2 - 1 = 61$ and $f'(2) = 6(2)^5 - 1 = 191$.

Therefore we have $x_1 = 2 - \frac{61}{191} = \frac{321}{191} \approx 1.681$

The later iterations are shown implemented in python.

[5]:
```
# now implement in code to check our work for first iteration and provide␣
↪additional iterations
# Our f(x) function
def f(x):
    y = x**6-x-1
    return y
# derivative
def df(x):
    y = 6*x**5-1
    return y


x=2
print(x)
for i in range(0,4):
    x = x-f(x)/df(x)
    print(x)
```

```
2
1.6806282722513088
1.4307389882390624
1.2549709561094364
1.1615384327733131
```

The secant method is a modification of Newton's method, replacing the derivative $f'(x)$ with a difference-based approximation, namely $f'(x_k) \approx \frac{f(x_k) - f(k_{k-1})}{x_k - x_{k-1}}$.

So the iteration of the secant method is given by $x_{k+1} = x_k - \frac{x_{k-1} f(x_k) - x_k f(x_{x_k - 1})}{f(x_k) - f(x_{k-1})}$, requiring two initial points.

Firstly, we have $x_0 = 2$ and $x_1 = 1$. So we have $f(2) = 2^6 - 2 - 1 = 61$ and $f(1) = 1^6 - 1 - 1 = -1$. We therefore have $x_2 = \frac{1*61 - 2*(-1)}{61 - (-1)} = 63/62 = \approx 1.0161$

Then the remaining iterations are shown in python code below.

[32]:
```
# Our f(x) function
def f(x):
    y = x**6-x-1
    return y

x0=2
x1=1
print(x0);print(x1)
for i in range(0,4):
    fx0 = f(x0)
    fx1 = f(x1)
    xtemp = x0
    x0 = (x0*fx1 - x1*fx0) / (fx1-fx0)
    x1 = xtemp
    print(x0)
    print(x1)
```

```
2
1
1.0161290322580645
2
1.0306747541311725
1.0161290322580645
1.1756889442904006
1.0306747541311725
1.1236790653714195
1.1756889442904006
```

3. Consider the equation $e^{100x}(x-2) = 0$. Apply Newton's method several times with $x_0 = 1$. What do you observe?

Below we implement the Newton's method for the function several times and observe the outputs. We see that there is a stready decline in the value of x, not a convergence. This may be because the true root is $x = 2$, and our choice of initital value is too far from that.

[16]:
```
# Our f(x) function
def f(x):
    y = math.e**(100*x)*(x-2)
    return y
# derivative
def df(x):
    y = 100*math.e**(100*x)*(x-2) + math.e**(100*x)
    return y


x=1
print(x)
```

```python
for i in range(0,10):
    x = x-f(x)/df(x)
    print(x)
```

```
1
0.98989898989899
0.9797989999989798
0.9697000097980198
0.9596019994079774
0.949504949530219
0.9394088414324873
0.9293136569269025
0.919219378349024
0.9091259885379105
0.899033470817122
```