Numerical Analysis - Homework 3

Chris Hayduk

Question 1

In [17]:
```
import math
import numpy as np

def secant(x0, x1, f, max_n):
    fx0 = f(x0)

    n = 0

    error = abs(x0-x1)
    ep = 10**(-10)

    while error > ep and n <= max_n:
        print("\n Iteration: ", n, ", x_n: ", x0, ", x_{n+1}: ", x1, ", Error:
", error)
        n += 1
        fx1 = f(x1)

        if fx1 - fx0 == 0:
            print("f(x1) = f(x0)); Division by zero")
            break

        x2 = x1 - fx1*(x1-x0)/(fx1-fx0)

        error = abs(x2 - x1)

        x0 = x1
        x1 = x2
        fx0 = fx1

secant(3, 2, lambda x: math.exp(-x) - math.cos(x), 6)
```

```
 Iteration:  0 , x_n:  3 , x_{n+1}:  2 , Error:  1

 Iteration:  1 , x_n:  2 , x_{n+1}:  0.8706020782571979 , Error:  1.129397921
742802

 Iteration:  2 , x_n:  0.8706020782571979 , x_{n+1}:  1.198554234317791 , Err
or:  0.32795215606059314

 Iteration:  3 , x_n:  1.198554234317791 , x_{n+1}:  1.3229955468185879 , Err
or:  0.12444131250079682

 Iteration:  4 , x_n:  1.3229955468185879 , x_{n+1}:  1.2914674589747202 , Er
ror:  0.031528087843867736

 Iteration:  5 , x_n:  1.2914674589747202 , x_{n+1}:  1.2926813363329115 , Er
ror:  0.0012138773581913398

 Iteration:  6 , x_n:  1.2926813363329115 , x_{n+1}:  1.2926957264424896 , Er
ror:  1.4390109578155119e-05
```

Question 2

We know that $g(\alpha) = \alpha$ and $|g'(\alpha)| < 1$. Since g' is continuous, there exists a neighborhood around α such that $a = \alpha - \varepsilon < \alpha < b = \alpha + \varepsilon$ for some $\varepsilon > 0$ satisfying $|g'(a)|, |g'(\alpha)|, |g'(b)| < 1$.

Since $|g'(x) < 1|$ for all $a \leq x \leq b$, any change in $g(x)$ will be smaller than the corresponding change in x. Thus, $a \leq x \leq b \implies g(a) \leq g(x) \leq g(b)$

Furthermore, since g and g' are continuous over $\mathbb{R}$, we know that g and g' are continuous on $a < \alpha < b$ as defined above.

Thus, we may apply Theorem 3.4.2.

By S2 in Theorem 3.4.2, for any initial estimate x_0 in $[a, b]$, the iterates x_n will converge to α.

Hence, if x_0 is sufficiently close to α, then the iteration $x_{n+1} = g(x_n)$ converges to α.

Question 3

a) Let $g(x) = cosh(x) - 1 - \frac{1}{2}x^2$.

$g(0) = cosh(0) - 1 - \frac{1}{2}0^2 = 1 - 1 - 0 = 0$

$g'(x) = sinh(x) - x \implies g'(0) = 0$

$g''(x) = cosh(x) - 1 \implies g''(0) = 0$

$g'''(x) = sinh(x) \implies g'''(0) = 0$

$g''''(x) = cosh(x) \implies g''''(0) = 1 \neq 0$

Thus, by 3.60 in the textbook, 0 is a root of multiplicity 4.

b) Since α is a root of multiplicity m for $f(x)$, we can write $f(x) = (x - \alpha)^m h(x)$ where $h(x)$ is some continuous function with $h(\alpha) \neq 0$.

The definition of Big-O notation states that $f(x) = O(g(x))$ as $x \to a$ if $\exists c \in \mathbb{R}$ such that $|f(x)| \leq c|g(x)|$ as $x \to a$.

Since $h(x)$ in the equation above is a continuous function, then $h(x)$ attains a maximum on any closed interval $[a, b] \in \mathbb{R}$. Let $c = \max_{[a,b]} |h(x)|$

Then, $|f(x)| = |(x - \alpha)^m h(x)| = |(x - \alpha)^m||h(x)| \leq |(x - \alpha)^m|c$.

Now let $g(x) = (x - \alpha)^m.

This yields $|f(x)| = c|g(x)|$, which satisfies the definition of Big-O notation.

Thus, $f(x) = O((x - \alpha)^m)$

Question 4

$g_c(x) = x + cf(x) \implies g_c'(x) = 1 + cf'(x)$

$\alpha - x_{n+1}$ is an approximation of the error at each step.

$\alpha - x_{n+1} \approx g_c'(\alpha)(\alpha - x_n) \implies |c| < 1$ will lead to a smaller error term at each iteration and thus faster convergence.

Question 5

In [15]:
```python
#x: initial guess
#f: function
#df: derivative of function
#max_n: max number of iterations
def newton(x, f, df, max_n):
    x0 = x

    n = 0

    ep = 10**(-10)

    error = 1

    while error > ep and n <= max_n:
        print("\n Iteration: ", n, ", xn: ", x0, ", Error: ", error)
        fx0 = f(x0)
        dfx0 = df(x0)

        if(dfx0 == 0):
            print("The derivative is 0.")
            break

        x1 = x0 - fx0/dfx0

        error = abs(x1 - x0)

        x0 = x1

        n += 1

newton(1, lambda x: x**5 - 2.4*x**3 + 0.64*x**2 + 1.536*x - 0.73728,
       lambda x: 5*x**4 - 7.2*x**2 + 1.28*x + 1.536, 10)
```

```
 Iteration:  0 , xn:  1 , Error:  1

 Iteration:  1 , xn:  0.9371428571428569 , Error:  0.06285714285714306

 Iteration:  2 , xn:  0.8933040293040273 , Error:  0.04383882783882964

 Iteration:  3 , xn:  0.8631001953922838 , Error:  0.030203833911743527

 Iteration:  4 , xn:  0.8424870997854577 , Error:  0.020613095606826115

 Iteration:  5 , xn:  0.8285184471959752 , Error:  0.013968652589482433

 Iteration:  6 , xn:  0.8191005672521435 , Error:  0.009417879943831697

 Iteration:  7 , xn:  0.8127736132321934 , Error:  0.006326954019950093

 Iteration:  8 , xn:  0.8085336806711613 , Error:  0.004239932561032167

 Iteration:  9 , xn:  0.8056971548348056 , Error:  0.00283652583635563

 Iteration:  10 , xn:  0.8038016925795347 , Error:  0.00189546225527093
```

From Ch 3.5, $\lambda = \frac{m-1}{m}$ and $\alpha - x_n \approx \lambda(\alpha - x_{n-1})$

Combining these two results, we get $\lambda \approx \lambda_n \equiv \frac{x_n - x_{n-1}}{x_{n-1} - x_{n-2}}$

Using Iteration 10:

$\lambda \approx \lambda_{10} \equiv \frac{x_{10} - x_9}{x_9 - x_8} \approx 0.668233735 \approx \frac{2}{3}$.

Thus, we can take the second derivative of $f(x)$ and still preserve α as a root.

$f''(x) = 20x^3 - 14.4x + 1.28$

In the case of $f''(x)$, α is a simple root.

In order to run Newton's Method on $f''(x)$, we must also find $f'''(x)$:

$f'''(x) = 60x^2 - 14.4$

In [18]:
```
newton(1, lambda x: 20*x**3 - 14.4*x + 1.28,
       lambda x: 60*x**2 - 14.4, 10)
```

```
Iteration:  0 , xn:  1 , Error:  1

Iteration:  1 , xn:  0.8491228070175438 , Error:  0.15087719298245617

Iteration:  2 , xn:  0.80417759696571 , Error:  0.04494521005183383

Iteration:  3 , xn:  0.800034448988395 , Error:  0.004143147977314965

Iteration:  4 , xn:  0.8000000023732067 , Error:  3.4446615188366e-05

Iteration:  5 , xn:  0.8 , Error:  2.373206631212099e-09
```