

**Instructions:**

- Your submission will consist of six files (and nothing else):
    - `bisection.m`
    - `newton.m`
    - `findbracket.m`
    - `newtonbisection.m`
    - `results1.txt`
    - `results2.txt`
  - **Very Important:** Create a single compressed (.zip) folder with these files. Name it `LastNameFirstNameProject`, e.g. `SchangTomProject.zip`
- 

In this assignment, we will address two issues with the Bisection method and Newton's method:

- Finding an interval  $[a, b]$  for the Bisection method, with  $f(a)$  and  $f(b)$  having different signs.
- Combining the excellent convergence properties of Newton's method with the guaranteed root-finding (robustness) of the Bisection method.

1a (Bisection): Implement the Bisection Method for Root Finding.

```
function p = bisection(f, a, b, tol)
```

- `p`: approximation to the root
- `f`: function handle
- `a`: left endpoint of initial interval
- `b`: right endpoint of initial interval
- `tol`: absolute error tolerance for root

In particular, your code should return an error if `f(a)` and `f(b)` have the same sign, in which case the intermediate value theorem does not guarantee bisection can successfully find a root.

1b (Newton): Implement Newton's Method for Root Finding

```
function p = newton(f, df, p0, tol)
```

- `p`: approximation to the root
- `f`: function handle
- `df`: function handle of derivative
- `p0`: initial guess
- `tol`: absolute error tolerance for root

2. Implement a MATLAB function `findbracket` with signature

```
function [a, b] = findbracket(f, x0)
```

which finds an interval  $[a, b]$  around  $x_0$  such that  $f(a)f(b) < 0$  (i.e.  $f(a)$  and  $f(b)$  have opposite signs) according to the following method:

1. Set  $a = b = x_0$  and  $dx = 0.001$
2. Set  $a = a - dx$ . If  $f(a)f(b) < 0$ , terminate.
3. Set  $b = b + dx$ . If  $f(a)f(b) < 0$ , terminate.
4. Multiply  $dx$  by 2 and repeat from step 2.

3. Implement a MATLAB function `newtonbisection` with signature

```
function p = newtonbisection(f, df, a, b, tol)
```

combining Newton's method and the Bisection method according to the following strategy:

1. Start with  $p = a$
2. Attempt a Newton step  $p = p - \frac{f(p)}{f'(p)}$
3. If  $p$  is outside of  $[a, b]$ , set  $p = \frac{a + b}{2}$
4. If  $f(p)f(b) < 0$ , set  $a = p$ , otherwise set  $b = p$ .
5. Terminate if  $|f(p)| < \text{tol}$
6. Repeat from step 2.

This function will be like a combination of `newton` and `bisection`.

4. Run your function `newtonbisection` using  $f(x) = \sin(x) - e^{-x}$  on the interval  $[1.9, 30]$ :

```
f = @(x) sin(x) - exp(-x);
df = @(x) cos(x) + exp(-x);
x = newtonbisection(f, df, 1.9, 30, 1e-8)
```

Present the result in a table showing for each iteration the method used (Newton or Bisection),  $a$ ,  $b$ ,  $p$ , and  $f(p)$ . Save this as `results1.txt`. You don't need to do anything fancy or write a function to get the requested data, just temporarily print/display some of the results and format them in a text file.

*Hint:* To generate the table, it is easiest to print out the needed information (i.e. the method used,  $a$ ,  $b$ ,  $p$ , and  $f(p)$ ) each iteration inside the function `newtonbisection` using `fprintf` or `disp`. You can then comment out these print statements in part 5.

5. Use your combined `findbracket` and `newtonbisection` to solve for the roots of  $f(x) = \sin(x) - e^{-x}$  with  $x_0 = -3, -2, -1, \dots, 9, 10$ :

```
f = @(x) sin(x) - exp(-x);
df = @(x) cos(x) + exp(-x);
for x0 = -3:10
    [a, b] = findbracket(f, x0);
    x = newtonbisection(f, df, a, b, 1e-8);
    disp([x0, a, b, x])
end
```

Present your results in a table showing  $x_0$ ,  $a$ ,  $b$ , and  $x$ . Save this as `results2.txt`.

*Note:* If you added some sort of print statement to your `newtonbisection` function for part 4, you probably want to comment them out for part 5.