MECH 221 Computer Lab 3

Root Finding with Newton's Method

Instructions

Write a function called `newton` which implements Newton's method for finding approximate roots of equations. The function should have the following properties:

- ☐ `newton` has 5 input parameters `f`, `Df`, `x0`, `tolerance` and `max_steps` where:
  - ☐ `f` is a function handle representing a real-valued function $f(x)$
  - ☐ `Df` is a function handle representing the derivative $f'(x)$
  - ☐ `x0` is a starting point near a root of $f(x)$
  - ☐ `tolerance` is the desired accuracy of the method (ie. the program ends when it has found a value c satisfying $|f(c)| <$ `tolerance`)
  - ☐ `max_steps` is the maximum number of iterations before forcing the algorithm to stop
- ☐ `newton` returns 3 outputs `root`, `num_steps` and `total_time` where:
  - ☐ `root` is a value c which satisfies $|f(c)| <$ `tolerance` (or it is set to `NaN` if no root is found)
  - ☐ `num_steps` is the number of iterations of the algorithm performed in the process of finding an approximate root (or it is set to `NaN` if no root is found)
  - ☐ `total_time` is the time elapsed during the implementation of Newton's method (or it is set to `NaN` if no root is found)
- ☐ `newton` displays a summary when complete. For example, if $f(x) = x^2 - 2$ then:

```
>> [r,s,t] = newton(@(x) x^2 - 2,@(x) 2*x,1.4,0.00000001,10);
Found root 1.41421356421356 after 2 iterations in 0.00042557 seconds.
```

  Another example, if $f(x) = x^2 - 7$ then:

```
>> f = @(x) x^2 - 7; Df = @(x) 2*x;
>> [r,n,t] = newton(f,Df,2,10^(-7),10);
Found root 2.64575131106469 after 4 iterations in 0.00014795 seconds.
```

  However, the summary should indicate that no root was found if the program terminates when the number of iterations exceeded `max_steps`. For example:

```
>> f = @(x) cos(x) - x; Df = @(x) -sin(x) - 1;
>> [r,n,t] = newton(f,Df,50,10^(-7),10);
Program terminated after 10 iterations.  No root found.
```

Finally, the summary should indicate that no root was found if the program terminates if $f'(x_n) = 0$. For example:

```
>> f = @(x) 1 - x^2; Df = @(x) -2*x;
>> [r,n,t] = newton(f,Df,0,10^(-7),10);
Program terminated because f'(x)=0.  No root found.
```

In these last two degenerate cases, the function returns `NaN` for all three output parameters `root`, `num_steps` and `total_time`.

- ☐ Write a description of your function (including its input and output parameters) in the lines right below the first line which contains the `function` keyword. Include your name and student number.
- ☐ Do **not** use the MATLAB function `fzero`.

When you have completed each item above, submit your M-file (called `newton.m`) to Connect.

Hints

Given a function $f(x)$, Newton's method starts with an initial guess x_0 for a root of $f(x)$ (ie. a solution of the equation $f(x) = 0$) and then computes a sequence of approximations of a root of $f(x)$ by the recursive formula

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

If the initial value x_0 is chosen near an actual root of $f(x)$, then the sequence converges quickly to an actual root. There are several ways to write an implementation of Newton's method. Here are a few hints:

1. It is possible to use either a `for` loop or a `while` loop to implement Newton's method.
2. A `for` loop would execute for values n from 1 to `max_steps` but with a `break` if the condition $|f(x_n)| <$ `tolerance` is met.
3. A `while` loop would continue to execute if $|f(x_n)| \geq$ `tolerance` and the number of iterations of Newton's method is less than `max_steps`.
4. Use `tic` and `toc` at the appropriate places in your code and set `total_time` to the total elapsed time.
5. Use `disp` to display a summary of the process in the command window. If the process was stopped by the number of iterations exceeding `max_steps`, the summary should indicate no root was found and then set `root` to `NaN` (this represents *not a number* in MATLAB).