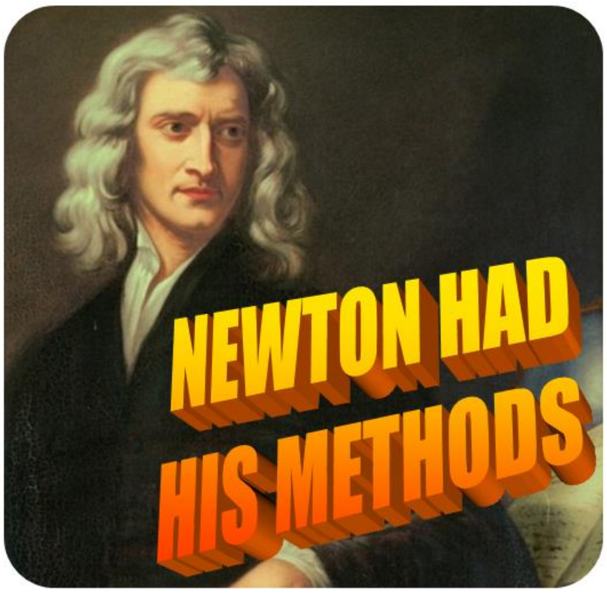


Self Study 5- Newton Had His Methods

Introduction

In this self study exercise you're going to use Newton Raphson's Method to solve a problem. In doing so, you will create a user-defined function that will do the calculations for you, called from the main program you will also write.

The Scenario

The diameter of a bicycle wheel is 710mm and the valve is set at the lowest point of the wheel. If the wheel is rolled forward until the valve is N mm away from its original position, through what angle (radians) has the wheel turned?

Assuming the valve is 300mm from the centre of the wheel the equation to be solved is:

$$355x - 300\ \sin x = N$$

where x is the angle turned through in radians.

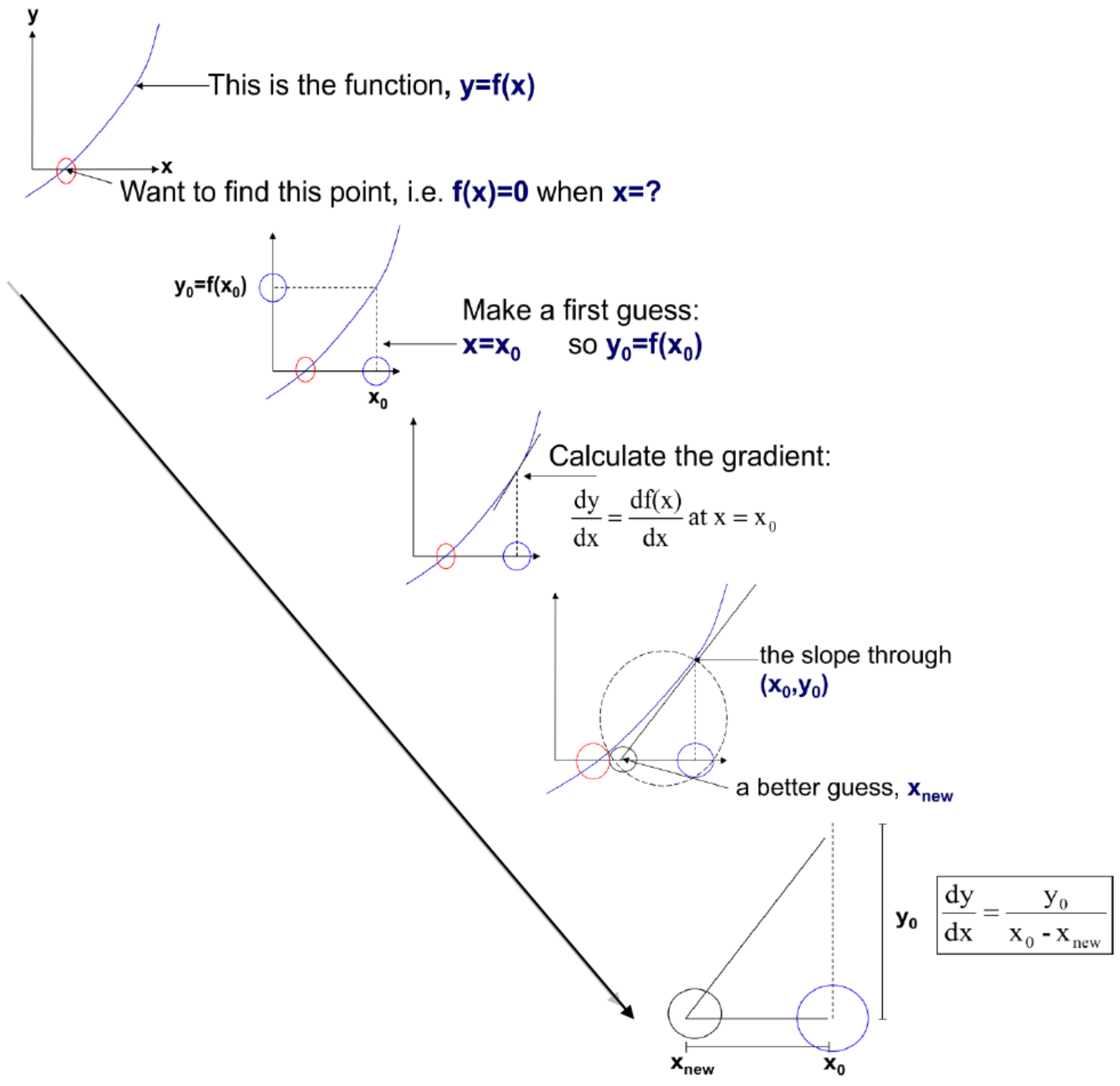
A Brief Overview of Newton's Method

Newton's Method or Newton Raphson Method is an iterative solver for finding the solution to an equation. A first guess is used that then produces a better approximation, the process is repeated and a new better approximation is derived and so on... We use it when we can't explicitly find the solution to an equation. The method is explained below in figure 1. From this we following equation:

$$x_{new} = x_0 - \frac{y_0}{\left(\frac{dy}{dx}\right)}$$

$$x_{new} = x_0 - \frac{f(x_0)}{f'(x_0)}$$

This can then be repeated replacing x_0 with x_{new} and a new x_{new} calculated. The solution can be found when x_{new} has converged on the solution, usually defined as when the value of x_{new} remains the same between iterations to a specific level of accuracy e.g. the same to 6 decimal places.



y
x
This is the function, y=f(x)
Want to find this point, i.e. f(x)=0 when x=?
$y_0=f(x_0)$
Make a first guess:
$x=x_0$ so $y_0=f(x_0)$
x_0
Calculate the gradient:
$\frac{dy}{dx}=\frac{df(x)}{dx}$ at $x=x_0$
the slope through (x_0,y_0)
a better guess, x_{new}
y_0
$\frac{dy}{dx}=\frac{y_0}{x_0-x_{new}}$
x_{new}
x_0

Figure 1:Newton's Method derivation

Instructions

Our function is:

$$f(x)=355x-300\sin x-N=0$$

We want to solve it when $N=100$ but so that we have more versatility we will be using a user defined function for this task and `N` will be the second input argument and the first will be `x0`. Your program will output a value of x for when f(x) = 0 to 6 decimal place.

The following variables will be used within your code:

```
x0
n
xold
xnew
change
solution
convergence
step
x_final
```

1. Now open the editor and save a new .m file as `bike_problem.m`. As always, you should place a comment describing the function of the program at the start. Also clear the command window and workspace programatically.

2. The next step is to initialise some variables. First create two row vectors, initialised to 15 values of zero, one called `convergence` and one called `solution`. You can use the zeros function for this (QG 1.4).

3. Now create initialise some scalar values. `x0, step, change` and `n` and set them to 5, 1, 100 and 100 respectively. Place the value of `x0` in the first column of row vector `solution`.

4. Create a variable called `xold` and set it to `x0`.

5. Now you are going to use a `while` loop that will run until our convergence criteria is met. The `while` loop will include a call to our user-defined function that we will create shortly. Your `while` loop should run until the difference between `xold` and `xnew` is less than 0.000001. We will assign the difference to the variable `change` which we initialised to a value of 100 in step 3. Write the start of the `while` loop with a conditional statement including the variable `change`. (See QG 7)

6. Inside your `while` loop, increment `step` by 1.

7. Now you are going to create your user-defined function. This function will be passed `xold` and the value `n` and will return a value for `xnew,` that is next best guess at a solution. Open a new script file by going to File -> New -> Script. Save this new file as `newton_method.m` in your working directory.

8. To define this file as a function the first line of code should be of the format:

```
function output = function_name(inputs)
```

In our case, the output will be a variable `xnew`, our function_name is `newton_method` and our inputs are `xold` and `n` giving: `function xnew = newton_method(xold, n).` Below the first line, add a comment that will form the part of the help information of the function.

9. Within this function file you should implement the following equation:

$$x_{new} = x_0 - \frac{f(x_0)}{f'(x_0)}$$

where x_0 is `xold`, and $f(x_0)$ is `355*xold-300*sin(xold)-N.` You can work out what $f'(x_0)$ will be by finding the derivative of $f(x_0)$. The user-defined function will return the value `xnew`. Type the word `end` at the bottom of the function file to define the end of the function. Save `newton_method.m`.

10. Go back to `bike_problem.m`. Now call the `newton_method.m` function from within the `while` loop you defined. The script used to call it should look like the first line of code in the function file (see step 8), without the word `function`.

11. Still within the `while` loop we want to calculate `change`. To do this we want the absolute difference between `xnew` and `xold`. Remember the `abs` function you used in self-study 1.

12. Place the value `change` in to the vector `convergence` you defined in step 2 at the index `step`. (See QG 2.13)

13. Place the value `xnew` into the vector `solution` defined in step 2 at the position `step`.

14. Assign the value of `xnew` to the variable `xold` and end the `while` loop. This loop should now iterate, calling the `newton_method` function each time and will stop when `change` is less than 0.000001. After loop has ended, `xnew` will contain our solution.

15. Now we want to reduce the size of the vectors `convergence` and `solution` so they just contain the data from index 1 up to the value at index `step`.

16. Now to visualise some of the data. We want to show how our guess converged to the correct answer. Plot `solution` against a vector called `iteration` you can create, the latter should run from 0 to `step-1`.

Now open a figure window. Use `figure(1)` in your code. This will open a figure window and set figure number 1 to the active figure. Produce a plot plotting the `solution` against `iteration`. Label the plot correctly using:

```
xlabel('label')
ylabel('label')
```

17. Finally set create a variable `x_final` that contains the last value in `solution` or `xnew`. Save both .m files and run in MATLAB. Once happy, submit via scriptcheck following the instructions below.

18. To convert your code into one that ScriptCheck can run (because you can't submit multiple files) you will need to copy and paste your user defined functions into the bottom of your main code so that newton method will become a local function. To do this, copy and paste your bike_problem.m file into the text box on scriptcheck. Below this type the word `end.` Next copy your function file newton_method.m and paste this on the bottom of the file after the word end. The general structure should look like this:

```
Your bike_problem.m code

end

Your newton_method.m code.
```

Submit your code to ScriptCheck and hopefully get 100%! https://scriptcheck.leeds.ac.uk