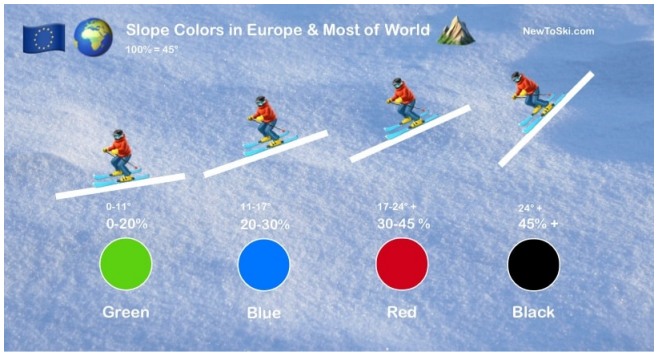
Nonlinear equations: Newton's method

MATH2070: Numerical Methods in Scientific Computing I

Location: http://people.sc.fsu.edu/~jburkardt/classes/math2070_2019/nonlinear_newton/nonlinear_newton.pdf

---



Even in zero visibility, a skier knows which way is downhill!

Rootfinding (Newton version)

Given $f(x)$, with derivative $f'(x)$, and a starting point x, estimate a root x^* for which $f(x^*) = 0$?

1 Newton's linear model for f(x)

Suppose that we are looking for a root x^* of $f(x) = 0$, and that $f(x)$ is at least twice continuously differentiable. Then, at a point x sufficiently close to x^*, we can write:

$$f(x*) = f(x) + f'(x) * (x^* - x) + O(x^* - x)^2$$

Noting that $f(x^*) = 0$, and dropping the $O(x^* - x)^2$ terms, we arrive at Newton's estimate for the location of the root:

$$x* \approx x - \frac{f(x)}{f'(x)}$$

Thus, to estimate the root of a function, we try to find a starting point that we hope is already close, and we must be prepared to evaluate the derivative of the function as part of the iteration.

We hope that, once the iteration is close to the root, the norm of the function will decrease on every step, and the size of the step $|x^i - x^{i-1}|$ will decrease quadratically, as it becomes similar to the size of the actual (but unknowable) error $|x^* - x^{i-1}|$.

2 Newton's method pseudocode

The pseudocode for Newton's method is very similar to that for previous algorithms. However, we only pass in a single starting value `x`, and along with the function `f(x)` we now must include the derivative function `fp(x)`.

```
## newton_pseudocode.txt

function newton ( )

Input: f(), fp(), x, xtol, ftol, itmax

it = 0
dx = 0
alpha = 0

Begin loop

  it = it + 1

  Set xold to x, and set  x to x - f(x) / fp(x)

  Set dx_old to dx, and set dx = | x - xold |

  Set alpha_old to alpha
  if dx_old is not 0, set alpha to dx / dx_old
    if alpha_old is not 0, set r = log ( alpha ) / log ( alpha_old )

  if x - xold is less than xtol and |f(x)| is less than ftol, success
  if it > itmax, failure

End loop

Output: values of updated x and it
```

Listing 1: Pseudocode for Newton's method

3 Example: Kepler's Equation

Our example of a Kepler function has the form:

$$f(x) = 5 - x + 2 * \sin(x)$$

so the corresponding derivative function would be:

$$f'(x) = -1 + 2 * \cos(x)$$

To use Newton's method, in addition to our file *kepler.m*, we would need to supply an additional file *kepler_fp.m* of the form:

```
function value = kepler_fp ( x )

  value = - 1.0 + 2.0 * cos ( x );

  return
end
```

Listing 2: kepler_fp.m

Let us apply our Newton code to this problem. We can call our MATLAB Newton solver with this script:

```
x = 3.0;
xtol = 0.00000001;
ftol = 0.00000001;
itmax = 50;
```

```
5
6 [ x, it ] = newton ( @(x)kepler(x), @(x)kepler_fp(x), x, xtol, ftol, itmax );
```

Listing 3: Newton's method applied to the Kepler function

The algorithm returns after just four steps. Here are the intermediate results:

it	x	f(x)
0	3.000000000000000	2.28224
1	3.765856211130886	0.065144
2	3.790693912837725	0.00036471
3	3.790834550863825	1.19566e-08
4	3.790834555474780	-2.22045e-16

While the values $f(x)$ seem to be showing quadratic convergence, the true quadratic behavior is in the successive values of x, in which the number of correct digits seems to roughly double on each step.

4 Exercise: The jump function

As an in-class exercise, let's see what we need to do to apply Newton's method to a new example, the *jump* function, whose plot looks a little like a ski jump. The function is defined by:

$$f(x) = e^{\cos(x)} - x/2 + 2$$

Do the following:

1. Copy the file *newton.m* from the website;
2. Create a function file *jump.m*;
3. Create a derivative file *jump_fp.m*;
4. Plot $f(x)$ between 0 and 10. Use this plot to pick a good starting point x. Choose x to be an integer;
5. Write a script, or interactively issue the commands, which specify the input to the *newton()* function;
6. Report your starting point, your solution, and the number of iterations required.

As with the secant method, our best results occur if the starting point is in the region near the root where the function seems to be behaving linearly. If the starting point is further away, then Newton's method can be very slow, or wander aimless, or even diverge to infinity.

5 Example: Convergence rate for the Lambert function

Our Lambert function has the form

$$f(x) = x * e^x - 742.0657955128830$$

For Newton's method, we also need the derivative:

$$f'(x) = (x + 1) * e^x$$

We can use a starting value of $x = 3$ and call the `newton()` function to seek a root. But now, we are interesting in observing the convergence rate of the algorithm.

As we have done for previous algorithms, we can estimate the order of convergence r as we come near the root by comparing the differences between successive iterates. Newton's method should in general exhibit quadratic convergence, so the value of r should tend to 2.

If we turn on the print statements inside `newton()` which monitor the convergene rate estimate, here is what we see for a calculation with the Lambert function:

it	α	$\log(\alpha)$	r
2	0.108328	-2.2226	
3	0.991943	-0.00808959	0.00363971
4	0.988803	-0.0112604	1.39197
5	0.982003	-0.0181604	1.61276
6	0.965831	-0.0347668	1.91442
7	0.925889	-0.0770008	2.21478
8	0.83009	-0.186221	2.41843
9	0.630036	-0.461978	2.48081
10	0.331472	-1.10421	2.39018
11	0.0889238	-2.41998	2.19159
12	0.00718688	-4.9355	2.03948
13	5.11763e-05	-9.88023	2.00187

Looking at the values of α, it's not so easy to see quadratic convergence, but the $\log(\alpha)$ column makes it easier to see.

6 Exercise: The multi function

Consider the problem of finding a root of another new example, called the "multi" function. This function has the form

$$f(x) = x^4 - 6x^2 + 8x - 3$$

with derivative

$$f'(x) = 4x^3 - 12x + 8$$

- Plot the function over [-3.5,+3.0], and notice there seem to be two places where $f(x) = 0$;
- Using the starting point $x = 3$, call the `newton()` function to get a root. Note the number of iterations seems high;
- Turn on the print statement in `newton()` which reports the convergence rate r. What seems to be different for this problem?
- The actual value of the root is an integer, so if `newton()` didn't quite get there, round the result to an integer. Verify that $f(x)$ is exactly 0 at this value of x.
- Now verify that it is also true that $f'(x) = 0$. Look at the plot again to see that this is so. When a function and its derivative are both zero at a value x, what does this say about x?
- What was said in class about the behavior of Newton's method in this case?

7 No assignment for this lab!