Math 311

Numerical Methods

2.3: The Newton-Raphson Method

Solutions of Equations of One Variable

S. K. Hyde

Burden and Faires, any ed.

Winter 2025

1 Introduction

- The Newton-Raphson method is one of the most powerful and well-known methods for solving a root-finding problem ($f(x) = 0$)
- To introduce it, we will derive it from Taylor Polynomials.
- Let p be the root of $f(x) = 0$. Expand $f(x)$ in a Taylor polynomial (with $n = 2$) about $x = x_0$. So

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + f''(\xi(x))\frac{(x - x_0)^2}{2} \tag{1}$$

- Since p is the root, then $f(p) = 0$. Plugging $x = p$ into equation (1) yields

$$f(p) = 0 = f(x_0) + f'(x_0)(p - x_0) + f''(\xi(p))\frac{(p - x_0)^2}{2}$$

- If we assume that $|p - x_0|$ is small, then $|p - x_0|^2$ is negligible and

$$0 \approx f(x_0) + f'(x_0)(p - x_0)$$

- Solving for p yields:

$$p \approx x_0 - \frac{f(x_0)}{f'(x_0)}$$

- Let $p = p_{n+1}$ and $x_0 = p_n$. Then we have the modern day Newton's Method:

$$p_{n+1} = p_n - \frac{f(p_n)}{f'(p_n)}$$

- Newton's Method Graphical interpretation: We will use a Desmos.com page I have created. It is at:

`https://www.desmos.com/calculator/w7uijbttse`

Newton-Raphson Algorithm

- Input: initial approximation p_0; tolerance TOL; max number of iterations N_0.
- Output: Approximate solution p or message of failure

Step 1 for ($k = 1$ to N_0) do steps 2-4

  Step 2 set $p = p_0 - f(p_0)/f'(p_0)$;

  Step 3 if <STOPPING CONDITION>, then output(p); stop;

  Step 4 set $p_0 = p$

Step 5 output("Method failed after N_0 iterations"); stop;

Stopping Condition

- At some point in a program, the solution will be found. How do we know when it is complete? We need a stopping criterion.
- Here are some ideas for a stopping criterion:

$$|p_N - p_{N-1}| < \varepsilon \qquad \frac{|p_N - p_{N-1}|}{|p_N|} < \varepsilon \qquad |f(p_N)| < \varepsilon$$

- There are issues using ANY of these. Any of them might lead to a solution that is not correct. Look at pg 42 in 5th edition right after (2.1), (2.2), and (2.3) for a discussion of the issues.
- Also build into your algorithm a limit to the number of iterations. (This is what N_0 is for.)

2 Examples

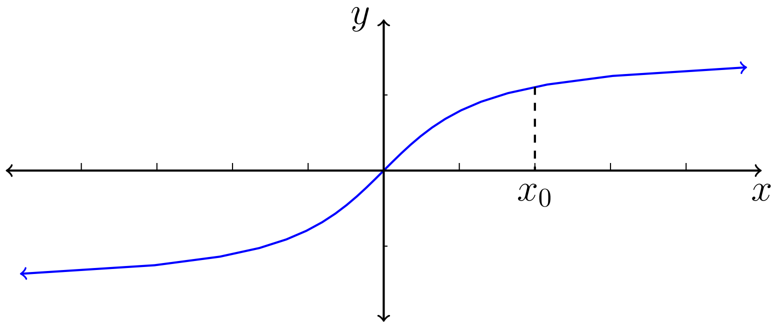
- $f(x) = \cos x - x = 0$
  - ★ $p_{n+1} = p_n - \dfrac{\cos p_n - p_n}{-\sin p_n - 1}$
  - ★ If we start with $p_0 = \frac{\pi}{4}$, then this requires only 4 iterations!
- $f(x) = x^3 + 4x^2 - 10 = 0$
  - ★ $p_{n+1} = p_n - \dfrac{p_n^3 + 4p_n^2 - 10}{3p_n^2 + 8p_n 0}$
  - ★ If we start with $p_0 = 1.5$, then this requires only 3 iterations!
- Babylonian Square Root Algorithm:
- $f(x) = x^2 - a = 0$
  - ★ $x_{n+1} = x_n - \dfrac{f(x_n)}{f'(x_n)} = x_n - \dfrac{x_n^2 - a}{2x_n} = \dfrac{2x_n^2 - x_n^2 + a}{2x_n} = \dfrac{x_n^2 + a}{2x_n} = \dfrac{1}{2}\left(x_n + \dfrac{a}{x_n}\right)$
  - ★ $x_{n+1} = \dfrac{1}{2}\left(x_n + \dfrac{a}{x_n}\right)$
  - ★ If we start with $x_0 = 1$, then this requires only 4 iterations for 11 digits of accuracy!

3 PROBLEMS!!!!

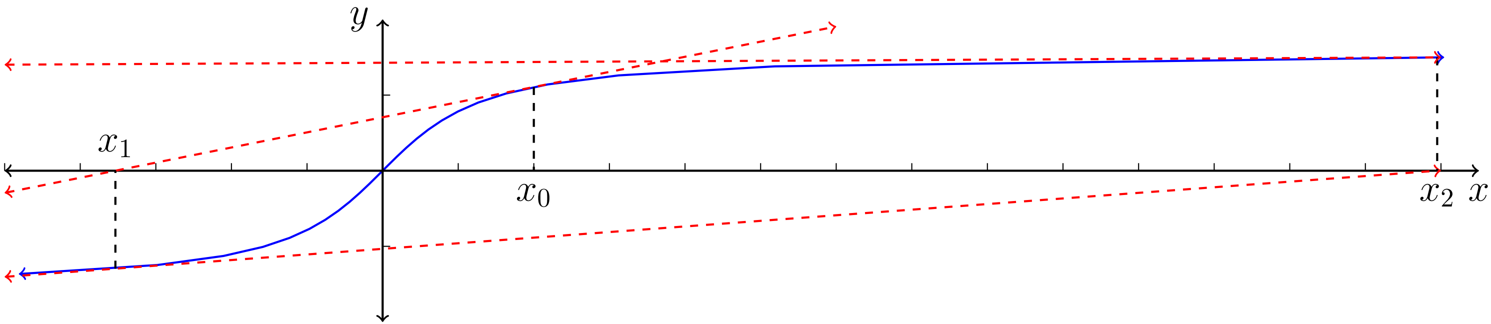
- Not everything is wonderful about Newton's Method!
- If you start too far from the solution, it might diverge. For example,
- Let's solve $\tan^{-1} x = 0$ with Newton's method. $f'(x) = \dfrac{1}{1+x^2}$
- So it follows that

$$x_{n+1} = x_n - \frac{\tan^{-1} x}{\frac{1}{1+x^2}} = x_n - (1+x^2)\tan^{-1} x$$

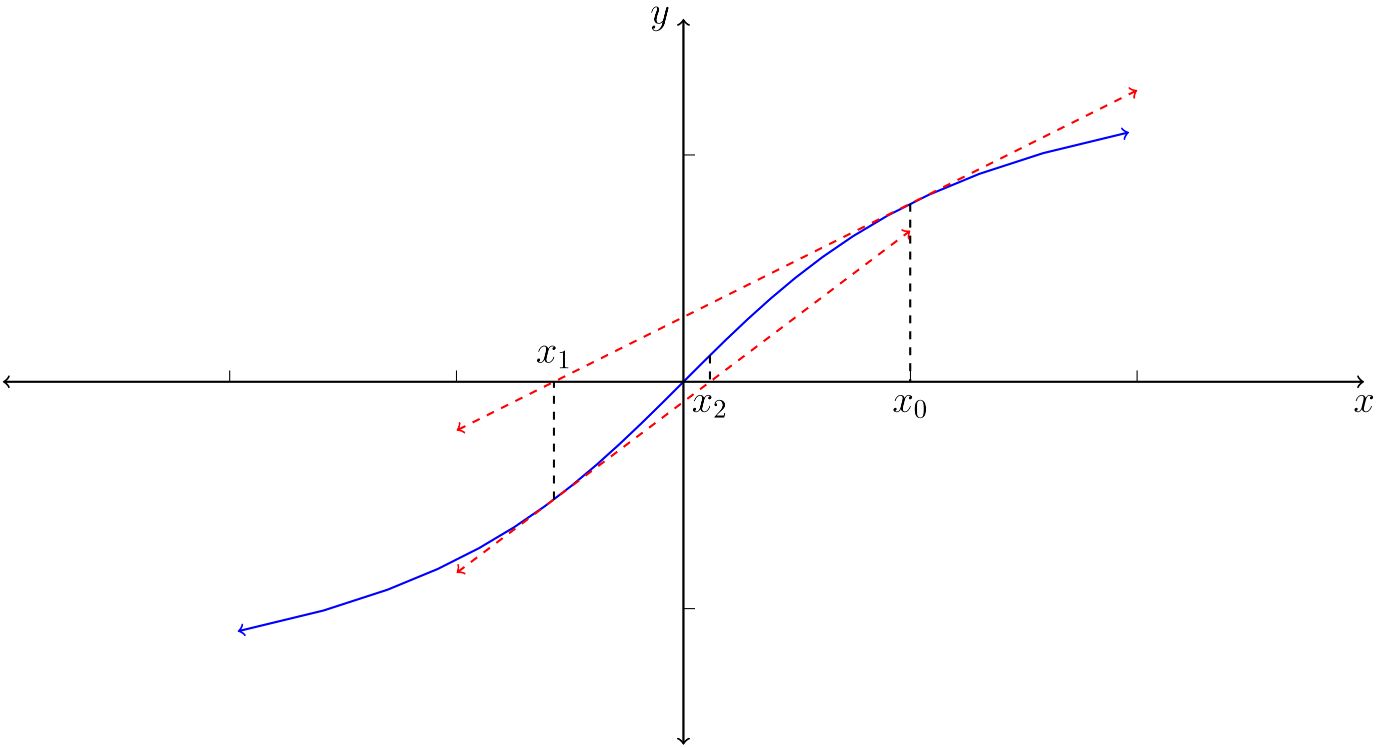
- Depending on the starting point, it MAY OR MAY NOT converge!



- For example, suppose $x_0 = 2$. In this case, the iterations proceed like this:

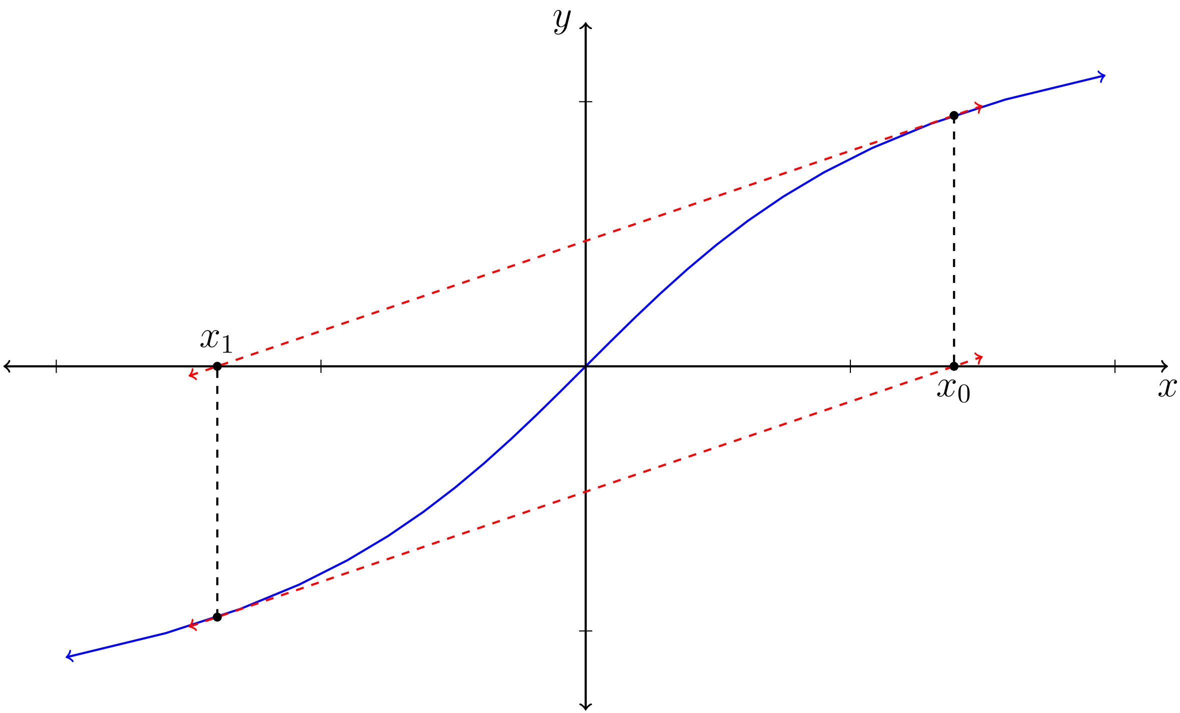


- Clearly, Newton's method keeps getting larger in absolute value for each passing iteration. It bounces back and forth from $-\infty$ to ∞.
- If we choose $x_0 = 1$ as our initial guess, then Newton's converges quite quickly.



y
x_1
x_2
x_0
x

- There is a "sweet spot" where it bounces back and forth and won't converge.
- Can you find it? (It's a good use of Newton's Method!)
- I'll give the first person who finds it (shows me how they used Newton's to solve it) extra credit!



- Note that if we used Bisection method, then it WOULD converge! Bisection is better in that case.
- However, if we have a good initial guess, then Newton's is better, because it will converge FAST!
- Let's analyze Newton's method using the theory in Section 2.2

Theorem: Convergence of Newton's (Thm 2.5)

Let $f \in C^2[a,b]$. If $p \in [a,b]$ such that $f(p) = 0$ and $f'(p) \neq 0$, then there exists a $\delta > 0$ such that Newton's Method generates a sequence $\{p_n\}_{n=1}^{\infty}$ converging to p for any initial approximation $p_0 \in [p-\delta, p+\delta]$

Proof. See the book for a full proof. Since Newton's method is a fixed point problem, then

$$g(x) = x - \frac{f(x)}{f'(x)}$$

The key to convergence (and speed) was to find $k \in (0,1)$ such that

$$|g'(x)| \leqslant k < 1.$$

The derivative of g is

$$g'(x) = \frac{d}{dx}\left(x - \frac{f(x)}{f'(x)}\right) = 1 - \frac{f'(x)f'(x) - f(x)f''(x)}{[f'(x)]^2} = 1 - 1 + \frac{f(x)f''(x)}{[f'(x)]^2}$$
$$g'(x) = \frac{f(x)f''(x)}{[f'(x)]^2}$$

Note that since $f(p) = 0$, then $g'(p) = 0$ as well! Note that NEAR p (the solution), g' is close to zero!! This should converge really fast! How fast? We'll analyze it next section. □

- A major difficulty of Newton's Method is that you HAVE to have know the derivative.
- It is either impossible or it's really complicated.
- For example, suppose that
$$f(x) = \frac{x^2 \cos(xe^x)}{\log(\sin x)}.$$
- Finding the derivative is possible, but difficult.
- We can circumvent this problem by approximating the derivative!

- A well known approximation of $f'(x)$ is

$$f'(x_0) = \lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0}$$

- Let's use the last two iterations to figure an estimate for the slope. So let $x_0 = p_{n-2}$ and $x = p_{n-1}$. Then the modified method is

$$p_n = p_{n-1} - \frac{f(p_{n-1})}{\left[\dfrac{f(p_{n-1}) - f(p_{n-2})}{p_{n-1} - p_{n-2}}\right]} \qquad \text{or} \qquad p_n = p_{n-1} - \frac{f(p_{n-1})(p_{n-1} - p_{n-2})}{f(p_{n-1}) - f(p_{n-2})}$$

The Secant Method

Define
$$p_n = p_{n-1} - \frac{f(p_{n-1})(p_{n-1} - p_{n-2})}{f(p_{n-1}) - f(p_{n-2})}$$

Starting with TWO initial guesses, p_0, and p_1, then as long as the initial guesses are close enough to the solution, then the sequence generated by p_n will converge to the solution p such that $f(p) = 0$.

- The Secant method suffers from the same problems of Newton's method.

- If the initial guesses are not chosen appropriately, then p_n will not converge to the solution.

Secant Algorithm

- Input: initial approxs p_0, p_1; tolerance TOL; max number of iterations N_0
- Output: Approximate solution p or message of failure

Step 1 set $q_0 = f(p_0)$;
$q_1 = f(p_1)$;

Step 2 for ($k = 2$ to N_0) do steps 3-5

Step 3 set $p = p_1 - q_1(p_1 - p_0)/(q_1 - q_0)$.

Step 4 if <STOPPING CONDITION>, then output(p); stop;

Step 5 set $p_0 = p_1$;
$q_0 = q_1$;
$p_1 = p$;
$q_1 = f(p)$;

Step 6 output("Method failed after N_0 iterations"); stop;

Secant Method Graphical Interpretation: We will use a Desmos.com page I have created. It is at:

https://www.desmos.com/calculator/5skucesaqu

- Compared to Newton's Method:
  - Not as fast (not so good!)
  - Does not require $f'(x)$ evaluation. (Really good!)
  - Each step only requires ONE $f(x)$ evaluation (Newton's requires ONE $f(x)$ AND ONE $f'(x)$. (Good!)
- Secant and Newton's are also used to refine the answer obtained from another method, like Bisection.

- Bisection is great in that the root is always between successive iterations (called bracketing the root). This guarantees convergence for Bisection, but isn't guaranteed with Newton's and Secant Method.
- So let's try a hybrid method that combines Secant and Bisection.
- This method is called the Method of False Position.
- I have no idea why it is called False Position.

Method of False Position Algorithm

- Input: initial approxs p_0, p_1; tolerance TOL; max number of iterations N_0
- Output: Approximate solution p or message of failure

Step 1 set $q_0 = f(p_0)$;
$q_1 = f(p_1)$;

Step 2 for ($k = 2$ to N_0) do steps 3-7

Step 3 set $p = p_1 - q_1(p_1 - p_0)/(q_1 - q_0)$.

Step 4 if <STOPPING CONDITION>, then output(p); stop;

Step 5 set $q = f(p)$;

Step 6 $\boxed{\text{If } q \cdot q_1 < 0, \text{ then set } p_0 = p_1; q_0 = q_1.}$

Step 7 set $p_1 = p$; $q_1 = q$;

Step 8 output("Method failed after N_0 iterations"); stop;

- Note the difference between Secant and False Position is Step 6. (circled above).
- While the changes to Secant guarantee convergence, it comes at a cost.
- The Worst case is <u>linear</u> convergence (just as slow as Bisection).

- This is undesirable. In fact, in my experience, False Position converges pretty slowly. The problem comes from when the root is bracketed between a region of positive or negative concavity. This makes it so the exchange is just like Bisection. Hence, the slow convergence.
- There is a modification to the algorithm called the Illinois Method that fixes the problems of False Position. We will not cover it in class. (Code for it is given on our website).