Lecture Transcript
Newton-Raphson Method

Hi and welcome to this next lecture on Data Structures and Algorithms. In this session, we 'll discuss another algorithm to find roots of an equation. So recall that we are looking at the roots of f(x)=0. We discuss the bisection method, which half the possible search range at every iteration. So bisection requires an interval [a,b] to be specified in which you expect the answer to occur. On the other hand, Newton-Raphson which we'll discuss today doesn't require an interval, it starts with a point estimate, rather than an interval estimate. So, as compared to the bisection method it requires only one initial guess of the root. This is an advantage and if the method converges it performs faster than bisection, that's because the search range is not just based on fixed pre-decided factor of half reduction in the search page. It actually varies with the shape and curvature of the function. So in practice rate depends on curvature of the function. The rate at which the interval shrink is what we are referring to here as rate. To more specifically given a function we are going to look at the curvature through its derivative. So f prime x(sub)i is the rate of change or decrease or increase of f at x(sub)i.

Now, we will canonically talk about the rate of change as rate of increase, if this f prime happens to have a negative value of v interpreted as a decrease. So what you going to do here is look at f prime and we find that well, you need to decrease, you need to move in this direction. How much should you move down? Well, you should keep moving down this direction of decrease until you find the x-intercept. This x-intercept is what we find to be a next approximation. x(sub)i +1 is the next x-intercept based on f prime x(sub)i. Let's look at theta, the angle made by this decent direction. What we expect tan theta to be is this height, which is f(x(sub)i-0) divided by x(sub)i- x(sub)i+1. So if the x(sub)i was the initial guess of the root, the idea is to find an improved estimate x(sub)i+1 as the x-intercept of the tangent to the curve at f(x(sub)i). The slope f prime x(sub)i, which is tan theta is a f(x(sub)i)-0, the 0 is basically that is what the x-intercept x(sub)i+1 has as such its y coordinate. So, f(x(sub)i) divided by (x(sub)i)-(x(sub)i+1), so this re- arrangement gives you (x(sub)i+1) as (x(sub)i) minus ratio of the function to its slope at (x(sub)i). Idea is to keep repeating this process till you get closer to the root within some desirable tolerance as find it out earlier, you grow, you are moving based on the rate of change of the function.

The function f could have small rate of change. So small rate of change will means small values of f prime and if f prime is small because f prime sits in the denominator. This will mean that the reciprocal will be large and therefore lead to large steps. On the other hand, the function could have a very large rate of change. In which case, f prime is large and this will mean that the reciprocal of large f prime will lead to small steps. This is actually quite initiative. This function is changing very slowly you might want to move fast. The function is changing too rapidly you

might want to move very slowly basically you don't want to miss the root in the second case and you don't want to take too long to get to the root in the first case. So, here is the overall algorithm you are going to have certain max number of iteration and certain tolerance limit, you are going to evaluate f prime x at every step and until a point that f prime x becomes very small which means the rate of change is negligible what you want is f prime x to be ideally 0 no rate of change as a necessary condition you keep finding the updates x(sub)i + 1. What are some issues? well, first problem is you need a closed form expression for the derivative.

So, programmatically, finding the derivative of a function is challenging. So if the function f has a known form you could have a closed form expression for the derivative. If you need to do this programmatically you would also land up with certain numerical errors. There are also one some classical cases when the function fails to converge to the root. The first example is overshooting. So it is possible that function where to cross a root. However, it's possible that subsequently, the function has a very different shape. So, if your x(sub)i initially were here, you suppose move to x(sub)i + 1 and landed up on the other side due to some reason. It's possible that you overshoot so much that x(sub)i + 1 is actually on this side. This can happen if the curve that we show here is so small that this intercept lands up getting you to a point x(sub)i + 1 on the other side. There are also other possibilities, your f prime itself quite might very small in which case you land up dividing by 0. Division by zero is when derivative x is closed to 0. You can have a wrong initial estimate that is x(sub)i which is here. For example, the x(sub)i could be here in which case you make a move in this direction which might never get you to 0. So this is the example of wrong initial estimate and finally one might oscillate. So you might have an x(sub)i which is reasonably initialize somewhere here, but it's possible that you oscillate between two iterates on either side of the x-axis.

Thank you.