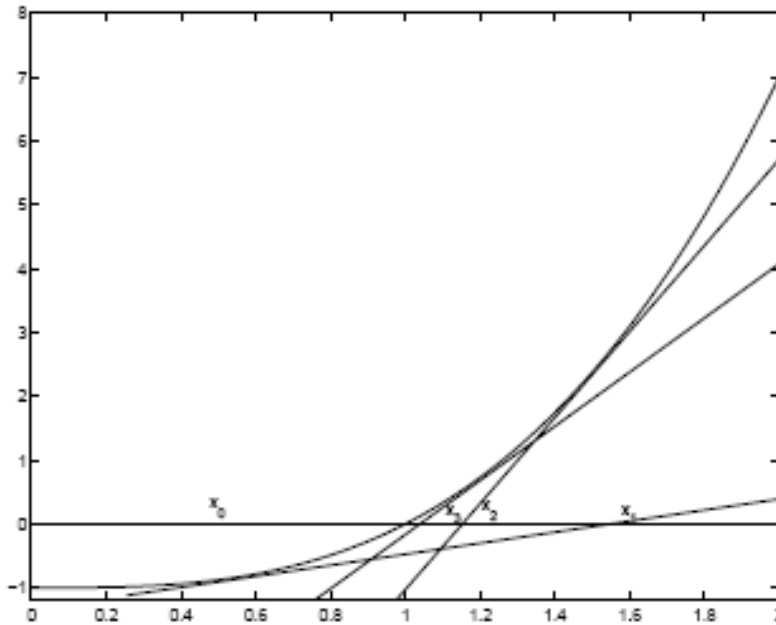


## Finding Roots: Newton's Method

### 1. Newton's Method

Newton's method involves finding the tangent line to a curve and then finding out where that tangent line has a root.



The equation of the tangent line to a function  $f(x)$  at  $x_0$  is given by

$$y - f(x_0) = f'(x_0)(x - x_0)$$

If we let  $y = 0$  (since this means the tangent line has a root) and solve for  $x$  and call it  $x_1$ , we get

$$x = x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

this is our new starting point to find the next approximation of  $x^*$ .

So in general the Newton's method had the following form:

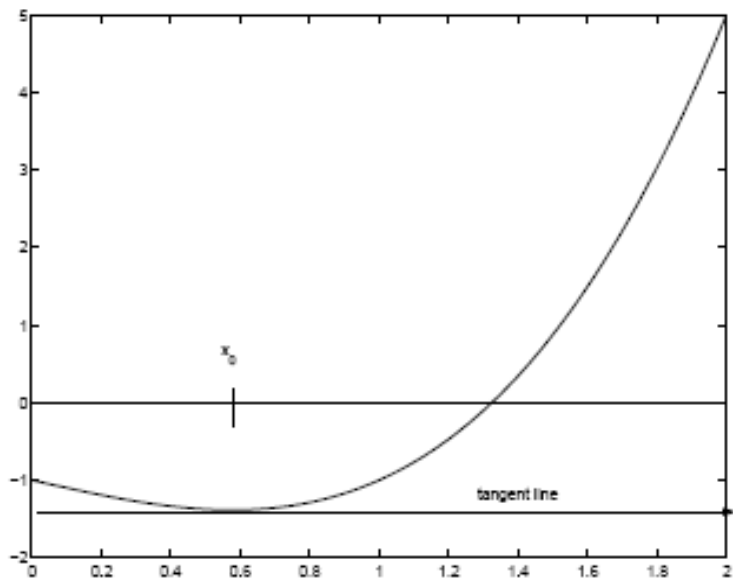
$$x_n = x_{n-1} + \Delta x$$

where

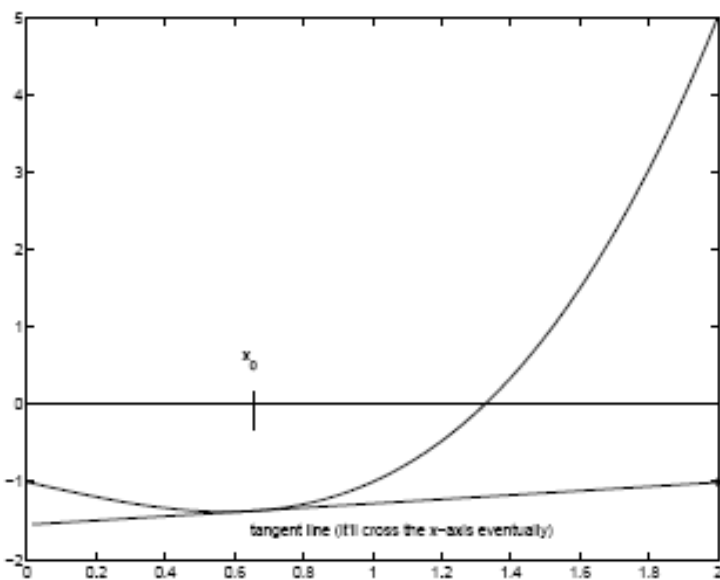
$$\Delta x = -\frac{f(x_{n-1})}{f'(x_{n-1})}$$

### Advantages/Disadvantages

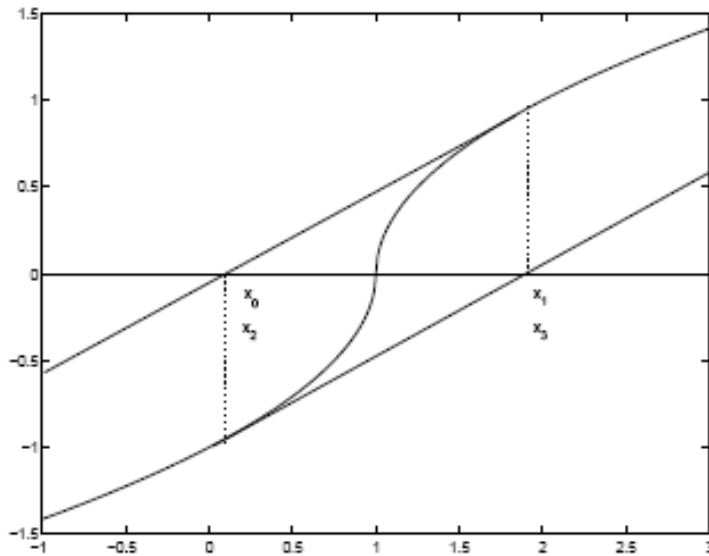
- + Newton's Method is the fastest method of the three we will discuss.
- Newton's Method only converges if you start sufficiently close to the root.
- We have to have formulas for both  $f(x)$  and  $f'(x)$ .
- If  $f'(x) = 0$  at any point in the iteration, the method fails



+/- If  $f'(x)$  is small, then your estimate goes flying off. However, it still converges, but it takes a while.



- If the function oscillates, then Newton's method will never converge, because of the vertical tangent ( $f'(r)$  does not exist)



+/- Newton's Method may not converge to the root you expect it to.

### ***Newton Algorithm:***

Inputs: fun, x0, xtol, ftol

x = x0

max\_it = k % here k is the maximum iterations allowed  
iter = 0

while iter < max\_it  
    iter = iter + 1

        f = f(x)

        df = f'(x)

        dx = -f/df

        xnew = x + dx

        fnew = f(xnew)

        if  $|x_{new} - x| < xtol$  or  $|f_{new}| < ftol$

            break

        end

        x = xnew

end

r = xnew

Example:

Use my version of newton.m to find an approximation to the root of  $y = e^{-x} + x - 2$ .

First, plot the function to get an initial guess to the root.

Notice that in order to use **Newton's Method** we have to redefine our function m-file so that it returns both the function and its derivative.

```
function [y,yprime] = ch6ex1fun(x)

% evaluates the function y = e^-x + x - 2
% and its derivative yprime = -e^-x + 1 for a vector x.

y = exp(-x)+x-2;
yprime = -exp(-x) + 1;
```

Then, in the workspace

```
>> x0 = 3; % this is the initial guess
>> r = newton('ch6ex2fun',x0)

r =

1.84140566043696
```