Newton Raphson's Method for finding a root

This method is applicable for a continuous and differentiable function f(x), provided a starting approximation to the unknown root, say x_0 is given. It consists of the following steps.

1. Given x_0, find the value of f(x) corresponding to it, i.e., f (x_0), the point on the curve is (x_0 , $f(x_0)$)
2. Draw a tangent to the curve f(x) at the point (x_0 , $f(x_0)$). The slope of the tangent at (x_0 , $f(x_0)$) is $f'(x_0)$, where f'(x) is the derivative of f(x). The equation of the tangent is $y - f(x_0) = f'(x_0)(x - x_0)$
3. Find the intersection of the tangent with the x-axis (i.e., y = 0). This gives the point x, call this point x_1.

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

4. Repeating steps 1 to 3 again with x_1 gives a new value, call it x_2 and so on. In general after steps 1 to 3 have been applied for n+1 terms, the approximation to the root is given by $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$

N-R method is known to converge quickly to a root when the conditions are favourable. The value of the term $f'(x_n)$ plays a role in it.

Example for illustration: Consider the same polynomial, with roots 3 and 7, viz., $p(x) = x^2 - 10x + 21$. The result of applying Newton-Raphson is summarized below. For the same polynomial and under similar conditions, Newton-Raphson appears to be converging faster than interval-halving.

Iteration	x_n	$p(x_n)$	$p'(x_n)$	x_{n+1}
0	20.0			
1	20.0	221	30	**12.6333**
2	12.6333	54.2678	15.2667	**9.07868**
3	9.07868	12.6356	8.15735	**7.52969**
4	7.52969	2.39935	5.05939	**7.05546**
5	7.05546	0.224899	4.11091	**7.00075**
6	7.00075	0.00299263	4.0015	**7**
7	7	1.90735e-06	4	**7**

Table 4 : Root finding Newton Raphson

Application of Newton-Raphson (N-R):

Finding square roots – Let $\sqrt{a}$, the square root of a real number a be computed. $\sqrt{a}$ is a solution of the polynomial $x^2 - a = 0$. Using Newton-Raphson iterative formulation for finding a root, $x_{n+1} = x_n - \frac{p(x_n)}{p'(x_n)}$, where p'(x) is the derivative of polynomial p(x).

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

In this formulation of square root computation, we need one *, one + and one / operation on floating point numbers.

Example : To find the square root of a number, say 255.84, using N-R method, produces the following sequence of approximations.
Approx root after 1 iteration : 128.42
Approx root after 2 iterations : 65.2061
Approx root after 3 iterations : 34.5648
Approx root after 4 iterations : 20.9833
Approx root after 5 iterations : 16.5879
Approx root after 6 iterations : 16.0056
Approx root after 7 iterations : 15.995
Approx root after 8 iterations : 15.995
Square Root found is 15.995 after 8 Iterations of Newton raphson
Square Root found using sqrt() function in maths library, sqrt(255.84), is 15.995

Finding N-th root of a real : Finding square root is a special application of N-R, the general problem for finding the n-th roots of a real a, is formulated along similar lines :

$a^{1/N}$ is a solution of the polynomial $x^N - a = 0$. Using Newton-Raphson iterative formulation for finding a root, $x_{n+1}=x_n-\frac{p(x_n)}{p'(x_n)}$, where p'(x) is the derivative of polynomial p(x).

$$x_{n+1}=x_n-\frac{x_n^N-a}{Nx_n^{N-1}}=\frac{1}{N}((N-1)x_n+\frac{a}{x_n^{N-1}})$$