Roots of Functions and Polynomials

The transfer function obtained by using polynomial approximation will be represented in the symbolic form by the rational function

$$F(s) = \frac{\sum_{i=0}^{n} a_i s^i}{\sum_{i=0}^{m} b_i s^i} = \frac{N(s)}{D(s)}$$

In many applications we need to know zeros and roots of this function, so the following form is desired

$$F(s) = k \frac{\prod_{i=1}^{n} (s - z_i)}{\prod_{i=1}^{m} (s - p_i)}$$

where z_i are roots (zeros) of N(s) and p_i are roots of $D(s)$. The best know general method for finding roots is the Newton-Raphson method. This is an iterative method that uses function and its derivative values. We start with an initial point X_0 and expand the function $f(x)$ around this point:

$$f(x_1) = f(x_0) + f^{'}(x_0)(x_1 - x_0) + ...$$

If we neglect higher order terms and want to find x_1 such that $f(x_1) = 0$
then

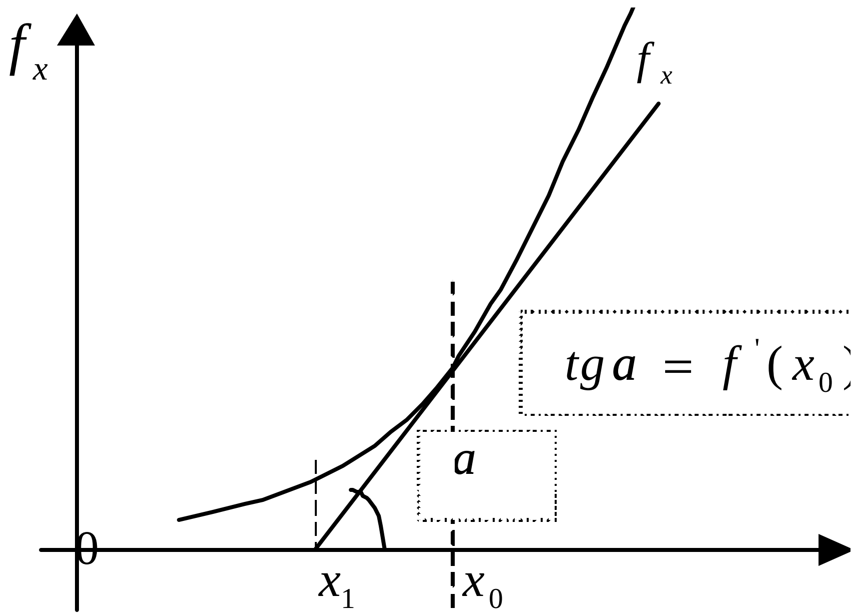
$$(x_1 - x_0) f^{'}(x_0) \cong -f(x_0)$$

and

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

Because this is an approximated solution we have to iterate.

The iterations can be illustrated as shown on the following figure



The Newton-Raphson method is defined by the following algorithm:

1. Set k=0 and select x_0
2. Calculate $\Delta x_k = -f(x_k)/f'(x_k)$
3. Calculate $x_{k+1} = x_k + \Delta x_k$
4. If $|\Delta x_k < e|$ stop , else set k=k+1 and go to step2.

The method converges quickly to the solution,

provided that initial point x_0 was correctly chosen.

The method which always converges to a root of polynomial $P_0(x)$ is called Laguerre's method and is defined by

$$x^{k+1} = x^k - \frac{np_n(x^k)}{p_n'(x^k) \pm \sqrt{H^k}}$$

where

$$H^k = (n-1)\left[(n-1)\left\{P_n'\left(x^k\right)\right\}^2 - nP_n\left(x^k\right)P_n''\left(x^k\right)\right]$$

and $P_n(x)$ is a polynomial of degree n. It is advantageous to first compute roots with small absolute values, thus initial estimate should start with $x^0 = 0$.

Example (Laguerre's method)

Find one root of the polynomial

$$P_5 = 8x^5 + 12x^4 + 14x^3 + 13x^2 + 6x + 1$$

We have $P_5'(x) = 40x^4 + 48x^3 + 42x^2 + 26x + 6$

And $P_5^{''}(x) = 160x^3 + 144x^2 + 84x + 26$

Start at $x^0 = 0$

Calculate $P_5(x^0) = 1$, $P_5^{'}(x^0) = 6$, $P_5^{''}(x^0) = 26$

$$H^0 = 4\left[4 \cdot 6^2 - 5 \cdot 1 \cdot 26\right] = 56, \quad \sqrt{H^0} = 7.48$$

so

$$x^1 = x^0 - \frac{5P_5(x^0)}{P_5^{'}(x^0) \pm \sqrt{H^0}} = \frac{-5 \cdot 1}{6 + 7.48} = -0.37$$

$\uparrow$

selected to have $\left|x^1 - x^0\right|$ small

Second iteration:

$P_5(x^1) = 0.02$, $P_5^{'}(x^1) = 0.45$, $P_5^{''}(x^1) = 6.5$

$$H^1 = 4\left[4 \cdot (0.45)^2 - 5 \cdot 0.02 \cdot 6.53\right] = 0.628, \quad \sqrt{H^1} = 0.79$$

$$x^2 = x^1 - \frac{5P_5(x^1)}{P_5^{'}(x^1) \pm \sqrt{H^1}} = -0.37 - \frac{-5 \cdot 0.02}{0.45 + 0.79} = -0.45$$

Third iteration:

$P_5\left(x^2\right) = 0.0012$, $P_5^{'}\left(x^2\right) = 0.0.71$, $P_5^{''}\left(x^2\right) = 2.78$

$$H^2 = 4\left[4 \cdot (0.071)^2 - 5 \cdot 0.0012 \cdot 2.78\right] = 0.014,$$

$$\sqrt{H^2} = 0.118$$

$$x^3 = x^2 - \frac{5P_5\left(x^2\right)}{P_5^{'}\left(x^2\right) \pm \sqrt{H^2}} = -0.45 - \frac{-5 \cdot 0.0012}{0.071 + 0.118} = -0.48$$

Exact solution to this problem is:

$x_1 = x_2 = x_3 = -0.5$, $x_4 = -j$, $x_5 = j$

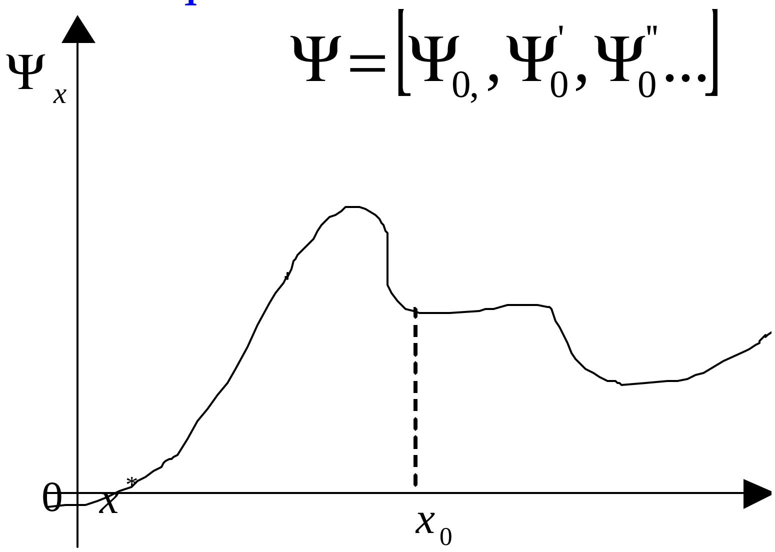
Noniterative method based on Taylor expansion

In the Newton-Raphson method the increment Δ is used to estimate the next iterate

$$\Delta = -\frac{f_0}{f_0'} \; and \quad x_1 = x_0 + \Delta$$

We can generalize this approach by using higher order derivatives to estimate the increment of the function value which brings it directly to a solution of the nonlinear equation.

$$\Psi = \left[\Psi_{0,}, \Psi_0', \Psi_0'' \ldots\right]$$



$$\Psi = \left[\Psi_0, \Psi_0', \Psi_0'', \ldots \Psi_0^k\right]$$

where $\Psi_0^i = \dfrac{d^i f(x_0)}{dx^i}$

The following algorithm uses derivatives at x_0 to find x^*

1. Set Count = 0
2. Formulate equation

$$\Psi(\Delta) = \Psi_0 + \Psi_0^{'}\Delta + \Psi_0^{''}\frac{\Delta^2}{2} + ...\Psi_0^k\frac{\Delta^k}{2} = 0$$

3. Find

$$\Delta_{count} = \min_i \left(\frac{-\Psi_0 i!}{\Psi_0^i}\right)^{\frac{1}{2}}$$ (Delta(i) in the algorithm)

- notice that a more precise estimate of delta can be found from the polynomial equation in 2.

4. Find function value and its derivatives in the (recomputed elements of the expansion)

$\Psi^i(x_0 + \Delta_{count})$ (function Psi in the algorithm)

by differentiating equation in 2. As we approach the solution Δ_{count} gets smaller and smaller.

5. Increase count by 1 replace x_0 by $(x_0 + \Delta_{count})$ and repeat 2, 3, and 4 until $\Delta_{count} < e_{ps}$
6. Solution

$$\Delta = \Delta_0(1 + \Delta_1(1 + \Delta_2(1 + ..(1 + \Delta_{count}))))$$

$$x^* = x_0 + \Delta$$

Notice that in this algorithm function and its derivatives are only calculated at the initial point x_0.

The algorithm uses $\Psi_0^i = \dfrac{d^i f(x_0)}{dx^i}$ to evaluate function and its derivatives at new point, rather than repeating system analysis to get them

The Matlab code is as follows

```
function x = solve-2(F,x0)
            % Nonlinear Equations Solver
            % Janusz Starzyk
             % F is a vector of a function and its
             % derivatives computed at x0
Psi(:)=F;
eps=le-14;
count=0;
deltak=l;
          % the main loop
while abs(deltak)>eps
count=count+l;
                     % solve for delta
           for i=l:(size(F)-l)
                     Delta(i)=(-Psi(l)/(Psi(i+l)/fact(i)))^(1/i);
           end
           deltak=min(Delta(:));
           delta(count)=deltak,
                      % precompute elements of the expansion
```

```
        for i=l:size(F)
                FnDeln(i)=Psi(i)*deltak^(i-1);
        end
                % find functions psi
        for i=l:size(F)
                Psi(i)=0;
                for j=i:size(F)
                        Psi(i)=Psi(i)+FnDeln(j)/fact(j-i);
                end
        end
end    % delta converged
del=l;
for i=count:-1:1
        del=l+delta(i)*del;
end
x=x0+del-1;

function y=fact(i)
                % this function evaluates factorial of i
y=1;
for ii=l:i
      y=y*ii;
end
```

The above functions are stored as solve-2.m and fact.m respectively