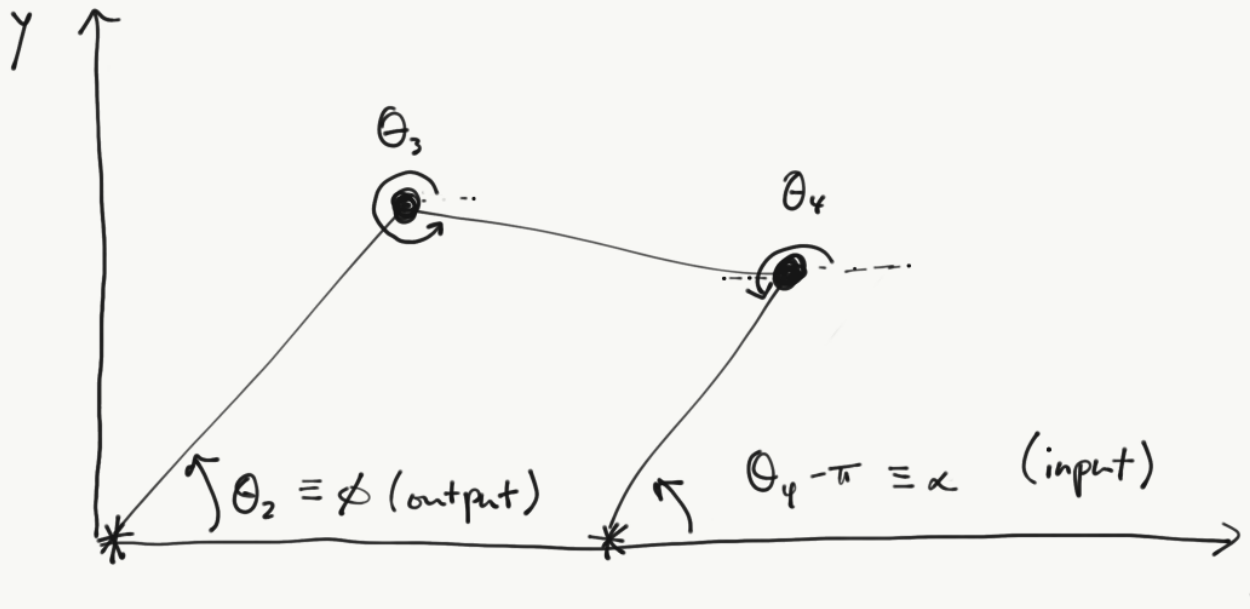
Nonlinear Equations

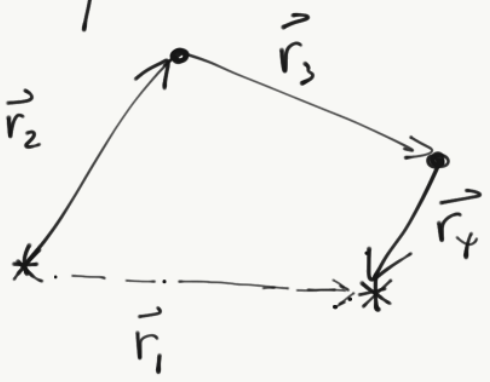
- Solutions of nonlinear equations are necessary in a wide range of applications.
- The canonical nonlinear problem is that of finding the roots of a function. Ie find

$$\{x_i\} \quad \text{s.t.} \quad f(x_i) = 0$$

- The book gives the example of a four bar linkage:



Vectorially we have:



$$\vec{r}_1 = \vec{r}_2 + \vec{r}_3 + \vec{r}_4$$

(x) $r_1 = r_2\cos\theta_2 + r_3\cos\theta_3 + r_4\cos\theta_4$

(y) $0 = r_2\sin\theta_2 + r_3\sin\theta_3 + r_4\sin\theta_4$

- Nonlinear equations can be extremely difficult to solve by hand, e.g.:

$$x - e^{-x} = 0 \qquad \text{(transcendental)}$$

- Often you just have to plot things or start guessing...

Root finding

- Root finding procedures (finding zeros of $f(x)$) generally share several basic steps:

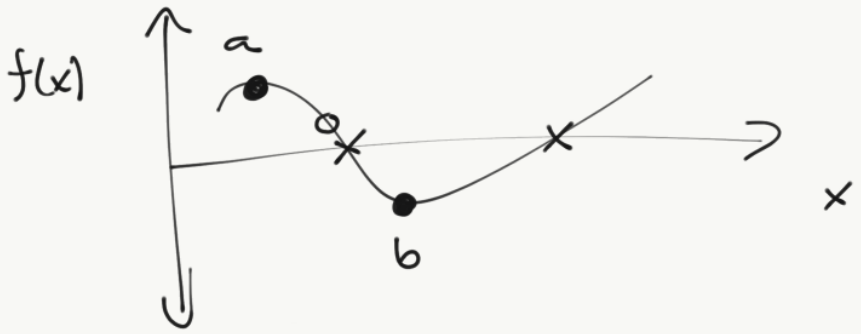
  1) bounding (restriction of solution to some interval, $a \leq x_i \leq b$)
  2) refinement (update solution to reduce error)

- Bounding
  a) graphing or search
  b) prior knowledge
  c) solution of approximate model
  d) prior solution in sequence of iterations.

- Refinement
  a) guessing
  b) "closed" domain methods
  c) open domain "

- See figure 3.6 & 3.2.4
- Root finding algorithm features
  a) max # of iterations
  b) error checking on whether $f'(x) \to 0$
  c) convergence test (x_i not changing or $f(x_i)$ small)
  d) checking that the root actually works.

- Interval halving is a straightforward way to implement a search method for root-finding
- If the value of the function a two different points changes sign, & that function is continuous than the root lies between those points:



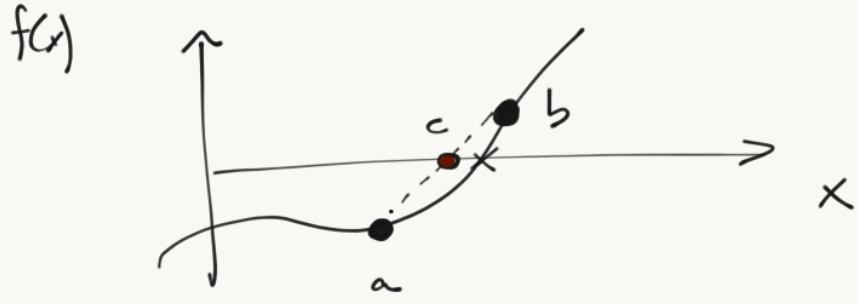
- Just guessing one could estimate that the root is halfway in between

$$x_i \sim \frac{a+b}{2}$$

- We can then choose the half interval (a,c) or $(c,b]$ that contains the root (fn. changes sign) & iterate until $f(c) = 0$ to some desired tolerance.

- False position is a refinement of interval halving.

- Rather than taking c @ the midpoint of (a,b), we approximate a line connecting $f(a)$ & $f(b)$ & find where that line crosses zero to estimate c.



(value of fn. at left ↑ "boundary")

- The line is defined by $y = m(x-c) + y_a$

$$m \approx \frac{f(b)-f(a)}{b-a} \; ; \quad y(x) = \frac{f(b)-f(a)}{b-a}(x-a) - f(a)$$

$$c: \qquad 0 = \frac{f(b)-f(a)}{b-a}(c-a) + f(a)$$

↑

(position @ which $y(x) = 0$) ∴

$$c = a - f(a) \Big/ \frac{(b-a)}{f(b)-f(a)}$$

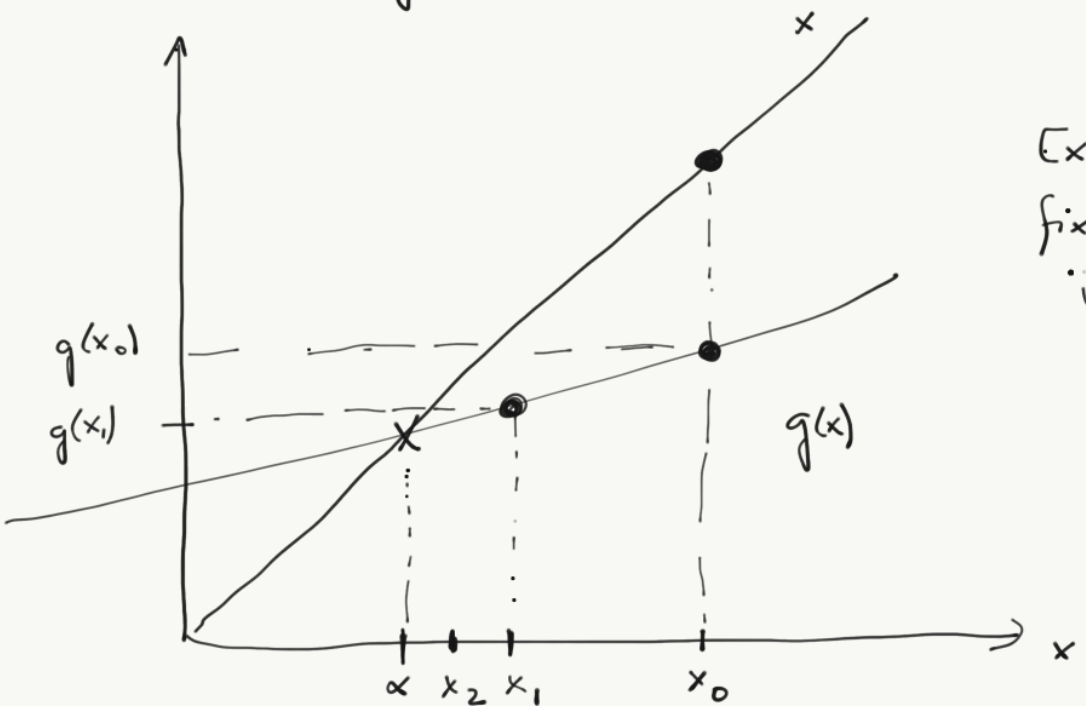
- Now as before we choose a new interval containing the root & iterate to convergence.

Fixed-point iteration

- Open domain methods do not require a root to be "bracketed"
  a) fixed-point iteration
  b) Newton's method
  c) secant method
- Fixed point iteration:
  1) recast $f(x) = 0$ into $x = g(x)$
  2) guess x_0 (initial root estimate)
  3) iterate to convergence:

$$x_{i+1} = g(x_i)$$

- This is similar in concept to Jacobi iterations for linear equations - guess solution & use it to solve remaining vars.



Ex. of fixed-point iteration

- Iterations (of any method) continue until some convergence is achieved.

$$|x_{i+1} - x_i| \le \varepsilon \quad \underline{\text{or}} \quad |f(x_{i+1})| \le \varepsilon'$$

- Suppose the true root (unknown) is denoted by α; the error is then $x - \alpha$.

$$x_{i+1} = g(x_i) \quad \therefore \quad x_{i+1} - \alpha = g(x_i) - \alpha$$

- At the solution $\alpha = g(\alpha)$:

$$x_{i+1} - \alpha = g(x_i) - g(\alpha)$$

thus: $$e_{i+1} = g(x_i) - g(\alpha)$$

- let us expand $g(\alpha)$ in a Taylor Series: (about x_i)

$$g(\alpha) = g(x_i) + g'(x_i)(\alpha - x_i) + \dots$$

then: $$e_{i+1} \approx g(x_i) - \{ g(x_i) + g'(x_i)(\alpha - x_i) \}$$

$$\approx -g'(x_i) \underbrace{(\alpha - x_i)}_{e_i}$$

(linear convergence: e_{i+1} is linear fn. of e_i)

$$\Rightarrow \boxed{e_{i+1} = g'(x_i)\, e_i} \Rightarrow |e_{i+1}| = |g'(x_i)\, e_i|$$

$$\boxed{|e_{i+1}| / |e_i| = |g'(x_i)| \quad (\text{must be} < 1)}$$

Newton's Method

- Converges quadratically, but requires derivative eval.

$$f'(x_i) \approx \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}$$ (truncated Taylor series)

- Solving for x_{i+1}, we find:

$$f'(x_i)(x_{i+1} - x_i) = f_{i+1} - f_i$$

$$f'_i x_{i+1} = f_{i+1} - f_i + f'_i x_i$$

$$x_{i+1} = \frac{f_{i+1} - f_i}{f_i'} + x_i$$

- The purpose of an iteration like this is to drive the root to zero, viz. $f_{i+1} = 0$

$\therefore$

$$\boxed{x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}}$$

Newton method!

- Convergence can be quantified by noting similarity to fixed-point iteration:

$$x_{i+1} = g(x_i) \quad \text{w/} \quad g(x_i) = x_i - \frac{f(x_i)}{f'(x_i)}$$

- For fixed point $|g'(x_i)| < 1$ converges.

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

$$g'(x) = \frac{\cancel{(f')^2} - \cancel{(f')^2} + f''f}{(f')^2} = \frac{f''f}{(f')^2}$$

- Evaluating @ root $x = \alpha$ → (goes to zero @ $x=\alpha$)

$$g'(\alpha) = \frac{f''(\alpha) f(\alpha)}{(f')^2} = 0 \quad \text{(convergent!)}$$

- Similar to before $e = x - \alpha$

$$e_{i+1} = x_{i+1} - \alpha = x_i - \frac{f_i}{f_i'} - \alpha$$

$$= x_i - \alpha - f_i / f_i' = e_i - f_i / f_i'$$

- Expanding about x_i:

$$f(\alpha) = f(x_i) + f'(x_i)(\alpha - x_i) + \frac{1}{2} f''(x_i)(\alpha - x_i)^2 + \ldots$$

- At root $f(\alpha) = 0 \Rightarrow$

$$f(x_i) = f'(x_i)(x_i - \alpha) - \frac{1}{2} f''(x_i)(x_i - \alpha)^2$$

$$= f'(x_i) e_i - \frac{1}{2} f''(x_i) e_i^2$$

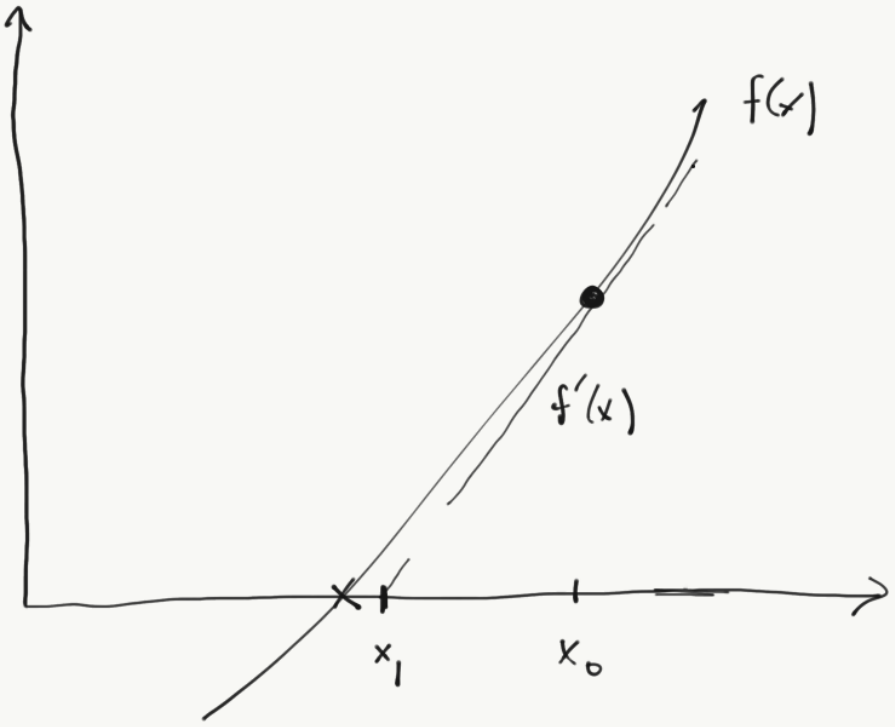
$$\therefore \quad e_{i+1} = e_i - \frac{f'(x_i) e_i - \frac{1}{2} f''(x_i) e_i^2}{f'(x_i)} = \frac{1}{2} \frac{f''(x_i)}{f'(x_i)} e_i^2$$

- In the case that derivatives are too expensive to evaluate they can be approximated:

$$\tilde{f}'(x_i) \approx \frac{f(x_{i+1}) - f(x_i)}{(x_{i+1} - x_i)}$$

$$x_{i+1} = x_i - \frac{f(x_i)}{\tilde{f}'(x_i)}$$ (approx. Newton's method)

Ex. of Newton's method



Secant Method

- Bears a lot of similarity to approximate Newton's method
- Given two initial approximations of the root (x_0, x_1), we take:

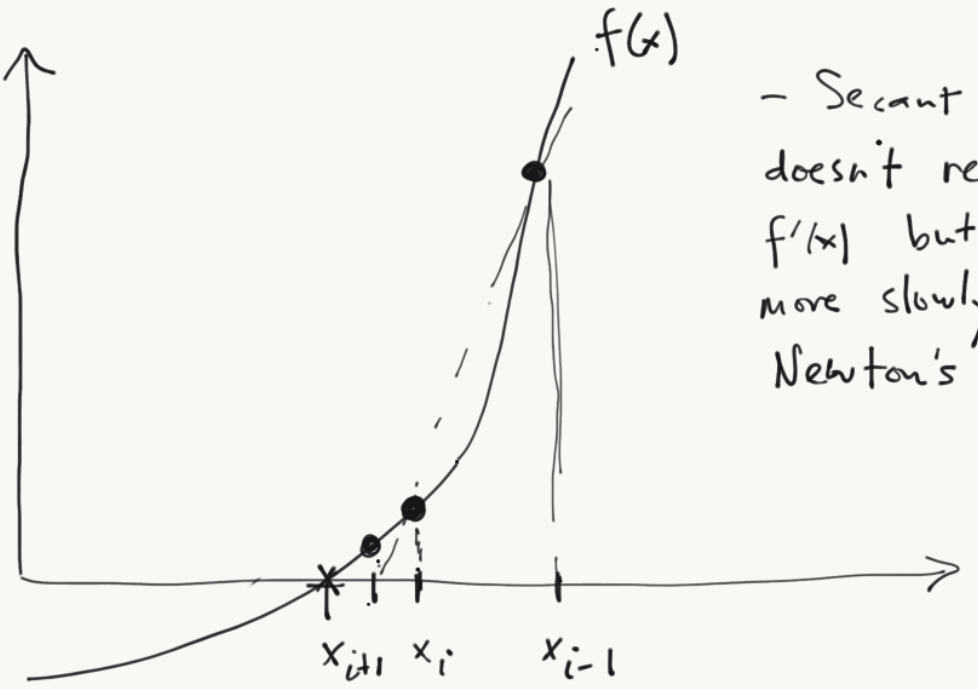
$$g'(x_i) \approx f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}}$$

- This derivative is used w/ Newton-style iteration:

$$x_{i+1} = x_i - \frac{f(x_i)}{g'(x_i)}$$

- This is equivalent to finding root of secant line as next approx. to root of $f(x)$.

(Ex. of secant method)



- Secant method doesn't require $f'(x)$ but converges more slowly than Newton's

Muller's Method

- Muller's method takes 3 initial points & approximates next root iteration from a quadratic form fitting these.
- Given x_i, x_{i-1}, x_{i-2} we find x_{i+1} from:

$$g(x) = a(x - x_i)^2 + b(x - x_i) + c$$

- With a, b, c determined from enforcing $g(x)$ passes thru f @ x_i, x_{i-1}, x_{i-2}

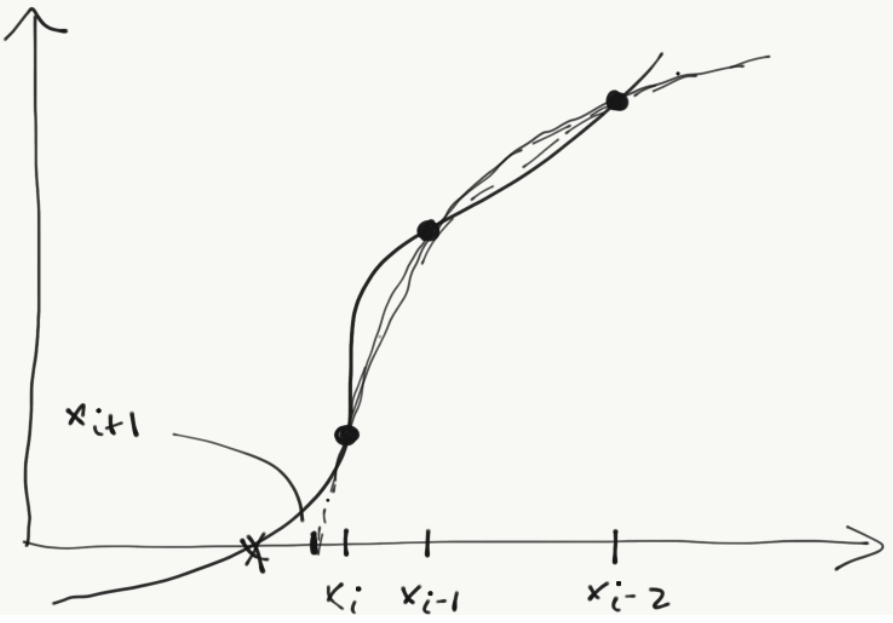
$$f_i = c$$

$$f_{i-1} = a(x_{i-1} - x_i)^2 + b(x_{i-1} - x_i) + c$$

$$f_{i-2} = a(x_{i-2} - x_i)^2 + b(x_{i-2} - x_i) + c$$

- a, b solved using latter two eqns. (see book)

Ex: Muller's method.



- Once a, b, c known, we find root of quadratic to estimate next value of root of $f(x)$:

$$x_{i+1} = x_i - \frac{2c}{b \pm \sqrt{b^2 - 4ac}}$$

- We choose root having $(\pm) \rightarrow \text{sign}(b)$ to prevent overly large changes $\Delta x = x_{i+1} - x_i$

Polynomial roots

(contains material from 4.2)

- Polynomials are of the form:

$$P_n(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n$$

- Fundamental thm of algebra: n^{th} degree polynomial has n roots.
- For linear & quadratics eqns can solve roots directly:

$$P_1(x) = ax + b \Rightarrow x = -b/a$$

$$P_2(x) = ax^2 + bx + c \Rightarrow x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

if roots have Im part they occur in complex conjugate pairs.

- Descartes' rule of signs: number of positive roots of $P_n(x)$ is number of sign changes in nonzero coeffs or smaller by an even integer. Number of negative roots found by considering $P_n(-x)$ in the same manner.
- Root-finding tends to be ill-conditioned in that small changes to P_n coeffs can greatly change roots.
- Polynomial evaluations require lots of multiplications due to x^2, x^3, x^4 and so on.

- Nested multiplication can reduce number of operations needed to evaluate polynomial e.g.

$$P_4(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4$$

$$= a_0 + x(a_1 + x(a_2 + x(a_3 + x a_4)))$$

- More generally, nested multiplication goes like:

$$P_n(x) = a_0 + x(a_1 + x(a_2 + \dots x(a_{n-1} + x a_n)\dots))$$

which can be evaluated thru the sequence:

$$b_n = a_n$$
$$b_i = a_i + x b_{i+1} \quad (i = n-1, 0)$$

$$\Rightarrow \underline{b_0 = P_n(x)}$$

- Polynomial division is defined by:

$$P_n(x) = (x-N)\, Q_{n-1}(x) + R$$

(N: divisor; $Q_{n-1}(x)$: quotient (degree $n-1$); R: remainder)

also note that, generally

$$P_n(N) = R$$

note $P_n(N) = 0 \Rightarrow (x-N)$ is a factor of $P_n(x)$
$\Rightarrow N$ is a root

- From poly-divison we have the following derivative constraint:

$$P_n'(x) = Q_{n-1}(x) + (x-N)\,Q_{n-1}'(x)$$

- Therefore, at the root N:

$$P_n'(N) = Q_{n-1}(N)$$

$$Q_{n-1} = \sum_{i=1}^{n-1} b_i x^{i-1}$$

$$P_n(x) = \sum_{i=0}^{n} a_i x^i$$

- Given a number, N, we can perform divison as follows:

$$P_n(x) = (x-N)\,Q_{n-1}(x) + R$$

$$\sum_{i=0}^{n} a_i x^i = (x-N) \sum_{i=1}^{n-1} b_i x^{i-1} + R$$

$$\sum_{i=0}^{n} a_i x^i = \sum_{i=1}^{n-1} b_i x^i - \left(\sum_{i=1}^{n-1} b_i x^{i-1} \right) N + R$$

- Polynomial representations are unique so we can equate coefficients of the powers:

$$i' = i-1$$
$$i = i'+1$$
$$i = 1 \Rightarrow i' = 0$$
$$i = n-1 \Rightarrow i' = n-2$$

$$\sum_{i=0}^{n} a_i x^i = \sum_{i=1}^{n-1} b_i x^i - N \left(\sum_{i'=0}^{n-2} b_{i'+1} x^{i'} \right) + R$$

$$a_i = b_i - N b_{i+1} \Rightarrow b_i = a_i + N b_{i+1}$$

- First & last terms are special cases:

$$b_n = a_n$$

$$b_0 \equiv R = a_0 + N b_1$$

- So synthetic division can be summarized as:

$$b_n = a$$

$$b_i = a_i + N b_{i+1} \quad (i = n-1, 1)$$

$$R = a_0 + N b_1$$

Newton's method for polynomials

- Newton's method works normally for simple (i.e non repeated) roots.
- For polynomials it can be desirable to <u>deflate</u> the polynomial once a root is found

$$P_n(x) = (x - \alpha_1) Q_{n-1}(x)$$

- The remaining roots are then those of $Q_{n-1}(x)$
- The roots off $Q_2(x)$ can be found via quadratic formula. This offers a straight forward way to converge on all roots...
- Multiple roots are handled fine via Newton's method & there a some algorithms to speed convergence (see 3.5.2.3.
  a) multiplicity factor
  b) modified function in Newton's method
- For multiple roots need to insure $f'(x) \neq 0 \Rightarrow$ use a relative convergence criteria based on $\Delta x = x_{i+1} - x_i$

- For complex roots Newton's method can converge if an initial complex guess is chosen. (Bracketing methods do not work)

- Often deflated root solutions are refined through an additional iteration step.

Systems of nonlinear equations

- Newton's method can readily be adapted to solve multi-variate problems, e.g.:

$$f(x,y) = 0 \quad ; \quad g(x,y) = 0$$

- Expanding in a two-variable Taylor series:

$$f(x,y) = f(x_i,y_i) + \frac{\partial f}{\partial x}(x_i,y_i)(x-x_i) + \frac{\partial f}{\partial y}(x_i,y_i)(y-y_i) + \dots$$

$$g(x,y) = g_i + \left[\frac{\partial g}{\partial x}\right]_i (x-x_i) + \left[\frac{\partial g}{\partial y}\right]_i (y-y_i) + \dots$$

- To solve this system, we wish to drive f, g to zero $\Rightarrow f(x,y), g(x,y) \to 0$.

$$\begin{bmatrix} (\partial f/\partial x)_i & (\partial f/\partial y)_i \\ (\partial g/\partial x)_i & (\partial g/\partial y)_i \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} = \begin{bmatrix} -f_i \\ -g_i \end{bmatrix}$$

followed by the simple update $\quad x_{i+1} = x_i + \Delta x$

$$y_{i+1} = y_i + \Delta y$$

- More generally, for n equations, n unknowns:

$$f_j(x_k) = 0$$

$\{x_k\}$: unknowns

$\{f_j\}$: nonlinear equations.

- Define a matrix:

$$D_{jk} = \frac{\partial f_j}{\partial x_k}$$

$$\underline{\underline{D}}\,\underline{\Delta x} = -\underline{f}$$

$$\underline{x}_{it1} = \underline{x}_i + \underline{\Delta x}$$

} So @ each iteration a linear system must be solved!