Chapter 3 Solving Nonlinear Equations

3.1 Introduction

The nonlinear function of unknown variable *x* is in the form of

where *n* could be non-integer. **Root** is the numerical value of *x* that satisfies $f(x) = 0$. Graphically, the root is the point where the function *f(x)* crosses or touches the x-axis.

A simple nonlinear equation is quadratic equation $f(x) = ax^2 + bx + c = 0$, the roots of it can be found analytically by quadratic formula as

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

In many situations, however, it is very difficult or impossible to find the analytical roots of the non-linear equation. In this case, numerical methods and desired accuracy of solution are often required. These numerical methods can be divided two types:

1). **Bracketing Method**: search for a root inside an interval **(x_l, x_r).**

- Graphical method
- False position (also called regula falsi or linear interpolation) method
- Bisection method

2). **Open Method**: search for a root starting from an initial given guess point **x_0.**

- Newton's method
- Secant method
- Fixed -point iteration

3.2 Bracketing Method

Intermediate value theorem:

Bracketing method: An initial interval that includes the solution is identified. Then, by using a numerical scheme, the size of the interval is successively reduced until the distance between the endpoints is less than the desired accuracy of the solution. It includes two steps:

1). Subdividing large interval (a, b) into n smaller subintervals as (a, a_1), (a_1, a_2), …, (a_{n-1}, b)

2). Examine signs at both ends of each subinterval. There are two possibilities of the two signs:

a) $f(x_l)$ and $f(x_u)$ same signs, i.e. $f(x_l)f(x_u) > 0 \rightarrow$then No roots or Even number of roots

b) $f(x_l)$ and $f(x_u)$ different signs, i.e. $f(x_l)f(x_u) < 0$, according to intermediate value theorem$\rightarrow$then One roots or Odd number of roots

Exceptions: *f(x)* id discontinuous function.

The algorithm (not MATLAB code) of bracketing method is:

```
Given: f(x), xmin, xmax, n
dx = (xmax-xmin)/n % Size of bracket interval
xl = xmin % Initialize left side of test bracket
i = 0 % Initialize counter
while i < n
  i = i + 1
  xr = xl + dx
  if f (xa) and f (xb)<0 %f(xl) and f(xr) have different
sign
    save [ xa , xb ] for further root finding
  else
  xl = xr % search next interval
end
```

3.2.1 Graphical Method

Graphical method: Sketch the graph of $f(x)$, then search for the points that intersects x-axis, these points are the roots of $f(x) = 0$.

Example 3.1 Estimate the root of $f(x) = \frac{667.38}{x}(1 - e^{-0.147x}) - 40 = 0$ by graphical method.

From the plot, we observe that it crosses x axis between (12, 16), and estimate the root roughly as $x \approx 15$. However, $f(15) = -0.4133$, thus, graphical method is not precise.

3.2.2 Bisection Method

Bisection method: is a bracketing method for finding a numerical solution of equation $f(x)=0$ when it is known that within a given interval $[a, b]$, $f(x)$ is continuous and the equation has a solution.

The algorithm is implemented in the following steps:

1). Choose a search interval $[a, b]$ which has sign-change for $f(x)$ (i.e. $f(a)f(b)<0$);
2). Calculate the midpoint of the interval $x_m = (a+ b) / 2$, use it as an estimate of the numerical solution;
3). Select subinterval from $[a, x_m]$ or $[x_m, b]$, use the one that has sign-change for $f(x)$ (i.e. contains solution) as new search interval $[a, b]$, then repeat steps 2)-3) until the error satisfies your expectation.

The algorithm of bisection method is:

```
initialize: a = . . . , b = . . .
for k = 1, 2, . . .
    xm = a + (b-a)/2
    if sign(f (xm)) = sign(f (a))
       a = xm
   else
      b = xm
   end
```

```
  if converged, stop
end
```

Example 3.2 Estimate the root of $f(x) = \frac{667.38}{x}(1 - e^{-0.147x}) - 40 = 0$ by bisection method between the interval (12, 16) until estimated relative error less than 0.5%.

Iteration	*a*	*b*	xm	ε_a	ε_t
1	12	16	14		5.279
2	14	16	15	6.667	1.487
3	14	15	14.5	3.448	1.896
4	14.5	15	14.75	1.695	0.204
5	14.75	15	14.875	0.840	0.641
6	14.75	14.875	14.8125	0.422	0.219

Note: The method converges slowly to find an ideal solution. It may fail when the function *f(x)* is tangent to and does not cross x-axis.

3.2.3 False Position (Regula Falsi or Linear Interpolation) Method

False position method: is another bracketing method for finding a numerical solution of equation *f(x)*=0 when it is known that within a given interval [*a*, *b*], *f(x)* is continuous and the equation has a solution. The endpoints at the interval (a, b) are connected by a straight line, the point where the line crosses the x-axis is thought as the estimate of the numerical solution.

Line can be expressed as: $y= sx +t$

Then $\Delta y=s\Delta x$

i.e.
let $y=0$
so $x=x_c=$ (3.1)

The algorithm is implemented in the following steps:

1). Choose a search interval [a, b] which has sign-change for $f(x)$ (i.e. $f(a)f(b)<0$);
2). Find the cross point xc by equation 3.1, use it as an estimate of the numerical solution;
3). Select subinterval from [a, x_c] or [x_c, b], use the one that has sign-change for $f(x)$ (i.e. contains solution) as new search interval [a, b], then repeat 2)-3) until the error satisfies your expectation.

Note:

3.3 Open Method

3.3.1 Newton's Method

Newton's method: also called Newton-Raphson method, is a open method for finding a numerical solution of equation $f(x)=0$ when it is known that $f(x)$ is continuous and differentiable and the equation has a solution near a given point.

Figure below shows procedure:

Newton's method can be derived by using Taylor series around x_i:
$f(x)=$

Assume x_{i+1} is the root, then
$f(x_{i+1})=f(x_i)+f'(x_i)(x_{i+1}-x_i)=0$
thus
$x_{i+1}=$ (3.2)

The algorithm is implemented in the following steps:

1). Choose a point x_i as an initial guess of the numerical solution;
2). Find the next guess x_{i+1} by equation (3.2), use it as an estimate of the numerical solution;
3). Calculate the estimated relative error ε_a, check whether the error satisfies your expectation or not. If not, use x_{i+1} as new initial guess x_i, then repeat 2); otherwise, x_{i+1} is the solution. **Stop iteration by checking 1)** $\varepsilon_a \le \varepsilon$ **or 2) tolerance** $|f(x_i)| \le \delta$ **.**

Note: Newton's method converges fast when there is a solution. The expression of derivative (slope) of function *f(x)* should be found and used to calculate the next guess of solution, sometimes it is difficult to determine the expression, then slope should be found numerically.

Newton's method may fail under the following conditions:

Case 1: Inflection point in vicinity of root

Case 2: Oscillate around local maximum or minimum

Case 3: Jump away for several roots

Case 4: Disaster from zero slope

Example 3.4 Write MATLAB function file that applies Newton's Method to find the root for $x - x^{1/3} - 2 = 0$ with true relative error less than 0.005%.

3.3.2 Secant Method

Secant method: It is good for function whose derivative expression is difficult to evaluate. It is a open method for finding a numerical solution of equation $f(x)=0$, it uses two points in the neighborhood of the solution to determine a new estimate for the solution.

In Newton's method, we know $x_{i+1}=$, when it is hard to find the expression of f'(x), Newton's method doesn't work. However, we can rewrite $x_{i+1}=$

Thus, secant method needs **two** initial data points, and is an approximated form of Newton's method.

Example 3.5 Use secant method to estimate root of $e^{-x} - x = 0$. Initial estimate $x_{-1} = 0$ and $x_0 = 1.0$.

1st iteration:
$x_{-1} = 0,\ f(x_{-1}) =$
$x_0 = 1,\ f(x_0) =$

$x_1 =$

2nd iteration:

3rd iteration:

3.3.3 Fixed-point Iteration Method

Fixed-point iteration method: It is another open method for finding a numerical solution of equation *f(x)*=0, the steps includes:

1). Rewrite original equation f(x) = 0 into another form x = g(x);
2). Select initial value x_0, and let $x_i = x_0$;
3). Update new x_{i+1} as a function of old x_i using iteration function x=g(x), i.e. $x_{i+1} = g(x_i)$
4). Check whether it reaches convergence, i.e.

$$\varepsilon_a = \left| \frac{x_{i+1} - x_i}{x_{i+1}} \right| \times 100\% \leq \text{ expected error}$$

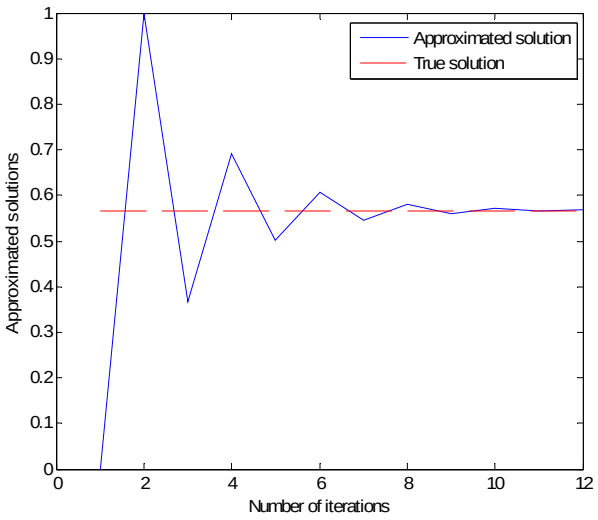
If yes, x_{i+1} is the root; otherwise, let $x_i = x_{i+1}$, and repeat steps 3)-4).

Note: 1). Sometimes, it is hard to derive *x=g(x)* from *f(x)=0*. Under this situation, you can rewrite *f(x)=0* as *f(x)+x –x=0*, then *x=f(x)+x* is your *g(x)*.

2).For some problem, it may not converge. When the derivative of g(x) in the neighborhood of the fixed point has an absolute value that is smaller than 1, i.e. $|g(x)'|<1$, the fixed-point iteration method converges.

Example 3.3 cont. Find root of $f(x) = x - e^{-x} = 0$ by fixed-point iteration method starting from $x_0=0$ until $\varepsilon_a \leq 1\%$.

i	x_i	ε_a (%)
0	0	
1	1	100
2	0.3679	171.8
3	0.6922	46.9
4	0.5005	38.3
5	0.6062	17.4
6	0.5454	11.2
7	0.5796	5.90
8	0.5601	3.48
9	0.5711	1.93
10	0.5649	1.11



3.4 Roots of Polynomial

A polynomial of order n is

where *a*'s = constant coefficients, *f(x)*=0 is special kind of nonlinear equation.

Roots of polynomials:
(1) *n*th-order equation has *n* real or complex roots
(2) If *n* is odd, at least one root is real.
(3) Complex roots exist with conjugate pairs (*r* + *mi* and *r* - *mi*)

3.5 MATLAB built-in Function for Solving Nonlinear Equations

There are two built-in functions:
1). fzero(`function', x_0)
It solves an equation (in the form of *f(x)=0*) with one variable. `function' is the expression of the function *f(x)*, x_0 is a value of x near to where the function crosses the axis. Thus, the user needs to know approximately where the solution is.
2). roots(P)
It only finds the roots of n-order polynomial

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots a_n x^n = 0$$

Where the row vector P consists of the coefficients of the polynomial and P=[a_n a_{n-1}…a_0].

Example 3.6 Find the roots of $f(x) = 10 + 8x + 5x^3 + 9x^4 = 0$ using MATLAB roots function.

Example 3.7 Find the root of Example 3.5 using MATLAB fzero function, use x_0=0.5.