1 Solving Nonlinear Equations

- solve "$f(x) = 0$"
  - where $f(x)$ is a function of x.
  - The values of x that make $f(x) = 0$ are called the roots of the equation.
  - They are also called the zeros of $f(x)$.
- The following non-linear equation can compute the friction factor,f:

$$\frac{1}{\sqrt{f}} = \left(\frac{1}{k}\right) ln(RE\sqrt{f}) + \left(14 - \frac{5.6}{k}\right)$$

- The equation for f is not solvable except by the numerical procedures.

1. **Interval Halving (Bisection)**. Describes a method that is very simple and foolproof but is not very efficient. We examine how the error decreases as the method continues.

2. **Linear Interpolation Methods**. Tells how approximating the function in the vicinity of the root with a straight line can find a root more efficiently. It has a better "rate of convergence".

3 **Newton's Method**. Explains a still more efficient method that is very widely used but there are pitfalls that you should know about. Complex roots can be found if complex arithmetic is employed.

4 **Muller's Method**. Approximates the function with a quadratic polynomial that fits to the function better than a straight line. This significantly improves the rate of convergence over linear interpolation.

5 **Fixed-Point Iteration: $x = g(x)$ Method**. Uses a different approach: The function $f(x)$ is rearranged to an equivalent form, $x = g(x)$. A starting value, x_0, is substituted into $g(x)$ to give a new x-value, x_1. This in turn is used to get another x-value. If the function $g(x)$ is properly chosen, the successive values converge.

1.1 Interval Halving (Bisection)

- Interval halving (bisection), an ancient but effective method for finding a zero of $f(x)$.
- It begins with two values for x that bracket a root.

- **The function $f(x)$ changes signs at these two x-values** and, if $f(x)$ is continuous, there must be at least one root between the values.
- The test to see that $f(x)$ does change sign between points a and b is to see if $f(a) * f(b) < 0$ (see Fig. 1).

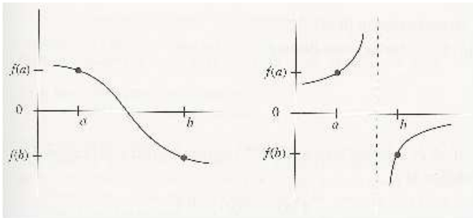

Figure 1: Testing for a change in sign of f(x) will bracket either a root or singularity.

The bisection method then

- successively divides the initial interval in half,
- finds in which half the root(s) must lie,
- and repeats with the endpoints of the smaller interval.

- A plot of $f(x)$ is useful to know where to start.

An algorithm for halving the interval (Bisection):

To determine a root of $f(x) = 0$ that is accurate within a specified tolerance value, given values x_1 and x_2, such that $f(x_1) * f(x_2) < 0$,
Repeat
Set $x3 = (x_1 + x_2)/2$
If $f(x_3) * f(x_1) < 0$ Then
Set $x_2 = x_3$
Else Set $x_l = x_3$ End If
Until $(\lvert x_1 - x_2 \rvert) < 2 * tolerance\ value$

- Think about the multiplication factor, 2.
- The final value of x_3 approximates the root, and it is in error by not more than $\lvert x_l - x_2 \rvert / 2$.

- The method may produce a false root if $f(x)$ is discontinuous on $[x_1, x_2]$.

```
>> format long e
>> fa=1e-120;fb=-2e-300;
>> fa*fb
ans =     0
>> sign(fa)~=sign(fb)
ans =     1
```

- **Example**: Apply Bisection to $x-x^{1/3}-2=0$. **m-file: demoBisect.m**

  ```
  >> demoBisect(3,4)
  ```

- **Example**: Bracketing the roots of the function, $y = f(x) = sin(x)$. **m-file: brackPlot.m**

  ```
  >> brackPlot('sin',-pi,pi)
  >> brackPlot('sin',-2*pi,2*pi)
  >> brackPlot('sin',-4*pi,4*pi)
  ```

- Now, try with a user (you!) defined function;

$$f(x) = x - x^{1/3} - 2$$

  ```
  >> brackPlot('fx3',?,?)
  ```

  In both example, try with different intervals.

- **Example**: The function; $f(x) = 3x + sin(x) - e^x$
- Look at to the plot of the function to learn where the function crosses the x-axis. MATLAB can do it for us:

```
>> f = inline ( ' 3 *x + sin ( x) - exp ( x) ')
>> fplot ( f, [ 0 2 ]) ; grid on
```

- We see from the figure that indicates there are zeros at about $x = 0.35$ and 1.9.

- To obtain the true value for the root, which is needed to compute the actual error ⇒ MATLAB

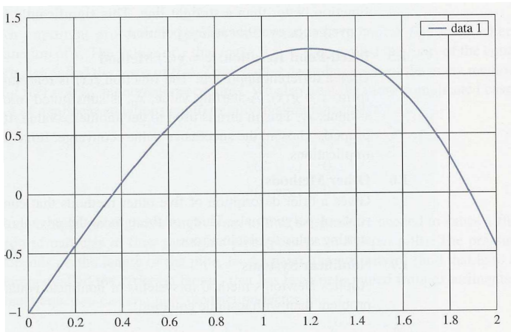

Figure 2: Plot of the function: $f(x) = 3x + sin(x) - e^x$

Table 1: The bisection method for $f(x) = 3x + sin(x) - e^x$, starting from $x_1 = 0, x_2 = 1$, using a tolerance value of 1E-4.

Iteration	X_1	X_2	X_3	$F(X_3)$	Maximum error	Actual error
1	0.00000	1.00000	0.50000	0.33070	0.50000	0.13958
2	0.00000	0.50000	0.25000	−0.28662	0.25000	−0.11042
3	0.25000	0.50000	0.37500	0.03628	0.12500	0.01458
4	0.25000	0.37500	0.31250	−0.12190	0.06250	−0.04792
5	0.31250	0.37500	0.34375	−0.04196	0.03125	−0.01667
6	0.34375	0.37500	0.35938	−0.00262	0.01563	−0.00105
7	0.35938	0.37500	0.36719	0.01689	0.00781	0.00677
8	0.35938	0.36719	0.36328	0.00715	0.00391	0.00286
9	0.35938	0.36328	0.36133	0.00227	0.00195	0.00091
10	0.35938	0.36133	0.36035	−0.00018	0.00098	−0.00007
11	0.36035	0.36133	0.36084	0.00105	0.00049	0.00042
12	0.36035	0.36084	0.36060	0.00044	0.00024	0.00017
13	0.36035	0.36060	0.36047	0.00013	0.00012	0.00005

- A general implementation of bisection (**m-file: bisect.m**)
- It is shown above how *brackPlot* can be combined with *bisect* to find a single root of an equation.
- The same procedure can be extended to find more than one root if more than root exists. Consider the code Use an appropriate 'myFunction', a suggestion is *sine* function.

The root is (almost) never known exactly, since it is extremely unlikely that a numerical procedure will find the precise value of x that makes $f(x)$ exactly zero in floating-point arithmetic.

- The main advantage of interval halving is that it is guaranteed to work (continuous & bracket).

```
>> solve('3*x + sin(x) - exp(x)')
ans=
.36042170296032440136932951583028

>> xb=brackPlot('fx3',0,5);
>> bisect('fx3',xb,5e-5)
ans =    3.5214
>> bisect('fx3',[3 4],5e-5,5e-6,1)
ans =    3.5214
```

- The algorithm must decide how close to the root the guess should be before stopping the search (see Fig. 3).

- This guarantee can be avoided, if the function has a slope very near to zero at the root.

- Because the interval $[a, b]$ is halved each time, the number of iterations to achieve a specified accuracy is known in advance.

- The last value of x_3 differs from the true root by less than $\frac{1}{2}$ the last interval.

- So we can say with surely that

$$error\ after\ n\ iterations < \left|\frac{(b-a)}{2^n}\right|$$

- When there are multiple roots, interval halving may not be applicable, because the function may not change sign at points on either side of the roots.

- The major objection of interval halving has been that it is **slow to converge**.

- Bisection is generally recommended for finding an approximate value for the root, and then this value is refined by more efficient methods.

1.2 Linear Interpolation Methods

1.2.1 The Secant Method

- Bisection is simple to understand but it is not the most efficient way to find where $f(x)$ is zero.

- Most functions can be approximated by a straight line over a small interval.

```
xmin=...; xmax=...;
Xb=brackPlot('myFunction',xmin,xmax);
for k=1:size(Xb,1)
    x(k)=bisect('myFunction',Xb(k,:));
    fprintf('Suspected root at %f gives f(x)=%f \n',
              x(k),myFunction(x(k)));
end
```

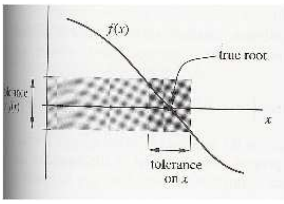

Figure 3: The stopping criterion for a root-finding procedure should involve a tolerance on x, as well as a tolerance on $f(x)$.

- The secant method begins by finding *two points on the curve* of $f(x)$, hopefully near to the root.
- As Figure 4 illustrates, we draw the line through these two points and find where it intersects the x-axis.
- If $f(x)$ were truly linear, the straight line would intersect the x-axis at the root.
- The intersection of the line with the x-axis is not at $x = r$ (root) but it should be close to it.
- From the obvious similar triangles we can write

$$\frac{(x_1 - x_2)}{f(x_1)} = \frac{(x_0 - x_1)}{f(x_0) - f(x_1)} \Longrightarrow x_2 = x_1 - f(x_1)\frac{(x_0 - x_1)}{f(x_0) - f(x_1)}$$

- Because f(x) is not exactly linear, x_2 is not equal to r,
- but it should be closer than either of the two points we began with. If we repeat this, we have:

$$x_{n+1} = x_n - f(x_n)\frac{(x_{n-1} - x_n)}{f(x_{n-1}) - f(x_n)}$$

- The net effect of this rule is to set $x_0 = x_1$ and $x_1 = x_2$, after each iteration.

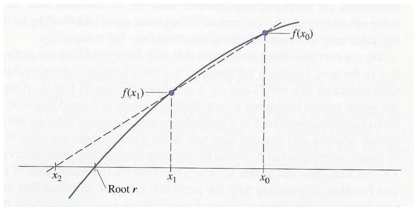

Figure 4: Graphical illustration of the Secant Method.

Table 2: The Secant method for $f(x) = 3x + sin(x) - e^x$, starting from $x_0 = 1, x_1 = 0$, using a tolerance value of 1E-6.

Iteration	x_0	x_1	x_2	$f(x_2)$
1	1	0	0.4709896	0.2651588
2	0	0.4709896	0.3722771	2.953367E-02
3	0.4709896	0.3722771	0.3599043	−1.294787E-03
4	0.3722771	0.3599043	0.3604239	5.552969E-06
5	0.3599043	0.3604239	0.3604217	3.554221E-08

At x = .3604217, tolerance of .0000001 met!

- The technique we have described is known as, the secant method because the line through two points on the curve is called the secant line.
- **An algorithm for the Secant Method:**

> To determine a root of $f(x) = 0$, given two values,
> x_0 and x_1, that are near the root,
> If$|f(x_0)| < |f(x_1)|$ Then
> Swap x_0 with x_1
> Repeat
> Set $x_2 = x_1 - f(x_1) * \frac{(x_0 - x_1)}{f(x_0) - f(x_1)}$
> Set $x_0 = x_1$, Set $x_1 = x_2$
> Until $|f(x_2)| < tolerance\ value$

- Table 2 shows the results from the secant method for the same function that was used to illustrate bisection.
- An alternative stopping criterion for the secant method is when the pair of points being used are sufficiently close together.
- If the method is being carried out by a program that displays the successive iterates, the user can interrupt the program should such improvident behavior be observed.

- If $f(x)$ is not continuous, the method may fail.
- If the function is far from linear near the root, the successive iterates can fly off to points far from the root, as seen if Fig. 5.

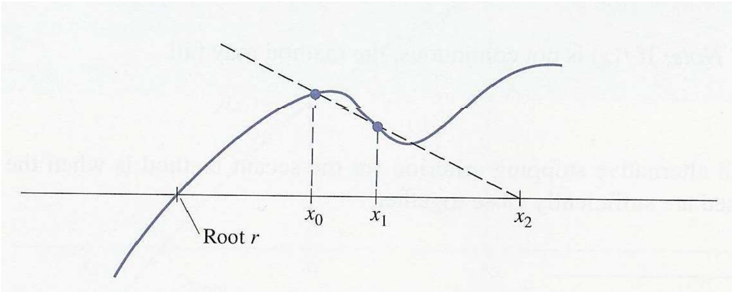

Figure 5: A pathological case for the secant method.

- If the function was plotted before starting the method, it is unlikely that the problem will be encountered, because a better starting value would be used.

1.2.2 Linear Interpolation (False Position)

- A way to avoid such pathology is to ensure that the root is bracketed between the two starting values and remains between the successive pairs.
- When this is done, the method is known as linear interpolation (regula falsi).
- This technique is similar to bisection except the next iterate is taken at the intersection of a line between the pair of x-values and the x-axis rather than at the midpoint.
- Doing so gives **faster convergence** than does bisection, but at the expense of a more complicated algorithm.

Table 3: Comparison of methods, $f(x) = 3x + sin(x) - e^x$, starting from $x_0 = 0, x_1 = 1$.

	Interval halving		False position		Secant method	
Iteration	x	$f(x)$	x	$f(x)$	x	$f(x)$
1	0.5	0.330704	0.470990	0.265160	0.470990	0.265160
2	0.25	−0.286621	0.372277	0.029533	0.372277	0.029533
3	0.375	0.036281	0.361598	$2.94 * 10^{-3}$	0.359904	$-1.29 * 10^{-3}$
4	0.3125	−0.121899	0.360538	$2.90 * 10^{-4}$	0.360424	$5.55 * 10^{-6}$
5	0.34375	−0.041956	0.360433	$2.93 * 10^{-5}$	0.360422	$3.55 * 10^{-7}$
Error after 5 iterations	0.01667		$-1.17 * 10^{-5}$		$<-1 * 10^{-7}$	

(Exact value of root is 0.360421703.)

- **An algorithm for the method of false position:**

  To determine a root of $f(x) = 0$, given two values of x_0 and x_1 that bracket a root: that is, $f(x_0)$ and $f(x_1)$ are of opposite sign,
  Repeat
  Set $x_2 = x_1 - f(x_1) * \frac{(x_0 - x_1)}{f(x_0) - f(x_1)}$
  If $f(x_2)$ is of opposite sign to $f(x_0)$ Then
  Set $x_1 = x_2$,
  Else
  Set $x_0 = x_2$
  End If
  Until $|f(x_2)| < tolerance\ value.$

- If $f(x)$ is not continuous, the method may fail.

- Table 3 compares the results of three methods-interval halving (bisection), linear interpolation, and the secant method for $f(x) = 3x + sin(x) - e^x = 0$

- Observe that the **speed of convergence** is best for the secant method, poorest for interval halving, and intermediate for false position.

1.3 Newton's Method

One of the most widely used methods of solving equations is Newton's method (Newton did not publish an extensive discussion of this method, but he solved a cubic polynomial in *Principia* (1687)).

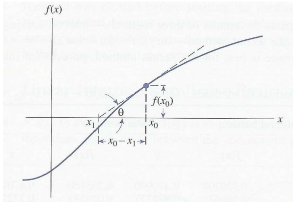

Figure 6: Graphical illustration of the Newton's Method.

- The version given here is considerably improved over his original example.
- Like the previous ones, this method is also based on a linear approximation of the function, but does so using a tangent to the curve (see Figure 6).
- Starting from a single initial estimate, x_0, that is not too far from a root, we move along the tangent to its intersection with the x-axis, and take that as the next approximation.
- This is continued until either the successive x-values are sufficiently close or the value of the function is sufficiently near zero.
- The calculation scheme follows immediately from the right triangle shown in Fig. 6.

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

and the general term is:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \; n = 0, 1, 2, \ldots$$