In engineering, many systems can be described by single input/output relations where \( x \) represents the input and \( y \) represents the output.

\[
\text{Input(s)} \quad \rightarrow \quad y = f(x) \quad \rightarrow \quad \text{Output(s)}
\]

If the function describing the system is known, it is a simple matter to determine the output \( y \) if the input \( x \) is specified. This process is often called analysis. On the other hand, suppose you wanted to design the system so that a given output \( y_0 \) was generated. This question is equivalent to solving \( f(x_0) = y_0 \). If the function \( f(x) \) were invertible (usually it is not), we could at once write the solution as \( x_0 = f^{-1}(y_0) \).

**Examples**

Find \( x_0 \) for the given function \( f(x) \) and design output \( y_0 \).

1. \( f(x) = x^2 \quad y_0 = 16 \). In this case, the inverse function clearly exists and is of the form \( f^{-1}(y) = \pm \sqrt{y} \).
   
   Therefore, \( x_0 = \pm \sqrt{y_0} = \pm \sqrt{16} = \pm 4 \)

2. \( f(x) = e^x \quad y_0 = 3 \). In this case, the inverse function clearly exists and is of the form \( f^{-1}(y) = \ln(y) \).
   
   Therefore, \( x_0 = \ln(y_0) = \ln(3) \)

You may think from the above examples that any function has an inverse and therefore the design problem can be solved in all cases. This is far from true.

**Examples**

Find \( x_0 \) for the given function \( f(x) \) and design output \( y_0 \).

\( f(x) = x^5 - 4x^4 + 3x^2 + 8 \quad y_0 = 14 \). In this case, the inverse function does not exist. If you can find one, you have done the impossible. N.H. Abel in 1823 proved that the general polynomial whose degree \( N > 4 \) can not be inverted (solved) analytically.
We can plot the function $y = f(x)$ and look at where the curve intersects the function $y = y_0$. In the above, the $x$ domain had to be adjusted by trial and error several times before the best domain for viewing was obtained. The ‘o’ marks the intersection and was plotted as a separate point whose value was obtained by successive use of the **zoom** function. The intersection occurs at $x_0 = 3.8228$.

This procedure is indicative of the design methodology in general. In the first step, the output is calculated for all possible inputs. The collection of input/output pairs is then compared to the desired output. It should be clear that this methodology can be extended to multi-input multi-output functions.

In the above example, only one input value was obtained but it is clear from the graph that output values exist in which have more than one input value.

**Problem**
From the graph, give an example of an output value with 2 input values, 3 input values, 4 input values, 5 input values, 6 input values

**Question?**
What are your observations and can you explain them?
The above example utilized a polynomial function but it is clear that any function can be used as long as we can graph the function response.

While the above problem was expressed in terms of the intersection of two curves (the function curve and the output curve), we could solve the problem by finding the roots (zeros) of the auxiliary function \( \tilde{f}(x) = f(x) - y_0 \). In terms of this function, the design inputs are the roots of \( \tilde{f}(x) = 0 \). Therefore, if we can find the roots of the auxiliary function, we can always obtain the design inputs.

Example:

\[
\begin{align*}
  f(x) &= x^5 - 4x^4 + 3x^2 + 8 \\
  y_0 &= 14 \Rightarrow \tilde{f}(x) = x^5 - 4x^4 + 3x^2 + 6 = 0
\end{align*}
\]

Therefore, we look to see what Matlab tools and procedures can be used to find the roots of functions. At this time, we only consider functions of the form \( y = f(x) \) i.e. single input, single output.

**Polynomial Functions**

A polynomial of degree \( n \) is of the form

\[
f(x) = a_0 x^n + a_1 x^{n-1} + \cdots + a_{n-1} x + a_n
\]

*Note:* An \( N^{\text{th}} \) degree polynomial has \( (N+1) \) terms and \( (N+1) \) coefficients.

From algebra, you learned that an \( N \)th degree polynomial has \( N \) roots which are either real or occur in complex conjugate pairs. In the graphical approach, only real roots are obtained. This explains why it was possible for a 5\(^{\text{th}} \) degree polynomial to have only one real root. The other 4 occur as complex conjugate pairs (2 sets of 2) and will not appear as intersections on a real \( x \)-\( y \) plot. We will study complex numbers in Lesson 5.

**A. Finding real roots using graphical methods (the zoom function)**

Example

Find the real roots of the polynomial \( y = x^2 - 4x + 2 \)

To do this, we must first plot the function on a finite interval and see where it intersects with the \( y = 0 \) (\( x \)-axis) curve.
\[ x = [-5:.01:5]; \]
\[ y = x^2 - 4x + 2; \]
\[ \text{plot}(x,y) \]

**Questions?**
Are there any real roots? (Does the function cross the x-axis).
To get good accuracy, we need to zoom the graph using the zoom feature.
In this example, we guessed at the interval and our guess was sufficient to see all the roots.

* Not all design questions have an answer.

**Problem**
Numerically obtain the roots of the above equation and compare to those obtained using the quadratic formula.

**Note:** Be careful to use enough different domain values to be sure that you are not missing any intersections.

**Problem**
Find the roots to \[ y = x^2 + 4x + 20 \] and compare with the quadratic formula.

**Questions?**
What are your observations for the previous problem?
Polynomials are special functions that Matlab can generate in a very simple way. This is done using the Matlab `polyval` function.

\[ \text{yvect} = \text{polyval(coeffvect, xvect)} \] where coeffvect is a vector storing the polynomial coefficients in descending order and xvect are the domain (x) values where the polynomial needs to be calculated.

**Example**
Generate the polynomial \[ y = 2x^2 - 4x + 1 \] on the interval \[-5 \leq x \leq 5\]
\[ \text{xvect} = [-5:.01:5]; \]
\[ \text{coeffvect} = [2 -4 1]; \]
\[ \text{yvect} = \text{polyval(c, xvect)} \]
\[ \text{plot(xvect, yvect)} \]
Problem
Find all real roots of \( y = x^4 - 5x^3 - 6x^2 + 3x + 1 \)

B. Finding real and complex roots using the roots function

Matlab has a special function that calculates all roots (real and complex) of a polynomial. The syntax is simply

\[
>> \text{xroots} = \text{roots}(	ext{coeffvect}).
\]

All that is needed is the coefficients of the polynomial.

Problem

1. Compare roots of \( y = x^4 - 5x^3 - 6x^2 + 3x + 1 \) obtained using the \text{roots} function to the real roots obtained from graphical method.
2. Repeat for \( y = x^4 + 2x^2 + 1 \);

Transcendental Functions

Many functions exist that are not of polynomial type. Examples include radicals (fractional powers), rational functions (Ratios of Polynomials), trigonometric, exponential, logarithmic or any combination thereof. It is clear that the graphical method is capable of finding the roots.

Example

Find all roots of \( y = \cos(x) - 0.2x^2 \) on the interval \( -\frac{\pi}{2} \leq x \leq \frac{\pi}{2} \)

\[
x = [-\text{pi}/2: \text{pi}/200: \text{pi}/2];
y = \cos(x) - 0.2x^2;
\text{plot}(x,y,x,\text{zeros(length(x),1)},'--')
\]

% The zeros argument is designed to draw the x-axis so the intersection is clearly visible.
The answer after zooming at the two intersection points is +1.2159, -1.2159

A. Numerical Methods

We discuss two numerical techniques for finding the zeros of a one variable function, namely the Direct Method and the Newton-Raphson Method.

**Direct Method**

This is particularly useful technique when the equation \( f(x) = 0 \) can be written in the form of \( x = I(x) \) instead of \( f(x) = 0 \).

\( I(x) \) is then called an iteration function, and it can be used for the generation of the sequence: \( x_{k+1} = I(x_k) \).

We assume that \( x_{k+1} \) is the value of \( x \) at the iteration \( (k+1) \) which is based on the value of \( x \) at the iteration \( (k) \) \( (x_{k+1} = I(x_k)) \), we keep calculating \( x \) for many iterations until the value of any newly calculated \( x \) is similar to the value of \( x \) at the previous iteration \( (x_{k+1} = x_k) \).

We say that this method converges to a solution.

To guarantee that this method gives accurate results in a specific case, the function \( I(x) \) should be **continuous** and it should satisfy the contraction condition:

\[
|I(x_n) - I(x_m)| \leq s |x_n - x_m|
\]
The difference between the function at two arguments is smaller than the difference between the arguments themselves.
Where $0 \leq s < 1$; [see book page 125 for proof].

Note that the value of $x$ at the first beginning (iteration=1) must be known in order to calculate the value of $x$ at the next iteration, we call this value of $x$ at the iteration 1, the initial guess, and we assume it to be anything, the closer our assumption is to the correct value of $x$, the faster the method converges.

Example
Find the zero of the function: $y = x - \sin(x) - 1$

Solution
Finding the zero of $f(x)$ is solving the equation $f(x)=0$; we can write the function $y=f(x)=0$ as $x=I(x)$, by defining $I(x)=\sin(x)+1$; $\Rightarrow y=x-I(x)$, hence $x=I(x)$;
$I(x)$ is continuous, since $I(x)$ is $\sin(x)-1$, and the contraction condition is satisfied since the difference between two “sines” is always smaller than the difference between their arguments (the angles)
At the zero, the iterative form can be written as:

$$x_k = \sin(x_{k-1}) + 1$$

Then using MATLAB we can calculate the value of $x$ for as many as iteration we would need by performing the code below:

```matlab
x(1)=1;  %value of initial guess
for k=2:20
    x(k)=sin(x(k-1))+1;
end
x
```

```
x =
1.0000 1.8415 1.9636 1.9238 1.9383 1.9332 1.9350 1.9344 1.9346 1.9345 1.9346 1.9346 1.9346 1.9346 1.9346 1.9346 1.9346 1.9346
```

As can be noticed, we needed about 11 iterations to get the correct or fixed value of $x$ (accurate to the 4th digits).

Problem
Redo the same problem but use the while loop
Newton-Raphson Method

This method requires the knowledge of both the function and its derivative. The method makes use of the geometrical interpolation of the derivative being the tangent at a particular point, and that the tangent is the limit of the chord between two close points on the curve. It is based on the fact that if \( f(x_1) \) and \( f(x_2) \) have opposite signs and the function \( f \) is continuous on the interval \([x_1, x_2]\), we know from the intermediate Value theorem of calculus that there is at least one value \( x_c \) between \( x_1 \) and \( x_2 \), such that \( f(x_c) = 0 \). A sufficient condition for this method to converge is that \( f'(x) \) and \( f''(x) \) have constant sign on an open interval that contains the solution \( f(x) = 0 \); in that case, any starting point that is close enough to the solution will give successive Newton’s approximations that converge to the solution.

Let \( x_{\text{guess}} \) and \( x \) have the same meaning as in the iterative method; therefore, \( f(x) = 0 \), and the definition of the derivative results in the equation:

\[
x = x_{\text{guess}} - \frac{f(x_{\text{guess}})}{f'(x_{\text{guess}})}
\]

This relation can now be the basis of an iterative function given by:

\[
x_k = x_{k-1} - \frac{f(x_{k-1})}{f'(x_{k-1})}
\]

The fixed point (limit) of \( x \), can be obtained for the same initial guess and tolerance, in a smaller number of iterations in the Newton-Raphson method than in the Direct Iterative method.

Problem

Find the zero of the function below using the Newton-Raphson method and compare with the results obtained earlier for the Direct Iterative method.

\[
Y = f(x) = x - \sin(x) - 1
\]

B. MATLAB fzero & fsolve built-in functions

Problem

Find the roots of \( y(x, a) = \ln \left( \frac{x}{a} \right) - (x - a) \) for \( a = 0.5 \), \( a = 2 \), \( a = 5 \).
Note that one trivial root (x=a) always exists. While the graphical method is very powerful, it requires a great deal of user intervention. However, Matlab is able to obtain the roots of arbitrary functions using the \texttt{fzero} function.

To use \texttt{fzero}, you need two things
1) A function file holding the function to be set to zero
2) A reasonable guess of the root.

It should be clear that a reasonable guess can only occur if you have either some physical knowledge which can estimate the root(s) or you use graphics to get an idea where the roots should occur.

\begin{verbatim}
Function file f1.m

###########################################################
function y=f1(x)
y=cos(x)-0.2*x.^2;
###########################################################

The \texttt{fzero} function call using a guess x=-1
>>fzero('f1',-1)
   -1.2519

Questions?
What happens if we use \texttt{fzero('f1',1)}?
What happens if we use \texttt{fzero('f1',0.1)}?

Problem
Use \texttt{fzero} to find the non-trivial roots of \( y(x,a)=\ln\left(\frac{x}{a}\right)-(x-a) \)
for a=2 and a=5 and compare with the graphical method.

The MATLAB command \texttt{fzero} is quite suitable for finding the zero of a function of one variable. However, we must use \texttt{fsolve} for the case of two-variable function.

Example
Find all intersection points between any two functions \( f_1(x) \) and \( f_2(x) \).
This is equivalent to finding all the roots to \( F(x)=f_1(x)-f_2(x) \) so any of the above root finding methods are applicable.