MCE511 Mathematica
Quick Introduction to Functional Programming

INSTRUCTOR: OA Fakinlede
www.oafak.com
oafak@unilag.edu.ng; fakinlede.Omotayo@lmu.edu.ng; oafak@hotmail.com

Department of Mechanical Engineering, Landmark University

Quick Introduction

* In the following Slides, we present a quick introduction to the Mathematica software in the areas of you design training as I shall be using it.
* A wise student can continue learning beyond the class requirements thereby achieving literacy in a powerful programming environment that can be useful for most other courses you can take.
* Mathematica is programmable with its own language. It supports symbolics, graphics as well as numerical computations. If you don't know how to program, last chance? Will you take it?

Required Tools

* One of our main software packages for this course is Wolfram Research Mathematica version 10 or better. This is available to you in the CSIS lab. You can get a personal copy of version 10 by purchasing a Raspberry Pi
* We will use both the Numeric and Symbolic capabilities of this software as an aid to design computations.
* The graphical capacity of the package is extremely good. Mathematica could, in many cases, be a good alternative to programming in a High level language.

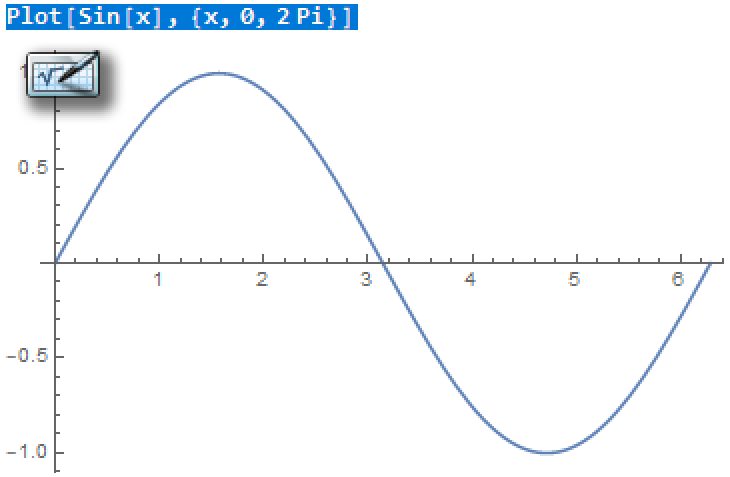
Other Literature

* A .pdf copy of the print manual of version 5 of Mathematica is available for you to download. This is probably the last time Wolfram made a print book for the software. The on-line help system is vastly superior to any book you can find. There are illustrated answers in codes you can copy and execute immediately. This is usually the simplest way to learn – by imitation!
* There is also a small nook called "the Mathematica Cookbook". It is easy for beginners.

Notation & Conventions

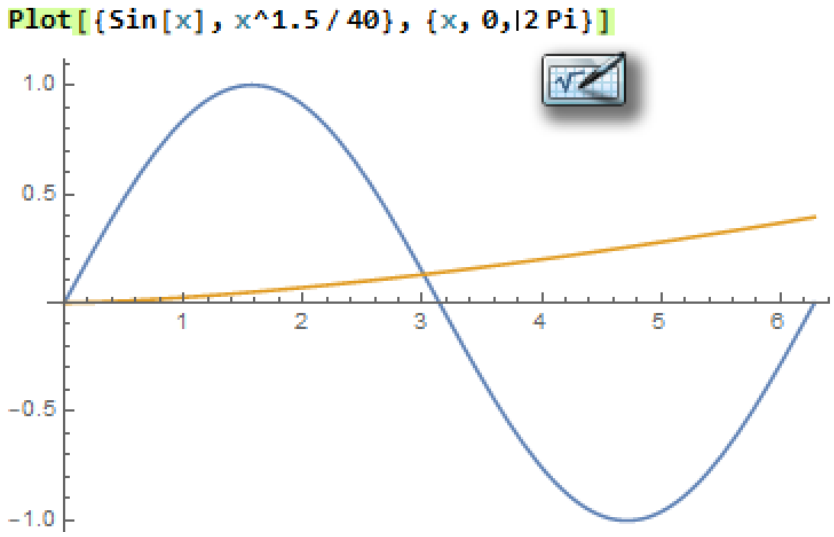
* Built-in functions: In Mathematica, built-in functions are easily recognizable:

1. They have names similar to the mathematical functions such as Sin, Cos, Log, etc. They are ALWAYS capitalized (first letter only) and their arguments are listed in square brackets. For example;
2. Sin[x] appearing here is the familiar function Plot taking Sin[x] as well as the list in braces is another function resulting in the plot shown.



Notation

* Plot [{Sin[x],x^1.5/40},{x,0,2 Pi}]



* Here, we are plotting two functions, the Pi is pre multiplied by 2. The space indicates multiplication here. The two arguments of the Plot function are lists in curly braces

Notation

* Integrate[a x^2,x] is a function taking the expression to be integrated as its first argument while the variable of integration is the second. This has the meaning: $\int ax^2 dx$. Note the space between a and x^2. If we had written ax^2 it will think that ax is itself a variable.
* If we are interested in the definite integral, the second argument will be a list of the variable and its domain: Integrate[a x^2,{x,1,4}] means $\int_1^4 ax^2 dx$
* Furthermore, Log[x,b] is $\log_x b$; and {x,0,2Pi,2} means we have the value of x ranging from zero to 2π in steps of 2. The default when steps is not stated is 1.

List Processing

* Lists are created, as we have seen using braces. Some lists, such as matrices require nesting of the braces as in:
* As we can see, once the matrix M is defined, we can request operations on it. The operand required by these functions is the nested list we started with. := is a delayed assignment – invoked only when needed. = is an immediate assignment.

```
M := {{1, 2, 3}, {3, 4, 5}, {-1, 1, 7}}

M[[1]][[2]]

2

Det[M]

-8

Eigenvalues[M]
```

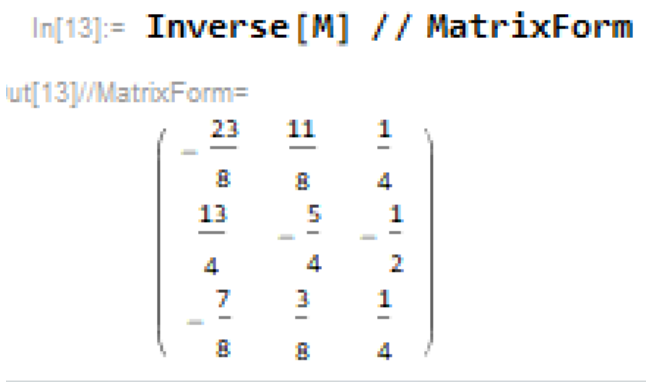
$$\{8, 2+\sqrt{5}, 2-\sqrt{5}\}$$

```
Inverse[M]
```

$$\left\{\left\{-\frac{23}{8}, \frac{11}{8}, \frac{1}{4}\right\}, \left\{\frac{13}{4}, -\frac{5}{4}, -\frac{1}{2}\right\}, \left\{-\frac{7}{8}, \frac{3}{8}, \frac{1}{4}\right\}\right\}$$

Output Format

* In the last example, we can control how our answer is given. In the following example, we request, that the result of the inverse be given to us in the usual matrix format. One way of doing this is to issue the command and make the output request in a "postfix" form as shown. The double slash initiates the postfix request.



Equations, Equality & Assignment

* One confusion introduced by an error in early programming languages never got corrected. The use and abuse of the equal sign.
* One of the first things that show you know how to program is the recognition that "=" does not indicate equality! Instead, in most programming languages, it means "assign the result of the computation on the right to the variable on the left".
* Mathematica, like most modern programming languages, uses "==" to indicate equality. An example below will show its usage:

```
Solve[{2 x + y - z == 10, -5 x + y + z == -1, x - 2 y - 5 z == -1}, {x, y, z}]
```

```
{{x → 1, y → 6, z → -2}}
```

Nonlinearity and Undetermined Values

* Mathematica will attempt to solve your equations even if some parameters are not yet determined. It does not matter if some of the equations are linear or not:
* Consider the following pairs of simultaneous equations:

```
In[17]:= Solve[{a x + b y == 1, x - y == 2}, {x, y}]
```

$$\text{Out[17]= } \left\{\left\{x \to -\frac{-1-2b}{a+b},\ y \to -\frac{-1+2a}{a+b}\right\}\right\}$$

```
In[19]:= Solve[{x^2 + y^2 == 1, x + y == a}, {x, y}]
```

$$\left\{\left\{x \to \frac{1}{2}\left(a-\sqrt{2-a^2}\right),\ y \to \frac{1}{2}\left(a+\sqrt{2-a^2}\right)\right\},\right.$$
$$\left.\left\{x \to \frac{1}{2}\left(a+\sqrt{2-a^2}\right),\ y \to \frac{1}{2}\left(a-\sqrt{2-a^2}\right)\right\}\right\}$$

Number Types

* Mathematica supports several number types. It will attempt to keep computations in their most accurate form whenever possible. The function Head[] with the expression as argument will tell you what numerical type Mathematica is using.
* Supported types include Integer, Rational, Complex, Real, Power (of Integers) etc.
  The fact that the sine of $\frac{\pi}{4}$ is $2^{1/2}$ which is fully expressible in integer terms is the reason why Mathematica will still treat it, not as expected as a numerical real number but simply as a power. The precision remains exact. Reals can be specified to any number of decimals as requested. Try N[Sqrt[2],100]

```
In[25]:= Sin[Pi / 2]
Out[25]= 1

In[27]:= Head[Sin[Pi / 4]]
Out[27]= Power

In[28]:= Sin [Pi / 4]
Out[28]= 1/√2
```

Everything is a Function!

* Every computation in Mathematical is a function call. This is one of its attributes as a functional programming environment. For example, you will get the right result if you ask mathematica to simply add two or more numbers as in: 2+3; or 1+20+15. However, underlining these are the full form of the expressions and you can check by the function FullForm[ ].
* From which you can see that the full form of the expression 2+3 is actually the function call, Plus[2,3] and 2 *3 is Times[2,3]

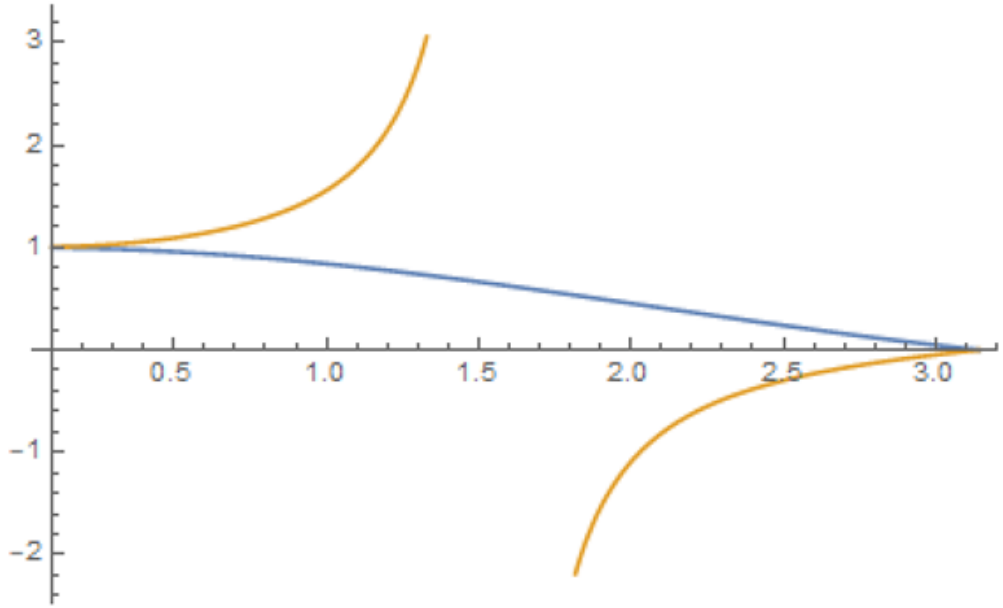
```
In[31]:=
        FullForm[g + h]

ut[31]//FullForm=
        Plus[g, h]

In[32]:= FullForm[x * y * z]

ut[32]//FullForm=
        Times[x, y, z]
```

Limit Example

* You always knew that as x approaches zero, there is no distinction between x and its sine or tangent. A simple Mathematical plot shows this:

In[33]:= `Plot[{Sin[x] / x, Tan[x] / x}, {x, 0.1, Pi}]`

Out[33]=



Precision & Accuracy

* Symbolics allow you to do exact math in Mathematica wherever possible. There are times when you are forced to do numerics and here precision and accuracy become issues of importance.
* Treat precision as the number of digits in the decimal representation of your value while accuracy is the number of digits after the decimal point. You may request for these as shown in:

```
In[35]:= Table[With[{x = 10^n + 1 / 17}, {N[x, 10],
           N[x, {Infinity, 10}]}], {n, 0, 5}] // TableForm
```

```
Out[35]//TableForm=
1.058823529     1.058823529
10.05882353     10.058823529
100.0588235     100.0588235294
1000.058824     1000.058823529
10000.05882     10000.0588235294
100000.0588     100000.0588235294
```

Notational Variety

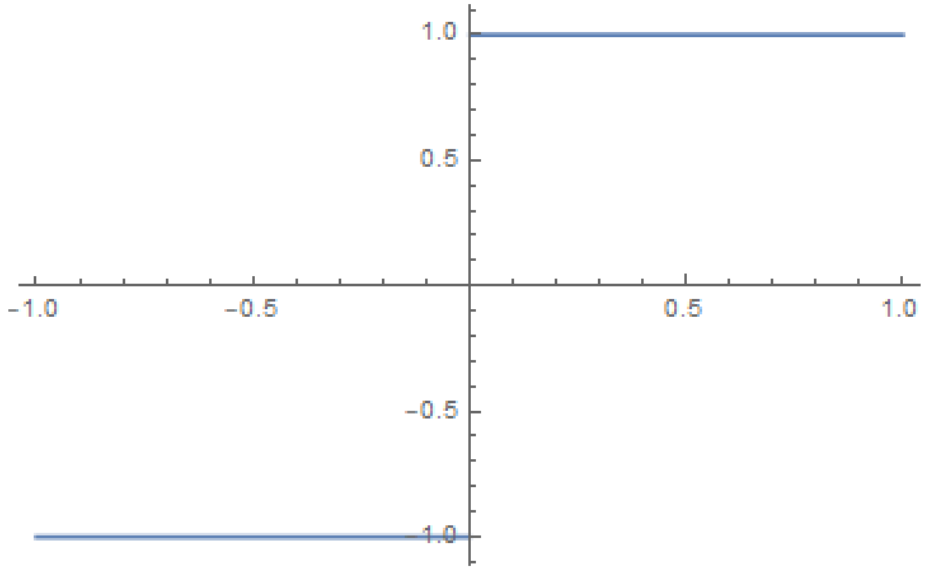
* The same computation may be submitted to Mathematica in a number of different ways. Here are a few examples:

Method	Examples
Function	Divide[Sin[x],x], N[Sqrt[2]]
Infix	Sin[x]/x
Postfix	Sqrt[2]//N (explicitly requests numeric result
Prefix	N@1/2 (Want the numeric in a prefix way)

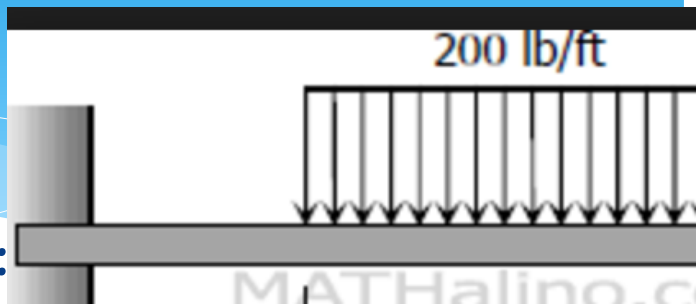
Piecewise Functions

* Engineers meet with functions that are discontinuous or defined differently in different domains. The piecewise function of Mathematica allows you to do this easily. For example, the step function at the origin:

```
Plot[Piecewise[{{-1, x < 0}, {1, x > 0}}], {x, -1, 1}]
```



Practical Uses



* For the benefit of those who want to know the practical applications of some of these, note the following:
* When you apply a point load or a sudden impulse to a vibratory system, the way to express that in the differential governing equations is take the magnitude and multiply by a unit impulse (Also called the Dirac Delta Function).
* This Dirac function (arguably not a real function to mathematical purists, but a distribution) is the derivative of the unit step function. The latter is useful on its own when a surface is free of loading for a portion and you have uniform loading in the rest of the space.

Fourier Approximation of the Step Function

* Next is a Fourier Series Approximation of the same Step Function:
* Can you get a better approximation?
* What do you need to change in the above code?
* Try it!

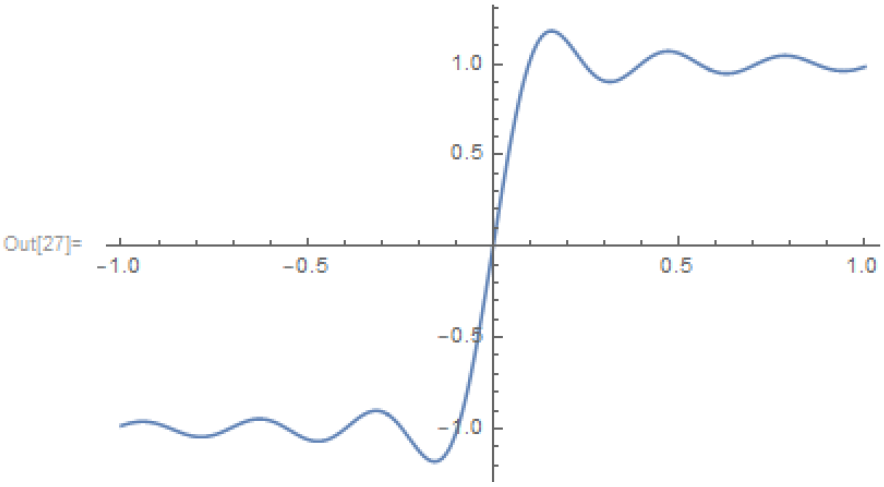
In[26]:=
```
FourierSinSeries[Piecewise[{{-1, x < 0}, {1, x > 0}}], x, 20]
```

Out[26]=

$$\frac{4\sin[x]}{\pi}+\frac{4\sin[3x]}{3\pi}+\frac{4\sin[5x]}{5\pi}+\frac{4\sin[7x]}{7\pi}+\frac{4\sin[9x]}{9\pi}+\frac{4\sin[11x]}{11\pi}+\frac{4\sin[13x]}{13\pi}+\frac{4\sin[15x]}{15\pi}+\frac{4\sin[17x]}{17\pi}+\frac{4\sin[19x]}{19\pi}$$

In[27]:=
```
Plot[%, {x, -1, 1}]
```

Out[27]=



Interval Arithmetic

* In Mathematica, it is possible to work with a range of values rather than specific values. This is implemented by the functionality of Interval Arithmetic. A simple example will be used to illustrate this concept:

In[19]:=

```
Clear[error1, error2, mass, velocity, kineticEnergy];
error1 = 0.01; error2 = 0.005; mass = Interval[{1.10 - error1, 1.10 + error1}];
velocity = Interval[{7.50 - error2, 7.50 + error2}];
kineticEnergy = (1 / 2) mass velocity^2
Abs[Subtract[kineticEnergy[[1]][[1]], kineticEnergy[[1]][[2]]]]
Subtract @@ kineticEnergy[[1]] // Abs
```

Out[22]= `Interval[{30.6154, 31.2604}]`

Out[23]= `0.645`

Out[24]= `0.645`

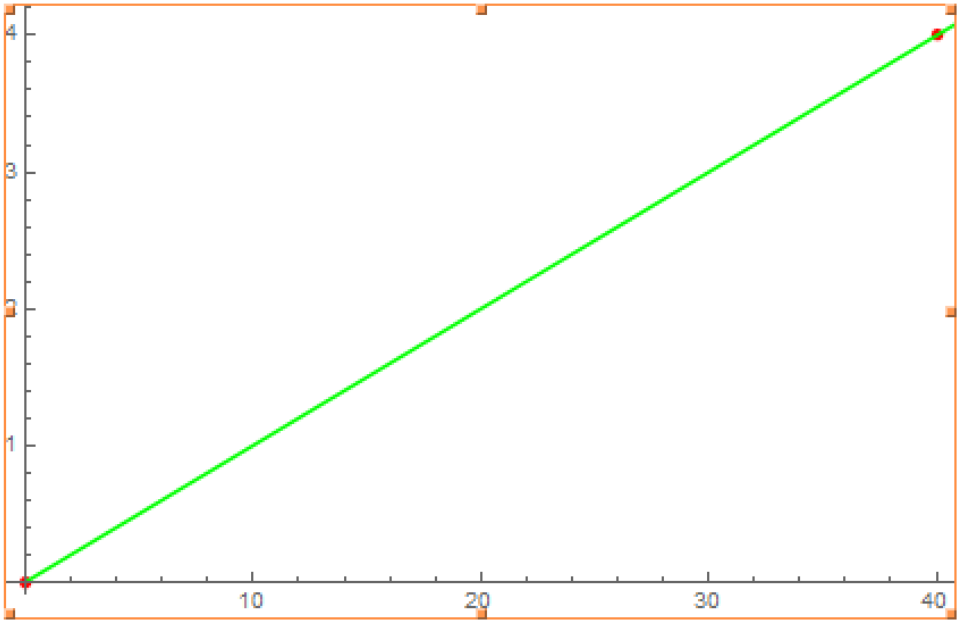
Help: Instant, Detailled

* You can seek for assistance at any point in your program by simply using the double question marks ??. If you append a particular command to this, Mathematica will give you specific guidance to the use of that particular function with ready examples you can test out.
* You can also take the slow lane and seek for details in the Mathematica reference book installed in every installation. This is the best way to learn for most people.

Linear Interpolation

* The simple walking problem of last week was solved by you using linear interpolation.

```
ChurchMove := {{0, 0}, {40, 4}}
F1[x_] := InterpolatingPolynomial[ChurchMove, x]
Show[ListPlot[ChurchMove, PlotStyle → Red],
 Plot[F1[x], {x, 0, 60}, PlotStyle → Green, AspectRatio -> Full]]
```



Interpolating Polynomial

* We have only two points. The Interpolating Polynomial gives us a line from which we can find any intermediate time, given a distance and vice versa. We are also able to extrapolate by a simple extension of the same line.
* Suppose a surveyor comes up with ten altitudes and linear positions on a road project such as,

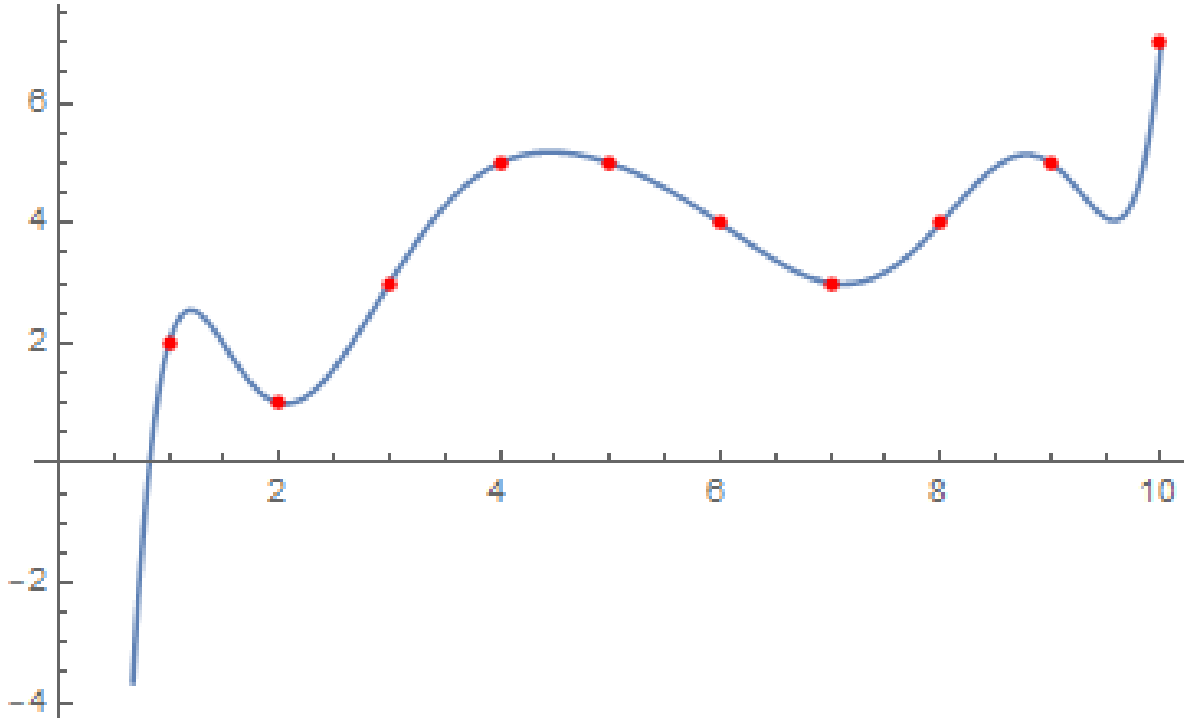
pts:={{1,2},{2,1},{3,3},{4,5},{5,5},{6,4},{7,3},{8,4},{9,5},{10,7}}

* What options do we have to obtain an integrable function? Several! We can,
  * Do a regression and obtain the best fit,
  * Obtain a 9^{th} order polynomial fit.

Runge's Phenomenon

In this particular case, the polynomial is,

$-72.+211.67\,x-240.07\,x^2+143.129\,x^3-50.3851\,x^4+11.0701\,x^5-1.53819\,x^6+0.131432\,x^7-0.0062996\,x^8+0.000129519\,x^9$



Lagrange

* For points $\{\{x_1, y_1\}, \ldots \{x_n, y_n\}\}$, the $(n-1)th$ order Lagrange Interpolating polynomial is computed with,

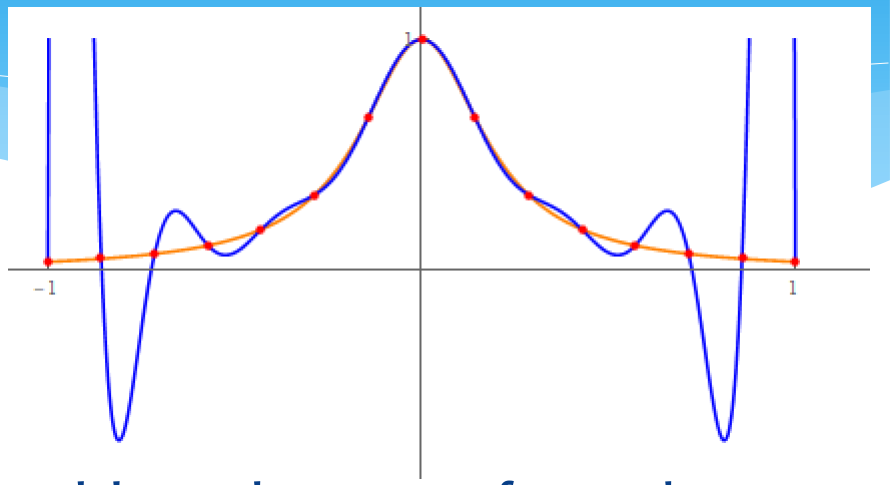
$$P(x) = \sum_{j=1}^{n} P_j(x)$$

* Where,

$$P_j(x) = \prod_{i=1, i \neq j}^{n} \frac{x - x_i}{x_i - x_j}$$

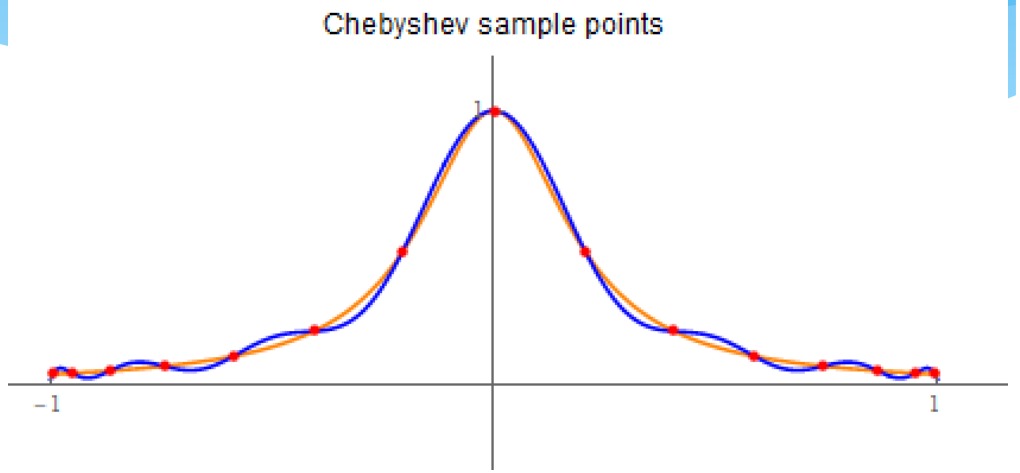
However, it is a well known fact that such interpolations have undesirable oscillating end effects called Runge's phenomenon. A critical case of this is shown below:

Runge's Phenomenon



* This leads to inaccuracies.
* Increasing the order of polynomials used does not necessarily give superior results.
* One strategy to solve this problems borrows from the old drafting craft when manual "splines" were used for curve fitting. The idea is to take a number of nearby points at a time and create regularity conditions using control points.
* This leads to the use of spline interpolation.

Non-Uniform Sampling Points

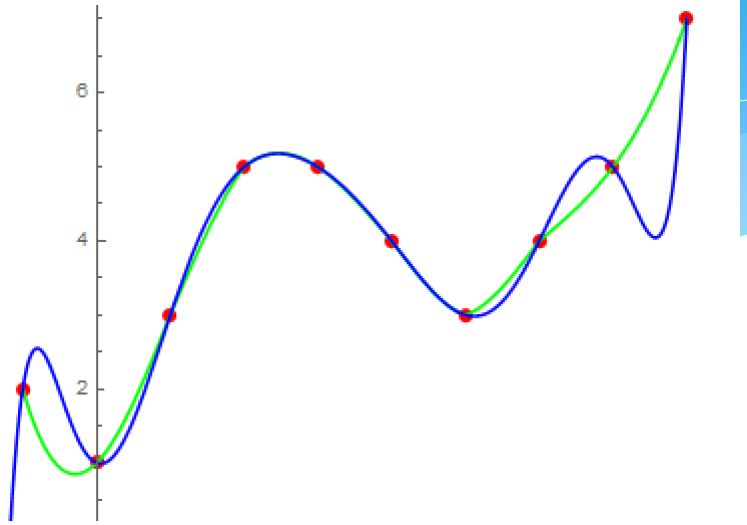


* **Chebychev Points:**

The same interpolation problem we dealt with earlier can be improved upon by using non equally spaced points.

* These Chebychev points are zeros of Chebychev polynomials of the first kind.
* NURBS uses non-equal spacing and splines at the same time.

Spline Fitting

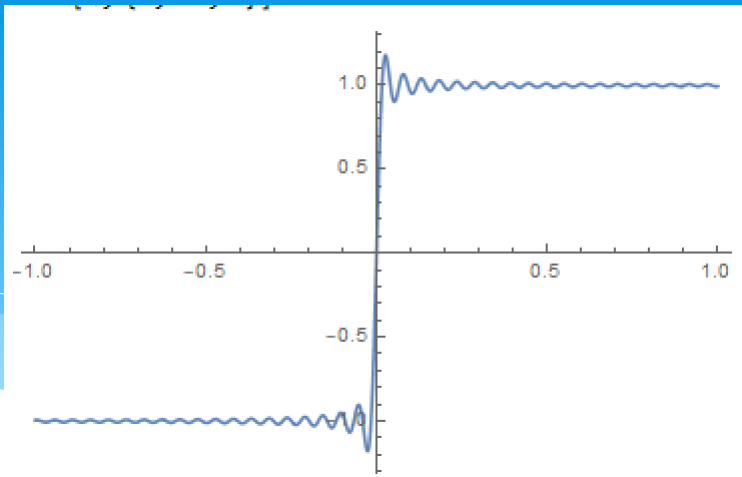


* The green curve is made up of piecewise splines and gives a superior fit. In our Fusion 360, we shall obtain superior control of the splines with tangential and curvature controls.
* This way, we will be able to create 2- and 3-D objects with greater control using spline interpolation.
* B-Splines, Bezier, NURBS and T-Splines are different refinements. Nurbs gains much of its advantage in the use of **N**on-**U**niform **R**ational **B**asis **S**plines. This is because the sampling points are not equally spaced.

Basis Functions

* What do Taylor Series, Fourier Expansion, Lagrange Polynomials, etc. have in common?
* The fact that certain functions are complete in the sense that they can be used to represent other functions to any level of accuracy desired.
* Just like in Cartesian coordinates, vectors i, j, and k form a basis because any other vectors can be represented by them.
* Simple Polynomials, $\{1, x, x^2, \dots\}$ as well as trigonometric functions, $\{\sin x\,, \cos x, \sin 2x\,, \cos 2x\,, \dots\}$ are the basis functions for Taylor and Fourier Series respectively. There are more complicated functions that are also complete. These are all basis functions.

Gibbs Phenomenon



* Earlier in the notes, I showed you that we can fit virtually any function with the Fourier Series
  It is easily demonstrated that the more terms we take, the more accurate we can get.
* Again, just like the polynomial expansion, increasing the terms creates the unwanted Gibbs phenomenon as shown above results from the first 120 terms of the Fourier Expansion. Note the oscillations at the point of discontinuity.
* In Mathematica, InterpolatingPolynomial[] implements the Lagrange while Interpolation[] implements the splines.

Program Files

* Two of the Mathematica Files used in these notes are shared with you:

1. Lagrange Bezier Interpolation https://1drv.ms/u/s!AgbbD-KyVrKGhbgOpfe30Yd_tV2NEw
2. Piecewise https://1drv.ms/u/s!AgbbD-KyVrKGhbgOpfe30Yd_tV2NEw

They are Mathematica notebooks. You can download and work with these files for better understanding.