# The jgraph project

jgraph is a Java Swing project. It is a pure java application written using plain Java 8 features. It does not use any external libraries. It requires a JRE (*version 8 and above*). It allows users to build and plot mathematical expressions (functional, parametric or polar) and apply calculus to the functions. The main application console allows users to select one of the following function types (*blue arrow points at a drop down list*)

* Functional

*Example:* ***y = 2x + cos(x)***

* Parametric

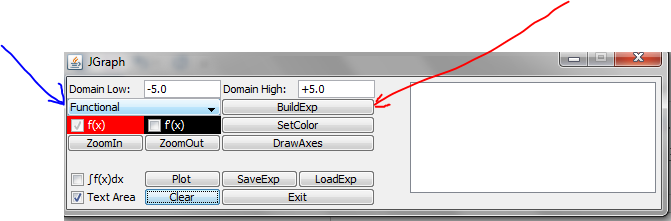
*Parametric form is a pair of equations where x and y are both functions of s*

*Example:* ***y = cos(s)****;* ***x = sin(s)***

* Polar

*This form uses r and Ɵ (r is the distance from origin and Ɵ is the angle that r makes with the horizontal axis)*

*Example:* ***r = 2Ɵ***



After selecting the function type as explained above, a user should click on the button labelled **BuildExp** (*button pointed at by red arrow*) to build expressions through [ExpressionBuilder](#ExpressionBuilder). An expression contains terms – simple or complex.

A simple term has a coefficient and an exponent – for e.g. 2*x*2. In the above case the coefficient is 2 and the exponent is 2. Let’s build this as shown in [case1](#ExpressionBuilder).

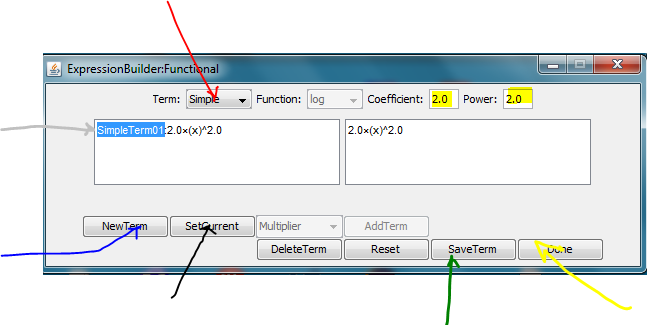
A complex expression could be one of the 3 trigonometric functions (*sin, cos or tan*), the hyperbolic trigonometric functions (*sinh, cosh or tanh*), the inverse trigonometric functions (*arcsin, arccos or arctan*), log10 function (*base 10*), lne function (*base e*), power function (*xx*), square-root, cube-root, ex or ex-1 or a combination of all of the above to make something like shown below (*functional form*):

cos(sin(*x*)) + *x*ln(*x*) + 5*x*

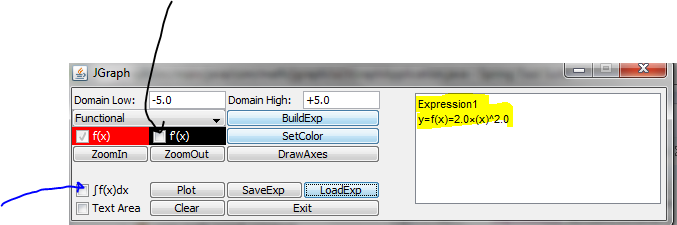
Let’s build this complex expression as shown in [case2](#ExpressionBuilderCase2).

ExpressionBuilder (case 1):

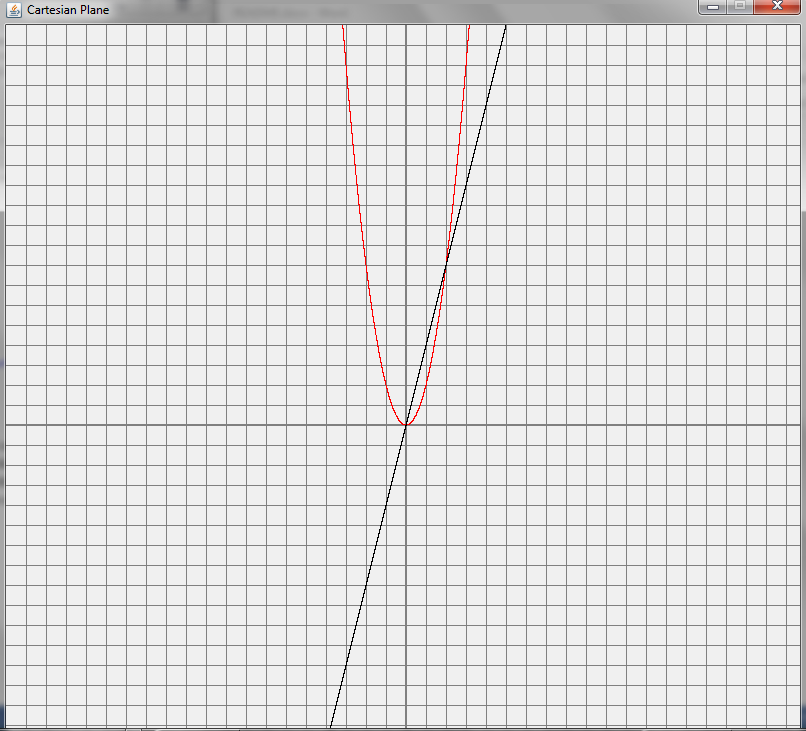
Let’s use **ExpressionBuilder** to build an expression that has a simple term 2*x*2. Click the ‘**BuildExp’** button in the [Fig 1](#Fig1). This will open up a new UI (shown below)



Select ‘**Simple’** from the drop down list (*red arrow*). Edit the ‘**Coefficient’** and the ‘**Power’** to values shown above. Double click on text in the left text area (*grey arrow*) to select the text. Click the button labelled ‘**NewTerm’** (*blue arrow*). The new expression 2*x*2 now appears in the left text area. Click on the button labelled ‘**SetCurrent’** (*black arrow*). The expression is copied in the right text area. Click on button labelled ‘**SaveTerm’** (*green arrow*) followed by clicking on the button labelled ‘**Done’** (*yellow arrow*). The ExpressionBuilder window closes. The saved expression appears in the main console application as shown below.



Click on button labelled ‘**Plot**’ to plot the function on a Cartesian plane frame. Optionally select the checkbox labelled **f’(x)** (*black arrow*) to plot the first derivative of the function. Optionally select the checkbox labelled **ʃ f(x) dx** (*blue arrow*)to calculate and paint the area bounded by the function and the x-axis. See the results on the Cartesian plane below.



The function 2x2 is plotted in red; its derivative d/dx (2x2) is plotted in black. A little intuition shows that by selecting a negative coefficient (in [Fig 1](#Fig1) ) the parabola would appear inverted. Selecting 0 for power (in [Fig 1](#Fig1)) would plot a line parallel to x-axis.

ExpressionBuilder (case 2):

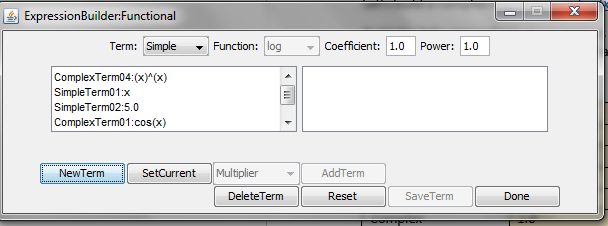
Let’s build a complex expression [cos(sin(*x*)) + *x*ln(*x*) + 5*x*](#ComplexExpression1). Select **Functional** in [Fig 1](#Fig1).

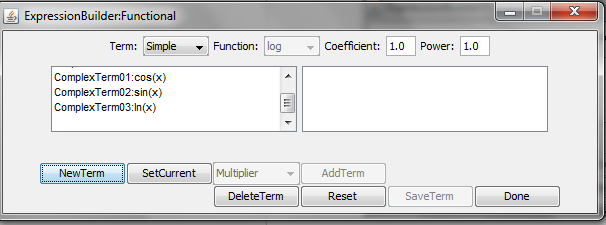
Build the following individual expressions (simple and complex) using the ExpressionBuilder interface. The order in which they are built does not matter.

Table 1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Term** | **Coefficient** | **Power** | **Function** | **Result** |
| Simple | 1.0 | 1.0 | --- | x |
| Simple | 5.0 | **0.0** | --- | 5 |
| Complex | 1.0 | 1.0 | cos | cos(x) |
| Complex | 1.0 | 1.0 | sin | sin(x) |
| Complex | 1.0 | 1.0 | ln | ln(x) |
| Complex | 1.0 | 1.0 | x^x | xx |

For each item select either ‘**Simple’** or ‘**Complex’** as the table shows and click on the button labelled ‘**NewTerm’**

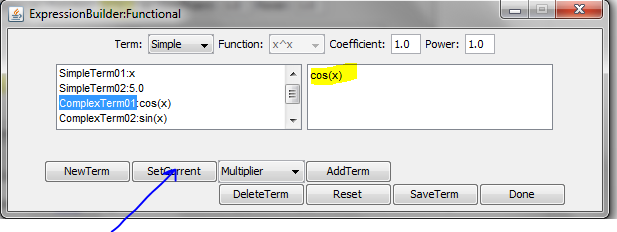




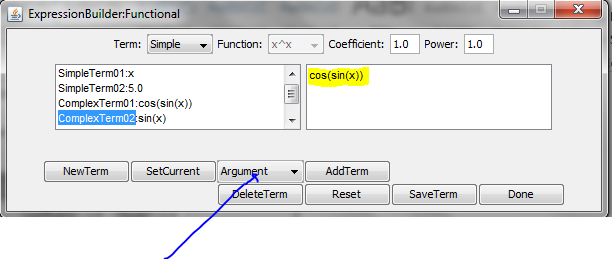
The question is how do we construct **cos(sin(x))** from individual cos(x) and sin(x) functions and similarly 5x from x^x and 5 here.

Select ComplexTerm01 (*by double clicking on it*) in the left text area. Next, click on ‘**SetCurrent’**. This action copies cos(x) in the right text area.

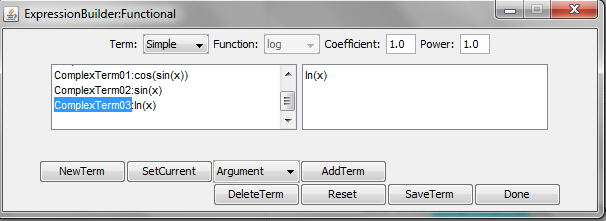
*Note: Depending on the order in which one creates these (simple and complex) terms your complex term cos(x) could have a different position number (in this case it is ComplexTerm****01****)*.



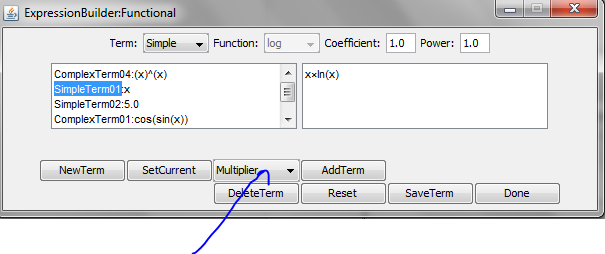
Then select ComplexTerm02 (*by double clicking on it*) in the left area. Select ‘**Argument’** from the drop down list (*blue arrow*). Click on button labelled ‘**AddTerm’** to change cos(s) into cos(sin(x); refer to the right and left text areas. Click on ‘**SaveTerm**’.



Now select ComplexTerm03 (*by double clicking on it*), click in ‘**SetCurrent**’ to copy it into right text area.

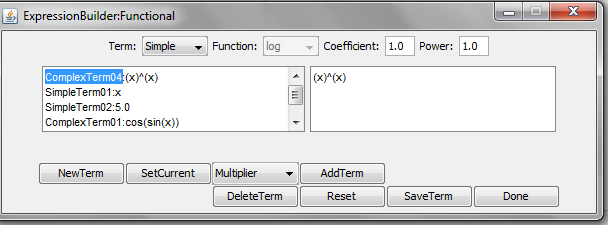


Select SimpleTerm01 (*by double clicking on it*) and select ‘**Multiplier’** from the drop down list (*blue arrow*). Click on ‘**AddTerm’** to change ln(x) into xln(x) (*refer to right and left text areas*). Click on ‘**SaveTerm**’.

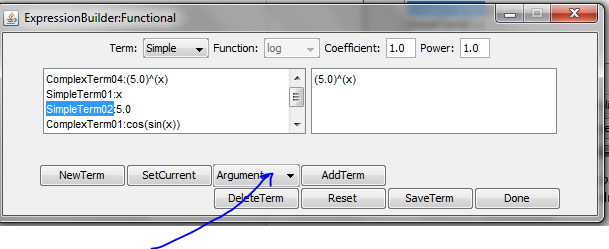


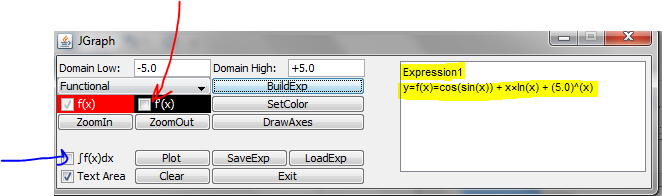
At this point your expression generated so far is cos(sin(x)) + *x*ln(x). We still need the term 5x.

Last step is to create 5x out of x^x and 5. Select ComlexTerm04 (*by double clicking on it*), click on ‘**SetCurrent’**. This should copy x^x (*x raised to x i.e. xx*) to right text area.



Select SimpleTerm02 (*by double clicking on it*) and select ‘**Argument’** from the drop down list (*blue arrow*). Click on ‘**AddTerm’** to change x^x into 5^x (*refer to right and left text areas*). Click on ‘**SaveTerm**’.



Click on the button labelled ‘**Done’** to close the ExpressionBuilder UI. The control is returned back to the main application console frame. It shows the recently built expression in the text area (*as shown*). 

Optionally you could check the checkbox labelled f’(x) (*see red arrow*) for plotting the first derivative of the function and the checkbox labelled **ʃ f(x) dx** (*blue arrow*)to calculate and paint the area bounded by the function and the x-axis. Click on Plot to see the results on the Cartesian plane (*see below*).

