Extending WolframAlpha Functionality

WolframAlpha, http://www.wolframalpha.com, is a wonderful free access online computation program. WolframAlpha can take a variety of "verbal" inputs, e.g.

(i) "integrate x squared dx"

(ii) "temperature Ithaca, NY May 19, 2013 at 10:30"

(iii) "benzene molecule"

(iv) "smallest planet"

(v) "solve 2x-cos(x)+6=17"

(vi) "definition of syzygy"

However, many students primarily use WolframAlpha as a mathematical problem solving tool. While the website takes "verbal" inputs, this can lead to confusion and increases computational time (which is limited for free users). WolframAlpha's abilities can be greatly extended using Mathematica commands. Here we give a sample of commands that WolframAlpha will take that would otherwise be difficult to communicate to the program otherwise. [Note depending on the input, some of these commands will still time out. Some are also finicky and will/won't work from time to time.]

Functions

`Exp[number]` : Gives e^x.

```
Exp[1] = E
```

`Log[b,number]` : Gives the log base b of the number.

```
Log[10,100] = 2
```

`Log10[number]` : Gives the base-10 log of the number.

```
Log10[100] = 2
```

`Log[number]` : Gives the log base e of the number, i.e. $\ln$.

```
Log[E] = 1
```

`Sin[number]` : Gives $\sin x$.

```
Sin[Pi] = 0
```

Note that you use a similar definition to obtain $\cos x, \tan x, \sec x, \csc x, \cot x, \sinh x, \cosh x, \tanh x$, et cetera. Also note that one uses, Pi for π, E for e, and `I` for i.

`Gamma[number]` : Gives $\Gamma(x)$.

```
Gamma[1/2] = sqrt(pi)
```

Numbers

`N[number,digits]` : Gives the inputed number to a given number of digits.

```
N[Pi,30] = 3.14159265358979323846264338328
```

`Abs[number]` : Gives the absolute value of the inputed integer.

```
Abs[-1729] = 1729
```

`Floor[number]` : Gives the closest integer less than the given number.

```
Floor[25.4] = 25
```

`Ceiling[number]` : Gives the closest integer greater than the given number.

```
Ceiling[25.4] = 26
```

`RandomInteger[{min,max}]` : Gives a random integer between the given bounds.

```
RandomInteger[{-5,17}] = -4
```

`RandomReal[{min,max}]` : Gives a random real between the given bounds.

```
RandomReal[{-5,17}] = 12.3818
```

`Divisors[number]` : Gives the divisors of the number.

```
Divisors[1729] = 1, 7, 13, 19, 91, 133, 247, 1729
```

`GCD[number list]` : Gives the greatest common divisor of the numbers.

```
GCD[148,254,388] = 2
```

`LCM[number list]` : Gives the least common multiple of the numbers.

```
LCM[148,254,388] = 1 823 212
```

`Mod[number,divisor]` : Gives the remainder when dividing the number by the divisor.

```
Mod[12536424,7] = 5
```

`Prime[number]` : Gives the nth prime.

```
Prime[1729] = 14 759
```

`PrimeQ[number]` : Determines if the number is prime or not.

```
PrimeQ[1387] = False
```

`PrimePowerQ[number]` : Determines if the number is a power of a prime or not.

```
PrimePowerQ[1884] = False
```

`SquareFreeQ[number]` : Determines if the number is divisible by the square of an integer.

```
SquareFreeQ[12] = False
```

`Divisible[number,divisor]` : Determines if the number is divisible by the divisor.

```
Divisible[12,4] = True
```

`CoprimeQ[number,number]` : Determines if the numbers are relatively prime.

```
CoprimeQ[142,98] = False
```

`FactorInteger[integer]` : Factors the given integer.

```
FactorInteger[235235] = 5 x 7 x 11 x 13 x 47
```

`IntegerExponent[number, divisor]` : Gives the highest power of the divisor that divides the number.

```
IntegerExponent[32640,2] = 7
```

`IntegerDigits[number,base]` : Gives the digits of the number in the given base.

```
IntegerDigits[18,2] = {1,0,0,1,0}
```

`DigitCount[number,base,digit]` : Gives the number of times the digit appears in the number when written in the given base (leave out the digit to get a list of digits).

```
DigitCount[18,3,2] = 1
```

`FromDigits[digits,base]` : Gives the number in base 10 from the digits of the number in the given base.

```
FromDigits[{1,0,1,1},2] = 11
```

`Rationalize[number]` : Gives the fraction form of the given number.

```
Rationalize[12.45] = 249/20
```

Equations

`Solve[equation,variable]` : Solves the equation with respect to the variable.

```
Solve[x^2-2==4,x] = -2, 2

Solve[x^2-2<2,x] = -2 < x < 2
```

`NSolve[equation,variable]` : Numerically solves the equation with respect to the variable.

```
NSolve[x^2Sin[x]-21==2,x] = 25.1327
```

`FindRoot[function,variable,point]` : Finds a root of the given function near the given point.

```
FindRoot[x^2 - Sin[x]+2,{x,1}] = 0.450185
```

`DSolve[equation,function,variable]` : Solves the differential equation.

```
DSolve[y'[x]==2y[x],y[x],x] = C[1]Exp[2x]
```

Polynomials

`Apart[function]` : Separates a rational function.

```
Apart[1/((x-2)(x-3))] = 1/(x-3) - 1/(x-2)
```

`Together[function]` : Combines terms into a common denominator

```
Together[1/(x-5)+1/(x-3)]=(2x-8)/((x-5)(x-3))
```

`Expand[function]` : Expands out a product.

```
Expand[(x+2)^3] = x^3 + 6x^2 + 12x + 8

Expand[(x-2)(x+3)] = x^2 + x - 6
```

`Factor[function]` : Factors a function.

```
Factor[x^2+x-6] = (x - 2)(x + 3)
```

`GroebnerBasis[polynomials,variables]` : Finds a Gröbner basis for the given polynomials in the given variables.

```
GroebnerBasis[{x^2-y^2,x^3+2y},{x,y}] = {-4y + y^5, 2xy + y^4, x^2-y^2}
```

`PolynomialGCD[polynomials]` : Finds the greatest common divisor of the given polynomials.

```
PolynomialGCD[x^2+4x-12,x^2-x-2] = x - 2
```

`PolynomialLCM[polynomials]` : Finds the least common multiple of the given polynomials.

```
PolynomialLCM[x^2+4x-12,x^2-x-2] = x^3 - 7x^2 + 4x + 12
```

`PolynomialQuotient[polynomial,divisor,variable]` : Finds the quotient of the result of dividing the polynomial by the given divisor.

```
PolynomialQuotient[x^3+2,x-1,x] = x^2 + x + 1
```

`PolynomialRemainder[polynomial,divisor,variable]` : Finds the remainder upon dividing the polynomial by the given divisor.

```
PolynomialRemainder[x^3+2,x-1,x] = 3
```

`PolynomialMod[polynomial,base]` : Gives the polynomial in the given base.

```
PolynomialMod[x^3+2x^2+x+2,2] = x^3 + x
```

`IrreduciblePolynomialQ[polynomial]` : Determines if the polynomial is irreducible over the rationals.

```
IrreduciblePolynomialQ[x^2-3] = False
```

`Discriminant[polynomial,variable]` : Gives the discriminant of the polynomial

```
Discriminant[a x^2+b x+c,x] = b^2 - 4ac
```

`Decompose[polynomial,variable]` : Writes a polynomial as a composition of polynomials if possible.

```
Decompose[x^3+1,x] = {x+1,x^3}
```

`Coefficient[polynomial,variable,power]` : Gives the coefficient of the given power of x in the given polynomial.

```
Coefficient[(x+3)^8,x,4] = 5670
```

`CoefficientList[polynomial,variable]` : Gives the coefficients of the polynomial starting at the 0th power.

```
CoefficientList[(x+2)^2,x] = {4,4,1}
```

`Cancel[function]` : Cancels terms in a rational function, if possible.

```
Cancel[(x^2-4)/(x-2)] = x + 2
```

`Numerator[polynomial]` : Give the numerator of a number or function.

```
Numerator[2/3] = 2
```

```
Numerator[(x-2)/(x+3)] = x - 2
```

`Denominator[polynomial]` : Gives the denominator of a number or function.

```
Denominator[3/5] = 5
```

```
Denominator[(x-2)/(x+3)] = x + 3
```

Calculus

`D[function,variable]` : Gives the derivative of the function with respect to the given variable(s).

```
D[x^3,x] = 3x^2
```

```
D[x^3,{x,2}] = 6x
```

```
D[x^2y^3,x,y] = 6xy^2
```

`D[function,variable]` : Gives the integral with respect to the given variable(s). Note for multiple integrals, the order of integration should be given to Wolfram in the *reverse* order normally written, e.g. $dx\ dy\ dz$ would be given z first, y next, and x last.

```
Integrate[x Exp[x],{x,0,1}] = 1
```

```
Integrate[20xy^2,{y,0,1},{x,0,y}] = 2
```

`Sum[sequence,variable,start,end]` : Finds the sum of the sequence from the start to end term.

```
Sum[1/2^n,{n,0,6}] = 127/64
```

```
Sum[1/2^n,{n,0,Infinity}] = 2
```

`NSum[sequence,n,start,end]` : Numerically finds the sum of the sequence from the start to the end term.

```
NSum[Sin[n],{n,1,20}] = 0.99822188441978185
```

`Series[function,variable,center,order]` : Finds the series for the function up to the given order at the given center.

```
Series[Sin[x],{x,0,4}]
```

`FourierSeries[function,variable,n]` : Gives the nth order Fourier series expansion for the given function.

```
FourierSeries[x^2/2,x,5]
```

`Product[sequence,variable,start,end]` : Gives the product of the terms of the sequence from the start to the end.

```
Product[3^x,{x,1,3}] = 729
```

`Limit[function,x->point]` : Gives the limit of the function as the variable goes to the point.

```
Limit[Sin[x]/x,x->0] = 1
```

`SumConvergence[series,variable]` : Determines whether the given series converges.

```
SumConvergence[1/n^2,n] = converges
```

`FindMinimum[function,variable,point]` : Finds a minimum of the given function near the given point.

```
FindMinimum[x^2+1,{x,1/2}] = 1
```

`FindMaximum[function,variable,point]` : Finds a maximum of the given function near the given point.

```
FindMaximum[1-x^2,{x,1/2}]=1
```

`NMinimize[function,variable]` : Finds the minimum value for the given function.

```
NMinimize[1+x^2,x] = {1,{x->0}}
```

`NMaximize[function,variable]` : Finds the maximum value for the given function.

```
NMaximize[1-x^2,x] ={1,{x->0}}
```

`Table[sequence,variable,start,end]` : Gives a table of the sequence from the start to the end.

```
Table[1/n^2,{n,1,5}] = 1, 1/4, 1/9, 1/16, 1/25
```

Plotting

`Plot[function,variable,min,max]` : Plots the given function between the given bounds.

```
Plot[Sin[x],{x,0,Pi}]
```

`Plot3D[function,x,min,max,y,min,max]` : Plots the given function between the given bounds.

```
Plot3D[Sin[x y],{x,0,2Pi},{y,0,2Pi}]
```

`RegionPlot[inequality,x,min,max,y,min,max]` : Plots the inequality over the bounds given.

```
RegionPlot[x^2+y^2<1,{x,-1,1},{y,-1,1}]
```

`ParametricPlot[xfunction,yfunction,t,min,max]` : Plots the parametric function between the given bounds.

```
ParametricPlot[{Cos[t],Sin[t]},{t,0,2Pi}]
```

`StreamPlot[xfunction,yfunction,x,min,max,y,min,max]` : Plots the stream plot for the given vector field.

```
SreamPlot[{-y,x},{x,-5,5},{y,-5,5}]
```

`VectorPlot[xfunction,yfunction,x,min,max,y,min,max]` : Plots the vector field over the given region.

```
VectorPlot[{-y,x},{x,-5,5},{y,-5,5}]
```

`VectorPlot3D[function,yfunction,zfunction,x,min,max,y,min,max,z,min,max]` : Plots the vector field over the given region.

```
VectorPlot3D[{-x,y,-z},{x,-3,3},{y,-3,3},{z,-3,3}]
```

`ParametricPlot3D[xfunction,yfunction,zfunction,s,min,max,t,min,max]` : Plots the parametric function over the given region.

```
ParametricPlot3D[{Cos[t],Sin[t],t},{t,0,2Pi}]
```

```
ParametricPlot3D[{Cos[t]Sin[s],Sin[t]Sin[s],Cos[s]},{s,0,Pi},{t,0,2Pi}]
```

`LogPlot[function,variable,min,max]` : Gives the log plot of the given function.

```
LogPlot[2^x,{x,0,10}]
```

`LogLinearPlot[function,variable,min,max]` : Gives the log-linear plot of the given function.

```
LogLinearPlot[2^x,{x,1,10}]
```

`LogLogPlot[function,variable,min,max]` : Gives the log log plot of the function.

```
LogLogPlot[Exp[2^x],{x,0,20}]
```

`ContourPlot[function,x,min,max,y,min,max]` : Gives the contour plot of the function over the given region.

```
ContourPlot[Sin[x y],{x,-Pi,Pi},{y,0,2Pi}]
```

`ContourPlot3D[function,x,min,max,y,min,max,z,min,max]` : Gives the contour plot of the function over the given region.

```
ContourPlot3D[x^2+y^2+z^2,{x,-2,2},{y,-2,2},{z,-2,2}]
```

```
ContourPlot3D[x^2+y^2+z^2==1,{x,-2,2},{y,-2,2},{z,-2,2}]
```

`Piecewise[value,condition,....]` : Gives the piecewise function over the given values and conditions.

```
Plot[Piecewise[{{Sin[x],x>=0},{-x,x<0}}],{x,-2,2}]
```

Linear Algebra

`Cross[vector,vector]` : Gives the cross product of the two vectors.

```
Cross[{1,1,3},{4,-1,2}] = {5,10,-5}
```

`Dot[vector,vector]` : Gives the dot product of the two vectors.

```
Dot[{1,1,3},{4,-1,2}] = 9
```

`VectorAngle[vector,vector]` : Gives the angle between the two vectors.

```
VectorAngle[{1,0,1},{0,-1,0}] = Pi/2
```

`Projection[vector,vector]` : Gives the vector projection of the first vector onto the second.

```
Projection[{2,4,-1},{3,3,4}] = {21/17,21/17,28/17}
```

`Normalize[vector]` : Normalizes the given vector.

```
Normalize[{3,3,5}] = {3/sqrt(43),3/sqrt(43),5/sqrt(43)}
```

`RowReduce[matrix]` : Row reduces a matrix.

```
RowReduce[{{1,3,5},{-1,4,2},{1,2,2}}] = {{1,0,0},{0,1,0},{0,0,1}}
```

`Transpose[matrix]` : Finds the transpose of the matrix.

```
Transpose[{{1,2},{3,5}}] = {{1,3},{2,5}}
```

`Nullspace[matrix]` : Gives the nullspace of the matrix.

```
Nullspace[{{1,1},{1,1}}] = {{-1,1}}
```

`MatrixRank[matrix]` : Gives the rank of the matrix.

```
MatrixRank[{{1,1},{1,1}}] = 1
```

`Eigenvalues[matrix]` : Gives the eigenvalues of the matrix.

```
Eigenvalues[{{1,1},{1,1}}] = {2,0}
```

`Eigenvectors[matrix]` : Gives the eigenvectors of the matrix.

```
Eigenvectors[{{1,1},{1,1}] = {{1,1},{-1,1}}
```

`Det[matrix]` : Gives the determinant of the matrix.

```
Det[{{1,1},{1,1}}] = 0
```

`PseudoInverse[matrix]` : Gives the pseudo-inverse of the matrix.

```
PseudoInverse[{{1,1},{1,1}}] = {{1/4,1/4},{1/4,1/4}}
```

`LinearSolve[A,b]` : Solves the equation $Ax = b$ for x.

```
LinearSolve[{{1,-1},{1,1}},{2,3}] = {5/2,1/2}
```

`MatrixPower[matrix,power]` : Gives the given power of the given matrix.

```
MatrixPower[{{1,2},{1,1}},4] = {{7,10},{5,7}}
```

`JordanDecomposition[matrix]` : Gives the Jordan form to the matrix. The output is { similar matrix, jordan form}.

```
JordanDecomposition[{{1,-1},{-2,1}}]
```

`CharacteristicPolynomial[matrix]` : Gives the characteristic polynomial for the matrix.

```
CharacteristicPolynomial[{{1,1},{1,1}}] = x^2 - 2x
```

`ConjugateTranspose[matrix]` : Gives the conjugate transpose of the matrix.

```
ConjugateTranspose[{{1,3},{-2,4}}] = {{1,-2},{3,4}}
```

`Tr[matrix]` : Gives the trace of a matrix.

```
Tr[{{1,2},{-3,5}}] = 6
```

Statistics

`Mean[numbers]` : Finds the average of the given numbers.

```
Mean[{1,3,5,7,9}] = 5
```

`Variance[numbers]` : Finds the variance of the given numbers.

```
Variance[{1,3,5,7,8}] = 10
```

`StandardDeviation[numbers]` : Finds the standard deviation of the given numbers.

```
StandardDeviation[{1,3,5,7,9}] = sqrt(10)
```

`Median[numbers]` : Finds the median of the given numbers.

```
Median[{1,3,5,7,9}] = 5
```

`Quantile[numbers,n]` : Finds the nth quantile of the list of numbers.

```
Quantile[{1,3,5,7,9},2] = 5
```

`Commonest[numbers]` : Finds the mode of a list of numbers.

```
Commonest[{1,1,3,5,7,9}] = 1
```

`RootMeanSquare[numbers]` : Finds the root mean square of a list of numbers.

```
RootMeanSquare[{1,3,5,7,9}] = sqrt(33)
```

`GeometricMean[numbers]` : Finds the geometric mean of a list of numbers.

```
GeometricMean[{1,2,4,4,8}] = 2^(8/5)
```

`InterquartileRange[numbers]` : Finds the interquartile range (IQR) of the list of numbers.

```
InterquartileRange[{1,3,5,7,9,11}] = 6
```

`Skewness[numbers]` : Finds the skewness of a list of numbers.

```
Skewness[{1,3,5,7,9}] = 0
```

`Kurtosis[numbers]` : Finds the kurtosis of a list of numbers.

```
Kurtosis[{1,3,5,7,9}] = 17/10
```

`Sort[numbers]` : Sorts a list of numbers.

```
Sort[{3,9,7,1,5}] = {1,3,5,7,9}
```

`Tally[numbers]` : Finds the tallies of a list of numbers.

```
Tally[{1,1,3,5,5,7,7,7}] = {{1,2},{3,1},{5,2},{7,3}}
```

`Differences[numbers]` : Finds the difference between adjacent members in the list of numbers.

```
Differences[{1,4,6,5,9}] = {3,2,-1,4}
```

`Accumulate[numbers]` : Finds the accumulation of the numbers in a list.

```
Accumulate[{1,4,6,5,9}] = {1,5,11,16,25}
```

`Min[numbers]` : Finds the minimum of a list of numbers.

```
Min[{1,4,6,5,9}] = 1
```

`Max[numbers]` : Finds the maximum of a list of numbers.

```
Max[{1,4,6,5,9}] = 9
```

`Total[numbers]` : Finds the sum of a list of numbers.

```
Total[{1,4,6,5,9}] = 25
```

`Fit[points,polynomial form,variable]` : Finds best polynomial to fit the points.

```
Fit[{ {1,1}, {4,2}, {6,-1} },{1,x,x^2},x] = -0.366x^2 + 2.166 x - 0.8
```

`FindFit[points, expression,constants,variable]` : Find the best fit for the points given the form.

```
FindFit[{ {1,1},{2,4},{3,5} },a Log[b x]+c,{a,b,c},x]={a->3.71,b->1.20,c->0.42}
```

`Binomial[n,m]` : Finds the binomial coefficient n choose m.

```
Binomial[5,3] = 10
```

`Multinomial[numbers]` : Gives the multinomial coefficient of the sum of the values.

```
Multinomial[2,3,3] = 560
```

`Factorial[n]` : Gives the value n!

```
Factorial[4] = 24
```

`Factorial2[n]` : Gives the double factorial n!!

```
Factorial2[6] = 48
```

`RandomChoice[numbers]` : Gives a random element from a list of numbers.

```
RandomChoice[{-1,4,2,2,13}] = 2
```

`RandomSample[numbers,samplesize]` : Gives a random sample from a list of numbers with no number chosen more than once.

```
RandomSample[{-1,4,2,2,13},2] = {13,4}
```

Complex Analysis

`Re[number]` : Gives the real part of the complex number.

```
Re[2+3I] = 2
```

`Im[number]` : Gives the imaginary part of the complex number.

```
Im[2+3I] = 3
```

`Conjugate[number]` : Gives the complex conjugate of the number.

```
Conjugate[2+3I] = 2 - 3I
```

`Arg[number]` : Gives the argument of the complex number.

```
Arg[2+2Pi] = Pi/4
```

`Residue[function,variable,point]` : Gives the residue of the function at the given point.

```
Residue[1/Sin[Pi z], {z,0}] = 1/Pi
```