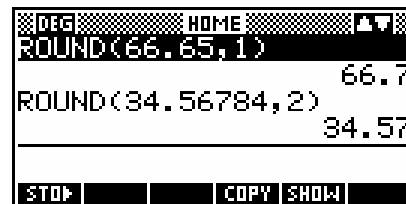# My favourite functions from the MATH menu

## *ROUND(<num>,<dec.pts>)*

This function rounds off a supplied number to the specified number of decimal places (d.p.).

```
DEG        HOME
ROUND(66.65,1)
                     66.7
ROUND(34.56784,2)
                    34.57

STO▶            COPY SHOW
```

Eg.   Round 66.65 to 1 d.p.
         Use: **ROUND(66.65,1)** = 66.7
         Round 34.56784 to 2 d.p.
         Use: **ROUND(34.56784,2)** = 34.57

This function is also capable of rounding off to a specified number of significant figures (s.f.). To do this, simply put a negative sign on the second argument.

```
DEG        HOME
ROUND(32345,-2)
                    32000
ROUND(3405.63475,-6)
                  3405.63

STO▶
```

Round 32345 to the nearest thousand.
Use: **ROUND(32345,-2)** = 32000
Round 3405.63475 to 6 s.f.
Use: **ROUND(3405.63475,-6)** = 3405.63
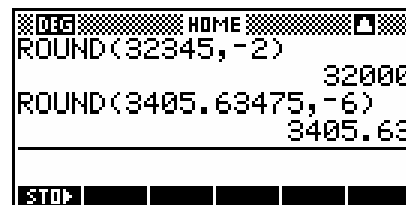
PREDY(<x-value>)

This function predicts the y value for a pair of columns set up as bivariate data in the Statistics aplet. This is discussed in more detail in the section covering the Statistics aplet, but a brief summary will be given here.

It assumes that:
   i.   the bivariate data is entered into a pair of columns (eg. **C1** and **C2**, with **C1** containing the independent data and **C2** the dependent data),
   ii.   these columns have been specified in the **SYMB** view to be paired bivariate data,
   iii.   this data set has been graphed in the **PLOT** view and that the **FIT** screen key has been used to plot the line of best fit for the pair of columns,

If these conditions are satisfied then the function **PREDY(3.5)** will produce a predicted y (dependent) value for the x (indep.) value of $3\cdot5$.

```
DEG        HOME
√(3²-4)
              2.2360679775
PREDY(3.5)
              36.0763157895

STO▶
```
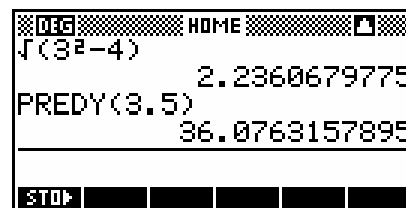
Residuals can easily be produced by going to the **HOME** view and performing the calculation **C2 – PREDY(C1)▶C3**. This subtracts the predicted values from the *y* values and places the residuals into column **C3**.

ABS(<real>) or ABS(<complex>)

The absolute function, which is found on the keyboard above the
left bracket key, returns the absolute value of a real number.

Eg.   **ABS(-3)** returns a value of 3.
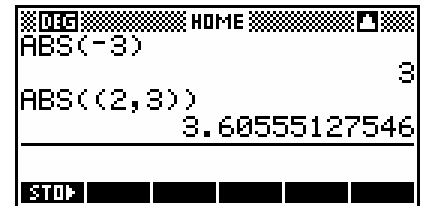
When you use the absolute function on a complex number $a + bi$
it returns the magnitude of the complex number as $\sqrt{a^2 + b^2}$. Note the requirement for
double brackets in this case. The outer pair are those of the function **ABS(...)**. The inner
pair are those of the number in *(a,b)* format.

SIGN(<real>) or SIGN(<complex>)

This function is found in the Real group not the Complex group
but is very useful with complex numbers and so is also covered
here. If given a vector/complex number *(a,b)*, **SIGN** will return
another vector/complex number which is a unit vector in the
direction of *(a, b)*. Note again the requirement for doubled
brackets.

i.e. **SIGN((A,B))** returns $\left(\dfrac{a}{\sqrt{a^2 + b^2}}, \dfrac{b}{\sqrt{a^2 + b^2}} \right)$.

This is very useful, not just in complex numbers, but also in vector problems.
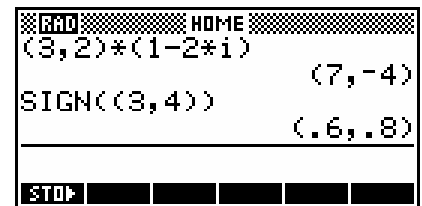
The 'Convert' group of functions

This is a set of functions which allow conversion of metric units
to imperial and vice versa. The list is not very extensive but does cover the more
common ones. The syntax is the same for all.

For example, the →**C** function converts to Celsius from the
equivalent Fahrenheit. To convert 212 degrees Fahrenheit to
Celsius just type →**C(212)**.   To convert 100 kg to pounds, use
→**LBS(100)** as shown in the screen shot.
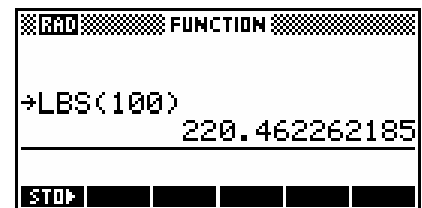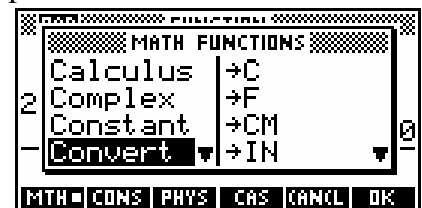
Available functions are:

→**C**	- converts Fahrenheit to Celsius	→**F**	- converts Celsius to Fahrenheit
→**CM**	- converts inches to centimeters	→**IN**	- converts centimeters to inches
→**L**	- converts litres to US gallons	→**GAL**	- converts US gallons to litres
→**KG**	- converts pounds to kilograms	→**LBS**	- converts kilograms to pounds
→**KM**	- converts miles to kilometers	→**MILE**	- converts kilometers to miles
→**DEG**	- converts radians into degrees	→**RAD**	- converts degrees into radians

RREF(<matrix>)

This function takes an augmented matrix of size n by n+1 and transforms it into reduced row echelon form, with the final column containing the solution.

Eg.   The system of equations

$$\left.\begin{array}{l} x - 2y + 3z = 14 \\ 2x + y - z = -3 \\ 4x - 2y + 2z = 14 \end{array}\right\}$$

is written as the augmented matrix

$$\begin{bmatrix} 1 & -2 & 3 & | & 14 \\ 2 & 1 & -1 & | & -3 \\ 4 & -2 & 2 & | & 14 \end{bmatrix}$$

M1	1	2	3	4
1	1	-2	3	14
2	2	1	-1	-3
3	4	-2	2	14

1

| EDIT | INS | GO→ | BIG | | |

which is then stored as a 3x4 real matrix **M1**.

```
▓DIG▓▓▓▓▓▓▓▓ HOME ▓▓▓▓▓▓

RREF(M1)▶M2
[[1,0,0,1],[0,1,0,-2]…

STO▶
```

We now use the function **RREF** to change this to reduced row echelon form and store it as **M2**.

This gives the final result shown in the matrix **M2** on the right, giving a solution of (1, -2, 3).

M2	1	2	3	4
1	1	0	0	1
2	0	1	0	-2
3	0	0	1	3

1

| EDIT | INS | GO→ | BIG | | |

The huge advantage of this function is that it allows for inconsistent matrices which can't be solved by an inverse matrix.

For example, suppose we use the system of equations below, in which the third equation is a linear combination of the first two but the constant is not consistent with this - ie no solution.

$$\begin{cases} x + y + z = 5 \\ 2x - y = -6 \\ 3y + 2z = 13 \end{cases}$$ If we solve this in the same way as before, the matrix which results is:

M2	1	2	3	4
1	1	0	.333333	0
2	0	1	.666667	0
3	0	0	0	1

1

| EDIT | INS | GO→ | BIG | | |

The final line of [0 0 0 1] indicates no solution.
A final line of [0 0 0 0] would have indicated infinite solutions.

TRN(<matrix>)

This function returns the transpose of an n x m matrix.

For example, if $M1 = \begin{bmatrix} 2 & 3 \\ 1 & -2 \\ 0 & 4 \end{bmatrix}$ then TRN(M1) would return $\begin{bmatrix} 2 & 1 & 0 \\ 3 & -2 & 4 \end{bmatrix}$.

DET(<matrix>)

This function finds the determinant of a square matrix. See page **Error! Bookmark not defined.** for an example of its use in finding an inverse matrix.

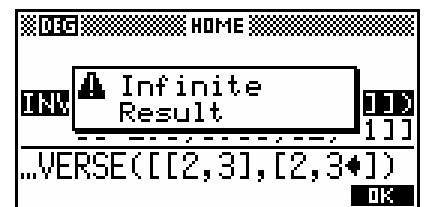Eg.　If $A = \begin{bmatrix} 2 & 3 \\ -1 & 5 \end{bmatrix}$ then find det(A).

　　　Ans:　det (A) = 2x5-3x(-1)
　　　　　　　　= 13

```
DIG              HOME
DET([[2,3],[-1,5]])
                    13

STO▶
```

INVERSE(<matrix>)

This function produces the inverse matrix of an n x n square matrix, where possible. A fully worked example of the use of an inverse matrix to solve a 3 by 3 system of equations is given in the chapter on Using Matrices on page **Error! Bookmark not defined.** and in Appendix A on page **Error! Bookmark not defined.**.

An error message is given (see right) when the matrix is singular (det. zero).

```
DIG              HOME
INVERSE([[2,3],[4,5]])
   [[-2.5,1.5],[2,-1]]

STO▶
```

```
DIG              HOME
     ⚠ Infinite
INV    Result              1]]
...VERSE([[2,3],[2,3◆])
                    OK
```

Note: Some people write the inverse matrix as a fraction (one over the determinant) multiplied by a matrix, so as to avoid decimals and fractions within the inverse matrix. The calculator does not do this. If you want the matrix with the determinant factored out, then evaluate **DET(matrix)** first, record the fraction and then evaluate **DET(matrix) * INVERSE(matrix)** to obtain (usually) a non-fractional matrix.

i.e.　$A = \begin{bmatrix} 2 & 3 \\ 4 & 5 \end{bmatrix} \Rightarrow A^{-1} = \dfrac{1}{-2}\begin{bmatrix} 5 & -3 \\ -4 & 2 \end{bmatrix}$

```
DIG          HOME      ▣
[[2,3],[4,5]]▶M1
           [[2,3],[4,5]]
DET(M1)
                    -2

STO▶
```

```
DIG          HOME      ▣
DET(M1)
                    -2
DET(M1)*INVERSE(M1)
        [[5,-3],[-4,2]]

STO▶
```

Remember that the inverse matrix is not just the matrix, but the fraction times the matrix.

POLYFORM(<expression>,<var_name>)

This is a very powerful and useful polynomial function. It allows algebraic manipulation and expansion of an expression into a polynomial. The expected parameters for the function are firstly the expression to be expanded, and secondly the variable which is to be the subject of the resulting polynomial. If the expression contains more than one variable then any others are treated as constants.

Eg. 1   Expand $(2x-3)^3 (x-1)^2$

   Result: $8x^5 - 52x^4 + 134x^3 - 171x^2 + 108x - 27$
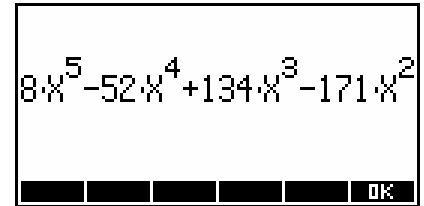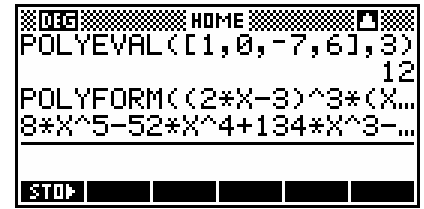
The resulting polynomial is shown both as it appears in the **HOME** view and as it appears after pressing the **SHOW** key. Once it appears in the **SHOW** window, of course, it can be scrolled right and left to see the missing terms.

Eg. 2   Expand $(3a-2b)^4$

This function contains two variables, **A** and **B**, which must be expanded separately.

The first expansion, treating **A** as the variable, is done using the expression **POLYFORM((3A-2B)^4,A)**. As you can see if you examine the view after pressing **SHOW**, the expansion of the expression in terms of **A** has been done, but the terms involving **B** are not fully evaluated.
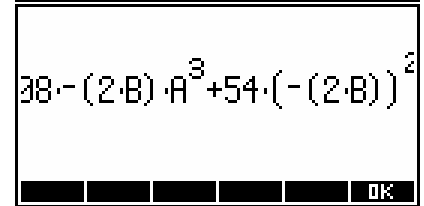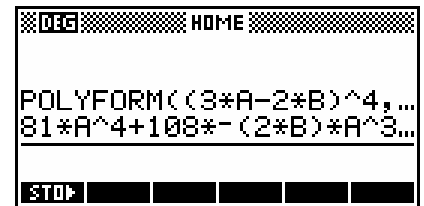
The solution to this is to use **POLYFORM** again. Use the **MATH** menu to fetch the **POLYFORM** function to the edit line, then move the cursor up to the partially evaluated expression that was the result of the previous **POLYFORM**. Copy it into the edit line and add a comma, a **B** and an end bracket. Pressing **ENTER** will now evaluate the terms involving **B**.
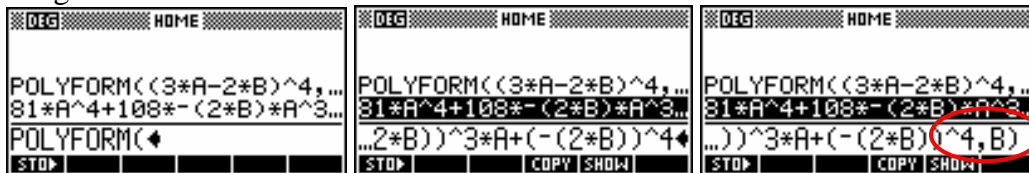
After pressing **ENTER** the for the second evaluation, the result is shown right (after pressing **SHOW**).
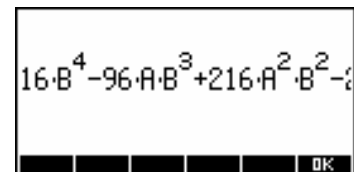
POLYROOT([coeff1,coeff2,…])

This function returns the roots of the polynomial whose coefficients are specified. The coefficients must be input as a vector in square brackets.

```
POLYROOT([1,0,-7,6])
              [1,2,-3]
```

Eg.   Using our earlier function of $f(x) = (x-2)(x+3)(x-1) = x^3 - 7x + 6$ we can enter the coefficients as [1, 0, -7, 6].

As you can see in the screen shot, the roots of 2, -3 and 1 have been correctly found.

See one of the other Tips & Tricks pages for details better ways of doing this and on finding roots of real and complex polynomials using the CAS on the hp 40gs.

COMB(<n>,<r>)

This function gives the value of ^nC_r using the formula $^nC_r = \dfrac{n!}{(n-r)!r!}$.

Eg.   Find the probability of choosing 2 men and 3 women for a committee of 5 people from a pool of 6 men and 5 women.

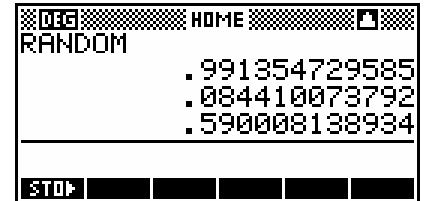$$p = \frac{\dbinom{6}{2}\dbinom{5}{3}}{\dbinom{11}{5}} = 0.3247$$

```
COMB(6,2)
                    15
COMB(6,2)*COMB(5,3)/C…
          .324675324675
```
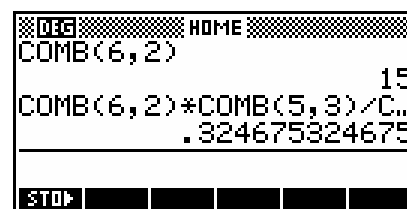
Notes:

1. The reason for the single '**COMB(6,2)**' above the main calculation is to save time. Rather than using the **MATH** menu for every entry of the **COMB** function, you can enter it once and then **COPY** it repeatedly, changing the parameters each time.

2. For large values of **N** (>150 or so) it is important to use the **COMB** function rather than using **N!/((N-R)!*R!)** because the massive values involved in calculating the factorials will cause inaccuracies internally. The **COMB** function has internal methods built in which avoid this and give accurate answers.

3. If you are <u>solving</u> for the value of **N**, such as in a Binomial probability calculation then you must use **N!/((N-R)!*R!)** rather than **COMB**.

RANDOM

This function supplies a random 12 digit number between zero and one. If you want a series of random numbers, just keep pressing **ENTER** after the first one.

```
OIG        HOME        
RANDOM
            .991354729585
            .084410073792
            .590008138934

STO▶
```

Eg. Produce a set of random integers between 5 and 15 inclusive.

Use the expression **INT(RANDOM*11)+5**
The **RANDOM*11** produces a range from **0** to **10.999999**. This is then dropped down to the integer below by the **INT** function, giving a range of integers **0,1,2,3,...,10**. The final adding of **5** gives the correct range.

```
OIG        HOME        
INT(RANDOM*11)+5
                       7
                      14
                      12

STO▶
```

INT(RANDOM*6)+1)

RANDSEED(<number>)

It is important to realize that the values produced by the **RANDOM** command are not truly random numbers. Inside the calculator is a mathematical procedure (an algorithm) which uses a 'seed' number to produce them. Unfortunately, when taken straight out of the box, two calculators will produce exactly the same sequence of "random" numbers! This can be a problem. For example, a class set of calculators doing a simulation of dice might all produce the same numbers.
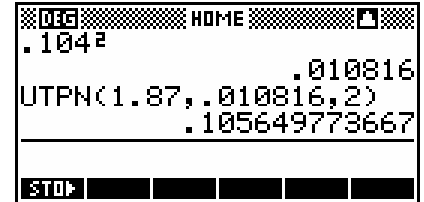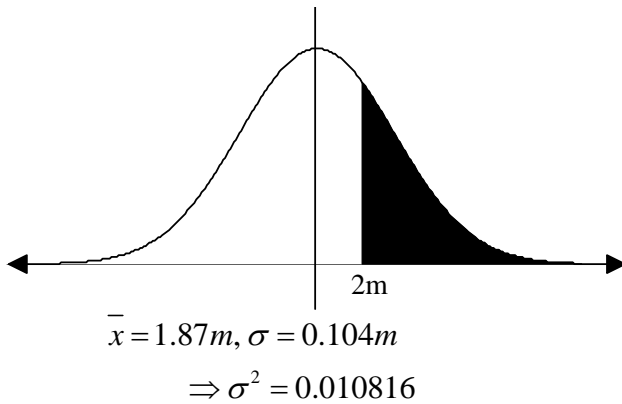
The simple solution is to seed the random number algorithm with different values for each machine. This is done using the function **RANDSEED(...)** and, providing each person uses a different number, it should only be necessary to do it once (unless the calculator is reset). The simplest method is to use **RANDSEED(Time)** since the value of **Time** will be different on each calculator. Since the same seed will produce the same sequence of numbers, this can be convenient for teachers in test situations.

```
RAD        FUNCTION        


RANDSEED(Time)
                       1

STO▶
```
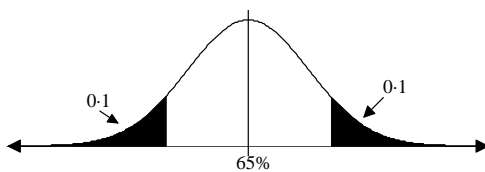
UTPN(<mean>,<variance>,<value>)

This function, the 'Upper-Tail Probability (Normal)', gives the probability that a normal random variable is greater than or equal to the value supplied. Note that the variance must be supplied, NOT the standard deviation.

Eg. 1. Find the probability that a randomly chosen individual is more than 2 meters tall if the population has a mean height of 1.87m and a standard deviation of 10.4cm

```
░░░░░░░░░░░░░░ HOME ░░░░░░░░░░▫░
.104²
                        .010816
UTPN(1.87,.010816,2)
                  .105649773667

STO▶
```

$$\bar{x} = 1.87m, \sigma = 0.104m$$

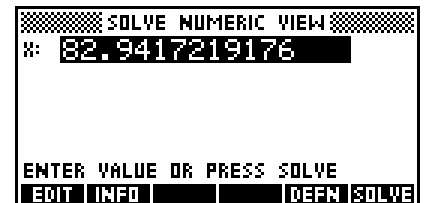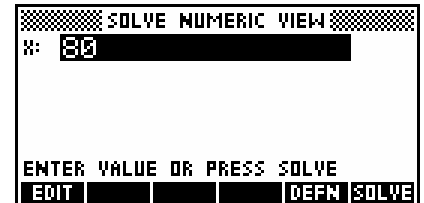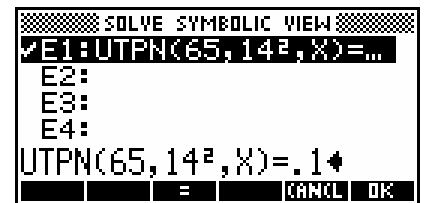$$\Rightarrow \sigma^2 = 0.010816$$

Ans: P(height>2m) = 0.1056

Eg. 2. The population of Year 12 Applicable Mathematics students had a mean exam score of 65% and a standard deviation of 14%. What two scores will cut off the top and bottom 10% of students?

i.e. Find x_0 such that $P(x > x_0) = 0 \cdot 1$

Using the Solve aplet (right) we can reverse the normal direction of the **UTPN** function.

Enter the expression to be solved for into the **SYMB** view as shown above, then switch to the numeric view. Enter a guess of 0.8 (80%) and then press **SOLVE**.

```
░░░░░░ SOLVE SYMBOLIC VIEW ░░░░░░
✔E1:UTPN(65,14²,X)=…
 E2:
 E3:
 E4:
UTPN(65,14²,X)=.1◆
░░░░░░░░░░░░ = ░░CANCL░ OK ░
```

```
░░░░░░ SOLVE NUMERIC VIEW ░░░░░░
X:  80

ENTER VALUE OR PRESS SOLVE
 EDIT ░░░░░░░░░░░ DEFN SOLVE
```

```
░░░░░░ SOLVE NUMERIC VIEW ░░░░░░
X:  82.9417219176

ENTER VALUE OR PRESS SOLVE
 EDIT INFO ░░░░░░░ DEFN SOLVE
```

The second value can be found by using the symmetry properties of the Normal Distribution, but it is probably just as fast to go back to the **SYMB** view, change the 0.1 to 0.9 and then re-**SOLVE**. Remember that an **EDIT** key is provided in the **SYMB** view to allow you to change the expression without having to retype it.

Final answer…   47.06% and 82.94% are the cut-offs.

Calculator Tip
The normal order for the arguments in the **UTPN** function is **UTPN(mean, variance, value)** and this results in the upper-tailed probability. However, many textbooks work with the lower-tailed probability instead. Fortunately the function can easily be adapted for this.

If you instead enter the function's parameters as **UTPN(value, variance, mean)**, reversing the normal positions of *value* and *mean*, then the probability given will be the lower-tailed value. The reason for this lies in the symmetry properties of the normal curve. It's probably easier just to do **1-UTPN(mean, variance, value)** though!