Math Tablet

Scientific Computing for Windows Mobile

User's Guide

Updated May 2, 2007

version 2.1

Contents

Click on underlined text to jump to that section

Additional scripts can be downloaded at www.statsnow.net

Email: support @statsnow.net Web: www.statsnow.net

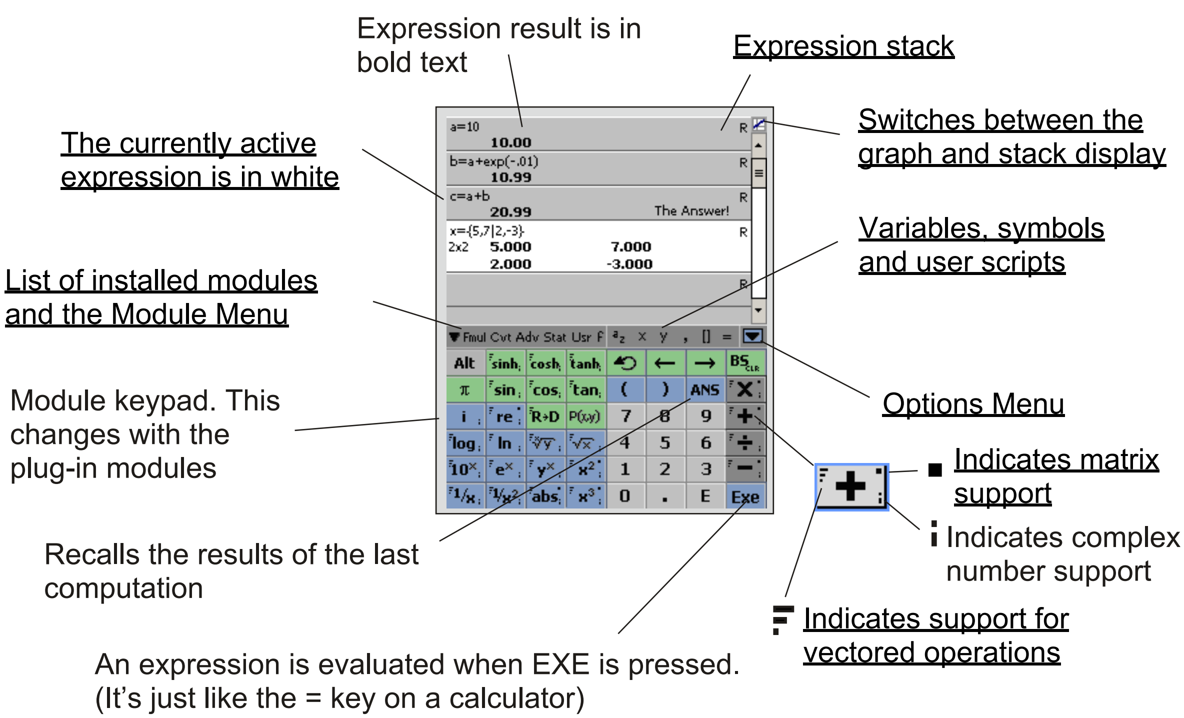
Math Tablet is a programmable scientific analysis package for the PocketPC. It is simple enough to use to balance your checkbook, yet powerful enough to solve your tough numerical problems.

Math Tablet features built in graphing capabilities and a unique expression stack, which keeps track of your calculations and lets you easily modify or correct your work. Math Tablet includes hundreds of scientific functions including matrix operations, complex numbers and hexadecimal calculations.   In addition, Math Tablet's features can be extended by using its built in scripting language or through 3rd party plug-in modules.

Key features of Math Tablet include:

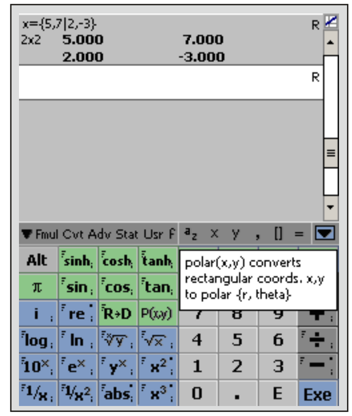
- Algebraic and RPN entry modes
- Matrix and complex number support
- Hundreds of scientific and statistical functions
- Arbitrary bases
- Numerical integrations and differentiation
- Differential equation solvers
- Polynomial and nonlinear equation solvers
- User variables, constants, functions and formulas
- Unit conversion and computations using mixed units
- User defined keyboard layouts
- Statistical tests with multiple data sets
- Graphing of multiple functions
- Results displayed as decimal, hexadecimal or fractions
- Import and exporting of data to comma separated data files
- Vectored operations on data
- A scripting language for creating custom features
- Plug-in support for adding new features to Math Tablet

Math Tablet's screen is divided into two sections. The upper section shows the expression stack or graph. The lower section shows the keypad for entering expressions.



Getting Help

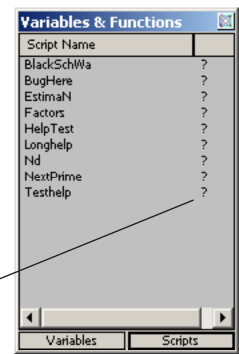
Math Tablet provides four options for obtaining help. These options are described below. The help options are arranged in order from the most convenient to the most complete.



On Screen Popup Help
Most Math Tablet functions support popup help. To see the popup help for a particular function, tap and hold the pen on the key for that function. Use popup help when you can' t remember the parameter for a function, or you forgot the details for using that function. Tap on the popup to cancel the help message, or tap on any key to cancel the help message and continue using Math Tablet. Popup help also works with user defined key created with the User module.

Workspace Help
You can access information about a workspace by selecting the Workspace Info... option from the Options menu. This information is provided by the user or creator of a user script to document the workspace.

You can also view this information by selecting the "?" from the Variable and Function popup.



Tap on the ? to get help
for a particular script

On Line Help
You can view more detailed help on Math Tablet by selecting Help from the Start menu. Math Tablet includes on line help for all of its modules. You can add help files by placing a HTM file inside of Math Tablet's program folder. When you restart Math Tablet it will automatically add the new help to its table of contents.

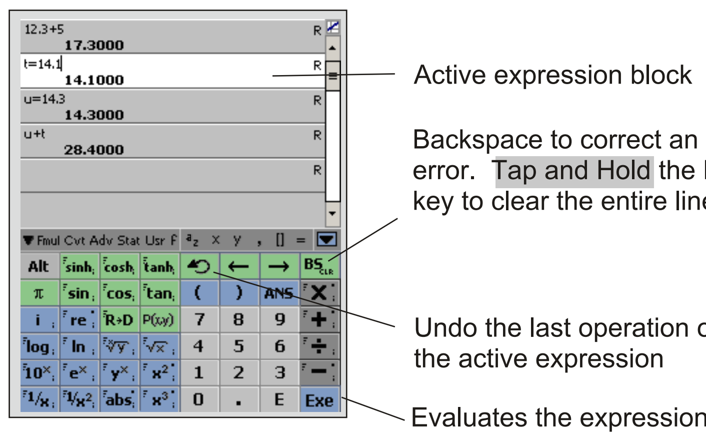
User's Guide
If you still need to learn more about Math Tablet you should consult this manual.

Expressions, or mathematical formulas, are entered onto the expression stack using using the keypad on the half bottom of the screen, or by using one of the built-in entry methods on your handheld - such as handwriting recognition. You can enter your expressions using either an Algebraic entry method or using Reverse Polish Notation (RPN). Both Algebraic and RPN entry methods utilize the expression stack in a similar manner. If you plan on using RPN, you should read this section to familiarize yourself with how the expression stack works, then <u>read about</u> <u>RPN</u>.

> Hint: Press the Up arrow on your handheld's navigation pad to see the standard keyboard

Expressions are entered into the active block. The active expression block is colored white. To activate a block, tap on the block near where the expression text is located. If the block is blank, tap anywhere in the block.

Once a block is active, enter and edit the expression using the keypad, arrow, and BS (backspace) keys. When the expression is complete, press the EXE(execute) key on the bottom right of the keypad, the RETURN key on the handheld's keyboard, or tap on another block. This enters the expression into the block and evaluates it. The result of the expression is shown in bold text beneath the expression. If the active block is the last block in the expression stack, a new block is automatically added and selected as the active block. Expressions are not saved until you press EXE or tap on another block. If you make a mistake, press the undo button, , and the previous value in the active block will be restored.



Math Tablet evaluates expressions based on common operator hierarchy. It performs multiplication and division first, then addition and subtraction. Math Tablet assumes that variables or values written together without an operator should be multiplied. If two or more variables are written together, Math Tablet will multiply them only if they are all single letter variables. See the examples below. These examples assume that the variables x,y,s,e,n and data are all defined.

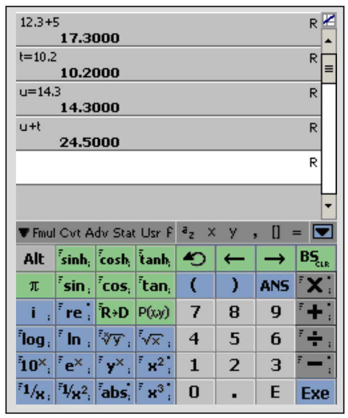
3x+5 is evaluated as (3*x) + 5
10sin(4pi + 5) is evaluated as 10*sin((4*pi)+5)
(10.2)(23.1) is evaluated as 10.2*23.1
1/3 + 2xy is evaluated as (1/3) + (2*x*y)
sen(10+pi) is evaluated as s*e*n*(10+pi)
12data is evaluated as 12*data
12sdata can not be evaluated because two variables must be multiplied and "data" is not a single letter variable

> Remember! You enter your expression using Algebraic entry as you would write it on paper. To evaluate sin(3) you tap "sin" then "3" to create the expression "sin(3)". To evaluate this expression tap EXE. This is opposite of many calculators in which you would enter "3" and then tap "sin".

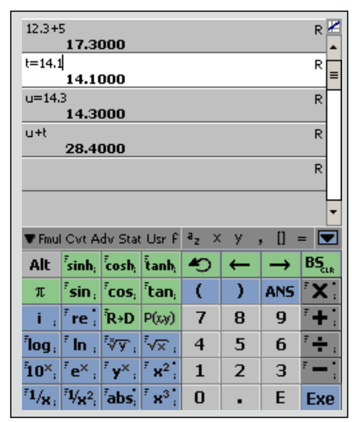
Expression Stack

The expression stack makes it easy to keep track of your work, to repeat steps in your calculations and to edit and correct your work. The expression stack is a list of expression blocks. The oldest blocks are at the top of the screen, the newer blocks are below them. Math Tablet remembers your last 100 expressions. When you edit an expression, Math Tablet automatically re-evaluates all of the expression blocks below the one you re-evaluated. Thus, you can use the expression stack to set up a problem and then vary the parameters of the problem and easily recompute the answers. Math Tablet also leaves the edited equation active so that you can easily change its value again and recompute the results by pressing EXE. Below is an example illustrating the expression stack.

The following expressions were entered into Math Tablet



Next the “t” expression was edited. “t” was changed from 10.2 to 14.1. As soon as EXE is tapped, all expressions below the “t” expression were re-evaluate. The updated stack is shown on the right.



Math Tablet also contains powerful scripting capabilities which let you control the order in which expression in a workspace are evaluated and lets you create your own functions. See the Scripting Module.

The expression stack features and short-cuts make it easy to enter, modify and view your expressions. These features are identified in the figure below.

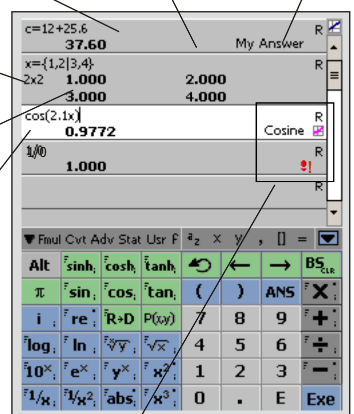
Tap and hold on an expression block to see the Expression Menu

Labels identify important blocks. See the Expression Menu

Tap on the equation to edit the equation and make that expression active

Matrix answers include the size of the matrix. A ".." below the size means that not all of the matrix could be shown.

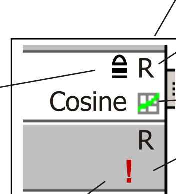
Tap on matrix answers to see the entire matrix in a matrix viewer



Tap in the area to the *left* of the answer to paste the ANS function corresponding to that expression into the currently active expression. If the expression was assigned to a variable, the variable will be pasted in the the active expression. Use this feature in combination with the Auto Assign feature to create calculations that build on each other. *See* the Options Menu.

Tap on the R or D to switch between Degrees and Radians

Indicates a locked expression. See the Expression Properties.



Tap here to enable/disable graphing of an expression. When you tap this spot, the plot box will appear

If an error occurs, the expression will be marked with a "!". Tap on the "!" to see an expanded error message

Plot this expression using the selected color

Don't plot this expression

Clear the plot selection on all expressions

Expression Menu & Properties

To view the Expression Menu, tap and hold the pen in an expression block.

Opens up the Expression Properties dialog (see below)

Executes the workspace as a script (see the Scripting Module)

Disables graphing of all expressions

Copies the answer from the selected expression to the clipboard. You can then paste the answer to other applications

Exports the answer to a comma separated (CSV) data file which can be opened in a text editor or in many spreadsheet applications

Use to copy and paste from one block to another. To copy text from an expression, first select the text then tap and hold in the block but away from the text. The menu will display "Copy Text". To copy and entire expression, tap and hold without selecting any text.

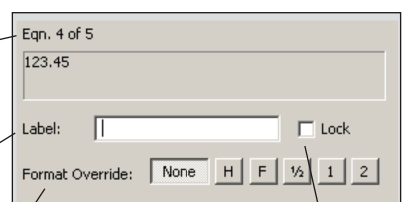
Converts the expression into a user function and opens up the User Function dialog

Adds a blank expression before the selected expression

Deletes the expressions from the stack

Hides all expressions without a label or Shows all expressions.

To view the Expression Properties, select Properties... from the Expression Menu



Math Tablet remembers your last 100 expressions. You can also save your work so you can edit or review it later.

Edit the expression's label. Use labels to identify key steps in your calculations. This makes it easier to modify your calculations when you come back to them later.

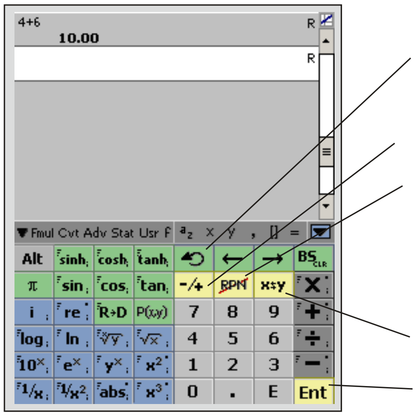
Format overrides. Use to display the answer in specific format. See Format Overrides

Locks an expression. Locked expression will not be automatically updated if the stack changes. You can update a locked expression by making it the active expression and pressing EXE.

RPN Entry

Math Tablet lets you enter expressions using either an Algebraic entry method, or Reverse Polish Notation (RPN). To switch to RPN mode, select RPN Mode from the Options menu. When in RPN mode, the keypad background changes from blue to green as shown below. RPN can only be used when you are entering values on the last equation block on the stack. If you make another block active, Math Tablet will temporarily return to Algebraic mode. You can tell what mode you are in by the color of the keypad.

The following section highlights the unique features of Math Tablet's RPN mode and assumes that you are already familiar with the RPN entry method. Remember, *RPN is optional*, if you are not comfortable using RPN you may want to skip this section.



Undoes the last RPN operation. Only one level of Undo is supported.

Sets the sign of a value

Temporarily disables RPN mode. RPN mode is automatically re-enabled if you press the EXE key, or activate another expression block and then reactive the last expression block in the stack

Exchange the last two entries on the stack

Enter or duplicate a value on the stack

RPN Keypad with green background

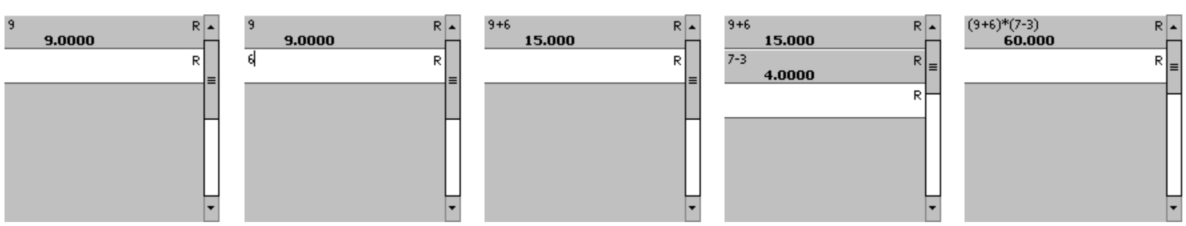
Entering Expressions:

Math Tablet uses an expression based RPN entry method. You enter an expression as you would using "normal" RPN, however, as you enter your expression Math Tablet generates the equivalent Algebraic expression and displays it in the stack. This lets you review and edit the steps you used in your computations.

As you enter expressions Math Tablet automatically grows the stack downward. For this reason RPN expressions must always be entered from the last expression on the stack.

The following example shows how the expression stack changes as the equation (9+6)*(7-3) is entered.

Stack contents:



Keys entered:

9 Ent **6** **+** **7 Ent 3 -** *****

Editing the stack contents:
Since Math Tablet uses an expression based RPN stack, you can edit the RPN stack contents at any time.

1) Tap on the incorrect expression to make it the active expression. Math Tablet will temporarily enter Algebraic mode.
2) Edit the expression. You can edit operators and values.
3) Press EXE to evaluate the new expression
4) Tap on the last expression block in the stack to make it active. Math Tablet will automatically return to RPN mode.

Undoing an operation:
Use the key to undo the last RPN operation and return the stack to its previous state. Only one level of undo is supported.

Entering Function Parameters:
Functions requiring only one parameter, such as "sin", are entered as done normally in RPN. sin(10.2) is entered as **10.2 sin**, note that **10.2 Ent sin** produces the same result, but is less efficient.

Functions requiring two parameters, such as "polar", (P(x,y) on the scientific keypad), can be entered two different ways. To enter: polar(5,6)

Method one: **5 Ent 6 polar** (polar is P(x,y) on the scientific keypad). **5 Ent 6 Ent polar** is the same, but is less efficient.

Method two: **5 Ent 6 , polar** This method uses the "," operator to create an expression 5,6 then evaluates that expression using the polar function

Functions requiring more than two parameters, or functions with optional parameters must be entered using the second method. That is, you must build up the parameter list using the comma operator and then evaluate the function. Alternately, you can disable RPN mode using the RPN key and enter the function in Algebraic mode. RPN mode will resume as soon as you press EXE.

Assigning and Recalling Variables:
To assign a value to a variable use the "=" operator. The "=" operator reverses the order of the entries so that you enter the value first, then the variable you wish to assign it to. For example to enter the expression a = 12+6, enter: **12 Ent 6 + a =**

To assign or recall an element from a matrix variable, use the [ ] as normal. When you press the [ ] key, Math Tablet temporarily enters Algebraic mode so that you can enter the parameter inside of the [ ]. Press Exe to enter the expression and resume RPN mode.

Using the ANS Function:
You can use the *ANS(n) function* to use previous stack values in an expression. See Expression Stack Features for information on recalling values from expression blocks. Math Tablet will automatically adjust the parameter "n" when it "pops" the stack. However, Math Tablet can not evaluate the ANS function if n<1, instead it will replace ANS with the actual value from the stack.

Using Matrices:
To enter a matrix, you need to enter the elements on the matrix first using the "," and " |" operators. Then enter "{}" to enter the values into a matrix.

To create the matrix $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$

You would enter **1 Ent 2 , 3 | 4 , { }** (the | and { } keys are located on the Advanced keypad)

An alternate method is to temporarily disable RPN mode by pressing the key. Then enter the matrix using Algebraic mode. When you press EXE, RPN mode will automatically resume.

Vectored Expressions:
To enable an expression for vectored operations press the key (in the Advanced module) after entering the expression. The *"V:" operator* will be placed at the beginning of the expression.

Entering Strings:
To enter a string value, press the " ", double quotes key. This temporarily disables RPN mode so that you can enter the string value. RPN mode is resumed when you press Exe.

Using the Complex Numbers:
You can enter complex constants two ways. For example, to enter 3+4i use:
1) **3 Ent 4 i +**
2) **3 Ent 4 Ent i * +**

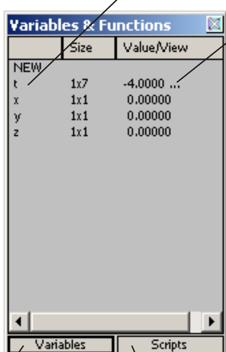
Changing Sign:
The key is used to change the sign of a value. The key functions differently depending on the state of the stack.
1) If the active block is empty, changes the sign of the previous expression block
2) If the active block holds a value that does not use scientific notation, changes the sign of the value
3) If the active block holds a value in scientific notation (E), changes the sign of the exponent. For example, to enter -1.2E-6 use: **1.2 -/+ E -/+ 6 Ent** where **-/+** is the key.

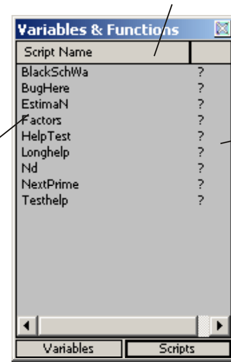
Stack Management:
Math Tablet's stack is slightly different than a traditional RPN calculator stack. First, the stack size is not fixed, but grows and shrinks as you make calculations. Second, you can edit or recall any value on the stack. Thus, Math Tablet does not need stack functions such as roll-up or roll-down. Below are some common calculator RPN stack operations and instructions on how to accomplish a similar function in Math Tablet.
1) Roll-Down: Tap and Hold on the expression you want at the bottom of the stack and select, Delete All Below from the pop-up menu.
2) Roll-Up: Instead of changing the stack, simply tap on the expression you wish to use. This will enter the ANS() function onto the stack and let you use the value stored in that location in the stack. This method also preserves all of your prior work.
3) Exchange X & Y: Use the key.
4) Duplicate X entry: Press **Ent** when the last block is active and empty

Recalling Variables and Scripts

You can enter the variables using the symbol bar shown below.

Tapping on the  symbol brings up the variable and function dialog box shown below. This box has two views. Tapping on the Variables or Scripts button at the bottom of the box switches between the two views. Variable are discussed on the following pages. Scripts are workspaces that can be run as a user program and are developed using the Scripting module.





The variable view list variables which have been assigned a value first, followed by user defined functions and then unassign variables. Scripts are listed in alphabetical order.

Creating Variables

Math Tablet lets you create up to 100 variables per workspace. Variable can have any name with the following restrictions.

"i" is reserved for sqrt(-1).
You can not use any existing built in function names
Variable names must can not contain numbers or symbols
Variable names are limited to 11 characters

A variable can hold any of the following type of data

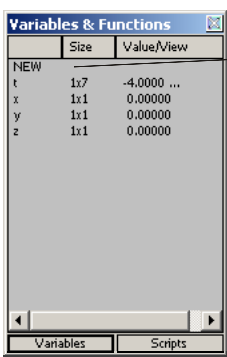
Scalar - a single complex value
Matrix - a matrix of complex values. The matrix can hold up to 64,000 values.
String - a string of characters, such as "Hello".

To create a variable:

1) equate a variable name to a value using "=". The expression, data = 13+7. will automatically create a variable named "data" if it doesn't already exist.

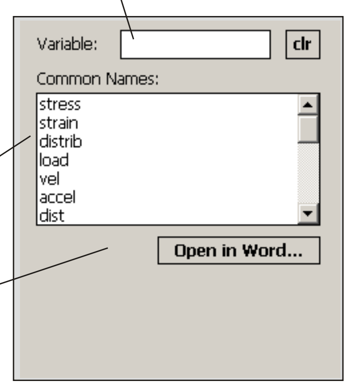
or

2) select NEW from the Variable and Function popup. This opens up the variable creation dialog shown below. Enter the name of the variable and press OK.



opens the variable creation dialog

enter the variable name here



a list of common variable names you might want to use

edit the list of common variable names to match your preferences

Lines beginning with non-characters are comment lines. You can add comments to your Common Names list to separate out different types of names.

To delete all unused variable select Delete Variables from the <u>Workspace sub menu</u>.

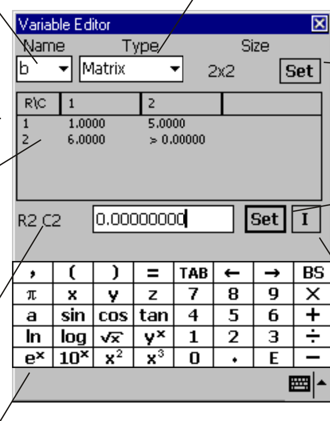
Press the Up arrow on your handheld's navigation pad to see the standard keyboard. Use this in combination with the first variable creation method above to name your variables.

Variable Editor

You assign values to variables using the “=” in an expression as discussed in the previous section, or by using the variable editor shown below. The variable editor is accessed through the Options Menu .

Select the variable you want to edit from this list

The variable type is listed here. Unused variables are shown as “empty”. You can change this value to change the variable type



Sets the number of rows and columns for a matrix variable

The value of the variable is shown here

To select a different matrix element, tap on the desired element. The selected element in a matrix will be highlighted

To change the value of a variable or matrix element, type in any valid scalar expression and press Set

Currently selected row and column

Quick keyboard for entering values. Use the TAB key to enter a value and move to the next element in an array. Press ⌨ to see the standard PocketPC keyboard

Use to import data from a text file into a variable. The text file must be formatted so the rows of the matrix are separated by a carriage return (return key) and columns of the matrix are separated by a comma, a tab or other nonnumeric character. A CSV (comma separated values) file is an example of a valid file. Import can only import real values.

Import is useful for transferring spread sheet data from a desktop PC to your handheld. Simply export the spread sheet data as a CSV file, transfer CSV file to your handheld and import the data into a variable.

Matrices

Math Tablet supports real and complex matrices.

Creating a Matrix:

You create a matrix by enclosing the row and column values in braces {}. Separate columns by commas and rows by a vertical line, |. (The **mat** command, available in older versions of Math Tablet, is also support.) Nested braces are not supported.

a = {1,2,3|4,5,6} creates the matrix $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$

Assigning Individual Values:

You can assign values to individual element, rows or columns, in a matrix variable by using the brackets [] and specifying the row, column. Use a ":" or a number less than 1 to assign entire rows or columns. Use "m:n" to assign a range of columns or rows, from m to n.

a[2,1]=9	assigns the element in row 2, column 1 to the value 9
a[1,:] = a[2,:]	assigns the values in row one of "a" to be the values in row 2 of "a"
a[2,2:3] = {9,10}	assigns the values 9 and 10 to columns 2 through 3 of row 2 of "a"
b = a	assigns the entire matrix stored in "a" to the variable "b"

The index values with in [] may be variables or expressions, but must be scalars.

Recalling Individual Values:

You can recall individual element, rows or columns, in a matrix variable by using the brackets [] and specifying the row and column. If you wish to recall an entire row or column replace the number with a ":" or a number less than 1. Use "m:n" to recall a range of columns or rows, from m to n

If a = {1,2,3|4,5,6} then

a[1,2]	recalls the value, 2, from the first row and second column
a[4]	recalls the fourth element, 4. Elements are numbered by rows
a[:,1:2]	recalls columns 1 through 2 of every row. The result is $\begin{bmatrix} 1 & 2 \\ 4 & 5 \end{bmatrix}$
a[:,1]	recalls the first column $\begin{bmatrix} 1 \\ 4 \end{bmatrix}$. Alternately you could write a[0,1]

You can recall elements from any matrix expression, not just variables. For example:

inv(a)[1,:]	recalls the first row from the matrix resulting from inv(a)
(a + 2a)[2,2]	recalls the element from the second row, second column of the matrix (a+ 2a)

When recalling matrix values from expressions, the [] has the same precedence as multiplication and division.

Strings

To save a string to a variable, place quotes around the text. For example a = "sin(x)". Matrix strings are not supported. You can also use the [ ] notation to recall or set individual characters, or groups of characters in a string.

Vectored Operations

Vectored operations let you perform operations on a matrix as if the matrix were a list of data. Vectored operations are all scalar operations which act independently on each element of a matrix. Vectored operation are useful for analyzing data sets, or for generating data for plotting.

In general, any scalar operator can act on a matrix by specifying that the expression block is to be evaluated using vectored operations. You specify vectored operations by preceding the expression with a "V:" (the key). Operations which support vectored use are marked on the keypad with a

Following are two examples comparing matrix and vectored operations

	Matrix	Vectored
Expression:	$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} = \begin{bmatrix}7 & 10\\6 & 22\end{bmatrix}$	$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix} = \begin{bmatrix}1 & 4\\9 & 16\end{bmatrix}$
Math Tablet Command:	{1,2|3,4}{1,2|3,4}	V: {1,2|3,4}{1,2|3,4}
Expression:	$\sin(\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix})$ result not defined in Math Tablet	$\sin(\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}) = \begin{bmatrix}0.8415 & 0.909\\0.1411 & -0.757\end{bmatrix}$
Math Tablet Command:		V: sin({1,2|3,4})

Math Tablet lets you mix matrices with scalars when doing vectored operations.

To see how vectored operations can be used to solve a problem, see Examples #7 and Example #8 at the end of this document.

Arbitrary Bases

Math Tablet lets you enter and display values in an arbitrary base.

Entering Values
To enter a value in an arbitrary base precede the value with 0bx where "b" is the base. If the base is greater than 10 use the letters A-Z for successive digits. For example:

110011 base 2 is entered as 02x110011

A231 base 12 is entered as 012xA231

FF23B base 16 is entered as 016xFF23B

Base 16 (hexadecimal) values can be entered without the base value. 016xFF23 and 0xFF23 are equivalent.

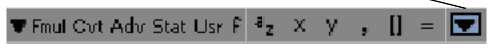
Values from different bases may be mixed in a single expression.

Displaying Values
Values are displayed in the currently specified format. Using the Base format setting you can specify the base in which all expressions, or specific expressions, are displayed. Math Tablet displays only unsigned integers when values are displayed in a base, other than base 10.

For details on changing the display format, see Format Overrides and Display Formats

Options Menu

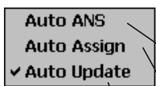
You access the Options Menu by tapping on the ▾ key



The items on the Options Menu are described below.

Use to change how Math Tablet displays results.
See Display Formats

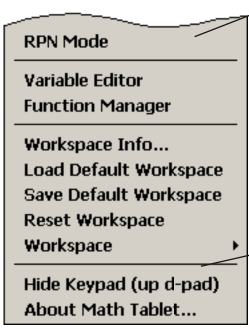
Tap on “Wksp”, “Set 1”, or “Set 2” to quickly switch to a different format. The currently used format is shown in “[ ]”



Automatically begins each express with ANS, which recalls the results of the previous expression. Use this mode to chain expressions.

Math Tablet will automatically start each expression with a variable assignment “a=“, where “a” will change with each new expression. The first expression will be “a=”, the next will be “b=” and so on up to “h=”. The assignments will then restart with “a=”. Tapping just below the variable assignment will recall that variable into the active expression.

Uncheck this to keep Math Tablet from automatically updating the expression stack. This is useful when developing scripts.



Enable/Disable RPN entry mode. See RPN Entry

Opens the variable editor or function editor dialogs

Use to manage your workspaces

Shows or hides the numeric and module keypads, and shows or hides the built in keyboard. Pressing the Up navigation button (cursor pad) on the front of your handheld will also show or hide the keypad

You can use CTRL Q from the keyboard to quit Math Tablet

Display Options

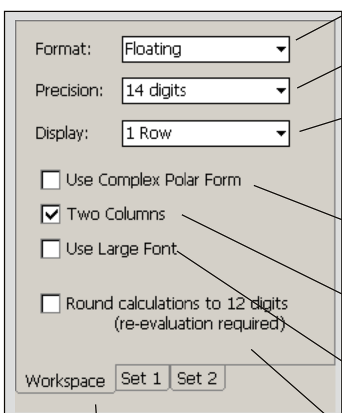
Math Tablet lets you view your work in several different formats. Use the Format... dialog on the Options menu to change this format.

Math Tablet lets you set up three different format configurations that you can easily switch between using the dialog box below or the Options Menu Custom format setting are useful, if for example, you perform a lot of complex math and regularly need to switch between rectangular and polar form. You can set up a format set for each format and quickly switch between the them using the Options Menu. There are three format sets.

Workspace: This format setting is associated with the current workspace and will be save and loaded as you save and load the workspace. Every workspace can have its own Workspace format.

Set 1 and *Set 2*: These format settings are perminent settings that do not change when you load workspaces. These are save when you quit MathTablet and loaded when you run Math Tablet.

The Format dialog is explained below.



select how the values are formatted

select the number of significant digits displayed

Sets the number of values that will show in an expression block when the answer is a matrix. If there are more values than can be shown a ".." is placed in the expression block. You can see all the values by tapping on the answer in the expression block. "Scripting" hides the expression block answer and is useful when writing scripts in Math Tablet.

displays complex numbers in polar form, otherwise rectangular form is used

Shows answers in a two column format when possible. Available only in "Floating" format

increase the font size used for the expression stack and some dialog boxes

internally round calculations to 12 digits. This can help to minimize zero values occuring a very small values. Once enabled, you will need to reevaluate expression on the stack.
Warning: Do not enable this option if you need maximum precision in your calculations.

Use to select bewteen different configurations you have set up. To use a setting, select that setting and press OK. You can also quickly switch between settings using the Options Menu

Format
Determines the layout or format to use when displaying numbers

Fixed Point displays values using the specified number of values to the right of the decimal. Very large, or very small values are displayed in scientific notation, using the specified number of digits.

Floating Point displays values using either scientific notation or fix point depending on the magnitude of the number. Floating Point automatically chooses the most efficient format.

Scientific Notation displays values always using scientific notation with one digit to the left of the decimal point

Engineering Notation displays values using scientific notation where the exponent is always a multiple of three

Fractions displays values as fractions. These may be approximations of the actual decimal value. The maximum value for the denominator is determined by the Precision setting.

Hexadecimal displays values as an unsigned 32 bit value in base 16. Decimal numbers are rounded to the nearest whole number.

Base displays values as an unsigned 32 bit value in an arbitrary base. Values are preceded with a 0x (base 16) or a 0bx, where b is the base of the value. The Precision setting is used to specify the desired base. Decimal numbers are rounded to the nearest whole number.

Precision
Determines the number of digits to show

Auto displays a variable number of digits. Values are displayed with up to eight significant digits. Trailing zeros are never shown.

2-14 displays a value with a fixed number of digits. Trailing zeros may be present. Math Tablet may automatically reduce the precision of complex values if they can not fix in the display area.

Fractional values (available only with the Fraction format) This value specifies the largest value of the denominator. Values are rounded to the nearest multiple of this value.

Truncate Small Zeros
Rounds values with a magnitude less than 10^{-14} to zero. The rounded value will be shown as +0 or -0 to indicate the rounding direction. This is useful if you are not concerned with very small values which may have resulted from internal roundoff. Available only under the “Floating” format.

Complex Values - Polar Form
You can view complex values using a polar format by selecting “Use Polar Form” . The polar form displays the magnitude and angle from the positive real axis, such as 7.0711@45°. The units for the angle corresponds to the angle setting for the expression block. Changing the angle setting will update the expression’s output. If you regularly use polar form, you should also refer to the “@” operator in the Scientific module, which lets you enter complex values directly in polar form.

When Polar Form is selected, complex values shown in the Variable editor or popup always use a unit of radians.

When Polar Form is not selected complex values are displayed in rectangular coordinates, such as 5+5i

> Math Tablet always retains answers internally to full machine precision (approximately 15 digits) even if the displayed value has been rounded.

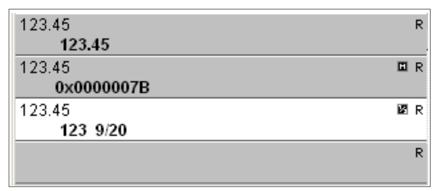
Format Overrides

Math Tablet nominally applies the format setting specified by the user to all expressions (see Display Options). Format overrides let you specify different formats for each expression. This is especially useful if you often work between base 10 and base 16. With a format override you can quickly switch between the two bases.

There are five format overrides. They are indicated by the following icons on the screen.

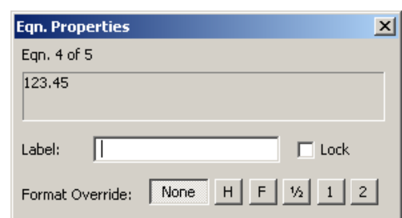
- **H** Hexadecimal format
- **F** Floating point format showing 16 digits
- **1/2** Fractional form format
- **1** Use the format specified in the format Set 1 (see Display Options).
- **2** Use the format specified in the format Set 2 (see Display Options).

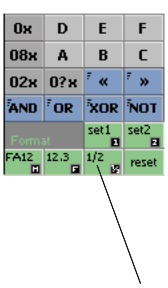
When an expression is using a format override, one of these icons appears on the far right of the expression as shown.



The format override can be set using the expression properties box, or through the updated Hex modules.

Using the expression property box , select one of the format override buttons to set the expression override.





The format override can also be set using the Hex module. The updated hex module includes buttons for setting the formats.

Use the hex module's format overrides to perform quick base conversions.

1) enter an expression in base 10
2) tap on the H (hex) format override

Format override buttons

Use Format Overrides to perform base conversion. Set one of the custom format sets to the desired base. The use format override to force the output to be displayed in that base.

Math Tablet lets you create up to 30 single line functions. More complex functions can be created using Math Tablet's scripting language. To create a single line function either enter the desired function on the expression stack and select "Make Function" from the Expression Menu or select the Function Manager from the Options Menu .

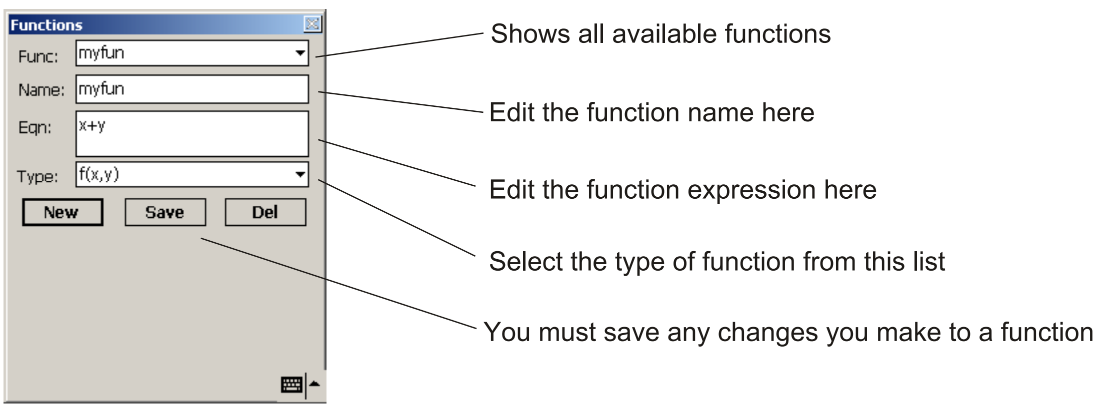
Functions must take one of four forms. Select the form, based on how many parameters your function requires:

Constant
f(x)
f(x,y)
f(x,y,z)

Before a function is evaluated, the values of x,y and z in the function expression are assigned the values passed to the function. The function is then evaluated. The resulting value is returned to the expression calling the function.

Function names are limited to 20 characters and can not contain any symbols or numbers.

The Function Manager is shown below.



User functions are saved as part of the current workspace. If you clear the workspace, or quit Math Tablet without saving the workspace you will loose your function definitions. If you save your user functions into the Startup workspace then the functions will be loaded whenever you Reset the workspace. See the Managing Workspaces section for more information.

You can create more complex functions using Math Tablet's scripting language. Script functions are saved independent of the current workspace.

In general, User Functions execute faster than scripts, however User Functions are limited to one line and are only available in workspaces in which they have been explicitly loaded.

Graphing

Math Tablet can graph up to 20 different expressions simultaneously. To enable graphing, tap on the area below the R or D on the right side of the expression box after you have evaluated the expression by pressing Exe. See the Expression Stack Features .

Math Tablet can graph expressions that are a function of “x”, explicit expression that are a function of both “x” and “y”, expressions in polar coordinates, and parametric equations. See the following table for details.

	Function Type	Enter this on the expression stack	Example: To plot this	Example: Enter this
rectangular	y=f(x)	f(x)	y=sin(x)	sin(x)
rectangular	f(x,y)=0	f(x,y)	$x^2+y^2=9$	$x^2+y^2 - 9$
polar	r=f(theta)	f(theta)	r=3*theta	3*t
polar	f(r, theta)=0	f(r, theta)	sin(8*theta)+3r=9	sin(8*t)+3r - 9
polar	where 0<theta<360 To change this range, change the polar rotation value in the graph setting.			
parametric	x=f(u), y=g(u) where 0<u<1	f(u) g(u)	x=sin(360u), y=cos(360u)	sin(360u) cos(360u)
parametric		Note: f(u) and g(u) must be entered on successive expressions on the stack. Only the f(u) expression should be marked for graphing.		

When you switch to the graphing view, Math Tablet will automatically graph the last 20 expressions that are marked for graphing with a ( ). See the Expression Stack Features .

The graph is automatically updated when ever you make a change to the expression stack.

Graphing

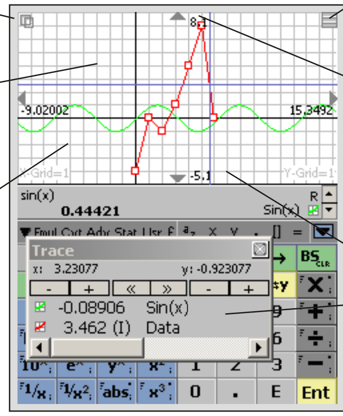
To view the graphs tap on the graph button (). Graphs can be viewed in half screen or full screen mode. Half screen mode lets you edit expressions while viewing the graph. Full screen provides many of the commonly used graph features as buttons and does not require a tap and hold to access them. To switch between the two modes press the zoom () button on the graph.

Tap to enable full screen graphing

Half Screen Mode

Tap to toggle between the graphing view and the expression stack view.

To zoom in, drag the pen to draw a box around the desired area



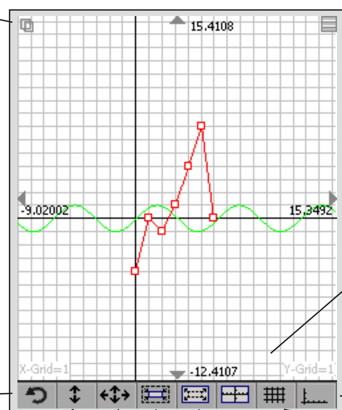
Tap on the arrows to scroll the graph. Math Table may have to recalculate the graph values when you scroll in the x direction. Y values are stored with the current graph so scrolling or scaling the y axis can be performed quickly.

Tap and hold the pen on the graph to see the Graphing Menu

Tap on the graph to bring up the "trace" box (see the following page) and show the traced location with a grey line.

Tap to enable half screen graphing

Full Screen Mode



Common graphing functions. You can also use a tap and hold on the graph to access these functions See the Graphing Menu

Previous Scaling

Settings

Grids

Scale Y only

Zoom in

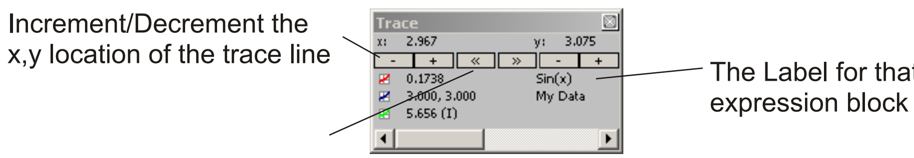
Default Axis

Scale all

Zoom out

Trace Box

To view the trace box, tap on the graph. This brings up the trace box and places cross hairs at the tapped location. The trace box lets you see the actual value of any data point on the graph.



The trace box displays the actual y value for each curve intersecting the *vertical* trace line. If a curve intersects the vertical trace line more than once, the value for curve value closest to the horizontal trace line is used.   When the polar grid is shown, trace value are displayed in polar coordinates using degrees

In addition the traced value may include a symbol to explain the how the value was obtained. Those are explained below:

(I) The curve was drawn by connecting a straight line between discrete data points (see the "plot" command in the Graphing Module). The value displayed was found by interpolating the y value at the current traced x value

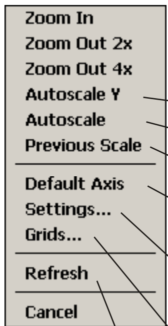
(--) The curve does not intersect the vertical trace line.

(*) The curved was generated with an implicit plot, such as y^2+x^2-9. Math Tablet had to solve an equation using a numerical approximation method in order to complete the requested plot. Values marked with a "*" are accurate to 1/10 the current zoom level.

(blank) The value displayed corresponds to the exact y data value for that curve. This y value corresponds to the x value shown at the top of the trace box. If a data point for the curve is close to the trace line, but not exactly under it, the trace box shows both the x and y value for that point. This can occur when data points are drawn using the plot or polar commands from the Graphing Module and the data point falls between two adjacent pixels on the display. Remember that the display has a limited resolution, so the trace line can not physical land on every possible data point at a given zoom level. Math Tablet takes care of the problem by giving you the exact x,y coordinate.

> You can copy a value from the trace window to the active expression by tapping on the value in the Trace Box when in half screen mode. Use this, in conjunction with the "plot" command and line option ("-"), to get interpolated values from your data.

To view the Graphing Menu, tap and hold the pen on the graph, or select one of the buttons at the bottom of the full screen graph.



Zooms the graph in and out. You can also draw a box around a desired area to zoom the graph in. Math Tablet may need to recompute graph values if you zoom in or out.

Sets the y scaling so that all of the graph can be seen.

Sets the x and y scaling so that all of the graph can be seen.

Restores the previous x and y scaling

Resets the axis limits to the default values and redraws the graphs.

Opens the Settings dialog shown below

Opens the <u>Grids dialog</u> described on the next page

Recomputes and redraws the graphs

When in full screen graphing mode, you can also select one of the following button from the bottom of the screen

Undo zoom

Autoscale y or x and y

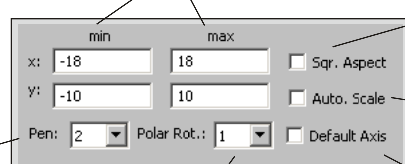
Zoom in and out

Use Default Axis

Change the grid and default axis settings

The default axis settings dialog is shown below

The graphing limits



Changes the thickness of the pen used to draw the graph. If you want the setting to be permanent, change the settings and <u>save the workspace as the "startup workspace"</u>

Makes the x and y axis scaling uniform. When the scaling is uniform, circles look like circles and not ovals

Automatically scales the graph so all the data is visible

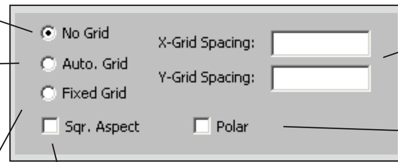
Changes the default limits. These are the values which are recalled when "Default Axis" is selected from the Graphing Menu

Sets the number of revolutions for which polar plots will be graphed. This set the range of "t" in polar graphs.

Use the grids dialog shown below to set the background grid for the graph.

No grid is shown

Math Tablet automatically chooses a grid size for the graph. The grid may change as you zoom in or out.



User specified grid spacing

Show a polar grid

The grid spacing is specified by the user. If the grid spacing is so tight that the handheld's screen can not resolve the grid, the grid will not be shown.

Makes the x and y axis scaling uniform. When the scaling is uniform, circles look like circles and not ovals

Math Tablet displays the current grid spacing values on bottom half of the graph.

Math Tablet includes a Graph Module and Scripting Module which lets you perform more advanced graphing.

Managing Workspaces

Math Tablet lets you organize your work in workspaces. A workspace contains all the expressions you have entered on the expression stack, your variables, your user defined functions and settings. Math Tablet has three different types of workspaces.

Start-Up Workspace

This is the workspace you see when you first run Math Tablet, or when you *"Reset Workspace"* the workspace, using the Options Menu . Use this workspace to store user defined functions, variable values you commonly use and preferred format settings.

Default Workspace

This workspace is saved automatically when you quit Math Tablet and automatically reloaded when you resume Math Tablet. The Default workspace can also be used as temporary storage. For example, if you wanted to perform some calculation on the side and then resume your original work, you could use the Save Default Workspace before you started your side calculation and the Load Default Workspace when you were done.

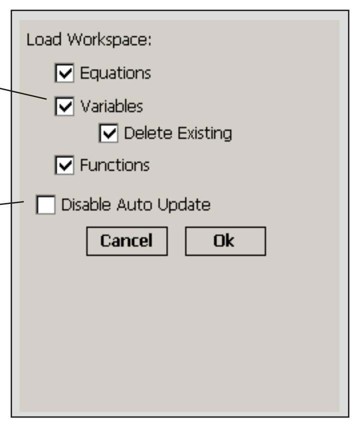
User Workspace

These are workspaces that you save and load at any time.

When you load a workspace you will automatically be prompted with the dialog box shown below. This allows you to load in all, or just part of the workspace. Math Tablet updates your current workspace with the new part you loaded in, retaining the other parts of your current workspace.

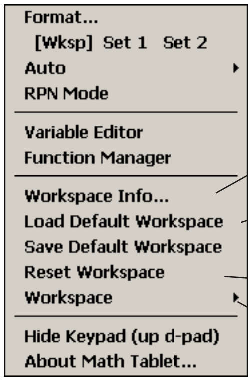
Select which components of the workspace you wish to load

Disables the automatic expression updating (Auto Update). Select this if you are loading a workspace script in order to edit it, and do not want Math Tablet to automatically update the expression stack after the workspace is loaded. You can enable Auto Update at any time using the Options Menu.



IMPORTANT!	To bypass the Start-Up and Default workspace, tap and hold the pen on the screen while Math Tablet is loading.

To manage your workspaces, use the *Options menu* and the Workspace submenu shown below.

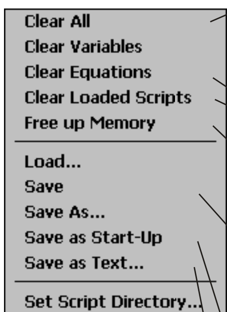


Show or edit comments for the workspace

Loads or saves the default workspace. This workspace is also saved when you quit Math Tablet.

Reloads the Start-Up workspace. If you have not saved a Start-Up workspace, this clears the current workspace.

Displays the workspace menu shown below



Clears all of the workspace except for your user defined functions. These you must delete using the Function Manager .

Deletes any unused variables and clears all other variables. The variables x,y,z,t are always defined

Clears Equations on the Expression stack, or Scripts which have been compiled by Math Tablet,

Releases temporary memory used in computations

Loads and saves workspaces. When you load a workspace you will be asked which components of the workspace you wish to load. You can also disable automatic updating of expression when you load a workspace. See Auto Update,

“Save” changes to “Save name” where name is the name of the last loaded or saved workspace. Use this after you made a change to a workspace.

Saves the current workspace as the Start-Up workspace.

Saves the current workspace as a text file which can be open in Word, or transferred to a desktop PC and printed

Sets the Script Directory. All script functions (workspaces) must be stored in the Script Directory You must have one workspace stored in a directory before you can set it as the Script Directory.

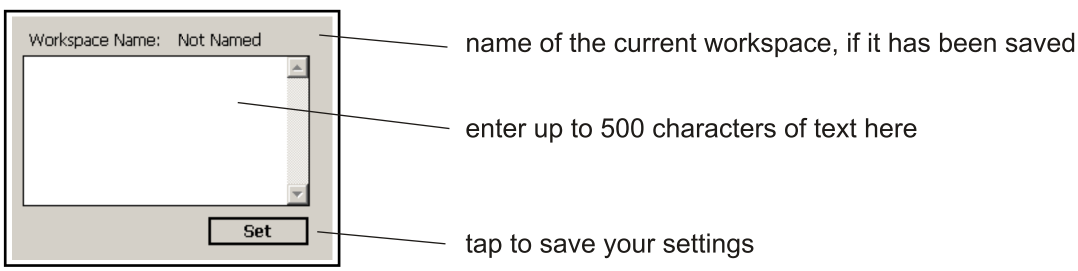
Documenting Your Work

Math Tablet provides several ways for you to document your work. This is important if you plan on reusing your workspaces or scripts at a later date.

Labels
You can add short Labels to any expression by choosing Properties from the Expression Popup menu. Labels are usefuls for indicating units or for documenting some simple aspect of the expression. Labels are limited to 30 characters and are also displayed on the trace when the expression is graphed.

Comment Lines
You can delegate an entire expression block for documentation by making the first character of the expression the "*" character. When this is done Math Tablet makes the block only one line high and indicates that the line is not active with a special error message. This method is similar to using comments in a program. Comment lines are limited to 100 characters.

Workspace Info.
You can provide more detailed documentation for each workspace using the Workspace Info. dialog, shown below, from the Options Menu. Workspace Info. is limited to 500 characters. This information is also displayed if the user accesses the popup help for a key which has been assigned to a user script or workspace, or if a user selects the "?" from the Variable and Function popup.



On Line Help
You can extend Math Tablet's on line help by creating your own htm help file and placing it in Math Tablet's program directory. You must then quit and restart Math Tablet before the online help will be updated. (Use CTRL Q to quit Math Tablet).

Modules and Module Menu

Modules are plug-in libraries that extent the capabilities of Math Tablet. Modules can provide new features and functions, or they can simply be a custom keyboard that simplifies access to features in other modules. Documentation for creating your own modules using C++ is available on request.

To install a module, copy the module's dll and help htm files to the Math Tablet directory and restart Math Tablet. You can have up to 16 modules installed at once.

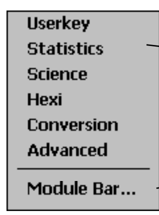
To switch between modules use the Options bar discussed below

You can list five modules here for quick access.

Show a list of all user scripts

Use the Module Menu to select any module, or change the modules shown on the bar. You can have up to 16 modules installed at once.

The Module Menu is accessed using the ▼ key



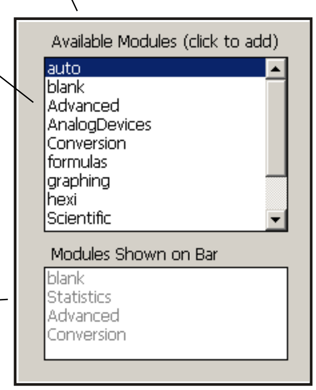
Lists all available modules. The keypad will change when you select a given module

Use this to change the four modules that are shown on the Options bar

Tap on an item to add it to the Options bar.

The "blank" item does not show any module.

The "auto" item creates spots that change to show your last used module. These items change when you select a module from the Module Menu that is not already showing on the module bar. This option is useful if you normally work between more than 4 modules.



Modules which will appear on the Options bar

Remember, you can create custom functions using Math Tablet's built in scripting language. For many applications, this is much easier than creating a C++ module.

The following pages describe the modules and feature that are included with Math Tablet. For each module the description lists:

1) the key you must press to access the function
2) the corresponding text that is entered in the expression block. This is the text you must enter if you use handwriting recognition
3) a description of how the function works
4) the module that must be installed in order to use this function. Some modules use features from other modules. “Built in” means that the feature does not require any modules.

The following function are always available on the right half of the keypad

Key	Function	Description	Module
+	+	add, strings are concatenated	Built in
−	–	subtract	Built in
X	*	multiply, if the operands are a scalar and a matrix, every element in the matrix is multiplied by the scalar.	Built in
÷	/	divide, you can not divide two matrices. If a matrix is divided by a scalar, every element in the matrix is divided by the scalar	Built in
ANS	ANS	recalls the results of the previous expression block on the stack. Use to chain calculations together.	Built in
	ANS(n)	recalls the results of the nth previous expression block on the stack. ANS(1) is the previous block, ANS(2) is two blocks up... n must be greater than zero. Use to chain calculations together. To automatically enter the ANS function, tap on a non-active expression block just to the left of the answer and below the actual expression. Always use the ANS(n) form when using RPN. Math Tablet will automatically adjust n as the stack changes	Built in

Sci Scientific Module

The Scientific module provides many common scientific and engineering functions. To access some functions you must press the ALT key first

Trigonometric

Key	Function	Description	Module
sinh	sinh(x)	hyperbolic sine	Sci
cosh	cosh(x)	hyperbolic cosine	Sci
tanh	tanh(x)	hyperbolic tangent	Sci
$sinh^{-1}$	asinh(x)	arc-sinh	Sci
$cosh^{-1}$	acosh(x)	arc-cosh	Sci
$tanh^{-1}$	atanh(x)	arc-tanh	Sci
sin	sin(x)	sine	Sci
cos	cos(x)	cosine	Sci
tan	tan(x)	tangent	Sci
sin^{-1}	asin(x)	arc-sine	Sci
cos^{-1}	acos(x)	arc-cosine	Sci
tan^{-1}	atan(x)	arc-tangent	Sci
	atanyx(y,x)	arc-tangent returns result in correct quadrant	Sci
R→D	deg(x)	converts x from radians to degrees	Sci
D→R	rad(x)	converts x from degrees to radians	Sci
P(x,y)	polar(x,y)	converts x,y to polar coordinates r,t (theta)	Sci
R(r,θ)	rect(r,t)	converts r,t (theta) to rectangular coordinates x,y	Sci
π	pi	(3.1415…)	Sci

Scientific

Key	Function	Description	Module
log	log(x)	log base 10	Sci
10^x	10^(x)	10^x	Sci
ln	ln(x)	log base *e*	Sci
e^x	exp(x)	e^x	Sci
y^x	y^x	y^x	Sci
$\sqrt[x]{y}$	nrt(x,y)	the xth root of y	Sci
1/x	1/(x)	1/x Inverse	Sci
$1/x^2$	$1/(x)^2$	$1/x^2$	Sci
x^2	x^2	x squared	Sci
x^3	x^3	x cubed	Sci
abs	abs(x)	absolute value of x, or magnitude of the complex value x	Sci
$\sqrt{x}$	sqrt(x)	square root of x	Sci

Complex

Key	Function	Description	Module
conj	conj(x)	complex conjugate	Built in
re	re(x)	real part of x	Built in
im	im(x)	imaginary part of x	Built in
i	i	sqrt(-1) the imaginary number	Built in
arg	arg(x)	the argument of x. This is zero except for complex numbers.	Sci
$ye^{\theta i}$	y@x	enters a value in complex polar form where y is the magnitude and x is the angle. This uses the formula y(cos(x)+sin(x)i). x and y must be real.	Sci

The Advanced module provides matrix and advanced features, such as equation solvers.

Matrix

Key	Function	Description	Module
{ }	{}	use to creates a matrix. Separate columns with a comma and rows with a |.	Built in
|	|	use to separate rows when entering a matrix	Built in
T	T(x)	matrix transpose	Built in
det	det(x)	matrix determinate	Built in
inv	inv(x)	matrix inverse	Built in
I	I(x)	creates an x by x identity matrix	Built in
zero	zero(r,c)	creates an r by c matrix of all zeros	Built in
rref	rref(x)	computes the row reduced echelon form of x. x must have more columns than rows.	Built in
eig	eig(x,“var_{opt}”)	computes the eigenvalues of x. The eigenvectors are stored in the optional variable, var. Example: eig(x,“v”) finds the eigenvalues of x and stores the eigenvectors in the variable v	Adv
[A|B]	augc(a,b,....)	combines matrices by augmenting columns	Adv
[A/B]	augr(a,b,...)	combines matrices by augmenting rows	Adv
	aug(a,b,...)	combines matrices into a single column vector	Adv
fill	fill(a,b,c)	creates a matrix with one column. The row values start at a, end at c and are incremented by b.	Adv
ones	ones(r,c)	creates an r by c matrix of all ones	Adv
	V:	forces the expression to use vectored operations.	Built in

Advanced

Key	Function	Description	Module
rk	rk(func(x,y),x_0)	solves the nonlinear differential equation of the form dy/dx = func(x,y) with y(0) = x_0 func(x,y) must be supplied as a text value — either a literal or a string variable. The solution will be plotted when the expression block is plotted Example: rk(“x+y”,3) solves the equation dy/dx = x+y with y(0) = 3	Adv
roots	roots(a_n, a_{n-1}, … a_0) or roots(m)	solves for the roots of the polynomial of the form $a_n y^n + a_{n-1} y^{n-1} + \ldots + a_1 y + a_0 = 0$ Maximum n = 18. If “m” is a matrix, then m = {a_n, a_{n-1}, … a_0}	Adv
d/dx	ddt(func(x),$dx_{optional}$)	differentiate func(x) at the current value of x. dx is an optional parameter specifying the increment used in differentiation. The default value for dx = 0.01, func(x) is supplied as text. Example: ddt(“sin(x)”)	Adv

Advanced

Key	Function	Description	Module
∫dx	integ(func(x),a,b,Opt)	integrate func(x) between the limits *a* and *b* using Romberg algorithm. func(x) must be supplied as a text value — Example: integ("sin(x)",1,2) Opt is optional parameter" > 1, use Simpsons method with Opt partitions < 1, specifies tolerance for Romberg method "x" use Simpsons when graphed for fast graphing	Adv
solve	solve(func(var),a,b,$var_{optional}$)	solves for a solution to the nonlinear eqn. func(var) = 0 where a< var < b var is an optional parameter which specifies which variable to solve for. The default value for var is "x". var must be supplied as a text value — either a literal or a string variable — ie "y". *See example #3*	Adv
eval	eval(txt)	evaluates the expression stored as text txt must be supplied as a text value eval must re-parse txt each time it is evaluated. Graphing expressions which contain eval can take up to twice as long as graphing the expression directly.	Built in
lti	lti(a_n, a_{n-1}, … a_0, f(x),IC_{opt}) or lti(m,f(x),IC_{opt})	solves the linear time invariant differential equation of the form: $a_n y^{(n)} + a_{n-1} y^{(n-1)} + \ldots + a_1 y^{(1)} + a_0 y = f(x)$ $y^{(n)} = d^n y/dx^n$ Maximum n = 9 f(x) must be supplied as a text value — either a literal or a string variable, ie "sin(x)". If "m" is a matrix, the a_n coefficients are $m = \{a_n,\ a_{n-1}, a_{n-2}, \ldots a_0\}$ The solution will be plotted when the expression block is plotted. lti uses a Runge-Kutta method with a fixed time step based pn the x axis range. Zoom in on the graph to improve the accuracy. IC_{opt} is an optional matrix holding the intial conditions, from the highest to lowest derivative. Initial conditions are set to zero if IC_{opt} is not provided Example: lti(1,2,1,"sin(x)") solves $d^2y/dx^2 + 2dy/dx + y = \sin(x)$	Adv
size	size(a)	returns the number of rows and cols in "a" as a matrix {rows, cols}	Adv

Hex Hexadecimal Module

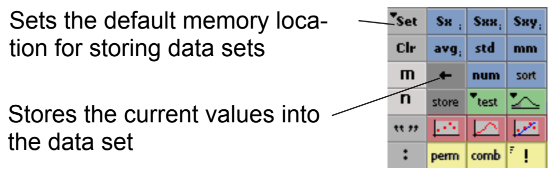
The Hexadecimal module provides bitwise functions and makes it easier to use Math Tablet's built in hexadecimal features. Hexadecimal operations assume an unsigned 32 bit integer.

Hexadecimal

Key	Function	Description	Module
0x	Ox	use to enter hexadecimal values. OxFFA2 enters the hexadecimal value FFA2	Built in
AND	x _AND_ y	bitwise x AND y	Hex
OR	x _OR_ y	bitwise x OR y	Hex
XOR	x _XOR_ y	bitwise x XOR y (exclusive OR)	Hex
NOT	_NOT_ x	bitwise NOT x	Hex
»	x >> y	shift right - bit shift x right y bits	Hex
«	x << y	shift left - bit shift x left y bits	Hex

The Statistics module provides functions for entering and analyzing data sets. Data sets are stored in a n x 2 matrix variable, where n is the number of data points. The first column of the matrix contains the independent value of the data and the second column contains the dependent value. If the data set is one dimensional, then the first row can simply contain an integer value representing the current observation.

To make it easier to enter data, the statistics module includes two special keys: the Set and the Store keys.



The statistical keypad

The *Set* key lets you select a set of matrix variables to use to store your data. When a variable is selected it is shown in the key below the Clr variable key. The matching variable is shown in the key below . The statistics modules matches the following variables as pairs: m n, o p, and q,r. You are not required to use these variables. They are provided only for convenience.

The *Store* key can be used to quickly load data into a variable.
The *Store* key does the following:
1) enters the “store” function on to the expression stack
2) automatically enters the name of the currently selected variable
3) sets the row value to the next free row in the matrix variable
4) executes the expression. This loads the data values into the matrix.

For example to enter the following data into the “m” variable:

x	3	5	9	11
y	2	3	21	13

Enter the following:

1) Tap Set and pick “m” as the data variable
2) Clear the contents of “m” by entering clrv(“m”)
3) Enter “3,2" with out the quotes and press the key
   The expression store(1,”m“,3,2) will be entered on the stack and executed
4) Enter “5,3" with out the quotes and press the key
5) Continue entering the rest of the data

You can verify the contents of “m” by entering “m” on the expression stack and executing that expression. The partial contents of “m” will be displayed. Tapping on the displayed value of “m” will open the variable window so you see all the values of “m”.

If the data set contains only one set of values, the y values for example, then you would enter the data as discussed above, but only enter a single value and press . The store function will automatically assigns an x value to the data and stores your entered value as the y value. See the following statistics function documentation.

Some statistical functions can operate directly on the data stored in a variable without having to “load” the data set. The reference below refers to a variable data set as var, where var is the name of a variable enclosed in quotes, such as “a”. Operation performed directly on variables require less memory. avg(“m”) computes the average of values in the variable “m” directly. avg(m) loads the values in “m” onto the evaluation stack and then computes the average of those values.

Statistical

Key	Function	Description	Module
avg	avg(val) or avg(x)	Computes the average value for each column in the matrix, or row if the matrix has only one row	Stat
std	std(val) or std(x)	Standard deviation of each column in the matrix, or row if the matrix has only one row	Stat
Sx	sx(val) or sx(x)	Sum of the rows in each column in the matrix	Stat
Sxx	sxx(val) or sxx(x)	Sum of the square of each value in a column, or row if the matrix has only one row	Stat
Sxy	sxy(val) or sxy(x)	Sum found by multiplying the elements in a row together and then summing the result for each row.	Stat
num	num(val) or num(x)	Number of rows in val	Stat
	plot(var)	Scatter plot of the values stored in var. The x values are stored in the first column, the y values in the second. *See Graphing Module*.	Graph
	plot(var,“-”)	Line plot of the values stored in var. The x values are stored in the first column, the y values in the second. *See Graphing Module*	Graph
test	mtest(var,x)	Tests the hypothesis H_0: mean(var)= x H_1: mean(var)- x var = the variable holding the data set. The second column of the data set is used. x = the test mean value The t statistic is t = (mean(var) — x)* sqrt(N)/std(var) where mean(var) = mean of the data set stored in var std(var) = standard dev. of the data stored in var N = number of items in the sample mtest return a 1x2 matrix of the form: 1. the t parameter 2. the p value for the 2 sided test	Stat

Statistical

Key	Function	Description	Module
test	mdtest(var1,var2) mdtest(var1)	Tests the hypothesis H_0: mean(var1)= mean(var2) H_1: mean(var1)- mean(var2) var = the variable holding the data set. The second column of the data set is used. x = the test mean valueStudent t-test for equal means on data sets with equal variances. If two var locations are provided, mdtest compares the second columns of each var. If a single var is provided, mdtest compares the first and second columns. The t statistic is t = (MeanVar1 — MeanVar2)/(s*sqrt(1/N1+1/N2)) and s = sqrt(((N1-1)*s1*s1 + (N2-1)*s2*s2)/(N1+N2-2)) where MeanVar1 = mean of the data set stored in var1 MeanVar2 = mean of the data set stored in var2 N1 = number of values in var1 N2 = number of values in var2 s1 = std. dev of values in var1 s2 = std. dev of values in var2 mdtest return a 1x2 matrix of the form: 1. the t parameter 2. the corresponding two sided p value	Stat

Statistical

Key	Function	Description	Module
test	pairtest(var1,var2) pairtest(var1)	Tests the hypothesis H_0: mean(var1-var2)= 0 H_1: mean(var1-var2)- 0 using a paired t test. var = the variable holding the data set. If two var locations are provided, mdtest compares the second columns of each var. If a single var is provided, mdtest compares the first and second columns. The t statistic is t = (mean(var1-var2))* sqrt(N)/std(var) where mean(var) = mean of the data set stored in var std(var) = pooled standard dev. of the data stored in var N = number of items in the sample mtest return a 1x2 matrix of the form: 1. the t parameter 2. the p value for the 2 sided test	Stat

Statistical

Key	Function	Description	Module
test	sdtest(var,var) sdtest(var)	Tests the hypothesis H_0: Var(var1)/Var(var2)= 1 H_1: Var(var1)/Var(var2))- 1 If two var locations are provided, sdtest compares the second columns of each var. If a single var is provided, sdtest compares the first and second columns. The f statistic is f = (s1*s1)/(s2*s2) where s1 = std. dev of values in var1 s2 = std. dev of values in var2 Values are automatically ordered such that s1<s2 sdtest return a 1x2 matrix of the form: 1. the f parameter 2. the p value corresponding two sided probability from the f-distribution	Stat
	lsq(var) or lsq(x)	least square curve fit of the values in var. The x values are stored in the first column, the y values in the second. lreg return a 1x3 matrix of the form: 1. a 2. b, where y=ax+b is the best fit line 3. r, the regression coefficient If lreg is plotted, it plots the line defined by y=ax+b	Stat
!	x!	factorial of x. Noninteger values of x are truncated before the factorial is computed.	Stat
mm	mm(var) or mm(x)	Finds the Minimum, Maximum values for each column in var, or row if the matrix has only one row. mm return the min, and max values in a matrix	Stat
sort	sort(var,n) or sort(x,n)	Sorts the columns of the matrix stored in var according to the values in the n th column, or by row if the matirx has only one row	Stat
perm	perm(n,m)	Permutation of n things taken m at a time	Stat
comb	comb(n,m)	Combination of n things taken m at a time	Stat

Statistical

Key	Function	Description	Module
	norm(x,m,s)	Computes the one sided probability in the left tail of a normal distribution. m is the distribution mean, s is its standard deviation. The default value is 0 and 1.	Stat
	invnorm(p,m,s)	Computes the value corresponding to the one sided probability in the left tail of a normal distribution. p is the probability,m is the mean, s is its standard deviation. The default value is 0 and 1.	Stat
	tdist(x,v)	Computes the one sided, left tail, probability of the t-distribution at x with v degrees of freedom.	Stat
	invtdist(p,v)	Computes the value corresponding to the one sided, left tail, probability of the t-distribution. p is the probability with v degrees of freedom.	Stat
	fdist(x,v,w)	Computes the one sided probability of the f-distribution at x with v,w degrees of freedom.	Stat
	invfdist(p,v,w)	Computes the value corresponding to the one sided, left tail, probability of the f-distribution. p is the probability with v,w degrees of freedom.	Stat
	chi(x,v)	Computes the one sided, left tail, probability of the chi-squared distribution at x with v degrees of freedom.	Stat
	invchi(p,v)	Computes the value corresponding to the one side, left tail, probability of the chi-squared distribution. p is the probability at x with v degrees of freedom.	Stat
	binom(x,np)	Computes the probability that y<= x when sampled from n event with individual probabilities of p. The corresponds to the one sided, left tail, cummulative probability for the binomial distribution. distribution for x events taken degrees of freedom.	Stat
	invbinom(P,np)	Computes the x such that the probability y<= x when sampled from n event with individual probabilities of p, is P. The corresponds to the one sided, left tail, cummulative probability for the binomial distribution.	Stat
store	store(n,var,x,y) store(n,var,y)	<u>Stores the value (x,y) in variable var in row n</u>. If x is not provided, x is assumed to be the same as the row number, n.	Stat
Clr	clrv(var)	clears the contents of var and reset the matrix size to zero rows	Stat

Conversion Module

The Conversion module makes it easy to work with different systems of units. This module has three features: the default base units, integrated unit conversion, and user defined conversions and constants.

Default Base Units
There are two base units: meter-kg-sec or ft-lbm-sec The base units are set using the MKS or FPS button. Once set, the default units are indicated with a diamond on the buttons. When ever you apply units to a value, that value is automatically converted to the default unit.

The base FPS units are:

ft - feet
lbm - pounds (mass)
(not slugs)
lbf - pounds (force)
s - seconds

The base SI units are:

m - meters
kg - kilograms (mass)
N - newtons (force)
l - liters

If, for example, you have the default units set to meters (SI units) and you enter

4FT + 6IN + 12CM

the answer will be given in meters as 1.4916

If you change the base units to feet and re-evaluate that expression block, the answer will be in feet as 4.8937

If, you have the default units set to meters (SI units) and you enter 2GAL the answer will be given in meters^3 as 0.0004 since meters are the default units, and meter^3 is the unit of volume. *When in doubt, you can specify the units on both sides of the > as discussed below*.

Math Table does not check to see that you are using consistent units. You can for example enter 6FT+3GAL and get a number which is meaningless. The conversion module keypad is color coded. Compatible units are grouped together with the same background color.

Integrated Unit Conversion
You can convert between compatible units using the > key as follows.
Current Units > New Units The value is converted from the old units to the new units.

For example: 6FT>IN returns the answer 72, which is the length in inches.

The > works regardless of the current base units. However, if you are converting from the base units you do not need to specify the units on the left side of the > operator.

For example if you have the base units set to feet, but you want the answer in inches you would enter (1.5)>IN

which returns 18. That is 1.5 ft = 18 in

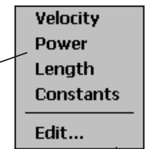
If you change the base units to meters and reevaluate the expression it will convert 1.5 meters to inches.

You can also convert between compound units. For example
(144IN^2)>(FT^2) returns 1
12(FT/SEC)>(MI/HR) returns 8.1818

User Defined Constants and Conversions
In addition to the built in unit conversions, you can create your own conversion constants. These constants are simply values that convert between one unit and another. They are not integrated into Math Tablet like the previously discussed conversions. However, you can organize and add new conversion as you need them.

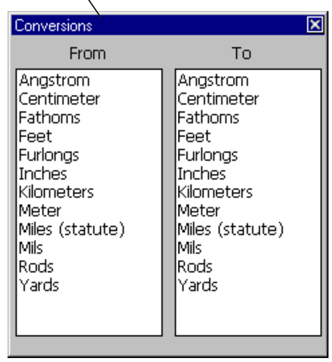
The constants and conversions are accessed using the Constant button Const

Pressing the Constant buttons brings up the following menu:



Use to select from a list of available conversion constants categories. Selecting a category brings up the selection box shown below

Select the units you want to convert to and from. The conversion constant will be entered on the expression stack as soon as you make two selections.



Use the Edit menu option to edit conversion values, add new values or create new categories. Selecting Edit opens up the <u>Units editor.</u>

Choosing an Appropriate Conversion Method

You can perform unit conversions in Math Tablet using either the built in units conversion feature or using the User Defined Constants. In many cases you can convert between units using either method. In general you should use the built in conversions if possible because it makes it easier to track your work and identify errors. For example, it is much easier to figure out what you did if you enter

145.4(FT/SEC)>(MI/HR)

than if you enter

145.4(0.681818)

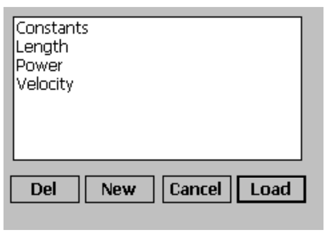
Units Editor

The Units Editor is a separate application that lets you create and edit your own conversion constants. Conversion constants are grouped in to sets of compatible units, i.e. length, area, velocity.... Each set can hold up to 200 units, and you can have as many sets as you need. Within each set there is a “base” unit. This can be any convenient unit. To add units to a set, you provide the conversion factor from the base unit to the new unit. You need provide only this one conversion factor. Math Tablet automatically derives conversion factors for all the other units defined in that set.

The Units Editor stores conversion constants in two forms: unit conversions and constants. Unit conversions let you convert between units. Constants are single unrelated values, like the “the speed of light”. Constants use a special base unit called “Constant”.

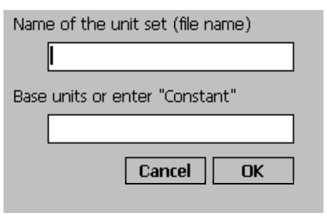
Unit sets are stored as separate files which must be placed in the same directory as the Math Tablet application. The Units Editor handles this automatically. You can also add unit sets directly by copying existing unit set files to your Math Tablet application directory.

The Units Editor is described below.



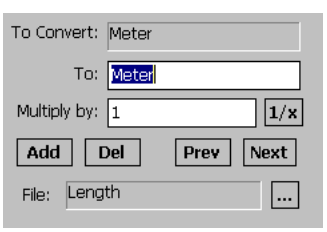
The start up screen lets you select which unit set you wish to edit. You can also create new unit sets or delete existing sets.

To edit an existing unit set select “Load”



When you create a new unit set you must provide the name of the unit set and the “base” unit for that set. The unit set name corresponds to the file name for that unit set. The base unit is the common unit for which you will have to provide conversion factors for all other unit. Once a base unit is selected it can not be changed.

If you want to create a list of constants, enter “Constant” for the base unit.



Once you’ve created a unit set you can add, delete or edit the units in that set. When you add a unit you must provide the name of the unit and the conversion factor to as many significant digits a possible. 10 digit are recommended.

The 1/x button inverts the current value in the Multiply by box. Use this if you know the conversion factor to the base unit instead of from the base unit.

The ... button brings up the start up screen and lets you switch to a different unit set.

The Conversion module provides many common scientific and engineering conversions. To access some functions you must press the ALT key first. Conversion factor can be combined to create compound conversions, such as FT/SEC

Conversion Functions

Key	Function	Description	Module
>	x > y	Converts from one unit to another. Starting Units > Desired Units. Note: a>b = b/a	Cvt
Units	Units	Change the base unit between metric and US.	Cvt
°C→°F	CF(x)	Converts the value x from degrees Celsius to degree Fahrenheit	Cvt
°F→°C	FC(x)	Converts the value x from degrees Fahrenheit to degrees Celsius	Cvt
x^2	x^2	Squared - use to create compound units	Adv
x^3	x^3	Cubed - use to create compound units	Adv

Conversion Units

Key	Abreviation	Unit	Module
m	M	meter	Cvt
mm	MM	millimeter	Cvt
cm	CM	centimeter	Cvt
km	KM	kilometer	Cvt
ft	FT	feet	Cvt
in	IN	inch	Cvt
yd	YD	yard	Cvt
mi	MI	mile	Cvt
kg	KG	kilogram	Cvt
g	G	gram	Cvt
lbm	LBM	pound mass	Cvt
oz	OZ	ounce mass	Cvt
l	L	liter	Cvt
cc	CC	cubic centimeter	Cvt
gal	GAL	gallon (liquid)	Cvt
cin	CIN	cubic inches	Cvt
pint	PT	pint (liquid)	Cvt
qt	QT	quart (liquid)	Cvt
btu	BTU	B.T.U. (th)	Cvt
joule	J	joule	Cvt
ft-lb	FTLB	foot-pound	Cvt
kwh	KWH	kilowatt-hour	Cvt
cal	CAL	calorie	Cvt
psi	PSI	pounds per square inch	Cvt
pascal	PA	pascal	Cvt
atm	ATM	atmosphere	Cvt
N	N	newton	Cvt
lbf	LBF	pound force	Cvt
ozf	OZF	ounce force	Cvt
dyne	DYNE	dyne	Cvt

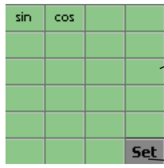
Time and Date Conversions

Key	Function	Description	Module
sec	SEC	seconds	Cvt
min	MIN	minutes	Cvt
hr	HR	hours	Cvt
day	DAY	days	Cvt
mdy	MDY(x)	returns a text string indicating the month, day, year and time corresponding the the value x. Where x is the number of seconds since Jan 1, 1601	Cvt
sdate	SDATE(m,d,y)	returns the number of seconds since Jan 1, 1601 where m= month, Jan = 1, Feb = 2, ... d= day y= year You can use the time conversion to perform date calculations. For example, to determine the time and date which is 36 days, 23 hours and 5 minutes from July 4, 1964 at 1:42 PM a=SDATE(7,4,1964)+13HR+42MIN *note 13hr + 42min = 1:42pm* MDY(a+36DAY+23HR+5MIN) *this offsets the time “a” by 36days, 23hrs 5min* *and returns the corresponding day* “12:47:00 on Mon Aug 10, 1964" You can compute the number of days between Jan 4, 2003 and July 5, 2004 as follows: SDATE(7,5,2003)-SDATE(1,4,2003) ANS>DAY *converts seconds to days* 548	Cvt
	NOW	returns the number of seconds since Jan 1, 1601	Cvt

User Module

The User module lets you create a custom keyboard layout. This makes it easy to combine features you commonly use from different modules into one custom module.

The User module has 23 blank keys and a Set key.



Blank keys the user can assign

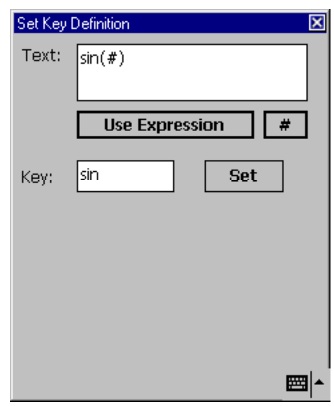
Use to assign keys. Tap twice to save your settings. Settings are automatically saved when you quit Math Tablet

To assign a function to a blank key:

1) Tap on the Set key.

2) Tap on the key you wish to change. This brings up the key definition box shown below.

3) Enter the text the key should produce when you tap it. The key's text can be any text you wish up to 60 characters. It can be a simple constant or a combination of functions.   Tapping on the "Use Expression" button, copies the text from the currently active expression. Thus, you can build your text in the expression stack and then copy it to a user key. Finally, place a # where you wish the cursor to be placed when you tap the user key. The # symbol will be removed from the text when the user key is pressed.



4) Enter the label for the key. The label is limited to seven characters or the size of the key.

5) Tap Set to close the key definition box

A useful key assignment is the open/close parenthesis.

Text: (#)
Key: (...)

When pressed this key enters a pair of parenthesis and positions the cursor between them

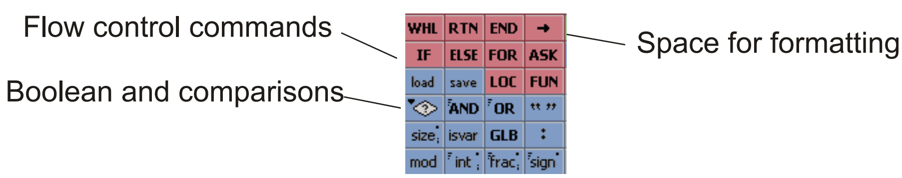
A useful "hack":
You can create additional user key modules by duplicating the Userkey.dll module file and giving it a different name - you must keep the dll extension. The new module will act just like the original Userkey module, but you will be able to assign a different set of keys to the keyboard. This would give you two user definable keyboards. The disadvantage is that both modules will have the same icon when shown on the module keypad.

Scpt Scripting Module

The Scripting module lets you write scripts (short programs) which can be run directly in Math Tablet. This makes it easy to add advanced features to Math Tablet with out having to write a full module using C++.

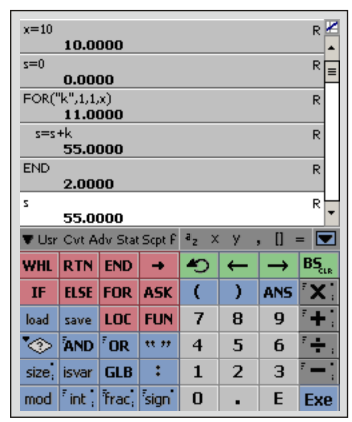
A Math Tablet scripts is a series of expressions which are executed in a particular order. The script can contain most valid Math Tablet expression. In addition to the standard Math Tablet expressions, the script modules provides special functions for controlling the flow of the script. These include, IF-THEN-ELSE, FOR and WHILE constructs. Scripts can contain local variables and can be called recursively - with in the memory limits of Math Tablet.

A script is written by creating a workspace which implements your function. The script module provides access to commands for controlling the flow of the script. The scripting keypad is shown below. The functions are described on the following pages. The paragraphs below highlight a few important feature of Math Tablet's scripting capabilities.



A simple script

A script is created by "writing a program" in a Math Tablet workspace. Each expression in the workspace corresponds to one line in the program. The following simple script computes the sum of the first "x" whole numbers.



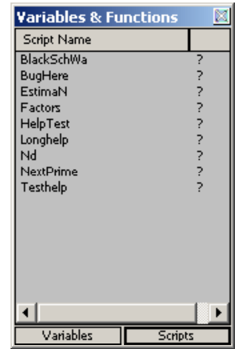
Running a script with in its workspace

You can run a script directly with in the workspace where it was created by tapping and holding the pen on a desired expression and selecting "Run from here.." from the expression menu. The script is run until it encounters a RETURN or end of the workspace. When running a script in this manner only expressions which are executed by the script are updated. This is different from the normal updating of the expression stack where all expression below the active expression are executed.

This method is useful for running simple scripts and for testing your scripts. The disadvantage is that this method takes more memory than running the script as a function (see the next page) and does not support local variables or parameter passing.

Running a Script as a Function (A Script Function)

You can run a script from a different workspace by using the name of the script (the workspace's file name) as a function in an expression block. The value returned from the script is the value in the RETURN function of the script. To minimize memory use, Math Tablet does not load the variables and user functions for the script's workspace. Memory is further conserved because Math Tablet does not store the result of each expression block. Thus you can not use the ANS command in a Script Function.



List of Scripts

To type a Script Function into the active expression, select the *Variable and Functions popup* and then tap on the Script button at the bottom of the window.

IMPORTANT *Restrictions for Running a Script as a Function:*

1) The script's name must begin with a capital letter.
2) The script's name must be less than 20 characters long and can not contain any numbers or symbols.
3) The script's workspace file must be located in the Script Directory.
5) Function scripts do not support the ANS function.
6) *User functions* must be defined in the workspace running the function script. User functions are not saved with the script

Local Variables: Script Functions automatically create the local variables, x, y, z, a, b, c. You can create additional local variables using the LOCAL command. Any variable which is not explicitly declared as local is a global variable. Math Tablet does *not* follow local variable scoping rules used in C++. Local variable created in one function are *not* available in functions called with in that funcion.

Parameter Passing: You can pass up to six parameter to a Script Function. These parameters are automatically stored in the local variables x, y, z, a, b, c respectively when the script executes unless you force Math Tablet to pass the parameters into other variables using the FUNC function. Note that FUNC automatically declares all of its parameters as local variables.

Compiling: Math Tablet automatically loads and compiles Script Functions as you use them. Before a function can compile, all the variables used in that function must be declared as local variables, or available as global variables in the workspace in which the function is running. You can ensure that a variable is available globally by using the GLOBAL function.

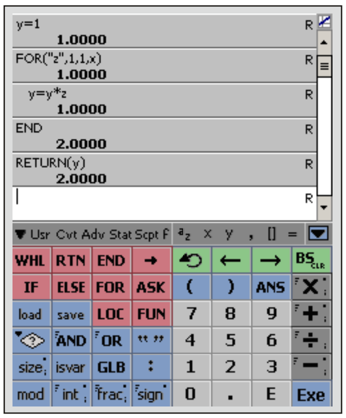
Math Tablet can store up to 20 pre-loaded and compiled scripts at once. If you call on more than 20 scripts Math Tablet must unload and reload scripts as they are used. When you save a script, Math Tablet automatically unloads the script and then reloads and compiles it when you next use it. This ensures that you are always using the most current version of the script. You can force Math Table to unload all compiled scripts using the Clear Loaded Scripts or Clear All menu options.

Script Functions Name Search Order: When you enter a function name in an expression block Math Tablet searches for the function in this order: 1) built in functions, 2) user functions in the workspace, 3) functions in modules , and finally 4) scripts in the Script Directory. Thus you should use unique file names for Script Function workspaces.

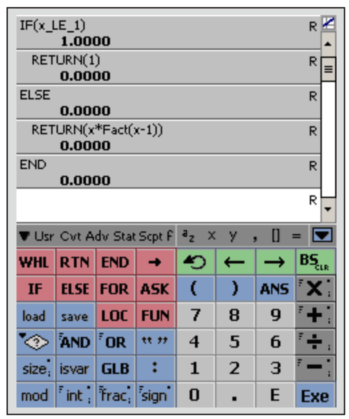
> You can abort the currently running script by tapping on the screen. Math Tablet dislays a message which tells you which script you aborted and on what line.

Example Script

The following example shows two different implementations of a factorial command. The factorial of x is the product of all positive integers, excluding zero, less than or equal to x. The first script uses a FOR loop, the second uses recursion. Both scripts assume that they will be executed from another workspace as a Function Script and that "x" holds the value for which the factorial is computed. (Recall that in a Function Script, the function parameters are automatically stored in the local variables x, y and z.)



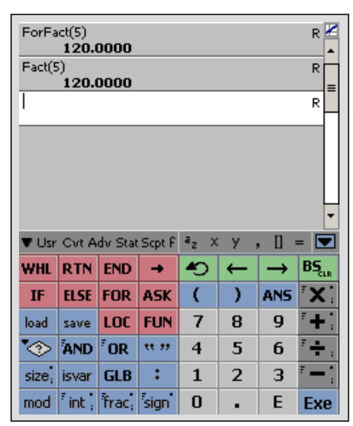
This script uses a FOR loop to compute the factorial of "x". This script was saved as "ForFact"



This script uses recursion to compute the factorial of "x". It's useful to save this script as "Fact" before entering the last RETURN. This forces a "Fact" item on the Variable and Functions pop-up in the Script view so that the text "Fact" can be easily entered into the expression block. The function must then be re-saved after it is completed.

This script uses recursion and is limited by the stack depth in Math Tablet. This version of Fact will fail for values greater than 35.

Once the script has been saved it will appear on the Variable and Functions pop-up in the Script view . The workspace to the right shows how the two versions of the script can be called as Script Functions. Remember that the ForFact and Fact workspaces must be saved in the "Script Directory".



The Scripting module provides commands to create and run scripts in Math Tablet.

Flow Control Constructs

Key	Function	Description	Module
FOR	FOR("x",a,b,c)	Implements a For loop. The counter can be any variable. (replace x with the variable name). a = the starting value for the counter b = the value added to the counter at the end of each loop c = the ending value for the counter. The loop is terminated when the counter is greater than c for positive values of b, or less than c for negative values of b FOR must have a following matching END	Built in
WHL	WHILE(a)	Implements a While loop. The loop continues to execute while the value "a" is non-zero. WHILE must have a matching END	Built in
IF	IF(a)	Implements an IF conditional branch. The branch executes the following block of expressions - up to the matching ELSE if present - when the value "a" is non-zero. IF must have a following matching END	Built in
ELSE	ELSE	Use to create an IF-ELSE conditional branch. When paired with an IF, the ELSE block of expression is executed when the IF conditional is zero. ELSE must have a preceding matching IF and a following matching END	Built in
END	END	Use to terminate the end of a block of expressions in a FOR, WHILE or IF construct	Built in
RTN	RETURN(value)	Exits a function and returns the value.	Built in
LOC	LOCAL("sqp") LOCAL("s,q,p")	Creates a set of local variables in the currently running script. "sqp" or "s,q,p" is a text string which contains the variables which will be local. x, y, z, a, b, c are always local and do not need to be declared in LOCAL. You must separate variable names by commas if they are not single character variables. LOCAL must be invoked before you use any local variables.	Built in

Key	Function	Description	Module
GLB	GLOBAL(“sqp”) GLOBAL(“s,q,p”)	Ensures that the variables s,q,p are global variables. If they do not exist in the workspace in which the script is running, GLOBAL creates them. You should declare all global variable you use in a script with the GLOBAL command. This guarantees your script will run even if the workspace does not currently contain the variables. GLOBAL should be invoked before you use any global variables	Built in
FUN	FUNC(“s,p,q”)	Creates the local variables s, p, q and stores the function parameter in the variables respectively. You can have up to 6 function parameters. If FUNC is not used, function parameter values are automatically stored in x,y,z,a,b,c. FUNC returns the number of parameter the user actually supplies when calling the function.	Built in
isvar	isvar(“var”)	Returns 0 is var is not a variable. Returns 1 if it is a local variable. Returns 2 if it is a global variable.	Built in
load	load(“name”,“var”) load(“name”)	Loads the variable “var” from the workspace file “name”. If “var” is omitted all of the variables in the file are loaded.	Built in
save	save(“name”)	Saves all of the variables in the workspace, or running script, to the file “name” in the current script directory. “name” must be a string variable or literal.	Built in
ASK	ASK(“text”) ASK(“text”,“var”,...)	Displays a text message for the user. If the text end in a “?”, the user is prompted for a Yes, No answer. ASK returns a 0 for a No reply and a 1 for a Yes reply Prompts the user for variable values and stores the values in the variables. Returns a value less than 0 if an error occurs. ASK can prompt for *up to* three variables. For each variable you must provide a text prompt and the corresponding variable. For example: ASK(“Numerator”,“x”, “Denominator”,“y”) prompts the user the “Numerator” and “Denominator” stores those values in the variables x and y respectively.	Scpt

Conditional Tests

Key	Function	Description	Module
LT	x _LT_ y	Returns 1 if x<y, otherwise 0	Scpt
GT	x _GT_ y	Returns 1 if x>y, otherwise 0	Scpt
EQ	x _EQ_ y	Returns 1 if x=y, otherwise 0	Scpt
NE	x _NE_ y	Returns 1 if x- y, otherwise 0	Scpt
LE	x _LE_ y	Returns 1 if x^2 y, otherwise 0	Scpt
GE	x _GE_ y	Returns 1 if x^3 y, otherwise 0	Scpt
AND	x _AND_ y	Returns non-zero if both x and y are non-zero otherwise it returns zero	Hex
OR	x _OR_ y	Returns non-zero if either x or y are non-zero otherwise it returns zero	Hex

Misc. Functions

Key	Function	Description	Module
mod	mod(x,y)	Computes x modulo y. The result is the remainder from the division x/y.	Scpt
frac	frac(x)	Returns the fractional part of x	Scpt
int	int(x)	Returns the integer part of x	Scpt
sign	sign(x)	Returns -1 if x<0 otherwise returns 1	Scpt
→		Inserts spaces into the active expression. This is useful for formatting scripts.	Scpt

The Graphing module extends the graphing capabilities of Math Tablet. It provides special functions for plotting data stored in a matrix, for creating polar plots, and for creating custom plotting functions.

The graphing functions discussed below supplement the graphing capabilities built into Math Tablet. You should use the *built in capabilities* for plotting equations which are simple functions of the independent variable x.

As in all Math Tablet plotting, the expression block must be *selected for drawing*, before the graphing commands will *actually graph*.

Plotting Data In a Matrix
The "plot" command graphs data stored in a matrix as (x,y) pairs. Each (x,y) pair must be stored in a column of the matrix. To store n data points you would use an nx2 matrix. For example, the coordinates (1,4), (3,5), (3,6) could be store in the variable u as:

    u= {1,4|3,5|3,6}

The points could be plotted by

    plot("u")

When the variable name is enclosed in quotes, Math Tablet takes the data directly from the variable storage location (programmers would call this "pass by reference"). You can also plot data directly as

    plot(u) or plot({1,4|3,5|3,6})

In this case the matrix is copied to the expressions block and then plotted from that data (programmers call this "pass by value"). The graph in both cases is the same, however for large data sets plot("u") is more memory efficient because the contents of u do not have to be copied to the expression block before they are graphed.

The plot command can also graph single dimensioned matrices. When the matrix has only one row or column, the data is plotted as the y value, and the x value corresponds to the matrix index. So for example:

    plot({1,4,5,3})

is the same a plotting

    plot({0,1| 1,4| 2,5| 3,3})

You can control how the graph looks using optional parameters which are discussed on the following pages. The default line style for "plot" is "#", (See the next page)

Polar Plots
The “polarplot” function can be used to generate a polar coordinate graph from an equation or from data tabulated in a matrix. Polar plots from tabulated data works just like the “plot” command, except that the columns of the matrix are (r,theta) instead of (x,y).

The polar command can also be used to plot a polar equation. The equation must be entered in quotes, *must be at least two characters long*, and must be written as r = f(t) where t =theta. For example:

polarplot(“2t”,“#”)

plots the spiral r = 2*theta marking each data point with a box

Polarplot nominally plots equations for 0<theta<360 (or 0<theta<2pi if you are in radians). You can change the plot range by specifying the starting and ending values. Note that the polarplot command calculates a discrete set of data point corresponding the specified equation. This number is calculated automatically, but is limited to 500 data points. If the plotting range is too large, for example 0<theta<10000, the plot may not be smooth and you may need to break the range into smaller segments.

As in the “plot” command, you can control how the polar plot looks using optional parameters which are discussed on the following pages. The default line style for “polarplot” is “-”.

Custom Plots
You can generate custom plots using the moveto, lineto commands. These functions let you draw on the graphics screen as if you controlled a pen plotter.

moveto(x,y) moves the “pen” to the specified location on the screen

lineto(x,y) draws a line from the current “pen” location to the specified location.

cleargraph() clears the sequence of moveto, lineto commands

drawgraph() draws the sequence of moveto,lineto commands. “drawgraph” has the same optional parameters as the plot and polarplot commands for controlling the line style. For example, drawgraph(“x”) marks the points on the graph with an “x”. The default style for drawgraph is “-” .

Changing the Line Style

The plot, polarplot and drawgraph commands accept optional parameters that let you specify the color and style of the plot. The line style parameter has the form:

"MLC"

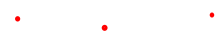
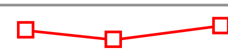
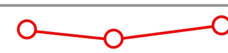
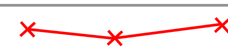
where M = indicates the marker used for each data point
L = indicates whether the data points are connected with a line
C = indicates a color override. If you do not provide a color, the color selected in the expression block is used.

M, L and C are all optional and can be provided in any order.

Color Parameters

C Parameter	Color
"r"	red
"g"	green
"b"	blue
"p"	purple
"t"	teal
"w"	yellow
"y"	grey
"k"	black
none	use expression block color

Line Style Parameters

ML Parameters	Style
"#"	
"o"	
"x"	
"."	
"#-" or "-#"	
"o-" or "-o"	
"x-" or "-x"	
"-"	

Examples:

plot(a,"#-t")	plots the contents of "a" using a teal line with squares on the data points
plot(a,"t")	plots the contents of "a" using a teal line with the default style settings
polarplot("2t","x")	plot the polar eqn. "2t" by marking the data points with an "x"

Plotting From a Script

You can use any of the graphing commands with in a script function. When you use a graphing command with in a script function the graph is shown only if you have selected the particular script function for graphing and the function inside the script is marked for graphing. This makes is easy to create complex custom graphing functions. See the *Bode plot example*,

If a script function does not contain any plotting commands, then the script function is plotted just like any other function in Math Tablet. This means that one of the script's parameters must be specified as "x".

The Graphing module commands are listed below

Plotting Functions

Key	Function	Description	Module
cont if graph	CONTIFGRAPHING() or CONTIFGRAHING(rtn)	Quits the script if the script is not being graphed and returns the value in rtn, if provided. Use this it skip portions of scripts that do not need to run when the script is not being graphed. This can make scripts run faster when not graphed.	Built-in
move	moveto(x,y)	Moves the pen to the x,y location on the graph	Graph
line	lineto(x,y)	Draws a line from the current pen location the the x,y location specified.	Graph
cleargraph	cleargraph()	Clears the sequence of moveto, lineto commands	Graph
drawgraph	drawgraph(Opt1,Opt2)	Draws the sequence of pen moves specified by lineto, moveto when the expression block is selected for graphing. Opt1 is an optional string parameter which specifies the line style use to draw the graph. <u>Those parameters are specifiedon the previous pages</u>. Opt2 is an optional string parameter which specified an additional label for the graph.	Graph
plot	plot(var,Opt1, Opt2)	Plots the data in the matrix var var can be a variable in quotes, such as “a”, or an expression such as 2a+b (no quotes). In both cases var must result in a matrix. If the matrix is n (rows) x 1 (columns) the data is plotted sequentially from the row data. If the matrix is n (rows) x m (columns) the data is plotted sequentially from the first two columns of the matrix where the first two columns represent (x,y) data pairs. Opt1 is an optional string parameter which specifies the line style use to draw the graph. <u>Those parameters are specified on the previous pages</u>. Opt2 is an optional string parameter which specified an additional label for the graph. Examples: plot({1,2,3,4},“oy”,“plot1") plot({1,2,3,4},”-#“) plot({1,2|2,5|4,6})	Graph

Plotting Functions (continued)

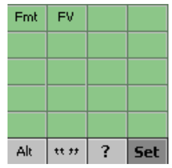
Key	Function	Description	Module
polar	polarplot(exp,b,e, Opt1, Opt2)	Plots the polar expression in exp. exp must be a string value which is a function of the variable "t" (theta). For example: polar("3t+cos(t)") Plots the polar equation: r = 3t+cos(t) IMPORTANT: exp must contain at least two characters! b,e are optional parameters which specify the range of values for which t is evaluated, b<t<e Nominal values are 0,360 for expressions in degrees and 0,2pi for expressions in radians. b and e must be provided as a pair. Opt1 is an optional string parameter which specifies <u>the line style as discussed on the previous pages.</u> Opt2 is an optional string parameter which specified an additional label for the graph.	Graph
polar	polarplot(var, Opt, Opt2)	Plots the values in the matrix var as polar data. var can be a variable in quotes, such as "a", or an expression such as 2a+b (no quotes). In both cases var must result in a matrix. If the matrix is n (rows) x m (columns) the data is plotted sequentially from the first two columns of the matrix where the first two columns represent (r,theta) data pairs. Opt1 and Opt2 and the same as described above.	Graph
label	label(x,y,"text",Opt2)	Prints the "text" at location x,y on the graph. <u>Opt2 is an optional string parameter specifying color</u>	Graph
title	title(x,y,"text",Opt2)	Prints the "text" at the *screen location* x,y on the graph, where (0,0) is the upper left corner and (239,129) is the lower left. Note that the y axis is upsidedown. <u>Opt2 is an optional string parameter specifying color</u>	Graph
text	text(x,"format$_{opt}$")	Coverts the value x to a text string. format is a optional string parameter that specifies the formatting using C sprintf format parameters.	Graph

Supporting Plotting Functions

Key	Function	Description	Module
fill ones	fill, ones	<u>Use create matrices</u>.	Adv
o – x □	- x o #	Graph style parameters.	Graph
{ } $r_1\|r_2$	{} |	<u>Use to create a matrix</u>.	Built-in
[A⋮B]	augc, auga	<u>Use to combine matrice*s*</u>	Adv
	V:	<u>Specifies vectored operations</u>	Built-in

The Formula module lets you create formulas relating several variables and equations and then solve for any of the unknown variables. You can solve these formulas using a custom dialog box, or directly in the expression stack. The Formula module also provides a custom keypad where you can assign a key to your formulas.

Keys for user defined formulas

More user keys

Use to create formulas

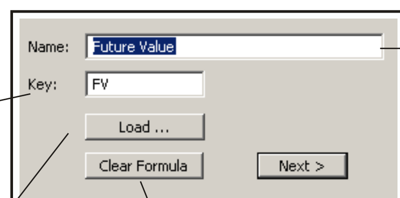
Creating a Formula

1) Tap on the Set key

2) Tap on the key you wish to assign that formula to. This opens up the formula creator dialog shown below. If a formula has already been assigned to a key, this lets you edit the formula.

3) Enter a name for the formula in the top edit box. You should choose a unique name that you have not already used for a formula. This is only required if you want to evaluate the formula directly in the expression stack using the “fm” command.

The name shown on the user key



Formula name. This is used to access the formula directly in the workspace using the “fm” command

Load formulas from a file. Use this to extract formulas others have written and save them to your set. The formula files (*.fdt) you are loading from must be locate in the My Documents folder of your PocketPC

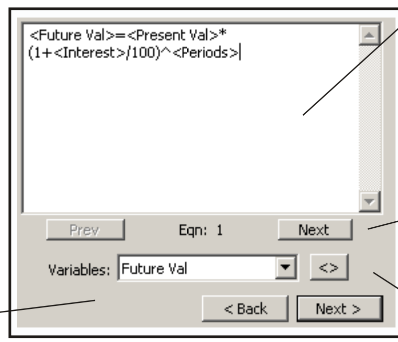
Clears all formulas for this key

4) Enter the formula in the formula edit box. The formula can contain any valid Math Tablet expression. Specify variables in your formula by enclosing them in pointed brackets “< >”. A formula can have up to 14 unique variables and 12 unique but related equations. Variable names can contain any characters except for “<” or “>”, and can have up to 20 characters, formulas can include up to 150 characters. The formula may optionally include an “=”. If it does not include an “=”, the formula is assumed to have the form, formula = 0. Examples of valid formulas are:

Future value formula
<Future Val>=<Present Val>*(1+<Interest>/100)^<Periods>

Power
<volts> = <ohms>*<amps>
<power> = <volts>*<amps>

The formula goes here. Surround variables with < >. You can have up to 14 variables.



Show the next or previous equation if you have multiple equations in your formula. You can have 12 equations

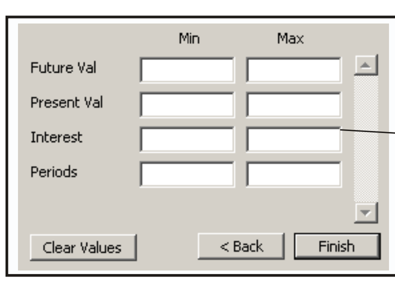
List of variables uses so far. This us updated when you press Next or Prev. Select a variable to enter it

Enters a <> for defining variables

Quick keyboard for entering values. Press ⌨ to see the standard PocketPC keyboard

5) Enter the search ranges for each variable value. If you leave these values blank, Math Tablet will try to bound the solution automatically.

Search ranges are used only when Math Tablet is solving for a single value. Setting an appropriate search range is important in formulas which are undefined for certain value, or where a division by zero can occur.



Enter the minimum and maximum value for each variable

Evaluating a Formula from the Keypad

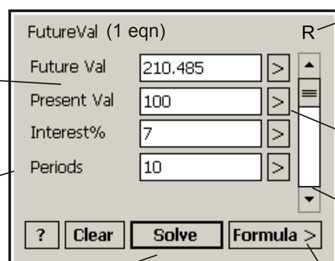
1) Tap on the key representing the formula you want to evaluate. This opens the formula dialog for this formula.
2) Enter the known values in their appropriate locations. You can enter either a constant, or any other valid Math Tablet expression.
3) Leave one or more of the variable boxes blank. These are the variables you wish to solve for. You can only have as many unknowns as there are equations in your formula. This number is shown at the top of the formula dialog. You can provide an initial guess for a value by using a "?"

   To provide an initial guess enter ?# where # is the initial guess.
   To provide a range enter ?{L,U} where U is the lower bound and U is the upper bound.
   For example ?10 starts the search at 10; ?{1,5} searches for a solution only between 1 and 5

4) Press "Solve" and Math Tablet will find a solution to your formula. If multiple solutions exist, Math Tablet will display the first value it finds.

   To find the solution to the formula for the same unknown, but using different known parameters, just change the value of the known parameters and press solve. You do not need to clear the unknown parameter value on successive solves. If you want to change which variable is solved for, clear the box for that variable and press "solve" again.

Enter the known values in the appropriate box. Leave the unknown box blank, or enter an initial guess by preceding the value with a ?



Tap to change between degrees and radians. This changes to a **!** when an error occurs. Tap the **!** to see the error message.

Copy that value to the expression stack

Scroll to see more variables if applicable

Solves for the unknown (blank) value and ? values

Create an expression which will evaluate the formula from the expression stack using the "fm" command.

Formula Errors

Formula errors can result from three sources

1) *Formula Syntax Errors*. These occur if the formulas are not entered as valid expressions. When you try to solve a formula Math Tablet reports the equation which contains the error and the specific error

2) Executable Errors. These occur if an error occurs in one of your equation while the formula is being solved. The might occur if you have a division in a formula and the solver happens to select a test value which causes the divisor to be zero. Executable errors are ignored while the formula is being solved but reported using a **!** on the dialog box. Many errors of this type will not affect the solution.

3) In some cases the solver may not be able to solve the equations because they are "singular". This can happen even if you provide enough know values and provide good initial guesses. Singular equations occur when there is not enough information to solve the problem - regardless of the method used. In this case you must change your formula equations.

Evaluating a Formula from the Expression Stack
Formulas can be solved directly from the expression stack using the fm function. The fm function has the following syntax:

fm("name",value1,value2...)

For example, to solve the future value formula on the previous page for present value, given future value = 250, interest = 3, and periods = 12, you would enter:

fm("Future Val",250,?,3,12)

To solve the formula with an intial guess of 500, enter

fm("Future Val",250,"?500",3,12)

You can graph formula values by replacing the independent value with "x". To plot the previous example for present value verses periods enter:

fm("Future Val",250,?,3,x)

and select the expression for plotting. This plot will show present value verses periods, where the x axis is the number of periods.

If the formula is solving for multiple unknowns, use a ? or "?value" for each unknown. fm will return the values in a matrix

Technical Details
The Formula module uses a nonlinear equation solver to find the unknown values.

When there is only one unknown it uses a method loosely based on the Secant Method. This method normally requires that you provide an upper and lower bound for the solution. In the formula module this bound is calculated automatically by performing a couple of iterations based on Newton's method starting at the initial guess (or zero if no guess is provided), until a bound on the solution is found. Overall this method is method is fairly robust. In certain cases, it may not be able to bound the solution, even if you provide a guess using the "?". In this case you should use the "solve" function and explicitly provide an upper and lower bound on the solution.

If the formula contains multiple equations and multiple unknowns, the equation solver uses a method loosely based on Newton's method. This method requires a multidimensional search and is not as robust as the method employed when there is only one unknown. There are several cases in which the solver may have problems:

1) If the equations are poorly scaled the solver may not be able to find the solution without an initial guess. In this case you should provide an initial guess using the "?" as described above.

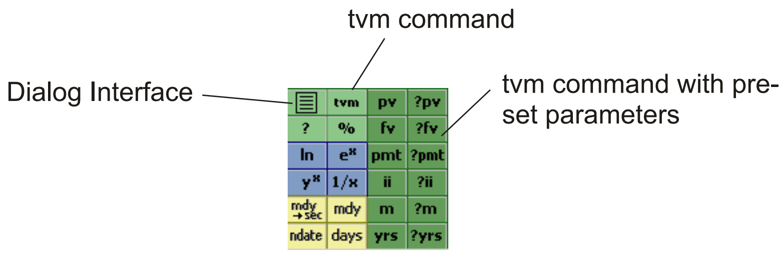
2) If you do not provide enough known values the solver may not be able to solve the equations. In this case it will solve the equations is can, and leave the rest blank.

3) In some cases the solver may not be able to solve the equations because they are "singular", as discussed on the Formula Errors section.

Formula Functions

Key	Function	Description	Module
	fm(name,var1,var2...)	Solves the formula named name for the unknown value where var = ? or is an initial guess provided as "?value" where value is the initial guess. name must be the name of an existing formula. Name must be a text string var1, var2... must be known values. These values must be entered in the same order as the in the dialog for the formula. One of the variables should be replaced with a ?. The function will solve for that value.	Formula
?	?	Enters the "?" string. This is used to identify the unknown value in the fm command	Formula

The Time Value of Money (TVM) module lets you solve financial problems involving the time value of money. The TVM module includes a dialog interface, similar to the Formula module, and a command line interface.



The TVM module uses the following parameters. These parameters are stored in variables described below

pv = present value
fv = future value
pmt = payment per period
ii = % interest (enter 10 for 10%)
m = number of payment periods per year
yrs = number of years

Given any five of these values, TVM will solve for the remaining value using the following formula:

$$fv = pv(ii/100+1)^{(m*yrs)} - pmt((ii/100+1)^{(m*yrs)}+1)/(ii/100)$$

You can solve for these values two different ways

Using the TVM function

tvm(pv,fv,pmt,ii,m,yrs)

To solve for an unknown value, use the TVM function and replace the unknown value with a ?. TVM will return the unknown value. For example tvm(-100,?,1,8,1,10) returns the future value of $100 put into an account for 10 years at 8% interest.

If you store values in predefined variables names, as listed above, you can use the function on the popup pad to enter the TVM equation quickly. For example if you have already assign:

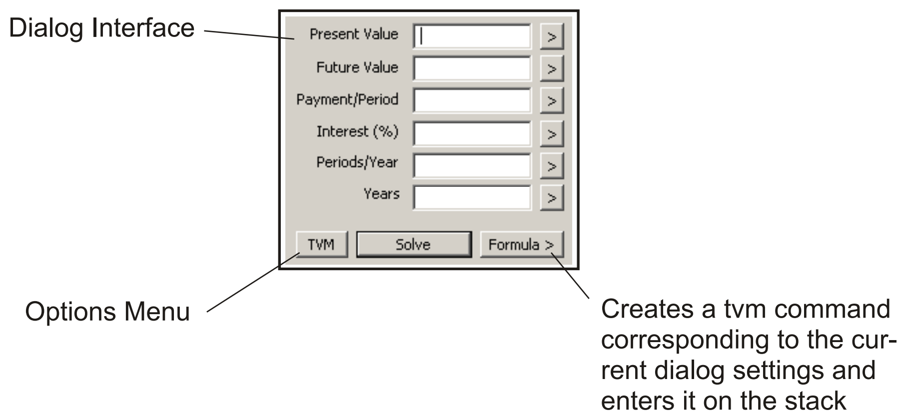
pv = -100
pmt = 1
ii = 8
m=1
yrs=10

Enter: tvm(pv,?,pmt,ii,m,yrs) which solves for the future value.. The TVM keypad includes keys which enter the tvm automatically. The ?fv key, for example, enters the tvm function so that it solves for fv.

Using the TVM dialog box

The TVM dialog lets you solve the tvm equations using a dialog form. Enter the known values into the dialog box. Leave the unknown value blank. Press Solve to solve for the unknown.

The dialog' s option menu can be used to load and save values to and from the workspace into the dialog's edit boxes.



Options Menu	
Load Values	Loads specific variables into the currently selected edit box.
Load All TVM Values	Loads the tvm variables (pv, fv, pmt…) into their respective edit boxes.
Save All TVM Values	Save the values in the edit boxes into their respective tvm variables on the workspace. If the variables don' t exist they are created.
Reveal TVM Values	Copies the values stored in the tvm variables on the workspace into their respective edit boxes
Create Formula	Creates a tvm formula based on the current entries in the edit boxes. The formula will be created with a ? in the entry that you last solve for in the dialog box.

TVM Time Value of Money

The TVM module commands are listed below

TVM Functions

Key	Function	Description	Module
	tvm(pv,fv,pmt,ii,m,yrs)	Solves for the unknown parameter in the time value of money problem. Enter the unknown value as ?	TVM
mdy →sec	mdytsec(mmddyyyy)	Returns the number of seconds since Jan 1, 1601, where mmddyyyy is the month, day, year	TVM
ndate	ndate(mmddyyyy,nd)	Returns the date, as mmddyyyy, nd days from the specified date	TVM
days	days(mmddyyyy, mmddyyyy)	Returns the number of days between two dates	TVM
%	x % y	Returns x * (y/100)	TVM

Example #1

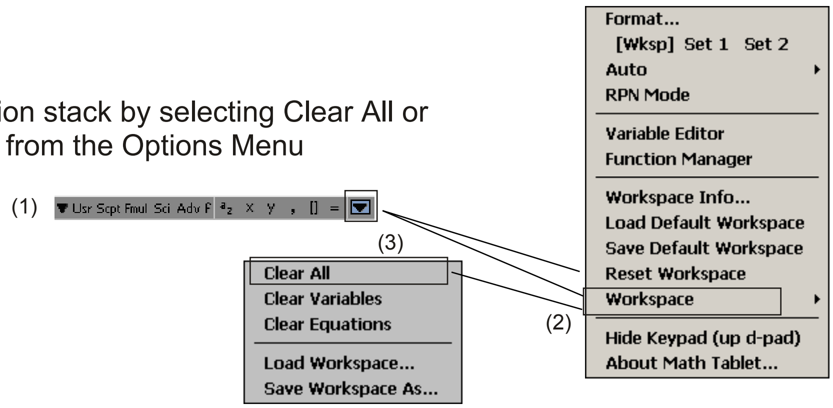
The following example problems illustrate how you can use Math Tablet to solve a variety of problems.

Problem: Balancing a checkbook. This example illustrates how Math Tablet's power can be put to use in even simple (yet often frustrating) problems.

Solution: Enter the starting balance and then subtract check values using the ANS feature to maintain a running total.

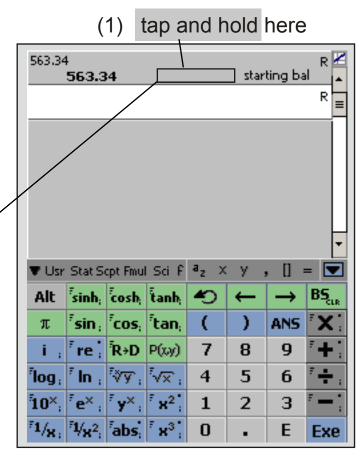
Step-by-Step:

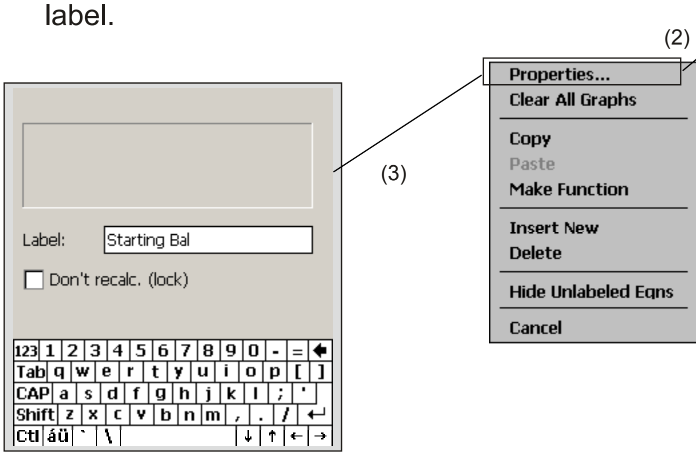
1. Clear the expression stack by selecting Clear All or Reset Workspace from the Options Menu



2. Tap on the first expression block and use the key-pad to enter the balance as shown. Then press EXE

3. Add the "starting bal" label to the equation by tap-ping and holding the pen on the lower portion of the block where you just entered the balance ($563.34). The select "Properties..." and enter the label.

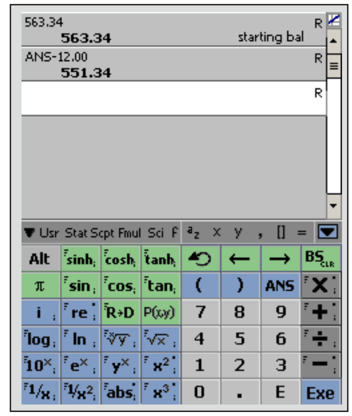




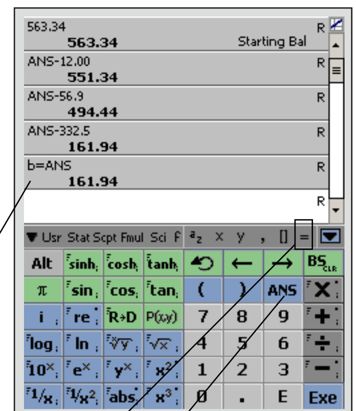
(continued)

4. Subtract the first check entry ($12.00) from the current balance. Use the ANS key to recall the current running balance. Press EXE when you have completed the expression.

   Short-Cut. Set the Auto ANS feature from the Options menu and each expression will automatically begin with ANS

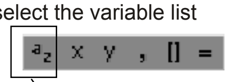


5. Continue subtracting check entries from the running balance as shown.

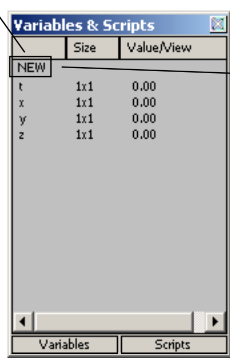


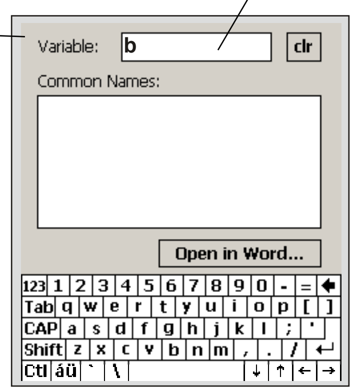
6. You can store the final balance in a variable if you wish so that you can recall it later. In the example the balance is stored in the variable "b".

(1) select the variable list



(2) select NEW





(4)

(5)

(3) Enter "b" and press OK This creates a new variable named "b" and enters it in the expression. If you want to use this variable again, tap and select "b" from the list.

The advantage of using Math Tablet for such a simple arithmetic problem is that Math Tablet remembers each entry you make on the expression stack. If your final balance does not match your bank statement you can review all the checks that you have entered. If you make a mistake, simply go back, change the value of the miss-entered check and press EXE. Math Tablet will automatically recompute your new balance. You can also save your workspace so that you can recall it next month and continue where you left off.

Problem: Find the solution to the simultaneous equations

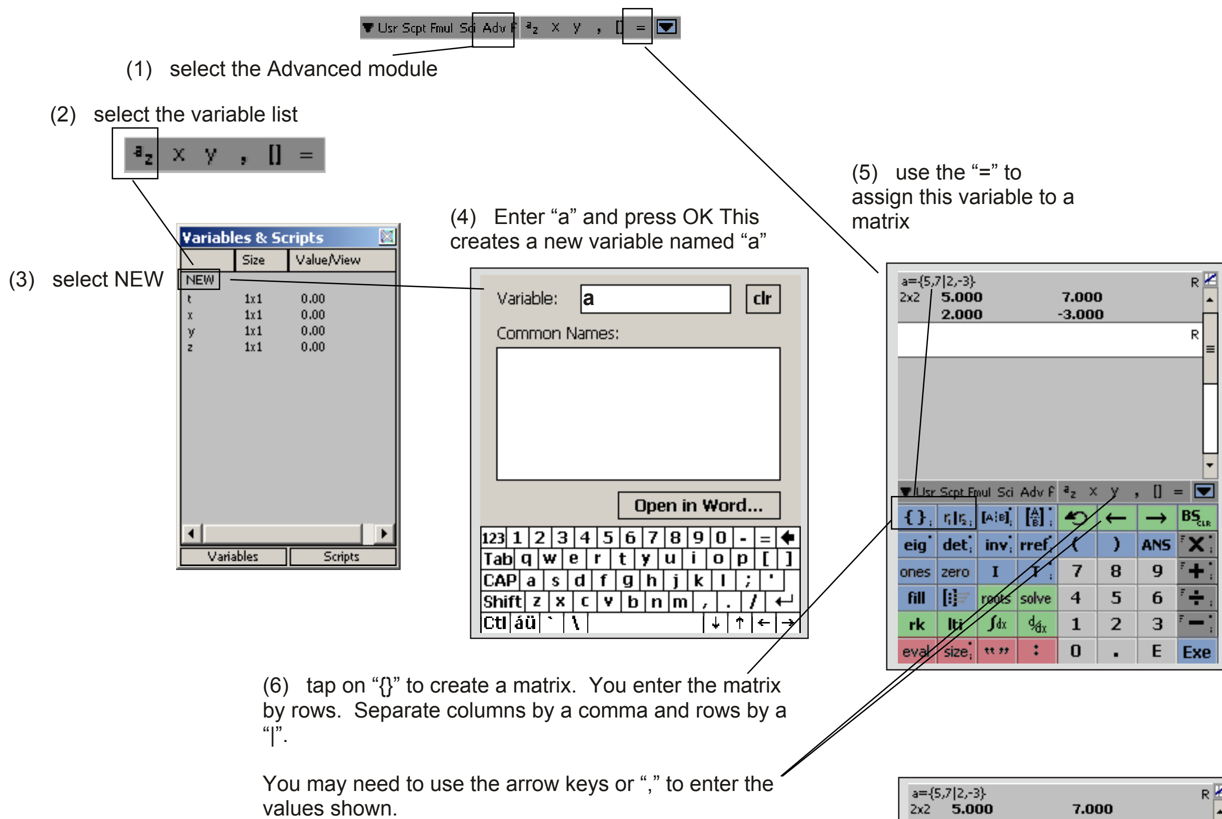
$$5x+7y \;= 2$$
$$2x - 3y \;= -4$$

Solution: Write the equations in matrix form Ax = B and solve for x here x = inv(A)*B

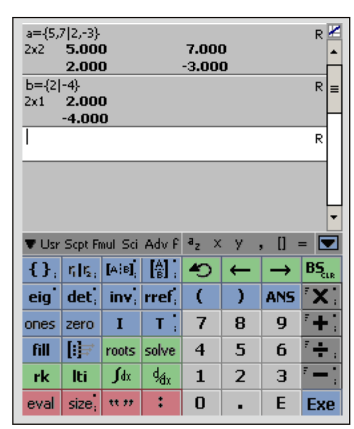
Step-by-Step:

1. Clear the expression stack if desired. This is not necessary, you can always leave your old work on the expression stack in case you want to review it later. Refer to step 1 of the previous example for instructions on clearing the expression stack.

2. Define the matrix A



3. Define the matrix B as shown using the same technique as in step 2



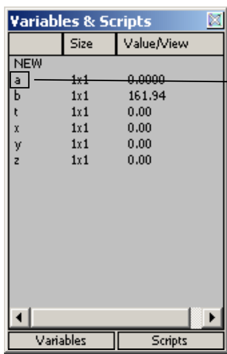
(continued)

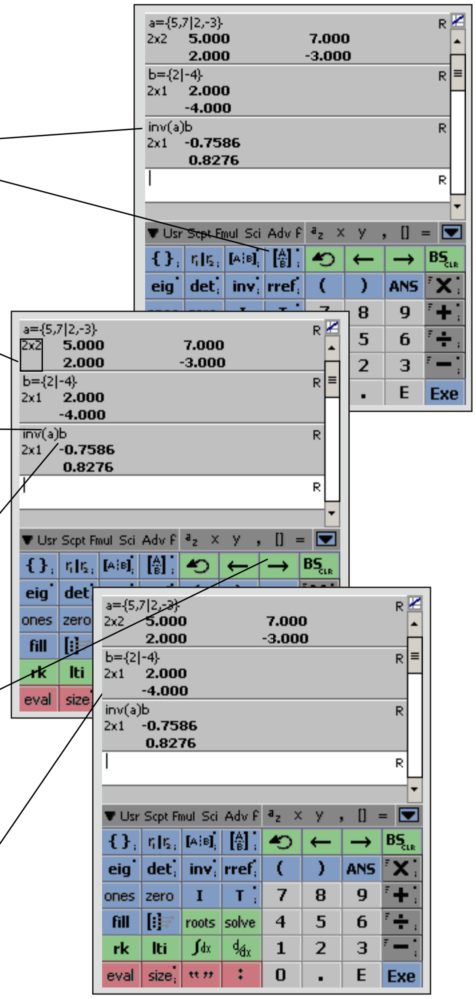
4. Solve for the solution using the inverse of “a”

(1) tap “inv” to enter the inverse function

(2) tap directly below the “a” in the “a” expression block. This enters “a” into the active expression. In our case, we will have “inv(a)”. It is easy to recall a variable that you have used in a previous expression using this method.

Alternately you can tap on ^{a}z and tap on “a” or “b” from the variable list





(3) use the right arrow to move the cursor to the outside of the parenthesis.

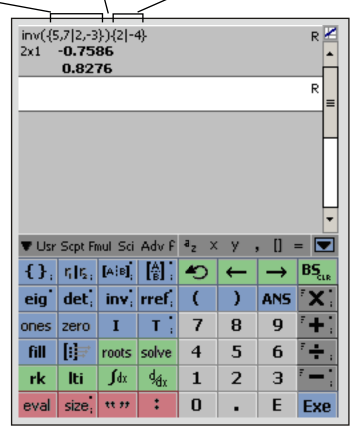
(4) enter “b”, recalling it from the previous expression, by tapping just below the “b” in the previous expression.

Alternately you could compute the solution in one expression as shown below, or you could create an augmented matrix and use the “rref” function.

implied multiplication

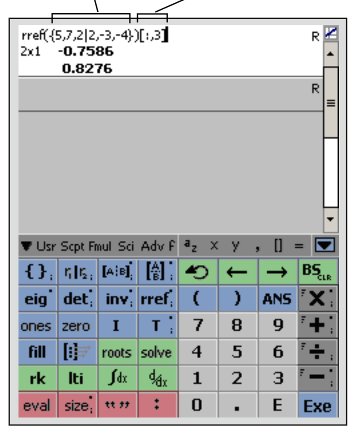
a matrix

b matrix



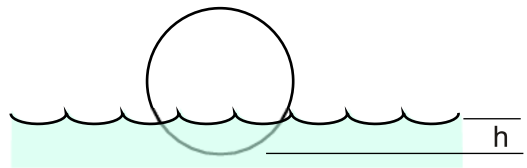
augmented system matrix

extract the 3rd column from the rref results



Example #3

Problem: Find the distance a sphere of radius (r) will sink in water (h) for the following values of specific density for the sphere: 0.2, 0.4, 0.6



Solution: The equation representing the force balance between the sphere's weight and its boyancy is given as:

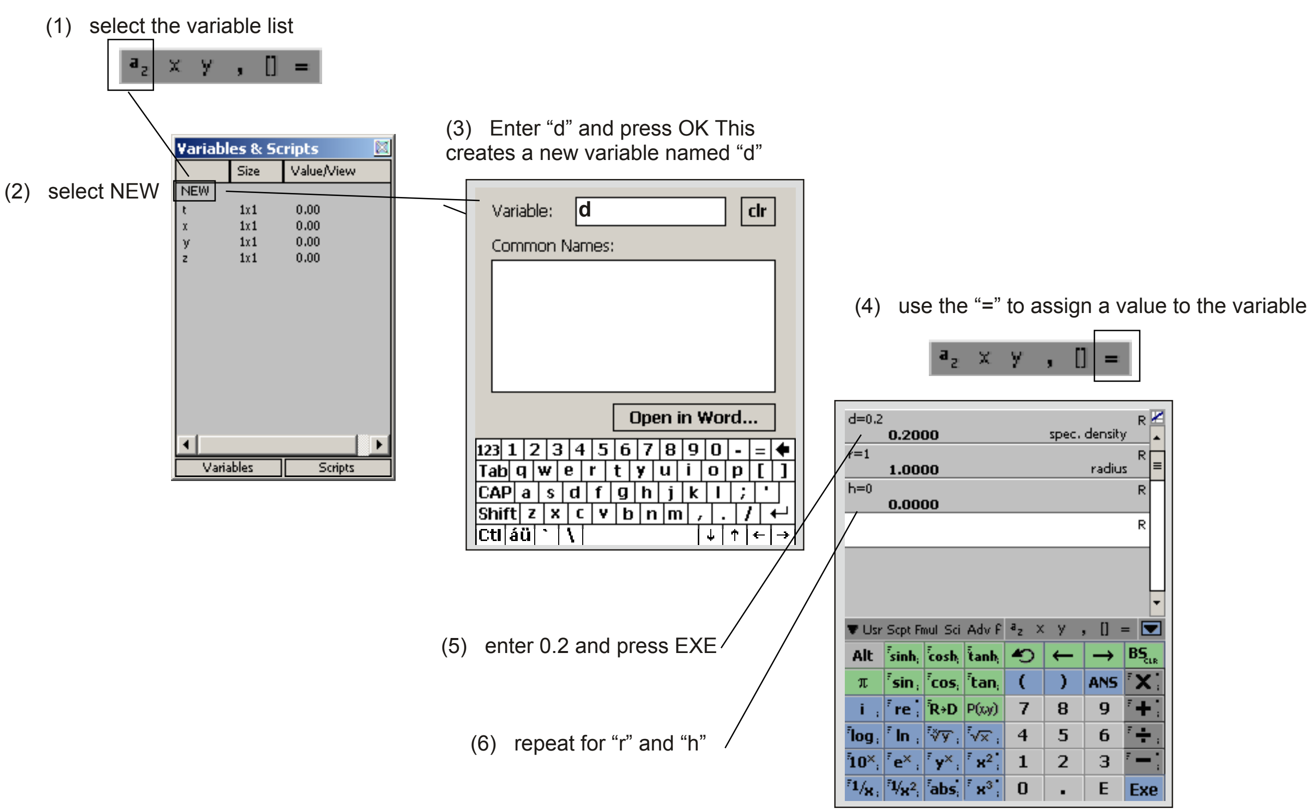
$$\frac{Pi}{3}(3rh^2\text{-}h^3) - \frac{4Pi}{3} r^3 d = 0$$

Solve for the height (h) for different values of specific density (d)

Step-by-Step:

1. Clear the expression stack if desired. This is not necessary, you can always leave your old work on the expression stack incase you want to review it later. Refer to step 1 of the first example for instructions on clearing the expression stack.

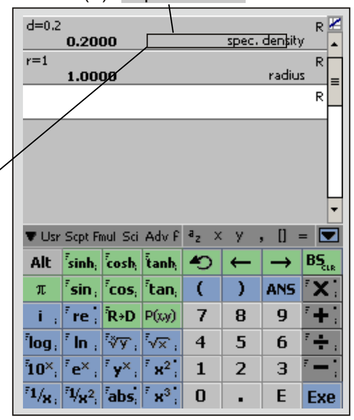
2. Enter the values for the parameters "d" and "r". These values will be defined using variables so that their values can be easily modified later.



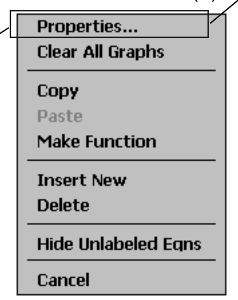
(continued)

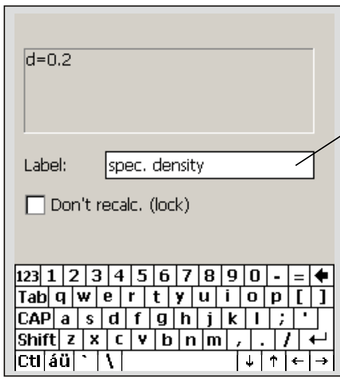
3. Add the “spec. density” and “radius” labels to the equations by tapping and holding the pen on the lower portion of the appropriate blocks. Then select “Properties...” and enter the label.

(1) tap and hold here



(2)

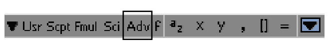




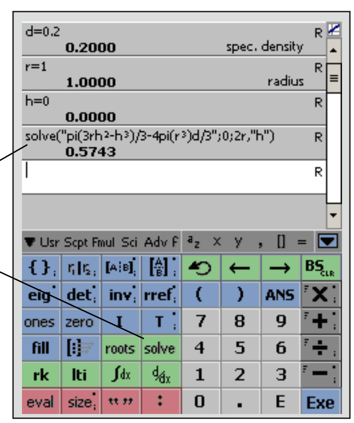
(3)

4. Solve the nonlinear equation for “h” using the “solve” function.

(1) switch back to the Advanced module

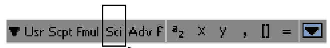


(2) tap on “solve” to enter the solve function



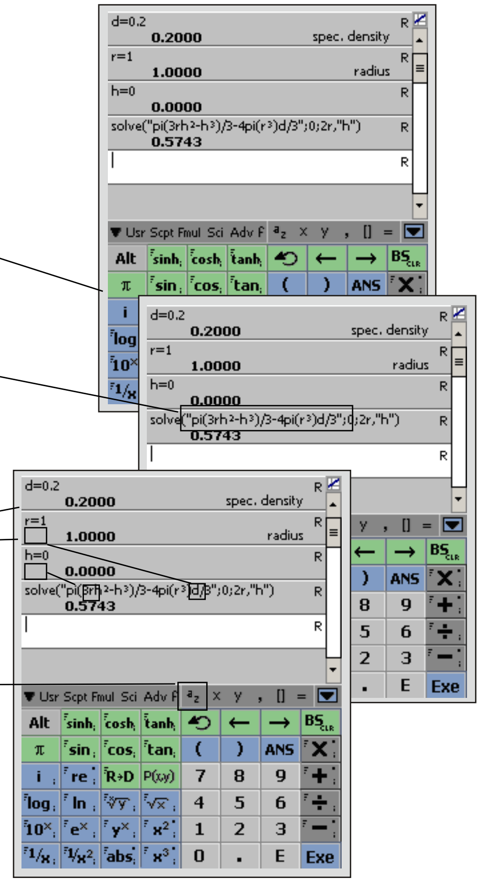
(continued)

(3) select the Scientific module so you can enter the equation to be solved

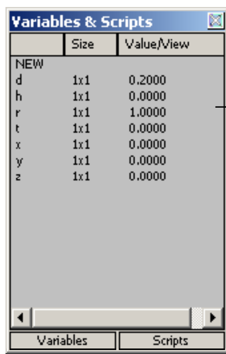


(4) use the scientific keypad to enter the equation. This should be entered inside of the quotes in the solve function. See step (5) below for a short-cut for entering variables used in previous expressions.

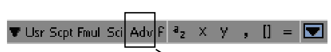
(5) enter “r” or “d”, recalling them from the previous expression, by tapping just below the variable in the previous expressions. This takes practice, so be patient.



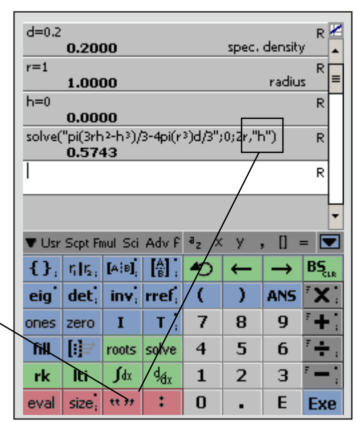
Alternately you can select the variable from the variable list



(6) switch back to the Advanced module so you can enter the “h” (in quotes) as the last parameter of the solve function



(7) tap EXE to evaluate the expression



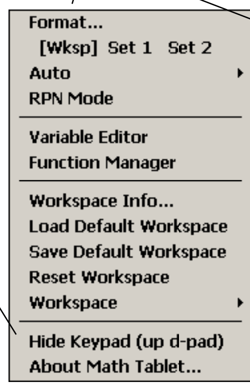
(continued)

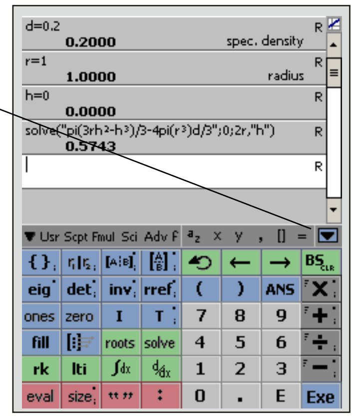
Useful Alternate Method

You can use the PocketPC keyboard to define variables or enter expressions.

(1) Open the Options Menu

(2) Tap Hide Keypad



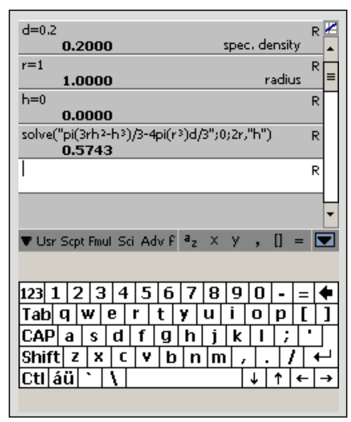


OR (1) Press the UP direction of the cursor pad on your PocketPC. Press UP again to hide the keyboard

(3) Type in all, or some of, the expressions using the keyboard

RETURN will enter an expression

If you enter “r=1" MathTablet will automatically create the variable “r” for you, so you don’t have to use the method on the previous page

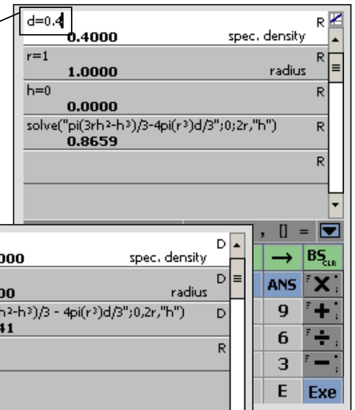


(continued)

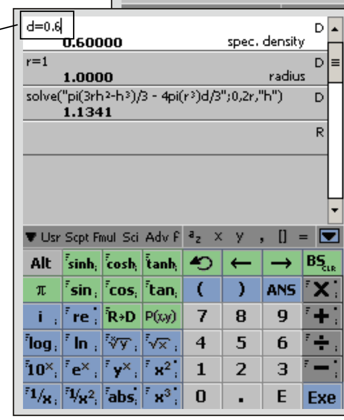
5. Re-solve the equation for different values of "d".

(1) tap on the "d" expression and change the value for "d". Tap EXE when the change is complete.

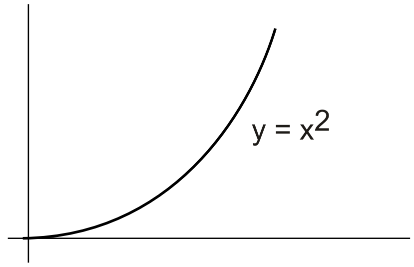
The solution changes to reflect the change in "d"



(2) repeat step (1) for d = 0.6



Problem: Find the area under the curve for 0>x>10



Solution: The area can be found by integrating the curve. This integral can be written as a single, or a double integration.

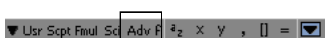
$$\text{Area} = \int_0^{10} x^2 dx \quad \text{or} \quad \text{Area} = \int_0^{10}\int_0^{x^2} 1 \, dy \, dx$$

Step-by-Step for Single Integration:

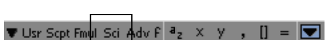
1. Clear the expression stack if desired. This is not necessary, you can always leave your old work on the expression stack incase you want to review it later. Refer to step 1 of the first example for instructions on clearing the expression stack.

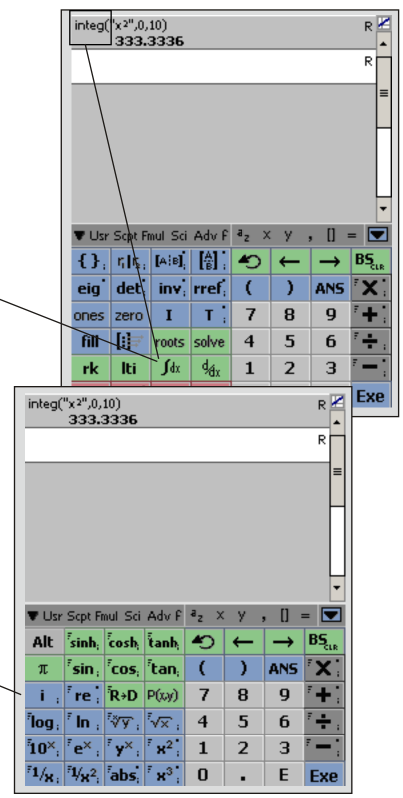
2. Enter the formula for the integration

(1) switch to the Advanced module so you can enter the integration function



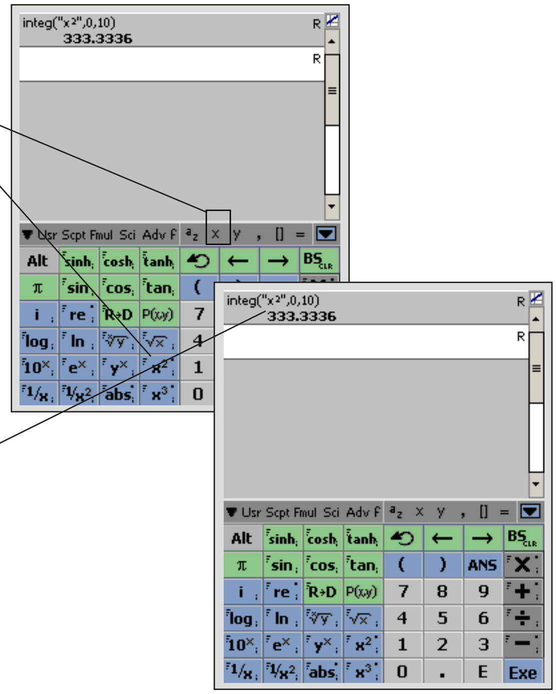
(2) switch to Scientific module





(continued)

(3) place the cursor between the “” and enter x^2



(4) use the arrow keys to enter the integration limits 0 and 10. Then press EXE to evaluate the integration

Step-by-Step for Double Integration:

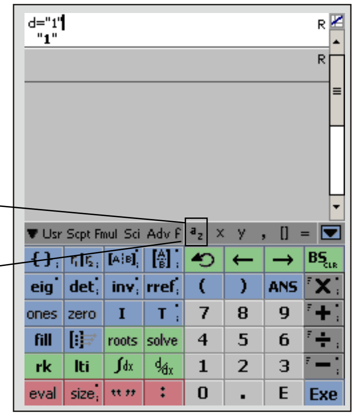
1. Clear the expression stack if desired. This is not necessary, you can always leave your old work on the expression stack incase you want to review it later. Refer to step 1 of the first example for instructions on clearing the expression stack.

2. A double integration requires that you integrate a function that uses the integration function. The integral would be written like this:

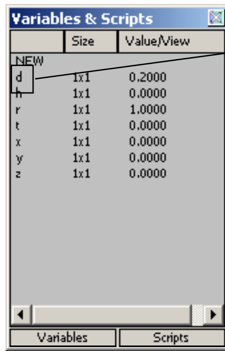
   integ(“integ("1",0,x)”,0,10)

   Notice that the outside integration function is integrating a function (the value in quotes) which itself is an integration function. The problem is that this inner integration function also has the expression to be evaluated placed in quotes. Thus, the inner set of quotes is embedded inside the outer set of quotes. This is not allowed in Math Tablet (or in most software). The solution is to replace the inner quoted function “1" with a string variable.

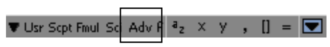
(continued)



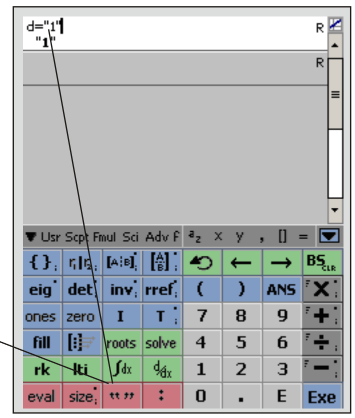
(1) enter “d” by using the variable list. Select “d” from the list. If it doesn’t exist, select NEW and create a new “d” variable



(2) set it equal to a string by placing the value 1 in quotes



Aletrnate, use the PocketPC keyboard to enter the expression. See the Alternate Method in <u>Example #3</u>



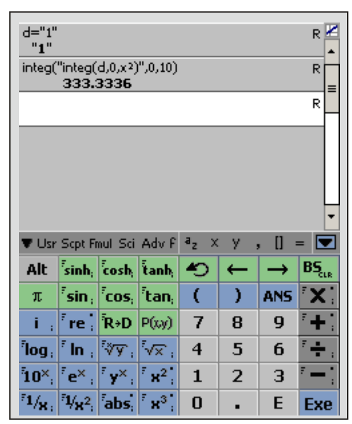
(3) press EXE

4. Enter the double integration formula as shown and press EXE

(1) switch to the Advanced module so you can enter the integration function twice.

(2) switch to the Scientific module so you can enter the integration limits of x^2 .

(3) press EXE when the expression is completed



Problem: Graph the path of a projectile shot with an initial velocity of (p) and an angle of (u)

Solution: The equations governing a projectile are as follows

$y = (g/2)\,(x/w)^2 + v(x/w)$ and $x = w\,t$

where
- y = vertical position
- x = horizontal position
- g = acceleration of gravity (-32.2 ft/s/s)
- t = time
- w = initial horizontal velocity
- v = initial vertical velocity

If the initial angle is u and the initial velocity is p then

$w = p*\cos(u)$
$v = p*\sin(u)$

Note: The workspace for this example is provided as part of the software installation. Use "Load Workspace..." to load in "example5.mtw" if you desire. Then skip to step 5.

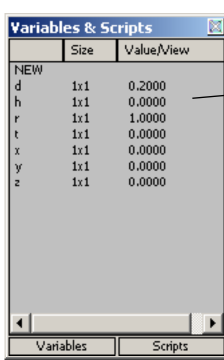
Step-by-Step:

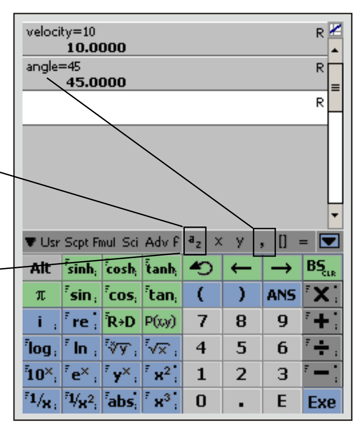
This examples focuses on graphing in Math Tablet. It assumes that you have reviewed the previous examples and are familiar with the basic operations of Math Tables.

1. Clear the expression stack if desired. This is not necessary, you can always leave your old work on the expression stack incase you want to review it later. Refer to step 1 of the first example for instructions on clearing the expression stack.

2. Enter the expressions for "velocity" and "angle" as shown

(1) Select NEW from the Variable popup and create a new variable "velocity" or use the keyboard to enter the expression



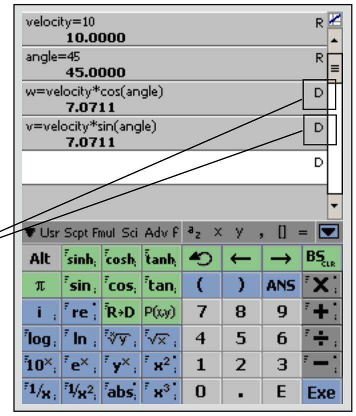


(2) repeat for the "angle" expressions

(continued)

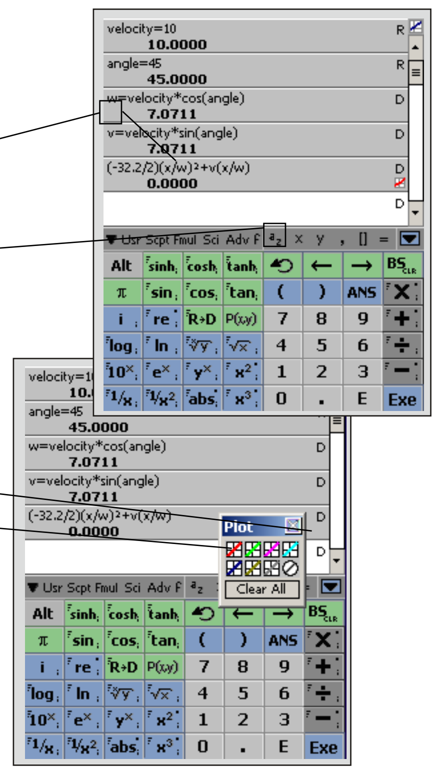
3. Enter the expressions for w and v. Use the variable dialog to access the variables. You may also have to switch to the Scientific mode to enter the “sin” and “cos” functions.

   Make sure you switch the angle calculations to degrees



(1) click here to switch between degrees and radians

4. Enter the expressions for the position verses “x” as shown. Then mark the expression to be graphed. Press EXE when completed.



(1) tap directly below the “w” or “v” in the previous expressions to enter them into the currently active expression.

Alternately you can use the variable dialog

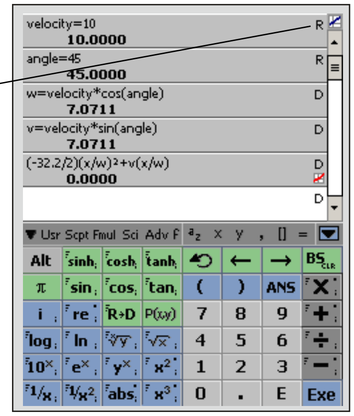
(2) click here and

(3) select a color to mark the expression to be graphed

(continued)

5. Switch to the graph screen to see the graph screen

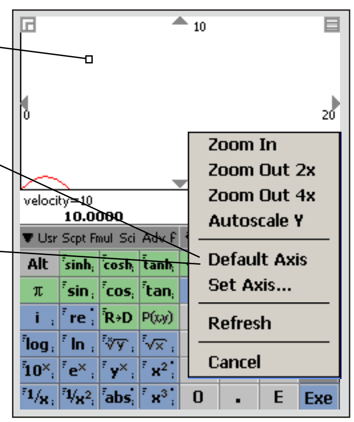
(1) click here to change to the graph screen

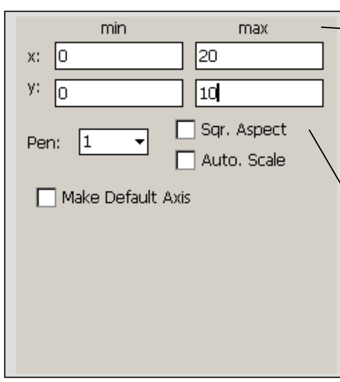


6. Change the graphs axis to $0 < x < 20$ and $0 < y < 10$.

(1) tap and hold the pen on the graph

(2) select Set Axis... from the popup menu





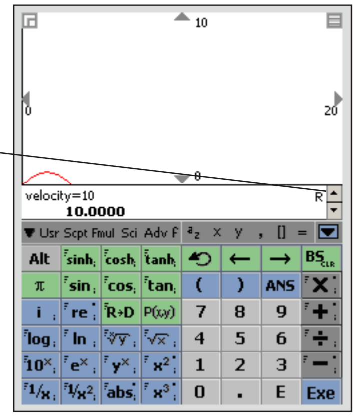
(3) Change the axis values to those shown

(4) Tap OK to apply the values

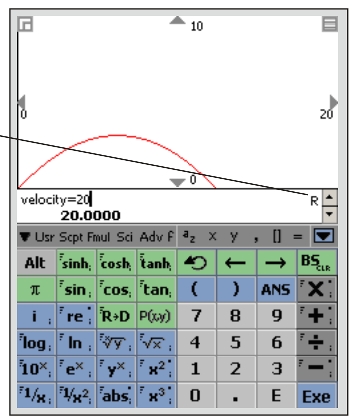
(continued)

7. The graph now shows the relevant portion of the projectile's trajectory. Scroll the expression blocks until you can see the velocity expression. You could also switch back to the expression stack view and edit the value there. Change the value of velocity and press EXE. The graph is updated to show the new trajectory.

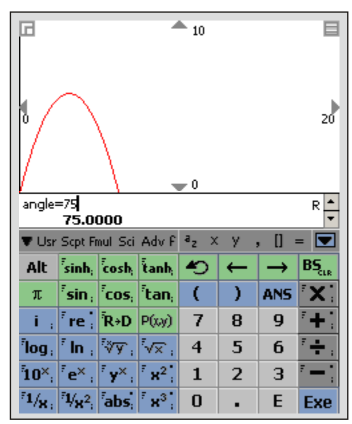
(1) scroll the expression stack until you can see the velocity equation



(2) scroll the expression stack so you can see the velocity expression. Edit the velocity expression and press EXE. The graph is updated



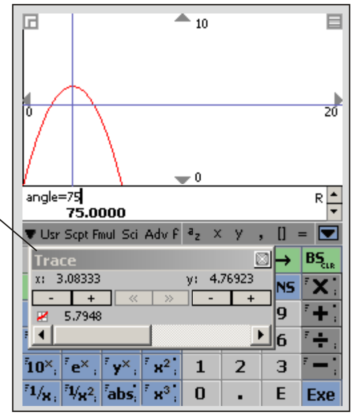
(3) scroll the expression stack so you can see the angle expression. Edit the angle expression and press EXE. The graph is updated



You can experiment with different velocities and angles and see how the trajectory changes.

(continued)

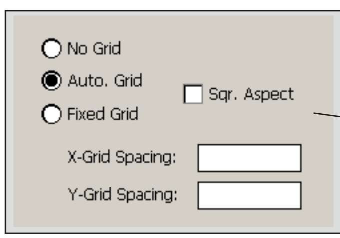
8. Tap anywhere on the graph to bring up the trace box. This shows the value of the curve where it crosses the trace marks

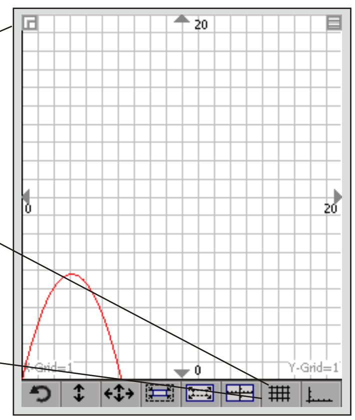


9. Enable full screen graphing with a grid

(1) tap the zoom box to enable full screen graphing

(2) tap the grid button to bring up the grid dialog and select auto grid





Alternately you can tap and hold the pen on the graph and select Grid from the popup menu

Example #6

Problem: Determine the first "n" prime numbers

Solution: Write a script which searches for prime numbers. To determine if a number is prime, the script will attempt to divide the candidate by all prime numbers less that that number. If it can not be evenly divided by any of these primes, the candidate value must be prime and is added to the prime number list.

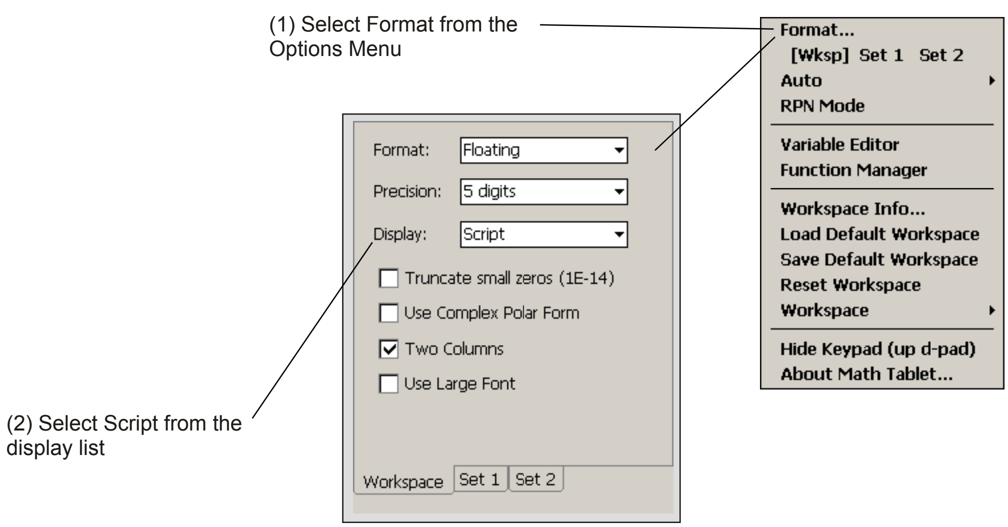
For reference, the first eight prime numbers are: 2, 3, 5, 7, 11, 13, 17, 19.

Step-by-Step:

This examples focuses on scripting in Math Tablet. It assumes that you have reviewed the previous examples and are familiar with the basic operations of Math Tables. This example also assumes that you are familiar with a computer programming language such as BASIC, FORTRAN, Pascal or C.

1. Clear the expression stack and make sure that you have set the script directory.

2. From the options menu select Format->Scripting. This eliminates the results from the expression block and allows you to see more expressions on the screen at one time.

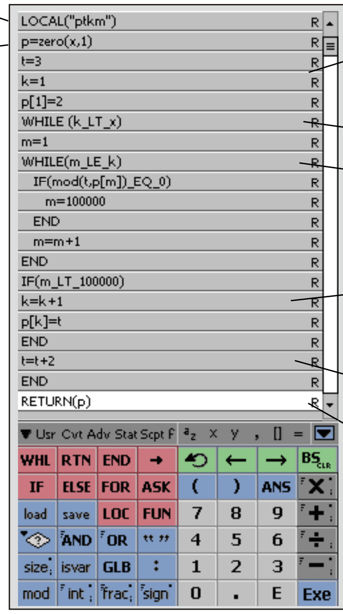


3. Switch to the Scripting module and enter the expressions as shown

Declare the local variables p,t,k,m. x is also used but it is automatically local

The first parameter passed to the script is automatically stored in x. This is used to determine the number of primes to find, so allocate a matrix to hold the values.

The screen has been extended in this figure. You will have to scroll to see all the expression blocks



t is the current candidate prime
k is the number of primes found so far

Loop until we have found x primes

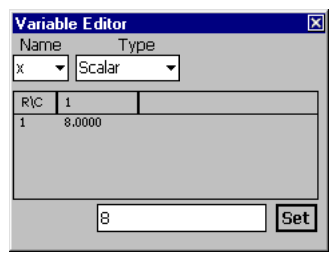
Loop through all the currently known primes and check to see if any divide evenly into the current candidate (t)

If a number is not evenly divisible by any of the known primes, add it to the prime list.

Get the next number for consideration.

Return the matrix that holds the answer

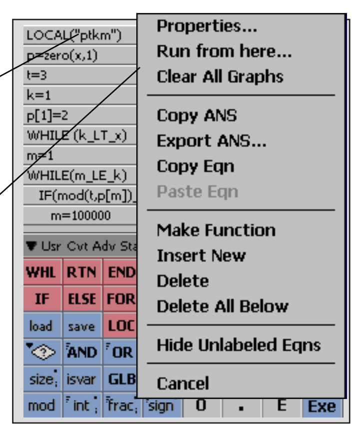
4. To test the script, use the Variable Editor from the Options Menu to set the variable “x” to a value of 5. This simulates what will happen you call the script from another workspace and pass it a parameter.



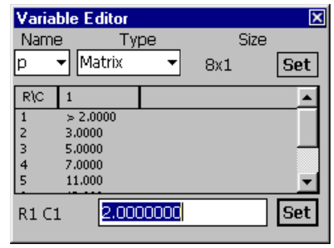
5. Tap and hold the pen on the first expression in the script and select “Run from here...”

(1) tap and hold

(2) select Run from here...



6. Use the Variable Editor to check the value stored in the variable “p”. It should contain the first 8 prime numbers. When you run this script from another workspace the value of “p” will be returned from the script.



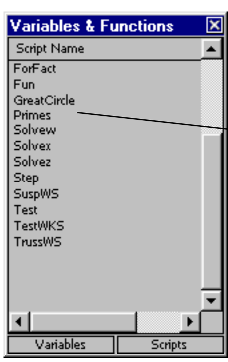
Once the script is working properly, you can save the script and use it as a function in another workspace. This is the real power of scripting.

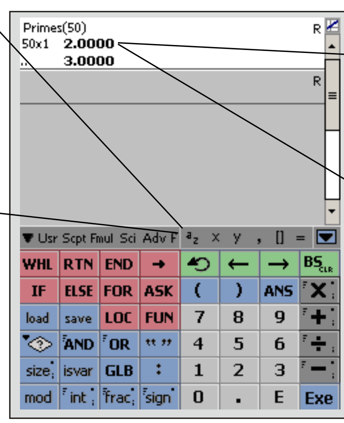
7. Save the workspace as “Primes.mtw” in the script directory . The function will require a single parameter. The parameter is automatically stored in the local variable “x” once the script runs. In this case the parameter is, x, the number of primes to find.

8. Clear the workspace and reset the display options to something other than Scripting. This is so that you can see the results of the script when you run it.

9. From the Variable and Functions popup, select Primes from the Script view and pass it a single parameter value as shown.

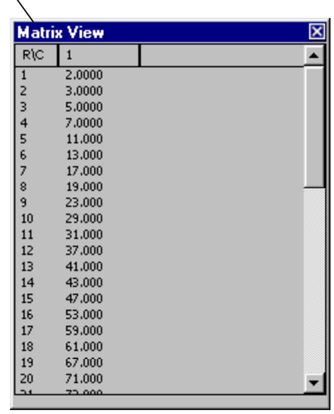
(1) tap on a_z, then Scripts, finally tap on Primes from the script list.





(2) Type in the value for the first parameter. Press EXE to run the script.

(3) the primes are stored in the matrix returned by the script. Tap on the results to open the matrix in the matrix viewer



Example #7

Problem: Plot the expression y = sin(x). This simple example illustrates the different ways you can graph in Math Tablet.

Solution: This graph will be generated three different ways
1) Using Math Tablets automatic function graphing capabilities
2) By generating discrete data points and grahing using the plot function
3) By writing a script to perform the graphing

Step-by-Step:

This examples assumes that you have reviewed the previous examples and are familiar with the basic operations of Math Tables.

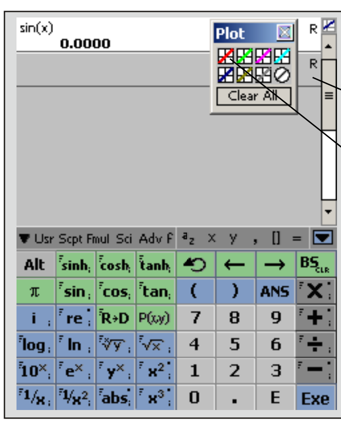
1. Clear the expression stack (optional).

Method I Built In Plotting

2. Enter the equation as a function of "x" onto the expression block

3. Enable plotting of this this function by tapping just beneath the R or D. You will see the plot box as shown below. Select the color for the plot.

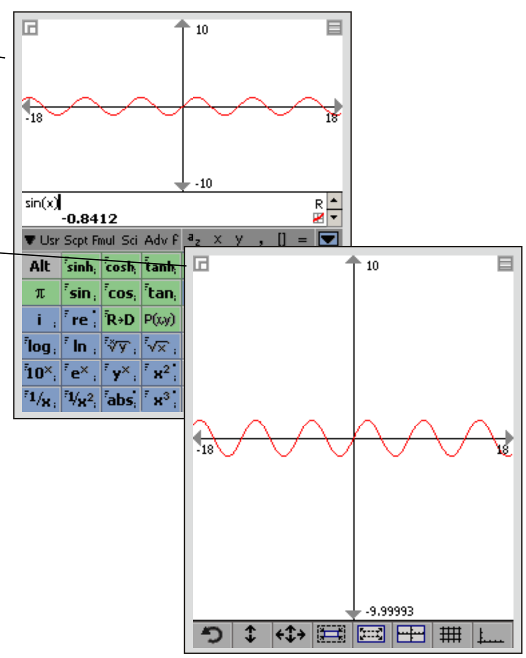
4. Show the graph by selecting the graphing screen



(2) Tap here to show the graph

(1) Tap here then here to enable graphing

Tap here switch between full and half screen graphing



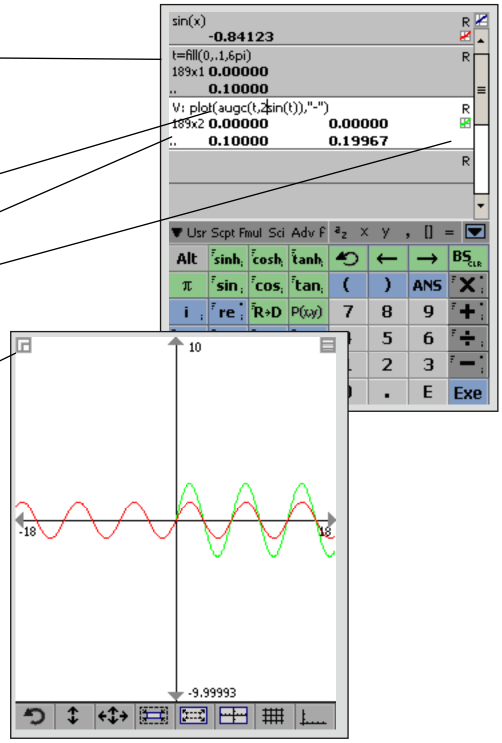
5. Optional: You can change the graph limits by tapping and holding the pen on the graph and selecting "Set Axis..." from the pop-up menu.

   You can rescale the graph by tapping and holding the pen on the graph and then selecting "Autoscale" from the popup menu

(continued)

Method II Plot Function

6. Create a list of points at which to evaluate the function using the "fill" function as shown



(1) use the "fill" command to create a array of point from 0 to 6pi radians, spaced every 0.1 radian

(2) create an array where the first column contains the values from step (1) and the second column contains the "sin" of those values. Note we use <u>vectored operation</u> to perform this in one expression.

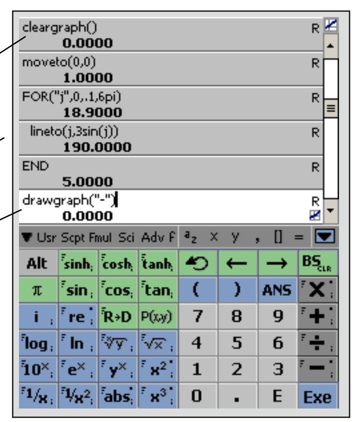
(3) "plot" the array of points, select the expression for graphing. The "plot" command is part of the graph module

(4) show the graph. The graph show here is in full screen. Tap here to switch between full and half screen

NOTE: In this example the second plot is scale by a factor of 2 so that both plots are visible on the screen

Method III Plotting Using a Script

7. Enter the script shown to the right



(1) clear the "draw graph" graph buffer and move to a starting location

(2) loop through all the values and draw a line from the last location to the new location.

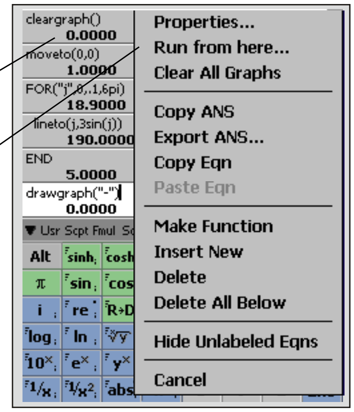
(3) draw the graph using "drawgraph" and select the expression for graphing

(continued)

8. Run the script. Tap and hold the pen on the first expression in the script to bring up the <u>expression popup menu</u>. Select “Run From Here...” to execute the script. This draws the series of lines in to the graphic buffer.

(1) tap and hold the pen here to see the popup menu

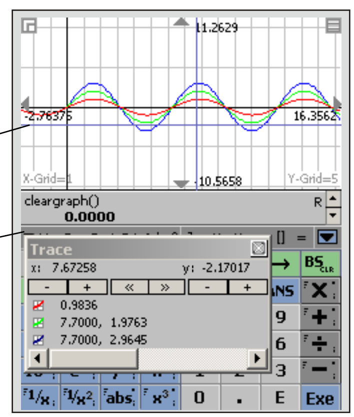
(2) select “Run from here...” to execute the script and load the plot into the graph buffer



NOTE: In this example the second plot is scale by a factor of 3 so that both plots are visible on the screen

9. Show the graph

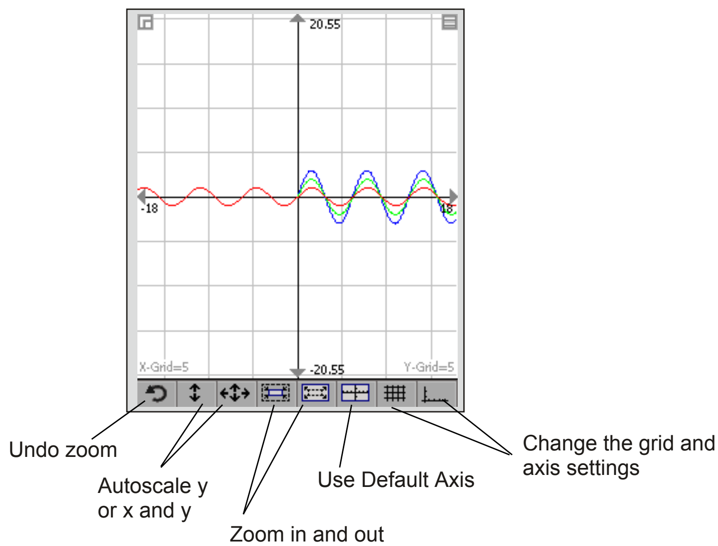
You can trace the graph by tapping once on the graph. The trace box only shows known values. You may see a “--” if the trace line is over a location where an exact data point is not available

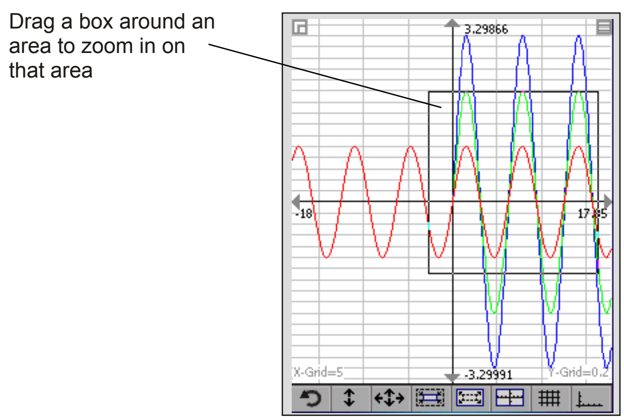


You can rescale the graph by tapping and holding the pen on the graph and then selecting “Autoscale” from the popup menu. You can show the grid by selecting “grid” from the same popup menu.

You can rescale the graph by tapping and holding the pen on the graph and then selecting “Autoscale” from the popup menu. You can show the grid by selecting “grid” from the same popup menu.

When in full screen mode, you can tap and hold the pen or use the buttons at the bottom of the graph.





Problem: Create a “Bode” plot showing the gain and phase for a transfer function as a function of input frequency.

Note: Bode plots are commonly used signal processing and control systems applications. On a bode plot, the input frequency is plotted on a log scale, the output gain is plotted a decimals (db), and the output phase shift is plotted in degrees.

Solution: Write a plotting script function. The input to the script will be the transfer function and the frequency limits of the plot.

Step-by-Step:

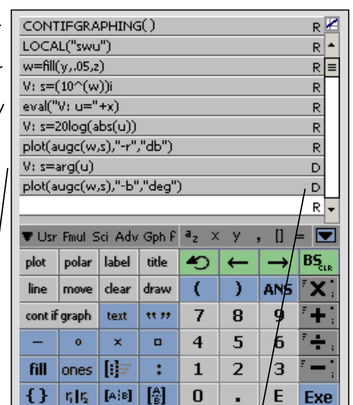
This examples assumes that you have reviewed the previous examples and are familiar with the basic operations of Math Tables.

1. Clear the expression stack
2. Enter the script shown

The script uses the following parameters:
x = a string containing the transfer function as f(s)
y = log(smallest frequency)
z = log(largest frequency)

(1) keep the function from running unless it is plotting

(2) create the frequency space to evaluate the transfer function. The frequencies are created using a “log” axis



(3) evaluate the transfer function at every value in the frequency space. First create the text string:
“V: u= f(s)”
where f(s) is the actual transfer function. Then evaluate this function using <u>vectored operation</u>. This evaluates the function at every value in the frequency space.

(4) compute the gain and phase of the results. Again, use vectored operations to perform the analysis on all of the values at once

Important: You must mark this function for plotting in order for it to plot when used as a script. If you are showing only one line, then you must tap and hold near the D, select Properties... and select it for graphing

3. Save the script as “Bode” by selecting “workspace” then “save as...” from *the options menu*.

(continued)

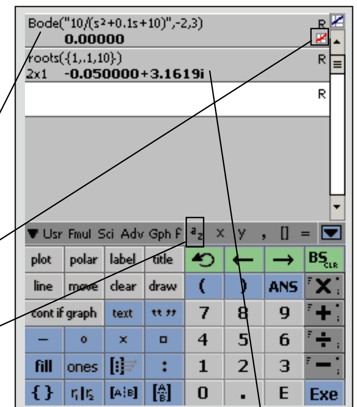
The Bode function can now be use to create a bode plot for any transfer function. In this example we will analyze the transfer function

$$\frac{Y}{U} = \frac{10}{s^2+0.1s+10}$$

4. Clear the expression stack

5. Execute the Bode function as shown.

6. Check the results by computing the poles of the transfer function. They should be where the "peak" in this bode plot occurs.

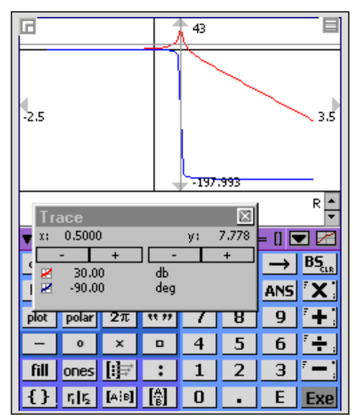


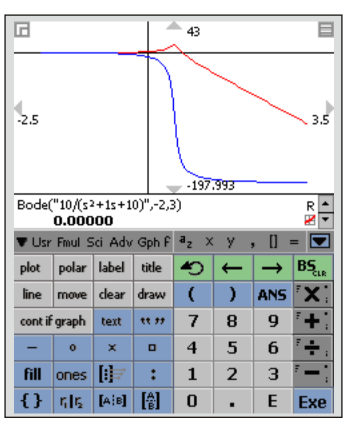
(2) enter the parameters for the bode function. Also select the expression for graphing

(1) tap here, then select the Script view and select Bode

(3) compute the poles of the system for comparison

7. Show the graph. Then tap on the graph to bring up the trace box and place the cross hairs at the plot's peak. The peak occurs at 0.5 on the x axis. This corresponds to a frequency of $10^{0.5}$ rad/sec (because we plotted using log of the frequency) or 3.16 rad/sec. 3.16 matches the value of the pole computed in step 6.





8. You can easily see how the parameters in the transfer function affect the bode plot. For example, change the 0.1 in the denominator to 1.0 and press EXE. Math Tablet automatically re-evaluates and graphs the updated transfer function.