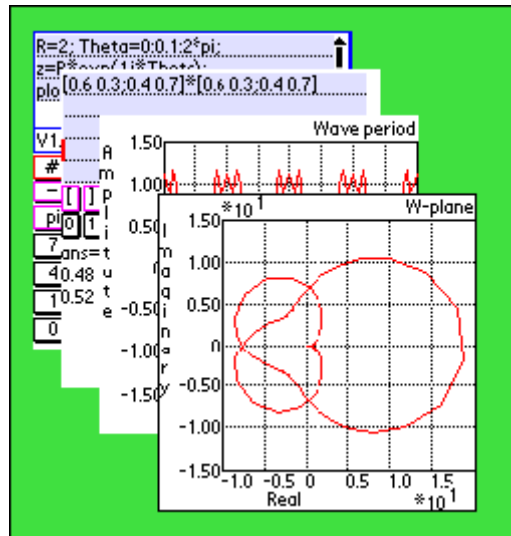


MtrxCal

The matrix calculator for the palm platform.



User manual

Version 1.80

ADACS LLC

Advanced Digital & Analog Consulting Service

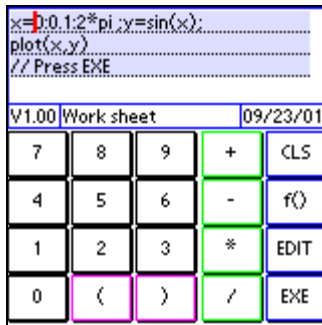
www.adacs.com

Table of contents

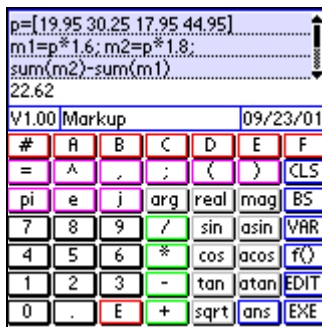
SCREEN SHOTS.....	3
HOW TO USE MTRXCAL.....	5
ADVANTAGE OF A MATRIX CALCULATOR.....	6
MATRIX MULTIPLICATION.....	6
MATRIX INVERSE.....	8
FUNCTIONS.....	9
ARITHMETIC OPERATORS	9
RELATIONAL OPERATORS	9
LOGICAL OPERATORS.....	9
FUNCTIONS	9
COMPLEX FUNCTIONS	10
MATRIX FUNCTIONS	10
GRAPHICAL FUNCTIONS	11
TRIGONOMETRY FUNCTIONS	11
HYPERBOLIC FUNCTIONS	11
FLOW CONTROL.....	12
SPECIAL FUNCTIONS.....	12
MTRXCAL SPECIAL FUNCTIONS.....	12
EXAMPLE SCRIPTS.....	13
NET PRESENT VALUE.....	13
WAVE PERIOD.....	14
SOLVE QUADRATIC	15
SINE TEST.....	15
RC NETWORK.....	16
DIODE CIRCUIT	16
CLOSED CONTOUR.....	17
LEGAL AND DISCLAIMERS.....	18
CONTACT INFORMATION.....	18

MtrxCal is the most powerful programmable matrix calculator especially designed for the palm platform. Enter numerous values in one matrix and perform calculations on all the values at the same time. Plot all the values with a simple command using a linear or even logarithmic scale. Yes, even auto-scaling is supported. Let MtrxCal do the work for you!

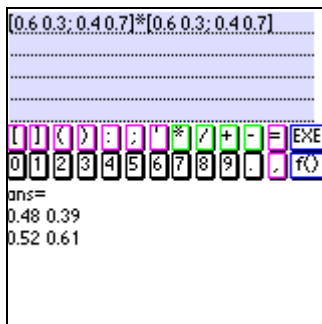
Screen shots



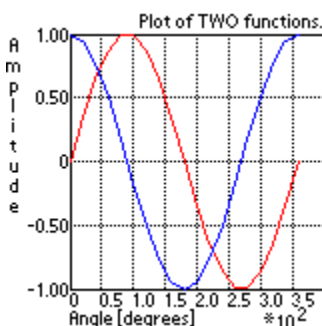
Select between several screens. The layout of the screens can be easily changed by editing a little text file. These text files contain the text for each button separated by a comma.



The only difference between this screen and the screen above is the text file used to assign the buttons. Notice how the top three lines are evaluated when the exe button is pressed and the result is displayed on the last line. The matrix p contains four values and is multiplied in line two by 1.6 and 1.8. A scroll bar is used when the script contains more than three lines.

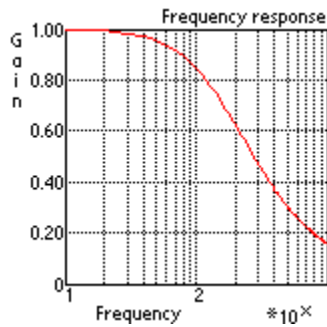


This is the matrix screen. The buttons are smaller. These buttons also can be easily changed by editing a little text file.



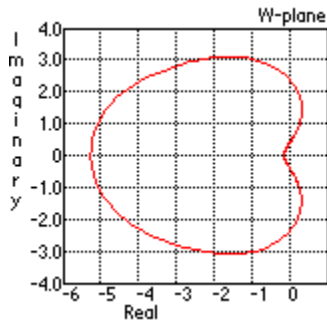
```
x=0:20:360;
r=x./180*pi;
plot(x,sin( r ),cos( r ))
title('Plot of TWO functions.')
xlabel('Angle [degrees]')
ylabel('Amplitude')
```

Note: The last three lines are only to print the labels and title. These are not needed for drawing the plot. To save space on the limited screen a multiplication factor is used for the x-axis. In this case 100.



```
R=1000 ;C=1e-6;
Fc=1/(2*pi*R*C)
f=[10:10:1000];
w=1j*2*pi*f;
H=1./(1+w.*R*C);
semilogx(f,abs(H))
title('Frequency response')
xlabel('Frequency')
ylabel('Gain')
```

Note: The second line is not terminated with a semi-colon. Therefore the result of the second line will be displayed.



```
R=0.6;
Theta=0:0.03:2*pi;
z=R*exp(1j*Theta);
H=(z-1)./(z+1).*(z.^2+z+1));
plot(H)
title('W-plane')
xlabel('Real')
ylabel('Imaginary')
```

How to use MtrxCal

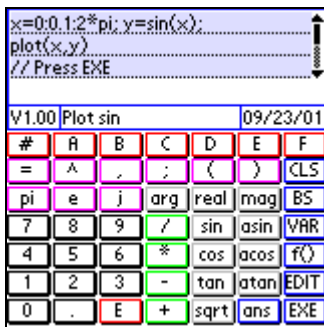


Fig. 1



Fig. 3

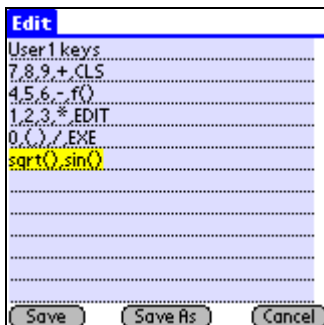


Fig. 5

After starting MtrxCal for the first time the screen as shown in fig. 1 appears. When you only use a palm calculator to balance your check book this might be a bit intimidating. Press the menu button beside the graffiti and the menu should appear. Select keys and select 'user 1'. The screen as shown in fig. 2 should appear. Press menu again and now select progs as shown in fig. 3. select Edit and fig. 4 should appear. Tap on the text 'User 1 keys' and a little text file should appear as shown in fig. 5. This file, without the last line, was used to create the screen in fig. 2. Now add the last line and press save. The screen as shown in fig. 6 should appear. This is just an example of how easy it is to change the screen. Everybody uses a calculator differently and this allows you to put the buttons where you like them and only the buttons you use.

MtrxCal has two types of screens. A special screen for matrix calculations and a main screen. The special screen for matrix calculations is shown when you select matrix under keys in the menu. The matrix screen has only room for two lines of buttons in the middle of the screen. The main screen allows for a lot more lines of buttons. The only thing fixed is the area in which the buttons are displayed and this depends on the screen type.

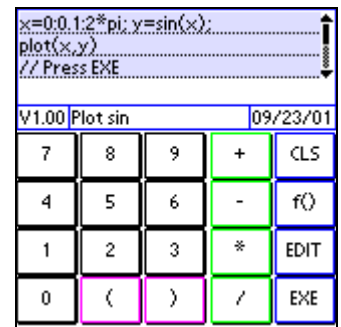


Fig. 2

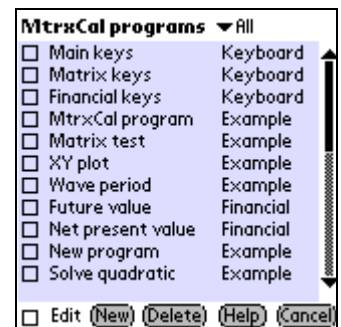


Fig. 4

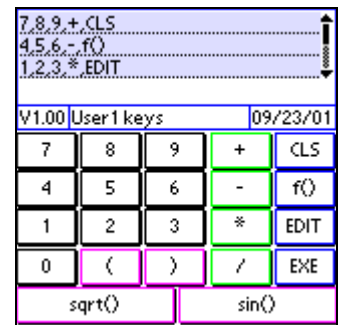


Fig. 6

Advantage of a matrix calculator

Lets just start with a simple example. Image you are selling products with prices of \$19.95 \$30.25 \$17.95 and \$44.95. You want to increase the mark-up from 1.6 to 1.8. What is the difference if you sell one of each?

```
P=[19.95 30.25 17.95 44.95];
m1=p*1.6;          m2=p*1.8;
sum(m2)-sum(m1)
```

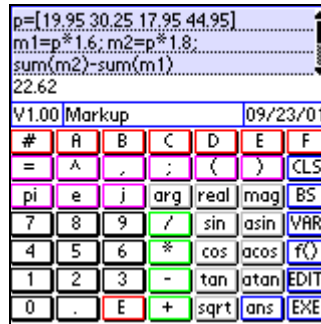


Fig. 1

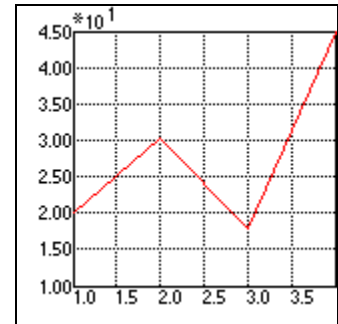


Fig. 2

Enter the script, well the three lines, as shown in Fig. 1 and press [exe] at the bottom. Change the last line to `plot(p)` and press [exe] and Fig. 2 appears. Normal calculators work with one value at a time. MtrxCal works with as many values as there are in the matrix. If the markup is not the same for each product you can create an other matrix like `m=[1.6 1.7 1.6 1.8]` and enter `tot=p.*m`. Notice the period in front of the *. This is needed when you would like to multiply each element, value, individually. Without the period MtrxCal will use the matrix multiply instead.

-

Matrix multiplication

Let A and B denote two matrices that have this characteristic: The number of columns in A equals the number of rows in B. These matrices are conformable with respect to each other, and they can be combined by an operation known as the multiplication of matrices. Let C denote the matrix formed by multiplying A and B. Matrix C is the product of A and B, and the operation is expressed symbolically as $C=AB$.

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + a_{i3}b_{3j} + \dots + a_{in}b_{nj} \quad (1)$$

Well are you impressed already? When I see this kind of formula for the first time my first thought is yeh yeh just give me a practical example of how useful it is and then I might be impressed.

Let take a look at an example with two cities A and B. Every year 40% of the people in city A move to city B. Every year 30% of the people in city B move to city A.

After one year we have in city A: $0.6A + 0.3B$ (2)

After one year we have in city B: $0.4A + 0.7B$ (3)

After two years we have in city A:

$$0.6 (0.6 A + 0.3 B) + 0.3 (0.4 A + 0.7 B) = 0.48A + 0.39B \quad (4)$$

After two years we have in city B:

$$0.4 (0.6 A + 0.3 B) + 0.7 (0.4 A + 0.7 B) = 0.52A + 0.61B \quad (5)$$

Ok let me stop here and leave it up to the reader to keep on going for the next couple of years.

Let's rewrite the first year into a matrix: $y^1 = \begin{bmatrix} 0.6 & 0.3 \\ 0.4 & 0.7 \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} 0.6A & 0.3B \\ 0.4A & 0.7B \end{bmatrix}$ (6)

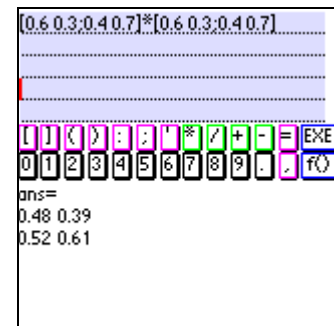
For year two we just write:

$$y^2 = \begin{bmatrix} 0.6 & 0.3 \\ 0.4 & 0.7 \end{bmatrix} \left(\begin{bmatrix} 0.6 & 0.3 \\ 0.4 & 0.7 \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} \right) \quad (7)$$

Ok time to enter an equation into MtrxCal. First select the matrix screen and then type:

`[0.6 0.3; 0.4 0.7] * [0.6 0.3; 0.4 0.7]`
and press `[exe]`.

The screen on the right should appear.



Look back at the equations (4) and (5)

Instead of `[0.6 0.3; 0.4 0.7] * [0.6 0.3; 0.4 0.7]` you could write the follow script:

```
n=2;  
y=[0.6 0.3; 0.4 0.7];  
p=y^n
```

If you calculated the populations after 5 years then you can verify the result by changing $n = 2$ to $n = 5$ and press [exe].

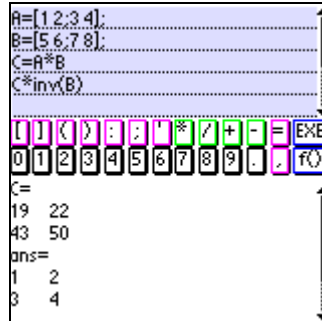
Apply equation (1) to the two matrices in equation (7) and here is what MtrxCal did to multiply the two matrices.

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{bmatrix} \quad (9)$$

Matrix inverse

When $A * B = C$ then $A = C * inv(B)$

A=[1 2;3 4];
 B=[5 6;7 8];
 C=A*B
 C*inv(B)



Functions

Arithmetic operators

+	Plus
-	Minus
*	Matrix multiply
.*	Array multiply
/	Matrix divide
./	Array divide
^	Matrix power
.^	Array power
\	Solves linear equations. Solves a square coefficient matrix A and a single right-hand side column vector b. The solution, $x=A\backslash b$, is the same size as b. Example: $A=[2\ 3; 4\ 5]$; $b=[4\ 6]'$; $x=A\backslash b$ Press exe and MtrxCal will show the result. It is that simple! Note: don't forget the `'-sign to transpose b.

Relational operators

==	Equal to
~=	Not equal to (matlab syntax)
!=	Not equal to (CplxCalPro syntax)
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
	Example: $x=[1:4]$; $x>=3$

Logical operators

&	Logical AND
	Logical OR
~	Logical NOT

Functions

sqrt(x)	sqrt(x) is the square root of the elements of X. Complex results are produced if X is not positive.
Exp(x)	The exponential of the elements of X, e to the X. For complex $z=x+i*y$, $\exp(z) = \text{EXP}(x)*(\cos(y)+i*\sin(y))$.
Log(x)	log(x) is the natural logarithm of the elements of X. Complex results are produced if X is not positive. This differs from CplxCalPro!

Log10(x)	Log10(x) is the base 10 logarithm of the elements of X. Complex results are produced if X is negative.
Floor(x)	Rounds to the nearest integers towards minus infinity. Floor(3.7)=3
ceil(x)	Rounds to the nearest integers towards infinity. Ceil(3.4)=4
round(x)	Rounds the elements of X to the nearest integers. round(3.4)=3

Complex functions

real(x)	REAL(X) is the real part of X.
imag(x)	IMAG(X) is the imaginary part of X.
conj(x)	CONJ(X) is the complex conjugate of X.
abs(x)	abs(X) is the absolute value of the elements of X. When X is complex, abs(X) is the complex modulus (magnitude) of the elements of X.
angle(x)	angle(H) returns the phase angles, in radians, of a matrix with complex elements. Example: x=[2:4]; abs(x+4j)

Matrix functions

[]	Use brackets to create arrays. a=[2 3;4 5] will create an 2-by-2 array and store the values in variable a.
()	Use parentheses to return one element of an array. Return columns is not supported yet.
'	Transpose matrix.
ones(m,n)	ones(n) is an n-by-n matrix of ones. ones(m,n) or ones([m,n]) is an m-by-n matrix of ones.
zeros(m,n)	zeros(n) is an n-by-n matrix of zeros. zeros(m,N) or zeros([m,n]) is an m-by-n matrix of zeros.
eye(m,n)	eye(N) is the N-by-N identity matrix. eye(M,N) or eye([M,N]) is an M-by-N matrix with 1's on the diagonal and zeros elsewhere.
size(x)	Size of matrix. d = size(x), for m-by-n matrix x, returns the two-element row vector d = [m, n] containing the number of rows and columns in the matrix.
rand(m,n)	rand(N) is an N-by-N matrix with random entries, chosen from a uniform distribution on the interval (0.0,1.0).
randn(m,n)	randn(N) is an N-by-N matrix with random entries, chosen from a normal distribution with mean zero, variance one and standard deviation one.
length(x)	Length of vector. length(x) returns the length of vector x. It is equivalent to max(size(x)) for non-empty arrays and 0 for empty ones.

det(x)	det(x) is the determinant of the square matrix x.
inv(x)	inv(x) is the inverse of the square matrix x.
conv(a,b)	c = conv(a, b) convolves vectors A and B. The resulting vector is length length(a)+length(b)-1. If a and b are vectors of polynomial coefficients, convolving them is equivalent to multiplying the two polynomials.
max(x)	max(x) is the largest element in X
min(x)	min(x) is the smallest element in X
sum(x)	sum(x) is a row vector with the sum over each column.
mean(x)	mean(x) is the mean value of the elements in X
std(x)	std(x) returns the standard deviation.
var(x)	var(x) returns the variance of X.

Graphical functions

plot(...)	plot(x,y) plots vector y versus vector X. plot(y) plots the columns of y versus their index. If y is complex, plot(y) is equivalent to plot(real(y),imag(y)). In all other uses of plot, the imaginary part is ignored. plot(x,y1,y2) plots vector y1 and y2 versus vector x. <i>Note: this is different than the matlab function, will be the same soon.</i>
semilogx(...)	semilogx(...) is the same as plot(...), except a logarithmic (base 10) scale is used for the x-axis.
semilogy(...)	semilogy(...) is the same as plot(...), except a logarithmic (base 10) scale is used for the y-axis.
loglog(...)	LOGLOG(...) is the same as PLOT(...), except logarithmic scales are used for both the X- and Y- axes.
title(str)	title('text') adds text at the top of the current axis.
xlabel(str)	xlabel('text') adds text below the x-axis on the current axis.
ylabel(str)	ylabel('text') adds text beside the Y-axis on the current axis.
logspace(...)	logspace(d1, d2) generates a row vector of 50 logarithmically equally spaced points between decades 10^{d1} and 10^{d2} . logspace(d1, d2, N) generates N points.
clf()	Clears the graphical screen.

Trigonometry functions

sin(x)	Returns the sine of the elements of X.
cos(x)	Returns the cosine of the elements of X.
tan(x)	Returns the tangent of the elements of X.
asin(x)	Returns the inverse sine of the elements of X.
acos(x)	Returns the inverse cosine of the elements of X.
atan(x)	Returns the inverse tangent of the elements of X.

Hyperbolic functions

sinh(x)	Returns the hyperbolic sine of the elements of X.
---------	---

cosh(x)	Returns the hyperbolic cosine of the elements of X.
tanh(x)	Returns the hyperbolic tangent of the elements of X.
asinh(x)	Returns the inverse hyperbolic sine of the elements of X.
acosh(x)	Returns the inverse hyperbolic cosine of the elements of X.
atanh(x)	Returns the inverse hyperbolic tangent of the elements of X.

Flow control

	IF statement condition. The general form of the IF statement is
if	if (condition)
	statements - Will be executed when condition is true.
else	else
	statements - Will be executed when condition is false.
	end

Special functions

clear	Removes all variables from the workspace.
pause	Wait until user taps on the screen.

MtrxCal special functions

	Set display format. t: 0-float, 1-eng, 2-sym, 3-hex, 4-bin, 5-oct, 6-pol, 7-date, 8-sexagesimal w: width of number (0-15) p: precision of number (0-15) tr: trailing zeros. (0 or 1)
fmt(t,w,p,tr)	
gclr()	Clears the graphical screen.
keybrd(file)	Read a text file and changes the keypad accordingly.
iskey('button')	Returns true when last button pressed is equal to button.
ghlin(A)	Draw horizontal lines. The scaling of the last plot is used. If the plot functions is not called the range 0 to 159 is used. Example: ghlin([20,40,120]) will draw three horizontal lines.
gvlin(A)	Draw vertical lines. The scaling of the last plot is used. If the plot functions is not called the range 0 to 159 is used. Example: gvlin([20,40,60,120]) will draw four vertical lines.

Example scripts

Net Present value

The net present value of a cash flow C_f is returned by the formula:

$$NPV(C_f, i) = \sum_t \frac{C_f_t}{(1+i)^t}$$

Suppose we are given the opportunity to make an investment of \$4000 which will provide a return of \$1000 next year and \$2000 for each following two years. What is the internal rate of return on the investment if the discount rate (the cost of borrowing \$4000) is 7%.

How do we calculate this in MtrxCal?

Net present value

Script name.

```
Cf=[-4000 1000 2000 2000];
i=0.07;
t=0:length(Cf)-1;
```

Create an array of cash flow.

Assign 7% to a variable

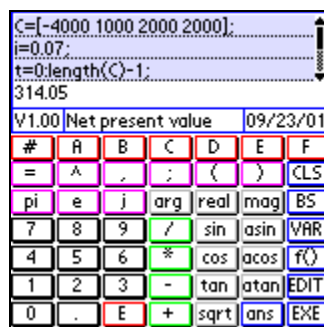
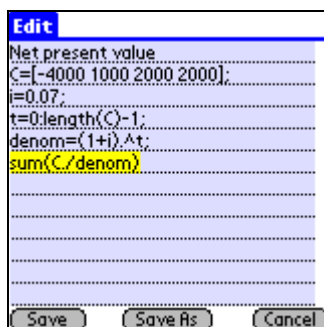
Create a range from 0 to 3, length(Cf) will return four so we subtract one.

```
Denom=(1+i).^t
sum(Cf./Denom)
```

Create an array with values for the denominator.

Here we just divide each element in array Cf by each element in the array Denom and sum the result.

Notice the period before the ^ and the /. Without the period before the / MtrxCal will perform a matrix divide and that is not what we want in this case.



Now you can change the values in the Cf array and the percent to calculate the net present value for different investments.

The text of this script can be copied, without making any changes, to matlab and will give the same answer.

Wave period

Some of you might be familiar with fourier transforms but if you are not familiar with it don't worry. This example shows how a square wave can be represented by sinusoidal waveforms.

$$f(t) = \frac{4 \sum_{h=1}^H b_{h-1} * \cos(h * w * t)}{\pi} \quad H = 5 \quad B = 1, 0, -\frac{1}{3}, 0, \frac{1}{4}$$

This can be implemented in several different ways but the main purpose of the script is to show how easy and powerful MtrxCal is. Notice how the arrays are multiplied and added in the example. The same can be done with an even smaller script in MtrxCal but would be more difficult to understand. Also notice how long it takes to calculate and plot the graph.

Wave period

t=-2.0:0.05:2;

w=2*pi;

y1=cos(w*t);

y3=-cos(3*w*t)/3;

y5=cos(5*w*t)/5;

y=4*(y1+y3+y5)/pi;

plot(t,y)

Name of script

Create a range from -2 to 2 in steps of 0.05.

Omega.

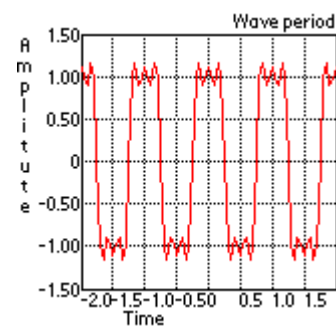
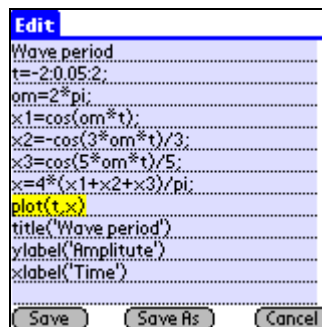
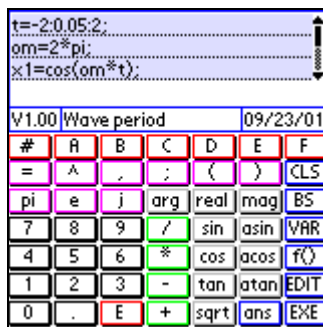
First harmonic.

Third harmonic.

Fifth harmonic.

Add the sinusoidal waveforms.

Plot y versus the time, t.



Notice how the arrays are multiplied and added in the example above.

When an array only has one element you don't have to use the .* operator to multiply.

Solve quadratic

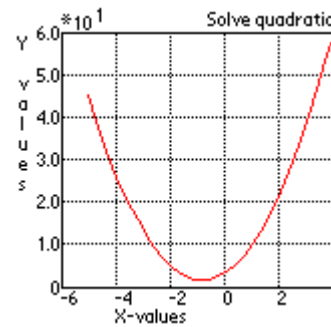
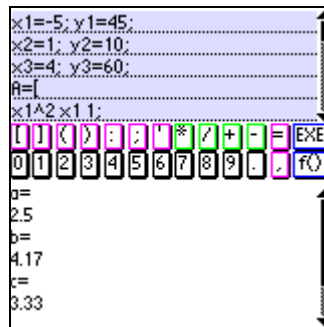
Enter the x,y coordinates of three points and this script will calculate the a,b and c for the quadratic equation:

$$f(x) = ax^2 + bx + c$$

The script will solve three linear equations for three unknowns. You can also try to fit the data point with different equations by changing the matrix A.

Solve quadratic

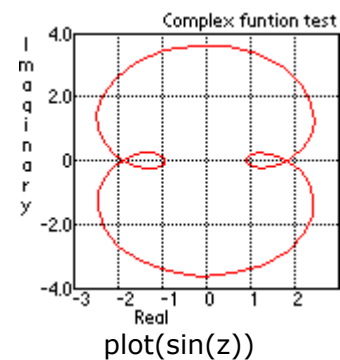
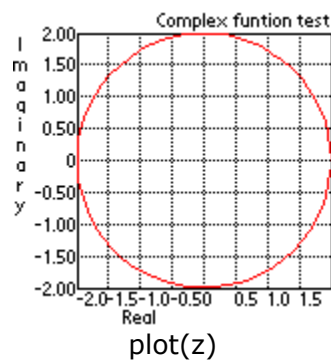
```
x1=-5; y1=45;
x2=1; y2=10;
x3=4; y3=60;
A=[
x1^2 x1 1;
x2^2 x2 1;
x3^2 x3 1;
];
B=[y1 y2 y3]';
C=A\B;
a=C(1)
b=C(2)
c=C(3)
x=x1:(x3-x1)/30:x3;
y=a*x.*x+b*x+c;
plot(x,y)
```



Sine test

Sine test

```
R=2;
Theta=0:0.1:2*pi;
z=R*exp(1j*Theta);
plot(sin(z))
```



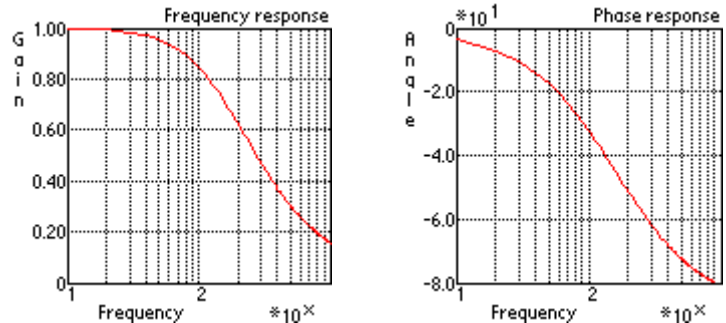
Notice the size of the script and how the plot function is used to plot an array of complex numbers. I use several little test scripts. This is just one I use to test the complex functions. This script has the advantage that all four quadrants are use and the radius can be easily changed.

RC network

The frequency response is calculated using the Laplace transform $H(s) = \frac{1}{sC} \cdot \frac{1}{R + \frac{1}{sC}} = \frac{1}{1 + sRC}$

Replacing s by $j\omega$ and taking the absolute value of $H(s)$ give the frequency response.

```
R=1000 ;C=1e-6;
Fc=1/(2*pi*R*C)
f=[10:10:1000];
w=1j*2*pi*f;
H=1./(1+w.*R*C);
semilogx(f,abs(H))
title('Frequency response')
xlabel('Frequency')
ylabel('Gain')
```



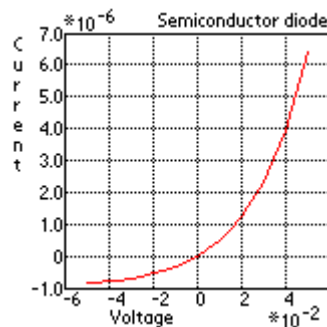
Replacing $\text{abs}(H)$ with $\text{angle}(H)$ gives the phase angle in radians. Replace $\text{abs}(H)$ with $180 \cdot \text{angle}(H) / \pi$ to get the angle in degrees. The cutoff frequency is also printed in line two and is 159.15 Hertz. The angle at the cutoff is -45 as shown in the Phase response.

Diode circuit

In a circuit containing a resistor and a diode the voltage across the diode is v . The current through the diode can be calculated from:

$i = I_0(e^{40v} - 1)$ where I_0 is a constant called the reverse saturation current.

```
IO=1e-6;
v=-0.05:0.005:0.05;
i=IO*(exp(40*v)-1);
plot(v,i)
title('Semiconductor diode')
xlabel('Voltage')
ylabel('Current')
```



Notice that entering the labels is not needed for just plotting the graph. However it does make the screen look better.

Closed contour

$$H(z) = \frac{z-1}{(z+1)(z^2+z+1)}$$

Let's take the function and create a closed contour with a radius of 1.1

Now lets see if all the poles are within the contour.

```
R=1.1;
Theta=0:0.03:2*pi;
z=R*exp(1j*Theta)
H=(z-1)./((z+1).*(z.^2+z+1));
plot(H)
title('W-plane')
xlabel('Real')
ylabel('Imaginary')
```

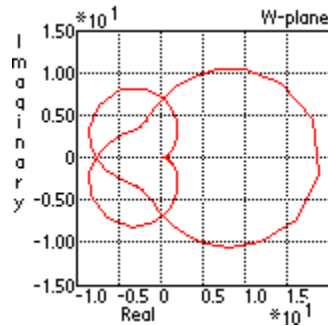


Fig. 1

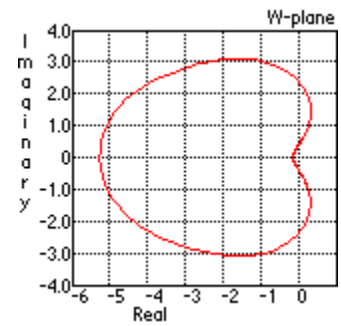


Fig. 2

The radius in fig. 1 is 1.1 and the radius in fig. 2 is 0.6

Even if the script is not totally clear to you it does show how easy it is to impress people with just a little script. We like to encourage people to write these scripts on any other calculator for the palm platform.

Legal and Disclaimers

ADACS LLC makes no warranty, either expressed or implied, including but not limited to any implied warranties of merchantability and fitness for a particular purpose, regarding any programs or book materials and makes such materials available solely on an "as-is" basis. In no event shall ADACS LLC, be liable to anyone for special, collateral, incidental, or consequential damages in connection with or arising out of the purchase or use of these materials, and the sole and exclusive liability of ADACS LLC regardless of the form of action, shall not exceed the purchase price of this application. Moreover, ADACS LLC, shall not be liable for any claim of any kind whatsoever against the use of these materials by any other party.

Contact information

ADACS LLC

Advanced Digital & Analog Consulting Services

12076 Marsh Hen Lane

Tega Cay, SC 29708

Phone: 803.833.8312

Fax: 803.547.4667

Email: support@adacs.com

Web site: www.adacs.com

Any comments or suggestions are very welcome!