

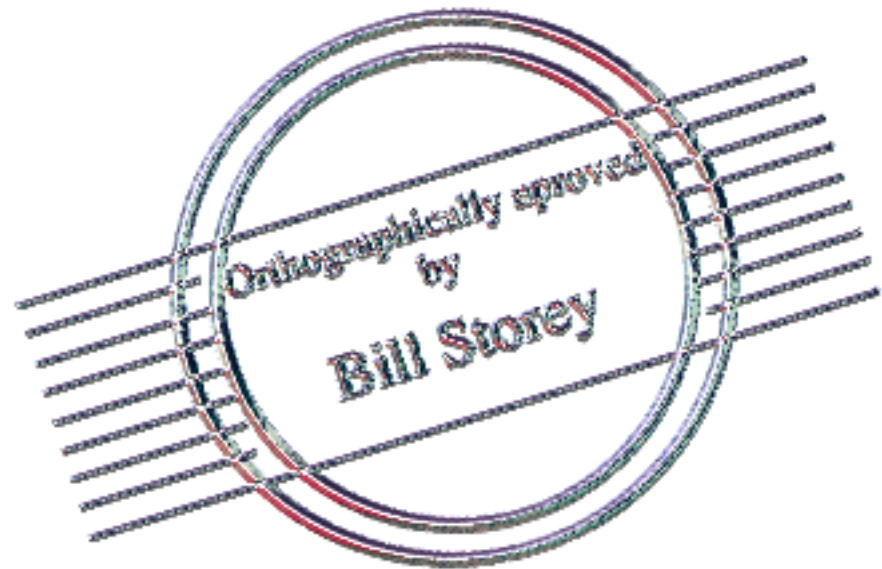
Complex Numbers with the HP49G

By Nick Karagiaouoglou

Many, many special thanks to:

Alban Schann for at least 2763 ideas about the marathon
Bill Storey for reading and correcting at least 3982 grammars
Veli-Pekka Nousiainen for the endless humour

... and everybody else who remotely helped.



Complex numbers with the HP49G - Part 1

Hi everybody!

This is the start of the second marathon, the Complex Number Marathon which is especially dedicated to VPN and JHM, who have been inspiring this group with his ideas and humour for years. (Boy, is it already such a long time? ;-))

We are going to start with the easier things and proceed to more complicated as the miles behind us get more. (Unconventional idea! ;-)) During this marathon, we are also going to use stuff from the first marathon (trigonometry), so that at the end we will have the combined strengths of both runs.

So let the games begin!

Let's configure the HP49G, so that we have all the same settings. First we set angle mode to radians. You can do that in many ways:

- I. Type RAD, press key [ENTER] RAD is the command the sets angle mode to radians.
- II. Press key [MODES], then press [ARROW DOWN] twice, then
 - A. press key [ALPHA], then key [] for entering an "R", which is the first letter of "Radians" or
 - B. Press menu key [CHOOSE], then
 1. Press key [ARROW DOWN] or key [ARROW UP] as many times as necessary, until "Radians" is selected in the popup menu , or
 2. Press key [ALPHA], then key [] for entering an "R", which is the first letter of "Radians",
 - C. then press key [ENTER] or menu key [OK], and then press key [ENTER] or menu key [OK] again.

III. Press simultaneously keys [LEFT-SHIFT]-[MODES]. This takes you to the old style configuration main menu. Press menu key [ANGLE] and then menu key [RAD]. The menu key changes to [RAD■] to indicate the current setting. Or,

IV. Enter -17,

A. Type SF and press key [ENTER], or

B. Press simultaneously keys [LEFT-SHIFT]-[MODES], press menu key [FLAG] and then menu key [SF]

SF is the command that sets a given flag, in this case flag -17, the system flag which, if set, let's the HP49G be in radians angle mode.

(Would you believe me, if I say that there are still different variations for settings? ;-)) Now a small RAD should appear at the top left of the screen, as an indicator of the setting for the radians angle mode.

Let's go on with settings. We want a rectangular coordinates system, so press simultaneously keys [LEFT-SHIFT]-[MODES] to go to the old style configuration main menu. Press menu key [ANGLE] and then menu key [RECT], which changes to [RECT■] . A small XYZ also appears at the top of the screen as an indicator of a rectangular coordinates system. Last thing, press key [NXT] to go to the second page of the menu, and then press menu key [MODES]. This takes you to the modes menu again. Press menu key [FLAG]. Enter -27 and the press menu key [SF]. When the system flag -27 is set, then the HP49G displays symbolic complex expressions in the form $\text{realPart} + \text{imaginaryPart} i$. (I'm an old fashioned guy, as you see, but you could try yourself to find the other possible ways for this settings.) These are the only settings necessary for reproducing the things that we do next.

The simplest complex thing on the HP49G, is the complex number, which the HP49G displays according to the conventions of mathematics as (a,b) with a the real and b the imaginary part. Let's enter a complex number from the command line. Press [LEFT SHIFT]

Complex numbers with the HP49G - Part 1

and then key [-]. The HP49G puts a pair of parentheses on the command line and moves the cursor between them. Now type 1.23 for the real part. We can use either blanks or one comma to separate the real from the imaginary part. If you want the comma, then press [RIGHT SHIFT] and [SPC]. But it is more convenient to only press [SPC], because this takes only one key press. (You know, using less key presses can be a life saver on the HP49G. ;-)) Now type the imaginary part 3.21 and press [ENTER]. Stack level 1 contains now: (1.23,3.21). The same way enter (1,1). Did you notice something?

Though you entered both the real and the imaginary parts as integers without decimal point, stack level one contains now (1.,1.). The HP49G converted silently both parts, to real numbers with decimal point.

Now, let's do the same from the EQW. In the EQW you don't use parentheses to enter complex numbers. Type 1.23 and the press [RIGHT SHIFT] and [SPC]. (Pressing only [SPC] doesn't work here.) The HP49G puts the real part in parentheses, puts a comma after the real part and waits for you to enter the imaginary part. Enter 3.21 and press ENTER. Now stack level 1 contains $1.23 + 3.21 i$. This is mathematically identical to (1.23,3.21) but the HP49G considers this as an algebraic which simply happens to contain the imaginary unit i . If you didn't drop the complex numbers that you entered from the command line, press SWAP and then TYPE. The HP49G returns 1. because this is the object type of complex numbers. Now press SWAP and TYPE again. The HP49G returns 9. because this is the object type of algebraic expressions. We are going to examine such complex algebraic expressions that evaluate to complex numbers in the next part, but for now we keep in mind, that the EQW enters not complex numbers but algebraic expressions, quite opposed to the EQW of the HP48 which enters complex numbers. Another difference to the HP48 is that the HP49G displays complex numbers always in one line, making it impossible to see the real and imaginary parts at once, while the HP48 uses two lines, one for the real and one for the imaginary part. Does anybody know if this is possible with the HP49G?

Now, let's see what can be done with complex numbers. There is a

menu with commands for complex numbers. Press [LEFT SHIFT] and then key [SYMB] to go to the MTH menu. Press key [NXT] and then menu key [CMPLX] to go to the complex menu. (There is another complex menu which you get by pressing [RIGHT SHIFT] and key [1] but it doesn't contain the commands R->C and C->R which we are going to use.) Enter now the complex number (2.56,3.74). (For the imaginary part type first 3.74 or at least the first digit 3 and then press key [+/-].) Press [ENTER] to make a copy of this and press the menu key [RE]. The function RE¹ returns the real part of a complex number. Press key [ARROW RIGHT] to swap stack levels 1 and 2 and press ENTER again to get another copy of the complex number. As you have guessed the function IM returns the imaginary part of the complex number, so press menu key [IM] for getting -3.74.

Swap levels 1 and 2 and press menu key [C->R] to split the complex number to its real constituents, which are of course the same with the real and imaginary part. The function OBJ-> (not in this menu) with a complex number as argument does the same like the function C->R, but it has also some differences, which we will examine later. Again, as you can imagine, the function R->C does the opposite of C->R. It takes two real numbers, the real part from stack level 2 and the imaginary part from stack level 1, and combines them to a complex number. The function R->C accepts also integers as arguments or integers and reals mixed up.

Make a copy of the complex number. Now press menu key [ABS] to get the magnitude of the complex number. Swap, make a copy and press [ARG]. The function ARG gives the angle of a complex number in the current angle mode. (Don't confuse this with the command LASTARG, which returns the arguments that were consumed by the last operation.) Swap and make a copy again. Press key [NXT] to go to the second page of the menu. Press menu key [SIGN]. This function returns the sign of a complex number. Swap and copy and press [NEG] to negate the complex number. Last thing, swap and press [CONJ] to get the complex conjugate of the complex number.

¹ "Re" is also used in the greek language. It is the short version of the word "moré", which was used in the past, for calling somebody "stupid". So did we, greek people, believe that finding someone stupid is knowing his or her real part? ;-)

Complex numbers with the HP49G - Part 1

We have seen that the functions ABS and ARG return the magnitude and the angle of a complex number, but can the HP49G represent complex numbers in the form (r, \angle) , r and \angle being the magnitude and the angle respectively? We wouldn't ask, if it weren't possible. ;-) Enter the complex numbers $(2.5, 3.5)$, $(-5.6, 2.3)$ and $(-1, -1)$. They are all represented as (a, b) , a and b being the real and the imaginary parts, because the current coordinate system is rectangular. Now, press simultaneously keys [LEFT-SHIFT]-[MODES], press menu key [ANGLE] and then menu key [CYLIN] to switch to polar coordinates. A dramatic change on the whole display takes place. First of all, all complex numbers are now displayed in the form (r, \angle) . (\angle is character number 128 of the HP49G built-in font.) The menu key changes to [CYLI■] to indicate the current setting. And on the top of the screen you see $R \angle Z$ as a further indicator for coordinates mode. You can also use spherical coordinates to do the same. Press menu key [SPHER]. The indicator at the top of the screen changes to $R \angle \angle$, and the menu key changes to [SPHE■], but the complex numbers stay the same because only the part $R \angle$, the radius and the first angle of the coordinate system are important for complex numbers. (There is however a difference between polar and spherical coordinates, which we will talk about at the vectors and matrices marathon.) Switch now back to polar coordinates.

Now, press [ARROW DOWN] to view the rest of the complex number, which hides off screen to the right. As you can see, the angle is expressed in radians, because the current angle mode is RAD. Press ENTER to put the number back to the stack, and then press menu key [DEG] to set the angle mode to degrees and make Veli-Pekka happy. Now the angle parts have changed. Press [ARROW DOWN] again and take a look at the angle of the complex number $(1, 1)$. You see that the angle is -135. and not 225., which tells you that the HP49G counts angles of complex numbers from -90° to 90° , or from -180° to 180° . Can we also enter complex numbers in the form (r, \angle) ? Yes, and independently of the current coordinates mode. Switch to rectangular coordinates and press [LEFT SHIFT], [-] to get a pair of parentheses

on the command line. Type 1.5 (the radius) and then press keys [ALPHA], [RIGHT SHIFT], [6] to put the angle character \angle after the radius. Now type 45 and press ENTER. Now the complex number $(1.5, \angle 45)$ is on stack level 1. While in polar coordinates, let's enter a complex number in the form (a, b) with a and b the real and imaginary parts respectively. Enter for example $(1, 1)$. The complex number is automatically converted to $(1.41421356237, \angle 45)$ accordingly to the current coordinates and angle mode. So if you enter a complex number as (a, b) the HP49G shows it always in the form that corresponds to the current coordinate system and angle mode.

We have used here the special character \angle to tell the HP49G, "what follows is an angle". But how do we know if there is some key sequence that produces this character, and which this key sequence is? Let's make a small excursion to the another picturesque place of the HP49G, the character catalogue. (You can go there from everywhere by pressing the keys [RIGHT SHIFT] and [CAT], so be prepared that it will be very crowded because it can be visited so easily. ;-)) You get a screen with all characters of the current font. You can move the current selection from one character to the other using the arrow keys.

When the character \angle is selected, you see $\angle \blacktriangleright 6$ at the bottom left

of the screen, which tells you that the key sequence for this character is [ALPHA], [RIGHT SHIFT], [6]. Then you see at the bottom middle of the screen the character number of the currently selected character. You can use this number as argument for the function CHR. For example if you enter 128 and press CHR, the function returns the string " \angle ". You also see three menu items over the keys F4, F5, F6. The first one, [MODIF], allows you to modify the currently selected character. The second one, [ECHO1], copies the currently selected character to where you come from, (command line, EQW etc.) and then exits the character catalogue and returns you right where you were before you started the character catalogue. The third, [ECHO] echoes the currently selected character and doesn't exit the character catalogue, allowing you to accumulate more than one special characters in the

Complex numbers with the HP49G - Part 1

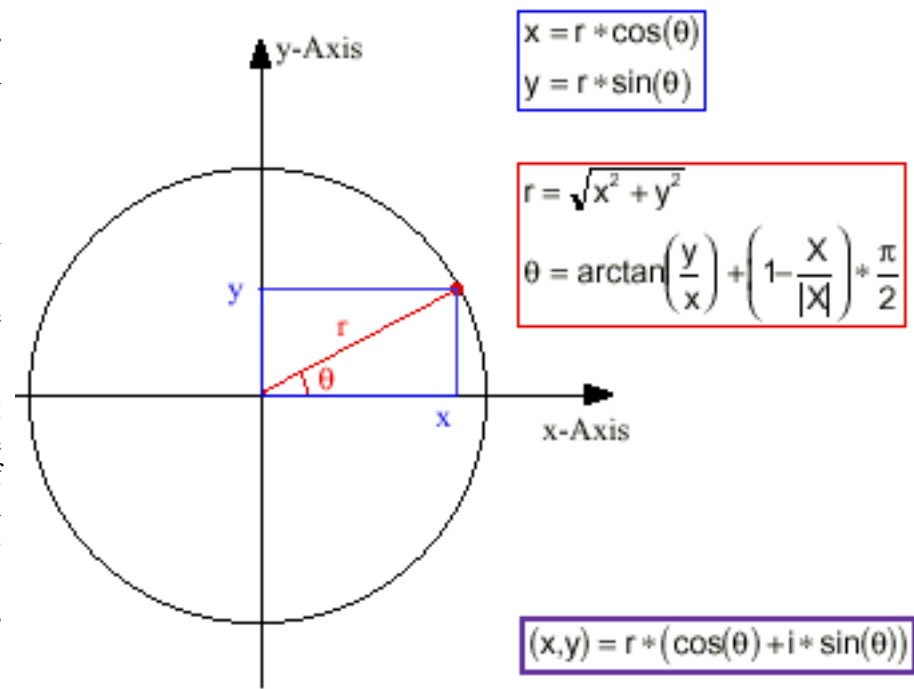
object that you want to edit, until you press ENTER or CANCL. The inhabitants of this place, the characteroids, have some strange habits. They won't allow you to turn the calculator off, while you are in their land. They put an External on the stack (behind your back) which examined with Nosy seems to be machine code for FALSE. JHM, can you tell us if this is true? Now let's return to our main path again, carrying the memories of the nice place and its strange inhabitants in our minds.

A little care is required, when entering complex numbers as (r, \angle) . As the HP49G has no other way to guess if your angle is in radians or degrees, it assumes always that the angle that you type in is measured in units of the current angle mode. That means, if you enter $(1, \angle 45)$ and the HP49G is in degrees mode, then it assumes that you mean 45 degrees. But if you enter $(15., \angle 45)$ and the HP49G is in radians mode, then it assumes that you mean 45 radians. Unfortunately you cannot specify the angle using units. You cannot enter $(1, \angle 45.^\circ)$. So you must first switch to the appropriate angle mode, or convert the angle to the current units first, and then enter the complex number.

Another thing to always be aware of, is that the command R->C always assumes that the numbers on stack level 1 and 2 are the imaginary and the real parts of a complex number, independently of current angle mode settings. While your HP49G is in polar and degrees mode enter the numbers 1 and 1 and press R->C. The command converts the two numbers to the complex number $(1, 1)$ and then displays it according to the current modes as $(1.41421356237, \angle 45)$. It doesn't assume that the 1. on stack level 2 is the magnitude and the 1. on stack level 1 is the angle in degrees, which would give the complex number $(1., \angle 1.)$.

We can think of many ways to make our lives easier considering complex number input. Assume for example that you would like to be able to enter a complex number in polar form (r, \angle) , specifying the

units for the angle, and letting the HP49G make the conversion of the angle to the current angle mode. One possible way would be to enter the magnitude, then enter the angle with units and then letting the HP49G compose the complex number. Of course we could check units and modes and act accordingly but this is cumbersome because so many combinations of angle modes and angle units have to be checked. So, let's look for an easier way. You know of course the relations between the rectangular and the polar form of a complex number:



Now, it turns out that the functions SIN and COS also accept angles with units as arguments, and return the correct result independently of angle mode. That means, that if you for example enter 1_r and press

Complex numbers with the HP49G - Part 1

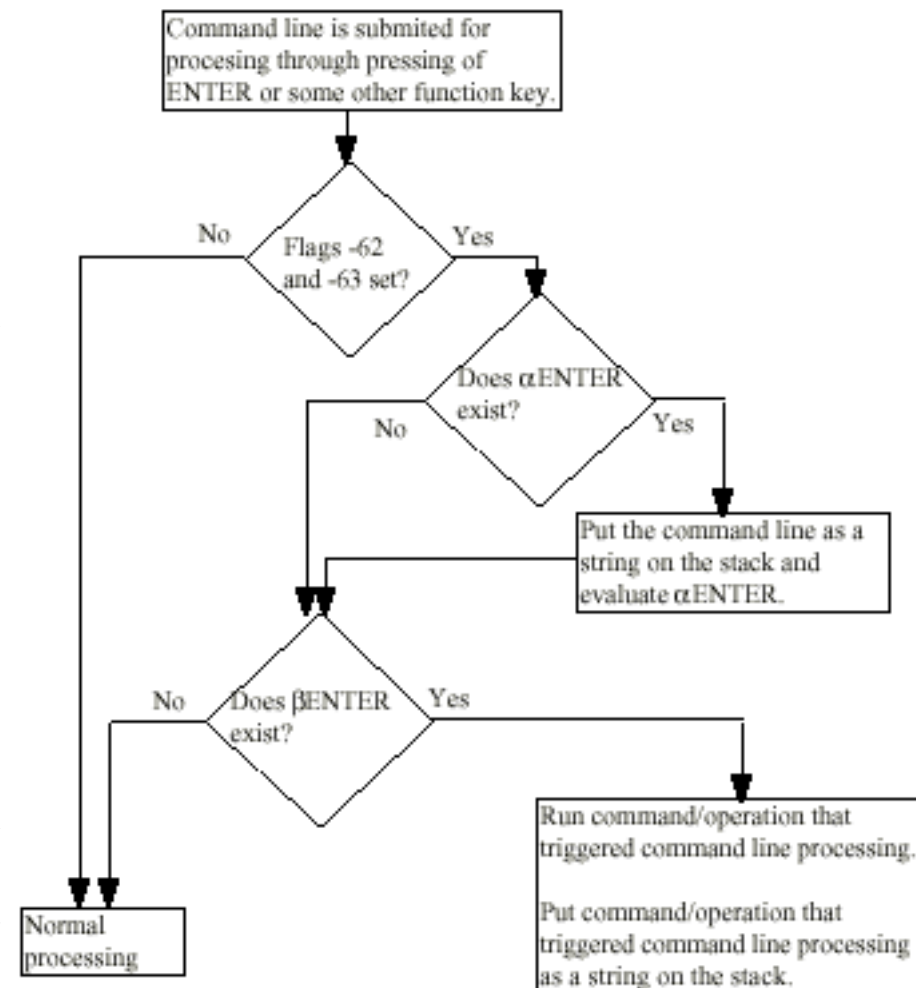
COS, then the HP49G will calculate the COS of 1_r (or approximately $57.3_^\circ$) and not the COS of 1 degree, even if it is currently in degrees angle mode. This is exactly what we need for our program.

```
<<
  DUP COS      @Make a copy of the angle
               @for later. Calculate COS.
  SWAP SIN      @Use copy of angle to
               @calculate SIN
  R->C          @Compose complex number
               @
  *            @Multiply by r
>>
```

STOre it in 'r->C'. This little program ("Programmli" in swiss german, right Thomas?), will accept a radius and an angle in any units regardless of current angle mode, and will return the correct complex number, represented according to the current mode for coordinates and angles. If you don't supply a unit with the angle, then the current unit is assumed.

If you followed faithfully everything here then your HP49G must be in polar degrees mode. If not, then configure it now. Enter 2.5 for radius and then 1.5_r for angle. Press r->C. The program returns $(2.5, \sqrt{85.9436693696})$, which shows that the input angle of 1.5_r has been converted automatically to $85.9436693696_^\circ$ according to the current angle mode degrees. But is it correct? Switch to radians mode. Now you see $(2.5, \sqrt{1.5})$ which let us breath again, because the programmli is small but does its work nice. Now, some guys out there might say, "yes, but I still can't enter the whole complex number at once as $(2.5, \sqrt{1.5})$ or as $(2.5, \sqrt{85.9436693696})$ from the command line." (Last part of the sentence comes presumably from VPN ;-). But the HP49G wouldn't be the calculator that we know and love, if it couldn't be heavily customised. Another not widely known feature of this small wonder is vectored enter capability. What is this strange thing? Well, you know that the key [ENTER] compiles

the command line, creates an object from the contents of the command line, and then acts according to the rules of the OS about objects. The vectored enter capability allows you to tell the HP49G how the command line must be interpreted and compiled/executed. First of all, vectored enter is activated if flags -62 (USER keys) and also -63 are set. (Don't set these flags yet, we need some things more.) When these flags are set, then the HP49G behaves like in the flow chart below. It



Complex numbers with the HP49G - Part 1

first checks to see if there is a variable named `ENTER` in the current path. If so, then the text of the command line is put on the stack and

`ENTER` is evaluated. Then, if a variable named `ENTER` exists, the command that caused the processing of the command line is first evaluated, then it is put on the stack as a string, and then variable `ENTER` is evaluated.

For example, imagine that you just type:

1 2

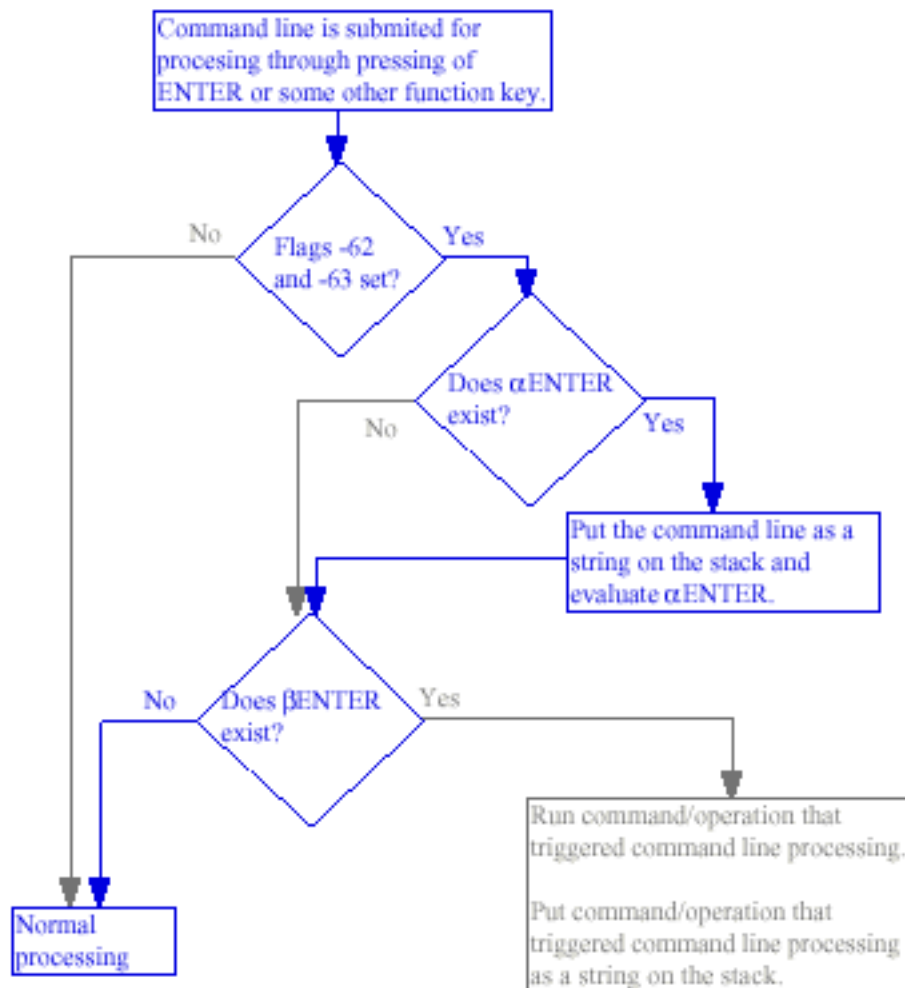
on the command line, and you press `[+]`. Then, the command line text is the string "1 2". In vectored enter mode, if variable `ENTER` exists, then this string is put on the stack. The program in `ENTER` must now do something with this string, parse it, evaluate it and so on. Then, if

`ENTER` also exists, the command `+` is evaluated, the string `"+"` is put on the stack and `ENTER` is evaluated. At this point, I must confess that I think that the evaluation of the command `+` (or any other command that triggered processing of the command line) is not a very well thought out thing. It makes it impossible to do something completely different with the command line, because if you have `ENTER`, then you must provide the arguments for the following `+` (or any other command). If the command `+` were not executed, then you could just do anything, without caring if `+` gets its arguments. In case you want the command to be executed, you could just take the command string `"+"` and do an `OBJ->` on it, after you have extracted the right arguments from the command line and put them on the stack.

Anyway, vectored enter gives us a very detailed way to control what happens upon command line completion. Don't confuse this with the user key definitions. Vectored enter is much like a unix shell. The advantage is the complete control over command line processing. The disadvantage is that we must do command line parsing ourselves, and this proves to be a tedious task, if we want to catch all possible errors in the command line. (Syntax etc.)

The next flow chart shows our strategy, for making possible to enter complex numbers with units of angle. Our path is coloured blue. We get the entered command line as a string and give it to `ENTER`. Then

`ENTER` parses the command line. It looks for complex numbers of the form `(realPart \ imaginaryPart_unit)`. If it finds such a complex



Complex numbers with the HP49G - Part 1

number, it converts to a valid complex number, using the same procedure like the "Programmli". It then replaces in the command line string the invalid complex number (the one with units), with the valid complex number, and uses OBJ-> to make an object (or sequence of objects) that will be used by the command that triggered command line processing (be it ENTER, + or anything.)

Of course the difficult thing is to parse the command line. How can we do that? The user might enter just about anything. Nobody can guarantee that the user enters a command line that only contains one complex number with units of angle. What if the user doesn't use " \ " but "," as separator between the radius and the angle? And what is to do, when the user enters an erroneous command line? You see, parsing is kind of an art. But because this is a complex numbers marathon and not a "Write your own shell marathon", we make a compromise. Our

ENTER should be able to extract and convert exactly one invalid complex number that contains " \ " as the separator of radius and angle. In case of other error, we return the erroneous command line string on the stack and do nothing more.

Here it is:

```
<<
  DUPDUP "(" POS OVER ")" POS SUB
  -> cmdstr cmplx
  <<
    IF
      cmplx " \ " POS
    THEN
      cmplx 2 OVER " \ " POS 1 - SUB
      cmplx DUP " \ " POS 1 + OVER SIZE 1 - SUB
      -> r q
      <<
        IF
          r SIZE NOT q SIZE NOT OR
        THEN
          cmdstr OBJ->
```

```
ELSE
  IFERR
    r OBJ->
  THEN
    DROP cmdstr
  ELSE
    IFERR
      q OBJ->
    THEN
      DROP2 cmdstr
    ELSE
      IF
        OVER DUP TYPE NOT SWAP
        TYPE 28. == OR
        OVER TYPE 13. == AND
      THEN
        DUP COS SWAP SIN R->C *
        ->NUM ->STR
        cmdstr cmplx ROT SREPL
        DROP OBJ->
      ELSE
        DROP2 cmdstr OBJ->
      END
    END
  END
END
END
END
>>
ELSE
  cmdstr OBJ->
END
>>
```

STOre this in ENTER. The character " " is key sequence [ALPHA],[RIGHT SHIFT],[F1]. Now enter -63 SF and -62 SF. A small string "USR" appears now at the top of the screen at the right of the current path. Switch to radians angle mode and polar coordinates.

Complex numbers with the HP49G - Part 1

Let's check if it does its work. First we check if it works right for inputs that don't contain polar complex numbers with units for the angle. Type 1 2 and press [+]. Nothing unusual happens. You just get the result 3. Press key [LEFT SHIFT] and [SPC] to get a on the stack. Press key [SIN]. Right result, you get a zero. Type (3,4) and press [ENTER]. You get (5., √.927295218002). Type (3., √3.14) and press keys [RIGHT SHIFT], [÷] to get the angle 3.14 on the stack. Seems to work all right for "normal cases". And now the long awaited moment of the truth. Type (3 √90_°). The character "_" is [RIGHT SHIFT],[.]. The character "°" for degrees is [ALPHA], and then simultaneously keys [RIGHT SHIFT] and [6]. Now press [ENTER] and enjoy! The HP49G has faithfully followed the rules and converted our input to (3., √1.57079632679), using the current angle mode. Switch to degrees mode. The complex number in stack level 1 is now (3., √89.9999999997), that is an angle of approximately 90°. (You don't get exactly 90_° because 1.57079632679 radians is not exactly $\frac{\pi}{2}$. Type (18 √2.0944_r).

The small "r" is for units of radians. And 2.0944_r is approximately 120_°. Press [ENTER]. You get the complex number (18., √120.000280612).

Of course our "shell" will error out, if you enter something like for example (3,#5), because it doesn't handle such syntax errors. It will also error out if you type (3.2 √47_°) (1.3 √123.3_°) and press [ENTER] to enter two complex numbers in polar form with units of angle at once. But you get the idea. You can change ENTER so that it does more checking if you like. For example the program could check in a loop all occurrences of such complex numbers and convert them all to syntactically valid complex numbers. The limit of possibilities is in your fantasy.

A bit more on angles. Sometimes you are in radians angle mode (except if you are Veli-Pekka ;-), but you would like to see an angle

in degrees. For example, using the command ARG you find the angle of a complex number. If you are in radians mode, then you get an angle of, say, 2.1345 radians. You don't have to re-enter the complex number, switch mode and, use ARG again. There is the built-in command R->D, which converts a unitless number, that represents an angle in radians, to degrees. The opposite does D->R, it converts a unitless number from degrees to radians. Unfortunately, there are no commands for conversion to and from grad, but it is easy to program that.

Another way to convert between angles, is to use units. Suppose for example that you have 35 on stack level one, representing an angle of 35°, and you want to know the angle in grad. Press [RIGHT SHIFT] and then [6] to go to the units menu. Press key [NXT] twice and then the menu key [ANGL]. You get a menu with angle units. While 35 is on stack level 1, press the menu key [°]. If a number is on stack level one, and you press a unit menu key, then the unit is multiplied with the number. Now your HP49G knows that the number 35 is meant as degrees. Press [LEFT SHIFT] and then the menu key [grad]. The number on stack level 1 is now expressed in grads, 38.888888888_grad. Pressing [LEFT SHIFT] and then a unit menu key while a number with a unit is on the stack, converts the number to the unit of the menu key (if the units are compatible). If you press [RIGHT SHIFT] and then a unit menu key while a number with or without a unit is on the stack, you divide the number with the unit of the menu key. If you want to get the number part of, say 38.888888888_grad, then you use command UVAL. The command CONVERT is the programmable analogon of converting through pressing [LEFT SHIFT] and then a unit menu key. It takes the number with a unit that you want to convert from stack level 2 and the new unit from stack level 1, and returns the number in units of stack level 1. For example, with 38.888888888_grad on stack level 2 and 1_r on stack level 1, CONVERT returns .610865238198_r. There is also a programming help for this. While you edit a program and you are at a unit menu, pressing [LEFT SHIFT] and then a unit menu key, puts '1_unit' CONVERT in the program. ('1_unit' can be for example '1_°', '1_r' or any other unit. And by the way, real

Complex numbers with the HP49G - Part 1

numbers with units, like 38.8888888888_grad are a distinct type of object on the HP49G. Their object type is 13. This gives us the possibility to make programs that run differently if the user inputs a unitless number or a unit.

Still more on angles. Sometimes you want to convert decimal degrees to degrees, minutes and seconds. For example, let's say you have 120.6578999 degrees and you want to know the angle in degrees, minutes, seconds. (The number has to be unitless.) The built-in function ->HMS does this conversion and returns 120.392843964 in this case. The result means 120°39'28" and the rest of the digits is parts of a second. The opposite conversion is done with the built-in command HMS->. (These two commands can also be used for the conversions decimal hours to/from hours, minutes, seconds.)

Let's see now, what functions the HP49G provides for complex numbers. Almost any function that you can use for real numbers, you can also use for complex numbers. You can use the normal +, -, *, / for addition, subtraction, multiplication and division. You can also use SIN, COS, EXP etc. just like you do with real numbers. There is no such thing like for example CSIN for finding the sine of a complex number. The HP49G uses the same function name for real and complex numbers (where appropriate). Each UserRPL function does a lot of arguments checking before it starts calculating. It first checks if there are enough arguments. If not it errors with "Too few arguments". If there are enough arguments, then it checks what argument type the arguments have. If the arguments have types that the function doesn't deal with, then it shows the error "Bad argument type". If the arguments have valid types, then it goes on processing.

A little care is needed when you calculate roots or raise complex numbers to broken exponents. The HP49G behaves different, depending on the argument types. But first let's take a look at the mathematics hidden behind the key $\sqrt[n]{Y}$ and Y^X for complex numbers. For any complex number :

$$(a,b) = r (\cos(\theta) + i \sin(\theta)) \quad (1)$$

where r is the radius and θ is the angle of the complex number. If we raise (1) to the power n , we have:

$$(a,b)^n = r^n (\cos(\theta) + i \sin(\theta))^n \quad (2)$$

Now, from the trigonometry marathon we know that:

$$\cos(\theta) + i \sin(\theta) = e^{i\theta} \quad (3)$$

We substitute $e^{i\theta}$ for $\cos(\theta) + i \sin(\theta)$ in (2) and so we obtain:

$$(a,b)^n = r^n e^{i n \theta} \quad (4)$$

Now we substitute $\cos(n\theta) + i \sin(n\theta)$ for $e^{i n \theta}$ in (4) and we find:

$$(a,b)^n = r^n (\cos(n\theta) + i \sin(n\theta)) \quad (5)$$

The last relation (5) is known as Moivre formula, and is valid for any power n . Now, if the power n is of the form $\frac{1}{m}$, then we are actually calculating $\sqrt[m]{(a,b)}$, the m^{th} root of the complex number (a,b) . The m^{th} root of the complex number $(a,b) = r (\cos(\theta) + i \sin(\theta))$ is a complex number $(x,y) = (\cos(\theta) + i \sin(\theta))$ which satisfies:

$$(x,y)^m = \sqrt[m]{r} (\cos(\theta) + i \sin(\theta))^m = r (\cos(m\theta) + i \sin(m\theta))$$

From this we derive for the radius of (x,y) :

$$r = \sqrt[m]{r} \quad (6)$$

Complex numbers with the HP49G - Part 1

For the angle of (x,y) we derive:

$$= \frac{n}{m} + \frac{n}{m} \quad (\text{remember TRISOL?}) \quad (7)$$

In (7) the integer values for n are $0, 1, \dots, m-1$. For $n = m$ we have again the same root as for $n = 0$. That means that there are two square roots with $n = 0, 1$, three cubic roots with $n = 0, 1, 2$, etc. of (a,b) .

For example, the three cubic roots of $(a,b) = r (\cos(\theta) + i \sin(\theta))$ are:

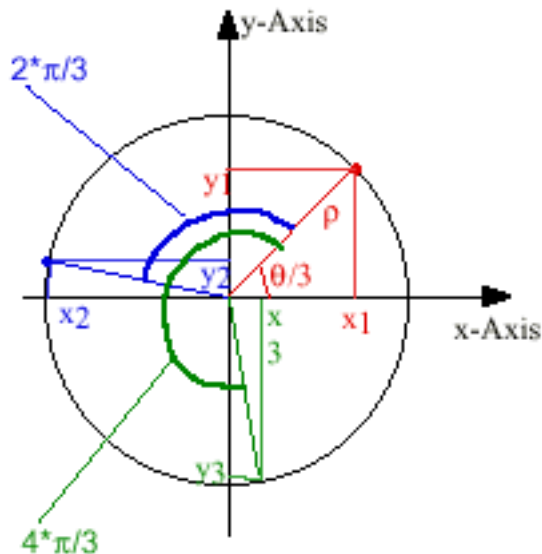
$$\sqrt[3]{r} \cos \frac{\theta}{3} + i \sin \frac{\theta}{3} \quad (i)$$

$$\sqrt[3]{r} \cos \left(\frac{\theta}{3} + \frac{2\pi}{3} \right) + i \sin \left(\frac{\theta}{3} + \frac{2\pi}{3} \right) \quad (ii)$$

$$\sqrt[3]{r} \cos \left(\frac{\theta}{3} + \frac{4\pi}{3} \right) + i \sin \left(\frac{\theta}{3} + \frac{4\pi}{3} \right) \quad (iii)$$

These three cubic roots sit on the angles of a regular triangle, if we plot them on the complex plane, with the real and imaginary part as x - and y - components respectively.

Similarly the four 4th roots sit on the angles of a regular square, the five 5th roots sit on the angles of a regular pentagon, and in general the m mth roots sit on the angles of a regular m -gon. And speaking about pentagon, as you



can see, even the deafness of the United States relies on the validity of mathematics of complex numbers. ;-)

Now the question is, which of the m^{th} roots should the HP49G give when we ask it to take m^{th} root of a complex number? It turns out the HP49G answers differently according to the types of the arguments that are given for example to the function $\sqrt[m]{Y}$. Let's make a table of results for all possible combinations of arguments Y and X for the function $\sqrt[m]{Y}$, arguments that are evaluable to complex or real numbers. Though symbolic expressions that evaluate to complex numbers are examined in the next part of this marathon, we include them in our table now, so that we have all possibilities in one place. The table contains strictly the direct results of the function. If some of these results can be evaluated to numbers using other commands, then a description of this evaluation is put in the paragraphs following the table. The table is divided into regions as follows:

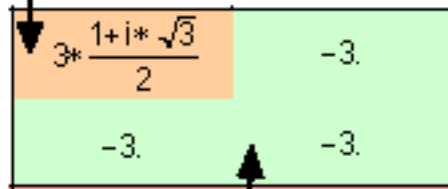
- 1) The regions of combinations of arguments X and Y that have the same numeric values are enclosed in solid black bounds. For example the region of combinations of X and Y arguments that have the same numeric values, is shown below.

		Numerical value of y is -27. Numerical value of x is 3.	
		B	C
X	$\sqrt[m]{Y}$	-27	-27.
2	3	$3 \cdot \frac{1+i\sqrt{3}}{2}$	-3.
3	3.	-3.	-3.
		XROOT Error:	XROOT Error:

Complex numbers with the HP49G - Part 1

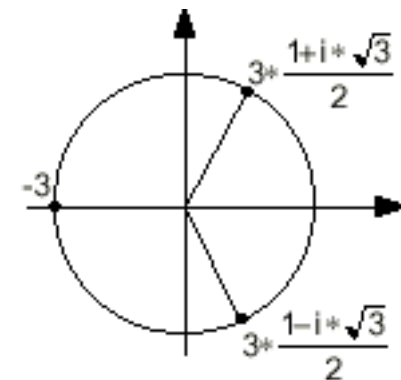
- 2) Inside such regions, all results that directly or after some manipulations return the same numeric values, are coloured with the same colour. For example the region of the results from a numeric values of $Y=27$. and $X=3$. are divided into an orange sub-region, that contains all results with numeric value (1.5,2.59807621136) and a green sub-region that contains all results with numeric value of -3.

Orange for all results that evaluate to (1.5,2.59807621136)



Green for all results that evaluate to -3.

same? Why is the cubic root of -27 in B2 not -3? As said earlier, there are three cubic roots of -27. If we plot them, we see that the returned root is the one, that comes first if we start at the x-axis and follow the circle counter-clockwise. This is the behaviour of the function $\sqrt[3]{Y}$, when we give it integer arguments. It always finds the "first" root, "first" meaning the one that we meet if we start at the x-axis and go counter-clockwise.



If any of the real-valued arguments is of type 0. (real number), then the *principal value* is returned. (Green coloured sub-region of the table.)

- 3) Over the table we see the settings under which the results are obtained. If the command or manipulations of its results also change modes, then this is described in the paragraphs following the table.

Settings

RAD XYZ C =			
	A		
1	x	$\sqrt[3]{y}$	y

Now let's examine the table. We see that the whole row 4 and column D are red, and red implies bad things. (Why is danger always red?) That means, that complex numbers as arguments for $\sqrt[3]{Y}$ are not allowed at all. The region B2-C3 contains results for arguments that evaluate to numeric values of $Y=27$. and $X=3$. But why are the results not the

The next paragraphs describe also how to get numerical values of the results shown in the table, when the function $\sqrt[3]{Y}$ returns a symbolic expression.

RAD XYZ C =					
	A	B	C	D	E
1	$\sqrt[3]{x}$	$\sqrt[3]{y}$	-27	-27.	(2.,3.)
2	3	$3 * \frac{1+i*\sqrt{3}}{2}$	-3.	XROOT Error: Bad Argument Type	$\sqrt[3]{2+3*i}$
3	3.	-3.	-3.	XROOT Error: Bad Argument Type	$\sqrt[3]{2+3*i}$
4	(1,2.)	XROOT Error: Bad Argument Type	XROOT Error: Bad Argument Type	XROOT Error: Bad Argument Type	XROOT Error: Bad Argument Type
5	$1+2*i$	$1+2*\sqrt{-27}$	$1+2*\sqrt{-27}$	XROOT Error: Bad Argument Type	$1+2*\sqrt{2+3*i}$

Complex numbers with the HP49G - Part 1

Cell B5: ->NUM on $^{1+2}\sqrt{-27}$ results in:
->NUM Error:
Bad Argument Type.
Press TSIMP to convert $^{1+2}\sqrt{-27}$ to:

$$e^{\frac{(3-6i)\text{LN}(3)+(2+i)}{5}}$$

Now press ->NUM to obtain:

$$(5.2385418165-4.32370051244i)$$

Cell C5: ->NUM on $^{1+2}\sqrt{-27}$. results in:
->NUM Error:
Bad Argument Type.
Press TSIMP to convert $^{1+2}\sqrt{-27}$. to:
 $e^0 e^{(2+.4i)\text{LN}(27)}$. This also switches the HP49G to approximate mode. You can press now ->NUM to obtain:

$$(5.2385418165-4.32370051244i)$$

Cell E2: ->NUM on $\sqrt[3]{2+3i}$ results in:
->NUM Error:
Bad Argument Type.
Press TSIMP to convert $\sqrt[3]{2+3i}$ to:

$$\sqrt[6]{13} e^{\frac{i \text{ATAN} \frac{3}{2}}{3}}$$

Now press ->NUM to obtain

$$(1.41585661836-.493403534104i)$$

Cell E3: ->NUM on $\sqrt[3]{2+3i}$ results in:
->NUM Error:
Bad Argument Type.

Press TSIMP to convert $\sqrt[3]{2+3i}$ to:

$$e^0 e^{(0.3333333333333333i) \text{ATAN} \frac{3}{2} + .1666666666666667 \text{LN}(13)}$$

This also switches the HP49G to approximate mode.

You can press now ->NUM to obtain:

$$(1.41585661836-.493403534104i)$$

Cell E5: ->NUM on $^{1+2}\sqrt{2+3i}$ results in:
->NUM Error:
Bad Argument Type.

Press TSIMP to convert $^{1+2}\sqrt{2+3i}$ to:

$$e^{\frac{(4+2i) \text{ATAN} \frac{3}{2} + (1-2i) \text{LN}(13)}{10}}$$

Press ->NUM to obtain:

$$(1.8197326743-.595841009675i)$$

Complex numbers with the HP49G - Part 1

Let's make the same table for $\sqrt[n]{Y}$ but for polar coordinates mode.

RAD R↔Z C =					
	A	B	C	D	E
1	$x \sqrt[n]{y}$	-27	-27.	(2,3.)	2+3*i
2	3	$3 * e^{\frac{\pi * i}{3}}$	-3.	XROOT Error: Bad Argument Type	$\sqrt[3]{2+3*i}$
3	3.	-3.	-3.	XROOT Error: Bad Argument Type	$\sqrt[3]{2+3*i}$
4	(1,2.)	XROOT Error: Bad Argument Type	XROOT Error: Bad Argument Type	XROOT Error: Bad Argument Type	XROOT Error: Bad Argument Type
5	1+2*i	$1+2*i \sqrt[n]{-27}$	$1+2*i \sqrt[n]{-27.}$	XROOT Error: Bad Argument Type	$1+2*i \sqrt[n]{2+3*i}$

In polar coordinates mode, the HP49 returned the "first" solution written in polar form. It returns also the i of the exponent written in its polar form, $e^{\frac{i}{2}}$. EXPAND i in polar mode and you also get $e^{\frac{i}{2}}$.

Cell B5: ->NUM on $1+2\sqrt{-27}$ results in:
->NUM Error: Bad Argument Type.

Press TSIMP to convert $1+2\sqrt{-27}$ to:

$$e^{\frac{3\sqrt{5} e^{i \frac{2 \text{ATAN} \frac{1}{2}}}{2} - \text{LN}(3) + \sqrt{5} e^{i 2 \text{ATAN} \frac{1}{2}}}{5}}$$

This is the (complicated) polar form of the cell B5 in the previous table. Now press->NUM to obtain:

$$(6.79240064216, 4.32370051242).$$

Cell B2: You can use ->NUM on $3 e^{\frac{i}{3}}$ to get the numerical value: (2.99999999999, 1.0471975512).

The result $3 e^{\frac{i}{3}}$ seems to be erroneous at first glance. But if you press POWEXPAND, you get exactly the same as in cell B2 of the previous table. The expression $e^{\frac{i}{2}}$ in the exponent is equal to i so that the returned result is equal to $3 e^{\frac{i}{3}}$.

(Explanation: $e^{\frac{i}{2}} = \cos \frac{1}{2} + i \sin \frac{1}{2} = 0 + i 1 = i$)

To show that $e^{\frac{i}{2}} = i$ on the HP49G, enter $e^{\frac{i}{2}}$ and press SINCOS.)

Cell C5: ->NUM on $1+2\sqrt{-27}$ results in:
->NUM Error:
Bad Argument Type.

Press TSIMP to convert $1+2\sqrt{-27}$ to:
 $e^0 e^{(.4472135995, -1.10714871779) \text{LN}(-27.)}$

This also switches the HP49G to approximate mode.

Complex numbers with the HP49G - Part 1

You can press now ->NUM to obtain:

(6.79240064216 \ 4.32370051242)

Cell E2:

->NUM on $\sqrt[3]{2+3i}$ results in:
->NUM Error:
Bad Argument Type.

Press TSIMP to convert $\sqrt[3]{2+3i}$ to:

$$\sqrt[6]{13} e^{\frac{i}{3} \operatorname{ATAN} \frac{3}{2}}$$

Now press ->NUM to obtain:

(1.53340623701 \ .32759790775)

Cell E3:

->NUM on $\sqrt[3]{2+3i}$ results in:
->NUM Error:
Bad Argument Type

Press TSIMP to convert $\sqrt[3]{2+3i}$ to:

$$e^0 e^{(.333333333333, 1.57079632679) \operatorname{ATAN} \frac{3}{2} + .166666666667 \operatorname{LN}(13)}$$

This also switches the HP49G to approximate mode.
You can press now ->NUM to obtain:

(1.53340623701 \ .32759790775)

Cell E5:

->NUM on $\sqrt[1+2i]{2+3i}$ results in:
->NUM Error:
Bad Argument Type.

Press TSIMP to convert $\sqrt[1+2i]{2+3i}$ to:

$$e^{\frac{2\sqrt{5} e^{i \operatorname{ATAN} \frac{1}{2}} \operatorname{ATAN} \frac{3}{2} + \sqrt{5} e^{i \frac{2 \operatorname{ATAN} \frac{1}{2}}}{3} \operatorname{LN}(13)}{10}}$$

Press ->NUM to obtain:

(1.9147985572 \ .31643112684)

You can make such tables yourself for other combinations of modes. I didn't include such cases like the cubic root of 27, because the result 3 can be also understood through the rule "first root in counter clockwise direction". But of course if they are helpful for you, you can include them in your table.

We continue with the next table (next page). As you of course know, instead of calculating $\sqrt[XY]{Y}$ we can also do $X^{\frac{1}{Y}}$.

Complex numbers with the HP49G - Part 1

RAD XYZ C =					
	A	B	C	D	E
1	y^x	-27	-27.	(2.,3.)	$2+3*i$
2	$\frac{1}{3}$	$3*\frac{1+i*\sqrt{3}}{2}$	-3.	\wedge Error: Bad Argument Type	$\sqrt[3]{2+3*i}$
3	.3333333333	(15.,2.59807621135)	(15.,2.59807621135)	(1.45185661836 .493403534103)	$(2+3*i)^{.3333333333}$
4	$\frac{1}{(1,2.)}$	(5.23854181649 -4.3237005124)	(5.23854181649 -4.3237005124)	(1.81997326743 -.595841009874)	$(2+3*i)^{(2,-4)}$
5	$\frac{1}{1+2*i}$	$(-27)^{\frac{1}{1+2*i}}$	$(-27.)^{\frac{1}{1+2*i}}$	$(2.,3.)^{\frac{1}{1+2*i}}$	$(2+3*i)^{\frac{1}{1+2*i}}$

Cell E2: ->NUM on $\sqrt[3]{2+3i}$ results in:
->NUM Error:
Bad Argument Type.

Press TSIMP to convert $\sqrt[3]{2+3i}$ to:
 $\sqrt[6]{13} e^{\frac{i \text{ATAN } \frac{3}{2}}{3}}$.

Now press ->NUM to obtain:
 $(1.41585661836-.493403534104)$.

Cell E3: ->NUM on $(2+3i)^{.3333333333}$ gives:
 $(1.41585661836-.493403534104)$

Cell B2: You can use ->NUM on $3 \frac{1+i\sqrt{3}}{2}$ to get the numerical value (15.,2.59807621135).

Cells B3,C3: The result (15.,2.59807621135) is the same as in B2.
 $Y^{\frac{1}{x}}$ doesn't behave like $\sqrt[x]{Y}$! Compare these results with the first table.

Cells B5,C5: ->NUM on $(-27)^{\frac{1}{1+2i}}$ and $(-27.)^{\frac{1}{1+2i}}$ gives:
(5.23854181649-4.3237005124).

Cell D2: This is the only case that results in error.

Cells D5,E4,E5: ->NUM on $(2.,3.)^{\frac{1}{1+2i}}$, $(2.,3.)^{(2,-4)}$, $(2+3i)^{\frac{1}{1+2i}}$, gives (1.41585661836-.493403534104).

One method to get all n^{th} roots of a complex number, is to solve the equation $w^n = z$ for w . For example let's find all cubic roots of -27. We enter $w^3 = -27$ and w . Then we can use one of the solving commands of the HP49G. I choose ZEROS in case I just want to see the solutions. ZEROS returns the solutions list:

$$-3 \frac{1-i\sqrt{3}}{2} \quad -3 \frac{1+i\sqrt{3}}{2} \quad -3$$

If we would like to let a program find all real solutions, then we could use ZEROS to get a list of solutions and then check the angle of the arguments. The command ARG returns 0 or π for real positive or negative arguments. When we have the solution list, we can use for example:

Complex numbers with the HP49G - Part 1

```

1
<<
  DUP ->NUM ARG
  IF
    DUP          @is angle not 0
    ->NUM -      @and not
    AND
  THEN
    DROP
  END
>>
DOSUBS

```

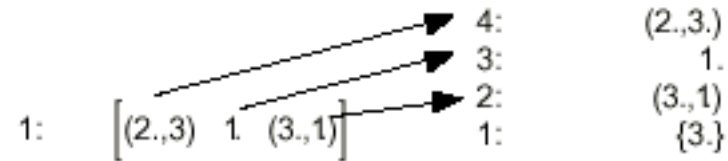
to get only the real roots. (Program DROPCMPLX)

And there is still another (beautiful) method, which we will examine at the vectors and matrices marathon.

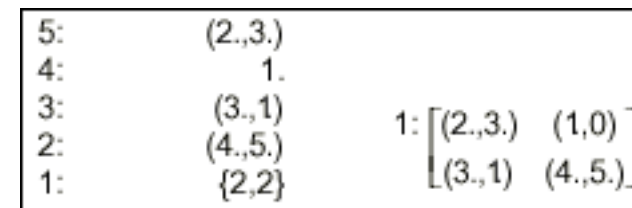
Speaking about vectors and matrices, let's take a short look at the complex numbers as elements of vectors and matrices. A numeric vector can contain real and/or complex elements. If the vector contains only real numbers as elements, then it is of type 3. If it contains only complex numbers then it is of type 4. If it is mixed, then it gets complicated. Enter the vector $[1. \ (2,3)]$. (1. with decimal point.) In the command line you see one real number and one complex number inside the vector. But if you press [ENTER], the real number 1. is converted to the complex number $(1.,0.)$. The vector contains now only complex numbers and is of type 4. But if you enter $[1. \ (2,3)]$ (1 is now integer without decimal point), and press [ENTER], then the vector doesn't change. Because it contains one integer, it is of type 29, the new array type on the HP49G. So watch out what you enter, or some commands for real/complex vectors may/will not work. The complex numbers are displayed according to the current mode, even inside such "numerical vectors".

The construction of such vectors is easy, as there are many commands for this. To make an n-dimensional vector out of n real and/or complex

numbers that are on the stack, you enter n (the number of elements) and press ->ARRAY. If the elements are in a list like for example in $\{(2.,3.) \ 1. \ (3.,1.)\}$ you can also use AXL to turn the list to an array, but the constructed vector $[(2.,3.) \ 1. \ (3.,1.)]$ is no more of type 3 or 4, though it contains only numbers. You must apply ->NUM with the vector $[(2.,3.) \ 1. \ (3.,1.)]$ as argument, to turn it to a type 4 vector. The vector looks exactly the same after ->NUM, but it has changed type. To explode a vector to its elements, use OBJ->, or ARRAY->, or V->. OBJ-> puts the elements on the stack, starting at the left of the vector. It also puts the number of elements in a list at stack level 1. ARRAY-> does the same like OBJ->. It is retained for compatibility with the HP28. V-> puts only the elements and no element count on the stack.



Numeric complex matrices can be also made using ->ARRAY. You just put all elements on the stack and then put a list with two numbers, the number of rows and the number of columns, on the stack. For example:



OBJ->, ARRAY-> do the opposite, while V-> doesn't work for matrices.

Complex numbers with the HP49G - Part 1

The commands R->C and C->R work also with vectors and matrices. If you put a matrix/vector with the real parts as elements on stack level 2 and a matrix/vector with the imaginary parts as elements on stack level 1 then you can compose the complex vector with R->C. Both matrices/vectors must have the same dimension. For example:

```

2: [2. 3. 4.]
1: [4. 8. 12.]  1: [(2.,4.) (3.,8.) (4.,12.)]

2:      1 2
      3 4
1:      5 6
      7 8
      → 1: (1.,5.) (2.,6.)
           (3.,7.) (4.,8.)

```

C->R does the opposite, it creates two matrices/vectors containing the real and imaginary parts of a given matrix/or vector. Note the difference between OBJ-> and C->R for a complex matrix/vector. While both commands do the same with complex numbers, they act differently when fed with matrices/vectors.

And now for a strange phenomenon! If you put two complex numbers, like for example (2.,3.) and (3.,4.) on the stack and use R->C, then you get the error "Bad argument type". Good! R->C expects real numbers and so errors out. But put two complex matrices/vectors of the same dimensions on the stack and press R->C. What is going on here? It seems that the HP49G determined that it has to build a new 2-

```

2: [(1,2.) (3.,4.)]
1: [(5.,6.) (7.,8.)]  1: [(1,5.) (2.,6.)]

```

dimensional vector, and then it started building it using real and imaginary parts of both vectors one after the other, until the new vector had 2 elements. The group says it is a bug, JYA declares that to a feature, and Nick thinks in silent mode, what could this back door be used for?

We will see how such "features" can be used for reasonable purposes in the "sets and mappings marathon", somewhere in the far future.

The other functions for complex numbers that can be also used for matrices/vectors with complex elements are:

RE: Returns an array with the real parts of each element.
IM: Returns an array with the imaginary parts of each element.
NEG: Negates each array element.
CONJ: Conjugates each matrix element.

ABS works but it doesn't return the magnitude of each element, but the Euclidean norm of the array. If you want an array with the magnitude of each element, put the array on the stack, enter << ABS >> and then press MAP.

ARG doesn't work for arrays. If you want an array with the angles of each element, put the array on the stack, enter << ARG >> and then press MAP.

MAP and the many other commands for construction and explosion of arrays will be examined in detail later in this marathon and in the "vectors and matrices marathon".

Talking about bugs, errh, features ;-), here you are another one. Enter 'n', 1+2 i 3+4 i 'n'. Press [] to calculate the sum. You get the error "Bad argument type", which is OK. But now go to the EQW,

enter n and press [ENTER] to go to the stack with this algebraic

object. Now press EVAL or EXPAND and you get the result 13 i. What the heck happened? Well, you can understand this better if you

enter n and press EXPAND. The result is $-\frac{n_0^2 - n_0 - (n_1^2 + n_1)}{2}$.

Now enter $n_0 = (1.,2.)$, press SUBST to substitute the value for n_0 , enter $n_1 = (3.,4.)$, press SUBST again to substitute the value for n_1 ,

Complex numbers with the HP49G - Part 1

press EXPAND and you get 13 i. The HP49G has found how to calculate symbolically the sum of the form $\sum_{n=n_0}^{n_1} n$, and then it blindly put the values for n_0 and n_1 in the formula. Don't panic, you got your towel! Let's you find a way to put something else in n_0 and n_1 in the sum $\sum_{n=n_0}^{n_1} n$, so that the HP49G will calculate symbolically sums of

programs or even directories. (What wonderful stuff for a meta-mathematics marathon!) First of all, it is impossible to enter something "inappropriate" for the sum-index or the summand right from the EQW or the command line, because the built-in mechanisms for syntax proof will not allow this. But we have the commands ->ALG, ->LST and COMP-> of the built-in menu 256. These commands are exactly what we need. (Enter 256 MENU to see these commands.) We enter a sum first, just to use it as place holder for our strange experiments with sums. Go to the EQW and enter:

$$\sum_{n=0}^1 n$$

Take that to the stack and press ->LST. When an algebraic is on stack level 1, the command ->LST turns it to and RPN-List. Now stack level 1 contains:

$$\{n \ 0. \ 1. \ n \ }$$

Press ENTER a couple of times to make some copies of this list, as we need it for several strange experiments. Now, edit this list and replace the summing bounds with two vectors, so that the list now is:

$$\{n \ [0. \ 0. \ 0.] \ [1. \ 1. \ 1.] \ n \ }$$

Now press ->ALG, which can take a list from stack level 1 and return its algebraic equivalent. Now you have the sum:

$$\begin{matrix} [1. & 1. & 1.] \\ n \\ n=[0. & 0. & 0.] \end{matrix}$$

which is reason enough to raise your arms in the air and scream. (I did that, and our neighbours look at me very strange since then.) If you now press EVAL, then you get the error "Bad Argument Type". If you press ->NUM then you get the error "Undefined Name". Don't EXPAND, your HP49G will presumably crash! Obviously, the sum can be displayed, but it can't be EVALuated or ->NUMerated.

Press ->LST to turn the algebraic object to a list and edit the list, so that it now is:

$$\{n \ << (0.,0.) \ ABS \ >> \ << (3.,4.) \ ABS \ >> \ n \ }$$

(You got it, we try to put programs as the summing limits. Now, press ->ALG. The sum that you get is:

$$\begin{matrix} << (3.,4.) \ ABS \ >> \\ n \\ n=<< (0.,0.) \ ABS \ >> \end{matrix}$$

a sum that goes (theoretically) from the magnitude of (0.,0.), which is 0., to the magnitude of (3.,4.) which is 5..

Press EVAL. After a while you get the result 'Invalid_Object'. This is a name (type 6.) which is obviously used when the HP49G wants to tell you that it found something illegal while evaluating. Press DROP to save the HP49G from illegality and crime. But we still don't give up. Let's try to put programs without the delimiters "<<" and ">>" as summing limits. We can't enter such programs directly, but we can compose them using ->PRG, which takes a list with various objects and turns it to a program with exactly the objects of the list. Enter first a 2. We need it to put our composed program at position 2 of the list that represents the sum. Now, enter the list $\{(0.,0.) \ ABS\}$, and press ->PRG. The object at stack level 1 is now:

Complex numbers with the HP49G - Part 1

(0.,0.) ABS

which is a program without the delimiters "<<" and ">>". It still can run as a normal program does. Now stack level 3 should contain the list {n 0. 1. n }, stack level 2 should contain the number 2 and stack level 1 should contain the program (0.,0.) ABS. Now press PUT. This command puts the object at stack level 1 at the position given at stack level 2 in the list given at stack level 3. (It can do more, but for now it suffices to know how it works with lists.) The result at stack level 1 is:

{n (0.,0.) ABS 1. n }

The object (0.,0.) ABS looks like two objects in the list, but it is still a single object. Now enter a 3, and then enter the list {(3.,4.) ABS}. Press ->PRG to convert the list to the program (3.,4.) ABS. Press PUT. The list on stack level 1 is now:

{n (0.,0.) ABS (3.,4.) ABS n }

Now press ->ALG and see the result:

(3.,4.) ABS
n
n=(0.,0.) ABS

Press EVAL. The result is 15., as it must be. The moral of the story is, if you have sums of which the summing limits must first be calculated through some programs, then you can put these programs without delimiters at the place holders for the summing limits in the sum. We make a small program to automate this.

```
<< -> sumidx low high summand
<<   sumidx
      low ->LST           @turn low limit to list
      TAIL REVLIST        @make program without
      TAIL REVLIST        @delimiters "<<" and
      ->PRG                @">>"
      high ->LST          @turn high limit to list
      TAIL REVLIST        @make program without
      TAIL REVLIST        @delimiters "<<" and
      ->PRG                @">>"
      summand
      { } HEAD            @Put command      on stack
      5 ->ALG             @Make algebraic
>>
>>
```

Here we see the second way that ->ALG works. When a number n is on stack level 1, the command takes n arguments from the stack and composes an algebraic object out of these objects. STORE it in ->. Now, enter n, << (0.,0.) ABS >>, << (3.,4.) ABS >>, and n and press ->. The result is:

(3.,4.) ABS
n
n=(0.,0.) ABS

which you can evaluate to 15.

Having programs as summation limits leads us to the next question: Can programs without delimiters be also used as summands? Let's see: DROP any objects from the stack until the list {n 0. 1. n } is on stack level 1. Enter a 4 (the position where the summand will be put. Now enter the program:

Complex numbers with the HP49G - Part 1

```
<<
  0.
  1. 3.
  FOR I
    I n * +
  NEXT
>>
```

Press ->LST, TAIL, REVLIST, TAIL, REVLIST ->PRG. The program now has lost its delimiters and is now:

```
0. 1. 3. FOR I I n *
+ NEXT
```

Now press PUT. The list on stack level 1 is now:

```
{n 0. 1. 0.1. 3. FOR I I n * + NEXT }
```

Again, the whole sequence of objects 0. 1. 3. FOR I I n * + NEXT is a single object, a program. Press ->ALG. Now you have the algebraic object:

```
1.
  0. 1. 3. FOR I I n * + NEXT
n=0.
```

If you EVAL this, you get 6., the right result. So putting programs without delimiters as summands seems to work. We can modify our program -> to do this:

```
<< -> sumidx low high summand
<<   sumidx
      low ->LST      @turn low limit to list
      TAIL REVLIST   @make program without
      TAIL REVLIST   @delimiters "<<" and
      ->PRG          @">>"
      high ->LST     @turn high limit to list
```

```
TAIL REVLIST   @make program without
TAIL REVLIST   @delimiters "<<" and
->PRG          @">>"
summand ->LST @turn summand to list
TAIL REVLIST   @make program without
TAIL REVLIST   @delimiters "<<" and
->PRG          @">>"
{ } HEAD       @Put command on stack
5 ->ALG        @Make algebraic
```

```
>>
>>
```

STOre this in -> . Let's try it: Enter n, << (0.,0.) ABS >>, << (3.,4.) ABS >>, and the program:

```
<<
  0.
  1. 3.
  FOR I
    I n * +
  NEXT
>>
```

for the summand. Press -> to get:

```
(3.,4.) ABS
  0.1. 3. FOR I I n * + NEXT
n=(0.,0.) ABS.
```

Press EVAL to get 90. the correct result.

A little bit more on out freak sums. You know of course that vectors are not allowed in algebraics, don't you? Well it turns out that they are allowed, but indirectly. Let's work it out. Enter the dummy sum $\sum_{n=0}^1 n$ again and press ->LST to make it a list. Now enter the list:

Complex numbers with the HP49G - Part 1

$$\{[n \ n^2 \ n^3] \text{ ABS}\}$$

Press ->ALG to be astonished once. The result is:

$$[[n \ n^2 \ n^3]]$$

a nice vector in an even nicer algebraic.

Now enter a 4, SWAP and PUT, to put the algebraic with the vector at position 4 of the list. Press ->ALG. Astonishment again, because you get the sum:

$$\sum_{n=0}^1 [[n \ n^2 \ n^3]]$$

Press EVAL to get the correct result 1.73205080757, and wonder why the "manual" doesn't even mention this, and why it is impossible to enter such things directly in the EQW. (But you know the method. Write a program that does it ;-))

Enough astonishing things. You can experiment further for yourself and don't forget to inform us about your results. Let's return again to our main path.

The numerical solver of the HP49G can only find numeric solutions of equations with real variables. It will not find complex numeric solutions. But we know that when the magnitude of a complex quantity are 0, then the quantity itself is also 0. That means that we can build the magnitude of a complex equation and solve resulting equation instead, for the real and imaginary part. Let's suppose, we have the equation $z^2 - z = 0$ and we want to solve for z , which is complex. Using ABS we find that the absolute value of $z^2 - z$ is:

$$\sqrt{\text{RE}(Z)^4 - 2 \text{RE}(Z)^3 + (2 \text{IM}(Z)^2 + 1) \text{RE}(Z)^2 - 2 \text{IM}(Z)^2 \text{RE}(Z) + \text{IM}(Z)^4 + \text{IM}(Z)^2}$$

We can replace in this equation $\text{RE}(Z)$ with a variable, say REZ, and $\text{IM}(Z)$ with IMZ and store the resulting equation in EQ. Then we give initial values to REZ and IMZ. Then we can solve the equation for REZ. If we find a root we stop. If we find an extremal value, we keep this value for REZ and solve the equation for IMZ. If we find a root we stop. If we find an extremal value we keep that for IMZ we repeat the procedure until we find a root.

In the times of GUIs the new interface of the numeric solver is perhaps more popular. But the old style interface has big advantages. For my gusto, the biggest advantage of the old style solver interface, is that you can build up a custom solver menu, that doesn't have to contain only variables but just about anything a menu can contain. Let's see how we can build such a solver menu.

The general structure of such a solver menu is:

{ equation { menu list } }.

The sub list { menu list } may contain names of variables of the equation or programs. It looks like:

```
{ varname1 varname2 ...
  { "label1"
    { unshifted action
      left-shifted action
      right-shifted action
    }
    { further action } ...
  }
}
```

The variables "varname1" "varname2" appear in the old solver menu (menu 30) as normal solver variables, that is their labels are written

Complex numbers with the HP49G - Part 1

black in a white box. They act like normal solver menu variables, that is unshifted press to store a value, left-shifted press to solve for the variable and right shifted press to recall the variable on the stack. The labels of the actions appear written white black background and their actions are taken from the sub sub sub list

```
{ unshifted action
  left-shifted action
  right-shifted action
}
```

Now, there are also variations of this schema. When an action menu item has only one unshifted action, then we can enter its sub list as:

```
{ "label1" unshifted action }
```

so that the menu list of the custom solver menu is like:

```
{ varname1 varname2 ...
  { "label1" unshifted action }
  { further actions } ...
}
```

The different sub lists can be freely intermixed, that means you can have at the same time menu keys with only one unshifted action and menu keys with unshifted, left-shifted and right-shifted actions.

The list { equation { menu list } } is stored in EQ instead of a single equation. Then pressing 30 MENU, builds the custom solver menu.

We want to have a menu with the variables of the equation and with an additional program, that does what has been said above, namely solve

for the real part, if a zero found stop. If an extremal value found, solve for the imaginary part. If a zero found stop. If an extremal value found, solve for the real part again. And so on until we have the numeric solution, or until we exceed a certain number of runs, say 50. When ready, the program should show us the numeric solution as a complex number, and the numeric value of the equation with the current values of the variables. So here we go:

```
<<
-> eq solvar
<<
EQVARS PURGE           @Clean up any variables
IFERR                  @from previous runs.
  'ORIGVAR' RCL PURGE
THEN
  DROP
END
{ EQ EQVARS ORIGVAR
  ORIGEQ } PURGE
eq 'ORIGEQ' STO         @Store original equation
solvar 'ORIGVAR' STO    @and variable
"Finding magnitude+"    @Use ABS to find
1 DISP                 @magnitude of original
eq ABS                  @equation
"Replacing RE(" solvar +
" )
with RE" + solvar + "... " +
1 DISP
"'RE(" solvar + ")"' + @Build REvar and replace
OBJ->                  @all occurrences of RE(var)
"RE" solvar + "'" +   @with it, using the
OBJ-> 2 ->LIST         @built-in command
  MATCH DROP          @ MATCH
"Replacing IM(" solvar +
" )
with IM" + solvar + "... " +
1 DISP
"'IM(" solvar + ")"' + @Build IMvar and replace
```

Complex numbers with the HP49G - Part 1

```

OBJ->                @all occurrences of IM(var)
"'IM" solvar + "" +  @with it, using the
OBJ-> 2 ->LIST        @built-in command
MATCH DROP           @ MATCH
"Finding new variables..."
1 DISP
LNAME AXL SORT REVLIST @Build new variables list
DUP 'EQVARS' STO      @Store it in EQVARS
"Building solver menu..." @make custom solver menu
1 DISP
{ { "NCSOL"           @Label
  <<                  @Action is a program
    1 50
    FOR I              @Do 50 times
      RCEQ HEAD        @Recall equation
      EQVARS           @Get first or second
      IF               @of new variables
        I 2. MOD       @alternating.
      THEN
        1
      ELSE
        2
      END
      GET
      "Solving for "
      OVER + "..." + 1 DISP
      DUP RCL ROOT DROP @Solve for it.
      "Iteration " I + @Show iteration number
      2 DISP
      EQVARS 1          @Show current values
      << DUP S~N "=" + @of new variables
        SWAP RCL + >>
      DOSUBS
      { 3 4 } DISP
      "EQ=" EQVARS EVAL @Show current value
      R->C 'ORIGVAR'    @of original equation
      RCL STO ORIGEQ ->NUM +
      5 DISP

```

```

"Press any key to stop."
6 DISP
IF                      @If original equation
  RCEQ HEAD ->NUM @solved with current
  ABS NOT           @values or we have
  I 50 OR           @reached 50 attempts
THEN                @then increase iteration
  'I' 50 STO+       @index I by 50, so that
END                 @we exit the loop at any
                   @case.
IF                  @If key was pressed,
  KEY               @then drop key code
THEN                @and increase iteration
  DROP 'I' 50 STO+ @index, so that we
END                 @exit the loop
NEXT
EQVARS 1
<< RCL >> DOSUBS      @Store (real, imaginary)
OBJ-> DROP R->C        @in original variable.
'ORIGVAR' RCL STO     @Return tagged solution
'ORIGVAR' RCL DUP     @and tagged numeric value
RCL SWAP S~N ->TAG    @of original equation.
ORIGEQ ->NUM "EQ"
->TAG
>> } } +              @Add action to the menu
2 ->LIST STEQ 30 MENU @Store equation, go to
>>                   @old numeric solver menu.

```




Store this program in CONSOL (COMplex Numeric SOLve).

Now, let's try it. Switch to complex mode, enter the equation $Z^2 + Z = 0$ and the variable Z. Press CONSOL. The program runs and some messages are dancing on the top of the screen. When it ends,


Complex numbers with the HP49G - Part 1




you are presented a new custom solver menu:





The keys   corresponding to F1 and F2, are input keys. You enter a number and press the key F1 or F2, to store initial guess values in the variables REZ or IMZ, which are the real and imaginary part of the original variable Z. The program makes these variables automatically. Enter for example 4. and press  to store a guess value of 4. in IMZ. At the top of the screen you see:


IMZ: 4

to inform you about the current value in IMZ. Store also a guess value 3. in REZ using the menu key .



Let's try first to solve the equation using the keys  . Press left-shift and then . The message "Solving for REZ" appears at the top of the screen, while the numeric solver tries to find a solution. If you press any key except [ON], then you can watch the whole process, as the solver determines better approximations for the solution.


When the best possible solution is found, then the solver writes a message at the top of the screen, to indicate what the found solution represents, and the tagged value of the solution on the stack. In our case it shows the message "Extremum" to indicate, that the found value for REZ, -.500000017497 is not a solution, but a value that minimises the equation as good as possible, when IMZ is 4. If you press the menu key , then EXPR: 16.25 is returned, which tells you that the value of the equation with the current values for REZ and IMZ is 16.25.

Now press left-shift and then the menu key , to let the solver solve for IMZ. After a while you get the message "Extremum" again

and IMZ:3.62807589005E-9 on stack level 1. Pressing  returns Expr: .249999999999, i.e. we approach a solution.

You see where this goes. We alternate solving for REZ and IMZ until the solver shows the message "Zero" or "Sign Reversal". "Zero" means that the equation is 0. with the current solution values. "Sign reversal" means that the solver didn't find a value that is a root of the equation within the precision 12 digits of the HP49G, but found two adjacent values for which the equation changes sign, and thus are very near a solution if the equation doesn't have problems like jumps, poles etc. between these two values.

Now let's find the solution with the menu key . Store the initial guess value 4. in IMZ and 3. in REZ. Press the menu key . You see the alternating messages "Solving for IMZ..." and "Solving for REZ..." a couple of times, additional information about the current values in REZ and IMZ and also the value of the original equation for the current values of REZ and IMZ. After a while the solution Z:(0.,0.) and the value of the original equation EQ:(0.,0.) are returned on the stack.

Let's try another example. Enter the equation $Z - \cos(Z)$ and the variable Z to solve for. Press CONSOL. When the program is ready, you see the solver menu. Enter initial guess values -1. for REZ and 2. for IMZ. Press the menu key . After some iterations, the values for REZ and IMZ don't change anymore. Press any key except [ON] to stop the solver. After a while you have the current solution Z:(-2.4869,1.8093) and the value of the original equation EQ:(.000004310.00000177) on the stack.

You can add your items to the menu, that for example do some calculations with the current values of the variables, or store them in other variables to have them later, or for other purposes.

Ending this first part of the complex numbers marathon, I add two things that are of interest. The first item of the list that is stored in EQ,

Complex numbers with the HP49G - Part 1

can also be a list of equations. Then you could for example add a menu item that cycles the current equation and its set of variables.

The label for the menu key can be a string or a GROB (graphics object) that is exactly 21x8 pixels. So you can customise the look of your menu keys in a much more detailed way than string allow.

Enough for now. Next time we will mess around with symbolic complex objects and learn about the confusing modes combinations of the HP49G. Hope you enjoyed it,

Complex greetings,
Nick.

Complex numbers with the HP49G - Part 2

Hi everybody!

Beginning this second part of the Complex Numbers Marathon, I first must say a big big "thank you" to all people out there, that encourage me to continue posting their thoughts and comments and the interest that they show for the marathon. My mail account is now heavier ;-)
Many warm thanks from me and Trabakoulas to all of you.

In the first part of the complex marathons we used programs with the commands SREPL, HEAD, TAIL, DISP, POS, SUB and also the clause IFERR THEN ELSE END. I didn't describe what they do, because this would make the first part even bigger. But because these commands are eventually new to some people, they will be described at the end of this part.

We will continue our marathon with symbolic expressions that contain no variables and can be directly evaluated to complex numbers using the commands \rightarrow NUM or XNUM. An example for such an expression would be $1 + 1i$, which is type 9. (for symbolic expression), but can be converted to (1.,1.) (type 1. for complex number) using \rightarrow NUM. Let's find a nice abbreviation for "symbolic expressions that can be directly evaluated to complex numbers using \rightarrow NUM or XNUM". Let's call them senn, an acronym for Symbolics Evaluable to Normal Numbers. (In swiss german this means also "shepherd", which fills Trabakoulas with joy ;-)) So from now on when we talk about a senn, we actually mean some symbolic expression, like for example $1 + 1i$ or $\frac{2}{1-8}$, than doesn't contain any variables and can be converted to a real or complex number using \rightarrow NUM or XNUM.

Let's first setup our HP49Gs for the sake of identical results. We configure complex mode, rectangular coordinates and radians angle mode. We need an additional setting now. Press [MODE] to go to the calculator modes screen, and then press the menu key [FLAGS]. Press [blue-shift] and then the key [arrow down] to go one page downwards in the flag catalogue. Now select flag 27 and check it using the menu key [CHK]. A small " " character must be visible to the left of this

flags and the description to the right must be: 'X+Y*i' \rightarrow 'X+Y*i'. Press [ENTER] twice to accept these settings and return to the stack. Instead of this you could also enter -27 SF, which sets system flag -27. Now, what is that flag good for? Well, when it is set, complex senns are displayed as $X + Y i$, with X the real and Y the imaginary part. Let's try to enter the senn $1 + 1i$ from the equation writer. Press [EQW], type 1 and press [red-shift], [SPC], type again 1 and press [ENTER]. The senn in the stack now takes the form $1 + 1i$, though in the equation writer it was (1,1). Let's try to enter the same senn a bit different. Go to the equation writer and type $1 + 1i$. You get the imaginary unit i by pressing [blue-shift] and then the key [TOOL]. Press [ENTER]. As you can see the two senns are completely identical.

Now, we enter a more complicated senn, $\frac{1}{2} - 3 + \frac{\sqrt{3}}{2} - 1i$. Go to

the equation writer and enter $\frac{1}{2} - 3$. If you now press [red-shift], [SPC] to put the comma in the senn and separate the real from the imaginary part, then the EQW looks like $\frac{1}{2} - (3,)$. This shows that the HP49G considers only the first term before the comma to be the real part of the senn. To tell the HP49G what is meant as the real part, you must select the whole real part. In our example you must press the key [arrow right] twice. The part $\frac{1}{2} - 3$ is then selected, it is displayed inverse. Now, you can press [red-shift], [SPC] and the EQW looks like $\frac{1}{2} - 3,$, which shows that the HP49G accepted the whole term as the real part of the senn. Enter now the imaginary part, so that the senn is complete $\frac{1}{2} - 3, \frac{\sqrt{3}}{2} - 1i$. Now enter this to the stack. It changes immediately to $\frac{1}{2} - 3 + \frac{\sqrt{3}}{2} - 1i$.

Complex numbers with the HP49G - Part 2

"Yes", you will say, "but can we enter senns without having to go to the equation writer? Can we enter them right from the command line?" Of course we can. Remember the difference between a senn and a complex number? The first is an algebraic expression, type 9. The second is a complex number, type 1. How do you enter algebraic objects from the command line? Right, you press [red-shift], [EQW] first, to put a pair of single quotes on the command line, and then you enter your expression. The same you do for a senn. Press [red-shift], [EQW] and then type, for example $(1/2 - 3, \sqrt{3}/2 - 1)$ between the single quotes. If you now press [ENTER], the senn will be displayed

as $\frac{1}{2} - 3 + \frac{\sqrt{3}}{2} - 1 \ i$ on stack level 1. Alternatively you can press

[red-shift], [EQW] and then type $1/2 - 3 + (\sqrt{3}/2 - 1) \ i$ between the single quotes.

Now, let's see what happens to senns when we change coordinates and/or angle mode. Enter $1 + 1 \ i$ and press CYLIN to change to polar coordinates. While complex numbers (type 1.) change to reflect mode changes, the senn remains the same. With $1 + 1 \ i$ on stack level 1,

press EXPAND and look how the senn is now written, $\sqrt{2} \ e^{i\frac{\pi}{4}}$. This is the polar form of the complex senn $1 + 1 \ i$. Its radius is $\sqrt{2}$ and its angle $\frac{\pi}{4}$. You remember the relation $e^{i\theta} = \cos(\theta) + i \sin(\theta)$ from the trigonometry marathon.

Combine it with the relation $(x,y) = r (\cos(\theta) + i \sin(\theta))$ and you get exactly what the HP49G

did, namely the complex senn in its polar form $r e^{i\theta}$. If you want, you can press SINCOS to convert the complex exponential to SIN and

COS functions. The result is $\sqrt{2} \frac{\sqrt{2}}{2} + i \frac{\sqrt{2}}{2}$. If you now press

EXPAND, the HP49G returns the polar form again. Change coordinates back to rectangular and press EXPAND again to get what we started with, $1 + 1 \ i$. Let's try an example that is a bit more

complicated. Enter $\frac{1}{2} - 3 + \frac{\sqrt{3}}{2} - 1 \ i$. Change to polar coordinates

mode and press EXPAND. The returned result is the polar form of the senn, which is also more complicated, but correct:

$$\frac{(16 - 2\sqrt{3}) \sqrt{488 + 61\sqrt{3}}}{61} e^{i \operatorname{ATAN} \frac{2-\sqrt{3}}{5}}$$

Change to rectangular coordinates mode and press EXPAND again. Now the senn is in the form:

$$\frac{(8 - \sqrt{3}) \sqrt{488 + 61\sqrt{3}}}{61} e^{i \operatorname{ATAN} \frac{2-\sqrt{3}}{5}}$$

No back transform has been done from the complex exponential to real and imaginary parts of the senn. So we have our first thing to be aware of, when we work with senns. Polar forms may not be transformable back to rectangular forms by simply changing to rectangular coordinates and pressing EXPAND. Let's see what happens when we explicitly tell the HP49G to find the real and imaginary part of the senn. Press [ENTER] to make a copy of the senn, and then press RE. The HP49G returns:

$$\frac{\operatorname{COS} \operatorname{ATAN} \frac{2-\sqrt{3}}{5} - \sqrt{488 + 61\sqrt{3}} (8 - \sqrt{3})}{61}$$

Now we see perhaps a bit clearer what the problem is. The cosine of

$\operatorname{ATAN} \frac{2-\sqrt{3}}{5}$ can't be symbolically calculated. For "easy"

Complex numbers with the HP49G - Part 2

angles, like for example $\frac{1}{4}$, the built-in routines return the sines and cosines (or equivalently the complex exponentials) as simple symbolics, like for example -1 or $\frac{\sqrt{2}}{2}$, but for more complicated things they don't do that. It is up to us to decide what must be done further. In this case, we remember TEXPAND from the trigonometry marathon and apply it to the last result. This returns the ugly expression:

$$\frac{-1 \frac{1}{(16-2\sqrt{3})\sqrt{61(8+\sqrt{3})}} + \frac{2-\sqrt{3}}{5} \sqrt{488+61\sqrt{3}}(8-\sqrt{3})}{0 \frac{1}{(16-2\sqrt{3})\sqrt{61(8+\sqrt{3})}} + \frac{2-\sqrt{3}}{5} \sqrt{488+61\sqrt{3}}(8-\sqrt{3})} \frac{61}{2}$$

which we can EXPAND to get $\frac{-5}{2}$, the real part of the senn that we

started with, $\frac{1}{2} - 3 + \frac{\sqrt{3}}{2} - 1$ i. The same way, we can use IM,

TEXPAND and EXPAND, to obtain the imaginary part $-\frac{2-\sqrt{3}}{2}$.

Sometimes we can also follow another way. We can use ->NUM to turn the polar form to a complex number, and then use XQ to turn it to a senn again, hoping that the HP49G will find the real and imaginary

parts that we started with. Enter again $\frac{1}{2} - 3 + \frac{\sqrt{3}}{2} - 1$ i, switch to polar coordinates and press EXPAND to get the polar form:

$$\frac{(16-2\sqrt{3})\sqrt{488+61\sqrt{3}}}{61} e^{i \text{ATAN} \frac{2-\sqrt{3}}{5}}$$

Now switch back to rectangular form and press ->NUM and then XQ.

$$\text{The result is } \frac{-5}{2} + i \frac{-5432}{40545}$$

which shows that for the real part we were lucky, but the imaginary part couldn't be exactly found, so the HP49G returned a ratio that is equal to the numeric value of $-\frac{2-\sqrt{3}}{2}$ up to 12 digits of accuracy.

While we are at it, let's see what the differences between the related commands ->NUM, XNUM, I->R on the one side, and on the other side ->Q, ->Q+, XQ, R->I.

->NUM will turn any senn to a real or complex number, if the condition is true, that all operations in the symbolic expression are convertible to numbers through built-in operations. For example, if the symbolic expression contains $\sqrt[3]{(2.,1.)}$, using ->NUM will error with "Bad argument type". (See part 1 of this marathon.)

XNUM will do the same as ->NUM, but will also change to approximate mode. This mode change is carried out, even if the senn can't be turned to a number, for example because it contains some "illegal" operation.

I->R will turn any integer to a real number. It turns only integers to reals. If you use it for, say $\frac{3}{2}$, it will simply return I R $\frac{3}{2}$.

->Q converts some given real or complex number to a ratio that is equal to the number in the 12 digit approximation. It also changes mode to exact. This command returns strictly quotients. For example, if you

Complex numbers with the HP49G - Part 2

enter 1.41421356237 and use $\rightarrow Q$, the result will not be $\sqrt{2}$ but $\frac{961664}{665857}$.

$\rightarrow Q$ does what $\rightarrow Q$ does, but also can return non-rational results and it doesn't change mode to exact. If you feed it with 1.41421356237 it will return $\sqrt{2}$. The ability to find such non-rational expressions is not unlimited. If you enter for example $\sqrt{2} - \sqrt{3}$, press $\rightarrow NUM$, and then use $\rightarrow Q$, then the result is no more the original expression but $\frac{-3480}{10949}$.

XQ gives the same results like $-Q$, but it also changes to exact mode.

The commands $\rightarrow Q$, $\rightarrow Q$ and XQ will also accept other types of arguments, any try to convert any occurrences of real numbers to quotients (or non-rational expressions). For example, XQ converts the expression $1.41421356237 * X$ to $\sqrt{2} X$.

R \rightarrow I accepts real numbers without decimal part and converts them to integers. It will error with "Bad argument type" for any other argument.

Let's now continue with the possible ways to enter senns in polar form. Of course, as you can imagine, we can enter a senn as a complex exponential. To make an example, the complex senn $1 + 1i$ can be

entered as $\sqrt{2} e^{i\frac{\pi}{4}}$. But the question is, if we can enter such senns using the notation (r, \angle) . The answer is yes, but I found this "yes" quite accidentally while writing this part. You can enter such a thing from the command line. But you *must* enter it as $'(r, \angle)'$ using single quotes and a comma *and* the character \angle to separate radius and angle. Enter for example $'(1, \angle /3)'$ in the command line, and press [ENTER]. The complex senn is immediately converted to

$1 \cos \frac{\pi}{3} + 1 \sin \frac{\pi}{3} i$, which is the same senn in the notation (realPart,imaginaryPart). This automatic conversion is carried out regardless of coordinates mode. If you EXPAND

$1 \cos \frac{\pi}{3} + 1 \sin \frac{\pi}{3} i$ while you are at rectangular coordinates

mode, you will get $\frac{1+i\sqrt{3}}{2}$. But if you EXPAND the expression

while in polar coordinates mode, you will get $\frac{2 e^{i\frac{\pi}{2}}}{2}$. The strange thing is that the 2 of the numerator and the 2 of the denominator are not cancelled out using EXPAND. You must COLLECT this to get $e^{i\frac{\pi}{2}}$. (!)

If you apply EXPAND on $1 \cos \frac{\pi}{3} + 1 \sin \frac{\pi}{3} i$ in rectangular

coordinates mode, and you get $\frac{1+i\sqrt{3}}{2}$, then you can also switch

afterwards to polar coordinates and apply EXPAND on $\frac{1+i\sqrt{3}}{2}$. The

result is again $\frac{2 e^{i\frac{\pi}{2}}}{2}$.

The conversion to the polar representation $r e^{i\angle}$ using EXPAND in polar coordinates mode, has also its limits. For example enter the senn $'(2, \angle \sqrt{2})'$. It will be converted to $2 \cos(\sqrt{2}) + 2 \sin(\sqrt{2}) i$. If you now switch to polar coordinates and press EXPAND, then you get $2 e^{i\frac{\pi}{2}} \sin(\sqrt{2}) + 2 \cos(\sqrt{2})$. Only the imaginary unit i has been

Complex numbers with the HP49G - Part 2

converted to its polar notation $e^{i\sqrt{2}}$. Instead EXPAND in polar coordinates, you can use the appropriate combination of EXPLN, TSIMP, LIN and EXP2POW in rectangular coordinates. Switch to rectangular coordinates and enter $(2, \sqrt{2})$, which is automatically converted to $2 \cos(\sqrt{2}) + 2 \sin(\sqrt{2}) i$. Press EXPLN to obtain:

$$2 \frac{e^{i \frac{\text{LN}(2)}{2}} + \frac{1}{e^{i \frac{\text{LN}(2)}{2}}}}{2} + 2 \frac{e^{i \frac{\text{LN}(2)}{2}} - \frac{1}{e^{i \frac{\text{LN}(2)}{2}}}}{2 i}$$

EXPLN converts trigonometric functions to complex exponentials.

Now press LIN to get $2 e^{i \frac{\text{LN}(2)}{2}}$ and then EXP2POW to convert this to $2 e^{i\sqrt{2}}$. LIN linearises products of exponentials. For example it will convert $e^a e^b$ to e^{a+b} . EXP2POW converts expressions of the form $e^{a \text{LN}(b)}$ to b^a and $e^{\frac{\text{LN}(b)}{a}}$ to $\sqrt[a]{b}$, when this doesn't lead to ambiguous expressions.

As we can see, senns can lead to complicated expressions, so perhaps we should make a small visit to the editing environments and commands of the HP49G, because sometimes we would like to edit a small part of an expression, add or remove something etc. In general there are two editing environments built-in. The command line and the "best" editing environment. The latter is the EQW for algebraic objects, the matrix editor for matrices and vectors and the PICTURE environment for GROBs. Of course there are lot of programs available, for editing objects, but we present only built-in capabilities here. (Or else I would be writing marathons until the end of time ;-))

When we want to edit something in the environment that best fits the object, we simply press the key [arrow down]. This automatically invokes the best editing environment and takes the object at stack level 1 to that environment for editing. So, when a senn is on stack level 1

and you press [arrow down], the HP49G starts the equation writer and puts the senn there for editing. But this action, pressing the key [arrow down] is a non-programmable operation. The programmable command for this, is the command EDITB. When you put this command in a program then the HP49G puts the object at stack level 1 to the best possible editing environment and the program halts. You can edit the object as long as you wish, add new terms, EXPAND etc. You can even leave the editing environment temporarily, do something in the stack or in any other editing environment, and then return. The HP49G "accumulates" the pending returns and while the object is still being edited, the program from which the command EDITB was executed is still halted. When you press ENTER (or CANCL) the edited object is put on stack level 1 again, and the program continues.

When you want to edit the contents of a variable in the best suitable environment, you use the command VISITB. This command takes the name of a variable from stack level 1, a variable that exists in the current directory, and puts its *contents* in the best suitable editing environment. When the user presses ENTER or CANCL again, the edited contents are stored in the variable, and the program continues. An additional difference between EDITB and VISITB, that you must be aware of, is that EDITB returns the edited object on stack level one when you press ENTER or CANCL, but VISITB doesn't return anything to the stack.

Sometimes we want to edit some object in the command line, regardless if there is some better suitable editing environment. To do so, we press the keys [left-shift] and then [arrow down]. The object in stack level 1 is then taken to the command line, where it can be edited. This is also a non-programmable operation. The programmable command for this action is EDIT. In programs it behaves like EDITB, with the difference that the object is taken in the command line. The analogous command editing contents of variables in the command line, is VISIT.

When the program should give the user the opportunity to only view but not edit an object, then the command SCROLL can be used. This is a "universal viewer" which allows viewing but no editing.

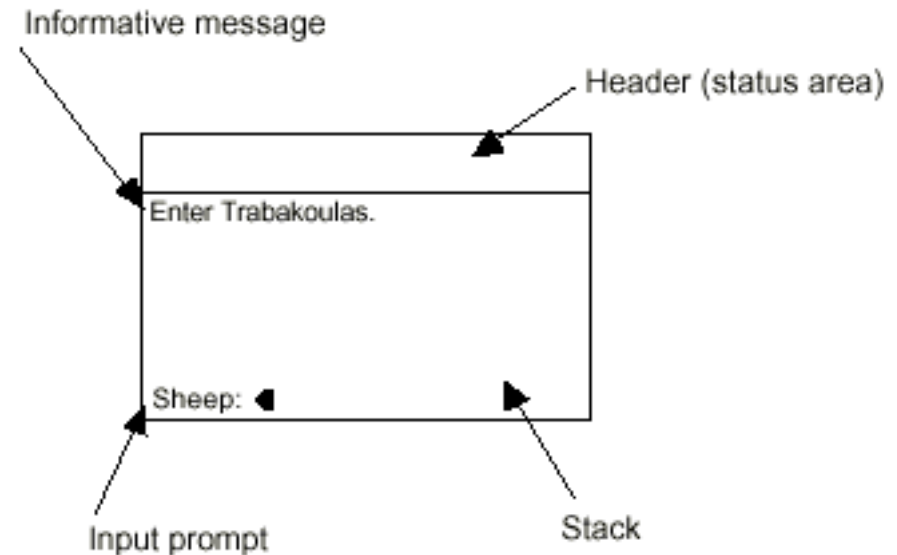
Complex numbers with the HP49G - Part 2

Let's now return to our main path again. You remember that there is no built-in way to enter complex numbers in polar form with units of angle. We had to use the vectored enter capability to do that. So we would think that there is also no built-in way to enter senns in polar form with units of angle. But here we get quite an unexpected behaviour. From the command line enter `'(3, \2_r)`. Be aware to type the comma character `,` and the angle character `\`. Surprisingly enough, the HP49G converts this to $3 \cos(2_r) + 3 \sin(2_r) i$ (!) So we think that, perhaps typing the comma character `,` and the angle character `\` when we want to enter complex numbers in the notation `'(r, \ _r)` with units of angle, could work. And it does! So if you don't want to use the vectored enter facility, you can enter your complex number as a senn in polar form with units of angle, and then use `->NUM` or `XNUM` afterwards to make it a complex number. Suppose for example that you would like to enter the complex number $(2.3 \sqrt{2.5_r})$ but you don't want to use vectored enter for some reason. You can enter the senn `'(2.3, \2.5_r)` instead, which is automatically converted to $2.3 \cos(2.5_r) + 2.3 \sin(2.5_r) i$ upon pressing [ENTER].

Then you can press `->NUM` to convert this to a complex number that is displayed according to the current settings for angle unit and coordinates system.

There is still (at least) an alternative (and a little bit cumbersome) way which doesn't use vectored enter. And this is: Make your own shell. It sounds to be complicated and admittedly it is not very easy, but it is also not so hard, as you might believe. The HP49G has many sophisticated commands for input. One of them is `INPUT`. In its simplest form, it takes a string from stack level 2 and a string from stack level 1. The string from stack level 2 is used to display an informative input message under the separation line between the stack and the header. The string from stack level 1 is used as the "input prompt". The arrow cursor blinks at the left of the input prompt awaiting your input. Enter for example the two strings "Enter Trabakoulas" and "Sheep:" and press `INPUT`. (You can find `INPUT`

and other related commands by pressing the keys [left-shift], [CAT], [NXT] and the the menu key [IN].) Now the display is:



Type 123 and press [ENTER]. The input prompt string concatenated with your input "123" is now on stack level 1. What should happen with this string, extract data etc., is up to the program that uses the command `INPUT`.

If we use this command in a loop, then we can use an empty string as informative message, and build an input prompt string that represents the stack (pseudo-stack) and concatenate some character with it, say `">"` as the command prompt, to let things look like a command line from an operating system. The program should parse the input and if it finds a senn in the form `'(r, \ _unit)`, where `r` and `_unit` are magnitude and angle, it converts it to the syntactically correct form $r \cos(_unit) + r \sin(_unit) i$. (Or any syntactically correct form, that you want to have.) The program should also include some civilised form of exiting, say by checking if the user entered "EXIT".

Complex numbers with the HP49G - Part 2

In this case it should clean up and return the user to the stack. Let's do such a program.

```
<<
DEPTH ->LIST          @Make a list with stack
                        @contents.
20. ""                @Enter depth of pseudo
                        @stack and an empty command
                        @string.
-> stack sDepth cmd @Store in local
                        @variables.

<<
  WHILE
    cmd "EXIT"         @While the command isn't
                        @"EXIT"
  REPEAT
    stack OBJ-> DROP @Explode stack list
    ""                 @Informative message
                        @(empty).
    sDepth stack SIZE
    IF
      DUP2 >           @If pseudo-stack depth is
                        @greater than number of
                        @objects in stack
    THEN
      1. + "" UNROT     @Build up string that
                        @looks like "upper"
                        @empty stack levels
      FOR I
        I R->I ":"
      " + + -1.
      STEP
    ELSE
      DROP2 ""
    END
    IF
      stack SIZE 0 >   @If the stack isn't
                        @empty
    THEN
      stack DUP SIZE   @Make a list with the
      DUPDUP sDepth MIN @stack objects to fill
      - 1 + SWAP SUB    @the "lower" pseudo
                        @stack levels.
      1
      <<
        stack SIZE NSUB @Build a string for
        - 1 + R->I ":"   @each stack level
        + SWAP ->STR +
        "
      >> DOSUBS         @Do to each object in
                        @stack list.
      IF
        DUP SIZE 1 >    @If resulting list
                        @has more than one
                        @elements
      THEN
        LIST            @Concatenate strings
      ELSE
        HEAD
      END
      +
    END
    "-----
    @String of
    @exactly 22 "-"
    @characters in bold
    @style and one ">"
    @in normal style,
    @separation line
    @between the pseudo stack
    @and the command prompt.
    @">" will look like the
    @input prompt.
    + cmd +
    INPUT
  " +
  >"

```

Complex numbers with the HP49G - Part 2

```

DUP
"-----
>"
@This is exactly the same
@string with the same
@styles as above

POS 30 + OVER
SIZE SUB
DUPDUP
"(' " POS
OVER ")'" POS
1 +
SUB
-> cmdstr cmplx
<<
  IF
    cmplx " \ " POS
  THEN
    cmdstr cmplx
    DUP 1 OVER
    " \ " POS 1 -
    SUB ", " +
    cmplx
    DUP " \ " POS
    OVER SIZE
    SUB +
    SREPL DROP
  ELSE
    cmdstr
  END
>>
DUP
'cmd' STO
IFERR
  OBJ->
  @If input can't be
  @converted to an object
  @Make error message
  @and display it until
  @a key press.
THEN
  "Error "
  ERRN +":

```

At this place you can
put your own code for
parsing

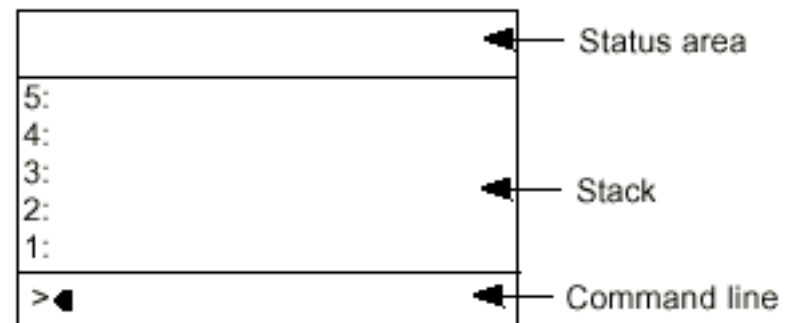
```

" + ERRM +
"
Press key to continue."
+ 1 DISP 0. WAIT
CLEAR
ELSE
  DEPTH ->LIST @Else make list of stack
  'stack' STO @Store in 'stack'
  IF
    cmd
    "EXIT" @If command isn't "EXIT"
  THEN
    " " 'cmd' @Store empty string
    STO @in 'cmd'
  END
END
END
stack OBJ-> @End of WHILE REPEAT
DROP2 @Explode stack list
@Drop element count, 'EXIT'
>>
>>

```

Store this program in SHELL, clear flag -55, to enable last arguments recovery, clear the stack and run it. It is not necessary to clear the stack, but we do that, so that the following description of the screen is the same with what your HP49G displays.

When you start the program, the screen of the HP49G shows:



Complex numbers with the HP49G - Part 2

Type for example 41.23 and press ENTER. Now the display is:

5:
4:
3:
2:
1:41.23
>◀

Type 28.93 and then +. Then press ENTER. You see:

5:
4:
3:
2:
1:70.16
>◀

Now type ' $4 \sqrt{2}r$ ' and press ENTER. The display is now:

5:
4:
3:
2:70.16
1:'4*COS(2_r)+4*SIN(2_r)*i'
>◀

If the stack becomes higher than the 5 displayed objects, you can simply press the key [arrow up] to view the other objects that are off screen. But remember to always return to the command line, before inputting anything, because otherwise the input will not be recognised. Press now [right shift] and then [backspace]. The command CLEAR is put on the command line. Press ENTER and the stack will be cleared. Let's try something that will produce an error. Press + and then ENTER. You see:

Error # 201h:
Too few arguments
Press key to continue
4:
3:
2:
1:
> + ◀

Press any key, edit the command line to 2 3 + and then press ENTER. Now the result 5 is on stack level 1. Type EXIT and the program quits, leaving the stack contents on the "real" stack.

Of course the command line parsing is only rudimentary and it will fail in many cases. But you can add your perfect parser code at the place indicated in the program listing so that you have your own personalised shell. As you can see the possibilities are endless.

In the above program we used the command INPUT with its simplest arguments, two strings. But this command allows more control of the command line. It can take a string at stack level 2 and a list at stack level 1, which specifies the content and the interpretation of the command line. The list can contain one or more of the following objects in any order:

- 1) The string that will be used as input prompt.
- 2) A real number or a list of two real numbers which specify

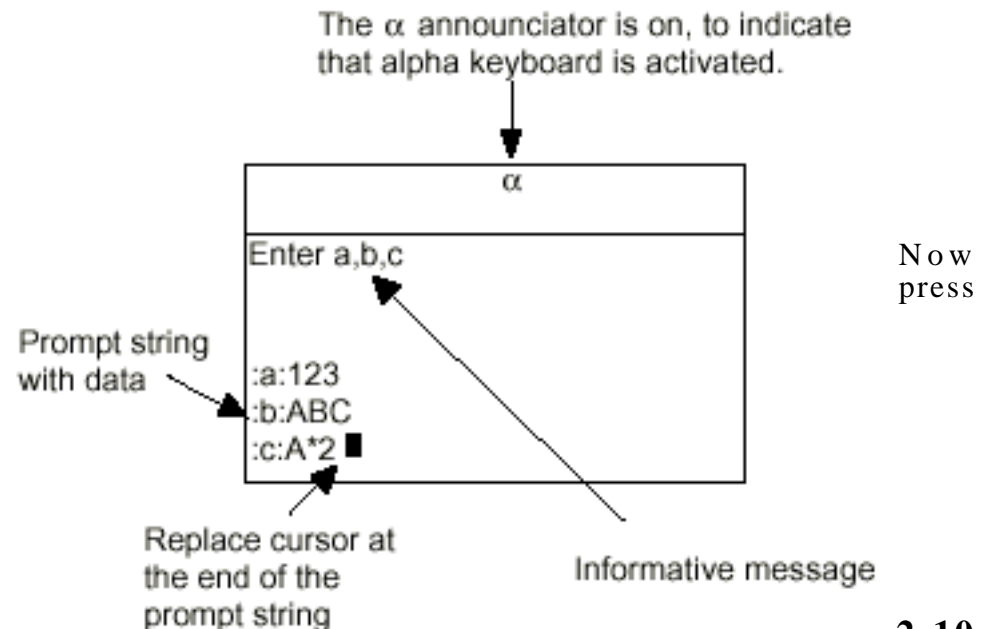
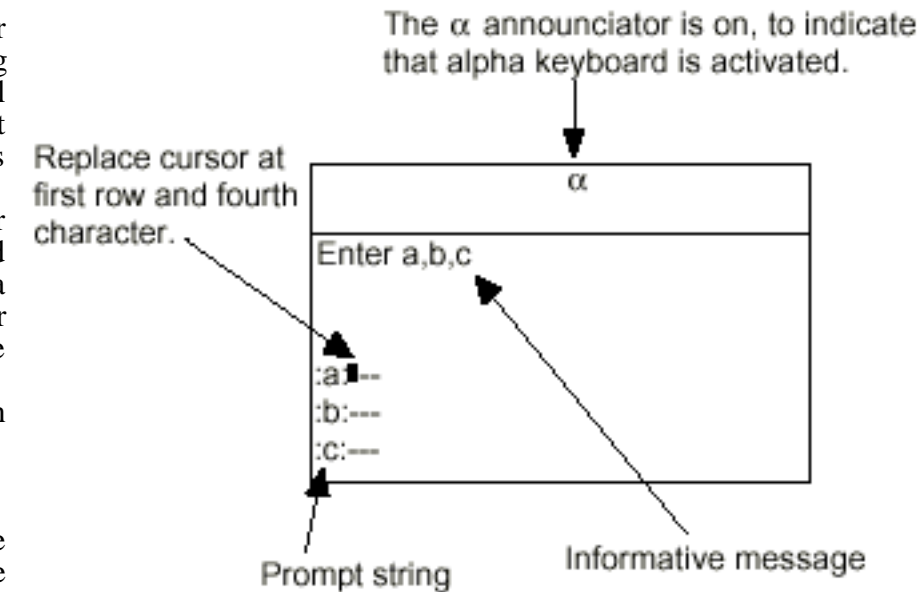
Complex numbers with the HP49G - Part 2

exactly where in the prompt string the blinking cursor should appear.

- a) When a real number is used, then it specifies that the cursor should appear at the nth character of the prompt string, starting at the left of the first line of the prompt string. When the real number is positive, then the insert cursor is specified. When it is negative, then the replace cursor is specified. The 0. specifies the end of the prompt string (default).
 - b) When a list of two real numbers is used, then the first number specifies the row of the prompt string where the cursor should appear. A positive row number specifies insert cursor, a negative specifies replace cursor. The second real number specifies that the cursor should appear at the nth character of the specified row.
- 3) The parameter ALG without any quotes. When present, then algebraic entry mode is activated.
 - 4) The parameter α without any quotes. When present the alpha-keyboard will be activated.
 - 5) The parameter V without any quotes. When present, then the contents of the command line, that is the prompt string and the input data, will be automatically verified when you press ENTER. When this string composes one or more valid objects, then the prompt string concatenated with the input data is put on the stack. When some invalid object is contained in this string, then "Invalid Syntax" is displayed and you are prompted again for data.

Let's make an example. Enter "Input a,b,c". Then enter the list: { ":a:--- :b:--- :c:---" {-1. 4.} V } and press ENTER. ("␣" is the displayed character for new line when a string is in a list.) Now press INPUT. The HP49G display looks like the picture at the top of the right column.

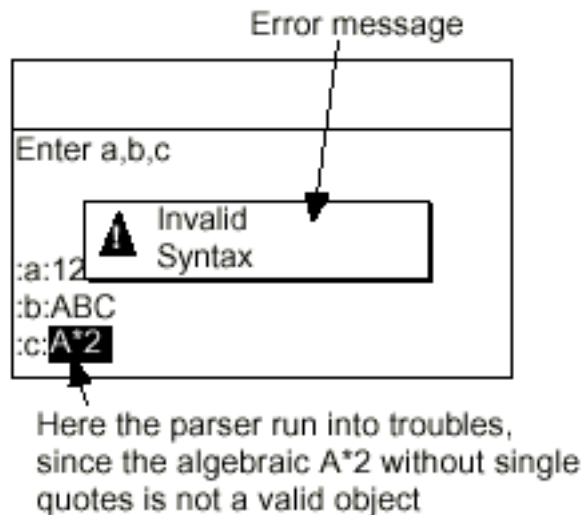
If you start typing, then the input data replaces the "-" characters. Use the arrow keys to move the cursor to the appropriate positions and enter the data: 123 for a, ABC for b, A*2 for c. Note that you don't have to lock alpha keyboard, since this is automatically done because the input list contained the parameter α . The picture to the right shows what the display looks like.



Complex numbers with the HP49G - Part 2

[ENTER]. The HP49G responds:

Press [F6] for "OK" (or [ENTER] if you want). The error message disappears, and the input screen appears again, but this time the syntactically invalid input is highlighted, to help you see where the error occurred. Use the arrow keys to put the cursor at the character "A" on the third line any type 'A*2'. Press [ENTER]. The string is put on stack level 1:



```
" :a:123  
:b:ABC  
:c:'A*2'
```

You can use now this string as you like to extract the contained data. You can understand better what the parser considers to be a valid object, if you ask yourself the question: "Would the HP49G error if I would enter this data from the command line?" For example, think what would happen, if you would enter A*2 from the command line. (Try it out.)

If you want to enter syntactically invalid objects which the program changes to syntactically correct objects, then you shouldn't use the parameter V in the list for the command INPUT. This is exactly what the program SHELL does. It doesn't use the built-in verifying procedure, but it parses and corrects itself.

Now it is time to look at the mysterious flag -27. Many of you have

already played with this and eventually wondered why some senns are displayed as (X,Y) and some other in the form $X + i Y$ when this flag is clear. To display such senns in the form (X,Y) , the HP49G must first recognise them. It turns out that it can only recognise senns that are exactly in the form $X + i Y$. The slightest deviation from this form already causes displaying them just as they were input.

Let's do some examples. First clear the flag -27 and set rectangular coordinates and radians angle mode. Go to the EQW and enter the senn $3 + 2 i$. Now press [ENTER]. The senn is put on the stack and is displayed as $(3,2)$. You can immediately see that this is a senn (type 9.) and not a complex number (type 2.) because the real and imaginary parts don't have the decimal point. Now, go again to the EQW and enter $3 + 2 i$. The only change is that we changed the positions of i and 2 in the imaginary part. Press [ENTER]. As you can see, this little change was enough for displaying the senn as $3 + 2 i$ and not $(3,2)$. Furthermore, enter the simple senn $1 + i$. Even this is not displayed as $(1,1)$ because it is not in the form $X + i Y$. You have to enter it as $1 + 1 i$, if you want the HP49G to display it as $(1,1)$. And the senn $1 - 3 i$ must be entered as $1 + (-3) i$ in order to be displayed as $(1, -3)$. Alternatively you can go to the EQW and press [1] [right shift] [SPC] [3] [+/-] [ENTER].

I already hear the voices: "Yes, but I want all complex senns to be displayed as (X,Y) . Why can't the HP49G do that?" Well, it can't do that automatically for any possible syntax of a senn. (Remember, parsing can be a complex story ;-). But we can write another programmi for this. We only need to find the real and the imaginary part of a complex senn, multiply the imaginary part with i and add the two parts. Look at the following program (on the next page):

Complex numbers with the HP49G - Part 2

```
<<
  DUP RE EXPAND
  SWAP IM EXPAND i * +
  { '&A-&B*i' '(&A,-&B)' } MATCH DROP
  { '&A+i' '(&A,1)' } MATCH DROP
>>
```

Store this program in ->CPAIR. Why do we need the first MATCH? Well, when we have a senn like for example $(16 - 23 i) (25 + i \sqrt{3})$, the first EXPAND converts it to $400 - 575 i + (23,16) \sqrt{3}$. Then the command sequence RE EXPAND returns $400 + 23 \sqrt{3}$ and the command sequence IM EXPAND returns $-(575 - 16 \sqrt{3})$. Multiplying the last result with i returns $-((575 - 16 \sqrt{3}) i)$ and the following addition returns $400 + 23 \sqrt{3} - ((575 - 16 \sqrt{3}) i)$, which is not in the form $X + Y i$ and so cannot be displayed as (X,Y) . The

MATCH does good because it brings this complex senn in the form $400 + 23 \sqrt{3} + -(575 - 16 \sqrt{3}) i$ which is of the form $X + Y i$ and so can be displayed as $(400 + 23 \sqrt{3}, -(575 - 16 \sqrt{3}))$. We also have a second MATCH, to turn senns with an imaginary part of 1 to the appropriate form. For example, if we had a senn like $3 + i$, then finding its imaginary part with IM would return a 1. Multiplication with i returns i and not 1 i because of auto-simplification. Addition of the real and the imaginary part results in $3 + i$ which is not in the form $X + Y i$. The second MATCH converts the senn to $3 + 1 i$, which is then displayed as $(3,1)$. Now, let's try some examples. Enter $1 + i$ and press ->CPAIR. The result is $(1,1)$. Enter $1 - i$ and press ->CPAIR again. The result is $(1,-1)$. Let's go for a more complicated senn. Enter $\frac{1+i}{1-i}$ and press ->CPAIR. You get i as result. (The same could be also

obtained with EXPAND.) Now, you may think that the program should return (0,1) in this case. But there is one reason not to do so. The resulting senn is one that is purely imaginary. It has no real part. So it appears more reasonable (to me) to have it in this form, exactly as we would want the number 3 to be displayed as 3 and not as (3,0).

Enter the senn $(16 - 23 i) (25 + i \sqrt{3})$. Press EXPAND. The result $400 - 575 i + (23,16) \sqrt{3}$ is not what we want. Press COLLECT. The result $-(25 i) + \sqrt{3} (23,16)$ is also not what we want. But if you press ->CPAIR, you get the complex senn displayed as $(400 + 23 \sqrt{3}, -(575 - 16 \sqrt{3}))$, that is in the form (X,Y) .

An even more complicated example. Enter:

$$\frac{\frac{2}{i-1} + \frac{3}{i-2}}{3 - \sqrt{6}}$$

EXPAND returns $-\frac{(33,24) - (8 - 11 i) \sqrt{6}}{75}$. COLLECT returns $-\frac{((i \sqrt{2} + \sqrt{3}) (11,8) \sqrt{3})}{3 \cdot 5^2}$. Press ->CPAIR. Now the result is $-\frac{33 - 8 \sqrt{2} \sqrt{3}}{75}, -\frac{24 + 11 \sqrt{6}}{75}$, a complex in the form (X,Y) .

Another example. Enter $5 \sqrt{5 - \sqrt{9}} + \frac{2}{9} i \sqrt{3} (4 - i^3)$. Pressing EXPAND returns $\frac{(180,45) \sqrt{2} - (2 - 8 i) \sqrt{3}}{9}$. COLLECT returns

Complex numbers with the HP49G - Part 2

$\frac{(45 + i \sqrt{6}) (4 + i) \sqrt{2}}{3^2}$. But pressing ->CPAIR returns what we want: $\frac{180 \sqrt{2} - 2 \sqrt{3}}{9}, \frac{45 \sqrt{2} + 8 \sqrt{3}}{9}$

Of course, you don't need to EXPAND or COLLECT first, if you want to use ->CPAIR. This was done in the examples for demonstration purposes only.

Now that we have a program that converts complex senns to a form that can be displayed as (X,Y) when flag -27 is clear, perhaps we could also write a program, that can factor a complex senn and convert each factor to a form that can be displayed as (X,Y). The idea is mainly to factor the senn and then use ->CPAIR for each factor. That means, we want to obtain all factors of the senn, say in a list, and then use DOSUBS, to apply the program ->CPAIR to each factor. The command FACTORS seems suitable here, because it returns a list of factors and multiplicities. We must of course watch out, because the list contains not only the factors but also the multiplicities. The command NSUB is a good help, because it returns the order number of the list element being processed. We know that FACTORS returns a list with the factors occupying the odd positions, while multiplicities occupy the even positions. So we must check if NSUB returns an odd number. If so, we apply ->CPAIR. If not then we raise the factor to the power given by the processed list element.

```
<<
FACTORS 1          @List of factors, multiplicities
<<
  IF NSUB 2 MOD    @If processed element in odd
                  @place
  THEN ->CPAIR     @Then apply ->CPAIR
  ELSE ->Q ^       @Else raise to power
  END
>>
DOSUBS
```

```
IF
  DUP SIZE 1 ==    @If list has only one element
THEN
  HEAD             @Then get this element
ELSE
  LIST             @Else product of elements
END
```

>>

Store this program in CPFACT. Note that we use ->Q to turn the multiplicity from a real number to an integer. If we don't use it then raising a factor to a power that is a real number, would return a number, which is not what we want. (We would turn the senn to a complex number this way.) Note also that parallel list processing allows also to use a list element in combination with previous elements. That is why raising the factor (previous element) to the power (current element) works. We can imagine that DOSUBS "sees" the list elements in a kind of virtual stack, the first element in the list occupying the highest level, the last occupying stack level 1. The command then just "combs" this virtual stack, allowing us to keep track of virtual stack level with the command NSUB. The picture on the next side may help to understand this better.

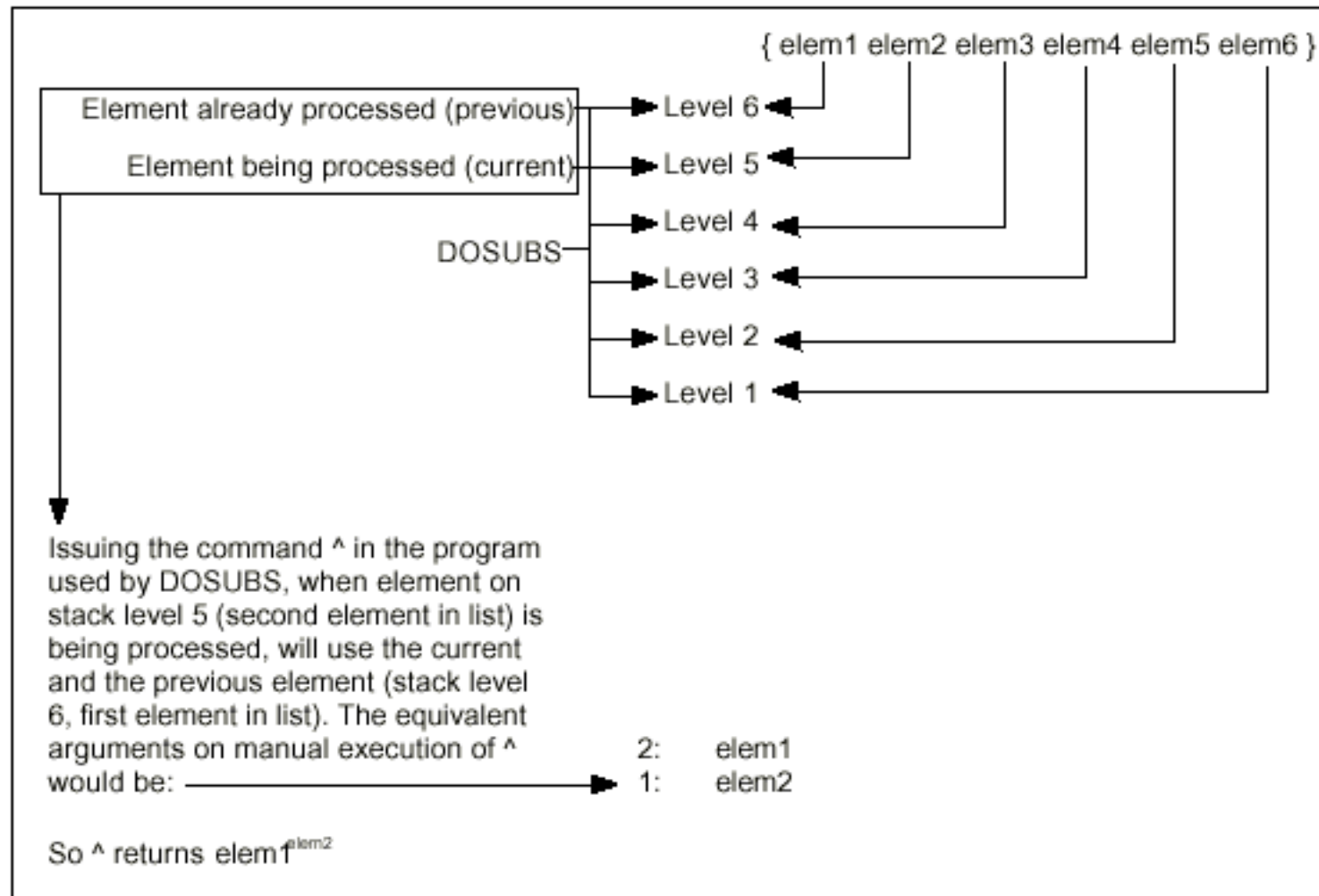
Let's try some examples. Enter $1 - i$ and press CPFACT. The result is $i^{(-1)} (1,1)$. Enter $\frac{1+i}{1-i}$ and press CPFACT. The result is i . Now a more

complicated one. Enter $(16 - 23 i) (25 + i \sqrt{3})$ and press CPFACT. The result $(23,16) (\sqrt{3}, -25)$ has also each factor displayed as (X,Y).

Enter $5 \sqrt{5 - \sqrt{9}} + \frac{2}{9} i \sqrt{3} (4 - i^3)$ and press CPFACT to obtain

the result $\frac{1}{9} (4 \sqrt{2}, \sqrt{2}) (45, \sqrt{6})$.

Complex numbers with the HP49G - Part 2



It might not be obvious, but something is strange here. The program returned $(23, 16)$ and $(\sqrt{3}, -25)$ as factors of $(16 - 23i)(25 + i\sqrt{3})$. But these are not the simplest factors that the HP49G can find for the used senn. (We'll see what *simple* means in this case later.) To understand this, enter $23 + 16i$ and then press FACTOR. The result is $i(2 - i)(11 - 6i)$, which shows that $23 + 16i$ is further factorable. Enter $(16 - 23i)(25 + i\sqrt{3})$ and press FACTOR again.

Complex numbers with the HP49G - Part 2

This returns $(16 - 23i) (-25i + \sqrt{3})$. Though FACTOR can factor $23 + 16i$ alone, it doesn't factor it when it is part of a bigger expression! The reason for this not very understandable behaviour is not known to me, but let's hope that some of the gurus out there will have mercy and tell us what is going on. Anyway, that doesn't mean that we accept this to be not avoidable in our program. If the factors are further factorable, then we factor the factors and see what happens. This is a good opportunity to use nested DOSUBS in our program. So let's make CPFACT better.

```
<<
  FACTORS 1      @Find factors
  <<            @Outer DOSUBS procedure starts
    IF
      NSUB 2 MOD      @If we have a factor
    THEN
      FACTORS 1      @Factor the factor
      <<            @Inner DOSUBS procedure starts
        IF
          NSUB 2 MOD      @If we have a power
          NOT
        THEN
          XQ            @Make it exact
          ^            @Raise factor of factor to
                      @the power
          ->CPAIR
        END
      >>            @Inner DOSUBS procedure ends
    DOSUBS
  IF
    DUP SIZE      @If list of factors of factor
    1 ==          @has only one element
  THEN
    HEAD          @Get that element
  ELSE
    LIST          @Else make product
  END
```

```
END
IF
  NSUB 2 MOD NOT@If we have a power
THEN
  XQ            @make it exact
END
>>            @Outer DOSUBS procedure ends
DOSUBS
IF
  DUP SIZE      @If list of factors
  1 ==          @has only one element
THEN
  HEAD          @Get that element
ELSE
  LIST          @Else make product
END
>>
```

Store this in CPFACT and try the previous examples again. For example if you clear flag -27, factoring $(16 - 23i) (25 + i\sqrt{3})$ with this version of CPFACT returns $i(2, -1) (11, 6) (\sqrt{3}, -25)$. Now the question is, will it be enough to find eventual factors of each factor of the complex senn? What if finding the factors of the factors of the senn, brings the same problems like finding the factors of the senn? Well, then we have to write CPFACT either with a loop, inside of which we factor the factors, then factor the factors of the factors of the factors, and so on, until nothing changes. Or we write the program so that it recursively calls itself, until no further factorisation can be achieved. Or we consider some of the properties of complex numbers and use our knowledge before we go further. Set flag -27 and let's go.

Many of you have undoubtedly already noticed, that we started factoring, without first asking what "factorisation" means for complex numbers. We know that some integers have no other divisors than themselves and 1, and such integers are called prime. Integers that do have other divisors than themselves and 1, are not prime. The nonprime integers can be written as products of prime integers. They

Complex numbers with the HP49G - Part 2

are factored in one and only one way. There are no integers that have the property of ambiguous factorisation in prime integers. In other words, there is only one product of prime integers that evaluated results in a given integer. Can these concepts be transferred to complex numbers? What is an integer when we regard complex numbers? What is a "prime complex number"? Is factorisation of "nonprime" complex numbers also unambiguous?

Let's start by giving the definition of complex integers, *Gaussian integers*² as they are called in mathematic language.

A complex number $a+bi$ is a Gaussian integer when a and b are integers.

Examples for such Gaussian integers are: $1+3i$, $2-i$ and so on. The norm of a Gaussian integer is defined by:

$$\text{norm}(a + bi) = a^2 + b^2$$

Now, there are Gaussian integers which are prime, we call them Gaussian primes. It may sound unfamiliar, but many things in the complex plain are quite a surprise, as we will see. Let's first give the definition of Gaussian primes.

Gaussian primes are Gaussian integers $z=a+bi$ having one of the following properties.

- 1. If both a and b are nonzero then, z is a Gaussian prime iff a^2+b^2 is an ordinary prime.**
- 2. If $a=0$, then bi is a Gaussian prime iff $|b|$ is an ordinary prime and $b \equiv 3 \pmod{4}$.**
- 3. If $b=0$, then a is a Gaussian prime iff $|a|$ is an ordinary prime and $a \equiv 3 \pmod{4}$.**

² Honouring Gauss who has done much in the field of complex numbers. The question regarding Gauss is, in which field known in his times he didn't do much.

The word "iff" is used here as an abbreviation for "if and only if".

Gaussian integers can be uniquely factored in terms of Gaussian primes up to powers of i and rearrangements.

Let's find some examples for Gaussian primes. The Gaussian integer $-5 + 2i$ is a Gaussian prime because $(-5)^2 + 2^2 = 29$ and 29 is an ordinary prime. The Gaussian integer $3 + 4i$ is not a Gaussian prime because $3^2 + 4^2 = 25$ and 25 is not an ordinary prime. Until now the astonishment is kept in medium ranges, but consider the following example. The ordinary prime 11 is a Gaussian integer of the form $z = a + bi$, with $b = 0$. It is also a Gaussian prime because 11 is an ordinary prime and $11 \equiv 3 \pmod{4}$, which means that $11 - 3$ is divisible by 4. But the ordinary prime 13 is not a Gaussian prime, because $13 - 3$ is not divisible by 4. This means that our old known prime number 13, is no more a prime in the complex word! It is factorable in terms of Gaussian integers!

This quite unexpected fact brings many mathematical and HP49Gical questions. One of the HP49Gical questions is: Has the HP49G any built-in function to factor ordinary primes that are not Gaussian integers? Ha! Of course it has. Enter 13, press [left shift], [1] and then the menu key [INTEG]. Press [NXT] to go to the next page of the menu. Now you should see PA2B2 over the key [F5]. Press [F5] and you get the result $3 + 2i$. This result means $13 = (2 + 3i)(2 - 3i)$, or in other words that the ordinary prime number 13 can be factored with the Gaussian prime $3 + 2i$ and its complex conjugate Gaussian prime $3 - 2i$. To verify this: Press [ENTER] to make a copy and use CONJ to get $3 + 2i$ on stack level 1. Press [*] and [EXPAND]. You get 13 !! The command PA2B2 will error with "Z is not = 1 mod 4" if you give it an ordinary prime that is also a Gaussian prime. Enter 11 and press PA2B2 to see that. This also gives us the information, that the "other" ordinary primes n which are not Gaussian primes, all have the common property $n \equiv 1 \pmod{4}$, that means that the difference $n - 1$ is divisible by 4. If you give PA2B2 an odd integer which is not

Complex numbers with the HP49G - Part 2

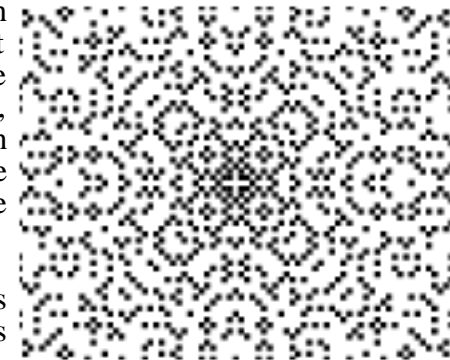
even an ordinary prime, like for example 21, then it errors with "Z is not prime". But for even integers which are not even ordinary primes, it errors again with "Z is not = 1 mod 4". This can be considered as a minor bug, because the problem in this case is that we don't even have an ordinary prime, but it is "not a very buggy bug".

If you have time enough (about one hour), let's make a plot of Gaussian primes on the complex plane. We want the HP49G to check all Gaussian integers $a + bi$ with a from -65 to 65 and b from -31 to 32.

```
<<
{ (-65.,-31)      @PICT lower left corner
  (65.,32.)       @PICT upper right corner
  X 0. (0.,0.)    @indep var, resolution, plot origin
FUNCTION Y } @plot type, depend. var.
'PPAR' STO        @Store plot parameter. For our program
                  @only the lower left corner, the upper
                  @right corner and the plot origin are
                  @important.
ERASE              @Erase any previous drawings.
{ #0 #0 }         @Display PICT starting at Pixel
PVIEW             @coordinates { #0 #0 } (upper left)
-65 65
FOR I              @X-coordinate loop
  -31 32
  FOR J            @Y-coordinate loop
    IF
      I 0 ==      @Criterium 2 for
      J ABS ISPRIME? AND @Gaussian primality
      J 3 - 4 MOD NOT AND
      J 0 ==      @Criterium 3 for
      I ABS ISPRIME? AND @Gaussian primality
      I 3 - 4 MOD NOT AND
      OR
      I SQ J SQ +  @Criterium 1 for
      ISPRIME?     @Gaussian primality
      OR
```

```
      THEN
        I J R->C    @Build complex (I,J)
        PIXON       @Turn on the pixel at (I,J)
      END
    NEXT
  NEXT
7 FREEZE           @Let the display persist after
>>                @program comes to an end.
```

Store the program in 'GAUSSPLOT' and run it. (It takes about one hour to run.) The HP49G displays an empty screen, and gradually plots each Gaussian integer it finds. By the time a nice pattern build-up, which in the middle of the plot looks like:



The first few ordinary primes which are also Gaussian primes are 3, 7, 11, 19, 23, 31, 43, ...

To speed up the plot you can of course use a smaller region to plot, by restricting the loops of the X- any Y-coordinates to smaller values, say I and J from -20 to 20.

The same plot could be made with the built-in plot type TRUTH, I as the independent variable, J as the dependent variable and the program:

```
<<
I 0 ==
J ABS ISPRIME? AND
J 3 - 4 MOD NOT AND
J 0 ==
I ABS ISPRIME? AND
I 3 - 4 MOD NOT AND
OR
I SQ J SQ +
```

Complex numbers with the HP49G - Part 2

ISPRIME?

OR

>>

stored in 'EQ'. But we must be careful then. The plot ranges and the resolution must be such, that the HP49G calculates integer values as X- and Y-coordinates, "integer values" meaning reals without any fractional parts. When you for example have a plot range for the X-coordinate going from -10 to 10, and a resolution of 1., then the values -10,-9,...,9,10 will be used as X-coordinates. But when the X-range is, say from -10 to 10, and the resolution is 2.5, then the values -10,-7.5,...,7.5,10 will be used as X-coordinates. As you can see there are values which will be not appropriate for example for checking if they are prime.

Now that we know what complex factorisation in terms of Gaussian primes means, let's examine some examples. Enter $9 - 8i$ and press

FACTOR. The result is $\frac{(2+i)(5+2i)}{i}$. $2+i$ and $5+2i$ are

Gaussian primes. Enter $-4 - 8i$ and press FACTOR to obtain the

expression $\frac{(1+i)^4(2-i)}{i^3}$. And now another unexpected thing: Enter

$145 + 58i$ and press FACTOR. Boom! The HP49G errors with "Bad Argument Value"!!. Try to FACTOR $10 - 5i$. The same error again.

The same mysterious behaviour for $39 + 26i$, $68 - 17i$. What is so different for these Gaussian integers? What makes them special, so special that the HP49G not only can't factor them, but it also errors out, when we tell it to factor them? Well, let's look at the example $10 - 5i$. You can immediately see that it is divisible by 5, which

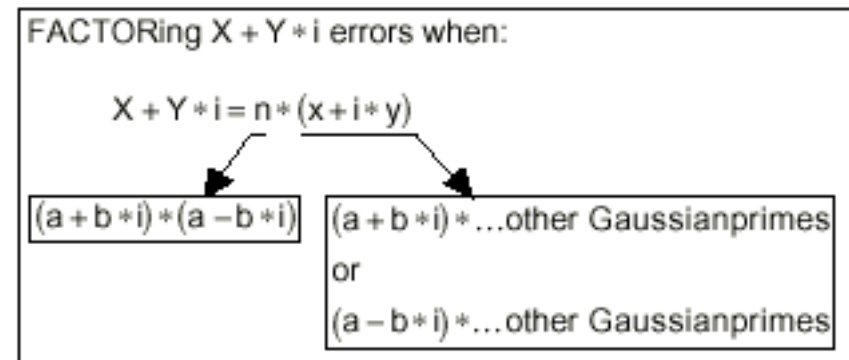
means $10 - 5i = 5(2 - i)$. Now remember that 5 isn't a Gaussian prime. If you enter 5 and press PA2B2, you get $2 + i$, which shows that 5 can be factored by means of Gaussian primes to $(2+i)(2-i)$.

That also means that $10 - 5i = (2+i)(2-i)(2-i)$ or

$10 - 5i = (2+i)(2-i)^2$. Here you can see the general rule for this

behaviour. When FACTOR errors out trying to factor a Gaussian

integer, then the factorisation contains a Gaussian prime factor of an ordinary prime raised to some power *and* additionally the complex conjugate of the Gaussian prime factor of the ordinary prime. The following picture may turn this more understandable:



And how does FACTORS behave in such cases? Well, let's try. Enter $10 - 5i$ and press FACTORS to get the list

$\{i \quad -3 \quad 2+i \quad 1.5 \quad 2-i \quad 1.5\}$. We are ready to scream

"Hurrahhhh!", but unfortunately this result is wrong. The broken exponents make us suspicious. We can't go on for ever with suspicious minds, so we should prove this. Press XQ to turn the decimal exponents to ratios. The list now turns to

$i \quad \frac{-3}{10} \quad 2+i \quad \frac{3}{2} \quad 2-i \quad \frac{3}{2}$. Enter a 1 and then the program:

```
<<
IF
  NSUB 2 MOD NOT
THEN
  ^
END
>>
```

Then press DOSUBS to raise each factor to its corresponding power.

Complex numbers with the HP49G - Part 2

The resulting list is:

$$\sqrt[10]{i^{(-3)}} \quad \frac{(1-2i+i\sqrt{5})\sqrt{4+2\sqrt{5}}}{2}^3 \quad \frac{(1-2i-i\sqrt{5})\sqrt{4+2\sqrt{5}}}{2}^3$$

Press LIST to convert this to a product and then EXPAND to get $\frac{5\sqrt{5}}{\sqrt[10]{i^3}}$. Press TSIMP to convert this complex to $\frac{5\sqrt{5}}{e^{\frac{3i}{20}}}$. Take this to

the EQW, select the denominator and press LIN. Press ENTER. Now you have $\frac{5\sqrt{5}}{e^{\frac{3i}{20}}}$ on stack level one, which shows that you have a

complex number with absolute value $5\sqrt{5}$ and phase (angle) $-\frac{3}{20}$.

Is this correct? Enter $10-5i$ again, press ENTER to make a copy and press ABS to find the magnitude. The HP49G returns $5\sqrt{5}$. OK, the magnitude is correct. SWAP and press ARG. The result is

$\text{ATAN} \frac{-1}{2}$. Oops! If you press \rightarrow NUM, you get the number -

.463647..., but $-\frac{3}{20} \rightarrow$ NUM returns -.471238..., quite near but not

equal to the correct angle $\text{ATAN} \frac{-1}{2}$! EXPAND the product

$$\frac{(1-2i+i\sqrt{5})\sqrt{4+2\sqrt{5}}}{2}^3 \quad \frac{(1-2i-i\sqrt{5})\sqrt{4+2\sqrt{5}}}{2}^3,$$

(the last two factors returned by the command FACTORS for the Gaussian integer $10-5i$), the result is $5\sqrt{5}$. That means that the product of these factors returns the correct magnitude of $10-5i$. The

remaining factor $\sqrt[10]{i^{(-3)}}$ must thus be the part that determines the angle of $10-5i$, which is directly found (using ARG) to be equal to

$\text{ATAN} \frac{-1}{2}$. If you enter $\sqrt[10]{i^{(-3)}}$ and press TSIMP LIN you get $e^{\frac{3i}{20}}$,

which clearly shows that the *first factor in the list is the reason for the wrong angle of the factored number*. The factorisation of such problematical Gaussian integers *errors out when you use FACTOR or returns factors that form the original Gaussian integer but out of phase when you use FACTORS*. We again find something that should have been said right from the start in the description of these commands. It is at the boundaries of irresponsibility to omit such critical parts in the documentation, because especially the behaviour of FACTORS, which silently gives the wrong answer, can lead to catastrophic results. I wouldn't like to be an engineer that calculates the wrong answers with his HP49G in his pocket, only because the ladies and gentlemen at HP think that everybody must be also a CAS expert, when (s)he wants to use their products.

Enough yelling! Now that we now the problem, let's make a program that gives the right answers. The idea is to check first if the command FACTOR errors. If so, we know that the Gaussian integer is divisible by an ordinary prime that is not a Gaussian prime at the same time. We search to find this ordinary prime and we divide the given Gaussian integer by it. Then we try to FACTOR the result of the division. Again, if FACTOR errors, we know that there must be another ordinary but not Gaussian prime in the factors of the given Gaussian integer. We repeat the procedure until FACTOR doesn't error, in which case we use FACTORS to find all factors and multiplicities. For such cases like $(23+16i)(-(25i)+\sqrt{3})$, we know that $23+16i$ wouldn't be factored, though it can be factored to $i(2-i)(11-6i)$. We thus wrap the whole in a loop and apply repeatedly to each found factor, until nothing changes. We are going to split it in several programs. Enter first the following program:

Complex numbers with the HP49G - Part 2

```

<<
RCLF 1 {} -> z flags n f
<<
  -55 CF          @Enable last arguments recovery
  WHILE
    "Factoring" 1 DISP
    IFERR
      z FACTOR    @Does FACTOR error out?
    THEN
      DROP        @Drop z, return 1. to keep on
      1.          @searching prime divisors
    ELSE
      FACTORS XQ   @Find list of factors and
      1           @multiplicities
      <<          @Convert them to list
      IF          @of factors
        NSUB 2 MOD
        NOT
      THEN
        ^
      END
      >> DOSUBS
      'f' STO+ 0.  @Put found factors in list f,
                  @return 0. to stop search.
    END
  REPEAT
    DO            @Outer DO-Loop
      DO          @Inner DO-Loop
        n NEXTPRIME @Find next prime, store in n
        'n' STO
        "Searching n: " @Show current n
        n + 1 DISP
      UNTIL      @Until we have a prime that
        n 1 - 4 MOD @is no Gaussian prime
      NOT
    END          @Inner DO-Loop ends
    IF
      "Check " n + @Show current checked prime
      2 DISP
    THEN
      z n IDIV2    @Integer division z/n
      ->NUM 0. ==  @If z divisible by n
    THEN
      "Positive"
      3 DISP
      'z' STO      @Store result of division in z
      n PA2B2      @Factor n in terms of
      DUP CONJ     @Gaussian primes
      2 ->LIST
      'f' STO+     @Put in factors list f
      'n' 1 STO-   @Subtract 1 from n.
                  @We do that because perhaps z
                  @is again divisible by n. This
                  @way the next NEXTPRIME will
                  @return the same n again, and so
                  @we check again divisibility.
                  1. @To exit the inner DO-Loop
    ELSE
      DROP        @Drop the result of division
      "Negative"
      3 DISP
      0.          @To remain in the inner DO-Loop
    END
  UNTIL
  END            @Outer DO-Loop ends
  END            @While-Loop ends.
  f             @Return factor list
  flags STOF     @Store flags
  >>
  >>
  <<
    1 ->LIST      @Wrap senn in list
    DO

```

Store this in GF. The program takes a senn from the stack and returns a list with its factors. Now we need another program that runs GF in a loop until the list of factors doesn't change.

Complex numbers with the HP49G - Part 2

```

    DUP
    << GF >> MAP      @Apply GF to each element
    1 << OBJ-> DROP >> @Explode each sub list so that
    DOSUBS             @the factors list remains flat
UNTIL
    DUP2 SAME          @Until nothing changes
    ROT DROP
END
IF
    DUP SIZE 1. ==     @If only one element in list
THEN
    HEAD               @Then take it
ELSE
    LIST               @Else product of elements
END
>>

```

Store this program in GFACTOR. Enter now $(16 - 23i)(25 + i\sqrt{3})$ and press GFACTOR. The result $i(2 - i)(11 - 6i)(25 + i\sqrt{3})$ shows that it worked. Enter $\frac{(1575 + 1350i)\sqrt{2} - (60 - 70i)\sqrt{3}}{9}$ and press GFACT. After some seconds you get the result:

$$\frac{1}{9} (2 + i) (4 + i) \sqrt{2} 5 \sqrt{3} (i \sqrt{2} + 15 \sqrt{3}).$$

Now let's think again about the reason of this discovery journey. We started with a program ->CPAIR that converts any given senn to the form $a + bi$, because we wanted any senn to be displayed as (a,b) when flag -27 is clear. Then we wanted to display also each factor of a factored senn the same way, so we started programming CPFACT. Then we discovered some strange behaviour of the command FACTOR, when it factors for example $(23,16)$ alone or as a part of a product of senns. Then we wandered for the first time what factoring actually means for complex numbers, and so we found a new world full of unexpected things, waiting to be discovered. Now we return

from this journey, with knowledge that can be used for a new CPFACT, that behaves better than the old. (The old would give wrong answers for, say $10 - 5i$, which is even worse than to error out.) We can use mainly the same method as in GF but in addition we use the algorithm used in ->CPAIR for each factor, to convert it to the form $a + bi$, which can be displayed as (a,b) , when flag -27 is clear. Store the program:

```

<<
    DUP RE SWAP IM i * +
    { '&A-&B*i' '(&A,-&B)' } MATCH DROP
    { '&A+i' '(&A,1)' } MATCH DROP
>>

```

in ->CP. Then enter the following program:

```

<<
    RCLF 1 {} -> z flags n f
    <<
        -55 CF          @Enable last arguments recovery
    WHILE
        "Factoring" 1 DISP
    IFERR
        z FACTOR        @Does FACTOR error out?
    THEN
        DROP            @Drop z, return 1. to keep on
        1.              @searching prime divisors
    ELSE
        FACTORS XQ       @Find list of factors and
        1                @multiplicities
        <<               @Convert them to list
            IF            @of factors
                NSUB 2 MOD
                NOT
            THEN
                ^
            ELSE

```

Complex numbers with the HP49G - Part 2

```

->CP          @Convert to a+b*i
END
>> DOSUBS
'f' STO+      0. @Put found factors in list f,
END           @return 0. to stop search.
REPEAT
DO            @Outer DO-Loop
DO            @Inner DO-Loop
n NEXTPRIME   @Find next prime, store in n
'n' STO
"Searching n: " @Show current n
n + 1 DISP
UNTIL         @Until we have a prime that
n 1 - 4 MOD   @is no Gaussian prime
NOT
END           @Inner DO-Loop ends
IF
"Check " n +  @Show current checked prime
2 DISP
z n IDIV2     @Integer division z/n
->NUM 0. ==   @If z divisible by n
THEN
"Positive"
3 DISP
'z' STO       @Store result of division in z
n PA2B2       @Factor n in terms of
DUP CONJ      @Gaussian primes
2 ->LIST
<< ->CP >>
MAP           @Convert to a+b*i
'f' STO+      @Put in factors list f
'n' 1 STO-    @Subtract 1 from n.
              @We do that because perhaps z
              @is again divisible by n. This
              @way the next NEXTPRIME will
              @return the same n again, and so
              @we check again divisibility.
1.           @To exit the inner DO-Loop
              >>
ELSE
DROP         @Drop the result of division
"Negative"
3 DISP
0.           @To remain in the inner DO-Loop
END
UNTIL        @Outer DO-Loop ends
END           @While-Loop ends.
f            @Return factor list
flags STOF    @Store flags
>>

```

Store the program in GFCP. It is essentially the same as GFACTOR with the addition of ->CP to convert each factor to the form $x+yi$

Now, store the following program in CPGFACT:

```

<<
1 ->LIST      @Wrap senn in list
DO
DUP
<< GFCP >> MAP @Apply GF to each element
1 << OBJ-> DROP >> @Explode each sub list so that
DOSUBS        @the factors list remains flat
UNTIL
DUP2 SAME     @Until nothing changes
ROT DROP
END
IF
DUP SIZE 1. == @If only one element in list
THEN
HEAD          @Then take it
ELSE
LIST          @Else product of elements
END
>>

```


Complex numbers with the HP49G - Part 2

Let's try it. Clear flag -27. Enter $10 - 5i$ and run the program. The result is $(2, -1) (2, 1) (2, -1)$. (Now, if you like, you could think of additional code that returns this as $(2, -1)^2 (2, 1)$.) Try also to factor $(16 - 23i) (25 + i \sqrt{3})$ with CPGFACT. The result should be $i (2, -1) (11, 6) (\sqrt{3}, -25)$.

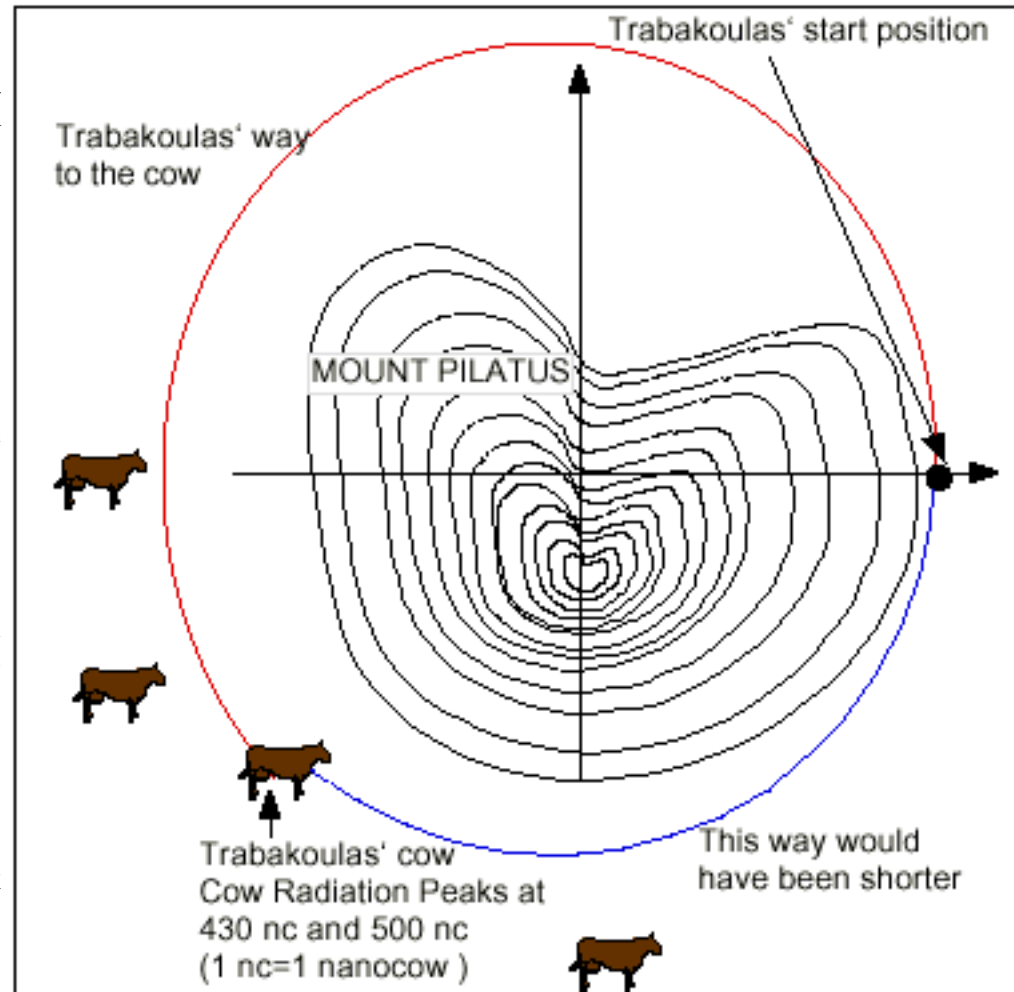
Let us now listen to what Trabakoulas has to say, because yesterday he complained, that he found the coordinates of a cow that was lying on the complex plane, but her angle was in the wrong interval. You remember the formula given in part 1 for the angle of a complex quantity? Of course you do, but I write it here again:

$$= \arctan \frac{y}{x} + 1 - \frac{x}{|x|} \frac{2}{2}$$

Well, this is the general formula used by the HP49G to find the angle when the provided arguments are symbolic arguments, to which senns also belong. Trabakoulas used his HP49G which is equipped with a CRR (Cow Radiation Receiver by Rrobotics®) and measured the x and y coordinates of the cow to be 1 km westwards and 1 km southwards from his position. Now, Mount Pilatus stood between him and the cow, so he decided to find her angle and go around the mountain, instead of climbing up from the one side and down from the other. (Greek genius!) He entered $-1 - i$ and pressed ARG. The HP49G returned $\frac{\pi}{4} + 2\pi$. He

pressed EXPAND and then he got $\frac{5\pi}{4}$. So he started walking counter clockwise and he reached the cow. (Greek genius, did I say that? ;-)) But the next day he thought that he would have also make it to the cow, if he had walked

clockwise to the angle $-\frac{3\pi}{4}$. Because the paths on the mountains can be really difficult to walk, he was very angry that the HP49G didn't give the answer in the interval from $-\pi$ to π . And he got even angrier when I said to him that he would get the angle $-\frac{3\pi}{4}$ if he had enter



Complex numbers with the HP49G - Part 2

the coordinates not as the senn $-1-i$ but as the complex number (type 0.) $(-1., -1.)$. "What do complex numbers have, that senns don't?", he asked. (Remember what Senn means? ;-)) He was about to change profession, but thanks goodness Alban Schann proposed along with other really nice things an alternative formula which returns angles in this interval:

$$= 2 \operatorname{ATAN} \frac{Y}{X + \sqrt{X^2 + Y^2}}$$

But unfortunately the simple program

```
<<
  DUP RE SWAP IM -> X Y
  '2*ATAN(Y/(X+ (X^2+Y^2)))'
>>
```

will not work for coordinates $x < 0$ and $y = 0$. (If the cow is exactly at the opposite side of Mount Pilatus, that is.) So we need a program that works in any case. (Or at least in any possible case considering cow coordinates.)

```
<<
EXPAND      @This will turn senns like -1+0*i to
             @integers like -1
IF
  { 28. 0. 1. } @If input is integer, real, or
  OVER TYPE POS @complex
THEN
  ARG      @Then simply use ARG, as it returns the
           @angles in the right interval for such
           @arguments
ELSE
  DUP RE SWAP IM -> X Y @Else use Alban's idea
  '2*ATAN(Y/(X+ (X^2+Y^2)))'
  EXPAND
  IF      @If result contains ATAN
```

```

  DUP ->LST
  { ATAN } HEAD
  POS
THEN      @then check if   is part of the result
  DUP ->NUM
  ->Q EXPAND
  IF      @If algebraic object   (!) is there
  {      } ->ALG
  DUP 2 LIST
  MATCH
THEN
  NIP
ELSE
  DROP
END
END
END
>>
```

We store this in ALARG, honouring Alban for his idea. Perhaps you wonder why we don't simply check if is in the list that is returned by `->LST`. Well, it doesn't work. Though you can see the in the list, entering and pressing POS returns 0! Matching simply with `{ }` also doesn't work because you get "bad argument type". We have to make an algebraic object that only contains and then put it twice in the list, which is used by the command `MATCH`. Matching with this list doesn't change the algebraic object because is replaced by , but it returns a 1, if some existed in the algebraic, or 0 if none existed, which can be used to decide if we keep the result with or with ATAN. Store the program in ANGLE and test it with some integers, reals, complex numbers or senns. The program returns numeric results for numeric inputs and symbolic results for symbolic inputs. I think Trabakoulas will remain a Senn. ;-)

You of course already noticed that the programs in this part of the complex marathon use often the commands DOSUBS and MAP. Both commands apply some procedure to all elements of some composite

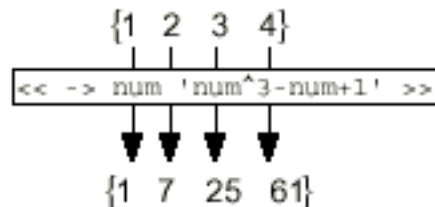
Complex numbers with the HP49G - Part 2

objects. But what are the differences? Let's take a more detailed look.

The command DOSUBS, as already said, applies a procedure, a program, to each group of n elements of a list. It takes the list from stack level 3, the number of elements n in each group from stack level 2 and the program from stack level 1. First of all, you can see that even programs can be arguments for commands. Let's make an example. Suppose you have the list of numbers $\{1\ 2\ 3\ 4\}$ and you want to find for each of them $\text{number}^3 - \text{number} + 1$. Enter this list. Now, think as if you had only one of these numbers, say 3, and you wanted to find $3^3 - 3 + 1$. You need only one argument for the program that will calculate this, namely the number itself. Thus the number of elements in each group is 1. Enter 1. Now how would you program the calculation? One of the many ways would be:

```
<< -> num 'num^3-num+1' >>
```

Enter this program. Now press DOSUBS. The result is the list $\{1\ 7\ 25\ 61\}$ which contains the results of the program, applied to each number contained in the original list.



A second example using three elements for each group. Suppose you have the list of algebraic objects $\{X\ X^2\ X^3\ X^4\ X^5\}$ and you want to find $\frac{\text{elem}_m}{\text{elem}_{m+2}}$ for each three adjacent elements in the list. Enter the list. You need three elements, so the number of elements in each group is 3. Enter 3. Now think as if you had only one such group of three adjacent elements, say elem2, elem3 and elem4 on the stack. You would write a program like for example:

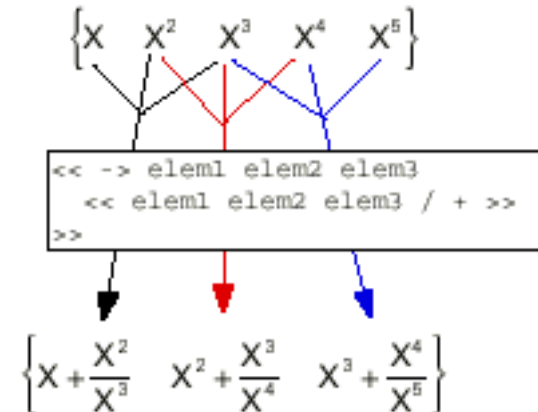
```
<< -> elem1 elem2 elem3
```

```
<< elem1 elem2 elem3 / + >> >>
```

Enter the program and press DOSUBS. The resulting list is:

$$X + \frac{X^2}{X^3} \quad X^2 + \frac{X^3}{X^4} \quad X^3 + \frac{X^4}{X^5}$$

DOSUBS applied the program to the group of elements X, X^2, X^3 , then to the group of elements X^2, X^3, X^4 and then to the group of elements X^3, X^4, X^5 , when it reached the last element and stopped processing.



It is not even necessary for the program to return only one result. Suppose you want to find the absolute value and the angle for each element in the following list of complex quantities

$(1,0) \ (1,1) \ 3-2i \ 2e^{i\frac{\pi}{3}}$. Enter the list, then enter 1 and then

enter the program:

```
<<
  DUP ABS SWAP ARG
>>
```

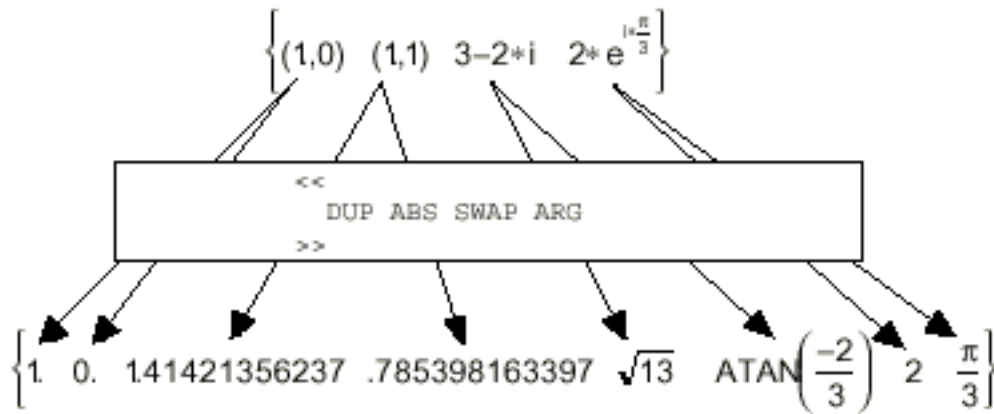
Now press DOSUBS. After a while you get the resulting list:

1. 0. 1.41421356237 .785398163397

$$\sqrt{13} \quad \text{ATAN} \frac{-2}{3} \quad 2 \quad \frac{\pi}{3}$$

The program run for each element and returned its absolute value and its angle, as the picture on the next page shows.

Complex numbers with the HP49G - Part 2



The commands NSUB and END SUB are related to the command DOSUBS. The command NSUB returns the number of the group of elements being actually processed. The command END SUB returns the total number of groups in the list. Both numbers are returned as real numbers on the stack. Both commands work only inside the program that is applied to each group of elements. Suppose for example that you have a big list, and for each element you calculate its sine. If you enter the big list, then a 1, the program:

```
<<
  NSUB " of " + END SUB +
    1 DISP
  SIN
>>
```

and press DOSUBS, then the HP49G will display "n of m" at the top line of the screen, n being the number of the element being processed and running from 1 to the number of elements in the big list, and m being the number of elements in the list. Of course NSUB and END SUB can also be used for other purposes, like for example to decide what should be done according to the position of a group in the list.

When a list element is itself a list, then it is treated as a whole. DOSUBS holds at the current nesting level and doesn't also examine group of elements of sub lists. If for example you have the list $\{\{1\ 2\ 3\}\ \{3\ 4\ 5\}\ \{4\ 5\ 6\}\}$ and you enter 1 and then the program `<< 5 POS >>`, then DOSUBS will return $\{0\ 3\ 2\}$, a list containing the positions of 5 in each sub list. But if you enter the list $\{\{1\ 2\ 3\}\ \{3\ 4\ 5\}\ \{4\ 5\ 6\}\}$, then a 1 and then the program `<< ->STR >>`, then the result is the list $\{"1\ 2\ 3"\ "3\ 4\ 5"\ "4\ 5\ 6"\}$, in which each sub list is turned to a string. The result is not $\{"1" "2" "3"\ "3" "4" "5"\ "4" "5" "6"\}$, in which each element of the sub lists is turned to a string. If you want to use DOSUBS and to go a step downwards in the nesting, then you must nest DOSUBS. An example for such a nesting using the same list like above: Enter the list $\{\{1\ 2\ 3\}\ \{3\ 4\ 5\}\ \{4\ 5\ 6\}\}$, then 1 and then the program:

```
<< 1 << ->STR >>
  DOSUBS
>>
```

Now, press DOSUBS. The result is the list:

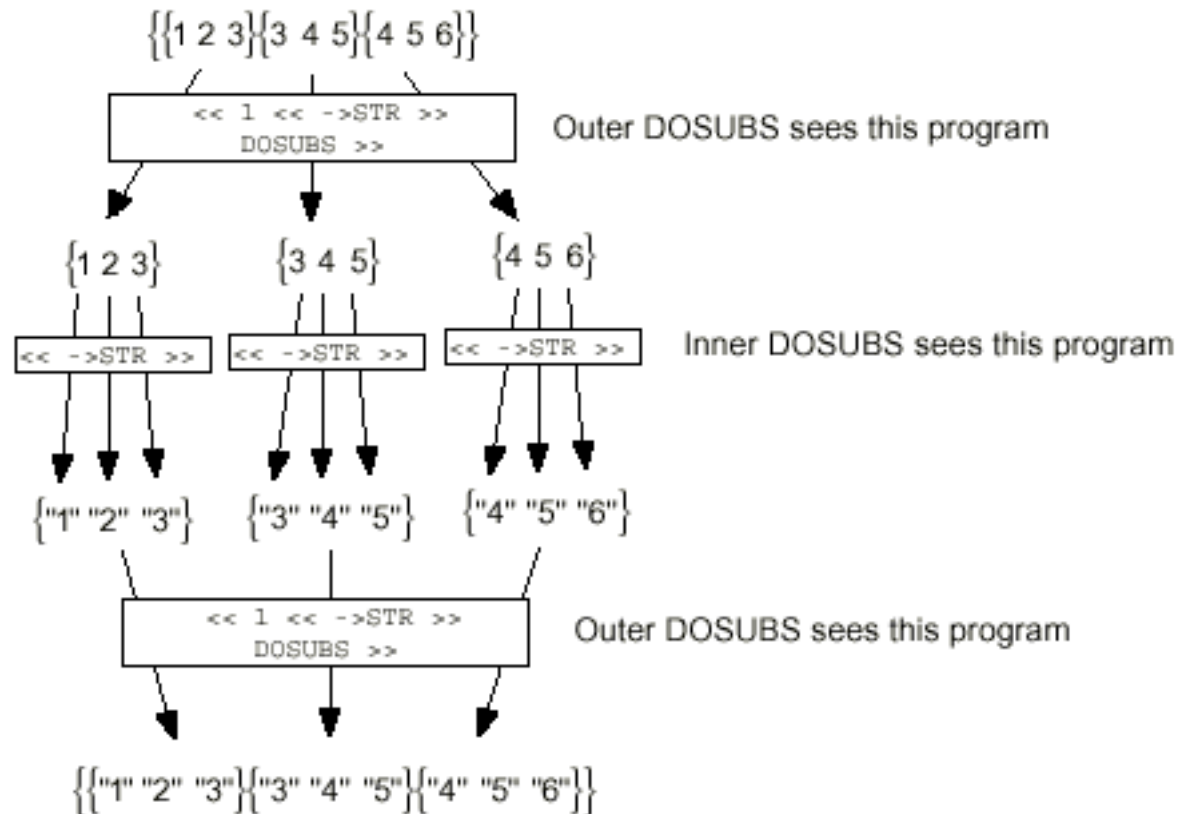
```
{{"1" "2" "3"} {"3" "4" "5"} {"4" "5" "6"}}
```

How did that work? The DOSUBS that you invoke by pressing the key, applied the program to each element of the list, that is it took each sub list and passed it to the program. The program received each time a list, entered a 1, entered the program `<< ->STR >>` and did another DOSUBS, this time to the sub list. The picture on the next page demonstrates this.

It should be noticed here that many times using DOSUBS is not necessary. Many commands know how to process lists up to the first level of nesting. Suppose for example that you want to find the square

Complex numbers with the HP49G - Part 2

root of each element of the list {1 2 3 4 5}. You don't need to use DOSUBS, because the built-in command `√` can also handle lists, returning the list of square roots of each element: {1 √2 √3 2 √5}. But it does only up to the first nesting level. That means that entering {{1 2 3 4 5} {2 3 4 5 6}} and pressing [], will error out with "Bad Argument Type" because the command `√` received the elements of the first nesting level, that is the elements of the list {{1 2 3 4 5} {2 3 4 5 6}} which are themselves lists. But if you enter the list {{1 2 3 4 5} {2 3 4 5 6}}, then a 1, then the program `<< >>` and then press DOSUBS, then the result will be the list {{1 √2 √3 2 √5} {√2 √3 2 √5 √6}}. The DOSUBS command applies the program `<< >>` to each sub list. The program itself contained the command `√`, which got a list of integers as argument, and so it returned the list of the square roots. The picture on top of the left column of the next page demonstrates this.



The behaviour of DOSUBS allows easy list flattening when it is used in combination with recursively program calling. Consider the program:

```
<<
  1 CF
  1
  <<
    IF
      DUP TYPE 5. ==
    THEN
```

```
      OBJ-> DROP 1 SF
    END
  >>
  DOSUBS
  IF
    1 FS?
  THEN
    FLATTEN
  END
>>
```

Complex numbers with the HP49G - Part 2

Store this in FLATTEN. It flattens a list with any nested sub lists, which themselves can contain sub lists, and so on. Any combination of nesting levels will be flattened to a single list that contains all elements, sub elements, sub sub elements, and so on.

Note also that DOSUBS will not process arrays. But you can convert an array to a list of lists using AXL. Then you can process the list as you wish, and reconvert to an array using AXL again. But one thing must be said here. The re-conversion of the list to an array using AXL, will produce always a symbolic array (type 29.) even if you started with a real array (type 3.) or a complex array (type 4.) If it is important not to lose the type of the array, then you can use the command OBJ-> or ARRY-> to explode the array to its elements, ->LIST to construct the list out of the elements and then OBJ-> or LIST-> to explode the list and ->ARRY to reconstruct the array.

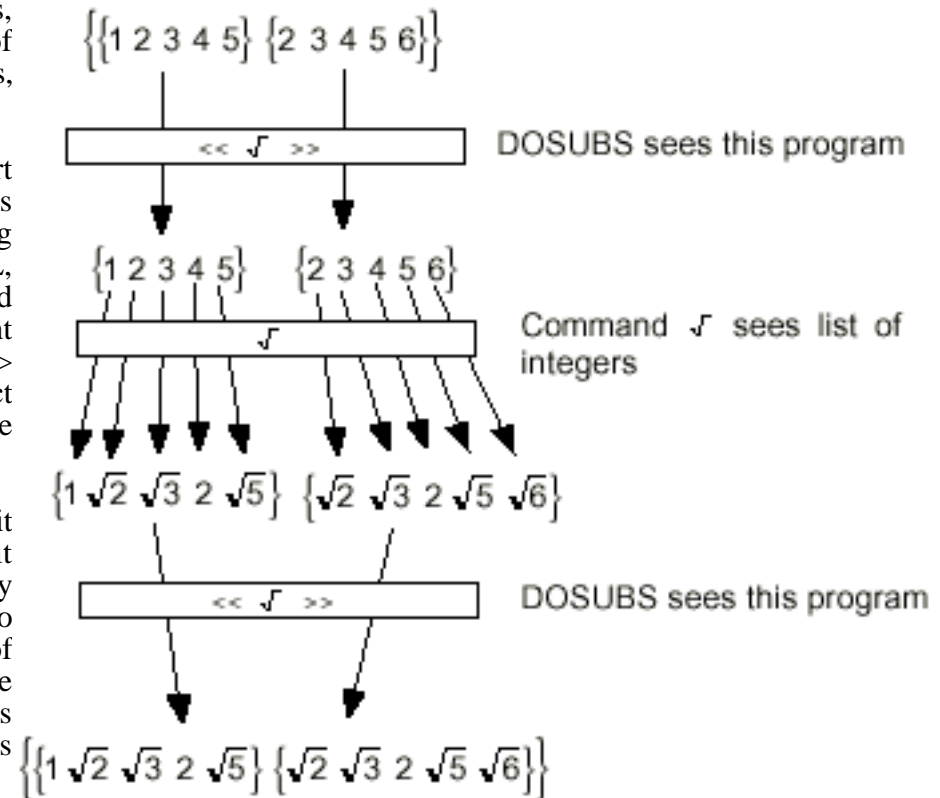
On with MAP. This command is similar to DOSUBS because it processes elements of lists. But it has also differences. First of all, it will process also array elements. The command takes a list or any array from stack level 2 and a program from stack level 1, which it applies to each list or array element. As you already see there is no number of elements which are going to be processed as a group. This is because the command automatically goes through the nesting levels until it finds something that is not a list or an array, and then attempts to process this something.

A first easy example: Enter the list {1 2 3} and then the program:
`<< SQ >>`

Press MAP. The result is the list {1 4 9}, a list with the squares of each number.

Now enter {1 2 {3 4}} and the same program `<< SQ >>` and press MAP. The result is the list {1 4 {9 16}}. MAP "went down" through all the nesting levels and returned a list with the results and with exactly the same structure.

Since MAP always goes through all nesting levels, you can't process



eventually contained sub lists as a whole. And you also can't use it to process contained elements as groups. A program that needs more than one arguments can't use the appropriate number of list elements as arguments. However if all but one arguments for the program are constant, then the program itself can of course put those constant arguments on the stack and then continue processing. If you for example want to multiply each list element of the list {1 4 {9 16}} with 2, then enter the list and the program `<< 2 * >>` and then press MAP. The result is {2 8 {18 32}}.

When processing lists with MAP, then the used program can take only

Complex numbers with the HP49G - Part 2

one argument at a time, but it can return more than one results. If more than one results are returned, then they are put in a list, which replaces the element from which the results are achieved. For example, if you enter the list $\{2\ 8\ \{18\ 32\}\}$, the program

`<< DUP 2 / >>` and press MAP, then the result is $\{\{2\ 1\}\ \{8\ 4\}\ \{18\ 9\}\ \{32\ 16\}\}$.

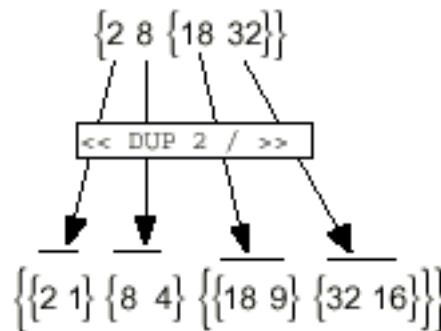
Don't try to explode the additional lists by adding OBJ-> DROP at the end of the program that is used as argument for MAP. Remember, the program will be used to each element of the list. So for example entering $\{(1.,0.)\ (0.,1.)\}$, then the program `<< DUP 2 / OBJ->`

`DROP >>` and then pressing MAP, doesn't return the list $\{(1.,0.)\ (.5,1.)\ (0.,1.)\ (0.,.5)\}$ but the list

$\{\{(1.,0.)\ .5\}\ \{(0.,1.)\ 0.\}\}$. How does this happen? Well, think

again of a list element alone, say $(1.,0.)$, on stack level 1. The program issues DUP, so you have $(1.,0.)$ on stack level 2 and on stack level 1. Then the program says 2 / so you have $(1.,0.)$ on stack level 2 and $(.5,0.)$ on stack level 1. Then the program says OBJ-> and this command is applied to the element on stack level 1, which is $(.5,0.)$,

so the stack goes to $(1.,0.)$ on stack level 3, $.5$ on stack level 2 and $0.$ on stack level 1. The command DROP drops the $0.$ from stack level 1 and so $(1.,0.)$ comes to stack level 2 and $.5$ to stack level 1. Now the program has finished and the two objects $(1.,0.)$ and $.5$ are on the stack. They are wrapped in a list, because the output was more than one object. The list is put where the first processed element, the complex number $(1.,0.)$ was in the original list, namely at position 1. The same is repeated with the second element of the original list.



MAP can also apply a program to each element of an array. But the

result of the program must be a valid matrix element. If an invalid matrix element is created by the program, then MAP errors with "Bad Argument Type". That also means that the program should return only one object because otherwise the resulting objects are wrapped in a list, and lists are not valid array elements. Another thing to be consider is that the resulting array is always type 29. for symbolic array, even if you started with an array containing only real or complex numbers. If you want to reconvert the array to its original type, you should use the command sequence OBJ-> ->ARRAY or ARRAY-> ->ARRAY after MAP has finished its work.

If you have a list of arrays like for example $\begin{matrix} 1 & 2 & 5 & 6 \\ 3 & 4 & 7 & 8 \end{matrix}$ and a program like `<< >>`, then pressing MAP returns the list

$\begin{matrix} \sqrt{1} & \sqrt{2} & \sqrt{5} & \sqrt{6} \\ \sqrt{3} & 2 & \sqrt{7} & 2 & \sqrt{2} \end{matrix}$. This shows that MAP goes

"downwards" the nesting levels, no matter if they are sub lists or arrays.

Last thing for today, the promised descriptions of some commands used so far. Let's start with SREPL. This command takes a string from stack level 3, a string from stack level 2 and a string from stack level 1. It replaces all occurrences of string of stack level 2 in the string of stack level 3 with the string in stack level 1. It returns the new string and the number of replacements made. It is case sensitive. Example:

3: "HP49G" SREPL 3: "HP58G∞"
2: "49G" 2: 1
1: "58G∞" 1:

REPL can do many things. For the time being we examine how it replaces parts of strings. It takes a string from stack level 3, a number from stack level 2 and a string from stack level 1. It replaces the characters in string of stack level 3, with the characters in string of stack level 1, starting at position in string of stack level 3 given by the

Complex numbers with the HP49G - Part 2

number of stack level 2. Example:

```
3: "Greek shepherd"  REPL  3:
2:      7             2:
1:  "cowboy "         1: "Greek cowboy "
```

HEAD returns the first element of a list or the first character of a string.

TAIL returns all but the first elements of a list as a list, or all but the first characters of a string as a string. It returns an empty list/string if the original list/string had only one element/character.

DISP takes an object from stack level two and a number from stack level one. It displays the object at the line indicated by the number at stack level 1. Objects are not displayed in pretty print, but as they appear in the command line. Strings are displayed without double quotes.

POS takes a list/string from stack level 2 and an object/string from stack level 1. It returns the position of the object in the list or the position of the string of stack level 1 in the string of stack level 2. If the object/string is not in the list/string, then 0. is returned.

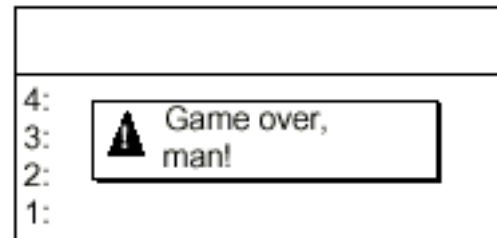
SUB can also do many things, but we examine here only its workings with lists or strings. It takes a list/string from stack level 3, the start position from stack level two and the end position from stack level one. It returns the portion of the list/string starting at the starting at the start position and ending at the end position.

IFERR THEN ELSE END is used to trap errors. The general syntax is:

```
IFERR
  commands1
THEN
  commands2
ELSE
  commands3
END
```

When an error occurs executing the command sequence `commands1`, then the command sequence is abandoned and the program continues with command sequence `commands2`. If flag -55 is clear, then the arguments that caused the error are returned to the stack. If this flag is set, then these arguments are not returned to the stack. If no error occurs executing `commands1`, then the program continues with the command sequence `commands3`. When the clause has been finished, the program continues with commands after END. The ELSE part can be omitted. The commands ERRM, ERRN, and ERR0 are related to errors. ERRM returns the last error message as a string. ERRN returns the last error number as a binary integer. ERR0 clears both ERRM and ERRN. And while we are at it, DOERR creates an error. It can take a binary integer or a real number or an integer and does the corresponding error of the HP49G. Or it takes a string, does an error and uses this string as the error message. Example:

"Game over, man!" DOERR , will produce:



That means the game is over for today. Hope you enjoyed it! next time we'll be messing around with symbolic complex quantities and some confusing modes combinations and assumptions for variables.

```
[Greetings Nick] << 2 ^ >> MAP
```

↓

```
[Greetings2 Nick2]
```


Complex numbers with the HP49G - Part 3

Hi everybody!

Before starting the third part, let me correct an error of the last part which the awoken eyes of Veli-Pekka have discovered. On page 2-30 the example for usage of command REPL with strings, had a wrong argument on stack level 2. The correct number should be 7. Thank you very much for noticing this Veli-Pekka!

We continue today with the third part of the Complex Numbers Marathon, which promises to be very interesting, since we have Obi Van Trabakoulas and Darth Kojak as guest stars. Oh, and of course because we start one of the most complex routes of this marathon, namely symbolic complex quantities and the settings that influence their interpretation on the CAS of the HP49G.

Actually Trabakoulas didn't plan to be a Jedi Knight. He never cared about such "adventures for smart guys", as he says. But yesterday the police visited him at his place and started asking questions about how many cows he has and if he knew about allowed limits of cow radiation. And he continues with a smoking pipe between his lips:

"My son, they came here with their space ships in the night, do you understand this? - with all lights on, and the cows were so scared that next morning they gave cheese instead of milk! - Those imperial stupids! Don't they ever sleep? The whole world sleeps at night, even the cows! The smiling baldie (Darth Kojak) dared call my cows rubbish, so I took old Alma (the boss cow) and directed her to attack his... - you know the part where his sits on." (That's why Darth Kojak breaths so heavy, especially when getting up!)

From now on Trabakoulas is Obi Van Trabakoulas the cow warrior and he doesn't need photon swords and other technology miracles, since old Alma does the work very nice. May the cow radiation be with him!

Now back to our subject. We are going to look at symbolic complex expressions, that is complex algebraic expressions that contain variables in which no value has been stored, like for example x and other. The HP49G has a lot of settings and of course even more combinations of

them, that allow you to control it exactly as you wish, when it comes to complex quantities. But because the settings are so many, working with the HP49G is also sometimes a bit confusing, because you might get unexpected results and think that the calculator starts getting crazy.

Let's first configure the HP49G so that we all start with the same settings. There are many settings that influence the workings of the HP49G regarding complex quantities. So let's use STOF to make all settings at once. We will be discussing about all these settings and flags one after the other in the examples that follow. Now, enter the list `{#A003008D010FF0h #0h #190100402000028h #0h}` and use STOF to store all flags at once. Press [blue-shift] and then [APPS] to start the filer. Go to the directory CASDIR find the variable REALASSUME. If you find it, select it and press the menu key [EDIT]. Edit the list to `{X Y t S1 S2}` if necessary and press [ENTER]. If the variable REALASSUME doesn't exist in CASDIR, then press the menu key [NEW] to create a new variable. Enter the list in the input field labelled with "Object:". In the input field labelled with "Name:" enter REALASSUME. Press menu key [OK] and exit the filer. We'll see what REALASSUME is for in the following examples.

Let's first examine what real and complex mode is for. You can control this mode with the flag -103. When this flag is set, the HP49G is in complex mode and a small capital "C" appears at the top of the screen. When this flag is clear, the HP49G is in real mode (like now) and a small capital "R" appears at the top of the screen. What can be controlled with this mode? Well, purge variable A if it exists in the current path, enter 'A' and press RE. The result is A. Enter again 'A' and press IM. Now the result is 0. This example shows that for the HP49G the variable A is a real quantity, because it finds the real part of A to be A itself, and the imaginary part of A to be 0, and these results are valid for any real variable. Let's try some other function. Enter the equation $A^2 + 1 = 0$, then enter 'A' and press SOLVE. The result is an empty list because there are no real solutions for this equation. That means again, that the HP49G in real mode considers generic (formal) variables to be real. If you try to factor

Complex numbers with the HP49G - Part 3

$A^2 + 1$ with FACTOR, then you get $A^2 + 1$ as result, which again shows that the HP49G "sees" the variable A as a real quantity. Let's do some more exploration. Enter 'A' and press ABS. The HP49G returns $|A|$, and this also shows the same general behaviour. But there are also exceptions. Enter 'A' again and press ARG. Now the result is:

$$\text{ATAN} \frac{\text{IM}(A)}{\text{RE}(A)} + 1 - \frac{\text{RE}(A)}{| \text{RE}(A) |} \frac{1}{2}$$

and the HP49G has switched to complex mode. (Look at the small "C" at the top of the display.) When the HP49G returns results with $\text{IM}(A)$ or $\text{RE}(A)$, it considers variable A to be a generic complex quantity which has a real part $\text{RE}(A)$ and an imaginary part $\text{IM}(A)$. But why the differences? Do some of the functions like ABS, RE and so on, consider generic variables to be real, while other functions like ARG consider them to be complex? Well, to find the answer, switch back to real mode by pressing simultaneously the keys [blue-shift] and [TOOLS]. Enter 5 and press ARG again. The result is 0 but the HP49G has again switched to complex mode. This shows that the function ARG first switches the HP49G to complex mode. In complex mode variables are considered to be in general complex (with some exceptions, which we will examine later on), that is $\text{RE}(A) = A$ and $\text{IM}(A) = 0$ in general. The result of ARG is then calculated, and because ARG has already switched the calculator to complex mode, it returns the above result that contains $\text{RE}(A)$, and $\text{IM}(A)$. Switch back to real mode and EXPAND the expression

$$\text{ATAN} \frac{\text{IM}(A)}{\text{RE}(A)} + 1 - \frac{\text{RE}(A)}{| \text{RE}(A) |} \frac{1}{2}. \text{ You get } \frac{(|A| - A)}{\text{SQ}(A)} \frac{|A|}{2}.$$

Press EXPAND again and you get the result $\frac{A^2 - A}{2 A^2} \frac{|A|}{2}$. Press

EXPAND another time to obtain $-\frac{|A| - A}{2 A}$, which again shows

that in real mode generic variables are real for the HP49G. We will see later on, how we can get this result right from the start. Let's add some more complicated examples. Enter e^Z and press RE or IM. You get the expected results, e^Z or 0. Enter $Z^2 + 1$ and press ABS. The result is $Z^2 + 1$, which is correct when Z is real because then the quantity $Z^2 + 1$ can't be less than 0, and so $|Z^2 + 1| = Z^2 + 1$. Enter $Z^2 - 1$ and press ABS. Now the result is $|Z^2 - 1|$. This is also correct, because in this case the absolute value depends on the value of Z and we didn't give the HP49G any information about this value. "But can we do that?", you might ask. "Can we give the HP49G any additional information about such formal variables?". And the answer is, "Oh yes, we can!". Assume for example that Z is some physical quantity, say a length, that in your specific problem can't be less than 1. In such a case you enter $Z \geq 1$ and use the command ASSUME. This command adds the relation $Z \geq 1$ to the list REALASSUME. (It also returns its argument to the stack, though the normal behaviour in RPN mode would be to remove it from the stack, but that's a different story. ;-)) From now on calculations that somehow have to do with Z will take into account that Z is greater than or equal to 1. If you now enter $Z^2 - 1$ and press ABS, the calculator returns $Z^2 - 1$ (and not $|Z^2 - 1|$) because it knows that $Z \geq 1$ and so the quantity $Z^2 - 1$ is always greater than or equal to 0, which means that $|Z^2 - 1| = Z^2 - 1$.

Let's verify this by examining the case $0 \leq Z \leq 1$. Remove first all assumptions already made for Z . Enter 'Z' and then use the command UNASSUME, which removes all assumptions for Z . Now enter $Z \leq 1$ and use ASSUME. Enter $Z \geq 0$ and press ASSUME again. (If you now take a look at REALASSUME you will see that it contains the assumption $Z \geq 1$ AND $Z \geq 0$, which is one of the ways that the HP49G can express the relation $0 \leq Z \leq 1$. Now enter $Z^2 - 1$ again and press ABS. The result $-(Z^2 - 1)$ is compatible with our assumptions for Z . Remove again all assumptions for Z by entering

Complex numbers with the HP49G - Part 3

'Z' and using the command UNASSUME.

In general there are two kinds of assumptions. The first kind is to simply assume that some variable is real. You do that by entering the variable and using the command ADDTOREAL, which adds the variable to the list REALASSUME. This kind of assumption simply tells the HP49G that the variable is real. The second kind is a relation like for example $Z \neq 0$ which tells the HP49G that the variable is real and that its value has the constraints given by the relation on stack level 1. The first kind of assumptions doesn't seem to be very useful in real mode, since in real mode formal variables are considered to be real. But this kind of assumption has great usefulness. We have already seen that some functions, like for example ARG, switch the calc to complex mode and return results that are compatible with the assumption that variables are complex. But if you enter 'Z' in real mode, use ADDTOREAL to add Z to the real variables, and then press ARG, then the result is no more the big expression with $\text{RE}(Z)$ and

$\text{IM}(Z)$ from the previous page, but simply $1 - \frac{Z}{|Z|^2}$. This kind of

assumptions is also very useful when working in complex mode, which we will examine later on.

Now, remove again assumptions for Z by entering 'Z' and using the command UNASSUME. Also switch back to real mode. Enter $Z \neq 0$ and use ASSUME to inform the HP49G that Z is greater than or equal to 0. Enter $Z^2 - 1$, then 'Z' and press SOLVE. The HP49G returns only one solution, namely the solution $Z = 1$ which is compatible with the assumption $Z \neq 0$. We see that SOLVE considers assumptions in this case. But this is not always the case. Let's try a system of linear equations. Enter the vector of equations $[Z + 1 + X = -2 \quad X + 3 = 4]$ and then the vector of unknowns $[X \quad Z]$. Now press SOLVE again. The HP49G returns a list with the solutions vector $\{[X = 1 \quad Z = -4]\}$ which contains a solution $Z = -4$, that is not compatible to the assumption $Z \neq 0$! So you can't lay back and expect that the solutions

will be always compatible to your assumptions! In this case SOLVE seems to use some of the algorithms involved in the command GBASIS, and these algorithms doesn't seem to care about assumptions. Remove now all assumptions for Z.

Let's check further properties of real mode. Enter $Z^3 - Z^2 + Z - 1$, then enter 'Z'. Press SOLVE. The HP49G returns $Z = 1$. The equation $Z^3 - Z^2 + Z - 1$ has also the solutions $Z = i$ and $Z = -i$ but these solutions are filtered out, since we are in real mode and so Z must be real. If you enter $Z^3 - Z^2 + Z - 1$ and press FACTOR, then the HP49G returns $(Z - 1)(Z^2 + 1)$ instead of $(Z - 1)(Z + i)(Z - i)$, that is it stays in the real domain. But if you enter $(Z - 1)(Z + i)(Z - i)$, press EXPAND first, and then solve for Z, you get the solutions $\{Z = 1 \quad Z = i \quad Z = -i\}$. Similarly, if you enter $(Z - 1)(Z + i)(Z - i)$, press EXPAND first, and then FACTOR, you get $(Z - 1)(Z + i)(Z - i)$. This is because the command EXPAND switched to complex mode, since it had to expand an expression that contained i, the imaginary unit.

Here we should also note the difference between FACTOR and COLLECT. Leave the HP49G in complex mode, enter $Z^3 - Z^2 + Z - 1$ and press COLLECT. Though we are in complex mode, the result is not $(Z - 1)(Z + i)(Z - i)$ but $(Z - 1)(Z^2 + 1)$. The command COLLECT does its work over the integers, and so it factors staying in the real domain when possible. What does this "when possible" means? Let's have an example. Enter $Z^3 + iZ^2 - Z - i$, switch to real mode and press COLLECT. The result is $(Z + 1)(Z - 1)(Z + i)$. That shows that COLLECT can return also complex factors, when the expression to be COLLECTed already contains some complex terms and can't be factored solely over the integers.

Now that we know about assumptions, perhaps you have the question in mind: "What happens if we make two incompatible assumptions?"

Complex numbers with the HP49G - Part 3

For example, suppose you have made the assumptions $Z \geq 0$, and after some days you forgot about it. Then you make the assumption $Z \leq 0$. The list REALASSUME now contains the assumption $Z \leq 0$ AND $Z \geq 0$. If you now try to solve for example $Z^2 + Z - 2$ for Z , then you will get an empty list as solution though the equation $Z^2 + Z - 2 = 0$ has the solutions $Z = 1$ and $Z = -2$. It can be of great importance to check the list REALASSUME from time to time, or to remove all assumptions made for some variables, after solving some problem, in order to start "with a clean sheet of paper".

Furthermore, what happens if we manually put some assumption in REALASSUME, which can't be used with the command ASSUME? You could for example add $\text{SIN}(Z) \geq 0$ in REALASSUME by editing the list manually. If you later enter $\text{SIN}(Z)$ and press ABS, then the result is $|\text{SIN}(Z)|$ and not $\text{SIN}(Z)$, which shows that such assumptions are not used. Only assumptions of type Name Expression or Name Expression will work. Note that you can also make one assumption that depends on another assumption. For example you can make the assumption $T \geq 0$ and the the assumption $Z \geq T^2$. If you then enter Z and press ABS, the result will be Z , because the HP49G knows that Z is greater than or equal to T^2 and that T is greater than or equal to 0, which together results in $Z \geq 0$.

Remove now the assumptions for T and Z and let's examine another flag that influences such results, like those which we have talked about until now. We are going to examine flag -119, which controls rigorous/non-rigorous mode. When this flag is set we are at non-rigorous mode. When it is clear we are at rigorous mode. You can set or clear this flag with SF and CF like any other flag. But you can also press [MODES], then the menu key [CAS] and then check or uncheck the option _Rigorous. When in non-rigorous *and* real mode the HP49G assumes that for all generic variables A , the relation $|A| = A$ is true. That means all variables are real and greater than or equal to 0. Switch to real non-rigorous mode, and enter Z . Now press ABS. The

result is Z and not $|Z|$, which shows that the HP49G considers Z to be real and greater than or equal to 0. Enter Z again and press ARG.

The HP49G returns now the result $\text{ATAN} \frac{\text{IM}(Z)}{\text{RE}(Z)}$ and not

$\text{ATAN} \frac{\text{IM}(Z)}{\text{RE}(Z)} + 1 - \frac{\text{RE}(Z)}{| \text{RE}(Z) |} \frac{1}{2}$, which we saw on page 3-2. This makes sense partially only. It is correct not to return the part

$1 - \frac{\text{RE}(Z)}{| \text{RE}(Z) |} \frac{1}{2}$, if we consider that $\text{RE}(Z) = Z$ (because we started

at real mode) and $|Z| = Z$ (because we started at non-rigorous mode), and the two relations together result in $| \text{RE}(Z) | = \text{RE}(Z)$. And so we

have $1 - \frac{\text{RE}(Z)}{| \text{RE}(Z) |} \frac{1}{2} = 0$, and so only the term $\text{ATAN} \frac{\text{IM}(Z)}{\text{RE}(Z)}$

remains as the result of ARG. But because we know also that $\text{RE}(Z) = Z$, we have that Z is real and so $\text{ATAN} \frac{\text{IM}(Z)}{\text{RE}(Z)} = 0$. The

result should be 0 in order to make sense completely. It is not that this result is wrong, it is only that the HP49G uses the modes combination real/non-rigorous only partially. You can of course switch back to real mode and press EXPAND, which returns the expected 0.

The non-rigorous mode is not taken into consideration by all commands. For example if you SOLVE $Z^2 + Z - 2 = 0$ for Z in real/non-rigorous mode, then you get the solutions $\{Z = 1 \ Z = -2\}$, though non-rigorous mode implies mathematically that $|Z| = Z \ Z \geq 0$. So perhaps it is better to think of non-rigorous mode as the mode where the CAS takes only the relation $|Z| = Z$ into consideration, but not the relation $Z \geq 0$. Let's examine this a bit better. Switch to real/non-rigorous mode, enter $Z \geq 0$ and use ASSUME to add this assumption to REALASSUME. Now enter Z

Complex numbers with the HP49G - Part 3

again and press ABS. The result is $-Z$ and it shows that assumptions override non-rigorousity. If you now enter Z again and press ARG, then you get $1 + \frac{Z}{Z} \frac{Z}{2}$, which is correct for a real Z that is less than or equal to 0. The non-rigorousity relation $|Z| = Z$ is only used for variables with no assumptions at all.

Now switch back to real rigorous mode. The last option that we examine, is the flag -128. When this flag is clear, then complex variables are allowed, in the sense that the HP49G will consider formal variables to be complex, if it has no other means of determining if they are real or not. When the flag is set, then all formal variables are considered to be real. You can set/clear this flag with SF or CF, or with the flag browser. Let's look at this a bit closer. Set flag -128 and enter Z . Use UNASSUME to remove the assumptions made for Z . Now press ABS. The result is $|Z|$. This is not very surprising. But if you

enter Z again and press ARG, then the result is $1 - \frac{Z}{|Z|} \frac{Z}{2}$ and this

shows that the HP49G considered Z to be real, though ARG has switched the calculator to complex mode. The result also shows that the HP49G made no further assumptions about Z , neither $Z > 0$, nor $Z < 0$. Switch back to real mode. If you now enter $Z > 0$ and then use

ASSUME, then entering Z and pressing ARG returns $1 - \frac{Z}{Z} \frac{Z}{2}$

which shows that the calculator considered Z to be real and also greater than or equal to 0, which means ARG must return 0, or an equivalent result.

This mode (only real variables, flag -128 is set) is particularly useful, when we work with physical quantities, for which we know right from the start, that they are all real.

Switch now back to real rigorous mode, enter Z and press UNASSUME and then clear flag -128 to allow complex variables

again.

From what has been said until now, you may think that when complex variables are somehow "deactivated", be it through assumptions or mode settings, then we can only work with real quantities. But this is not the case. Consider for example the two variables X and Y , which are in the list REALASSUME per default. These two variables are of course real for the HP49G, but we can make a complex quantity with them: $X + i Y$. If you enter this in real mode and press for example RE, then the HP49G returns X , because it knows that X and Y are real. If you enter $X + i Y$ and press IM, the HP49G returns Y for the same reason. The function ABS will return $\sqrt{X^2 + Y^2}$ and switch the HP49G to complex mode. Switch back to real mode and try another example. Enter e^Z and press ABS. The result is e^Z , because of the reasons described on page 3-2. But if you enter e^{X+iY} and press ABS, then the result is $|e^X| \sqrt{\sin(Y)^2 + \cos(Y)^2}$ and the HP49G is in complex mode. The HP49G has turned e^{X+iY} to $e^X (\cos(Y) + i \sin(Y))$ and then has found the absolute value of the latter expression. Switch back to real mode and press EXPAND. You get $e^X \sqrt{\sin(Y)^2 + \cos(Y)^2}$, because the HP49G knows that $e^X > 0$ for real X . If you now use the function TRIG (trigonometry marathon), then you get e^X , the end result. If you enter e^{X+iY} and press RE in real mode, then you get $e^X \cos(Y)$, the real part of $e^X (\cos(Y) + i \sin(Y))$.

Of course assumptions and/or non-rigorousity are also used in calculations with complex quantities of the form $X + i Y$, where X and Y are real.

You already noticed that some commands, like for example EXPAND, will switch the calculator to complex mode, when the expression being EXPANDED directs them to do so. There has been already a lot of discussions and wishes about the behaviour of the HP49G regarding

Complex numbers with the HP49G - Part 3

such mode switches. Some want them to happen automatically as needed, others want the HP49G to ask first what to do, and quite a few people (like me) don't like this behaviour at all. (Consider for example Mathematica or Maple. Perhaps their inner workings do some kind of "switch" to real or complex mode. But such "switches" are kept out of your way.) The current version of the system/CAS software of the HP49G uses a flag, namely flag -120, for determining if such mode changes are performed automatically, or if you should be asked each time such a mode change is about to be performed. When flag -120 is set, then we have "silent mode on", which means that mode changes will be performed without asking you if you want them to be. When this flag is clear, then "silent mode" is off, meaning that the calculator will ask you each time, when it wants to change modes. (And ruin any programs that have to run unattended!) If you have the time and curiosity, just switch to real mode and clear flag -120. Then go to the EQW and enter $(X + i Y)^2$. Press EXPAND. You are presented a pop-up with the question "Complex mode on?" and the choices "Yes" and "No". If at this point you choose "Yes", then everything will be fine. But if you say "No", then the calculator errors out with "EXPAND Error: Mode Switch Cancelled"! Judge for yourself if this behaviour is reasonable or not, but for now set flag -120 again, so that such questions are suppressed and the HP49G does its work without asking you to choose.

By the way, if flag -120 is clear and you press [blue-shift] and then [TOOL] to enter the imaginary unit i while you work in the normal stack with the normal command line, then the change modes dialogue appears again. But if you under the same circumstances enter 'i', that is the imaginary unit enclosed in single quotes, then the dialogue doesn't appear. The question for mode changes appears only when the HP49G has to do something with the input, evaluate it, EXPAND it and so on. The single quotes tell the HP49G to take the input expression and just put it on the stack as is, without any changes, and so the dialogue doesn't come up. But with the flag -120 clear (silent mode off), if you press [blue-shift] and then [TOOL] while you are in the EQW, then no change mode dialogue appears. You can even press [ENTER] to put the imaginary unit on the stack without any question about switching to

complex mode! (If you cleared flag -120, set it now, to turn silent mode on again.)

We have covered all (?) possible combinations of settings that influence the workings of the HP49G, when it is in real mode and has to do with complex quantities. All above information, except the adventures of Obi Van Trabakoulas, is summarised on the table titled "What the HP49G does with complex quantities in real mode" on the next page. What? You want to know what happens under the same settings but in complex mode? You curious person want me to write another five pages about the behaviour of the HP49G? OK, OK, I'll do that, but remember to bring me some donuts when you visit me at the neurological clinic ;-).

First of all switch your HP49G to complex, rigorous mode and set complex variables allowed. We are going to repeat the same examples for better understanding the correspondence of behaviours. Here we go again. Purge variable A if it exists in the current path and remove all assumptions for it. Enter A and press RE. The result is $RE(A)$.

Enter A again and press IM. The result is $IM(A)$. For the HP49G the variable A is now a complex quantity, consisting of a real part $RE(A)$ and an imaginary part $IM(A)$. Enter the equation $A^2 + 1 = 0$, enter A and press SOLVE. The result is the solution list $\{A = i \quad A = -i\}$. (Press EXPAND to turn it to $\{A = -i \quad A = i\}$). That means again, that the HP49G in complex mode considers formal variables to be complex. If you try to factor $A^2 + 1$ with FACTOR, then you get $(A + i)(A - i)$ as result, which again shows that the HP49G considers variable A to be a complex quantity. Now, enter A and press ABS.

The HP49G returns $\sqrt{RE(A)^2 + IM(A)^2}$, the general form of the magnitude of the complex variable A. Enter A again and press ARG,

to obtain $ATAN \frac{IM(A)}{RE(A)} + 1 - \frac{RE(A)}{|RE(A)|} \frac{1}{2}$, the general form of the angle of a complex quantity A.

Complex numbers with the HP49G - Part 3

What the HP49G does with complex quantities in real mode

	Without assumptions about formal variables	With assumptions about formal variables
Rigorous mode is on	Formal variables are in general considered to be real. Exception when some commands like ARG cause switching to complex mode. In such cases expressions containing $RE(A)$ and/or $IM(A)$ are returned. Switching back to real mode and EXPAND will turn $RE(A)$ and/or $IM(A)$ to A and/or 0 . Any algebraic manipulations that include absolute values are performed in the mathematically strictly correct way, without assuming anything about formal variables in the most general form and returning $ A $ when necessary. Solving equations returns only real solutions if some exist, or no solutions at all if only complex solutions exist. When only real variables are allowed (flag -128 set), then commands like ARG which switch to complex mode, simplify $RE(A)$ and/or $IM(A)$ to A and/or 0 . When flag -120 is clear (silent mode off) you are always asked before any mode change is performed.	Formal variables are in general considered to be real and assumptions for them are taken into consideration. When commands like ARG cause switching to complex mode then if assumptions for variables are present, these are also used. Any algebraic manipulations that include absolute values are performed considering assumptions about formal variables. When no assumptions are present, then the most general forms are returned, like for example $ A $, when necessary. Solving equations returns only real solutions if some exist, or no solutions at all if only complex solutions exist. The solutions are <i>only in some cases</i> compatible to assumptions. When only real variables are allowed (flag -128 set), then commands like ARG which switch to complex mode, simplify $RE(A)$ and/or $IM(A)$ to A and/or 0 . When flag -120 is clear (silent mode off) you are always asked before any mode change is performed.
Rigorous mode is off	Formal variables are in general considered to be real. Exception when some commands like ARG cause switching to complex mode. In such cases expressions containing $RE(A)$ and/or $IM(A)$ are returned. Switching back to real mode and EXPAND will turn $RE(A)$ and/or $IM(A)$ to A and/or 0 . Any algebraic manipulations that include absolute values are performed considering that for any formal variable A the assumption $ A = A$ holds. This assumption is only used to replace $ A $ by A but not in the sense $A \geq 0$. Solving equations returns only real solutions if some exist, or no solutions at all if only complex solutions exist. When only real variables are allowed (flag -128 set), then commands like ARG which switch to complex mode, simplify $RE(A)$ and/or $IM(A)$ to A and/or 0 . When flag -120 is clear (silent mode off) you are always asked before any mode change is performed.	Formal variables are in general considered to be real. Exception when some commands like ARG cause switching to complex mode. In such cases expressions containing $RE(A)$ and/or $IM(A)$ are returned. Switching back to real mode and EXPAND will turn $RE(A)$ and/or $IM(A)$ to A and/or 0 . Any algebraic manipulations that include absolute values are performed considering your assumptions. If you didn't make any assumptions, then for any formal variable A the assumption $ A = A$ is used. This assumption is only used to replace $ A $ by A but not in the sense $A \geq 0$. Solving equations returns only real solutions if some exist, or no solutions at all if only complex solutions exist. The solutions are <i>only in some cases</i> compatible to assumptions. When only real variables are allowed (flag -128 set), then commands like ARG which switch to complex mode, simplify $RE(A)$ and/or $IM(A)$ to A and/or 0 . When flag -120 is clear (silent mode off) you are always asked before any mode change is performed.

Complex numbers with the HP49G - Part 3

In complex mode the HP49G considers any formal variable A to be $\text{RE}(A) + i \text{IM}(A)$. Enter e^Z and press RE or IM. You get the results $e^{\text{RE}(Z)} \text{COS}(\text{IM}(Z))$ and $e^{\text{RE}(Z)} \text{SIN}(\text{IM}(Z))$. How does the HP49G derive this? Well, first of all Z is now for the HP49G a complex quantity and thus $Z = \text{RE}(Z) + i \text{IM}(Z)$. So the expression e^Z is for the HP49G equivalent to $e^{\text{RE}(Z) + i \text{IM}(Z)}$. If you enter the latter expression and press TEXPAND, then you get $e^{\text{RE}(Z)} e^{i \text{IM}(Z)}$. From the trigonometry marathon we remember that for $e^{i \text{IM}(Z)}$ the relation holds: $e^{i \text{IM}(Z)} = \text{COS}(\text{IM}(Z)) + i \text{SIN}(\text{IM}(Z))$. (If you want to check this out for yourself, take the expression $e^{\text{RE}(Z)} e^{i \text{IM}(Z)}$ in the EQW, select the sub-expression $e^{i \text{IM}(Z)}$ and press SINCOS.) So we have the relation: $e^{\text{RE}(Z)} e^{i \text{IM}(Z)} = e^{\text{RE}(Z)} (\text{COS}(\text{IM}(Z)) + i \text{SIN}(\text{IM}(Z)))$ or $e^{\text{RE}(Z)} e^{i \text{IM}(Z)} = e^{\text{RE}(Z)} \text{COS}(\text{IM}(Z)) + i e^{\text{RE}(Z)} \text{SIN}(\text{IM}(Z))$. Now you can see that the HP49G calculated correctly real and imaginary parts of e^Z .

Enter $Z^2 + 1$ and press ABS. The HP49G returns the result $\sqrt{\text{RE}(Z)^4 + (2 \text{IM}(Z)^2 + 2) \text{RE}(Z)^2 + \text{IM}(Z)^4 - 2 \text{IM}(Z)^2 + 1}$. Is that correct? Well, if we want to check that, we must do some mathematics ourselves. For any complex quantity Z we have the relation: $\text{ABS}(Z) = \sqrt{\text{RE}(Z)^2 + \text{IM}(Z)^2}$. For the complex quantity $Z^2 + 1$ we have the relation:

$$\begin{aligned} Z^2 + 1 &= (\text{RE}(Z) + i \text{IM}(Z))^2 + 1 \\ &= \text{RE}(Z)^2 - \text{IM}(Z)^2 + 2 i \text{RE}(Z) \text{IM}(Z) + 1 \end{aligned}$$

The real part of $Z^2 + 1$ is $\text{RE}(Z)^2 - \text{IM}(Z)^2 + 1$ and its imaginary part is $2 i \text{RE}(Z) \text{IM}(Z)$. If we square the real and imaginary parts and add

the results, we get:

$$(\text{RE}(Z)^2 - \text{IM}(Z)^2 + 1)^2 + (2 i \text{RE}(Z) \text{IM}(Z))^2, \text{ or}$$

$\text{RE}(Z)^4 + (2 \text{IM}(Z)^2 + 2) \text{RE}(Z)^2 + \text{IM}(Z)^4 - 2 \text{IM}(Z)^2 + 1$, which under the sign of the square root is exactly what the HP49G has already told us a long time ago.

Now, what about assumptions?. Let's check that. Add Z to the list of real variables, enter $Z^2 + 1$ and press ABS. The result is $|Z^2 + 1|$, which shows that the HP49G took our assumptions only partially into consideration, because it recognised that Z is real and so the absolute value of $Z^2 + 1$ is $|Z^2 + 1|$. But $Z^2 + 1$ can't be less than 0 if Z is real, and so we have $|Z^2 + 1| = Z^2 + 1$. The HP49G didn't realise that, though it does in real mode! What if we make some explicit assumption, say $Z \geq 1$? Enter Z and press UNASSUME. Now enter $Z \geq 1$ and use ASSUME. Enter again $Z^2 + 1$ and press again ABS. The result is again $|Z^2 + 1|$ and not $Z^2 + 1$ as it would be if we did exactly the same in real mode! The fact that a variable is assumed to be real will be considered in complex mode, but nothing more in this case. But the inconsistencies go even further! While the assumption $Z \geq 1$ is still valid, enter $Z^2 - 1$ and press ARG. The HP49G returns 0, which shows that in this case it did realise that $Z^2 - 1$ can't be less than 0, when $Z \geq 1$! Clear all assumptions for Z , enter $Z \geq 1$ and use ASSUME. Now enter $Z \leq -1$ and use again ASSUME to put the assumption $Z \geq 1$ AND $Z \leq -1$ in the list REALASSUME. Enter $Z^2 - 1$ and press ABS. The HP49G returns $|Z^2 + 1|$ which shows that the assumption $Z \geq 1$ AND $Z \leq -1$ was not taken into consideration. The HP49G only used the fact that Z is real. If it had used the assumption, it should have returned $-(Z^2 - 1)$ like it did in real mode. Now enter $Z^2 - 1$ and press ARG. The result is

Complex numbers with the HP49G - Part 3

$1 + \frac{Z^2 - 1}{Z^2 - 1} \cdot \frac{1}{2}$, which you can EXPAND to $\frac{1}{2}$. This shows that in

this case the HP49G took the assumption $Z \neq 1$ AND $Z \neq -1$ into consideration and found out that under these circumstances $Z^2 - 1$ is negative and so its angle is π . I think this is a very inconsistent behaviour which can drive the users crazy.

Now, let's see how SOLVE behaves with assumptions. Remove again assumptions for Z . Enter Z and use ADDTOREAL to tell the HP49G that Z is a real variable. SOLVE $Z^2 + 1 = 0$ for Z . The HP49G returns the list $\{Z = i \ -1 \ Z = -i\}$, which shows that the assumption that Z is real was not taken into consideration. Now, make the assumption $Z \neq 0$. SOLVE $Z^2 - 1 = 0$ for Z . This time the HP49G returns only one solution, the solution $Z = 1$ which is compatible with the assumption $Z \neq 0$. This assumption was taken into consideration! With the same assumption SOLVE $Z^4 - 1 = 0$ for Z . Now the HP49G returns $\{Z = i \ Z = -i \ Z = 1\}$. The complex solutions and the real solution compatible with the assumption were returned. Assumptions of the first kind, namely those which simply state that a variable is real, are not taken into consideration, while assumptions of the second kind, namely those which say something about the value of a variable are taken into consideration. Let's try again a system of linear equations with the same assumption $Z \neq 0$. Enter the vector of equation $[Z + 1 + X = -2 \ X + 3 = 4]$, then the vector of unknowns $[X \ Z]$ and press SOLVE again. The HP49G returns a list with the solutions vector $\{[X = 1 \ Z = -4]\}$ which contains a solution $Z = -4$, that is not compatible to the assumption $Z \neq 0$! This time the assumption of the second kind $Z \neq 0$ was not taken into consideration! Remove the assumption, enter Z and use REALASSUME. Enter the vector of equations $[Z - (1+i) - i \ 3 \ X \ X + i \ Z = i + 2]$, the vector of unknowns $[X \ Z]$ and press SOLVE. The result

$X = -\frac{3-2i}{2} \ Z = -\frac{7i}{2}$ is also incompatible to the assumption

of the first kind, namely that Z is real! We can clearly see with these examples, that the assumptions are used in a totally inconsistent and unorganised way in complex mode. Add the fact that assumptions are hidden in a list which you can't easily recall and view, and you have a feature that is there only for... just saying that it is there. It is totally unusable in complex mode, and only of partial usefulness in real mode. Remove now again all assumptions for Z .

Let's check how FACTOR and COLLECT behaves when the HP49G is in complex mode. If you enter $Z^3 - Z^2 + Z - 1$ and press FACTOR, then the HP49G returns $(Z - 1)(Z + i)(Z - i)$, the expected result. (Well, at least *one* expected result ;-)) Enter $Z^3 - Z^2 + Z - 1$ again and press COLLECT. The result is $(Z - 1)(Z^2 + 1)$ in accordance to the result in real mode.

Let's examine now rigorous/non-rigorous mode and hope that the chaotic behaviour with assumptions will not repeat itself. Press [MODES], then the menu key [CAS] and then uncheck the option _Rigorous. As already said, when in non-rigorous and real mode the HP49G assumes (or should assume ;-)) that for all formal real variables A , the relation $|A| = A$ is true. That means all formal variables are real and greater than or equal to 0.

Enter Z . Now press ABS. The result is $\sqrt{\text{RE}(Z)^2 + \text{IM}(Z)^2}$, which shows that the HP49G considers Z to be complex and calculates the absolute value accordingly. This is compatible with the fact that we are in complex mode, since in this mode all formal variables without any assumptions are considered to be complex. Enter Z again and press ARG. The HP49G returns now the result $\text{ATAN} \frac{\text{IM}(Z)}{\text{RE}(Z)}$ and not

Complex numbers with the HP49G - Part 3

$\text{ATAN} \frac{\text{IM}(Z)}{\text{RE}(Z)} + 1 - \frac{\text{RE}(Z)}{|\text{RE}(Z)|^2} \frac{1}{2}$, which we saw on page 3-2.

Why the difference and is that OK? Well, if you would calculate the angle of the complex number $1 + 1i$ with the formula

$= \text{ATAN} \frac{\text{IM}(Z)}{\text{RE}(Z)}$, you would get $\frac{\pi}{4}$, which is OK. But if you

would calculate the angle of the complex number $-1 - 1i$ with the same formula, you get again $\frac{\pi}{4}$, which is wrong, as the angle of this

complex number is $-\frac{3\pi}{4}$ (or $\frac{5\pi}{4}$ if you like angles between 0 and

2π). The formula $\text{ATAN} \frac{\text{IM}(Z)}{\text{RE}(Z)} + 1 - \frac{\text{RE}(Z)}{|\text{RE}(Z)|^2} \frac{1}{2}$ calculates the

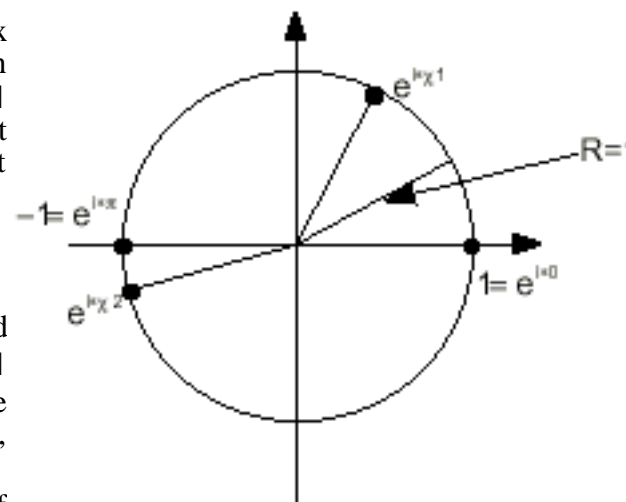
correct result for all cases, and since the HP49G in complex mode gave

us $\text{ATAN} \frac{\text{IM}(Z)}{\text{RE}(Z)}$ as the angle of the complex number Z , it gave us a

wrong result! *The lesson is again, complex non-rigorous mode is to be used with care, or better to not be used at all!*

Now, what happens if we SOLVE in non-rigorous mode? Enter $Z^2 + Z - 2 = 0$ and solve it for Z in complex/non-rigorous mode. The solutions $\{Z = 1, Z = -2\}$ are mathematically not compatible with $|Z| = Z - Z = 0$, which non-rigorous mode implies. But at least it is consistent to the behaviour in real/non-rigorous mode. (The CAS takes only the relation $|Z| = Z$ into consideration, but not the relation $Z = 0$. Since no assumptions are made for Z , this variable is complex and so $|Z| = Z$ doesn't apply.) Add Z to the list of real variables and SOLVE $Z^2 - 2i = 0$. You get the solutions $\{Z = (1+i), -1, Z = 1+i\}$. As you can see, the assumption of the first kind was not used. ASSUME now $Z = 0$ and SOLVE $Z^2 - 1 = 0$. The solution returned is $Z = 1$. The

other solution, $Z = -1$, is not returned, in accordance to the assumption of the second kind, which shows that this assumption *was* used. Remove all the assumptions for Z again. Solve $|Z| = 1$. The result is $Z = e^{ix1}$ which we interpret as a complex number Z having magnitude of 1 and lying somewhere on the unit circle. Non-rigorousity was not used in this case, because it should imply that the solutions are $Z = 1$ or $Z = -1$, that is complex numbers with magnitude of 1 lying not somewhere on, but at the angles 0 or π on the unit circle.



Add Z to the list of real variables and SOLVE $|Z| = 1$ again. You get the same result, $Z = e^{ix1}$. You also get this result, if you ASSUME $Z = 0$ or anything else. The use of assumptions is almost useless in complex non-rigorous mode.

Now switch back to complex rigorous mode. The last option that we examine, is the flag -128 for allowing or disallowing complex variables. Set flag -128 to disallow complex variables and enter Z . Use UNASSUME to remove the assumptions made for Z . Now press ABS. The result is $|Z|$, which is OK since we told the HP49G that all formal variables are real. The same if you enter Z again and press

ARG. The result is $1 - \frac{Z}{|Z|} \frac{1}{2}$ and this shows that the HP49G

considered Z to be real. If you now enter $Z = 0$ and then use ASSUME, then entering Z and pressing ABS returns Z , which is

Complex numbers with the HP49G - Part 3

correct under these circumstances. (Z is real and greater than or equal to 0.) Entering Z and pressing ARG returns 0 which shows again that the calculator took our assumptions into consideration. If you unassume Z , assume $Z = 0$ and find the absolute value of Z then the calculator

will return $-Z$. Finding the angle of Z returns then $1 + \frac{Z}{2}$ which

is also correct but not expanded to $\frac{1}{2}$. It looks like flag -128 works good in combination with assumptions. :-)

Clear flag -128 now, to allow complex variables again. You should beware of the fact that in complex mode an expression like for example $a + b \cdot i$ without any assumptions for a and b , is not equivalent to a complex quantity with real part a and imaginary part b . Both formal variables a and b are complex themselves! That means that for example the calculator returns $\text{RE}(a) - \text{IM}(b)$ and not a , when you enter $a + b \cdot i$ and press RE. Don't be irritated by the fact that entering $X + Y \cdot i$ and pressing RE returns X . This is simply because X and Y are per default in the list REALASSUME, and so it is likely that they are there when you work with the calculator. The default contents of REALASSUME are the variables X , Y , t , $S1$ and $S2$.

We finish our panic run through modes by noticing that the flag -120 works like in real mode. Now that we have all (?) possible combinations of settings that influence the behaviour of the HP49G regarding complex quantities, we summarise this information on the table titled "What the HP49G does with complex quantities in complex mode" on the next page. If someone discovers any special behaviour of the HP49G under some certain combination of settings/assumptions, then please post it to the group, so that we all know how many more dangers, monsters and other surprises wait for us in the flags jungle.

In the next part of the complex marathon we'll solve complex problems and do plots with complex quantities. Have fun, 'till the next time.

ASSUME(GREETINGS MAXR)

Nick.

Complex numbers with the HP49G - Part 3

What the HP49G does with complex quantities in complex mode		
	Without assumptions about formal variables	With assumptions about formal variables
Rigorous mode is on	<p>Formal variables are in general considered to be complex. Any algebraic manipulations that include absolute values are performed in the mathematically strictly correct way, without assuming anything about formal variables in the most general form and returning $\sqrt{\text{RE}(Z)^2 + \text{IM}(Z)^2}$ when necessary. Solving equations returns all possible real and complex solutions. When only real variables are allowed (flag -128 set), then commands like ARG which switch to complex mode, simplify $\text{RE}(A)$ and/or $\text{IM}(A)$ to A and/or 0. When flag -120 is clear (silent mode off) you are always asked before any mode change is performed.</p>	<p>Formal variables are in general considered to be complex when no assumptions are present for them. When assumptions are present the great chaos takes its way. Results are <i>only in some cases</i> compatible to assumptions. When only real variables are allowed (flag -128 set), then commands return results that are correct under the assumption that all formal variables are real. When flag -120 is clear (silent mode off) you are always asked before any mode change is performed.</p>
Rigorous mode is off	<p>Formal variables are in general considered to be complex. BEWARE AS IN THIS MODE YOU CAN GET WRONG RESULTS! Assumptions don't seem to be used at all. When flag -120 is clear (silent mode off) you are always asked before any mode change is performed.</p>	<p>Formal variables are in general considered to be complex. Assumptions are <i>in some cases</i> taken into consideration. If only real variables are allowed (flag -128 set), then commands like ARG which switch to complex mode, simplify $\text{RE}(A)$ and/or $\text{IM}(A)$ to A and/or 0. When flag -120 is clear (silent mode off) you are always asked before any mode change is performed.</p>

Complex numbers with the HP49G - Part 4

"Let's solve it...", said Trabakoulas
when he diluted sugar in coffee,
"..let's enter the hall of fame!"

You see, to solve and to dilute
in (old) greek language is the same.³

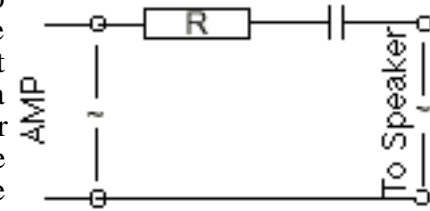
While we enjoy Obi Van Trabakoulas and his poetical eruptions, we are going to use what we have found so far, to solve some problems with the help of complex quantities. He is going to give us many ideas, sometimes a bit twisted ideas, but nonetheless helpful. And when he goes too far, we can always give him another cup of coffee and a pipe with greek tobacco, so that he keeps silent for a couple of minutes. (Procedure also known as "greek breakfast" ;-))

We start with an easy example taken from the physics of alternating current, dedicated to the one and only Rcoho (and his thousands of machines ;-)). We'll see also how much graphing can be done even with simple examples and how we can influence the details of the plots. So let's go.

Nick doesn't like the sound coming out of the amplifier for his electric guitar. It is full of bass (low frequencies) and somehow sounds like a loudspeaker box without tweeter. So he decides to place a high pass filter between the output of the amplifier and the speaker, to diminish the amplitude of low frequencies and so make the sound better.

³ L ö d
ö alysi t ö L
t ö p e

Such a filter is very easy to construct, if someone neglects some problems that come with it. It consists in its simplest form of a resistance and a capacitor connected in series (one after the other) with the outlets of the amplifier. The capacitor with the capacity C will let higher frequencies pass easier than low frequencies. The resistor R is needed for damping, that is to keep the ratio of high to low frequencies in reasonable amounts, so that the overall gain in high frequencies is not extraordinary high compared to low frequencies. (And it is also needed to make the problem a bit more interesting. ;-))



First make a new directory called RCCIRCUIT. Enter RCCIRCUIT and press CRDIR. This creates the directory in the current directory. If you now go to the variables menu, you'll see a new menu label "RCCIR" with a tab over it, which indicates that this variable is a directory. Press the soft menu key with this label, to go to the directory RCCIRUIT. Note that the header of the display now indicates the current path. (The command CRDIR is in menu PRG/MEM/DIR. You press [blue-shift] and then [CAT] to get the menu PRG, then press [F2] to go to the menu MEM, and then you press [F5] to go to the menu DIR. The fifth command in this menu is CRDIR.)

The capacitor's "resistance" Z to allow different frequencies to pass is given by the formula: $Z_c = \frac{1}{i 2 \pi f C}$, f being the frequency and C the capacity of the capacitor. As you can see, when the frequency is low (bass), the resistance will be high and vice versa. At the limiting case of $f = 0$ the resistance is ∞ which means that no DC will pass at all. When f (very high frequencies), then $Z_c = 0$ (frequencies pass without loss). The resistance of the whole circuitry is given by the sum of all resistors, since we have connected them in series:

Complex numbers with the HP49G - Part 4

$$Z_c = R + \frac{1}{i 2 \pi f C}$$

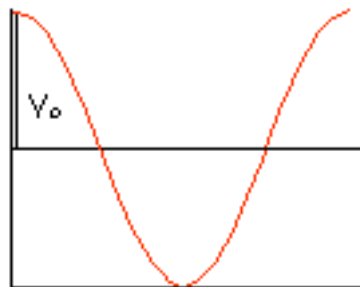
Enter the formula $Z = R + \frac{1}{i 2 \pi f C}$ in the equation writer and press [ENTER] to put it on the stack. Now press DEF ([blue-shift] and then [2]). A new variable Z will be created, which contains $R + \frac{1}{i 2 \pi f C}$. Noticed something? Well, normally we use DEF to define use functions. DEF takes an equation of the form $F(\text{arguments}) = \text{mathManipulationOfArguments}$, like for example $F(X) = \text{SIN}(X)^2$, and creates a program which looks like:

```
<< -> X
      'SIN(X)^2'
>>
```

It stores this program then in variable F in the current directory. But when the argument for DEF is an equation of the form $F = \text{mathManipulation}$, where F is simply a name with no arguments enclosed in parentheses, then DEF simply stores the right hand side in the variable at the left hand side. In our case, the equation was $Z = R + \frac{1}{i 2 \pi f C}$ and so DEF stored $R + \frac{1}{i 2 \pi f C}$ in Z.

Now, we know that the quantities R, f, and C are all real and greater than 0. So we make assumptions for them. Enter $R = 0$ and use ASSUME, to add this assumption in REALASSUME. Do the same with $f = 0$ and $C = 0$.

The amplifier delivers the voltage $V = V_0 \cos(2 \pi f t)$ where f is the frequency and t is time. (The HP49G includes t in the list REALASSUME per default. If for some reason you have deleted t from REALASSUME, then



enter t and use ADDTOREAL to put it back in the list of assumptions.) V_0 is the maximum of the sinusoidal voltage. The

current I at the outlets of the filter is given by: $I = \frac{V}{Z}$. In case of a short circuit at the outlets of the filter (no additional resistance), the current is: $I = \frac{V}{R + \frac{1}{i 2 \pi f C}}$.

Putting the voltage in this formula we derive: $I = \frac{V_0 \cos(2 \pi f t)}{R + \frac{1}{i 2 \pi f C}}$

Now, because the resistance is complex, we replace $V_0 \cos(2 \pi f t)$ with $V_0 e^{i 2 \pi f t}$ to make our work easier. When we derive some result we will always have to take the real part of this result using RE, because $V_0 \cos(2 \pi f t)$ is the real part of $V_0 e^{i 2 \pi f t}$.

Enter $I = \frac{V_0 e^{i 2 \pi f t}}{R + \frac{1}{i 2 \pi f C}}$ and press DEF to put the right hand side in a

variable named I. Press the soft menu button for I, to put the quantity $\frac{V_0 e^{i 2 \pi f t}}{R + \frac{1}{i 2 \pi f C}}$ (current) on the stack. Because V_0 is only a scaling

factor we can store some real value in it, like 1, 2.5, or any other real. For simplicity enter 1, then enter V_0 and press STO. (The number without decimal point is not a real for the HP49G, but an integer. But for now it is better to store an integer in V_0 to ensure symbolic results when we use algebraic manipulation commands.)

If your HP49G is in complex mode, switch to real mode. Now we

Complex numbers with the HP49G - Part 4

must take the real part of the quantity $\frac{V_0 e^{j2\pi f t}}{R + \frac{j}{2\pi f C}}$, so press RE.

After some seconds the HP49G returns:

$$\frac{\cos(2\pi f t) V_0 R + V_0 \sin(2\pi f t) \frac{-1}{C f}}{SQ(R) + SQ \frac{-1}{C f}}$$

Press EXPAND to convert this to:

$$-\frac{2\pi f C \sin(2\pi f t) - 4\pi^2 f^2 R C^2 \cos(2\pi f t)}{4\pi^2 f^2 R^2 C^2 + 1}$$

The simple sinusoidal voltage $V_0 \cos(2\pi f t)$ gives rise to this much more complicated current, which behaves like a linear combination of a sine and a cosine of $2\pi f t$. We remember that any sum of a sine and a cosine of the form $a \sin(x) + b \cos(x)$ can be

$$TCOLLECTed to \sqrt{a^2 + b^2} \cos(x + \theta) \text{ with } \theta = \frac{2 \operatorname{atan} \frac{y}{x}}{2}.$$

So now press TCOLLECT and wait for some seconds, wait a little bit longer, wait even longer and at the end the HP49G returns:

$$\frac{2\pi f C \sqrt{4\pi^2 f^2 R^2 C^2 + 1}}{4\pi^2 f^2 R^2 C^2 + 1} \cos 2\pi f t + \operatorname{ATAN} \frac{2\pi f C (4\pi^2 f^2 R^2 C^2 + 1)}{4\pi^2 f^2 R C^2 (4\pi^2 f^2 R^2 C^2 + 1)}$$

EXPAND this huge thing to get the expression:

$$\frac{2\pi f C \sqrt{4\pi^2 f^2 R^2 C^2 + 1} \cos \operatorname{ATAN} \frac{1}{2\pi f R C} + 2\pi f t}{4\pi^2 f^2 R^2 C^2 + 1}$$

We see that the voltage $V_0 \cos(2\pi f t)$ gives rise to a current with

amplitude $\frac{2\pi f C \sqrt{4\pi^2 f^2 R^2 C^2 + 1}}{4\pi^2 f^2 R^2 C^2 + 1}$, frequency f and *phase*

shift $\operatorname{ATAN} \frac{1}{2\pi f R C}$. We could of course store the real current

in I, but we would lose the already stored complex current. So first of all let's rename the existing variable I to CMPLI. Enter CMPLI (the new name first), enter 'I' in quotes and then enter RENAME. The already existing variable I is now renamed to CMPLI. Now enter I again and press STO. The real current is stored in variable I now.

Now, as already said, the amplitude of the current is given by

$$\frac{2\pi f C \sqrt{4\pi^2 f^2 R^2 C^2 + 1}}{4\pi^2 f^2 R^2 C^2 + 1} \text{ and the phase shift is given by}$$

$$\operatorname{ATAN} \frac{1}{2\pi f R C}. \text{ Because we will work with this quantities}$$

now, press the soft menu key for I, press [arrow down] to take the current to the EQW and using the arrow keys select

$$\cos \operatorname{ATAN} \frac{1}{2\pi f R C} + 2\pi f t. \text{ Then press CUT and}$$

[BACKSPACE] to remove this cosine. Press [ENTER] to put the

remaining expression $\frac{2\pi f C \sqrt{4\pi^2 f^2 R^2 C^2 + 1}}{4\pi^2 f^2 R^2 C^2 + 1}$ on the stack.

Store it in AMPLI.

Complex numbers with the HP49G - Part 4

Now, go to the EQW and press PASTE to put the expression

$\text{COS ATAN } \frac{1}{2 f R C} + 2 t f$ there. Then, select

$\text{ATAN } \frac{1}{2 f R C}$, press COPY and then [CANCEL] to return to

the stack. Press PASTE and [ENTER] to put $\text{ATAN } \frac{1}{2 f R C}$ on

the stack. Enter (key [ALPHA], then key [red-shift] and then key [COS]) and press STO to store the phase shift in variable .

Let's take a look at the amplitude. What happens with it when we go to very high frequencies? You guess right, we are going to find a limit. Press the soft menu key for AMPLI. Enter $f =$ and then press lim. Then wait a couple of seconds, drink a cup of coffee, go for a walk around the block, and when you come back the HP49G has news for you: The limit value of the amplitude of the current for very high frequencies is $\frac{1}{R}$. This is the result that the HP49G shows to you. But

it has also done some additional stuff behind your back. First of all take a look at the header of the screen. You can see that now the variable VX has been set to f (!). Enter the variable that was VX previously (it is likely that it was X) and then type STOVX and press [ENTER] to restore your VX. This is cumbersome but at least the HP49G shows what it has done. But there is also something that it didn't tell you anything about. Go to the directory CASDIR which resides in the HOME directory and take a look at REALASSUME. We have made the assumption $f = 0$, didn't we? But now the variable f is in the list REALASSUME *without any assumption*! The HP49G silently removed the assumption and just put the variable f in the list REALASSUME, presumably because the CAS works that way when finding a limit. This is not bad, but leaving the changed assumption in REALASSUME and not telling us anything about that, is not very nice, don't you think? So enter $f = 0$ and use ASSUME to restore our assumption. Go back now to the directory RCCIRCUIT.

Now, Nick wants at a frequency of 1000 Hz the amplitude of the current to be one half of the limit value for very high frequencies, that

is $\frac{1}{2 R}$. We can use this constraint to find C as a function of R.

Press the soft menu key for AMPLI. Press SWAP to bring the limit value on stack level 1. Press [2] and then [\div] to get one half of that value. Press [=] to make an equation. Now we must replace f with 1000. Enter $f = 1000$ and press SUBST. Now we can solve for C. Enter C and press SOLVE. After a short time the HP49G says: "Not reducible to a rational expression", which makes Nick very sad. But we don't give up! Let's go the dangerous way! Press SWAP to bring the equation on stack level 1, and then SQ to square both sides of the equation. (When we do such a thing, we must check our solutions for validity by back substitution.) Now, press EXPAND, SWAP to bring C on stack level 1, and then SOLVE. The HP49G goes then ratatat-ratatat and it returns the solutions list:

$$C = -\frac{1}{|6000| R} \sqrt{3} \quad C = \frac{1}{|6000| R} \sqrt{3}$$

The question that goes beyond my skills in philosophy is, why the absolute value |6000| isn't simplified to 6000 automatically, but as Trabakoulas says, the ways of the CAS are mystical! So press EXPAND to expand and simplify the two solutions, so that the list now is:

$$C = -\frac{\sqrt{3}}{6000 R} \quad C = \frac{\sqrt{3}}{6000 R}$$

Right away we can throw away the first solution, because we already made the assumption $C = 0$ and $R = 0$. (If R is greater than 0, then the first solution $-\frac{\sqrt{3}}{6000 R}$ is less than 0, which doesn't satisfy our assumption $C = 0$.) So the only candidate that we must check

Complex numbers with the HP49G - Part 4

through back substitution is the second solution $C = \frac{\sqrt{3}}{6000 R}$.

Extract the second solution by entering 2 and then pressing GET. The command GET is in menu PRG/LIST/ELEM. It can take a list, and an integer (or a real without fractional part), and returns the list element whose index is equal to the integer (or real). Now press DEF to store the solution $\frac{\sqrt{3}}{6000 R}$ in variable C. Now we must examine if this solution is OK. Press AMPLI and then EXPAND. Because $\frac{\sqrt{3}}{6000 R}$ is now stored in C in the current path, the command EXPAND uses that value whenever it encounters C in the expression of AMPLI. The result is $\frac{f}{\sqrt{f^2 + 3000000} R}$. We found the solution for the Nick's frequency $f = 1000$, so enter $f = 1000$ and press SUBST. Then EXPAND and you get $\frac{1}{2 R}$, which shows that the solution is correct. (Thanks goodness!)

Now, apart from the general expression for the amplitude and phase, we have also a special amplitude and special phase where $C = \frac{\sqrt{3}}{6000 R}$. We should store these special cases in separate variables, so that we have them handy when needed. Press AMPLI and EXPAND to get $\frac{f}{\sqrt{f^2 + 3000000} R}$ again. As you can see, this amplitude doesn't explicitly depend on C, because we already found C as a function of R using Nick's constraint. Enter N.AMPLI and press STO to store this special case. Now in the variables menu press [] and then EXPAND. The result is $\text{ATAN} \frac{1000 \sqrt{3}}{f}$, the expression for

the phase when C is equal to $\frac{\sqrt{3}}{6000 R}$. Store this in variable N.

And now it's time for some plots. We want to see what the curves of N.AMPLI and N. look like as functions of the frequency f. First of all press N.AMPLI to bring the expression $\frac{f}{\sqrt{f^2 + 3000000} R}$ on the stack. You of course have noticed that this expression also depends on R, so if we make a 2-D graph with the frequency f as independent variable and we don't somehow supply a value for R, the evaluation of the function for different frequencies will not return a number, and so nothing will be plotted. We are going to use several methods to supply such a value for R.

Let's start with the first method: Substitute several different values for and create a list of functions to plot, each one having a different value for R. Enter the list {'R = 1' 'R = 2' 'R = 3'} and press SUBST. This results in three distinct substitutions. The result is a list with three functions $\frac{f}{\sqrt{f^2 + 3000000} R}$, where in the first R has been replaced by 1, in the second R has been replaced by 2 and in the third R has been replaced by 3:

$$\frac{f}{\sqrt{f^2 + 3000000} 1} \quad \frac{f}{\sqrt{f^2 + 3000000} 2} \quad \frac{f}{\sqrt{f^2 + 3000000} 3}$$

Now, press STEQ to store this list of functions in EQ, a system reserved variable which contains the current function (or a list of functions) to plot. If there is something that has a huge number of options then this is graphing, so we are going to setup the plot first. Press simultaneously the keys [blue-shift] and [F4] to go to the screen PLOT SETUP. The first input field is labelled "Type:". If the plot type isn't already "Function", then press the menu key CHOOS ([F2]) and use the arrow keys to select this plot type. Press [Arrow down] twice

Complex numbers with the HP49G - Part 4

to go to the field labelled "Indep". Type 'f' and press [ENTER]. This makes f the independent variable. Check the options _Simult to plot the three functions simultaneously (and not one after the other) and _Connect to connect the plotted points (and not just plot the points unconnected). If the menu label AXES displays without a small square, press the corresponding menu key, so that the label appears with the small square. This indicates that the axes will also be drawn and also brings the additional input fields for options of the axes. Now, go to the field "H-Tick". We want a plot from $f = 0\text{Hz}$ to $f = 20000\text{Hz}$, so a tick on the horizontal axis every 1000 Hz seems reasonable. Enter 1000 in this field. For vertical axis tick "V-Tick" we use a value of 0.1 because we know that the amplitude goes for high frequencies to the value $\frac{1}{R}$, which for $R = 1$ is 1. Disable the option _Pixels because the data in the fields "H-Tick" and "V-Tick" are not pixels. We don't want an axis tick every 1000 pixels horizontally and every 0.1 pixel (!) vertically, but every 1000 Hz and 0.1A. (We plot amplitude of current against frequency.)

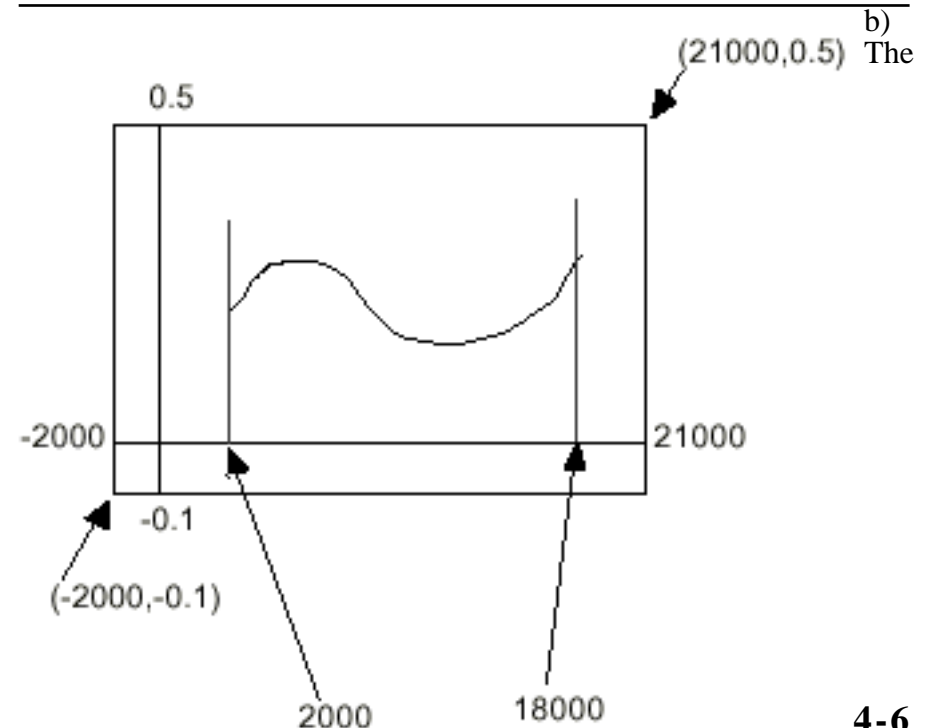
Now, press simultaneously the keys [blue-shift] and [F2] to go to the screen PLOT WINDOW - FUNCTION. Enter for "H-View" the values -2000 and 21000. That means that the plot window contains horizontally coordinates from -2000 to 21000. For "H-View" enter the values -0.2 and 1. Now, in the field "Indep Low:" enter 0 and in the field "High:" enter 20000. That means that the function will be plotted from $f = 0$ to $f = 20000$. Perhaps a bit of explanation is needed here. When you setup the values for "H-View" and "V-View" you adjust which part of the XY-plane will be visible on the screen, like shown in the picture to the right.

The fact that the horizontal coordinates of the viewing space goes from -2000 to 21000 doesn't also mean that the coordinates used for the independent variable also start at -2000 and end at 21000. One could use for example values for the independent variable from 2000 to 18000 (like in the picture) or any other values. The HP49G allows the user to specify separately viewing space and plot range. When we only setup "H-View" and "V-View" but don't specify "Indep Low" and "High",

then the values of "H-View" are used as independent variable range. When we do specify "Indep Low" and "High" then these are the values that are used as start and end of the independent variable range.

Now, we also want the axes to be labelled. Unfortunately the screens for plot setup don't include this capability of the HP49G. (Oh yes, it can do more with axes.) So the only built-in way to specify axes labels is the command AXES. This command changes the fifth element of the reserved variable PPAR, where plot parameters are stored, and so Bruce Willis uses it quite frequently. It can take as argument:

- 1) A complex number specifying the coordinates of axes intersection. When we give AXES a complex number as argument, then this number replaces the fifth element of PPAR.
- 2) A list containing any or all of the following elements:
 - a) A complex number that specifies the coordinates of axes intersection.



Complex numbers with the HP49G - Part 4

axes ticks parameters. This can be a real number or a binary integer for specifying the distance of ticks in user coordinates or in pixels. Or it can be a list with two real numbers or two binary integers for specifying separately distance of ticks on the x- and y-axis in user coordinates or in pixels.

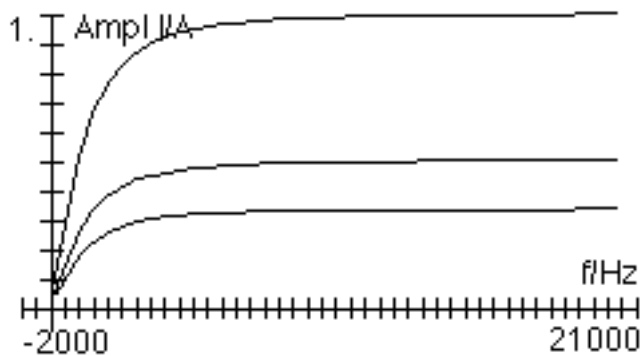
- c) A string to be used as label for the x-axis. If none specified then the independent variable is used per default.
- d) Another string that will be used as label for the y-axis.

When such a list is used as argument for AXES, then only the specified parameters are replaced in PPAR.

If you followed the instructions above you must be still in the screen PLOT WINDOW - FUNCTION. (Or your HP49G has shut off.) Exit the screen by pressing ENTER so that the settings so far are accepted. Now, let's make the list that we'll use as argument for AXES. We want the labels "f/Hz" and "Ampl. I/A" for the x- and y-axis, so we enter both strings. Then, we enter 2 and use the command ->LIST (menu PRG/LIST) to make a list out of these parameters.

And now the time has come to plot our functions. You can press simultaneously the keys [blue-shift] and [F2] to go to the screen PLOT WINDOW - FUNCTION again and then press the soft menu keys [ERASE] ([F5]) and [DRAW] ([F6]). This will ERASE previous contents of PICT and draw the plot according to our settings. You will be transferred to the PICTURE environment, where editing of the graph and further exploring can be done. But wait!

Where are the axis labels? Well, the command DRAW from the soft of the plot setup screens, only draws the functions and the axes (if drawing axes option was



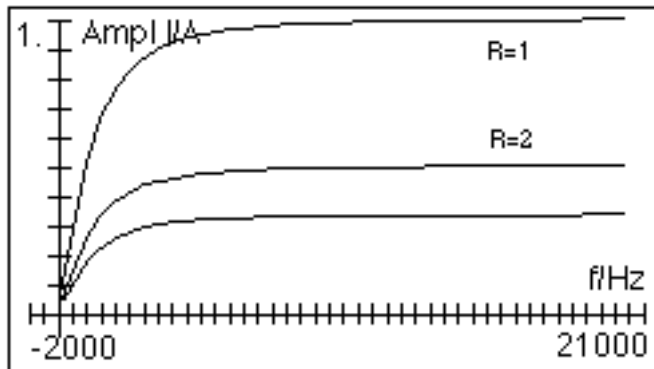
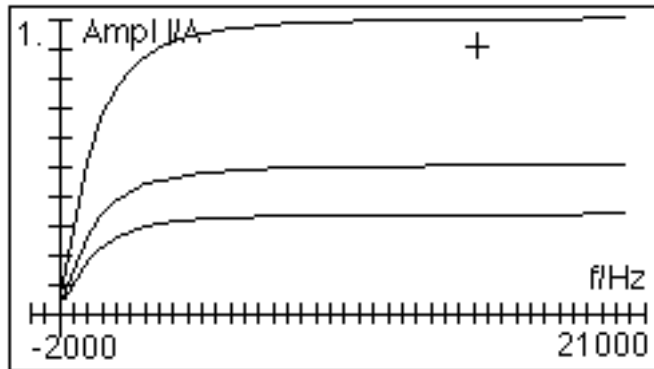
enabled.) But it doesn't draw the labels of the axes. When the drawing is done you are presented a menu with the menu item EDIT ([F5]). Press that menu key and then press [NXT] to go to the second page of the menu. Now you see the label "LABEL" over [F3]. Press this to get labels on the axes. Or you can use the command sequence ERASE DRAX DRAW LABEL and then PICTURE to do the same from the command line.

While in the PICTURE environment you can press the key [-] to let the menu disappear and see the plot in all its glory. Now, should we add annotations? We could for example add the value of R as annotation for each one of the curves. Press CANCEL until you are on the stack again. Enter the string "R=1" and then 1. Now press [blue-shift], [CAT], [NXT] and then [F1] (soft menu key for GROB menu). The first menu command is labelled "->GROB". We are going to use the command ->GROB to convert the string "R=1" to a GGraphs Object. This command takes the object that will be converted to a GROB from stack level 2, and the number 0 or 1 or 2 or 3 from stack level 1. This number determines the font size that will be used, that is 1 for mini font, 2 for medium size and 3 for big. The number 0 has a special meaning for algebraics. When an algebraic, like for example X^2 , is converted to a GROB, then the numbers 1, 2 and 3 cause the GROB to be in non-textbook form, that is X^2 will be converted to the GROB $X^{\wedge}2$. But when we supply a 0, then the GROB is in textbook form, that is X^2 will be converted to the GROB X^2 . When we use 0, then the size of the used font is determined by flag -81. When this flag is set, then the resulting GROB of the algebraic will be created using mini font, that is X^2 will be converted to the GROB x^2 . When this flag is clear, then the GROB of the algebraic object will be created using the current font.

So, now that we have the string "R=1" and a 1 on the stack, we press the soft key for ->GROB and the GROB $R=1$ is returned to the stack. We must put it on the graph now. So press [arrow-left] to activate the PICTURE environment again. Press [F5] (soft key from EDIT menu), then press [NXT] twice to go to the third page of the menu. Use the arrow keys to take the cross hairs to the point indicated at the

Complex numbers with the HP49G - Part 4

following picture. (It doesn't have to be exactly there, just put it under the highest curve.) Press [F5], the soft key for REPL. This replaces the current PICT with the GROB at stack level 1, starting at the coordinates of the cross hairs. The part of the PICT that will be replaced starts at the coordinates of the cross hairs and ends at coordinates determined by the size of the GROB on stack level 1 in pixels. We will add the annotation "R=2" using



commands from the stack. To do so, we need the GROB to which we are going to add the annotation, that is the current PICT. Press STO. When in the PICTURE environment, pressing STO copies the current PICT to the stack. Now, move the cross hairs with the arrow keys until it is some pixels over the second curve. Press [ENTER]. When in the PICTURE environment pressing [ENTER] just puts the current coordinates of the cross hairs on the stack as a complex number. (We need this for our annotation.) Press CANCEL until you are on the stack again. Now the stack contains a copy of the current PICT and a complex number. Enter "R=2", 1 and press ->GROB. Now press REPL. The result is a GROB that results when we replace a portion of the GROB on stack level 3 with the GROB on stack level 1 starting at the coordinates on stack level 2. REPL. We have seen on page 2-29 that

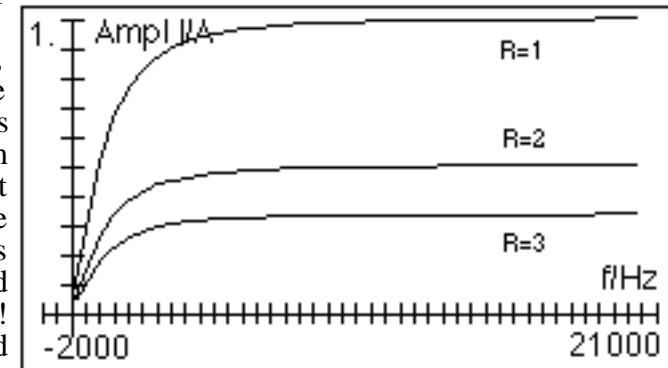
this command also replaces characters in strings. And it can also replace objects in lists and elements in matrices. Once again we see that many commands are thought for multiple kind of arguments. They act according to the argument types that they are fed with. Kind of overloading, so to speak.

The resulting GROB with the two annotations is only on the stack but not in PICT, because we worked on a copy of the PICT on the stack. To put this GROB in the current PICT, just enter PICT (without quotes!) and press STO. Now the PICT contains the GROB with the two annotations.

Before we add the third annotation, let's look at this thingie, the PICT. What is this? Well, PICT is a command (object type 19.) but it also acts like a variable. A special kind of variable, that is. You can recall PICT, or store a GROB in it, but it never shows in the variables menu. Think of it like of a shortcut for "the current contents of the graphics environment". It is not the same like a GROB on the stack. When you use it, remember to use it without quotes.

Now, let's go on with the third annotation. First enter PICT without quotes. Then press [arrow-left] to go to the graphics (or PICTURE) environment. Use the arrows to position the cross hairs somewhere under the lowest curve and press [ENTER] to put the coordinates on the stack. Press [CANCEL] to exit the graphics environment. Enter "R=3", then 1

and press ->GROB. Now, press REPL. The three arguments disappear from the stack, but where is the result? Press [arrow-left] and be astonished! The third annotation was



Complex numbers with the HP49G - Part 4

put in PICT. So, if you think of PICT like "the current contents of the graphics environment" or perhaps even better as a pointer to those contents, it is easy to understand what happened. We gave the command REPL the special argument PICT, which "points" to the contents of the graphics environment. So it replaced the appropriate portion of those contents with the annotation GROB on stack level 1, starting at the coordinates at stack level 2. The plot now contains three curves and axis labels and annotations for the curves.

Press now [F3] to start tracing the plotted curves. The menu label "TRACE" becomes now "TRAC•" to indicate that tracing is activated. Press [F2] to see the current coordinates of the cross hairs. You can move the cross hairs to higher or lower x-coordinates with the keys [arrow-left] and [arrow-right]. The displayed x- and y-coordinates are updated accordingly. If you keep the arrow keys pressed, then the cross hairs move continuously along the first curve. You can use the keys [arrow up] and [arrow down] to jump from one curve to the other. If you press [blue-shift] and then [arrow-down] then the currently traced equation is shown for about 1 second on the top of the screen.

Should we photograph this plot as evidence for the capabilities of the HP49G? OK, press STO and then [CANCEL] to return to the stack. The plot has been copied to stack level 1. Now you can store it in some variable so that you can view it again later.

When we plot such functions, like $\frac{f}{\sqrt{f^2 + 3000000} R}$, that in addition to the independent variable contain one parameter, then we can also do the following. Declare the parameter to another independent variable and do a 3-D plot. We can do a 3-D plot with independent variables f and R in this case. Recall the current equation on the stack and press HEAD to get the first element of the list. Edit the expression $\frac{f}{\sqrt{f^2 + 3000000} n}$ (where n represents one of the numbers 1, 2 or 3 - when you jumped back and forth between the equations the list of equations in EQ was rotated accordingly - and replace this n with our

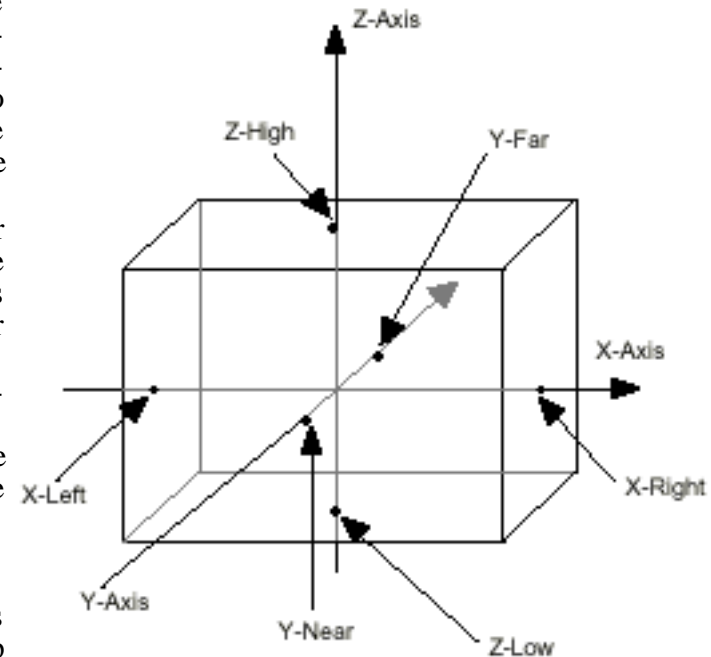
parameter R , so that the expression becomes again

$\frac{f}{\sqrt{f^2 + 3000000} R}$. Now press STEQ to store this in EQ. There are

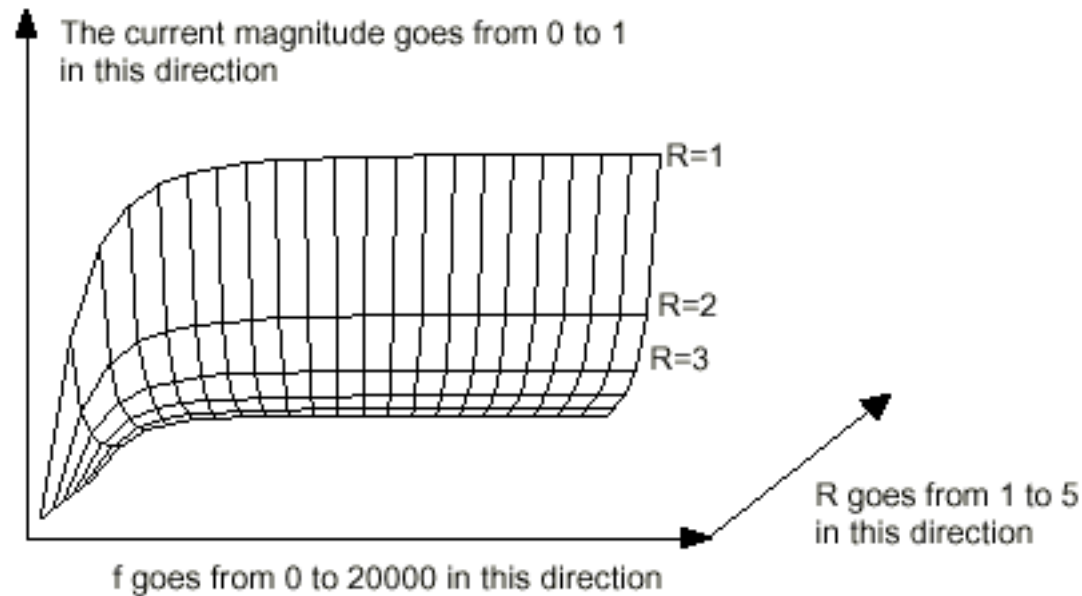
several types of 3-D plots built-in. FAST-3D is presumably the most widely known. But we are going to use the plot type WIREFRAME first, so that this plot type doesn't get unemployed.

Press [blue-shift] and [F4] simultaneously to get the PLOT SETUP screen. Choose WIREFRAME plot type, f for independent and R for dependent variable. Don't worry about R being called "dependent". For 3-D plots this input field just means the second independent variable. Press now [blue-shift] and [F2] simultaneously to get the PLOT SETUP - WIREFRAME screen. The input fields X-Left, X-Right, Y-Near, Y-Far, Z-Low and Z-High determine the view volume, as shown on the picture on the next page. Enter for X-Left, 20000 for X-Right, 1 for Y-Near, 5 for Y-Far, -0.2 for Z-Low and 1 for Z-High. The inputs for X-Left and X-Right are also used as range of the variable f . The inputs for Y-Near and Y-Far are also used as range for variable R . Z-Low and Z-High determine the height of the viewing space.

Now it is time to setup

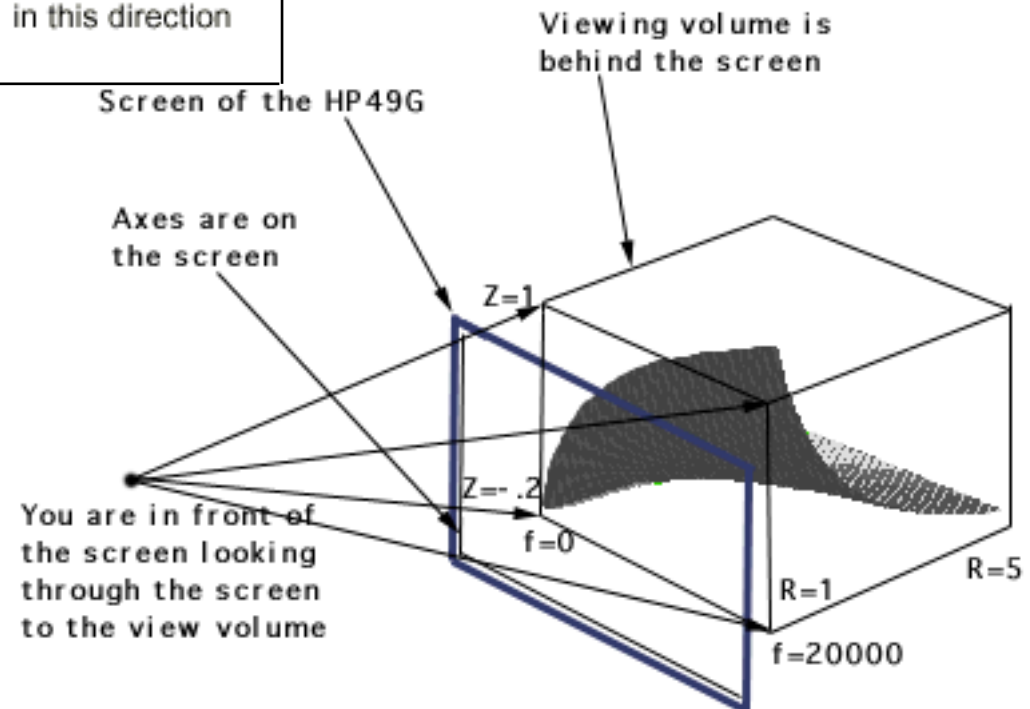


Complex numbers with the HP49G - Part 4



$R = 1$, $R = 2$ and $R = 3$ are annotated accordingly. Think of the curve with $R = 1$ as being nearer to you, $R = 2$ a bit further away from you, $R = 3$ even further away and so on. We can of course try to put some annotations, but this can get harder than for simple function plots. We can of course make the GROB $R = 1$, then go to the PICTURE environment, move the cross hairs at the appropriate position and press REPL. But if we use the command REPL programmatically, then we can't just use the coordinates, say $X = 20000$ and $Y = 1$. You see, the HP49G has two separate reserved variables for 2D- and 3D-plots. The reserved variable for 2D-plots is the well known PPAR. The reserved variable for 3D-plots is the not so well known VPAR. When the HP49G draws a wire frame plot it uses VPAR (and some

the parameters XE, YE and ZE, which are the coordinates of the point in space where the observer of the plot (you) sits and looks at the plot. Enter 10000 for XE, -10 for YE and 0 for ZE. One restriction for YE is that you can't enter a number that is greater than Y-Near, that is you can't sit in the viewing space or behind it. The input fields "Step Indep" and "Depnd" are the number of steps in X- and Y-direction. We want in X-direction a plot point every 1000 Hz, so enter 21 for "Step Indep". Similarly enter 5 for "Depnd" so that we have one plot point every 1 Ohm. Press [F5] (ERASE) and [F6] (DRAW). The HP49G starts with big Y-coordinates, that is it first plots the function for $R = 5$ and $f = 0$ to $f = 20000$. Then it draws the function for $R = 4$ and $f = 0$ to $f = 20000$, and so on, until the 1st curve for $R = 1$ and $f = 0$ to $f = 20000$. When ready the plot reminds us of our first plot. But you see the limitations. There are no axes, no labels, nothing that helps you to understand what is front and what is back. So I put some additional things on the picture above, to make clear what this thing represents. The curves of constant



Complex numbers with the HP49G - Part 4

parameters in PPAR). But when it executes such commands like REPL then it finds the location the replacement by using PPAR. So you could think that placing the same values for view volumes in VPAR and PPAR would do the trick, but there are more difficulties. The values of the view range of the Y-coordinate in PPAR are meant from the bottom to the top of the screen, while in VPAR are meant from nearer to the observer to the virtual depth behind the screen. The values for the Z view volume in VPAR are meant (almost) from the bottom to the top of the screen, so they correspond physically (that is for the screen) to the Y view volume of PPAR. The plotting of a 3D-surface on a 2D-medium like a screen is always made by means of some type of projection, that is to find a point with given coordinates X, Y and Z on the 2D-medium we must calculate the projection of its real coordinates on the two dimensional coordinates system of the medium. We can of course set the X view volume of PPAR from 0 to 20000 and the Y view volume of PPAR from -0.2 to 1 (so that it is identical with the Z range of VPAR). Then we can use DRAX to draw axes, so that we at least have the X- and the Z-axis. But, big but, very big but: The axes that we draw are meant as if they were on the screen, while the surface is behind the screen. Because the perspectivistic shift of the axes and the plot are not the same, the position of the axes will not be correct as axes of the 3D-plot.

While we are talking about view volumes for 2D- and 3D-plots, the explanation of the contents of PPAR and VPAR and how they are used for the plot types that we talked about so far, becomes more and more important. So let us examine these reserved variables and the commands that can be used to programmatically setup the parameters in these variables.

Let's start with PPAR and how its contents are used with the FUNCTION type plot. PPAR is a list with 7 items:

$\{(x_{\min}, y_{\min}) (x_{\max}, y_{ax}) \text{ indep res axes ptype depnd}\}$

For the plot type FUNCTION these elements have the following meaning:

(x_{\min}, y_{\min})

A complex number which specifies the lower left corner of the display range. Default value is $(-6.5, -3.1)$. The programmable command for setting this parameter is PMIN. This command takes a complex number from the stack and puts it in the first position of PPAR.

(x_{\max}, y_{ax})

As you might have imagine, a complex number which specifies the upper right corner of the display range. Default value is $(6.5, 3.1)$. The programmable command for setting this parameter is PMAX. This command takes a complex number from the stack and puts it in the second position of PPAR.

There are two additional commands for setting up the display range. The first is XRNG. It takes two real numbers from the stack (the minimum and maximum of the view range of the X-axis) and sets up the parameters x_{\min} of (x_{\min}, y_{\min}) and x_{\max} of (x_{\max}, y_{ax}) . The other is YRNG and as you can think it does the same for the Y-axis.

indep

This element can be either a name specifying the independent variable of the expression that we want to plot. Or, as we already have seen in our function plot with annotations, it can be a list which contains the name of the independent variable and the minimum and maximum of the plotting range. This allows to have different values for the viewing and plotting range. Default for this parameter is X.

There are two programmable commands that can be used for setting this parameter. We have the command INDEP, which can take as arguments:

The name of the independent variable. If a name is

Complex numbers with the HP49G - Part 4

given to INDEP then this name replaces the third element of PPAR, that is if you already have specified a plotting range, then this will be lost and the viewing range from the parameters (x_{\min}, y_{\min}) and (x_{\max}, y_{\max}) will be used.

A list which contains the name of the independent variable. In this case the independent variable is replaced but an existing plotting range will not be touched.

A list with the name of the independent variable, a real number that specifies the minimum of the plotting range and a real number that specifies the maximum of the plotting range.

A list with two numbers specifying the minimum and maximum of the plotting range. The independent variable remains untouched.

Or lastly two real numbers that specify the minimum and the maximum of the plotting range. The independent variable remains untouched.

res

A real number that specifies the resolution of the plot in user coordinates or a binary integer that specifies the resolution in pixels. Default value is 0, which on the HP49G means a plot point every two pixels. The command that sets this parameter is RES and it can take either a real or a binary integer as argument. If the argument is a real number r , then the HP49G plots a plot point every r units of horizontal coordinate the plot. If it is a binary integer n , then it plots a plot point every n pixels of horizontal direction the plot.

axes

This element is either a complex number specifying the coordinates of intersection of the axes. Or a list that has one or more of the following elements in order. A complex number specifying the coordinates of

intersection of the axes, a list that specifies the tick marks of the axes and two strings that are used as labels for the X- and the Y-axes. Commands for this parameter are:

AXES, which takes as arguments a complex number representing the coordinates of axes intersection, or a list that has the parameters listed above.

ATICK, which sets up the distance between tick marks on the axes. This command takes as arguments either a real number that specifies the distance between tick marks in user units for both axes, or a list with two real numbers that specify this distance separately for the X- and Y-axis, or a binary integer that specifies the distance between tick marks in pixels for both axes, or a list with two binary integers that specify this distance separately for the X- and Y-axis.

ptype

One of the plot types available on the HP49G out of the box. These are: BAR, CONIC, DIFFEQ, FUNCTION, GRIDMAP, HISTOGRAM, PARAMETRIC, PARSURFACE, PCONTOUR, POLAR, SCATTER, SLOPEFIELD, TRUTH, WIREFRAME, YSLICE and FAST-3D. The commands for setting the appropriate plot type are the same like the parameters above, that is, if you want to set up the plot type function from a program, you just enter the command FUNCTION.

depnd

A name that specifies the dependent variable. The command is DEPND and its arguments are the same as the arguments for INDEP. Note that a plot range for the independent variable is only used for the plot type TRUTH but is ignored otherwise. Default is Y.

Let's now move on to VPAR. This is the reserved variable where the parameters for 3D-plots are stored. (Well, not only 3D but any other

Complex numbers with the HP49G - Part 4

plots that have two independent variables.) Its contents is a list of the form:

x_{Left} x_{Right} y_{Near} y_{Far} z_{Low} z_{High} x_{min} x_{max} y_{min} y_{max}
 x_{Eye} y_{Eye} z_{Eye} x_{Step} y_{Step}

For the plot type WIREFRAME these elements have the following meaning:

- x_{Left} and x_{Right} : Real numbers that specify the start and the end of the view space in X direction, that is the width of the view space. Corresponding command is XVOL which takes two real numbers as arguments.
- y_{Near} and y_{Far} : Real numbers that specify the start and the end of the view space in Y direction, that is the depth of the view space. Corresponding command is YVOL which takes two real numbers as arguments.
- z_{Low} and z_{High} : Real numbers that specify the start and the end of the view space in Z direction, that is the height of the view space. Corresponding command is ZVOL which takes two real numbers as arguments.
- x_{min} and x_{max} : These parameters are (unfortunately) not used by the plot type WIREFRAME. Corresponding command is XXRNG.
- y_{min} and y_{max} : These parameters are (also unfortunately) not used by the plot type WIREFRAME. Corresponding command is YYRNG.

Because the last four parameters are not used by the plot type WIREFRAME, it is not possible to set up a different view and plot range.

x_{Eye} , y_{Eye} and z_{Eye} : Real numbers which represent the point in space from which the plot is viewed. Corresponding command is EYEPT which takes three real numbers as arguments.

x_{Step} and y_{Step} : Real numbers that specify the number of plot points in X and in Y direction. The two commands for these parameters are NUMX and NUMY which take a real number as argument.

The plot type wire frame also uses some parameters of the variable PPAR. These are, the parameter indep which specifies the name of the first independent variable (X variable), depnd which specifies the name of the second independent variable (Y variable) and PTYPE which for this plot type is WIREFRAME.

Now that we know what the parameters VPAR and PPAR are and how they are used for the plot type WIREFRAME, we can make an animation that shows how the looking of our plot changes, when move, that is when the parameters x_{Eye} , y_{Eye} and z_{Eye} change. For example let as move from $x_{\text{Eye}} = 10000$ to $x_{\text{Eye}} = -10000$ in steps of 2000 and keep on looking at the plot. We make a small program that draws a frame of the animation for each value of x_{Eye} and then animates the frames.

```
<<
'f/( (f^2+300000)*R)' STEQ @store equation
'f' INDEP 'R' DEPND @setup 1st. and 2nd.
@independent variables.
WIREFRAME @setup plot type
0. 20000. XVOL @setup view range
1. 5. YVOL
-0.2 1 ZVOL
10000. -10000.
FOR I @start loop
```

Complex numbers with the HP49G - Part 4

```

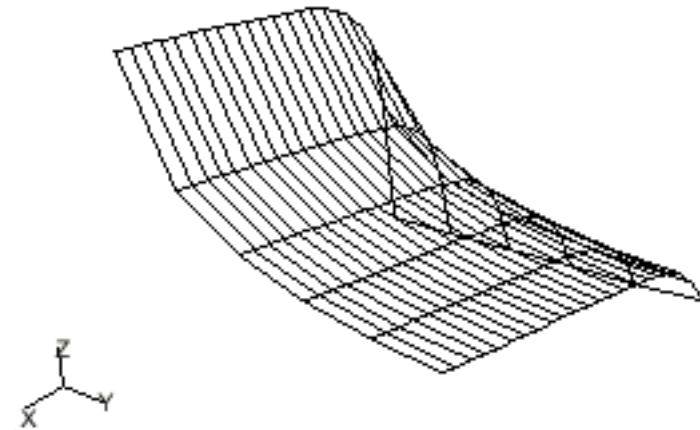
I -10. 0. EYEPT      @setup eye point
ERASE DRAW           @draw function
PICT RCL             @copy picture to stack
-2000
STEP                 @decrement I by 2000

```

>>

Let the program run and sing along "movie star" while the frames are made. When tired of singing, drink a cup of coffee while you look excited how the HP49G goes to Hollywood. Hey! we are making a movie. And like any good movie, our movie just takes time to be made;-) About 15 minutes. When the program finishes then you have 11 GROBs on the stack, each one is a frame of the animation. Now enter 11 and then press ANIMATE. (Press [blue-shift], [CAT], [NXT], soft key [GROB], [NXT] and the soft key [ANIMA].) Enjoy your travel looking at the 3D- surface. When you're tired press [CANCEL]. The animation stops but the GROBs and the number 11 are left on the stack. (We wouldn't like to start making the movie again;-) The command ANIMATE can take as argument either a number specifying the number of frames to be animated, or a list with the following elements: The number of frames, a list containing two binary integers that specify the pixel coordinates of the screen where you want to animate the GROBs, a number specifying the delay between each frame in seconds and a number that specifies how many times the animation should run. If this last number is 0., then the animation is played one million times or until you press [CANCEL]. To have an example of the second type of argument, drop the number 11 from the stack and enter the list {11 {#20d #20d} .1 5}. Press ANIMATE and look how the animation area starts at coordinates #20d (20 pixels from the left edge of the screen) and #20d (20 pixels from the top edge of the screen). This can be very useful when you have GROBs that are smaller than #131 pixels wide and #64 pixels high (the size of the screen) and you want to animate them on a specific area of the screen, while leaving some other part of the screen contents (stack or PICT) visible.

Because the plot type FAST-3D is the "modern" variant of a 3D plot on the HP49G, let's give it a try with the same function. Press

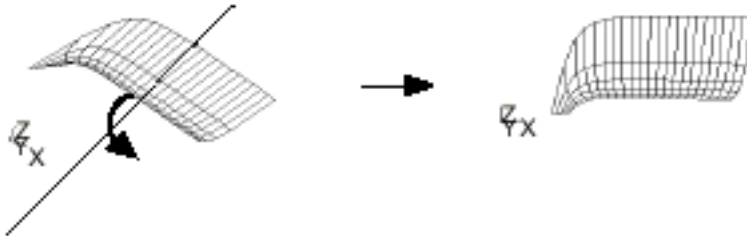


simultaneously [blue-shift] and [F4] to go to the PLOT SETUP screen, and choose FAST-3D plot type. Now press simultaneously [blue-shift] and [F2] to go to the PLOT WINDOW - FAST3D screen. The viewing parameters are set from the last plot, so press the soft key DRAW. A nice processing bar appears, which shows how much of the plot has been calculated. This plot is made much faster than the WIREFRAME plot. When the HP49G finishes calculating, it displays the graph, along with a cartesian coordinates system at the lower left corner of the screen to help you a little bit understanding what is front and what is back. So you can see that the values for the x coordinate (frequency f) grow in the direction somewhere in the middle to lower left. The values for the y coordinate (resistance R) grow in the direction somewhere in the middle to lower right. And the values for the z coordinate (amplitude of current) grow in the direction somewhere in the middle to the top of the screen. This kind of plot allows rotating using the arrow keys and zooming in and out using [+] and [-]. Let's rotate the plot so that it has approximately the same orientation in space like the previously made



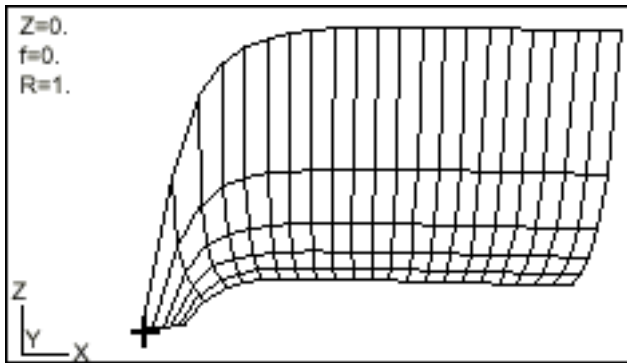
Complex numbers with the HP49G - Part 4

WIREFRAME. Hold the key [arrow-right] press to rotate the plot along the perpendicular axis of the viewing space. This axis of the rotation is not the Z axis of the plot, though for certain orientations of the plot it can coincide with it. Now keep the key [arrow up] pressed to rotate again. The



plot has now (almost) the same orientation as the previously made WIREFRAME. You can trace also this kind of plot. Press the menu key TRACE. The cross-hairs cursor appears at $f = 0$, $R = 1$, $Z = 0$ and the

coordinates are shown at the top left corner of the screen. Pressing [arrow-down] moves the cross hairs to higher frequencies and pressing [arrow-up] to lower frequencies. (Quite the opposite of what you might expect but anyway,



expectations are not always the right method working with the HP49G ;-)). Pressing [arrow right] moves the cross hairs to higher values of R , while pressing [arrow left] moves it to lower values. In this tracing mode zooming in and out using the keys $[+]$ and $[-]$ is still possible, so as Cow Dundee Trabakoulas says, use it mate!

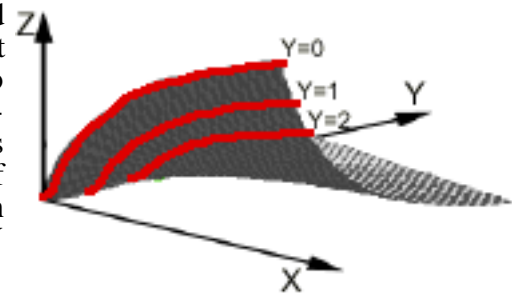
If you exit this plot by pressing [CANCEL] or the menu key [EXIT] you are transferred to the PICTURE environment and the plot can be edited. You can press the menu key [EDIT] and start adding annotations, replacing parts of the plot and so on. If you press [CANCEL] again, then you are transferred back to the PLOT

WINDOW - FAST3D screen. Note also that when a FAST3D plot is made, it simply overwrites any previously made plots. So if you want to overlay several plots, then you must press STO while you are in the PICTURE environment. (Not the environment where the plot is initially shown!!!) Or do PICT RCL from the command line after the plot has been done. Then you can use the command REPL to overlay this GROB with other GROBs.

Though this kind of plot is much faster than WIREFRAME it also has its limitations. The eye point parameters are not used at all, so you can't view the plot from different points in space. This is not so bad because you can imitate the view from different points by rotating. But you can't imitate the looking of the plot while "flying by", you can only rotate it. Talking about limitations, here comes the one I consider worst. There is no way to programmatically make an animation of a rotational movement of the plot. You may alter the parameters in VPAR as you like, but when you issue the command DRAW, the plot is always drawn in the same orientation. Rotational movements of FAST3D plots are only for interactive use.

So WIREFRAME can be viewed on fly by, while the FAST3D plot can be viewed "flying around it". The question is why both things were not combined in a single more powerful type of plot. Questions, questions and no answers ;-)

We continue with another plot type that can be used for expressions that depend on two variables, the plot type Y-SLICE. This is kind of a mixture of a 3D and 2D plot. It works by plotting the curves of a three dimensional surface that have constant values of the second independent variable. It "cuts" slices perpendicular to the Y-axis, thus the name Y-SLICE. The plot starts drawing with the curves of higher Y values and goes on drawing curves with lower Y values.



Complex numbers with the HP49G - Part 4

Let's plot our function with a plot type Y-SLICE. Go to the PLOT SETUP screen and choose a plot type Y-SLICE. In this screen check the option _Save Animation so that all GROBs with the curves are saved after the animation has finished. Now, go to the PLOT WINDOW - Y_SLICE screen and set the values for X-Left: 0. , for X-Right: 20000 , for Y-Near: 1 , for Y-Far: 5 , for Z-Low: -0.2 and for Z-High: 1. Also, enter for Step Indep 21 (one plot point every 1000 Hz) and for Depnd: 5. (one curve) every 1 Ohm. Press the menu keys [ERASE] and [DRAW]. Five curves are drawn and then animated. If you press [CANCEL] while the animation runs, then the currently shown frame remains on the screen and you can trace it by pressing the menu key [TRACE]. Use the keys [arrow left] and [arrow right] to move along the curve. You can also have the current coordinates of the cross hairs by pressing the menu key [(X,Y)]. If you want to show the menu again you can press again [F2]. Press [F3] to leave trace mode. While in trace mode you can use the keys [arrow down] and [arrow up] to jump to curves with lower or higher values of the Y coordinate (R in this case). The cross hairs jumps to the point of the next curve that has the same X coordinate (in our case the same frequency), but unfortunately the curve itself is not shown. :(If you now exit the plotting environment you'll have to answer the question, "I did check the option _Save Animation. Why weren't the frames of the animation saved somewhere?" Well, that's a good one. For which my logic isn't sufficient to provide an answer. Another good question is why the parameter for Step Indep: is no more 21 when we exit the plot environment, but is set to 20 instead.

How are the parameters of VPAR and PPAR used for this plot type? In VPAR we have:

x_{Left} x_{Right} y_{Near} y_{Far} z_{Low} z_{High} x_{min} x_{max} y_{min} y_{max}
 x_{Eye} y_{Eye} z_{Eye} x_{Step} y_{Step}

x_{Left} and x_{Right} Real numbers that specify the start and the end of the view space in X direction, that is the width of the view space. Corresponding command is XVOL which takes two real numbers as arguments.

y_{Near} and y_{Far} : Real numbers that specify the start and the end of the view space in Y direction, that is the depth of the view space. Corresponding command is YVOL which takes two real numbers as arguments.

z_{Low} and z_{High} : Real numbers that specify the start and the end of the view space in Z direction, that is the height of the view space. Corresponding command is ZVOL which takes two real numbers as arguments.

x_{min} and x_{max} : These parameters are (unfortunately) not used by the plot type Y-SLICE. Corresponding command is XXRNG.

y_{min} and y_{max} : These parameters are (also unfortunately) not used by the plot type Y-SLICE. Corresponding command is YYRNG.

Because the last four parameters are not used by the plot type Y-SLICE, it is not possible to set up a different view and plot range.

x_{Eye} , y_{Eye} and z_{Eye} : Not used with Y-SLICE.

x_{Step} and y_{Step} : Real numbers that specify the number of plot points in X and in Y direction. The two commands for these parameters are NUMX and NUMY which take a real number as argument.

Let's test the what happens with the parameter Step Indep and the frames of the animation if we do all from the command line. Enter 21, use NUMX and then ERASE and DRAW. When the animation is displayed press [CANCEL] to stop it and return to the stack. Aha! We have the number of the frames (5.) in stack level 1 and the frames on stack levels 2 to 6. That means we can save the frames for later use by doing the plot from the command line. Now, recall VPAR and check the second last parameter. It still is 21, while using the PLOT

Complex numbers with the HP49G - Part 4

WINDOW - Y-SLICE screen it was changed to 20. So the graphical interface plays games with us, allowing us to check the option _Save Animation and then not saving the frames, and also allowing us to enter 21 for the parameter Step Indep and then setting it back to 20 behind our backs. My Jedi Lords and Knights, I may mention that this behaviour is not the expected behaviour, but as already said at least 19265 times, working with the HP49G often means to forget your expectations. Let's take a look at the parameters of PPAR when the plot type is Y-SLICE. PPAR is:

$\{(x_{\min}, y_{\min}) (x_{\max}, y_{\max}) \text{ indep res axes ptype depnd}\}$

Of these parameters Y-SLICE only uses:

- indep** A name specifying the independent variable of the expression that we want to plot. Default for this parameter is X. The command INDEP can be used for setting this parameter.
- res** A real number or a binary integer that specifies the resolution of the plot in user coordinates or in pixels. Default is 0, which means a plot point every two pixels. The command that sets this parameter is RES and it can take either a real or a binary integer as argument.
- ptype** One of the plot types available on the HP49G out of the box. These are: BAR, CONIC, DIFFEQ, FUNCTION, GRIDMAP, HISTOGRAM, PARAMETRIC, PARSURFACE, PCONTOUR, POLAR, SCATTER, SLOPEFIELD, TRUTH, WIREFRAME, YSLICE and FAST-3D. The commands for setting the appropriate plot type are the same like the parameters above, that is, if you want to set up the plot type function from a program, you just enter the command YSLICE.
- depnd** A name that specifies the second independent variable. Default is Y. The command for this parameter is DEPND.

parameters in PPAR. We must set up the parameters (x_{\min}, y_{\min}) and (x_{\max}, y_{\max}) to be exactly the same like x_{Left} , z_{Low} , x_{Right} and z_{High} of the variable VPAR. Enter 0 and 20000 and issue the command XRNG, then enter -0.2 and 1 and issue the command YRNG. Now, if we ERASE DRAX and DRAW again, the axes will be drawn but before drawing the first frame they will be erased again. If we ERASE, first DRAW and then DRAX, the axes will be drawn after all slices are drawn, so the axes will be drawn only on the slice shown when we press CANCEL during the animation. And in addition to this, it is not the slice with the axes that is returned to the stack but the slice before the axes were drawn. That means: The HP49G draws all slices and starts animating. Then you press [CANCEL] and the HP49G stops the animation and immediately returns all the frames of the animation to the stack. Then it draws the axes but now it is too late, because the frames that we need are already on the stack. Well, how can we then get the axes on all frames? We use the simple plot type FUNCTION. We can make a little program that draws one curve after the other with axes and anything else that we want and returns each drawn plot to the stack:

```
<<
'f/(+(f^2+300000)*R)' STEQ @store equation
'f' INDEP @setup independent var.
FUNCTION @setup plot type
0. 20000. XRNG @setup view range
-0.2 1. YRNG
{ 1000. 0.2} ATICK @setup axes ticks
0. RES @setup resolution
5. 1.
FOR I @start loop
  I 'R' STO @store value in R
  ERASE DRAX DRAW @draw function and axes
  PICT RCL @copy picture to stack
-1
STEP @decrement I by 1
>>
```

What about axes? Well, we can draw them correctly using the

Complex numbers with the HP49G - Part 4

When the program finishes running you have 5 GROBs on the stack. If you enter 5. and press ANIMATE, then the same animation like the animation of Y-SLICE will be shown, but this time with axes. Sometimes the solution is achieved by following the non-standard way.

Until now we have plotted the amplitude of the current as a function of the frequency and the resistance, so it is time to plot the phase of the amplitude which is stored in variable N. . Go to the PLOT SETUP screen and choose plot type FUNCTION. In the input field EQ: enter 'N. ' with quotes. (Yes, we can also store names that contain expressions in EQ.) Enter 1000. for H-Tick. The phase goes from 0 to

$\frac{\pi}{2}$, or from 0 to 90 degrees. We want a tick on the Y axis every 10

degrees but if we are at RAD mode how can we enter 10 degrees easily without changing mode? Well, go to the input field V-Tick, type 10 and then press [blue shift], [SYMB], menu key [REAL], [NXT], [NXT] and then menu key [D->R] (degrees to radians). Now the command line contains 10 D->R. Press [ENTER]. The whole command line is evaluated and the resulting number 0.174... is put in the input field V-Tick. Lastly, uncheck the option _Pixels.

Now, go to the PLOT WINDOW - FUNCTION screen and set H-View from 0 to 20000, V-View from -0.2 to 1.57 (you can type $\frac{\pi}{2}$ / and then press ->NUM to enter the approximate number for $\frac{\pi}{2}$ - the symbolic itself won't work as the plotters need numeric parameters for the viewing volume) . Press [ERASE] and [DRAW] to plot. Here you have the plot of phase against frequency.

But wait! Until now we have plotted the current amplitude (in a dozen of ways) and the phase alone. Could we somehow combine the two curves in one single plot, so that we can see both dependencies on frequency at once? You guess it right, we can. And we can do it with so many methods that it really becomes difficult to choose.

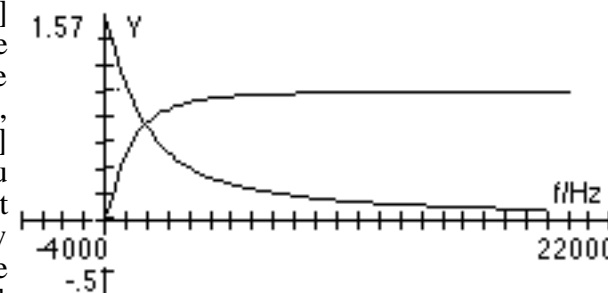
The easiest method that we can use is to put the two expressions for current amplitude and phase in a list and store this list in EQ. Go to the

variables menu, press N. and N.AMPLI and then enter 2 and use the command ->LIST. Now press STEQ. And now the settings. First we setup the axes from the stack because the setup screens don't contain all possible settings. Enter the list $\{(0.,0.) \{1000. .2\} "f/Hz" ""\}$ and then enter AXES. We have

set axes intersection at the point (0.,0.) x-axis ticks every 1000 Hz, y-axis ticks every 0.2 units, x-axis label to "f/Hz" and no label (empty string) for the y-axis. We don't use a label for the y-axis because the one function is current amplitude and the other is phase. Enter 'Y' and press DEPND to set Y as the dependent variable, because at the previously made plots it has been set to R. Because the label for the y-axis has been set to an empty string, the name of the dependent variable, that is Y, will be used as y-axis label. Go to the PLOT SETUP screen and check that the parameters are: f for INDEP, 1000 for H-TICK, 0.2 for V-Tick. Now go to the PLOT WINDOW - FUNCTION and check that H-View goes from -4000 to 22000, V-View from -.5 to 1.57 (approximately $\frac{\pi}{2}$), Indep Low is 0 and High is 20000. Press the

menu keys [ERASE] [DRAW]. When the plot is done press the menu key [EDIT], then the key [NXT] and then the menu key [LABEL] to plot the axes labels. Now the plot looks quite nice. Press [NXT] and then the menu key [PICT] to go back to the graphing environment. You can [TRACE] both curves and also show the coordinates under the cross hairs with [(X,Y)]. Using the two keys [arrow left] and [arrow right] you move along one curve. To jump to the other curve you can use the keys [arrow up] and [arrow down].

But perhaps you already have noticed that this method can also have some problems. If you remember the last program, then you know that it left a 1 stored in R when it run. The expression for the amplitude

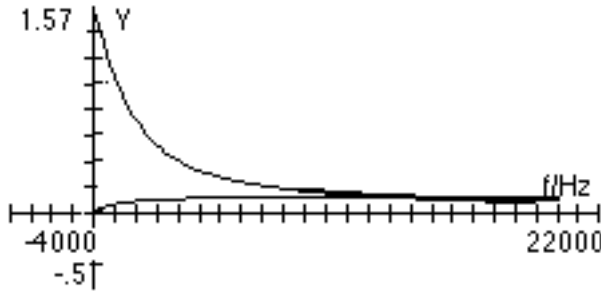


Complex numbers with the HP49G - Part 4

contains also the variable R: $\frac{f}{\sqrt{f^2 + 3000000} R}$. Now you see that for

other values stored in R the simultaneous plot will bring problems. For example store 10 in R and ERASE DRAX LABEL and DRAW. Now the plot looks not so nice. The amplitude of the current has become too low to be displayed

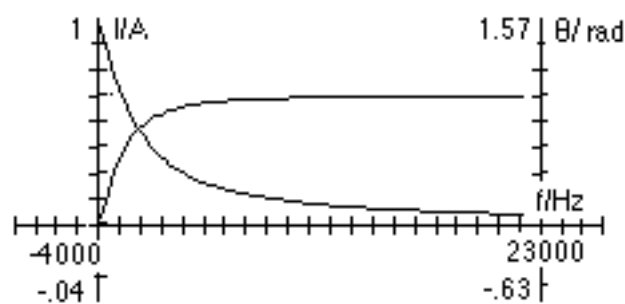
nicely in the same plot as the phase. But here come the commands AXES to save us from the misery. We can plot the amplitude with axes and labels with a V-View that lets the curve be drawn nicely, and then move the vertical axes to the right, set the appropriate V-View for the phase and draw again *without erasing the first plot*. This will create a plot with two vertical axes, one for the amplitude and one for the phase. We only must be careful to set the two different V-Views so that the ratio of the maximum to the minimum vertical view value is the same for both V-Views, or else the x-axes will not be drawn at the same place for both plots. Sounds much more complicated than it really is. Let's do that thoroughly and step for step from the command line. (Can you find which of the following things can be accomplished in some setup screen and which of them can't?)



- 1) Go to the variables menu and press N.AMPLI. Press STEQ to store the expression for the amplitude in EQ.
- 2) Use the command FUNCTION to set plot type FUNCTION.
- 3) Enter the list {f 0. 20000.} and press INDEP to set f as the independent variable and plot range from 0 to 20000.
- 4) Enter -4000. and 23000. and use the command XRNG to set the H-View.

- 5) Enter -0.04 and 0.1. and use the command YRNG to set the V-View. The ratio of the maximum to the minimum values of the vertical view is 10 to -4. We keep that in mind for the next plot.
- 6) Enter the list {(0.,0.) {1000. .01} "f /Hz" "I/ A"} and use AXES to setup the axes.
- 7) Use the command sequence ERASE DRAX LABEL DRAW to draw the plot with axes and labels.
- 8) Go to the variables menu and press N. . Press STEQ to store the expression for the phase in EQ.
- 9) Enter , then 2 and press [/] and [->NUM]. Press [ENTER] to make a copy of the number. Multiply by 4 and divide by 10, then negate. Press 2 and then use RND to round the number at 2 decimal places. Press [arrow right] to swap the two numbers and then enter 2 and RND again. The two numbers -0.63 and 1.57 will be used as the V-View and have the same ratio as the values for the previous plot. That's why we multiplied by 4 and divided by 10. We also rounded the up in order to have nice short labels of the axis minimum and maximum values when they are labelled later. Now press YRNG.
- 10) Enter {(20000.,0.) {1000. .175} "f/Hz" " /rad"}, and use the command AXES This will set the point where the axes cross in the new plot to (20000.,0.), the ticks to {1000. 0.175}, and "f /Hz" and " /rad" as labels for the axes.
- 11) Now DRAX LABEL and DRAW.
- 12) Press the key [arrow left] to look at the new plot. The left vertical axis is for the amplitude. It goes from -0.04 to 1 and has a tick every 0.01 amperes. The right vertical axis is for the phase. It goes from -0.63 to 1.57 and has a tick every 0.175 units, that is approximately every 10 degrees.

Complex numbers with the HP49G - Part 4



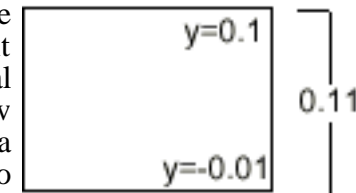
Now you know enough to make even boxed plots. But as always we also have a problem. We can only TRACE the last function that we plotted, in this case the phase.

Another way to put several curves on the same picture is to plot them stacked, that is one above the other. We can adjust the V-View so that the curve itself doesn't extend from the bottom to the top of the screen, but only over a part of it vertically. We then plot one expression and cut the part of the PICT that contains the curve. Then we do the same with the second curve and we "join" the two resulting plots. We are going to do that also step for step, and by the way learn some new commands for GROBs.

- 1) First of all, press **N.AMPLI** and then **STEQ** to store the expression for the amplitude in EQ.
- 2) We set the axes now. Enter $\{(0.,0.) \text{ "f /Hz" "I/ A"}\}$ and press **AXES**. As you see we don't set ticks for the axes yet. This is because we must first calculate what the V-View must be, in order to have the curve for the amplitude occupying, say only a third of the screen vertically. So we don't know yet what a reasonable distance between the ticks for the y-axis could be.
- 3) Enter -4000. and then 23000. Press **XRNG**.
- 4) Type **AUTO** and press **[ENTER]**. The command **AUTO** autoscales the plot, that is it set the V-View so that the whole curve is visible and that the curve extends vertically from the bottom to the top of the screen. Doing this allows to know what V-View is necessary for having no points of the curve lying off screen vertically. If we know that we can also calculate what V-

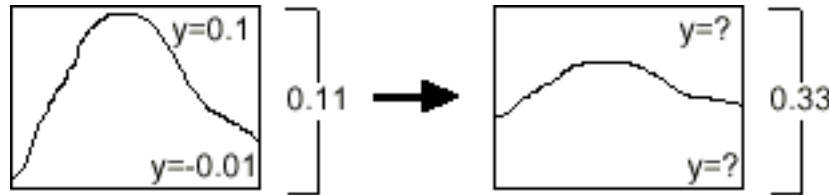
View is necessary to "squeeze" the curve vertically to a certain part of the screen.

- 4) **PURGE** variable **f**. You may have noticed that after **AUTO** finishes you have a new variable in the current directory. The command **AUTO** uses the independent variable. It stores there 40 equidistant values taken from the plot range and it evaluates the expression in EQ with these values, in order to find the appropriate V-View. Since we could forget that our independent variable contains some number, we **PURGE** it to avoid unexpected results later. Actually I think that this should be done by **AUTO** automatically (or else what **AUTO** is that?) because we so far didn't explicitly store any value in .
- 5) Recall **PPAR**. Enter 1 and 2 and then use the command **SUB** to get a list containing the first and the second element of **PPAR**. We need these parameters to calculate the new V-View.
- 6) Press **OBJ->** to explode the list and **DROP** the element count. Press **[DUP2]** to make a copy of the two complex numbers.
- 7) Press **IM** to get the imaginary part of the complex number at stack level 1. This number represents the coordinates of the upper right corner of the plot, so the imaginary part is the high value of V-View. Round it to 2 decimal places. Press **[SWAP]** and do the same to the second complex number. Now you can see that the vertical view goes from -0.01 to 0.1.
- 8) Press **[DUP2]** again to copy the two numbers. Press **[-]**. The result 0.11 is the length of the vertical view range. Multiply it by 3. Now we know that we have to choose a V-View length of 0.33 in order to squeeze the curve vertically so that it occupies one third of the screen. But we don't know what the low and the high values of V-View must be. We only know that their difference must be 0.33. It is clear that we must

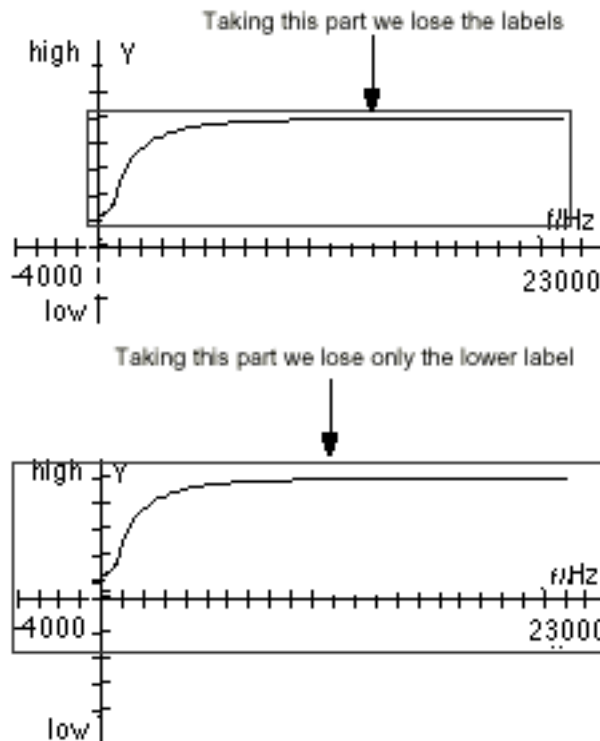


Complex numbers with the HP49G - Part 4

decide what one of both values should be. If we decide this, we can calculate the second. For this decision we consider that we



are later going to cut the part of the screen to which the curve extends vertically when plotted. When we do so we eventually lose both labels of the vertical axis. So it seems better to let one of both values be the same and modify the other. For example, if we leave the higher value of the vertical view at 0.1 and calculate a new lower value, then the plotted curve will extend from the top of the screen to some place at about one third of the screen height starting from the top. Then we can cut the part that contains the curve so that

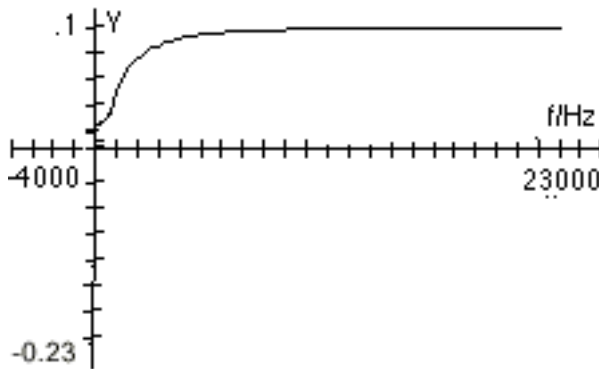


we only lose the label for the lower value of the V-View.

- 9) Let's follow this idea. Press [DUP] to make a copy of the new length. We know that the overall length of the vertical view range will be 0.33. Now we can decide what the distance of the ticks should be. Let's say, we want ten ticks on the vertical axis. So we divide by 10. The distance of the ticks for the vertical axis will be 0.033. Let the distance of the ticks on the horizontal axis be 2000. Enter {2000.}, press [SWAP] and then [+]. The command + besides adding other things also adds new elements to lists or lists to lists. Now use ATICK to specify the distance of tick labels.
- 10) Stack level three contains the higher value of the V-View. If we subtract it from 0.33 we get the *absolute value* of the new lower value. (Think why it is the absolute value.) Press PICK3 to get a copy of the object on stack level 3 to stack level 1. Press [-]. The absolute value of new lower value is 0.23. Now, we know that the old lower value was -0.01, so the new must also be negative. Working this way we can see that the value must be negative but what if we want to make a program? Well, then we can add an IF THEN ELSE END clause, and check if the old lower value is lower than 0. or not. But for now press [NEG]. The new lower value of V-View must be -0.23. Press [NIP] to drop the object at stack level 2 as we don't need it anymore.
- 11) Press SWAP to put the lower and the higher value in the right order on the stack and then use YRNG to adjust the new V-View.
- 12) Use the command sequence ERASE DRAX LABEL DRAW to plot the curve and draw axes and labels. Now the plot looks like we wanted to have it. The curve is on the top third of the screen and so taking just this part will cost only one label.
- 13) Now that the plot is done we can use the coordinates of the viewing range that are left on the stack to cut the part of the

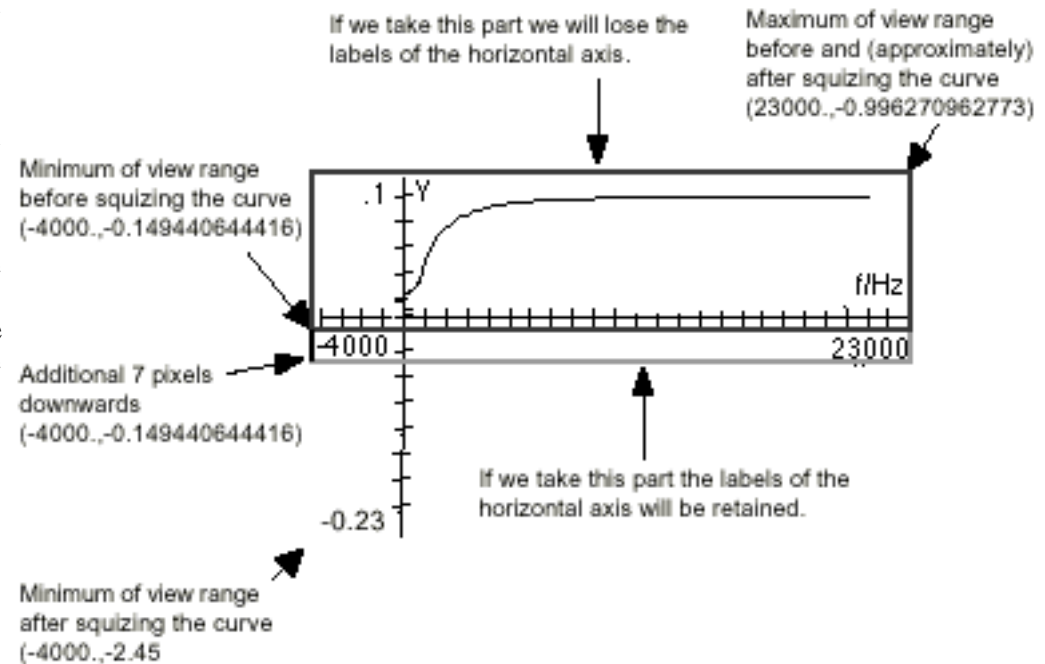
Complex numbers with the HP49G - Part 4

PICT that contains the curve, the labels of the horizontal axis and the top label of the vertical axis. Remember, the coordinates that are now on the stack, were created by the command AUTO before we



squeezed the curve. But we must do a little bit of additional work. Why? Because the curve is contained between these coordinates but the horizontal labels are eventually *outside* these coordinates. How can we change the y-coordinate of the minimum of the view range making sure that the labels are then contained in the part that we cut? Well, to answer this we must examine the way that the pixels correspond to coordinates a bit more thoroughly. We already know that the range of values of coordinates in PICT is defined by the two first parameters in PPAR. These coordinates are *user coordinates* and they are described with complex numbers. Pixels of functions are plotted according to these parameters. But the HP49G has another way to address pixels on GROBs. This is by counting pixels in x-direction starting at the left and at the y-direction starting at the top of the GROB. Note the difference for the y-direction. *While the user y-coordinates (in general) grow upwards, the pixel y coordinates always grow downwards.* Fortunately for us, HP has included two commands that transform user to pixel coordinates and vice versa. These two commands are C->PX and PX->C. You find them if you press [blue-shift], [CAT], [NXT], menu key [PICT] and [NXT]. These commands save our lives in this case. The labels added to the

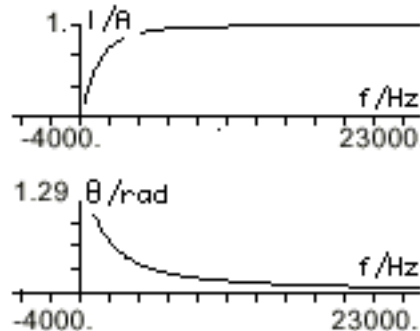
horizontal axes always extend vertically 7 pixels under the horizontal axis. That means that if we diminish the user coordinate of the minimum of the view range by that amount, then the labels will certainly be contained in the part of the PICT that we cut. So let's do it. Press [SWAP] to get the minimum coordinates of the view range on stack level 1. Press C->PX. The complex number is transformed to a list of two binary integers. The user coordinate that was transformed is the same as {#0d #21d}, that is 0 pixels from the left edge of the PICT and 21 pixels from the top of the PICT. If the binary integers that you HP49G displays don't have the character "d" at the end, then you are probably not in decimal mode. Type DEC and press [ENTER] to switch to decimal mode. Now, we want additional 7 pixels for the y-coordinate. Enter the list {#0d #7d} and use ADD to add 0 pixels to the first and 7 pixels to the second pixel coordinate. The command ADD



Complex numbers with the HP49G - Part 4

performs addition of elements of lists. It does + to corresponding elements. The elements can be any objects that the command + can add, that is reals, numbers and so on. ADD also accepts a list and a single object in any order and adds the object to each element of the list. It also accepts two single objects and then it acts like +. This command is in menu MTH/LIST. Back to our coordinates. We want the pixel coordinates to be transformed back to user coordinates, so we press PX->C. The result is the complex number (-4000.,-.15) which shows that the pixel y-coordinate increased from #21d to #28d but the user y-coordinate decreased from approximately -0.15 to -0.53. SWAP the two numbers.

- 14) Enter PICT and UNROT. Press SUB. (SUB is in menu PRG/LIST and also in menu PRG/GROB) Now the part of the PICT that contains the curve and all labels except the lowest is on the stack.
- 15) We are going to put this GROB and the next one together, so we should somehow insert something between the two, so that they are separated on the common plot. Enter the two binary integers #131d and #3d and press [BLANK]. This command creates a blank GROB with the horizontal and vertical sizes given as binary integers. BLANK is in menu PRG/GROB.
- 16) Now press GROBADD to join the two GROBs, the plot and the blank in one GROB vertically.
- 17) Repeat the same procedure with the expression N. and the appropriate data for AXES etc. and then GROBADD the new plot with the one that is already on the stack.

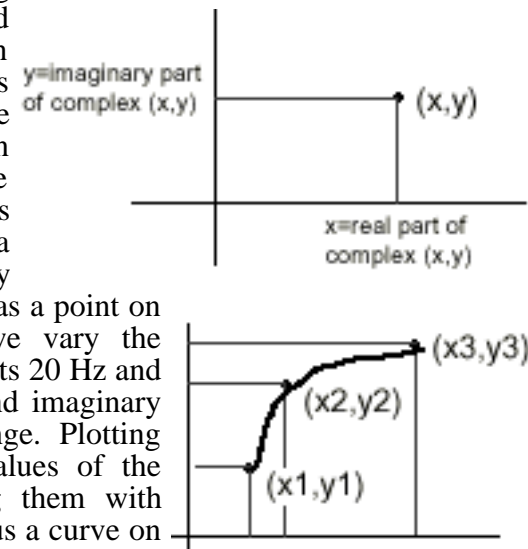


The result is a GROB that contains both curves on one GROB so that they can be seen simultaneously on the screen. Now that you know how it works, perhaps you could make a program that takes a list of expressions and other parameters and creates similar GROBs that contain more than one plots.

Up to this point we plotted real quantities that can be calculated out of the complex current. But the HP49G allows also plotting complex quantities in several ways. Let's take a closer look at these plots and how we can use them to visualise the behaviour of our complex quantities.

The complex current is a function of the frequency. It is given by the formula
$$\frac{V_0 e^{i 2 \pi f t}}{R + \frac{1}{i 2 \pi f C}}$$
. The factor $\frac{V_0}{R + \frac{1}{i 2 \pi f C}}$ is the quantity

with which the oscillating part is multiplied and which doesn't depend on the time. We have used this to find the real amplitude and phase and plot them in many ways. For some certain frequency this complex quantity has a certain real and imaginary part, which can be plotted as a point on the complex plane. If we vary the frequency between the limits 20 Hz and 20000 Hz then the real and imaginary parts will of course change. Plotting many points for many values of the frequency and connecting them with small line segments gives us a curve on the complex plane. If we do such a plot for the complex current we have two dependent values on the plot, the real and imaginary part of the current. The independent variable (the



Complex numbers with the HP49G - Part 4

frequency) doesn't explicitly appear as a coordinate. This is called a parametric plot. For our happiness the HP49G can do such plots out of the box.

Enter V0 (in which we have stored previously a 1), go to the variables menu and press the menu key for Z (in which we have stored

$R + \frac{1}{i 2 f C}$ previously). Then press [÷]. EXPAND the expression

and you get $\frac{f}{f - 1000 i \sqrt{3}}$. We will do a parametric plot of this

expression. Press STEQ to store this in EQ. Now we must setup the plot. Got to the screen PLOT SETUP and choose PARAMETRIC as the plot type. Independent variable must be . Enter 0.1 as H-tick and V-Tick and uncheck the option _Pixels. Now go to the screen PLOT

WINDOW - PARAMETRIC. Enter a horizontal view from 0. to 1. and a vertical view from 0. to 0.5. The independent variable should go from Low: 0. to High: 20000. If something different than "Default" is in the input field of Step: then go to that field, press [NXT] and press the



menu key [RESET]. Select "Reset value" from the popup and press [ENTER]. Uncheck the option _Pixels. Now press [NXT] and then the menu keys [ERASE] and [DRAW]. The HP49G draws the plot. When ready you can TRACE it and pressing the menu key [(X,Y)] you can see the coordinates under the cross hairs. The HP49G shows the coordinates f:0 (0.,0.). This means, when the frequency is Hz then the real part is 0. and the imaginary part is 0.. Press the key [arrow right] to watch the changes of the complex current as the frequency varies. If you press [ENTER] in TRACE mode then the HP49G puts a list {f (x,y)} on the stack, giving you the value of the frequency and

the real and imaginary parts of the point on the curve where the cross hairs is. If you repeatedly press [arrow right] you will notice that the cross hairs movement gets slower. This show that both the real and imaginary parts change less and less when the frequency goes to higher values. The behaviour matches the behaviour of the real amplitude of the current, which we have already plotted against the frequency. The

amplitude and so also the real and imaginary parts of the current go against a limit value when the frequency goes higher.

This kind of plot uses the parameters in PPAR differently:

(x_{min}, y_{min})

A complex number which specifies the lower left corner of the display range. Default value is $(-6.5, -3.1)$.

(x_{max}, y_{ax})

A complex number which specifies the upper right corner of the display range. Default value is $(6.5, 3.2)$.

indep

A list which contains the name of the independent variable and the minimum and maximum of the plotting range, that is the lower and upper bounds between which the independent variable varies. If only the name of the independent variable is included in this list, then the values of (x_{min}, y_{min}) and (x_{max}, y_{ax}) are used as plotting range, which in general will give you undesired results. This is because the range in which the independent variable varies is not on any part of the plot explicitly.

res

A real number that specifies the step between values of the independent variable in user coordinates. Default value is 0, which specifies a step of $\frac{1}{130}$ of the difference between the maximum and the minimum values of the plotting range of the independent variable.

axes

A list that has one or more of the following elements in order. A complex number specifying the coordinates of intersection of the axes, a list that specifies the tick marks of the axes and two strings that are used as labels for the X- and the Y-axes.

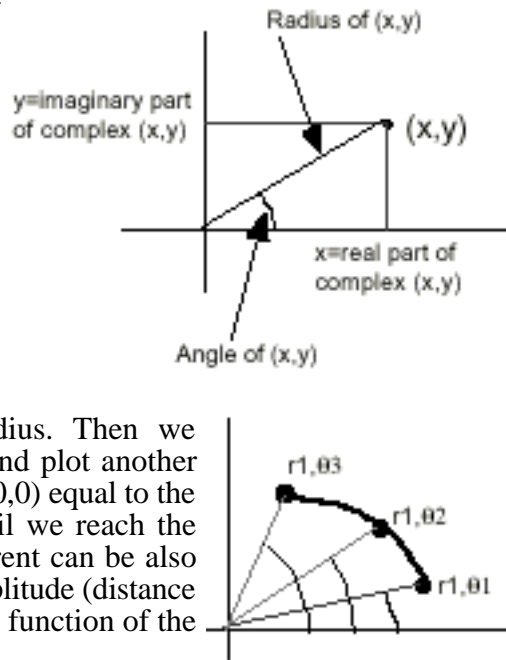
ptype

The command name PARAMETRIC.

Complex numbers with the HP49G - Part 4

depnd Name specifying the label of vertical axis. Used when the doesn't contain a label for the vertical axis.

This plot has shown us how the real and imaginary parts of the complex current change when the frequency changes. You of course know that a point on the complex plain can be specified by giving its real and imaginary parts, or by giving its radius and phase (angle). A new kind of plot can be done using these two quantities. We start at the first value of the angle and plot a point which has a distance from (0,0) equal to the radius. Then we change the angle a little bit and plot another point having a distance from (0,0) equal to the new radius. We continue until we reach the final angle. The complex current can be also plotted this way, that is its amplitude (distance from (0,0)) can be plotted as a function of the angle.



We already have found the expressions for the amplitude and the phase and stored them in variables N.AMPLI and N. . Go to the variables menu and press the menu keys for N.AMPLI and N. . The expressions for the amplitude (radius) $\frac{f}{\sqrt{f^2 + 3000000} R}$ and phase

(angle) $\text{ATAN} \frac{1000 \sqrt{3}}{f}$ are now on the stack. As you can see, in

this case we don't have the radius as a function of the angle. We must first somehow transform the radius, so that it becomes a function of the angle. So now the CAS of the HP49G comes to help us. Enter . You

get a sort of by pressing [ALPHA] twice, typing the letter "O" and then pressing [red shift] and [9]. The resulting character can be thought of being the greek letter , despite the protesting commends of Trabakoulas. ("What? No real ? How dare they?") Then press

[SWAP] and [=] to get the equation $\theta = \text{ATAN} \frac{1000 \sqrt{3}}{f}$. (We

don't use the greek letter for denoting the angle, because we have already stored an expression there.) Now perhaps you see what we are going to do. If we solve this equation for f, we can SUBSTitute the result in the expression for the amplitude. Press SOLVE to get the

solution $f = \frac{1000 \sqrt{3}}{\text{TAN}(\theta)}$ which is the frequency as a function of the

angle. Press SUBST, switch to real mode and EXPAND. The expression for the amplitude is now

$\frac{\sqrt{3} \text{TAN}(\theta) |\text{TAN}(\theta)|}{\text{SQ}(\text{TAN}(\theta)) \sqrt{\text{TAN}(\theta)^2 + 1} \sqrt{3}}$. Now, we know from our

previous plots that goes from $\frac{\pi}{2}$ to 0. That means that $\text{TAN}(\theta)$ is positive and so $|\text{TAN}(\theta)| = \text{TAN}(\theta)$. We could add the assumptions

0 and $\frac{\pi}{2}$ and try to EXPAND again, expecting that the HP49G would recognise that under these assumptions $|\text{TAN}(\theta)| = \text{TAN}(\theta)$. But it won't. "Limitations show you that you are in real life, son!", as Trabakoulas says. So edit the expression

manually and make it $\frac{\sqrt{3} \text{TAN}(\theta) \text{TAN}(\theta)}{\text{SQ}(\text{TAN}(\theta)) \sqrt{\text{TAN}(\theta)^2 + 1} \sqrt{3}}$. EXPAND

it again and you have: $\frac{\sqrt{\text{TAN}(\theta)^2 + 1}}{\text{TAN}(\theta)^2 + 1}$. Do we remember that? If not,

Complex numbers with the HP49G - Part 4

we press TRIGCOS and we get $\frac{\cos(\)^2 |\cos(\)|}{\text{SQ}(\cos(\))}$. EXPAND this res

and you get $|\cos(\)|$, which you can edit to $\cos(\)$ for the same reason as above. This is the dependency of the amplitude from the angle. A simple cosine! This makes us sing with joy because even in a complex marathon we still like simple things, don't we? ;-)

Now that we have the amplitude as a function of the angle, let's plot. We are going to use the type of plot "radius against angle" or in other words POLAR. Press STEQ to store $\cos(\)$ in EQ. In the screen PLOT SETUP choose a plot type POLAR and independent variable (that is the character θ , Trabakoulas looks angry again ;-)). In the screen PLOT WINDOW - POLAR enter H-View from 0. to 1. and V-View from 0. to 0.5. Indep should go from 0 to 1.57 (that's approximately $\frac{\pi}{2}$). Press ERASE and DRAW. The plot looks exactly

like the previous one, it is only plotted in the opposite direction. (Can you tell why?) If you TRACE it displaying coordinates, you can see how the amplitude varies as the angle goes from 0. to $\frac{\pi}{2}$. Pressing [arrow right] in TRACE mode moves the cross hairs to the left. This is because pressing this key, just puts a higher value in θ , which makes the amplitude smaller. The parameters in PPAR are used as follows:

(x_{\min}, y_{\min}) A complex number which specifies the lower left corner of the display range. Default value is $(-6.5, -3.1)$.

(x_{\max}, y_{\max}) A complex number which specifies the upper right corner of the display range. Default value is $(6.5, 3.2)$.

indep A name specifying the independent variable or a list which contains the name of the independent variable and the minimum and maximum of the plotting range.

axes

A real number that specifies the step between values of the independent variable in user coordinates. Default value is 0, which specifies a step of 2 degrees or 2 grads or $\frac{\pi}{90}$ radians.

pptype

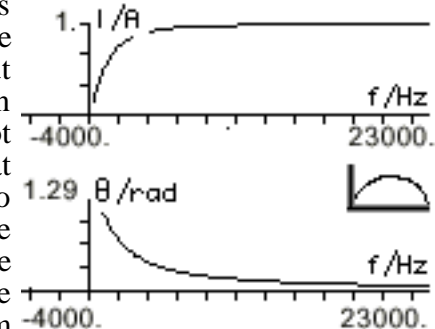
A list that has one or more of the following elements in order. A complex number specifying the coordinates of intersection of the axes, a list that specifies the tick marks of the axes and two strings that are used as labels for the X- and the Y-axes

depnd

POLAR

Name specifying the label of vertical axis. Used when the θ doesn't contain a label for the vertical axis. Note however that the dependent variable is not the same as the labels of the vertical axis. The dependent variable is the radius, the distance of the plotted point from the point (0,0).

If you press the menu keys [ZOOM] and the [ZOUT] then the plot is redrawn with a zoomed out view range, that is it is drawn smaller. Can you cut the small plot out and put it in the plot that contained two stacked curves, so that you at once can see the dependency of amplitude and phase from the frequency and the dependency of the amplitude from the phase?



After so many plots and parameter discussions we perhaps should relax for a while, especially because many many other plots will be

Complex numbers with the HP49G - Part 4

discussed later in this part. So let's do something funny taken out of the secret book of treasures of the HP49G.

If you remember the start of this part of the complex marathon (which is ages in the past), we stored the complex resistance

$$R + \frac{1}{i 2 f C} \text{ in variable Z, the complex current } \frac{V0 e^{i 2 f t}}{R + \frac{1}{i 2 f C}} \text{ in}$$

variable CMPLI, 1 in V0, the expression for the amplitude $\frac{2 f C \sqrt{4 f^2 R^2 C^2 + 1}}{4 f^2 R^2 C^2 + 1}$ in variable AMPLI, the expression

$$\text{ATAN} \frac{1}{2 f R C} \text{ in , the expression } \frac{\sqrt{3}}{6000 R} \text{ in variable C}$$

and some different values in R. Now suppose that we take the expression for and we want to see how it depends on R. If we put it on stack and EXPAND, then any possible evaluation and expansion of all variables is performed, and since even R has a value we get as result

$$\text{the expression } \text{ATAN} \frac{1000 \sqrt{3}}{f} \text{ which is not what we wanted. But}$$

the good old HP49G has a forgotten command which it inherited from his ancestor, the HP48. It is the command SHOW. It fits very good for our purposes because it does exactly what we want. (And because we already made movies, so the SHOW was good, I hope ;-)). The command works in two ways, according to the arguments we give to it. Recall the variable on the stack, enter 'R' with quotes type SHOW

$$\text{and press [ENTER]. The result is } \text{ATAN} \frac{1}{2 f R \frac{\sqrt{3}}{6000 R}}.$$

The command has searched and evaluated all variables except R itself, and returned a result where variables were replaced with their contents until R was visible. Of course in this case R cancels out but there are cases where this forgotten command can be a real treasure. Imagine for

example that you have the expression $\frac{a-b}{A+B}$ and you have stored

$$\frac{a^2+b-1}{3} \text{ in A, } \frac{\text{COS}(a) b}{\text{SIN}(b) A} \text{ in B and } 3 a^2 - a \text{ in b. If you want to}$$

resolve all variables that appear in $\frac{a-b}{A+B}$ until variable b shows, then

you enter $\frac{a-b}{A+B}$, then 'b' (in quotes) and use SHOW to get the result

$$\frac{a-b}{\frac{a^2+b-1}{3} + \frac{\text{COS}(a) b}{\text{SIN}(b) \frac{a^2+b-1}{3}}}. \text{ As you can see the following}$$

happened. On the numerator no change was carried out, because variable a doesn't contain anything and because b was the variable that we wanted to be shown. Variable A in the denominator was replaced by its contents $\frac{a^2+b-1}{3}$. Since again a has no contents and b is what we want to see, no further actions were performed. Variable

B in the denominator was replaced by its contents $\frac{\text{COS}(a) b}{\text{SIN}(b) A}$. Here

we have the variable A again, which when resolved gives a result that contains b. So it was resolved and the expression $\frac{\text{COS}(a) b}{\text{SIN}(b) \frac{a^2+b-1}{3}}$

was the result that was put in place of variable B in the original expression. If you enter $\frac{a-b}{A+B}$ and 'a', and then use SHOW, the

$$\text{result is } \frac{a - (3 a^2 - a)}{\frac{a^2 + 3 a^2 - a - 1}{3} + \frac{\text{COS}(a) (3 a^2 - a)}{\text{SIN}(3 a^2 - a) \frac{a^2 + 3 a^2 - a - 1}{3}}}, \text{ because all}$$

Complex numbers with the HP49G - Part 4

occurrences of b were replaced with the contents of b , in which variable b is present. Lastly, if you enter $\frac{a-b}{A+B}$, then 'A', and then

use SHOW, you get the result $\frac{a-b}{A + \frac{\cos(a)}{\sin(b)} b}$, where only B was

replaced with its contents which also include A . This is the first way of how SHOW works. It makes all implicit references to the variable in stack level 1 explicit. But if you enter an expression and then a list with variables, then the implicit references of all variables *except those in the list*, will be made explicit. For example entering $\frac{a-b}{A+B}$ and the list

{ b B }, and using SHOW, returns $\frac{a-b}{\frac{a^2+b-1}{3} + B}$. This results in the

following way: Variable a doesn't contain anything, it doesn't refer to some other variable, so any occurrence of a will be left unchanged. Variable b was in the list, so its implicit references will not be resolved. Variable A contains references to a and b , so these references are made explicit. And variable B was in the list, so its implicit references will not be resolved. Notice the big difference between the two ways. Giving an expression and a variable name to SHOW, makes all occurrences *to* this variable explicit. Giving an expression and a list of names to SHOW, makes all occurrences *of* variables that are not in the list explicit.

What does this have to do with complex math? Well, working with such things as alternating voltage and current, especially when there are many elements like resistors, capacitors and coils in your circuit, quickly leads to very complicated expressions. Often some sub expressions are stored to some variables and in the formulas we work with these variables, instead working with their complicated contents. It is a real advantage to be able to resolve exactly what we want, so that we can emphasise on the dependency of some certain variable, etc. This is the controlled evaluation of variables, which again shows that the

HP49G might be often a confusing machine, but the flexibility of its command set is still unreached by any other mathematics tools of this size.

Oh, you might wonder what the funny thing is, considering SHOW.

Well, consider the expression $1 + \frac{1}{X}$. Now replace X with the

expression itself and you get $1 + \frac{1}{1 + \frac{1}{X}}$. Go on repeating this

substitution. The expression grows and grows, but it keeps being, well somehow similar to itself. Without giving many details of the really extremely interesting behaviour of this kind of expression, let it

be said here that each time we replace X with $1 + \frac{1}{X}$, we get nearer and

nearer to the number 1.61803398875...., *no matter what X is!!!* Note that this is not a limit in the sense that we approach some value by letting X approach some other value. It is a limit which we

approach by repeating the procedure of replacing X with $1 + \frac{1}{X}$. X

itself can be 1, -4.5 or even (1,1) (!). The question is, how can we program this on the HP49G? How can we say, "Repeat this substitution procedure a certain number of times?". We can do that in many ways. One of these would be to define the recursion:

$$\text{REC1}(n) = \text{IFTE } n == 1, 1 + \frac{1}{X}, 1 + \frac{1}{\text{REC1}(n-1)}$$

Enter this and press [DEF]. Now enter 1 and press the menu key

[REC1]. The result is $\frac{X+1}{X}$ which is equal to the expanded form of

$1 + \frac{1}{X}$. Enter 2 and press again [REC1] to obtain the second member

Complex numbers with the HP49G - Part 4

of this repeated substitution, $\frac{2X+1}{X+1}$, which is equal to the expanded form of $1 + \frac{1}{1 + \frac{1}{X}}$. But perhaps you don't want the expanded form.

And perhaps you would like to build successively all members starting from the first up to the nth. Then you can use SHOW in a funny, and for some people "illegal" way. Here's how to do it. Enter 1+A, then B, and press [STO]. Enter $\frac{1}{B}$, then 'A' (in quotes!!) and press STO again.

You might already observed that we have a circular reference here. (While for computing devices this can be a horror, for mathematics it is very very interesting!). Now do exactly what is described here, and be careful to put the variables in quotes when needed. Enter 'B'. *Enter it in quotes and don't EVAL it or else the HP49G will error out with "Circular Reference"*. Enter 'A' (again in quotes) and press SHOW. The result is 1+A. Now enter 'B' (yes, in quotes) and press SHOW again. The result is $1 + \frac{1}{B}$. Then again 'A' and SHOW to get $1 + \frac{1}{1+A}$.

You get it now. Alternating 'A' SHOW and 'B' SHOW, we get successively the members of this remarkable family of expressions. If you want all members from the first to the nth, then the small program will do that for you automatically:

```
<<
  'B'
1 ROT
START
  'A' SHOW
  'B' SHOW
  DUP
NEXT
DROP
>>
```

Give it a number n and it will return all members from the first to the

nth, where instead of X we have B in the expressions. You see clearly here how even circular references can be used for interesting mathematic experiments, if we use the appropriate commands. The command SHOW is the more controlled analogon to EVAL, and so it allows us to circularly refer to variables, controlling how much evaluation will be done. The funny thing is to that circular references are treated like one of the worst things that can happen, and at the same time hiding such a useful command somewhere in the endless catalogue of commands in the HP49G. And the question of Trabakoulas is: "how can someone not love this machine?"

End of philosophic thoughts (for now ;-)). Let's get back to our frequencies. Nick wanted to make a filter, in order to give the sound of his guitar more high frequencies. Now he wonders, what does the signals coming out of the outlets to the loudspeakers (after the filter) look like compared to the voltage supplied at the outlets of the amplifier (before the filter)? Well, if we plot both simultaneously we can compare them. The voltage supplied by the amplifier is given by $V_0 \cos(2\pi f t)$, where we already stored 1 in V0, and where f is the frequency and t is the time. Enter $V_0 \cos(2\pi f t)$ and press [EXPAND] to get $\cos(2\pi f t)$. We already stored the expression for the current after the filter in variable I. Recall I on the stack and press [EXPAND] to get the expression

$$\frac{f \sqrt{f^2 + 3000000} \cos \operatorname{ATAN} \frac{1000 \sqrt{3}}{f} + 2\pi f t}{f^2 + 3000000}$$

Enter 2 and press [->LIST]. Nick takes 1000 Hz as test frequency, so enter f = 1000 and then press [SUBST] and [EXPAND]. The list

$$\text{now is } \cos(2000\pi t) \frac{\cos \frac{(6000\pi t + 1)}{3}}{2} . \text{ Press STEQ.}$$

Go to the PLOT SETUP screen and choose type of plot FUNCTION.

Complex numbers with the HP49G - Part 4

Independent variable is t . Enter 0.0001 as horizontal and 0.1 as vertical ticks. Go to the PLOT WINDOW - FUNCTION screen. The frequency of the signal is 1000 Hz, that is it oscillates one time in 0.001 seconds. Let's setup a plot for taking a look at two oscillations, that is horizontal view from 0 to 0.02. vertical view from -1 to 1. Reset the independent variable plotting range to "Default". Now, press [ERASE] and [DRAW]. The plot shows what happens with the voltage of 1000 Hz (red curve). The

corresponding current after the filter (blue curve) has half the amplitude (that is what we wanted) but also is out of phase (what we didn't want). And Nick starts asking himself if that was what he

wanted. Music is not a single cosine of a single frequency. If each frequency has a different phase shift and a different amplitude, then what will happen to a signal consisting of several frequencies? What will it look like? Let's help the musician, who thought he could be also an electronics expert. (He didn't know what he was doing, right Rcobo? ;-)) We build up signal of 3 frequencies and three different amplitudes, as an example of the voltage supplied by the amplifier before the filter. Enter again $V0 \cos(2 \pi f t)$ and the list

$\{V0 = .9 \ V0 = .8 \ V0 = .7\}$ and press [SUBST]. Enter the list $\{f = 500 \ f = 1000 \ f = 2000\}$ and press [SUBST] again. Press [SLIST] (Menu MTH/LIST). The result is $.9 \cos(2 \pi 500 t) + .8 \cos(2 \pi 1000 t) + .7 \cos(2 \pi 2000 t)$

Now we do the same for the current after the filter. Recall variable I, press EXPAND. Then enter 'V0' (in quotes) and press [*] (because we stored the expression for the current in 'I' after we worked with $V0=1$).

Now, enter again $\{V0 = .9 \ V0 = .8 \ V0 = .7\}$, press [SUBST], enter $\{f = 500 \ f = 1000 \ f = 2000\}$ and [SUBST] again. Press [SLIST] and then 2 [->LIST]. Store this in EQ. Go to the PLOT WINDOW - FUNCTION

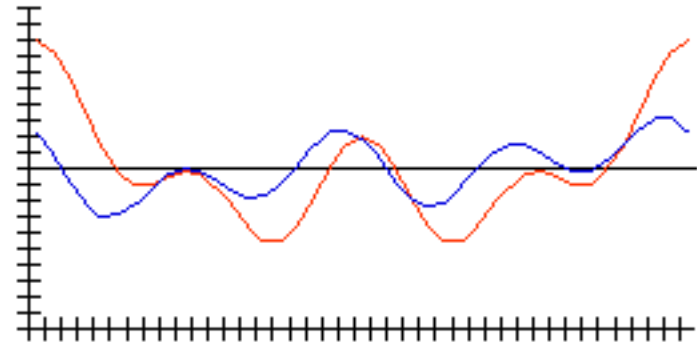
screen and enter a vertical view from -3 to 3. [ERASE] and [DRAW] that. The

signal after the filter (blue) doesn't quite look like what it should be. It

is totally out of shape. And it is a real wonder that we can still recognise Metallica, when we turn the bass high on the equaliser. Our ears and brain must somehow make an FFT and find out which frequencies belong to which instrument. Imagine what a calculating power is needed to do this with an orchestra of about 100 instruments playing simultaneously! (The other moral of the story is: "Buy a good amp, Nick, and let filters aside.;-))

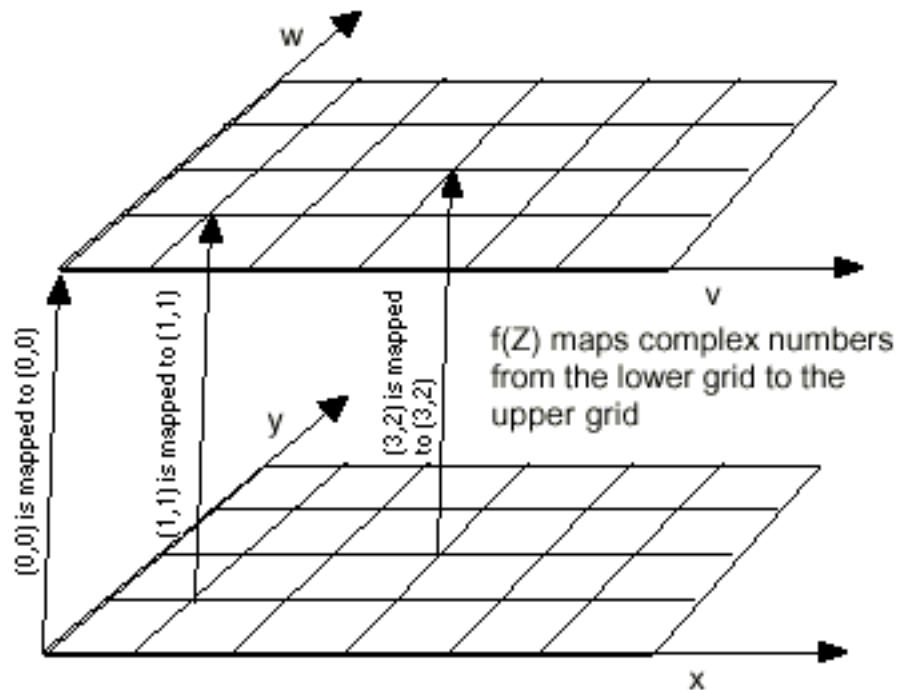
We are coming closer to the end of this part and of the complex marathon, but I couldn't resist giving yet another plot here. There is a plot type built-in to the HP49G, which can be a great help when dealing with complex quantities, and which we will use extensively at the complex analysis marathon. It is the plot type GRIDMAP which I have the feeling is not very well known by the majority of the users.

Let's describe first what this plot does. Exactly like defining and plotting functions of a variable in over the real numbers, we can define functions and also plot them over the complex numbers. That is, we define a function $U = f(Z)$, where U and Z are complex. Any complex number can be written in the form $x + i y$. Defining such a function over the complex numbers, we have two independent and two



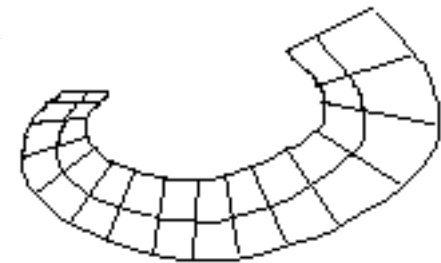
Complex numbers with the HP49G - Part 4

dependent variables, because $Z = x + i y$ and $U = v + i w$. If we want to plot such a function, showing all coordinates (x, y, v, w) we need to be able to perceive a 4-dimensional space (or to think if there is some kind of projection of the 4-dimensional space to our 3-dimensional space.) Or, we can also let x and y out of the plot, and only take v and w . Consider for example the very simple function $v + i w = f(Z) = f(x + i y) = x + i y$ which maps any complex number to itself. If we forget about the lower grid in the picture below and we only take the upper grid, then we just have the plot type GRIDMAP of the HP49G.



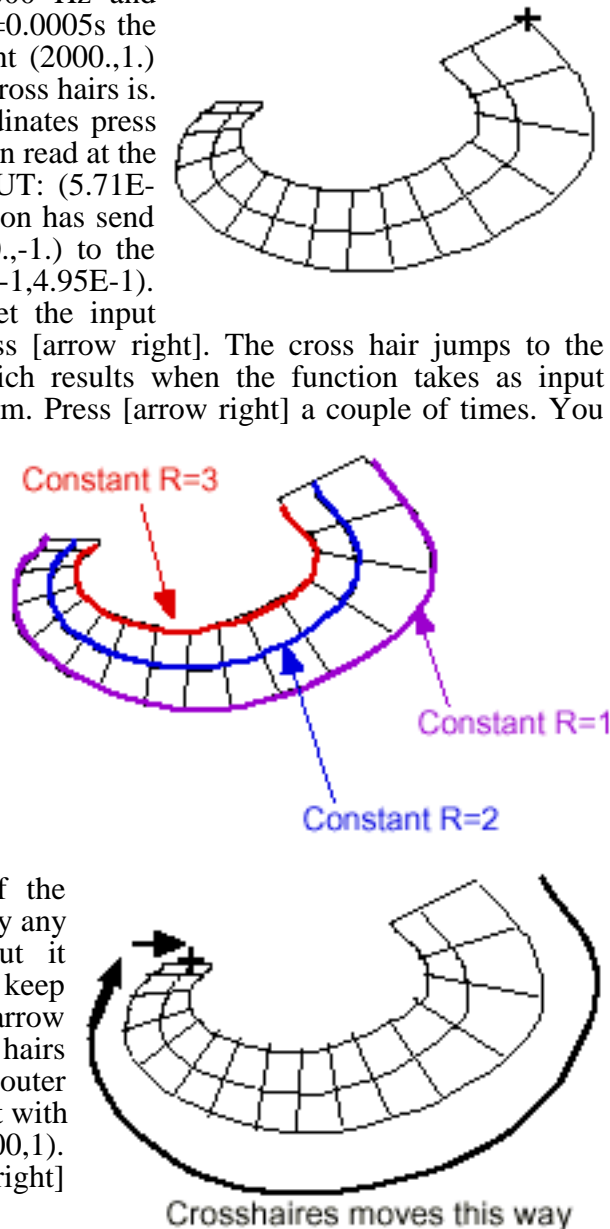
We already had the complex function $\frac{V_0 e^{i 2 \pi f t}}{R + \frac{1}{i 2 \pi f C}}$ in variable CMPLI. Purge variable R, recall CMPLI on the stack and EXPAND it. The result is $\frac{f e^{i 2 \pi f t}}{(f - 1000 i \sqrt{3}) R}$. As you can see, this is a

complex function of three variables. If we consider it for a certain time t , then it still depends on two variables, namely f and R . A complex function that depends on two real variables, that sounds like it is good for plotting in with plot type GRIDMAP. So let's do it. First store the function in EQ. We will plot with variable f going from 500 Hz to 2000 Hz, so that the frequency of 1000 Hz (Nick's key frequency) is included. This frequency finishes an oscillation in 0.001 seconds. We will plot the function for a particular value of t , so store 0.0005 in variable 't'. That means, that the picture we will look at, will be at the middle of one oscillation of signals with a frequency of 1000 Hz. Go to the PLOT SETUP screen and choose plot type GRIDMAP. Set independent variable 'f' and dependent variable 'R'. Go to the PLOT WINDOW - GRIDMAP screen. Enter for X-Left -0.8 and for X-Right 0.8. Enter for Y-Near -0.8 and for Y-Far 0.8. This is the viewing window. Now enter for XXLeft 500. and for XXRight 2000. (The frequency will go from 500 to 2000.) Enter for YYNear 1. and for YYFar 3. (The resistance will go from 1 to 3.) Enter for Step Indep 16 and for Depnd 3 to adjust number of steps the for frequency and for the resistance. Press [ERASE] and then [DRAW]. When the plot is done it looks like the picture on the right. But what is that thing? We can understand better by tracing. Press the menu key [TRACE] and then the menu key [(X,Y)]. Be patient because the HP49G needs considerable time to react. The cross hairs jumps to the upper right point of grid. The coordinates shown are "INPUT {2000. 1 }". That means that the source

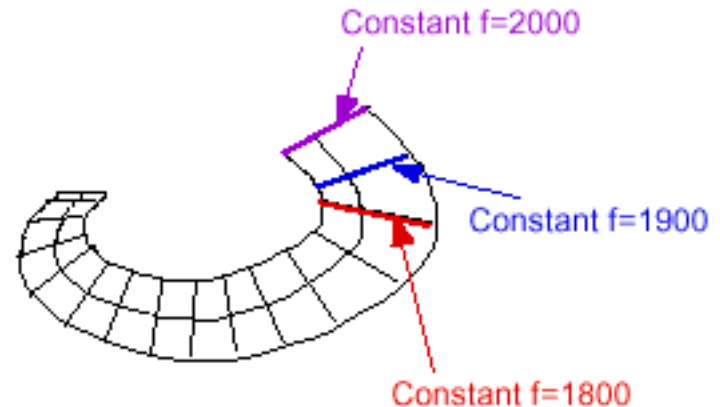


Complex numbers with the HP49G - Part 4

coordinates were $f=2000$ Hz and $R=1\Omega$. At the time $t=0.0005$ s the function sends the point $(2000.,1.)$ to the point where the cross hairs is. To see the target coordinates press [F2] again. Now you can read at the screen bottom "OUTPUT: $(5.71E-1,4.95E-1)$ ". The function has send the source point $(2000.,-1.)$ to the target point $(5.71E-1,4.95E-1)$. Press [F2] twice to get the input coordinates again. Press [arrow right]. The cross hair jumps to the point on the grid, which results when the function takes as input $f=1900$ Hz and $R=1\Omega$. Press [arrow right] a couple of times. You can see that the resistance R is always 1Ω , while the frequency varies to smaller values. This means that the bent curves that go around the centre point of the plot belong are curves of constant resistance R . Now you might think that pressing the key [arrow up] or [arrow down] would move along curves of the same constant frequency any varying resistance. But it doesn't! You have to keep on pressing the key [arrow right] until the cross hairs has moved along the outer curve and is at the point with input coordinates $(500,1)$. Then, pressing [arrow right]



moves the cross hairs to the point with input values $(500,2)$. Anyway, the curves of constant frequency are those which extend in direction from the centre point of the plot to its edges. The crossing points of the curves in this grid, are complex values of the alternating current for distinct combinations of frequency and resistance and for the particular time $t=0.0005$ s.



For example the grid point with the input values $f=1000$ Hz and $R=1\Omega$ has output values If you TRACE the plot and have patience enough to keep waiting until "INPUT {999.99999999}" appears on the bottom of the screen and the cross hairs is at the most outer curve, then you can press [F2] to let the HP49G show the output , and then [ENTER] to put the output values on the stack. (It is a little bit annoying that the HP49G doesn't show "INPUT {1000 1}", though we set up the number of steps for the frequency to 16.) Press [CANCEL] until you are back to the stack. Press [ENTER] to make a copy of the complex number. If you now press [ABS] the result will be .499999999996 which is approximately 0.5, the amplitude of the current with a frequency of 1000 Hz, as Nick wanted to have. Press [arrow right] (shortcut for [SWAP]) and then [RE]. The result is the value of the alternating current with a frequency of 1000 Hz at the time $t=0.0005$ seconds. Before we go on exploring what happens as the time passes by, we notice that no axes were drawn. This kind of plot doesn't draw any axes, but we can add axes that perfectly fit to our viewing range. To do so, we must set the parameters in PPAR identically to the parameters in VPAR which are used by GRIDMAP. We had a viewing

Complex numbers with the HP49G - Part 4

range horizontally and vertically from -0.8 to 0.8. So enter -0.8, then 0.8 and use XRNG to set H-View for PPAR. Use the same values for the command YRNG. Then enter the list $\{(0,0.) \{.1 .1\} "RE(I)" "IM(I)"\}$ and use AXES to adjust the axes parameter. Now, use the command sequence DRAX LABEL to add axes with labels to the plot. Type GRIDMAP and press [ENTER] to switch back to the plot type GRIDMAP. If you now press [arrow left] (shortcut for switching to the plotting environment) you will see the gridmap plot with axes and will still be able to TRACE the plot. When you traced enough return to the stack.

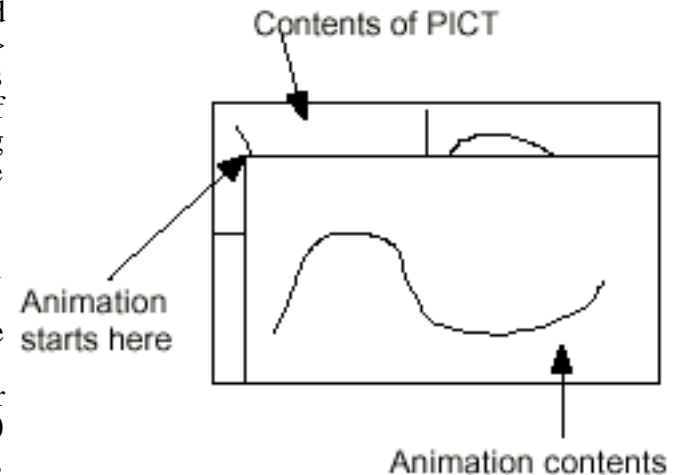
Let's explore now what happens as the time passes by, that is how the grid evolves with time. To so we must put several different values and re-plot, so that we can then animate the plots at different times. When the same is done many times, it is better to program the calculator instead of doing everything manually. So enter the following program:

```
<<
'f*EXP(2*i*t*f* )/((f-1000*i*f*+3)*R)'
STEQ                               @Store equation
GRIDMAP 'f' INDEP                 @Setup GRIDMAP plot
'R' DEPND                         @and independent variables
-0.8 0.8 XVOL                      @Setup viewing range
-0.8 0.8 YVOL
500. 2000. XXRNG                  @Setup plotting range
1 3 YYRNG
16 NUMX 3 NUMY                    @Setup plot steps
-0.8 0.8 XRNG                     @Setup PPAR parameters
-0.8 0.8 YRNG                     @for drawing the axes
{(0,0) {0.1 0.1}}                 @and the labels
"RE(I)" "IM(I)"                   @correctly.
AXES
0 0.01
FOR I                               @Loop over time
  I 't' STO                        @Store value in variable 't'
  ERASE PICT                       @Put the title "t=value"
  {#0 #0}                          @at the upper right corner
  "t=" t + "s" +                  @of the PICT.
```

```
1 ->GROB REPL
DRAX LABEL DRAW @Draw axes, labels and gridmap.
PICT RCL        @Recall PICT on stack
.00005
STEP            @Increase time by .00005 seconds
TEXT           @Switch back to stack display
LCD-> PICT STO  @return a GROB of this and store
               @it in PICT.
{ 21. {#0d #14d} @Animate the GRIDMAP plots
  .2 0. }        @so that the header of the
ANIMATE          @stack screen is visible.
```

>>

The command sequence LCD-> PICT STO was added because of the following reason: When the command ANIMATE shows an animation starting at some screen coordinates other than than {#0 #0} (upper left), the contents of the PICT are shown on the part of the screen where no animation is shown. Because we wanted the header of the stack display to be shown over the animation, we switch to the stack display with the command TEXT, then we make a GROB with the contents of the current display (which is the stack with its header) using the command LCD->, and then we store this GROB in PICT.



When the program finishes it shows an animated GRIDMAP of the

Complex numbers with the HP49G - Part 4

function $\frac{f e^{i 2 \pi f t}}{(f - 1000 i \sqrt{3}) R}$ for t going from 0 to 0.01 seconds. You

can see how the different frequencies move around with different angular frequencies, that is the frequency of 2000 Hz needs less time to fulfil one circle (or period) than the frequency of 500 Hz. The different points of the grid also seem to vary their distance from the centre of the plot, but this is only an illusion that is created because the aspect ratio of the plot is not 1. (The horizontal direction contains about twice as many pixels for the same length as the vertical direction.) perhaps you repeat the program run replacing the line -0.8 0.8 XVOL with -1.6 1.6 XVOL and also the line -0.8 0.8 XRNG with -1.6 1.6 XRNG. Then all points of the grid will retain their distance from the centre of the plot for all times. Algebraically that means that the absolute value of

$\frac{f e^{i 2 \pi f t}}{(f - 1000 i \sqrt{3}) R}$ doesn't depend on time. We already have found

that absolute value. It was $\frac{f}{\sqrt{f^2 + 3000000} R}$, which really is time

independent. As the points of the grid move around, the projection of them on the x-Axis oscillates. It oscillates in exactly the same way as the expression for the value of the current

$\frac{f \cos \text{ATAN} \frac{1000 \sqrt{3}}{f} + 2 \pi f t}{\sqrt{f^2 + 3000000} R}$ does, for the same values of f

and R , because it is the real part of $\frac{f e^{i 2 \pi f t}}{(f - 1000 i \sqrt{3}) R}$.

Finishing this rather lengthy part of the complex marathon, let's look at the meaning of the parameters in VPAR and PPAR for the plot type GRIDMAP. The variable VPAR contains the list:

x_{Left} x_{Right} y_{Near} y_{Far} z_{Low} z_{High} x_{min} x_{max} y_{min} y_{max}
 x_{Eye} y_{Eye} z_{Eye} x_{Step} y_{Step}

x_{Left} and x_{Right} Real numbers that specify the start and the end of the view space in X direction, that is the width of the view space. Corresponding command is XVOL which takes two real numbers as arguments.

y_{Near} and y_{Far} Real numbers that specify the start and the end of the view space in Y direction, that is the depth of the view space. Corresponding command is YVOL which takes two real numbers as arguments.

z_{Low} and z_{High} : Two real numbers used as the plotting range for the first independent variable, that is the range of values in which this variable varies. Corresponding command is XXRNG.

x_{min} and x_{max} : Two real numbers used as the plotting range for the second independent variable, that is the range of values in which this variable varies. Corresponding command is YYRNG.

y_{min} and y_{max} : Real numbers that specify the number of plot points in X and in Y direction. The two commands for these parameters are NUMX and NUMY which take a real number as argument.

The other parameters in VPAR are not used.

The variable PPAR contains the list:

indep A name specifying the independent variable of the

Complex numbers with the HP49G - Part 4

expression that we want to plot. Default is X.
Corresponding command is INDEP.

ptype For this plot type the parameter is GRIDMAP, the
same as the command that sets this plot type.

depnd A name that specifies the second independent
variable. Default is Y. The command for this
parameter is DEPND.

Last thing to say here, **a big big big thanks to Alban Schann**,
who read and read the marathons again and again, and sent me so many
corrections that I must wonder where my mind was when I wrote the
stuff. It must have been Trabakoulas and his Ouzo ;-)

There is going to be another complex thing, the complex analysis
marathon in future. But next comes the calculus marathon. (Is that you
smiling captain? ;-)) I hope you enjoyed this. Keep the math fun high
and take care.

eⁱ greetings
Nick.