Microsoft™ COM for Complex Variables

Dr. G. B. Swartz
Professor Mathematics
Monmouth University
W. Long Branch, NJ 07764
Swartz@monmouth.edu

ICTCM
The Twelfth Annual International Conference on
Technology in Collegiate Mathematics
Burlingame, CA
November 5, 1999

Abstract

Our senior mathematics course in Complex Variables uses technology for complex arithmetic and mappings. Microsoft COM technology is used to produce a component in Visual Basic code that can be used in programs or in Excel. The component encapsulates the data structure and operations for complex arithmetic. The component is used in programming projects for complex valued computations and plotting. We show example projects and how to create the COM complex component.

Introduction

Complex numbers are composed of two real quantities. Arithmetical operations on such manipulate the two and produce results using real arithmetic. We usually program this one quantity at a time. However in a course focused on Complex Variables this is unproductive work for the participants. We create for them a component that allows reference to a complex number and its arithmetic by normal programming constructions. The technology that allows us to do this with ease is the Microsoft COM technology for encapsulating a data structure and its operations in a programming unit that can be referenced in a program. The component is connected to an Excel spreadsheet using Visual Basic Application (VBA is the macro language built into Excel) to compute the complex values. The class project included: the computation of parallel impedance's, plotting the complex valued output of a bandpass filter, using complex valued functions to estimate the real derivative of a function, to make a 3d plot of the magnitude of a complex valued function of a complex variable, and finally to show how a circle in the complex plane is mapped by a complex valued function of a complex variable.

Components are written in Visual Basic 6, tested and compiled as an active X dll. The extension of the compiled file is that of a dynamic linked library, a dll. Components have the advantage that they are protected from the user so that a user cannot modify the code and they provide a natural syntax to the inherently complicated syntax of complex arithmetic.

The Complex Com

The complex com is written to support the following complex operations. These are grouped as properties and methods, which are classified as subroutines or functions.
Under properties we have Re and Im for the real and imaginary part of the complex number. Under methods of the type subroutines we have: Create(real, imag), Recip() for the reciprocal of the current complex number, Add(z) and Mult(z) for addition of z or multiplication of z by the current complex number. Under functions we have: Magnitude() for the absolute value and FormatComplex(Optional n=4) which returns a string formatted with n decimal places of the Cartesian format for a complex number.
At the end of this paper we will extend the Com to include a dependent Com, Circle, to show how that is done.

Basic Complex Arithmetic

We illustrate the basic of complex arithmetic and show how it would be done using ComComplex. Let z1=a+bi and z2=c+di. The x term is called the real part of z, in Com it is z1.Re; and y is the imaginary part, z1.Im.The sum is a+c + (b+d)i, in Com it would be done as z1.Add z2 and z1 would contain the sum. Multiplication results in ac-bd + (bc+ad)i, in Com as z1.Mult z2. For example (3+4i).Mult (1-2i) results in 11-2i. The magnitude is $|z|=sqrt(x^2+y^2)$ and in com as a=z.magnitude. The format command makes z1 into a string for printing, z1.FormatComplex() results in "3.00 + I(4.00)". The Create command makes the object, thus to establish z1 in code we issue the command: z1.Create(real:=3,imag:=4), using named arguments.

Using Excel

To use these commands on an Excel spreadsheet we need to use VBA to take a cell value and send it to the com and then use VBA to send the result back to the cell. To have Excel multiply two complex numbers, z1 and z2, you enter 3 and 4 into cells A1 and B1, enter 1 and –2 into C1 and D1. You select the next two cells, E1 and F1, and enter the function =Multiply(a1,b1,c1,d1) and array enter the function. Array enter means to press CTRL+SHIFT+ENTER. The function in VBA calls Com and returns the answer, 11 and –2 to the two cells. In the VBA code, which you get to by selecting Tools/Macro/Visual Basic Editor, you enter:

```
Function Multiply(a,b,c,d)
Dim z as Complex, u as Complex
Dim P(1,1)
Set z = New Complex
z.Create a, b
Set u = New Complex
u.Create c, d
z.Mult u
P(0,0)=z.Re
P(0,1)=z.Im
Multiply=P
End Function
```

The Dim commands identify the Com objects and the array P, and the Set command instantiates each object. To return values to the Excel spreadsheet we must use an array, P, and assign it to the name of the function, as in the last line. To connect the Complex.dll file to the program you select in the VB Editor the menu: Tools/Reference/Browse and click on the file complex.dll.

Parallel Impedance Project

The first project is to find the parallel impedance Z of a circuit with impedance Z1 in parallel with a circuit of impedance Z2. The parallel impedance Z=1/(1/Z1+1/Z2) is a fairly difficult complex calculation. We did this with a resonant circuit of an inductor with Z1=iwL and Z2=i/wC. This has infinite Z at wo=1/sqrt(LC), as is well known. If resistance is added to L, as it would be due to coil resistance, the point of max|Z| is the solution of a quintic in w and is not a simple formula, we found using Maple that the point of resonance is given approximately by the following formula, this corrects a published formula in Boylestad, *Introductory Circuit Analysis, 7ed*,Merrill, 1994. We found w0adj=wo*sqrt(1-($R^2C/L)^2$/2). We used Maple to investigate this in a symbolic manner.

Maple code and approximate resonant frequency for R+iwL in parallel with i/(wC):

```
> Z1:=A+I*w*L; Z2:=B+I/(w*C);
> Z:=simplify(evalc(1/(1/Z1+1/Z2)));
> mag:=simplify(evalc(abs(Z)^2)):
> dmag:=simplify(diff(mag,w)):
> sol:=solve(dmag=0,w):
> w1:=sol[5];
```

This led us to investigate computation with a balanced circuit of R in both the inductor and capacitor. We found, what is a new result, that the resonance does not detune. The project plots the selectivey as |Z|/Zmax, where Zmaz=R(1+L/$R^2C)^2$/2. The VBA code is:

```
Function parallel(a, b, c, d)
Dim p(1, 2)
Dim z1 As New Complex
Dim z2 As New Complex
z1.Create a, b
z2.Create c, d
z1.recip
z2.recip
z1.Add z2
z1.recip
p(0, 0) = z1.Re
p(0, 1) = z1.Im
parallel = p
End Function
```

The worksheet output is in Table 1, and the charts are in Figs 1 and 2.

Table 1

w	A	L	B	C	Z		w	Mag	
0.200	0.200	0.400	0.200	-20.000	0.2083	0.4060	0.200	0.4563	A=1
0.400	0.200	0.800	0.200	-10.000	0.2375	0.8636	0.400	0.8957	L=2w
0.600	0.200	1.200	0.200	-6.667	0.3060	1.4483	0.600	1.4803	
0.800	0.200	1.600	0.200	-5.000	0.4717	2.3092	0.800	2.3569	B=1
1.000	0.200	2.000	0.200	-4.000	0.9654	3.8269	1.000	3.9468	C=-4/w
1.200	0.200	2.400	0.200	-3.333	3.2879	7.2052	1.200	7.9199	
1.400	0.200	2.800	0.200	-2.857	19.7020	2.7860	1.400	19.8980	
1.600	0.200	3.200	0.200	-2.500	5.0985	-8.5723	1.600	9.9739	
1.800	0.200	3.600	0.200	-2.222	1.7469	-5.3283	1.800	5.6074	
2.000	0.200	4.000	0.200	-2.000	0.9654	-3.8269	2.000	3.9468	
2.200	0.200	4.400	0.200	-1.818	0.6665	-3.0108	2.200	3.0837	
2.400	0.200	4.800	0.200	-1.667	0.5191	-2.4997	2.400	2.5530	
2.600	0.200	5.200	0.200	-1.538	0.4347	-2.1483	2.600	2.1918	
2.800	0.200	5.600	0.200	-1.429	0.3813	-1.8908	2.800	1.9289	
3.000	0.200	6.000	0.200	-1.333	0.3451	-1.6933	3.000	1.7281	
3.200	0.200	6.400	0.200	-1.250	0.3193	-1.5364	3.200	1.5692	
3.400	0.200	6.800	0.200	-1.176	0.3002	-1.4084	3.400	1.4400	
3.600	0.200	7.200	0.200	-1.111	0.2855	-1.3017	3.600	1.3326	
3.800	0.200	7.600	0.200	-1.053	0.2740	-1.2112	3.800	1.2418	
4.000	0.200	8.000	0.200	-1.000	0.2648	-1.1334	4.000	1.1640	
4.200	0.200	8.400	0.200	-0.952	0.2572	-1.0657	4.200	1.0963	
4.400	0.200	8.800	0.200	-0.909	0.2510	-1.0062	4.400	1.0370	
4.600	0.200	9.200	0.200	-0.870	0.2458	-0.9533	4.600	0.9845	
4.800	0.200	9.600	0.200	-0.833	0.2413	-0.9061	4.800	0.9377	
5.000	0.200	10.000	0.200	-0.800	0.2375	-0.8636	5.000	0.8957	
5.200	0.200	10.400	0.200	-0.769	0.2343	-0.8251	5.200	0.8577	
5.400	0.200	10.800	0.200	-0.741	0.2314	-0.7901	5.400	0.8233	
5.600	0.200	11.200	0.200	-0.714	0.2289	-0.7580	5.600	0.7918	
5.800	0.200	11.600	0.200	-0.690	0.2267	-0.7286	5.800	0.7631	
6.000	0.200	12.000	0.200	-0.667	0.2248	-0.7015	6.000	0.7366	
6.200	0.200	12.400	0.200	-0.645	0.2230	-0.6764	6.200	0.7122	
6.400	0.200	12.800	0.200	-0.625	0.2215	-0.6531	6.400	0.6896	
6.600	0.200	13.200	0.200	-0.606	0.2201	-0.6314	6.600	0.6687	
6.800	0.200	13.600	0.200	-0.588	0.2188	-0.6112	6.800	0.6492	
7.000	0.200	14.000	0.200	-0.571	0.2176	-0.5922	7.000	0.6310	
7.200	0.200	14.400	0.200	-0.556	0.2166	-0.5745	7.200	0.6140	
7.400	0.200	14.800	0.200	-0.541	0.2156	-0.5578	7.400	0.5980	
7.600	0.200	15.200	0.200	-0.526	0.2148	-0.5421	7.600	0.5831	
7.800	0.200	15.600	0.200	-0.513	0.2140	-0.5272	7.800	0.5690	
8.000	0.200	16.000	0.200	-0.500	0.2132	-0.5132	8.000	0.5557	
8.200	0.200	16.400	0.200	-0.488	0.2126	-0.4999	8.200	0.5432	
8.400	0.200	16.800	0.200	-0.476	0.2119	-0.4873	8.400	0.5314	
8.600	0.200	17.200	0.200	-0.465	0.2114	-0.4754	8.600	0.5203	
8.800	0.200	17.600	0.200	-0.455	0.2108	-0.4640	8.800	0.5097	
9.000	0.200	18.000	0.200	-0.444	0.2103	-0.4532	9.000	0.4996	

Fig. 1

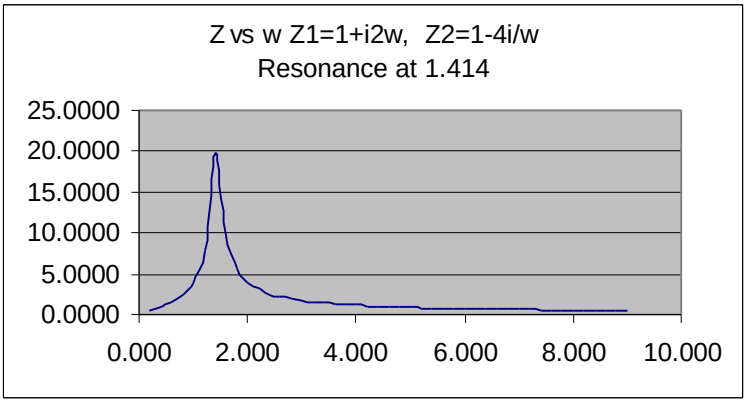
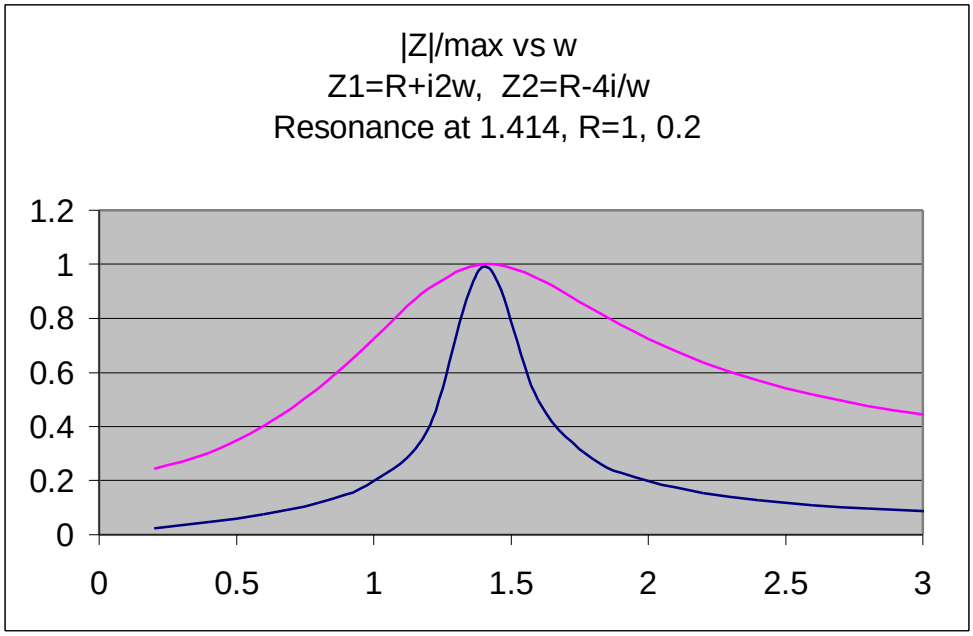


Fig. 2

Array Enter the function into E2:F2 using Ctrl + Shift + Enter.
{=Parallel(A2,B2,C2,D2)}

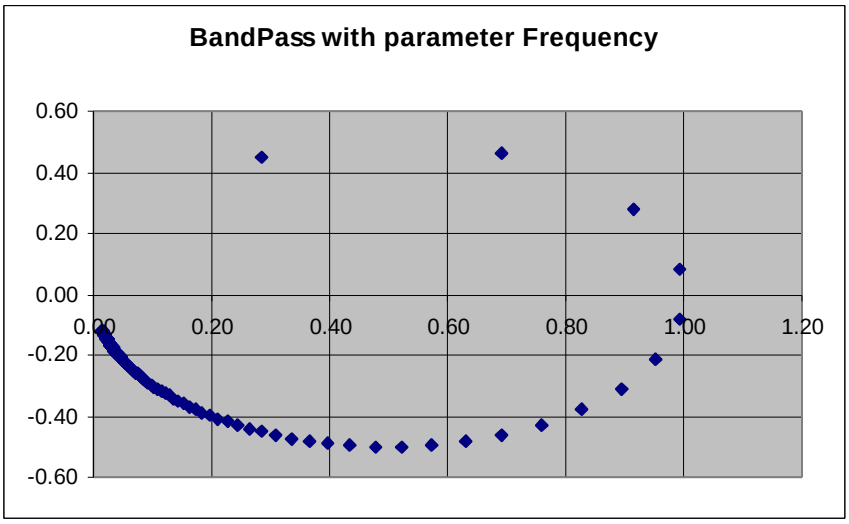


BandPass Filter Project

The second project is to plot the complex valued output of the transfer function for a low pass filter, LP, a high pass filter, HP and their product, a bandpass filter, BP. Most engineers are accustomed to think of these plots in terms of magnitude where they show a gain of 1 over the pass region of frequencies and near 0 outside the pass range. However, the output is a complex valued quantity, so in this project we plot the Cartesian complex number of the output as frequency varies. The code in the worksheet is:

```
Function BandPass(f, q)
Dim p(1, 2)
Dim z As Complex
Set z = New Complex
Dim u As Complex
Set u = New Complex
f1 = -1 / f: i2 = q * f
z.Create real:=1, imag:=f1
u.Create real:=1, imag:=i2
z.recip
u.recip
z.mult u
p(0, 0) = (1 + q) * z.Re
p(0, 1) = (1 + q) * z.Im
BandPass = p
End Function
```

The chart is:



Estimate of Real Derivative

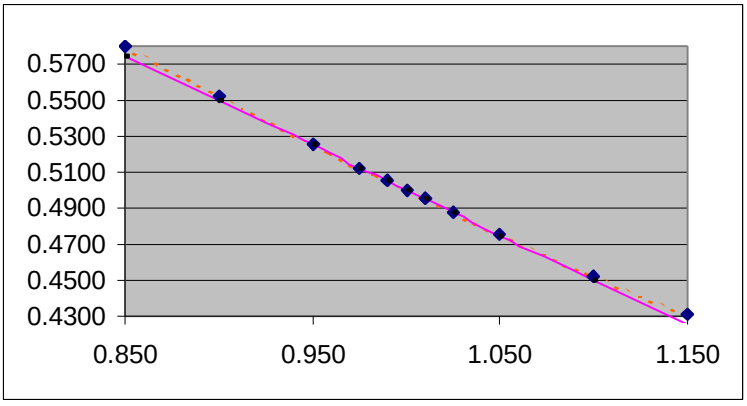
The real derivative of a real valued function, f(x), can be obtained from IM(f(x+ih)/h) for a small value of h. This estimate is better than the usual (f(x+h)-f(x))/h or (f(x+h)-f(x-h))/2h. The VBA code is:

```
Function diff(a, h)
Dim z As Complex
Dim w As Complex
Set z = New Complex
z.Create real:=a, imag:=h
Set w = Witch(z)
diff = w.Im / h
End Function
```

The function diff call a function Witch(z) which is the Witch of Agnassi, $1/(1+z^2)$. The code for it illustrates how to write a function that returns an object, in this case a complex object:

```
Function Witch(z As Complex) As Complex
Dim w As Complex
Set w = New Complex
z.mult z
kk = z.Re + 1
jj = z.Im
w.Create real:=kk, imag:=jj
w.recip
Set Witch = w
End Function
```

The result is shown in the chart showing the tangent line at x=1:



The worksheet is set up in terms of h for the value of x in cell A1, the function is called as =diff(A3,A1). The worksheet is in Fig 3.

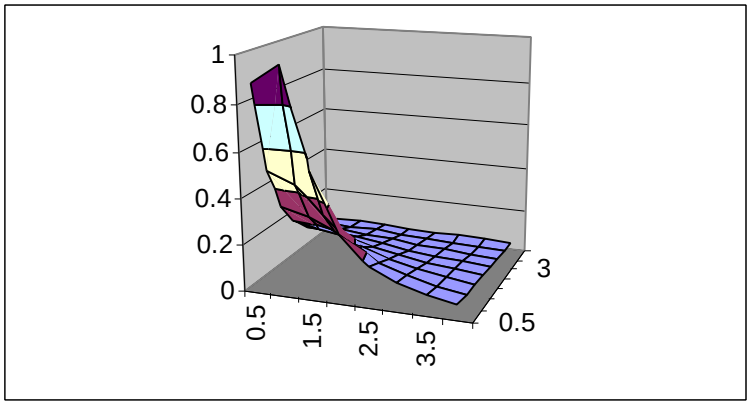
1

h	m
0.150	-0.49994
0.100	-0.49999
0.050	-0.50000
0.025	-0.50000
0.010	-0.50000

Fig. 3.

3D Plot of A Complex Valued Function

Visualizing a map of a complex valued function of a complex variable is not easy due to the high dimension of the graph space. We can, however, show the magnitude of the function over the complex plane in a 3D plot. This project shows the magnitude of the Witch function on the plane defined by [0..3] X [0..2].



The worksheet is set up as an array of the real and imaginary parts of the domain:

	A	B	C	D	E	F	G	H	I	J	K
1			=Witch2($B3,C$2) in each cell				Imag				
2		0.000	0.000	0.500	1.000	1.500	2.000	2.500	3.000	3.500	4.000
3		0.000	1.000	1.333		0.800	0.333	0.190	0.125	0.089	0.067
4		0.500	0.800	0.894	0.970	0.555	0.294	0.179	0.120	0.087	0.065
5		1.000	0.500	0.496	0.447	0.332	0.224	0.152	0.108	0.081	0.062
6	Real	1.500	0.308	0.298	0.267	0.217	0.165	0.124	0.094	0.072	0.057
7		2.000	0.200	0.194	0.177	0.152	0.124	0.099	0.079	0.063	0.052
8		2.500	0.138	0.135	0.125	0.111	0.095	0.080	0.066	0.055	0.046
9		3.000	0.100	0.098	0.092	0.084	0.075	0.065	0.055	0.047	0.040
10		3.500	0.075	0.074	0.071	0.066	0.060	0.053	0.047	0.041	0.036
11		4.000	0.059	0.058	0.056	0.053	0.049	0.044	0.040	0.035	0.031

The code in VBA in the Excel project for the worksheet is:

```
Function Witch2(a, b)
Dim w As Complex
Set w = New Complex
Dim z As Complex
Set z = New Complex
z.Create real:=a, imag:=b
z.mult z
kk = z.Re + 1
jj = z.Im
w.Create real:=kk, imag:=jj
w.recip
Witch2 = w.magnitude
End Function
```

How to Create a COM Object

The process to make the COM object is to write a class in Visual Basic 6, VB6, and compile it as a DLL. Start VB6 with an Active X dll project. Add a class module to the project. Make sure its instancing property is set to 5-Multiuse on the property sheet for the class. Make the name property of the class: complex. The properties of a class are viewed in the property view which can be selected from the View menu or by pressing F4. Write the following code and do not use the class builder wizard as it adds too much code, more than we need for this simple class. The code is shown below. You can add a form to test the code, but VB6 automatically changes the instancing property and project properties when you do, you must change them back after testing when you remove the form. Compile the class by selecting from the menu File/Make dll.

Code for the COM Complex Class in VB6

```
Private mvre
Private mvim
Public Property Get Re()
  Re = mvre
End Property
Public Property Get Im()
  Im = mvim
End Property
Public Sub Create(real, imag)
  mvre = real
  mvim = imag
End Sub
```

This part of the class defines private internal values to hold the real and imaginary parts of the complex number. The Create subroutine is used to set these values. The values are made available to the calling program using the property functions Get Re and Get Im. These return the internal values of the real and imaginary parts.

```
Public Function magnitude()
  magnitude = Sqr(mvre * mvre + mvim * mvim)
End Function
Public Function FormatComplex(Optional n = 4) As String
  FormatComplex = FormatNumber(mvre, n) & " + i" _
    & "(" & FormatNumber(mvim, n) & ")"
End Function
```

The magnitude function in the class returns the magnitude of the number, using the internal values. The format function returns a string formatted to show the number in Cartesian format as 1.00 + I(2.00). This uses an optional argument n with default value of 4. If the calling program uses the syntax without the n, the program will use 4 in it's place. Functions must have the function name used on the left side of an assignment statement to return the value to the calling program.

```
Public Sub add(z As Complex)
  mvre = mvre + z.Re
  mvim = mvim + z.Im
End Sub
Public Sub mult(z As Complex)
Dim mvre1
  mvre1 = mvre * z.Re - mvim * z.Im
  mvim = mvre * z.Im + mvim * z.Re
  mvre = mvre1
End Sub
Public Sub recip()
  Dim a
  a = mvre * mvre + mvim * mvim
  mvre = mvre / a
  mvim = mvim / a
End Sub
```

The operations of add, multiply and inverse are written as subroutines to act on the internal values of the complex number. The calling program will use these with syntax of z.add u where z and u have been created as complex numbers. The real and imaginary parts of the argument are used with the internal real and imaginary parts to calculate the results that update the internal values. Nothing is returned, only the internal values are modified. The Dim statements indicate to the compiler that a new variable is in use. If we use in VB6, the option explicit feature, this prevents an error, otherwise the Dim is not necessary.

Map a Circle by Squaring

In our final project we will take points on a circle, square them and visualize the resulting shape. We expect another circle, but as we will see this is not so if the circle crosses the imaginary axis. The COM will be modified to add a class for Circle. The Circle class will be a complex number on the circle. The circle will be created with a center at coordinates (a, b) with radius r. We will write a method to return the center of the circle and one to move the point to t radians on the circle. The Circle class will use the complex number class and so is a dependent class. The method of writing a dependent class is shown in the Circle class code:

Class circle:

```
Private mvrecenter
Private mvimcenter
Private mvradius
Private mvz As Complex
Private Sub Class_Initialize()
  Set mvz = New Complex
End Sub
Sub CCreate(a, b, r)
  mvrecenter = A
  mvimcenter = b
  mvradius = r
  mvz.Create real:=mvrecenter, imag:=mvimcenter
End Sub
```

The private variables maintain the internal state of the circle. The complex class is connected to this class in the Class_Initialize event which is called automatically when the circle object is instantiated with a set statement. The Ccreate method sets the center and radius of the internal values as a complex object.

```
Function Eval(t) As Complex
Dim x
Dim y
Dim u As Complex
Set u = New Complex
x = mvradius * Cos(t)
y = mvradius * Sin(t)
u.Create real:=x, imag:=y
u.add mvz
Set Eval = u
End Function
```

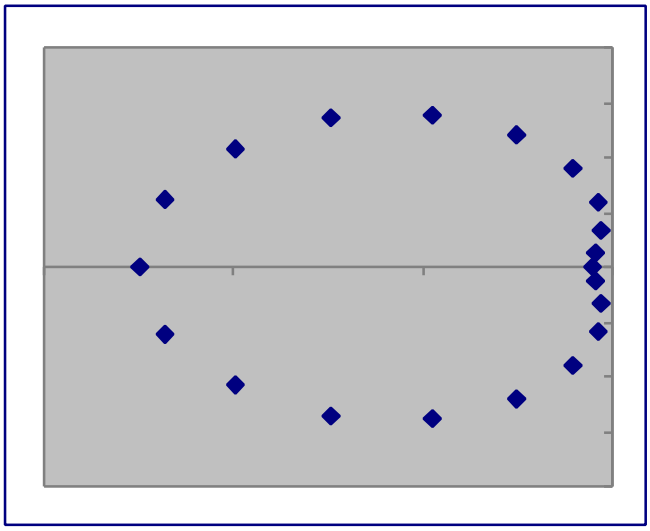
The function Eval(t) returns a complex object with a value on the circle at radian measure t.

```
Function CCenter()
Dim cc As Complex
Set cc = New Complex
cc.Create real:=mvrecenter, imag:=mvimcenter
Set CCenter = cc
End Function
```

The function Ccenter() returns the center of the circle and may be used for documentation purposes.

A chart of the output is in Fig. 4.

Fig. 4.



References

- Lomax, Paul, *VB & VBA In A Nutshell*, O'Reilly, 1998
- Walkenbach, John, *Excel 2000 Programming*, IDG, 1999
- www.j-walk.com