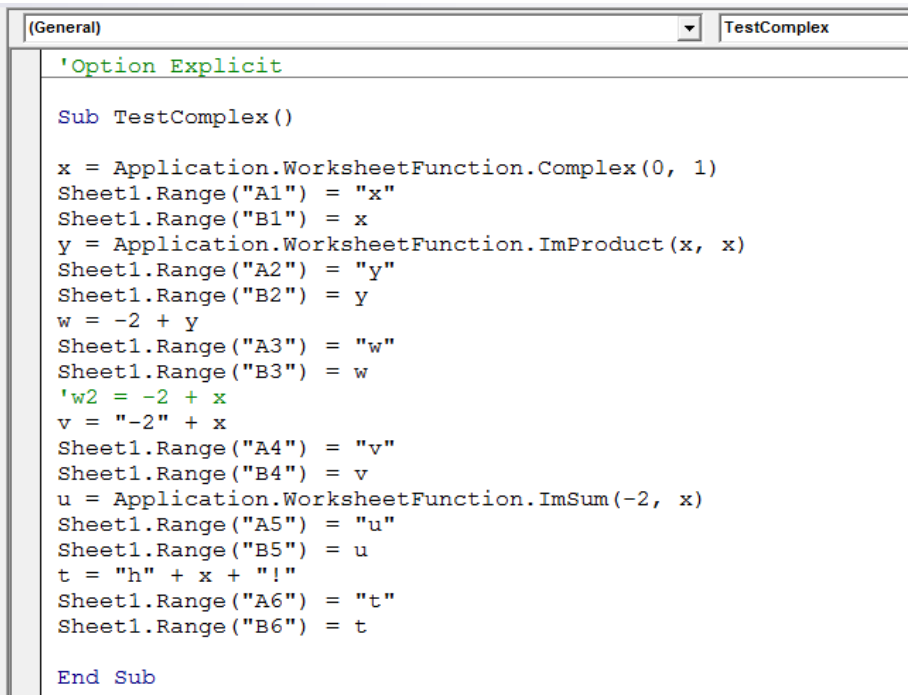
Hochschule RheinMain
Prof. Dr. D. Lehmann

SS 2021

5. Übungsblatt zur Vorlesung
Finanzmathematik mit Excel und VBA

Aufgabe 1) Debugging-Tools: Öffnen Sie den VBA-Editor, fügen Sie ein neues Modul ein und geben Sie dann folgenden Code ein:




```
'Option Explicit

Sub TestComplex()

x = Application.WorksheetFunction.Complex(0, 1)
Sheet1.Range("A1") = "x"
Sheet1.Range("B1") = x
y = Application.WorksheetFunction.ImProduct(x, x)
Sheet1.Range("A2") = "y"
Sheet1.Range("B2") = y
w = -2 + y
Sheet1.Range("A3") = "w"
Sheet1.Range("B3") = w
'w2 = -2 + x
v = "-2" + x
Sheet1.Range("A4") = "v"
Sheet1.Range("B4") = v
u = Application.WorksheetFunction.ImSum(-2, x)
Sheet1.Range("A5") = "u"
Sheet1.Range("B5") = u
t = "h" + x + "!"
Sheet1.Range("A6") = "t"
Sheet1.Range("B6") = t

End Sub
```

Achten Sie darauf, dass der `Option Explicit` Befehl herauskommentiert ist. Führen Sie das Macro aus und versuchen Sie genau zu verstehen, was passiert. Aktivieren Sie jetzt den `Option Explicit` Befehl und führen Sie das Makro erneut aus. Sie bekommen eine Fehlermeldung, da Sie jetzt gezwungen sind, sämtliche Variablen explizit zu deklarieren. Das ist in diesem Fall jedoch nicht so klar, da nicht klar ist, welchen Datentyp die Funktion

`Application.WorksheetFunction.Complex()`

zurückgibt. Um das herauszufinden, sind die Debugging-Tools 'Locals Window' und 'Break Points' oder Halte-Punkte sehr nützlich. Tun Sie zunächst den `Option Explicit` Befehl wieder herauskommentieren, so dass das Makro wieder funktioniert. Lesen Sie sich dann in Urtis_ExcelVBA unter

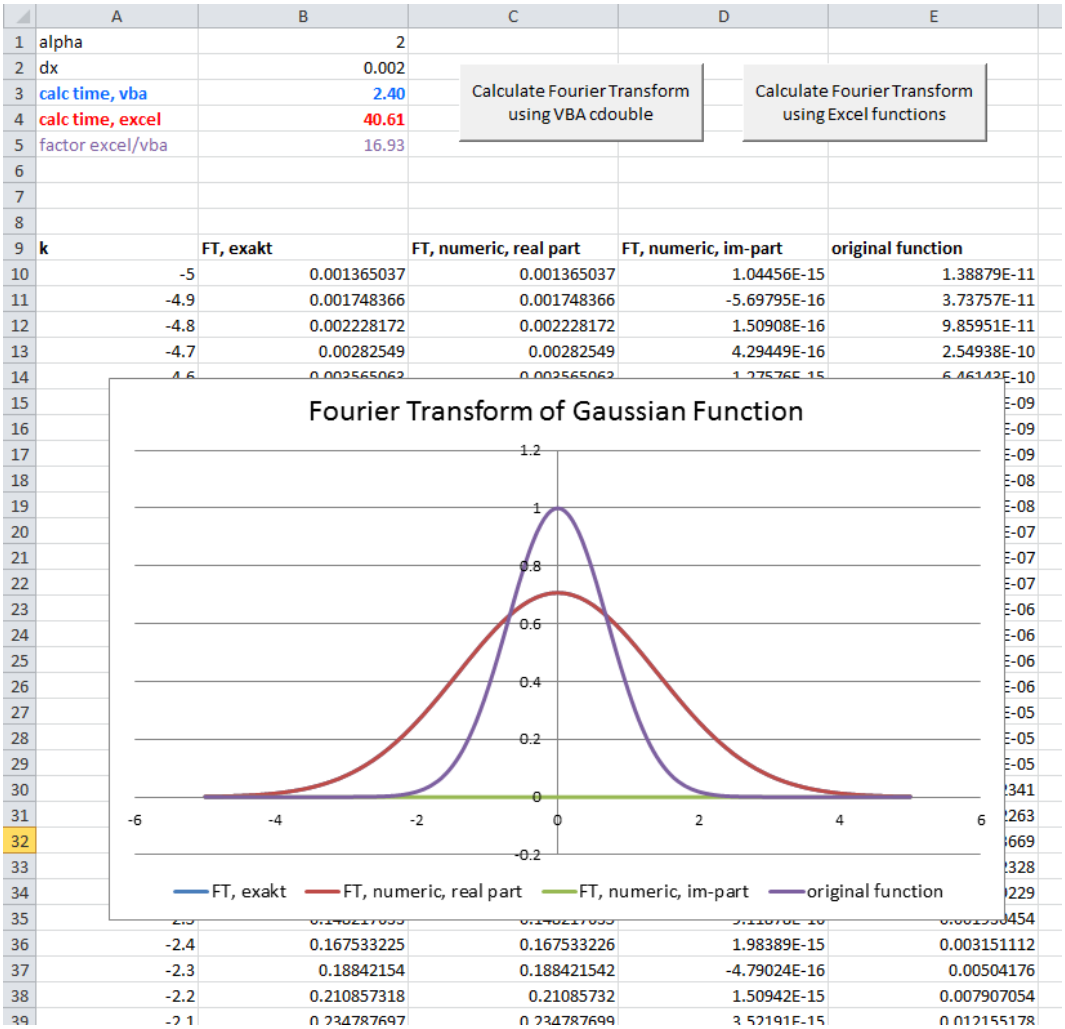
http://hsrm-mathematik.de/SS2021/semester4/ExcelVBA/book.pdf

auf Seite 198 unten die 6 Zeilen 'The Debugging Toolbar' durch und machen Sie diese Toolbar sichtbar in Ihrem VBA-Editor. Schauen Sie sich dann auf Seite 199 oben rechts die Abbildung 17-6 an und klicken Sie in Ihrer Debugging-Toolbar auf den Locals Window Knopf, so dass

dieses Fenster sichtbar wird. Gehen Sie dann zurück zum Macro `Sub TestComplex()` und bewegen Sie den Eingabe-Prompt an das Ende der `End Sub` Zeile. Drücken Sie dann die F9-Taste, damit haben Sie einen Halte-Punkt gesetzt. Führen Sie dann mit der Taste F5 das Makro aus. Die code-execution stoppt in der letzten Zeile und Sie können sich jetzt alle Variablen im Locals-Window anschauen. Achten Sie insbesondere auf die Datentypen.

Ausgestattet mit diesen zusätzlichen Informationen, versuchen Sie jetzt das Makro zum Laufen zu bringen, auch wenn der `Option Explicit` Befehl aktiviert ist.

Aufgabe 2) Performance Excel-Funktionen vs. VBA-Funktionen, Part II: Laden Sie sich von der Vorlesungshomepage das Sheet `week5.xlsm` herunter und öffnen Sie den VBA-Editor. Fügen Sie ein neues Modul ein und nennen Sie es etwa 'UsingExcelFunctions'. Kopieren Sie den Code vom `Sub CalcFourierTransform()` in dieses neue Modul und tun Sie dann dieses kopierte Macro umbenennen zu, etwa, `CalcFTwithExcelFunctions()`. Modifizieren Sie jetzt den Code dieses Moduls derart, dass der benutzerdefinierte Datentyp `cdouble` und die damit verbundenen Funktionen wie `cplus`, `cmult` und `cexp` nicht mehr benutzt werden. An deren Stelle sollen die Excel-Funktionen für komplexe Zahlen, die man im Insert Function Dialog-Fenster unter der Kategorie 'Engineering' findet, verwendet werden. Die Rechenzeit soll wieder auf das Sheet geschrieben werden. Ihre Ausgabe könnte also folgendermassen aussehen:



	A	B	C	D	E
1	alpha	2			
2	dx	0.002			
3	calc time, vba	2.40			
4	calc time, excel	40.61			
5	factor excel/vba	16.93			
6					
7					
8					
9	**k**	**FT, exakt**	**FT, numeric, real part**	**FT, numeric, im-part**	**original function**
10	-5	0.001365037	0.001365037	1.04456E-15	1.38879E-11
11	-4.9	0.001748366	0.001748366	-5.69795E-16	3.73757E-11
12	-4.8	0.002228172	0.002228172	1.50908E-16	9.85951E-11
13	-4.7	0.00282549	0.00282549	4.29449E-16	2.54938E-10
36	-2.4	0.167533225	0.167533226	1.98389E-15	0.003151112
37	-2.3	0.18842154	0.188421542	-4.79024E-16	0.00504176
38	-2.2	0.210857318	0.21085732	1.50942E-15	0.007907054