Comparing Matlab to Excel/VBA

Jake Blanchard
University of Wisconsin - Madison
August 2007

Surveys

- Excel
- VBA
- Solver
- Iteration

Overall Comparison

- Matlab is
  - Faster
  - More powerful
  - More comprehensive
- Excel is
  - Ubiquitous
  - Familiar to more engineers
  - Constrained optimization is much easier
  - Linear (but non-polynomial) curve fits are easier

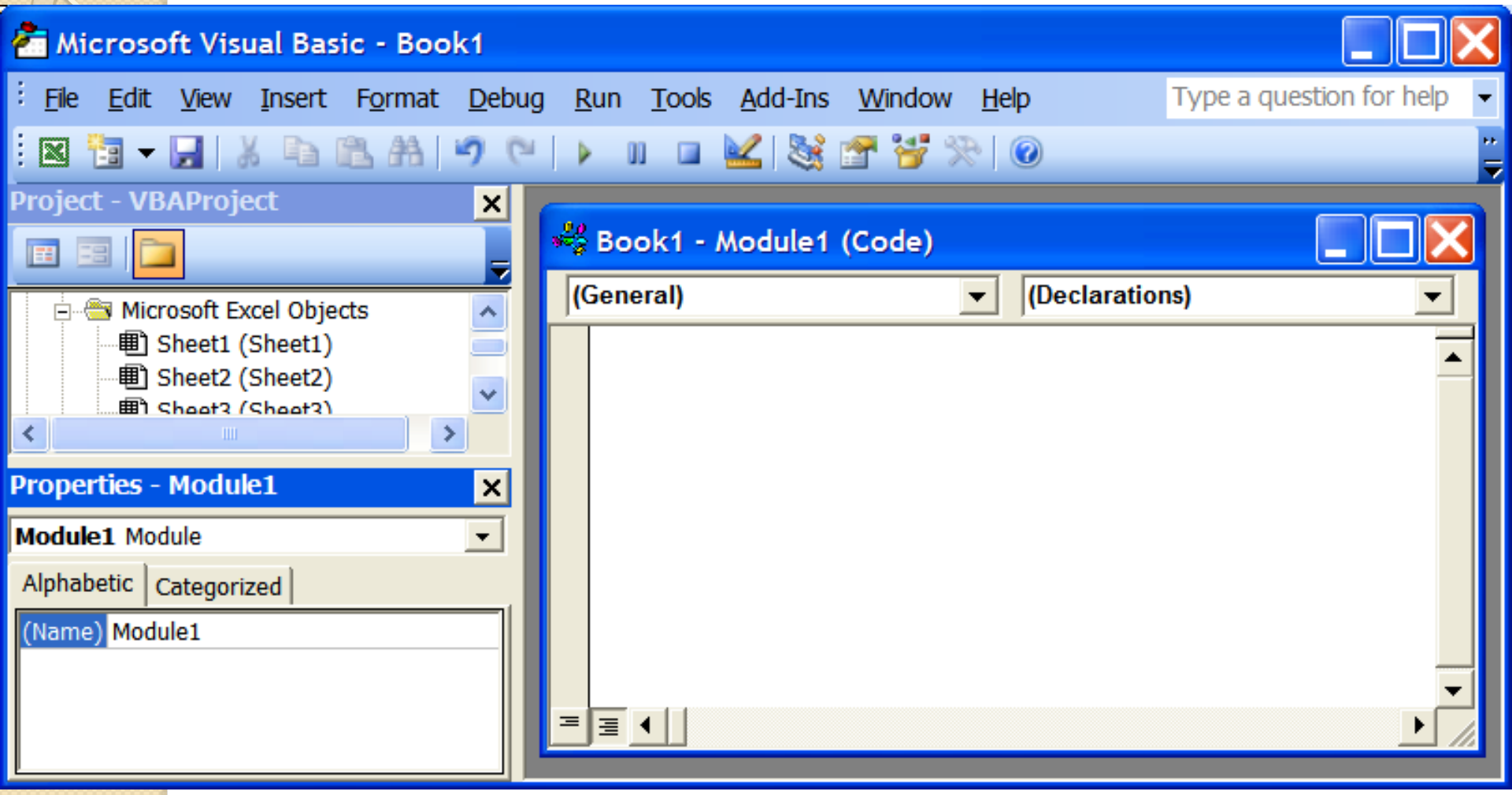
VBA Macros

- Macros allow one to add significant power to Excel
- They are small programs that can be called from a spreadsheet
- You can create functions or subroutines
- If you want to get fancy, you can add a user interface as well

Using Macros

- Macros are written in a Basic-like language called Visual Basic for Applications
- Excel comes with a separate macro editor
- To create a macro, go to Tools/Macro/Visual Basic Editor, then within the Editor go to Insert/Module

You should get this...

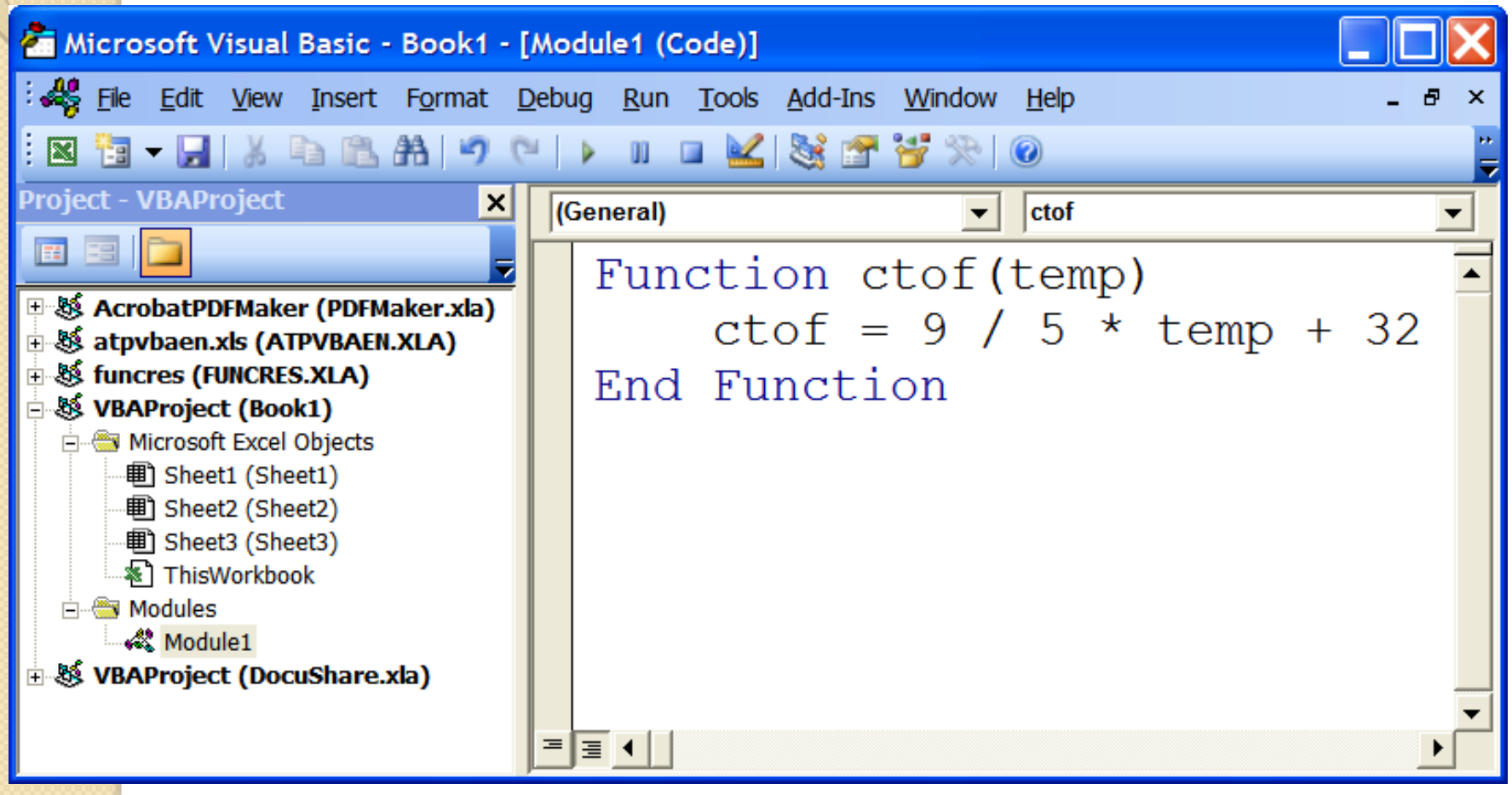


Creating a Function

- Suppose we want to create an Excel function that takes a temperature in Celsius and converts to Fahrenheit
- We would type the following in a module:

```
Function ctof(temp)
    ctof = 9 / 5 * temp + 32
End Function
```

Now we have this...



Using the function

- Then you can go to the spreadsheet and type **=ctof(100)**
- Or, you can put the value of "100" into cell A1 and then type **=ctof(A1)** into some other cell
- In fact, this function can be used just as any built-in Excel function can be used

The Macro Language

- **Operators:** +, -, *, /, ^, Mod
- **Comparison:** =, <, >, <=, >=, <>
- **Logical Operators:** And, Eqv, Imp, Not, Or, Xor
- **Intrinsic Functions:** Abs, Cos, Sin, Tan, Atn (arc tangent), Exp, Log (natural), Sgn, Sqr (square root), Rnd (random number)

Flow Control

```
If condition Then
      statements
Else
    statements
End If
```

```
If x=0 Then
      f=1
Else
    f=sin(x)/x
End If
```

Flow Control

```
For counter=start To end
    statements
Next
```

```
For i=1 To 100
    sum=sum+i
Next
```

Flow Control

```
Do Until condition
    statements
Loop
```

```
i=1
x=1
Do Until i=50
    x=x*i
    i=i+1
Loop
```

Flow Control

```
Do While condition
    statements
Loop
```

```
i=1
x=1
Do While i<50
    x=x*i
    i=i+1
Loop
```

A factorial routine

```
Function fact(Z)
    x = 1
    ans = 1
    Do Until x = Z
        ans = ans * x
        x = x + 1
    Loop
    fact = ans
End Function
```

Another Solution

```
Function fact(Z)
    ans = 1
    For i = 1 To Z
        ans = ans * i
    Next
    fact = ans
End Function
```

Root-Finding

- Use fzero function in Matlab
- Use Solver in Excel
- Either are pretty simple
- Solver not as "automated" as the rest of Excel
- Solver Demo

Root-Finding Macro

```
Function newtroot(guess)
    x = guess
    tolerance = 0.0001
    Do
        xold = x
        x = x - fff(x) / fprime(x)
        diff = Abs((xold - x) / x)
    Loop Until diff <
   tolerance
    newtroot = x
End Function
```

```
Function fff(x)
    fff = x * Sin(x) - 1
End Function

Function fprime(x)
    fprime = Sin(x) + x *
   Cos(x)
End Function
```

Quadrature

- **quadl** function in Matlab
- No built-in routine in Excel
- Can easily add one in VBA
- I've provided a simple Simpson's routine
- Matlab routine is adaptive

Trapezoidal Macro

```
Function trap(a, b, N)
h = (b - a) / N
t = 0.5 * ff(a) + 0.5 * ff(b)
If N > 1 Then
    For i = 1 To N - 1
        x = a + i * h
        t = t + ff(x)
    Next
End If
trap = h * t
End Function
```

```
Function ff(x)
    ff = Sin(x)
End Function
```

Simpson's Rule

```
Function simp(a, b, N)
h = (b - a) / N
t = ff(a) + ff(b)
m = 4
For i = I To N / 2
    x = a + h * i
    xx = b - h * i
    t = t + m * ff(x) + m * ff(xx)
    If x = xx Then
        t = t - m * ff(x)
    End If
    If m = 4 Then
        m = 2
    Else
        m = 4
    End If
Next
simp = h / 3 * t
End Function
```

Solving initial value problems

- **ode45** routine in Matlab
- Others for more exotic equations
- Nothing in Excel
- I've supplied a fixed-time step RK routine
- We give up adaptive routine
- I once published an adaptive routine one could use

Runge-Kutta Routine

```
Function rk(t, y, dt)
  k1 = dt * f(t, y)
  k2 = dt * f(t + dt / 2, y + k1 / 2)
  k3 = dt * f(t + dt / 2, y + k2 / 2)
  k4 = dt * f(t + dt, y + k3)
  rk = y + (k1 + 2 * (k2 + k3) + k4) / 6
End Function

Function f(t, y)
    f = 1 + t + Sin(t * y)
End Function
```

2nd ORDER ODEs

```
Sub rk2(t, x, y, dt)
    k1 = dt * f2(t, x, y)
    l1 = dt * g2(t, x, y)
    k2 = dt * f2(t + dt / 2, x + k1 / 2, y + l1 / 2)
    l2 = dt * g2(t + dt / 2, x + k1 / 2, y + l1 / 2)
    k3 = dt * f2(t + dt / 2, x + k2 / 2, y + l2 / 2)
    l3 = dt * g2(t + dt / 2, x + k2 / 2, y + l2 / 2)
    k4 = dt * f2(t + dt, x + k3, y + l3)
    l4 = dt * g2(t + dt, x + k3, y + l3)
    x = x + (k1 + 2 * (k2 + k3) + k4) / 6
    y = y + (l1 + 2 * (l2 + l3) + l4) / 6
End Sub
```

Macro to Write Results to Sheet

```
Sub writeODE2()
NumPoints = Range("Npoints")
tNot = Range("tnot")
xNot = Range("xnot")
yNot = Range("ynot")
dt = Range("dt")
[C1].Select
ActiveCell.Value = "t"
ActiveCell.Offset(0, 1).Value = "x(t)"
ActiveCell.Offset(0, 2).Value = "y(t)"
ActiveCell.Offset(1, 0).Value = tNot
ActiveCell.Offset(1, 1).Value = xNot
ActiveCell.Offset(1, 2).Value = yNot
t = tNot
x = xNot
y = yNot
For i = 1 To NumPoints
Call rk2(t, x, y, dt)
t = t + dt
ActiveCell.Offset(i + 1, 0).Value = t
ActiveCell.Offset(i + 1, 1).Value = x
ActiveCell.Offset(i + 1, 2).Value = y
Next
End Sub
```

Monte Carlo Analysis

- =Rand() in sheet to get a uniform random number from 0 to 1
- Rnd for the same thing in VBA
- Histograms can be generated in the Analysis Toolpak

VBA for Random Numbers

```
Function getavg(N)
Count = 0
For i = 1 To N
    Count = Count + Rnd
Next
getavg = Count / N
End Function
```

Creating a GUI

- Can do it in Matlab
- Much easier in VBA
- Demo

Questions?