Midterm Review

Larry Caretto
Mechanical Engineering 209

Computer Programming for Mechanical Engineers

April 4, 2017

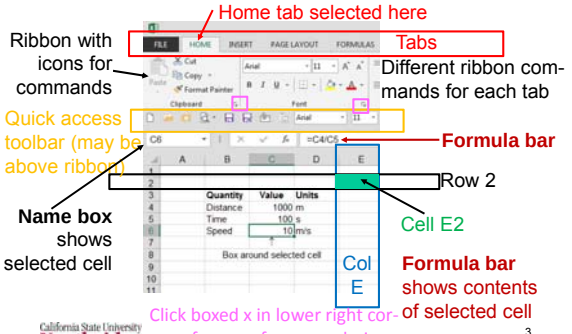
California State University Northridge

Outline

- Excel spreadsheet basics
- Use of VBA functions and subs
- Declaring/using variables and constants
- Operators and precedence
- If and ElseIf statements
- Loops: For, Do While, Loop Until
- What to expect on closed-book midterm
  - Bring your VBA information sheet

California State University Northridge 2

Midterm Review: Excel Basics



California State University Northridge 3

Using Excel

- Entering and copying formulas
  - Understanding absolute/relative references
- Excel functions like sin(), exp(), etc.
- Formatting cells
- Creating and modifying plots
- Creating macros (subs without arguments) can be called from worksheet
- Advanced topics like data validation and range names not on midterm

California State University Northridge 4

Using VBA

- Transfer to and use of editor
- Writing functions and subs
  - Functions/subs transfer information through the argument list, **based on position in list**
  - A macro is a sub with no arguments
  - Information transfer to/from macros by statements like the following
    - Range("A1").Value = x * y — Write property
    - Z = Cells(12,13).Value — Read property

California State University Northridge 5

Sample Exam Question

Write a function to compute the density, ρ, of an ideal gas from the equation ρ = MP/RT, where P is the pressure, R is the universal gas constant = 8.314462, and M is the molar mass. Your function should have R as a constant and have P, T, and the name of the gas as input parameters. Assume that you have an available function, getMolarMass(name) that finds M from the input substance name.

California State University Northridge 6

Solution to Sample Question

```
Const universalGasConstant as Double = 8.314462
Function idealGasDensity(substanceName As _
   String, pressure As Double, temperature As _
                                 Double) As Variant
    'Computes ideal gas density

    Dim molarMass As Variant
    molarMass = getMolarMass(substanceName)
    idealGasDensity = molarMass * pressure _
           / (universalGasConstant * temperature)
    End If
End Function
```

How would you call this function to find the density of air for P = 500 and T = 600?

```
Rho = idealGasDensity("Air", 500, 600)
```


A Further Question

- The getMolarMass function used for the previous solution returns the integer -9999 when it cannot find the molar mass
- Use this information to revise the previous function to return the message "ERROR: Molar mass not found for ####", where #### is the substance name input by the user

Solution with Error Checking

```
Const getMolarMassError as Double = -9999
Const universalGasConstant as Double = 8.314462
Function idealGasDensity(substanceName As _
   String, pressure As Double, temperature As _
                                 Double) As Variant
    'Computes ideal gas density

    Dim molarMass As Variant
    molarMass = getMolarMass(substanceName)
    If molarMass = getMolarMassError Then _
        idealGasDensity = "ERROR: Molar mass not"_
      & " found for substance: " & substanceName
    Else
        idealGasDensity = molarMass * pressure _
            / (universalGasConstant * temperature)
    End If
End Function
```

Further Example Question

- Write the formula in cell B5 below to compute the density of nitrogen using the pressure and temperature shown.

	A	B
1		
2	Substance	Nitrogen
3	Pressure	101.325
4	Temperature	400
5	Density	=idealGasDensity(B2,B3,B4)

- May also have question about using worksheet with constant or range name

Example Answer as Sub

```
Const getMolarMassError as Double = -9999
Const universalGasConstant as Double = 8.3144621
Sub idealGasDensity()
    Dim substanceName As String
    Dim pressure As Double
    Dim temperature As Double
    Dim molarMass As Variant
    substanceName = Range("B2").Value
    pressure = Range("B3").Value
    temperature = Range("B4").Value
    molarMass = getMolarMass(substanceName)
    If molarMass = getMolarMassError Then _
        Range("B5").Value = "Masslookup error"
    Else
        Range("B5").Value = molarMass * pressure _
            / (universalGasConstant * temperature)
    End If
End Sub
```

Will need to use Range().Value on midterm

Declaring Variables/Consts

- Use the following syntax

```
Dim x As Double  'Distance
Dim k As Long
Dim s As String, d as Date
Const g As Double = 9.80665
```

- Do ***not*** use the following syntax

```
Dim x, y As Double
```

- Use `Option Explicit` to find errors in undeclared variables

Data Types

- Integer (-32,768 to 32,767) 4 bytes
- Long (-2,147,483,648 to 2,147,483,647) 8 bytes
- Single (-3.402623E38 to -1.401298E-45, 0, 1.491298E-45 to 3.402823E38) 4 bytes
- Double (-1.797…E308 to -4.94…E-324, 0, 4.940…E-324 to 1.797…E308) 8 bytes
- String (storage depends on length)

California State University Northridge 13

Variable Names

- Rules for variable names
  - Names are not case sensitive
    - The VB editor will maintain consistent capitalization in your variables, but if you type pressure it is the same as typing PRESSURE
  - The first character must be a letter or an underscore
  - After the first character you may use letters, numbers or underscores
  - Must be less than 255 characters
  - Cannot use VBA reserved words

California State University Northridge 14

Assignment Statements

- Assigns a value to a variable
  - Examples shown below

```
degreesToRadians = PI / 180
force = mass * accel
```

- What is really being done here?
  - Names refer to computer memory locations
  - The value of the expression on the right of the = sign is assigned to the memory location of the variable on the left
    - Can have statements like x = x + 1

California State University Northridge 15

String Variables

- String variables hold text information
- Declare string variables using Dim <variable> As String
  - E. g.: `Dim s As String, msg as String`
- String constants defined with quotation marks: “ME 209” is a string constant
  - String assignment: `s = “ME 209”`
- Concatenate (join) two strings with the concatenation (&) operator

```
msg = “This course is ” & s
```

California State University Northridge 16

Operators in Precedence Order

Parentheses (Evaluate me first)
Exponentiation ^
Unary minus – (*E.g.* –x)
Multiply/Divide * /
Integer Division \
Remainder Mod (*E.g.* 7 Mod 4 = 3)
Addition/Subtraction + –
String concatenation &
Relational < <= = >= > <>
Logical in following order: Not And Or

California State University Northridge 17

Writing Equations

- Important to remember operator precedence rules
  - Use parentheses to get desired result
  - Example: $z = \frac{p+qz}{s^2 r + \cos y}$
  - Write a statement to compute z

```
z = (p + q * z) / (s^2 * r + cos(y))
```

  - Write statement on one line (or continuation)
  - Use x*y to multiply x times y
  - Parentheses for multiplication/division

California State University Northridge 18

Relational/Logical Operators

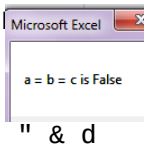
- Relational operators compare other variables and have true or false results
  - VBA operators are <, <=, =, >, >=, <>
  - Not equal <> is less than or greater than
- Logical operators (Not, And, Or) combine Boolean (true/false) results
  - **x And y** is true if both x and y are true
  - **x Or y** is true if either x or y is true
  - **Not x** is true if x is false; is false if x is true

California State University Northridge 19

Beware This Error

- What is the result of the following code

```
Sub Test()
    Dim a As Long, b As Long
    Dim c As Long, d As
    a = 0: b = 0: c = 0
    d = a = b = c
    MsgBox "a = b = c is " & d
End Sub
```



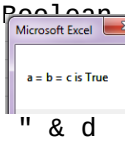
a = b is True; True becomes a one for final comparison 1 = c

California State University Northridge 20

Correct Way

- What is the result of the following code

```
Sub Test()
    Dim a As Long, b As Long
    Dim c As Long, d As Boolean
    a = 0: b = 0: c = 0
    d = a = b And a = c
    MsgBox "a = b = c is " & d
End Sub
```



a = b and a = c are both evaluated as True before the And

California State University Northridge 21

If – Else If

```
If <condition1> Then
   <Statements done if condition1 is true>
ElseIf <condition2> Then
   <Statements done if condition2 is true>
ElseIf <condition3> Then
   <Statements done if condition3 is true>
<May be other conditions>
Else
   <Statements done if all conditions false>
End If
<Execute here after any statements done>
```

A <condition> is TRUE or FALSE

May have zero or many ElseIf's

Else statement is optional

California State University Northridge 22

If – ElseIf Explained

- If any condition is true, the statements following the If or Else If are executed
- Once any statements are executed transfer to the first statement after the End If
- Statements for only the first true condition are executed
- The Else block is optional
  - If no conditions are true those statements are executed

```
If <condition1> The
   <Statements don
Else If <condition2>
   <Statements don
Else If <condition3>
   <Statements don
<May be other cond
Else
   <Statements don
End If
<Execute here after
```

California State University Northridge 23

One Line If Example

```
'Compute base pay for all hours
'then add overtime if
'applicable
pay = hours * rate
If hours > 40 Then pay = pay _
    + 0.5 * rate * (hours - 40)
MsgBox "Pay = " & pay
```

California State University Northridge 24

If Without Else Example

```
'Compute base pay for all hours
pay = hours * rate
If hours > 40 Then
'Add overtime pay
    pay = pay + 0.5 * rate _
              * (hours - 40)
End If
MsgBox "Pay = " & pay
```


If-Else Example

- Test for complex roots in solution quadratic equation $ax^2 + bx + c = 0$
  - Complex if $b^2 - 4 * a * c < 0$, real otherwise

```
If b^2 - 4 * a * c < 0 Then
    MsgBox "Complex Roots"
Else
    MsgBox "Real Roots"
End If
```


If – ElseIf Example

```
Function test(x As Double) As String
    If x < 0 Then
        test = "Negative"
    ElseIf x > 0 Then
        test = "Positive"
    Else
        test = "Zero"
    End If
    MsgBox "x is " & test
End Function
```

Can use "Else" instead of "ElseIf x = 0" because we know that x = 0 if we get to Else

Note use of string constants, string variables, and & operator

Boolean Variables

- Boolean variables have two possible values: TRUE or FALSE
- Can set Boolean variables equal to a relational or logical expression
- Declare variables as Boolean

```
Dim converged As Boolean
```

- Example of use

```
converged = diff < absErr Or _
    diff < relErr * Abs(reference)
```


Looping

- Looping is code that repeats the same operations with different data
- There are four kinds of loops in VBA
  - Count-controlled loops (For Loop)
  - Conditional Loop (Do While Loop)
  - Another conditional Loop (Loop Until)
  - Infinite Loop (Do … Loop; requires Exit)
- Choice between these is often a user preference

The Do While Loop

- The Do While <condition> loop executes as long as the <condition> is true
- The Do While x < 10 ends only when x ≥ 10 at the start of the loop
- In the operation of the loop, a new value of x is tested before the loop starts
- Once the loop starts all statements before the "Loop" statement are completed until a new test is made unless there is a conditional exit

Loop Until Command

- An alternative to the Do While … Loop
  - Starts with the command Do
  - Followed by loop body with all code executed in loop
  - Concludes with Loop Until <condition>
- Loop continues as long as <condition> is false when tested at the end of the loop
- Loop is exited when <condition> is true
- Loop Until condition opposite of Do While

California State University Northridge 31

Count-controlled Loops

- Basic structure of VBA For loop

```
For <counter> = <start> To <finish>
   <statements>
Next <counter>
```

- <counter> is a long/integer variable that increases by one each time through loop
- <start> is the initial value of <counter>
- <last> is the final value of <counter>
- <statements> executed repeatedly may use value of <counter>

California State University Northridge 32

Count-controlled Loops II

- Modified structure of VBA For loop

```
For <counter> = <start> To _
        <finish> Step <increment>
   <statements>
Next <counter>
```

- Here <counter> is changed by <increment> each time through loop
- Loop is executed nTimes times where nTimes = Int((<finish> – <start>) / <increment>) + 1
  - Not executed if nTimes ≤ 0
  - Executed once if <start> = <finish>

California State University Northridge 33

Square Root Trial and Error

- Find a, the square root of A, by the following iteration equation
  - $a_{k+1} = \frac{a_k}{2} + \frac{A}{2a_k}$ a_0 is initial guess; a_1, a_2, a_3, etc. are subsequent values
  - Do we believe this works? Look at an example of finding $\sqrt{9}$ with initial guess of 1
    - Here A = 9 and a_0 = 1
    - $a_1 = \frac{a_0}{2} + \frac{A}{2a_0} = \frac{1}{2} + \frac{9}{2(1)} = 5$
    - $a_2 = \frac{a_1}{2} + \frac{A}{2a_1} = \frac{5}{2} + \frac{9}{2(5)} = \frac{25+9}{10} = \frac{34}{10}$

California State University Northridge 34

Do While Example

```
Function mySqrt(A As Double) As Variant
   Const maxRelErr = 1E-12 : maxIterations = 100
   'Dim statements omitted

    mySqrt = 1 : iterations = 0 : converged = False
    Do While Not converged And _
                    iterations < maxIterations
        mySqrt = mySqrt / 2 + A / (2 * mySqrt)
        iterations = iterations + 1
        converged = Abs(mySqrt ^ 2 - A) < _
                              maxRelErr * Abs(A)
    Loop
    If Not converged Then mySqrt = _
                                   "No Convergence"
End Function
```

$a_{k+1} = \frac{a_k}{2} + \frac{A}{2a_k}$

California State University Northridge 35

Same Example With Loop Until

```
Function mySqrt(A As Double) As Variant
   Const maxRelErr = 1E-12 : maxIterations = 100
   'Dim statements omitted

  iterations = 0 : mySqrt = 1
  Do
    mySqrt = mySqrt / 2 + A / (2 * mySqrt)
    iterations = iterations + 1
    converged = Abs(mySqrt ^ 2 - A) < _
                              maxRelErr * Abs(A)
  Loop Until converged Or iterations >= _
                                   maxIterations
  If Not converged Then mySqrt ="No Convergence"
End Function
```

Opposite of Do While condition

California State University Northridge 36

Same Loop Until As a Sub

```
Sub mySqrtsub() As Variant
    Dim mySqrt As Double, A as Double
    A = Range("A1").Value
      'Const and other Dim statements omitted
  iterations = 0 : mySqrt = 1
  Do
    mySqrt = mySqrt / 2 + A / (2 * mySqrt)
    iterations = iterations + 1
    converged = Abs(mySqrt ^ 2 - A) < _
                              maxRelErr * Abs(A)
  Loop Until converged Or iterations >= _
                                 maxIterations
  If Not converged Then mySqrt ="No Convergence"
  Range("A2").Value = mySqrt
End Function
```

Opposite of Do While condition

Northridge

Same Example With For Loop

```
Function mySqrt2(A As Double) As Variant
  Const maxRelErr = 1E-12 : maxIterations = 100
  'Dim statements omitted


  mySqrt2 = 1  'No other initializations
  For iteration = 1 To maxIterations
    mySqrt2 = mySqrt2 / 2 + A / (2 * mySqrt)
    If Abs(mySqrt2 ^ 2 - A) < maxRelErr * _
                          Abs(A) Then Exit Function
  Next iteration
     'maxIterations exceeded if we get here
  mySqrt2 ="No Convergence"
End Function
```

Northridge 38

Tracing Loops

```
x = 1 : term = x : sum = term
For n = 1 to 10
      term = term * x / n
      sum = sum + term
Next k
```

Exercise: What is sum for n = 6?

Initialization: x = 1 term = x = 1 sum = term = 1
n=1: term = 1 * 1 / 1 = 1; sum = 1 + 1 = 2
n=2: term = 1 * 1 / 2 = 1/2; sum = 2 + 1/2 = 5/2
n=3: term = 1/2 * 1 / 3 = 1/6; sum = 5/2 + 1/6 = 16/6
n=4: term: 1/6 * 1 / 4 = 1/24; sum: 16/6 +1/24 = 65/24
n=5: term :1/24*1/5=1/120;sum: 65/24+1/120=163/60

Northridge 39

What will be on the midterm?

- Similar to quizzes / VBA statement list
- Interpreting equations in Excel cells
- Writing VBA code for mathematical expressions
- Writing and tracing code that uses If-Then-ElseIf statements
  - Use of MsgBox, Range.Value, Cells.Value
- Tracing code that uses all statements covered in class, including loops

Northridge 40

Sample Question

- What is the output from this VBA sub if user tries to enter numbers 3, 6, -12, 4?

```
Sub midterm()
   Dim n as long, sum as long
   sum = 0
   n =  InputBox ("Enter an integer:")
   Do While n > 0
     sum = sum + n
     n =  InputBox ("Enter an integer:")
   Loop
   Msgbox("The sum is :" & sum)
End Sub
```

9

Northridge 41

Another Sample Question

- What is the output from this VBA sub if user tries to enter numbers 3, 6, -12, 4?

```
Sub midterm()
   Dim n as long, sum as long
   sum = 0
   Do
     n =  InputBox ("Enter an integer:")
     sum = sum + n
   Loop Until n <= 0
   Msgbox("The sum is :" & sum)
End Sub
```

-3

Northridge 42