Using Tools and Writing Code for Roots of Equations

Larry Caretto
Mechanical Engineering 309

Numerical Analysis of Engineering Systems

February 12, 2014

California State University Northridge

Outline

- MATLAB fzero and Excel Goal Seek
- Root finder codes for various methods
  - Pass function names for f and f' so there is no need to change code for new problems
    - Workaround required for VBA
- Function code in MATLAB for passing function name into root-finder code
- Use of variables for current iteration instead of arrays for x_i iterations
- Model code for root finders

MATLAB Function Handles

- We have seen that we can define functions in MATLAB in a fname.m file
- We can also pass the name of a function into another function
  - For example to solve for roots of f(x) = 0 we can pass a **function handle** for f(x) defined as: fNameHandle = @fName
  - We can also define anonymous functions at the command line >>fH=@(x) x^2 + 2
    - Does not use an equal sign to define function

MATLAB fzero

- MATLAB command to solve f(x) = 0
  - >> x = fzero(fun, x0) tries to find root of function with handler, fun, near x0
  - fun is function handle for a MATLAB function
    - For a function myFunction.m you can enter @myFunction for the first argument, fun
    - Anonymous functions: >> fH=@(x) x^3 – sin(x) can be used as >> x = fzero(fH, x0)
  - x0 may be a scalar (or 2D array that must bracket root)
  - >> help fzero for more information about fzero, options parameters, and return values

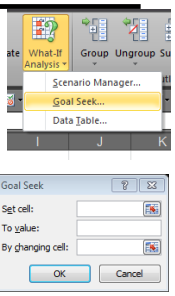
EXCEL Goal Seek

- For worksheet with several calculations, what is the value of an input cell that will set a specified results cell to a certain value?
- *E.g.* what compression ratio gives 55% efficiency?

	M	N	O	P
1		Diesel Cycle		
2		inletP	100	kPa
3		inletT	280	K
4		gasConstant	0.287	kJ/kg-K
5		compressionRatio	10	
6		heatInput	800	kJ/kg
7		bottomDeadCenterV	0.8036	m^3/kg
8		topDeadCenterV	0.08036	m^3/kg
9		initialU	199.74	kJ/kg
10		compressionEndT	686.18	K
11		compressionEndP	2450.65	kPa
12		compressionEndH	698.12	kJ/kg
13		heatInputEndH	1498.12	kJ/kg
14		heatInputEndT	1387.18	K
15		heatInputEndV	0.16245	m^3/kg
16		expansionEndT	814.92	K
17		expansionEndP	291.04	kPa
18		expansionEndU	603.96	kJ/kg
19		heatRejection	404.22	kJ/kg
20		netWork	395.78	kJ/kg
21		efficiency	49.5%	
22		meanEffectivePressure	647.23	kPa

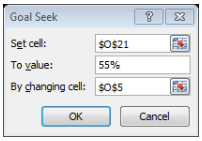
Using Goal Seek

- Start Goal Seek from What-If Analysis on Data tab
  - In resulting Goal Seek dialog
    - Specify "Set cell:" box as cell to be assigned a desired value
    - Enter desired value in "To value:" box
    - Specify cell to be changed in "By changing cell:" box
    - Click OK to execute
- See example on next chart



EXCEL Goal Seek Example

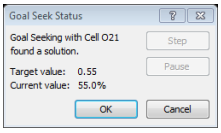
- Find compression ratio to give 55% efficiency
- Start Goal Seek and enter values below



	M	N	O	P
1		Diesel Cycle		
2		inletP	100	kPa
3		inletT	280	K
4		gasConstant	0.287	kJ/kg-K
5		compressionRatio	10	
6		heatInput	800	kJ/kg
7		bottomDeadCenterV	0.8036	m^3/kg
8		topDeadCenterV	0.08036	m^3/kg
9		initialU	199.74	kJ/kg
10		compressionEndT	686.18	K
11		compressionEndP	2450.65	kPa
12		compressionEndH	698.12	kJ/kg
13		heatInputEndH	1498.12	kJ/kg
14		heatInputEndT	1387.18	K
15		heatInputEndV	0.16245	m^3/kg
16		expansionEndT	814.92	K
17		expansionEndP	291.04	kPa
18		expansionEndU	603.96	kJ/kg
19		heatRejection	404.22	kJ/kg
20		netWork	395.78	kJ/kg
21		efficiency	49.5%	
22		meanEffectivePressure	547.23	kPa

California State University Northridge

Goal Seek Result



- Can accept or reject new results
- Compare with previous slide to show changes in intermediate results

	M	N	O	P
1		Diesel Cycle		
2		inletP	100	kPa
3		inletT	280	K
4		gasConstant	0.287	kJ/kg-K
5		compressionRatio	13.40157	
6		heatInput	800	kJ/kg
7		bottomDeadCenterV	0.8036	m^3/kg
8		topDeadCenterV	0.05996	m^3/kg
9		initialU	199.74	kJ/kg
10		compressionEndT	762.98	K
11		compressionEndP	3651.85	kPa
12		compressionEndH	781.01	kJ/kg
13		heatInputEndH	1581.01	kJ/kg
14		heatInputEndT	1456.25	K
15		heatInputEndV	0.11445	m^3/kg
16		expansionEndT	760.12	K
17		expansionEndP	271.47	kPa
18		expansionEndU	559.74	kJ/kg
19		heatRejection	360.00	kJ/kg
20		netWork	440.00	kJ/kg
21		efficiency	55.0%	
22		meanEffectivePressure	591.69	kPa

California State University Northridge

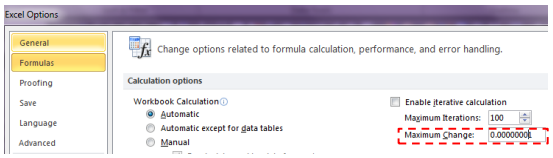
Before and After

Changes in all Results

	M	N	O	P	M	N	O	P
1		Diesel Cycle				Diesel Cycle		
2		inletP	100	kPa		inletP	100	kPa
3		inletT	280	K		inletT	280	K
4		gasConstant	0.287	kJ/kg-K		gasConstant	0.287	kJ/kg-K
5		compressionRatio	10			compressionRatio	13.40157	
6		heatInput	800	kJ/kg		heatInput	800	kJ/kg
7		bottomDeadCenterV	0.8036	m^3/kg		bottomDeadCenterV	0.8036	m^3/kg
8		topDeadCenterV	0.08036	m^3/kg		topDeadCenterV	0.05996	m^3/kg
9		initialU	199.74	kJ/kg		initialU	199.74	kJ/kg
10		compressionEndT	686.18	K		compressionEndT	762.98	K
11		compressionEndP	2450.65	kPa		compressionEndP	3651.85	kPa
12		compressionEndH	698.12	kJ/kg		compressionEndH	781.01	kJ/kg
13		heatInputEndH	1498.12	kJ/kg		heatInputEndH	1581.01	kJ/kg
14		heatInputEndT	1387.18	K		heatInputEndT	1456.25	K
15		heatInputEndV	0.16245	m^3/kg		heatInputEndV	0.11445	m^3/kg
16		expansionEndT	814.92	K		expansionEndT	760.12	K
17		expansionEndP	291.04	kPa		expansionEndP	271.47	kPa
18		expansionEndU	603.96	kJ/kg		expansionEndU	559.74	kJ/kg
19		heatRejection	404.22	kJ/kg		heatRejection	360.00	kJ/kg
20		netWork	395.78	kJ/kg		netWork	440.00	kJ/kg
21		efficiency	49.5%			efficiency	55.0%	
22		meanEffectivePressure	547.23	kPa		meanEffectivePressure	591.69	kPa

Setting Goal Seek Accuracy

- Go to File Tab | Options | Formulas
- Set small value for Maximum Change:



- Do not enable iterative calculations
  - This is a useful tool, but it stoops flagging circular reference errors

California State University Northridge

Root Finder Code: Basic Ideas

- Code for root-finder function is written with call to function for equation, f(x)
  - Allows root-finder code to be applied to any function without changes
  - User must write function for equation to be solved, *i.e.* what we call f(x)
    - Specific function for user equation will usually have another name, *e.g.* axMinusSinX
  - To use Newton's method the root solver code will call a function for fPrime
    - Name will be like axMinusSinXPrime

California State University Northridge

MATLAB Root Finder Example

- User functions (in two separate files, axMinusSinx.m, axMinusSinXPrime.m)

```
function f = axMinusSinX( x )
    f = 0.05x - sin(x)
end

function fPrime = axMinusSinXPrime( x )
    fPrime = 0.05 - cos(x)
end
```

California State University Northridge

MATLAB Root Finder Example

- Root-finder function header

```
Function x = findRoot(f, fPrime, …
initialGuess)
```

- Look at Newton's Method example

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

- Reference to function in Newton's Method root-finder

```
xNew = xOld – f(xOld)/fPrime(xOld)
```


MATLAB Root-Finder Call

- From the command line enter the following

```
>> x = rootFinder ( @axMinusSinX, …
@axMinusSinXPrime, initialGuess)
```

- This call will make root-finder use axMinusSinX for f and axMinusSinXPrime for fPrime
  - The @ sign creates a "function handle" which is necessary for passing a function name (instead of the usual variable) to a function in MATLAB.

Root-finder Code

- Typically do **not** use arrays for x_i
- Use variables for recent iterations only
  - *E.g.* x1, x2, x3 or xOlder, xNewer, xNext
  - Secant method example x1 = x_{k-1}, x2 = x_k, and x3 = x_{k+1} and f1 = f(x1), f2 = f(x2), *etc*.
  - Computing $x_{k+1} = x_k - f_k(x_k - x_{k-1})/(f_k - f_{k-1})$
  - x3 = x2 – f2 * (x2 – x1) / (f2 – f1)
  - Before next iteration do the following in the order shown
    - x1 = x2 : x2 = x3 : f1 = f2 : f2 = f(x2)

What does this do?

Previous Iteration	$x_{i+1} = x_{15}$	$x_i = x_{14}$	$x_{i-1} = x_{13}$
Program Variables	X3 →	X2 →	X1
Next Iteration	$x_{i+1} = x_{16}$	$x_i = x_{15}$	$x_{i-1} = x_{14}$

- Note that setting X1 = X2 for the next iteration moves x_{14} from X2 to X1
- Similarly setting X2 = X3 moves x_{15} from X3 to X2 for the next iteration

Passing Function Names

- Many programming languages allow the name of a function to be passed as a parameter to another procedure
  - MATLAB function handles allow this
- For numerical analysis this allows a complete separation of the general code for the algorithm from the equations for the specific problem definition
  - For root solver want code to compute f(x) to be separate from solution method code

VBA Passing Function Names

- Does not exist but have a workaround that requires four steps
  1. Write the usual function to compute f(x) for the function you are trying to solve: f(x) = 0
  2. Use a string variable to represent your "f" function in the root-solver function header
  3. Call the root-solver function with the function name expressed as a string
  4. Use a statement like the following to compute f(x) in the root-solver function

```
f1 = Application.Run(f, x1)
```

VBA Passing Function Names II

1. Example: function to solve $e^{ax}\sin(bx) = .5$

```
Function eaxsinbx(x As Double) As Double
   Const a As Double = 0.02
   Const b As Double = 0.15
   eaxsinbx = exp(a*x) * sin(b*x) - 0.5
End Function
```

2. Example of root-solver function header

```
Function myRootSolver(f As String, _
x1 As Double, x2 as Double) As Double
```

– Here f represents function name and x1 and x2 the initial guesses passed into the solver

VBA Passing Function Names III

3. Example of root-solver call with initial guesses of 2 and 4

```
X = myRootSolver("eaxsinbx", 2, 4)
```

– Recall function header:

```
myRootSolver (f As String, _
      x1 As Double, x2 as Double
```

4. Use statements like the following in root solver to evaluate f(x) for any x value

```
f1 = Application.Run(f, x1)
```



```
Function secant(x1 As Double, x2 As Double, f As _
     String, Optional maxIter As Integer = 100, _
Optional maxRelErr As Double = 1E-10) As Variant
    Dim x3 As Double, f1 As Double, f2 As Double
    Dim iteration As Integer

    f1 = Application.Run(f, x1)   'Computes f(x1)
    f2 = Application.Run(f, x2)   'Computes f(x2)
    For iteration = 1 To maxIter
        x3 = x2 - f2 * (x2 - x1) / (f2 - f1)
        If Abs(x3 - x2) <= maxRelErr * Abs(x3) Then
            secant = x3: Exit Function
        Else
            x1 = x2: x2 = x3: f1 = f2
            f2 = Application.Run(f, x2)
        End If
    Next iteration
    secant = "No Convergence in Secant Routine"
End Function
```

Use: =secant (0, 2,"myF")

Write VBA code for Function _ myF(x As Double) as Double

Root-finder pseudo Code

```
Function rootFinder( f, fp, x1, x2)
maxIter = 100  :  maxRelErr = 1e-12
f1 = f(x1)   :   f2 = f(x2)
For iterations = 1 to maxIter
     x3 = new iteration from method
     f3 = f(x3)
     if abs(x3 – x2) <= …
                 maxRelErr * abs(x3)
          rootFinder = x3
          exit Function
     end if
     x1 = x2 : x2 = x3  :  f1 = f2
```

Initial guesses

Root-finder pseudo Code

```
For iteration = 1 to maxIter
     x3 = new iteration from method
     f3 = f(x3)
     if abs(x3 – x2) <= …
                 maxRelErr * abs(x3)
          rootFinder = x3
          exit Function
     end if
     x1 = x2 : x2 = x3 : f1 = f2
     f2 = f(x2)
Next iteration
rootFinder = -9999   !error code
```


Interpreting pseudo code

- Iteration formulas and updates depend on method
- Number and type of initial guesses depend on method
  - Some require two guesses to bracket root
  - Newton method requires one
  - Secant method can take one initial guess, x_{Guess}; user can create $x1 = x_{Guess}(1 - \alpha)$ and $x2 = x_{Guess}(1 + \alpha)$ [See note below]
  - Use α between 0.05 and 0.25

If $x_{Guess} = 0$, $x1 = 1 - \alpha$; $x2 = 1 + \alpha$

Interpreting pseudo code II

- For MATLAB
  - Loop: `for iterations = 1:maxIter`
  - All keywords are lower case
  - Use "end" at end for loops and ifs
  - Use return to exit function
- For VBA
  - Need workaround to pass functions through header
  - Use variant type function to return text error

California State University Northridge 25

MATLAB Function Returns

- MATLAB functions can return more than one variable
  - Could use this to get both f and f'

```
function[f fp] = x3_SinXNt( x )
    f = x^3 - sin(x)
    fp = 2*x^2 - cos(x)
```

Call root finder as follows (initial guess 1)
x = newt(@x3_sinxNt, 1)

- Root-finder function header

```
x = newt(fct, initialGuess)
```

- Use in root finder: `[f fp] = fct(xOld)`

```
xNew = xOld - f/fp
```

California State University Northridge 26

Program Assignment Three

- Submission files
  - Word file with discussion and MATLAB operations and results
  - Excel file with Excel and VBA results
- Find all roots of $f(x) = 0.0005x^4 = \cos(x) = 0$ between $-2 \le x \le 8$ (written as [-2 8])
  - Use MATLAB fzero (copy command window to Word document)
  - Use goal seek method of Excel
  - Write MATLAB and VBA code

California State University Northridge 27

Exercise for Tonight

- Sketch $f(x) = 0.005x^4 - \cos(x)$ and determine approximate location of roots
- Use MATLAB fzero to find roots in [-2 8]
  - Copy commands and results to Word file
- Use Excel Goal Seek to all roots in [-2 8]
  - Goal seek will give answer in same cell as initial guess
  - Use following structure

This cell starts with initial guess

Counter	Initial Guess	Answer	f(x)

This cell records the initial guess

California State University Northridge 28

Required Output

- What output is required for second assignment, exercise one, question number 8?
- You should provide the following:
  - A listing of your function (can use Type)
  - The calls to your function with the input values clearly specified
  - The resulting plots
  - Use comments as appropriate
    - Start MATLAB comments with a %

California State University Northridge 29