Using Tools and Writing Code for Roots of Equations

Larry Caretto
Mechanical Engineering 309
Numerical Analysis of Engineering Systems
February 12, 2014

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: \( \text{fNameHandle} = @ \text{fName} \)
      - We can also define anonymous functions at the command line
        \( \text{fH}(x) = x^2 + 2 \)
        - Does not use an equal sign to define function

MATLAB fzero
- MATLAB command to solve \( f(x) = 0 \)
  - \( x = \text{fzero}(\text{fun}, x0) \) tries to find root of function with handler, fun, near \( x_0 \)
  - \( \text{fun} \) is function handle for a MATLAB function
  - For a function \( \text{myFunction.m} \) you can enter \( @\text{myFunction} \) for the first argument, \( \text{fun} \)
  - Anonymous functions: \( \text{>> fH}(x) = x^3 - \sin(x) \) can be used as \( \text{>> x = fzero(fH, x0)} \)
  - \( x_0 \) may be a scalar (or 2D array that must bracket root)
  - \( \text{>> help fzero} \) for more information about \( \text{fzero} \), 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?

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

<table>
<thead>
<tr>
<th>Goal Seek</th>
<th>Y Value</th>
<th>55%</th>
</tr>
</thead>
<tbody>
<tr>
<td>From cell</td>
<td>OK</td>
<td>Cancel</td>
</tr>
</tbody>
</table>

Goal Seek Result

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

Before and After

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

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

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
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
  \[ x_{\text{new}} = x_{\text{old}} - f(x_{\text{old}})/f'(x_{\text{old}}) \]

MATLAB Root-Finder Call

- From the command line enter the following
  \[ >> x = \text{rootFinder}( \@axMinusSinX, ... @axMinusSinXPrime, \text{initialGuess}) \]
- 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. \( x_1, x_2, x_3 \) or \( x_{\text{Older}}, x_{\text{Newer}}, x_{\text{Next}} \)
  - Secant method example \( x_1 = x_{k-1}, x_2 = x_k, \) and \( x_3 = x_{k+1} \) and \( f_1 = f(x_1), f_2 = f(x_2), \) etc.
- Computing \( x_{k+1} = x_k - f_k(x_k - x_{k-1})/(f_k - f_{k-1}) \)
- \( x_3 = x_2 - f_2 \cdot (x_2 - x_1) / (f_2 - f_1) \)
- Before next iteration do the following in the order shown
  1. \( x_1 = x_2 : x_2 = x_3 : x_1 = f_2 : f_2 = f(x_2) \)

What does this do?

<table>
<thead>
<tr>
<th>Previous Iteration</th>
<th>( x_{i+1} = x_{15} )</th>
<th>( x_i = x_{14} )</th>
<th>( x_{i-1} = x_{13} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Variables</td>
<td>( X3 )</td>
<td>( X2 )</td>
<td>( X1 )</td>
</tr>
<tr>
<td>Next Iteration</td>
<td>( x_{i+1} = x_{16} )</td>
<td>( x_i = x_{15} )</td>
<td>( x_{i-1} = x_{14} )</td>
</tr>
</tbody>
</table>

- 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
     \[ f1 = \text{Application.Run}(f, x1) \]
VBA Passing Function Names II

1. Example: function to solve $e^{ax}\sin(bx) = .5$
   
   ```vba
   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
   
   ```vba
   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
   
   ```vba
   X = myRootSolver("eaxsinbx", 2, 4)
   ```

   - Recall function header:
     ```vba
     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
   
   ```vba
   f1 = Application.Run(f, x1)   'Computes $f(x_1)$
   f2 = Application.Run(f, x2)   'Computes $f(x_2)$
   ```

Root-finder pseudo Code

```vba
Function rootFinder( f, fp, x1, x2)  
maxIter = 100  :  maxRelErr = 1e-12  
f1 = f(x1) :  f2 = f(x2)  Initial  
For iterations = 1 to maxIter guesses  
x3 = new iteration from method  
f3 = f(x3)  
if abs(x3 - x2) <= ...  
maxRelErr * abs(x3) Then  
rootFinder = x3  
exit Function  
end if  
x1 = x2  :  x2 = x3 :  f1 = f2  
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
  - Use $\alpha$ between 0.05 and 0.25

```vba
If xGuess = 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

MATLAB Function Returns

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

```matlab
function [f fp] = x3_sinXnt( x )
    f = x^3 - sin(x)
    fp = 2*x^2 - cos(x)
end
```

• Root-finder function header

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

• Use in root finder:

```matlab
[f fp] = fct(xOld)

xNew = xOld - f/fp
```

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 \(-2 \leq x \leq 8\)
  – Use MATLAB fzero (copy command window to Word document)
  – Use goal seek method of Excel
  – Write MATLAB and VBA code

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

```
<table>
<thead>
<tr>
<th>Counter</th>
<th>Initial Guess</th>
<th>Answer</th>
<th>f(x)</th>
</tr>
</thead>
</table>
```

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 %