California State University Northridge
College of Engineering and Computer Science
Mechanical Engineering Department

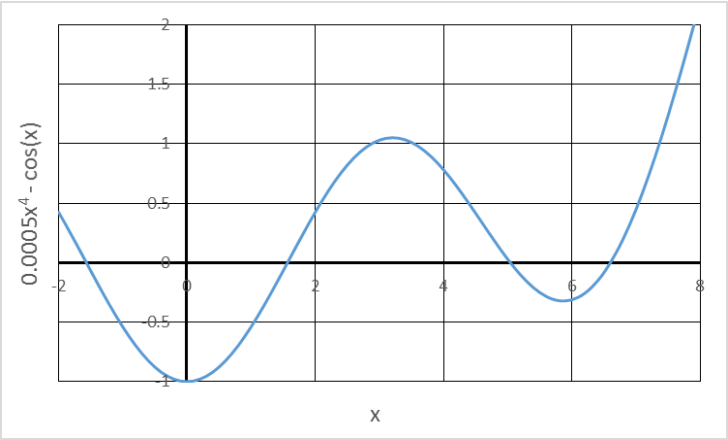
Mechanical Engineering 309
Numerical Analysis of Engineering Systems

Fall 2012 Number: 14835 Instructor: Larry Caretto

Solutions to Third Programming Assignment

All the assignments below should be tested on the equation $0.0005x^4 = \cos(x)$. Graph this equation to determine the number of roots it has in the range $-2 \le x \le 8$ and their general location.

The Excel plot of the equation $f(x) = 0.0005x^4 - \cos(x)$, at the left, shows that there are four zeroes between -2 and 8. These are located near x = -2, x = 2, x = 5 and x = 7. These values will be used for the initial guesses. For methods that require root bracketing the initial guesses will be [xPlus, xMinus] = [-2,1], [1, 2], [4, 6], and [7, 6].



1. **Use the fzero command of MATLAB with different initial guesses to find all the roots of $0.0005x^4 = \cos(x)$.**

To use this command we must first write a function to compute the equation in the form f(x) = 0. Here we choose $f(x) = 0.0005x^4 - \cos(x)$. The MATLAB function for this f(x), named ax4_cosx, is shown below.

```
function f = ax4_cosx( x )
% ax4_cosx computes a*x^4 - cos(x)
%  Used as input to root solvers to find zeros of this function
    a = 0.0005;
    f = a*x^4 - cos(x);  %Note the semicolons to avoid printout
end
```

Once we have written this function, we can use it in the `fzero` command with different initial guesses. These commands and their results are shown below. Note the ampersand (@) in front of the function name when the function name is passed as a MATLAB function handle to the MATLAB `fzero` function. In the command sequence below the accuracy of the solution is checked by evaluating the function at the answer provided by fzero.

```
>> format longe
>> fzero(@ax4_cosx,-2)
ans = -1.567775635746529e+00
>> ax4_cosx(ans)
ans = -3.989863994746656e-17
>> fzero(@ax4_cosx,2)
ans = 1.567775635746529e+00
>> ax4_cosx(ans)
ans =-3.989863994746656e-17
>> fzero(@ax4_cosx,5)
ans = 5.041214539367116e+00
>> ax4_cosx(ans)
ans =-2.775557561562891e-16
>> fzero(@ax4_cosx,7)
ans = 6.601685296468693e+00
>> ax4_cosx(ans)
```

Jacaranda Hall 3314 Mail Code Phone: N/A
Email: lcaretto@csun.edu 8348 Fax: 818.677.7062

```
ans = 4.440892098500626e-16
```

2. Write a MATLAB routine that uses the secant method and apply it with different initial guesses to find all the roots of $0.0005x^4 = \sin(x)$. Your method should be able to accept a function name in the same way that `fzero` does.

The function is shown below.

```
function xNew = falsePosition(f, xPlus, xMinus)
%falsePosition -- uses the false position method to find a root of f(x) = 0
%User input: f is the name of the function whose root you want
%            xPlus  is one initial guess for the root where f(x) > 0
%            xMinus is another initial guess for the root where f(x) < 0
    maxIterations = 100;
    maxRelErr = 1e-14;
    fPlus = f(xPlus);
    fMinus = f(xMinus);
    xNew = xPlus;             %arbitrary choice for initial error check
    for iterations = 1:maxIterations
        xNewOld = xNew;
        xNew = xMinus - fMinus * (xPlus - xMinus) / (fPlus - fMinus);
        if abs(xNewOld - xNew) <= maxRelErr * abs(xNew);
            return    %returns value x defined two lines above
        end
        fNew = f(xNew);
        if fNew > 0
            xPlus = xNew;
            fPlus = fNew;
        else
            xMinus = xNew;
            fMinus = fNew;
        end
    end
    errorMessage = 'No convergence in false position routine'
end
```

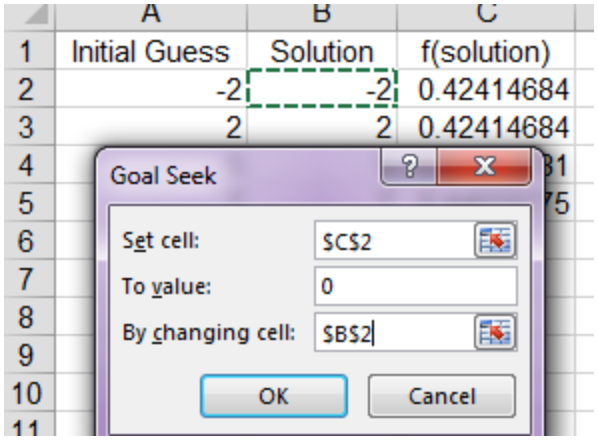
The calls to this function and the results are shown below. Again, the solution is checked by evaluating the function at the solution provided by the ans variable. Note the similarity to the results for task 1.

```
>> falsePosition(@ax4_cosx,-2,-1)
ans = -1.567775635746529e+00
>> ax4_cosx(ans)
ans = -3.989863994746656e-17
>> falsePosition(@ax4_cosx,1,2)
ans = 1.567775635746529e+00
>> ax4_cosx(ans)
ans = -3.989863994746656e-17
>> falsePosition(@ax4_cosx,4,6)
ans = 5.041214539367116e+00
>> ax4_cosx(ans)
ans = -2.775557561562891e-16
>> falsePosition(@ax4_cosx,7,6)
ans = 6.601685296468688e+00
>> ax4_cosx(ans)
ans =    -4.440892098500626e-15
```

3. Use the goal seek method of the Excel worksheet with different initial guesses to find all the roots of $0.0005x^4 = \cos(x)$.

The worksheet used for this calculation is shown at the right in equation view. Prior to the solution, the different initial guesses are entered in columns A and B.

	A	B	C
1	Initial Guess	Solution	f(solution)
2	-2	-2	=0.0005*B2^4-COS(B2)
3	2	2	=0.0005*B3^4-COS(B3)
4	5	5	=0.0005*B4^4-COS(B4)
5	7	7	=0.0005*B5^4-COS(B5)



The goal seek method will change the value in column B and the value in column A will remain as a record of the initial guess.

The Goal Seek dialog is selected on the **Data** Tab from the **What-If Analysis** icon in the **Data Tools Group**. As shown at the left, the Goal Seek dialog is set up to set the f(x) cell in C2 to zero by changing the value in cell B2. The current value in cell B2, -2, is the initial guess for the iteration process. After clicking OK a final dialog (not shown here) allows the user to accept or reject the proposed solution. (Goal Seek may not find the root; in this case the user is notified that no solution is found. For this exercise, Goal Seek is repeated for each different root sought. The final results for all four roots are shown in the screen view at the right. Before using goal seek the error

	A	B	C
1	Initial Guess	Solution	f(solution)
2	-2	-1.567776	-1.383E-15
3	2	1.567776	-1.383E-15
4	5	5.041215	-1.055E-11
5	7	6.601685	1.2212E-15

tolerance was adjusted by the menu sequence **File → Options → Formulas** , and in the **Calculation Options** section of the resulting dialog the **Maximum Change** was set to 10^{-10} to improve the accuracy of the Goal Seek results.

4. Write a VBA routine that uses Newton's method to find all the roots of $0.0005x^4 = \cos(x)$ in the range $-2 \le x \le 8$. This should be written as a user-defined function that takes an initial guess from the worksheet as its argument and returns the solution to the worksheet. Your root finder should require you to write two simple, one-line functions to compute f(x) and df/ dx. Your Newton's method code should then accept a string name for both the function that computes f and the function that computed df/dx; it should the Application.Run method to evaluate arbitrary functions that are input as strings to the root solver as discussed in class. This will allow you to use your root solver for any function without changes.

The VBA code is shown below; although not required for this assignment, the function header shows how optional parameters are used for the maximum iterations and the maximum relative error. If the user does not enter a value for these parameters, the ones shown in the function header are used. The user can choose to enter different values for either or both of these parameters. The VBA code below includes the code for Newton's method and the code for the two functions required to compute the function and its derivative. The Newton's method code could be used without changes to solve any other problem if the necessary functions to compute f(x) and f'(x), the equation to be solved and its first derivative, were available.

```
Option Explicit
Const a As Double = 0.0005  'a set here to ensure consistency in f and f' functions

Function newton(x As Double, f As String, fPrime As String, _
```

```
            Optional maxRelErr As Double = 0.0000000001, _
            Optional maxIterations As Long = 100) As Variant

    'Uses the Newton method to find the root of an equation in the form f(x) = 0
    'User must write two functions that are called by this routine
    '    Function fName(x As Double) As Double computes f(x), the equation to be solved
    '    Function fNamePrime(x As Double) As Double computes f'(x),
    '                                the derivative of the equation to be solved
    'Any name may be used for these functions; the specific names are passed to the
    'newton function as strings as described below

    'Inputs to the program
    '   x is the initial guess for x
    '   f is a string giving the name of the user's fName function as "fName"
    '   fPrime is a string giving the derivative function name as "fNamePrime"
    '   maxRelErr (optional) is the maximum allowed relative error in the solution
    '   maxIterations (optional) is the maximum iterations; after these iterations,
    '                   the program assumes that it is in an infinite loop and exits

    'If the solution converges, the function returns the x value that sets f(x) = 0
    'If the solution does not converge, the function returns "No convergence"

    Dim oldX As Double  'old value of x in iterations
    Dim k As Long       'iteration counter

    For k = 1 To maxIterations
        oldX = x
        x = x - Application.Run(f, x) / Application.Run(fPrime, x)
        If Abs(x - oldX) <= maxRelErr * Abs(x) Then
            newton = x
            Exit Function
        End If
    Next k
    newton = "No convergence"
End Function
Function ax4CosX(x As Double)
    ax4CosX = a * x ^ 4 - Cos(x)
End Function
Function ax4CosXPrime(x As Double)
    ax4CosXPrime = 4 * a * x ^ 3 + Sin(x)
End Function
```

The figure below shows the worksheet that calls the newton function; the worksheet is in the equation-display mode. The same function that is used in the Newton's method routine, ax4cosx, is used here to compute the value of f(solution). (Row identifiers not shown; rows shown are 1 to 5.)

F	G	H
Initial Guess	Solution	f(solution)
-2	=newton(F2,"ax4cosx","ax4cosxPrime")	=ax4cosx(G2)
2	=newton(F3,"ax4cosx","ax4cosxPrime")	=ax4cosx(G3)
5	=newton(F4,"ax4cosx","ax4cosxPrime")	=ax4cosx(G4)
7	=newton(F5,"ax4cosx","ax4cosxPrime")	=ax4cosx(G5)

The normal view below shows the solutions and the values of $f(x_{solution})$ for each initial guess.

F	G	H
Initial Guess	Solution	f(solution)
-2	-1.56777563575	-4.00E-17
2	1.56777563575	-4.00E-17
5	5.04121453937	-2.86E-16
7	6.60168529647	-5.03E-16