### Excel & Visual Basic for Applications (VBA)

- Algorithm development: bisection
- Spreadsheet prototyping: bisection
- VBA programming: bisection
- Debugging and documentation

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Algorithm Design
- Statement of the objective
- Written step-wise description of the method
- Flowchart the method

### Prototyping
- Develop an Excel spreadsheet to test the method and demonstrate its effectiveness & limitations

### Program Development
- Write, code and test VBA program

### Documentation
- Commenting of code
- Nomenclature tables
- Users’ guide
Case Study Problem

Develop a user-defined function in VBA that solves for a root of a nonlinear equation, given the evaluation of the equation on the spreadsheet based on input cells, and using the bisection method.

Start with a picture of the method

The Bisection Method  finding a root of $f(x)=0$

1) obtain two starting estimates for the root, $x_1$ and $x_2$
2) evaluate $f(x_1)$ and $f(x_2)$
3) check that $f(x_1)$ and $f(x_2)$ are of opposite sign. If not, the starting estimates are not appropriate, stop; otherwise, go on
4) compute the midpoint between $x_1$ and $x_2$: $x_{mid}$, and $f(x_{mid})$
5) if this is not the first $x_{mid}$, check to see whether the change between this $x_{mid}$ and the last one is small, below some tolerance threshold. if it is below the tolerance, stop and return $x_{mid}$ as the solution; otherwise, continue
6) if $f(x_{mid})$ and $f(x_1)$ have the same sign, let $x_1$ become $x_{mid}$; otherwise let $x_2$ become $x_{mid}$
7) return to step 4)
\begin{align*}
  h^3 - 3Rh^2 + \frac{3V}{\pi} &= 0 \\
  \text{or} \\
  f(h) &= 0
\end{align*}

We will try a radius of 10 feet (3.05m) and a liquid volume of 500 gal (1.9m$^3$).

Initial estimates: $h_1 = 0$ \quad $h_2 = 2*R$
VBA Code Development

Spreadsheet

VBA Code

```
Function Bisect(x1, x2)
    h1 = x1
    h2 = x2
    h = Bisect(h1, h2)
    ' End Function

Function f(x)
    ' End Function
```

Excel spreadsheet prototype
[done before, see Class 10]
**VBA Code for Bisect Function**

```vba
Function Bisect(xl, x2)
    Dim xmid, xold, tol As Single
    tol = 0.000001
    xold = xl
    If f(xl) * f(x2) > 0 Then
        Bisect = "bad initial guesses"
    Else
        Do
            xmid = (xl + x2) / 2
            If Abs((xmid - xold) / xmid) < tol Then
                Exit Do
            End If
            xold = xmid
            If f(xl) * f(xmid) > 0 Then
                xl = xmid
            Else
                x2 = xmid
            End If
        Loop
        Bisect = xmid
    End If
End Function
```

**VBA Code for Equation Evaluation**

```vba
Function f(x)
    Dim Pi, V, R As Single
    Pi = 4 * Atn(1)
    V = Range("Volume")
    R = Range("Radius")
    f = x ^ 3 - 3 * R * x ^ 2 + 3 * V / Pi
End Function
```

**Spreadsheet used to test Bisect function**

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Development of Bisect Function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0 h1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>20 h2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td>4.31 h</td>
</tr>
<tr>
<td>6</td>
<td>10 Radius</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>500 Volume</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Development of Bisect Function</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>0 h1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>20 h2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>Bisect(h1val,h2val)</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>10 Radius</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>500 Volume</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Checking and debugging the Bisect function

Since Bisect is a function, you can’t step into it from the VBE, nor from Tools/Macro/Run

Put a breakpoint (F9) in the Function Bisect statement

```
Function Bisect(x1, x2)
Dim xmin, xold, tol As Single
tol = 0.000001
xmin = x1
```

Cause the Bisect function to calculate by editing the cell where the function is used changing one of the input cells to the function

```
Function Bisect(x1, x2)
Dim xmin, xold, tol As Single
tol = 0.000001
xold = x1
```

Single-step the code, checking values along the way

```
Function Bisect(x1, x2)
Dim xmin, xold, tol As Single
tol = 0.000001
xold = x1
If f(x1) * f(x2) > 0 Then
    Bisect = “bad initial guesses”
Else
    Do
        xmin = (x1 + x2) / 2
        If (f(x1) * f(xmin) < tol
            Exit Do
    End If
    xold = xmin
    If f(x1) * f(xmin) > 0 Then
        x1 = xmin
```

Skip over stepping through Function f(x), once you know it works ok, with Shift-F8 (“step over”) instead of F8
Skip past the repetitive loop execution by executing to cursor position with Ctrl-F8
Run continuously to end with F5 or Run button
Documentation

add comments to your VBA code:

name and date

description of function
    what it accomplishes
    input requirements
    warnings & limitations

nomenclature for variables

comments between and tagged onto
    statements