Washington Bridge Foundation Libraries

Programmable Software Components for Bridge Engineers

Getting Started

A member of the Alternate Route Project

WASHINGTON BRIDGE FOUNDATION LIBRARIES

Getting Started

Introduction

Ever wish you could do a rigid frame analysis directly in an Excel spreadsheet without cutting and pasting from your FEM software? Maybe you've dreamed of computing LRFD HL93 live load responses directly from MathCAD without having to write pages and pages of logic? Do you remember that sinking feeling when you found a bug in your spreadsheet two weeks after it was in widespread use by your colleagues?

Today most production engineers use tools like spreadsheets and MathCAD to create custom solutions to automate routine and repetitive calculations. Until recently, users of these tools were unable to create solutions of any appreciable complexity without being hampered by their limitations. For instance, to perform a continuous beam analysis in a spreadsheet you would either have to limit the number of spans to accommodate a closed form solution, or you would have to undertake a difficult and complex programming effort. Well not any more!

The Washington Bridge Foundation Libraries (WBFL) is a library of software components that enables engineers with minimal programming skills to supercharge spreadsheets and MathCAD documents to easily solve complex engineering problems. The WBFL also helps engineers to overcome many of the pitfalls associated with spreadsheets and other document-orientated solutions. The WBFL takes spreadsheet development to a new level.

Think of WBFL as an arsenal of ready-to-use software components that can be seamlessly integrated into spreadsheets, web pages, script files, and MathCAD documents – an arsenal that allows you to solve problems you never thought possible. The WBFL components can be used to perform a variety of engineering computations including coordinate geometry, section properties analysis, and plane frame structural analysis.

This document will introduce you to the WBFL and will show how the various components can be utilized to simplify and enhance the development of spreadsheet and MathCAD solutions.

Let’s Get Started

If this is the first time you are reading this document, we expect that you have just installed the WBFL components onto your computer and are wondering what to do next. This is good – you’ve come to the right place, keep on reading…

The first concept that you need to understand is that the components themselves are not computer programs – they are software libraries, in the form of Dynamic Link Libraries (.dll’s), that you can use to build *your* computer programs. For your convenience, this document contains detailed descriptions of two example programs. Also, there are many others examples provided with the installation.

By default, the components are installed into the `C:\Program Files\Washington State Department of Transportation\WBFL\` folder. The components (.dll’s), along with their associated help files, can be found there. This is also where this document and the example files are installed. Now is a good time to browse to this folder and get acquainted with the WBFL files. Go ahead and open up some of the help files and examples – soon they will become your primary tools for learning more about the WBFL components. We will talk soon in this document about what to do with the dll’s.

Prerequisites - What Do I Need To Know?

Although they are no doubt useful for many other disciplines, the WBFL Components were written with bridge engineers in mind. In fact, we expect that most users of this system will be licensed bridge engineers or structural engineers who don't have much programming experience. Which leads us to the next prerequisite:

In order to use the components, you must have a basic understanding of Visual Basic/VBA and/or programming with the Microsoft Component Object Model (COM). If your knowledge of programming is limited to what you learned in that FORTRAN class way back in college, you are likely in for a struggle. Don't fret though, Visual Basic is probably the most popular programming language in the world and there are lots of resources out there to help you learn it.

For first-time users of the LBAM system, we recommend the VBA programming environment in Microsoft Excel as a good place to get familiar with the system. Excel has a decent code editor and debugger, and it integrates well with the context-sensitive help provided with the LBAM system.

Note that Visual Basic (VB) and Visual Basic For Applications (VBA) are very similar, but are not the same language. Visual Basic is a full-featured programming environment that is used for creating stand-alone programs; while VBA is hosted within other programs such as Word, Excel, AutoCad, and Access. One good thing though: if you are familiar with one, it is easy to learn the other.

The WBFL Components

The WBFL consists of several components for your use:

• **FEM2D**	Plane Frame Finite Element Modeling
• **Geometry**	Geometric Primitives and Shape Modeling
• **LBAM**	Longitudinal Bridge Analysis Model components
• **Sections**	Section Property Analysis
• **Tools**	Miscellaneous Tools
• **UnitServer**	Unit Conversions, Base Unit Management, Unit Mode Management, and Display Unit Management

Detailed documentation for each component is provided in the form of an on-line help file. The help files have a .chm suffix and are given similar names to their corresponding dll’s located in the same folder.

Some development environments, such as Visual Basic and Excel's VBA, work to provide context-sensitive help for the components. This feature will be explored in the examples given later in this document.

The Examples

After you have finished reading this document, you will want to browse the many examples provided with the WBFL installation. The following table provides a short description of each example

Excel VBA Examples

This section provides examples of using the WBFL components in the Microsoft Excel Visual Basic for Applications (VBA) programming environment.

File Name	Description
CoordinateTransformation.xls	Uses the Geometry library to transform the coordinates of a point from on coordinate system to another
Culvert.xls	Uses the LBAM library to compute the live load response for a multi-cell culvert with an arched slab
Enveloper.xls	Demonstrates how to use the LBAM Enveloper component to compute the response envelope of simply supported response verses simple-for-dead, continuous-for-live-load response
GettingStarted.xls	Source for the example described in this document
Influence.xls	Compute influence lines for a continuous beam using the Fem2d library. Note that this is the hard way – using the LBAM makes it much easier
LiveLoader.xls	Live load response for a continuous, prismatic superstructure using the LBAM library
SDCL.xls	LRFD live load and limit state responses for a simple-for-dead-load, continuous-for-live-load analysis using the LBAM library
SectionProps.xls	Section properties of a built-up composite plate girder using the Section library
SimpleSample.xls	Example used in the getting started section of the LBAM library help file
T-BeamCapacity.xls	Moment capacity of a prestressed concrete T-beam using the RcCapacity library
TriangleBeamMomentCapacity.xls	Moment capacity of a triangular reinforced concrete beam using the RcCapacity library

MathCad Examples

This section provides examples of driving the components from the scripting environment in MathCAD

File Name	Description
GettingStarted.mcd	Source for the example described in this document
InfluenceLine.mcd	Computation of an influence line using the Fem2d engine

Scripting Examples

This section provides examples of driving the components from the scripting languages provided in HTML pages.

File Name	Description
Elevations.htm	Computes elevations, stations and offsets of points located along a bridge alignment.
GettingStarted.htm	HTML implementation of the example given in this document
Portal.htm	Deflection analysis of a portal frame using the Fem2d library

Visual Basic (VB) Examples

This section provides examples of driving the components from Visual Basic

File Name	Description
ArpNotice.vdp	Application that shows how to use the ARP Notice component found in the WBFLTools library
Convert.vdp	Unit conversion application using the WBFLTools library
MohrCircle.vdp	Compute principle stresses using the MohrCircle component in the WBFLTools library
PDelta.vdp	Compute the buckling capacity of a column using the P-Delta technique and the Fem2d library
UnitSystem.vdp	Demonstrates how to use the UnitSystem library to manage the use of units between different components

C++ (VC) Examples

This section provides examples of driving the components from C++

File Name	Description
ArpNotice.vdp	Application that shows how to use the ARP Notice component found in the WBFLTools library
LBAMDumper	Dump utility console application for the LBAM.

Using WBFL™ Components with Microsoft® Excel®

This document assumes you have a working knowledge of Visual Basic for Applications (VBA) and VBScript. If you are unfamiliar with VBA, we suggest you contact your local computing resource center and take an Advanced Excel class.

While you have been able to write macros in Microsoft Excel for quite some time now, the 97 version of the software was enhanced to support for Visual Basic for Applications (VBA). VBA gives you the ability to create custom programs that can perform just about any task you like within the Excel program. This section will provide an overview of using WBFL components with Microsoft Excel. The techniques presented here can be adapted to spreadsheet applications with similar features.

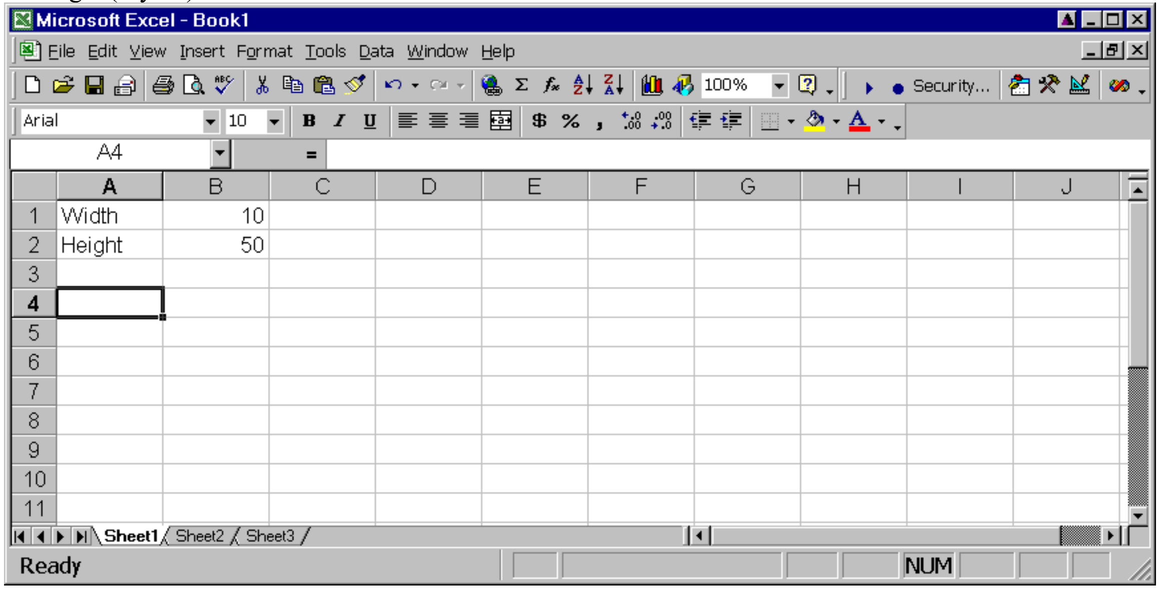
Example: Geometric Properties of a Rectangle

Let’s start with a simple example to demonstrate the basics of utilizing a WBFL component. In this example we will create a spreadsheet that computes the geometric properties of a rectangle. Obviously this is something you can do in Excel without the aid of external components, but keep in mind that this is a simple example. In a real problem your spreadsheet would be implementing a series of complex calculations. The calculation of geometric properties of a shape is only one intermediate step in the overall process.

Instead of putting formulas in the cells of the spreadsheet, we will create a macro to control the process. The macro won't actually compute the section properties. It will delegate that responsibility to the Rectangle component of the WBFL Geometry library. The spreadsheet is the user interface, the macro simply ferries data between the spreadsheet and the Rectangle component, and the Rectangle component does all the computational work

Let's get started:

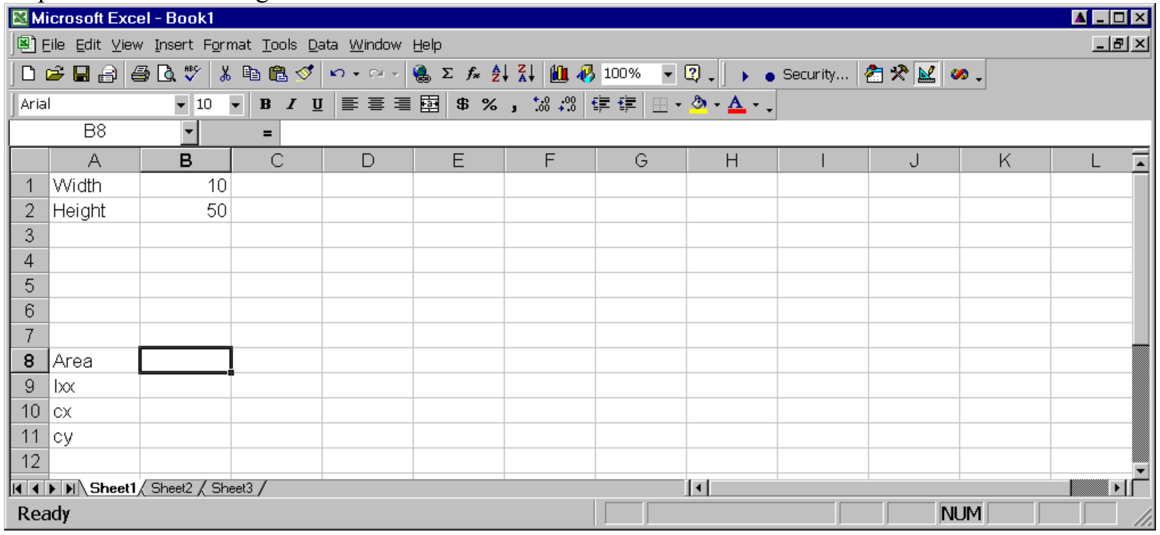
1. Start Excel and open a new workbook. Excel usually creates a new blank workbook for you when it starts.
2. Define the input to the problem. In cell A1, enter the title "Width". In cell B1 enter the width of the rectangle (say 10). In cell A2, enter the title "Height". In cell B2 enter the height of the rectangle (say 50).



	A	B	C
1	Width	10	
2	Height	50	

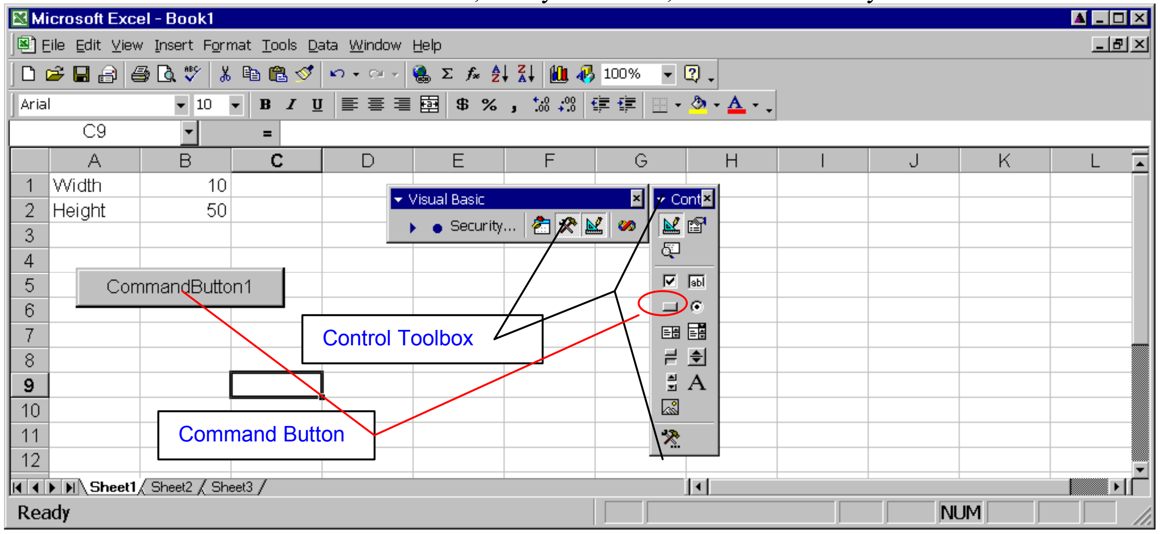
3. Setup a place to put the output. For this example lets have the output be Area, Ixx, and the centroid. Enter the titles "Area", "Ixx", "cx", and "cy" in cells A8 through A11. We will write the

output to cells B8 through B11



	A	B	C	D	E	F	G	H	I	J	K	L
1	Width	10										
2	Height	50										
3												
4												
5												
6												
7												
8	Area											
9	Ixx											
10	cx											
11	cy											
12												

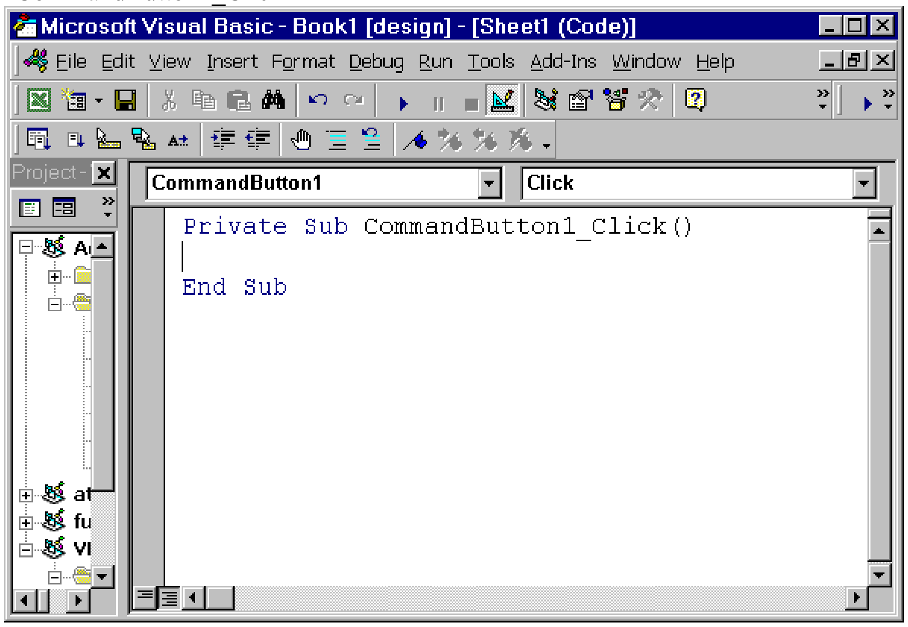
4. Add a command button to the spreadsheet.
Select View | Toolbars | Visual Basic to show the Visual Basic toolbar.
Click on the Control Toolbox button to show the Control Toolbox.
Click on the "Command Button" button and, with your mouse, draw a button on your worksheet



5. Add an event handler for the command button. An event handler is code (program instructions) that is executed when the button is pressed. Our event handler will compute the properties of the rectangle and report them back to the worksheet.

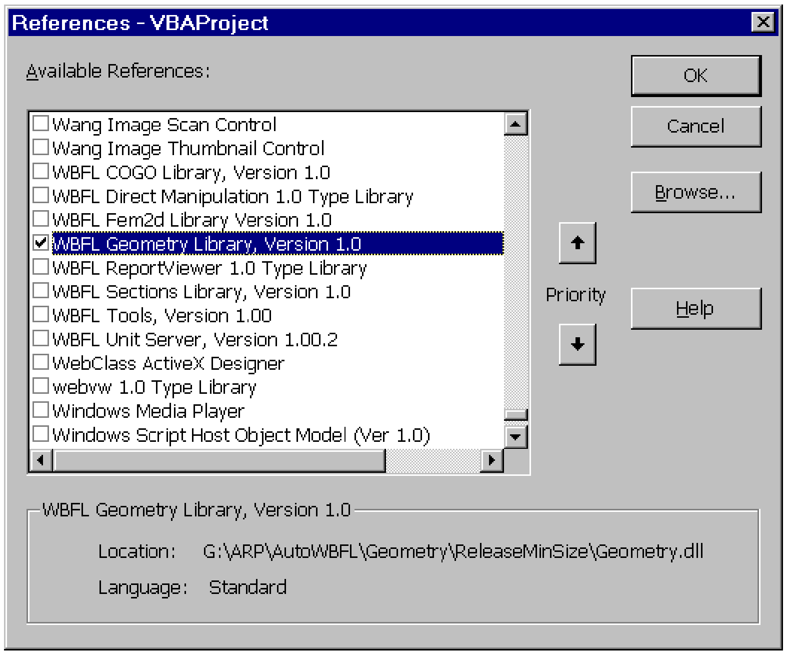
To start the event handler, double click on the Command Button you just drew on the worksheet. Excel will automatically open up the Visual Basic Editor and "stub out" an event handler called

"CommandButton1_Click"


Microsoft Visual Basic - Book1 [design] - [Sheet1 (Code)]
File Edit View Insert Format Debug Run Tools Add-Ins Window Help
Project -
CommandButton1
Click

```
Private Sub CommandButton1_Click()

End Sub
```

6. Before we can use the WBFL Geometry library, we have to reference it. That is, we need to tell Excel about the library. Select References from the Tools menu. This will present the References dialog. Scroll down to the bottom and check the box next to WBFL Geometry Library, Version 1.0. Press the OK button.



Available References:

- ☐ Wang Image Scan Control
- ☐ Wang Image Thumbnail Control
- ☐ WBFL COGO Library, Version 1.0
- ☐ WBFL Direct Manipulation 1.0 Type Library
- ☐ WBFL Fem2d Library Version 1.0
- ☑ WBFL Geometry Library, Version 1.0
- ☐ WBFL ReportViewer 1.0 Type Library
- ☐ WBFL Sections Library, Version 1.0
- ☐ WBFL Tools, Version 1.00
- ☐ WBFL Unit Server, Version 1.00.2
- ☐ WebClass ActiveX Designer
- ☐ webvw 1.0 Type Library
- ☐ Windows Media Player
- ☐ Windows Script Host Object Model (Ver 1.0)

WBFL Geometry Library, Version 1.0

Location: G:\ARP\AutoWBFL\Geometry\ReleaseMinSize\Geometry.dll

Language: Standard

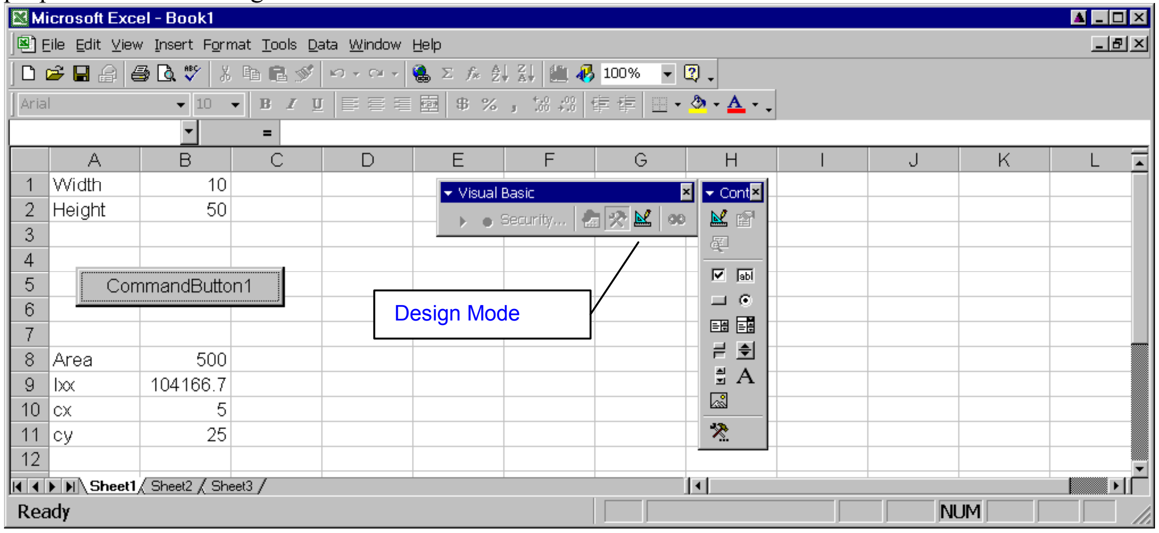
7. Now we have to write the macro. The macro will read the input from the spreadsheet, create a Rectangle object, position the rectangle, extract the section properties, and report them back to the spreadsheet. The code for doing this is shown below.

```
Private Sub CommandButton1_Click()
   'Gain access to the worksheet object
   Dim ws As Worksheet
   Set ws = Worksheets("Sheet1")
```

```
    'Extract the input parameters
    Dim width As Double
    Dim height As Double
    width = ws.Cells(1, "B")
    height = ws.Cells(2, "B")

    'Create a Rectangle object and set the
    'width and height
    Dim rect As New WBFLGeometry.rect
    rect.width = width
    rect.height = height

    'The center of the rectangle is at (0,0) by default
    'Move the bottom left corner to (0,0)
    Dim point As New WBFLGeometry.Point2d
    point.Move 0, 0
    rect.XYPosition.LocatorPoint(lpBottomLeft) = point

    'Get the geometric shape properties
    'of the rectangle
    Dim props As WBFLGeometry.ShapeProperties
    Set props = rect.Shape.ShapeProperties

    'Write the properties to the worksheet
    ws.Cells(8, "B") = props.Area
    ws.Cells(9, "B") = props.Ixx
    ws.Cells(10, "B") = props.Centroid.X
    ws.Cells(11, "B") = props.Centroid.Y
End Sub
```

8. Before we go on, let's try using the context-sensitive help. Select the text "`WBFLGeometry.Rect`" in the source and press the F1 key on your keyboard. This will bring up the help for the Rect object – pretty cool huh!

9. Now it's time to run the macro. Close the Visual Basic Editor window. Go to the Visual Basic Toolbar and make sure the Design Mode button is NOT pressed. Press the command button on the worksheet. If everything works correctly, cells B8-B11 will be filled in with the geometric properties of the rectangle.



	A	B
1	Width	10
2	Height	50
8	Area	500
9	Ixx	104166.7
10	cx	5
11	cy	25

10. Now, change width and/or height and press the command button again. The geometric properties will be updated.

Image that instead of a rectangle, you had a multi-cell box girder cross-section that was rotated due to roadway superelevation. Computing section properties, including the product of inertia and principle directions, using formulas in the cells of the spreadsheet would be a monumental task, but can be easily accomplished with the WBFL components.

The complete code for this example can be found in "C:\Program Files\Washington State Department of Transportation\WBFL\Examples\VBA\GettingStarted.xls"

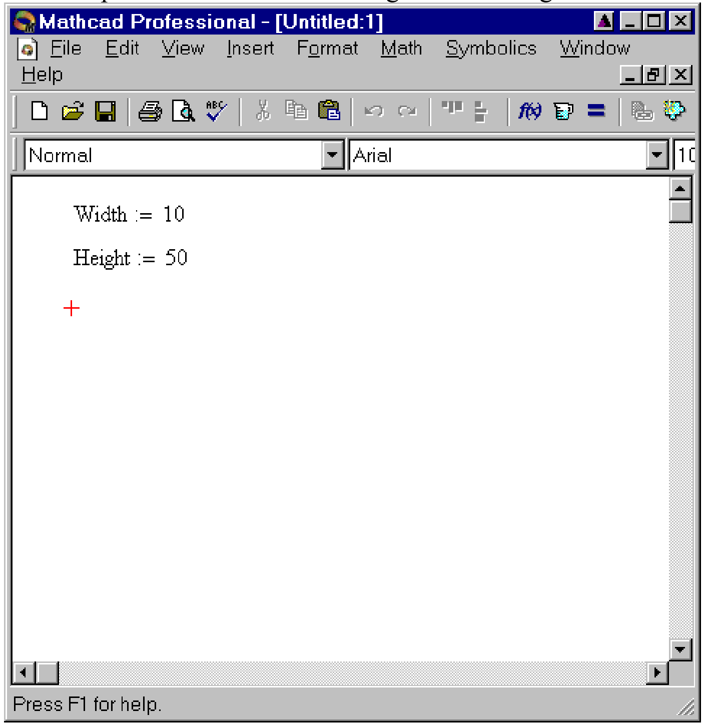
Using WBFL™ Components with MathCAD®

MathCAD is very popular with engineers because it presents calculations as if they were performed by hand. This is a great feature and one of the major advantages over Excel. However, when MathCAD documents are full of programming instructions and flow of control logic, they stop being human readable.

It is not well known that MathCAD provides its users with the ability to add Automation compatible controls into a worksheet and control them with JavaScript and VBScript. This example will demonstrate how to call WBFL components by computing the geometric properties of a rectangle. Again, this is a simplistic example. Imagine that computing geometric properties is one step in a detailed and complex process.

Example: Geometric Properties of a Rectangle

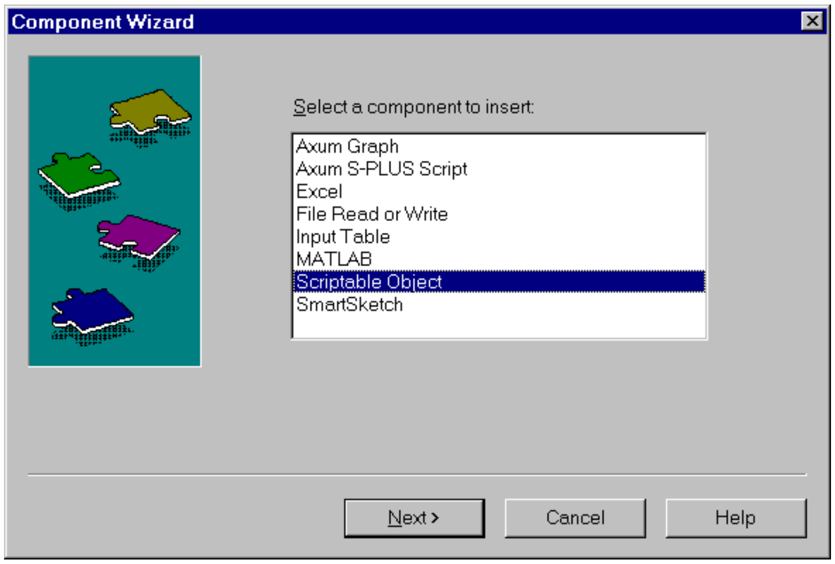
1. Start MathCAD and create a new worksheet.
2. Create inputs for the Width and Height of a rectangle


Mathcad Professional - [Untitled:1]
File Edit View Insert Format Math Symbolics Window Help
Normal
Arial
Press F1 for help.

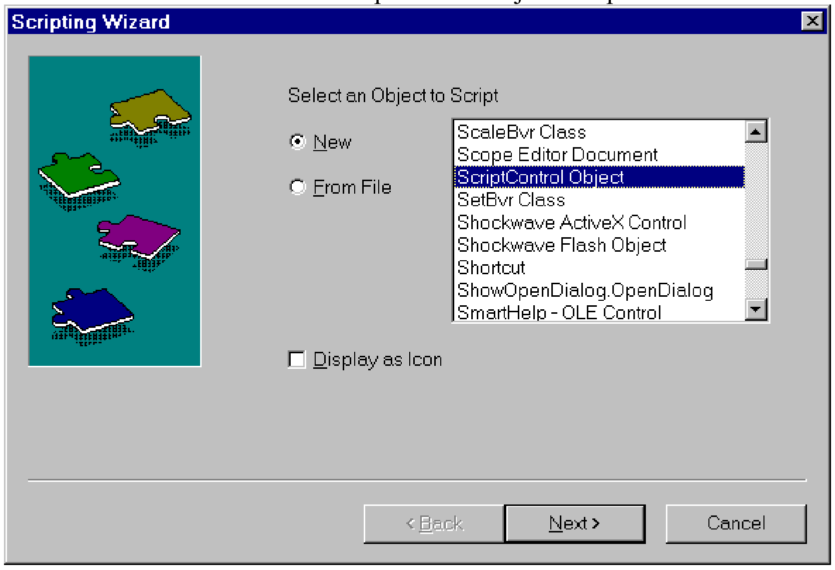
Width := 10

Height := 50

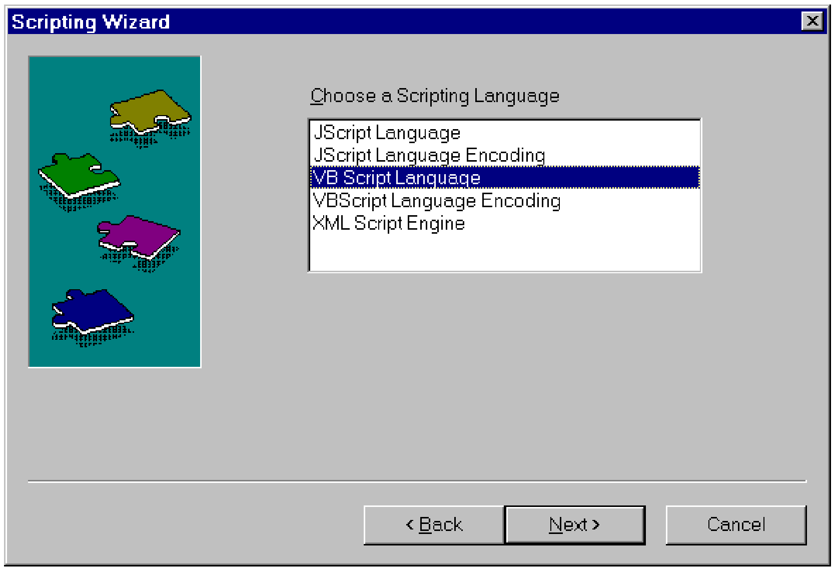
3. Next we need to insert the script component into MathCAD.
Select Component from the Insert menu. This will start the Component Wizard.
Select Scriptable Object and press the Next button.



Scroll down the list and select ScriptControl Object and press the Next button



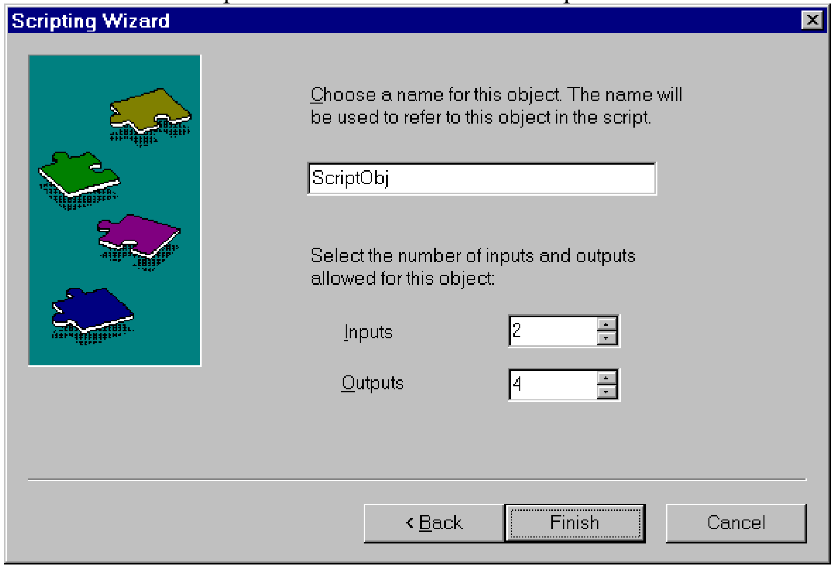
Select VB Script Language as the scripting language. Press the Next button


Scripting Wizard

Choose a Scripting Language

JScript Language
JScript Language Encoding
VB Script Language
VBScript Language Encoding
XML Script Engine

Set the number of Inputs to 2 and the number of Outputs to 4. Press the Finish button


Scripting Wizard

Choose a name for this object. The name will be used to refer to this object in the script.

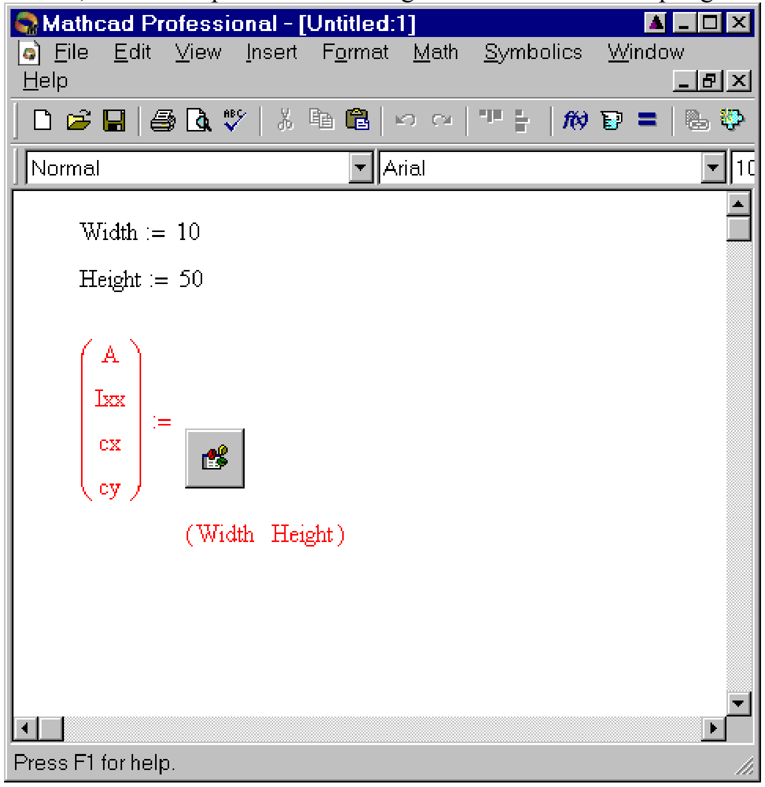
ScriptObj

Select the number of inputs and outputs allowed for this object:

Inputs 2

Outputs 4

4. A scripting component will be added to your worksheet. Enter the input variables along the bottom, and the output variables along the left side of the scripting component.


Mathcad Professional - [Untitled:1]
File Edit View Insert Format Math Symbolics Window Help
Normal
Arial
Press F1 for help.

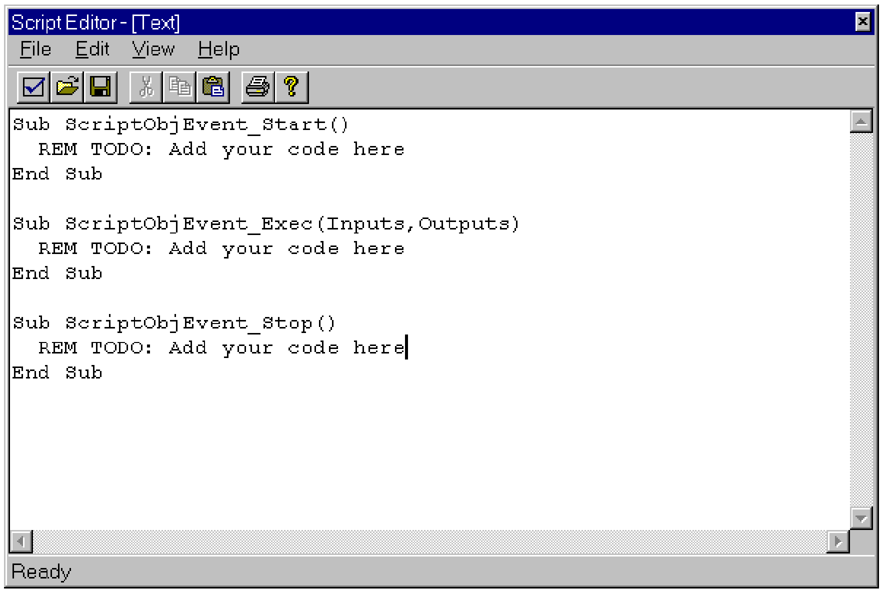
$Width := 10$

$Height := 50$

$$\begin{pmatrix} A \\ Ixx \\ cx \\ cy \end{pmatrix} := \quad (Width \;\; Height)$$

5. Next we have to write the script that actually computes the geometric properties of the rectangle. Right click on the scripting component and select Edit Script from the menu.

6. This will open the Script Editor window. You will see that MathCAD "stubbed out" three subroutines for us. This script editor isn't nearly as nice as the one in Excel, but its what you've got to work with.



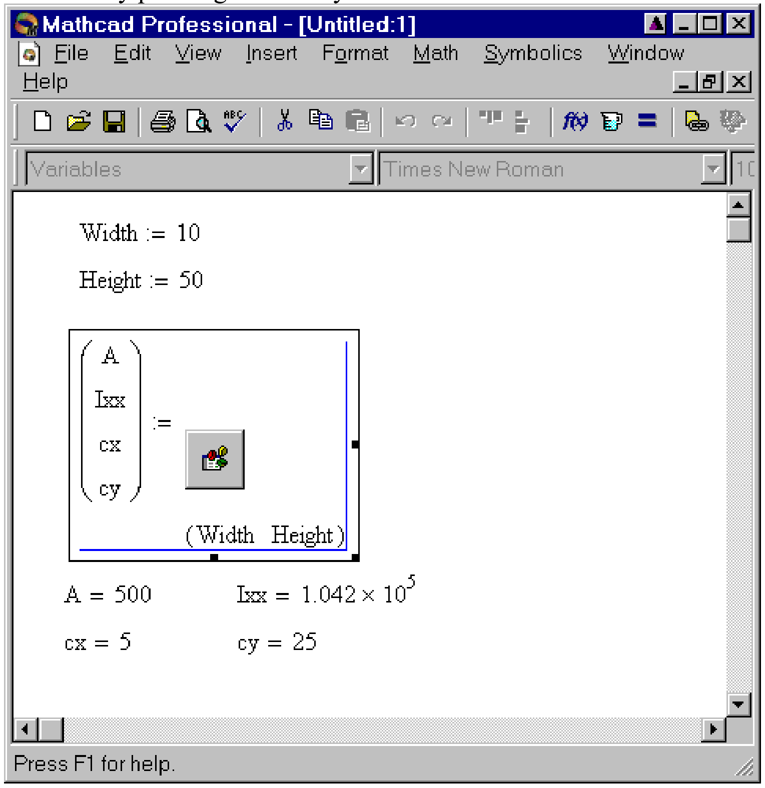
```
Sub ScriptObjEvent_Start()
  REM TODO: Add your code here
End Sub

Sub ScriptObjEvent_Exec(Inputs,Outputs)
  REM TODO: Add your code here
End Sub

Sub ScriptObjEvent_Stop()
  REM TODO: Add your code here
End Sub
```

The one we are interested in is ScriptObjEvent_Exec. The Inputs object contains the input variables. We will put the output in the Outputs object. For more information on the Inputs and Outputs objects see http://www.mathsoft.com/mathcad/library/mathconnex/scripted.htm.

7. Now we have to write the script. The script will read the input from the Inputs object, create a Rectangle object, position the rectangle, extract the section properties, and save the results into the Outputs object. The code for doing this is shown below.

```
Sub ScriptObjEvent_Exec(Inputs,Outputs)
  'Extract the input parameters
  Dim width, height
  width = Inputs(0).Value(0,0)
  height = Inputs(1).Value(0,0)

  'Create a Rectangle object and set the
  'width and height
  Dim rect
  Set rect = CreateObject("WBFLGeometry.Rectangle")
  rect.Width = width
  rect.Height = height

  'The center of the rectangle is at (0,0) by default
  'Move the bottom left cornder to (0,0)
  Dim point
  Set point = CreateObject("WBFLGeometry.Point2d")
  point.Move 0,0
  rect.XYPosition.LocatorPoint(7) = point

  'Get the geometric shape properties
  Dim props
  Set props = rect.Shape.ShapeProperties

  'Write the properties to the Output object
  Outputs(0).Value(0,0) = props.Area
  Outputs(1).Value(0,0) = props.Ixx
  Outputs(2).Value(0,0) = props.Centroid.X
  Outputs(3).Value(0,0) = props.Centroid.Y
End Sub
```

8. Finally, display the values of the output variables. You may have to force MathCAD to recalculate the sheet by pressing the F9 key


Mathcad Professional - [Untitled:1]
File Edit View Insert Format Math Symbolics Window Help
Variables
Times New Roman
Press F1 for help.

Width := 10

Height := 50

$\begin{pmatrix} A \\ Ixx \\ cx \\ cy \end{pmatrix} :=$ (Width Height)

A = 500 Ixx = 1.042×10^5

cx = 5 cy = 25

9. Change the input parameters and the output will automatically be updated if you have Automatic Calculation turned on.

A complete example of the geometric properties of a rectangle can be found in “"C:\Program Files\Washington State Department of Transportation\WBFL\Examples\MathCAD\GettingStart.mcd".

That was just the Beginning

That was just the beginning. The WBFL is comprised of several libraries, each with components that can make your spreadsheet and MathCAD documents more powerful then ever before.

The best way to learn more about the WBFL components is to study the on-line documentation and the examples. The complete documentation for each library can be found on the Start menu. You can also find additional examples of the WBFL components in C:\Program Files\Washington State Department of Transportation\WBFL\Examples.

The possibilities are limitless. Let your imagination be your guide...