

©2022 ANSYS, Inc. All Rights Reserved. Unauthorized use, distribution or duplication is prohibited.

# **Ansys ACT Customization Guide for Fluent**



ANSYS, Inc.
Southpointe
2600 Ansys Drive
Canonsburg, PA 15317
ansysinfo@ansys.com
http://www.ansys.com
(T) 724-746-3304
(F) 724-514-9494

Release 2022 R2 July 2022

ANSYS, Inc. and ANSYS Europe, Ltd. are UL registered ISO 9001: 2015 companies.

### **Copyright and Trademark Information**

© 2022 ANSYS, Inc. Unauthorized use, distribution or duplication is prohibited.

ANSYS, Ansys Workbench, AUTODYN, CFX, FLUENT and any and all ANSYS, Inc. brand, product, service and feature names, logos and slogans are registered trademarks or trademarks of ANSYS, Inc. or its subsidiaries located in the United States or other countries. ICEM CFD is a trademark used by ANSYS, Inc. under license. CFX is a trademark of Sony Corporation in Japan. All other brand, product, service and feature names or trademarks are the property of their respective owners. FLEXIm and FLEXnet are trademarks of Flexera Software LLC.

### **Disclaimer Notice**

THIS ANSYS SOFTWARE PRODUCT AND PROGRAM DOCUMENTATION INCLUDE TRADE SECRETS AND ARE CONFIDENTIAL AND PROPRIETARY PRODUCTS OF ANSYS, INC., ITS SUBSIDIARIES, OR LICENSORS. The software products and documentation are furnished by ANSYS, Inc., its subsidiaries, or affiliates under a software license agreement that contains provisions concerning non-disclosure, copying, length and nature of use, compliance with exporting laws, warranties, disclaimers, limitations of liability, and remedies, and other provisions. The software products and documentation may be used, disclosed, transferred, or copied only in accordance with the terms and conditions of that software license agreement.

ANSYS, Inc. and ANSYS Europe, Ltd. are UL registered ISO 9001: 2015 companies.

### **U.S. Government Rights**

For U.S. Government users, except as specifically granted by the ANSYS, Inc. software license agreement, the use, duplication, or disclosure by the United States Government is subject to restrictions stated in the ANSYS, Inc. software license agreement and FAR 12.212 (for non-DOD licenses).

### **Third-Party Software**

See the legal information in the product help files for the complete Legal Notice for ANSYS proprietary software and third-party software. If you are unable to access the Legal Notice, contact ANSYS, Inc.

Published in the U.S.A.

# **Table of Contents**

Introduction	
ACT Start Page and Tool Access	1
UDF Folder	1
Fluent Wizards	3
Fluent Wizard (MSH Input File)	3
Defining FluentDemo1	3
Defining Functions for FluentDemo1	6
Reviewing the Wizard in FluentDemo1	7
Fluent Wizard (CAS Input File)	
Defining FluentDemo2	9
Defining Functions for FluentDemo2	
Reviewing the Wizard in FluentDemo2	

# Introduction

This guide assumes that you are familiar with the general ACT usage information in the ACT Developer's Guide. This first section provides ACT usage information specific to Fluent:

**ACT Start Page and Tool Access** 

**UDF** Folder

While Fluent does not currently support using ACT to deliver custom features via extensions, it does support using UDF libraries (p. 1) for feature creation. Fluent also supports target product wizards, which are described in the subsequent section (p. 3).

#### Note:

For information on all ACT API changes and known issues and limitations that may affect your existing ACT extensions, see Migration Notes and Known Issues and Limitations in the *Ansys ACT Developer's Guide*.

### **ACT Start Page and Tool Access**

From a stand-alone instance of Fluent, you access the ACT Start Page by doing one of the following:

- During startup, select the **Load ACT** check box in the **Fluent Launcher**.
- When Fluent is running, click **Arrange the workspace** and then **ACT**, which toggles between showing and hiding the **ACT Start Page**.
- Enter the text command file/load-act-tool.

The **ACT Start Page** opens as a new component in the Fluent layout.

The **ACT Start Page** for a stand-alone instance of Fluent has an icon for accessing the Extension Manager. However, when Fluent is opened within Workbench, the **ACT Start Page** accessed by using one of the previous methods does not have the icon. This is because you must manage extensions from the **ACT Start Page** for Workbench.

Once accessed, the **ACT Start Page** and ACT tools are all used as described in the *Ansys ACT Developer's Guide*.

### **UDF** Folder

When creating an extension for Fluent, in the XML file, the element simdata> can have a child <udf>
element. This element specifies the location of the folder containing UDF libraries.

```
<simdata context="Fluent">
  <udf folder="udf_library"></udf>
  </simdata>
```

The path that you supply for the UDF folder uses the same logic as the path that you supply for the extension's IronPython script, where src is the path relative to the extension folder:

```
<script src="main.py"/>
```

Paths for both of these elements are relative to the extension folder. However, the element script>
requires a file relative path, while the element <udf> requires a folder relative path.

Sample XML code for a Fluent extension with the element <udf> follows. The extension FluentTest is a theoretical example.

```
<extension name="FluentTest" version="l" icon="icon.png">
    <guid>36DB3B25-C5E2-4E43-9A04-BA53AA7DC05A</guid>
    <author>Ansys, Inc.</author>
    <interface context="Fluent"></interface>
    <description>A sample extension for Fluent with UDF encapsulation</description>
    <script src="main.py" />
        <simdata context="Fluent">
        <udf folder="udf_library"></udf>
        </simdata>
    </extension>
```

Because the UDF libraries in the specified folder are packaged with the extension, when this extension is installed, these libraries can be loaded.

When you specify the UDF folder name in the element <udf>, do not specify a Fluent version. The extension automatically appends the Fluent version in which the extension is running to the name of the folder. However, because UDF libraries are version-dependent, you must name each of the real UDF folders with the specified name followed by an underscore and the version.

In the previous example, the name specified for the UDF folder is **udf\_library**. However, the real folders are named **udf\_library** followed by an underscore and then the Fluent versions. The extension searches for the name given in the element **<udf>** followed by the Fluent version in which you are running the extension.

Thus, the naming convention of the UDF folder in the XML file is <code>foldername</code>, and the naming convention for the real folder is <code>foldername\_Fluent\_version</code>.

# **Fluent Wizards**

You can use ACT to create target product wizards for Fluent. The supplied package ACT Wizard Templates contains a folder named Templates-FluentWizard. In this folder are two extension examples, FluentDemo1 and FluentDemo2, along with their sample input files. For download information, see Extension and Template Examples.

This section describes the supplied Fluent wizards for importing MSH and CAS files:

Fluent Wizard (MSH Input File)

Fluent Wizard (CAS Input File)

#### Note:

You use the Extension Manager to install and load extensions and the Wizards launcher to start a target product wizard.

# Fluent Wizard (MSH Input File)

The supplied extension FluentDemo1 includes a target project wizard for Fluent named Simple Analysis (Fluent Demo 1). This Fluent wizard imports an MSH file, runs a simple analysis, generates results, and displays the results in a custom panel in the user interface. This example can be used in either of the following scenarios:

- Within a Workbench project that contains a Fluent (with Fluent Meshing) system
- In a Fluent stand-alone project with Meshing Mode selected

The following topics describe the extension **FluentDemo1**:

**Defining FluentDemo1** 

Defining Functions for FluentDemo1

Reviewing the Wizard in FluentDemo1

# **Defining FluentDemo1**

The file FluentDemo1.xml follows. For both the elements <extension> and <wizard>, the attribute context is set to Fluent.

```
<extension name="FluentDemo1" version="1">
  <guid>A0A3E28C-E094-4B2D-8A91-A0B0884D1AE2</guid>
  <author>Ansys, Inc.</author>
  <description>This extension demonstrates how to create a wizard in Fluent (.msh file import).</description>
  <script src="main.py" />
  <interface context="Fluent">
```

```
<images>images</images>
</interface>
<uidefinition>
<layout name="ResultsView">
  <component</pre>
  name="Title"
  topAttachment=""
   topOffset="10"
  leftAttachment=""
  leftOffset="10"
   rightAttachment=""
  rightOffset="10"
  bottomOffset="10"
  bottomAttachment="Properties"
  widthType="Percentage"
   width="100"
  heightType="FitToContent"
  height="200"
  componentType="startPageHeaderComponent" />
  <component
  name="Properties"
   topAttachment="Title"
  leftAttachment="'
  leftOffset="10"
  rightAttachment=""
  rightOffset="10"
   bottomAttachment="Report"
  bottomOffset="10"
  widthType="Percentage"
   width="100"
  heightType="Percentage"
  height="20"
  componentType="propertiesComponent" />
  <component
  name="Report"
  topAttachment="Properties"
  leftAttachment="
   leftOffset="10"
  rightAttachment=""
  rightOffset="10"
  bottomAttachment="Chart"
  bottomOffset="10"
   widthType="Percentage"
   width="100"
  heightType="Percentage"
  height="40"
  componentType="helpContentComponent" />
  <component
  name="Chart"
   topAttachment="Report"
  leftAttachment="
  leftOffset="10"
  rightAttachment=""
  rightOffset="10"
   bottomAttachment="Submit"
  bottomOffset="10"
   widthType="Percentage"
   width="100"
  heightType="Percentage"
  height="40"
   componentType="chartComponent" />
  <component
  name="Submit"
   topAttachment="Chart"
   leftAttachment="'
   leftOffset="10"
   rightAttachment=""
   rightOffset="10"
  bottomAttachment=""
```

```
bottomOffset="10"
   widthType="Percentage"
   width="100"
  heightType="FitToContent"
  height="50"
   componentType="buttonsComponent" />
 </layout>
</uidefinition>
<wizard name="Simple Analysis (Fluent Demo 1)" version="1" context="Fluent" icon="wizard_icon">
 <description>Generate the mesh and solve the analysis.</description>
 <step name="Analysis" caption="Analysis" version="1">
  <callbacks>
   <onrefresh>ImportMesh</onrefresh>
   <onupdate>CreateAnalysis</onupdate>
  </callbacks>
  <step name="Results" caption="Results" version="1" layout="ResultsView@FluentDemo1">
  <callbacks>
   <onrefresh>CreateReport</onrefresh>
  </callbacks>
  </step>
</wizard>
</extension>
```

#### **Custom Layout Definition**

The element <uidefinition> defines a custom layout named ResultsView for the Fluent wizard. It is made up of five custom components: Title, Properties, Report, Chart, and Submit. The Properties, Report, and Chart components are views, and the Submit component is a button.

#### **Wizard Definition**

In the element <wizard>, the attribute name is set to Simple Analysis (Fluent Demo 1).

### **Step Definition**

The element <step> defines a step in the wizard. This wizard has two steps: Analysis and Results. For each step:

- The attribute layout specifies the custom layout to apply to the step. For the step
   Results, layout is set to ResultsView@FluentDemo1. The custom layout
   ResultsView is defined in the element <uidefinition>.
- Any element property
   specifies a property and property attributes to be used in the step.
  - For the step Analysis, the property velocity is defined. The attribute caption is set to velocity.

- For the step Results, the property pressure is defined. The attribute caption is set to Pressure.
- The element <callbacks> specifies callbacks to functions defined in the script main.py. For callback descriptions, see Reviewing the Wizard in FluentDemo1 (p. 7).

### **Defining Functions for FluentDemo1**

The IronPython script main.py follows. This script defines all functions executed by the callbacks in the extension's XMLfile:

- The function ImportMesh is invoked by the callback <onrefresh> in the step Analysis.
- The function CreateAnalysis is invoked by the callback <onupdate> in the step Analysis.
- The function CreateReport is invoked by the callback <onrefresh> in the step Results.

```
def ImportMesh(step):
           tui = ExtAPI.Application.ScriptByName("TUI")
           tui.SendCommand("""switch-to-meshing-mode""")
           tui.SendCommand("""/file/read-mesh [ {0} " """.format(System.IO.Path.Combine(
ExtAPI.Extension.InstallDir, "final.msh")))
           tui.SendCommand("""switch-to-solution-mode yes yes""")
           tui.SendCommand("""/display/mesh ok""")
           tui.SendCommand("""/display/views/restore-view right""")
def CreateAnalysis(step):
           tui = ExtAPI.Application.ScriptByName("TUI")
           res = System.IO.Path.GetTempFileName()
           img = System.IO.Path.GetTempFileName()+".jpg"
           curve = System.IO.Path.GetTempFileName()
           tui.SendCommand("""/define/boundary-conditions/velocity-inlet inlet no no yes yes no {0} no 0""".format(step
           tui.SendCommand("""/solve/initialize/hyb-initialization""", True)
           tui.SendCommand("""/solve/iterate 50""", True)
           tui.SendCommand("""/display/mesh ok""", True)
           tui.SendCommand("""/display/views/restore-view right""", True)
           tui.SendCommand("""/display/set/contours/filled-contours? yes""", True)
           tui.SendCommand("""/display/set/contours/surfaces wall inlet outlet ()""", True)
           tui.SendCommand("""/display/contour pressure 0 2""", True)
           \label{tui.SendCommand("""/display/save-picture "{0}" """.format(img), True)} % \[ \frac{1}{2} \left( \frac{1
           tui.SendCommand("""/surface/rake-surface rake-5 0 0 0 0 1 50""", True)
           tui.SendCommand("""/plot/plot-direction 0 0 1""", True)
           tui.SendCommand("""/plot/plot yes "\{0\}" yes yes no no pressure yes 0 0 1 rake-5 ()""".format(curve), True)
           tui.SendCommand("""/report/surface-integrals/area-weighted-avg inlet () pressure yes "{0}" """.format(res),
           val = 0.
           n = 0;
           with System.IO.StreamReader(res) as reader:
                       while n!=5:
                                  line = reader.ReadLine()
                                  if line!=None:
                       str = line.Substring(32)
                       val = System.Double.Parse(str, System.Globalization.CultureInfo.InvariantCulture)
           x=[]
           y=[]
           with System.IO.StreamReader(curve) as reader:
                       while n!=5:
                                  line = reader.ReadLine()
                                  if line!=None:
```

```
while not line.StartsWith(")"):
            tab = line.Split("\t")
           x.Add(System.Double.Parse(tab[0], System.Globalization.CultureInfo.InvariantCulture))
           y.Add(System.Double.Parse(tab[1], System.Globalization.CultureInfo.InvariantCulture))
           line = reader.ReadLine()
   System.IO.File.Delete(res)
   System.IO.File.Delete(curve)
   step.NextStep.Properties["pressure"].Value = val
   step.NextStep.Attributes["img"] = img
   step.NextStep.Attributes["x"] = x
   step.NextStep.Attributes["y"] = y
def CreateReport(step):
   graph = step.UserInterface.GetComponent("Chart")
   graph.Title("Static Pressure")
   graph.ShowLegend(False)
   graph.Plot(step.Attributes["x"], step.Attributes["y"], key="Static Pressure", color='g')
   help = step.UserInterface.GetComponent("Report")
   help.SetHtmlContent(ExtAPI.Extension.InstallDir,"""<center><img style="height:250px" src="{0}"></center>"
```

### **Reviewing the Wizard in FluentDemo1**

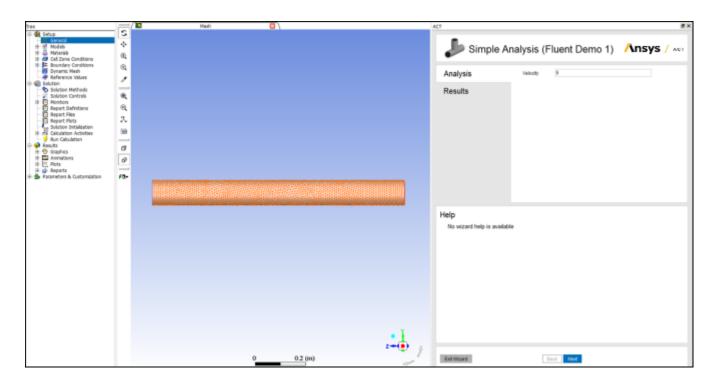
This section reviews the user interface and the processes performed in each step of the wizard Simple Analysis (Fluent Demo 1).

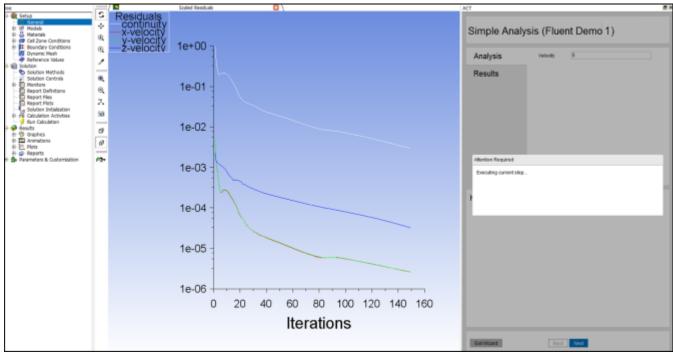
### **Reviewing the Analysis Step**

The first step is **Analysis**, which imports the mesh file and creates the analysis. It requires that a value be specified for the property **Velocity**.

In this step:

- The callback <onrefresh> invokes the function ImportMesh. This function imports the file final.msh and displays the mesh in the Mesh view.
- The callback <onupdate> invokes the function CreateAnalysis when Next is clicked. This
  function sets up the analysis.
  - The element property>
     defines the property Velocity. This property is shown in Fluent
    as a float field with a default value of 3. You enter the desired value. In this example, a value of
    5 is entered.
  - Also defined are additional actions to be performed as part of the analysis.
  - Intermediate results are shown dynamically in the Scaled Residuals window as the analysis is performed.





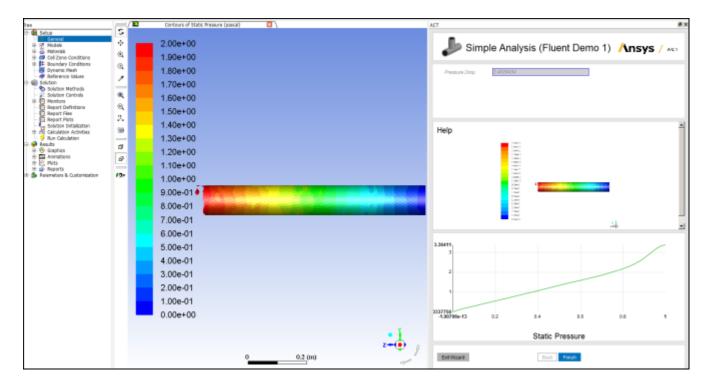
# **Reviewing the Results Step**

The second step is **Results**, which creates a report of the results.

In this step:

• The callback <onrefresh> invokes the function CreateReport.

- The element <layout> specifies the custom layout to apply to the results. This is a child element within the element <uidefinition>.
- The wizard takes the results obtained and displays them:
  - The element property
     defines the property
     PressureDrop
     This calculated value is displayed in Fluent as a ready-only field.
  - The static pressure results are shown in the **Graph** and **Contours** views.



# Fluent Wizard (CAS Input File)

The supplied extension FluentDemo2 includes a target project wizard for Fluent named Simple Analysis (Fluent Demo 2). This Fluent wizard imports a CAS file, runs a simple analysis, generates results, and displays the results in a custom panel in the user interface. This example can be used in either of the following scenarios:

- · Within a Workbench project that contains a Fluent or Fluent (with CFD-Post) system
- In a Fluent stand-alone project with **Meshing Mode** not selected

The following topics describe the extension FluentDemo2:

**Defining FluentDemo2** 

Defining Functions for FluentDemo2

Reviewing the Wizard in FluentDemo2

# **Defining FluentDemo2**

The XML extension definition file FluentDemo2.xml follows.

```
<extension name="FluentDemo2" version="1">
    <guid>49B358FB-EAFB-4678-BBBD-82E949B46F70/guid>
<author>Ansys, Inc.</author>
<description>This extension demonstrates how to create a wizard in Fluent (.cas file import).</description>
<script src="main.py" />
<interface context="Fluent">
 <images>images</images>
</interface>
 <uidefinition>
 <layout name="ResultsView">
   <component
   name="Title"
   topAttachment=""
   topOffset="10"
   leftAttachment=""
   leftOffset="10"
   rightAttachment=""
   rightOffset="10"
   bottomOffset="10"
   bottomAttachment="Properties"
   widthType="Percentage"
   width="100"
   heightType="FitToContent"
   height="200"
   componentType="startPageHeaderComponent" />
   <component
   name="Properties"
   topAttachment="Title"
   leftAttachment=""
   leftOffset="10"
   rightAttachment=""
   rightOffset="10"
   bottomAttachment="Report"
   bottomOffset="10"
   widthType="Percentage"
   width="100"
   heightType="Percentage"
   height="20"
   componentType="propertiesComponent" />
   <component
   name="Report"
   topAttachment="Properties"
   leftAttachment="
   leftOffset="10"
   rightAttachment=""
   rightOffset="10"
   bottomAttachment="Chart"
   bottomOffset="10"
   widthType="Percentage"
   width="100"
   heightType="Percentage"
   height="40"
   componentType="helpContentComponent" />
   <component
   name="Chart"
   topAttachment="Report"
   leftAttachment="
   leftOffset="10"
   rightAttachment=""
   rightOffset="10"
   bottomAttachment="Submit"
   bottomOffset="10"
   widthType="Percentage"
   width="100"
   heightType="Percentage"
   height="40"
   componentType="chartComponent" />
```

```
<component
   name="Submit"
   topAttachment="Chart"
   leftAttachment=""
   leftOffset="10"
   rightAttachment=""
   rightOffset="10"
   bottomAttachment=""
   bottomOffset="10"
   widthType="Percentage"
   width="100"
   heightType="FitToContent"
   height="50"
   componentType="buttonsComponent" />
 </layout>
</uidefinition>
<wizard name="Simple Analysis (Fluent Demo 2)" version="1" context="Fluent" icon="wizard_icon">
 <description>Generate the mesh and solve the analysis.</description>
 <step name="Analysis" caption="Analysis" version="1">
  <callbacks>
   <onrefresh>ImportModel</onrefresh>
   <onupdate>CreateAnalysis</onupdate>
  </callbacks>
  <step name="Results" caption="Results" version="1" layout="ResultsView@FluentDemo2">
  <callbacks>
   <onrefresh>CreateReport</onrefresh>
  </callbacks>
  </step>
</wizard>
</extension>
```

Definitions follow for elements in the XML extension definition file FluentDemo2.xml.

### **Custom Layout Definition**

The element <uidefinition> defines a custom layout named ResultsView for the Fluent wizard. It is made up of five custom components: Title, Properties, Report, Chart, and Submit. The Properties, Report, and Chart components are views, and the Submit component is a button.

#### **Wizard Definition**

The element <wizard> has the attribute name set to Simple Analysis (Fluent Demo 2).

#### **Step Definition**

The element <step> defines each step in the wizard. This wizard has two steps: Analysis and Results. For each step:

The attribute layout specifies the custom layout to apply to the step. For the step Results, layout is set to ResultsView@FluentDemo2. The custom layout ResultsView is defined in the element <uidefinition>.

- Any element property</pr>
   specifies a property and property attributes for the step.
  - For the step Analysis, the property velocity is defined. The attribute caption is set to Velocity.

For the step Results, the property pressure is defined. The attribute caption is set to Pressure.

The element <callbacks> specifies callbacks to functions defined in the script main.py
 For more information, see ??? (p. 13).

### **Defining Functions for FluentDemo2**

The IronPython script main.py follows. This script defines all functions executed by the callbacks in the XML extension definition file:

- The function ImportModel is invoked by the callback <onrefresh> in the step Analysis.
- The function CreateAnalysis is invoked by the callback <onupdate> in the step Analysis.
- The function CreateReport is invoked by the callback <onrefresh> in the step Results.

```
def ImportModel(step):
         tui = ExtAPI.Application.ScriptByName("TUI")
         tui.SendCommand("""/file/read-case "{0}" """.format(System.IO.Path.Combine(
ExtAPI.Extension.InstallDir, "final.cas")))
         tui.SendCommand("""/display/mesh ok""")
         tui.SendCommand("""/display/views/restore-view right""")
def CreateAnalysis(step):
         tui = ExtAPI.Application.ScriptByName("TUI")
         res = System.IO.Path.GetTempFileName()
         img = System.IO.Path.GetTempFileName()+".jpg"
         curve = System.IO.Path.GetTempFileName()
         tui.SendCommand("""/define/boundary-conditions/velocity-inlet inlet no no yes yes no {0} no 0""".format(ster
         tui.SendCommand("""/solve/initialize/hyb-initialization""", True)
         tui.SendCommand("""/solve/iterate 50""", True)
         tui.SendCommand("""/display/mesh ok""", True)
         tui.SendCommand("""/display/views/restore-view right""", True)
         tui.SendCommand("""/display/set/contours/filled-contours? yes""", True)
         tui.SendCommand("""/display/set/contours/surfaces wall inlet outlet ()""", True)
         tui.SendCommand("""/display/contour pressure 0 2""", True)
         tui.SendCommand("""/display/save-picture "{0}" """.format(img), True)
         tui.SendCommand("""/surface/rake-surface rake-5 0 0 0 0 1 50""
         tui.SendCommand("""/plot/plot-direction 0 0 1""", True)
         tui.SendCommand("""/plot/plot yes "{0}" yes yes no no pressure yes 0 0 1 rake-5 ()""".format(curve), True)
         \label{tui.SendCommand("""/report/surface-integrals/area-weighted-avg inlet () pressure yes "{0}" "".format(res), and the content of the co
         val = 0.
         n = 0;
         with System.IO.StreamReader(res) as reader:
                           line = reader.ReadLine()
                           if line!=None:
                                    n+=1
                  str = line.Substring(32)
```

```
val = System.Double.Parse(str, System.Globalization.CultureInfo.InvariantCulture)
   x=[]
   y=[]
   n = 0;
   with System.IO.StreamReader(curve) as reader:
        while n!=5:
           line = reader.ReadLine()
           if line!=None:
               n+=1
        while not line.StartsWith(")"):
            tab = line.Split("\t")
           x.Add(System.Double.Parse(tab[0], System.Globalization.CultureInfo.InvariantCulture))
           y.Add(System.Double.Parse(tab[1], System.Globalization.CultureInfo.InvariantCulture))
           line = reader.ReadLine()
   System.IO.File.Delete(res)
   System.IO.File.Delete(curve)
   step.NextStep.Properties["pressure"].Value = val
   step.NextStep.Attributes["img"] = img
   step.NextStep.Attributes["x"] = x
   step.NextStep.Attributes["y"] = y
def CreateReport(step):
   graph = step.UserInterface.GetComponent("Chart")
   graph.Title("Static Pressure")
   graph.ShowLegend(False)
   graph.Plot(step.Attributes["x"], step.Attributes["y"], key="Static Pressure", color='g')
   help = step.UserInterface.GetComponent("Report")
   help.SetHtmlContent(ExtAPI.Extension.InstallDir, """<center><img style="height:250px"
src="{0}"></center>""".format(step.Attributes["img"]))
```

### **Reviewing the Wizard in FluentDemo2**

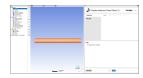
This section reviews the wizard interface and the processes performed in each step of the wizard Simple Analysis (Fluent Demo 2).

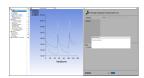
# **Reviewing the Analysis Step**

The first step is **Analysis**, which imports the CAS file and creates the analysis. It requires that a value be specified for the property **Velocity**.

In this step:

- The callback <onrefresh> invokes the function ImportModel. This function imports the file final.cas and displays the model in the Mesh view.
- The callback <onupdate> invokes the function CreateAnalysis when the step's Next button
  is clicked. This function sets up the analysis.
  - The element property
     defines the property Velocity
     It is shown in Fluent as a float field with a default value of 3. You can enter the desired value. In this example, a value of 2 is entered.
  - Also defined are additional actions to be performed as part of the analysis.
  - Intermediate results are shown dynamically in the Scaled Residuals window as the analysis is performed.





## **Reviewing the Results Step**

The second step is **Results**, which creates a report of the results.

In this step:

- The callback <onrefresh> invokes the function CreateReport.
- The attribute <layout> specifies the custom layout to apply to the results. Custom layouts are defined in the element <uidefinition>.
- The wizard takes the results obtained and displays them:
  - The element cproperty> defines the property PressureDrop. This calculated value is displayed in Fluent as a ready-only field.
  - The static pressure results are shown in the **Graph** and **Contours** views.

