PyTecplot and Tecplot 360 2017 R1

How to Enhance Visualization Productivity with Python

NASA Ames - March 30, 2017
John Goetz, PhD, Developer
Alan Klug, VP Customer Development

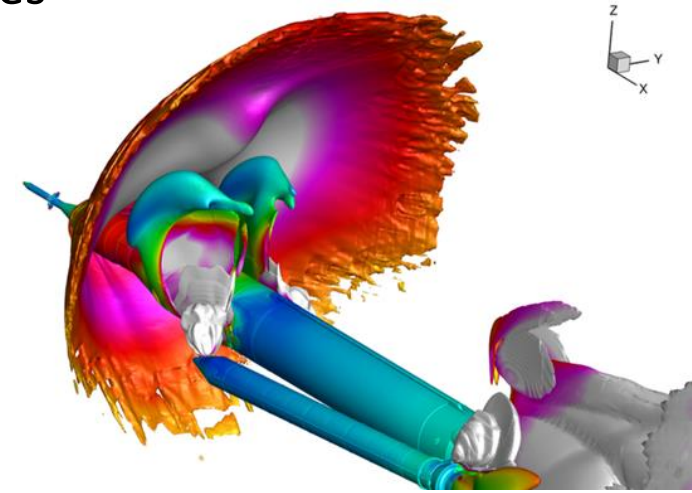


Contents

- Overview of Tecplot 360 EX 2017 R1
- Purpose of **PyTecplot**
- API Overview & Basic Usage
- Case Study (*Cp plot* with integration using **CFDA**)
- Other Examples & Advanced Usage
- Documentation & Support
- Future Work

Tecplot 360 EX 2017 R1 - What's New?

- Chorus included with 360 EX
- Access remote data with SZL Server
- Loader enhancements and returning features
- Automate with an all new Python API



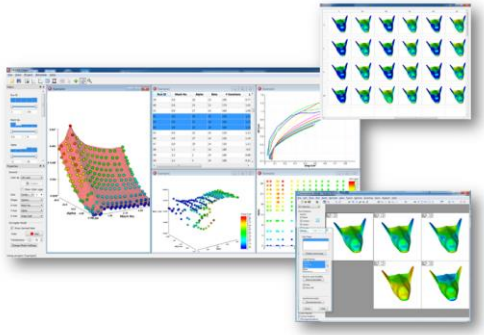
Tecplot 360 EX 2017 R1 - What's New?

BIG DATA

- SZL Technology
- SZL Server
- Under the hood improvements
- In situ viz (in development)

MANY SIMULATIONS

- Chorus



AUTOMATION

- PyTecplot

Tecplot 360 EX 2017 R1 - What's New?

- ANSYS, Abaqus, CFX, CGNS, PLOT3D, FLOW-3D, and NASTRAN loader improvements
- Variable Combining – merge variable names
- New slice tools - define with three points
- Batch SZL conversion
- Enhanced Cart3D support and TRIX reader
- 3D Mouse Support (Windows only)
- Getting Started Manual updated

Purpose of PyTecplot

- Enable complex operations in batch mode
- Allow for manipulating data
- Better programmatic flow control
- Much faster than Macro Language for data access and manipulation

Why Python?

- Popular and well known programming language
- Interoperates with C/C++ libraries (Tecplot's SDK) easily
- Massive wealth of built-in and third-party libraries
- *Python is just plain fun!*

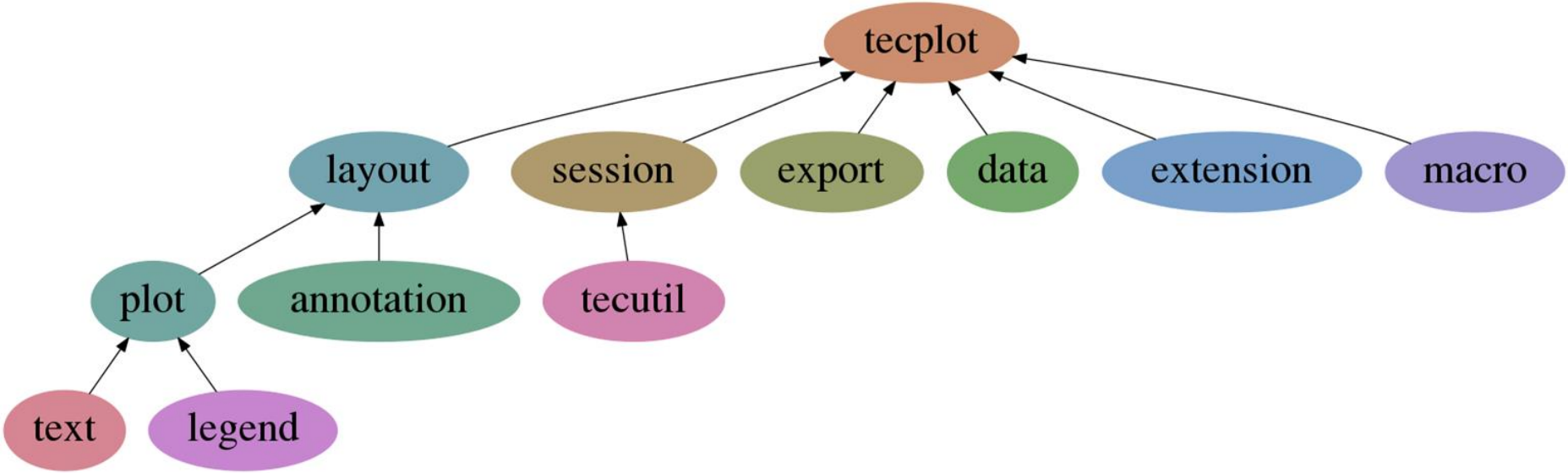
Why not extend the **macro** language?

- Limited syntax and difficult to write from scratch

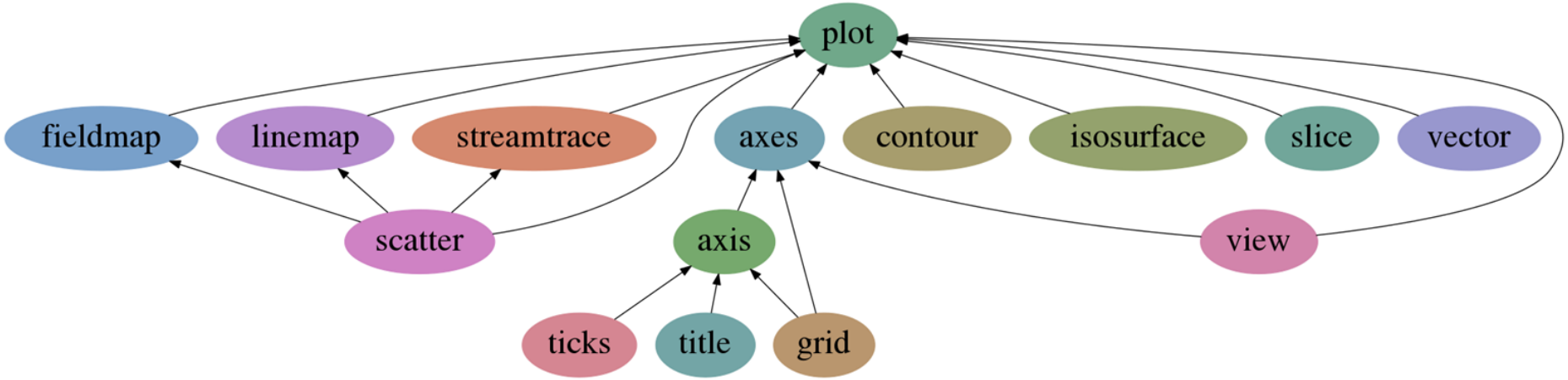
Python, Again!?

- There was a python API released in 2008
  - Thin wrapper around the ADK's "TecUtil Layer" (1000+ functions)
  - Largely state-based
  - Compiled and linked against a specific Python shipped with 360
  - Users had little control over extra packages (e.g. Numpy)
- **PyTecplot** is significantly different
  - Object oriented, organized into namespaces
  - Mostly stateless
  - Can be run in your favorite IDE
  - Not tied to a particular installation of Python
  - Does not run from within 360

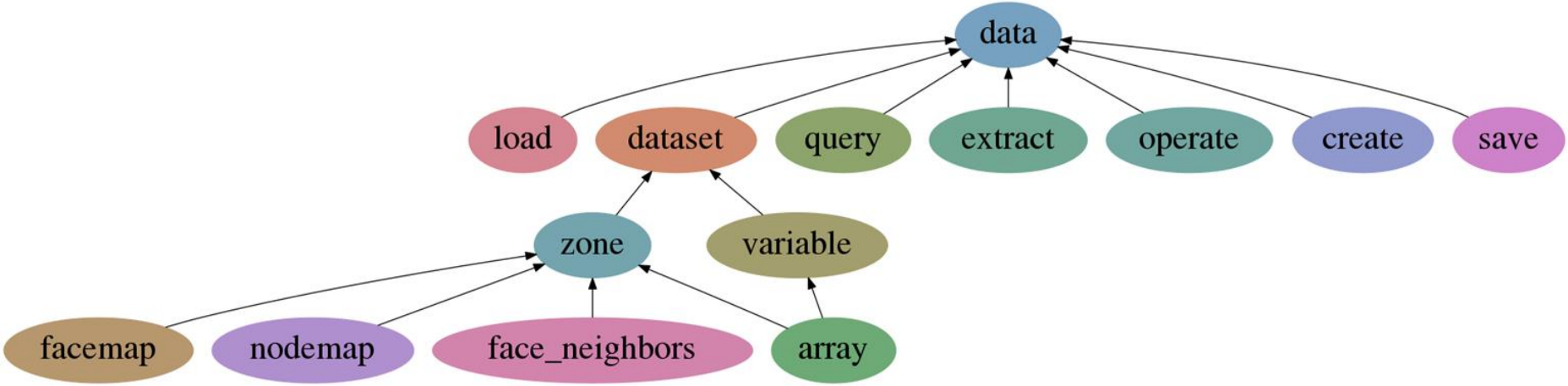
API Overview



API Overview: Plot Submodule



API Overview: Data Submodule



PyTecplot Installation

- Requirement:
  - Tecplot 360 2017 R1 or later
  - **64-bit Python 2.7, 3.4+**
- PyTecplot is available from Python's official package index servers:
  - `pip install pytecplot`
  - **A**lso available under Tecplot 360's installation directory
- Tecplot 360's "Interprocess" library is loaded at run-time
  - Directories must be visible to the loader
    - Windows: `PATH`, Linux: `LD_LIBRARY_PATH`, OSX: `DYLD_LIBRARY_PATH`
- Detailed notes can be found in the documentation

Basic Usage

```
# loads the Tecplot interprocess dynamic library
import tecplot as tp

# for convenience
from tecplot.constant import *

# acquires a Tecplot Batch license and loads the data
dataset = tp.data.load_tecplot('/path/to/my/data.plt')

# change frame to show 3D field plot with boundary surfaces turned on
frame = tp.active_frame()
plot = frame.plot(PlotType.Cartesian3D)
plot.activate()
plot.show_contour = True
for fmap in plot.fieldmaps:
    fmap.surfaces.surfaces_to_plot = SurfacesToPlot.BoundaryFaces

tp.export.save_png('myimage.png', width=600, supersample=3)
```

Cp Plot Case Study

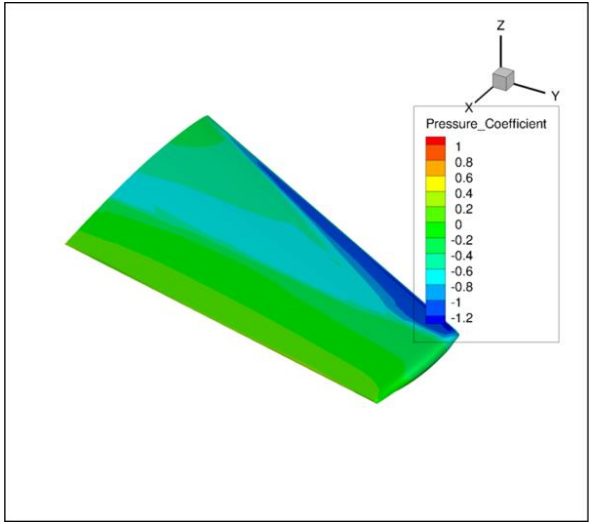
- Create surface slice at some point along wing
- Adjust plot to show typical **Cp plot**
- Integrate and include result as a text annotation

Cp Plot Case Study - Loading Data

```
import os
import tecplot
from tecplot.constant import *

examples_dir = tecplot.session.tecplot_examples_directory()
datafile = os.path.join(examples_dir, 'OneraM6wing',
                        'OneraM6_SU2_RANS.plt')
dataset = tecplot.data.load_tecplot(datafile)

# get handle to frame, change plot type, show the wing
frame = tecplot.active_frame()
plot = frame.plot(PlotType.Cartesian3D)
plot.activate()
plot.show_contour = True
plot.contour(0).variable = dataset.variable('Pressure_Coefficient')
tecplot.export.save_png('wing.png', 600, supersample=3)
```



Cp Plot Case Study - Extract Slice

```
# extract an arbitrary slice from the surface data on the wing
extracted_slice = tecplot.data.extract.extract_slice(
    origin=(0, 0.25, 0),
    normal=(0, 1, 0),
    slicesource=SliceSource.SurfaceZones,
    dataset=dataset)

extracted_slice.name = 'Quarter-span C_p'

# get x from slice
extracted_x = extracted_slice.values('x')

# copy of data as a numpy array
x = extracted_x.as_numpy_array()

# normalize x
xc = (x - x.min()) / (x.max() - x.min())
extracted_x[:] = xc
```

Cp Plot Case Study - Linemaps

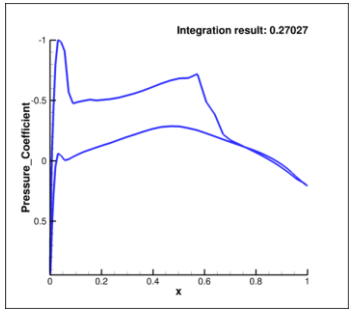
```
# switch plot type in current frame, clear plot
plot = frame.plot(PlotType.XYLine)
plot.activate()
plot.delete_linemaps()

# create line plot from extracted zone data
cp_linemap = plot.add_linemap(
    name=extracted_slice.name,
    zone=extracted_slice,
    x=dataset.variable('x'),
    y=dataset.variable('Pressure_Coefficient'))

# set style of linemap plot
cp_linemap.line.color = Color.Blue
cp_linemap.line.line_thickness = 0.8
cp_linemap.y_axis.reverse = True

# update axes limits to show data
plot.view.fit()
```

Cp Plot Case Study




```
# integrate over the normalized extracted slice values
# notice we have to convert zero-based index to one-based for CFDAnalyzer
tecplot.macro.execute_extended_command('CFDAnalyzer4', '''
    Integrate
    VariableOption='Average'
    XOrigin=0 YOrigin=0
    ScalarVar={scalar_var}
    XVariable=1
'''.format(scalar_var=dataset.variable('Pressure_Coefficient').index + 1))

# get integral result from Frame's auxiliary data
total = float(frame.aux_data['CFDA.INTEGRATION_TOTAL'])

# overlay result on plot in upper right corner
frame.add_text('Integration result: {:.5f}'.format(total), (60,90))

# export image of pressure coefficient as a function of x/c
tecplot.export.save_png('wing_pressure_coefficient.png', 600, supersample=3)
```

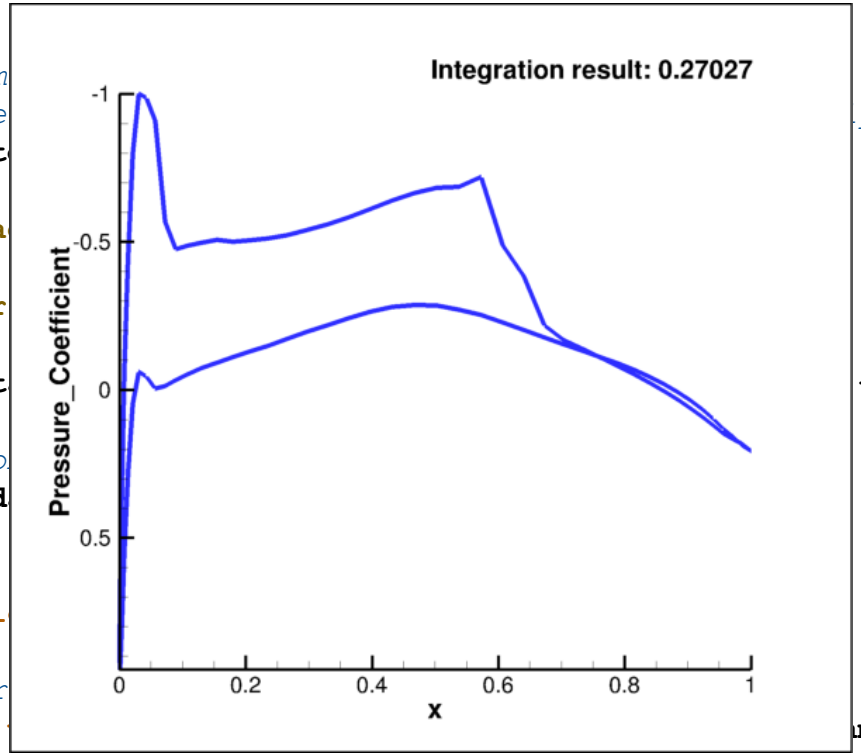
Cp Plot Case Study

```
# integrate over the norm
# notice we have to conve                                          yzer
tecplot.macro.execute_ext
    Integrate
    VariableOption='Avera
    XOrigin=0 YOrigin=0
    ScalarVar={scalar_var
    XVariable=1
'''.format(scalar_var=dat                                          + 1))

# get integral result fro
total = float(frame.aux_d

# overlay result on plot
frame.add_text('Integrati

# export image of pressur
tecplot.export.save_png('                                          mple=3)
```



Other Examples & Advanced Usage

- Many **examples** can be found throughout the documentation
- In the 360 installation under **`pytecplot/examples`**
- In **ZIP** file downloaded from *PyPI* servers

Examples: Streamtraces

18_streamtrace_ribbon.py – /tpw088/john/pytecplot/pytecplot/examples – [pytecplot] – Geany

File Edit Search View Document Project Build Tools Help

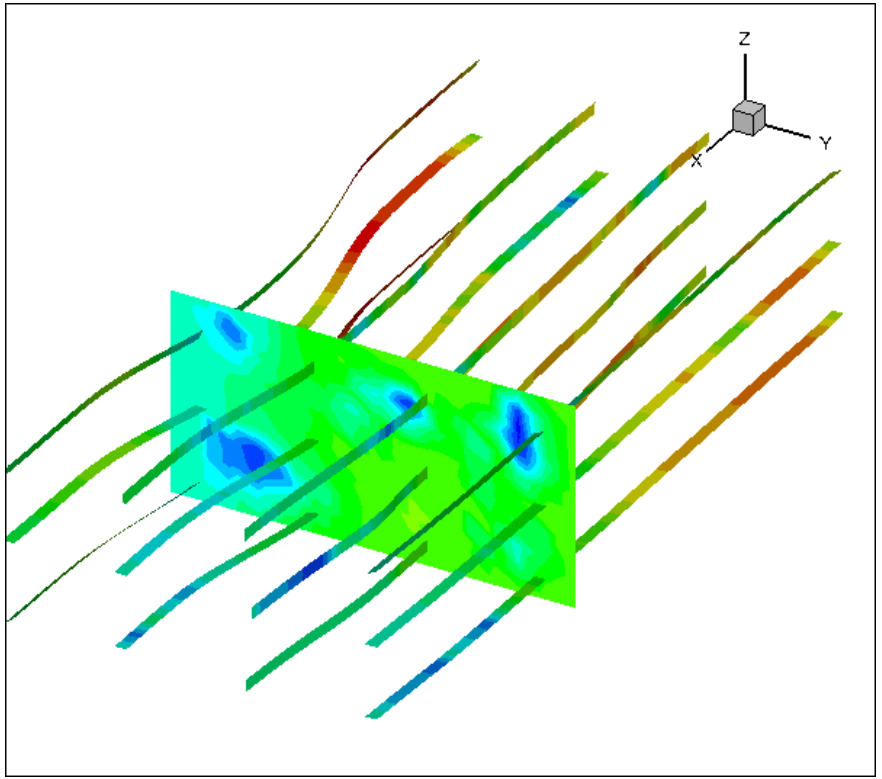
Symbols Documents Project

pytecplot/pytecplot/examples

- 00_hello_world.py
- 01_load_layou...ave_image.py
- 02_exception_handling.py
- 03_slices_along_wing.py
- 04_spherical_harmonic.py
- 05_timestep_delta.py
- 06_ndconvolution.py
- 07_execute_equation.py
- 08_save_data.py
- 09_probe_at_position.py
- 10_contour_filtering.py
- 11_linemaps.py
- 12_slices.py
- 13_wing_slices.py
- 14_isosurface.py
- 15_wing_mach_iso.py
- 16_print_zone_info.py
- 17_streamtrace_line.py
- 18_streamtrace_ribbon.py
- 19_streamtrace_2D.py
- 20_legend_line.py
- 21_legend_contour.py

```
1  import os
2  import tecplot
3  from tecplot.constant import *
4  import numpy as np
5
6  examples_dir = tecplot.session.tecplot_examples_directory()
7  datafile = os.path.join(examples_dir, '3D_Volume', 'ductflow.plt')
8  dataset = tecplot.data.load_tecplot(datafile)
9
10 frame = tecplot.active_frame()
11 frame.plot_type = tecplot.constant.PlotType.Cartesian3D
12
13 plot = frame.plot()
14 plot.contour(0).variable = dataset.variable('P(N/m2)')
15 plot.contour(0).levels.reset_to_nice()
16 plot.contour(0).legend.show = False
17
18 plot.vector.u_variable = dataset.variable('U(M/S)')
19 plot.vector.v_variable = dataset.variable('V(M/S)')
20 plot.vector.w_variable = dataset.variable('W(M/S)')
21
22 # Goal: create a grid of 12 stream trace ribbons on a slice
23 x_slice_location = .79
24 y_start = .077
25 y_end = .914
26 z_start = .052
27 z_end = .415
28
29 num_left_right_slices = 4  # Must be >= 2
30 num_top_bottom_slices = 3  # Must be >= 2
31
32
33 # Show and add a slice.
34 plot.show_slices = True
35 slice_0 = plot.slice(0)
36
37 # Set the x position of the slice.
38 slice_0.origin = [x_slice_location, 0, 0]
39
40 slice_0.contour.show = True
41
42 plot.show_streamtraces = True
43 streamtraces = plot.streamtraces
44 streamtraces.show_paths = True
45
```

line: 31 / 56 col: 0 sel: 0 INS SP mode: CRLF encoding: UTF-8 filetype: Python scope: unknown



Examples

```
00_hello_world.py
01_load_layout_save_image.py
02_exception_handling.py
03_slices_along_wing.py
04_spherical_harmonic.py
05_timestep_delta.py
06_ndconvolution.py
07_execute_equation.py
08_save_data.py
09_probe_at_position.py
10_contour_filtering.py
11_linemaps.py
12_slices.py
13_wing_slices.py
14_isosurface.py
15_wing_mach_iso.py
16_print_zone_info.py
17_streamtrace_line.py
18_streamtrace_ribbon.py
19_streamtrace_2D.py
20_legend_line.py
21_legend_contour.py
interpolation/
jupyter/
working_with_datasets/
```

Working with Datasets: FE Triangles

fe_triangles1.py – /tpw088/john/pytecplot/pytecplot/examples/working_with_datasets – [pytecplot] – Geany

File Edit Search View Document Project Build Tools Help

Symbols Documents Project

pytecplot/pytec..._with_datasets
- create_cell_ce...d_variable.py
- face_neighbors.py
- fe_triangles1.py
- fe_triangles2.py
- fe_triangles3.py
- fe_triangles4.py
- fe_triangles5.py
- ordered1.py
- ordered2a.py
- ordered2b.py
- ordered3.py
- polygons1.py
- polygons1b.py
- polygons2.py
- polyhedrons1.py
- polyhedrons1b.py
- polyhedrons2.py

```
"""Triangle Finite-element Data Creation (Part 1)

This script creates a quad of two triangles from scratch using the PyTecplot
low-level data creation interface. The general steps are:

1. Setup the data
2. Create the tecplot dataset and variables
3. Create the zone
4. Set the node locations and connectivity lists
5. Set the (scalar) data
6. Write out data file
7. Adjust plot style and export image

The data created looks like this::

    Node positions (x,y,z):

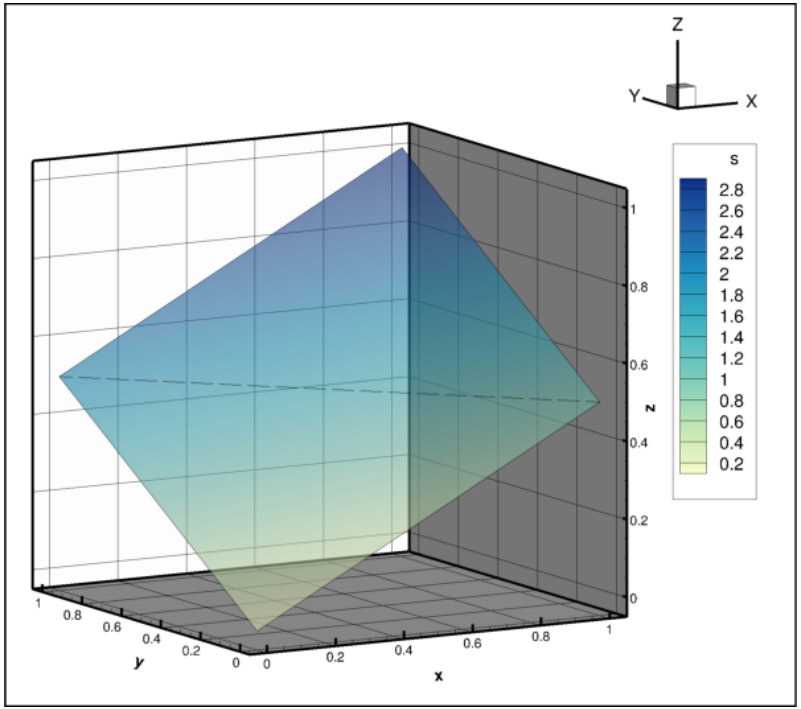
                    (1,1,1)
                   3
                  / \
                 /   \
     (0,1,.5)  2-----1  (1,0,.5)
                \   /
                 \ /
                  0
                    (0,0,0)

Breaking up the two triangular elements, the faces look like this. Notice the
first element (index: 0) is on the bottom::

    Element 1 Faces:
                      *
       (nodes 3-2)  1 / \ 0  (nodes 1-3)
                     /   \
                    *-----*
                       2
                         (nodes 2-1)

    Element 0 Faces:
                         (nodes 1-2)
                       1
                      *-----*
                      \   /
       (nodes 2-0)  2 \ / 0  (nodes 0-1)
                        *
```

line: 1 / 136 col: 0 sel: 0 INS SP mode: LF encoding: UTF-8 filetype: Python scope: unknown



Advanced Examples: "Working with Datasets"

```
fe_triangles1.py  ordered1.py   polygons1.py   polyhedrons1.py
fe_triangles2.py  ordered2a.py  polygons1b.py  polyhedrons1b.py
fe_triangles3.py  ordered2b.py  polygons2.py   polyhedrons2.py
fe_triangles4.py  ordered3.py
fe_triangles5.py
```

Hints for Using PyTecplot Effectively

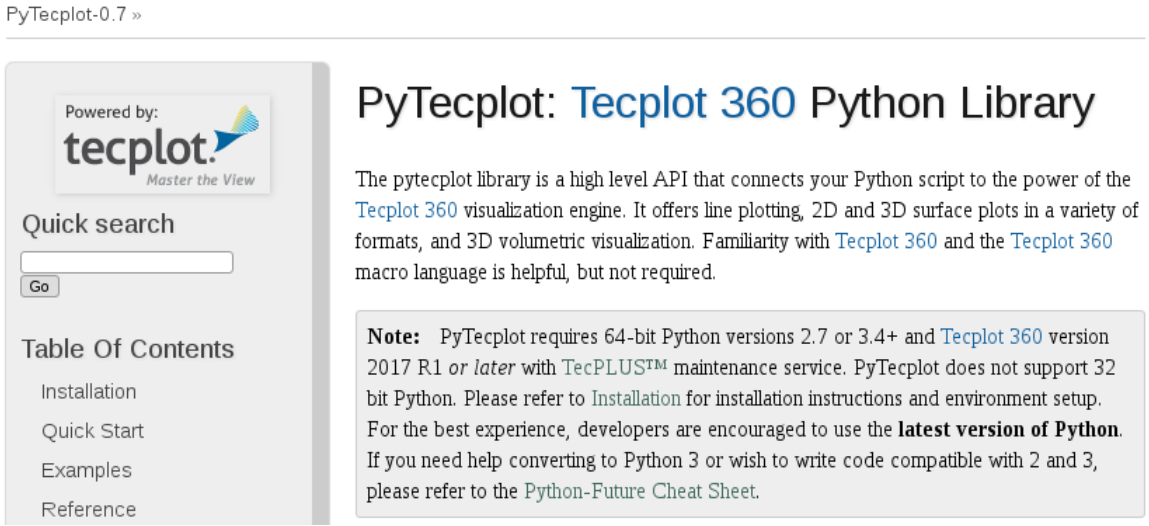
- Side-effects have been minimized
  - Can do most operations on an inactive frame
- Copy out to Numpy arrays when manipulating data
  - Alternative: `tecplot.data.execute_equation()`
- Let Python optimize runtime with: `python -O`
- We push updates of **PyTecplot** to *PyPI* frequently
- PyTecplot's HTML documentation is a great resource

Future Work

- Driving an Interactive Instance of Tecplot 360
- Recording **PyTecplot** from Tecplot 360
- Converting Macros into Python Scripts

Documentation and Support

- Documentation is found at: tecplot.com/docs/pytecplot
- General support inquiries: support@tecplot.com



Documentation and Support

- Alan Klug - VP Customer Development a.klug@tecplot.com
- John Goetz - PyTecplot Developer j.goetz@tecplot.com