A Quick Macro Panel Suite to Augment the X-Y Line Mapping Interface of Tecplot

Stephen J. Alter
Williamsburg, VA
altertalk@cox.net

April 2, 2014

Contents

List of Tables

Abstract

The Tecplot software offered by Tecplot Incorporated can be used for data analysis as well as data generation. Making use of the software for analysis can require the use of algorithms that require many repeated operations, and in some cases many different operations. A macro language was developed to augment the Tecplot software by enabling repetitive operations. From its inception though, the macro language has become a command rich coding structure for the development of complex algorithms. Some commonly used complex and repetitive algorithms, within the aerospace community, have been incorporated into a dynamic interface called the Quick Macro Panel (QMP), in Tecplot. The QMP is a widget or dialog that offers single button initiation of a macro sequence, and is available under the "Scripting..." menu of the main interface. By utilizing the QMP, the user interface becomes more robust for the analysis of solution quality and development of animation. This document serves as a guide to the algorithms and macros developed for a QMP suite of macros, to describe their operation and applicability. Also contained within this document is the macro coding and the algorithms used, as an illustration of what is possible within the Tecplot macro framework.

Chapter 1

Introduction

The original file provided by Tecplot Inc. for the Quick Macro Panel (QMP) has several different types of macros. These are provided by Tecplot Inc. to illustrate how the QMP is used. But in their original form, they were not user friendly. These macros do not consider the possible incorrect entries for user input that guide the operation of the macros. Additionally, these macros are just examples for the general user. They do not necessarily address the specific needs of an organization with respect to the operation of Tecplot, but rather serve as examples of how to use the QMP and how it can be useful in the general operation of Tecplot.

As noted, the aerospace community has very specific requirements for data analysis, as well as data generation. Hence, the QMP is one mechanism that can be used to customize the Tecplot operational environment towards efficient and effective analysis and data generation. However, use of the techniques requires a manual of sorts. This document is a guide to the macros contained herein, serves as a registration of the technology developed for data analysis and generation, and identifies a few ways the QMP can be used to augment the research and development environment.

This guide is separated into two more chapters. The first addresses the use of the animation augmentation to the X-Y line mapping. The following chapter addresses macros developed to enhance the operational environment typically used in generating animations. There is a final set of sections which cover macros that do not appear in the QMP, but are used by the QMP as subroutines. Though no examples are offered in this document for the results of the macros in the QMP, the the user is strongly urged to employ these macros to obtain a firm understanding of the powerful techniques the algorithms present.

Chapter 2

X-Y Line Map Animation

Most of the built-in functionality for animation in Tecplot is honed for two and three dimensional data sets. Animation can be performed by marching through a set of zones, changing indicies of an ordered zone, changing contour levels of a plot, and others. But similar functionality does not exist for X-Y line maps, the most general analysis plot of data within Tecplot. X-Y line maps can be animated similarly to zones by marching through them, however, there are no mechanisms to march through zones or indicies that comprise a line or set of symbols displaying data.

The following set of macros are designed to offer animation of X-Y line mappings by marching through zones defined in the mapping. Assumed in this set up is the format of the line mappings; **a set of XY line mappings using the structure that each mapping is a separate variable tied to all other line maps by referencing the same root zone**. Thus, by looping on the zone, an animation is made. Since no stepping through zones is also possible within the Tecplot user interface for X-Y line maps, this functionality is also inherent to the macros that are listed below.

To begin the process of animation, an initial zone has to be set so that incremental changes to the zone number can be made. The variable `|ThisZone|` is used as the reference for all zones. By changing the value of this zone, and using the macro function to update all X-Y line mappings SETXYMAPZONE, the user has the ability with the press of a single button in the QMP to advance or reverse the marching direction of the referenced zone for all X-Y line maps. Thus at start up of the QMP, the `|ThisZone|` variable is initialized as is the increment for animations. In addition to the specific zone for all mappings, another variable `|Detla|` controls the increment at which marching occurs. These two variables in essence couple all the animation macros together; they serve as the link that enables various animations to be performed.

```
$!VarSet |ThisZone| = 1
$!VarSet |Delta| = 1
```

With the referencing zone and increment set, the list in table 2.1 of animation macros are used to augment the animation functionality of Tecplot. In the following sections, each macro is detailed.

Macro Name	Description
Set X-Y LineMap Zone	Sets the `|ThisZone|` variable.
Zone Increment	Sets the amount to change the `|ThisZone|` variable.
Advance Zone (+)	Advances the displayed zone.
Reverse Zone (-)	Reverses the direction of the displayed zone.
Animate All	Animates the entire set of line mappings for screen or file output.
Animate Forward [+]	Animates the set of zones from the current `|ThisZone|` reference point to the end.
Animate Reverse [-]	Animates the set of zones from the current `|ThisZone|` reference point to the beginning.

Table 2.1: Table of animation routines in the quick macro panel suite.

2.1 Set X-Y LineMap Zone

The following macro is designed to set the zone referenced in all line maps established for the XY line plot type. This macro enables the user to change the current location from any zone to any other zone without having to select the zones in the `Mapping Style` interface. It also enables the user to reset the starting zone to the beginning without having to reverse an animation or go to the end zone for the same reason.

```
$!MacroFunction
  Name = "Set X-Y LineMap Zone"
  ShowInMacroPanel = Yes

  $!VarSet |NewZone| = -1

  $!While |NewZone| == -1

    $!PromptForTextString |NewZone|
      Instructions = "Enter zone for all X-Y LineMaps:"

    $!If |NewZone| < 0
      $!Pause "Cannot set zone to less than 0 (last zone)!"
      $!VarSet |NewZone| = -1
    $!EndIf

    $!If |NewZone| > |NumZones|
      $!Pause "Cannot set zone to more than last zone (|NumZones|)!"
      $!VarSet |NewZone| = -1
    $!EndIf

  $!EndWhile

  $!VarSet |ThisZone| = |NewZone|

  $!RunMacroFunction "SetXYMapZone" (|ThisZone|)

$!EndMacroFunction
```

2.2 Zone Increment

The following macro is designed to set the zone increment used in all other animation macros of this suite, for all line maps established for the XY line plot type. This macro enables the user to use the `Advance Zone (+)` and the `Reverse Zone (-)` to be used a s toggles between two different zones for solution comparison. It also enables the user to speed up the animation by using a coarser representation of the data and not using every single zone for an animation.

```
$!MacroFunction
  Name = "Zone Increment"
  ShowInMacroPanel = Yes

  $!VarSet |Delta| = -1

  $!While |Delta| == -1

    $!PromptForTextString |Delta|
      Instructions = "Enter zone increment for all XY line animations:"

    $!If |Delta| < 0
      $!Pause "Cannot set zone increment to less than 0 (Max-1)!"
      $!VarSet |Delta| = -1
    $!EndIf

    $!If |Delta| > |NumZones|
      $!Pause "Cannot set zone to more than the number of zones (|NumZones|)!"
      $!VarSet |Delta| = -1
    $!EndIf

    $!If |Delta| != -1
      $!If |Delta| == 0
        $!VarSet |Delta| = (|NumZones|-1)
      $!Endif
    $!EndIf

  $!EndWhile

$!EndMacroFunction
```

2.3 Advance Zone (+)

As a functionality of generic animation in Tecplot, the user is able to incrementally step through attributes that can be modified sequentially. However, this is not possible within the XY line maps either. Thus, the following macro was written to enable this capability. Using the `|Delta|` variable to define the increment, the referenced `|ThisZone|` variable is increased to march forward through the zone referenced in all mappings displayed. However, this macro also produces a warning when the last zone is reached and the user attempts to march beyond the limit.

```
$!MacroFunction
  Name = "Advance Zone (+)"
  ShowInMacroPanel = Yes

  $!VarSet |ThisZone| += |Delta|

  $!If |ThisZone| > |NumZones|
    $!Pause "Cannot advance past last zone!"
    $!VarSet |ThisZone| -= |Delta|
  $!EndIf

  $!RunMacroFunction "SetXYMapZone" (|ThisZone|)

$!EndMacroFunction
```

2.4 Reverse Zone (-)

As a functionality of generic animation in Tecplot, the user is able to incrementally step through attributes that can be modified sequentially. However, this is not possible within the XY line maps either. Thus, the following macro was written to enable this capability. Using the `|Delta|` variable to define the increment, the referenced `|ThisZone|` variable is decreased to march in reverse through the zone referenced in all mappings displayed. However, this macro also produces a warning when the first zone is reached and the user attempts to march before the limit.

```
$!MacroFunction
  Name = "Reverse Zone (-)"
  ShowInMacroPanel = Yes

  $!VarSet |ThisZone| -= |Delta|

  $!If |ThisZone| < 1
    $!Pause "Cannot reverse before first zone!"
    $!VarSet |ThisZone| += |Delta|
  $!EndIf

  $!RunMacroFunction "SetXYMapZone" (|ThisZone|)

$!EndMacroFunction
```

2.5 Animate All

The following macro is the root macro for the intent of the entire suite; to animates all zones. The intent for this macro was to provide nearly identical capabilities already available for zone animation for XY line map animation. Thus, an animation can be displayed on the screen or to a file. When generating an animation for a file, or a movie, several output formats are available, all of which are provided in this macro. Due to the difficulty in developing an add-on though, some coding has been added to extract the type of format to export an animation. If it fails to find something coherent, the user is prompted.

```
$!MacroFunction
  Name = "Animate All"
  ShowInMacroPanel = Yes
```

To begin this process ask if the animation is to be on screen or written to a file:

```
  $!RunMacroFunction "DisplayOrExport" ()
```

Animate and display and/or export the animation:

```
  $!VarSet |NumFrames| = ((|NumZones|-1)/|Delta|+1)

  $!Loop |NumFrames|

    $!VarSet |CurrZone| = ( (|loop|-1)*|Delta| + 1 )

    $!RunMacroFunction "SetXYMapZone" (|CurrZone|)

    $!If |Output| == 2
      $!ExportNextFrame
    $!EndIf

  $!EndLoop

  $!If |Output| == 2
    $!ExportFinish
  $!EndIf

  $!VarSet |ThisZone| = |NumZones|

$!EndMacroFunction
```

2.6 `Animate Forward [+]`

When evaluating data with incremental changes in the reference zone `|ThisZone|`, it may become necessary to begin an animation at a different location than the first zone for the mappings. This macro offers that capacity, utilizing the output features of section 2.5. The difference here is that the animation starts at the reference zone and moves forward by the increment set in section 2.2 to the last possible zone.

```
$!MacroFunction
  Name = "Animate Forward [+]"
  ShowInMacroPanel = Yes
```

To begin this process ask if the animation is to be on screen or written to a file:

```
  $!RunMacroFunction "DisplayOrExport" ()
```

Animate the XY line maps forward from `|ThisZone|` referenced as the beginning instead of zone one. The difference here from animating all the zones is the number of frames is the number of zones between `|ThisZone|` and the largest zone count for the data set.

```
  $!VarSet |NumFrames| = ( (|NumZones|-|ThisZone|-1)/|Delta|+1 )

  $!Loop |NumFrames|

    $!VarSet |CurrZone| = ( (|loop|-1)*|Delta|+|ThisZone| )

    $!RunMacroFunction "SetXYMapZone" (|CurrZone|)

    $!If |Output| == 2
      $!ExportNextFrame
    $!EndIf

  $!EndLoop

  $!If |Output| == 2
    $!ExportFinish
  $!EndIf

  $!VarSet |ThisZone| = |NumZones|
  $!RunMacroFunction "SetXYMapZone" (|ThisZone|)

$!EndMacroFunction
```

2.7 Animate Reverse [-]

Similarly to the forward animation in section 2.6, this macro enables the animation to begin in the middle or end of all zones and march backward to the beginning by the increment set in section 2.2 by `|Delta|`. The output capabilities of this macro are identical to those of section 2.5.

```
$!MacroFunction
  Name = "Animate Reverse [-]"
  ShowInMacroPanel = Yes
```

To begin this process ask if the animation is to be on screen or written to a file:

```
  $!RunMacroFunction "DisplayOrExport" ()
```

Animate the XY line maps forward from `|ThisZone|` referenced as the beginning instead of zone one. The difference here from animating all or forward is that the number of frames is the number of zones between `|ThisZone|` and the first zone for the data set.

```
  $!VarSet |NumFrames| = ( (|ThisZone|-2)/|Delta| + 1)

  $!Loop |NumFrames|

    $!VarSet |CurrZone| = ( |ThisZone| - (|loop|-1)*|Delta| )

    $!RunMacroFunction "SetXYMapZone" (|CurrZone|)

    $!If |Output| == 2
      $!ExportNextFrame
    $!EndIf

  $!EndLoop

  $!If |Output| == 2
    $!ExportFinish
  $!EndIf

  $!VarSet |ThisZone| = 1
  $!RunMacroFunction "SetXYMapZone" (|ThisZone|)

$!EndMacroFunction
```

Chapter 3

Special Views

The Tecplot software has 4 built-in views available to the user, including the X-Y, X-Z, Y-Z, and a "Default". These views are useful in resetting the plotting region, but do not offer variability. The planar views are fixed, while the default view can be set in the configuration file of Tecplot. However, once the configuration file is loaded, the "Default" view can not be changed.

A common task within Tecplot is to perform an analysis or investigation, and have a view in the plotting area that is desirable, but applied to more recent data. Instead of attempting to recall the operations as would be performed by using a layout or stylesheet, a simple saving of the view can be used. However, there is no mechanism within Tecplot to change the "Default" view. Rather, the user has to save the view characteristics to a stylesheet, and recall the style sheet to restore the view. The process of recalling the view has a similar drawback to reloading a layout file - multiple operations required to perform the recall. Additionally, there are occasions when the use of multiple views is desired. For all these reasons, the following set of macros have been created to provide four additional user defined views for the Tecplot interface.

The views are stored with the STORE VIEW macro which simply requires a number for the view. The succeeding four macros are "push button" operations to recall a specific view based on the number used in the STORE VIEW macro. The views themselves are stored as view-1,2,3,4.sty in the original directory where Tecplot was started. Utilizing this set of macros provides the user with rapid access to views and reloading data for analysis with the Tecplot software.

```
$!MacroFunction
  Name = "Store View"
  ShowInMacroPanel = True

  $!VarSet |View| = 0

  $!While |View| == 0

    $!PromptForTextString |View|
      Instructions = "Enter the view number to be used (1-4): "

    $!RunMacroFunction "CheckIntInput" (|View|)

    $!System "echo \"NumBadChar = |NumBadChar|      View = |View|\" "
    $!If |NumBadChar| > 0
      $!VarSet |View| = 0
    $!EndIf

    $!If |View| > 4
      $!VarSet |View| = 0
    $!EndIf

    $!If |View| < 1
      $!VarSet |View| = 0
    $!EndIf

    $!If |View| == 0
      $!Pause "Error: You must enter a number between 1 and 4!"
    $!EndIf

  $!EndWhile

  $!WriteStyleSheet "view-|View|.sty"
    IncludeContourLevels = True
    IncludeText = True
    IncludeGeom = True
    IncludeStreamPositions = True

$!EndMacroFunction

$!MacroFunction
  Name = "View-1"
  ShowInMacroPanel = True

  $!ReadStyleSheet "view-1.sty"
$!EndMacroFunction

$!MacroFunction
  Name = "View-2"
  ShowInMacroPanel = True

  $!ReadStyleSheet "view-2.sty"
$!EndMacroFunction

$!MacroFunction
  Name = "View-3"
  ShowInMacroPanel = True

  $!ReadStyleSheet "view-3.sty"
$!EndMacroFunction

$!MacroFunction
  Name = "View-4"
  ShowInMacroPanel = True
  $!ReadStyleSheet "view-4.sty"
$!EndMacroFunction
```

Chapter 4

Auxiliary Macros

All the macros in the macro panel of this suite have several segments of code used over and over. To reduce, or factor down, the number of lines needed to implement the capabilities, different routines are listed below but are not included in the selectable button list of the QMP. Macro functions in this set are more abstract and do not necessarily work on only one variable. One note of caution though is that all macros in Tecplot use global variables. None of the variables passed to them nor created inside them are local. Thus they can be referenced elsewhere.

4.1 SETXYMAPZONE

The following macro function is designed to update all XY line mappings to a specified zone. It is a segment that was used throughout the animation routines of this quick macro panel group, and was placed here as a result of factoring the software.

```
$!MacroFunction
  Name = "SetXYMapZone"
  ShowInMacroPanel = Nope

  $!Loop |NumLineMaps|
    $!LineMap [|loop|]
      Assign {Zone = |1|}
  $!EndLoop

  $!RedrawAll

$!EndMacroFunction
```

4.2 DISPLAYOREXPORT

The following macro enables the suite of macros above to output an animation to a file for movie inclusion in other software. If output to a file is requested, the macro requests the user to supply a file name using the Tecplot built-in dialog for selecting a file. If the file provided has a know export file type extension, this macro will detect it and set the output type accordingly. If such an extension is not provided or is unknown, the user will be asked to provide the extension type. If the extension type does not match `avi`, `mp4`, `wmv`, or `rm`, for AVI, Mpeg4, Windows Movie, or Raster Metafile, respectively, the macro will continue to prompt the user until an acceptable extension is provided.

```
$!MacroFunction
  Name = "DisplayOrExport"
  ShowInMacroPanel = No

  $!VarSet |Output| = 0

  $!While |OutPut| == 0

    $!PromptForTextString |Output|
      Instructions = " Animate to:\n   1) Screen\n   2) File\n Choice:"

    $!If |Output| == 1

    $!ElseIf |Output| == 2

      $!PromptForFileName |ExportFile|
        DialogTitle = "Export File Name:"
        FileFilter = "*.[amwr][vpm][i4v]"
        FileMustExist = no
```

Determine the type of output from the file name extension:

```
      $!ExtendedCommand
        CommandProcessorID = 'extendmcr'
        Command = 'STRING.LENGTH "|ExportFile|" NumChar'

      $!VarSet |iChar| = (|NumChar|-3)
      $!ExtendedCommand
        CommandProcessorID = 'extendmcr'
        Command = 'STRING.SUBSTRING "|ExportFile|" |iChar| |iChar| Dot'

      $!If "|Dot|" != "."

        $!VarSet |iChar| = (|NumChar|-2)
        $!VarSet |jChar| = (|NumChar|)
        $!ExtendedCommand
          CommandProcessorID = 'extendmcr'
          Command = 'STRING.SUBSTRING "|ExportFile|" |iChar| |jChar| Extension'

        $!If "|Extension|" == ".rm"
```

```
          $!VarSet |Extension| = "rm"

        $!Else

          $!VarSet |Prompt| = "Enter output animation type based on extension"
          $!VarSet |Prompt| = "|Prompt| (avi, mp4, etc.):"
          $!VarSet |Extension| = ""

          $!While "|Extension|" == ""

            $!PromptForTextString |Extension|
              Instructions = "|Prompt|"

            $!RunMacroFunction "CheckExtension" ()

          $!EndWhile

        $!EndIf

      $!Else

        $!VarSet |iChar| = (|NumChar|-2)
        $!VarSet |jChar| = (|NumChar|)
        $!ExtendedCommand
          CommandProcessorID = ’extendmcr’
          Command = ’STRING.SUBSTRING "|ExportFile|" |iChar| |jChar| Extension’

        $!RunMacroFunction "CheckExtension" ()

        $!While "|Extension|" == ""

          $!PromptForTextString |Extension|
            Instructions = "|Prompt|"

          $!RunMacroFunction "CheckExtension" ()

        $!EndWhile

      $!EndIf

    $!Else

      $!VarSet |Output| = 0

    $!EndIf

  $!EndWhile

  $!If "|Extension|" == "rm"
```

```
    $!VarSet |ExportFormat| = "RasterMetaFile"
  $!ElseIf "|Extension|" == "avi"
    $!VarSet |ExportFormat| = "AVI"
  $!ElseIf "|Extension|" == "mp4"
    $!VarSet |ExportFormat| = "MPeg4"
  $!ElseIf "|Extension|" == "wmv"
    $!VarSet |ExportFormat| = "WMV"
  $!EndIf

  $!If |Output| == 2

    $!ExportSetUp
      ExportFName = "|ExportFile|"
      ExportFormat = |ExportFormat|
      ExportRegion = AllFrames

    $!ExportStart

  $!EndIf

$!EndMacroFunction
```

4.3 CHECKEXTENSION

The following macro function determines the extension type for output to a file:

```
$!MacroFunction
  Name = "CheckExtension"
  ShowInMacroPanel = No

  $!If "|Extension|" == "rm"
  $!ElseIf "|Extension|" == "avi"
  $!ElseIf "|Extension|" == "mp4"
  $!ElseIf "|Extension|" == "wmv"
  $!Else
    $!VarSet |Extension| = ""
  $!EndIf

$!EndMacroFunction
```

4.4 CHECKINTINPUT

In order to enable more user friendly software, software that does not error out or abruptly end without recourse to enable the user to make a different choice, various inputs need to be checked from time to time to ensure they match the type of information expected. The following routine attempts to offer user friendliness by ensuring a numeric entry was provided. This check is done by ensuring that each character in the `|1|` string, which is the only argument received, are integer numbers.

```
$!MacroFunction
  Name = "CheckIntInput"
  ShowInMacroPanel = Nope

  $!ExtendedCommand
    CommandProcessorID = 'extendmcr'
    Command = 'STRING.LENGTH "|1|" NumChar'

  $!VarSet |NumBadChar| = 0

  $!Loop |NumChar|

    $!ExtendedCommand
      CommandProcessorID = 'extendmcr'
      Command = 'STRING.SUBSTRING "|1|" |loop| |loop| Char'

    $!VarSet |BadChar| = 0

    $!If "|Char|" == "0"
    $!ElseIf "|Char|" == "1"
    $!ElseIf "|Char|" == "2"
    $!ElseIf "|Char|" == "3"
    $!ElseIf "|Char|" == "4"
    $!ElseIf "|Char|" == "5"
    $!ElseIf "|Char|" == "6"
    $!ElseIf "|Char|" == "7"
    $!ElseIf "|Char|" == "8"
    $!ElseIf "|Char|" == "9"
    $!Else

      $!VarSet |BadChar| = 1

    $!EndIf

    $!VarSet |NumBadChar| += |BadChar|

  $!EndLoop

$!EndMacroFunction
```