QTPlaut and MatPlaut

User Manual

Beta version

The AUTO solution

interpolating and plotting

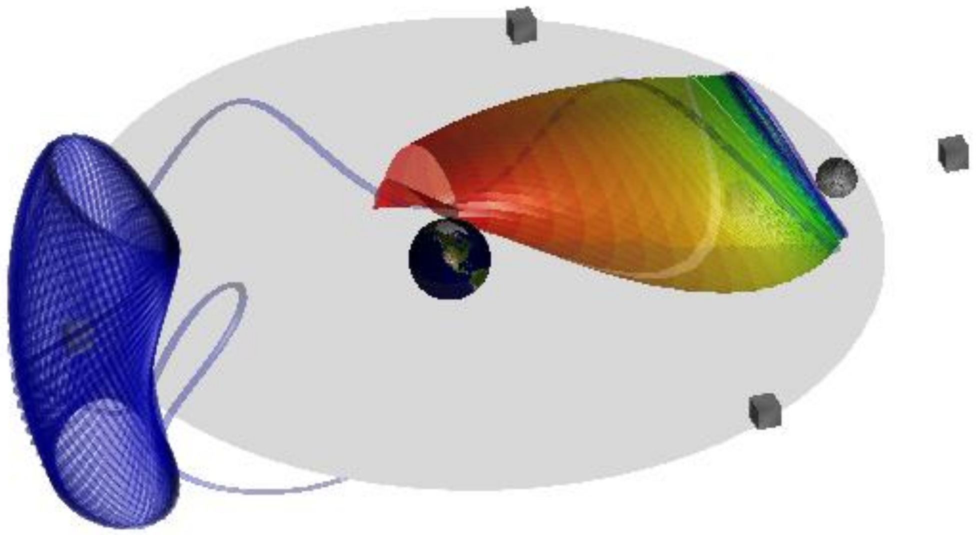
program package

Zhikai Wang, Concordia

July 1, 2008

This project is sponsored by Prof. Eusebius Doedel

Computer Science Department, Concordia University, Canada



Contents

1. Introduction

The QTPlaut is a program that can read multiple AUTO solutions. The solutions can be interpolated by the Lagrange interpolation method. The interpolated results can be put together into a '.mlf' file. The core of QTPlaut is the interpolations or simplifications by GUI. Such operations are possible by the frequent usage of dynamic memory allocation of templated C++ programming.

The MatPlaut is a combination of MATLAB '.m' files that plot the '.mlf' generated by QTPlaut. Again it utilizes object oriented programming techniques. But it is implemented with the MATLAB OO –programming. The great feature of MatPlaut is the transparency effects while plotting multiple manifolds or complex manifold. We may say to present the 2D or 3d projections of one or more AUTO solutions.

An AUTO solution usually contains large quantity of data. To simply them with MATLAB script program doesn't seem to be practical. The plotting functions like 'surf' or 'patch' also have limited data processing ability. So we have to turn to C++ memory management. As long as the main memory of the operation system permits, we have a container to put the reactants. The dynamic memory allocation makes possible the processing of a solution file without knowing in advance the number of branches it contains, how the mesh is set, and even how many solutions we want to put in the same scene. There are many packages that can implement the amazing transparency effects. But the MATLAB plotting functions are the easiest to program with. With them, we save huge quantity time than program with OpenGL or Open Inventor and etc.

2. Installation and setup

The QTPlaut needs openGL/Mesa and QT. Please make sure their libraries are installed in the target machine. The QT version 4.3.4 is used for QTPlaut development. Unzipping the QTPlaut zip/gz to a folder called 'qtplaut'. The folder name can be changed but the execution file built would have the exact name of this folder. After changing directory under this folder xxx, in a Linux terminal or a MS Windows DOS shell , you type 'qmake –project'. Then you open the xxx.pro file and add a line 'QT += opengl', casesensitive. Save the file then type 'qmake' without any arguments. In Linux you can use make, in MS Windows/Visual C++ you can use nmake to build the executable. Its' name will be xxx same as the folder name. Normally you have to add the following lines in your Linux .bashrc.

```
PATH=/usr/local/Trolltech/Qt-4.3.4/bin:$PATH
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib:/usr/local/Trolltech/Qt-4.3.4/lib
export PATH
export LD_LIBRARY_PATH
```

But they are the issues of QT installation while not those of QTPlaut.

The installation of MatPlaut is rather simple. Unzip the files to your working directory. The MATLAB version should be 7.5.0(R2007b), the functioning of earlier versions is not guaranteed.

3. Instructions

3.1 QTPlaut

You can call the QTPlaut program by clicking the executable directly or call from terminal command. The main Window of QTPlaut is divided into two parts. On the left is the plotting panel. On the right is the toolbox panel. The toolbox feature of QT4 makes possible large number of controls in a limited panel space.

3.1.1 Solutions tools, tool box

The solutions tools have four controls

3.1.1.1 Add a solution , button

By clicking this button, you can be navigated through a window to locate an AUTO solution file.

3.1.1.2 Uploaded solutions , combo box

You can load multiple solutions. All loaded solutions can be show in this combo box. Each solution has a unique ID which is not shown explicitly.

For each branch, only the branch header and the next NTPL lines are kept.

3.1.1.3 Drop a solution , button

Click this button to drop the current item shown in combo 3.1.1.2.

3.1.1.4 Detail a solution , button

Prompt a dialog of table showing the brief of the current solution

3.1.2 Scene tool, tool box

This tool box contains controls to build, plot and interpolate a manifold. In QTPlaut, a manifold is a 2D or 3D projection of an AUTO solution. It is simple. A scene presents one or more manifolds. Although a solution can be loaded more than once, it is not advised. If a manifold needs interpolation operations, it will be reconstructed from the raw solution data loaded, so a solution should always be kept after it is loaded. Both the manifolds and the solutions have their unique IDs. The manifold ID is shown explicitly in the built manifolds combo. The current manifold or solution is the current item in according combos.

About the dimensions, in MATLAB, x-axis orients from the left to the right of the screen, y-axis orients as the direction as the user’s eye vector pointing at the screen, z-axis from the screen bottom to the top. But in OpenGL, this is not the case. The x-axes are the same, in openGL the y-axis is from the screen bottom to the top, and the z-axis is a vector orthogonal to the screen but pointing to the user. The above describes the case that view vector is the user-screen vector without any transformation. In QTPlaut we keep the same axes orientation as MATLAB. So in the dimension settings group box, X+ means the openGL X+, equal to the real world X+ (dimension 1 in solutions, dimension 0 is always the time t). Y- means the OpenGL Y-, equal to the real world Z+ (dimension 3 in solutions). Z+ means the OpenGL Z+, equal to the real world Y+ (dimension 2 in solutions). In the case of 2D, X axis originates from left to the right of the screen and Y axis originates from the bottom to the top of the screen. No matter how tricky the transformations are , right hand rule is always kept.

3.1.2.1 Uploaded solutions, combo

Showing all the solutions loaded, highlighting the current solution we are working with.

3.1.2.2 Plotting dimensions, group box

Two exclusive radio-buttons , 2D or 3D

3.1.2.3 Dimension settings, group box

The orientation is discussed as above. In each of the three combos, all dimensions in the solution are shown. 0 is always for time t; 1,2,3... are as the sequence in AUTO solution.

3.1.2.4 Construct a manifold, button

Construct means project the current solution according to the dimension settings into 2D or 3D graphical representation, the manifold. Each manifold has a unique mother solution. A solution can have multiple presentation of manifolds drawn as orbits only, manifold only, both, or part of them. By default, the manifold is plotted using color-map HSV onto its orbits indices. In short, it is marked as diverse. A manifold present itself as the manifold (vertices, mesh, surface) or the orbits (showing all or single). A subset operation is possible by setting starting/end orbits and starting/end indices. The manifold can be single-colored or diverse-colored. So the orbits can also be single-colored or divers-colored.

3.1.2.5 Built manifolds, combo box

Showing the current manifold. It contains all the built manifolds with the unique IDs (starting from zero).

3.1.2.6 Drop a manifold, button

Drop the current manifold in the built manifolds combo.

3.1.2.7 Set Manifold Color, button

If the Manifold color settings below are chosen as single, a dialog will be prompted so you can choose the color.

3.1.2.8 Set Orbit Color, button

If the Orbits color settings below is chosen as single, a dialog will be prompted so you can choose the color.

3.1.2.9 Manifold color settings, group box

As discussed above.

3.1.2.10 Orbits color settings, group box

As discussed above.

3.1.2.11 Interpolation settings, group box

none. It doesn’t mean nothing. It means no interpolation, the original orbits/mesh are kept.

time. Along the evolution of time each orbit of the manifold is interpolated by n segments of time into n+1 points. n is set by a spin-box , the Time segments ,in the group box interpolation settings below.

arclength. Along the evolution of time each orbit of the manifold is interpolated by n segments of arc-length into n+1 points. n is set by a spin-box , the Arclength segments ,in the group box interpolation settings below.

reverse-time. Reverse to the evolution of time each orbit of the manifold is interpolated by n segments of time into n+1 points. n is set by a spin-box , the Time segments ,in the group box interpolation settings below.

reverse-arclength. Reverse to the evolution of time each orbit of the manifold is interpolated by n segments of arc-length into n+1 points. n is set by a spin-box , the Arclength segments ,in the group box interpolation settings below.

The following options set a segment scale rather than set the number of segments. In a solution, we can always find the maximum or minimum arc-length or the maximum or minimum time. So the step of interpolation can choose a uniform scale for all the branches/orbits, such as the max arc-length divided by n.

The interpolation always keeps the first point and the end point of a branch. n is always greater than 1, so the resulted manifold orbit will always contain at least two points. A branch that contains only one point should be avoided, while AUTO solution’s NTPL is always >> 2. But, for example, in the CR3BP, L1 family has some branch points shrink into one point. So the Lagrange interpolation has overlapped mesh points and divides zero. QTPlaut can deal with this case by decimating the redundant overlapping points.

The uniform scale interpolation such as Max Arc Div N or Min Arc Div N is very fit for the simplification of the CR3BP L1 family type. The outer orbit will contain more points and the inner less. So the resulted '.mlf''s size will be optimized.

All the interpolation settings are global, affecting all the orbits. But if you interpolate the only one single branch or part of the branches, QTPlaut will set the option to local interpolation or no change. In some extreme cases, the user may need to interpolate some specific orbits to achieve better visual effects. QTPlaut offers such functions that will be shown in 3.1.3 .

3.1.2.12 Interpolation segments, group box

Set n for time or arc-length interpolation respectively.

3.1.2.13 Manifold settings, group box

Hide. The manifold is not shown. Thus the manifold will not written to the '.mlf' file. In QTPlaut, what you see is what passed to MATPlaut.

Vertices, show manifold as vertices.

Mesh, the net mesh after triangulation.

Surface, it is left for future improvement.

3.1.2.13 Draw part of the manifold, group box

If a manifold has N orbits. A subset operation is possible by select the start/end orbits, bounded by [1, N].

Combing the starting/end indices setting, we can take a patch from a manifold. The indices are bounded by [0, M]; M is the maximum vertices count of all orbits.

3.1.2.14 Orbit style, group box

Tube or line. In fact in QTPlaut all orbits are drawn by line. But one is anti-aliasing the other is not.

3.1.2.15 Orbit settings, group box

Current orbit. If you set show orbits as single. This index decides the current orbit you see.

Tube size, line size, vertex size settings are useful to highlight part or the whole manifold in QTPlaut without transparency effects in QTPlaut.

3.1.2.16 Modify a manifold, button

This is another powerful tool to select orbits from manifold. It uses spread sheet type dialog. The user can drop, duplicate, re-sequence the orbits against the original solution. At the end, another dialog will popup. Remember YES is to save the change, NO keep the original before you call the modification.

The difference with the draw part of the manifold is that, draw part doesn't delete the orbits from the manifold, any interpolation operations still applies to all the orbits.

The result of modifying the manifold will remove or reset the orbits irrecoverable. To restore the original orbits, the user should drop the modified manifold and construct it from the solution again.

3.1.2.17 Write the scene to a file, button

What you see will be what you write to the '.mlf' file.

A solution's projection can be written to the '.mlf' as the manifold, the orbits, or both. If the manifold settings is 'hide', the manifold (in MATPLaut the surface)won't be written. If the show orbits is set as none, the orbits won't be written (you won't see those tubes). But 'all' orbits or 'single' orbit has no difference, they all mean write all orbits.

In '.mlf' there always are:

The total number of the solutions' projections

For a manifold, the list of all vertices, the list of triplets of how they are triangulated, the indices of the stripes. Stripe means the triangles between two orbits. MATPlaut needs it to apply colormap schemes.

For orbits, if shown, there is the list of end vertices' indices of each orbit, and is also the total number of orbits written.

Some tutorials at the end of this document will help to understand.

3.1.3 Local interpolations

If the user needs to interpolate some specific orbit or some sequential orbits, the user can use local interpolation.

3.1.3.1 Built manifolds

Show the current manifold.

3.1.3.2 Available orbits

Show the indices of all the orbits.

3.1.3.4 Set orbits to edit

Select start or end orbit, part to edit. If start=end, only one orbit is selected.

3.1.3.5 Triangulate Next Orbit

By default, during triangulation, each orbit is triangulated with the next orbit; the end orbit will do nothing. The trick here is that, you add an orbit twice, use the setting here not to be triangulated with itself. Then the orbit can have two interpolation settings. So it can help to form better triangle mesh when the two other orbits at either side of it have a big difference in arc-length.

3.1.3.6 Interpolation settings

Explained in 3.1.2.

3.1.3.7 Interpolation segments

Set n for interpolation

3.1.3.8 Subset operation, construct, append

Suppose you set n as 100, you will get 101 points for the interpolated orbit/orbits. Construct create a new orbit with vertices [start,end], bounded by [0 100]. Append takes the [start,end] part, bounded by [0 100], adds them to the end of the orbit/orbits you select. This trick makes it possible that one orbit can have more than one interpolation settings. The append feature is not fully tested.

3.1.4 Diagram transformation

The first slider rotates the scene about the OpenGL x-axis.

The second slider rotates the scene about the OpenGL y-axis.

The third slider rotates the scene about the OpenGL z-axis.

The fourth slider performs some limited zoom operations.

The mouse can perform some rotation operations.

Since MATPlaut can do far better in transformations, this part is kept as simple as needed.

3.2 MATPlaut

MATPlaut is a set of MATLAB '.m' files. It has a GUI for the users to ease the settings for plotting AUTO solutions. The GUI is event driven. So the effect of setting change can be seen immediately without the restarting of the program.

In your MATLAB command console, change the working directory to the directory where you unzip the MATPlaut files. In the command line, type 'MATplaut'. The GUI is started.

The out looking likes the QTPlaut, but MATLAB is not specialized at GUI design. It has only limited controls. However, it meets our requirement.

3.2.1 Input mlf file

In text input line, enter the full path file name 'xxx.mlf'. MATLAB GUI doesn't offer a navigation window to help you to find the file. If you just enter the xxx.mlf, MATPlaut will try to retrieve the file from the directory where you call MATPlaut.

3.2.2 Current Solution

In the combo box, there will be the indices list of the solutions you wrote to the '.mlf' file. The original solution file name of the current index with a unique ID will appear on the right. The following settings till camera motion will affect the current solution only. We should say affecting the current presentation of a solution.

3.2.3 Manifold color setting

The solution of AUTO's 3D/2D projection can be mapped onto a rectangular. One edge dotted by the indices of orbits, one edge dotted by time or arc-length. Colormap is a sequence of colors we can use to enhance the contrast.

So you will understand the difference between colormap onto arc-length or colormap onto orbits indices. Single color means only one color for the manifold. If you tick this option, you can click the set manifold color button to change the color.

3.2.4 Color map

All these thirteen color-maps are cited from MATLAB . So please check the MATLAB documentation for details.

3.2.5 Set Manifold Color

As stated above if a manifold is set to be single color, the color can be changed here through a very convenient color palette GUI.

3.2.6 Set orbits color

If in QTPlaut, you allow the plotting of orbits, the orbits all share one color and the color can be changed here. Among these orbits, if you want to highlight one or more with different colors, you may construct multiple manifolds from a same solution. Modify the manifolds to get the orbit/orbits you want to highlight then set different colors here.

3.2.7 Orbit style

Obviously tube option gives better visual effects. But line option is simple and fast to be plotted.

3.2.8 Tube size

You can edit the tube size in the first text field. The input should be positive number. Fault correction is implemented here. The second text field is the delta, you click the +, - button, the tube size will be increased or decreased by delta, but always be greater than zero.

3.2.9 Manifold Alpha

The fault correction trigger will always guarantee the alpha value is bounded by [0,1.0]. 0 means the manifold is totally transparent or invisible. 1 means it is opaque or no light could penetrate.

3.2.10 Orbit Alpha

It is convenient to set different alpha values for orbits and manifold respectively.

3.2.11 Camera Motion

3.2.12 Light Motion

3.2.13 Light

3.2.14 Light and camera polar angles

Please check MATLAB documents for ‘view’ and ‘lightangle’ commands.

3.2.15 Input r3b file (s.start)

In order to present the CR3BP more accurately, we parse the L1,L2,L3,L4,L5 positions and mu, thus the primaries’ positions, from the s.start file. It is the user’s responsibility to provide the correct s.start file corresponding to the solutions we are dealing with.

3.2.16 Earth size

3.2.17 Moon size

They are resizable. But the value should be none-zero. If they are positive, the larger primary will use the earth texture, the smaller primary will use the moon texture. For other CR3BP problem, the sizes could be set negative thus no textures are shown.

3.2.18 Liberation point size

We use some cubes to represent the liberation points. There are resizable.

3.2.19 Disk Alpha 0~1.0

Change the transparency setting of the CR3BP disk on the orbiting plane.

3.2.20 CR3BP Mu

L1,L2,L3,L4,L5 positions are acquired from s.start.

mu is acquired from the second line of the first branch of the s.start file.

The large primary is at (mu,0,0), the small primary is at (1-mu,0,0).

3.2.21 Build a .m script

Suppose you write a test1.m file to your MATLAB working directory. You may call it by type 'test1();' in your MATLAB command line console. What will you get?

The MATLAB gurus then can change the axes, add labels, export the image file and etc. for their publishing.

4. Tutorials

4.1 The coordinate system

a. Call an instance of QTPlaut.

b. Click ‘Add a solution’. By the navigation window, find the file ‘s.xyza’. Single click the file. Then click button ‘Open’.

c. The file name is shown in the ‘Uploaded solutions’ combo. You can click ‘Detail of a solution’ to get a brief. But this file is not generated by AUTO, it is a pseudo-solution file for tutorial purpose.

d. Then click the ‘Scene tools’ toolbox header. The panel for the Scene tools toolbox is revealed.

e. Click the long button ‘Construct a manifold’. In the ‘Built manifolds’ combo, the solution file name with a unique ID number will be shown.

f. In ‘Manifold settings’, click radio-button ‘hide’

g. In ‘Show orbits’, click radio-button ‘all’

h. In the ‘Diagram panel’ you can hold the left key and rotate the figure a little bit. The figure reveals that the x-axis has three orbits, the y-axis has five orbits, and the z-axis has seven orbits. Whenever you get confused with the orientation of the coordinate system, you can load this file. Or generate such a file meeting your requirement. In QTPlaut, the axes and label function are not implemented, since MATPlaut or MATLAB can do far better.

i. You can try to set ‘Manifold settings’ as ‘mesh’ then change back to ‘hide’.

j. Now use the scroll bar go to the bottom of the ‘Scene tools’ panel. Click ‘Modify a manifold’. In the right of the panel, click drop all. In the left panel select items with ‘LAB’ 1,2,3. This is like in a spread sheet software. You hold ‘shift’ key to select sequentially, ‘ctrl’ to select continually. Then click ‘Add’. Close the dialog. Click ‘Yes’ to confirm the changes. Now only the x-axis are shown.

k. Click ‘Construct a manifold’ again. In the ‘built manifolds’ combo, there will be two items. Highlight ‘s.xyza-1’. The unique ID start with zero. ‘hide’ manifold. ‘Show orbits’ ‘all’. Click the ‘Modify a manifold’ again, ‘Drop all’, ‘Add’ LAB 4,5,6,7,8 . Close. ‘YES’ to save changes.

l. Construct a ‘s.xyza-2’, repeat k, but add LAB 9,10,11,12,13,14,15. You can try to see the ‘mesh’s. But remember to ‘hide’ again for each manifold.

m. Click ‘Write the scene to a file’. Change the file name to ‘axes.mlf’. Click ‘save’. Close QTPlaut.

n. Go to MATLAB. Call MATPLaut, clear all and add ‘axes.mlf’ at the ‘add mlf file’edit line. You should add the full path name or copy this file to your working directory.

o. The ‘current solution’ index should be ‘1’ and name ‘s.xyza-0’. At ‘tube size’ clear all and type ‘0.05’ as the new tube radius, hit ‘enter’. Then click ‘Set orbit(s) color’. You can choose a color you believe it is red. Select index ‘2’ for ‘s.xyza-1’, change the ‘tube size’ and set the green color for it. Do the same thing with index ‘3’ for ‘s.xyza-2’, set color as blue. The green orbits may not shine. In the ‘camera and light polar angles’ set light ‘el’ as 35. Done.

The following two figures show the final results in QTPlaut and MATPlaut. In this tutorial, we try to clarify how the coordinate system is oriented. No matter how you rotate the figures, right hand rule is always followed. We also learn how to cut one solution file into three pieces, how to set the tube size and how to change the orbit(s) color.

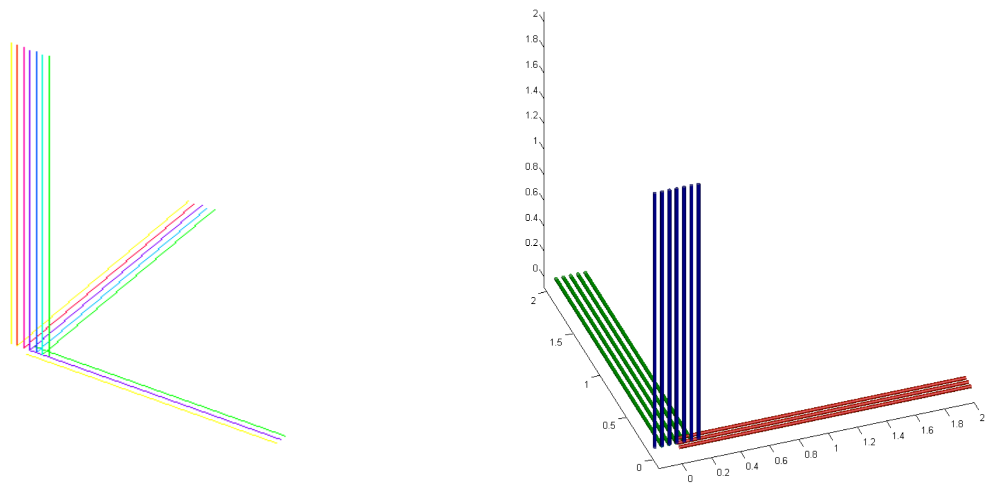
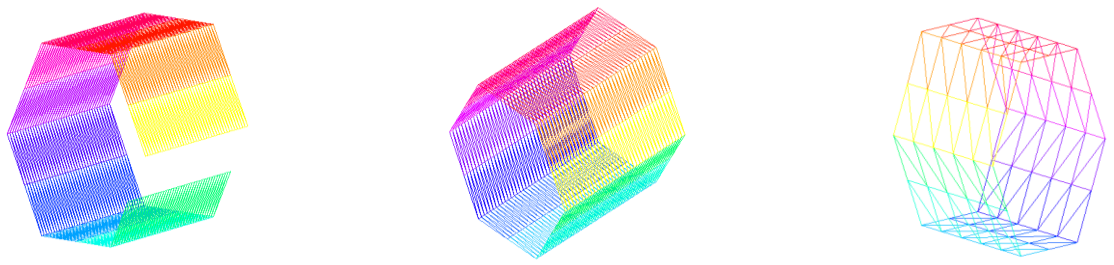


Figure 1: s.xyza in QTPlaut (left) and in MATPlaut (right)

4.2 The closed hex- prism

a. In QTPlaut, add solution ‘s.hexprism’ . This file should be in the unzipped QTPlaut folder or in the sample_files folder.

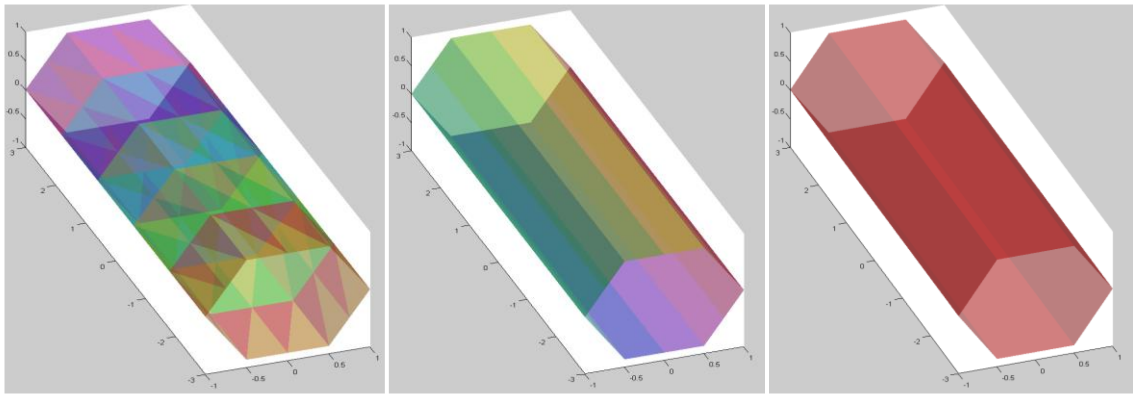
b. In scene tools ‘Construct a manifold’. In ‘Manifold settings’ choose ‘mesh’. ‘Modify a manfold’. ‘Drop all’. ‘Add all’. Select LAB 1 (if you click or highlight any cell for the line of LAB 1, that means selected.). close. ‘YES’ to save change. In ‘Interpolation Settings’ click ‘arclength’ or ‘reverse-arclength’, in ‘Interpolation segments’ change ‘Arclength segments’ to 5. ‘Write the scene to a file’ named ‘hexprism.mlf’. Close QTPlaut.



a Not closed b. Closed c. Closed and Interpolated

Figure 2:. s.hexprism in QTPlaut

c. ‘read’ ‘hexprism.mlf’ in MATPlaut. Change light polar angle el to 35. In the ‘Manifold color settings’ try the three different options. You can also try with other colormap options, manifold alpha value , view angles, light angles and etc. The scale in MATPlaut is different from QTPlaut. In QTPlaut the figure is scaled into a cube. In MATPlaut we use ‘daspect [1 1 1]’. For daspect please check MATLAB manual.



a. HSV mapped onto arc-length b. HSV mapped onto orbits indices c. Single color

Figure 3: s.hexprism in MATPlaut, the alpha value is 0.5

In this tutorial, we try to close a manifold, if the user believes the solution should be of a cylinder, a tube or a torus type. We also try with simple interpolation settings, apply different color schemes, and the user can use this naive pseudo-solution to explore all kinds of settings.

4.3 Interpolation and Mesh

a. 'Add a solution' file 's.intpl','Construct a manifold'. In 'Manifold settings' set 'mesh'. In 'Diagram transformation', use the slide bars to rotate and zoom the diagram that fits best.In 'Scene tools' groupbox 'Manifold color setings' set 'Single'. Click button 'Set Manifold Color', choose a color that fits best.
b. In 'Interpolation Settings' choose different interpolation settings.
c. Since the pseudo-solution we built has linear time-space relation, the difference between time and arc-length interpolation is not obvious. In the next tutorial, the users can experiment it by themselves.

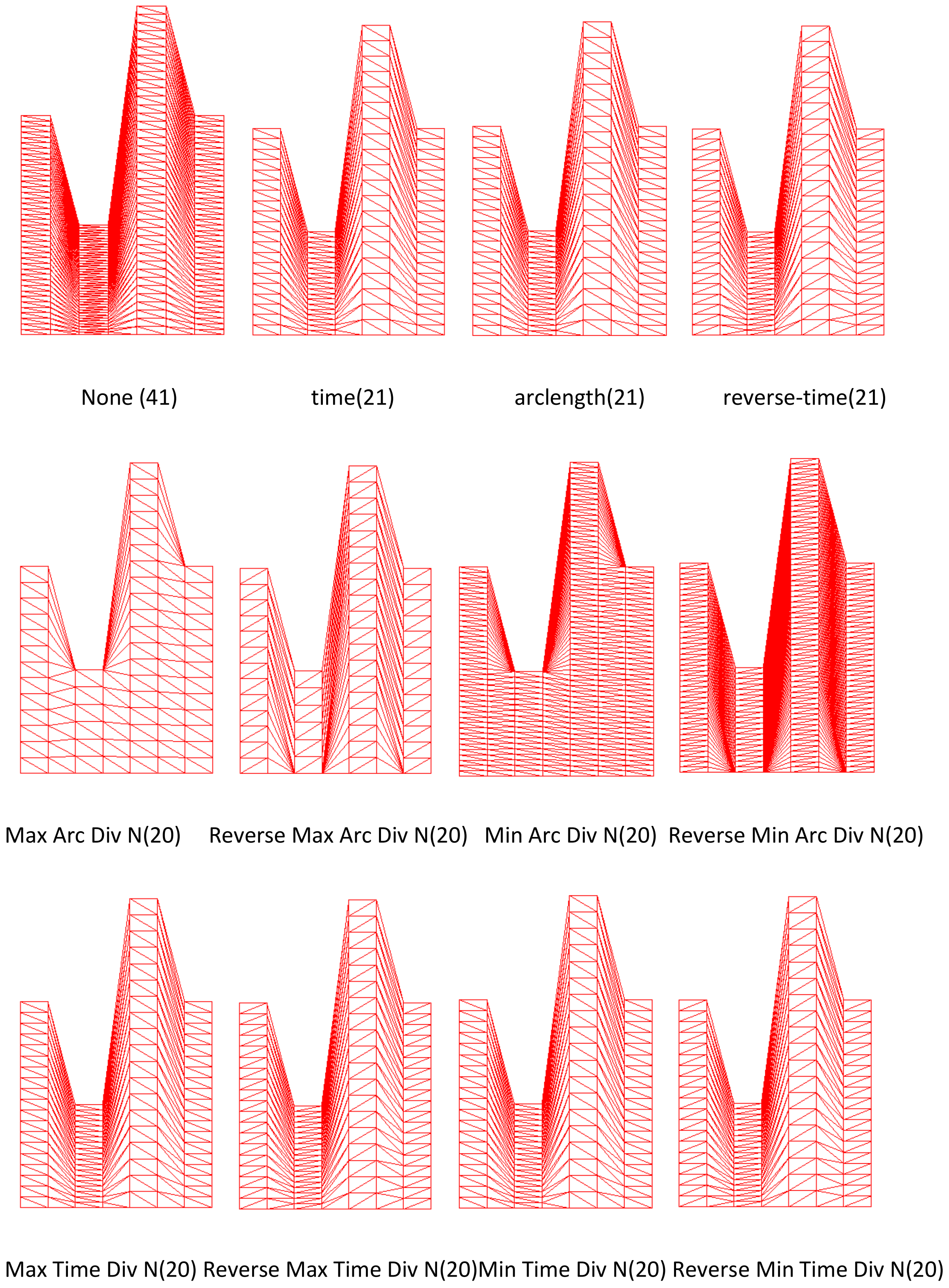


Figure 4: Tutorial 4.3 Interpolation and Mesh

This tutorial explains the various interpolation options and their mesh constructions.

4.4 The Lorenz manifold

a. Open an instance of qtplaut. 'Add a solution' file 's.17'. File s.17 is an AUTO solution computed by Prof. Eusebius Doedel for the famous Lorenz manifold. This file should be in the example-files folder. Its size is about 62.5MB. There is a smaller arc-length solution s.15. You may also try with this solution. 'Construct a manifold'. You may switch 'Manifold settings' between 'vertices' and 'mesh', but the 'vertices' option seems to be better. In the 'Diagram transformation', slide the first two bars to the left end, use the fourth bar to zoom the diagram. ‘Write the scene to a file' as 's17-full.mlf', it will take a while. QTPlaut can process very large file, the developer once loaded around 1G bytes solution files in a PC with 8G main memory (Fedora 8/I386-64). Your free memory should be at least twice the size of the files you plan to read.

b. In 'Interpolation Settings' tick 'reverse-arclength', in 'Interpolation segments' field 'Archlength segments' change '20' to '100'.'Write the scene to a file' as 's17-revarc100-allorbits.mlf'. in 'Interpolation segments' field 'Archlength segments' change '100' to '50'.'Write the scene to a file' as 's17-revarc50-allorbits.mlf'.in 'Interpolation segments' field 'Archlength segments' change '50' to '20'.'Write the scene to a file' as 's17-revarc20-allorbits.mlf'.

c. 'Modify a manifold' then 'Drop half', close , 'YES' to save changes. In 'Interpolation Settings' tick 'reverse-arclength', in 'Interpolation segments' field 'Archlength segments' change to '100','Write the scene to a file' as 's17-revarc100-halforbits.mlf'.'Archlength segments' change to '50','Write the scene to a file' as 's17-revarc50-halforbits.mlf'.'Archlength segments' change to '20','Write the scene to a file' as 's17-revarc20-halforbits.mlf'.

d. 'Modify a manifold' then 'Drop half', close , 'YES' to save changes.In 'Interpolation Settings' tick 'reverse-arclength',in 'Interpolation segments' field 'Archlength segments' change to '100','Write the scene to a file' as 's17-revarc100-halfhalforbits.mlf'.'Archlength segments' change to '50','Write the scene to a file' as 's17-revarc50-halfhalforbits.mlf'.'Archlength segments' change to '20','Write the scene to a file' as 's17-revarc20-halfhalforbits.mlf'. Close QTPlaut.

e. In MATPlaut try to open and plot these files respectively. It is good to restart MATPlaut each time.

f. However, MATLAB failed to plot the following files:
s17-full.mlf
s17-revarc100-allorbits.mlf
s17-revarc50-halforbits.mlf
s17-revarc20-allorbits.mlf
s17-revarc100-halforbits.mlf
s17-revarc50-allorbits.mlf
The experiments are done on a Pentium IV machine, 2G memory, Windows XP/SP2.
In next tutorial, we can eve plot file size greater than some of the above. The reason could be during rendering transparency effects, MATLAB puts some limits on the total number of vertices or the maximum layers of graphical objects to test the depth, or the maximum number of textures it can mix. In OpenGL/QT, we can plot unlimited graphical objects in theory. But to implement the transparency effect, the OpenGL programming job is so huge and it involves too many computer graphics techniques.

In this tutorial, we have in some senses tested the limit of QTPlaut and MATPlaut. By following the tutorials explored by the developer, the users can achieve a practical visual presentation of their mathematical solutions.

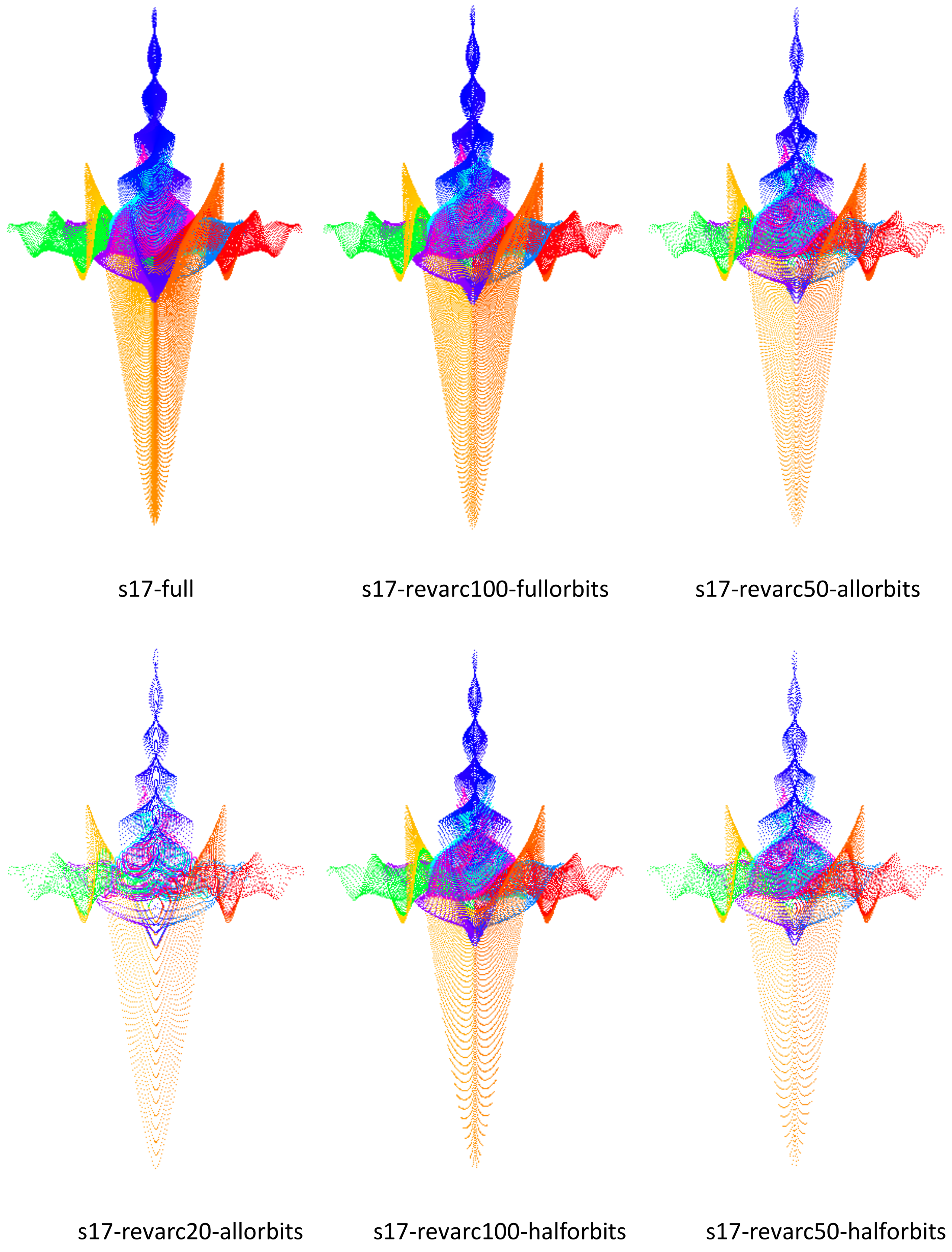
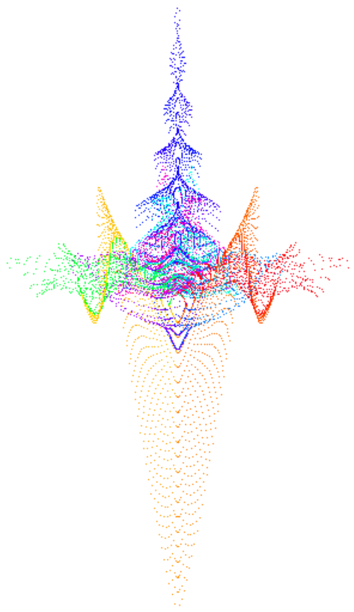
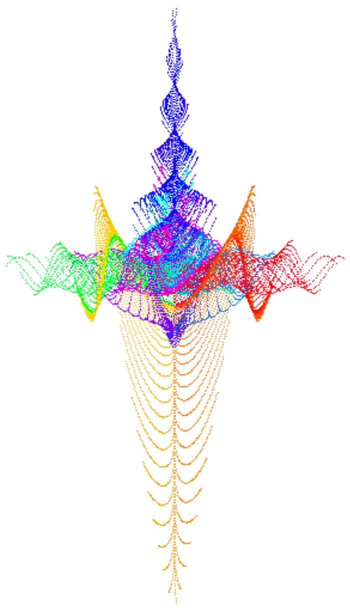
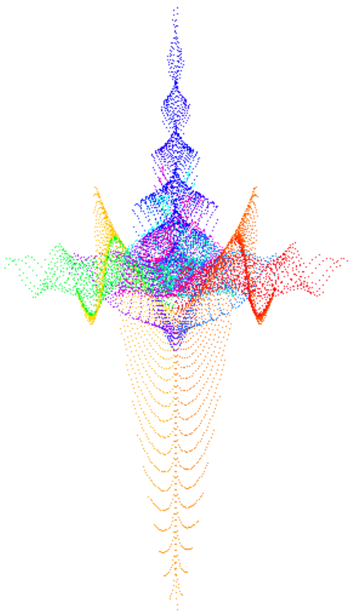


Figure 5: s.17 interpolated and plotted by QTPlaut -1


s17-revarc20-halforbits


s17-revarc100-halfhalforbits


s17-revarc50-halfhalforbits

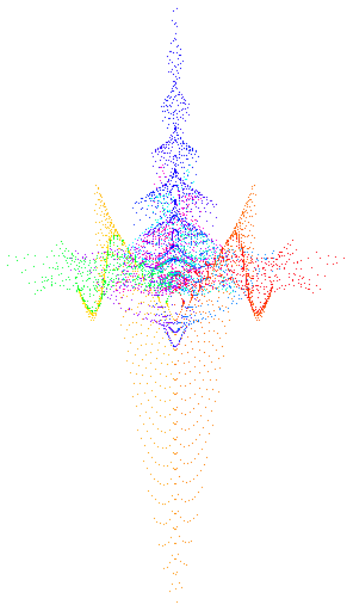
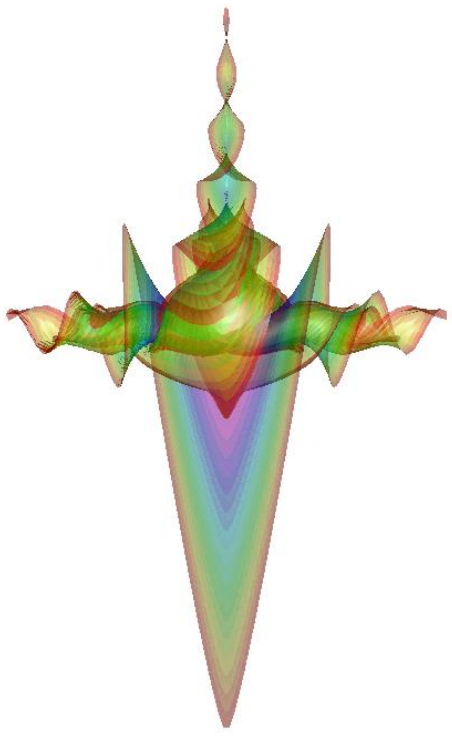
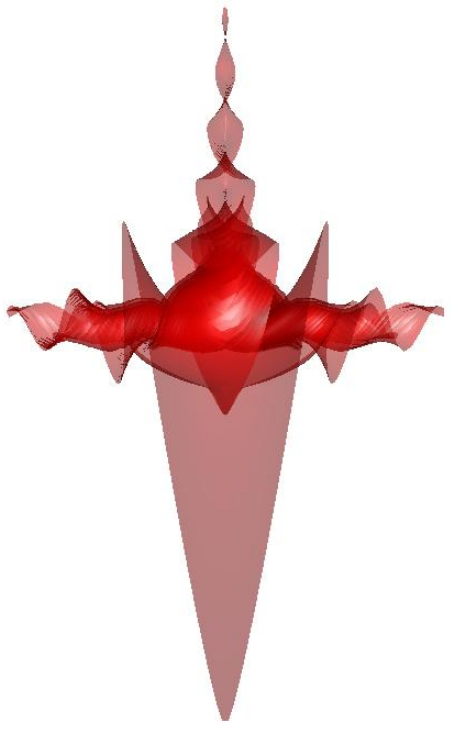

s17-revarc20-halfhalforbits

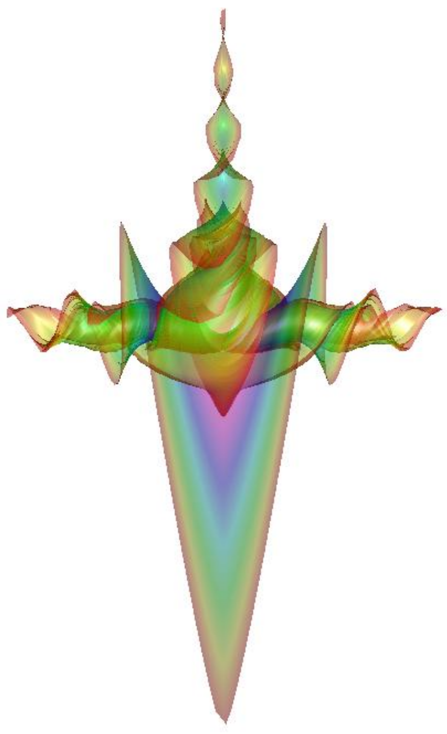
Figure 6: s.17 interpolated and plotted by QTPlaut -2



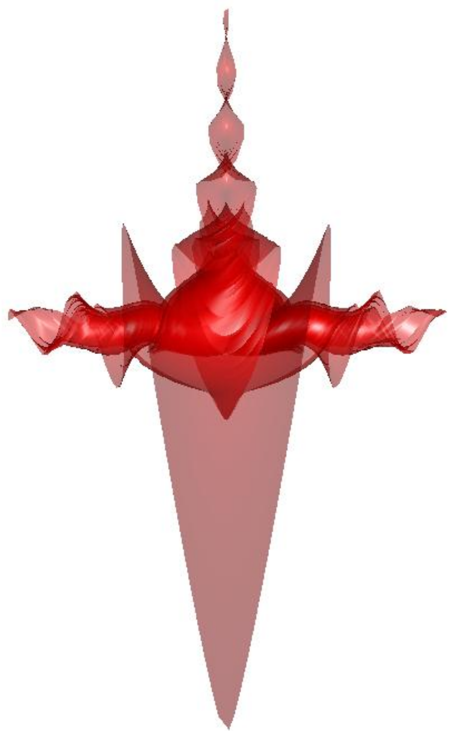
s17-revarc20-halforbits-1



s17-revarc20-halforbits-2

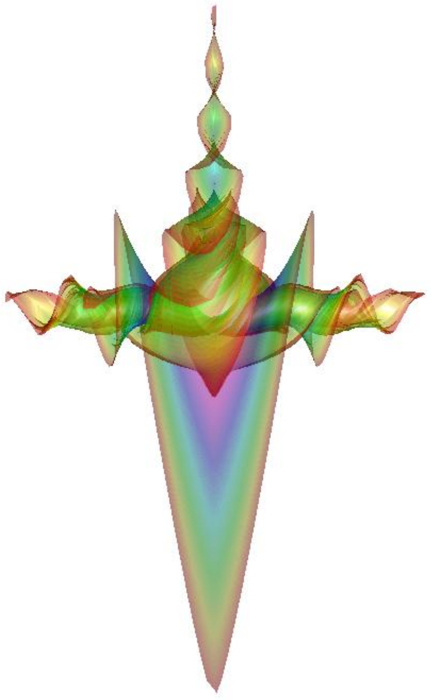


s17-revarc100-halfhalforbits-1

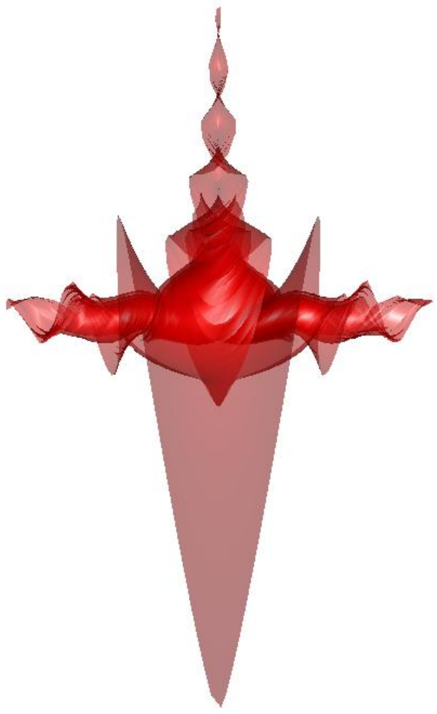


s17-revarc100-halfhalforbits-2

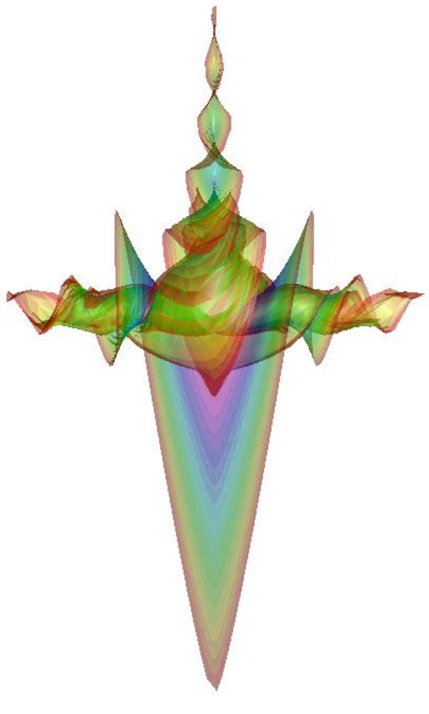
Figure 7: s.17 plotted by MATPlaut -1



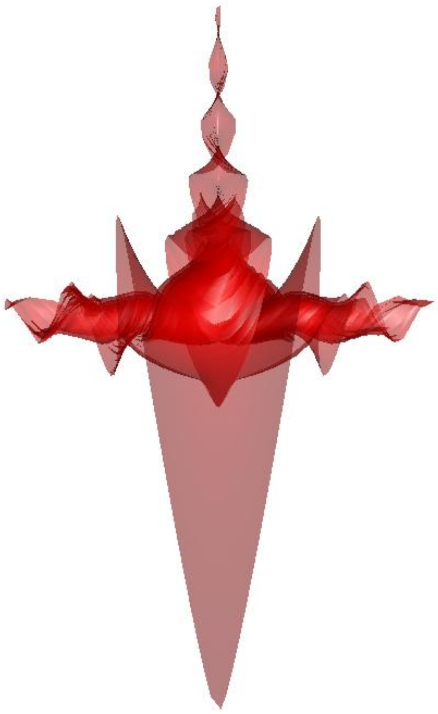
s17-revarc50-halfhalforbits-1



s17-revarc50-halfhalforbits-2



s17-revarc20-halfhalforbits-1



s17-revarc20-halfhalforbits-1

Figure 8: s.17 plotted by MATPlaut -2

4.5 The CR3BP mu= 0.063, H1a and hetH1a. This tutorial needs results of AUTO07p demo r3b.

a. Call an Instance of QTPlaut. 'Add a solution' with file 's.hetH1a' and 'Add a solution' with file 's.H1a' .Switch to tool-box 'Scene tools', in 'Uploaded solutions' select 's.hetH1a', 'Construct a manifold', 'Modify a manifold', 'Drop all', select LAB '34', close, 'YES' to save. In 'Manifold settings', tick 'hide'. In 'Show orbits' tick 'all'. In 'Uploaded solutions' select 's.H1a'. 'Construct a manifold'. In 'Built manifolds', select 's.H1a-1'. You select to see s.H1a as mesh or vertices. 'Write the scene to a file as 'h1a-m-hetH1a-o-full.mlf'.
b. In interpolation settings, select 'reverse-arclength', in 'Interpolation segments' set 'Archlength segments' as 100.'Write the scene to a file as 'h1a-m-hetH1a-o-revArc100.mlf'.
c. In interpolation settings, select 'reverse-arclength', in 'Interpolation segments' set 'Archlength segments' as 50.'Write the scene to a file as 'h1a-m-hetH1a-o-revArc50.mlf'.
d. In interpolation settings, select 'reverse-arclength', in 'Interpolation segments' set 'Archlength segments' as 20.'Write the scene to a file as 'h1a-m-hetH1a-o-revArc20.mlf'. Close QTPlaut.
e. Call MATPlaut. Try file 'h1a-m-hetH1a-o-full.mlf'. In ‘Current solution’ select ‘2’ , for ‘s.H1a-1’. In ‘Manifold color settings’ select ‘colormap onto orbits indices’. In ‘Current solution’ select ‘1’ , for ‘s.hetH1a-0’. In ‘Tube size’ change it to ‘0.02’, ‘Orbit Alpha’ change the value to ‘0.3’. In ‘Input r3b file’ enter ‘s.start’, if this file is not in the MATLAB working directory, the user should enter the full path name. Click ‘read’. The figure now looks like the one on the cover.
f. Try other files.
g. At the GUI right-bottom, ‘Build a .m script’ input file name ‘testr3b.m’ then click ‘write’, close the GUI. In MATLAB command line, type ‘testr3b();’, you will see a same figure as plotted through the GUI. Then you can change the script yourself, or add labels, change axes, export the figure and etc.

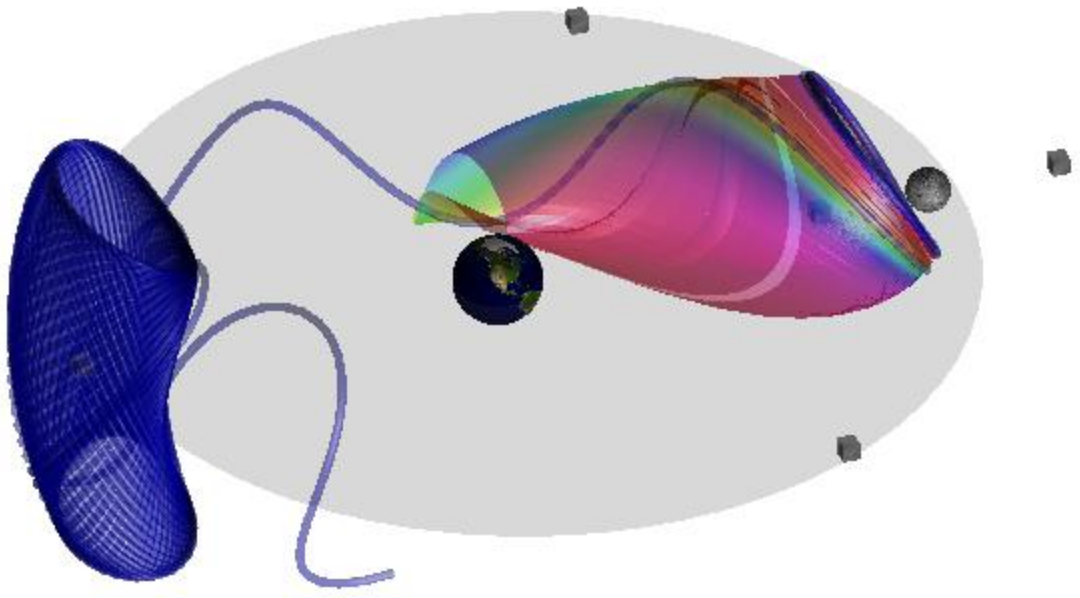


Figure 9: CR3BP h1a-m-hetH1a-o-full.mlf

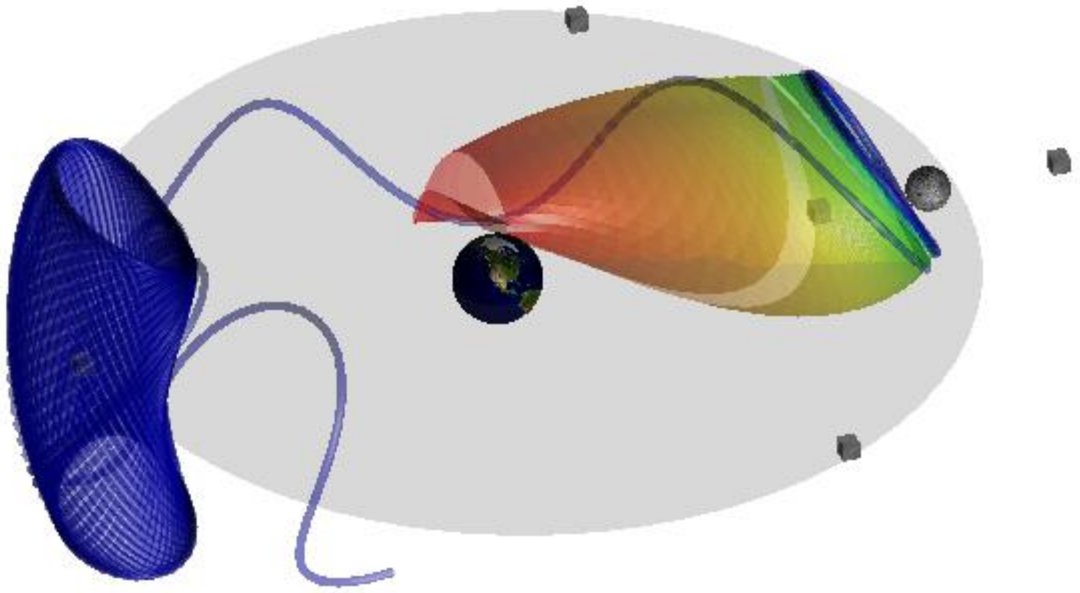


Figure 10: CR3BP h1a-m-hetH1a-o-revArc100.mlf

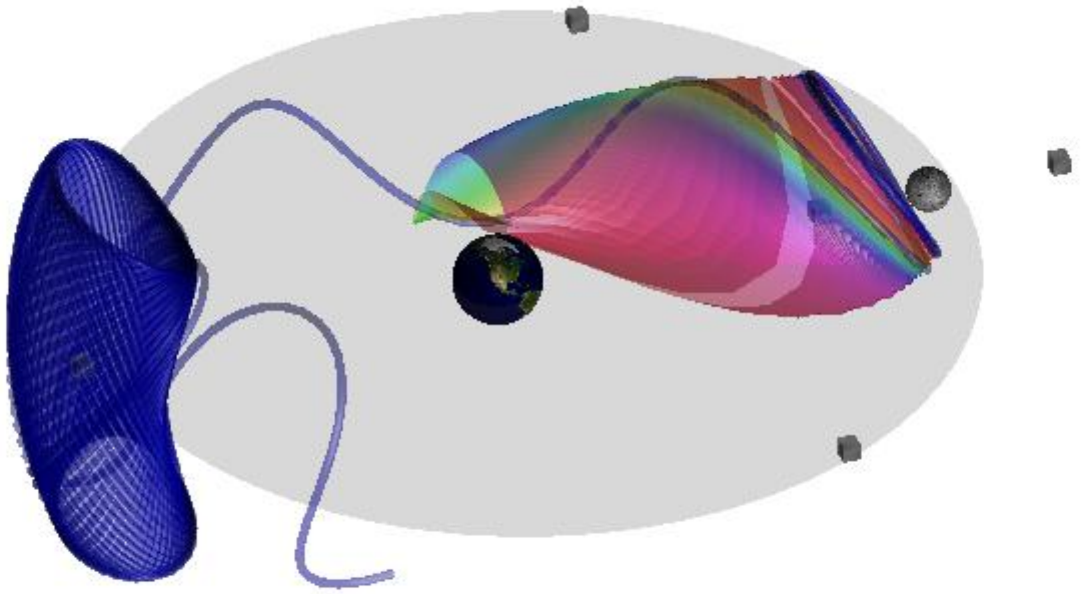


Figure 11: h1a-m-hetH1a-o-revArc50.mlf

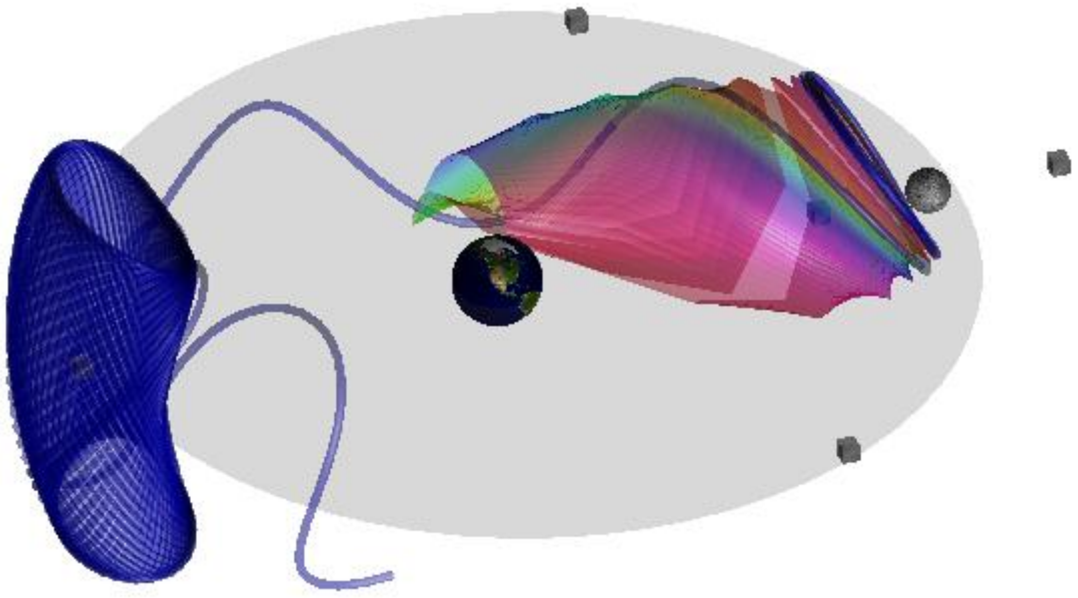


Figure 12: h1a-m-hetH1a-o-revArc20.mlf

4.6 The CR3BP, close manifold H1a

You may notice in tutorial 4.5. There is a gap in manifold H1a. We try to close this gap in this tutorial. There is no mathematical proof that the map should be closed or not. Here we just want to show that QTPlaut has such a function.

a. Call an instance of QTPlaut. 'Add a solution' file 's.H1a' , then 'Construct a manifold'. 'Modify a manifold', 'Drop half', select LAB '1' on the left sheet, then click 'Add'. close , 'YES' to save change. In 'Manifold settings' choose 'mesh'. In 'Interpolation Settings' tick 'reverse-arclength'. In 'Archlength segments' set value to '100'. If you rotate the figure, you can see at the starting part of this manifold, the mesh gets tangled. Now we try to decimate those tangled mesh.
b. Method 1. In 'Interpolation Settings' tick 'Reverse Max Arc Div N'. In 'Draw part of the manifold, decrease the 'End Index' until you can't see any lines you don't want. You can use the up arrow or down arrow to adjust the 'End Index'. Write the scene to file so you can see how it looks like in MATPlaut.
c. Method 2 local interpolation. Quit. Start a new QTPlaut instance. 'Add a solution' file 's.H1a' , then 'Construct a manifold'. 'Modify a manifold', 'Drop half', then select LAB 90 from the left sheet 'Add'. 'Add' LAB 1 after adding LAB 90. At right sheet, you need two LAB 90s, one LAB 1 at the end. In 'Interpolation Settings' tick 'reverse-arclength'. In 'Archlength segments' set value to '100'. In 'Manifold settings' choose 'mesh'.Switch to tool-box 'local interpolations'. Set 'start orbit' and 'End orbit' both to 46. 'Trianglize next orbit' set to 'No'. 'Interpolation Settings' tick 'reverse-arclength'. In 'Interpolation segments' set value to '100'. In 'Start End index' set start as '0' and end as '100'. Click 'Construct'. In 'Set orbits to edit' set both start and end '48'. 'Triangulate next orbit' set

to ‘No’. In ‘Interpolation segments’ set value to ‘130’.In ‘Start End index’ set start ‘0’ and end as ‘100’ , click ‘Construct’. Write a file for MATPlaut.

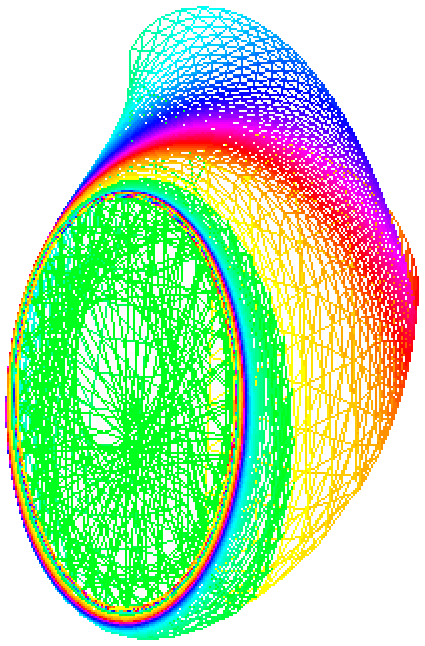


Figure 13: H1a closed, but mesh tangled at the starting part

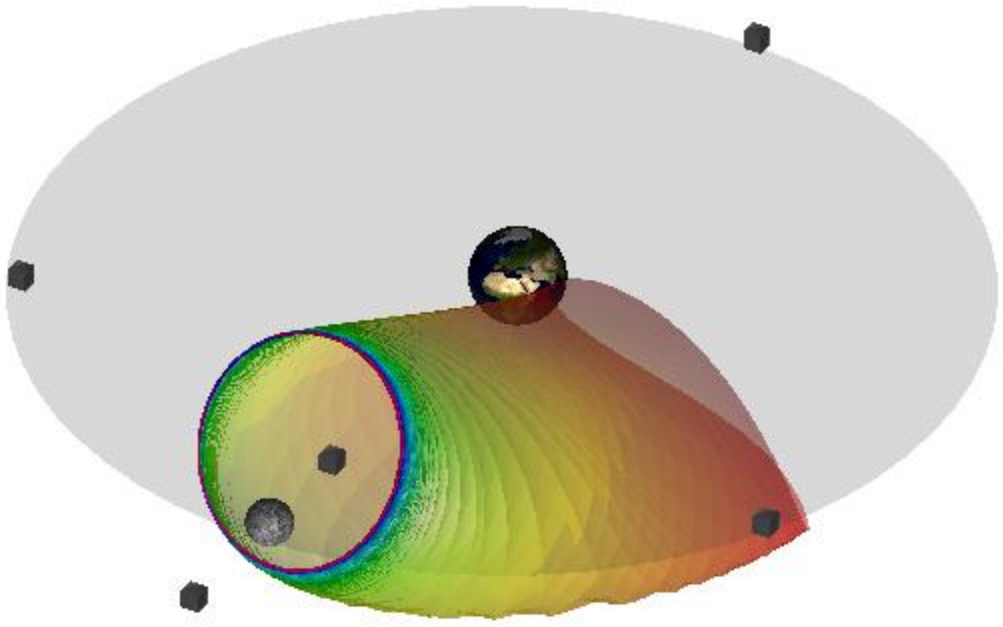


Figure 14: H1a closed, using subset operation

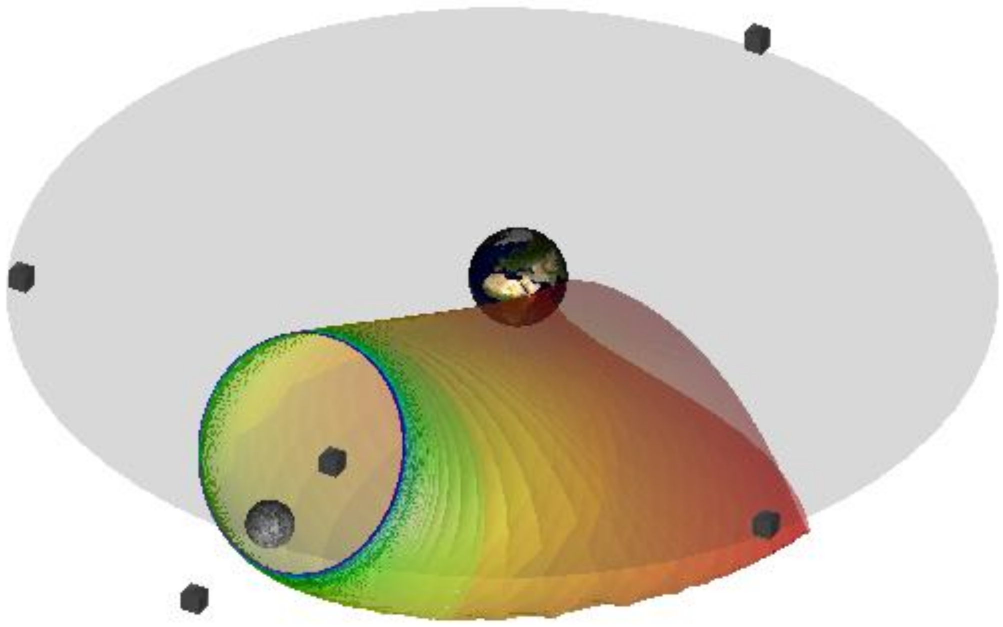
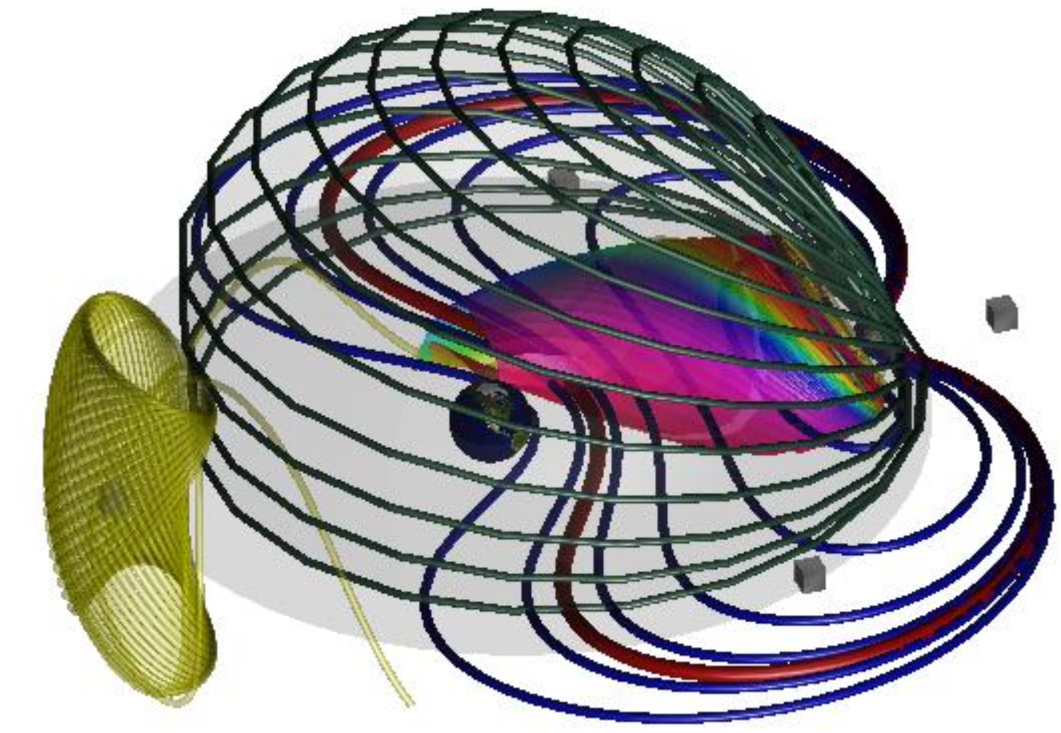


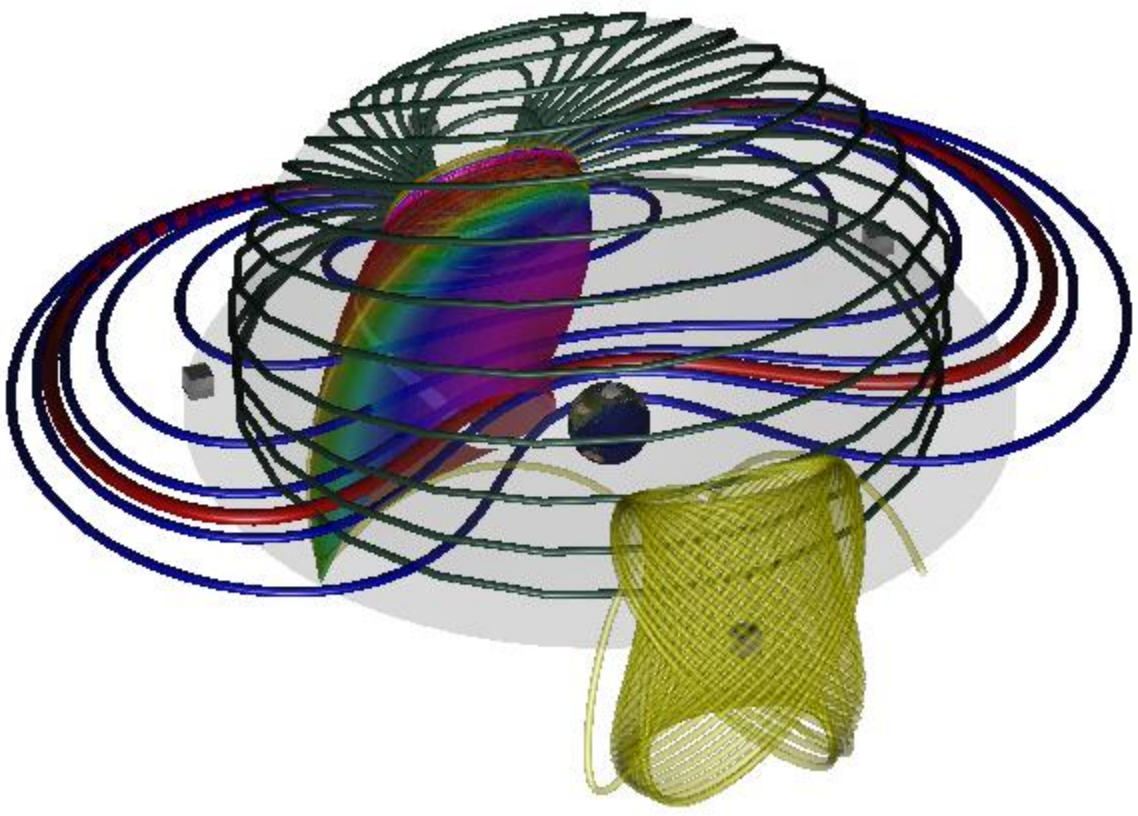
Figure 15: H1a closed, using local interpolations

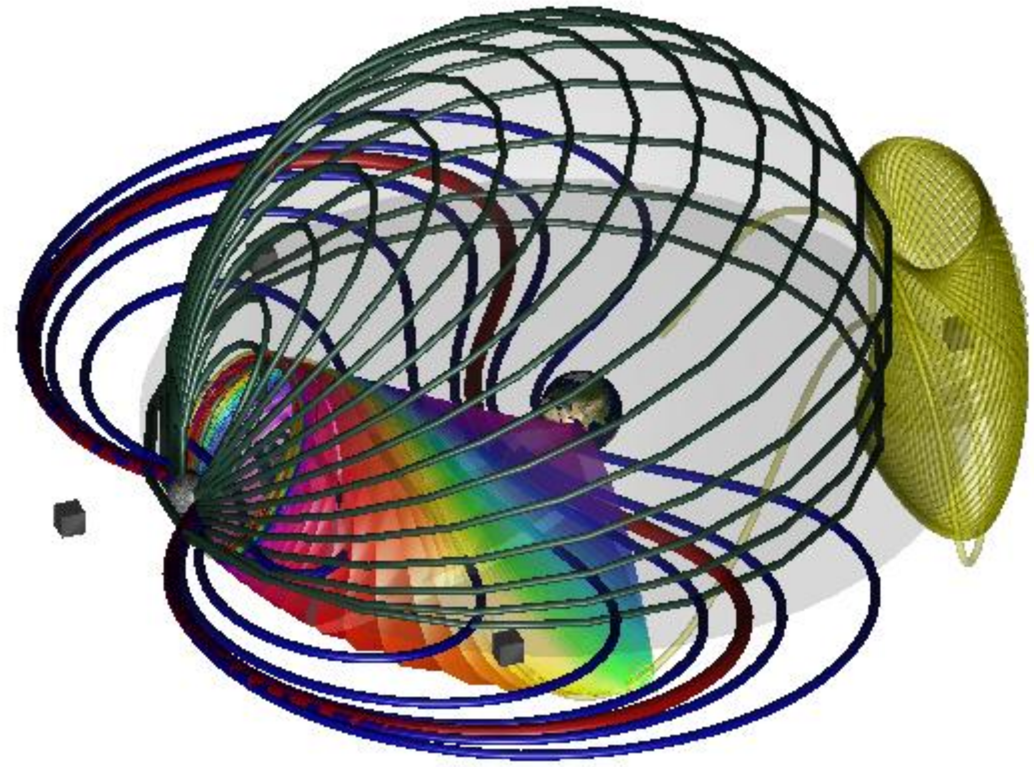
The user is suggested to study the underlying mechanism. This tutorial in some aspects proves the feasibility and usage of local interpolation.

4.7 The CR3BP, test a complex scenario

a. Use QTPlaut to read s.L1, s.H1, s.H1a, and s.hetH1a
b. Construct s.L1 twice. Use modify to keep one with LAB 6,10,14,18,22,26,38. Use local interpolation to set orbits 1~3 interpolated with arclength by 80 segments, remember to change the start index to 0, 4~7 with arclength by 100 segments. The other one of L1 only keep LAB 24.
c. Construct H1. Modify it by applying 'Drop half' twice. Set 'mesh' and show orbits all. Interpolate it with arclength by segments 20.
d. Construct H1a. Modify and drop half. Interpolate by 'Reverse Max Arc Div N' set N as 80. Set 'mesh' and show orbits 'none'.
e. Construct hetH1a. Modify and keep only one orbit. 'hide' manifold, show 'all' orbits. Write the scene to a file. Close QTPlaut. Have a look at the file size comparing the s. Files.
f. In MATPlaut, try to see if you can set the figure as the following.







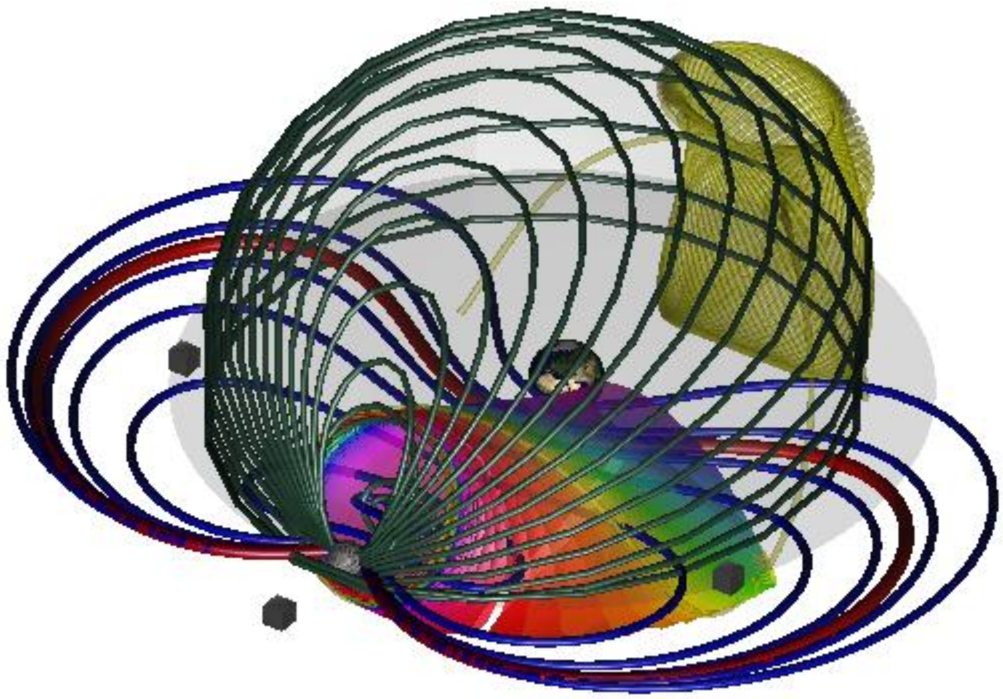


Figure 16: The CR3BP, test a complex scenario

5. Remarks

[1] AUTO, a software for continuation and bifurcation problems in ordinary differential equations, Eusebius Doedel. http://cmvl.cs.concordia.ca/auto/

[2] QT, a cross-platform application framework. http://www.trolltech.com/

[3] MATLAB, The Language of Technical Computing. http://www.mathworks.com/products/matlab/

[4] The earth texture, land_ocean_ice_2048.jpg , NASA.

[5] The moon texture, moonmap1k.JPG, http://planetpixelemporium.com/earth.html

[6] The files followed QT examples are stated in file headers. While, the remaining part can run independently by C++/OpenGL IOs.

[7] All MATPlaut codes follow MATLAB help document.