

Bifurcation using AUTO2000 and the Auto2000 Tellurium Plugin

Continue with some bifurcation

1.1 Introduction

The AUTO2000 plugin serves as a front-end for the AUTO2000 library, which is a library for continuation and bifurcation problems in ordinary differential equations ¹.

Current limitations: Multiple continuation parameters are not supported, i.e. only one parameter can be selected for any continuation problem.

Available properties in the auto2000 plugin are documented in the next section.

¹AUTO2000 by Eusebius J. Doedel , Randy C. Paffenroth, Alan R. Champneys, Thomas F. Fairgrieve, Yuri A. Kuznetsov, Bart E. Oldeman, Björn Sandstede and Xianjun Wang. See <http://www.dam.brown.edu/people/sandsted/publications/auto2000.pdf>.

1.2 Plugin Properties

The AUTO library have numerous properties that has been wrapped and made available to a plugin client. Each property is listed below with its data type, default value and a short description. For the exact usage and a more in detail description please consult the main AUTO2000 manual.

| Property Name | Data Type | Default Value | Description |
|--------------------------------|-----------|---------------|---|
| SBML | string | N/A | SBML document as a string. Model to be used by AUTO |
| TempFolder | string | "." | Folder used by auto and the plugin for saving temporary files |
| KeepTempFiles | bool | false | Boolean indicating if temporary files should be deleted after an AUTO session or not |
| ScanDirection | string | "Positive" | Parameter instructing AUTO how to sweep its principal continuation parameter |
| PrincipalContinuationParameter | string | N/A | The principal continuation parameter (PCP) is the first parameter that AUTO will sweep. Currently only one parameter is supported, which then by default is the PCP |

| | | | |
|-------------------|---------------|-----|---|
| BifurcationPoints | vector<int> | N/A | This integer vector holds the exact point number (in the sequence of all output data) for an AUTO solution point. It can be used together with the labels in the bifurcationlabels property to assist in plotting a bifurcation diagram |
| BifurcationLabels | stringList | N/A | The bifurcation labels string list holds the AUTO designated solution type label for a solution point, as found in the bifurcationpoints property. Consult the AUTO documentation for possible label types and their meaning |
| BifurcationData | telluriumData | N/A | The BifurcationData property holds the bifurcation diagram after a session. First column is the values of the selected parameter, and successive columns are selected species |

The following properties are used internally by the auto library. Depending on the problem at hand, they may need to be tweaked.

| | | | |
|-------|--------|-----|--|
| fort2 | string | N/A | Property containing the content of the AUTO temporary file, fort.2. Fort.2 is the input file for AUTO and created by the plugin |
| fort3 | string | N/A | Property containing the content of the AUTO temporary file, fort.3. The content of fort.3 file is undocumented in AUTO's documentation |
| fort6 | string | N/A | Property containing the content of the AUTO temporary file, fort.6. The content of fort.6 file is a bifurcation session summary |
| fort7 | string | N/A | Property containing the content of the AUTO temporary file, fort.7. The content of fort.7 file is a bifurcation diagram on success |
| fort8 | string | N/A | Property containing the content of the AUTO temporary file, fort.8. The content of fort.8 file contain various statistics from the session |
| fort9 | string | N/A | Property containing the content of the AUTO temporary file, fort.8. Diagnostic messages, convergence history, eigenvalues, and Floquet multipliers are written in fort.9 |

| | | | |
|------|-------------|-----|--|
| NDIM | int | 1 | The NDIM property correspond to the dimension of the system of equations |
| IPS | int | 1 | Constant defining the problem type |
| IRS | int | 1 | This constant sets the label of the solution where the computation is to be restarted. |
| ILP | int | 1 | Fold detection; 1=ON, 0=OFF |
| NICP | vector<int> | N/A | Property denoting the number of free parameters |
| ICP | int | N/A | Free parameters |
| NTST | int | 15 | The number of mesh intervals |
| NCOL | int | 3 | The number of collocation points per mesh interval |
| IAD | int | 3 | Mesh adaption every IAD steps; 0=OFF |
| ISP | int | 1 | Bifurcation detection; 0=OFF, 1=BP(FP), 3=BP(PO,BVP), 2=all |
| ISW | int | 1 | Branch switching: 1=normal, -1=switch branch (BP, HB, PD), 2=switch to two-parameter continuation (LP, BP, HB, TR) 3=switch to three-parameter continuation (BP) |

| | | | |
|------|--------|-------|--|
| IPLT | int | 0 | This constant allows redefinition of the principal solution measure, which is printed as the second (real) column in the fort.7 output-file. See AUTO manual for possible settings |
| NBC | int | 0 | Number of boundary conditions |
| NINT | int | 0 | Number of integral conditions |
| NMX | double | 1000 | Maximum number of steps |
| RL0 | double | 0.01 | The lower bound on the principal continuation parameter |
| RL1 | double | 30 | The upper bound on the principal continuation parameter |
| A0 | double | 0 | The lower bound on the principal solution measure |
| A1 | int | 10000 | The upper bound on the principal solution measure |
| NPR | int | 50 | Save the solution in the solution file every NPR continuation steps |
| MXBF | int | -1 | Automatic branch switching for the first MXBF bifurcation points if IPS=0, 1 |
| IID | int | 0 | Control diagnostic output; 0=none, 1=little, 2=normal, 4=extensive |

| | | | |
|-------|-------------|-------|--|
| ITMX | int | 8, | Maximum number of iterations for locating special solutions/points |
| ITNW | int | 5, | Maximum number of correction steps |
| NWTN | int | 3, | Corrector uses full newton for NWTN steps |
| JAC | double | 0, | User defines derivatives; 0=no, 1=yes |
| EPSL | double | 1e-8 | Property setting the convergence criterion for parameters |
| EPSU | double | 1e-8 | Property setting the convergence criterion for solution components |
| EPSS | double | 1e-6 | Property setting the convergence criterion for special points |
| DS | double | 0.001 | Session start step size |
| DSMIN | double | 1e-5 | Minimum continuation step size |
| DSMAX | double | 0.1 | Maximum continuation step size |
| IADS | int | 1 | Step size adaption every IADS steps; 0=OFF |
| NTHL | int | 0 | The number of modified parameter weights (for BVP) |
| THL | vector<int> | N/A | List of parameter weights |
| NTHU | int | 0 | The number of modified solution component weights (for BVP) |
| THU | vector<int> | N/A | List of solution weights |

| | | | |
|------|-------------|-----|--|
| NUZR | int | 0 | The number of user output points specified |
| UZR | vector<int> | N/A | List of values for user defined output |

Table 1.1: Plugin Properties

1.3 The `execute(bool inThread)` function

The `execute()` function will start a bifurcation session. Depending on the problem at hand, the algorithm may run for a long time.

The `execute(bool inThread)`, do support a boolean argument indicating if the execution of the plugin work will be done in a thread, or not. If set to false, i.e. executing `execute(false)`, the function will be a blocking function and will not return until the plugin work is done. If, on the other hand, the client are setting the argument to true, the `execute(true)` will return immediately and the plugin work will be executed in a thread. A client can use the `isPluginDone(plugin)` to query the status of the plugin progression.

The `inThread` argument defaults to **false**.

1.4 Plugin Events

The auto2000 plugin are using all of a plugins available plugin events, i.e. the *PluginStarted*, *PluginProgress* and the *PluginFinished* events.

The available data variables for each event are internally treated as *pass through* variables, so any data, for any of the events, assigned prior to the plugins execute function (in the `assignOn()` family of functions), can be retrieved *unmodified* in the corresponding event function.

| Event | Arguments | Purpose and argument types |
|----------------|--------------|---|
| PluginStarted | void*, void* | Signal to application that the plugin has started. Both parameters are <i>pass through</i> parameters and are unused internally by the plugin. |
| PluginProgress | void*, void* | Communicating progress of fitting. Both parameters are <i>pass through</i> parameters and are unused internally by the plugin. |
| PluginFinished | void*, void* | Signals to application that execution of the plugin has finished. Both parameters are <i>pass through</i> parameters and are unused internally by the plugin. |

Table 1.2: Plugin Events

1.5 Python example

The following Python script illustrate how the auto plugin can be invoked, how to set its properties and finally how to plot a bifurcation diagram.

```

1  from telplugins import *
2
3  try:
4      sbmlModel = "BIOMD0000000203.xml"
5      auto = Plugin("tel_auto2000")
6
7      #print auto.listOfPropertyNames()
8
9      #Setup Auto Property
10     auto.setProperty("SBML", readAllText(sbmlModel))
11
12     #Auto specific properties
13     auto.setProperty("ScanDirection", "Positive")
14     auto.setProperty("PrincipalContinuationParameter", "A")
15     auto.setProperty("PCPLowerBound", 10)
16     auto.setProperty("PCPUpperBound", 200)
17
18     #Max number of points
19     auto.setProperty("NMX", 5000)
20
21     #Execute the plugin
22     auto.execute()
23
24     # Bifurcation summary
25     print "Summary: " + auto.BifurcationSummary
26
27     #Plot Bifurcation diagram
28     pts = auto.BifurcationPoints
29     lbls = auto.BifurcationLabels
30     biData = auto.BifurcationData
31
32     biData.plotBifurcationDiagram(pts, lbls)
33
34     print "Done"
35
36 except Exception as e:
37     print "There was a problem: " + 'e'

```

Listing 1.1: Bifurcation example using a complex model.

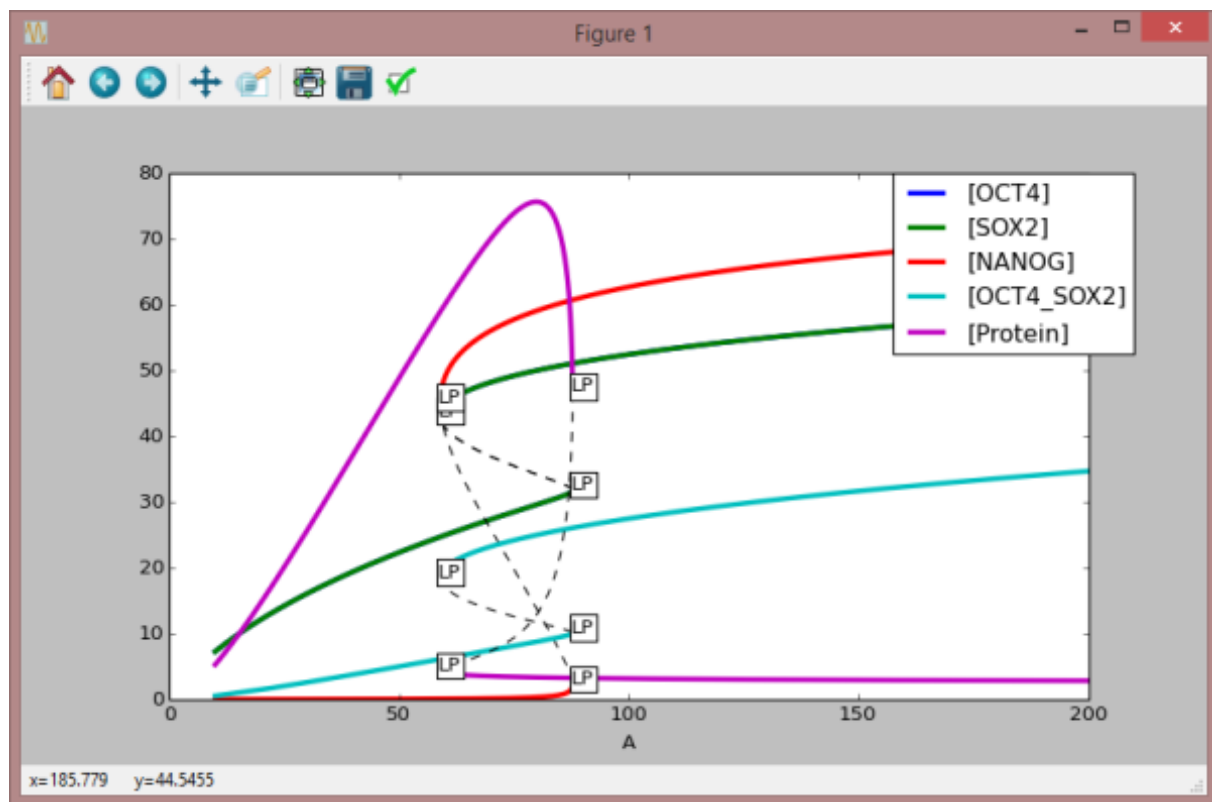


Figure 1.1: Output for the example script above.