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.

[1] AUTO2000 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	”.”	Tempfolder used by auto and the plugin for saving temporary files
mKeepTempFiles	bool	false	Boolean indicating if temporary files should be deleted after an AUTO session or not
mScanDirection	string	”Positive”	Parameter instructing AUTO how to sweep its principal continuation parameter
mPrincipalContinuationParameter	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 asssist in plotting a bifurcation diagram
BifurcationLabels	stringList	N/A	The bifurcation labels stinglist 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 succesive 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.

mFort2	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
mFort3	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
mFort6	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
mFort7	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
mFort8	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
mFort9	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 intervalls
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 trough* 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 *pass trough* parameters and are unused internally by the plugin.
PluginProgress	void*, void*	Communicating progress of fitting. Both parameters are *pass trough* parameters and are unused internally by the plugin.
PluginFinished	void*, void*	Signals to application that execution of the plugin has finished. Both parameters are *pass trough* 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.

```
from telplugins import *

try:
    sbmlModel ="BIOMD0000000203.xml"
    auto = Plugin("tel_auto2000")

    #print auto.listOfPropertyNames()

    #Setup Auto Propertys
    auto.setProperty("SBML", readAllText(sbmlModel))

    #Auto specific properties
    auto.setProperty("ScanDirection", "Positive")
    auto.setProperty("PrincipalContinuationParameter", "A")
    auto.setProperty("PCPLowerBound", 10)
    auto.setProperty("PCPUpperBound", 200)

    #Max numberof points
    auto.setProperty("NMX", 5000)

    #Execute the plugin
    auto.execute()

    # Bifurcation summary
    print "Summary: " + auto.BifurcationSummary

    #Plot Bifurcation diagram
    pts     = auto.BifurcationPoints
    lbls    = auto.BifurcationLabels
    biData  = auto.BifurcationData

    biData.plotBifurcationDiagram(pts, lbls)

    print "Done"

except Exception as e:
    print "There was a problem: " + ‘e‘
```

Listing 1.1: Bifurcation example using a complex model.

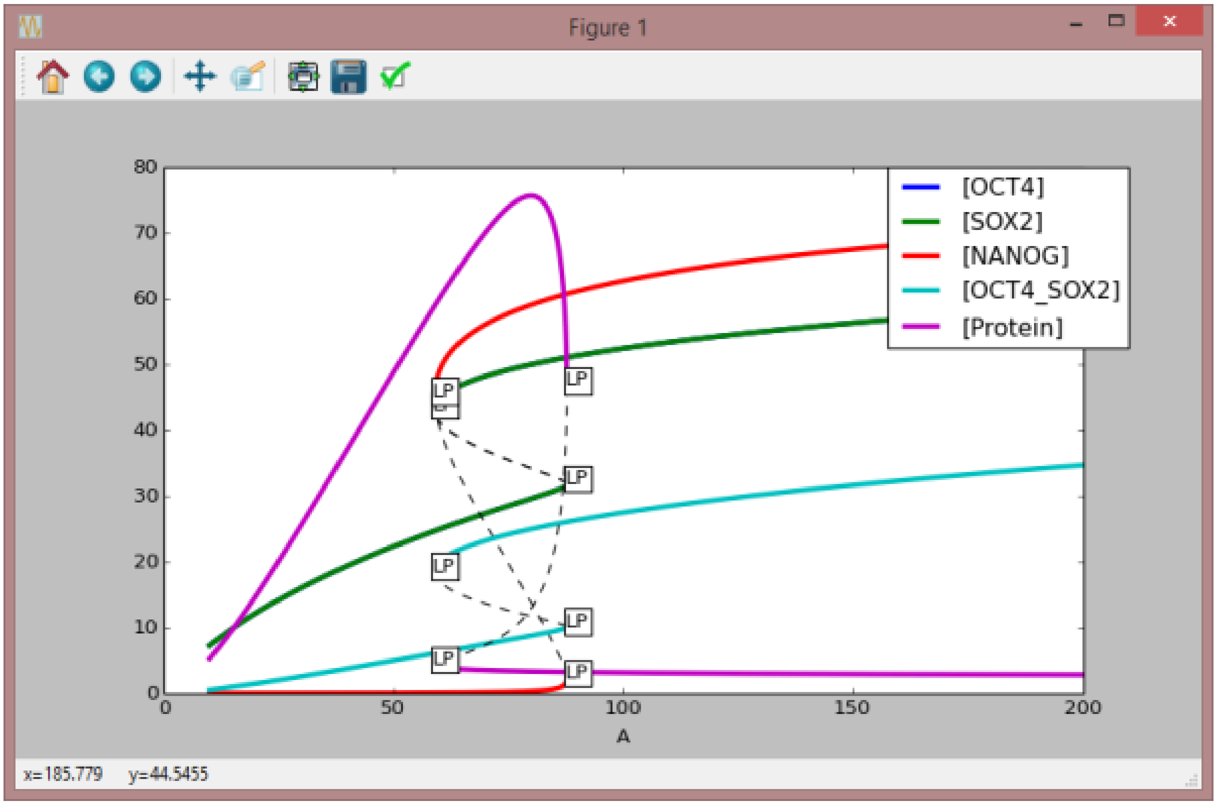

Figure 1.1: Output for the example script above.