AUTO

Continuation and Bifurcation software for Ordinary Differential Equations

AUTO軟體介紹

- Continuation(連續) and Bifurcation(分岔) software for Ordinary Differential Equations
- Windows系統透過命令提示字元 命令提示字元 操作
- AUTO軟體使用python語言
  - 需安裝python，呼叫AUTO的library使用指令
  - 撰寫AUTO執行檔使用python語言
  - PyPLAUT繪製平衡點
- 使用AUTO時，所需檔案:
  - 1. equation-file: fortran或c檔案，需安裝fortran或c編譯器
  - 2. constant-file:定義AUTO計算方式，如維度、參數數量、誤差…
  - 3. auto-file:將所有執行指令寫在檔案內，方便執行(非必要)

非線性系統步驟

Step 1. Modeling

$$\dot{x} = f(x,\mu), \qquad x \in R^n, \qquad \mu \in R^m$$

Step 2. Setting point(equilibrium point平衡點, or fixed point固定點)

$$0 = \dot{x} = f(x_e,\mu_e),\ \ \mu_e$$ ：外加參數or Input

$$\Rightarrow 0 = f(x_e,\mu_e)$$ →可透過MatLab解ODE

$$\Rightarrow x_e = h(\mu_e)$$

Step 3. Stability analysis

Let $\tilde{x} = x - x_e,\ \tilde{\mu} = \mu - \mu_e$

$$\Rightarrow \dot{\tilde{x}} = \dot{x} - \dot{x_e}$$ (兩邊微分)

$$= f(x,\mu) - 0$$

在(x_e,μ_e)展開→ $= f(x_e,\mu_e) + \frac{\partial f}{\partial x}(x_e,\mu_e)(x - x_e) + \frac{\partial f}{\partial \mu}(x_e,\mu_e)(\mu - \mu_e) + O(2)$

非線性系統步驟(續)

Step 3. Stability analysis(cont.)

$$\dot{\tilde{x}} = f(x_e, \mu_e) + \frac{\partial f}{\partial x}(x_e, \mu_e)(x - x_e) + \frac{\partial f}{\partial \mu}(x_e, \mu_e)(\mu - \mu_e) + O(2)$$

Let $\frac{\partial f}{\partial x}(x_e, \mu_e) \triangleq A \quad and \quad \frac{\partial f}{\partial \mu}(x_e, \mu_e) \triangleq B$

$$\Rightarrow \dot{\tilde{x}} = 0 + A\tilde{x} + B\tilde{\mu} + O(2)$$

$$\cong A\tilde{x} + B\tilde{\mu}$$ (線性系統)

AUTO重點在解: $0 = 0 + A\tilde{x} + B\tilde{\mu} + O(2)$

計算$\Rightarrow \tilde{x} = -A^{-1}B\tilde{\mu} + O(2)$

若A是invertible→一個點

若非→分歧(多點),bifurcation

AUTO所需檔案

以下舉例的專案名稱以*代表

1. The equations-file(*.f90 或 *.c)

   可以是fortran 或是 c檔案(範例皆以c檔案為主)

2. The constants-file(c.*)

3. The auto script file(*.auto)

Equations-file (*.c)

共有6個User-supplied routines　參考: AUTO-07P manual, Chapter 3

1. FUNC: 定義微分方程式 $\dot{x} = f(x, \mu)$
2. STPNT: 當IRS=0時，STPNT會被呼叫，在此定義起始解starting solution (x, μ)，並且此解不應該是分支點
3. BCND: 定義邊界條件
4. ICND: 定義積分條件
5. FOPT: 定義objective functional
6. PVLS: 定義“solution measures”

Equations-file (*.c) (續)

```
#include "auto_f2c.h"
#include <math.h>
int func (integer ndim, const double *u, const integer *icp,
    const double *par, integer ijac, double *f, double *dfdu, double *dfdp)
{
    /* System generated locals */
    integer dfdu_dim1 = ndim, dfdp_dim1 = ndim;
    /* User defined locals */
    double  x;
    double  mu;
    x=u[0];
    mu=par[0];
    f[0] = mu-x*x;
    return 0;
}
/* ---------------------------------------------------------------------- */
/* ---------------------------------------------------------------------- */
int stpnt (integer ndim, double t, double *u, double *par)
{
    par[0]=1;
    u[0]=1;
    return 0;
}
```

Constants-file (c. *)

參考: AUTO-07P manual, Chapter 10

```
# Default AUTO Constants file
e = '', s='', dat='', sv=''
unames = {}, parnames = {}
U = {}, PAR = {}
NDIM= 2, IPS = 1, IRS = 0, ILP = 1
ICP = [1]
NTST= 20, NCOL= 4, IAD = 3, ISP = 2, ISW = 1, IPLT= 0, NBC= 0, NINT= 0
NMX= 0, NPR= 0, MXBF= 10, IID = 2, ITMX= 9, ITNW= 5, NWTN= 3, JAC= 0
EPSL= 1e-07, EPSU = 1e-07, EPSS = 1e-05
DS = 0.01, DSMIN= 0.005, DSMAX= 0.1, IADS= 1
NPAR= 36, THL = {}, THU = {}
RL0=-1.7976e+308, RL1=1.7976e+308, A0=-1.7976e+308, A1=1.7976e+308,
UZR = {}, UZSTOP = {}, SP = [], STOP = []
IIS = 3, IBR=0, LAB=0, TY=''
NUNSTAB = -1, NSTAB = -1, IEQUIB = 1, ITWIST = 0, ISTART = 5
IREV = [], IFIXED = [], IPSI = []
```

Script file(*.auto)

Running AUTO using Python Commands. 參考: AUTO-07P manual, Chapter 4.14 (基本指令)

以此圖為範例說明

```
print "\n***Generate starting data***"
ab=load(e='saddle_node',c='saddle_node')
ab=run(ab)
ab = ab + run(ab,DS='-',IRS=1)
ab = relabel(ab)
save(ab,'saddle_node')
p = plot(ab)
p.config(stability='true')
```

*.auto內可以使用python指令與AUTO內部指令(如:run()、relabel() …)

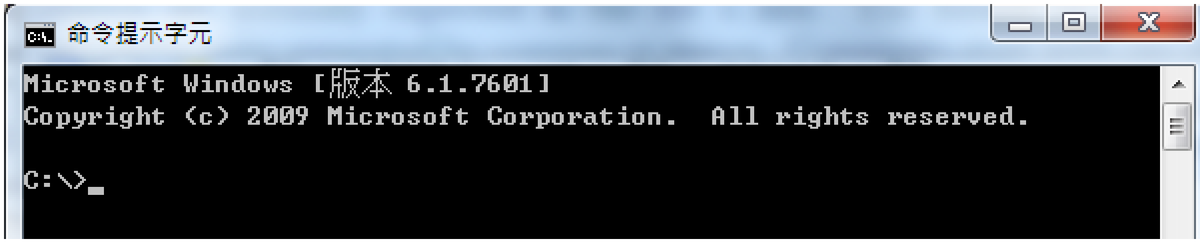
將所有指令寫在script file內，讓AUTO一起執行，不用分次輸入指令。

The Plotting tool pyPLAUT可以繪製分岔圖

參考: AUTO-07P manual, Chapter 4.11

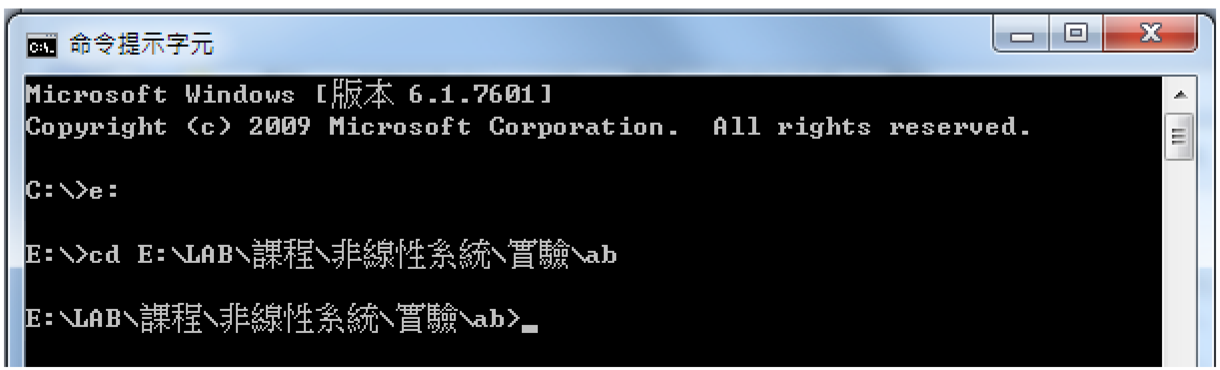
操作方式

1. 開啟命令提示字元



2. 切換至範例程式檔案所在位置(範例名稱:ab)

- E:\LAB\課程\非線性系統\實驗\ab



執行AUTO步驟

3.輸入auto.py

```
E:\LAB\課程\非線性系統\實驗\ab>auto.py
Python 2.7.6 (default, Nov 10 2013, 19:24:18) [MSC v.1500 32 bit (Intel
Type "help", "copyright", "credits" or "license" for more information.
(AUTOInteractiveConsole)
AUTO> cd ..
```

- 進入AUTO介面，一樣可以使用指令切換工作目錄(linux指令用法)

4.輸入指令ab=load(equation=‘ab’)或是ab=load(e=‘ab’)

- 載入equation-file
- 目錄內必須要有.f90或.c檔案
- 顯示Runner configured

```
AUTO> ab=load(equation='ab')
Runner configured
```

5.輸入指令ab=load(ab , constants=‘ab.1’)或是ab=load(ab ,c=‘ab.1’)

- 載入constants-file
- 目錄內必須要有c. 檔案(例如右方圖片中: c.ab , c.ab.1)
- constants-file命名規則為c.“檔案名稱”
- 顯示Runner configured

ab目錄內的檔案

```
ab.auto
ab.f90
autorc
c.ab
c.ab.1
c.ab.2
c.ab.3
clean.auto
plaut04.rc
```

執行AUTO步驟(續)

6. 輸入指令run(ab)
   - gfortran 編譯equation-file(ab.f90)，產生執行檔ab.exe
   - Starting ab …開始計算

```
AUTO> run(ab)
gfortran  -O -c ab.f90 -o ab.o
gfortran  -O ab.o -o ab.exe C:\auto\07p\lib\*.o
Starting ab ...
ab ... done
<_=bifDiag instance at 0x0282d830>
AUTO> _
```

   - 計算後，產生3個檔案
     - 1. fort.7 (b.*) : bifurcation diagram file
     - 2. fort.8 (s.*) : solution file
     - 3. fort.9 (d.*) : diagnostic messages, convergence history, eigenvalues, and Floquet multipliers are written in fort.9

Types of points

Type	Short Name	Number
No Label	No Label	
Branch point (algebraic problem)	BP	1
Fold (algebraic problem)	LP	2
Hopf bifurcation (algebraic problem)	HB	3
Regular point (every NPR steps)	RG	4
User requested point	UZ	-4
Fold (ODE)	LP	5
Bifurcation point (ODE)	BP	6
Period doubling bifurcation (ODE)	PD	7
Bifurcation to invariant torus (ODE)	TR	8
Normal begin or end	EP	9
Abnormal termination	MX	-9

Table 4.5: This table shows the various types of points that can be in solution and bifurcation diagram files, with their short names and numbers.

範例:Saddle-node bifurcation

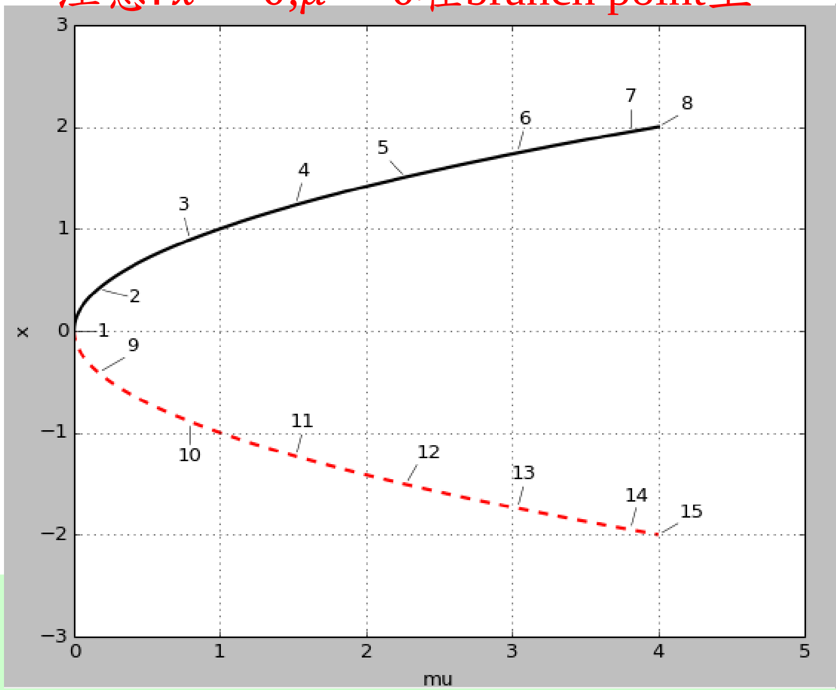
典型的saddle-node bifurcation微分方程式為：

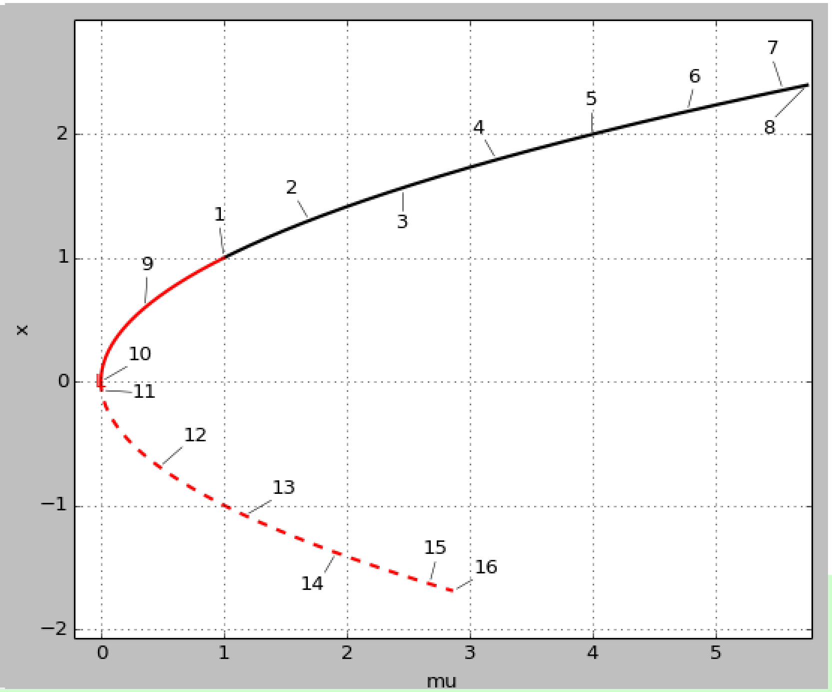
$$\frac{dx}{dt} = \mu + x^2$$

其中 x: 狀態變數state variable, μ: 分岔參數bifurcation parameter

AUTO計算結果: 左圖起始點為$x = 0$,$\mu = 0$；右圖起始點為$x =1$,$\mu = 1$。

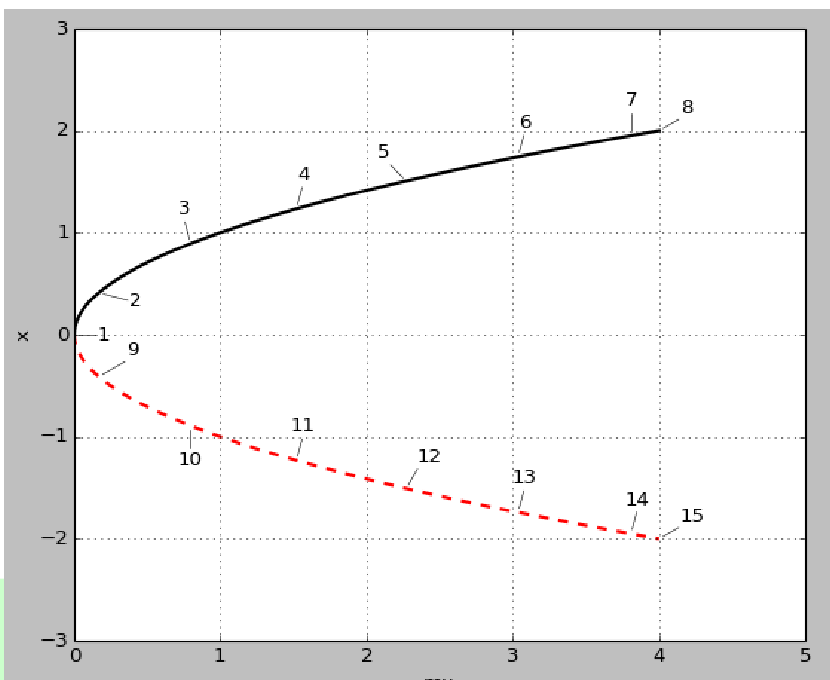
注意:$x = 0$,$\mu = 0$在branch point上

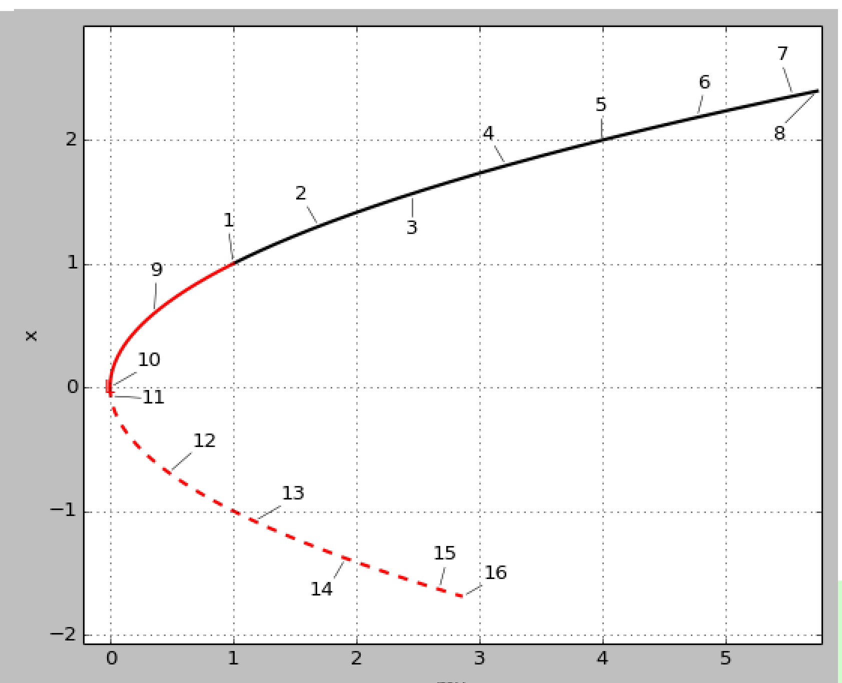




範例:Saddle-node bifurcation(續)

```
print "\n***Generate starting data***"
ab=load(e='saddle_node',c='saddle_node')
ab=run(ab)
ab = ab + run(ab,DS='-',IRS=1)
ab = relabel(ab)
save(ab,'saddle_node')
p = plot(ab)
p.config(stability='true')
```





練習: Transcritical bifurcation

Transcritical bifurcation的normal form為：

$$\frac{dx}{dt} = \mu x - x^2$$

其中 x: 狀態變數state variable, μ: 分岔參數bifurcation parameter

AUTO計算結果: