Stability of 2-Dimensional Systems: A Neural Network Approach

Nikos E.Mastorakis

Military Institutions of University Education (MIUE),
Hellenic Naval Academy, Chair of Computer Science,
Terma Hatzikyriakou, 18539, Piraeus, GREECE.

Tel-Fax: +301 777 5660

Abstract . In this paper, the problem of 2-D (two-dimensional) systems' stability is investigated using (artificial) neural networks. In particular, the conditions provided by Jury's test are used for the synthesis of appropriate neural networks with very easy hardware implementation. Examples are also given.

Key-words: Neural Networks, Applications of Neural Networks in Multidimensional Systems Stability, Multidimensional Systems. IMACS/IEEE CSCC'99 Proceedings, Pages: 1011-1018

1 Introduction

Two-dimensional (2-D) systems have been attracting the attention of scientific and engineering community working on the area of applied mathematics, systems and control as well as computer science since the early seventies.

Nowadays, 2-D systems form a multi-disciplinary field with direct impacts in many areas of science and engineering: *a) Image processing and its applications in biomedicine:* image deblurring, image filtering, X-ray enhancement, seismology, moving-objects recognition, remote-sensing, computer vision, robots vision. Radar and sonar technology, ultrasound echography, lightwave technology, thermography, blood cell analysis, computerized tomography. *b) Systems Theory:* General Systems Theory, Multivariable networks modelling, Large Scale systems analysis. *c) Applied Mathematics:* Applied electromagnetic problems, finite elements methods, solution of Laplace Equation, Helmholtz Equation and Wave Equation in cartesian and non-cartesian coordinates, underwater acoustics, elasticity problems, computerised strength of materials, heat diffusion. *d) Modelling:* Discretization of pollution, absorption processes, computerised fluid mechanics,etc.).

The stability of 2-D systems has received much attention during the last two decades and many stability theorems have been published. Excellent overviews of the stability problem and of the theorems and the tests associated with it can be found in [1÷3].

In this paper, the problem of 2-D system stability is examined on the light of Neural Networks. Neural Networks in their software version (simulation) are efficient and strong algorithms which can solve extremely challenging problems in science (statistics, optimization, stochastic models, simulation, etc.) and engineering (pattern recognition, telecommunication systems, image processing, control, noise cancellation, etc) [4,5]. The same is also true for their hardware implementation [6]

It is known that each neural network has at least two physical components: connections weights and processing elements. The combination of these two components creates a neural network topology. A very popular Neural Network (NN) for many engineering applications is the 3-layer Neural Network shown in Fig.1

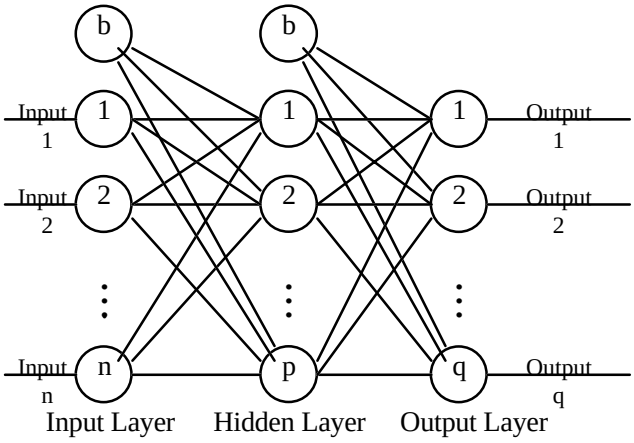

Fig.1: The General 3-layer Neural Network

Each Processing Element is represented by a circle. The interconnections are represented by the associated arrows. The Processing Elements with the letter "b" inside are bias Processing Elements. In this paper, learning of this Neural Network is first attempted by the Back-Propagation algorithm [4,5]. Each Processing Element in the Hidden Layer is characterized by the following input-output function (ramp threshold function).

$$y = f(x) = \begin{cases} 1 & if \quad x > 0 \\ 0 & if \quad x \le 0 \end{cases} \tag{1}$$

where x is the sum of all inputs in the particular Processing Element. The sigmoid threshold function, with the positive parameter k, is a continuous version of the ramp threshold function

$$y = f(x) = \frac{1}{1 + e^{-k \cdot x}} \tag{2}$$

The objective in this paper is to train the 3-layer Neural Network in order to recognize if a 2-D system is stable [1], [7]. It is reminded that a 2-D system with the transfer function

$$G(z_1, z_2) = \frac{B(z_1, z_2)}{A(z_1, z_2)} \tag{3}$$

is (Hurwitz, i.e. Bounded Input Bounded Output - BIBO) stable if and only if

$$A(0, z_2) \neq 0, \qquad \text{for} \qquad |z_2| \le 1 \tag{3.1}$$

$$A(z_1, z_2) \neq 0, \qquad \text{for} \qquad |z_1| \le 1, \qquad |z_2| = 1 \tag{3.2}$$

where we have assumed that $A(z_1, z_2)$ and $B(z_1, z_2)$ are coprime polynomials in the independent complex variables z_1 and z_2 as well as that there are no nonessential singularities of the second kind on the closed unit bidisk, i.e. there are no points (z_1, z_2) with $|z_1| \le 1$ and $|z_2| \le 1$ such that $A(z_1, z_2) = B(z_1, z_2) = 0$ simultaneously [8÷11].

It is also known that in 2-D systems' stability, we are interesting for (3.2) since (3.1) is an 1-D condition which is easy to check using any 1-D stability test. Condition (3.2) is more difficult since it includes two variables. Henceforth, we are dealing only with (3.2). We denote:

$$A(z_1, z_2) = \sum_{i_1=0}^{N_1} \sum_{i_2=0}^{N_2} a(i_1, i_2) z_1^{i_1} z_2^{i_2} \tag{4}$$

In various numerical experiments -see Appendix 1 (mainly for the training of Neural Network) the following condition equivalent to (3.1) & (3.2) conditions can be tested:

$$\sum_{i_1=0}^{\infty} \sum_{i_2=0}^{\infty} |h(i_1, i_2)| < \infty \tag{5}$$

where $h(n_1, n_2)$ is the impulse response of the system having the transfer function of (3). This condition is checked via the first program in Appendix 1, written in C language. Some problems with the dimensions of the matrix y seems to be overcome using the second program written in MatLab (for Windows '98).

2 Neural Networks for checking 2-D systems Stability.

To train a Neural Network, we need a great many patterns of polynomials $A(z_1, z_2)$ or equivalently of the coefficients of $\{a(i_1, i_2)\}$. Each polynomial

$A(z_1,z_2)$ is faced as a $(N_1+1)(N_2+1)$-tuple pattern of the coefficients $a(i_1,i_2)$. Since one coefficient of $A(z_1,z_2)$, for example $a(0,0)$, can be normalized to 1, each polynomial $A(z_1,z_2)$ is faced as a $(N_1+1)(N_2+1)-1 = N_1N_2+N_1+N_2$-tuple pattern of the coefficients $a(i_1,i_2)$. So for our 3-layer Neural Network (Fig.1) we have $n = N_1N_2+N_1+N_2$, $q=1$ (i.e we need only one output which will give 1 when the examined 2-D system is Stable and 0 when the examined 2-D system is unstable). We also select $p = N_1N_2+N_1+N_2$.

Unfortunately, after long calculations (using programs 1, 2 of Appendix 1 which check Condition (5) or program 3 which check Jury's test) and after a great number of examples (~ 2000) the result of the Neural Network training is not satisfactory because we obtain an error ~ 0.18. For this reason, we adopt another approach, which seems to be better. This approach is as follows:

For checking (3.2), we write Equation (4) as follows

$$A(z_1,z_2) = \sum_{i_1=0}^{N_1} a_{i_1}(z_2) z_1^{i_1} \qquad (6)$$

with $a_{i_1}(z_2) = \sum_{i_2=0}^{N_2} a(i_1,i_2) z_2^{i_2}$.

We consider z_2 constant, with $|z_2| = 1$ i.e. $z_2 = e^{j\theta}$ with $\theta \in [0,2\pi]$ constant. Then, we consider the classical Jury Test for 1-D systems. We form the Jury Table as follows (we always normalize a_0 to 1) where by $\bar{z}$ we denote the complex conjugate of z.

$$\begin{array}{lllllll}
a_0 & a_1 & a_2 & \dots & a_{N_1-2} & a_{N_1-1} & a_{N_1} \\
\bar{a}_{N_1} & \bar{a}_{N_1-1} & \bar{a}_{N_1-2} & \dots & \bar{a}_2 & \bar{a}_1 & \bar{a}_0 \\
b_0 & b_1 & b_2 & \dots & b_{N_1-2} & b_{N_1-1} & \\
\bar{b}_{N_1-1} & \bar{b}_{N_1-2} & \bar{b}_{N_1-3} & \dots & \bar{b}_1 & \bar{b}_0 & \\
c_0 & c_1 & c_2 & \dots & c_{N_1-2} & & \\
\bar{c}_{N_1-2} & \bar{c}_{N_1-3} & \bar{c}_{N_1-4} & \dots & \bar{c}_0 & & \\
\vdots & \vdots & & & & & \\
s_0 & s_1 & & & & & \\
\bar{s}_1 & \bar{s}_0 & & & & & \\
t_0 & & & & & &
\end{array}$$

where

$$b_k = det\begin{bmatrix} a_0 & a_{N-k_1} \\ \bar{a}_{N_1} & \bar{a}_k \end{bmatrix} \qquad k = 0,\dots,N_1-1 \qquad (7)$$

c's are formed by b's exactly in the same way that b's are formed by a's. The system in question is stable if we have $b_0 > 0, c_0 > 0, \dots, s_0 > 0, t_0 > 0$ for each z_2, with $|z_2| = 1$ i.e. $z_2 = e^{j\theta}$ with $\theta \in [0,2\pi]$.

This idea is implemented via the following Neural Network.

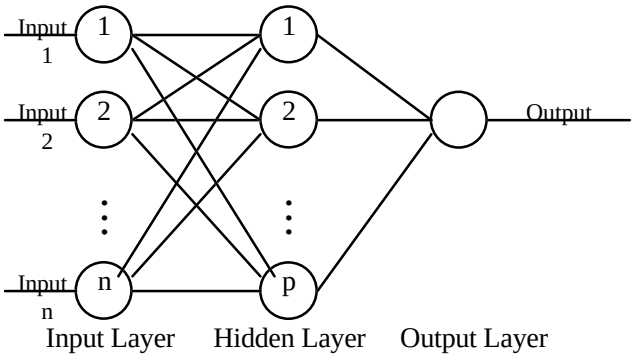

Fig.2: Neural Network based on the Jury Test

$n = 2N_1$ (for each complex coefficient, i.e. a_1, a_2, $\ldots, a_{N_1-2}$, a_{N_1-1}, a_{N_1}, we need one input for the Real Part and another for the Imaginary Part). The Processing Element in the Hidden Layer are dependent on the relations of the Jury Test. The unique Output yields 1 if and only if all the outputs from the Hidden Layer (which also are the inputs to the final Output) are 1 for all θ, with $\theta \in [0, 2\pi]$. This holds if and only if the considered 2-D system is *stable*. This unique Output is 0 if and only if some outputs from the Hidden Layer (which also are the inputs to the final Output) are 0 for some θ, with $\theta \in [0, 2\pi]$. This holds if and only if the considered 2-D system is *unstable*.

The following example presents a Neural Network for checking the stability of (3.2) in the special case with $N_1 = 2, N_2 = 2$. This Neural Network has a very simple Hardware Implementation [6].

So, we set in Input 1, x_1, the real part of $a_1 = a_1(z_2)$, in Input 1, x_2, the imaginary part of $a_1 = a_1(z_2)$, in Input 3, x_3, the real part of $a_2 = a_2(z_2)$ and in Input 4, x_4, the imaginary part of $a_2 = a_2(z_2)$. Furthermore, in the Hidden Layer, in the Processing Element 1, we have the total input $1 - x_3^2 - x_4^2$ while in the Processing Element 2, we have the total input $\left(1 - x_3^2 - x_4^2\right)^2 - \left(x_1 - x_1x_3 - x_2x_4\right)^2\left(x_2 + x_2x_3 - x_1x_4\right)^2$. Each Processing Element in the Hidden Layer is characterized by the ramp threshold function of Equation (1). The final output of the Neural Network is 1 if the two outputs of the Hidden Layer is 1 for all θ, with $\theta \in [0, 2\pi]$, else is 0. If this final output is 1 the considered 2-D system is *stable*, else is *unstable*.

The following numerical example is taken from [12].

$$f(z_1, z_2) = 8z_1^2 - 7z_1z_2 - 12z_1 - z_2 + 13 \tag{8}$$

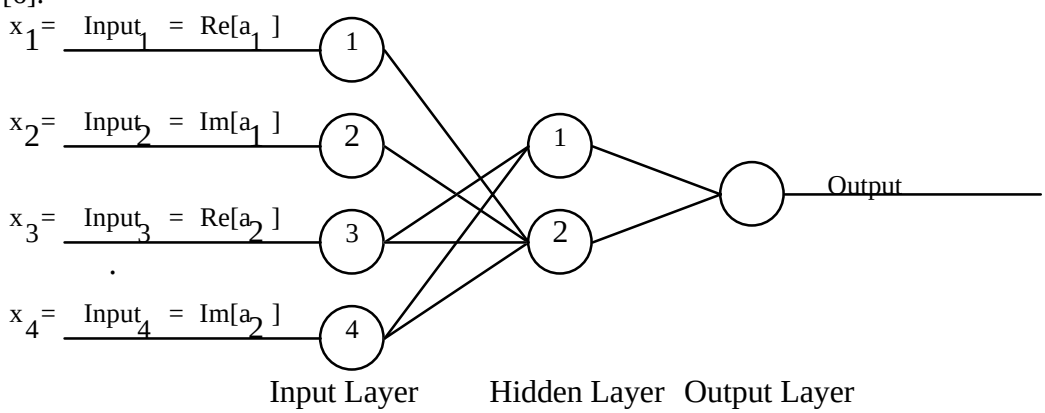

Fig.3: Neural Network based on the Jury Test in the special case with $N_1 = 2, N_2 = 2$

For this numerical example, the Input to Node 1 of the Hidden Layer while $\theta \in [0,2\pi]$ is designed in Fig.4

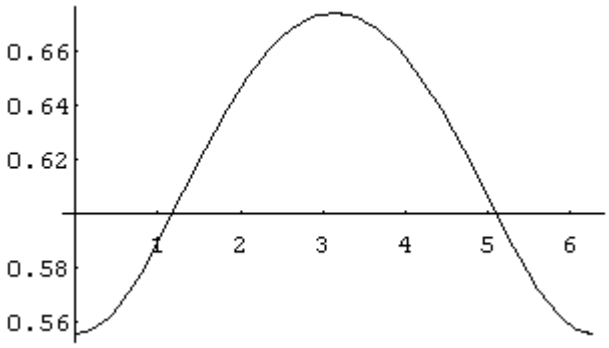

Fig.4: Input to Node 1 of the Hidden Layer

while the Output from Node 1 of the Hidden Layer ($\theta \in [0,2\pi]$) is designed in Fig.5

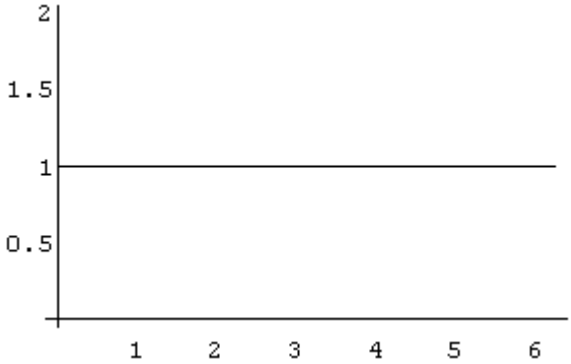

Fig.5: Output of Node 1 of the Hidden Layer

Also for this numerical example, the Input to Node 2 of the Hidden Layer while $\theta \in [0,2\pi]$ is designed in Fig.6

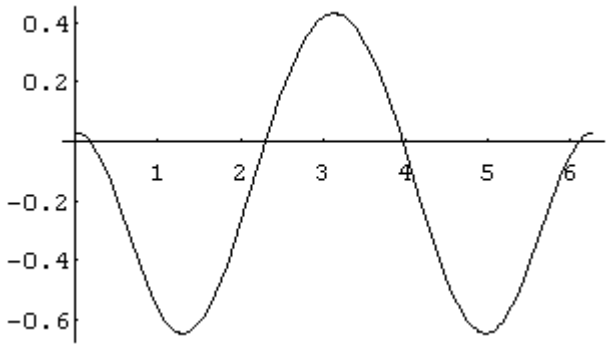

Fig.6: Input for Node 2 of the Hidden Layer

Also the output from Node 2 of the Hidden Layer is designed in Fig.7

Fig.7: Output of Node 2 of the Hidden Layer

We obtain that the Output of Node 2 of the Hidden Layer is not always 1 (for all θ, with $\theta \in [0,2\pi]$). Therefore our 2-D System is *unstable*. Equivalently, the final output of our Neural Network is 0.

3 Conclusion

In 2-D systems' stability problem, Artificial Neural Networks (or simply Neural Networks) can contribute and offer a new, modern and effective approach. These Neural Networks should be Hardwired Neural Networks in order to achieve a fast solution for the 2-D stability problem. Generally, these Hardwired Neural Networks can be proved very powerful tools in the whole 2-D systems study [13]. Work is in progress by the author in the area of Neural Networks applications in *m*-D Systems Stability (*m*>2).

References

[1] Jury, E.I., "Stability of multidimensional scalar and matrix polynomials", *Proceedings of the IEEE*, Vol.66, 9, pp.1018-10476, 1978.

[2] Jury, E.I., "Stability of multidimensional systems and related problems", *Multidimensional Systems - Techniques and Applications*, S.Tzafestas editor, New York : Marcell Dekker, 1985

[3] Kaczorek T., *2-D Linear Systems*, Springer-Verlang, Lecture Notes in Control and Information Sci., Berlin-Heidelberg, 1985.

[4] Mastorakis N., *Recent Advances in Circuits and Systems*, World Scientific Publishing Co., 1998,

[5] Eberhardt R., Simpson P. and Dobbins R., "Computational Intelligence PC Tools", Academic Press Professional, London, 1996.

[6] Woodburn R. and Murray A., "Pulse-Stream Techniques and Circuits for Implementing Neural Networks", Chapter 4, pp.103-119, in the "Neural Network Analysis, Architectures and Applications", Institute of Physics Publishing, Bristol and Philadelphia, 1997, (Browne A. editor),

[7] R.P.Roesser, "A discrete state-space model for image processing," *IEEE Trans. Automat. Contr.*, vol. AC-20, pp. 1-10, Feb. 1975.

[8] N K.Bose (Editor), *Multidimensional Systems Theory, Progress, Directions and Open Problems in Multidimensional Systems*, D. Reidel Publishing Company, Dordrecht, Holland 1985.

[9] N.E.Mastorakis," A Continuous Model for Two-Dimensional (LSI) Systems", *in the Plenary Session: Multidimensional Systems: New Models, The Relationship between Multidimensional Systems and Partial Differential Equations*. Circuits, Systems and Computers '96. International Conference with the approval and support of the Hellenic Navy General Staff. Hellenic Naval Academy, Piraeus, GREECE, July 15-17, 1996.

[10] T.S.Huang, "Stability of Two-Dimensional Recursive Filters", *IEEE Trans. Audio, Electroacoust.*, Vol. AU-20, pp.158-163, June 1972.

[11] N.K.Bose and E.I.Jury, "Positivity and Stability Tests for Multidimensional Filters", *IEEE Trans. Acoust., Speech, Signal Processing*, Vol. ASSP-22, pp.174-180, June 1974.

[12] M.N.S.Swamy, L.M.Roytman, and N.Marinovic, "A Necessary Condition for the BIBO Stability of 2-D Filters", *IEEE Trans. on Circ. and Syst.* - II: Analog and Digital Signal Processing, Vol.39, No.7, pp. 475-476, July 1992.

[13] Mastorakis N., *Recent Advances in Information Science and Technology*, World Scientific Publishing Co., 1998.

APPENDIX 1

C Program examing the stability of a 2-D system via the computation of the Impulse Response.

```
#include <stdio.h>

main()
{int i, j, k, l;
 float s, h, x, z;
 int N1=2;
 int N2=2;
 int M=100;
 float a[2][2];
 float y[100][100];
 char q;
 for(i=0;i<=N1;i=i++)
{ for(j=0;j<=N2; j=j++)
  {
     printf("a(%d,%d)=",i,j);
   scanf("%f",&x);
  a[i][j]=x;
  /* At this point, we give the coefficients */
  }}
for(i=0; i<=M; i=i++)
 { for(j=0; j<=M; j=j++)
  {y[i][j]=0;
/* At this point, we iniatilise OUTPUT y */
 }
}


h=0;
/* Variable h represents the Sum of the */
/* Absolute Values of the Impulse Response */

y[N1][N2]=1;

for(i=N1; i<=M; i=i++)
{
  for(j=N2; j<=M; j=j++)
 {
   if ((i==N1) && (j==N2)) {continue;}
   else
   {
   s=0;
   for(k=0; k<=N1; k=k++)
    {
     for(l=0; l<=N2; l=l++)
              {
               if ((k==0) && (l==0)) {continue;}
               else
               {
                s=s+a[k][l]*y[i-k][j-l];
                /* We Compute Impulse Response by its
Definition */;
             }
         }
      }
    y[i][j]=-s/(a[0][0]);
    z=y[i][j];
    printf("     %f",z);
    if (z>0) {h=h+z;} else {h=h-y[i][j];}
   if ((fmod(i,20)==0) && (fmod(j,20)==0) &&
(fmod(i,100)!=0))
   {printf("     %4.3f",h);}
/* We print the sum of the absolute */
/* values of the impulse response   */
/* at appropriate points */
  if     ((fmod(i,100)==0)     &&     (fmod(j,20)==0))
{printf("    %4.3f\n",h);}
    }
   }
  }
}
```

----------Some problems with the dimensions of the matrix y impose to write a MATLAB code as follows.

Matlab Program examing the stability of a 2-D system via the computation of the Impulse Response.

```
%  program stabil2 for MATLAB
%  for WINDOWS '95
%  This program examines the stability of a 2-%   D
system with N1 x N2 order
%  The program is based on the condition (4.1) %
of this paper.
%  We check the absolute summability of the
%  impulse respones

clear screen width height mwwidth mwheight left
bottom rect
flops(0);

M=400;
N1=2;
N2=2;

% At this point, we will give the coefficients
% ATTENTION: a(1,1) must be DIFFERENT
%  from Zero

a(1,1)=7;          % For our system this is a(0,0)
a(1,2)=2;          % For our system this is a(0,1)
a(1,3)=2;          % For our system this is a(0,2)
a(2,1)=2;          % For our system this is a(1,0)
a(2,2)=1;          % For our system this is a(1,1)
a(2,3)=1;          % For our system this is a(1,2)
a(3,1)=2;          % For our system this is a(2,0)
a(3,2)=1;          % For our system this is a(2,1)
a(3,3)=1;          % For our system this is a(2,2)

% At this point, we will initialise OUTPUT y
y1=zeros(1,M+1);
y2=y1';
y=y2*y1;
h=0;

y(N1+1,N2+1)=1;
t1=1;
t2=1;
for i=N1:M
```

```
for j=N2:M
   s=0;
       for k=0:N1
         for l=0:N2
            s=s+a(k+1,l+1)*y(i-k+1,j-l+1);
           end
       end
   y(i+1,j+1)=-s/(a(1,1));
   y(N1+1,N2+1)=1;
   h=h+abs(y(i+1,j+1));
        if i-20*fix(i/20)==0
           if j-20*fix(j/20)==0
            response(t1,t2)=h;
            if (20*t2>=M)
            t2=1;
            t1=t1+1;
            else t2=t2+1;
           end
        end
       end

            % We print the sum of the absolute
            % values of the impulse response
            % at appropriate points
end
end

response
```

Mathematica **Program checking the Neural Network of Fig.3 which is based on the Jury's Test.**

```
step=1;

For[i1=-1, i1<=1, i1=i1+step,
For[i2=-1, i2<=1, i2=i2+step,
For[i3=-1, i3<=1, i3=i3+step,
For[i4=-1, i4<=1, i4=i4+step,
For[i5=-1, i5<=1, i5=i5+step,
For[i6=-1, i6<=1, i6=i6+step,
For[i7=-1, i7<=1, i7=i7+step,
For[i8=-1, i8<=1, i8=i8+step,
katalog={i1,i2,i3,i4,i5,i6,i7,i8};
output=0;
If [(Abs[i1]>=Abs[1+i2])
    || ((i2)(i2)>=1), output=2];
a=1+i1 z+i2 z z;
b=i3+i4 z+i5 z z;
c=i6+i7 z+i8 z z;
a1=c/.z->1/z;
b1=b/.z->1/z;
c1=a/.z->1/z;
d=Simplify[a c1 - a1 c];
e=Simplify[a b1 - a1 b];
d1 = e/.z->1/z;
e1 = d/.z->1/z;
f=Simplify[d e1 - d1 e];
q1=Simplify[d/.z->Cos[th]+I Sin[th]];
q2=Simplify[f/.z->Cos[th]+I Sin[th]];
q3=Simplify[
      q1/.{Cos[th]->x, Cos[2 th]->2 x x -1}];
q4=Simplify[
      q2/.{Cos[th]->x, Cos[2 th]->2 x x -1,
            Cos[3 th]->4 x x x -3 x,
            Cos[4 th]->8 x x x x - 8 x x +1}];
If[(q3/.x->0)<=0, output=3];
If[(q4/.x->0)<=0, output=4];
w3=x/.N[Solve[q3==0,x]];
w4=x/.N[Solve[q4==0,x]];
For[i=1,i<3,i++,If[(Im[w3[[i]]]==0)
                    &&
                   (Re[w3[[i]]]<=1)
                    &&
                   (Re[w3[[i]]]>=-1),
                   output=5]];
For[i=1,i<5,i++,If[(Im[w4[[i]]]==0)
                    &&
                  (Re[w4[[i]]]<=1)
                    &&
                 (Re[w4[[i]]]>=-1),
                 output=6]];
If [output==0,katalog=Append[katalog,1];,
          katalog=Append[katalog,0];];
katalog>>> "C:\NN\C1.DAT";
Switch[output,
    0,STABLE>>>"C:\NN\A1.DAT",
    2,UNSTABLE1d >>> "C:\NN\A1.DAT",
    3,UNSTABLE2d_0 >>> "C:\NN\A1.DAT",
    4,UNSTABLE2f_0 >>> "C:\NN\A1.DAT",
    5,UNSTABLEdd >>> "C:\NN\A1.DAT",
    6,UNSTABLEff >>> "C:\NN\A1.DAT"]
]
]
]
]
]
]
]
]
```