ISSN : 2278 - 0181

THESIS

Thesis ID : IJERTTH0019

Adaptive Algorithms for Consensus and Synchronization

Federica Torrisi

Electric, Electronics and Computer Engineering Department
University of Catania

Published By

International Journal of Engineering Research and Technology
(www.ijert.org)

UNIVERSITY OF CATANIA

ELECTRIC, ELECTRONICS AND COMPUTER ENGINEERING DEPARTMENT

BACHELOR'S DEGREE IN COMPUTER ENGINEERING

FEDERICA TORRISI

ADAPTIVE ALGORITHMS FOR CONSENSUS AND SYNCHRONIZATION

DEGREE THESIS

Supervisor:

Prof. Mattia Frasca

Ing. Lucia Valentina Gambuzza

Academic Year 2017/2018

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181

Summary

1 Introduction 1

1.1 Structure of the thesis 3

2 Adaptive consensus 5

2.1 What is the "adaptive consensus" 5

2.2 Faults...7

2.3 MATLAB Simulations 9

2.3.1 Simulations with random faults 10

2.3.2 Simulation with deterministic faults...12

2.3.3 Simulation results.................................... 15

3 Adaptive synchronization of chaotic circuit 19

3.1 Rössler system 20

3.2 Chua system 28

4 Conclusions 37

Bibliografy…....…………………………………………………………..……39

Chapter 1
Introduction

A multi-agent system is characterized by a set of agents located in a certain environment and interacting with each other through appropriate organization. A set of nodes, or agents, interacting with each other, creates a more complex structure, called network of dynamic systems, in which each agent can communicate with its neighbours through links between the different nodes. In the past, it had to do with smaller networks, whose structure did not have complicated characteristics; very simple graphs were used to describe these networks and the structure of these graphs could be studied also by visual inspection.
Technological development has led to the appearance of larger and more complex networks, with less intuitive features and characterized by a very high number of nodes.

An important problem in complex networks is synchronization. In colloquial language the term synchronization is used to describe a certain degree of temporal correlation between two different processes. The term "global synchronization" refers to the fact that the synchronization property is applied to all units of the network and therefore the trajectories of all chaotic systems of a network overlap perfectly thanks to the coupling existing between them.

$$\dot{x}_i = f(x_i) + \sigma \sum_{j=1}^{N} a_{ij}(x_j - x_j)$$

This equation represents the synchronization of a network, the coupling of which is expressed by the adjacency matrix A.
$x_i \in \mathbb{R}^n$ is the state vector of the i-th oscillator, while n is the order of the system.

Chapter 1

In the last decades, the design of control systems has shifted from traditional centralized approaches, where all network information is concentrated in a few nodes and decisions are taken from a higher level of control, to distributed approaches, where each agents has and independently calculates local information, but decisions are taken in collaboration with other agents through the exchange of certain information without a superior supervision.

A typical, extensively studied scenario is that of consensus, whose purpose is to create control algorithms so that a group of agents reach an agreement on the final states through local interaction.

Therefore, the term "consensus" means the convergence of several agents to a common parameter through exchange of information between them; the objective is to bring variables of interest to a common value.

The term adaptive control refers to control laws that independently modify their parameters in order to adapt to changes that the system may undergo during the exercise of its functions.

Sometimes it can happen that in the system there are some faults. A fault is defined as a defect within a system that can lead to a failure of the system itself. The expected result in the presence of faults is that, as the percentage of faults increases, the consensus time increases accordingly.

In another context [1] a different result from expectations was found, namely that, in the presence of localized faults, the graph curve describing the consensus time at some point decreases sharply and then grows again.

Introduction

1.1 Thesis structure

The purpose of the project will be to describe some algorithms for consensus and synchronization in a network of dynamic systems, by the results obtained from various simulations using MATLAB.

In Chapter 2 the concept of adaptive consensus will be defined; the various types of faults and the way in which they can be generated will also be analyzed. At the end of the chapter the results obtained by simulations are shown, considering the different types of faults.

Chapter 3 will present the problem of adaptive synchronization of chaotic circuits, in particular the models of the Rössler circuit and of the Chua circuit will be described. The results of the simulations to change of some characteristic parameters of these systems are analyzed at the end of this chapter.

Finally, in Chapter 4 the results obtained from the previous simulations will be compared with the result presented in Article [1], to demonstrate whether it is actually possible to reproduce this behaviour.

Chapter 2

Adaptive consensus

2.1 What is the "adaptive consensus"

The term "consensus" indicates the achievement of an agreement on a certain amount of interest that depends on the status of all agents. The "consensus algorithm" is defined as that interaction law that regulates the exchange of information between an agent and its neighbours in the network.
Let's consider the simplest case in which the network is characterized by n integrator systems connected to each other, each of which is described by the dynamics:

$$\dot{x}_k = u_k$$

where x_k is the state of the system while u_k is the input of the integrators that comes from a communication protocol with other nodes/agents.

In this context, the synchronization equation of a network, defined in the previous chapter, can be written as follows:

$$\dot{x}_i = \sigma \sum_{j=1}^{N} a_{ij}\left(x_j - x_j\right)$$

Since in a network of integrators the following relationship applies:

$$f(x_i) = 0$$

Chapter 2

The consensus is defined "adaptive" because we take into account the terms a_{ij} that evolve over time; so it is possible to write them as a function of time: $a_{ij}(t)$. These terms adapt autonomously to the changes that the system may undergo, by means of a law defined as adaptive control law, or law of adaptation. When the following condition occurs, $a_{ij}(0) = 0$, there is no consensus.

If, instead, there is consensus, it is necessary to study the evolution of a_{ij}, through the following law:

$$\dot{a}_{ij} = (x_j - x_i)^2$$

For simplicity we put σ =1 and rewrite the previous formula in the following way:

$$\dot{x}_i = \sum_{j=1}^{N} a_{ij}(x_j - x_i) \qquad \text{con } i = 1, \dots, N$$

So, we found the law of adaptation, also defined as consensus protocol. Given a simple graph G with n vertices, its Laplacian matrix $L \coloneqq (l_{ij})_{nxn}$ is defined as:

$$L = D - A$$

where D is the degree matrix, a diagonal matrix containing information on the degree of each vertex of the graph, and A is the graph's adjacent matrix, a matrix of size $n\ x\ n$ where each row and each column correspond to the nodes and the elements a_{ij} of the matrix assume the value 0 if there is no arc (i, j) corresponding, otherwise 1.

Adaptive consensus

It is possible to write the previous relationship using the l_{ij} elements of the Laplacian matrix associated with graph G.

$$\dot{x}_i = -\sum_{j=1}^{N} l_{ij} x_j$$

where $x = col(x_1, x_{2,\dots,} x_n)$ e $l_{ij} = -a_{ij}$ con $i \neq j$

Ultimately, if we apply the consensus protocol to each system, we can write in vector form that

$$\dot{x} = -Lx$$

For the resolution of this first-order linear equation the Euler algorithm is used. Euler's method is the simplest numerical integration method for differential equations. It is used this method to integrate equations of coupled dynamic systems networks.

$$x_{k+1} = x_k - Lx_k dt$$

With dt the integration step.

2.2 Fault

A fault is defined as a defect within a system, which can result in a breakdown or failure of the system itself.

Sometimes it happens that in the matrix A there are some faults, that is points in which the value assumed by the a_{ij} elements is equal to 0; this means that there is no arc (i,j) corresponding and therefore the two nodes are disconnected.

$$A = \begin{bmatrix} * & * & \cdots \\ \vdots & \ddots & \vdots \\ * & \cdots & * \end{bmatrix} \qquad \text{with } * = 0$$

Chapter 2

The presence of these zeros in matrix A means that some a_{ij} cannot evolve.

We can distinguish two types of fault:

1. Random faults: these are faults present in random positions of the matrix
2. Deterministic faults: these are localized faults present in certain locations. These faults are divided into two categories:

   2.1. Deterministic faults – columns: whole columns of the matrix are zero

   2.2 Deterministic faults – area: defined area portions of the matrix are zero

$$\begin{bmatrix} \dots & 0 & \dots & \dots & 0 \\ 0 & \dots & \dots & \dots & \dots \\ \dots & \dots & 0 & \dots & \dots \\ \dots & \dots & \dots & 0 & \dots \\ 0 & \dots & \dots & \dots & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 0 & \dots & 0 & \dots \\ 0 & 0 & \dots & 0 & \dots \\ 0 & 0 & \dots & 0 & \dots \\ 0 & 0 & \dots & 0 & \dots \\ 0 & 0 & \dots & 0 & \dots \end{bmatrix} \qquad \begin{bmatrix} 0 & 0 & 0 & \dots & \dots \\ 0 & 0 & 0 & \dots & \dots \\ 0 & 0 & 0 & \dots & \dots \\ 0 & 0 & 0 & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \end{bmatrix}$$

1. *Random Faults* *2.1. Deterministic Faults - columns* *2.2. Deterministic Faults - rect*

The consensus time t_c is the time needed for the variables to converge. The result we expect when faults are present is that, as the percentage of faults increases, the consensus time increases accordingly. In another context [1] a counterintuitive result was found: in the presence of localized faults the graph curve describing the consensus time at a certain point decreases and then grows again. Through various simulations that will be described later, we want to check if this behaviour, different from expectations, is generalizable.

Adaptive consensus

2.2 MATLAB Simulation

The general scheme of MATLAB codes used for the simulations is always the same and is characterized by the following steps:

1. Definition of the integration parameters
2. Definition of the number of initial conditions, of the faults number and initialization of the three matrices for the storage of the obtained results. The dimensions of these three matrices are the same and are given by the number of initial conditions and by the fault number.
   i) IEAv: IAE means Integral Absolute Error. To obtain the IAE value the following relationship is used, in which the absolute error is time integrated
   $$IAE = \int_0^{+\infty} |e|\, dt$$
   ii) Ts: ts is the settling time, that is the time taken by the response to enter permanently in a range close to the regime value.
   iii) Faultsr: it contains the number of nodes set to zero.
3. Start of the for loop on the initial conditions and for loop on the faults.
4. Initialization of the conditions for the dynamic consensus: creation of a matric X0 characterized by random values, definition of an adjacency matric A, of the Laplacian matric L, of the matrix X, of the matric Az characterized only by values equal to 1, which is necessary to zero some links.
5. Cancellation of some links of the matrix Az, depending on the different type of faults that are applied to the system.
6. Integration of discretized equations using Euler methos: matrix A and consensus variables are updated.
7. Calculation of the error
8. Calculation of the consensus error
9. End of the for loop on the faults and for loop on initial conditions
10. Graphic display of results

Chapter 2

2.2.1 Simulation with random faults

In this first example of code, faults are generated randomly.

```
% Simulation with random faults

% integration parameters
dt = 0.001;
t = 0 : dt : 100;
N = 100;

numeroic = 20; % number of initial conditions
numerof = 13; % number of faults
% initialization of matrices for storage
IAEv = zeros(numeroic,numerof);
ts = zeros(numeroic,numerof);
faultsr = zeros(numeroic,numerof);

for ic = 1 : numeroic
    % random faults
    possloc = [];
    for ii = 1 : N,
      for jj = ii+1 : N,
        possloc = [possloc; ii jj];
      end
    end
    maxp = randperm(N*(N-1)/2);
    faultloc = possloc(maxp,:);
    fp = [200:200:4000]; % fp fraction of faults, fp <=maxp
    X0 = rand(N,1) - 0.5;
    % network
    for index_fault = 1 : numerof
        A = zeros(N);
        L = diag(sum(A,2)) - A; % Laplacian matrix
        Az = ones(N); % matrix to set the links to zero
        X = zeros(N, length(t)); % init. cond. for consensus
        X(:,1) = X0;
```

Adaptive consensus

```
            % generation of random faults
            for ip = 1 : fp(index_fault)
                Az(faultoc(ip,1), faultloc(ip,2)) = 0;
             end
            % integration of discretized equations
            for it = 2 : length(t)
                x = X(:, it-1);
                A = A + dt*Az.*(x*ones(1,N)-ones(N,1)*x').^2;
                L = diag(sum(A,2)) - A;
                X(:,it) = X(:,it-1) - dt*L*X(:,it-1);
            end
            % calculation of the error
            Xref = mean(X0)*ones(N,1);
            e = mean(abs(X-Xref*ones(1,length(t))),1);
            IAE = sum(e)*dt;
            IAEv(ic,index_fault) = IAE;
            %calculation of ts
            S = stepinfo([200; e'],[0;
            t'],0,'SettlingTimeThreshold',0.0002);
            ts(ic,index_fault) = S.SettlingTime;

            zeroed_nodes = sum(sum(Az==0))-sum(sum(diag(Az)==0));
            faultsr(ic,index_fault) = zeroed_nodes;

        end % for cycle on faults
    end % for cycle on initial conditions

    figure,plot(mean(faultsr,1),mean(ts,1))
    figure,plot(mean(faultsr,1),mean(IAEv,1))
```

Chapter 2

2.2.1 Simulation with deterministic faults

In this paragraph the behaviour of a system with deterministic faults will be analysed. Unlike random faults, in this case faults are located in very precise locations. It is possible to set to zero entire columns or portions of defined areas. The code for the simulation is the same as the previous one, the only difference is the way the faults are generated. Since we are considering deterministic faults, first we remove from the code the part in which causal faults are created and replace it with the part relating to the generation of deterministic faults. In case we want to set to zero the columns, before the start of the for loop on the initial conditions, we define an appropriate vector fp (fp stands for "fraction of faults") and a vector v that indicates the columns to be permutated.

```
% deterministic faults - columns
fp = [1:13];
v = randperm(N);
```

The vector v is randomly defined, so the columns are reset sequentially but randomly.
At this point, instead of the for loop for undoing links in the case of random faults, it is defined another cycle in which the entire columns are set to zero.

```
% deterministic faults - columns
for ii = 1 : fp(index_fault)
  Az(:; v(ii)) = 0;
end
```

Adaptive consensus

Suppose we get a vector v of this type:

```
v = [3 6 7 8 5 1 2 4 9 10];
```

Suppose we also have a faults number equal to 1 (index_fault=1) and a matrix Az of 5x5 dimensions.

Then, as reported in the following code, the column 3 of the matrix Az will be cancelled.

```
v(ii) = v(1) = 3
Az(:, 3)=0; % column 3 set to zero
```

$$Az = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \end{bmatrix}$$

Obviously, having a larger number of faults, more columns will be reset, but always in a determined way, as they are cancelled in full.

The code of a system in which we reset an area of the matrix Az rather than entire columns is very similar to the previous code. The difference is that we do not need a v vector but only a fp vector. In this specific case, the aim is to cancel a portion of the area that has the shape of a triangle.

Suppose we have a fp vector defined as follows:

```
% deterministic faults - rect
fp = [3 3; 8 8; 15 15; 22 22; 27 27];
```

Chapter 2

This vector specifies the positions of the matrix through which to construct the triangle of values to cancel.

Defined fp, the following for loop is used to cancel the desired area.

```
%deterministic faults - rect
a = 0;
for ii = 1 : fp(index_fault,1)
    a = a + 1;
    for jj = 1 : fp(index_fault,2)
      Az(ii,jj) = 0;
    end
end
```

Suppose, as in the previous case, to have a number of faults equal to 1 and a matrix Az of 5x5 dimensions.

Then, as shown in the following code, a portion of area is placed equal to zero, in particular the area of the triangle having as its base the number of rows, that is 3, and as its height the number of columns, that is 3; so the matrix Az is obtained.

```
fp(1,1)=3;
fp(1,2)=3;
```

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

2.2.3 Simulation results

Once describing the code for the consensus of dynamic systems in the presence of faults and how to generate such faults, the results obtained are analysed. In the simulations we did not generate the faults symmetrically, because we are interested in focusing our attention on the behaviour mentioned in article [1], in which the elements of matrix A are not eliminated symmetrically.

First, we analyze the behaviour of the error and the consensus time as the number of faults changes, based on the three types of faults previously described.

From Figure 2.1 it is clear that the error grows as the number of faults increases, regardless of how the faults are generated. However, this increase is most significant only in the case of deterministic faults where the columns are cancelled, since in the other two cases it is almost imperceptible.

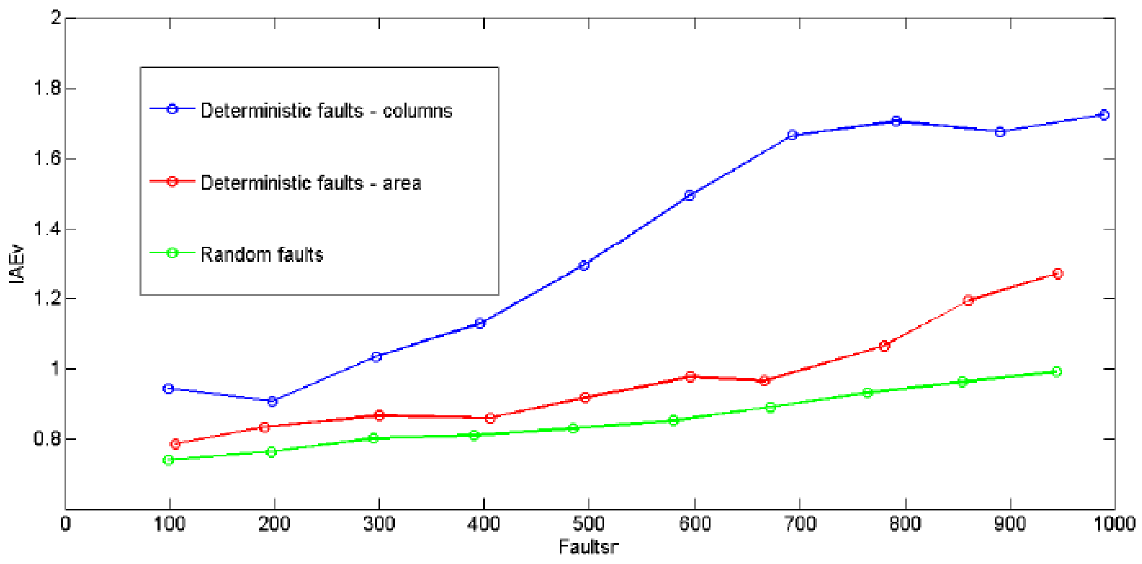

Fig. 2.1 Evolution of the error as a function of the faults number

Chapter 2

In the rest of the thesis, charts like the previous one will not be shown, because the obtained results are much similar to those shown in Figure 2.1. In general, we can say that the curve that describes the trend of the error increases as the number of faults increases, so it is defined as an increasing function.

As regards the consensus time, from Figure 2.2 it is noticeable that this increases as the number of faults increases.

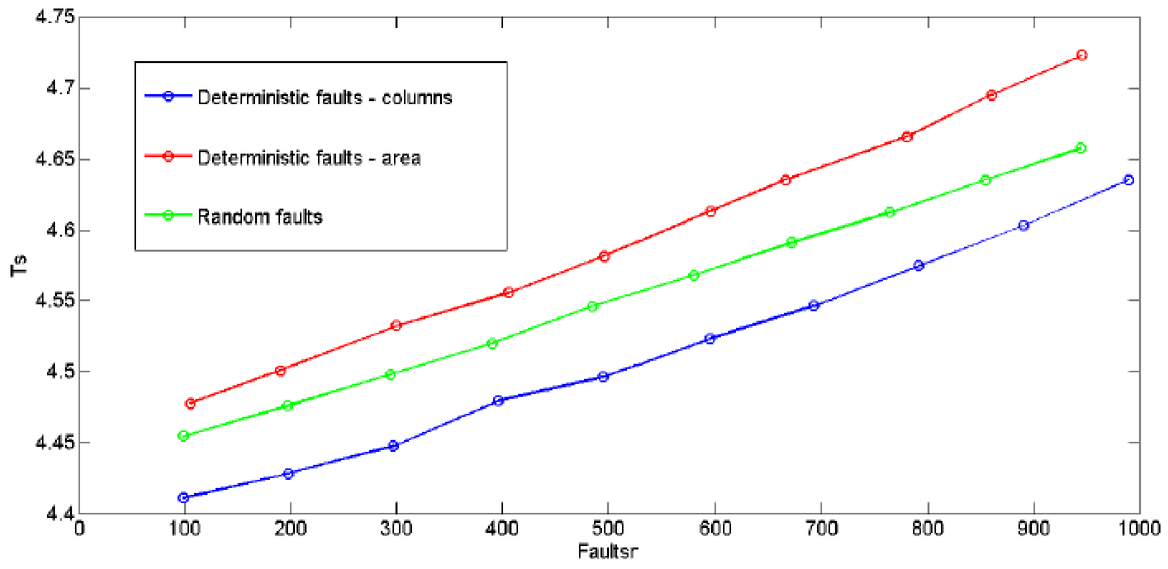

Fig. 2.2 Evolution of the consensus time as a function of the faults number

The results observed in Figure 2.2 are the typical results that we expected to obtain, that is that as the fault percentage increases, the consensus time increases accordingly. So, we are far from the paradoxical result described in article [1], as the curve describing the trend of consensus time does not decrease at any point, rather it is always increasing.

Adaptive consensus

In the codes described above, a K scale factor of 0.01 is used, present in the update formula of the consensus variables. Various simulations were made for the different types of faults, in which the value of K was changed and, from the results obtained, we noticed that the higher the value of K, the shorter the time interval within which the curve of consensus time evolves as the number of Faults varies. So, in order to be able to study more precisely the behaviour of a system, it is preferable to use low K values, in order to slow down the process of updating values. For this reason, K values of 0.01 or 0.001 will be used in the following simulations.

In the next chapter two chaotic systems will be studied; the code scheme of these systems is the same as that described in this chapter. In the following, when simulations are made with deterministic faults, we will consider only the case in which the columns are zeroed, because the obtained results in this case are much similar to those that we obtain when portions of area is set to zero, so let’s not report the latter results.

Chapter 3

Adaptive synchronization of chaotic circuit

A chaotic system is a dynamic system with a non-linear evolution, that depends strictly on the initial conditions; in fact, a variation in the initial conditions may involve considerable variations on the output results. In this chapter, two chaotic systems will be described, the Rössler system and the Chua system, and the problem of adaptive synchronization of a network of N circuits will be analysed.
By making various simulations, it is evident that there is a certain dependence on the gain, so certain behaviours of the system change according to the size of the matrix. For this reason, in the following simulations, N is set at 10.
In addition, in this chapter we will focus our study on systems in which deterministic faults are present, because in the article [1] the abnormal behaviour of the synchronization time curve is manifested in the presence of deterministic faults, so the results obtained with random faults are neglected.

Chapter 3

3.1 Rössler system

The Rössler system is a very simple third-order system and can be classified as chaotic systems. The equations that characterize it are three first order coupled differential equations:

$$\dot{x}_i = -y_i - z_i$$
$$\dot{y} = x_i + ay_i$$
$$z_i = b + z_i(x_i - c)$$

The system has three degrees of freedom, represented by the dynamic variables x, y and z; it also has only one nonlinear term, the product between z and x in the third equation.

The system parameters are a, b e c, fixed in this way:

$$a = b = 0{,}2 \qquad\qquad c = 9$$

so that the oscillator has a chaotic dynamic.

Let us consider N=10 identical Rössler oscillators coupled by the second variable of the node j, which acts on the dynamics of the second variable of the node i. Then we rewrite the above equations as follows:

$$\dot{\mathrm{x}}_\mathrm{i} = -\mathrm{y}_\mathrm{i} - \mathrm{z}_\mathrm{i}$$
$$\dot{\mathrm{y}}_\mathrm{i} = \mathrm{x}_\mathrm{i} + \mathrm{ay}_\mathrm{i} - \sigma \sum_{j=1}^{N} g_{ij} y_j$$
$$\mathrm{z}_\mathrm{i} = \mathrm{b} + \mathrm{z}_\mathrm{i}(\mathrm{x}_\mathrm{i} - \mathrm{c})$$

This formula is similar to the synchronization equation described in Chapter 1.

Adaptive synchronization of chaotic circuit

Below there is the code for the simulation of a network of Rössler systems. The scheme of the code follows the one described in the previous chapter, with the only difference that, after the integration parameters, the parameters of Rössler are defined and that the matrices for the definition of the initial conditions (xold, yold and zold) are always randomly defined but in a different way. The characteristic equations of the Rössler system are then reported within the loop for the integration of discrete equations.

Code for the simulation of a network of Rössler systems

```
%Simulation of a network of Rossler systems

% integration parameters
dt = 0.001;
t = 0 : dt : 1000;
N = 10;
% Rossler parameters
sigma = 1;
ar = 0.2;
br = 0.2;
cr = 9;

numeroic = 20; % number of initial conditions
numerof = 13; % number of faults
% initialization of matrices for storage
IAEv = zeros(numeroic,numerof);
ts = zeros(numeroic,numerof);
faultsr = zeros(numeroic,numerof);

%deterministic faults - columns
fp = [1:10];

for ic = 1 : numeroic
  for index_fault = 1 : numerof
    % initialization
```

Chapter 3

```
    xold = (30*rand(N,1)-15)/5;
    yold = (30*rand(N,1)-15)/5;
    zold = (30*rand(N,1)-15)/5;
    A = zeros(N);
    Az = ones(N);
    Gmatrix = diag(sum(A,2)) - A;
    x = zeros(N, length(t));
    y = zeros(N, length(t));
    z = zeros(N, length(t));
    x(:,1) = xold;
    y(:,1) = yold;
    z(:,1) = zold;

    % deterministic faults - columns
    for ii = 1 : fp(index_fault)
      Az(:,ii) = 0;
    end

    % integration of discretized equations
    for it = 2:length(t)
      A=A+dt*0.001*Az.*((xold*ones(1,N)-ones(N,1)*xold').^2);
      Gmatrix = diag(sum(A,2))-A;
      coupling = -Gmatrix*yold;
      dxdt = -yold-zold;
      dydt = xold+ar*yold+sigma*coupling;
      dzdt = br+zold.*(xold-cr);
      xold = xnew;
      yold = ynew;
      zold = znew;
      x(:,it) = xnew;
      y(:,it) = ynew;
      z(:,it) = znew;
    end
    % error calculation
    error = 0;
    for i = 1 : N
      for j = 1 : N
        Ee = (x(i,:)-x(j,:)).^2+(y(i,:)-y(j,:)).^2+(z(i,:)-
```

Adaptive synchronization of chaotic circuit

```
          z(j,:)).^2;
          error = error + Ee;
        end
      end
      error = sqrt(error/N/(N-1));
      IAE = sum(error)*dt;
      IAEv(ic, index_fault) = IAE;

      % consensus time calculation
      S = stepinfo([200;e'],[0;t'],0,'SettlingTimeThreshold',
      0.0002);
      ts(ic,index_fault) = S.SettlingTime;
      zeroed_nodes = sum(sum(Az==0))-sum(sum(diag(Az)==0));
     faultsr(ic, index_fault) = zeroed_nodes;
    end % for loop on faults
  end % for loop on initial condition

  figure,plot(mean(faultsr,1),mean(ts,1))
  figure,plot(mean(faultsr,1),mean(IAEv,1))
```

This code describes the behaviour of a Rössler system when half of the adjacent matrix columns are zeroed. The columns are cancelled sequentially, so in the case of the previous code the first 5 columns will be reset; however it makes no difference if we cancel the columns sequentially or randomly, because we care about the number of nodes reset, which in both cases is the same.

Chapter 3

Figure 3.1 shows how Rössler systems synchronize.

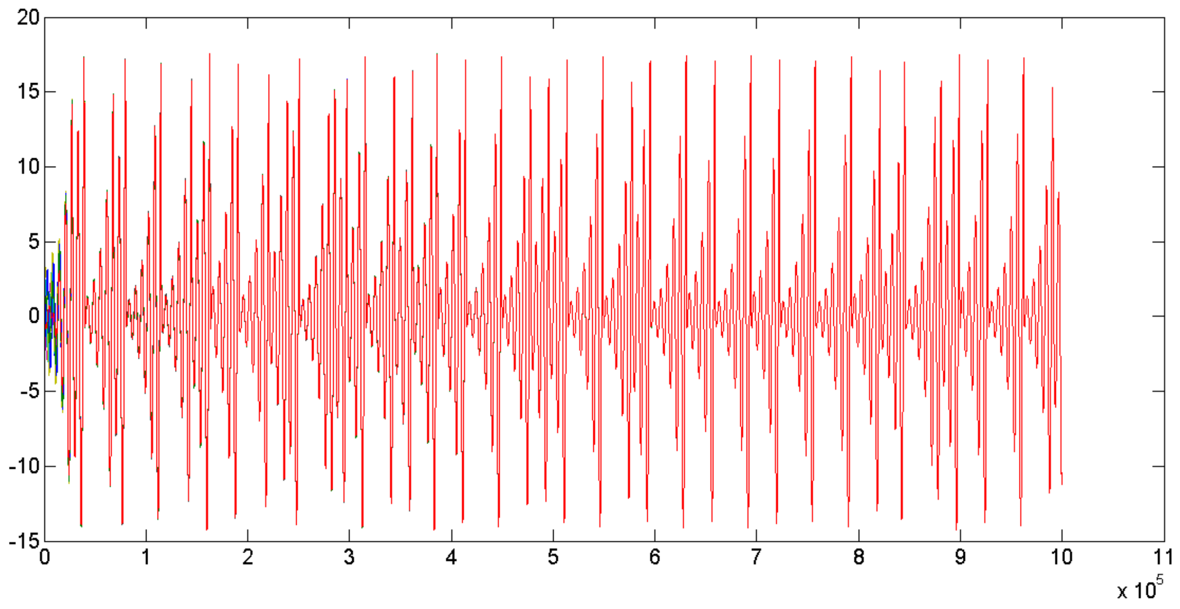

Fig. 3.1 Synchronization of a network of Rössler system

In Figure 3.2 the result of the first simulation made is shown

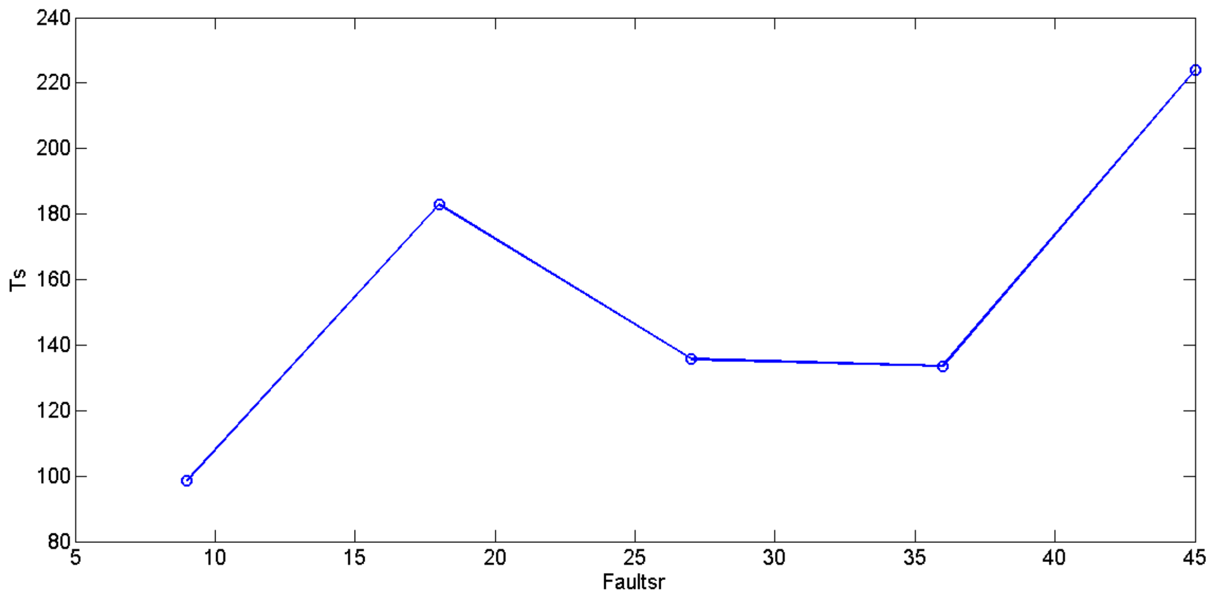

Fig. 3.2 Evolution of the consensus time according to the number of faults in a network of Rössler systems

Adaptive synchronization of chaotic circuit

Having considered a number of faults equal to 5, in the previous graph 5 markers can be distinguished; the first corresponds to the first column cancelled (then when 9 links are reset), the second to the second column cancelled (then when 18 links are reset), etc… Compared to the graphs obtained in the previous chapter, in this case the curve for a certain stretch decreases and then grows again, but we still do not get close to the paradoxical result of the article [1].

In the code we placed σ=1, but before fixing this value several simulations were made, from which it emerged that the synchronization depends on the value of σ. For example, for σ=0.75 the network never reaches synchronization, while, if this coupling coefficient increases, the network synchronizes immediately without a true adaptive evolution. In addition, for sigma 1 the system is quite stable and can synchronize, so in the subsequent simulations we will always use this sigma value. Synchronization does not only depend on σ, but also on other factors such as type and coupling mode. In order to study the behaviour of this system, we make appropriate modifications and record the results obtained.

First, we study the behaviour of the system as the definition of error changes.

In the above examples the error was calculated as follows:

```
% error calculation - Version1
error = 0;
for i = 1 : N
  for j = 1 : N
    Ee = (x(i,:)-x(j,:)).^2+(y(i,:)-y(j,:)).^2+(z(i,:)-z(j,:)).^2;
    error = error + Ee;
  end
end
```

Chapter 3

Now we do a simulation with a different approach to calculate the error. Rather than doing the two for cycles as in the previous example, we add the following relationship within the loop for the integration of discretized equations:

```
% error calculation - Version2
error(it) = mean((xnew-mean(xnew)).^2+(ynew-mean(ynew)).^2+(znew-
mean(znew)).^2
```

For clarity, the first method for error calculation is called Version1 (V1), while the second Version2 (V2).

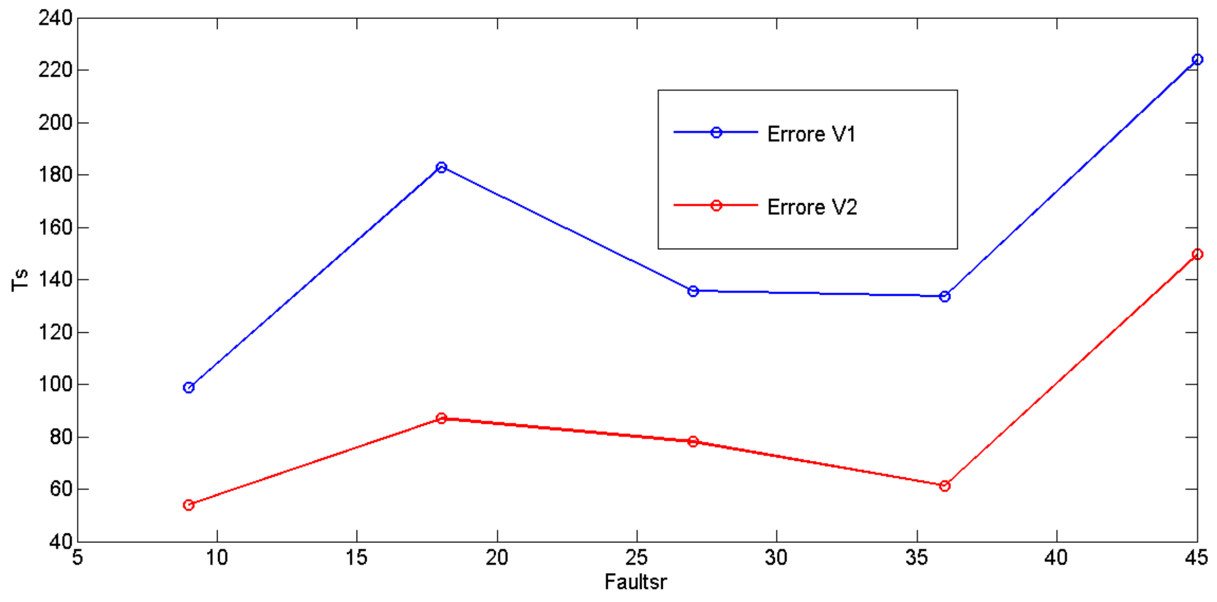

Fig. 3.3 Comparison of the trend of the consensus time as the number of faults changes when two different approaches to calculate the error are used in a network of Rössler systems

Adaptive synchronization of chaotic circuit

From Figure 3.3 we note that the main difference consists in the fact that, when we use the second approach for the calculation of the error, the variables reach the consensus before; for example when 9 nodes are cancelled, that is when the first column is reset, if it is used the Version1 the consensus is reached after about 98 ms, instead if it is used the Version2 only after about 57 ms.

The next change we're going to make is to update the weights and the state not at the beginning of the cycle for the integration of discretized equations, but at the end of the same cycle, in order to modify the matrix A with the updated values of xold, yold and zold. This simulation has been made both in the case in which the error is calculated using both Version1 and Version2.

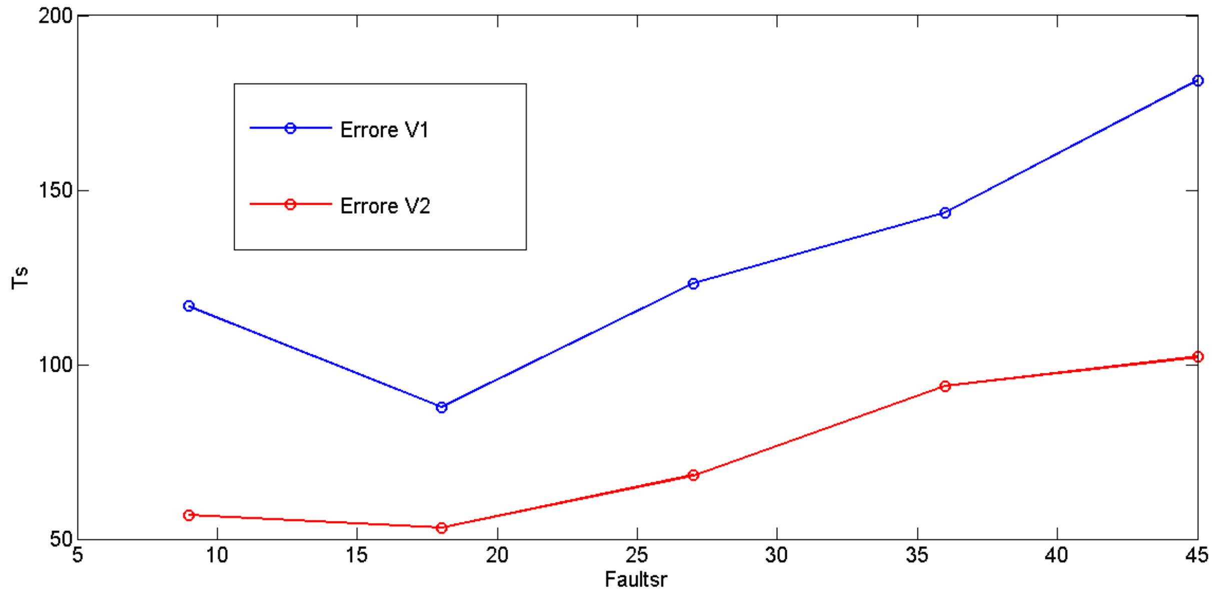

Fig. 3.4 Comparison of the trend of the consensus time as the number of faults changes when two different approaches to calculate the error are used and the position of the matrix A is modified in a network of systems of Rössler

From Figure 3.4 it is noticeable that the temporal jump between the annulment of the first column and the second one is minimal, almost negligible in the case we use the Version2 for the calculation of the error. We realize that, to varying of the position of the update of A, the behaviour of the diagram diverges a little from that in Figure 3.3, therefore we deduce that the system is much sensitive to the variation of determined parameters.

From the obtained results it is possible to assert that the two versions of the error calculation do not differ unless normalizations not carried out, therefore the difference in the results depends only in this non-normalization and in the initial conditions. Furthermore, we see that the curve is not close to that obtained in the above-mentioned article [1], as we would expect the consensus time to decrease sharply and then increase again when the number of faults increase, but as you can see from the previous graphs this behaviour is not assumed by these studied networks.

3.2 Chua system

The Chua circuit is a very simple electronic circuit that reproduces the phenomenon of chaos in non-linear dynamic systems. It was the first circuit to be built in order to reproduce chaotic behaviour. Its peculiarity is to be an autonomous circuit, that is, it does not need input signals.

The dimensional equations that describe the dynamics of this circuit are the following:

$$\dot{x}_1 = \alpha_c\big(x_2 - x_1 + g(x_1)\big)$$
$$\dot{x}_2 = x_1 - x_2 + x_3$$
$$\dot{x}_3 = -\beta_C x_2 - \gamma_C x_3$$

Adaptive synchronization of chaotic circuit

where $g(x_1)$ is a linear function defined in sections:

$$g(x_1) = -bx_1 + 0{,}5(b-a)(|x_1+1| - |x_1-1|)$$

with $a = -1{,}143$ and $b = -0.714$.

The other Chua parameters are chosen so that the system is chaotic:

$\alpha_c = 9, \beta_c = 14.286\ e\ \gamma_c = 0.$

Also in this case, the MATAB code is the same as that used for Rössler systems.

Code for the simulation of a network of Chua systems

```
% Simulation of a network of Chua systems

% integration parameters
dt = 0.001;
t = 0 : dt : 1000;
N = 10;
% Chua parameters
sigma = 1;
ac = -1.143;
bc = -0.714;
gamma = 0;
alpha = 9;
beta = 14.286;

numeroic = 20; % number of initial conditions
numerof = 13; % number of faults
% initialization of matrices for storage
IAEv = zeros(numeroic,numerof);
ts = zeros(numeroic,numerof);
faultsr = zeros(numeroic,numerof);

% deterministic faults - columns
fp = [1:10];

for ic = 1:numeroic
  for index_fault = 1:numerof
```

Chapter 3

```
% initialization
xold = -0.3 + 0.6*rand(N,1);
yold = -0.3 + 0.6*rand(N,1);
zold = -0.3 + 0.6*rand(N,1);
A = zeros(N);
Az = ones(N);
Gmatrix = diag(sum(A,2))-A;
x = zeros(N, length(t));
y = zeros(N, length(t));
z = zeros(N, length(t));
x(:,1) = xold;
y(:,1) = yold;
z(:,1) = zold;

% deterministic faults - columns
for ii = 1 : fp(index_fault)
  Az(:,ii) = 0;
end

% integration of discretized equations
for it = 2 : length(t)
  A=A+dt*0.001*Az.*((xold*ones(1,N)-ones(N,1)*xold').^2);
  Gmatrix = diag(sum(A,2))-A;
  couplingx = -Gmatrix*xold;
  couplingy = 0;
  couplingz = 0;
  fc = -bc*xold+(-ac+bc)/2*(abs(xold+1)-abs(xold-1));
  dxdt = alpha*(yold-xold+fc)-sigma*couplingx;
  dydt = xold-yold+zold-sigma*couplingy;
  dzdt = -beta*yold-gamma*zold-sigma*couplingz;
  xnew = xold+dt*dxdt;
 ynew = yold+dt*dydt;
 znew = zold+st*dzdt;
 xold = xnew;
  yold = ynew;
  zold = znew;
  x(:,it) = xnew;
  y(:,it) = ynew;
```

Adaptive synchronization of chaotic circuit

```
      z(:,it) = znew;
    end

    % error calculation
    error = 0;
    for i = 1 : N
      for j = 1 : N
        Ee = (x(i,:)-x(j,:)).^2+(y(i,:)-y(j,:)).^2+(z(i,:)-
        z(j,:)).^2;
        error = error + Ee;
      end
    end
    error = sqrt(error/N/(N-1));
    IAE = sum(error)*dt;
    IAEv(ic, indicefault) = IAE;

    % consensus time calculation
    S = stepinfo([200;e'],[0;t'],0,'SettlingTimeThreshold',
    0.0002);
    ts(ic,indicefault) = S.SettlingTime;
    zeroed_nodes = sum(sum(Az==0))-sum(sum(diag(Az)==0));
    faultsr(ic, indicefault) = zeroed_nodes;
  end % for loop on faults
end % for loop on initial conditions

figure,plot(mean(faultsr,1),mean(ts,1))
figure,plot(mean(faultsr,1),mean(IAEv,1))
```

In Figure 3.5 is shown the result obtained from the first simulation made

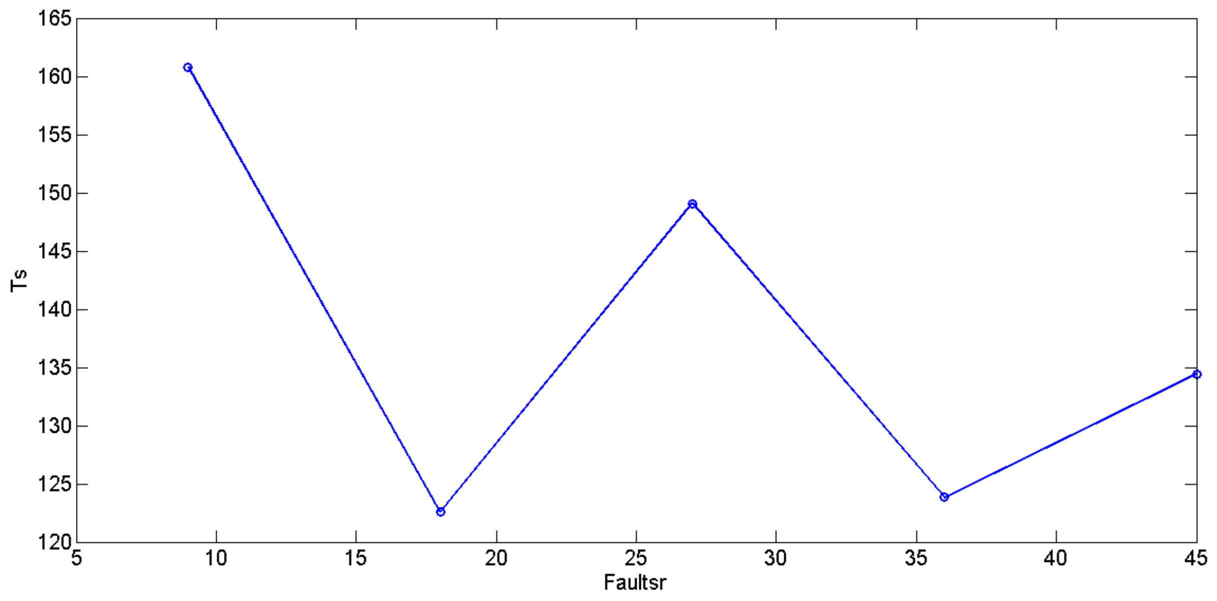

Fig. 3.5 Evolution of the consensus time according to the number of faults in a network of Chua systems

From this first result it is possible to notice that the trend is different from that of a Rössler system. We could expect that by making appropriate changes it is possible to approach the paradoxical behaviour observed in the article [1].

First we fix the sigma value to 1, because by means of various simulations similar to those we have done for a Rössler system, we realize that for this value the system is able to synchronize.

Adaptive synchronization of chaotic circuit

At this point, as in the previous case, we study the behaviour of the system as the definition of error changes, always considering the two versions described in the previous paragraph, that is the Version1 (V1) and the Version2 (V2).

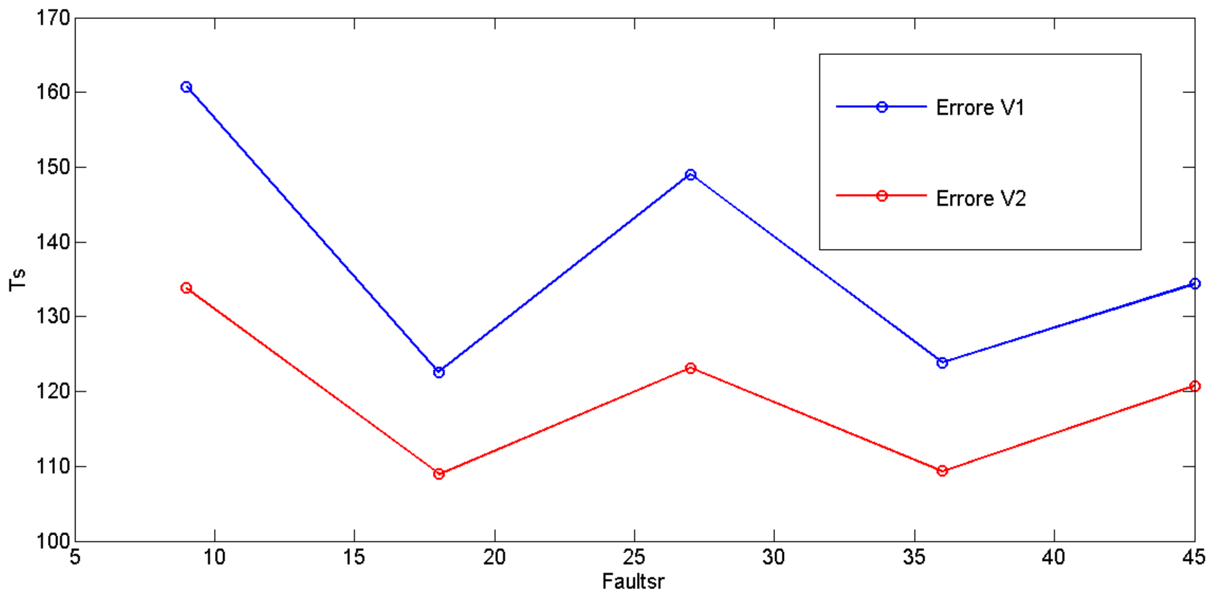

Fig. 3.6 Comparison of the trend of the consensus time as the number of faults changes when two different approaches to calculate the error are used in a network of Chua systems

From Figure 3.6 we observe that the trend of the two graphs is similar, the variables of the system in which the Version1 is used for the calculation of the error reach the consensus shortly after the variables of the other system. In the decreasing section of the curve, between the first two markers (the first corresponds to the cancellation of the first column and the second to the cancellation of the second column), the graph in which the Version1 is used has a greater slope than the other. Despite this, it isn't possible to deduce anything for sure, so we continue to do other simulations by varying other parameters of the system.

Chapter 3

Let's now analyze the behaviour of our system when the position of the matrix A is changed; as in the case of Rössler systems, let's update A at the end of the for loop for the integration of discretized equations.

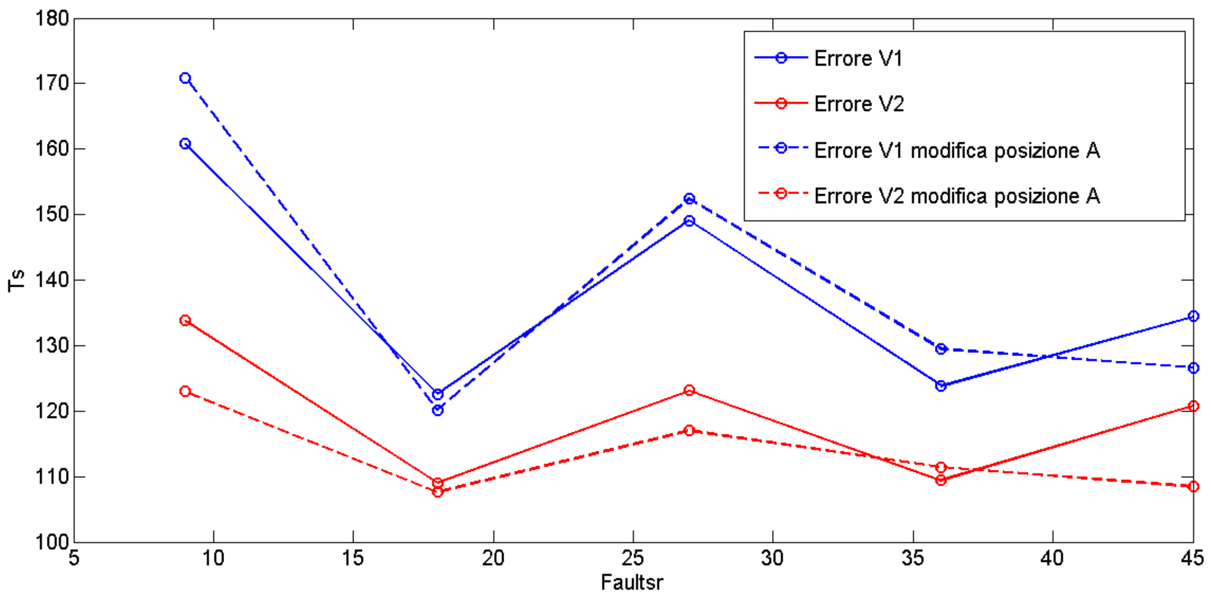

Fig. 3.7 Comparison of the trend of the consensus time as the number faults changes when two different approaches to calculate the error are used and the position of the matrix A is modified in a network of systems of Chua

In Figure 3.7 the results obtained by this simulation are compared with those of the previous simulations; we note that the results obtained by changing the position of A are very similar to those shown in Figure 3.6. For this reason, we can decide to keep the update of A at the beginning of the cycle for the integration of differential equations.

Adaptive synchronization of chaotic circuit

Among the simulations just made, we realize that the one that could be closer to the counterintuitive result that we have mentioned is the one in which we use the Version1 for the calculation of the error.

A further change that it is possible to make is to consider only the variable x for the calculation of the error, then we will use the following relationship:

```
% error calculation
error = 0;
for i = 1 : N
  for j = 1 : N
    Ee = (x(i,:)-x(j,:)).^2;
    error = error + Ee;
  end
end
```

The result of the simulation is shown in Figure 3.8

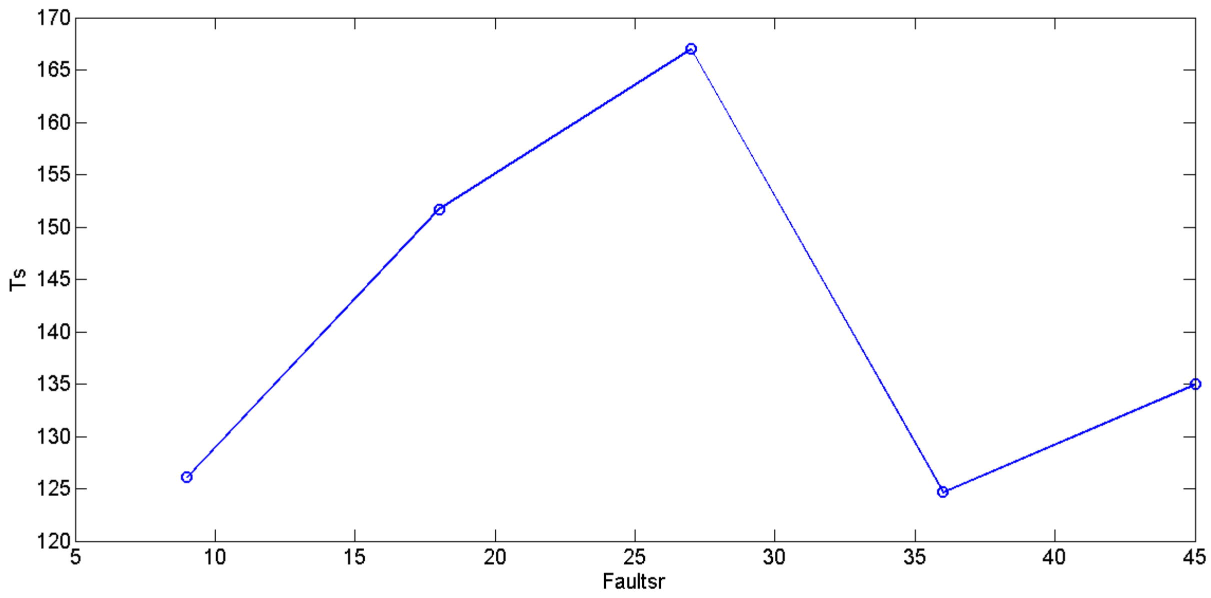

Fig. 3.8 Evolution of the consensus time according to the number of Faults in a network of Chua systems, considering only the variable x for the calculation of the error

Analysing the graph shown in Figure 3.8, it can be deduced that it is close to the result of the article [1], just because the decreasing stroke is sharper than the previous graphs, in fact the third column is zeroed after about 167,5 ms while the fourth one after about 124 ms.

We can conclude by saying that we have not been able to fully recover the paradoxical result of Article [1], not even by making various modifications to the chaotic circuits just described. However, only in one case it is possible to observe a different behaviour that is close to that of the above mentioned article, and this is the case of a network of Chua systems in which the error is calculated using Version1 and considering only the variable x.

Chapter 4

Conclusion

In this thesis, some adaptive algorithms for consensus and synchronization have been studied. After a brief description of complex networks and multi-agent systems, the concept of "adaptive consensus" was presented. Subsequently, the problem of the faults has been presented, which are system anomalies that may result in system failures. Each system behaves differently depending on how the faults are generated, in particular there may be random or deterministic faults. In the article [1] it was reported that, in the presence of deterministic faults, the system behaves differently from expectations, that is, the graph that describes the evolution of consensus time as the number of faults varies decreases sharply and then grows again. In this paper several simulations were made to understand if the result just described was general or not. We have not found conclusive evidence of the possibility of such a generalization.

In Chapter 2, simulations of networks of dynamic systems were made to vary the three types of faults and, in all three cases, the result was the same, that is, the consensus time increases as the number of faults increases.

In Chapter 3 two chaotic systems have been described, the Rössler system and the Chua system .From the various simulations it is possible to realize that in these circuits there is a very strong dependence on the order in which some parts of the code are written and on the definition of error. In fact, when certain parameters vary, the trend of the curve describing the evolution of the consensus time changes considerably.

Chapter 4

Only in a situation we have found a result that is close to the behaviour described in article [1] and it is precisely the case of a Chua system in which the error is calculated by Version1 and considering only the variable x: in this case there is a stretch where the curve decreases, although not abruptly as in the case of article [1].

In conclusion, from the various simulations we have been able to establish that the behaviour of the circuits, especially that of the chaotic circuits, is very sensitive to the variation of some parameters and such changes make it unstable. This is the reason why it was not possible to determine a law that accurately describes the behaviour of such systems in the presence of faults.

Bibliography

[1] L. V. Gambuzza, M. Frasca, L. Fortuna, V. Ntinas, I. Vourkas, G. Syrakoulis, *"Memristor Crossbar for Adaptive Synchronization*", IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 64, no. 8, 2017, pp. 2124-2133.

[2] M. Frasca, L. V. Gambuzza, A. Buscarino, L. Fortuna, *"Synchronization in Networks of Nonlinear Circuits"*

[3] MATLAB Documentation Center, *www.mathworks.it*

[4] WIKIPEDIA, *www.wikipedia.org*