Universidade de São Paulo
Escola de Engenharia de São Carlos
Departamento de Engenharia Elétrica

Alunos: Luisa Helena Bartocci Liboni
Rodrigo de Toledo Caropreso

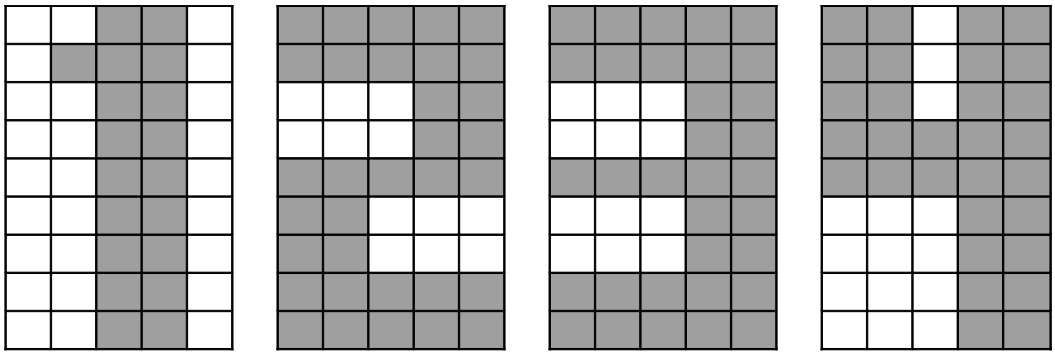
Data de Entrega: 28/05/2012

Redes Neurais Artificiais

(Prof. Ivan Nunes da Silva)

EPC-8

Um sistema de transmissão de imagens, representadas em 45 bits, codifica e envia os referidos sinais por meio de um link de comunicação. Ao chegar ao sistema de recepção, os sinais são novamente decodificados visando à recuperação fiel da imagem previamente enviada. As quatro imagens (informações) que estão sendo transmitidas são representadas pelas seguintes figuras:



Entretanto, durante a transmissão ao longo do link de comunicação, as informações são corrompidas por ruídos que deixam as imagens incompletas ou distorcidas após a sua decodificação pelo sistema de recepção.

Assim, implemente uma memória associativa, por meio de uma rede de Hopfield com 45 neurônios, para armazenar e recuperar os padrões definidos acima a partir da apresentação de versões distorcidas ou incompletas das imagens.

Considere que:

- Pixel branco é codificado com valor -1.
- Pixel escuro é codificado com valor +1.

- Cerca de 20% dos pixels são corrompidos aleatoriamente durante a transmissão, ou seja, alguns que valiam -1 passa a valer +1 e vice-versa.

1. Simule 12 situações de transmissão (3 para cada padrão), conforme apresentado abaixo.
2. Mostre em cada situação a imagem distorcida e a imagem limpa recuperada.
3. Explique o que acontece quando se aumenta excessivamente o nível de ruído.

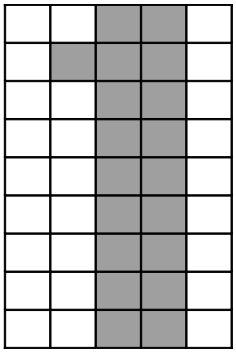


Imagem Transmitida (livre de ruído)

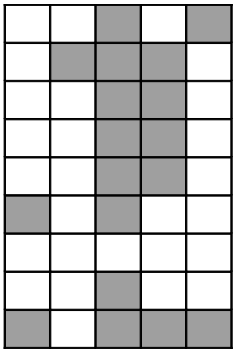


Imagem Distorcida (com ruído)

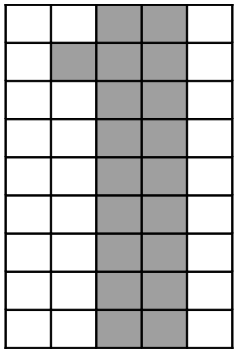


Imagem Limpa (sem ruído)

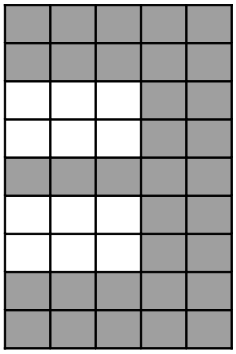


Imagem Transmitida

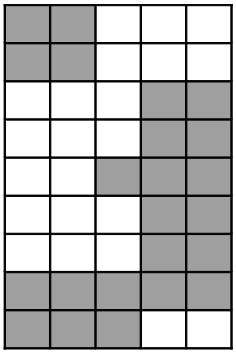


Imagem Distorcida

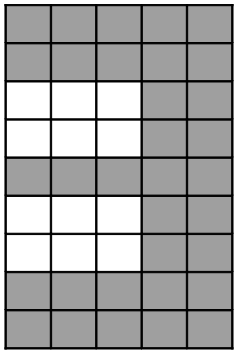


Imagem Limpa

Topologia da Rede de Hopfield:

- A matriz de pesos $\boldsymbol{W}$ é obtida pela regra do produto externo, que é dada por:

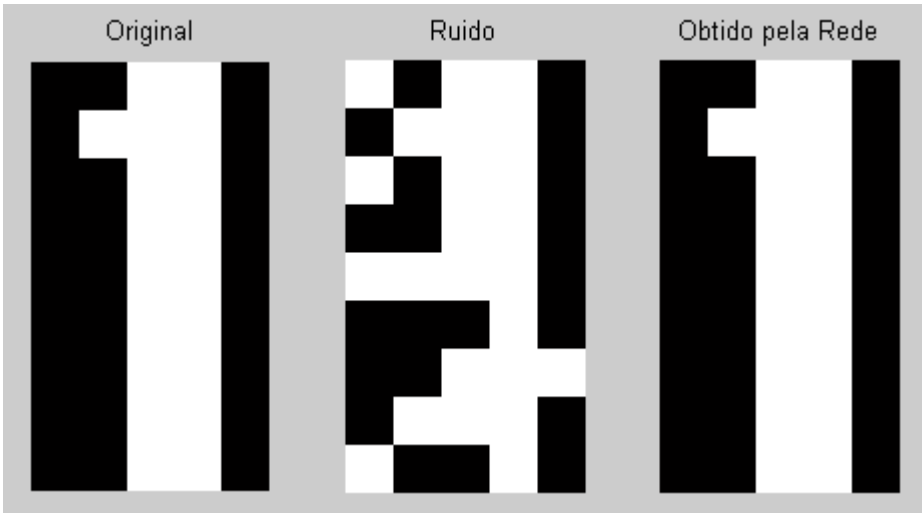
$$\boldsymbol{W} = \frac{1}{N}\sum_{k=1}^{P} \boldsymbol{z}^{(k)} \cdot (\boldsymbol{z}^{(k)})^{T} - \frac{P}{N} \cdot \boldsymbol{I}$$

- A função de ativação é a Tangente Hiperbólica com β muito grande.

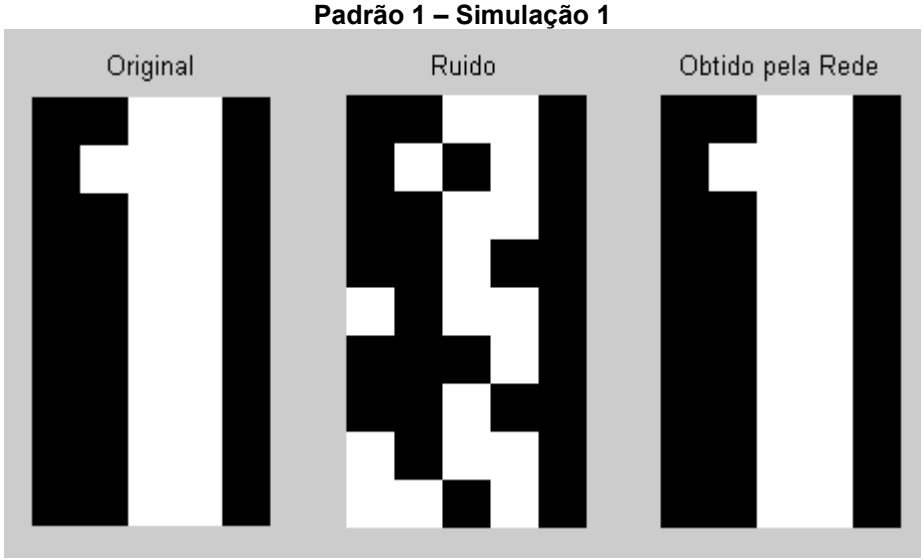
SOLUÇÃO:

1 – Foi criado um algoritmo para geração de ruído com taxa variável de interferência no sinal transmitido. O código-fonte (MatLab) encontra-se disponível no final do trabalho (anexo).

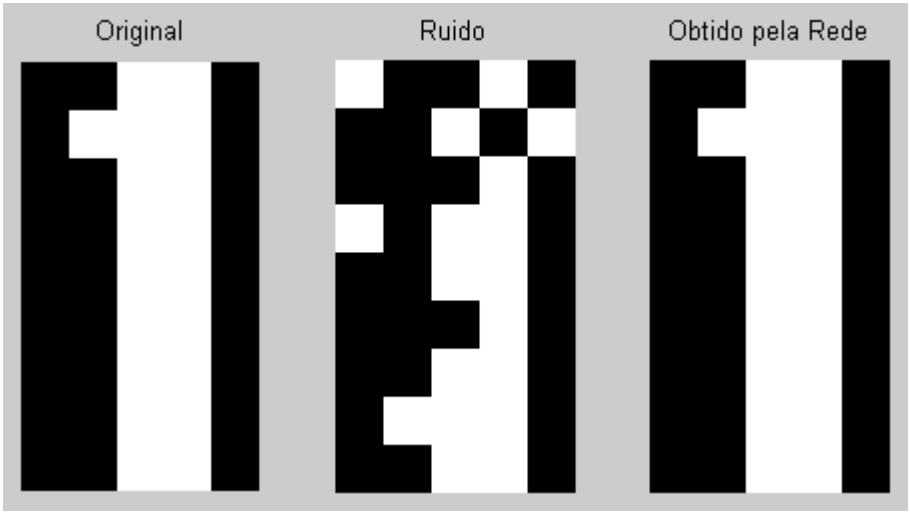
2 – Para uma taxa de interferência de 20% do sinal, os resultados obtidos nas 3 simulações para cada tipo de padrão seguem abaixo:



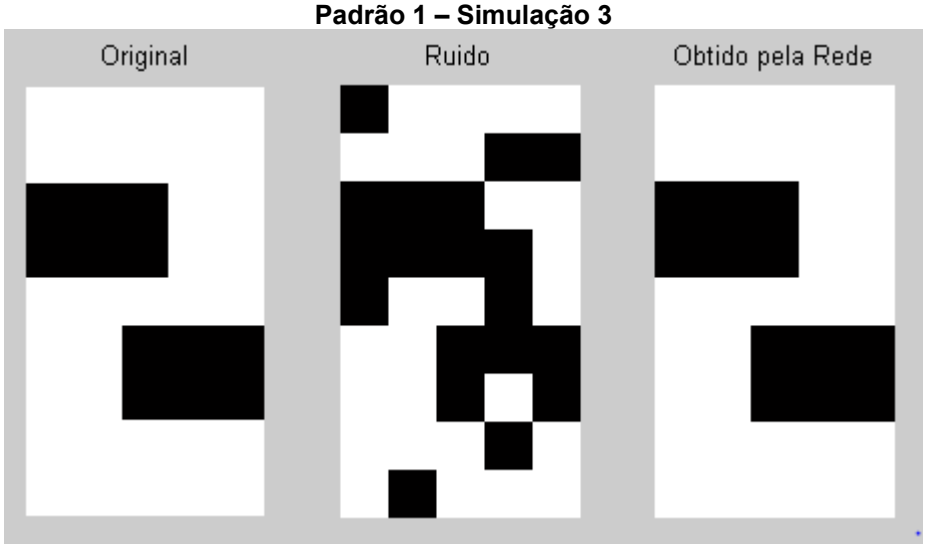
Padrão 1 – Simulação 1



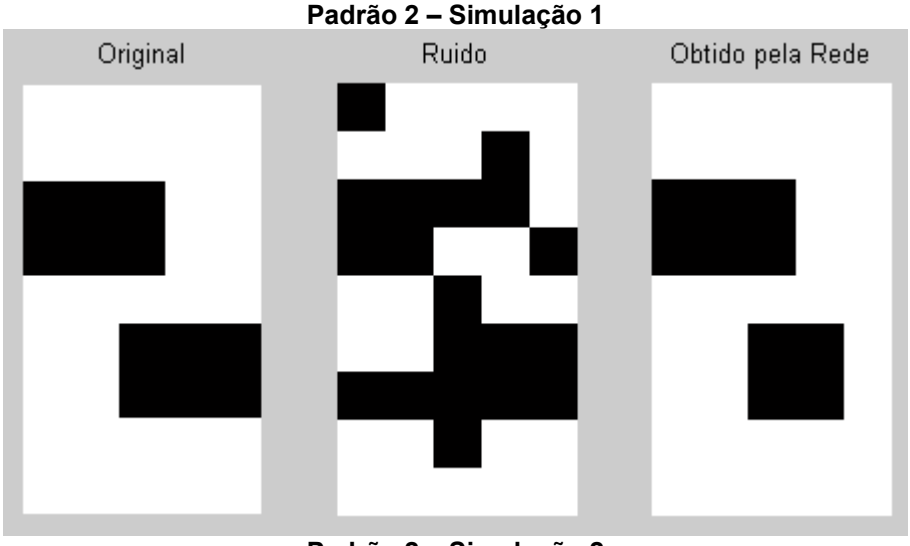
Padrão 1 – Simulação 2



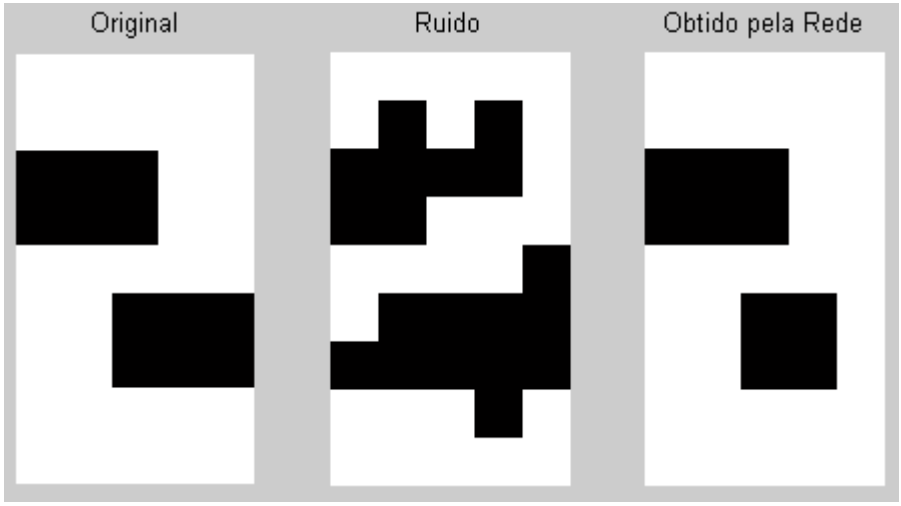
Padrão 1 – Simulação 3



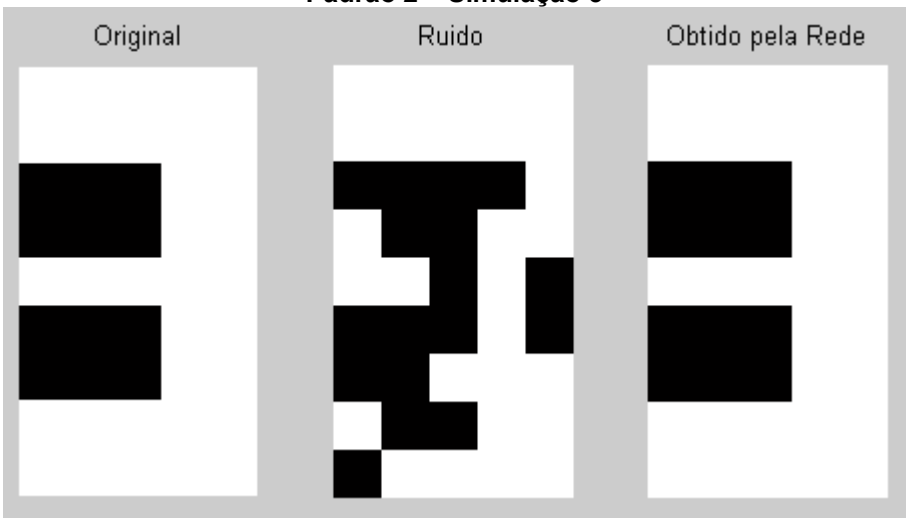
Padrão 2 – Simulação 1



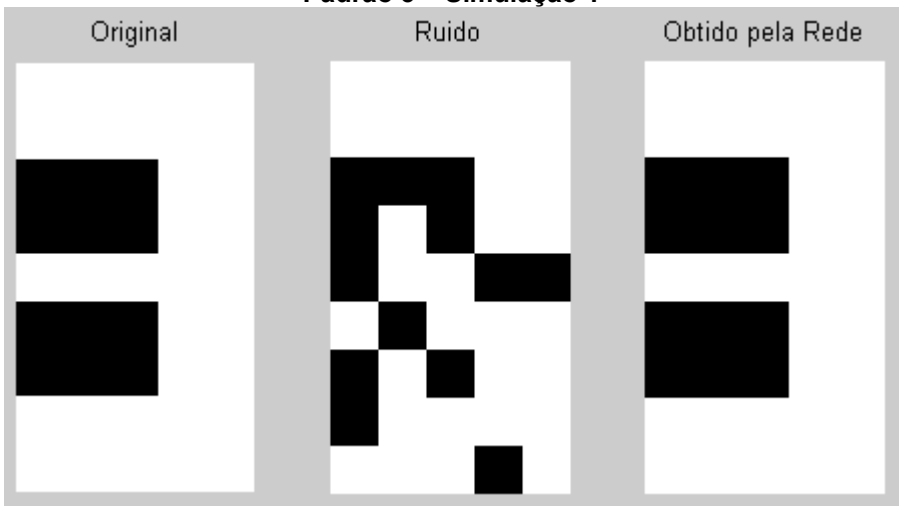
Padrão 2 – Simulação 2



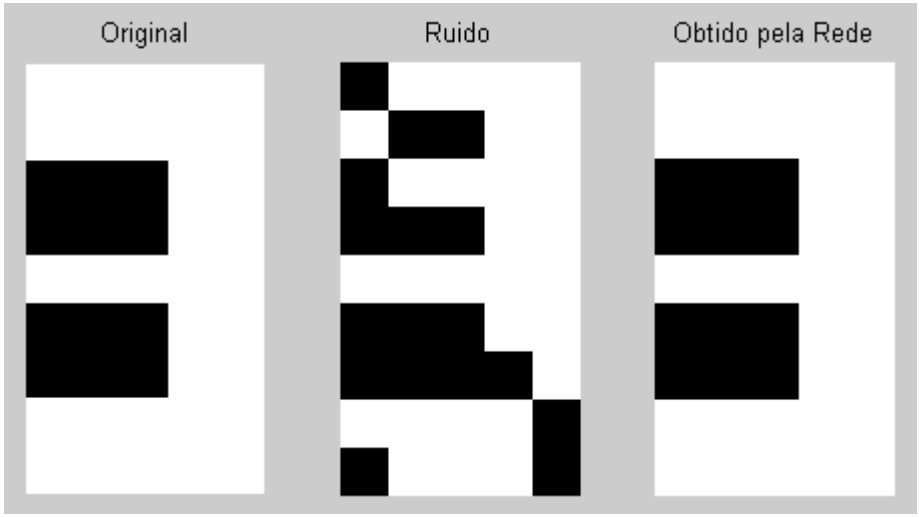
Padrão 2 – Simulação 3



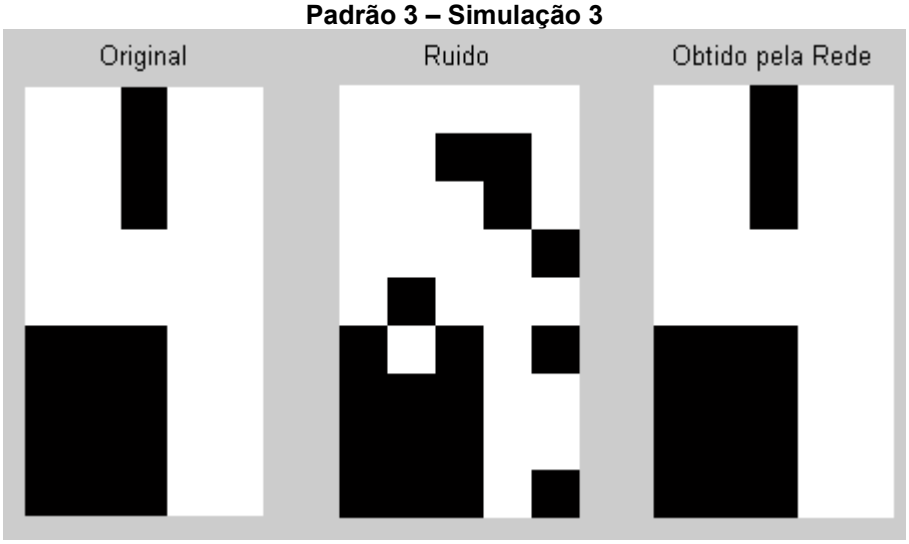
Padrão 3 – Simulação 1



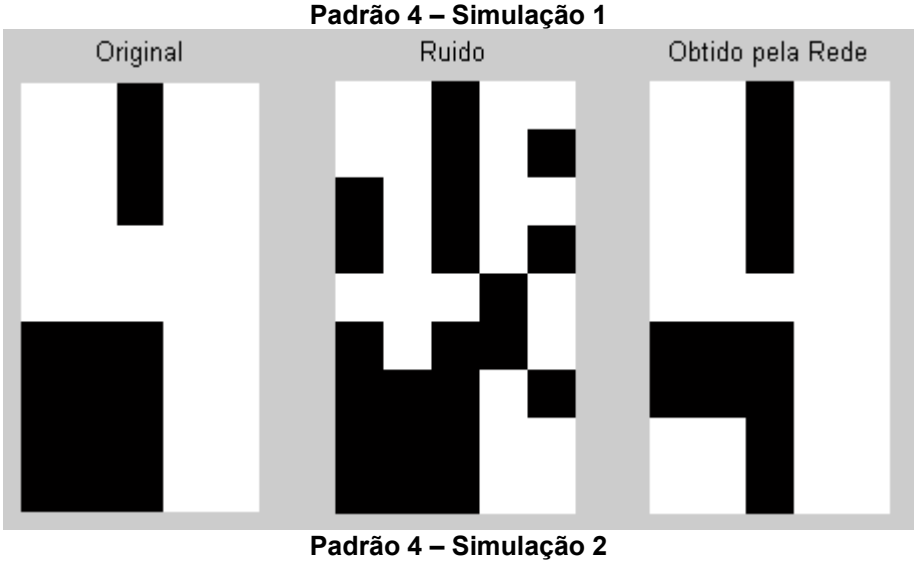
Padrão 3 – Simulação 2



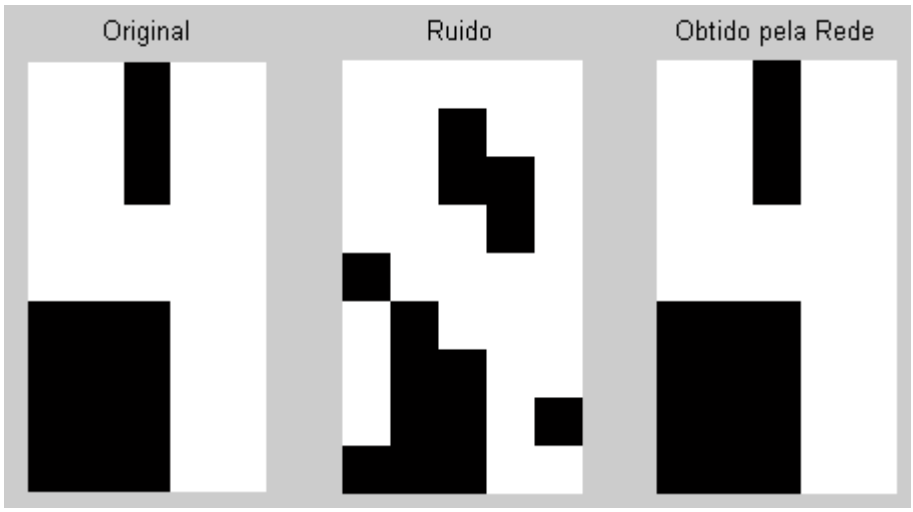
Padrão 3 – Simulação 3



Padrão 4 – Simulação 1



Padrão 4 – Simulação 2



Padrão 4 – Simulação 3

3 - Com o aumento excessivo do número de ruídos, a rede converge para outro ponto de equilíbrio estável (não do padrão original). Isso pode ser observado no resultado da figura 13, a qual a imagem original é de um “1”, mas foi recuperado um padrão de um “3”.

Pode ser também que a rede não consiga convergir a nenhum padrão, parando assim num ponto de equilíbrio instável. Isso pode ser observado na figura 14, a qual a imagem original é de um “3”, mas foi recuperado um padrão desconhecido. Apesar de o resultado obtido ser semelhante à imagem do número “2” mas com as cores invertidas, esse padrão não é uma referência (valor original) para rede.

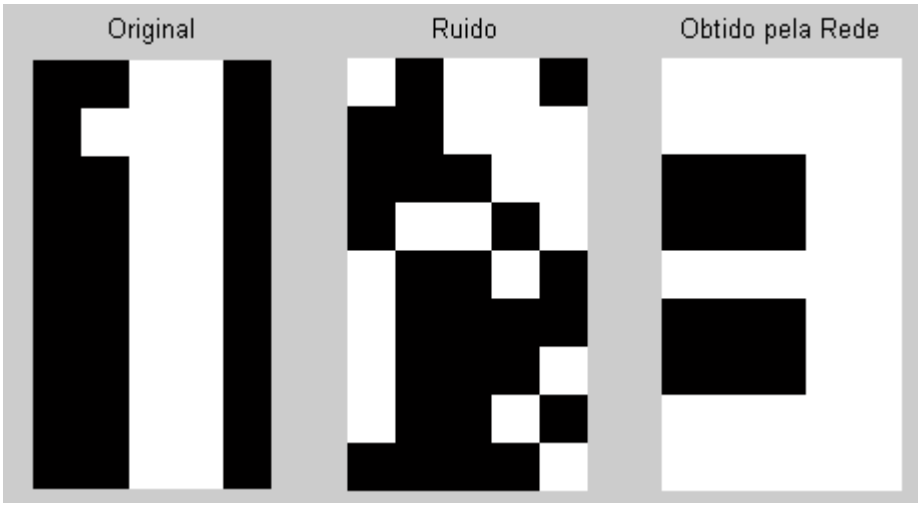


Figura 13 – Símbolo '1' com excesso de ruído convergindo para '3'

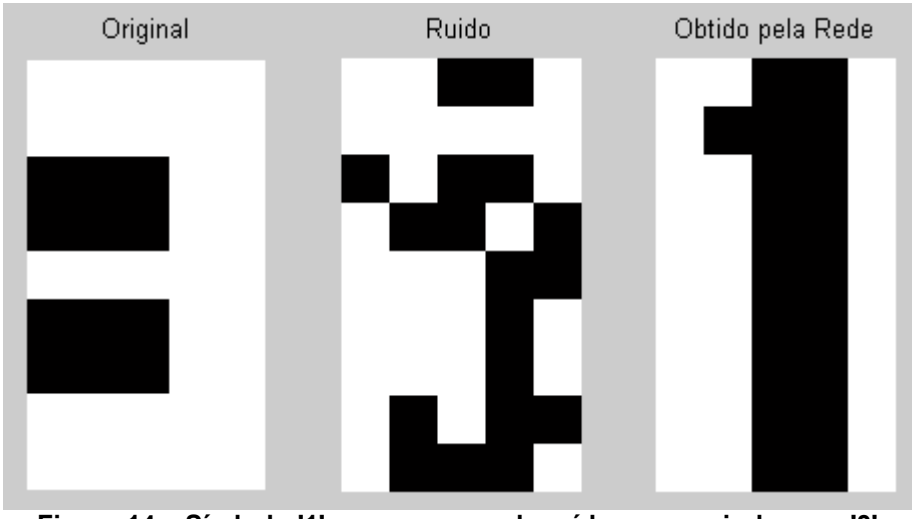


Figura 14 – Símbolo '1' com excesso de ruído convergindo para '3'

CÓDIGO FONTE

PRODUTO EXTERNO:

```
function [ W ] = ProdutoExterno( p, n, amostras )
%ProdutoExterno Realiza o PE das amostras
%   Detailed explanation goes here

soma = zeros(n,n);

for k=1:p
      soma = soma + amostras(:, k) * amostras(:, k)';
end;

W = soma / n - (p/n)*eye(n);
```

OPERAÇÃO DA REDE

```
function [ v_final ] = HopfieldOperacao( W, ib, x0 )
%HopfieldOperacao Executa a Rede de Hopfield
% W  -> Pesos
% ib -> vetor de limiares
% x0 -> vetor inicial de entradas

v_atual    = x0;
v_anterior = zeros(size(v_atual, 1), size(v_atual, 2) );

beta = 100;
epson = 1e-8;

while( sum( abs( v_anterior - v_atual ) ) > epson )
    v_anterior = v_atual;
    u = W * v_anterior + ib;
     v_atual = sign(u); %alem de ser discreto, tanh com beta muito alto é praticamente a funcao 'sinal'.
%     v_atual = ( 1 - exp( -2 * beta * u ) ) ./ ( 1 + exp( -2 * beta * u ) );
%      v_teste = tanh(u); a expressao acima da os mesmos resultados de tanh
%       quando beta =1 (com diferença uma distante casa decimal)
end;

v_final = v_atual;
```

GERAÇÃO DE INTERFERÊNCIA NO SINAL

```
function Zout = GeraRuido( Zin, NivelDeRuido )
%GeraRuido Gera um Ruido em 20% dos pixels da amostra
%   Detailed explanation goes here

%Quantidade de pixels que sofrera interferencia
n = NivelDeRuido * length(Zin);

Zout = Zin;
vpositions = [];
for i=1:n
      %Sorteia um pixel para sofrer 'ruido'
    b_newposition = false;
       while( ~b_newposition ) %este while garante que uma posicao ainda nao alterada pelo ruido (inedita) sera de fato sorteada
                position = ceil(rand * length(Zin)); %ceil arredonda pra cima, garantindo que a menor posiçao seja 1 e a maior 45
```

```
          if( isempty(find( vpositions
== position )) )
             %esta posicao é inetida ou
seja, nao foi alterada ainda pelo
             %ruido
             vpositions(i) = position;
             b_newposition = true;
         end;
     end;
      Zout(position) = Zout(position) *
(-1); %inverte o pixel = ruido
end;
```

FUNÇÃO AUXILIAR PARA EXIBIÇÃO DE IMAGEM

```
function [ out ] = AjeitaVetor( in )
%UNTITLED1 Ajeita o vetor para poder
ser visualizado como a matriz do EPC8
%   Detailed explanation goes here

out = [in(1:5); in(6:10); in(11:15);
in(16:20);    in(21:25);    in(26:30);
in(31:35); in(36:40); in(41:45);];
```

CARREGAMENTO DE DADOS

```
function [ Z1, Z2, Z3, Z4 ] =
CarregaDadosEntrada( input_args )
%CarregaDadosEntrada    Carrega    os
vetores de amostras
%   Detailed explanation goes here

%Numeros codificados em uma grade 9 x
5
%Numero '1'
Z1 = [ -1 -1  1  1 -1 ...
       -1  1  1  1 -1 ...
       -1 -1  1  1 -1 ...
       -1 -1  1  1 -1 ...
       -1 -1  1  1 -1 ...
       -1 -1  1  1 -1 ...
       -1 -1  1  1 -1 ...
       -1 -1  1  1 -1 ...
       -1 -1  1  1 -1 ...
     ]';

 %Numero '2'
Z2 = [  1  1  1  1  1 ...
        1  1  1  1  1 ...
       -1 -1 -1  1  1 ...
       -1 -1 -1  1  1 ...
        1  1  1  1  1 ...
        1  1 -1 -1 -1 ...
        1  1 -1 -1 -1 ...
        1  1  1  1  1 ...
        1  1  1  1  1 ...
     ]';
```

```
%Numero '3'
Z3 = [  1  1  1  1  1 ...
        1  1  1  1  1 ...
       -1 -1 -1  1  1 ...
       -1 -1 -1  1  1 ...
        1  1  1  1  1 ...
       -1 -1 -1  1  1 ...
       -1 -1 -1  1  1 ...
        1  1  1  1  1 ...
        1  1  1  1  1 ...
     ]';

 %Numero '4'
Z4 = [  1  1 -1  1  1 ...
        1  1 -1  1  1 ...
        1  1 -1  1  1 ...
        1  1  1  1  1 ...
        1  1  1  1  1 ...
       -1 -1 -1  1  1 ...
       -1 -1 -1  1  1 ...
       -1 -1 -1  1  1 ...
       -1 -1 -1  1  1 ...
     ]';
```

EXECUÇÃO DO EPC

```
clear;
close all;
clc;


%Numero de padroes usados: p (4)
%Numero de dados usados: n (45)

[Z1,      Z2,      Z3,      Z4]      =
CarregaDadosDeEntrada();

p = 4;
n = length(Z1); %45'
NivelDeRuido = 0.5; %no caso, 20%
SimulacoresPorAmostra = 3;

%Gera matriz de pesos da Rede
Z = [Z1 Z2 Z3 Z4];
W = ProdutoExterno(p, n, Z);

%Gera vetor de limiares
ib = zeros(45, 1);

%Execucao
for k=1:p % vai pegar 1 amostra por
vez
     for i=1:SimulacoresPorAmostra %3
ruidos por amostra
        %Gera ruido para uma amostra
            x0 = GeraRuido( Z(:, k),
NivelDeRuido );

        %Executa a Rede
```

```
        v_final = HopfieldOperacao( W,                    subplot(  1,3,1  ),
ib, x0 );                                 imshow(          original          ),
                                          title( 'Original' );
        % v_final                                          subplot(  1,3,2  ),
        % sum(v_final == Z1)              imshow( ruido ), title( 'Ruido' );
                                                          subplot(  1,3,3  ),
            %Monta Figura com dados: imshow( obtido ), title( 'Obtido pela
original, ruidoso e obtido pela rede    Rede' );
        figure                                        saveas( gcf, sprintf('%d_
        title( sprintf( 'Amostra: %d - %d.jpg', k, i) );
Simulação: %d', k, i ));                    end;
          original = AjeitaVetor(Z(:, end;
k)');
        ruido    = AjeitaVetor(x0');
                       obtido        =
AjeitaVetor(v_final');
```