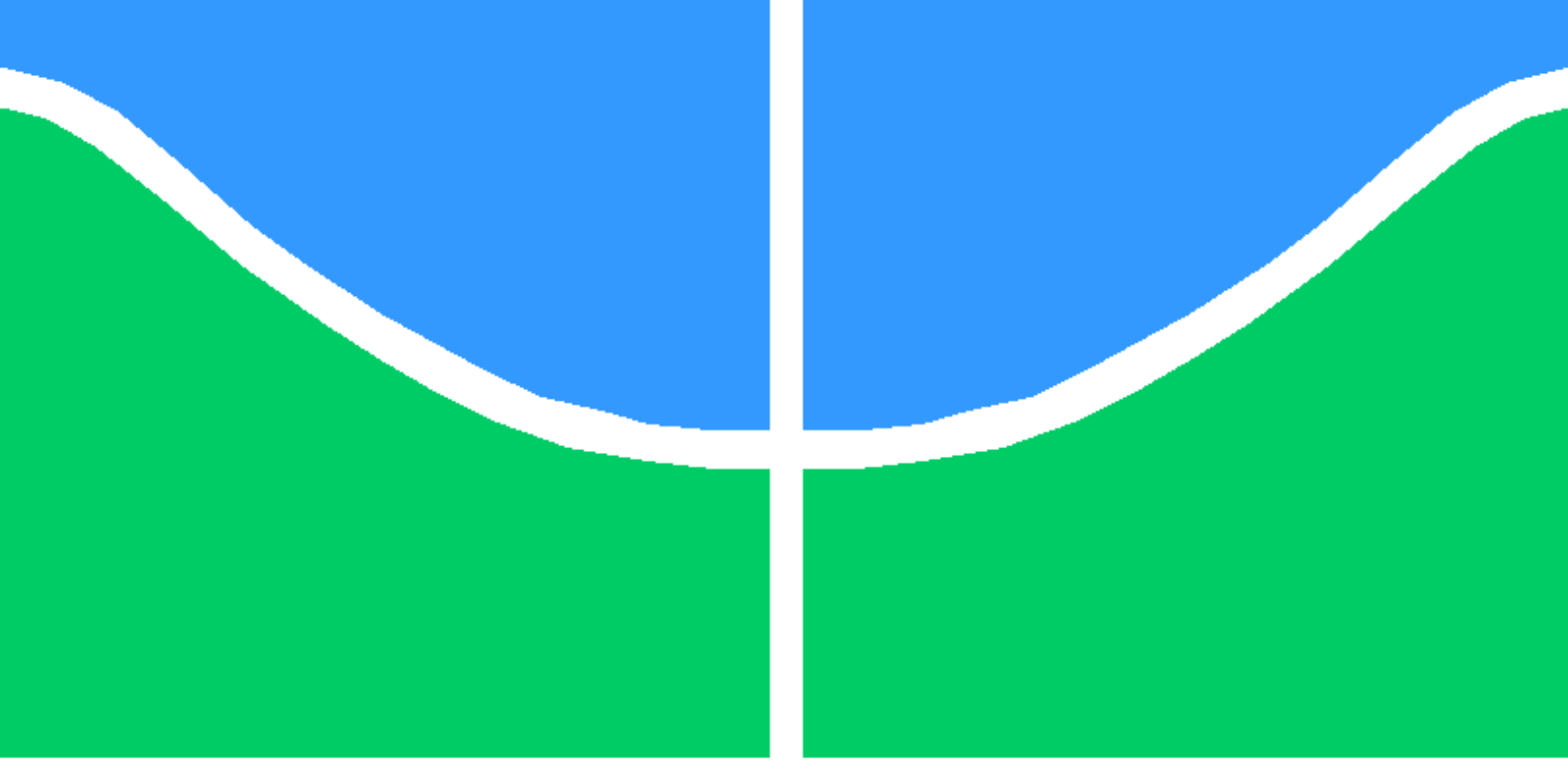

ANÁLISE DE COMPONENTES INDEPENDENTES EM MONTE CARLO PARALELO

HEITOR ALVES RIOS CAMPOS

TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ANÁLISE DE COMPONENTES INDEPENDENTES EM MONTE CARLO PARALELO

HEITOR ALVES RIOS CAMPOS

Orientador: PROF. DR. DANIEL GUERREIRO E SILVA, ENE/UNB

TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

BRASÍLIA-DF, 04 DE DEZEMBRO DE 2018.

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

ANÁLISE DE COMPONENTES INDEPENDENTES EM MONTE CARLO PARALELO

HEITOR ALVES RIOS CAMPOS

TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO ACADÊMICO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE EM ENGENHARIA ELÉTRICA.

APROVADA POR:

Prof. Dr. Daniel Guerreiro e Silva, ENE/UnB
Orientador

Prof. Dr. José Edil Guimarães de Medeiros, ENE/UnB
Examinador interno

Prof. Dr. Leonardo Rodrigues Araújo Xavier de Menezes, ENE/UnB
Examinador interno

BRASÍLIA, 04 DE DEZEMBRO DE 2018.

FICHA CATALOGRÁFICA
HEITOR ALVES RIOS CAMPOS
ANÁLISE DE COMPONENTES INDEPENDENTES EM MONTE CARLO PARALELO
2018xv, 52p., 201x297 mm
(ENE/FT/UnB, , Engenharia Elétrica, 2018)
Trabalho de conclusão de curso de graduação - Universidade de Brasília
Faculdade de Tecnologia - Departamento de Engenharia Elétrica

REFERÊNCIA BIBLIOGRÁFICA

HEITOR ALVES RIOS CAMPOS (2018) ANÁLISE DE COMPONENTES INDEPENDENTES EM MONTE CARLO PARALELO. Trabalho de conclusão de curso de graduação em Engenharia Elétrica, Publicação , Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 52p.

CESSÃO DE DIREITOS

AUTOR: HEITOR ALVES RIOS CAMPOS
TÍTULO: ANÁLISE DE COMPONENTES INDEPENDENTES EM MONTE CARLO PARALELO.

_____

HEITOR ALVES RIOS CAMPOS
SHIN QI 2 CONJUNTO 12 CASA 21

Agradecimentos

À minha namorada e companheira Mariana Santana pelo carinho, pelo companheirismo, pelo apoio, pelo estímulo e principalmente pela a paciência.

Ao meu orientador professor Daniel Guerreiro e Silva, por sua amizade, apoio e valiosa orientação ao longo deste trabalho.

Aos meus irmãos Arthur e Otto, pelo o apoio e incentivo.

Ao meu pai Jerônimo pela a educação e pela criação que me permitiu chegar aqui.

À todos os meu amigos pela a amizade, e presença no momentos difíceis.

Aos colegas de curso pelo o auxílio e companheirismo ao longo de todo curso.

Um agradecimento especial a toda minha família.

A todos que contribuíram para realização desta etapa da minha vida, que foram e são muito importantes para mim, meu sincero muito obrigado.

Resumo

Neste estudo verificamos se o uso de técnicas de análise de componentes independentes em amostras pseudo-aleatórias é capaz de melhorar a convergência de Monte Carlo paralelo. Uma das razões da lenta convergência do método de Monte Carlo paralelo é a correlação entre as amostras utilizadas para alimentar o método. Para fazer essa verificação utilizamos duas técnicas distintas de ICA: FastICA e mapas auto-organizáveis. Um método de Monte Carlo simples foi proposto para que fosse possível fazer essa verificação. Também foi feita a comparação dos resultados usando dois modelos de gerador de números pseudo-aleatórios. Os resultado obtidos mostraram que nas condições propostas neste estudo tanto o FastICA quanto mapas auto-organizáveis não foram capazes gerar amostras menos correlacionadas e portanto não são alternativas viáveis em otimizar o método de Monte Carlo paralelo.

$\boldsymbol{Palavras-chaves}$: Análise de componentes independentes, Mapas auto-organizáveis, Monte Carlo, Gerador de número pseudo-aleatório.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

Capítulo 1

Introdução

Uma definição formal de Método de Monte Carlo foi dada por Halton em 1970. Ele definiu o método como uma técnica para representar a solução de um problema como um parâmetro de uma população hipotética e, que usa uma sequência aleatória de números para construir uma amostra da população da qual estimativas estatísticas desse parâmetro possam ser obtidas.

O primeiro trabalho com método de Monte Carlo foi em 1947 com Jon Von Neuman e Stanislaw Ulam. Conforme colocado em [Ulam J. von Neumann 1947] e posteriormente em [Metropolis and Ulam 1949], eles propuseram usar uma simulação computacional em uma parte do projeto Manhattan, na Segunda Guerra Mundial. No projeto de construção da bomba atômica, Ulam e Jon Von Neumann consideraram a possibilidade de utilizar o método, que envolvia a simulação direta de problemas de natureza probabilística relacionados com o coeficiente de difusão do nêutron em certos materiais.

O nome do método se originou pelo uso da aleatoriedade e da natureza repetitiva das atividades realizadas em cassinos em Monte Carlo, Mônaco. O método de Monte Carlo tem sido utilizado há bastante tempo como forma de obter aproximações numéricas de funções complexas. Estes métodos tipicamente envolvem a geração de observações de alguma distribuição de probabilidades e o uso da amostra obtida para aproximar a função de interesse. O método é também referido como simulação estocástica e é um método relativamente simples e fácil de implementar.

Uma das desvantagens do método de Monte Carlo quando comparado com outros métodos numéricos é a sua convergência. A convergência deste método é lenta, da ordem de $1/\sqrt{N}$ em que N é o número de amostras. Sendo assim é desejável paralelizar o método de Monte Carlo para que a convergência seja mais rápida. Porém ao paralelizar Monte Carlo um problema surge: as amostras obtidas pelos geradores de números aleatórios para cada simulação são correlacionadas entre si, isso faz com o ganho em convergência não seja tao significativo [Hellekalek 1998].

Uma possível solução para minimizar a correlação destas amostras é o uso de uma téc-

nica chamada de análise de componentes independentes. A Análise de Componentes Independentes (ACI), ou *Independent Components Analysis* (ICA) em inglês, é um método computacional que tenta decompor um conjunto de sinais em componentes que sejam estatisticamente independentes entre si. Para isso, aos algoritmos clássicos assumem que estas componentes também tenham distribuições não-gaussianas. Um caso especial de ICA é à separação cega de fontes, ou *Blind Source Separation* (BSS), e um exemplo clássico de BSS é o "cocktail party".

Para entender o problema do "cocktail party", considere duas pessoas falando em uma sala fechada e que esta sala possua microfones que capturam os sinais de voz destas pessoas. O sinal obtido por tais microfones será uma mistura entre a voz de cada pessoa nesta sala. Esta situação está representada na Figura 1.1. O problema consiste em recuperar os sinais de voz de cada pessoa sabendo apenas os sinais capturados pelos microfones. A particularidade da separação cega de fontes, entre outras técnicas de filtragem, é que neste caso não é preciso conhecer as fontes originais nem a forma como os sinais foram misturados [Aapo Hyvärinen 2001].

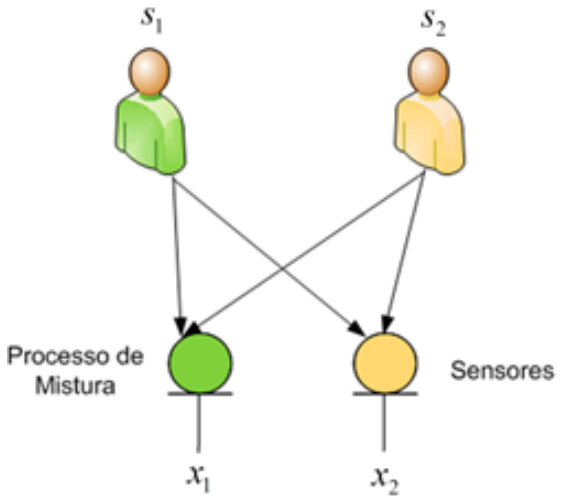

Figura 1.1: Modelo simplificado do problema "coktail party"com duas fontes.

Para melhor exemplificar o problema de separação cega de fontes, suponhamos um sistema com três sinais independentes: uma onda quadrada, uma onda triangular e uma onda senoidal, misturados de forma linear. A figura 1.2 representa este sistema.

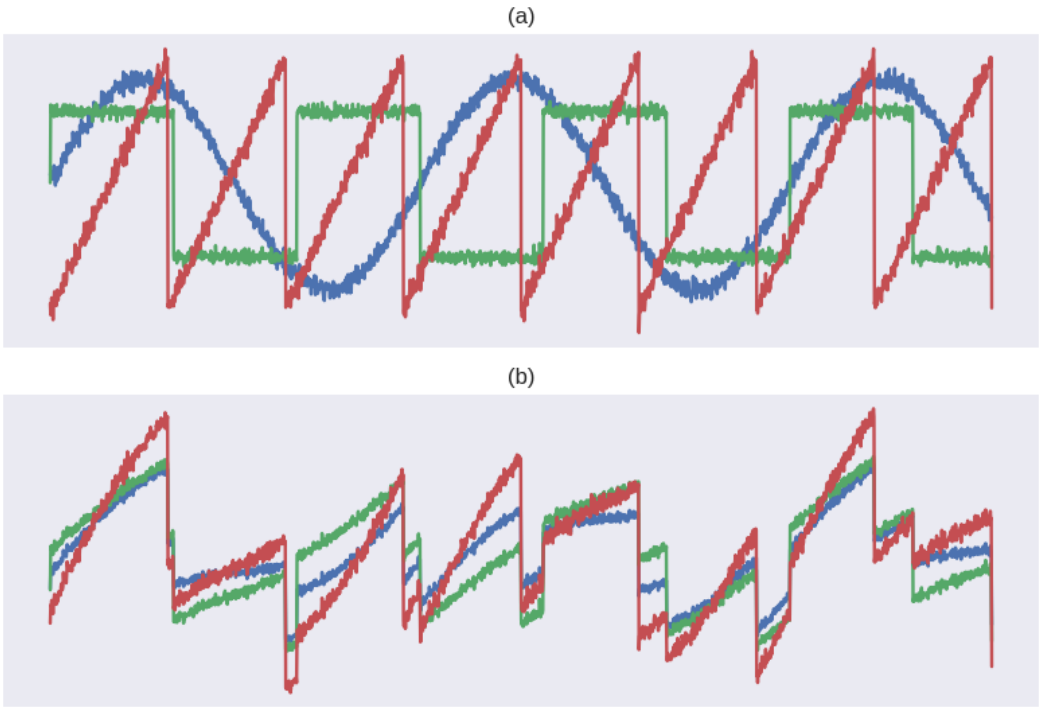

Figura 1.2: Exemplo de BSS com três fontes de sinais; (a) Representa os sinais originais; (b) Os sinais observados após a mistura.

Com base na figura a cima, o problema de separação cega de fontes é estimar a fontes originais (a) a partir apenas dos sinais observados (b). Várias técnicas de ICA foram desenvolvidas para poderem solucionar um problema como este e algumas delas serão comentadas neste estudo.

A análise de componentes independentes foi originalmente desenvolvida para resolver um problema análogo ao problema do "*cocktail-party*". Porém esta técnica se mostrou ser bastante interessante em outras abordagens. Considere, por exemplo, uma gravação da atividade elétrica do cérebro é dada por um eletroencefalograma (EEG). A informação do EEG consiste em registros do potencial elétrico em varias regiões do cérebro. Esta situação é bastante similar ao problema do "*cocktail-party*": desejamos encontrar as componentes originais da atividade cerebral, mas temos acesso apenas a uma mistura destes sinais. Portanto, ICA pode revelar informações interessantes sobre a atividade cerebral dando acesso a suas componentes independentes [Hyvärinen and Oja 2000].

Portanto, o uso de técnicas de ICA em amostras de números pseudo-aleatórios, que posteriormente serão usadas para fazer algum calculo numérico pelo método de Monte Carlo paralelo, pode fazer com que a convergência deste método melhore. Já que, se aplicação de ICA for bem sucedida, uma das causas da convergência lenta deste método será solucionada, que é a correlação entre as amostras utilizadas pelo método.

Para resolver o problema de descorrelacionar amostras pseudo-aleatórias duas técnicas de ICA se mostram promissoras: o FastICA, que é um algoritmo de ponto fixo e o mais utilizado em aplicações simples de ICA, e ICA por mapas auto-organizáveis, técnica que utiliza uma rede neural para poder estimar as componentes independentes [Hyvarinen 1999] [Pajunen et al. 1996]. A escolha do FastICA se deu pelo o fato de ser o algoritmo mais consolidado em ICA. Já ICA por mapas auto-organizáveis foi escolhido por ser uma técnica não-linear de ICA.

Objetivo Geral

O objetivo deste trabalho é verificar se o uso das técnicas de análise de componentes independentes, FastICA e mapas auto-organizáveis, é capaz de descorrelacionar amostras de números pseudo-aleatórios e por consequência melhorar a convergência de um problema de Monte Carlo paralelo.

Organização do Documento

No capítulo 2, é apresentada a revisão de análise de componentes independentes, incluindo uma técnica linear e outra abordagem não-linear, método de Monte Carlo e alguns pontos importantes para o entendimento do trabalho.

No capítulo 3, é explicada a metodologia usada neste trabalho.

No capítulo 4, são mostrados os resultados obtidos no trabalho.

E por fim, no capítulo 5 é apresentada as conclusões e ideias de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Neste capitulo será abordado os fundamentos teóricos suficientes para o entendimento deste trabalho. Para uma leitura mais aprofundada sobre os temas aqui tradados recomenda-se as referências citadas ao longo deste capitulo.

2.1 Simulação de Monte Carlo

O método de Monte Carlo é usado para determinar heuristicamente soluções numéricas através de parâmetros gerados aleatoriamente. Geralmente, estas soluções consistem em distribuições de valores, já que o problema é alimentado por um conjunto de parâmetros pseudo-aleatórios[Landau et al. 2015].

A integração de Monte Carlo é um caso especial do método de Monte Carlo, é baseada no teorema do valor médio:

$$I = \int_a^b f(x)dx = (b-a)\langle f \rangle \tag{2.1}$$

Este teorema é simples de compreender se pensar em integrais como sendo uma área. O valor da integral de alguma função $f(x)$ entre os pontos a e b é igual ao comprimento do intervalo $(b-a)$ vezes o valor médio da função neste mesmo intervalo $\langle f \rangle$ [Landau et al. 2015]. O algoritmo de integração de Monte Carlo usa pontos aleatórios para poder estimar a média da função. Para o cálculo da média de uma amostra, considere uma sequência $a \leq x_i \leq b$ com N números uniformemente distribuídos, então:

$$\langle f \rangle \simeq \frac{1}{N} \sum_{i=1}^{N} f(x_i) \tag{2.2}$$

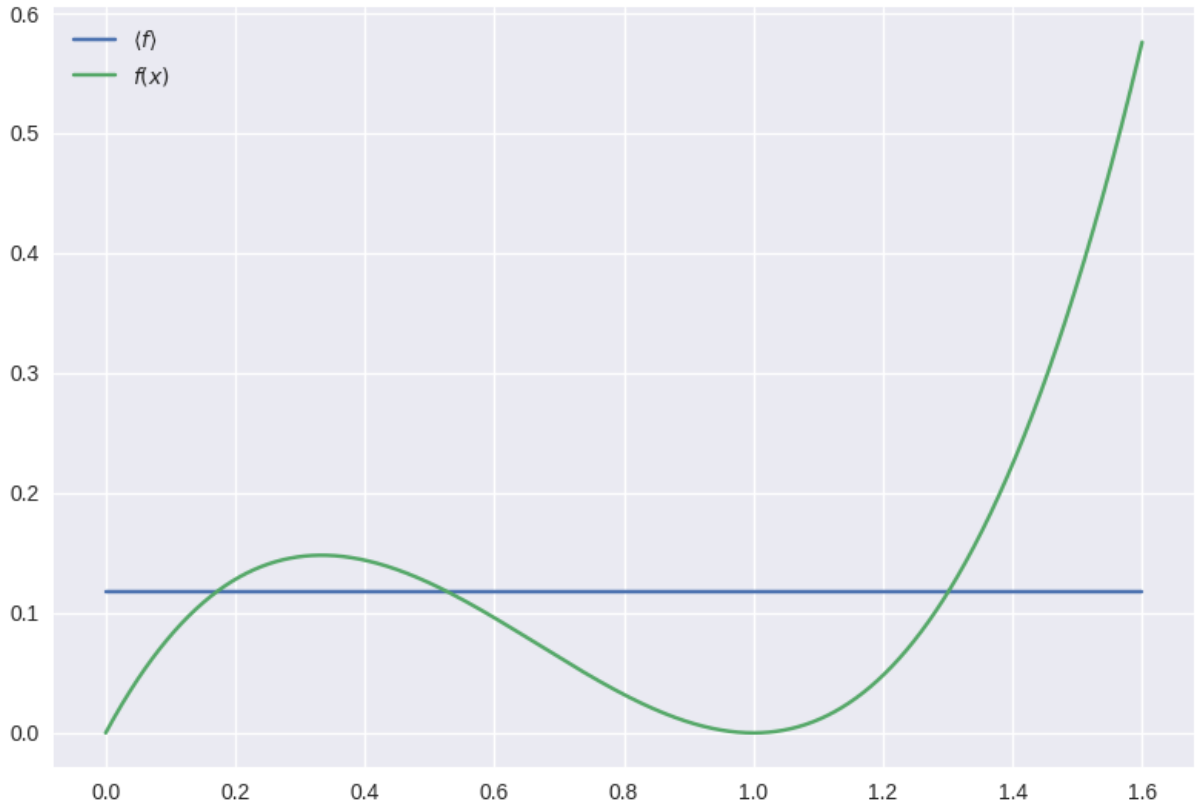

Figura 2.1: Representação do teorema do valor médio. A curva verde é uma função qualquer no intervalo [0;1.6] enquanto a curva em azul é a media dessa função no mesmo intervalo. Portanto a área sob ambas as curvas é a igual [Apostol 2007].

Usando a equação 2.1 e chegamos em uma forma simples de integração numérica:

$$\int_a^b f(x)dx \simeq (b-a)\frac{1}{N}\sum_{i=1}^{N} f(x_i) = (b-a)\langle f\rangle \tag{2.3}$$

Se o valor do número de amostras de $f(x)$ se aproximar de infinito, $N \to \infty$, ou se manter o número finito de amostras mas pegar a média entre infinitas integrações, o valor da equação 2.3 se aproxima do valor exato.

É importante enfatizar que a equação 2.3 só é valida se todos os números amostrados possuem a mesma chance de serem escolhidos, ou seja, são uniformemente distribuídos. Quanto mais uniforme a distribuição melhor será o resultado deste método numérico [Newman and Barkema 1999]. Nos métodos clássicos de Monte Carlo, como este, a uniformidade das amostras é mais importante do que a aleatoriedade destas [Dongarra et al. 2003]. Em casos em que os números amostrados não são uniformemente distribuídos uma outra técnica de Monte Calo pode ser usada:

$$I = \lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)} \tag{2.4}$$

Em que $p(x)$ é uma distribuição qualquer.

A incerteza no valor obtido pela integral após N amostras de $f(x)$ é calculada pelo o desvio padrão σ_I. Considerando que σ_f é o desvio padrão do integrando f na amostragem, então para distribuições normais temos que [Landau et al. 2015]:

$$\sigma_I \simeq \frac{1}{\sqrt{N}}\sigma_f \tag{2.5}$$

Logo, para N muito grande, o erro do valor obtido pela integral é proporcional a $\frac{1}{\sqrt{N}}$. Quando comparado com outros métodos numéricos de integração Monte Carlo possui uma convergência muito lenta. O método de Simpsons por exemplo possui o erro proporcional a $1/\sqrt{N^4}$. Porém o método de Monte Carlo é mais eficaz para integrais com varias dimensões isso porque o erro do método de Monte Carlo é proporcional a $1/\sqrt{N}$ independentemente do número de dimensões [Newman and Barkema 1999].

Paralelizar o método de Monte Carlo seria a forma mais intuitiva para melhorar a convergência deste método, já que o algoritmo é facilmente paralelizável. Porém ao se paralelizar o método alguma forma de paralelização na geração números amostrados também se faz necessária, e isso gera um problema: os números gerados em paralelo são de alguma forma correlacionados entre si. Por este motivo paralelizar Monte Carlo não gera tanto ganho em convergência [Hellekalek 1998].

2.2 Gerador de Números Pseudo-Aleatórios

Como visto na seção anterior para a aplicação do método de Monte Carlo se faz necessário a amostragem de um conjunto de números aleatórios. Para isso pode-se usar um gerador de números verdadeiramente aleatórios, porém uma característica deste tipo de gerador o torna indesejável: os resultados obtidos não são reproduzíveis. A reprodução de resultados do método de Monte Carlo é uma importante etapa para verificar se o resultado é válido ou não. Para isso utiliza-se então um gerador de números pseudo-aleatórios em detrimento de geradores verdadeiramente aleatórios.

Um gerador de números pseudo-aleatório é um algoritmo para gerar uma sequência de números em que suas propriedades se assemelhem a sequências de números verdadeiramente aleatórios. Uma sequência de números originados a partir de um um gerador pseudo-aleatório não é completamente aleatório, já que, a sequência segue um padrão determinístico. Isto é, sabendo o valor inicial, é possível prever qualquer número desta sequência.

Algumas características definem um bom gerador de números pseudo-aleatórios [Hellekalek 1998]:

- Ter distribuição uniforme;
- Os números gerados serem descorrelacionados;

- Os números gerados serem reproduzíveis;
- O período do gerador deve ser algumas ordens de grandeza maior que a quantidade de números gerados;

O gerador de números pseudo aleatório mais utilizado é o Mesenne Twister. Esse algoritmo é baseado no número primo de Mersenne, $2^{19937} - 1$. Este gerador tem o período muito grande, o que faz dele um método muito atrativo em aplicações como criptografia [Matsumoto and Nishimura 1998].

2.2.1 Registrador de Deslocamento com Retroalimentação Linear

Um tipo simples de gerador de números pseudo-aleatórios é o registrador de deslocamento com retroalimentação linear ou *Linear Feedback Shift Register* ou LFSR. Este gerador nada mais é do que um circuito registrador de deslocamento com realimentação feita através de operações lineares. A realimentação é realizada com a utilização de portas lógicas do tipo "ou-exclusivo".

O valor inicial do LFSR é chamado de semente, e como a operação do registrador é determinística, a sequência de números produzido pelo gerador é completamente dependente do estado anterior. Como o registrador tem um número finito de estados possíveis, eventualmente o gerador passa a gerar números repetidos[Poorghanad et al. 2008].

O período máximo de um gerador LFSR é $2^n - 1$ em que n é o número de bits do registrador usado.

A figura 2.2 mostra o modelo de um gerador LFSR de 4-bits em que o estado atual é 0001.

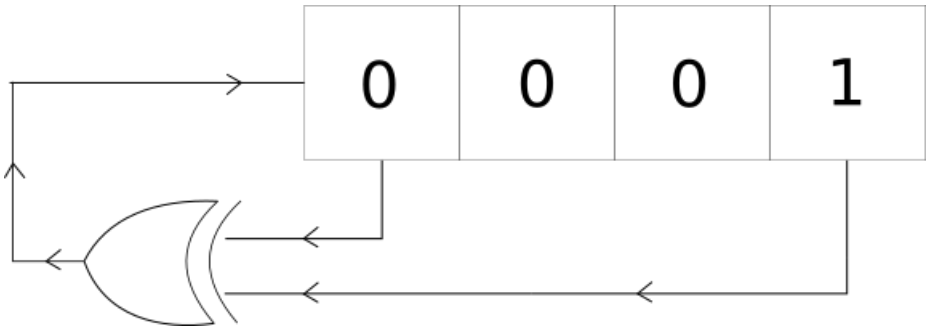

Figura 2.2: Registrador de deslocamento funcionando como um gerador LFSR com valor inicial de 0001. A realimentação é feita pela a operação "ou-exclusivo"nos bits 1 e 4

É possível, na tabela 2.1, observar que a saída do gerador não parece seguir nenhum padrão conhecido e portanto se parece com um gerador verdadeiramente aleatório. Na linha 16 da tabela 2.1 o valor do registrador volta a 0001, que é o valor inicial, e a partir desse ponto os valores são todos repetidos, portanto o período deste gerador é 15. O valor 0000

Registrador	Feedback	Saida
0001	1	1
1000	1	8
1100	1	12
1110	1	14
1111	0	15
0111	1	8
1011	0	11
0101	1	5
1010	1	10
1101	0	13
0110	0	6
0011	1	3
1001	0	9
0100	0	4
0010	1	2
0001	1	1
1000	1	8
1100	1	12

Tabela 2.1: sequência de valores obtidos usando um gerador LFSR de 4 bits e valor inicial 0001.

no registrador não pode ser usado como valor inicial porque caso isso aconteça o valor da retroalimentação será 0 e o registrador sempre permanece no estado 0000.

2.3 Análise de Componentes Independentes - ICA

Com o objetivo de estimar quantidades desconhecidas, sejam elas parâmetros, sequências de dados ou sinais, a partir de medidas conhecidas (obtidas por meio de sensores), deve-se utilizar um modelo de representação que seja capaz de expressar a relação entre as grandezas desconhecidas. Um modelo dos mais utilizados é o modelo linear, expresso por:

$$\boldsymbol{x} = \boldsymbol{A}\boldsymbol{s} \tag{2.6}$$

Onde $\boldsymbol{x}$ é o vetor das medidas realizadas pelos sensores e $\boldsymbol{A}$ é a matriz de mistura. A matriz $\boldsymbol{A}$ corresponde aos parâmetros de combinação dos sinais ou dados, sendo que os coeficientes ou pesos desta matriz são considerados constantes e desconhecidos, uma vez que não é possível determiná-los sem o conhecimento das propriedades do sistema de mistura. O vetor $\boldsymbol{s}$ é aquele que contém os sinais fonte ou dados originais [Aapo Hyvärinen 2001].

Devido a sua simplicidade e aplicabilidade, o modelo linear invariante no tempo tem sido utilizado para equacionar o problema de análise de componentes independentes.

Algumas outras considerações devem ser feitas ao modelo da equação 2.6, para a sua utilização em ICA:

- $\boldsymbol{s}$ é estacionário e de média zero;
- As componentes do vetor $\boldsymbol{s}$ são estatisticamente independentes;
- O número de sensores é maior ou igual ao número de fontes.

Assim, partindo do modelo linear (2.6) e do princípio de que as fontes originais são estatisticamente independentes, a análise de componentes independentes procura estimar um conjunto de fontes $\boldsymbol{y}$ também independentes, através de um sistema de separação $\boldsymbol{W}$ quando somente as medidas dos sensores $\boldsymbol{x}$ são conhecidas [Aapo Hyvärinen 2001]. Para isso, a restrição estatística utilizada é que a função densidade de probabilidade das fontes sejam não-gaussianas, ou que no máximo uma das fontes tenha a função densidade de probabilidade gaussiana. Esta restrição é utilizada, visto que, conforme o teorema do limite central [Athanasios and Pillai 1991], a soma de variáveis aleatórias gaussianas fornece uma distribuição de probabilidade conjunta também gaussiana, o que inviabiliza qualquer inferência sobre as fontes originais a partir das obervações dos sensores, ou seja, das misturas.

O modelo para a separação das misturas utilizando ICA pode , então, ser escrito como:

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x} \tag{2.7}$$

Na realidade, as formulações das equações (2.6) e (2.7) não são diferentes, uma vez que $\boldsymbol{W}$ pode ser considerada a inversa da matriz $\boldsymbol{A}$.

A partir da equação 2.7 o problema passa a ser a determinação do sistema ou matriz de separação $\boldsymbol{W}$, tal que as componentes de $\boldsymbol{y}$ sejam maximamente independentes.

Mais uma vez vale a pena destacar que este problema só será resolvido se, e somente se, as fontes originais forem consideradas não gaussianas, pois só assim a restrição de independência estatística é suficiente para a determinação dos coeficientes de $\boldsymbol{W}$ e das componentes $\boldsymbol{y}$. Desta forma, pode-se dizer que a não-gaussianidade das fontes originais é o ponto de partida para a elaboração de métodos para a solução do problema de ICA, que tem por objetivo encontrar componentes independentes para o caso especial em que as fontes são não-gaussianas.

2.3.1 Restrições Estatísticas

Para que se possa utilizar ICA é necessário que os sinais originais sejam estatisticamente independentes. Assim sendo, a matriz de separação $\boldsymbol{W}$ é determinada de tal forma que as componentes de $\boldsymbol{y_i}$ também sejam estatisticamente independentes. Isto significa que o valor de uma componente não fornece nenhuma informação sobre o valor das outra componentes.

Em termos estatísticos, as variáveis são mutuamente independentes se a função densidade de probabilidade conjunta de y_i puder ser fatorada no produto de suas funções densidade de probabilidade marginais. A equação (2.8) demostra essa relação para duas variáveis aleatórias x e y [Athanasios and Pillai 1991].

$$p_{x,y}(x,y) = p_x(x)p_y(y) \tag{2.8}$$

A equação (2.8) pode ser utilizada para definir a propriedade mais importante das variáveis aleatórias independentes. Dadas duas funções, g e h, sempre temos que [Hyvärinen and Oja 2000]:

$$E\{g(x)h(y)\} = E\{g(x_1)\}E\{h(y_1)\} \tag{2.9}$$

A equação (2.9) pode ser provada da seguinte maneira:

$$\begin{aligned} E\{g(x)h(y)\} &= \int\int g(x)h(y)p(x,y)dxdy \\ &= \int\int g(x)p(x)h(y)p(y)dxdy \\ &= \int g(x)p(x)dx \int h(y)p(y)dy = E\{g(x_1)\}E\{h(y_1)\} \end{aligned} \tag{2.10}$$

Uma forma ,fraca, de independência é a descorrelação. Duas variáveis aleatórias são descorrelacionadas se sua covariância é igual a zero, como definido pela equação (2.11)

$$E\{xy\} - E\{x\}E\{y\} = 0 \tag{2.11}$$

Com base nas equações (2.8) e (2.11) é possível comprovar que, se as variáveis aleatórias são estatisticamente independentes, elas são descorrelacionadas, mas a descorrelação não implica em independência. O único caso em que descorrelação implica em independência estatística ocorre quando as variáveis aleatórias são gaussianas, uma vez que estas variáveis são completamente descritas por suas estatísticas de segunda ordem [Athanasios and Pillai 1991]. Como a descorrelação não garante a independência, ela também não pode garantir a separação em componentes independentes. Assim, para garantir a separação é necessário recorrer a estatísticas de ordem superior. A utilização de estatísticas de ordem superior, por sua vez faz com que a separação , utilizando ICA, seja garantida somente para no máximo uma fonte gaussiana.

Não obstante, como a independência estatística implica em descorrelação, a maioria dos métodos de ICA restringe o problema da estimativa de fontes originais de tal forma que as componentes independentes estimadas sejam sempre descorrelacionadas. O procedimento

de descorrelação reduz o número de parâmetros livre, a serem determinados na separação, simplificando o problema.

2.3.2 Variáveis aleatórias não-gaussianas

Uma das principais restrições para a realização de ICA é que as componentes independentes sejam não gaussianas ou que no máximo uma das componentes seja gaussiana.

Considere um sistema de mistura com duas fontes gaussianas , s_1 e s_2. Assim, os sinais misturados resultantes, x_1 e x_2, são gaussianos, descorrelacionados e tem variância unitária. Como a função densidade de probabilidade de x_1 e x_2 é simétrica, ela não contem nenhuma informação sobre a matriz de mistura A. Desta forma, pode dizer que a matriz A não pode ser estimada, se mais do que uma das fontes originais for gaussiana [Hyvärinen and Oja 2000]. Isto confirma e reforça a afirmação de que a separação em componentes independentes só é possível se no máximo uma fonte original for gaussiana. Além disso, é importante reforçar a ideia de que a separação somente é possível através da utilização de estatísticas de ordem superior.

Por outro lado, para variáveis não gaussianas, a independência estatística é a principal garantia de que as fontes estimadas serão distintas e com base neste critério são definidas a maioria das estratégias para a separação cega utilizando ICA.

A função densidade de probabilidade para a distribuição gaussiana ou normal é definida pela seguinte função:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{2.12}$$

Onde μ e σ são, respectivamente a média e o desvio padrão.

Uma função densidade de probabilidade sub-gaussiana é tipicamente mais plana que a distribuição gaussiana. Um exemplo é a distribuição uniforme. Já a função densidade de probabilidade super-gaussiana tem tipicamente extremidades mais prolongadas e picos mais acentuados que os das distribuições gaussianas. Um exemplo é a distribuição Laplaciana. Sinais de voz e de música comumente apresentam distribuição super-gaussiana. A figura 2.3 compara a forma de distribuições super-gaussianas, sub-gaussiana e gaussianas.

Assim, vários métodos foram desenvolvidos baseados no princípio de que para variáveis não gaussianas a independência é suficiente para garantir que fontes estimadas serão distintas, validando desta forma a separação das fontes. Neste sentido, os métodos de ICA mais convencionais são: Maximização de não-gaussianidade, estimação de máxima verossimilhança e minimização da informação mútua [Aapo Hyvärinen 2001].

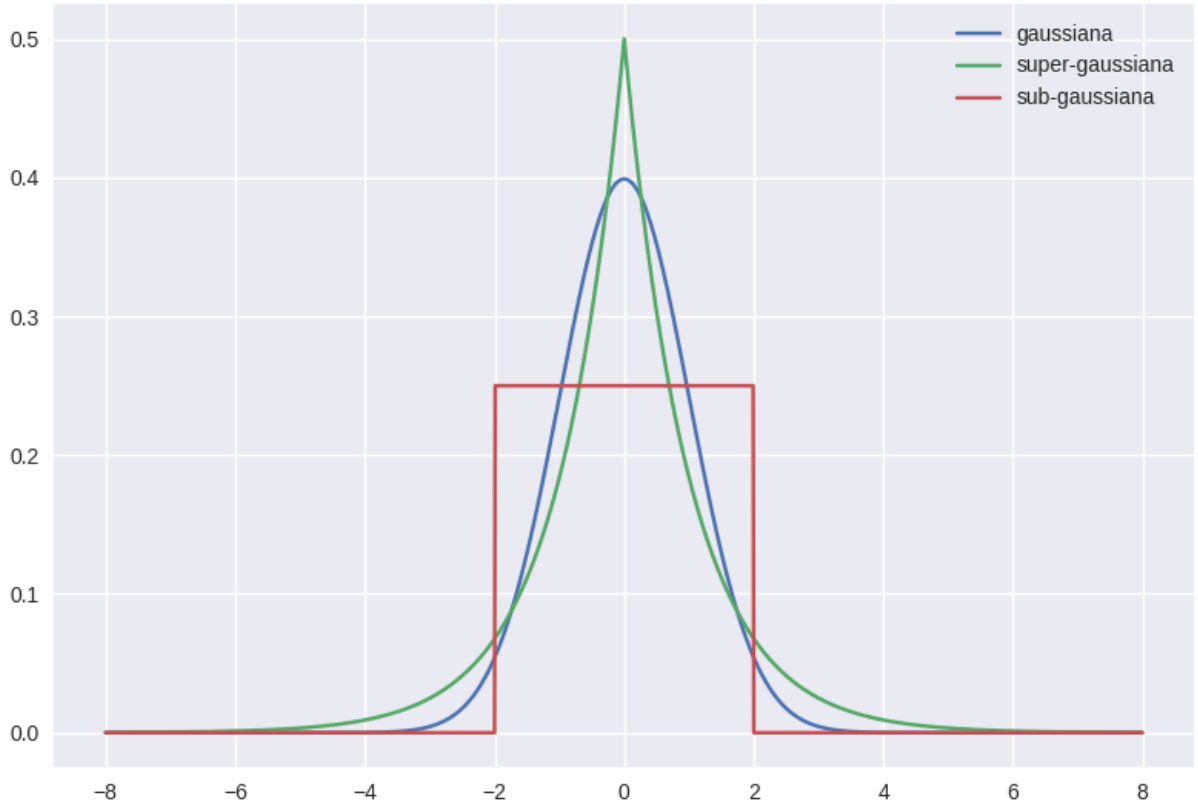

Figura 2.3: Comparação da função probabilidade de densidade de diferentes distribuições. A curva vermelha representa uma distribuição sub-gaussiana, a curva verde um distribuição super-gaussiana e a curva azul uma distribuição gaussiana.

2.3.3 Ambiguidades

Inerentes ao problema da separação cega exitem algumas ambiguidades ou indeterminações, que são mantidas pelo modelo geral de ICA da equação (2.6). As principais ambiguidades ou indeterminações são:

1. Não é possível determinar a variância das componentes independentes. A razão disso é que, como $\boldsymbol{s}$ quanto $\boldsymbol{A}$ são desconhecidos, qualquer escalar que multiplica uma das fontes pode ser cancelado dividindo a coluna $\boldsymbol{a_i}$ da matriz $\boldsymbol{A}$.

$$\boldsymbol{x} = \sum_i (\frac{\mathbf{1}}{\boldsymbol{\alpha_i}} \boldsymbol{a_i})(\boldsymbol{s_i \alpha_i}) \tag{2.13}$$

   Como consequência é possível recuperar a amplitude das componentes. Como elas são variáveis aleatórias, a maneira mais simples de fazer isso é assumindo que cada uma das componentes tem variância unitária, ou seja , $\boldsymbol{E\{s^2\}} = \mathbf{1}$. Assim, a matriz de mistura A pode ser estimada pelo método de ICA levando em conta esta restrição. Nota-se ainda, uma ambiguidade em relação ao sinal (positivo ou negativo) das fontes recuperadas.

2. Não é possível determinar a ordem das componentes independentes. A razão está no fato de que tanto os sinais originais quanto a matriz de mistura são desconhecidos. Isso pode ser demostrado através de uma matriz de permutação $\boldsymbol{P}$ e de sua inversa $\boldsymbol{P^{-1}}$, que podem ser substituídas no modelo da equação (2.6) resultando na equação (2.14), em que os elementos de $\boldsymbol{Ps}$ são variáveis independentes originais, mas uma outra ordem e a matriz $\boldsymbol{AP^{-1}}$ é a nova matriz de mistura desconhecida.

$$\boldsymbol{x = AP^{-1}Ps} \tag{2.14}$$

Assim, a separação de sinais utilizando análise de componentes independentes somente é possível sob as ambiguidades de permutação e escalonamento.

2.3.4 Algoritmo FastICA

Os primeiros algoritmos de ICA desenvolvidos apresentam problemas quanto utilizado em casos práticos, pois suas convergências são lentas. Para tentar minimizar este problema foi desenvolvida uma família de algoritmos baseados em iterações de ponto fixo denominada FastICA [Hyvarinen 1999].

Os algoritmos da família FastICA são diferenciados pela abordagem e pela função custo utilizada; todos eles visam encontrar componentes independentes através da maximização da negentropia, que é uma medida de não-gaussianidade, ou seja, distribuições gaussianas tem negentropia igual a zero.

Com relação à abordagem, existem duas versões do FastICA: uma que permite a recuperação de todas as fontes e outra que encontra uma das componentes, apenas. As duas abordagens utilizam qualquer função custo não paramétrica para estimar a negentropia das fontes [Hyvarinen 1999].

O algoritmo FastICA procura encontrar uma direção, ou seja, um vetor $\boldsymbol{w}$, cuja projeção $\boldsymbol{w^T x}$ maximiza a negentropia que pode ser aproximada na equação (2.15)

$$\boldsymbol{J_g(y) = [E\{G(y)\} - E\{G(\nu)\}]^2} \tag{2.15}$$

Onde $\boldsymbol{G}$ é uma função não quadrática e $\boldsymbol{\nu}$ é uma variável gaussiana de média zero e variância unitária. O algoritmo do FastICA pode ser simplificado da seguinte forma:

1. Escolha valores aleatórios para o vetor $\boldsymbol{w}$;
2. $\boldsymbol{w^+ = E\{xg(w^T x)\} - E\{g'(w^T x)\}w}$;
3. $\boldsymbol{w = w^+ / ||w^+||}$;
4. Repetir o passo 2 até convergir;

Note que a convergência aqui significa que o valores antigos e novos de $\boldsymbol{w}$ estão na mesma direção, ou seja, o produto interno é próximo de 1.

As esperanças $\mathbf{E}$ são estimadas através da média amostral de um número suficientemente grande de amostras dos dados de entrada para garantir a precisão.

Quando é necessário a estimativa de várias fontes, deve-se usar o algoritmo mostrado anteriormente, entretanto, aplicando-o separadamente aos membros de um conjunto de vetores $\{\boldsymbol{w_1}, \boldsymbol{w_2}, ..., \boldsymbol{w_n}\}$, tal que os membros $\boldsymbol{w_n}$ representam um vetor de pesos e $\boldsymbol{n}$ o número total de componentes que se deseja encontrar. Para evitar que resultados convirjam para o mesmo ponto, deve-se descorrelacionar as saídas $\boldsymbol{w_1^T x}$, $\boldsymbol{w_2^T x}$, ..., $\boldsymbol{w_n^T x}$, a cada iteração. Uma maneira simples de se alcançar isso usando um algoritmo iterativo foi proposta por [Hyvarinen 1999]. Sendo $\boldsymbol{W} = (\boldsymbol{w_1}, \boldsymbol{w_2}, ..., \boldsymbol{w_n})^T$

1. $\boldsymbol{W} = \boldsymbol{W}/\sqrt{||\boldsymbol{W}\boldsymbol{W}^T||}$;
2. $\boldsymbol{W} = \frac{2}{3}\boldsymbol{W} - \frac{1}{2}\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{W}$;
3. Repetir o passo 2 até convergir;

2.4 Mapas Auto-organizáveis

Na seção anterior apresentamos uma abordagem linear de análise de componentes independentes, equação 2.6, porém em muitas situações esse modelo é muito simples para descrever os sinais observados $\boldsymbol{x}$ de forma adequada. Portanto, Uma extensão natural para este modelo linear é um modelo não-linear. Para misturas invariantes no tempo, o modelo não-linear segue uma forma genérica:

$$\boldsymbol{x} = \boldsymbol{f}(\boldsymbol{s}) \tag{2.16}$$

Em que $\boldsymbol{x}$ é o vetor do sinal observado, $\boldsymbol{f}$ é uma função desconhecida de mistura e $\boldsymbol{s}$ é o vetor com as componentes independentes desconhecidas.

Assumindo que a dimensão do vetor $\boldsymbol{s}$ é igual a dimensão do vetor $\boldsymbol{x}$ por simplicidade, então, o modelo geral para ICA não-linear consiste em encontrar a função h: $\boldsymbol{R^n} \rightarrow \boldsymbol{R^n}$ que gere componentes que são estatisticamente independentes.

$$\boldsymbol{y} = \boldsymbol{h}(\boldsymbol{x}) \tag{2.17}$$

Para resolver um problema não-linear de ICA as técnicas apresentadas anteriormente não são suficientes. Para isso varias técnicas para resolver este problema foram apresentadas como: MISEP, que utiliza o principio do Infomax, e ICA por meio de função de base radial [Comon and Jutten 2010]. Uma das abordagens mais simples e mais antigas para solucionar

o problema de ICA não-linear é por meio de mapas auto-organizáveis. Este método foi originalmente proposto por [Pajunen et al. 1996].

O Mapa Auto-Organizável (*Self Organizing Map*), também chamado de rede de Kohonen ou pela abreviação no nome em inglês SOM, é uma arquitetura de rede neural artificial com aprendizado do tipo não-supervisionado [Haykin 1994], [Kohonen 2001]. A rede de Kohonen faz uma projeção não linear de dados de dimensão $\boldsymbol{d}$ em um mapa discreto de duas dimensões, mantendo a topologia dos dados de entrada to fielmente quanto possível.

Mapa auto-organizável pode ser modelado em uma rede de duas camadas, entrada (vetores de características) e a camada de saída (mapa de neurônios [Kohonen 2001]). O vetor de entrada é representado por $\boldsymbol{x_i}$ e tem $\boldsymbol{n}$ dimensões. Logo $\boldsymbol{x} = (\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x_n})$, como pode ser visto na figura 2.4. Sendo assim o conjunto de vetores $\boldsymbol{x}$ será denotado $\boldsymbol{X}$.

Um neurônio (ou unidade) $\boldsymbol{u}$ é associado a um vetor de pesos $\boldsymbol{w_u} = (\boldsymbol{w_{u1}}, \boldsymbol{w_{u2}}, ..., \boldsymbol{w_{un}})$, que assim como o vetor de entrada tem dimensão $\boldsymbol{d}$. Os vetores de peso devem ser inicializados de forma aleatória, com escolhas aleatórias no domínio da entrada [Kohonen 2001].

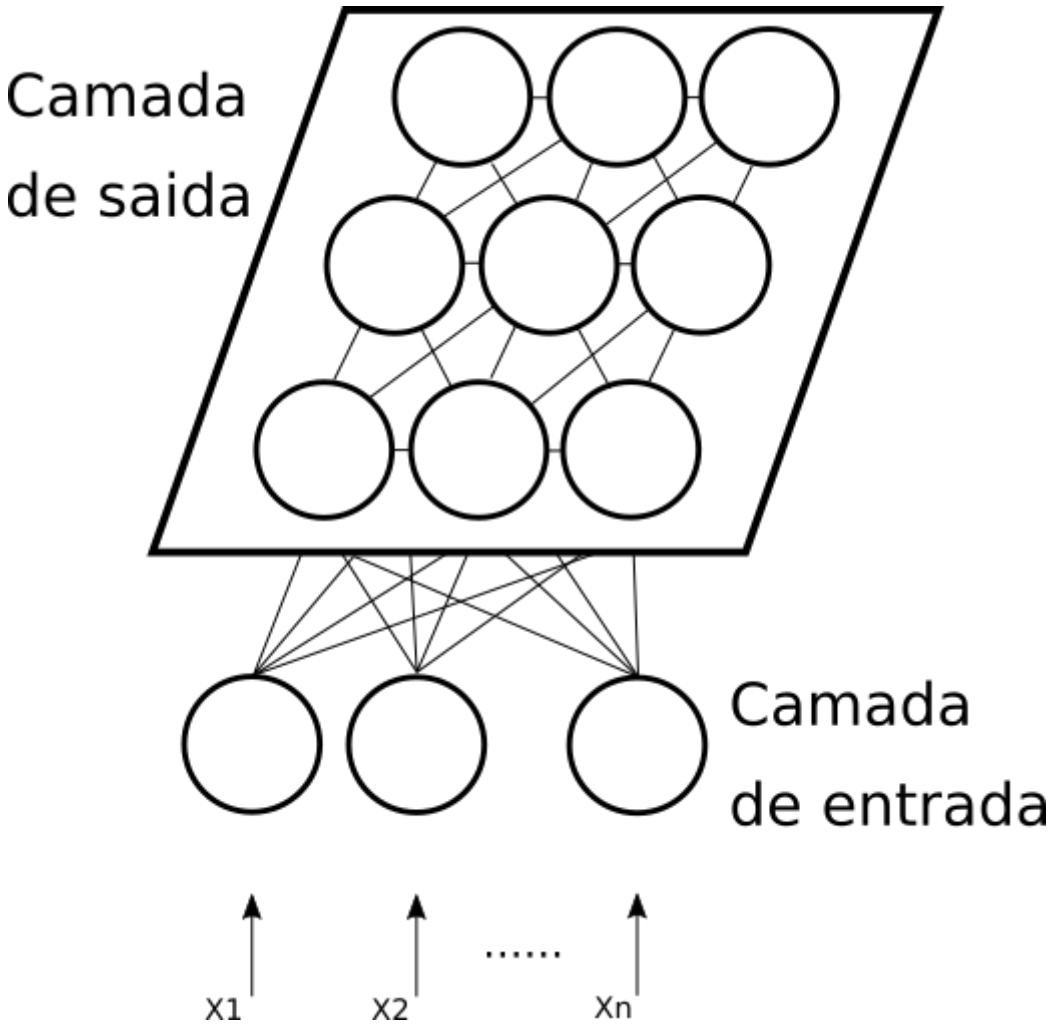

Figura 2.4: Modelo de mapas auto-organizáveis. A camada $\boldsymbol{x_i}$ representa a camada de entrada.

O treinamento do mapa auto-organizável começa com a seleção aleatória de um vetor do conjunto de treinamento, x_i. Então, o neurônio u com a menor distância entre seu vetor de pesos w_u e x_i no espaço euclidiano é, então, selecionado como o neurônio vencedor, denotado como c, de acordo com a equação 2.18.

$$c = argmin(||x_i - w_u||) \tag{2.18}$$

Em que a distância euclidiana é denotada como $||.||$

Podemos dizer que o neurônio c é o melhor representante de x_i. Para aumentar a probabilidade deste neurônio ser escolhido como vencedor novamente, caso o mesmo vetor x_i seja selecionado novamente nas iterações subsequentes do treinamento, a diferença entre o vetor de pesos do neurônio vencedor w_c e x_i é diminuída. Esta adaptação gradual do vetor de pesos é controlado pelo parâmetro de taxa de aprendizado α. Usualmente, a taxa de aprendizado diminui em função do número de iterações, assim, o vetor de pesos será adaptado mais fortemente no início do treinamento. Já no final do processo, quando α é pequeno, o mapa se ajusta de forma mais lenta, fazendo apenas o ajuste fino.

Para garantir que o mapa preserve a topologia, ou seja, preservar as relações de similaridade entre os vetores de treinamento e os neurônios do mapa, além do vetor de pesos do neurônio vencedor os vetores dos seus vizinhos também serão alterados. Define-se então uma relação de vizinhança, para que os neurônios mais próximos do neurônio vencedor c se ajustem mais que neurônios mais distantes de c. Uma das funções de vizinhança mais utilizadas é:

$$h_{ci}(t) = e^{||r_c - r_i||^2/2\sigma^2} \tag{2.19}$$

Em que h_{ci} é a força de adaptação, r_c é a coordenada do neurônio vencedor c, r_i é a coordenado de um neurônio vizinho i no mapa de saída. O parâmetro σ define o tamanho do raio da vizinhança e é um fator dependente do número de iterações.

Definida a taxa de aprendizado α e a função de vizinhança h_{ci}, o vetor de pesos w_u^+ do neurônio u é adaptado pela adição de uma parcela αh_{ci} do vetor diferença $[x_i - w_u]$ ao vetor w_u, de acordo com a equação 2.20.

$$w_u^+ = w_u + \alpha h_{ci}||x_i - w_u|| \tag{2.20}$$

Como consequência da equação 2.20, o vetor de pesos do neurônio vencedor e os pesos dos neurônios vizinhos se deslocam em direção ao vetor de treinamento x_i.

Uma forma resumida de uma iteração do algoritmo de aprendizado de um mapa auto-organizável pode ser descrito como:

1. Seleção aleatória de um vetor x_i do conjunto de treinamento;

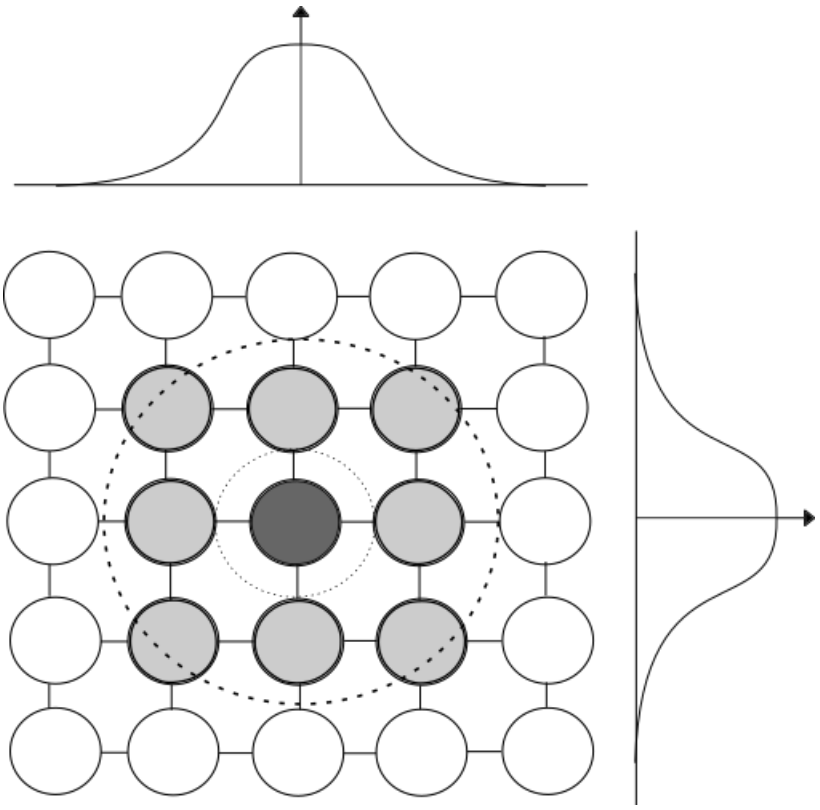

Figura 2.5: Representação da relação de vizinhança. A intensidade de adaptação dos neurônios é indicada pelos diferentes círculos em tons de cinza. O neurônio central representa o neurônio vencedor

2. Seleciona-se o neurônio vencedor

3. Adaptação do vetor de pesos do vencedor e seus vizinhos;

4. modificação da taxa de aprendizado e da faixa de vizinhança;

Ao final de cada iteração, o processo de treinamento é continuado com a seleção aleatória do próximo vetor $\boldsymbol{x_i}$, a ser considerado na nova iteração de treinamento.

Um mapa auto-organizável pode ser usado como uma técnica de ICA não-linear [Pajunen et al. 1996]. O mapa pode estimar a matriz inversa $\mathbf{W}$, da equação 2.7. Isso pode ser feito usando o vetor de sinais observados $\boldsymbol{x}$ como o vetor de entrada no mapa auto-organizável. Sendo assim, a coordenado do neurônio vencedor no mapa define a estimativa do vetor de sinais recuperados $\boldsymbol{y}$. Usando interpolação no mapa é possível fazer com que esta função fique continua[Ludwig et al. 1995].

Mesmo que o mapa auto-organizável consiga produzir vetores que sejam estatisticamente independentes entre si, essa propriedade não garante que as componentes obtidas pelo mapa são boas estimativas dos sinais fontes. Porém, com algumas restrições heurísticas, aparentemente os sinais fontes podem ser separados de forma grosseira. A densidade da mistura deve ter uma forma tal que o mapa auto-organizável consiga se adaptar a ela. Além disso essa adaptação natural deve resultar em uma separação correta. Uma classe de problema ra-

zoável em que essa descrição é valida consiste em fontes sub-gaussianas que primeiramente são misturadas de forma linear e posteriormente sofrem uma pequena distorção não-linear. Tipicamente, sinais com distribuições sub-gaussianas possui a densidade de probabilidade mais achatada em comparação com distribuições gaussianas [Pajunen et al. 1996].

Devido a forma mais plana de uma distribuição sub-gaussiana uma mistura linear de fontes sub-gaussianas produzem, em geral uma densidade próxima de um retângulo. Pequenas não-linearidades não distorcem tanto essa densidade com forma retangular, permitindo a adaptação correta do mapa auto-organizável.

Para demostrar tal técnica podemos utilizar duas fontes de sinais sub-gaussianos : um sinal senoidal e um sinal de ruido uniformemente distribuído. Usando a matriz de mistura $\boldsymbol{A}$:

$$\boldsymbol{A} = \begin{bmatrix} \mathbf{0.7} & \mathbf{0.3} \\ \mathbf{0.3} & \mathbf{0.7} \end{bmatrix} \tag{2.21}$$

Apos a mistura linear os sinais misturados, $\boldsymbol{v_k} = \boldsymbol{A s_k}$ foram distorcidos por uma função não-linear $\boldsymbol{f}$ aplicada separadamente em cada componente. resultando no vetor de mistura observado:

$$\boldsymbol{x_k} = [\boldsymbol{f}(\boldsymbol{v_k^1}), \boldsymbol{f}(\boldsymbol{v_k^2})]^{\boldsymbol{T}} \tag{2.22}$$

A função não-linear $\boldsymbol{f}$ utilizada neste exemplo foi $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{x^3} + \boldsymbol{x}$.

Como pode ser visto na figura 2.6 a técnica conseguiu preservar a forma dos sinais originais, porém, com um ruido atrelado ao resultado.

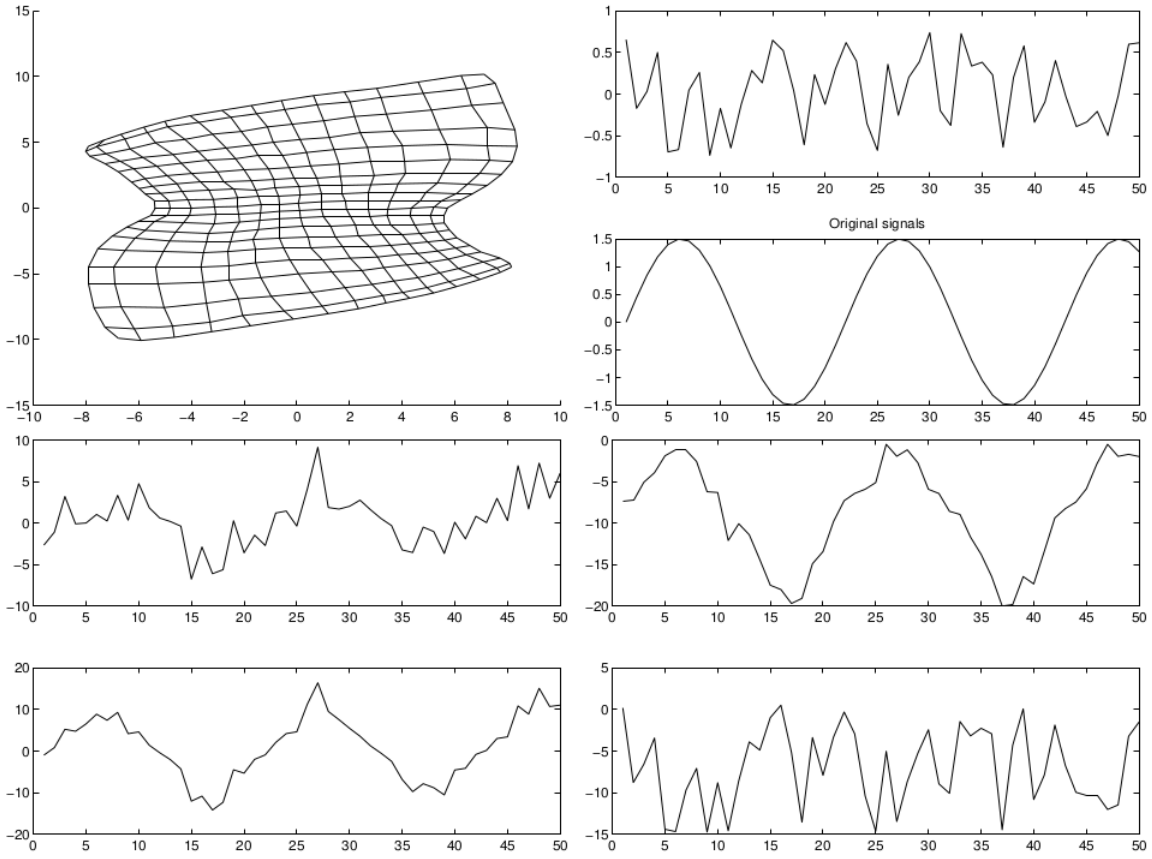

Figura 2.6: Canto superior esquerdo: Mapa auto-organizável; Canto superior direito: Fontes originais; Canto inferior esquerdo: sinais observados; Canto inferior direito: Sinais recuperados. [Pajunen et al. 1996]

Capítulo 3

Metodologia

A metodologia deste estudo foi dividida em duas partes: ICA linear e ICA não-linear. Para produzir resultados o mais confiáveis possíveis, ou seja, obter um erro relativo na ordem de grandeza de $\mathbf{0,01\%}$ escolhemos um grande número de amostras e uma quantidade razoável de paralelização de Monte Carlo. Com esse propósito serão feitas $\mathbf{20}$ simulações com $\mathbf{1.000.000}$ pontos cada. Estes pontos serão amostrados a partir de dois geradores de números aleatórios: Mersenne Twister e LFSR 16-bits.

A amostragem dos pontos necessários para o método de Monte Carlo é feita da seguinte maneira: como cada ponto na simulação é um par ordenado $(\boldsymbol{x_i}, \boldsymbol{y_i})$, o primeiro passo é criar um vetor $\boldsymbol{x}$ de tamanho $\mathbf{1.000.000}$ e, em seguida substituir cada valor $\boldsymbol{x_i}$ por um valor gerado pelo gerador pseudo-aleatório. Repetindo esse processo para componente $\boldsymbol{y}$ geramos uma amostra de pares ordenados necessários para uma simulação de Monte Carlo. Esse processo todo é feito para gerar uma amostra, então, repetimos ele até chegar a $\mathbf{20}$ amostras.

Com o conjunto de amostras pronto, aplicamos ICA e como resultado geramos um outro conjunto de amostras. Esses dois conjuntos serão utilizados no método de Monte Carlo a ser proposto e por fim iremos comparar o erro obtido a partir destes dois conjuntos. Com isso esperamos verificar se a aplicação de ICA nas amostras melhora a convergência de Monte Carlo paralelo.

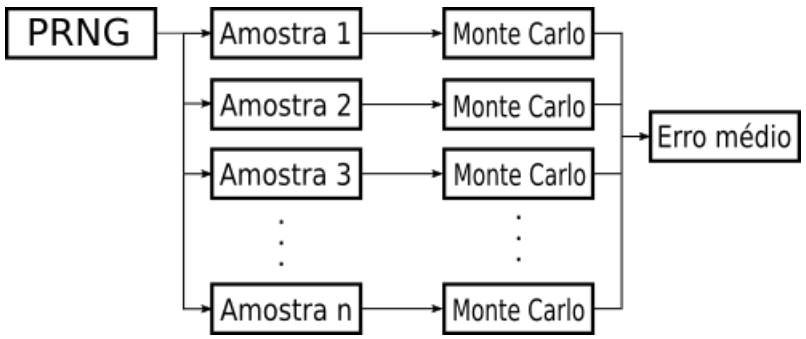

Figura 3.1: Diagrama de blocos do problema de Monte Carlo sem o uso de ICA.

O diagrama de blocos representado na figura 3.1 mostra de forma simplificada como

foi feita a primeira parte do estudo. Primeiramente amostramos um numero n de amostras pseudo-aleatórias que em seguida alimenta n ensaios de Monte Carlo. Assim obtemos o erro médio das n simulações. No caso, o valor n neste estudo é 20 como já foi citado. Em seguida realizamos o ensaio usando ICA como mostra o diagrama de blocos da figura 3.2.

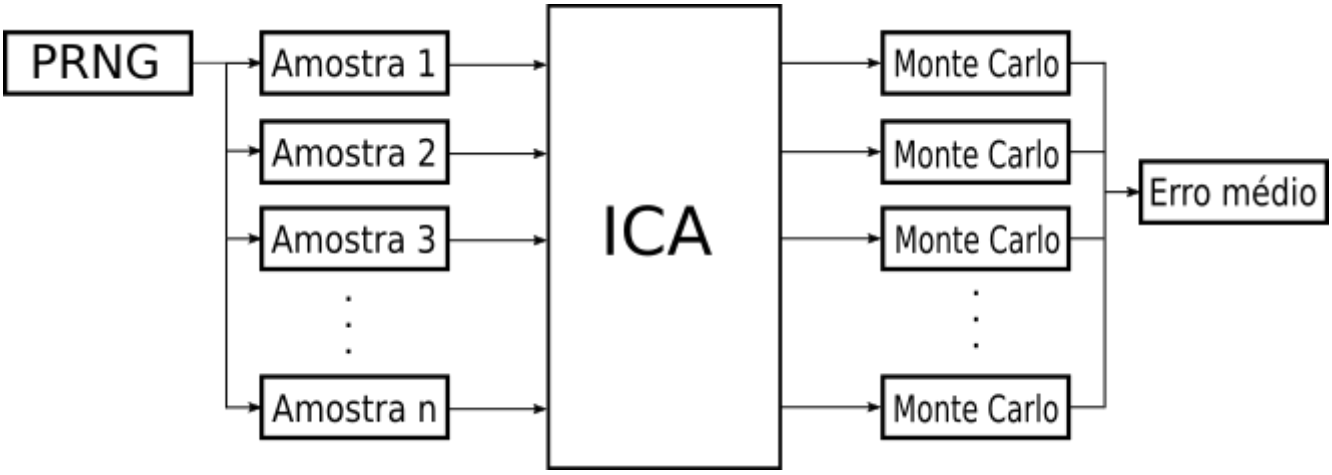

Figura 3.2: Diagrama de blocos do problema de Monte Carlo com o uso de ICA.

Nesta segunda etapa apos a amostragem a partir do gerador pseudo-aleatório aplicamos ICA nas n amostras obtidas. A análise de componentes independentes ira gerar como resultado outras n amostras que então alimentaram as realizações da simulação de Monte Carlo. Por fim obtemos o erro médio e comparamos com o valor obtido da etapa anterior.

3.1 Parte 1 - ICA Linear

Para demonstrar se o algoritmo FastICA, que é uma técnica linear de ICA, é capaz de tornar amostras obtidas por um gerador de números pseudo-aleatórios mais estatisticamente independentes, propomos uma simulação de Monte Carlo para então comparar a convergência da simulação antes e depois da aplicação do algoritmo nas amostras utilizadas por Monte Carlo. Ou seja, queremos verificar se o erro entre o valor analítico do problema proposto e o resultado da simulação de Monte Carlo tende a zero com uma amostra menor quando aplicado a técnica de ICA.

Com este objetivo, analisamos a simulação de Monte Carlo da seguinte curva: $y^2+x^2 = 1$, definida para $0 \leq x \leq 1$ e $0 \leq y \leq 1$, como apresentado na figura 3.3. Esta curva foi escolhida pela a simplicidade, e o conhecimento do seu valor analítico da área sob a curva que é $\pi/4$.

Para obtermos os valores numéricos a serem comparados, obtivemos um conjunto de números pseudo-aleatórios com um número N de amostras. Em seguida avaliamos o módulo do par ordenado (x_i, y_i). Caso o módulo seja menor que 1 o ponto se encontra abaixo da curva e então $p_i = 1$. Caso o módulo for maior que 1 então o ponto se encontra a cima da

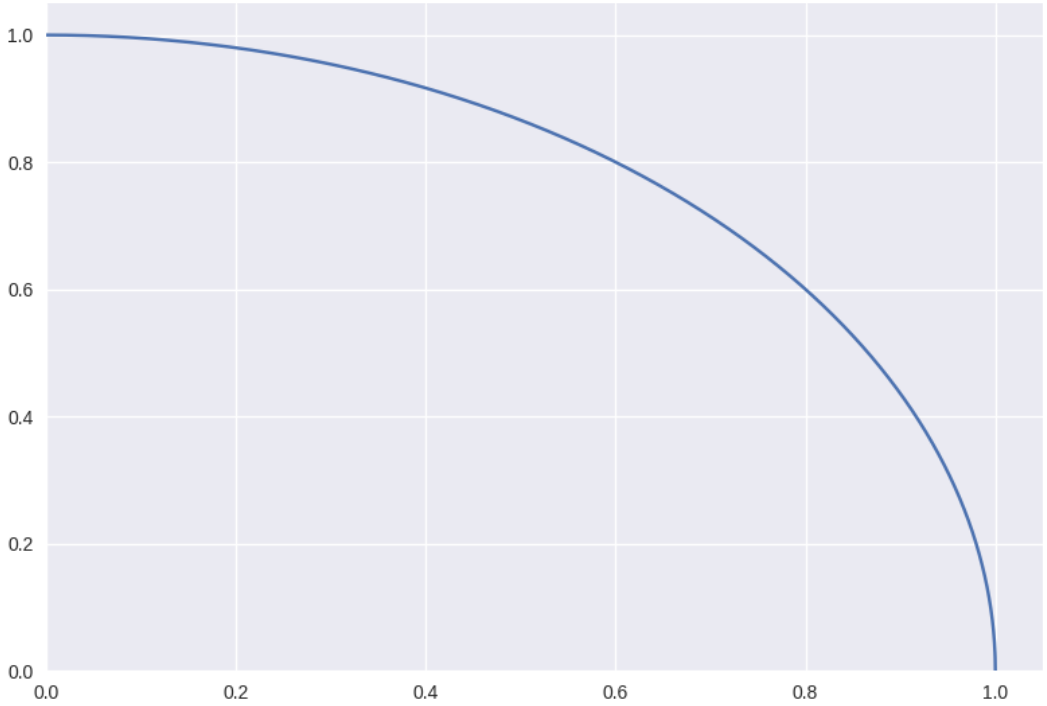

Figura 3.3: Curva representada pela função $y^2 + x^2 = 1$ no intervalo $0 \leq x \leq 1$ e $0 \leq y \leq 1$. Esta curva será usada como modelo

curva então $p_i = 0$. Ao fazer a seguinte operação:

$$A = \frac{1}{N} \sum_{i=1}^{N} p_i \tag{3.1}$$

obtivemos o valor numérico para a área sobre a curva estudada.

Assim comparando o valor analítico e o valor experimental traçamos a curva de erro. Também para fins de comparação traçamos o valor de da área multiplicada por 4. Este valor converge para o número π.

Os geradores de números pseudo-aleatórios utilizados neste estudo foram dois : Mersenenne Twister e Linear-shift Feedback Register 16-bits (LFSR). o gerador o Mersenne Twister foi escolhido por ser o gerador de números pseudo-aleatórios com período muito grande, $(2^{19937} - 1)$, e ser o mais usado. Já o gerador LFSR foi escolhido pelo o fato do período for ser, $(2^{16} - 1)$, isso faz com que as amostras obtidas por esse gerador tenham padrões evidentes. Por esse motivo, esperamos que o algoritmo FastICA obtenha melhores resultados nas amostras geradas pelo o gerador LFSR. A comparação entre duas amostras, uma de cada gerador, é apresentada na figura 3.4. Um exemplo de uma simulação feita com números gerados por um gerador LFSR pode ser visto na figura 3.5. Esta simulação foi feita com 500.000 pontos.

Agora aplicamos o algoritmo FastICA e repetimos a simulação Monte Carlo feita anteriormente e, então, comparamos o erro antes e depois de aplicar a técnica de ICA.

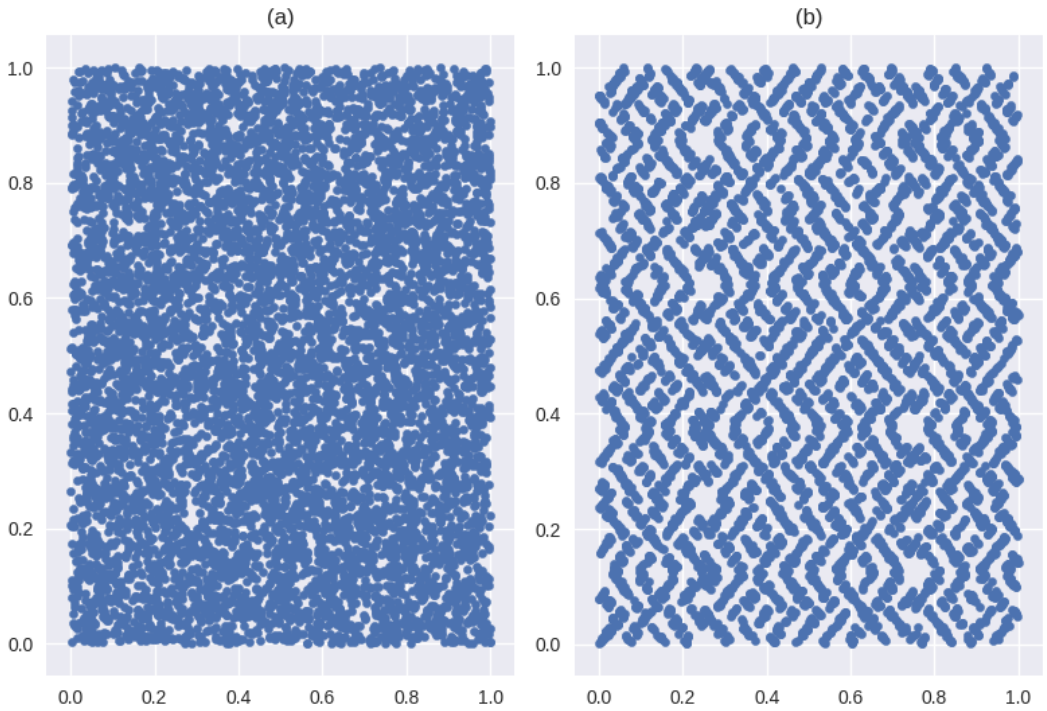

Figura 3.4: Comparação entre os amostras dos geradores de números pseudo-aleatórios: (a) Mersene Twsiter e (b) LSFR. Ambas as amostras tem **8000** pontos

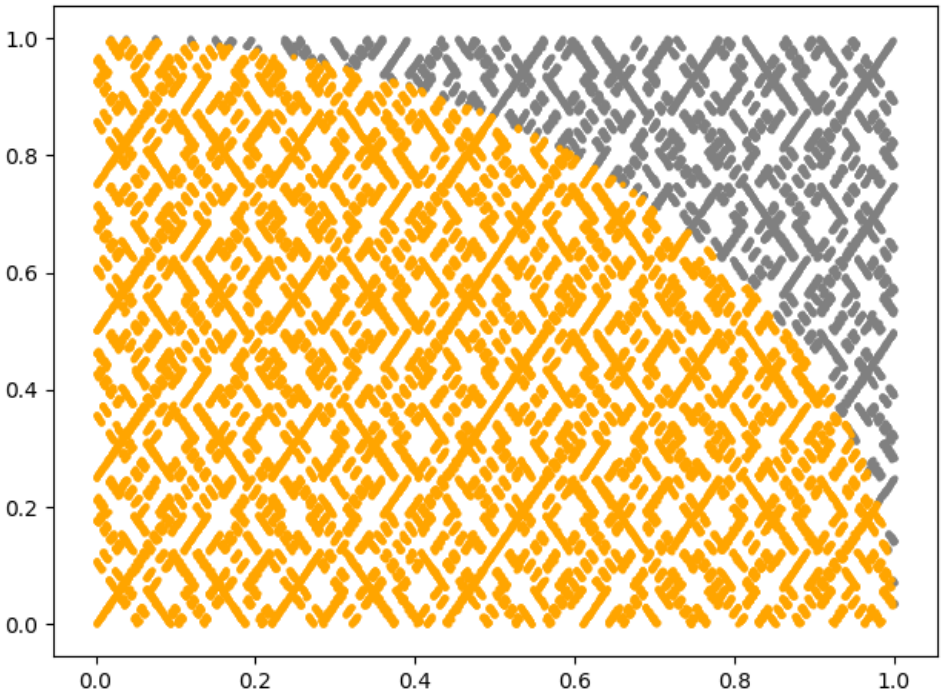

Figura 3.5: Simulação de Monte Carlo usando o gerador LSFR com 500000 pontos. Os pontos em laranja estão abaixo da curva estudada

3.2 Parte 2 - ICA Não-Linear

A segunda etapa do estudo é realizar o mesmo estudo que foi feito anteriormente porém ao invés de usar um algorítimo linear de ICA usamos a técnica de mapas auto-organizáveis. Este procedimento é uma técnica não-linear que utiliza uma rede neural para obtermos amostras estiticamente independentes. Como a aplicação de tal rede neural foi utilizado a implementação, escrita na linguagem de programação python, MiniSOM (`https://github.com/JustGlowing/minisom`) com algumas alterações.

O primeiro passo na utilização de mapas auto-organizáveis é determinar o número de neurônios que será utilizado na rede neural. Neste estudo utilizamos uma rede $\mathbf{40x40}$. O segundo passo foi determinar os parâmetros $\boldsymbol{\sigma}$ e a taxa de aprendizado $\boldsymbol{\alpha}$. Estes parâmetros foram determinados de forma experimental e foram: $\boldsymbol{\sigma} = \mathbf{2}$ e $\boldsymbol{\alpha} = \mathbf{0,0005}$. Com os parâmetros definidos a rede neural foi treinada usando o uma amostra de treinamento com $\mathbf{50000}$ pontos. Na figura 3.6 uma rede neural 10x10 foi treinada em uma mistura super-gaussiana com média 0 como formar de validar o a implementação de mapa auto-organizável MiniSOM.

Os sinais originais seguem a forma de uma distribuição "t de Student"com grau de liberdade pequeno (quanto o maior o grau de liberdade desta distribuição mais ela se aproxima de uma distribuição gaussiana). Usando a matriz de mistura $\boldsymbol{A}$ obtemos o resultado do quadro (a) da figura 3.6:

$$\boldsymbol{A} = \begin{bmatrix} \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{2} \end{bmatrix} \tag{3.2}$$

Como pode ser visto, os neurônios se adéquam à topologia da amostra. Este resultado serve para ilustrar o funcionamento da implementação MiniSOM

Após a rede neural ser treinada, usamos como entrada do mapa os mesmos conjuntos de amostras que foram utilizados na primeira parte deste estudo. Então, para cada ponto das amostras, é calculado o neurônio vencedor e as coordenadas deste neurônio são a saída do método. É feita então novamente a simulação de Monte Carlo para compararmos os resultados.

Para minimizar os erros de quantização devido ao número finito de neurônios (e suas respectivas coordenadas), aplicamos uma técnica de interpolação para mapas auto-organizáveis. Para cada dimensão do mapa o neurônio vencedor possui dois vizinhos. O vetor de distância entre o vetor de entrada $\boldsymbol{X}$ e os pesos do neurônio vencedor $\boldsymbol{W}_{w}^{(in)}$ é projetado sobre o vetor de cada neurônio vizinho ($\boldsymbol{W}_{d,l}^{(in)}$ e $\boldsymbol{W}_{d,r}^{(in)}$). Então $\boldsymbol{\alpha}_l$ e $\boldsymbol{\alpha}_r$ são calculados como a razão

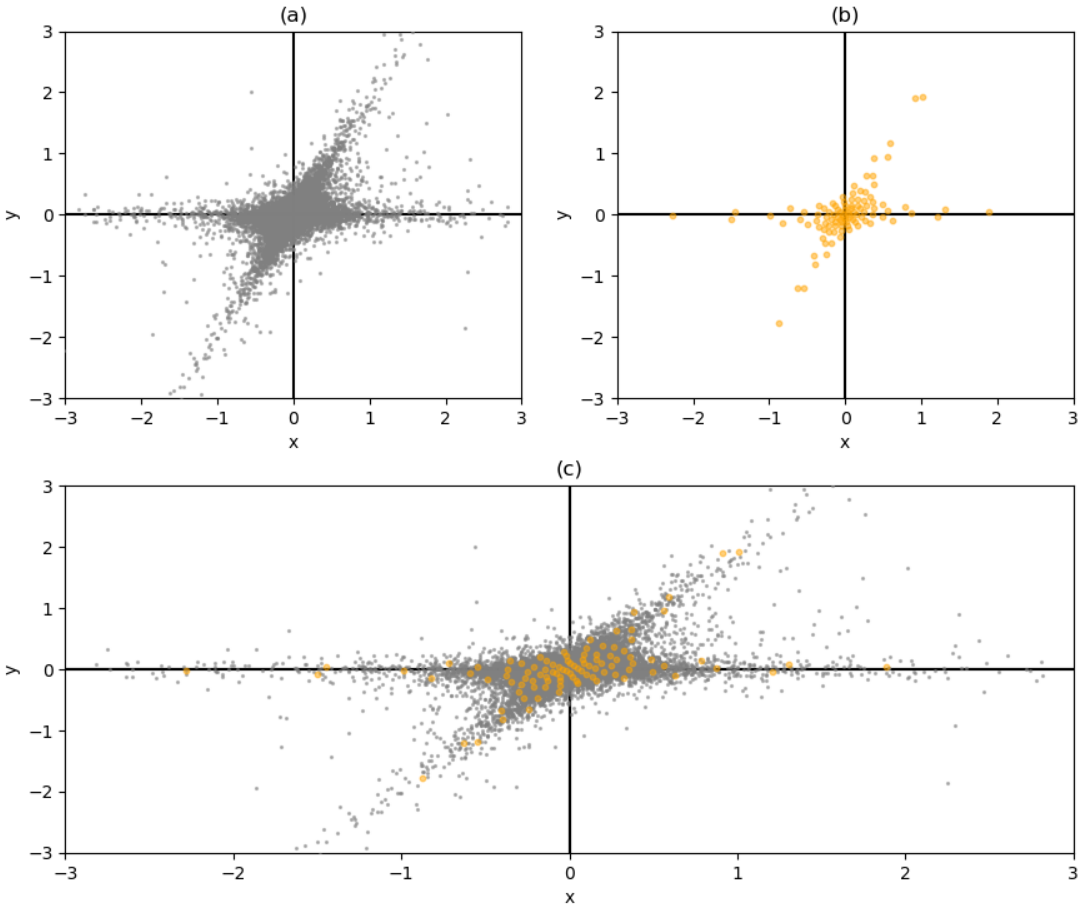

Figura 3.6: Validação da implementação de mapas auto-organizáveis. O gráfico (a) representa uma mistura de duas componentes com distribuição super-gaussiana. O gráfico (b) mostra os pesos do mapa auto-organizável após o treinamento. O gráfico (c) é a sobreposição do gráfico (a) e (b) demonstrando como a rede de neurônios se adequou a topologia da amostra.

entre a distância do neurônio vencedor e seu vizinho [Ludwig et al. 1995].

$$\boldsymbol{\alpha_{d,l}} = \frac{(\boldsymbol{X} - \boldsymbol{W_w^{(in)}})^T(\boldsymbol{W_{d,l}^{(in)}} - \boldsymbol{W_w^{(in)}})}{(\boldsymbol{W_{d,l}^{(in)}} - \boldsymbol{W_w^{(in)}})^T(\boldsymbol{W_{d,l}^{(in)}} - \boldsymbol{W_w^{(in)}})} \tag{3.3}$$

$$\boldsymbol{\alpha_{d,r}} = \frac{(\boldsymbol{X} - \boldsymbol{W_w^{(in)}})^T(\boldsymbol{W_{d,r}^{(in)}} - \boldsymbol{W_w^{(in)}})}{(\boldsymbol{W_{d,r}^{(in)}} - \boldsymbol{W_w^{(in)}})^T(\boldsymbol{W_{d,r}^{(in)}} - \boldsymbol{W_w^{(in)}})} \tag{3.4}$$

Assim, o resultado da interpolação é:

$$\boldsymbol{Y} = \boldsymbol{W_w^{(out)}} + \sum_{\boldsymbol{d=1}}^{\boldsymbol{dimenses}} \frac{\boldsymbol{\alpha_{d,l}}(\boldsymbol{W_{d,l}^{(out)}} - \boldsymbol{W_w^{(out)}}) + \boldsymbol{\alpha_{d,r}}(\boldsymbol{W_{d,r}^{(out)}} - \boldsymbol{W_w^{(out)}})}{\boldsymbol{k_d}} \tag{3.5}$$

Em que, $\boldsymbol{W_w^{(out)}}$ são as coordenadas do neurônio vencedor, $\boldsymbol{W_{d,l}^{(out)}}$ e $\boldsymbol{W_{d,r}^{(out)}}$ são as coor-

denadas de cada neurônio vizinho e k_d é o número de vizinhos na dimensão d (Nas margens $k_d = 1$).

Então, por fim usamos o resultado da interpolação do mapa auto-organizável para, mais uma vez, fazer a simulação de Monte Carlo do problema proposto e verificar se a convergência melhorou ou não.

Capítulo 4

Resultados

4.1 FastICA

Como foi explicado no capitulo 1 o foco do trabalho foi em comparar a convergência da simulação de Monte Carlo do problema proposto usando amostras antes e depois da aplicação de uma técnica de ICA. Todos os resultados aqui obtidos podem ser reproduzidos usando os códigos hospedado no seguinte endereço: `https://github.com/heitorcampos/ica_pseudo_rng`.

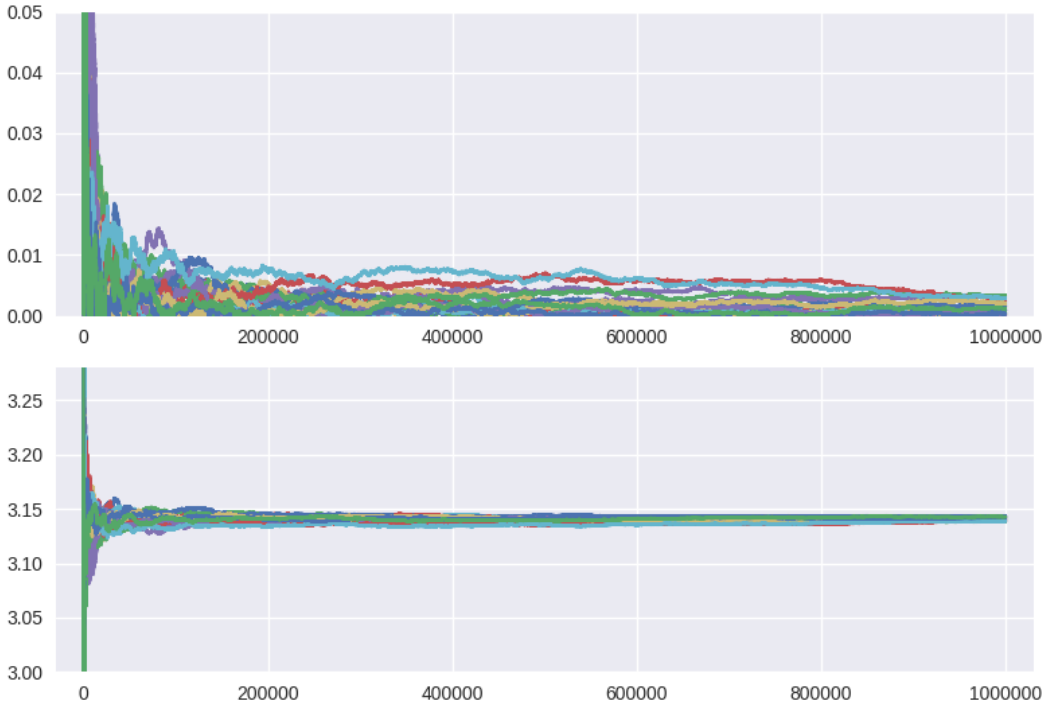

Figura 4.1: Resultado da simulação de Monte Carlo proposta utilizando o gerador Mersenne Twister. Cada cor representa uma simulação distinta com 1000000 pontos. O gráfico superior representa o valor do erro em função do número de mostras. O gráfico inferior representa o valor de π calculado numericamente a partir do resultado da simulação.

Os resultados a seguir, nas figuras 4.1 e 4.2, mostram as curvas de erro obtidas usando amostras dos dois tipos de geradores de números pseudo-aleatórios : Mersenne Twister e LFSR 16-bits. Foram feitas 20 simulações com um milhão de pontos cada, para cada tipo de gerador. Os valores médios de erro foram de $\mathbf{0,0009}$ e $\mathbf{0,0005}$ para as amostras Mersenne e LFSR respectivamente.

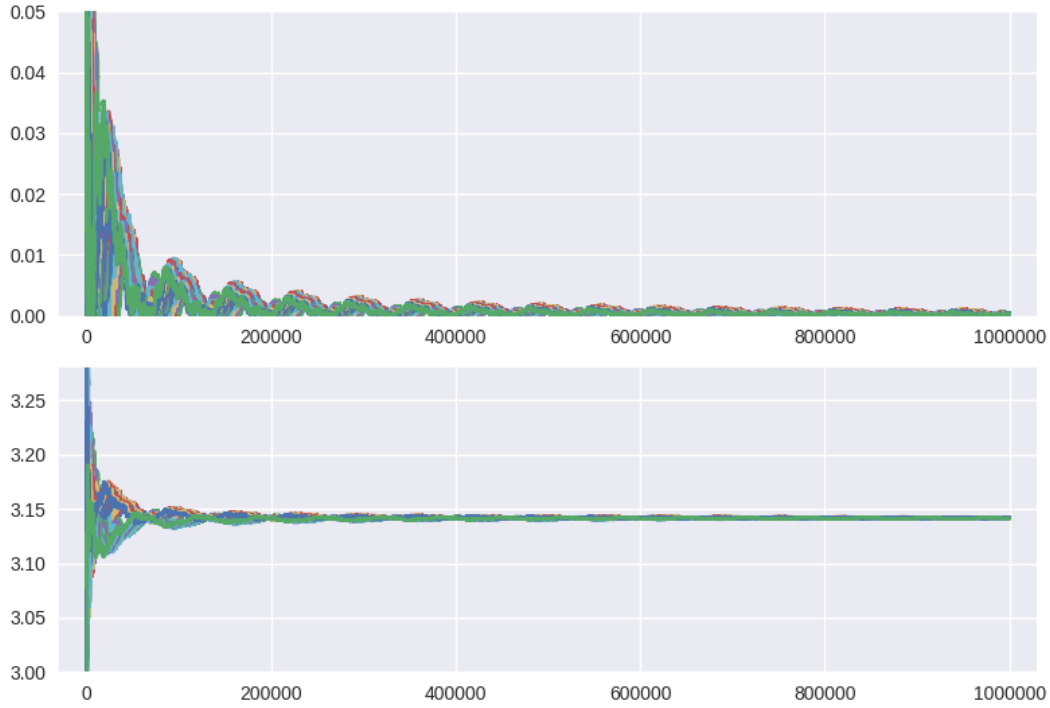

Figura 4.2: Resultado da simulação de Monte Carlo proposta utilizando o gerador LFSR 16-bits. Cada cor representa uma simulação distinta com $\mathbf{1000000}$ pontos. O gráfico superior representa o valor do erro em função do número de mostras. O gráfico inferior representa o valor de π calculado numericamente a partir do resultado da simulação.

É possível observar que a convergência do erro foi melhor com o gerador LFSR 16-bits. Ao comparar os histogramas em uma dimensão de ambas amostras observamos que o gerador LFSR 16-bits gerou uma amostra mais uniforme que o Mersenne Twister. Isso se deve justamente o fato deste gerador possuir um período muito menor do que o número de pontos amostrados. Como esses geradores de números pseudo-aleatórios podem ser vistos como uma maquina de estados, um número só pode ser amostrado novamente quando todos os outros números dentro do período também forem. Pensando então em uma amostra do tamanho do período do gerador é fácil perceber que todos os números tiveram a mesma frequência e portanto a mesma probabilidade de serem amostrados. Quando o próximo número for amostrado este terá a frequência duas vezes maior que o restante dos números da amostra o que causa uma distorção na uniformidade da amostra até que o número amostrado chegue a um múltiplo do período. Quando a quantidade de números amostrados é varias ordens de grandeza maior que o período este problema é minimizado. Por isso a amostra obtida a partir do gerador LFSR 16-bits foi mais uniforme quando comparada com a amostra gerada pelo Mersenne Twister.

Como já citado no capitulo 2 o problema aqui estudado é muito sensível a uniformidade das amostras [Dongarra et al. 2003], portanto o fato de o gerador LFSR 16-bits ter obtido resultados melhores é valido, mesmo tendo período baixo.

Outro fato interessante de se observar é como as curvas de erro do gerador LFSR 16-bits convergem em um padrão periódico. Este período observado se aproxima muito ao período do gerador, que é de 65.535.

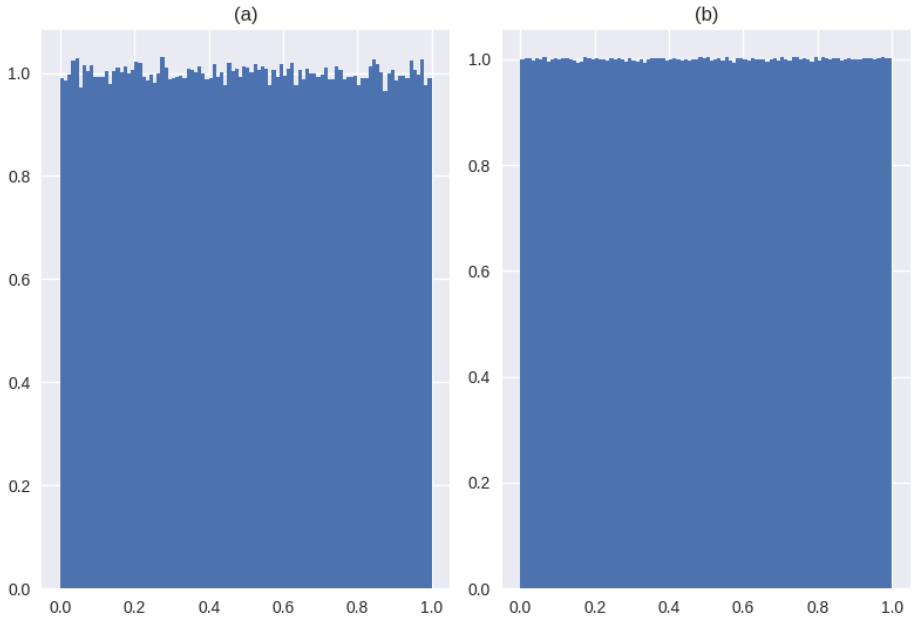

Figura 4.3: Histograma comparando amostras dos geradores (a) Mersenne Twister e (b) LFSR 16-bits

Agora aplicando o algoritmo FastICA nas amostras e em seguida traçando as curvas de erro novamente obtemos os resultados nas figuras 4.4 e 4.5. O erro obtido foi de $\mathbf{0,0012}$ e $\mathbf{0,0009}$ para o gerador Mersenne Twister e LFSR 16-bits respectivamente.

Analisando as imagens 4.4 e 4.5 e comparando com os resultados obtidos anteriormente nas figuras 4.1 e 4.2 observamos que o uso do FastICA não melhorou a convergência do problema, pelo o contrario, piorou marginalmente. Para entender a razão de tal fato analisamos o gráfico de disperção das amostras antes e depois do uso do algoritmo nas figuras 4.6. Como o conjunto de pontos amostrados pelo o Mersenne Twister já é muito homogêneo o resultado da aplicação do FastICA não mostra resultados aparentes. Porém o efeito do algoritmo é perceptível quando aplicado a amostra LFSR. Ao aplicar o FastICA a amostra foi apenas rotacionada, isto é, os pontos $(\boldsymbol{x_i}, \boldsymbol{y_i})$ se tornaram $(\boldsymbol{y_i}, \boldsymbol{x_i})$. Esse fenômeno é justamente uma das ambiguidades das técnicas de ICA.

Porem somente tal rotação não causaria com que a convergência diminuísse. Para entender melhor a razão de tal piora foram feitos os histogramas das amostras após o FastICA. Analisando a figuras 4.7 percebemos que o algoritmo alterou a uniformidade das amostras perto na margens. Como já foi dito o problema aqui estudado é muito sensível a uniformidade das amostras utilizadas e por isso o FastICA fez com a a convergência diminuísse.

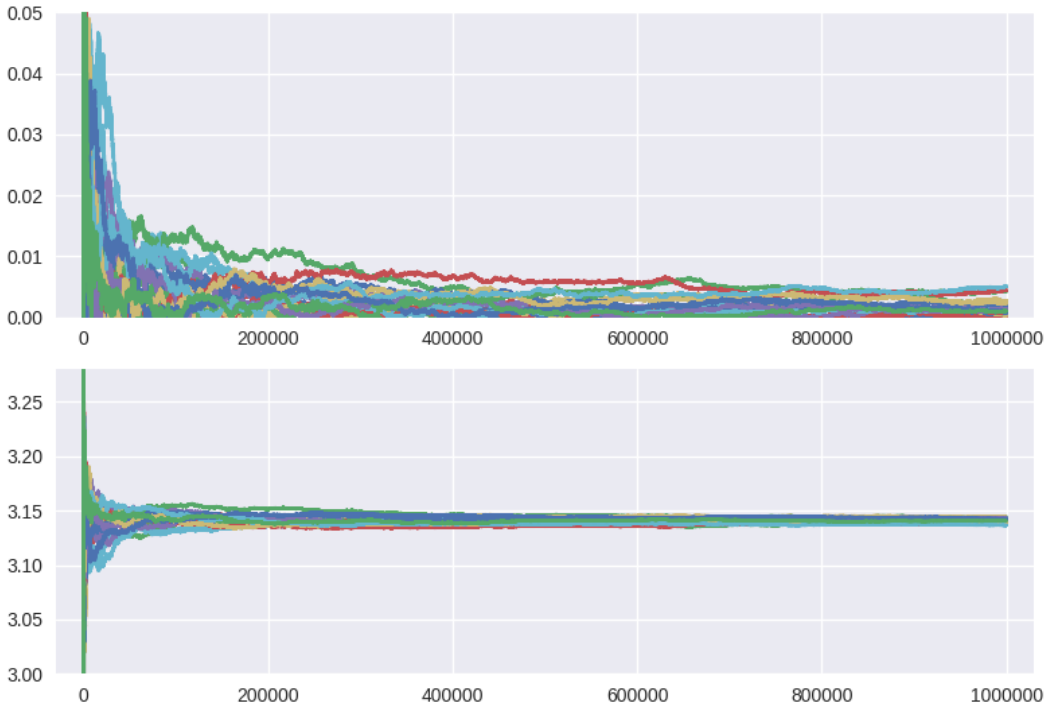

Figura 4.4: Resultado da simulação de Monte Carlo proposta utilizando amostras do gerador Mersenne Twister e aplicando o algoritmo FastICA. Cada cor representa uma simulação distinta com $\mathbf{1000000}$ pontos. O gráfico superior representa o valor do erro em função do número de mostras. O gráfico inferior representa o valor de π calculado numericamente a partir do resultado da simulação

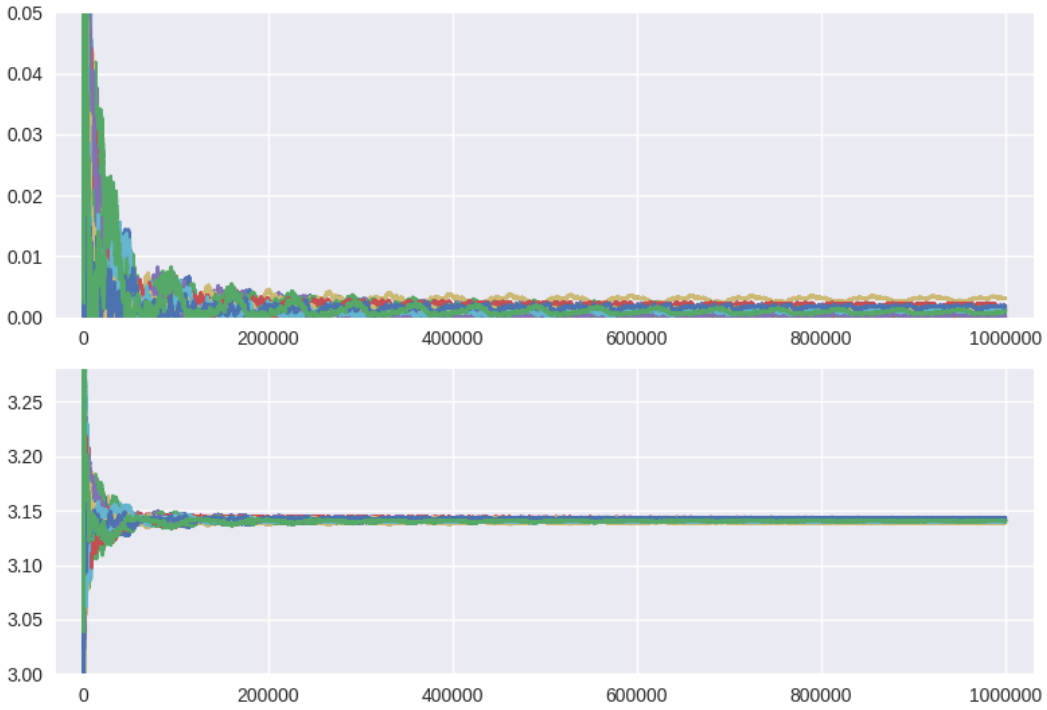

Figura 4.5: Resultado da simulação de Monte Carlo proposta utilizando amostras do gerador LFSR e aplicando o algoritmo FastICA. Cada cor representa uma simulação distinta com $\mathbf{1000000}$ pontos. O gráfico superior representa o valor do erro em função do número de mostras. O gráfico inferior representa o valor de π calculado numericamente a partir do resultado da simulação

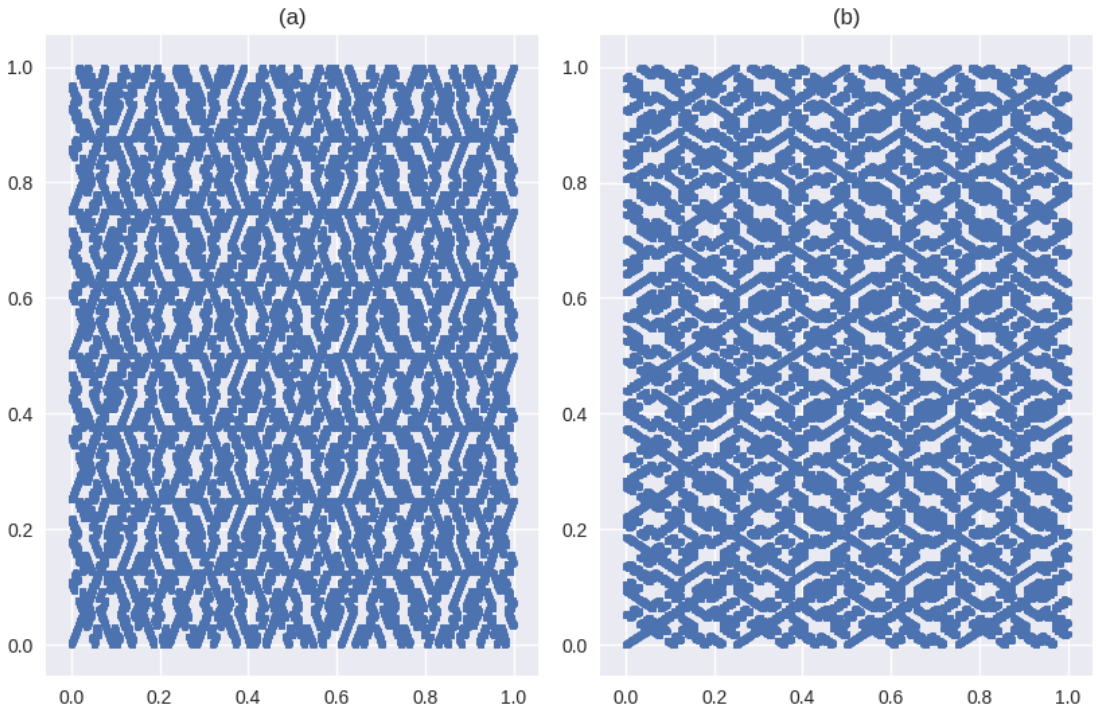

Figura 4.6: Gráfico de dispersão (a) mostra um conjunto com **500000** pontos obtidas a partir do gerador LFSR 16-bits. O gráfico (b) mostra a mesmo conjunto do gráfico (a) após a aplicação do FastICA.

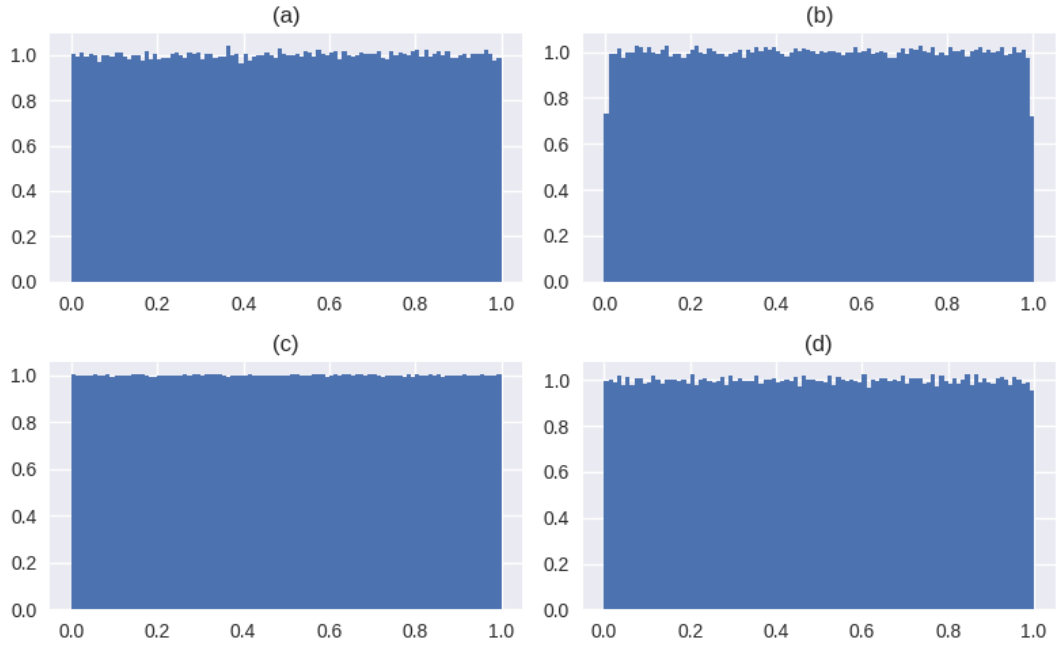

Figura 4.7: Comparação dos histogramas das amostras originais e após a aplicação do FastICA. (a) Amostra original (Mersenne Twister); (b) Amostra após o FastICA (Mersenne Twister); (c) Amostra original (LFSR); (d) Amostra após o FastICA (LFSR)

4.2 Mapas auto-organizáveis

Assim como na secção anterior o primeiro passo foi conferir a convergência, do problema de Monte Carlo proposto, antes da aplicação do método de ICA. Como já foi mostrado nas figuras 4.1 e 4.2. O gerador LFSR 16-bits obteve melhor resultado com erro de $\mathbf{0,0005}$ enquanto o gerador Mersenne Twister obteve erro de $\mathbf{0,0009}$ com 20 curvas e 1000000 pontos. Definimos então uma rede neural com 1600 neurônios dispostos em uma malha 40x40. Em seguida utilizamos um amostra de treinamento com $\mathbf{50.000}$ pontos para treinar a rede neural utilizada. A figura 4.8 mostra como o mapa com 1600 pontos , ou seja 40x40, evoluiu com a variação dos números de iterações de treinamento enquanto a figura 4.9 mostra o resultado final do mapa.

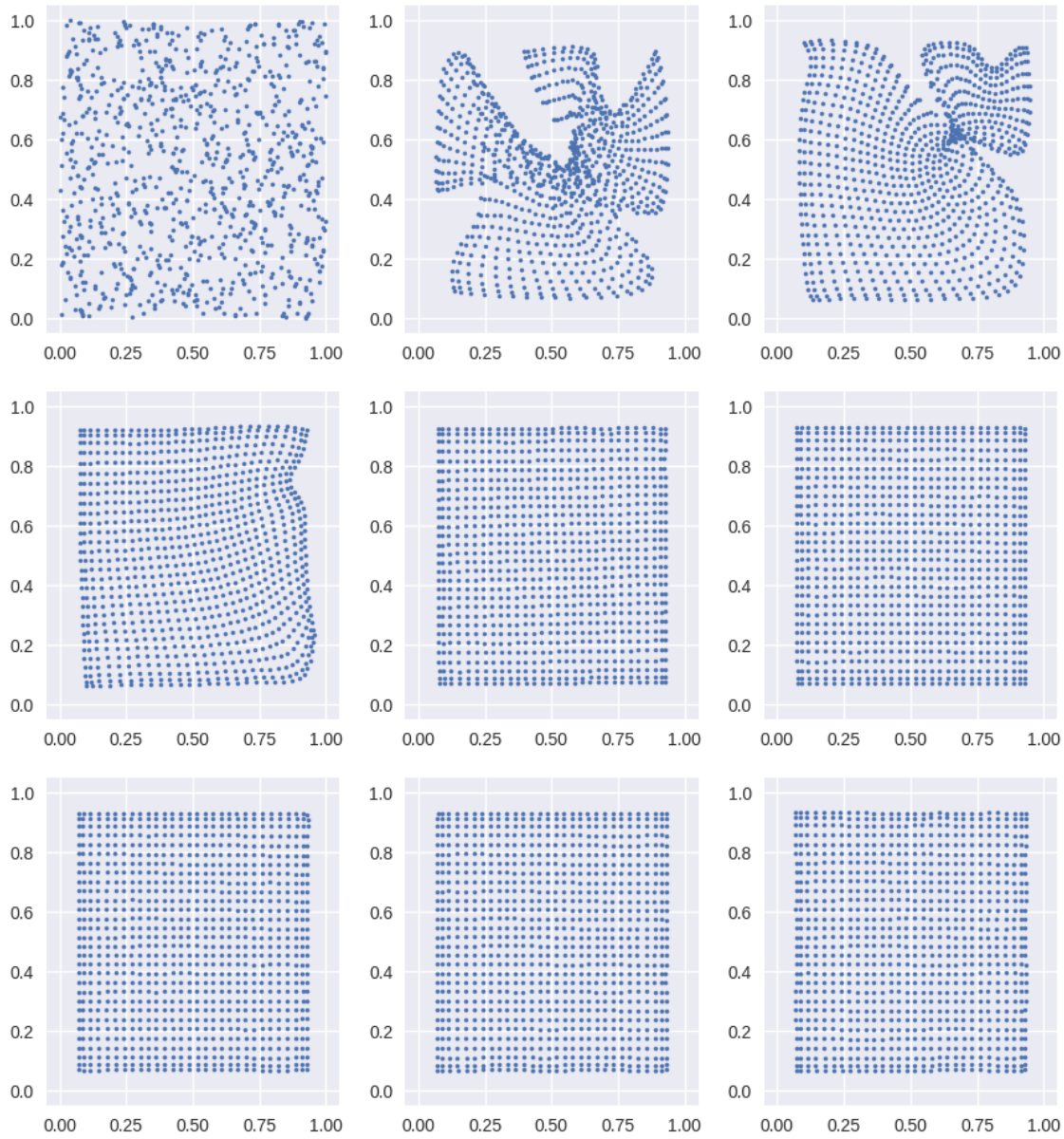

Figura 4.8: Treinamento do mapa auto-organizável. Para cada quadro foram feitas $\mathbf{1000000}$ de iterações.

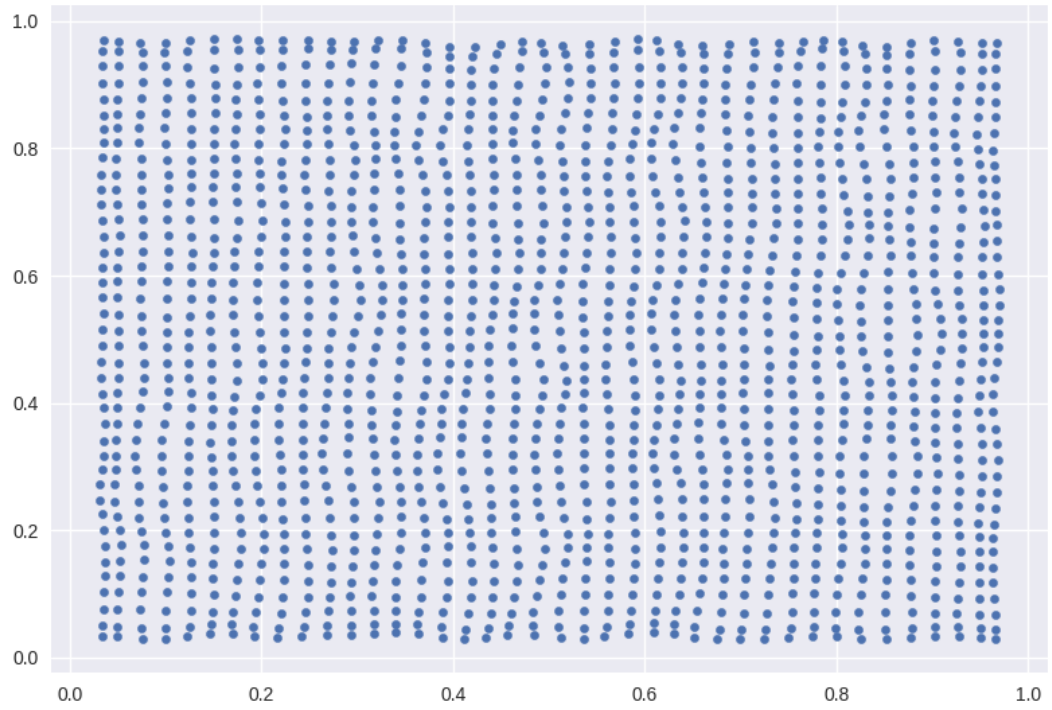

Figura 4.9: Curva a ser estudada

Para chegarmos na malha final foram feitas **10.000.000** iterações para obter a rede treinada na figura 4.9.

Podemos analisar também a qualidade do mapa auto-organizável observando o seu mapa de ativação. O mapa de ativação mostra quantas vezes cada neurônio foi vencedor dado uma amostra. A figura 4.10 mostra o mapa de ativação obtido utilizando a mesma amostra usada para treinar a rede. Como a amostra é homogênea o resultado que gostaríamos de obter é um mapa de ativação também homogêneo. Podemos observar que os neurônios no interior da malha foram ativados de forma bastante homogenia, porem os neurônios nas margens foram mais ativados e em especial os neurônios nas quinas foram ainda mais ativados. Isso acontece porque, como pode ser visto na figura 4.10, os neurônios que se encontram na margem do mapa são considerados neurônios vencedores de todos os pontos que se encontram fora do mapa auto-organizável.

Com o mapa treinado calculamos o neurônio vencedor para cada um dos 1000000 pontos das 20 amostras e então substituímos o ponto amostrado pelos geradores de números pseudo aleatórios pela posição do neurônio vencedor na malha do mapa auto-organizável. Como os neurônios da nossa malha são dispostos em uma malha 40x40 precisamos normalizar o resultado para que nossa amostra se restrinja a números entre 0 e 1.

O resultado do mapa auto-organizável possui um erro de quantização muito grande, já que o mapa é uma malha com **1600** posições possíveis como pode ser visto na figura 4.11. Isso compromete o resultado desta etapa, portanto para podermos diminuir o erro de quantização utilizamos a técnica de interpolação que foi explicada no capitulo 3. O resultado

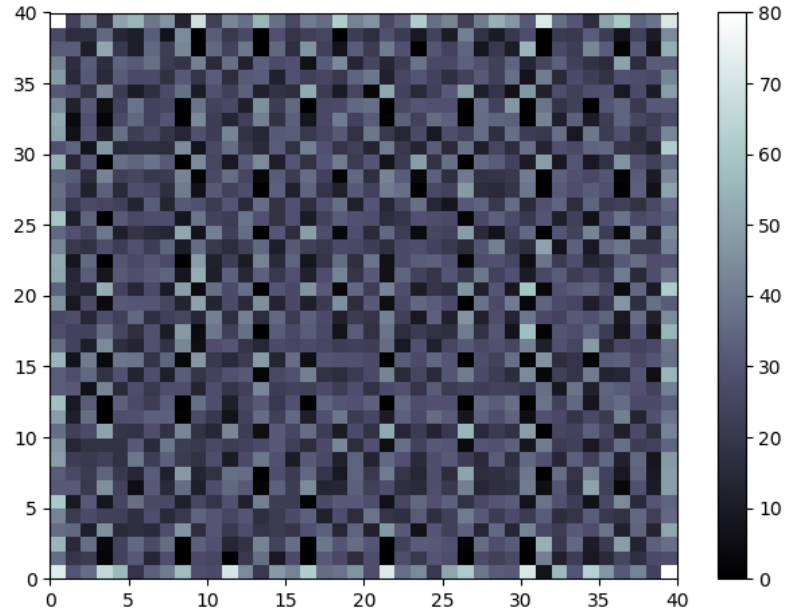

Figura 4.10: Mapa de ativação do mapa auto-organizável a partir da amostra de treinamento. Cada célula deste mapa representa um neurônio. Quanto mais claro o tom de cinza mais ativado foi o neurônio.

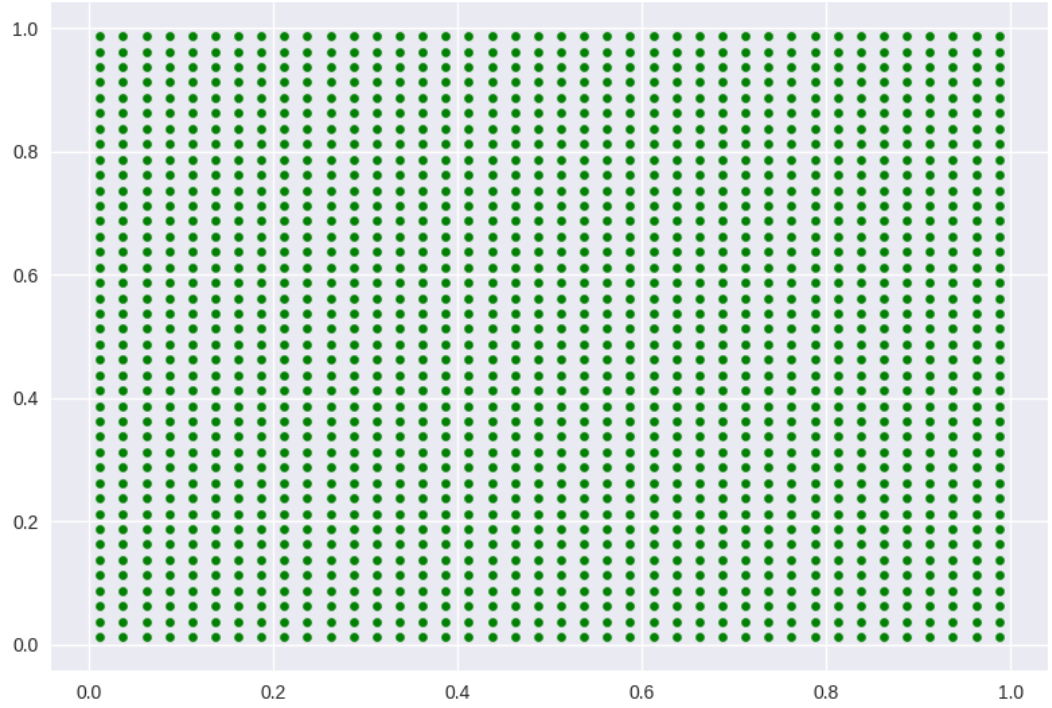

Figura 4.11: Saída do mapa auto-organizável. Neste resultado cada ponto representa mais de uma amostra.

obtido da interpolação pode ser visto na figura 4.12.

Como pode ser observado o resultado da interpolação se parece muito com a amostra original. Os pontos mais a margem da amostra sofrem mais distorções devido ao fato de que o neurônio vencedor destas amostras possui apenas 3 ou 2 vizinhos apenas. Outro fato

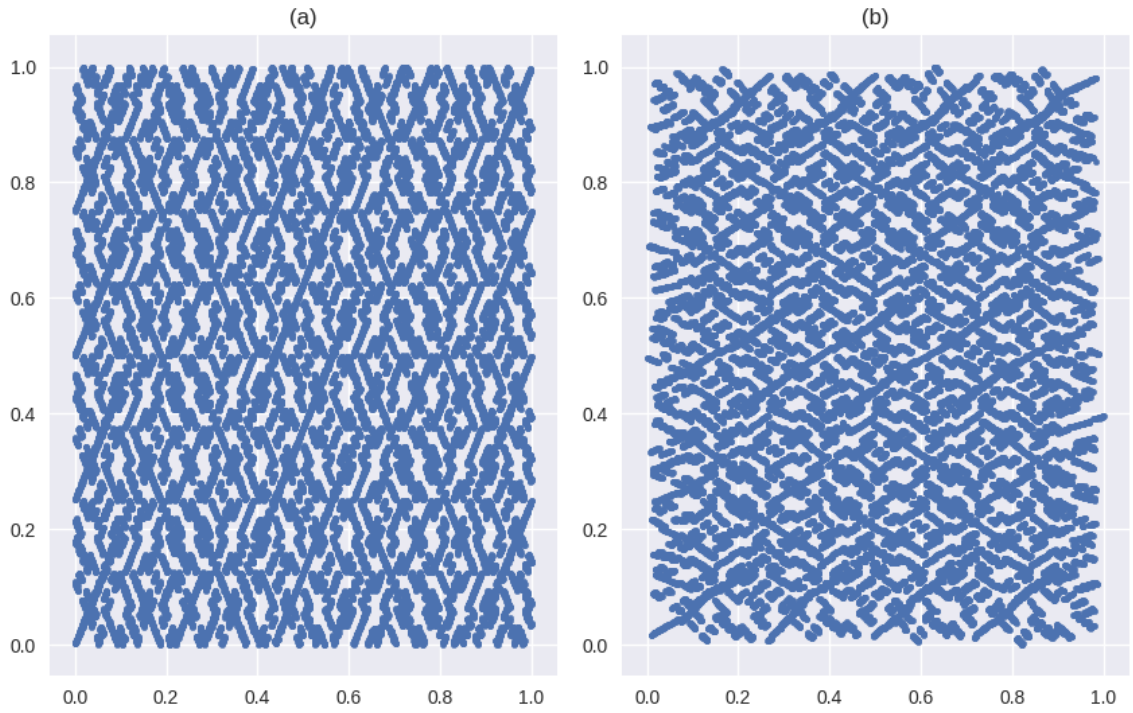

Figura 4.12: Gráfico de dispersão (a) mostra uma amostra com $\mathbf{500000}$ obtidas a partir do gerador LFSR 16-bits. O gráfico (b) mostra a mesma amostra do gráfico (a) após o método de mapas auto-organizáveis com interpolação.

interessante deste resultado é que parece que a amostra sofreu uma rotação assim como os resultados obtidos pelo o algoritmo FastICA. Os padrões nas amostras do gerador LFSR, consequência do seu baixo período, também se mantiveram.

O erro médio destas amostras para o problema proposto foi de $\mathbf{0,17}$ e $\mathbf{0,16}$, para o gerador Mersenne Twister e LFSR 16-bits respectivamente. Diferentemente método anterior, a aplicação de ICA por mapas auto-organizáveis faz com que o erro do problema de Monte Carlo não mais convergisse para 0 como mostram as figuras 4.13 e 4.14. Para entender melhor este comportamento fizemos o histograma de uma das 20 amostras que é mostrado na figura 4.15.

Como mostram os histogramas da figura 4.15, a técnica de mapas auto-organizáveis distorce muito a forma da distribuição a ponto de não se assemelhar mais como uma distribuição uniforme, comprometendo a convergência a convergência do algoritmo numérico. Tal comportamento já era esperado, uma vez que a aplicação da integração de Monte Carlo exige uma distribuição uniforme para que o erro convirja para zero, dado um número suficientemente grande de realizações.

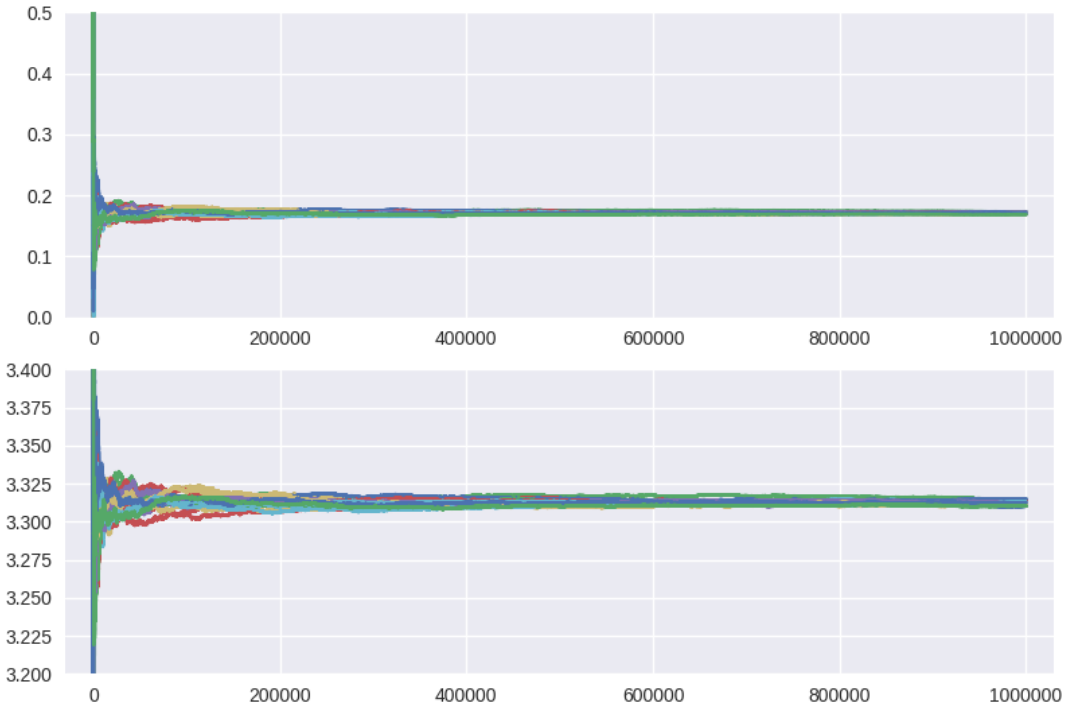

Figura 4.13: Resultado da simulação de Monte Carlo proposta utilizando o resultado do mapa auto-organizável com interpolação (Mersenne Twister). Cada cor representa uma simulação distinta com **1000000** pontos. O gráfico superior representa o valor do erro em função do número de mostras. O gráfico inferior representa o valor de π calculado numericamente a partir do resultado da simulação.

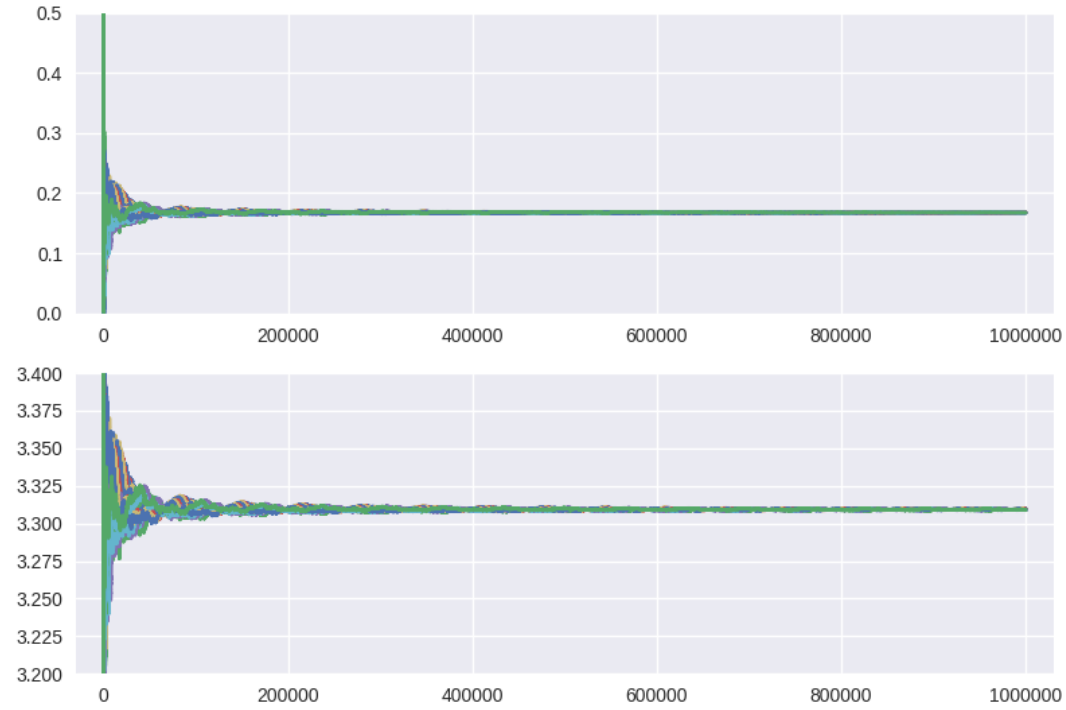

Figura 4.14: Resultado da simulação de Monte Carlo proposta utilizando o resultado do mapa auto-organizável com interpolação (LFSR). Cada cor representa uma simulação distinta com **1000000** pontos. O gráfico superior representa o valor do erro em função do número de mostras. O gráfico inferior representa o valor de π calculado numericamente a partir do resultado da simulação.

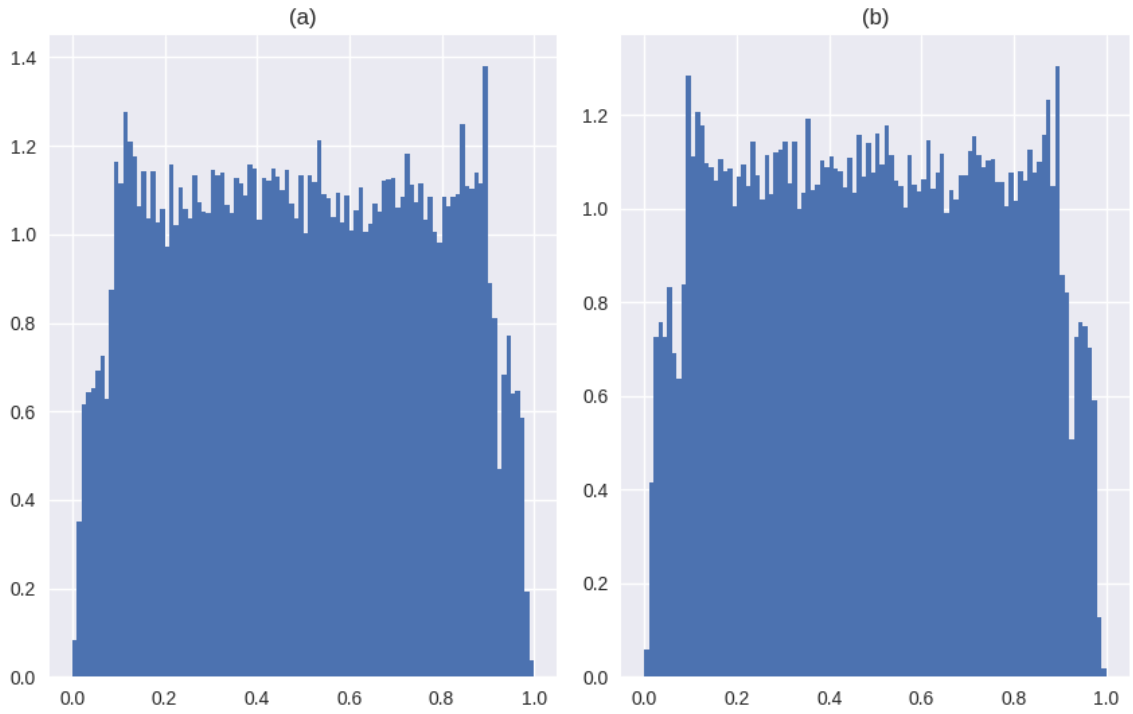

Figura 4.15: Histograma comparando amostras, a após o método de mapas auto-organizáveis e interpolação, dos geradores (a) Mersenne Twister e (b) LFSR 16-bits

Capítulo 5

Conclusão

Este trabalho teve como objetivo principal verificar se a análise de componentes independentes quando aplicada a uma amostra de números pseudo-aleatórios é capaz de tornar-la em uma amostra mais estatisticamente independente e por consequência melhorar a convergência do método de Monte Carlo quando paralelizado. Para isso usamos um problema de integração de Monte Carlo simples para podermos comparar os resultados antes e depois das técnicas de ICA. Neste trabalho foi usadas duas técnicas de ICA: uma linear, o FastICA, e outra não-linear, mapas auto-organizáveis.

Para isto tivemos que implementar um gerador de números pseudo-aleatórios LFSR 16-bits, que foi escolhido por ser um gerador com período pequeno. E para comparação usamos uma implementação pronta do gerador Mersenne Twister. Também foi utilizado uma implementação pronta do algoritmo FastICA. Para o método de mapas auto-organizáveis foi utilizada a implementação MiniSOM com modificações para atender o problema.

Os resultados obtidos utilizando a técnica de ICA linear, FastICA, mostram que tal algoritmo não foi capaz de gerar nenhuma descorrelação aparente nas amostras pseudo-aleatórias. A única mudança aparente entre o resultado do algoritmo e a amostra original é um rotação do gráfico de dispersão, este resultado já era previsto já que é uma das ambiguidades do ICA. Outro resultado interessante do FastICA é que houve uma leve mudança na densidade de probabilidade da amostra tornando ela um pouco menos uniforme. Devido a estes dois comportamentos, o FastICA não se mostrou ser uma técnica viável para melhorar a convergência de um problema de Monte Carlo paralelo. Se somarmos o tempo computacional adicional necessário pelo algorítimo e a convergência mais lenta podemos afirmar que o algoritmo FastICA não é uma alternativa viável para a otimização de Monte Carlo paralelo.

Na segunda parte deste estudo foi usada como técnica de ICA o mapa auto-organizável. Assim como o FastICA, o mapa auto-organizável não foi capaz de gerar nenhuma descorrelação aparente nas amostras. Foi feita uma interpolação no mapa para diminuir o erro de quantização, porém, como mostram os resultados, a forma das amostras foi distorcida, principalmente nas margens. Ao analisar o resultado interpolado percebemos que a o mapa

auto-organizável também rotacionou a mistura. A distorção gerada pelo mapa também pode ser vista na densidade de probabilidade da amostra. A distorção foi tanta que o quando usadas as amostras geradas pelo mapa no método de Monte Carlo o erro não mais convergiu para 0. Considerando o esforço computacional de uma mapa auto-organizável e as distorções geradas por este método concluímos que a técnica de ICA com mapas auto-organizáveis também não é uma alternativa viável para a otimização de Monte Carlo paralelo.

Mesmo não fazendo parte dos objetivos deste estudo, um resultado interessante foi mostrar que as amostras geradas pelo gerador LFSR 16-bits produzem um resultado melhor que as amostras geradas pelo Mersenne Twister quando utilizadas no método de Monte Carlo proposto neste trabalho. Este resultado é atribuído ao fato das amostras do gerador LFSR 16-bits serem mais uniformes que a do gerador Mersenne Twister.

Trabalhos Futuros

Agora que sabemos que as duas técnicas de análise de componentes independentes estudadas neste trabalho, FastICA e mapas auto-organizáveis, não são alternativas viáveis, nos resta estudar outras técnicas de ICA antes de descartamos a hipótese de que ICA pode melhorar a convergência de Monte Carlo paralelo.

Algoritmos mais avançados e complexos, para ICA não-linear, que se baseiam no princípio de informação mútua e máxima verossimilhança podem ser um bom ponto de partida para novos trabalhos. O livro "Handbook of Blind Source Separation: Independent component analysis and applications"[Comon and Jutten 2010] detalha algumas técnicas que poderiam ser testadas mais adiante.

Neste sentindo podemos citar alguns estudos que podem complementar o trabalho aqui desenvolvido:

- Aumentar o número de neurônios do mapa auto-organizável com a finalidade minimizar as distorções geradas.
- Estudar um outro problema que envolve Monte Carlo que envolva mais dimensões. Neste caso o problema da correlação das amostras é mais evidente.
- Estudar a aplicação da técnica de ICA não-linear baseada no principio de máxima verossimilhança proposto em [Matsumoto and Nishimura 1998]. Nesta aplicação é preciso conhecer a função densidade de probabilidade das fontes originais, o que, no nosso problema é um densidade uniforme.
- Estudar a aplicação da técnica de ICA que utiliza uma rede neural recorrente proposto por [Deville and Hosseini 2009]. Também é na estimação de máxima verossimilhança e pode trazer resultados interessantes.

Referências Bibliográficas

[Aapo Hyvärinen 2001] Aapo Hyvärinen, Juha Karhunen, E. O. (2001). *Independent Component Analysis*. John Wiley Sons, Inc.

[Apostol 2007] Apostol, T. M. (2007). *Calculus, Volume I, One-variable Calculus, with an Introduction to Linear Algebra*, volume 1. John Wiley & Sons.

[Athanasios and Pillai 1991] Athanasios, P. and Pillai, S. U. (1991). Probability, random variables, and stochastic processes. *Mc-Graw Hill*.

[Comon and Jutten 2010] Comon, P. and Jutten, C. (2010). *Handbook of Blind Source Separation: Independent component analysis and applications*. Academic press.

[Deville and Hosseini 2009] Deville, Y. and Hosseini, S. (2009). Recurrent networks for separating extractable-target nonlinear mixtures. part i: Non-blind configurations. *Signal Processing*, 89(4):378–393.

[Dongarra et al. 2003] Dongarra, J., Foster, I., Fox, G., Gropp, W., Kennedy, K., Torczon, L., and White, A. (2003). *Sourcebook of parallel computing*, volume 3003. Morgan Kaufmann Publishers San Francisco.

[Haykin 1994] Haykin, S. (1994). *Neural networks: a comprehensive foundation*. Prentice Hall PTR.

[Hellekalek 1998] Hellekalek, P. (1998). Don't trust parallel monte carlo! In *ACM SIGSIM Simulation Digest*, volume 28, pages 82–89. IEEE Computer Society.

[Hyvarinen 1999] Hyvarinen, A. (1999). Fast and robust fixed-point algorithms for independent component analysis. *IEEE transactions on Neural Networks*, 10(3):626–634.

[Hyvärinen and Oja 2000] Hyvärinen, A. and Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430.

[Kohonen 2001] Kohonen, T. (2001). *Self-Organizing Maps*. Springer-Verlag Berlin Heidelberg.

[Landau et al. 2015] Landau, R. H., Páez, M. J., and Bordeianu, C. C. (2015). *Computational physics: problem solving with Python*. John Wiley & Sons.

[Ludwig et al. 1995] Ludwig, L., Kessler, W., Göppert, J., and Rosenstiel, W. (1995). Som with topological interpolation for the prediction of interference spectra.

[Matsumoto and Nishimura 1998] Matsumoto, M. and Nishimura, T. (1998). Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 8(1):3–30.

[Metropolis and Ulam 1949] Metropolis, N. and Ulam, S. (1949). The monte carlo method. *Journal of the American statistical association*, 44(247):335–341.

[Newman and Barkema 1999] Newman, M. and Barkema, G. (1999). Monte carlo methods in statistical physics. 1999. *New York: Oxford*, 475.

[Pajunen et al. 1996] Pajunen, P., Hyvärinen, A., and Karhunen, J. (1996). Nonlinear blind source separation by self-organizing maps. In *In Proc. Int. Conf. on Neural Information Processing*. Citeseer.

[Poorghanad et al. 2008] Poorghanad, A., Sadr, A., and Kashanipour, A. (2008). Generating high quality pseudo random number using evolutionary methods. In *2008 International Conference on Computational Intelligence and Security*, pages 331–335. IEEE.

[Ulam J. von Neumann 1947] Ulam J. von Neumann, R. D. R. (1947). Statistical methods in neutron diffusion. *LAMS-551*.