Universidade Estadual de Maringá
Centro de Ciências Exatas
Departamento de Física

Trabalho de Conclusão de Curso

# Estudo dos padrões formados em simulações no jogo pedra-papel-tesoura sem rede no modelo de Lotka-Volterra

Acadêmica: Maria Clara Giovanna Abe

Orientador: Prof. Dr. Breno Ferraz de Oliveira

Maringá, 17 de abril de 2023

Universidade Estadual de Maringá
Centro de Ciências Exatas
Departamento de Física

Trabalho de Conclusão de Curso

# Estudo dos padrões formados em simulações no jogo pedra-papel-tesoura sem rede no modelo de Lotka-Volterra

Trabalho de Conclusão de Curso submetido à Universidade Estadual de Maringá, como parte dos pré-requisitos necessários para a obtenção do Grau de Bacharel em Física sob a orientação do Prof. Dr. Breno Ferraz de Oliveira.

Acadêmica: Maria Clara Giovanna Abe

Orientador: Prof. Dr. Breno Ferraz de Oliveira

Maringá, 17 de abril de 2023

# Sumário

# Lista de Figuras

# Agradecimentos

Gostaria de agradecer à todas as pessoas que direta ou indiretamente me ajudaram durante essa trajetória.

Primeiro à Deus.

Ao meu marido Rafael e ao meu filho Miguel, por me apoiar e incentivar em todos os momentos, eu amo vocês!

Aos aos meus pais Arali e Giovanni.

Aos meus amigos, Rafhael e Miguel, por poder compartilhar as alegrias e desesperos da graduação e da vida.

Por último, mas não menos importante, meus sinceros agradecimentos ao meu orientador Breno Ferraz de Oliveira, por ser o melhor professor que eu poderia ter e por sua imensa paciência.

# Resumo

O modelo de Lotka-Volterra no jogo RPS, do inglês *rock-paper-scissors*, trata de uma interação entre três espécies, na qual uma espécie é a presa e a outra espécie é o predador, interagindo de maneira cíclica. Nesse modelo, as ações de reprodução e predação acontecem simultaneamente. Neste trabalho, utilizamos os resultados de simulações estocásticas de modelos RPS, mostrando que em um modelo sem rede ocorre a formação do padrão de espirais. Também estudamos as propriedades dinâmicas, como a densidade das espécies em função do tempo.

# Introdução

O RPS (pedra-papel-tesoura) do inglês *rock-paper-scissors*, que é um jogo em que há uma interação cíclica entre três elementos, na qual uma espécie será a presa e a outra espécie será o predador [6]. A partir dele fizemos uso do modelo estocástico de Lotka Volterra, em que as espécies possuem 2 opções de ações: a predação-reprodução ou a mobilidade para realizar este trabalho.

Em 1975 foi publicado um dos primeiros trabalhos, em RPS, na qual Jackson e Buss observaram uma competição cíclica entre recifes de corais [7]. Para que aconteça essa competição entre os recifes de corais, eles usam uma capacidade que alguns organismos possuem, chamada alelopatia, na qual é produzida uma substância que pode ser nociva e influenciar no desenvolvimento de outros organismos. Assim é possível que ocorram essas relações, pois desencadeiam em uma monopolização de recursos. Existem outros organismos que também usam a alelopatia para que aconteça a competição cíclica, temos como exemplo os plânctons, entre cepas da bactéria *Escherichia coli* e entre plantas e fungos.

Depois, em 1996, houve o trabalho de Sinervo e Lively, que foi publicado pela revista Nature [8]. Eles observaram uma competição cíclica em lagartos da espécie *Uta stanburiana*, encontrado na faixa litorânea da Califórnia. Há machos com 3 colorações de garganta (Figura 1), e essas cores que foram definidas de forma genética mostram como eles cuidam de seu território.

Os lagartos mais territorialistas e agressivos possuem garganta alaranjada, os menos territorialista e menos agressivos possuem garganta azul escura e os que não defendem seu território são os que possuem garganta com listras amarelas (cor semelhante a das fêmeas).

A reprodução desses lagartos também acontece de forma cíclica, conforme observamos na Figura 2. Os lagartos amarelo possuem vantagem sobre o laranja, pois como o lagarto laranja possui muito território que é difícil de cuidar, e o amarelo que possui muito semelhança com as fêmeas, este consegue se misturar sem ser percebido e se reproduzir. O lagarto laranja possui vantagem sobre o lagarto azul, pois como o lagarto azul não é territorialista o laranja ”conquista” o espaço dele e suas fêmeas. Finalmente, como o azul possui vantagem sobre o amarelo e o ciclo se reinicia.

Essas relações podem ser demonstrados a partir de simulações, em que é possível adi-

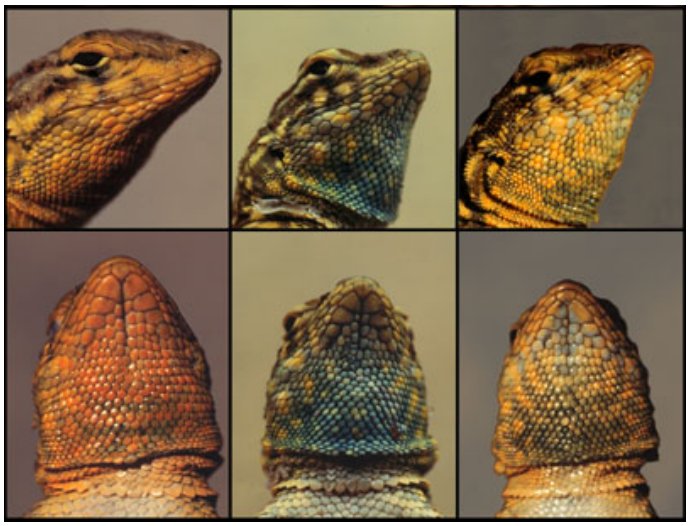

Figura 1: Ilustração dos três tipos de lagartos e suas gargantas [1].

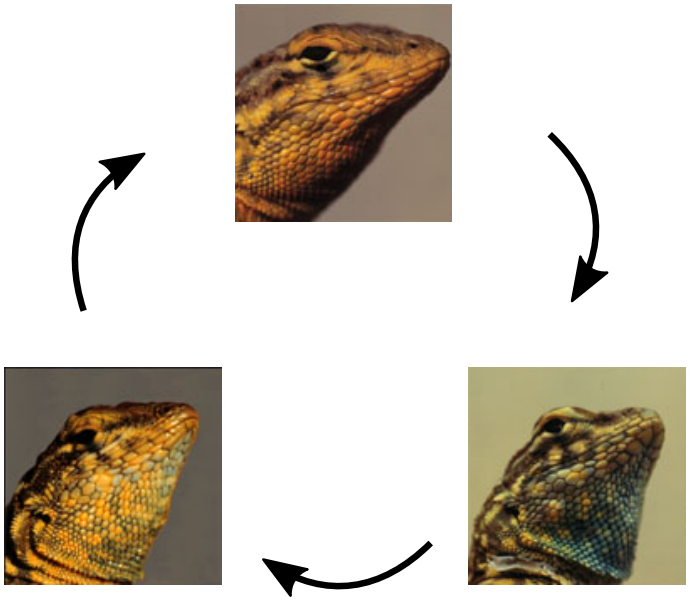

Figura 2: Hierarquia cíclica entre os lagartos, figura adaptada a partir do trabalho da Nature [1].

cionar interações como mobilidade, predação e reprodução às espécies. Em 2002 cientistas fizeram um experimento com três tipos de bactérias, observaram nele que havia interação cíclica com reprodução e predação, e assim conseguiram explicar o crescimento dessas bactérias e o padrão que foi encontrado [9]. Em 2007 outros cientistas realizaram simulações onde observaram que as bactérias também poderiam se mover, e essa mobilidade poderia contribuir para o crescimento ou extinção de uma espécie.

No capítulo 1, apresentamos o RPS, o modelo de Lotka-Volterra com rede, seguido pelo modelo sem rede que descreve matematicamente alguns exemplos de dinâmica de populações e será a forma como utilizaremos em nossa simulações. No capítulo 2, apresentamos os métodos computacionais que foram implementados na escrita do programa,

falaremos brevemente sobre transformada de Fourier, seguida por algumas noções de pariedade, a transformada de Fourier discreta e como utilizamos a regra do trapézio para discretizar as integrações e encerramos esse capítulo como uma aplicação, mostrando o uso da transformada de Fourier para limpar um sinal tipo seno com ruído. No capítulo 3, apresentamos os resultados obtidos e fizemos as considerações sobre o resultados das simulações. O código computacional utilizado se encontra no apêndice.

# Capítulo 1

# O jogo RPS e o modelo de Lotka-Volterra

Desde 2002 o RPS vem sendo utilizado em simulações para demonstrar o padrão de crescimento, as interações e formações de padrões ao longo do tempo [9, 10]. Nesse capítulo, iremos apresentar o jogo RPS e o modelo estocástico de Lotka Volterra com rede e sem rede.

## 1.1 RPS

O RPS que vem do inglês *rock-paper-scissors*, é o nosso conhecido pedra-papel-tesoura, ou Jokenpô. O Jokenpô é jogado em duplas, na qual cada um tem três opções de jogadas, a pedra (punho fechado), o papel (mão aberta com dedos encostados) ou tesoura (dedos indicador e médio abertos em V), para saber o vencedor é utilizado um sistema cíclico de três espécies sempre com uma presa e um predador [6], assim, pedra quebra a tesoura, a tesoura corta o papel e o papel cobre a pedra (veja a Figura 1.1).

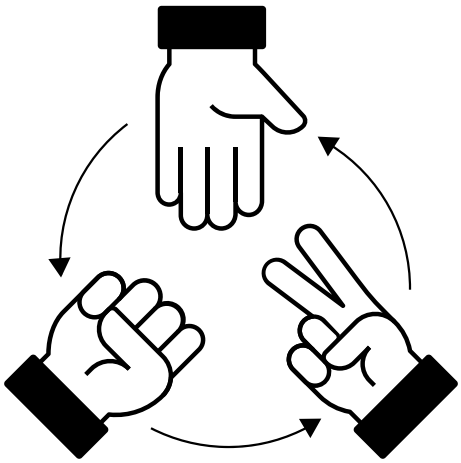

Figura 1.1: Representação que mostra a interação cíclica que acontece no jogo Jokenpô [2].

Nesse jogo todas as jogadas tem a mesma probabilidade de ganhar ou perder.

Existem outras variação do jogo RPS, mas a mais conhecida é a do jogo RPS, lizard e Spock, essa variação foi inventada por Sam Kass e popularizada após aparecer no seriado *The Big Bang Theory*, na qual um personagem ensina como se joga. Mesmo que há novos elementos nesse jogo, podemos observar que o modelo cíclico permanece (Figura 1.2).

Nessa versão além dos três elementos já conhecidos do RPS, temos o lagarto (mão em forma de fantoche) e o Spock (mão imitando uma saudação dos vulcanos em Star Trek). E adicionamos as seguintes interações entre os elementos: Pedra esmaga lagarto, lagarto envenena Spock, Spock derrete tesoura, tesoura decapita lagarto, lagarto come papel, papel refuta Spock e Spock vaporiza pedra.

Figura 1.2: Representação que mostra a interação cíclica que acontece no jogo pedra, papel, tesoura, lagarto e Spock [3].

Mesmo que seja muito interessante modelos com mais elementos, em nosso trabalho iremos focar em modelos de RPS com apenas três espécies.

## 1.2 Lotka-Volterra

Alfred J. Lotka (1880 – 1949) e Vito Volterra (1860 – 1940) foram matemáticos que, individualmente, trabalharam em um modelo para descrever a interação entre espécies.

Em 1925 Volterra analisou o acréscimo na população de tubarões e o decréscimo na população do peixe que era sua presa em um mar da Itália, mesmo que durante a Primeira Guerra Mundial não houvessem atividades pesqueiras. Com isso elaborou equações para expor esse ocorrido [11]. No mesmo ano, de 1925, Lotka publicou um livro [12] na qual estudou as relações entre predador e presa e apresentou a mesma equação de Volterra.

O modelo de Lotka-Volterra recebeu esse nome pois foi a unificação dos dois modelos. Eles foram os primeiros a tentar compreender a relação entre predador-presa a partir de

equações matemáticas.

Esse modelo não descreve as relações complexas que observamos na natureza, porém, um modelo com duas espécies é um grande passo para a compreensão de fenômenos.

Considerando um modelo predador-presa, na qual a densidade de presas é $x(t)$ e a densidade de predador é $y(t)$.

$$\frac{dx}{dt} = x(a - \alpha y) \ , \tag{1.1}$$

$$\frac{dy}{dt} = y(-b + \beta x) \ . \tag{1.2}$$

Podendo descrevê-las como:

(variações do número de presas) = (aumento natural) - (destruição pelos predadores)

(variações do número de predadores) = (mortes na ausência de presas) + (aumento causado pela alimentação)

as constantes $\alpha$ $\beta$, $a$ e $b$ são positivas e controlam as populações no sistema predador-presa [13] .

Nas equações (1.1, 1.2) percebemos que elas não prevem outra alimentação dos predadores, com isso há um decrescimento da população do predador em função da constante $b$, mas aumenta com uma densidade $x$ de presas.

Se houver escassez de predadores, também conseguimos observar nas equações (1.1, 1.2) que haveria um aumento exponencial da população de presas.

### 1.2.1 Modelo estocástico com rede de Lotka Volterra

O modelo de Lotka-Volterra no jogo RPS trata de uma interação entre 3 espécies de forma cíclica. Em um modelo com rede os indivíduos estão distribuídos em um lugar pré definido, como se estivessem em um tabuleiro de damas, ou uma toalha xadrez de piquenique (Figura 1.3).

Podemos observar essa distribuição na Figura 1.4, na qual temos uma rede $4x4$ com 16 espaços em $t = 0$. Percebemos que cada indivíduo se encontra em seu espaço e possui vizinhos ortogonais, na qual podem interagir. Consideramos que as bordas se conectam, tendo uma condição de contorno periódica, assim simulamos uma rede infinita para desprezar efeitos de borda.

Note que todos os espaços são preenchidos por uma das três espécies, nesse modelo não temos a opção de espaço vazio.

Para demonstrar estamos usando uma rede com 16 espaços, mas nada nos impede de usarmos redes maiores. Nesse modelo é possível duas ações: A mobilidade, na qual o indivíduo troca de lugar com o vizinho, como vemos na Figura 1.5, ou as ações de predação e reprodução que acontecem simultaneamente, como podemos observar na Figura 1.6.

Figura 1.3: Foto de um tabuleiro de damas. Fonte: [4].

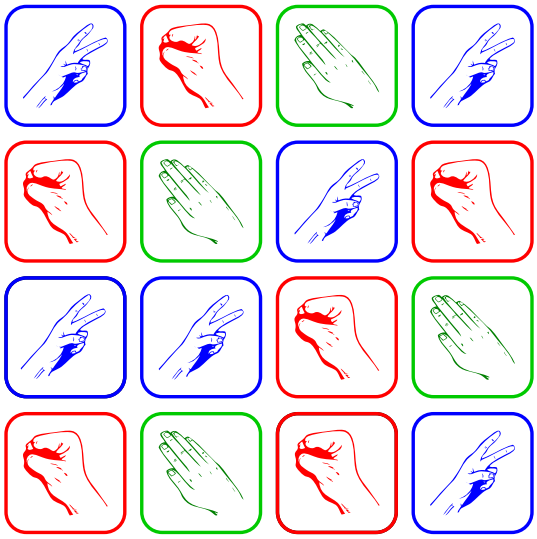

Figura 1.4: Representação de uma rede para o modelo de Lotka Volterra em t=0. Fonte: [5].

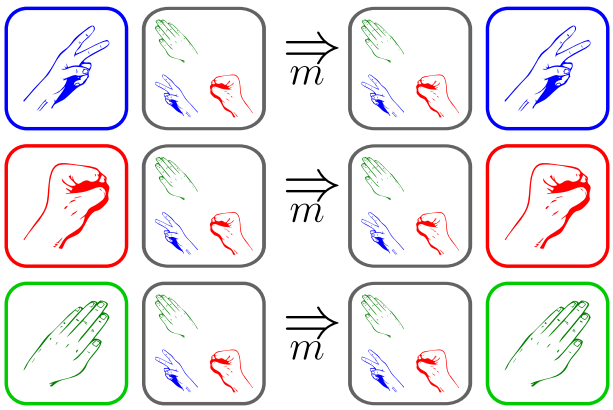


Figura 1.5: Ilustração da ação mobilidade para o modelo de Lotka-Volterra. Fonte: [5].

No modelo estocástico lidamos com amostragem indeterminadas, ou seja, aleatórias que ao realizar a simulação nos fornece resultados numéricos. Com rede estamos limi-

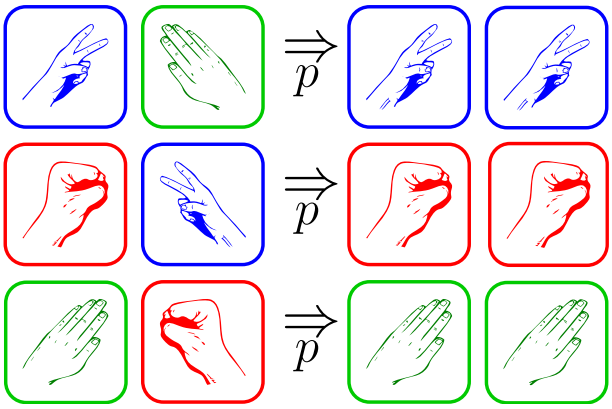


Figura 1.6: Ilustração da ação de predação-reprodução para o modelo de Lotka-Volterra. Fonte: [5].

tados aos quatro vizinhos ortogonais para realizar as ações de predação-reprodução ou mobilidade. Uma forma de alcançar mais indivíduos é utilizando o modelo sem rede, que iremos tratar a seguir.

### 1.2.2 Modelo estocástico sem rede

Para melhorar a interação entre os indivíduos e diminuir as limitações de espaço no nosso trabalho utilizaremos o modelo estocástico de Lotka-Volterra sem rede.

Nesse modelo, os indivíduos não possuem um lugar definido, e podem estar em qualquer lugar do espaço definido desde que aquele ponto não esteja ocupado. Assim, a rede não limita o número de indivíduos que cabem nela. Em um modelo com rede tamanho 3 x 3 comportaria no máximo 9 indivíduos, 1 para espaço, já no modelo sem rede não temos limitação do número de indivíduos de acordo com o tamanho no nosso espaço.

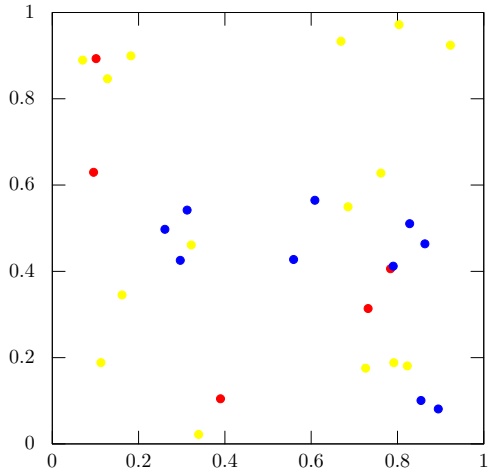


Figura 1.7: Ilustração de uma geração inicial em um modelo sem rede, com 50 indivíduos de 3 espécies diferentes.

Na Figura 1.7 observamos uma geração inicial ($t = 0$) de um modelo sem rede com 3

espécies e 50 indivíduos. Como as posições são geradas de maneira aleatória, note que os indivíduos não estão com espaçamentos simétricos entre si.

Agora que já entendemos a partir dessa explicação breve como funciona um modelo sem rede iremos relatar como acontece 1 geração na simulação.

1. Um elemento qualquer é sorteado de maneira aleatória, chamamos ele de ativo.

2. Um vizinho próximo a ele é sorteado, chamamos esse sorteado de passivo.

   Como nesse modelo não temos uma rede definida precisamos parametrizar um valor de raio para que aconteça essas interações. Por exemplo, para que aconteça mobilidade teremos $l_m = 0.02$ e para que aconteça a predação teremos $l_p = 0.02$. Observe que estou definindo os padrões de mobilidade e predação iguais, porém, como são parâmetros distintos poderiam ter valor também distintos.

3. Sorteamos, também de forma aleatória, uma ação: Predação ou mobilidade. Essas ações são sorteadas de acordo com a probabilidade que definimos no inicio do programa sendo $p_m$ a probabilidade de ser sorteada a ação de mobilidade e $p_p$ a probabilidade de ser sorteada a predação. Se usarmos $p_m = p_p = 0.5$ teremos probabilidades iguais de serem sorteadas as duas ações.

4. Caso a ação sorteada seja predação é verificado se o ativo é o predador do passivo, se for junto com a predação ocorre a reprodução, se não for nada acontece.

5. Caso seja sorteado a ação de mobilidade o ativo e o passivo trocam de posição.

6. Esse ciclo é feito com a quantidade de indivíduos que há na rede, por exemplo, se há 20 indivíduos esse ciclo é realizado 20 vezes. A geração é realizada na quantidade de indivíduos que há na simulação, mas, como eles são sorteados de forma aleatória pode acontecer de um mesmo ser sorteado mais de 1 vez e outro não ser sorteado.

# Capítulo 2

# Métodos computacionais

Nesse capítulo iremos expor e exemplificar o métodos computacionais numéricos que utilizamos para realizar as simulações, assim como adaptações que precisamos fazer como discretizar a transformada de Fourier. Também realizamos uma aplicação retirando o ruído de uma função seno.

## 2.1 Transformada de Fourier

A transformada de Fourier é uma transformada integral na qual sua função está em termos de funções de base sinusoidal, que é uma função que pode ter problemas com processamento de sinais. Mesmo que a transformada de Fourier não se limite à funções temporais podendo haver mesmo em funções espaciais [14], por convenção usamos o domínio temporal como original. Calculando a transformada de Fourier de uma função temporal obtemos uma função com valor de frequência complexo, o argumento complexo seria o deslocamento da base naquela frequência. Podemos também definir a reversão que chamamos de transformada inversa com a frequência como domínio, na qual reconstituímos uma função temporal original.

Para simplificar chamaremos $\omega$ de frequência e a usaremos como variável independente, mesmo sabendo que a "frequência angular" pode ser dada como $\omega = 2\pi\upsilon$ e sendo $t$ uma variável espacial, o $\omega$ significaria um número de onda $k = 2\pi/\lambda$, $\lambda$ é o comprimento de onda.

Estabelecendo que a transformada de Fourier é $g(\omega)$. Logo, a transformada de Fourier da função $f(t)$ pode ser definida como

$$g(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-i\omega t} f(t)dt, \tag{2.1}$$

Sabendo que existe a transformada inversa de Fourier, na qual possuindo a frequência e o comprimento de onda é possível recuperar a função original, a transformada inversa de Fourier pode ser definida como

$$f(t) = \int_{-\infty}^{\infty} e^{i\omega t} g(\omega) d\omega, \tag{2.2}$$

### 2.1.1 Propriedades de paridade

A transformada de Fourier possui propriedades que são importantes, a seguir listaremos, sem demonstração, algumas propriedades de paridade, as demonstrações podem ser observadas na referência [14].

1. Se $f(t)$ for uma função real,

   $Re[g(-\omega)] = Re[g(\omega)]$ e $Im[g(-\omega)] = -Im[g(\omega)]$,

   ou seja,

   $g(-\omega) = g*(\omega)$ [14].

2. Se $f(t)$ for uma função real e par, a transformada de Fourier $g(\omega)$ será real.

3. Se $f(t)$ for uma função real e ímpar, então a transformada de Fourier $g(\omega)$ será imaginária.

4. Se $f(t)$ é uma função imaginária, então as propriedades 1, 2 e 3, acima, serão válidas trocando apenas "real" por "imaginária".

## 2.2 Transformada de Fourier discreta

A transformada de Fourier na sua forma analítica 2.1 não pode ser utilizada em nosso estudo computacional, porque o código computacional não consegue realizar equações contínuas, logo, precisamos de uma forma numérica para que seja possível calcular a integral de acordo com os dados que produzimos em nossas simulações.

Uma solução para o nosso trabalho é usar a transformada de Fourier discreta, pois com ela as integrais são discretizadas, substituindo-as assim por somatórias que podemos realizar utilizando nossos dados. Além disso, a saída $g(\omega k)$ também é discreta.

### 2.2.1 Integral de Fourier discreta

Precisamos integrar os dados, logo, precisamos de uma forma de calcular a integral numericamente.

Lembre-se que ao calcular a integral estamos calculando a área abaixo da curva formada pela função $f(x)$.

A Figura 2.1 representa uma função $f(x)$ no plano cartesiano $xy$, abaixo de curva está destacado entre os pontos $a$ e $b$. Essa figura representa a equação a seguir:

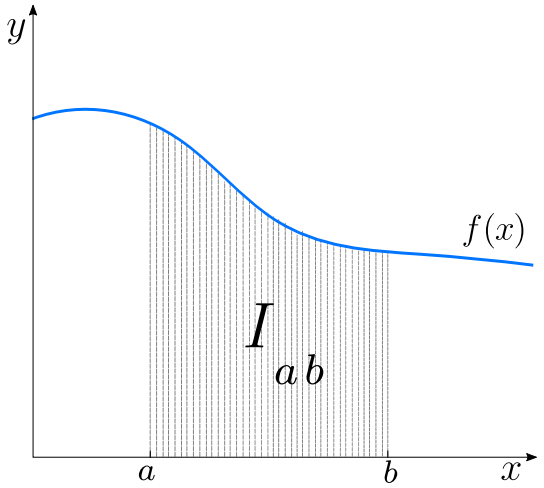


Figura 2.1: Função $f(x)$, na qual $I_{ab}$ é a integral da função entre os pontos $a$ e $b$.

$$I_{ab} = \int_b^a f(x)dx. \tag{2.3}$$

Uma aproximação satisfatória que temos para calcular essa integral é pela regra do trapézio [14].

Entre os pontos $a$ e $b$ vemos que a função $f(x)$ representa uma curva sem interrupções. Assim, uma forma de se aproximar do cálculo dessa área seria imaginar que temos várias trapézios. Sabemos que os trapézios não nos darão o valor exato, porém, se usarmos trapézios com pequenos valores de $x$ e preenchermos toda a área que pretendemos calcular podemos diminuir o erro.

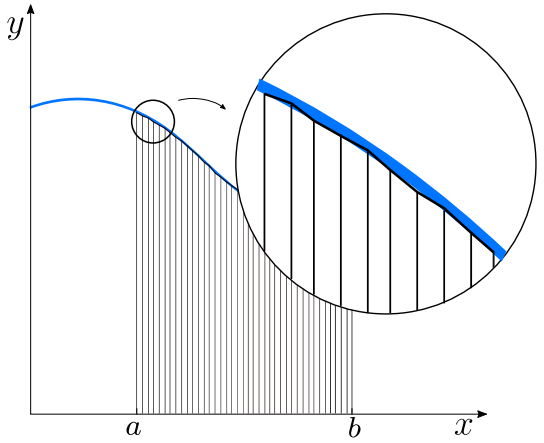


Figura 2.2: A área abaixo da curva pode ser calculada a partir de uma junção de vários trapézios.

Utilizamos os trapézios, pois além de ser uma forma geométrica que se adapta muito bem à curva, conhecemos o cálculo da sua área, que será necessário para o próximo passo.

Para saber a área pintada abaixo da curva mostrada da imagem 2.1 basta somar a área de todos os trapézios. Para as bases dos trapézios iremos usar $f(x)$ e $f(a + x)$, e

altura $\Delta x$. Logo,

$$I_{ab} \approx \frac{\Delta n}{n} \sum_{i=0}^{n} \frac{(f(x_i) + f(x_{i+1}))}{2}, \tag{2.4}$$

no qual $x_0 = a$ e $x_n = b$, e a função desloca $\Delta x$ a cada $i$.

Além da regra do trapézio existem outros métodos para calcular a integral discreta, como a regra do ponto médio ou a regra de Simpson [14]. Porém, o uso da regra do trapézio será o método que usaremos. De acordo com os dados que a simulação nos fornecem, como o da densidade de espécies ao longo do tempo, a partir dessa integral discreta iremos discretizar a transformada de Fourier.

Agora, usando (2.1), as propriedades de paridade e (2.4), e considerando que $f(t) = 0$ se $t$ não está dentro do intervalo $[a, b]$ e que temos uma quantidade de termos da integral $n$ grande para que possamos desconsiderar as correções, podemos definir a transformada de Fourier discreta como

$$g(\omega_k) = \Delta t \sum_{j=1}^{n} exp(-i\omega_k t_j) f_j, \tag{2.5}$$

com $f_j = f(t_j)$.

### 2.2.2 Aplicação

No modelo RPS a Transformada de Fourier possui muitas aplicações, como para evidenciar frequência característica. Como iremos exemplificar a seguir em um sinal com ruído, utilizando o que mostramos acima para remover o ruído do sinal.

Considere um sinal do tipo $f(x) = \text{sen}(nx) + \sigma$ na qual $\sigma$ representa um ruído qualquer, como podemos observar na Figura 2.3.

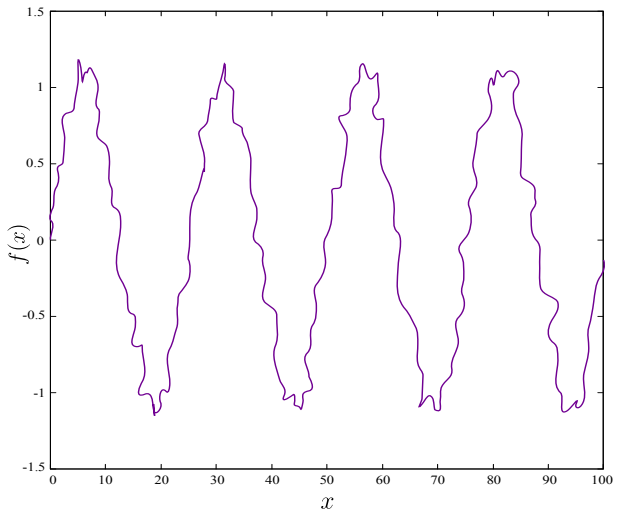


Figura 2.3: Ilustração de um sinal do tipo seno com ruído, função pode ser definida como $f(x) = sen(nx) + \sigma$

Primeiro iremos calcular a Transformada de Fourier da $f(x)$, e a partir dela a densidade espectral, que usamos para identificar a frequência característica do sistema.

A densidade espectral pode ser calculada usando a equação (2.6) e o resultado é mostrado na Figura 2.4:

$$S(\omega) = |g(\omega)|^2 \tag{2.6}$$

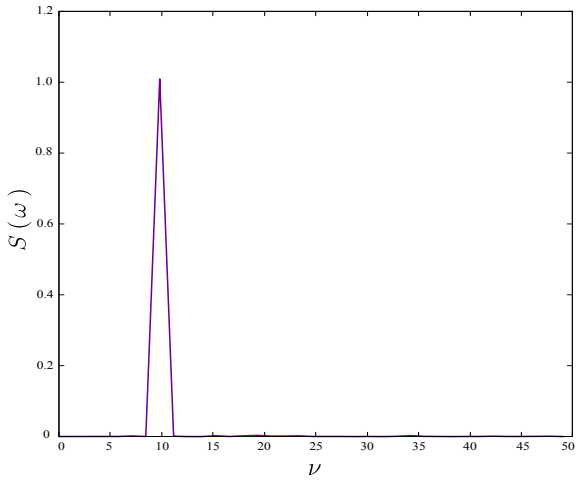


Figura 2.4: Densidade espetral do sinal $f(x)$.

Para obter o sinal limpo, encontramos a frequência característica e em qualquer ponto fora dela multiplicamos a transformada de Fourier por zero, para dessa forma minimizar todas as frequências com ruído, multiplicando o resultado obtido pela transformada inversa de Fourier teremos o sinal limpo. Percebemos que o sinal limpo pode ser representado por sen$(nx)$, que é o que esperamos, que após todos esse processo não houvesse o ruído $\sigma$. Podemos comparar o sinal inicial $f(x) = \text{sen}(nx) + \sigma$ e o sinal obtido $f(x) = \text{sen}(nx)$ na Figura 2.5.

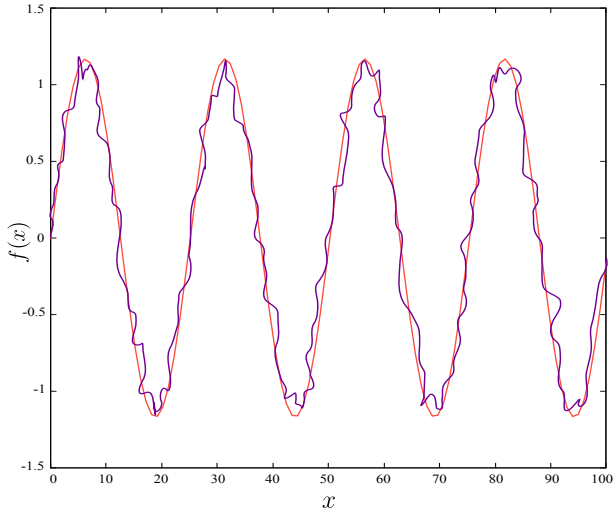


Figura 2.5: Comparação entre o sinal limpo e o sinal com ruído.

# Capítulo 3

# Resultados

No capítulo apresentado a seguir apresentaremos os resultados obtidos após o estudo que pontuamos nos capítulos 1 e 2 que foram utilizados para construir o código computacional que se encontra no Apêndice A.

Traremos simulações de RPS com modelo de Lotka -Volterra sem rede, sempre com 3 espécies.

Nessas primeiras simulações temos 1500 indivíduos e $t = 5000$ gerações, colocamos as condições de $p_m = p_p = 0.5$, assim as probabilidades da ação sorteada ser predação ou mobilidade são iguais, e $l_m = l_p = 0.02$ para que a distância de alcance do vizinho independente da ação também serem iguais.

Na Figura 3.1 temos dois painéis, o primeiro, da esquerda, mostra o tempo inicial $t = 0$, e conseguimos perceber que pelo tamanho do nosso espaço essa quantidade de indivíduos nos dará uma densidade de espécies baixa, o que sabemos de acordo com a nossa referência [15] que não nos mostrará após gerações a formação de espirais. Assim percebemos que o resultado encontrado no painel da direta, após $t = 5000$ está de acordo com a referência [16] do trabalho de M. Peltomäki e M. Alava que não diziam ser possível observar os padrões de espirais.

Utilizando os mesmos parâmetros de $l_m$, $l_p$, $p_m$, $p_p$ e número de indivíduos vamos observar como se comporta a densidade de espécies em função do tempo.

Ao realizar a densidade de espécies seguindo os parâmetros da Figura 3.1, na Figura 3.2 percebemos que a densidade de espécies se conserva ao longo do tempo. Ou seja, essa densidade de espécies $\rho_i$ vária entre 0.15 e 0.55, mas, há gerações em que alguma espécie se mostra de forma mais abundante e consequentemente outros de forma menor. Continuando as gerações as espécies trocam de posição, porém não há extinção de nenhuma espécie.

Complementando a Figura 3.2 temos a Figura 3.3 em que calculamos a densidade espectral da densidade de espécies. Para encontrar o valor da frequência característica precisamos lembrar da transformada de Fourier discreta. Na qual a partir do resultado dela teremos a frequência característica e os dados necessários para realizar a Figura 3.3.

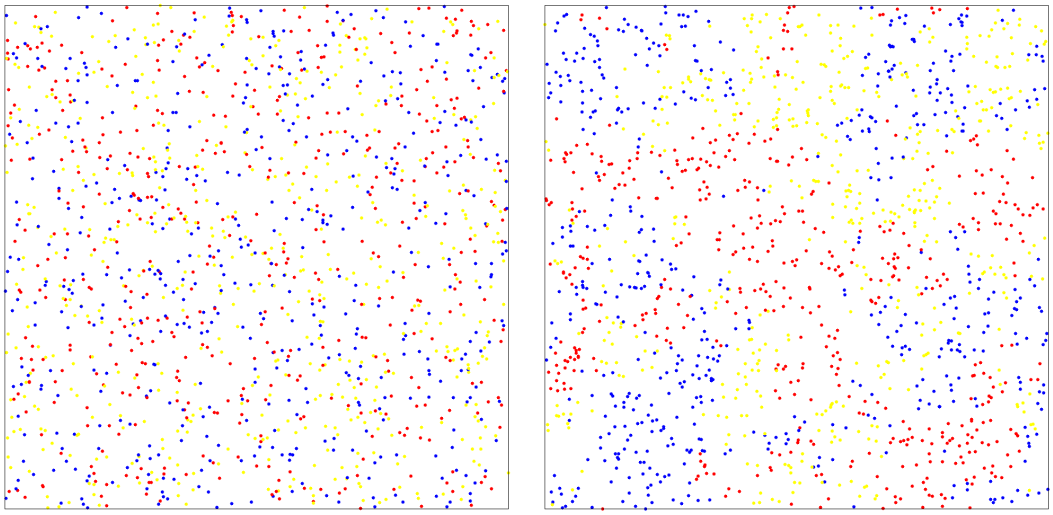

Figura 3.1: Configuração dos 1500 indivíduos no tempo inicial (painel da esquerda) e após $t = 5000$ gerações (painel da direita). Os parâmetros utilizados foram $p_m = 0.5$, $p_p = 0.5$, $l_m = 0.02$ e $l_p = 0.02$.

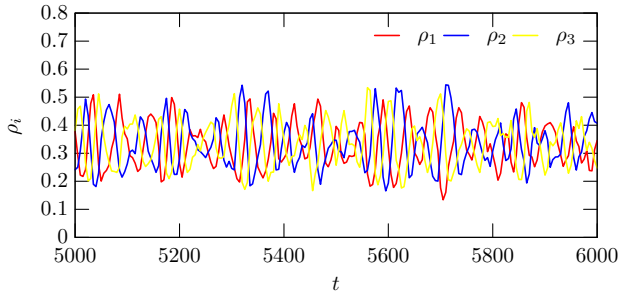


Figura 3.2: Evolução temporal das densidades das espécies. Os parâmetros utilizados nesse resultado, foram os mesmos utilizados para confecção da Figura 3.1

Ao realizar as simulações percebemos resultados similares para as três espécies, com isso para simplificar apresentamos apenas o resultado para a espécie 1. Em todas as espécies encontramos uma frequência característica média de 220. E o que isso nos indica? Que considerando um intervalo de 10 mil gerações ($t = 10000$), em aproximadamente 220 vezes a espécie 1 foi a mais abundante, a que se destacava. Nota-se que os parâmetros para sortear ações de predação e mobilidade são iguais ($p_m = p_m$), logo, era esperado que encontrássemos a mesma média de valores para as três espécies.

(Em nosso trabalho realizamos apenas a transformada de Fourier discreta para uma densidade de indivíduos, pois para cada transformada é necessário 10 mil simulações, se fossemos realizar para todas as simulações precisaríamos de um processador mais potente do que o disponível, como um *Cluster* e mesmo assim o processo seria muito demorado).

Esse é o resultado que esperávamos se considerarmos o trabalho de M. Peltomäki and M. Alava [16].

Porém, contrariando esse resultado, na próxima simulação apresentada observamos a

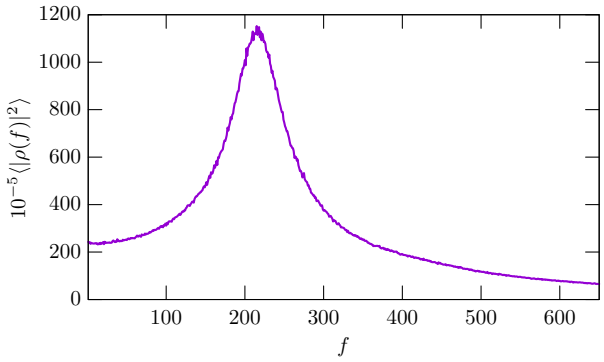


Figura 3.3: Densidade espectral da densidade da espécies 1. Notem que, o mesmo resultado é obtido para as outras espécies. A frequência característica é de 220, ou seja, em um intervalo de 10 mil gerações a espécie 1, em média, foi 220 vezes a espécie mais abundante. Os parâmetros utilizados nesse resultado foram os mesmos utilizados para confecção da Figura 3.1

mudança acontecer apenas mudando um parâmetro.

Na próxima simulação manteremos os parâmetros de distância para mobilidade e predação e probabilidade de haver mobilidade ou predação ($p_p = p_m = 0.5$ e $l_m = l_p = 0.02$) e mudaremos apenas a quantidade de indivíduos, agora temos 30000 indivíduos. Os resultados obtidos após esse aumento da quantidade de indivíduos podem ser observados na Figura 3.4.

Observando a Figura 3.4, no painel da esquerda temos o tempo inicial $t = 0$ na qual é muito semelhante ao que encontramos na Figura 3.1 mudando apenas a densidade de indivíduos que se encontra nesse painel. Já no painel da direita temos o tempo $t = 5000$, na qual percebemos claramente uma grande diferença se comparado com a figura 3.1 em que o tempo passado é igual mudando apenas a quantidade de indivíduos. Na Figura 3.4 foi possível observar a formação dos espirais, mesmo com a conservação da densidade de indivíduos. O que no trabalho [16] M. Peltomäki and M. Alava não diziam ser possível.

Essa constatação já havia sido realizada em trabalhos [15] e conseguimos, com uma densidade de indivíduos semelhante chegar ao resultado esperado.

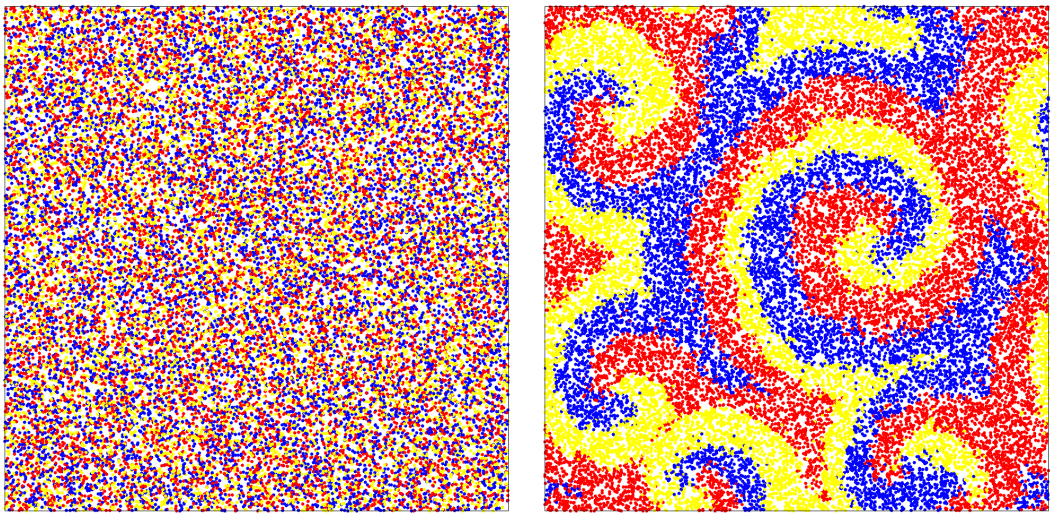

Figura 3.4: Configuração dos 30000 indivíduos no tempo inicial (painel da esquerda) e após $t = 5000$ gerações (painel da direita). Os parâmetros utilizados foram $p_m = 0.5$, $p_p = 0.5$, $l_m = 0.02$ e $l_p = 0.02$.

# Considerações finais

Nesse trabalho realizamos simulações RPS com três espécies no modelo de Lotka-Volterra sem rede. Após o estudo conseguimos mostrar assim como na referência [15] que existe a formação de espirais nas simulações quando há uma densidade de indivíduos alta, nos resultados mostramos que, para 1500 indivíduos não foi observado os padrões de espirais. Para 30000 indivíduos percebemos o padrão de espirais.

Ainda sobre a densidade de indivíduos percebemos que há uma conservação ao longo das simulações. Mostramos a densidade espectral da espécie 1 e constatamos que o mesmo ocorre para as espécies 2 e 3, ou seja, elas possuem aproximadamente a mesma quantidade de gerações em que cada espécie aparece em destaque e nas outras elas possuem densidade de indivíduos semelhantes.

Como perspectiva pretendemos realizar novas simulações e a transformadas de Fourier com outros valores de densidades.

# Apêndice A - Algoritmo Rock-Paper-Scissor (RPS) sem rede

## rps.h

Na linguagem **C** os arquivos que possuem extensões **.h** são chamados arquivos de cabeçalho (ou "headfile"). Esses arquivos tem função de organizar o seu código, neles estão definidas as bibliotecas que serão utilizadas e todas as definições de parâmetros necessários [17, 18].

Segue o código do arquivo **rps.h**:

```
1  //bibliotecas e definições
2
3  #include <stdio.h>
4  #include <stdlib.h>
5  #include <math.h>
6  #include <time.h>
7  #include <gsl/gsl_rng.h>
8
9  #define N 30000
10 #define Ns 3
11 #define t 5000
```

## rps.c

Os arquivos **.c** contém as declarações de todas as funções utilizadas pelo programa, neles podem conter o método numérico que será utilizado e, ainda, a função **main**. O arquivo **rps.c** contém as funções que foram usadas no nosso programa, como a função **main**, a função responsável por gerar os números aleatórios, além de incluir nossa função **output**, responsável de direcionar os valores em arquivos específicos e em seguida armazená-los na pasta **dat**.

Segue o código do arquivo **rps.c**

```c
#include "rps.h"

  struct DATA{
    int s;
    double x;
    double y;
  };

void op(int , struct DATA *);

int main(int argc, char **argv){
  int i, j, k, l;
  int n=0, o;
  double c;
  double theta, d, dx, dy, L;
  double q=0, r=0, u=0;

  struct DATA *p;
  p= (struct DATA*) calloc(N, sizeof(struct DATA));

  //condições iniciais - início
  gsl_rng_default_seed= (argc == 2) ? atoi(argv[1]) : time(NULL);
  gsl_rng *w= gsl_rng_alloc(gsl_rng_taus);

  for(i= 0; i< N; i++){
    p[i].x= gsl_rng_uniform(w);
    p[i].y= gsl_rng_uniform(w);
    p[i].s= gsl_rng_uniform(w)*3+1;
  }
  op(0, p);
  for(i= 0; i< N; i++){
    switch(p[i].s){
      case 1:
        q++;
        break;
      case 2:
        r++;
        break;
      case 3:
```

```
40          u++;
41          break;
42        }
43      }
44      FILE *saida= fopen("dat/prob.dat", "a");
45      fprintf(saida, "%e %e %e\n", (q/N), (r/N), (u/N));
46      fclose(saida);
47      //condições iniciais - fim
48
49      for(k= 0; k< t; k++){
50        l=0;
51        while(l< N){
52          i= gsl_rng_uniform(w)*N;
53          c= gsl_rng_uniform(w);
54          if(c< 0.5){
55            theta= gsl_rng_uniform(w)*2.0*M_PI;
56            p[i].x+= 0.02*cos(theta);
57            if(p[i].x > 1.0){
58              p[i].x-= 1.0;
59            } if(p[i].x < 0.0){
60                p[i].x += 1.0;
61            }
62            p[i].y+= 0.02*sin(theta);
63            if(p[i].y > 1.0){
64            p[i].y-= 1.0;
65            } if(p[i].y < 0.0){
66                p[i].y += 1.0;
67            }
68            l++;
69          }else{
70            o=-1;
71            L= 1.0;
72            for(j= 0; j< N; j++){
73              if(j != i){
74                dx= fabs(p[i].x - p[j].x);
75                dy= fabs(p[i].y - p[j].y);
76                if(dx > 0.5){
77                  dx -= 1.0;
78                }
```

```
79              if(dy > 0.5){
80                dy -= 1.0;
81              }
82              d= sqrt(dx*dx+dy*dy);
83              if(d < L){
84                if((p[i].s-1) == (p[j].s+1)%Ns){
85                  L= d;
86                  o= j;
87                }
88              }
89            }
90          }
91          if(o != -1){
92            if(L < 0.02){
93              p[o].s = p[i].s;
94              l++;
95            }
96          }
97        }
98      }
99
100
101     //densidade de individuos por geração
102     q=0, r=0, u=0;
103     for(i= 0; i< N; i++){
104       switch(p[i].s){
105         case 1:
106           q++;
107           break;
108         case 2:
109           r++;
110           break;
111         case 3:
112           u++;
113           break;
114       }
115     }
116     saida= fopen("dat/prob.dat", "a");
117     fprintf(saida, "%e %e %e\n", (q/N), (r/N), (u/N));
```

```
118       fclose(saida);
119       op(++n, p);
120     }
121     free(p);
122     gsl_rng_free(w);
123     return 0;
124   }
125
126   void op(int n, struct DATA *p){
127     int i, s;
128     FILE *out;
129     char name[100];
130
131     for(s= 0; s< 3; s++){
132       sprintf(name, "dat/p_%d-%d.dat", (s+1), n);
133       out= fopen(name, "w");
134       for(i= 0; i< N; i++){
135         if(p[i].s == (s+1)){
136           fprintf(out, "%e %e\n", p[i].x, p[i].y);
137         }
138       }
139     fclose(out);
140     }
141   }
```

## makefile

O arquivo **makefile** contém instruções para a compilação do programa RPS. Segue o código do arquivo **makefile**

```
1  COMPILER = gcc
2  FLAGS = -Wall -O2 -mtune=native
3  LIB = -lgsl -lgslcblas -lm
4
5  rps:
6    ${COMPILER} ${FLAGS} rps.c ${LIB}
7
8  clean:
9    rm -f dat/*
```

```
10    rm -f a.out
11
12  clean-fig:
13    rm -f plt/*.png plt/*.pdf
```

## dft.c

O arquivo **dft.c** contém as funções que foram utilizadas para realizar a transformada de Fourier discreta dos dados gerados anteriormente.

Segue o código do arquivo **dft.c**

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <math.h>
4   #include <complex.h>
5
6   int main(int argc, char **argv){
7     if(argc != 3){
8       printf("%s nome_do_arquivo_de_dados.dat n_de_linhas\n", argv[0]);
9       exit(1);
10    }
11
12    int t, f, n;
13    double x, y;
14    double         *g_t;
15    double complex *g_f;
16    FILE *arquivo;
17
18    arquivo= fopen(argv[1], "r");
19    n = atoi(argv[2]);
20    g_t= (double         *) calloc(n,   sizeof(double        ));
21    g_f= (double complex *) calloc(n/2, sizeof(double complex));
22    for(t= 0; t< n; t++){
23      if(fscanf(arquivo, "%lf %lf %lf", &g_t[t], &x, &y) != 3){
24        printf("não foi possível ler o arquivo\n");
25        exit(1);
26      }
27    }
28    fclose(arquivo);
```

```
29
30    for(f= 0; f< n/2; f++){
31      for(t= 0; t< n; t++){
32        g_f[f]+= g_t[t]*cexp(-2.0*M_PI*I*f*t/n);
33      }
34      g_f[f]*= 2.0/n;
35    }
36    g_f[0]*= 0.5;
37
38    arquivo= fopen("g_f.dat", "w");
39    for(f= 0; f< n/2; f++){
40      fprintf(arquivo, "%e\n", cabs(g_f[f]*g_f[f]));
41    }
42    fclose(arquivo);
43
44    double *g_i;
45    g_i= (double *) calloc(n, sizeof(double));
46    for(t= 0; t< n; t++){
47      for(f= 0; f< n/2; f++){
48        if(f < 6){
49          g_i[t]+= creal(g_f[f]*cexp(2.0*M_PI*I*f*t/n));
50        }
51      }
52    }
53
54    arquivo= fopen("g_i.dat", "w");
55    for(t= 0; t< n; t++){
56      fprintf(arquivo, "%e\n", g_i[t]);
57    }
58    fclose(arquivo);
59
60    free(g_i);
61    free(g_t);
62    free(g_f);
63    return 0;
64  }
```

# Referências Bibliográficas


[1] B. Sinervo and C. M. Lively, "The rock-paper-scissors game and the evolution of alternative male strategies," *Nature*, vol. 380, p. 240, Mar 21 1996.

[2] "Vetores por vecteezy." Disponível em: <https://pt.vecteezy.com/vetor-gratis/humano> Acessado em: 05 de Jan. de 2023.

[3] "Pedra, papel, tesoura, lagarto e spock." Disponível em: <https://suricatodigital.com/sheldon-cooper-pedra-papel-tesoura-lagarto-spock> Acessado em: 15 de Fev. de 2023.

[4] "Jogo de dama tamanho master geriatria." Disponível em: <https://br.pinterest.com/pin/576249714793992810/> Acessado em: 15 de Fev. de 2023.

[5] J. V. O. Silva, "Estudos da dinâmica de população no modelo rps fora do equilíbrio natural," 2023.

[6] R. Bose, "Effect of swarming on biodiversity in non-symmetric rock– paper–scissor game," *IET Systems Biology*, vol. 4, pp. 177–184, 2009.

[7] L. W. B. J. B. C. Jackson, "Alleopathy and spatial competition among coral reel invertebrates," *Nature*, vol. 72-12, pp. 5160–5163, (1975).

[8] R. D. Bini, "Estudo da biodiversidade utilizando o jogo pedra-papel-tesoura," 2017.

[9] M. W. F. B. Kerr, M. A. Riley and B. J. M. Bohannan, "Local dispersal promotes biodiversity in a real-life game of rock-paper-scissors," *Nature*, vol. 418, p. 171, (2002).

[10] M. M. T. Reichenbach and E. Frey, "Mobility promotes and jeopardizes biodiversity in rock-paper-scissors games," *Nature*, vol. 448, p. 1046, (2007).

[11] V. Volterra, "Fluctuations in the abundance of a species considered mathematically," *Nature*, vol. 118, pp. 558–560, (1926).

[12] A. J. Lotka, "Elements of physical biology," *Williams & Wilkins*, (1925).

[13] L. C. S. Lacerda, "O modelo de lotka-volterra e aplicações," 2015.

[14] C. Scherer, *Métodos Computacionais da Física*. Editora Livraria da Física, 2005.

[15] P. P. Avelino, D. Bazeia, L. Losano, J. Menezes, and B. F. de Oliveira, "Spiral patterns and biodiversity in lattice-free lotka-volterra models," 2017.

[16] M. Peltomäki and M. Alava, "Three- and four-state rock-paper-scissors games with diffusion," *Phys. Rev. E*, vol. 78, p. 031906, (2008).

[17] L. A. P. L. Jr., "Usando arquivos .h." Disponível em: <http://www.ppgia.pucpr.br/~laplima/ensino/tap/contents/02_arquivosh.html> Acesso em: 18 Ago. 2018.

[18] E. S. Dobay, *Programação em C*. IF-Universidade de São Paulo, (2012). Disponível em: <http://www.ppgia.pucpr.br/~laplima/ensino/tap/contents/02_arquivosh.html> Acesso em: 14 Jul. 2018.