**Título: Modelagem do Sistema Lotka-Volterra e Suas Aplicações em Ecologia Computacional com Python**

Diogo Takamori Barbosa

**Resumo:**

O modelo Lotka-Volterra, desenvolvido por Alfred Lotka e Vito Volterra no início do século XX, é uma ferramenta fundamental na ecologia que descreve as dinâmicas das populações de presas e predadores em um ecossistema. Este modelo oferece insights valiosos sobre as interações ecológicas e tem inúmeras aplicações em áreas que variam de ecologia a economia, epidemiologia e engenharia. Este artigo de pós-graduação explora a teoria por trás do modelo Lotka-Volterra, apresenta exemplos de modelos do mundo real e demonstra sua implementação em Python para análise e simulação computacional.

Palavras-Chave: Lotka-Volterra, Matemática, Epidemiologia, Presa-Predador, Economia.

**Introdução:**

O modelo Lotka-Volterra é uma ferramenta matemática que descreve a dinâmica de duas populações em um ecossistema: as presas e os predadores. As populações interagem entre si, e essas interações têm um impacto significativo na dinâmica das populações ao longo do tempo. As equações diferenciais que compõem o modelo permitem entender como a abundância de presas e predadores evolui e oscila no decorrer das estações e anos.

O modelo Lotka-Volterra é um exemplo clássico de um sistema dinâmico que pode ser aplicado em uma ampla gama de contextos, não apenas na ecologia, mas também em campos como economia, epidemiologia e engenharia. Essa versatilidade torna o entendimento do modelo e sua implementação prática em linguagens de programação, como Python, uma habilidade valiosa.

1. **Modelos Reais e Suas Abordagens:**

Neste artigo, exploraremos várias aplicações do modelo Lotka-Volterra em cenários do mundo real. Alguns exemplos notáveis incluem:

**Dinâmica de Populações de Presas e Predadores**: O modelo Lotka-Volterra tem sido usado para entender as oscilações nas populações de lebres e linces, lobos e alces, e outros exemplos de predadores e suas presas na natureza. Esses estudos têm contribuído para a conservação de espécies em risco.

**Economia**: O modelo encontra aplicação na economia para descrever a competição entre empresas e consumidores. Pode ser usado para analisar a relação entre o estoque de recursos naturais, como peixes, e a pesca comercial.

**Epidemiologia**: Na epidemiologia, o modelo pode ser usado para modelar a propagação de doenças infecciosas. As presas podem representar indivíduos suscetíveis, e os predadores podem representar indivíduos infectados.

Cada uma dessas aplicações requer adaptações do modelo Lotka-Volterra, levando em consideração as dinâmicas específicas das populações e das interações em questão. Esses modelos adaptados são fundamentais para entender as dinâmicas de sistemas complexos.

1. **Abordagem Computacional com Python:**

Uma das vantagens do modelo Lotka-Volterra é a sua implementação computacional. Python é uma linguagem de programação amplamente utilizada para modelagem e simulação de sistemas ecológicos. As vantagens da implementação computacional do modelo Lotka-Volterra incluem:

Facilidade de Implementação: Python é conhecido por sua simplicidade e legibilidade, o que torna a implementação das equações Lotka-Volterra mais acessível, mesmo para aqueles que não têm uma formação matemática avançada.

Flexibilidade: Python oferece uma ampla gama de bibliotecas e ferramentas para análise de dados, simulação e visualização. Isso permite que os pesquisadores ajustem e personalizem o modelo para se adequar a cenários específicos.

Visualização Gráfica: A capacidade de criar gráficos e visualizações é uma das principais vantagens do uso de Python. Isso permite que os pesquisadores observem as oscilações das populações de presas e predadores ao longo do tempo, tornando a interpretação dos resultados mais clara.

Análise de Sensibilidade: Com o uso de Python, é possível realizar análises de sensibilidade para entender como variações nos parâmetros afetam as dinâmicas do sistema. Isso é fundamental para prever o comportamento de populações em resposta a diferentes condições.

Integração com Dados Reais: Python permite a integração de dados reais, como dados de campo e séries temporais de populações, para validar e ajustar os modelos Lotka-Volterra. Isso torna o modelo mais robusto e aplicável a cenários do mundo real.

1. **Modelagem Matemática usando Lotka-Volterra.**
   1. **Modelo Matemático Lotka-Volterra para Dinâmica de Populações de Presas e Predadores**

O modelo Lotka-Volterra para a dinâmica de populações de presas (P) e predadores (Pr) é composto por duas equações diferenciais:

Equação das Presas:

dP/dt​=rp​P−aPPr​

Equação dos Predadores:

dP/dtr​​=bPPr​−dPr​  
​

dP/dt​ é a taxa de variação da população de presas.

P é a população de presas.

rp​ é a taxa intrínseca de crescimento das presas.

a é a taxa de predação (eficiência na conversão de presas em predadores).

Pr​ é a população de predadores.

dtdPr​​ é a taxa de variação da população de predadores.

Pr​ é a população de predadores.

b é a taxa de crescimento da população de predadores devido à disponibilidade de presas.

d é a taxa de mortalidade dos predadores.

Análise Matemática:

A análise matemática envolve a determinação de pontos de equilíbrio e sua estabilidade. Para simplificar, vamos considerar valores de parâmetros fictícios:

rp​=0.1 (taxa de crescimento intrínseca das presas)

a=0.02 (taxa de predação)

b=0.1 (taxa de crescimento dos predadores)

d=0.01 (taxa de mortalidade dos predadores)

Ponto de Equilíbrio:

Para encontrar o ponto de equilíbrio, igualamos as taxas de mudança a zero:

Para as presas (dP/dt​=0):

rp​P−aPPr​=0

Isso nos leva a dois casos:

a) P=0 (extinção das presas)

b) Pr​=arp​​ (ponto de equilíbrio não trivial)

Para os predadores (dPr/dt​​=0):

bPPr​−dPr​=0

Isso nos leva a dois casos:

a) Pr​=0 (extinção dos predadores)

b) P=bd​ (ponto de equilíbrio não trivial)

Vamos usar o ponto de equilíbrio não trivial para a simulação em Python:

**Simulação Computacional em Python:**

O Código em Python demonstra a evolução das populações de presas e predadores ao longo do tempo e plota um gráfico para visualização:

import numpy as np

import matplotlib.pyplot as plt

# Parâmetros do modelo Lotka-Volterra

alpha = 0.1   # Taxa de crescimento dos coelhos na ausência de predadores

beta = 0.02   # Taxa de predação dos coelhos pelas raposas

gamma = 0.1   # Taxa de morte das raposas na ausência de presas

delta = 0.01  # Taxa de crescimento das raposas devido à predação dos coelhos

# Condições iniciais

x0 = 40  # População inicial de coelhos

y0 = 9   # População inicial de raposas

# Configuração do tempo

T = 200

dt = 0.1

num\_steps = int(T / dt)

# Arrays para armazenar resultados

x\_values = np.zeros(num\_steps)

y\_values = np.zeros(num\_steps)

time\_values = np.zeros(num\_steps)

# Inicialização das populações iniciais

x = x0

y = y0

# Simulação do modelo Lotka-Volterra usando o método de Euler

for i in range(num\_steps):

    x\_values[i] = x

    y\_values[i] = y

    time\_values[i] = i \* dt

    # Equações de Lotka-Volterra usando o método de Euler

    dx = dt \* (alpha \* x - beta \* x \* y)

    dy = dt \* (delta \* x \* y - gamma \* y)

    # Atualização das populações

    x += dx

    y += dy

# Plotagem do gráfico bidimensional

plt.figure(figsize=(10, 6))

plt.plot(x\_values, y\_values, label='Dinâmica Populacional')

plt.title('Modelo Lotka-Volterra: Sistema Dinâmico Bidimensional')

plt.xlabel('População de Coelhos')

plt.ylabel('População de Raposas')

plt.legend()

plt.grid(True)

plt.show()

Comentários sobre o código:

Parâmetros do Modelo: Os parâmetros alpha, beta, gamma e delta representam as taxas de crescimento e predação das populações de coelhos e raposas no modelo Lotka-Volterra.

As populações iniciais de coelhos (x0) e raposas (y0) são definidas para iniciar a simulação.

Define o tempo total da simulação (T), o intervalo de tempo (dt) e calcula o número total de passos de tempo (num\_steps).

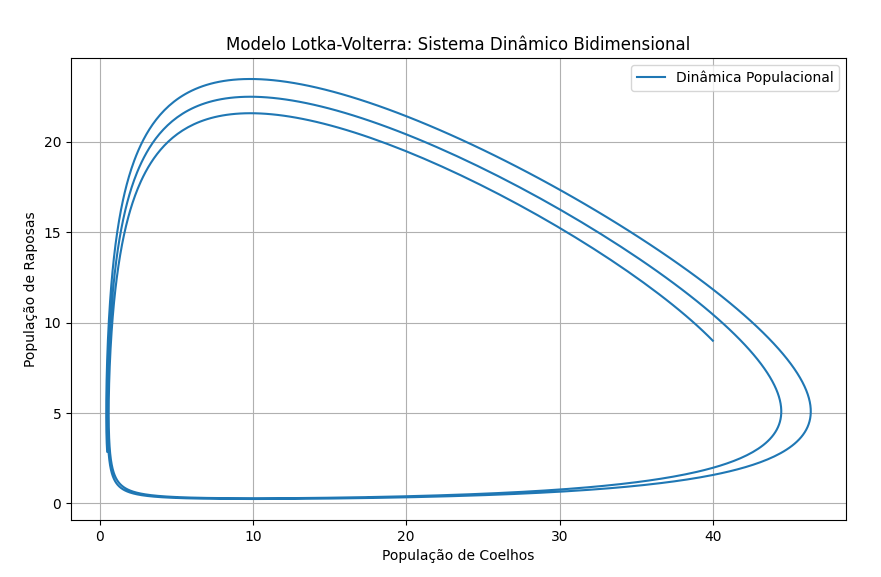
Arrays para Armazenar Resultados: Arrays x\_values, y\_values e time\_values são inicializados para armazenar os resultados da simulação.

Inicialização das Populações Iniciais: As populações iniciais são atribuídas às variáveis x e y.

Simulação usando o Método de Euler: As equações de Lotka-Volterra são resolvidas usando o método de Euler para cada passo de tempo.

Atualização das Populações: As populações de coelhos e raposas são atualizadas com base nas derivadas calculadas.

Plotagem do Gráfico Bidimensional: Um gráfico bidimensional é criado para visualizar a dinâmica populacional ao longo do tempo, mostrando as populações de coelhos e raposas.

Este código Python realiza a simulação da dinâmica de populações de presas e predadores ao longo do tempo com base no modelo Lotka-Volterra e plota um gráfico que demonstra as oscilações características entre as duas populações.

**Calculo Matemático do Código apresentado (Latex)**

O modelo Lotka-Volterra descreve a interação entre duas espécies, no seu caso, coelhos (x) e raposas (y). As equações diferenciais do modelo são dadas por:

\[

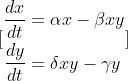
\begin{align\*}

\frac{dx}{dt} &= \alpha x - \beta xy \\

\frac{dy}{dt} &= \delta xy - \gamma y

\end{align\*}

\]



onde:

- \(x\) é a população de coelhos,

- \(y\) é a população de raposas,

- \(\alpha\) é a taxa de crescimento dos coelhos na ausência de predadores,

- \(\beta\) é a taxa de predação dos coelhos pelas raposas,

- \(\gamma\) é a taxa de morte das raposas na ausência de presas,

- \(\delta\) é a taxa de crescimento das raposas devido à predação dos coelhos.

A simulação numérica do modelo usando o método de Euler é feita pelas seguintes equações de atualização:

\[

x\_{i+1} &= x\_i + \Delta t \cdot (\alpha x\_i - \beta x\_i y\_i) \\

y\_{i+1} &= y\_i + \Delta t \cdot (\delta x\_i y\_i - \gamma y\_i)

\]

onde \(i\) é o índice da iteração, \(\Delta t\) é o intervalo de tempo e \(x\_i\), \(y\_i\) são as populações de coelhos e raposas no passo \(i\), respectivamente.

As etapas do cálculo matemático para o código Python apresentado podem ser apresentadas pelo seguinte exemplo matemático:

* Definição das Equações Diferenciais do Modelo:

\[

\begin{align\*}

\frac{dx}{dt} &= \alpha x - \beta xy \\

\frac{dy}{dt} &= \delta xy - \gamma y

\end{align\*}

\]

* Inicialização das Variáveis e Parâmetros:

- \( \alpha = 0.1 \), \( \beta = 0.02 \), \( \gamma = 0.1 \), \( \delta = 0.01 \)

- \( x\_0 = 40 \), \( y\_0 = 9 \)

- \( T = 200 \), \( \Delta t = 0.1 \)

- Número de passos: \( \text{{num\\_steps}} = \frac{T}{\Delta t} \)

* Inicialização das Arrays e Condições Iniciais:

- \( x\_{\text{{values}}}[0] = x\_0 \), \( y\_{\text{{values}}}[0] = y\_0 \), \( \text{{time\\_values}}[0] = 0 \)

* Simulação do Modelo usando o Método de Euler:

\[

x\_{i+1} &= x\_i + \Delta t \cdot (\alpha x\_i - \beta x\_i y\_i) \\

y\_{i+1} &= y\_i + \Delta t \cdot (\delta x\_i y\_i - \gamma y\_i)

\]

Para cada iteração \(i\):

- Calcular \(dx = \Delta t \cdot (\alpha x - \beta x y)\)

- Calcular \(dy = \Delta t \cdot (\delta x y - \gamma y)\)

- Atualizar \(x\) e \(y\) usando \(x += dx\) e \(y += dy\)

Plotagem do Gráfico Bidimensional:

- População de Coelhos (\(x\)) no eixo x

- População de Raposas (\(y\)) no eixo y

- Título: "Modelo Lotka-Volterra: Sistema Dinâmico Bidimensional"

- Rótulos dos eixos: "População de Coelhos", "População de Raposas"

- Legenda: "Dinâmica Populacional"

- Exibir o gráfico.

**Gráfico em relação ao tempo**

Podemos usar outro código para plotar um gráfico em função do tempo modificando a forma de plotar o gráfico. Para esse caso usaremos o seguinte código em Python:

import numpy as np

import matplotlib.pyplot as plt

# Parâmetros do modelo Lotka-Volterra

alpha = 0.1   # Taxa de crescimento dos coelhos na ausência de predadores

beta = 0.02   # Taxa de predação dos coelhos pelas raposas

gamma = 0.1   # Taxa de morte das raposas na ausência de presas

delta = 0.01  # Taxa de crescimento das raposas devido à predação dos coelhos

# Condições iniciais

x0 = 40  # População inicial de coelhos

y0 = 9   # População inicial de raposas

# Tempo

T = 200

dt = 0.1

num\_steps = int(T / dt)

# Arrays para armazenar resultados

x\_values = np.zeros(num\_steps)

y\_values = np.zeros(num\_steps)

time\_values = np.zeros(num\_steps)

# Inicialização

x = x0

y = y0

# Simulação

for i in range(num\_steps):

    x\_values[i] = x

    y\_values[i] = y

    time\_values[i] = i \* dt

    # Equações de Lotka-Volterra usando o método de Euler

    dx = dt \* (alpha \* x - beta \* x \* y)

    dy = dt \* (delta \* x \* y - gamma \* y)

    x += dx

    y += dy

# Plotagem

plt.figure(figsize=(10, 6))

plt.plot(time\_values, x\_values, label='Coelhos')

plt.plot(time\_values, y\_values, label='Raposas')

plt.title('Dinâmica Populacional: Modelo Lotka-Volterra')

plt.xlabel('Tempo')

plt.ylabel('População')

plt.legend()

plt.grid(True)

plt.show()

Comentários:

Parâmetros do Modelo: Os parâmetros (alpha, beta, gamma, delta) são as taxas que influenciam as interações entre as populações de coelhos e raposas no modelo Lotka-Volterra.

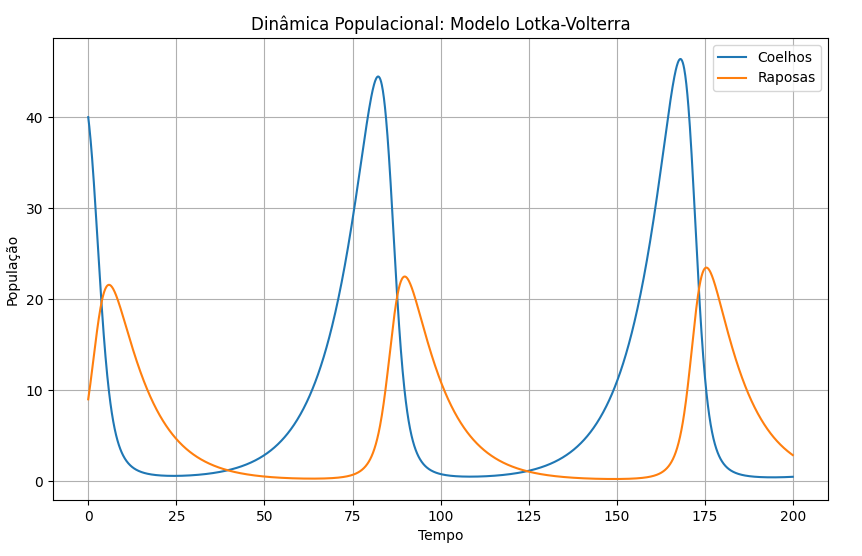
Condições Iniciais: As condições iniciais (x0 e y0) representam as populações iniciais de coelhos e raposas.

Tempo e Passos de Simulação: O tempo total de simulação (T) e o tamanho do passo de simulação (dt) são especificados, e o número de passos de simulação (num\_steps) é calculado com base nesses valores.

Arrays para Armazenar Resultados: Arrays (x\_values, y\_values, time\_values) são inicializados para armazenar os resultados da simulação ao longo do tempo.

Inicialização e Simulação: As populações iniciais são atribuídas a variáveis (x e y), e a simulação é realizada usando o método de Euler para iterativamente calcular as populações ao longo do tempo.

Plotagem: A dinâmica populacional é plotada ao longo do tempo para as populações de coelhos e raposas, com rótulos e legendas para melhor compreensão.



Para a Próxima análise será criado campos vetoriais para identificação das tendências da movimentação populacional. Campos vetoriais são representações matemáticas que associam a cada ponto em um espaço multidimensional um vetor. A ideia principal por trás de campos vetoriais é atribuir um vetor a cada ponto em um espaço. Na aplicação do modelo Lotka-Volterra, os campos vetoriais são representados pelas derivadas das equações diferenciais que descrevem a dinâmica populacional de coelhos e raposas.

As derivadas dx e dy nas equações de Lotka-Volterra representam os campos vetoriais implicitamente. Em cada ponto (x,y), essas derivadas indicam as taxas de variação das populações de coelhos e raposas. A direção das setas (ou vetores) no gráfico bidimensional é determinada pelas derivadas dx e dy em cada ponto. As setas apontam na direção em que as populações estão mudando mais rapidamente. A magnitude das setas indica a taxa de variação.

Pontos de equilíbrio do sistema são encontrados onde dx=dy=0. Esses pontos representam situações em que as populações não estão mudando. A estabilidade desses pontos pode ser avaliada pela direção das setas ao redor deles. Se as setas convergirem para um ponto, ele é estável; se divergirem, é instável. O padrão de setas pode indicar a presença de ciclos limite, onde as populações oscilam entre valores mínimos e máximos. Visualmente, o gráfico bidimensional mostra como as populações de coelhos e raposas evoluem ao longo do tempo, proporcionando uma visão intuitiva da dinâmica populacional. A análise dos campos vetoriais é crucial para entender a dinâmica de sistemas complexos. No caso de Lotka-Volterra, ela fornece insights sobre as interações predador-presa e a evolução das populações ao longo do tempo.

Para demonstrar o uso de campos vetorias no gráfico bidimensional usamos o seguinte código em python

# Importando bibliotecas necessárias

import numpy as np

import matplotlib.pyplot as plt

# Parâmetros do modelo Lotka-Volterra

alpha = 0.1   # Taxa de crescimento dos coelhos na ausência de predadores

beta = 0.02   # Taxa de predação dos coelhos pelas raposas

gamma = 0.1   # Taxa de morte das raposas na ausência de presas

delta = 0.01  # Taxa de crescimento das raposas devido à predação dos coelhos

# Condições iniciais

x0 = 40  # População inicial de coelhos

y0 = 9   # População inicial de raposas

# Configuração do tempo

T = 200

dt = 0.1

num\_steps = int(T / dt)

# Arrays para armazenar resultados

x\_values = np.zeros(num\_steps)

y\_values = np.zeros(num\_steps)

time\_values = np.zeros(num\_steps)

# Inicialização das populações iniciais

x = x0

y = y0

# Simulação do modelo Lotka-Volterra usando o método de Euler

for i in range(num\_steps):

    # Armazenando valores atuais das populações e tempo

    x\_values[i] = x

    y\_values[i] = y

    time\_values[i] = i \* dt

    # Equações de Lotka-Volterra usando o método de Euler

    dx = dt \* (alpha \* x - beta \* x \* y)

    dy = dt \* (delta \* x \* y - gamma \* y)

    # Atualização das populações usando as derivadas calculadas

    x += dx

    y += dy

# Configuração do plano (x, y) para os campos vetoriais

x\_range = np.linspace(min(x\_values), max(x\_values), 20)

y\_range = np.linspace(min(y\_values), max(y\_values), 20)

x\_derivatives, y\_derivatives = np.meshgrid(x\_range, y\_range)

# Calculando as derivadas nos pontos do plano para os campos vetoriais

dxdt = dt \* (alpha \* x\_derivatives - beta \* x\_derivatives \* y\_derivatives)

dydt = dt \* (delta \* x\_derivatives \* y\_derivatives - gamma \* y\_derivatives)

# Normalizando os vetores para melhor visualização

magnitude = np.sqrt(dxdt\*\*2 + dydt\*\*2)

dxdt /= magnitude

dydt /= magnitude

# Plotagem dos campos vetoriais

plt.figure(figsize=(12, 8))

plt.plot(x\_values, y\_values, label='Dinâmica Populacional')

plt.quiver(x\_derivatives, y\_derivatives, dxdt, dydt, scale=45, color='red', label='Campos Vetoriais')

plt.title('Modelo Lotka-Volterra: Campos Vetoriais')

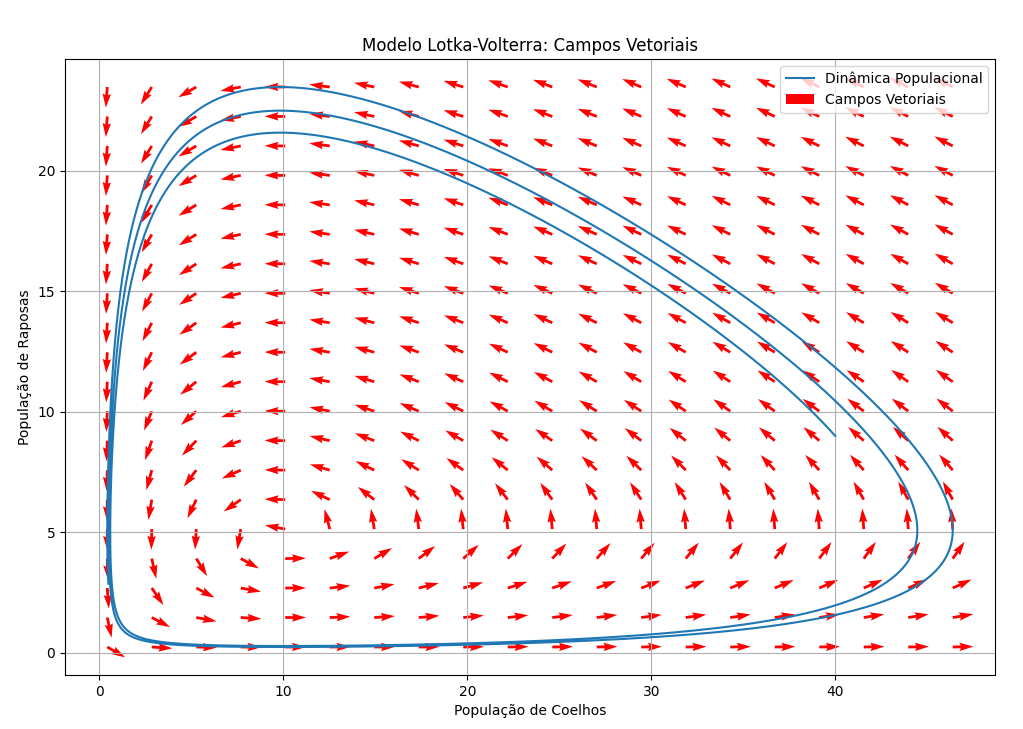
plt.xlabel('População de Coelhos')

plt.ylabel('População de Raposas')

plt.legend()

plt.grid(True)

plt.show()

Esse código cria um gráfico que mostra a dinâmica populacional ao longo do tempo e adiciona campos vetoriais para indicar as tendências do sistema. A plotagem do gráfico fica da seguinte forma:

O gráfico mostrará como as populações de coelhos e raposas evoluem ao longo do tempo com base nas equações Lotka-Volterra. Você pode observar padrões como ciclos populacionais, onde as populações de coelhos e raposas oscilam. Esses ciclos são características do comportamento predador-presa descrito pelo modelo. Vale ressaltar que essas simulações são simplificações e que as condições iniciais e os parâmetros escolhidos podem levar a diferentes resultados. Neste código, os vetores de tendência são adicionados ao gráfico usando a função quiver do Matplotlib, que adiciona setas para indicar a direção e magnitude dos vetores. Os vetores de tendência são calculados a partir das derivadas das equações Lotka-Volterra em cada ponto da simulação.

Demonstração matemática para exemplificação do código em python

Campos Vetoriais:

Os campos vetoriais são calculados nos pontos do plano (x, y). Vamos denotar \(u = \frac{dx}{dt}\) e \(v = \frac{dy}{dt}\). As derivadas são dadas por:

\[

\begin{align\*}

u &= \Delta t \cdot (\alpha x - \beta xy) \\

v &= \Delta t \cdot (\delta xy - \gamma y)

\end{align\*}

\]

Normalização dos Vetores:

Os vetores são normalizados para melhor visualização, dividindo cada componente pela magnitude:

\[

\begin{align\*}

\text{{magnitude}} &= \sqrt{u^2 + v^2} \\

u &= \frac{u}{\text{{magnitude}}} \\

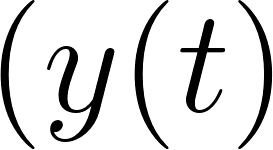
v &= \frac{v}{\text{{magnitude}}}

\end{align\*}

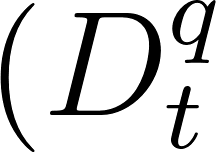
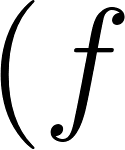
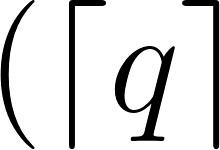
\]

* 1. **Modelo Matemático Lotka-Volterra para Dinâmica de Populações de Presas e Predadores com Cálculo Fracionário**

Equações Diferenciais Fracionárias (EDFs) são uma generalização das Equações Diferenciais Ordinárias (EDOs), onde a ordem da derivada ou integral é um número não inteiro. Em outras palavras, elas envolvem derivadas ou integrais de ordens fracionárias. A EDF mais comum é a Equação Diferencial Fracionária (EDF) de Caputo, que é uma generalização da derivada comum para ordens não inteiras.

A forma geral de uma EDF de Caputo de ordem [](https://www.codecogs.com/eqnedit.php?latex=(%20q%20#1)) para uma função [](https://www.codecogs.com/eqnedit.php?latex=(%20y(t)%20#1)) é dada por:

[](https://www.codecogs.com/eqnedit.php?latex=%20D%5Eq_%7Bt%7D%20y(t)%20%3D%20f(t%2C%20y(t)%2C%20D%5E%7B%5Clceil%20q%20%5Crceil%20-%201%7D_%7Bt%7D%20y(t)%2C%20D%5E%7B%5Clceil%20q%20%5Crceil%20-%202%7D_%7Bt%7D%20y(t)%2C%20%5Cldots%2C%20y'(t)%2C%20y(t))%20#1)

onde [](https://www.codecogs.com/eqnedit.php?latex=(%20D%5Eq_%7Bt%7D%20#1)) representa a derivada de ordem fracionária de Caputo, [](https://www.codecogs.com/eqnedit.php?latex=(%20f%20#1)) é uma função que descreve a dinâmica do sistema, e [](https://www.codecogs.com/eqnedit.php?latex=(%5Clceil%20q%20%5Crceil#1)) denota o arredondamento para cima de [](https://www.codecogs.com/eqnedit.php?latex=(%20q%20#1)) para o inteiro mais próximo.

As EDFs podem modelar fenômenos complexos em sistemas físicos, biológicos e outros, onde comportamentos não lineares e memória de longo prazo são relevantes. O cálculo fracionário oferece uma ferramenta matemática poderosa para descrever esses fenômenos.

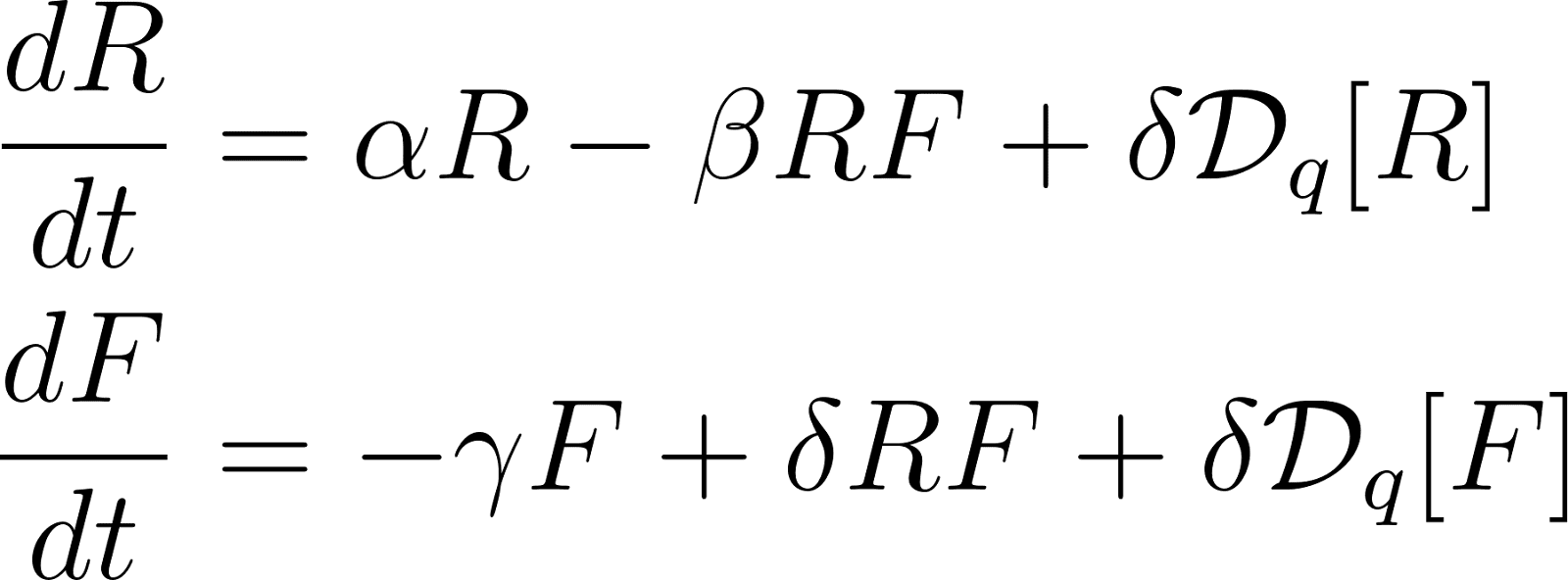
Para resolver EDFs numericamente, podem ser usados métodos específicos, como o método de diferenças finitas fracionárias, a transformada de Laplace fracionária ou outros métodos adaptativos.

Lidar com Equações Diferenciais Fracionárias (EDFs) pode ser mais complexo do que lidar com EDOs tradicionais. Em Python pode-se recorrer a outras abordagens e bibliotecas especializadas para lidar com cálculos fracionários. Vamos apresentar uma abordagem geral usando o método de diferenças finitas para a integração numérica de EDFs que permitem uma abordagem matemática para o modelo Lotka-Volterra.

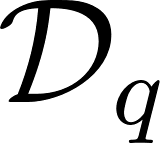
A resolução de um modelo Lotka-Volterra por meio de Equações Diferenciais Fracionárias (EDFs) pode ser feita usando métodos específicos para cálculo fracionário. Vamos apresentar uma abordagem usando o método de diferenças finitas fracionárias.

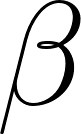
O modelo reformula Lotka-Volterra em termos de EDFs. O sistema de EDFs de Caputo para o modelo Lotka-Volterra:

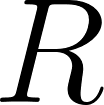
As equações Lotka-Volterra com derivadas fracionárias são representadas por:

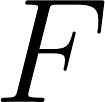
[](https://www.codecogs.com/eqnedit.php?latex=%5Cbegin%7Balign*%7D%5Cfrac%7BdR%7D%7Bdt%7D%20%26%3D%20%5Calpha%20R%20-%20%5Cbeta%20R%20F%20%2B%20%5Cdelta%20%5Cmathcal%7BD%7D_q%5BR%5D%20%5C%5C%5Cfrac%7BdF%7D%7Bdt%7D%20%26%3D%20-%5Cgamma%20F%20%2B%20%5Cdelta%20R%20F%20%2B%20%5Cdelta%20%5Cmathcal%7BD%7D_q%5BF%5D%5Cend%7Balign*%7D#1)

onde:

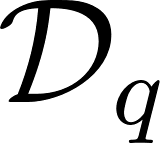
- [](https://www.codecogs.com/eqnedit.php?latex=%5Cmathcal%7BD%7D_q#1) representa a derivada fracionária,

- [](https://www.codecogs.com/eqnedit.php?latex=%5Calpha#1), [](https://www.codecogs.com/eqnedit.php?latex=%5Cbeta#1), [](https://www.codecogs.com/eqnedit.php?latex=%5Cgamma#1), [https://lh7-us.googleusercontent.com/JY4JLw5S3IkQH_U7y1dbsLV3hfPMjPcxNr-xd8Yv68DZAFJBizs1PwG4CZdZAbgMuEb6N7mNmA51tlWJResVzPWPeE2S4SqtcKSwUvzRyyNX7Km2DZmlCRS-qT4mr-qgzgThd2o4fiDmBOVTwSF2Ac4](https://www.codecogs.com/eqnedit.php?latex=%5Cdelta#1) são parâmetros do modelo,

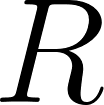
- [](https://www.codecogs.com/eqnedit.php?latex=R#1) é a população de coelhos (presas),

- [](https://www.codecogs.com/eqnedit.php?latex=F#1) é a população de raposas (predadores),

- [https://lh7-us.googleusercontent.com/OBNKDYe665uU0wVOri00ZWAFo0sXVDi1rAZ3Xjc3nSaoUHZ5WGqLjCTsPduZEMh4XiL3SSGUNfUtgO5GzIt3AwVZWz-hPmY6Vaj9X2zGunsvjP8VVSZFxJP2Z5uIJnk3CYexIFCDzqaKZrvEubnXGrM](https://www.codecogs.com/eqnedit.php?latex=q#1) é a ordem fracionária para a derivada.

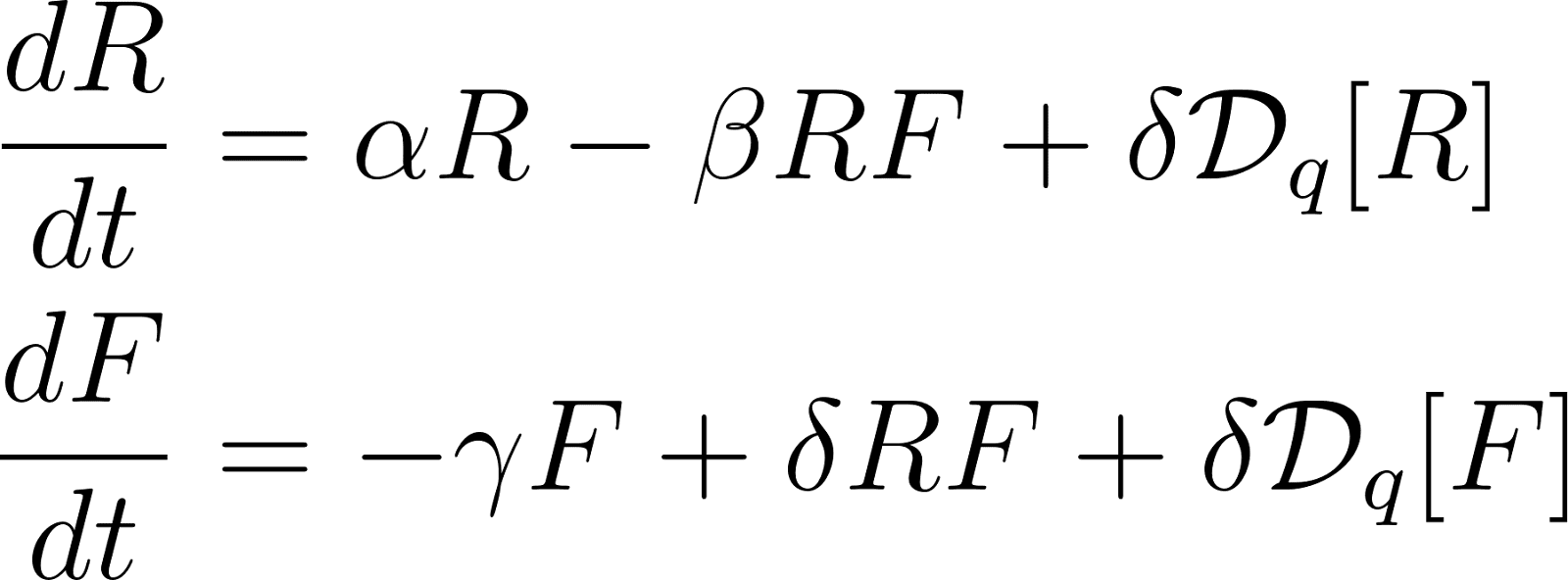
O método de diferenças finitas fracionárias é utilizado para discretizar as derivadas fracionárias. A fórmula para a derivada fracionária [](https://www.codecogs.com/eqnedit.php?latex=%5Cmathcal%7BD%7D_q#1) é dada por:

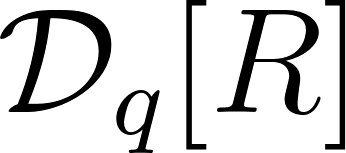
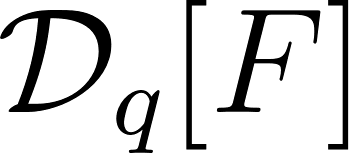
[](https://www.codecogs.com/eqnedit.php?latex=%5Cmathcal%7BD%7D_q%5BR%5D%20%3D%20(1%20-%20q)%20%5Ccdot%20R_i%20%2B%20q%20%5Ccdot%20R_%7Bi-1%7D#1)

Essa fórmula é usada para aproximar a derivada fracionária de [](https://www.codecogs.com/eqnedit.php?latex=R#1) em relação ao tempo.

Dentro do loop, as seguintes etapas são realizadas para cada iteração:

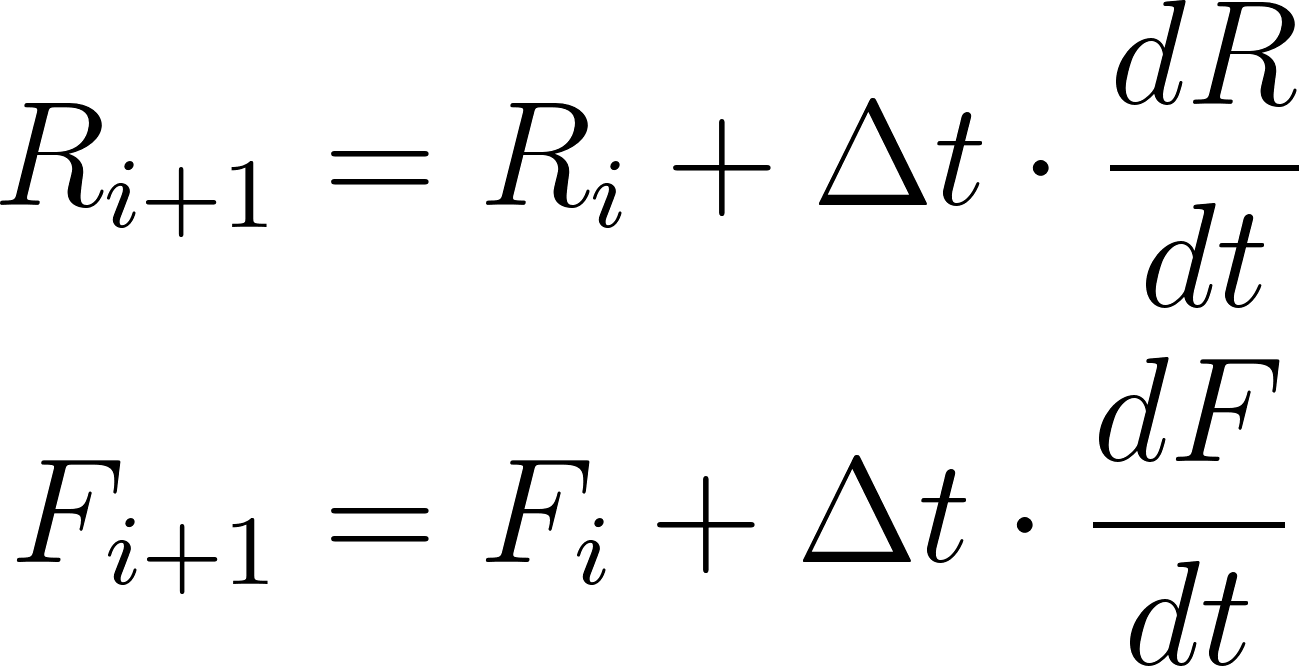
Cálculo das Derivadas

[](https://www.codecogs.com/eqnedit.php?latex=%20%20%20%5Cbegin%7Balign*%7D%20%20%20%5Cfrac%7BdR%7D%7Bdt%7D%20%26%3D%20%5Calpha%20R%20-%20%5Cbeta%20R%20F%20%2B%20%5Cdelta%20%5Cmathcal%7BD%7D_q%5BR%5D%20%5C%5C%5C%5C%20%20%20%5Cfrac%7BdF%7D%7Bdt%7D%20%26%3D%20-%5Cgamma%20F%20%2B%20%5Cdelta%20R%20F%20%2B%20%5Cdelta%20%5Cmathcal%7BD%7D_q%5BF%5D%20%20%20%5Cend%7Balign*%7D%20%20%20#1)

   Aqui, [](https://www.codecogs.com/eqnedit.php?latex=%5Cmathcal%7BD%7D_q%5BR%5D#1) e [](https://www.codecogs.com/eqnedit.php?latex=%5Cmathcal%7BD%7D_q%5BF%5D#1) são calculadas usando a fórmula da diferença finita fracionária.

Atualização das Populações

   As populações de coelhos e raposas são atualizadas usando o método de diferenças finitas fracionárias:

[](https://www.codecogs.com/eqnedit.php?latex=%20%20%20%5Cbegin%7Balign*%7D%20%20%20R_%7Bi%2B1%7D%20%26%3D%20R_i%20%2B%20%5CDelta%20t%20%5Ccdot%20%5Cfrac%7BdR%7D%7Bdt%7D%20%5C%5C%5C%5C%20%20%20F_%7Bi%2B1%7D%20%26%3D%20F_i%20%2B%20%5CDelta%20t%20%5Ccdot%20%5Cfrac%7BdF%7D%7Bdt%7D%20%20%20%5Cend%7Balign*%7D%20%20%20#1)

O código em Python fica da seguinte forma:

import numpy as np

import matplotlib.pyplot as plt

def fractional\_difference(y, alpha, dt):

    """

    Calcula a diferença fracionária usando o método de diferenças finitas fracionárias.

    Parâmetros:

    - y: Lista contendo os valores da série temporal.

    - alpha: Ordem fracionária para a derivada.

    - dt: Incremento de tempo.

    Retorna:

    - Valor fracionário calculado.

    """

    if len(y) >= 2:

        # Se houver pelo menos dois elementos em y, calcula a diferença fracionária.

        return (1 - alpha) \* y[-1] + alpha \* y[-2]

    else:

        # Se houver menos de dois elementos, retorna o último elemento.

        return y[-1]

# Parâmetros do modelo Lotka-Volterra

alpha = 0.1   # Taxa de crescimento das presas na ausência de predadores

beta = 0.02   # Taxa de predação (interação entre presas e predadores)

gamma = 0.1   # Taxa de diminuição dos predadores na ausência de presas

delta = 0.01  # Taxa de crescimento dos predadores em função das presas

q = 0.5       # Ordem fracionária para a derivada (pode ser ajustada conforme necessário)

# Condições iniciais

R0 = 40  # População inicial de coelhos (presas)

F0 = 9   # População inicial de raposas (predadores)

# Configuração do tempo

t\_max = 200  # Tempo máximo de simulação

dt = 0.1     # Incremento de tempo (passo)

time\_points = np.arange(0, t\_max, dt)  # Lista de pontos temporais

# Inicialização das populações

R = np.zeros(len(time\_points))  # Lista para armazenar a população de coelhos

F = np.zeros(len(time\_points))  # Lista para armazenar a população de raposas

R[0] = R0  # População inicial de coelhos

F[0] = F0  # População inicial de raposas

# Método de diferenças finitas fracionárias para resolver EDFs

for i in range(1, len(time\_points)):

    # Equações Lotka-Volterra discretizadas com derivadas fracionárias

    dRdt = alpha \* R[i-1] - beta \* R[i-1] \* F[i-1] + delta \* fractional\_difference(R[:i], q, dt)

    dFdt = -gamma \* F[i-1] + delta \* R[i-1] \* F[i-1] + delta \* fractional\_difference(F[:i], q, dt)

    # Atualização das populações usando o método de diferenças finitas fracionárias

    R[i] = R[i-1] + dt \* dRdt

    F[i] = F[i-1] + dt \* dFdt

# Criação de campos vetoriais

R\_vec = np.linspace(min(R), max(R), 20)

F\_vec = np.linspace(min(F), max(F), 20)

R\_grid, F\_grid = np.meshgrid(R\_vec, F\_vec)

dRdt\_grid = alpha \* R\_grid - beta \* R\_grid \* F\_grid

dFdt\_grid = -gamma \* F\_grid + delta \* R\_grid \* F\_grid

# Normalizando os vetores para melhor visualização

magnitude = np.sqrt(dRdt\_grid\*\*2 + dFdt\_grid\*\*2)

dRdt\_grid /= magnitude

dFdt\_grid /= magnitude

# Plotagem do gráfico bidimensional com campos vetoriais

plt.figure(figsize=(12, 6))

# Plotagem do campo vetorial

plt.quiver(R\_grid, F\_grid, dRdt\_grid, dFdt\_grid, scale=40, color='pink')

# Plotagem da evolução temporal de Coelhos e Raposas

plt.plot(R, F, label='Dinâmica Populacional', color='blue')

plt.title('Modelo Lotka-Volterra: Sistema Dinâmico Bidimensional com Campos Vetoriais')

plt.xlabel('População de Coelhos')

plt.ylabel('População de Raposas')

plt.legend()

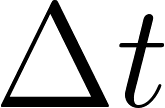
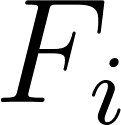
plt.grid(True)

plt.show()

Neste código, as equações do modelo Lotka-Volterra são discretizadas usando o método de diferenças finitas para aproximar as derivadas. O resultado é uma simulação temporal das populações de coelhos e raposas ao longo do tempo. O gráfico final mostra como as populações evoluem de acordo com o modelo.

O código fornecido implementa o modelo Lotka-Volterra com a adição de cálculos fracionários, representados pela ordem fracionária q . As equações diferenciais discretizadas usando o método de diferenças finitas são:

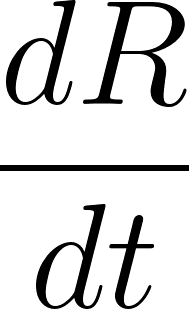
[](https://www.codecogs.com/eqnedit.php?latex=%20%5Cbegin%7Balign*%7D%20R_%7Bi%2B1%7D%20%26%3D%20R_i%20%2B%20%5CDelta%20t%20%5Ccdot%20%5Cleft(%5Calpha%20R_i%20-%20%5Cbeta%20R_i%20F_i%5Cright)%20%5C%5C%5C%5C%20F_%7Bi%2B1%7D%20%26%3D%20F_i%20%2B%20%5CDelta%20t%20%5Ccdot%20%5Cleft(-%5Cgamma%20F_i%20%2B%20%5Cdelta%20R_i%20F_i%5Cright)%20%5Cend%7Balign*%7D%20#1)

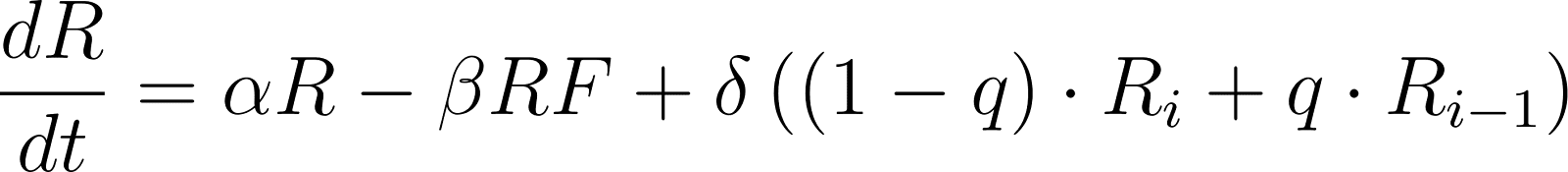
onde [https://lh7-us.googleusercontent.com/gdfwlKgxQPHEypJQrzTYrEzxSWQZKVuNDjKQ-JsGyoId7QO6-zUsT7nMOiU88WBnJb6fKAgc-iClR5dPm36GkEMd6470ZnkF9rFdB3_wBKsDcblnmQiK5Clfo_LvcKGN36e8MNP00X99vDggwZ3lSj8](https://www.codecogs.com/eqnedit.php?latex=%20i%20#1) é o índice da iteração, [](https://www.codecogs.com/eqnedit.php?latex=%20%5CDelta%20t%20#1) é o intervalo de tempo, e [](https://www.codecogs.com/eqnedit.php?latex=%20R_i%20#1), [](https://www.codecogs.com/eqnedit.php?latex=%20F_i%20#1) são as populações de coelhos e raposas no passo [https://lh7-us.googleusercontent.com/NWecmDKndSUvzWNbM4afzoeEAkyurzWobnJoyZ286SZAoa_h6zThy9qEDcpFGjEqNS1Beu_HAchVMjSOwpgtGoxGL1dJ70pxU9wNM5NqEqO5LgRMtevXu3Ye5xFySEGvhEa3nqVMmGFcudSSH4fGBxE](https://www.codecogs.com/eqnedit.php?latex=%20i%20#1), respectivamente.

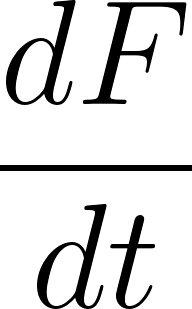
Cálculos Matemáticos:

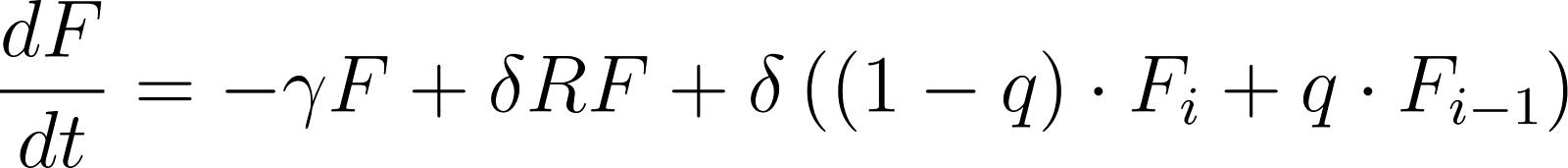
Vamos demonstrar os cálculos para o modelo Lotka-Volterra com derivadas fracionárias usando o método de diferenças finitas fracionárias, com os parâmetros fornecidos no código:

[](http://www.texrendr.com/?eqn=%5Cbegin%7Balign*%7D%5Calpha%20%26%3D%200.1%20%5C%5C%5Cbeta%20%26%3D%200.02%20%5C%5C%5Cgamma%20%26%3D%200.1%20%5C%5C%5Cdelta%20%26%3D%200.01%20%5C%5Cq%20%26%3D%200.5%20%5Cquad%20%5Ctext%7B(ordem%20fracion%C3%A1ria)%7D%5Cend%7Balign*%7D#1)

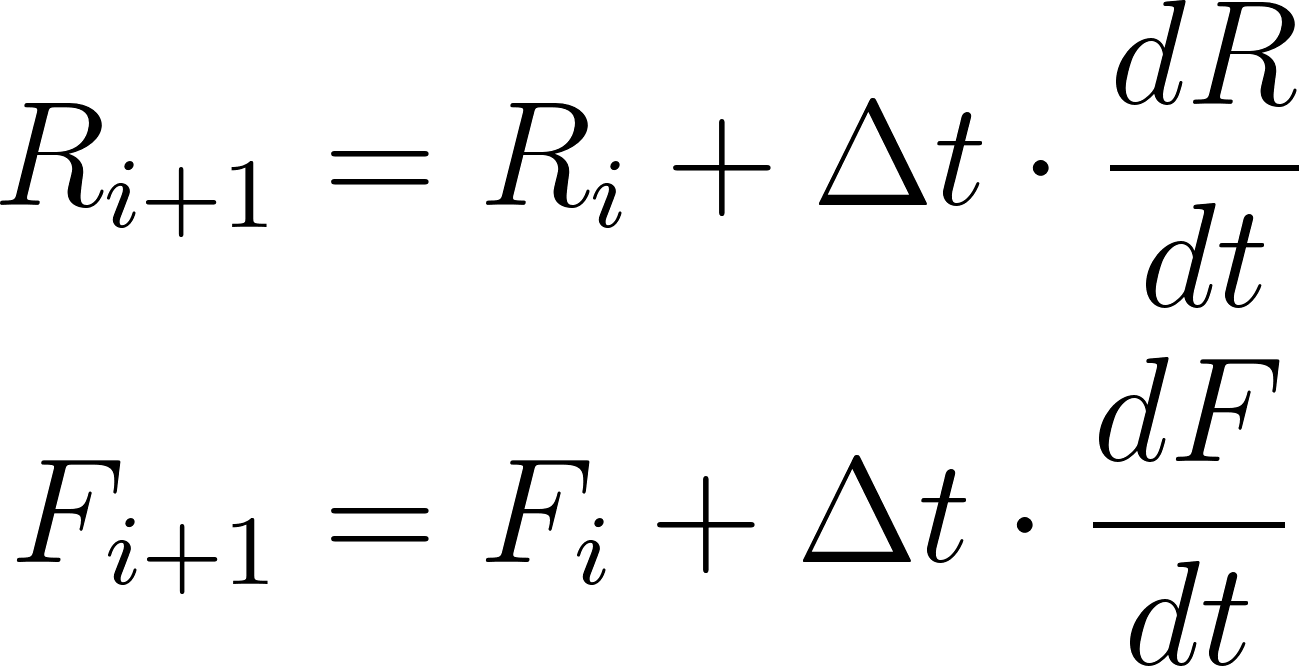
* Cálculo de [](https://www.codecogs.com/eqnedit.php?latex=%5Cfrac%7BdR%7D%7Bdt%7D#1):

[](https://www.codecogs.com/eqnedit.php?latex=%5Cfrac%7BdR%7D%7Bdt%7D%20%3D%20%5Calpha%20R%20-%20%5Cbeta%20R%20F%20%2B%20%5Cdelta%20%5Cleft(%20(1%20-%20q)%20%5Ccdot%20R_i%20%2B%20q%20%5Ccdot%20R_%7Bi-1%7D%20%5Cright)%20#1)

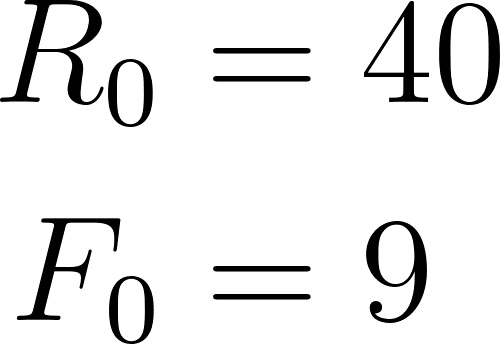
* Cálculo de [](https://www.codecogs.com/eqnedit.php?latex=%5Cfrac%7BdF%7D%7Bdt%7D#1):

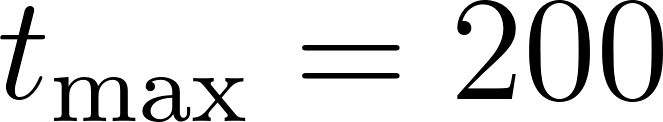
[](https://www.codecogs.com/eqnedit.php?latex=%20%5Cfrac%7BdF%7D%7Bdt%7D%20%3D%20-%5Cgamma%20F%20%2B%20%5Cdelta%20R%20F%20%2B%20%5Cdelta%20%5Cleft(%20(1%20-%20q)%20%5Ccdot%20F_i%20%2B%20q%20%5Ccdot%20F_%7Bi-1%7D%20%5Cright)%20#1)

* Atualização das Populações:

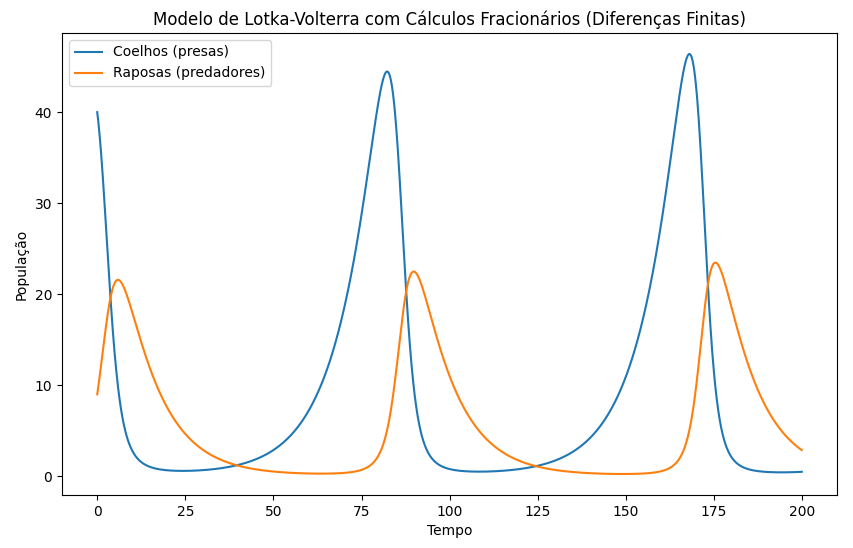
[](https://www.codecogs.com/eqnedit.php?latex=%5Cbegin%7Balign*%7D%20R_%7Bi%2B1%7D%20%26%3D%20R_i%20%2B%20%5CDelta%20t%20%5Ccdot%20%5Cfrac%7BdR%7D%7Bdt%7D%20%5C%5C%20F_%7Bi%2B1%7D%20%26%3D%20F_i%20%2B%20%5CDelta%20t%20%5Ccdot%20%5Cfrac%7BdF%7D%7Bdt%7D%20%5Cend%7Balign*%7D%20#1)

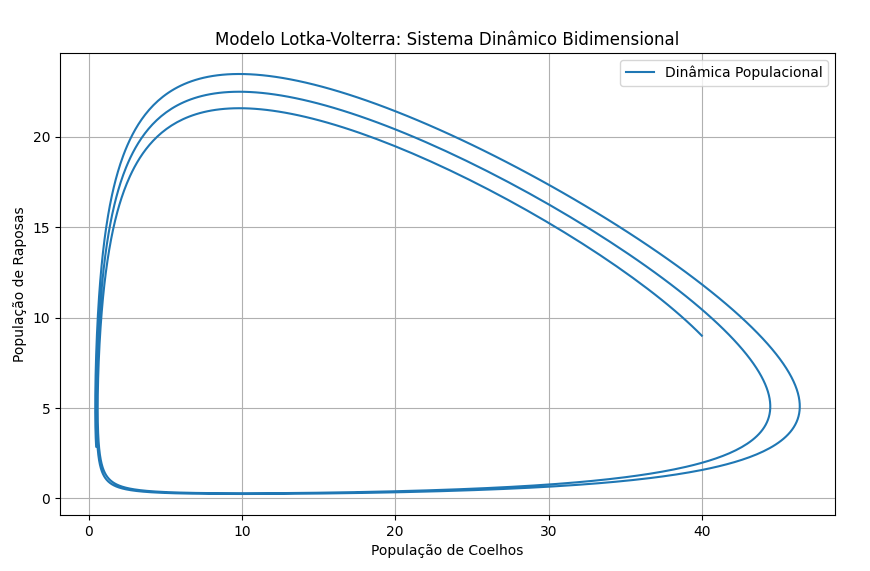
* Considere os seguintes valores iniciais:

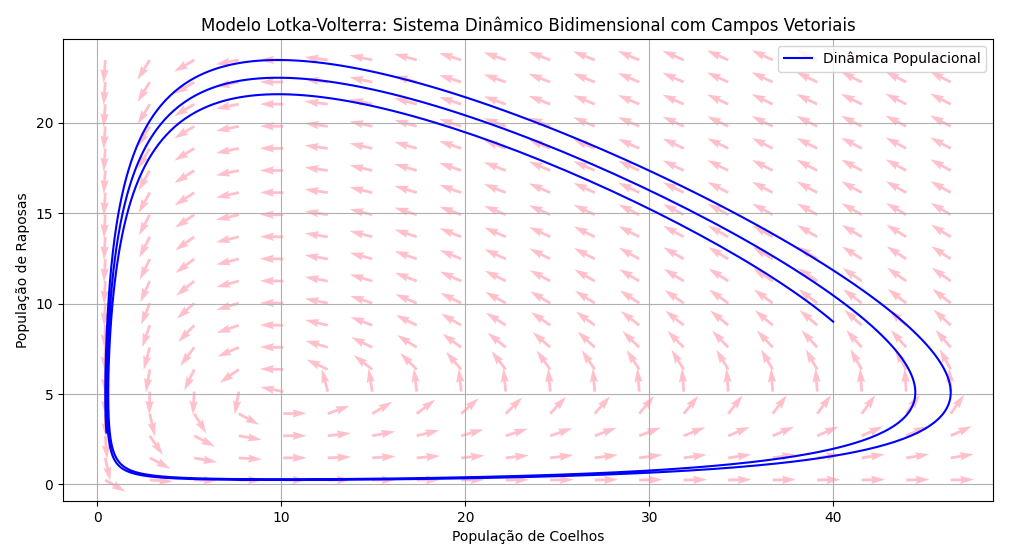
[](https://www.codecogs.com/eqnedit.php?latex=%20%5Cbegin%7Balign*%7D%20R_0%20%26%3D%2040%20%5C%5C%20F_0%20%26%3D%209%20%5Cend%7Balign*%7D%20#1)

e um incremento de tempo [](https://www.codecogs.com/eqnedit.php?latex=%5CDelta%20t%20%3D%200.1#1) com um tempo máximo de simulação [](https://www.codecogs.com/eqnedit.php?latex=t_%7B%5Ctext%7Bmax%7D%7D%20%3D%20200#1).

* Plotagem dos Resultados:

   - Gráfico das populações de coelhos e raposas ao longo do tempo.

- Gráfico das populações de coelhos e raposas Bidimensional

- Gráfico das populações de coelhos e raposas Bidimensional com Campos Vetoriais (Normalizados).

A introdução da função fractional\_difference(código Python), ou Calculo Fracionário, no cálculo das derivadas fracionárias representa uma abordagem alternativa para incorporar efeitos fracionários no modelo Lotka-Volterra em comparação com o cálculo por Equações Diferenciais Ordinárias (EDOs) padrão. A diferença fundamental está na forma como as derivadas fracionárias são tratadas. O método de diferenças finitas fracionárias é uma técnica numérica que discretiza a derivada fracionária usando diferenças finitas, levando em consideração a ordem fracionária q. Isso pode ser especialmente útil, quando se lida com sistemas complexos ou comportamentos não lineares que podem ser capturados de maneira mais precisa por derivadas fracionárias.

Em contraste, o cálculo por EDO padrão utiliza derivadas ordinárias, que assumem que as mudanças nas variáveis são proporcionais às próprias variáveis. Para modelos mais simples ou situações em que a linearidade é uma boa aproximação, as EDOs podem ser suficientes e são geralmente mais simples de resolver analiticamente.

O efeito específico da função fractional\_difference no modelo Lotka-Volterra dependerá do valor escolhido para a ordem fracionária q e das características específicas do sistema que está sendo modelado. Experimentar diferentes valores de q pode ajudar a entender como as derivadas fracionárias afetam a dinâmica do sistema.

Portanto, a introdução da função fractional\_difference, ou do Cálculo Fracionário traz uma abordagem numérica para incorporar derivadas fracionárias, permitindo maior flexibilidade na modelagem de sistemas dinâmicos complexos em comparação com a abordagem tradicional de EDOs.

**Referências:**

Murray, J.D. (2002). Mathematical Biology I: An Introduction. Springer.

Strogatz, S.H. (2018). Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering. CRC Press.

Soetaert, K., Petzoldt, T., & Setzer, R.W. (2010). Solving Differential Equations in R: Package deSolve. Journal of Statistical Software, 33(9), 1-25.

Python Software Foundation. (<https://www.python.org/>)

Podlubny, I. (1999). "Fractional Differential Equations: An Introduction to Fractional Derivatives, Fractional Differential Equations, to Methods of Their Solution and Some of Their Applications."

Diethelm, K., & Ford, N. J. (2002). "Analysis of fractional differential equations." Journal of Mathematical Analysis and Applications, 265(2), 229-248.

Podlubny, I., Chechkin, A., Skovranek, T., Chen, Y. Q., & Jara, B. M. (1999). "Matrix approach to discrete fractional calculus." Fractional Calculus and Applied Analysis, 2(3), 359-386.