


MAC2166 - Introdução à Computação - Grande áreas Civil, Mecânica, Química e Petróleo



[Início](#) / [Meus Ambientes](#) / [MAC2166 Civil, Mecânica, Química e Petróleo](#) / [Exercícios-programa](#) / [EP3: simulação de modelo para propagação de vírus](#)


 [Descrição](#)

 [Visualizar envios](#)

EP3: simulação de modelo para propagação de vírus

 **Data de entrega:** terça, 14 jul 2020, 23:59

 **Arquivos requeridos:** EP3.py, dados1.txt, dados2.txt, dados3.txt, dados4.txt, dados5.txt ( [Baixar](#))

 **Tamanho máximo de arquivo carregado:** 4 MiB

Tipo de trabalho:  Trabalho individual

Objetivos

Neste exercício-programa, você terá que desenvolver várias funções relacionadas com um modelo de simulação de contaminação por vírus. A função `main()` já está sendo fornecida pronta. Também são fornecidos os protótipos/cabeçalhos das funções a serem desenvolvidas. **Não** apague ou altere a função `main()`. Não altere os protótipos/cabeçalhos das funções (isto é, nome das funções e sua relação de parâmetros). Você deve preencher o corpo das funções solicitadas. Funções auxiliares adicionais podem também ser escritas.

Modelo SIR Episódico

O modelo SIR foi publicado em 1927 por Kermack e McKendrick ([Proc. R. Soc. Lond. A, 115:700-721](#)). Ele é um modelo determinístico relativamente simples utilizado até hoje para estimar a evolução de epidemias.

Em sua forma original, o modelo considera três variáveis básicas **S**, **I** e **R** que representam, respectivamente, indivíduos **S**uscetíveis mas ainda não infectados, indivíduos **I**nfectados (ou seja, doentes) e indivíduos **R**ecuperados, que são removidos do modelo (ou seja, que tiveram a doença no passado e agora não mais a transmitem – porque se curaram e adquiriram imunidade ou porque faleceram). Ele considera, ainda, uma população fixa sob análise de N indivíduos e duas constantes β e γ que representam, respectivamente, fatores de infecção e de recuperação característicos da doença.

Com estes parâmetros (β e γ), o modelo descreve a evolução temporal das variáveis **S**, **I** e **R** usando equações diferenciais codependentes:

$$\begin{aligned}\frac{dS}{dt} &= -\beta \frac{S I}{N} \\ \frac{dI}{dt} &= \beta \frac{S I}{N} - \gamma I \\ \frac{dR}{dt} &= \gamma I\end{aligned}$$

Este modelo não é suficiente para descrever em detalhes a evolução de uma epidemia como, por exemplo, a de COVID-19 que estamos vivenciando. Ele serve, entretanto, para descrever em linhas gerais como uma epidemia evolui. Um dos grandes problemas da aplicação do modelo ao caso do COVID-19 é que ele identifica o grupo dos infectados ao dos doentes, mas não prevê a existência dos assintomáticos: indivíduos infectados que contraem o vírus mas não desenvolvem sintomas. Vários estudos de populações bem demarcadas têm apontado que cerca de metade da população infectada adquire imunidade sem apresentar sintomas, mas em presídios norte-americanos que sofreram um teste massivo verificou-se que 96% da população já infectada era de assintomáticos ([Daniel P. Oran, AM, Eric J. Topol, MD, Prevalence of Asymptomatic SARS-CoV-2 Infection, Annals of Internal Medicine, 3 Jun 2020](#)).

A população e sua respectiva cardinalidade N deve considerar grupos que tenham possibilidade de contato e os parâmetros β e γ devem corresponder aos valores observados empiricamente e capazes de representar, respectivamente, quão transmissível se apresenta a doença e o tempo requerido para a recuperação de infectados. A transmissibilidade é determinada por características da doença em si e padrões de comportamento da população (distanciamento social, padrões de higiene, etc.).

Para simular este modelo computacionalmente, pode ser construída uma versão “episódica” em que o tempo é discretizado para poder ser tratado como uma sequência de números naturais (que podem representar, por exemplo, dias). As equações nesta versão “episódica” ficam assim:

$$\begin{aligned}S(t+1) &= S(t) - \beta \frac{S(t) I(t)}{N} \\ I(t+1) &= I(t) + \beta \frac{S(t) I(t)}{N} - \gamma I(t) \\ R(t+1) &= R(t) + \gamma I(t) \\ S(0) &= N - 1, I(0) = 1, R(0) = 0\end{aligned}$$

Neste EP produziremos um simulador SIR episódico e extrairemos alguns dados dele.

Tarefa 1

Construa uma função `SIR(N, Beta, Gama, Tmax)` que receba como parâmetros:

- ***N***, o tamanho da população (int)
- ***Beta***, o parâmetro correspondente a quão transmissível é a doença (float)
- ***Gama***, o parâmetro correspondente ao tempo necessário para se recuperar da doença (float)
- ***Tmax***, o valor máximo para a variável *tempo* a ser considerado nas simulações (int)

Esta função deverá retornar como resposta três listas contendo *Tmax* elementos, correspondendo respectivamente às variáveis ***S***, ***I*** e ***R*** em cada instante de tempo da lista 0, 1, 2, ..., (*Tmax*-1).

Exemplo de chamada da função SIR no Python Shell:

```
>>> S,I,R = SIR(10, 0.5, 0.1, 10)
>>> imprimeLista(S) # função fornecida pronta no EP3.py
9.0000 8.5500 7.9729 7.2585 6.4138 5.4712 4.4897 3.5445 2.7054 2.0154
>>> imprimeLista(I)
1.0000 1.3500 1.7921 2.3273 2.9392 3.5879 4.2106 4.7348 5.1004 5.2803
>>> imprimeLista(R)
0.0000 0.1000 0.2350 0.4142 0.6469 0.9409 1.2997 1.7207 2.1942 2.7042
```

Tarefa 2

Construa uma função `critic_SIR(N, Gama, Tmax, Beta_MIN, Beta_MAX, Beta_delta)` que receba como parâmetros:

- **N**, **Gama** e **Tmax** como na Tarefa 1
- **Beta_MIN**, **Beta_MAX** e **Beta_delta** (float) para representar o intervalo de valores $B = \{\beta_{MIN}, \beta_{MIN} + \Delta, \beta_{MIN} + 2\Delta, \dots, \beta_{MAX}\}$, sendo **Beta_MIN** = β_{MIN} , **Beta_MAX** = β_{MAX} e **Beta_delta** = Δ .

Esta função deverá retornar como resposta uma lista representando, para cada valor de β dentro do intervalo B, o valor máximo da variável **I** produzido pela função da Tarefa 1.

Exemplo de chamada da função `critic_SIR` no Python Shell:

```
>>> cSIR = critic_SIR(10, 0.1, 10, 0.05, 0.50, 0.05)
>>> imprimeLista(cSIR) # função fornecida pronta no EP3.py
1.0000 1.0000 1.2663 1.7439 2.3103 2.9424 3.6023 4.2421 4.8139 5.2803
```

Nas duas próximas tarefas utilizaremos os formatos PGM e PPM para produção de gráficos para apresentar visualmente resultados numéricos. Uma descrição detalhada destes formatos de arquivo, que são utilizados para o armazenamento de imagens, pode ser encontrada no seguinte link: https://www.ime.usp.br/~mac2166/ep3-2020/pgm_ppm.pdf

Tarefa 3

Construa uma função `gera_grafico_simples(L)` que receba como parâmetro uma lista **L** de valores (float) e construa um gráfico X-Y em que o eixo Y tenha como limite inferior o valor **Y_MIN=0** e como limite superior o inteiro **Y_MAX** definido pelo [teto](#) do valor máximo encontrado em **L**, e o eixo X tenha como limite inferior o valor **X_MIN=0** e como limite superior **X_MAX** a quantidade de elementos de **L** menos um. Este gráfico deve apresentar uma representação dos valores em **L**. O gráfico corresponderá a uma matriz **M** de dimensão **m x n**, sendo o número de linhas dado por **m = Y_MAX - Y_MIN + 1** e o número de colunas dado por **n = X_MAX - X_MIN + 1 = len(L)**. Os valores da matriz devem ser padronizados como **255 (branco)** para pontos que representem os valores contidos na lista L e **0 (preto)** para os demais pontos na imagem. Ou seja, cada ponto **(x,y) = (k,L[k])** deverá ser registrado em uma posição **M[i][j]** da matriz, sendo o índice i obtido por arredondamento de **Y_MAX - L[k]** para o inteiro mais próximo e **x=k=j**. Além de devolver a matriz resultante, a função deverá gravar o gráfico gerado na matriz **M** em um arquivo no formato **PGM**, com o nome padronizado "graf_simples.pgm".

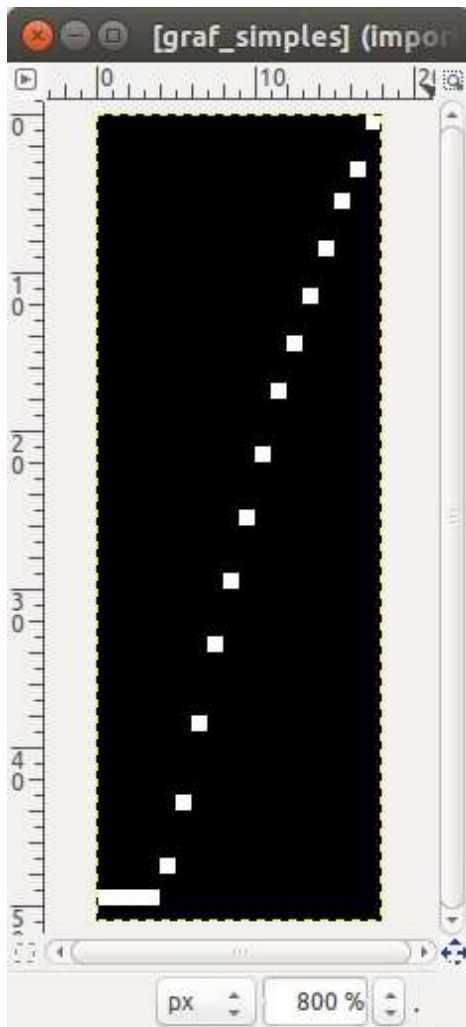
Exemplo de chamada da função `gera_grafico_simples` no Python Shell:

```
>>> cSIR = critic_SIR(10, 0.1, 10, 0.05, 0.50, 0.05)
>>> gera_grafico_simples(cSIR)
[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 255, 255], [0, 0, 0, 0, 0, 0, 255, 255, 0, 0], [0, 0, 0, 0, 0, 255, 0, 0, 0, 0],
[0, 0, 0, 255, 255, 0, 0, 0, 0, 0], [255, 255, 255, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]]
```

Abaixo é exibido o conteúdo do arquivo **PGM** correspondente "graf_simples.pgm":

```
P2
10 7
255
  0      0      0      0      0      0      0      0      0      0
  0      0      0      0      0      0      0      0      255    255
  0      0      0      0      0      0      255    255    0      0
  0      0      0      0      0      255    0      0      0      0
  0      0      0      255    255    0      0      0      0      0
255    255    255    0      0      0      0      0      0      0
  0      0      0      0      0      0      0      0      0      0
```

Para visualizar a imagem "graf_simples.pgm", você deve rodar seu programa em Python diretamente no seu computador. No VPL não é possível visualizar as imagens. Para visualizar a imagem, você deve abrir o arquivo "graf_simples.pgm" resultante com algum programa de edição/visualização de imagens, com suporte para o formato PGM. Abaixo é mostrado um exemplo de visualização no [Gimp](#), usando **cSIR = critic_SIR(100, 0.2, 100, 0.05, 0.90, 0.05)** e depois **gera_grafico_simples(cSIR)**.



Tarefa 4

Construa uma função `gera_grafico_composto(S,I,R)` que receba como parâmetros três listas de valores (float) e construa um gráfico X-Y em que o eixo Y tenha como limite inferior o valor **$Y_{MIN}=0$** e como limite superior o inteiro **Y_{MAX}** definido pelo [teto](#) do valor máximo encontrado em **$S \cup I \cup R$** , e o eixo X tenha como limite inferior o valor **$X_{MIN}=0$** e como limite superior **X_{MAX}** a quantidade de colunas de **S** menos um. Este gráfico deve apresentar uma representação dos valores em **S** , **I** e **R** , em formato de linhas superpostas com cores distintas. O gráfico deverá ser registrado em um arquivo no formato **PPM**, com o nome padronizado "graf_composto.ppm". A função deverá, também, retornar como resposta o conteúdo do arquivo **PPM** no formato de uma matriz de valores inteiros. Estes valores devem ser também padronizados como:

- **"255 0 0" (vermelho)** para pontos que representem os valores contidos na lista correspondente aos valores de **S** ,
- **"0 255 0" (verde)** para pontos que representem os valores contidos na lista correspondente aos valores de **I** ,
- **"0 0 255" (azul)** para pontos que representem os valores contidos na lista correspondente aos valores de **R** e

- “0 0 0” (preto) para os demais pontos na imagem.

No caso de sobreposição das curvas, as cores devem ser somadas (exemplo, sobreposição de **S** e **I** gera cor “255 255 0” (amarelo)).

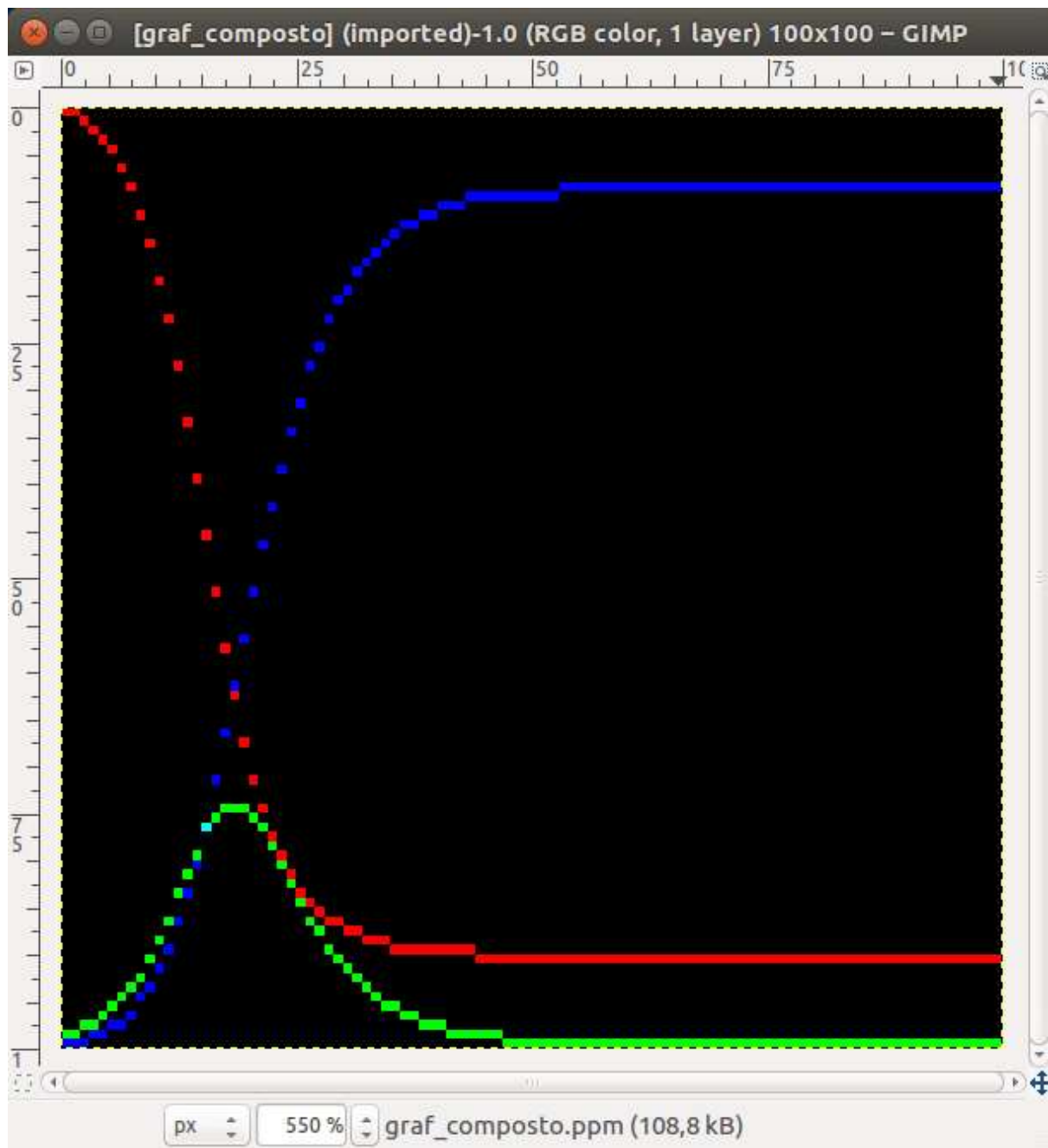
Exemplo de chamada da função `gera_grafico_composto` no Python Shell:

```
>>> S,I,R = SIR(10, 0.5, 0.1, 4)
>>> gera_grafico_composto(S, I, R)
[[255, 0, 0, 255, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 255, 0, 0, 255, 0], [0, 255, 0, 0, 255, 0, 0, 0, 0, 0, 0, 0], [0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255]]
```

Abaixo é exibido o conteúdo do arquivo **PPM** correspondente “graf_composto.ppm”:

```
P3
4 10
255
255 0 0 255 0 0 0 0 0 0 0 0
0 0 0 0 0 0 255 0 0 0 0 0
0 0 0 0 0 0 0 0 0 255 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 255 0 0 255 0
0 255 0 0 255 0 0 0 0 0 0 0
0 0 255 0 0 255 0 0 255 0 0 255
```

Para visualizar a imagem “graf_composto.ppm”, você deve rodar seu programa em Python diretamente no seu computador. No VPL não é possível visualizar as imagens. Para visualizar a imagem, você deve abrir o arquivo “graf_composto.ppm” resultante com algum programa de edição/visualização de imagens, com suporte para o formato PPM. Abaixo é mostrado um exemplo de visualização no [Gimp](#), usando **S,I,R = SIR(100, 0.5, 0.2, 100)** e depois **gera_grafico_composto(S, I, R)**.



Tarefa 5

Construa uma função `leitura_de_valores(nome_de_arquivo)` que receba como parâmetro um nome de arquivo e retorne valores para as seguintes variáveis: ***N***, ***Gama***, ***Tmax***, ***Beta_MIN***, ***Beta_MAX***, ***Beta_delta***, que deverão ser lidas de um arquivo texto com o nome indicado em ***nome_de_arquivo***. O arquivo texto deve ter um valor para cada uma das variáveis de retorno. Os valores devem ser fornecidos um em cada linha e devem ser transformados para o tipo de dados apropriado (int ou float, dependendo da variável). A função **`leitura_de_valores(nome_de_arquivo)`** é chamada pela função **`main()`** do seguinte modo:

```
Dados = input("Digite nome do arquivo: ");  
N, Gama, Tmax, Beta_MIN, Beta_MAX, Beta_delta = leitura_de_valores(Dados)
```

Exemplos de arquivos de entrada podem ser vistos nas abas "dados1.txt", "dados2.txt", "dados3.txt", "dados4.txt" e "dados5.txt". Não edite esses arquivos, pois eles serão usados nos testes automáticos do VPL.

Programa principal:

A função `main()` está sendo fornecida pronta e não deve ser alterada. Ela possui 7 modos de operação diferentes, visando testar as diferentes partes do seu programa. Os modos são:

1. Calcula 'SIR' e imprime os vetores S, I e R - leitura dos parâmetros via teclado.
2. Calcula 'critic_SIR' e imprimir o vetor resultante - leitura dos parâmetros via teclado.
3. Calcula 'critic_SIR' e imprimir o vetor resultante - leitura dos parâmetros de um arquivo fornecido.
4. Calcula 'critic_SIR' e testa matriz devolvida por 'gera_grafico_simples' - leitura dos parâmetros via teclado.
5. Calcula 'critic_SIR' e testa arquivo PGM no disco por 'gera_grafico_simples' - leitura dos parâmetros via teclado.
6. Calcula 'SIR' e testa matriz devolvida por 'gera_grafico_composto' - leitura dos parâmetros via teclado.
7. Calcula 'SIR' e testa arquivo PPM no disco por 'gera_grafico_composto' - leitura dos parâmetros via teclado.

Exemplos de entradas e saídas para cada uma das opções. As entradas fornecidas estão em azul e as saídas esperadas em vermelho.

Modo 1:

```
Digite modo do programa: 1  
Digite N: 10  
Digite Beta: 0.5  
Digite Gama: 0.1  
Digite Tmax: 10  
S = 9.0000 8.5500 7.9729 7.2585 6.4138 5.4712 4.4897 3.5445 2.7054 2.0154  
I = 1.0000 1.3500 1.7921 2.3273 2.9392 3.5879 4.2106 4.7348 5.1004 5.2803  
R = 0.0000 0.1000 0.2350 0.4142 0.6469 0.9409 1.2997 1.7207 2.1942 2.7042
```

Modo 2:

```
Digite modo do programa: 2  
Digite N: 10  
Digite Gama: 0.1  
Digite Tmax: 10  
Digite Beta_MIN: 0.05  
Digite Beta_MAX: 0.50  
Digite Beta_delta: 0.05  
1.0000 1.0000 1.2663 1.7439 2.3103 2.9424 3.6023 4.2421 4.8139 5.2803
```


Modo 3:

```
Digite modo do programa: 3
Digite nome do arquivo: dados2.txt
1.0000 1.0000 1.3514 2.1129 2.8484 3.4907 4.0423 4.5150 4.9291 5.2844
```

Modo 4:

```
Digite modo do programa: 4
Digite N: 10
Digite Gama: 0.1
Digite Tmax: 10
Digite Beta_MIN: 0.1
Digite Beta_MAX: 0.5
Digite Beta_delta: 0.1
[[0, 0, 0, 0, 0], [0, 0, 0, 0, 255], [0, 0, 0, 255, 0], [0, 0, 255, 0, 0], [0, 255, 0, 0, 0], [255, 0, 0, 0, 0], [0, 0, 0, 0, 0]]
```

Modo 5:

```
Digite modo do programa: 5
Digite N: 10
Digite Gama: 0.1
Digite Tmax: 10
Digite Beta_MIN: 0.1
Digite Beta_MAX: 0.5
Digite Beta_delta: 0.1
p2
5 7
255
 0 0 0 0 0
0 0 0 0 255
0 0 0 255 0
0 0 255 0 0
0 255 0 0 0
255 0 0 0 0
0 0 0 0 0
```

Modo 6:

```
Digite modo do programa: 6
Digite N: 10
Digite Beta: 0.50
Digite Gama: 0.1
Digite Tmax: 4
[[255, 0, 0, 255, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 255, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 255, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 255, 0, 0, 255, 0], [0, 255, 0, 0, 255, 0, 0, 0, 0, 0, 0, 0], [0, 0, 255, 0, 0, 255, 0, 0, 255, 0, 0, 255]]
```

Modo 7:

```
Digite modo do programa: 7
Digite N: 10
Digite Beta: 0.50
Digite Gama: 0.1
Digite Tmax: 4
p3
4 10
255
255 0 0 255 0 0 0 0 0 0 0
0 0 0 0 0 0 255 0 0 0 0
0 0 0 0 0 0 0 0 0 255 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 255 0 0 255
0 255 0 0 255 0 0 0 0 0 0
0 0 255 0 0 255 0 0 255 0 0 255
```

Arquivos requeridos

EP3.py