Timelines e controle de epidemias

Jones Martins Matheus Andrade Raphael Almeida Vitor Vasconcellos

Fevereiro 2021

Abstract

Realizamos um comparativo entre análises numéricas utilizando cadeia de Markov, equações diferenciais ordinárias e simulação de eventos discretos para o modelo epidemiológico SIS no casos de propagação de fake news em mídias sociais.

1 Introdução

O trabalho traça um paralelo entra a disseminação de Fake News em redes de mídias sociais como Facebook, Twitter, Instagram ou LinkedIn e a dinâmica de epidemias em redes, como o COVID-19.

O trabalho foi divido em etapas; cada uma com suas dificuldades e particularidades. Primeiramente foi definido o modelo matemático que seria base comparativa para as simulações. Depois, foi elaborada uma estratégia de simulação e por fim um cruzamento dos dados obtidos em cada uma das etapas.

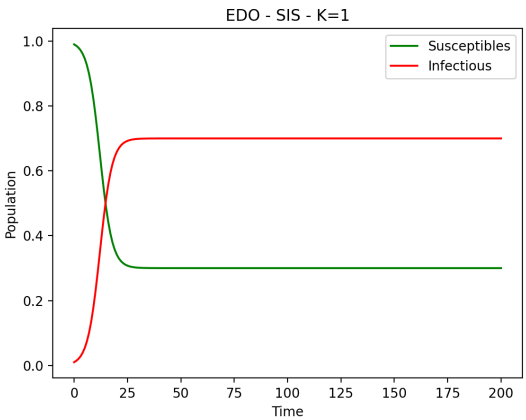
2 Modelos

Numerosas doenças infecciosas não conferem imunidade duradoura, como rotavírus, infecções sexualmente transmissíveis e muitas infecções bacterianas. Para essas doenças, um indivíduo pode ser infectado várias vezes ao longo da vida, sem imunidade aparente. Esse comportamento se enquadra na classe de modelo abreviada como SIS (Susceptible-Infectious-Susceptible). O problema proposto pelo trabalho também se enquadra nesse modelo; onde um *usuário* pode ser considerado *infectado* quando um ou mais *spots* da sua *timeline* possuem uma *fake news* e ele passa a ser *suscetível* quando não possui nenhuma *fake news* em sua *timeline*. Esse conjunto de equações tem um equilíbrio endêmico, pois os indivíduos em recuperação reabastecem o estoque de suscetíveis. Para o caso onde K=1 modelamos o problema da seguinte maneira:

$$\frac{dS}{dt} = -(\beta SI) + \gamma I$$

$$\frac{dI}{dt} = (\beta SI) - \gamma I$$

Onde β corresponde a **taxa de infeção**, γ **taxa de recuperação**, I é a **população infectada** e S é a **população suscetível**

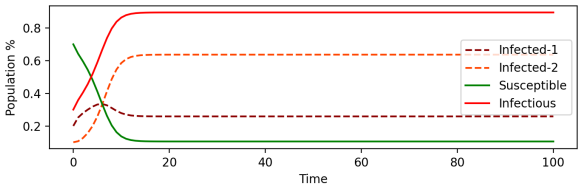


Para o caso onde K=2 a modelagem foi adaptada. A estratégia foi determinar duas populações de usuários infectados: I_1 que representa a população com apenas um spot infectado e I_2 que representa a população com seus dois spots infectados. Com isso precisamos também arbitrar um β_1 e um β_2 que são respectivamente as taxas de infecção quando o usuário possui uma fake news e duas fake news em sua timeline. Seguindo a modelagem:

$$\frac{dS}{dt} = \gamma I_1 - (\beta_1 I_1 + \beta_2 I_2)S$$

$$\frac{dI_1}{dt} = (\beta_1 I_1 + \beta_2 I_2)S - (\gamma I_1 + (\beta_1 I_1 + \beta_2 I_2)I_1)$$

$$\frac{dI_2}{dt} = (\beta_1 I_1 + \beta_2 I_2)I_1 - \gamma I_2$$



3 Simulação

Nossa estratégia inicial para a construção do simulador foi desenvolver um código genérico, com uma boa base de abstração, que fosse capaz de lidar com todos os casos propostos sem muitas alterações. Sendo assim, escolhemos o Python para desenvolvimento por ser uma linguagem de fácil acesso e bem suportada por diversas bibliotecas de matemática e estatística. A estrutura do nosso código esta dividida em duas partes:

3.1 Biblioteca

A primeira parte é uma biblioteca fake_news_sis_simulator que provê acesso ao ambiente de execução do simulador. O ambiente é composto de um classe Simulator que mantêm o estado durante a execução do simulador. Internamente utilizamos a biblioteca numpy para geração das amostras de distribuições necessárias para produção dos eventos a serem processados. Demais, administramos a fila de eventos com auxílio do módulo nativo heapq que provê as funcionalidades necessárias para construção de um fila de prioridade mínima.

Desse modo, a lógica básica do simulador é que em cada etapa da execução são gerados todos os possíveis eventos relativos ao compartilhamento entre usuários, ou recebimento externo, de noticias falsas ou genuínas, esses são então adicionados na fila de prioridade mínima para imediatamente ser extraído o próximo evento a ser executado dentre esse conjunto. Em seguida, é simulado a execução desse evento e reiniciada a fila de prioridade mínima, permitindo, assim, a execução do próximo passo. Por fim, o simulador gera estatísticas quanto as alterações ocorridas no ambiente para serem externamente analisadas.

Além disso, incluímos na biblioteca as implementações dos métodos números de Markov e EDO relativos ao modelo epidemiológico SIS, para podermos comparar seus resultados aos da simulação.

3.2 Jupyter Notebook

A segunda parte é um Jupyter Notebook no qual importamos a biblioteca mencionada acima e simulamos diversas rodadas de execução com diversos parâmetros. Essa etapa possui hyperparametros comuns a todos os métodos usados na análise, e também parâmetros individuais para a simulação e outros algorítimos.

No documento *.ipynb* podemos configurar o tempo de análise, número de indivíduos, taxas variadas, os valores das funções $f_0(a)$ e $f_1(a)$, além de outras configurações como nível de discretização da curva de Markov, etc.

A principio havíamos optado por usar o Colab. Entretanto, devido ao alto tempo demandado para rodar as simulações com intervalo de confiança significativo acabamos migrando para o Jupyter Lab.

O Jupyter, além da função de facilitar as execuções para parâmetros diversos e facilitar a depuração do código, também é a nossa ferramenta fundamental para geração de gráficos e analise dos métodos e resultados. Todas as imagens adicionadas a esse documento de alguma forma foram geradas por esses notebooks. Para a análise de cada método o documento gera gráficos individuais de execução e gráficos combinados com outros métodos para analise comparativa.

Uma das dificuldades que tivemos foi inicialmente detectada através dessa ferramenta, onde percebemos que o parâmetro de tempo da simulação divergia do parâmetro de tempo da EDO e da solução por cadeia de Markov. Para a EDO e a cadeia de Markov o parâmetro de tempo é em milissegundos, enquanto que para a simulação foi em segundos. Como utilizamos a mesma linguagem base, suspeitamos que essa divergência vem da forma que parametrizamos as taxas e tempos em cada método.

Na análise de dados, os gráficos foram essenciais para determinar se os métodos estavam funcionando. Percebemos uma possível instabilidade numérica em nossa cadeia de Markov e também problemas relacionados ao valor inicial em nossos métodos graças a interatividade fornecida pela ferramenta e os gráficos plotados.

Podemos visualizar na imagem abaixo onde temos um exemplo de gráfico combinado:

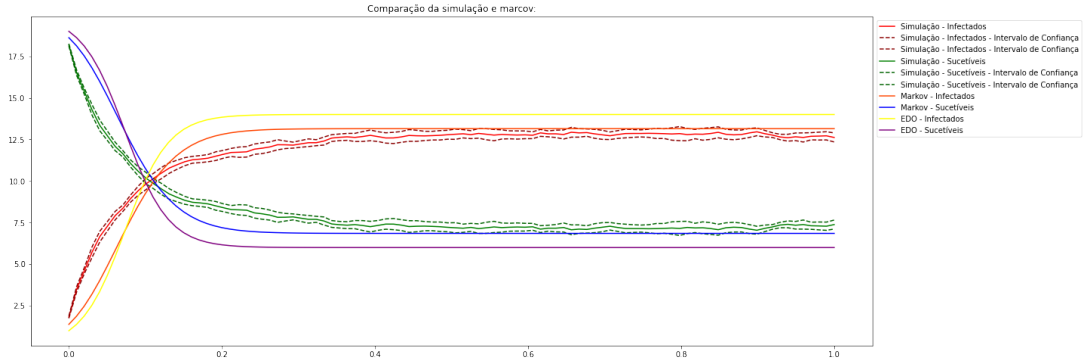


Figure 1: Exemplo de gráfico com mais de um método combinado

Desafios: Uma das grandes dificuldades de implementação foi construir um método de calcular os intervalos de confiança em um contexto continuo, no qual os dados das simulações são desalinhados em tempo e número de eventos.

Combinar esses dados não foi trivial. No notebook adicionamos uma função responsável por alinhar os dados, usando o conceito de janelas de tempo. Basicamente todo o intervalo de simulação e métodos numéricos foi dividido em intervalos temporais. Para cada dado de uma simulação dentro desses intervalos, tiramos a média e pegamos o tempo do intervalo como referencia. Além disso, se por acaso a duração do dado por tempo não for suficiente para alinhar com os outras iterações, implementemos um método de repetição do dado anterior nos intervalos restantes, afim de alinhar todas as análises e garantir o funcionamento do cálculo dos intervalos de confiança. Além disso, dada a complexidade de implementar o intervalo de convergência para $K \geq 2$ e a proximidade com o prazo de entrega do trabalho, não conseguimos disponibilizá-lo. O que dificultou a comparação mais direta entre os métodos. Dessa forma, ao longo desse documento, os algorítimos referentes a $K \geq 2$ muitas vezes não possuem gráficos comparativos.

Outro grande desafio foram os elevados tempos de execução. Considerando a excessiva duração de algumas execuções da simulação, a obtenção dos dados em quantidade suficiente para responder as perguntas e testar os algorítimos requeria muito tempo e não atendia a nossa necessidade. Para tentar mitigar esse problema, saímos da nuvem do Colab para o Jupyter local para ganhar um pouco de performance e diminuir o em tempo de execução.

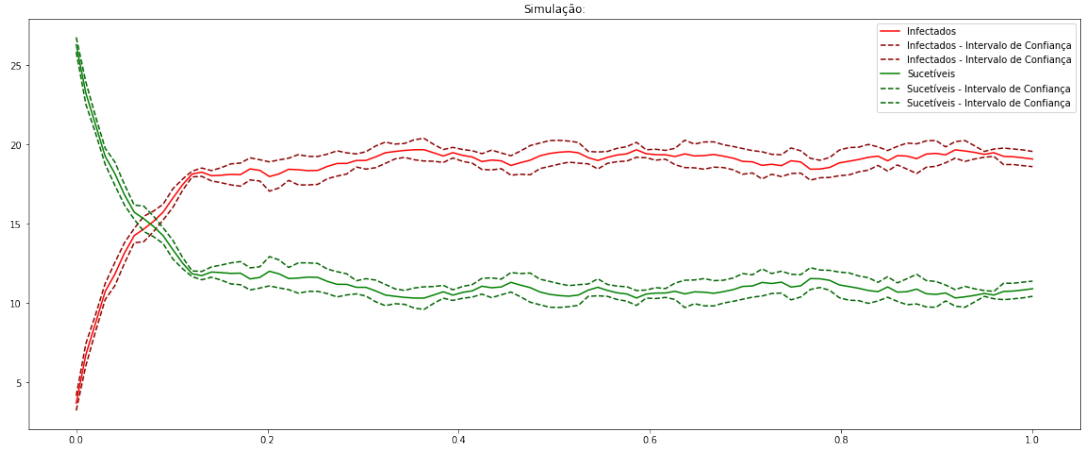


Figure 2: Exemplo do intervalo de confiança

3.3 Ambiente de depuração interativo

Desenvolvemos uma ferramenta de visualização interativa da execução do simulador, com intuito de facilitar a compreensão e a depuração de erros na lógica do memso. Ela pode ser acessada através do comando utilitário simulate depois da instalação da nossa biblioteca no sistema. O visualizador fornece informações internas do simulador e mostra em tempo real quais eventos estão sendo executados e estatísticas básicas sobre o estado dos usuários sendo simulados.

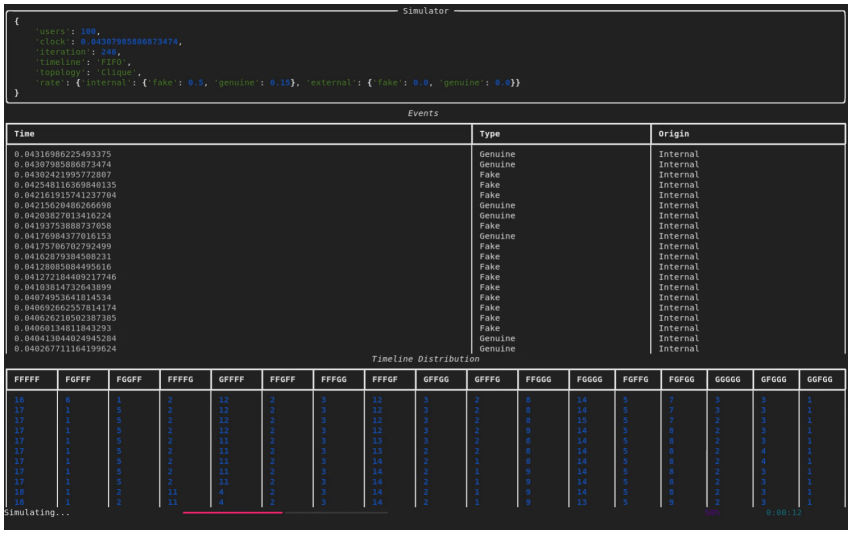


Figure 3: Ambiente de visualização interativo

3.4 Dificuldades

Inicialmente a abordagem que adotaríamos para a geração e execução de eventos seria a de constantemente popular a fila com novos eventos, sem limpá-la entre os passos, e usar o delta do tempo para diferenciar os eventos de cada passo. Isso se mostrou problemático, pois devido ao uso da fila de prioridade mínima, todos os eventos gerados seriam consumidos, o que não leva a uma boa simulação por causa da inter-dependência intrínseca entre o estado do ambiente de execução e quais eventos foram simulados nele durante tempo. Dessa forma, os eventos gerados num passo que não foram escolhidos se tornam obsoletos, pois foram gerados baseados num ambiente anterior a execução do evento escolhido. Por isso decidimos limpar a fila de eventos a cada passo antes de escolhermos o evento a ser simulado.

Durante a comparação dos resultados do simulador com os dos métodos de Markov e EDO percebemos algumas inconsistências em certos casos que nos levaram a descobrir um falha no modo que o simulador calculava as taxas de propagação das notícias entre usuários. Essa inconsistência se mostrava

mais drasticamente em casos com grandes quantidades de usuários nos quais a função resultante da análise dos resultados da simulação se mostrava deslocada no eixo horizontal. Conseguimos identificar o problema, entretanto não conseguimos corrigi-lo devido a proximidade com o prazo de entrega do entrega do trabalho.

Um problema que se mostrou durante a execução de casos com grandes quantidades de usuários foi a expressiva deterioração da performance do simulador. Alguns dos motivos para isso vem da escolha inicial do Python como linguagem para desenvolvimento da aplicação, que é conhecidamente mais lenta que contrapartidas como C e Rust. Outra razão, que identificamos, foi a falta de aproveitamento de alguns casos de otimização durante o uso do numpy. Especialmente na geração dos tempos dos eventos com a função np.random.exponetials, onde para cada evento gerado uma nova execução da função era feita, o que não é o ideal já que poderíamos gerar todos os tempos em uma unica execução da função. Entretanto não conseguimos corrigir esse falha de otimização devido a proximidade com o prazo do entrega do trabalho.

4 Markov

Para K = 1:

Modelamos o problema de $K = 1$ como uma cadeia de Markov onde os estados representam o número de indivíduos infectados. Essa cadeia possui $n + 1$ estados pois temos os estados referentes a todas as quantidades de infectados de 0 a $(n - 1)$. Como o modelo é *SIS*, os indivíduos podem transitar entre infectados e suscetíveis. O que faz com que a cadeia possa transitar entre mais ou menos infectados.

Como K = 1, só é possível que alguém receba uma mensagem falsa ou uma mensagem verídica. Ou seja, alguém está ou com a timeline completamente infectado ou não. Além disso, para K = 1, não faz sentido falar a respeito de *FIFO* ou *RND*, pois ambos os algoritmos são equivalentes.

Nossa decisão para $f_1(a)$ e $f_0(a)$ para o caso de K = 1 foi $f_1(a) = 1$ e $f_0(a) = 1$, uma vez que o número de posts no feed só pode ser igual a um, e que essas funções não precisariam influenciar nossos cálculos.

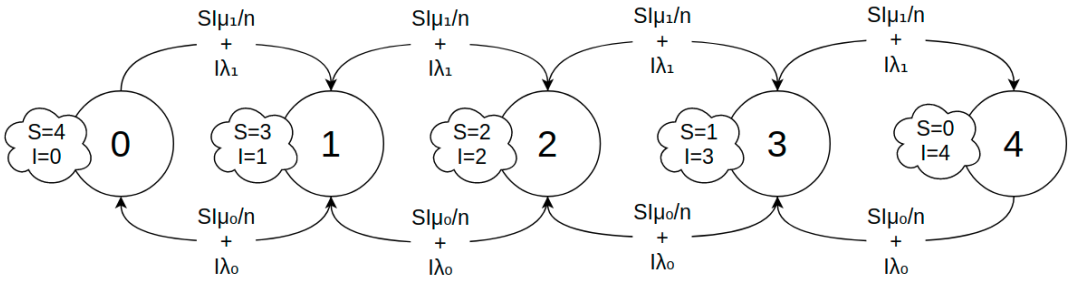


Figure 4: Modelo de cadeia usado no algorítimo de construção de Q

Para calcular a matriz de taxas fizemos um diagrama simplificado da cadeia e o traduzimos na matriz de taxas Q, que é uma matriz tri-diagonal. Em seguida usamos a função *exmp* para calcular a sua matriz de probabilidades. Entretanto, como desejávamos plotar a evolução da cadeia, fizemos os deslocamentos no tempo usando passos de tempo definidos para gerar a matriz Q deslocada no tempo e em seguida uma matriz com uma timeline dos resultados para cada condição inicial, ou seja, cada unidade de tempo, a partir de um estado inicial, gera uma matriz de probabilidades $P(t) = exp(Q * t)$. Essa matriz P(t) representa as probabilidades de se ter determinada quantidade de infectados em dado tempo para cada condição inicial. Em seguida aplicamos $\sum_{i=0}^{n} xP(X = x)$ para todos os possíveis estados para obter uma lista de infectados por intervalo de tempo. Com esses dados, fomos capazes de plotar a evolução da infecção no tempo e compará-la aos demais métodos. Podemos observar abaixo um exemplo de evolução gerada a partir da cadeia de Markov:

Uma percepção relevante é que algumas vezes a cadeia de Markov simulada ficava levemente deslocada em relação a simulação e a EDO. Percebemos que nossa abordagem na montagem da cadeia provavelmente esta gerando erros numéricos para grandes populações ou períodos de tempo muito longos. Sendo assim, atribuímos esses deslocamentos a esses erros. O deslocamento pode ser visualizado abaixo:

Quanto maiores os valores envolvidos na montagem da matriz Q e P, maior o deslocamento em relação as outras abordagens, até o momento em que essa abordagem diverge completamente. Apesar

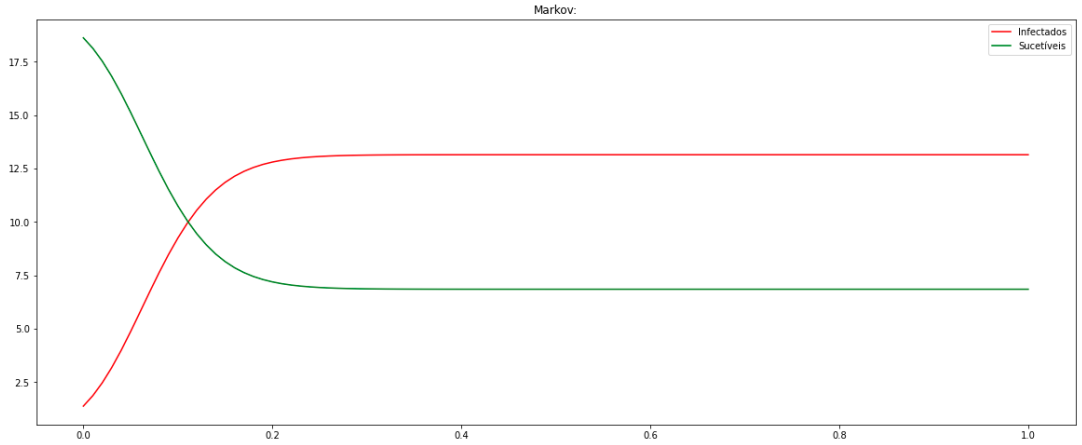


Figure 5: Exemplo de evolução da Cadeia de Markov

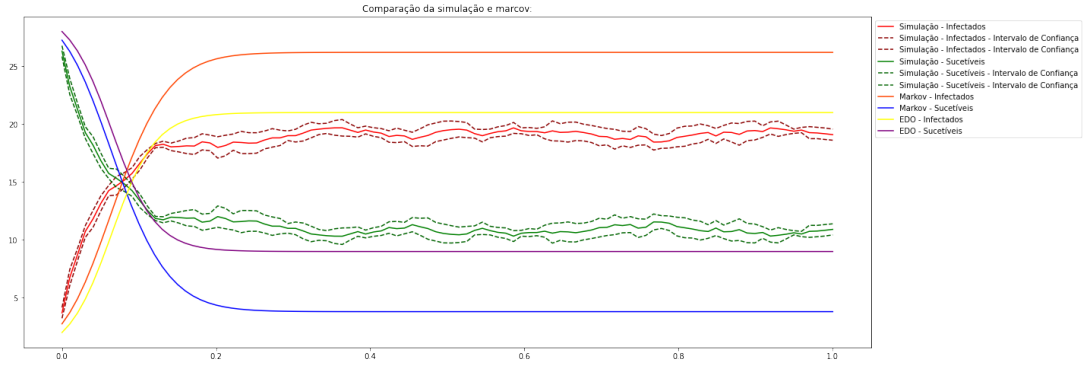


Figure 6: Exemplo de descolamento da cadeia de Markov

de nossa suspeita, seria relevante fazer uma investigação mais a fundo da razão do surgimento desses deslocamentos.

Mesmo com os deslocamentos em questão, percebemos que para variações nas taxas exógenas e endógenas nosso algorítimo se comportou bem e teve resultados satisfatórios também para o número de pessoas. Entretanto, percebemos que ao aumentar o número de infectados iniciais, nossa cadeia se a afasta cada vez mais da EDO e da simulação, indicando que pode haver um problema na definição das taxas da matriz Q ou no algorítimo que gera a evolução a partir da cadeia.

Apesar de todos esses fatores, essa abordagem se provou útil como segundo ou terceiro mecanismo de checagem de resultados e nos levou a outras descobertas anteriormente citadas.

Para K = 2:

Nosso método para K = 2 foi diferente. A abordagem que seguimos foi similar a do código MATLAB citado no texto base do trabalho. Deste modo, formamos recursivamente uma matriz de taxas cujos estados são ($n_0 0$, $n_0 1$, $n_1 0$, $n_1 1$), esses são valores válidos, nos quais sua soma é a população total N e não há valores negativos. Por exemplo, ($N-5$, 4, 0, 1), (10, $N-14$, 3, 1), etc. Esses estados são mapeados para posições de 0 a $N-1$. E então calculamos as probabilidades $P(t) = exp(Q*t)$ e utilizamos as probabilidades de (N, 0, 0, 0) e (0, 0, 0, N).

RND

Aqui, a matriz de taxas Q é preenchida com os 4 movimentos possíveis a partir de determinado estado levando em conta as taxas exógena e endógena.

FIFO

A mesma lógica da RND, mas com os 6 movimentos possíveis e suas respectivas taxas.

Resultados obtidos para FIFO:

Percebe-se que mantendo taxas iguais, exceto a taxa de recuperação exógena, que foi aumentada, a probabilidade de estarmos no estado completamente infectado diminuiu de 60% para 20%.

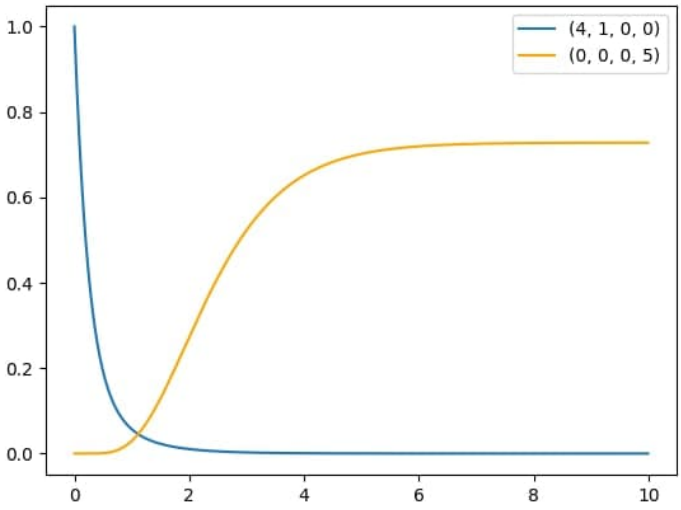


Figure 7: $\mu_0 = 0.2$, $\mu_1 = 0.5$, $\lambda_0 = 0$, $\lambda_1 = 0$

5 Perguntas e respostas

5.1 Sistema transiente: tempo até extinção da epidemia

Contexto: Quando consideramos $\lambda_0 = \lambda_1 = 0$, o sistema leva aproximadamente 4 unidades de tempo até atingir estado absorvente. Já quando consideramos $\lambda_0 = 0.1$ e $\lambda_1 = 0.2$, o sistema fica no estado em que todos os nós estão infectados em torno de 50% do tempo.

Pergunta 1: Por que a solução do sistema não é simétrica? Ou seja, por que no fim das contas não temos que com probabilidade 50% terminamos no estado all fake, e com 50% no estado all non fake? O que está quebrando a simetria?

Resposta 1: Pela primeira situação com $\lambda_0 = \lambda_1 = 0$, vemos que a cadeia apresenta taxas μ_0 e μ_1 diferentes e que ela é absorvida. Quando colocamos as taxas exógenas $\lambda_0 = 0.1$ e $\lambda_1 = 0.2$ temos o sistema em equilíbrio. Basicamente isso indica que no primeiro caso, a cadeia foi absorvida sendo todos os nós contagiados e também que $\mu_0 > \mu_1$. Logo, precisamos de uma taxa exógena λ_0 e λ_1 diferentes para equilibrar a desproporcionalidade de μ_0 e μ_1. Assim uma alocação $\lambda_0 = \lambda_1$ nesse caso, iria gerar um gráfico não simétrico.

Pergunta 2: Como se comporta a simulação para diferentes condições de estado inicial?

Resposta 2: Como estamos em um sistema transiente e potencialmente sem taxas exógenas, existem determinados estados iniciais que podem travar o sistema em um mesmo estado para sempre, ou faze-lo chegar a uma solução estacionária, logo, as condições iniciais afetam diretamente o sistema tanto em sua evolução, quanto em sua solução estacionária.

5.2 Sistema estacionário: solução estacionária

Contexto: Agora consideramos o caso em que há chegadas de mensagens exógenas. Ou seja, além do que é trocado na rede entre os nós, também chegam mensagens de fora. Note que quando iniciamos o sistema no estado em que nenhum nó tem notícia fake na timeline, ainda chegam notícias fake de fora, então o sistema continua evoluindo a partir daí (compare com o caso em que não havia chegadas exógenas, em que o sistema acabava no estado inicial para sempre).

Pergunta 1: Por que a solução do sistema nas figuras parece depender da condição inicial? A solução não deveria ser independente da condição inicial? Qual o mistério?

Resposta 1: A solução não deveria ser independente da condição inicial, porque a taxa de mensagens endógenas depende da quantidade de pessoas infectadas.

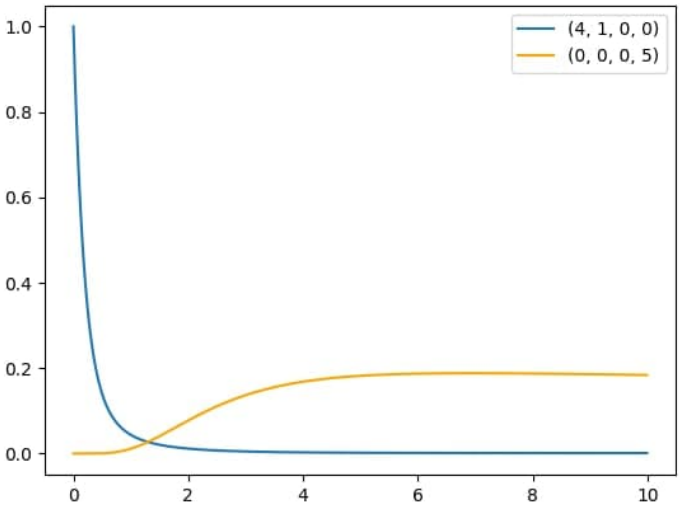


Figure 8: $\mu_0 = 0.2$, $\mu_1 = 0.5$, $\lambda_0 = 1$, $\lambda_1 = 0$

Pergunta 2: Como se comporta a simulação para diferentes condições de estado inicial?

Resposta 2: Como estamos em um sistema exógeno com ambas as taxas > 0, independente do estado inicial, essas taxas são suficientes para eventualmente dar tração ao sistema, de forma que eventualmente ele vai convergir para para a solução estacionária. O estado inicial afeta a velocidade de evolução e convergência, entretanto, o sistema irá inevitavelmente chegar a uma solução estacionária.

6 Conclusão

Esse foi um trabalho desafiador, no qual nos esforçamos para entregar respostas satisfatória. Entretanto, nos deparamos com diversas dificuldades e problemas descobertos de última hora que afetaram nossos resultados. Ademais, conseguimos desenvolver análises condizentes com nossas expectativas e algumas soluções proveitosas, como: o método de calcular o intervalo de confiança com base em dados com diferentes alinhamento; simulador com alto grau de interatividade que nos permitiu uma melhor compreensão do funcionamento do modelo. Abordamos também as dificuldades que tivemos, e algumas superamos, durante a analise com os métodos numéricos de Markov e EDO para o modelo SIS. Por fim, para futuros trabalhos seria interessante abordar soluções para os problemas que nos deparamos: na analise de Markov K=2; na falha nos cálculos das taxas infecção do simulador; na resolução dos casos iniciais diversos da cadeia de Markov K=1; melhorias de performance do simulador em casos com muitos usuários, possivelmente abordando uma solução de paralelização de sua execução.

7 Bibliografia

Dobrow, Robert P. "Introduction to Stochastic Processes with R" (2016).

Ige, Oluwatobiloba, "Markov Chain Epidemic Models and Parameter Estimation" (2020). Theses, Dissertations and Capstones. 1307. https://mds.marshall.edu/etd/1307

7.1 Links

`https://github.com/HeavenVolkoff/fake_news_sis_simulator`
`https://docs.python.org/3/library/heapq.html`
`https://numpy.org/doc/stable/reference/random/generated/numpy.random.exponential.html`

https://en.wikipedia.org/wiki/Poisson_point_process
https://en.wikipedia.org/wiki/Exponential_distribution
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.expon.html#scipy.stats.expon
https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.expm.html
https://www.overleaf.com