**UNIVERSIDADE FEDERAL DO RIO DE JANEIRO**

Instituto de Matemática

Centro de Ciências Matemáticas e da Natureza

Departamento de Ciência da Computação

UFRJ

**ANDRÉ FIGUEIREDO MUNIZ**

**DIEGO MACHADO DE SOUZA**

**FILIPE JOSÉ TEIXEIRA DE OLIVEIRA**

**GIOVANNI APARECIDO DA SILVA OLIVEIRA**

**PAULO RENATO CARVALHO DE AZEVEDO FILHO**

**RODRIGO WERNECK FRANCO**

**YAGO DE ARAUJO SERPA**

# Trabalho Final de Avaliação e Desempenho

**RIO DE JANEIRO, RJ**
**2019**

# 1. INTRODUÇÃO

Este relatório é o produto final da disciplina Avaliação e Desempenho, ministrada no curso de Graduação do BCC/UFRJ, tendo como objetivo descrever a experiência adquirida no estudo da implementação de uma simulação orientada a eventos, em que há fila sem prioridades e filas com distinção de classes com prioridade entre a classe 1 e a classe 2. Aliado a isso, o cálculo também é feito em sua forma analítica, usando Cadeias de Markov, e, consequentemente, há a comparação entre estas soluções.

## 1.1 Funcionamento

O simulador representa o funcionamento de uma fila M/M/1, a qual adota disciplinas de atendimento FCFS e LFCS por um total de 3200 rodadas. É possível para o simulador fazer distinção entre os tipos de fregueses na fila (dependendo da questão do trabalho) e, portanto, há uma modelagem de prioridade na simulação fazendo uso de uma variável booleana que tem função de flag. As estatísticas selecionadas para cada rodada são coletadas já considerando o simulador estando em equilíbrio. Dado isto, o tamanho da rodada é definido pela variável K_SAMPLES e, durante cada uma das 3200 rodadas, calcula-se o IC da média, IC da variância, número médio de pessoas na fila e IC do número de pessoas na fila. No sistema apresentado, um loop principal é executado, buscando o próximo evento da fila de eventos para, então, completar uma série de tarefas, executando as ações pertinentes e necessárias à cada evento e coletando os dados para os fins estatísticos desejados.

Uma representação da fila M/M/1 considerada em algumas simulações pode ser vista no diagrama apresentado na figura 1.1. A taxa de chegada é representada por $\lambda$ e a taxa de serviço é representada por $\mu$. Cada valor da taxa de serviço ($\mu$) como o da taxa de utilização ($\rho$) variam de acordo com as questões e serão explicitados em suas respectivas seções que se seguem.


Figura 1.1

## 1.2 EVENTOS

Dois eventos principais foram escolhidos para representar o funcionamento do sistema, sendo eles:

**Chegada ao sistema (CH).** Evento responsável por tratar uma chegada ao sistema, com casos particulares onde o servidor está ocioso ou ocupado. Caso o servidor esteja em estado *idle* ou ocioso, métricas relativas ao tempo de espera são atualizadas. Independente do estado do servidor, ao final deste evento um novo evento do tipo CH é gerado.

**Saída de serviço (SS).** Evento responsável por tratar uma saída de serviço e armazenar dados, como o tempo de partida do freguês. Caso a fila de espera não esteja vazia, dados adicionais referentes ao número de pessoas na fila são coletados. Ao fim deste evento, caso a fila de espera não esteja vazia, dependendo da política de atendimento um freguês é iniciado o serviço de um freguês da fila.

## 1.3 LINGUAGEM E SEU USO NA GERAÇÃO DE VARIÁVEIS ALEATÓRIAS

Para codificar a simulação foi usada a linguagem de programação Python na versão 3.7.4, por contar com operações pré-programadas, como listas, e do uso contínuo dos integrantes com ela. Isso permitiu aos integrantes focarem suas atenções à implementação dos algoritmos estudados e às análises e interpretações dos resultados encontrados.

A linguagem conta com um gerador de números aleatórios (PRNG - Python Random Number Generator), utilizado através da biblioteca random e que na realidade gera números pseudo-aleatórios através de um processo determinístico, o qual consiste em uma amostra da distribuição Uniforme no intervalo (0,1]. Na versão utilizada da linguagem, o método do PRNG é o Mersenne Twister, que baseia-se no número primo de Mersenne.

## 1.4 ESTRUTURAS INTERNAS

O simulador foi desenvolvido seguindo o paradigma de orientação a objetos e essencialmente consiste em quatro classes distintas. A classe *Event* representa a modelagem do mínimo que um evento necessita ter. A classe *Customer* representa a modelagem de um freguês no sistema. A classe *Simulator* representa a modelagem do simulador, com as estruturas de dados e métodos de interpretação, necessárias para a simulação. A classe *Statistics* representa uma série de estruturas feitas para a coleta de dados por rodada do simulador, além de conter os métodos para cálculo de média e variância.

A fila de eventos foi modelada como uma lista de objeto do tipo *Event*. Para salvar as informações dos fregueses há uma lista. A fila de espera foi modelada como uma lista de objetos do tipo *Customer*, fregueses que chegam ao sistema com ele em estado ocupado são adicionados à esta estrutura de lista.

Para coletar dados e gerar corretamente as estatísticas para cada rodada, foi modelada uma lista de entidades do tipo *Statistics*.

## 1.5 PRECISÃO DO INTERVALO DE CONFIANÇA

O intervalo de confiança (IC) da T-Student é calculado ao término da coleta de dados em todas as rodadas.

Através do método *calculate_t_student_nq*, no qual é definido um intervalo de confiança de 95% e 1 grau de liberdade, é possível calcular o número de clientes médio na fila. Análogamente, temos o método *calculate_t_student_w* para calcular o tempo de espera através de uma T-Student. Algumas das fórmulas para o cálculo do intervalo de confiança estão demonstradas abaixo.

A fórmula $p = t_{1-\alpha/2;n-1} \times \frac{\hat{\sigma}}{\hat{\mu}\sqrt{n}}$ faz a verificação para encontrar a precisão no IC da T-Student. Na fórmula, *n* é o número de rodadas, o qual é fixado em 3200 durante todo o trabalho, *n-1* é a quantidade de graus de liberdade na T-Student, $\hat{\mu}$ é uma média estimada que pode ser encontrada utilizando a fórmula $\sum_{n=1}^{n} \frac{X_i}{n}$ onde $X_i$ é a média do tempo de espera na fila da rodada *i.* O desvio padrão é dado por $\hat{\sigma}$, que é encontrado fazendo $\sum_{n=1}^{n} \frac{(X_i - \bar{X})^2}{n-1}$, dado que $\bar{X}$ é a média anteriormente encontrada.

Executado os cálculos anteriores, alterou-se repetidas vezes o valor de *k* até que o valor retornado pela precisão atenda o alvo de até *5%*.

**Desafios e Soluções**

Um dos primeiros desafios encontrados pelo grupo ao começar a realização do projeto foi com relação à seção 3 da descrição do trabalho, na qual o cenário abordado é uma fila com dois fluxos Poisson e sem prioridades. Tendo em vista que a Cadeia de Markov Tempo Contínuo necessária para modelar uma fila desse tipo deve ser infinita, temos que o número de pessoas na fila (Nq) pode variar de 0 a infinito, no entanto sua tendência é permanecer próximo do valor médio. Além disso, uma abordagem para resolver este tipo de cadeia de forma algorítmica é truncá-la, dado que esta não tem como ser resolvida através de um algorítmo se for infinita.

Uma solução possível para resolver esta questão da cadeia ser infinita é truncar a matriz que define os estados. O grupo resolveu este desafio adicionando um parâmetro na definição da função até onde esse truncamento pode ser feito. Um ponto muito relevante a ser analisado é que, com a finalidade de se obter um resultado mais preciso, testamos um número muito grande para truncamento, e foi observado que a execução da cadeia torna-se um tanto mais lenta quanto maior for este número de truncamento, todavia os valores se aproximam cada vez mais dos valores analíticos.

Através do código é possível gerar diferentes versões da tabela com maior ou menor profundidade de truncamento, dessa forma agilizando ou reduzindo a velocidade de execução, assim como aumentando ou diminuindo a precisão da cadeia.

Além disso, dentro da função de realizar o plot do gráfico ficam localizados os cálculos dos valores analíticos para cada um dos lambdas (dentro do mesmo for), equações do lambda, rho, cada um dos w's, que são todos valores analíticos. O sistema gera isso para todos os possíveis valores de lambda e plota no gráfico junto de cada um dos resultados da cadeia de markov.

## 1.6 MÉTODO BATCH

Tal método consiste na coleta de medidas durante uma rodada de tamanho K, para N rodadas, onde tanto K quanto N são pré-definidos. O método executa rodadas sucessivamente, sem a interrupção do simulador ao fim da rodada. A mesma semente é utilizada durante toda a simulação. A fase transiente desse método não precisa ser descartada, apesar de no escopo deste trabalho de simulação ela está sendo desprezada. A rodada é finalizada quando k fregueses forem atendidos pelo sistema, ou seja, k coletas estatísticas sejam feitas.

O sistema de cores foi implementado de tal forma a respeitar as seguintes regras:

1. O tamanho da rodada define o número de amostras que serão coletadas durante aquele intervalo.
2. Se houverem chegadas enquanto a última amostra da rodada i estiver sendo coletada, as mesmas terão as estatísticas descartadas na rodada i+1.

## 1.7 COLETA DAS ESTATÍSTICAS

As seguintes métricas foram definidas para calcular a média e a variância do tempo de espera da fila:

$X_{ik}$ ⇒ Amostra *k* da rodada *i*, a qual é dada por *tempo de simulação de entrada no servidor - tempo de simulação de chegada ao sistema*.

$X_i$ ⇒ Tempo médio de espera na fila da *i-ésima* rodada com *k* fregueses.

$\widehat{\mu}$ ⇒ Média estimada do tempo de espera na fila, a qual foi calculada com a soma dos tempos de espera de uma determinada rodada seguida da divisão pelo número de coletas, para, então, obter a média estimada por rodada.

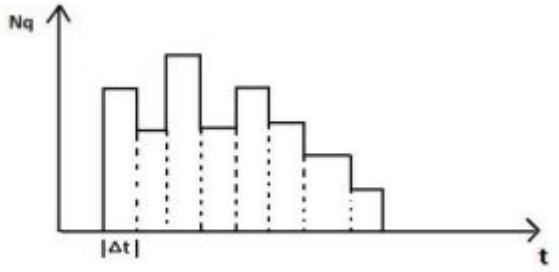
$\widehat{\sigma^2}$ ⇒ Variância estimada do tempo de espera na fila.

$p$ ⇒ Precisão para o intervalo de confiança usando a T-Student, dado pela fórmula apresentada na seção 1.4

Para encontrar o intervalo de confiança da média usando a T-Student utilizou-se:

$$\widehat{\mu} - t_{1-\alpha/2;n-1} \times \frac{\widehat{\sigma}}{\widehat{\mu}\sqrt{n}} \leq \mu \leq \widehat{\mu} + t_{1-\alpha/2;n-1} \times \frac{\widehat{\sigma}}{\widehat{\mu}\sqrt{n}}$$

O número médio de pessoas na fila de espera será dado pela expressão (área / tempo total), onde a área é a área do gráfico (Número de pessoas na fila de espera x Δt)




Cada modificação do número de pessoas na fila de espera influencia diretamente o crescimento ou decrescimento do eixo Y do gráfico em questão. Nq é representado pelo tamanho da lista de fregueses e Δt o tempo de simulação de saída de serviço da amostra i - tempo de simulação de saída de serviço da amostra (i-1).

Logo para cada rodada é feito a seguinte fórmula:

$$\frac{1}{tempo\ total} \sum_{i=1}^{n} Nq_{ik} * \Delta t = Nq$$

**Fórmulas**

**Média do número de clientes na fila $E[N_q]$ (FCFS)**

$$E[N_q] = \frac{\rho^2}{1-\rho}$$

**Variância do número de clientes na fila $V[N_q]$ (FCFS)**

$$V[N_q] = \frac{\rho^2(1+\rho-\rho^2)}{(1-\rho)^2}$$

**Média do tempo de espera na fila $E[W]$ (FCFS)**

Temos que $T = W + X$, logo:

$$W^*(s) = \frac{T^*(s)}{X^*(s)}$$

Assim a relação entre o número de chegadas ao intervalo em que ocorreram $N(z) = T(\tau - \tau z)$ nos force $T^*(s)$. Fazendo a substituição $z = \frac{\tau - s}{\tau}$:

$$T^*(s) = \frac{(1-\rho)s\gamma^*(s)}{s-\tau+\tau X^*(s)}$$

Logo:

$$W^*(s) = \frac{(1-\rho)s}{s-\tau+\tau X^*(s)}$$

Finalmente, se substituirmos a transformada conhecida $X^*(s) = \frac{\mu}{\mu + s}$ para um serviço exponencial, podemos linearizar a equação, deixando em função de s e assumindo s=0, conseguimos obter:

$$E[W] = -W^*(s) = \frac{\rho}{1-\rho}$$

**Variância do tempo de espera na fila $V[W]$ (FCFS)**

$$V[W] = \frac{\rho(2-\rho)}{(1-\rho)^2}$$

**Variância do tempo de espera na fila $V[W]$ (LCFS)**

$$V_{LCFS}[W] = \frac{\rho(2-\rho)+\rho^3}{(1-\rho)^3}$$

**Fórmulas da questão 8**

Para resolver a seção 8 do trabalho, consideramos o enunciado e as fórmulas propostas, clientes de 2 classes (1 e 2) com escalonamento FCFS dentro de cada classe e prioridade da classe 1 sobre a classe 2, sem preempção.

E[U] = trabalho pendente no sistema, no instante de chegada do cliente
E[W] = tempo médio de espera na fila de espera de um cliente típico deste sistema
E[G] = período ocupado do sistema
$E[Xr_i]$ = vida residual do serviço dos clientes do tipo *i*
E[Xr] = vida residual do tempo de serviço do servidor

E[W*i*] = tempo médio na fila de espera dos clientes do tipo *i*

$\lambda = \lambda_1 + \lambda_2$, $p_1 = \lambda_1/\lambda$ e $p_2 = \lambda_2/\lambda$

$E[W] = p_1E[W_1] + p_2E[W_2]$

$E[W_1] = \rho_1E[Xr_1]/(1 - \rho_1)$

$E[W] = \rho E[Xr]/(1 - \rho)$

$E[U] = \rho E[Xr]/(1 - \rho)$

$E[X] = (\rho_1/\rho)E[X_1] + (\rho_2/\rho)E[X_2]$

$E[Xr] = (\rho_1/\rho)E[Xr_1] + (\rho_2/\rho)E[Xr_2]$

$E[G] = E[X]/(1 - \rho)$

$E[U] = \rho_1E[W_1] + \rho_2E[W_2] + \rho E[Xr] = \lambda_1E[X_1]E[W_1] + \lambda_2E[X_2]E[W_2] + \rho E[Xr]$

$\rho_1E[W_1] + \rho_2E[W_2] + \rho E[Xr] = \rho_1/\rho E[W_1] + \rho_2/\rho E[W_2] = \rho E[Xr]/1 - \rho \Rightarrow \rho_1E[W_1] + \rho_2E[W_2] = \rho^2 E[Xr]/1 - \rho$

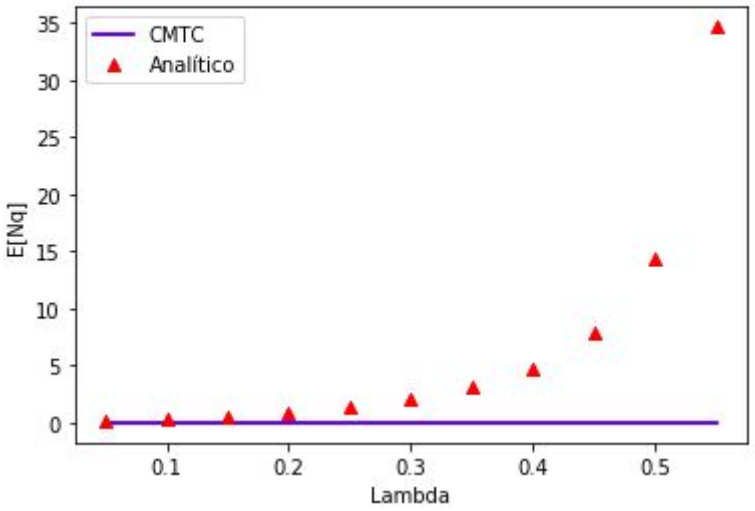
## 1.8 A SIMULAÇÃO

A simulação obteve os seguintes tempos de execução, em uma máquina virtual simulada através do site https://repl.it/. O nome do site RPL significa Read-Eval-Print-Loop (uma tradução literal seria Ler-Avaliar-Imprimir-Loop), e é um ambiente de programação interativo controlado através de um console, no qual várias pessoas podem editar o mesmo código simultaneamente, além de ser possível para todos ver o output no console.
A máquina virtual que o grupo usou possui as seguintes especificações:

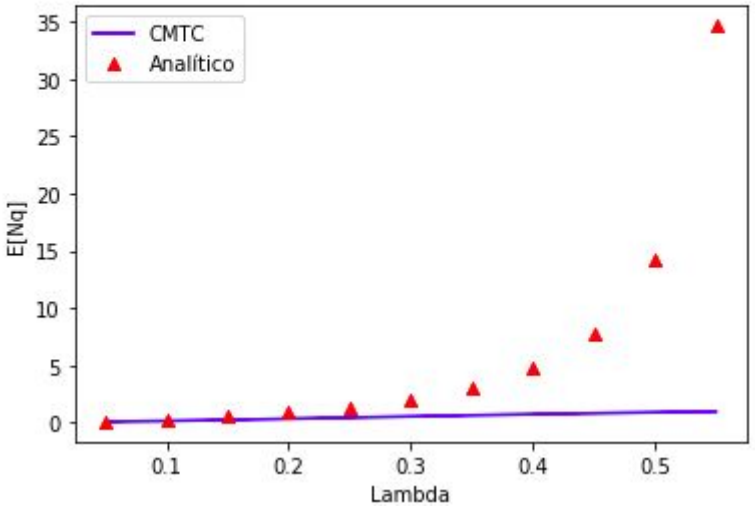
Sistema: Linux
Versão: #38~16.04.1-Ubuntu SMP Tue Jun 25 15:30:46 UTC 2019
Arquitetura: x86_64
Processador: intel i5 7400
Memória : 8GB

### Resultados em gráficos

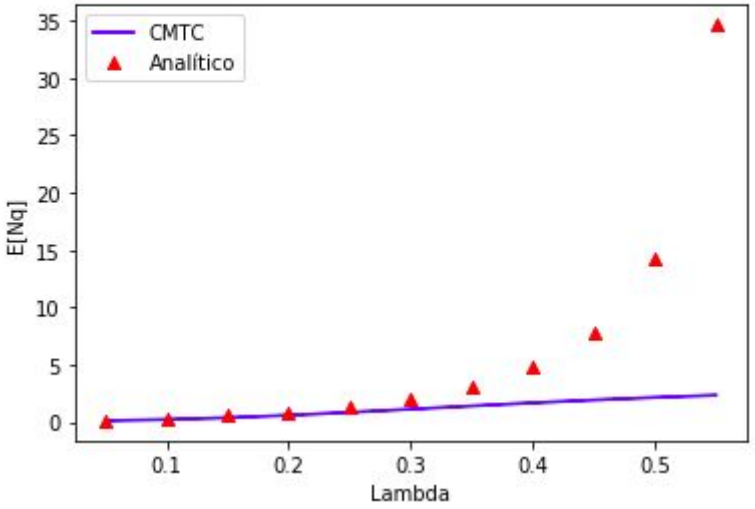
l1=[0.05, 0.55], l2=0.2
m1=1.0, m2=0.5
#states=1



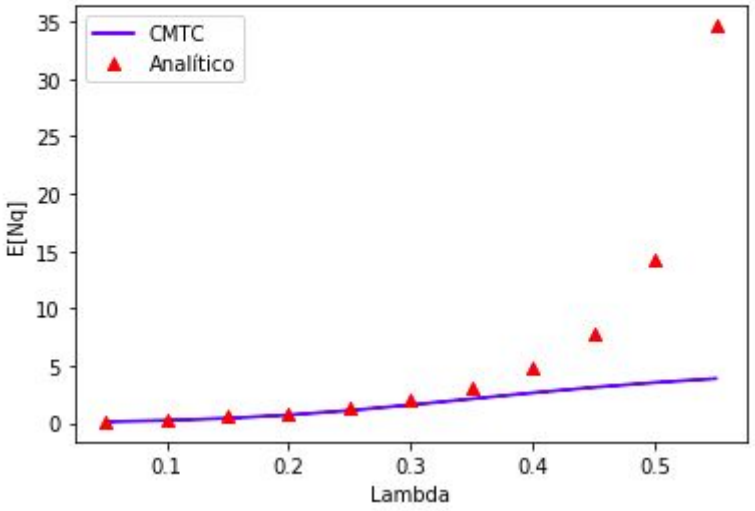

l1=[0.05, 0.55], l2=0.2
m1=1.0, m2=0.5
#states=6



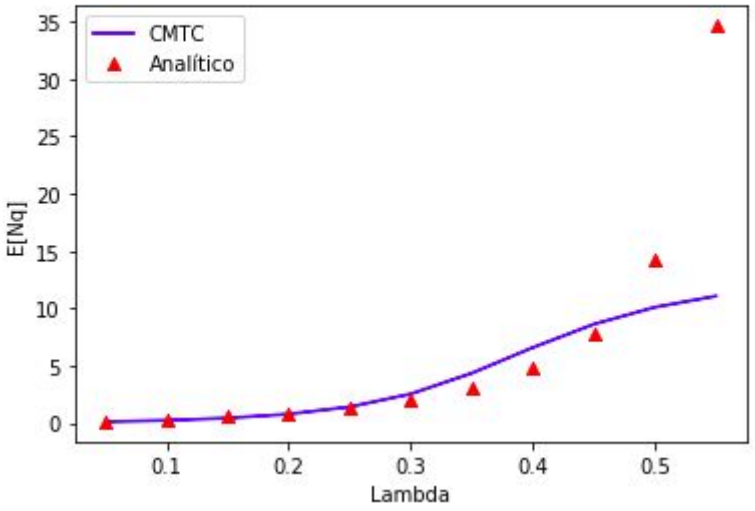
l1=[0.05, 0.55], l2=0.2
m1=1.0, m2=0.5
#states=15



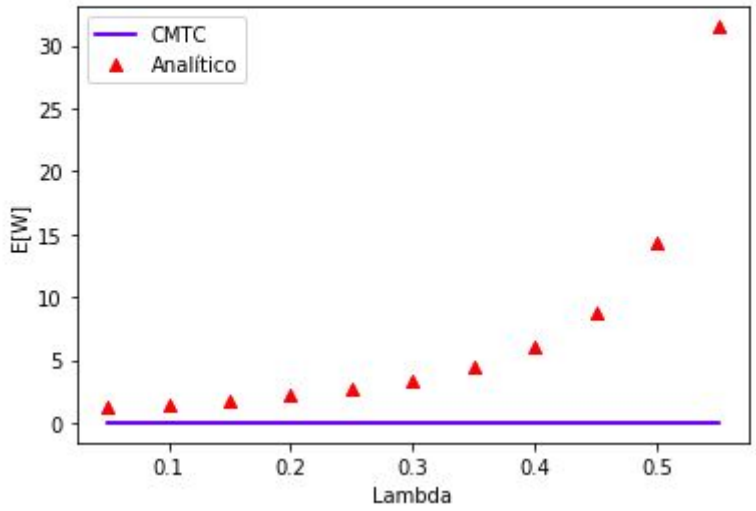

l1=[0.05, 0.55], l2=0.2
m1=1.0, m2=0.5
#states=28



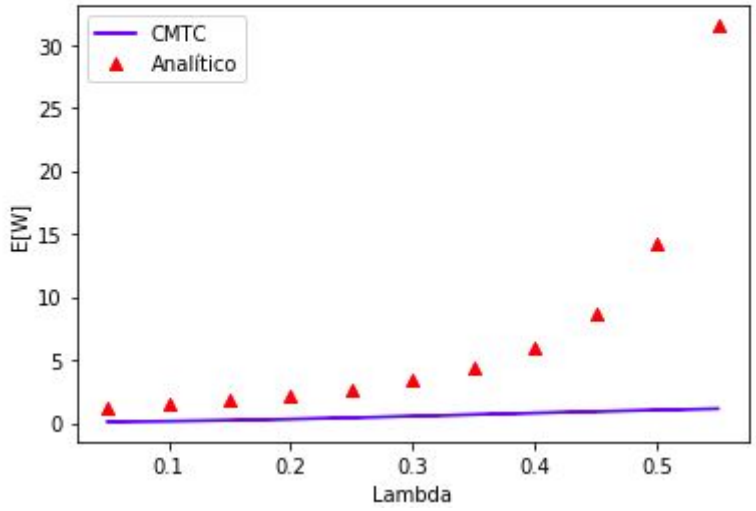
```
l1=[0.05, 0.55], l2=0.2
m1=1.0, m2=0.5
#states=120
```



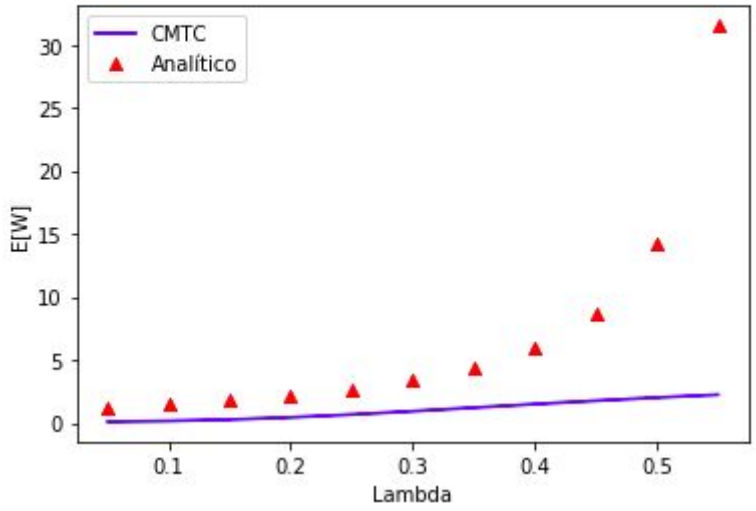
l1=[0.05, 0.55], l2=0.2
m1=1.0, m2=0.5
#states=1



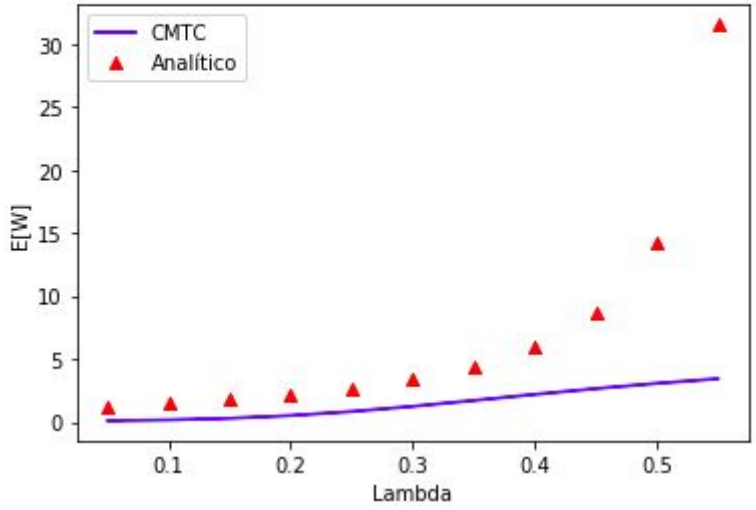

l1=[0.05, 0.55], l2=0.2
m1=1.0, m2=0.5
#states=6



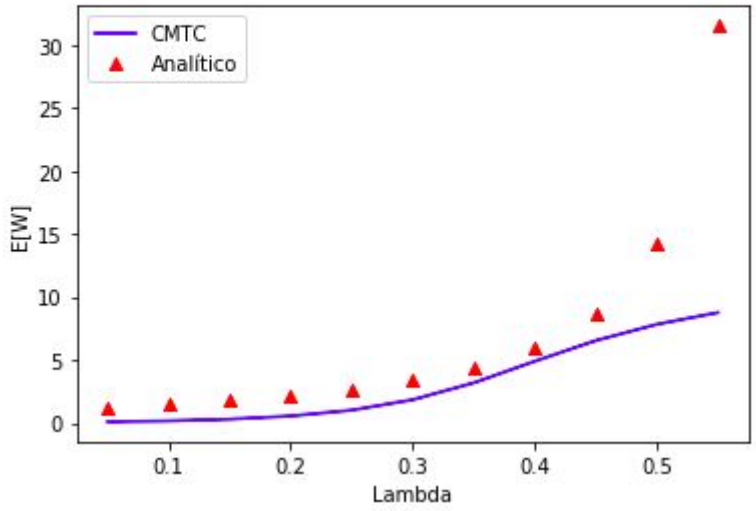
l1=[0.05, 0.55], l2=0.2
m1=1.0, m2=0.5
#states=15




l1=[0.05, 0.55], l2=0.2
m1=1.0, m2=0.5
#states=28



```
l1=[0.05, 0.55], l2=0.2
m1=1.0, m2=0.5
#states=120
```




**Código fonte**

O código do simulador e da cadeia de markov pode ser encontrado no github (https://github.com/filipeoliveira/MM1Simulator-AD2019.2) ou no repl.it (https://repl.it/@AndreMuniz/matplotlib-graph)