Universidade Federal do Rio de Janeiro

Departamento de Ciência da Computação

Simulador de Filas com Prioridade

Avaliação e Desempenho

Professor: Paulo Henrique de Aguiar Rodrigues

Grupo 10:	*DRE:*
Igor Carvalho de Paiva Fonseca	112214399
Mario Cesar Lestro Bonicenha	113054946
Matheus Pinheiro Pinto	112191208

Todos os integrantes participaram de todas as reuniões para discussão do trabalho, tanto presencialmente quanto remotamente. Todos participaram da implementação do simulador e da escrita deste relatório. Inicialmente, as discussões envolveram esboçar a estrutura do gráfico e iniciar a implementação, durante essa parte, todos trabalharam juntos presencialmente. Quando já havia um esqueleto do simulador, cada um remotamente continuou a implementação realizando as tarefas definidas na reunião. Por fim, algumas tarefas foram iniciadas mais especificamente. Mario ficou inicialmente com os testes de correção, gerando casos de teste determinístico, Matheus ficou mais responsável pelo cálculo das métricas ao final da simulação e Igor ficou com a apresentação gráfica dos resultados. Ambos os integrantes foram essenciais para a realização do trabalho.

12 de dezembro de 2017

Siglas

VA - Variável Aleatória

IC - Intervalo de Confiança

ρ - Taxa de utilização = 2λ.

λ - Taxa de entrada.

μ - Taxa de serviço.

T_1 - Tempo total de execução da fila 1.

X_1 - Tempo do primeiro serviço.

W_1 - Tempo de espera da fila 1.

N_1 - Número de pessoas na fila 1.

N_{q1} - Número de pessoas em espera na fila 1.

N_{s1} - Número de pessoas do tipo 1 executando.

T_2 - Tempo total de execução da fila 2.

X_2 - Tempo do segundo serviço.

W_2 - Tempo de espera da fila 2.

N_2 - Número de pessoas na fila 2.

N_{q2} - Número de pessoas em espera na fila 2.

N_{s2} - Número de pessoas do tipo 2 executando.

Sumário

Capítulo 1

Introdução

1.1 Funcionamento Geral do Simulador

O trabalho desenvolvido tem como objetivo simular o comportamento de um sistema com duas filas disputando um único servidor. A Fila 1, com maior prioridade, tem chegadas exógenas segundo uma *Poisson* com taxa λ e novas chegadas interrompem um possível serviço do freguês em execução da Fila 2, a qual não possui chegadas exógenas. Existe continuidade no serviço interrompido e ambas as filas operam no regime FCFS (*First Come First Served*). Quando um freguês do tipo 1 termina de executar o primeiro serviço, o mesmo é adicionado na Fila 2 para sua segunda execução.

Com isso, a implementação do trabalho conta com 7 classes:

- Simulador

  Esta é a classe principal da simulação. Após inicializar as estruturas (que serão descritas na seção 1.5 deste relatório), ela executa dois *loops* principais. O primeiro valida se foi alcançado o término da execução, gera o tempo até a chegada do próximo fregues e, depois, começa a tratar o *loop* da execução de um freguês no servidor. O próximo passo

consiste em atualizar o tempo restante de execução do mesmo, gerando eventos de fim de serviço e atualizando W_1 e W_2. Caso o tempo restante de execução do freguês no servidor seja menor do que o tempo até um próximo freguês chegar, o próximo freguês da fila é selecionado. Caso contrário, tratamos o evento de chegada de um novo freguês, atualizando T_1, T_2, X_1, X_2, N_{q1}, N_{q2}, N_{s1} e N_{s2}, e verificando se o sistema está vazio, colocando o novo freguês direto pra executar. Se, porém, tiver alguém executando, verificamos se é um freguês do tipo 1 ou do tipo 2. Caso seja de 1 o novo freguês será colocado na fila 1, pois não há interrupção na própria fila. Se o freguês for do tipo 2, ocorrerá uma interrupção, onde o programa devolve o freguês que estava executando para o início da fila 2 e coloca em execução o novo freguês.

- Fila

  Esta é a classe que representa a Fila do sistema. Ela contém um atributo representando a prioridade da fila e uma lista com os fregueses que estão nela. Além disso, ela implementa as funções de interação com a fila, isto é, adicionar um novo freguês, selecionar o próximo freguês que irá executar, voltar um freguês para fila e uma função para retornar o tamanho da mesma.

- Fregues

  Esta classe representa o Freguês que deverá executar no sistema. Ela contém os atributos necessários para cada freguês, que são o identificador, tempos de chegada nas filas 1 e 2, tempos de serviço da fila 1 e 2, tempo que ainda resta para executar, prioridade e a cor, que representa de qual rodada é o freguês. Ela também implementa uma função para

atualizar os atributos do freguês, que é utilizada no momento que ele termina a execução do serviço 1 e passa para a fila 2.

- Evento

  Classe que representa o evento gerado no sistema. Ela é composta pelo tempo que ocorreu o evento, o freguês que o gerou, o tipo do evento (chegada ou fim de serviço) e a prioridade (fila 1 ou 2).

- Metrica

  Esta classe é responsável pelo cálculo de todas as métricas após a execução da simulação. Ela possui uma função para calcular o valor das esperanças pedidas no trabalho através das amostras obtidas durante a execução. Ela também possui uma função que calcula a variância das amostras e o desvio padrão, utilizadas no cálculo dos intervalos de confiança.

- Plot

  Esta classe fica responsável por exibir os gráficos de E[W_1], E[W_2], E[N_{q1}], E[N_{q2}], E[N_{s1}] e E[N_{s2}] ao final da execução. Ela é composta por um conjunto de atributos que representam a lista de cada amostra de cada VA, coletadas durante a execução da simulação. Após exibir todos os gráficos, os mesmos são salvos dentro do diretório *./graficos*

- Utils

  Classe composta pela função geradora de um número aleatório dada uma taxa como parâmetro, podendo ser a partir de uma semente ou não.

1.2 Desenvolvimento e ambiente

A linguagem de programação escolhida para o desenvolvimento do trabalho foi *Python 3*, devido principalmente à sua facilidade de uso e pelo grupo já ter conhecimento da mesma. Percebemos que a performance de execução da simulação foi a esperada, considerando que o *Python* é uma linguagem mais lenta que C ou Javascript. Os gráficos das métricas foram desenhados utilizando o módulo *PyPlot* da biblioteca *Matplotlib*. Outras biliotecas também foram úteis para o cálculo das métricas solicitadas, como a *Numpy*, *Scipy* e *Prettytable*. O programa pode ser executado em qualquer Sistema Operacional, bastando apenas ter o *Python 3* instalado. Sua execução é explicada a seguir, na seção 1.4.

Os integrantes do grupo utilizaram seus próprios computadores para desenvolvimento e realização de testes, sendo estes:

- Sistema Operacional Windows 10, processador Intel i7 e 8GB de memória RAM.
- Sistema Operacional Windows 10, processador AMD X4945 e 8GB de memória RAM.
- Sistema Operacional Linux Mint, processador Intel i3 e 8GB de memória RAM.
- Sistema Operacional Ubuntu, processador Intel i5 e 8GB de memória RAM.

Analisando os testes executados, vimos que não tiveram muitas diferenças, mas os testes documentados da fase transiente e testes finais foram feitos nos computadores com Windows e as execuções dos testes determinísticos foram feitos no Ubuntu.

Os tempos de execução para um conjunto de 10000 fregueses leva em torno de 0.19 segundos para executar, enquanto para 10 vezes esse valor leva cerca de 2 segundos. Consideramos o tempo bastante aceitável e não prejudicou a otimização inicial dos parâmetros. Porém, ao colocar valores acima de 10 milhões, o programa consumia muita memória e demorava cerca de 4 minutos, o que não consideramos satisfatório para o sistema.

1.3 Organização

O grupo se organizou através do sistema de versionamento de código **Git** e o **GitKraken**, além do **Google Drive** para compartilhamento de gráficos e logs de execução. Além disso, a ferramenta utilizada para escrever o relatório foi a **ShareLatex**, que recentemente foi acrescida de uma outra ferramenta chamada **OverLeaf** e são bastantes usadas, pois permitem que mais de um colaborador desenvolva o documento ao mesmo tempo, o que facilitou bastante sua escrita. Na maior parte do tempo utilizamos o **Visual Studio Code** para o desenvolvimento e execução dos testes iniciais, porém, devido ao modo debbuger, percebemos que a execução ficava bem mais lenta do que se executássemos diretamente pelo terminal. Assim, passamos a rodar todos os testes na linha de comando.

1.4 Como executar o simulador

Para executar o simulador, primeiramente, é necessária a instalação de algumas bibliotecas do *Python3* que são utilizadas pelo simulador, via terminal. É importante lembrar que se faz necessário estar com permissões de administrador. Obs: caso esteja no Linux utilize *pip3 install* como abaixo.

Caso esteja no windows o equivalente seria *python3 pip install*

- Matplotlib

```
python -mpip install -U pip
python -mpip install -U matplotlib
```

- PrettyTable

```
pip3 install prettytable
```

- Scipy

```
pip3 install scipy
```

- Numpy

```
pip3 install numpy
```

Em seguida, o simulador poderá ser executado através do comando **python3 simulador.py** e deve ser seguido dos seguintes parâmetros:

- numero_rodadas
  Parâmetro que representa o número total de rodadas que a execução terá.

- fregueses_por_rodada
  Parâmetro que representa o número total de fregueses que serão servidos em cada rodada.

- fase_transiente

  Parâmetro que representa a quantidade de fregueses que serão utilizados durante a fase transiente para estabilizar o sistema.

- ρ

  Parâmetro que representa a taxa de utilização do sistema.

(Exemplo) python3 ./simulador.py 10 10000 20000 0.4
Neste caso, estamos executando o simulador com 10 rodadas, 10000 fregueses por rodada, 20000 fregueses na fase transiente e taxa de utilização $\rho = 0.4$

1.5 Eventos, Estruturas Internas, Métricas e Variáveis Aleatórias

Temos dois tipos de eventos no simulador: CHEGADA e FIM_SERVICO. Um evento de chegada ocorre quando um novo freguês chega no sistema ou quando um freguês chega na fila 2 (finalizado seu serviço 1). Um evento de fim de serviço ocorre quando um freguês termina o serviço 1 ou o serviço 2.

A análise da ordem dos eventos e corretude do sistema foi feito utilizando um modelo determinístico que será explicado no *Capítulo 2*.

Utilizamos todas as VA's listadas no início do trabalho e delas obtivemos as métricas de esperança, variância, desvio padrão e consequentemente, calculamos o intervalo de confiança desejado de 95%, utilizando-se, para isso, as amostras armazenadas durante a execução. Os cálculos das métricas não foram complicados, pois colocamos em prática os conhecimentos obtidos na disciplina de Estatística e Probabilidade e durante o início da disciplina de Avaliação e Desempenho.

As estruturas internas utilizadas foram, além de variáveis simples (in-

teiros) e listas, as instâncias das Classes já definidas anteriormente (Fila1, Fila2, Plot e Metricas). Importante lembrar que, inicialmente, calculamos a taxa λ=$\rho/2$, definimos μ=1 e utilizamos essas taxas na função para gerar as chegadas e os tempos de serviço. A geração desses números foi feito através da fórmula

$$\frac{-1 * log(1 - rand[0, 1))}{taxa}$$

com taxa igual a λ para a chegada e igual a μ para serviço. Também inicializamos uma variável que controla o número total de fregueses criados e outra para o número total de fregueses servidos, que são importantes para controlar e terminar o *loop* de execução.

1.6 Método Batch e conceito de Cores

O método escolhido para o cálculo das rodadas foi o Batch. Nele, as medidas são tomadas após certo intervalo de tempo e a simulação continua por outro período de tempo, onde novas medidas são tomadas. Não há necessidade de reiniciar a simulação, logo os fregueses da rodada anterior podem ainda estar no sistema esperando para terminar o serviço e isso irá influenciar em algumas VA's, como o W_2 e N_{q2}.

O conceito de cor foi implementado como um atributo do freguês que representa a rodada que ele pertence, garantindo que os cálculos das métricas são feitos considerando que cada amostra obtida pertence de fato à rodada que se encontra.

A implementação dessa estrutura se deram por utilizarmos uma validação para quando o número de fregueses pertencentes à rodada já tiver sido criada, mudamos para a próxima rodada.

1.7 Escolha de Parâmetros

A escolha de parâmetros se iniciou pela análise dos gráficos gerados para a escolha de possíveis fases transientes. Em seguida, foi observada a relação do aumento da taxa de utilização ρ com a estabilidade do sistema, que será descrita mais adiante no relatório.

Após definirmos as fases transientes, rodamos a simulação diversas vezes, analisando a proximidade com valores analíticos e buscando intervalos de confiança adequados para o trabalho.

A otimização dos parâmetros seguiu um processo empírico, após um conjunto de testes variando o número de rodadas e o tamanho de cada rodada, encontramos valores que achamos justos. Porém, não foi possível obter um resultado totalmente satisfatório para todos os valores de ρ, como será mostrado mais para frente.

ρ	rodadas	fregueses por rodada	fase transiente
0.9	15	10000	10000
0.8	30	800	20000
0.6	25	500	8000
0.4	10	1000	14000
0.2	10	2800	16000

Tabela 1.1: Tabela com parâmetros mais próximos do valor analítico

Capítulo 2

Testes de Correção

Para a comprovação da corretude do código e de toda a Simulação, vamos apresentar cenários onde conhecemos a saída esperada e comentar sobre as boas saídas obtidas a partir das métricas esperadas para as diferentes entradas e rhos.

2.1 Determinismo

Para o código, especificamente, foi implementada uma versão determinística onde podemos saber ao certo qual saída estamos esperando (diferentemente da versão exponencial). Realizamos dois cenários determinísticos para esse fim:

- Cenário 1: 2 fregueses executam consecutivamente sem interrupção e com ocorrência de tempos com o servidor ocioso;
- Cenário 2: 2 fregueses executam com a ocorrência de uma interrupção no meio.

No decorrer dos Cenários Determinísticos, o código funciona através de 3

vetores principais: chegada, tempo de serviço 1 (Xs1) e tempo de serviço 2 (Xs2). Cada posição dos vetores representa um freguês. No Cenário 1 temos as chegadas nos tempos 0 e 4, Xs1 de 1 segundo e 2 segundos e Xs2 de 5 segundos e 2 segundos. Os logs abaixo mostram como seguiu a simulação do Cenário 1.

```
0.0 seg - CHEGADA do fregues 0 de prioridade 1
1.0 seg - FIM_SERVIÇO do fregues 0 de prioridade 1
1.0 seg - CHEGADA do fregues 0 de prioridade 2
2.0 seg - FIM_SERVIÇO do fregues 0 de prioridade 2
4.0 seg - CHEGADA do fregues 1 de prioridade 1
5.0 seg - FIM_SERVIÇO do fregues 1 de prioridade 1
5.0 seg - CHEGADA do fregues 1 de prioridade 2
7.0 seg - FIM_SERVIÇO do fregues 1 de prioridade 2}
```

Como os logs demonstram, tudo sai como o esperado: O primeiro freguês de id 0 entra e executa suas duas tarefas; O servidor fica ocioso até o freguês de id 1 chegar e executar suas tarefas também; As Médias também saem como o esperado:

```
+-------+-------+-------+-------+-------+-------+---------+---------+
| E[T1] | E[W1] | E[X1] | E[T2] | E[W2] | E[X2] | Var[W1] | Var[W2] |
---+-------+-------+-------+-------+-------+-------+---------+------+
|  1.0  |  0.0  |  1.0  |  1.5  |  0.0  |  1.5  |   0.0   |   0.0   |
+-------+-------+-------+-------+-------+-------+---------+---------+
```

Para o Cenário 2 temos uma situação muito importante na simulação: interrupção. Um freguês novo chega no meio da execução de um freguês de prioridade 2. Os vetores de exemplificação de execução serão: Chegadas em

1 segundo e 4 segundos; X1s de 1 segundo cada e X2s de 1 segundo e 2 segundos. O log descreve como acontece a sequência de eventos:

```
1.0 seg - CHEGADA do fregues 0 de prioridade 1
2.0 seg - FIM_SERV do fregues 0 de prioridade 1
2.0 seg - CHEGADA do fregues 0 de prioridade 2
4.0 seg - CHEGADA do fregues 1 de prioridade 1
6.0 seg - FIM_SERV do fregues 1 de prioridade 1
6.0 seg - CHEGADA do fregues 1 de prioridade 2
9.0 seg - FIM_SERV do fregues 0 de prioridade 2
11.0 seg - FIM_SERV do fregues 1 de prioridade 2
```

Todos os eventos acontecem como o esperado. O momento crucial da simulação acontece no tempo 4 segundos, onde o freguês de prioridade 2 que estava a executar pára para o novo freguês de prioridade 1 começar a executar. A programação de eventos acontece normalmente após o ocorrido de acordo com as ordens dos fregueses nas filas.

E[T1]	E[W1]	E[X1]	E[T2]	E[W2]	E[X2]	Var[W1]	Var[W2]
1.5	0.0	1.5	6.0	2.5	3.5	0.0	0.5

Importante chamar atenção para algumas medidas: E[W2] será 2.5 pois o freguês 0 esperará por 2 segundos (entre 2 e 4 segundos) na fila 2 e o freguês 1 esperará por 3 segundos (entre 6 e 9 segundos) na fila 2. Logo teremos uma espera média na fila 2 de 2.5 segundos, exatamente como os logs nos mostram. Com os eventos vitais da simulação demonstrados para poucas pessoas

(para obtermos uma melhor visualização), os mesmos foram testados e funcionam para qualquer número de pessoas, obviamente. Assim testemunhamos que o código funciona corretamente e executa os passos exatos da simulação proposta.

2.2 Intervalos de Confiança para diferentes ρ

Com o objetivo de confirmar que os resultados da Simulação são os esperados e que o Intervalo de Confiança é de boa qualidade, escolhemos um número suficiente de rodadas e de fregueses para rodarmos o projeto com diferentes valores de ρ. O mais importante desse tópico é, na verdade, atestar que os resultados analíticos se encontram dentro desse intervalo também. Para isso, utilizamos da planilha disponibilizada pelo professor com resultados das médias para diferentes ρ. Os valores analíticos referentes ao $\rho = 0.2$ são:

Metrica	Valor_Analítico
E[W1]	0.11111
E[Nq1]	0.01111
E[T2]	1.52778
E[W2]	0.52778
E[N2]	0.15278
E[Nq2]	0.05277

Podemos perceber que, com a tabela abaixo, todos os valores analíticos se encontram no Intervalo de Confiança com precisão $< 5\%$, assim como o

esperado.

Metrica	Intervalo_inferior	Intervalo_superior	precisao
E[W1]	0.109977	0.111486	0.68%
E[Nq1]	0.011057	0.011372	1.41%
E[T2]	1.521302	1.531925	0.35%
E[W2]	0.519456	0.527804	0.8%
E[N2]	0.147738	0.155476	2.55%
E[Nq2]	0.050378	0.054051	3.52%

A fase transiente, que discutiremos no Capítulo seguinte, foi escolhida arbitrariamente a partir da análise dos gráficos. Na figura abaixo temos os gráficos referentes a simulação escolhida como exemplo para melhor visualização.

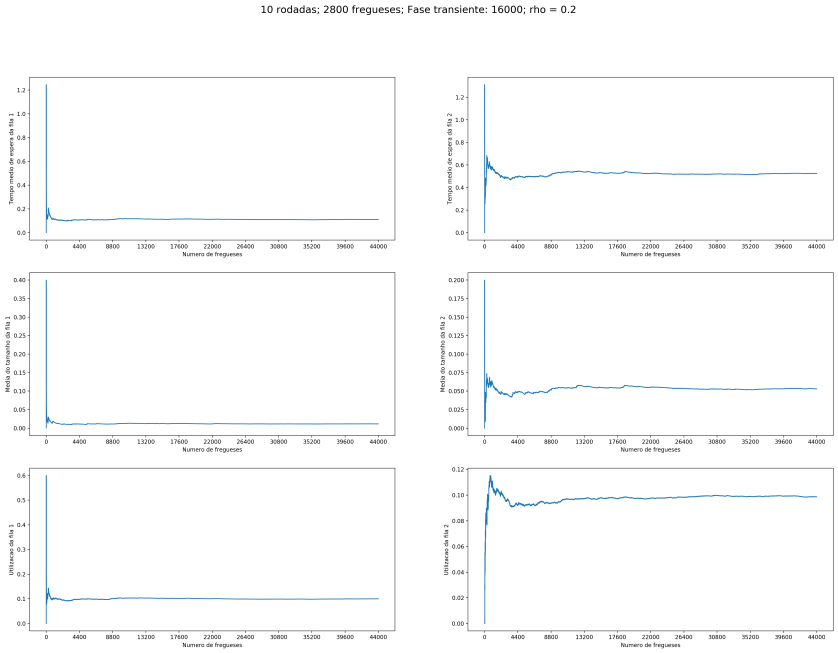

Figura 2.1: Gráficos para $\rho = 0.2$, 2800 fregueses, 10 rodadas e 16 mil de fase transiente

Abaixo deixaremos mais um exemplo de uma saída satisfatória com um $\rho = 0.4$.

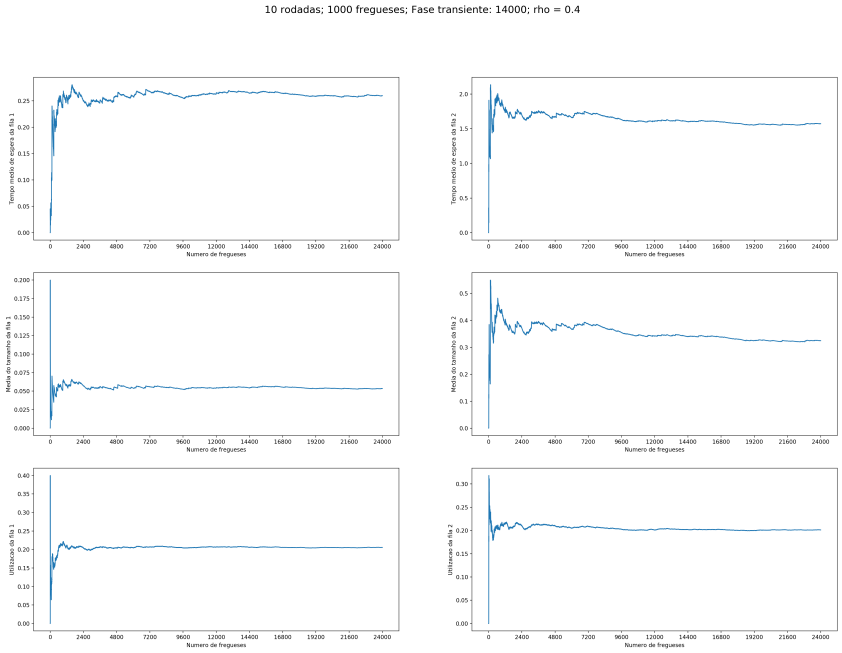

Figura 2.2: Gráficos para $\rho = 0.4$, 1000 fregueses, 10 rodadas e 14 mil de fase transiente

Os valores encontrados para $\rho = 0.4$ seguem na tabela abaixo.

Metrica	Intervalo_inferior	Intervalo_superior	precisao
E[T1]	1.242174	1.267912	1.03%
E[W1]	0.241435	0.25745	3.21%
E[X1]	0.999214	1.011986	0.64%
E[N1]	0.251453	0.257547	1.2%
E[Nq1]	0.049792	0.052408	2.56%
E[Ns1]	0.200739	0.206061	1.31%
E[T2]	2.493691	2.545621	1.03%

E[W2]	1.490443	1.544185	1.77%
E[X2]	1.001015	1.003668	0.13%
E[N2]	0.492993	0.505207	1.22%
E[Nq2]	0.293156	0.304644	1.92%
E[Ns2]	0.19856	0.20184	0.82%

E, por fim, os valores analíticos que se encontram todos dentro dos intervalos citados.

Metrica	Valor_Analítico
E[W1]	0.250000
E[Nq1]	0.050000
E[T2]	2.500000
E[W2]	1.500000
E[N2]	0.500000
E[Nq2]	0.300000

Capítulo 3

Estimativa da Fase Transiente

A fase transiente corresponde ao período inicial em que o sistema não se encontra em equilíbrio. Nesse momento, o cálculo das métricas diverge muito do valor analítico e, o objetivo desta seção é mostrar como foi escolhida a fase transiente para cada ρ observado.

Analisando os diversos testes realizados e os gráficos gerados, obtivemos o valor da fase transiente de forma arbitrária de acordo com nossas observações. A partir dos exemplos a seguir, iremos mostrar como foi feita a decisão de escolha da fase transiente.

Os parâmetros utilizados para os exemplos a seguir foram:

- 5 rodadas
- 20 mil fregueses por rodada
- fase transiente com 0 fregueses
- ρ=0.9

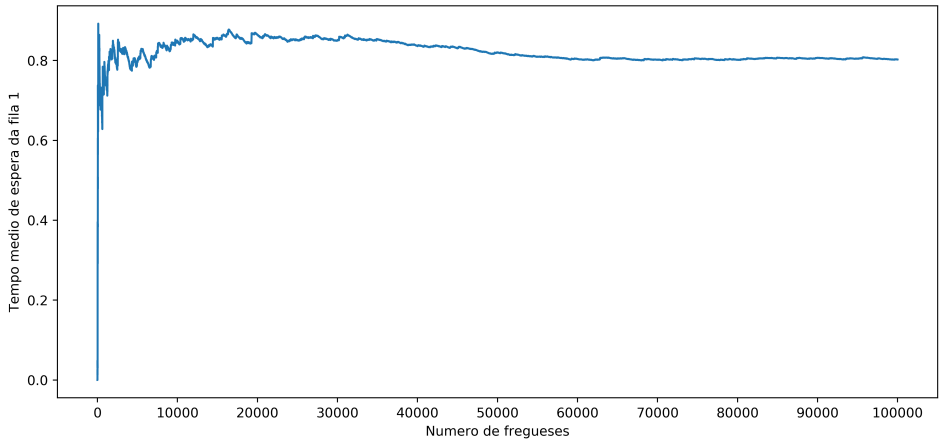

Figura 3.1: Média de N_{q1} necessita de pelo menos 30000 fregueses até alcançar uma possível estabilidade

Para as diferentes médias calculadas, são observadas diferentes necessidades quanto ao intervalo da fase transiente. É de extrema importância a minimização da variação de todos os parâmetros enviesados por esse momento inicial do sistema. Por exemplo, ao se escolher uma fase transiente pequena demais, acabamos por considerar valores muito diferentes dos desejáveis (que são alcançados no estado de estabilidade). A figura 3.2 mostra uma outra métrica da mesma simulação exemplificada acima. Podemos ver que o intervalo da fase transiente necessária para este é muito maior.

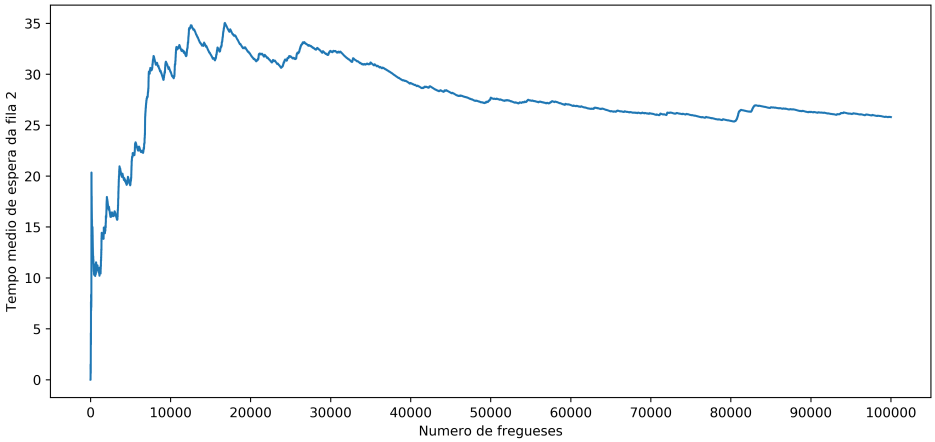

Figura 3.2: Média de N_{q2} necessita de pelo menos 50000 fregueses até alcançar uma possível estabilidade(5 rodadas; 20 mil fregueses; $\rho = 0.9$)

Foi possível observar que rodando o simulador para diversas sementes, a fase transiente foi diferente para cada uma delas, garantindo assim a independência de seu valor. Porém, observamos também que conforme o ρ aumenta, o tamanho da fase transiente também aumenta e isso foi seguido por todas as diferentes sementes testadas.

Para demonstrar que uma fase transiente ruim pode influenciar nos cálculos das métricas, decidimos fixar uma semente para geração dos números pseudo-aleatórios, garantindo assim uma comparação imparcial, vista na tabela 3.1. E com isso também poderemos confirmar a influência do valor de ρ no tamanho da fase transiente. Como decidimos o fim da fase transiente de acordo com a variação das métricas calculadas, até que o sistema se encontre em um possível equilíbrio, podemos garantir que a decisão é independente de qualquer semente. A tabela a seguir mostra as possíveis escolhas do tamanho da fase transiente para cada um dos ρ e médias analisadas.

ρ	$\overline{W_1}$	$\overline{W_2}$	$\overline{N_{q1}}$	$\overline{N_{q2}}$	$\overline{N_{s1}}$	$\overline{N_{s2}}$
0.2	20000	20000	25000	25000	15000	10000
0.4	30000	35000	25000	40000	15000	10000
0.6	30000	40000	30000	40000	15000	10000
0.8	35000	30000	35000	45000	15000	10000
0.9	40000	50000	40000	55000	15000	15000

Tabela 3.1: Exemplo com 5 rodadas, 20000 fregueses por rodada, $0.2 < \rho < 0.9$ e semente fixa

Ao compararmos os valores obtidos com os resultados analíticos, podemos perceber que a simulação com a fase transiente mais adequada apresenta métricas mais próximas da desejada, como podemos ver na tabela abaixo.

Resultado	$\overline{W_1}$	$\overline{N_{q1}}$	$\overline{T_2}$	$\overline{N_2}$	$\overline{W_2}$	$\overline{N_{q2}}$
Sem Transiente	0.6964	0.2832	12.3125	4.9322	11.3174	4.5395
Transiente	**0.5971**	**0.2301**	**10.4876**	**4.135**	**9.4719**	**3.7227**
Analítico	0.4286	0.1286	4.6428	1.3928	3.6428	1.09286

Tabela 3.2: Teste com 1000 fregueses, 10 rodadas, $\rho = 0.8$ e seed '51'.

A implementação da fase transiente no simulador consistiu em considerarmos a rodada de índice 0 do vetor de rodadas como representante do período transiente. Quando o programa atinge o número especificado como *fase_ transiente*, a mesma muda para a rodada seguinte e os próximos fregueses gerados já não irão compor a fase transiente. Vale lembrar que, apesar dos valores serem armazenados no vetor de rodadas, sempre com o índice 0, o cálculo das métricas só considera as rodadas 1 para cima, não interferindo nos valores finais. O armazenamento só foi mantido para a exibição final dos gráficos. Outro ponto importante do código foi a utilização de identificadores

negativos para marcar os fregueses da fase transiente e cada um, no momento da chegada no sistema, recebe um atributo cor, que corresponde à rodada atual do sistema.

Capítulo 4

Resultados e comentários pertinentes

A tabela abaixo é composta pelos valores analíticos das médicas de cada VA utilizada no trabalho. Com isso, nossos testes e otimizações tiveram como foco principal alcançar valores próximos o suficientes para que os valores analíticos se encontrem dentro do intervalo de confiança gerado.

ρ	$\overline{W_1}$	$\overline{N_{q1}}$	$\overline{T_2}$	$\overline{N_2}$	$\overline{W_2}$	$\overline{N_{q2}}$
0.9	0.81818	0.36818	26.36364	11.86364	25.36364	11.41364
0.8	0.66667	0.26667	11.66667	4.66667	10.66667	4.26667
0.6	0.42857	0.12857	4.64286	1.39286	3.64286	1.09286
0.4	0.25000	0.05000	2.50000	0.50000	1.50000	0.30000
0.2	0.11111	0.01111	1.52778	0.15278	0.52778	0.05278

Tabela 4.1: Tabela com valores analíticos

4.1 Testes para análise da fase transiente

4.1.1 Fase Transiente = 0

Como teste inicial vamos mostrar os resultados das métricas sem a escolha de fase transiente arbitrária. Estes resultados podem ser vistos na tabela a seguir:

Resultado	$\overline{W_1}$	$\overline{N_{q1}}$	$\overline{T_2}$	$\overline{N_2}$	$\overline{W_2}$	$\overline{N_{q2}}$
Sem Transiente	0.22187	0.0398	2.204867	0.4032	1.230011	0.2132
Analítico	0.25	0.05	2.5	0.5	1.5	0.3

Tabela 4.2: Teste com 1000 fregueses, 5 rodadas, $\rho = 0.4$ e seed '42'.

Como podemos analisar, as métricas calculadas pela simulação ainda não estão próximas da planilha disponibilizada com os valores analíticos. Visto isso, iremos analisar alguns dos gráficos gerados para escolha da fase transiente na figura 4.1 a seguir.

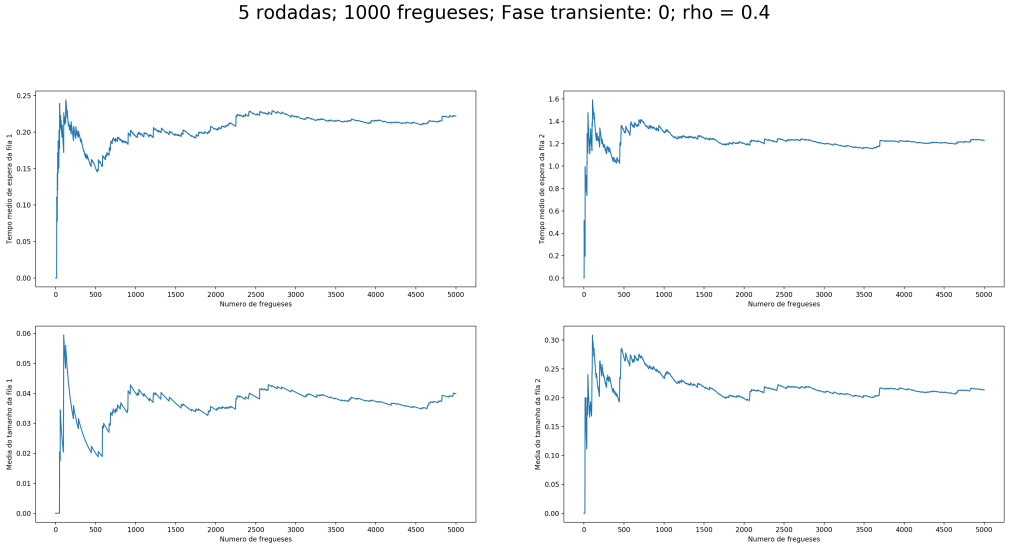

Figura 4.1: Gráficos das métricas: $\overline{W_1}$, $\overline{W_2}$, $\overline{Nq_1}$ e $\overline{Nq_2}$, respectivamente

Analisando estes gráficos podemos perceber que um tamanho de fase transiente bom seria de 3000.

4.1.2 Fase Transiente = 3000

Rodando o simulador com o tamanho da fase transiente 3000, as seguintes métricas foram calculadas:

Resultado	$\overline{W_1}$	$\overline{N_{q1}}$	$\overline{T_2}$	$\overline{N_2}$	$\overline{W_2}$	$\overline{N_{q2}}$
Com Transiente	0.234951	0.0416	2.467196	0.4924	1.444128	0.2904
Analítico	0.25	0.05	2.5	0.5	1.5	0.3

Tabela 4.3: Teste com 1000 fregueses, 5 rodadas, fase transiente = 3000, $\rho = 0.4$ e seed '42'.

Como se pode observar, ao utilizar uma fase transiente baseada nos gráficos, as métricas se aproximam dos valores analíticos. Sendo assim, podemos observar que a escolha do tamanho da fase transiente é essencial para o bom cálculo das métricas.

4.2 Alcançando Resultados Próximos

Com os repetitivos testes e análises para os diferentes valores de ρ, começamos a nos aproximar dos valores desejados. O objetivo foi buscar parâmetros, que variam para cada ρ, que pudesse deixar o intervalo de confiança suficientemente pequeno, para termos 95% de certeza e que o valor analítico esteja dentro deste intervalo.

Métricas	Valor analítico	Intervalo inferior	Intervalo superior	Precisão
$\overline{W_1}$	0.25000	0.23539	0.25759	4.5%
$\overline{N_{q1}}$	0.05000	0.047378	0.05416	**6.68%**
$\overline{T_2}$	2.50000	2.39796	2.516522	2.41%
$\overline{N_2}$	0.05000	0.466565	0.517473	**5.17%**
$\overline{W_2}$	1.50000	1.395047	1.519614	4.27%
$\overline{N_{q2}}$	0.30000	0.275533	0.319852	**7.44%**

Tabela 4.4: Exemplo com 13 rodadas, 800 fregueses por rodada, 10000 fase transiente e ρ=0.4

Como podemos verificar na tabela acima, todos os intervalos de confiança obtidos então englobando os valores analíticos. Porém, as métricas $\overline{N_{q1}}$, $\overline{N_2}$ e $\overline{N_{q2}}$ estão com a precisão acima de 5%. Com isso, vamos nos aproximando dos resultados finais desejados, faltando apenas pequenos ajustes nos parâmetros.

4.3 Resultados finais

Após alguns testes, alcançamos uma configuração de parâmetros satisfatória que gerava, em quase todos os casos, intervalo de confiança contendo o valor analítico, como mostram as tabelas abaixo.

4.3.1 $\rho = 0.2$

Métricas	Valor analítico	Intervalo inferior	Intervalo superior	Precisão
$\overline{W_1}$	0.11111	0.109977	0.111486	0.68%
$\overline{N_{q1}}$	0.01111	0.011057	0.011372	1.41%
$\overline{T_2}$	1.52778	1.521302	1.531925	0.35%
$\overline{N_2}$	0.15278	0.147738	0.155476	2.55%
$\overline{W_2}$	0.52778	0.519456	0.527804	0.8%
$\overline{N_{q2}}$	0.05278	0.050378	0.054051	3.52%

Tabela 4.5: Exemplo com 10 rodadas, 2800 fregueses por rodada, 16000 fase transiente e ρ=0.2

4.3.2 $\rho = 0.4$

Métricas	Valor analítico	Intervalo inferior	Intervalo superior	Precisão
$\overline{W_1}$	0.25000	0.241435	0.257450	3.21%
$\overline{N_{q1}}$	0.05000	0.049792	0.052408	2.56%
$\overline{T_2}$	2.50000	2.493691	2.545621	1.03%
$\overline{N_2}$	0.50000	0.492993	0.505207	1.22%
$\overline{W_2}$	1.50000	1.490443	1.544185	1.77%
$\overline{N_{q2}}$	0.30000	0.293156	0.304644	1.92%

Tabela 4.6: Exemplo com 10 rodadas, 1000 fregueses por rodada, 14000 fase transiente e ρ=0.6

4.3.3 $\rho = 0.6$

Métricas	Valor analítico	Intervalo inferior	Intervalo superior	Precisão
$\overline{W_1}$	0.42857	0.420673	0.428875	0.97%
$\overline{N_{q1}}$	0.12857	0.122482	0.129198	2.67%
$\overline{T_2}$	4.64286	4.612096	4.709075	1.04%
$\overline{N_2}$	1.39286	1.390194	1.426926	1.30%
$\overline{W_2}$	3.64286	3.600061	3.705579	1.44%
$\overline{N_{q2}}$	1.09286	1.092836	1.128444	1.60%

Tabela 4.7: Exemplo com 25 rodadas, 500 fregueses por rodada, 8000 fase transiente e ρ=0.6

4.3.4 $\rho = 0.8$

Métricas	Valor analítico	Intervalo inferior	Intervalo superior	Precisão
$\overline{W_1}$	0.66667	0.6582930	0.6699340	0.88%
$\overline{N_{q1}}$	0.26667	0.2634730	0.2720270	1.60%
$\overline{T_2}$	11.6667	11.299051	11.776312	2.07%
$\overline{N_2}$	4.66667	4.4977880	4.7034620	2.24%
$\overline{W_2}$	10.6667	10.298935	10.771808	2.24%
$\overline{N_{q2}}$	4.26667	4.1050080	4.3081580	2.41%

Tabela 4.8: Exemplo com 30 rodadas, 800 fregueses por rodada, 20000 fase transiente e ρ=0.8

4.3.5 $\rho = 0.9$

Métricas	Valor analítico	Intervalo inferior	Intervalo superior	Precisão
$\overline{W_1}$	0,81818	0.816033	0.823436	0.45%
$\overline{N_{q1}}$	0,36818	0.368108	0.370172	0.28%
$\overline{T_2}$	26,36364	25.431143	26.768573	2.56%
$\overline{N_2}$	11,86364	24.433306	25.770238	2.66%
$\overline{W_2}$	25,36364	11.443883	12.047104	2.57%
$\overline{N_{q2}}$	11,41364	10.993069	11.596771	2.67%

Tabela 4.9: Exemplo com 15 rodadas, 10000 fregueses por rodada, 10000 fase transiente e ρ=0.9

Todas as precisões calculadas estão dentro dos 5%, satisfazendo o intevalo de confiança proposto, menos as variâncias utilizando a *chi-square*, pois o número de rodadas foi menor que 3070.

4.3.6 Considerações Finais

Apesar todos os nossos resultados terem sido satisfatórios, não consideramos a variância utilizando o método *chi-square*, pois o mesmo só entraria na precisão de 5% com no mínimo 3070 rodadas. E isso tornava nossos resultados inviáveis, pois a precisão das outras métricas ficava muito próxima de 0% e nunca obtinhamos o valor analítido dentro do intervalo. Além disso, quando aumentávamos o número de fregueses e com um número de rodadas muito alto, a execução do simulador se tornava custosa ao armazenar os valores na memória.

Capítulo 5

Otimização

A otimização dos parâmetros seguiu um processo empírico e, após um conjunto grande de testes, escolhemos os valores que retornavam um melhor resultado. Porém, não foi possível obter um resultado totalmente satisfatório para todos os valores de ρ, como foi mostrado na seção anterior.

Com isso, chegamos em um resultado final, com os seguintes fatores mínimos para o nosso simulador:

ρ	fator mínimo
0.9	160000
0.8	44000
0.6	20500
0.4	24000
0.2	44000

Tabela 5.1: Tabela com valores analíticos

Como relatamos anteriormente, a precisão relacionada ao Chi Square não foi satisfatória para os casos mostrados acima. Isso acontece pois para conse-

guirmos uma porcentagem dentro da nossa meta de 5%, o número de rodadas precisa ser, pelo menos, 3070. Porém, ao executarmos o simulador com um número tão alto de rodadas, a precisão das médias fica muito pequena, impossibilitando assim ao nosso grupo de conseguir verificar alguma saída satisfatória para o Intervalo de Confiança diante dos resultados analíticos. Tendo afirmado nossa presente dificuldade no encontro de parâmetros que satisfizessem o resultado analítico (dentro de nosso Intervalo de Confiança do Chi Square), este ocorrido foi relatado na seção 6.1 deste relatório. Mesmo com os resultados não satisfatórios para uma das métricas desejadas, podemos perceber, porém, que as medidas estão sim muito próximas as desejadas. Abaixo mostramos um resultado muito perto do que deveria ser encontrado.

Metrica	Intervalo_inferior	Intervalo_superior	precisao
E[T1]	1.250255	1.250561	0.01%
E[W1]	0.250153	0.250326	0.03%
E[X1]	1.000102	1.000235	0.01%
E[N1]	0.251168	0.25127	0.02%
E[Nq1]	0.050442	0.050488	0.05%
E[Ns1]	0.200726	0.200782	0.01%
E[T2]	2.49452	2.494779	0.01%
E[W2]	1.496556	1.496651	0.0%
E[X2]	0.997905	0.998186	0.01%
E[N2]	0.501275	0.501424	0.01%
E[Nq2]	0.29934	0.299487	0.02%
E[Ns2]	0.201934	0.201938	0.0%
V[W1] t_student	0.548608	0.549174	0.05%
V[W2] t_student	5.687628	5.689774	0.02%
V[W1] chi_square	0.522431	0.577424	5.0%
V[W2] chi_square	5.414468	5.984414	5.0%

Figura 5.1: Log de saída do simulador

Como podemos ver no resultado acima para $\rho = 0.4$, número de rodadas = 3070, número de fregueses por rodada = 70 e fase transiente de tamanho 0, os resultados foram muito perto do desejado. Abaixo estaremos comparando os valores para melhor visualização e comparação.

Médias	Analítico	Intervalo Inferior
E[W1]	0.25000	0.250153
E[Nq1]	0.05000	0.050442
E[T2]	2.50000	2.49452
E[N2]	0.50000	0.501275
E[W2]	1.50000	1.496556
E[Nq2]	0.30000	0.29934

Tabela 5.2: Tabela com valores analíticos e intervalos para comparação

Capítulo 6

Conclusão

O desenvolvimento e construção deste trabalho sobre simulação foi de fundamental importância para a fixação da matéria dada em sala de aula. Também foi importante para entender na prática os detalhes da implementação de um simulador e o conceito de teoria de filas, assim como suas dificuldades, que iremos apresentar em seguida.

6.1 Dificuldades durante o desenvolvimento e Otimizações

Inicialmente, uma das dificuldades que foram encontradas foi saber como avaliar a corretude da lógica implementada. Porém, casos de testes determinísticos foram pensados para que os testes de correção fossem executados. E a partir daí foi possível observar que a saída esperada da *D/D/1*, foi de fato a que ocorreu, como mencionamos na seção 2.1 deste relátorio.

Outra dificuldade encontrada por nosso grupo: utilizar as amostras que obtivemos na implementação para calcular as métricas necessárias. Principalmente os cálculos da variâcia utilizando a *t-student* e a *chi-square*. Estes

cálculos parecem corretos, porém para a maioria das execuções as precisões estão muito pequenas, interferindo diretamente no cálculo do IC. Esta foi nossa última e maior dificuldade, encontrar ICs que abrangessem os valores analíticos disponibilizados na planilha. A baixa precisão acarretou em ICs muito pequenos, sendo assim na maior parte das execuções os valores analíticos ficam muito próximos do intervalo, porém, dificilmente dentro dele.

6.2 Possíveis melhorias

Devido às dificuldades que descrevemos acima, uma possível melhoria seria calcular a precisão de forma diferente, de maneira a obter um IC maior, onde na maioria das execuções os valores analíticos estivessem contidos nele. Gostaríamos também de implementar um módulo de testes automatizados, onde não precisaríamos realizar de forma manual a checagem dos valores analíticos em relação ao Intervalo de Confiança obtido.

Apêndice A

- Programa

O **código fonte** está na pasta *Simulador* e todas as classes estão no diretório raiz dessa pasta, bastando apenas estar nela para executar o programa conforme explicado inicialmente no relatório. Todas as classes estão bem comentadas e de fácil entendimento e o projeto está armazenado no **Git**, podendo ser encontrado em https://github.com/matheusp08/trabalho-ad-ufrj-2017-2.