

**FUNDAÇÃO UNIVERSIDADE FEDERAL DO AMAZONAS  
FACULDADE DE TECNOLOGIA  
ENGENHARIA DA COMPUTAÇÃO**

**PROGRAMAÇÃO EM TEMPO REAL**

**3ª SIMULAÇÃO DE SISTEMAS**

**Manaus – AM  
2017**

**FELIPE DE MENEZES SANTOS**

**3ª SIMULAÇÃO DE SISTEMAS**

Quarto Relatório da Disciplina de Programação em  
Tempo Real apresentado ao Curso de Engenharia da  
Computação.

**PROFESSOR: ANDRÉ CAVALCANTE**

**Manaus – AM  
2017**

## **SUMÁRIO**

<b>OBJETIVOS</b>	<b>4</b>
<b>INTRODUÇÃO TEÓRICA</b>	<b>5</b>
<b>RESULTADOS</b>	<b>6</b>
<b>CONCLUSÃO</b>	<b>18</b>
<b>REFERÊNCIAS</b>	<b>19</b>

## **OBJETIVOS**

A atividade tem como o propósito simulações do sistema do trabalho anterior e sua execução nem um Sistema Operacional não RT (Real-Time) e em um RTOS (Real-Time Operating System) e a comparação dos resultados alcançados.

## FUNDAMENTAÇÃO TEÓRICA

“No mundo atual, a rapidez nas decisões, nas comunicações e nas atividades em geral, se tornou um dos paradigmas dominantes na Sociedade da Informação. Utiliza-se cada vez mais o termo Tempo Real em diversas situações, às vezes com propriedade, outras apenas com objetivo comercial. De fato, o tempo está sempre presente em todas as atividades mesmo que não seja de forma explícita; as atividades computacionais seguem também essa regra.” [1]

“Para muitos problemas de engenharia atuais, os modelos matemáticos finais costumam ter uma complexidade extremamente elevada que se tornam praticamente impossíveis de serem resolvidos “à mão” pelo engenheiro. Assim, é muito comum o uso de computadores para resolvê-los que aplicam algum método numérico apropriado para o problema. Um dos métodos mais populares é o Método dos Elementos Finitos (MEF), sendo atualmente possível encontrar softwares comerciais especializados para todas as áreas da engenharia (mecânica dos fluidos, mecânica estrutural, vibrações e acústica, térmica,...). Esses softwares fazem uso dos modelos mais fundamentais da engenharia (os mesmos que são estudados nas disciplinas mencionadas anteriormente), porém são aplicados às complexas geometrias das peças das máquinas ou incorporam as não linearidades dos materiais utilizados na fabricação das peças. Independente do tipo de problema que esses softwares resolvem, eles apenas atuam na fase de resolução numérica do modelo, ainda é responsabilidade do engenheiro de selecionar corretamente as variáveis importantes para o seu projeto.” [2]

## PROBLEMA PROPOSTO

A atividade descreve um robô móvel com acionamento diferencial com modelo no espaço de estados seguinte:

$$\dot{x}(t) = \begin{bmatrix} \sin(x_3) & 0 \\ \cos(x_3) & 0 \\ 0 & 1 \end{bmatrix} u(t)$$
$$y(t) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} x(t)$$

Figura 1: Espaço de estados do problema

Onde  $x(t) = [x_c \ y_c \ \theta]^T$ , sendo  $(X_c, Y_c)$  a posição do centro da massa do robô e  $\theta$  a sua orientação  $u(t) = [v \ \omega]^T$  é a entrada do sistema, sendo  $v$  a velocidade linear e  $\omega$  a velocidade angular do robô. A saída do sistema é  $y(t)$ , sendo que a entrada assume os valores no intervalo abaixo:

$$u(t) = \begin{cases} 0 & , \text{ para } t < 0 \\ \begin{bmatrix} 1 \\ 0.2\pi \end{bmatrix} & , \text{ para } 0 \leq t < 10 \\ \begin{bmatrix} 1 \\ -0.2\pi \end{bmatrix} & , \text{ para } t \geq 10 \end{cases}$$

Figura 2: Intervalo de valores que a entrada assume.

## ESTRUTURA DE DIRETÓRIOS

O ambiente possui 3 diretórios: “exerc4”, “lib” e “include”. O diretório “exerc4” contém o main.c, um Makefile além do executável e dos arquivos com os dados que podem ser gerados pela aplicação; o diretório “lib” contém sistema.c, simulacao.c, libIn.c, libOut.c e libX.c que são os arquivos que contém as funções que são compiladas no Makefile deste diretório e gera a biblioteca “libsistema.a” que é chamada na main.c do diretório “exerc4”. O diretório “include” contém apenas os .h correspondentes dos arquivos .c que estão em “lib”.

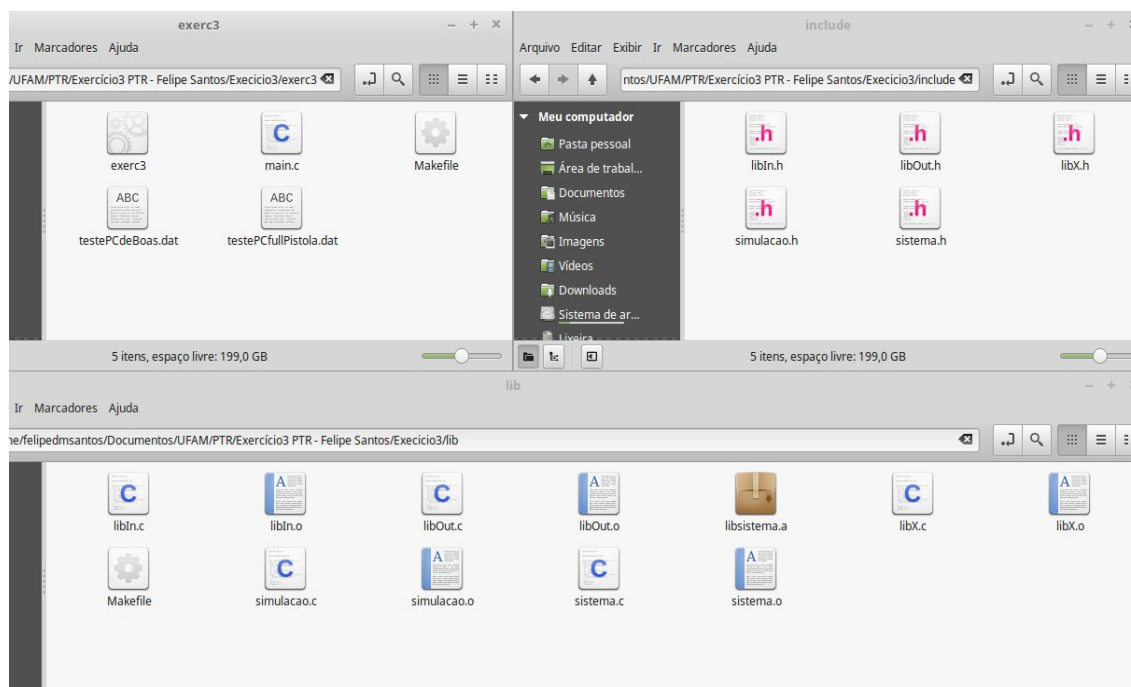
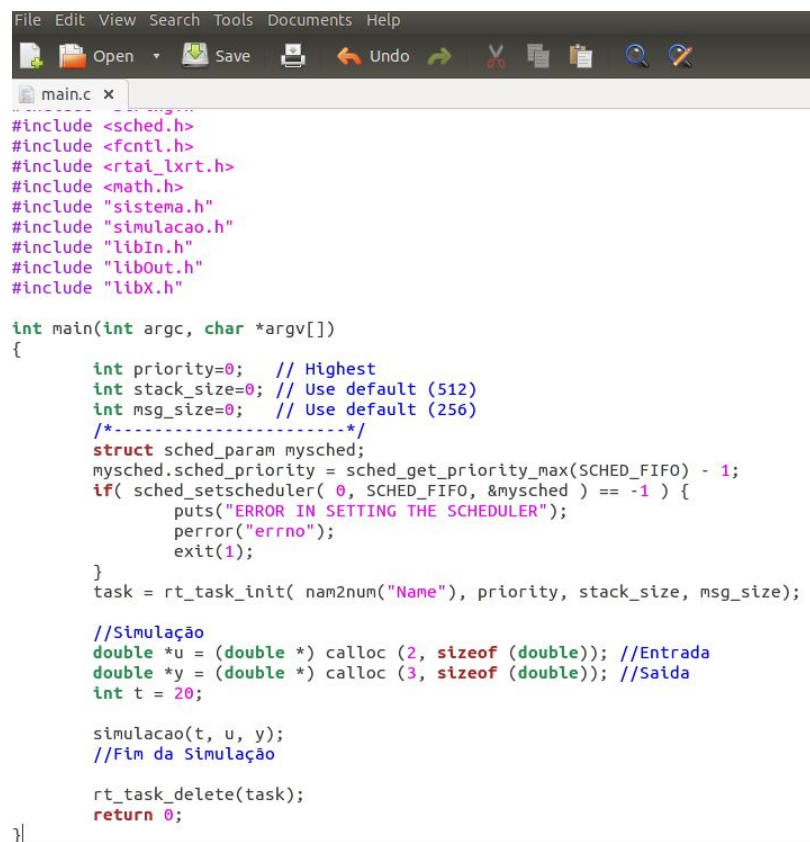


Figura 3: Os três diretórios e os arquivos que os mesmos contém.

## ARQUIVOS FONTES

No arquivos Makefile foram declaradas todas as dependências para cada arquivo de dentro do projeto, dessa forma possibilitando a sua execução abrindo-se um prompt de comando dentro dos diretórios “exerc4” e “lib” do projeto. O nome do executável é “exerc4” e pode ser aberto como mostra nas figuras que seguem. Em relação ao último exercício foram feitas as seguintes modificações na main.c com base no descrito na referência [4] para utilizar os recursos do Ubuntu 14.0 rodando RTAI 4.1 instalado no sistema de acordo com a referência [3].



```
File Edit View Search Tools Documents Help
main.c x
#include <sched.h>
#include <fcntl.h>
#include <rtai_lxrt.h>
#include <math.h>
#include "sistema.h"
#include "simulacao.h"
#include "libIn.h"
#include "libOut.h"
#include "libX.h"

int main(int argc, char *argv[])
{
    int priority=0; // Highest
    int stack_size=0; // Use default (512)
    int msg_size=0; // Use default (256)
    /*-----*/
    struct sched_param mysched;
    mysched.sched_priority = sched_get_priority_max(SCHED_FIFO) - 1;
    if( sched_setscheduler( 0, SCHED_FIFO, &mysched ) == -1 ) {
        puts("ERROR IN SETTING THE SCHEDULER");
        perror("errno");
        exit(1);
    }
    task = rt_task_init( nam2num("Name"), priority, stack_size, msg_size);

    //Simulação
    double *u = (double *) calloc (2, sizeof (double)); //Entrada
    double *y = (double *) calloc (3, sizeof (double)); //Saida
    int t = 20;

    simulacao(t, u, y);
    //Fim da Simulação

    rt_task_delete(task);
    return 0;
}
```



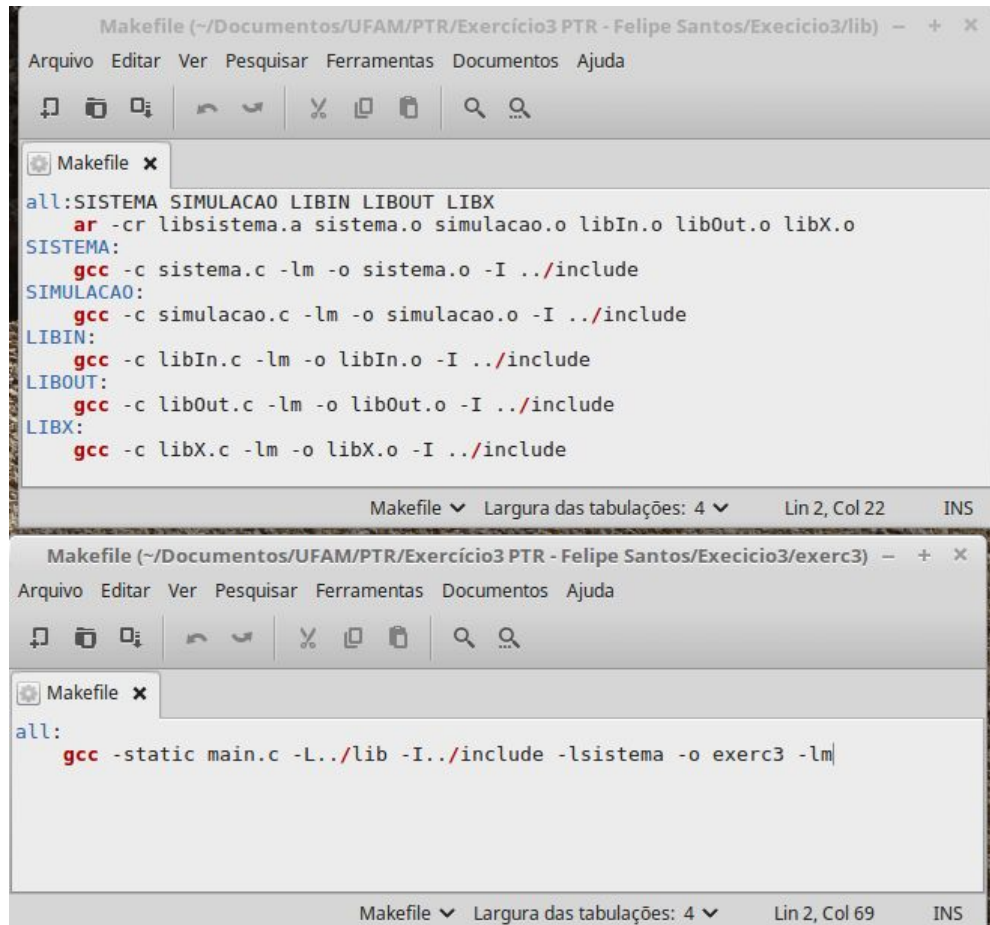


Figura 4: Arquivos Makefiles criados em cada diretório. percebe-se que é praticamente a mesma do exercício anterior.

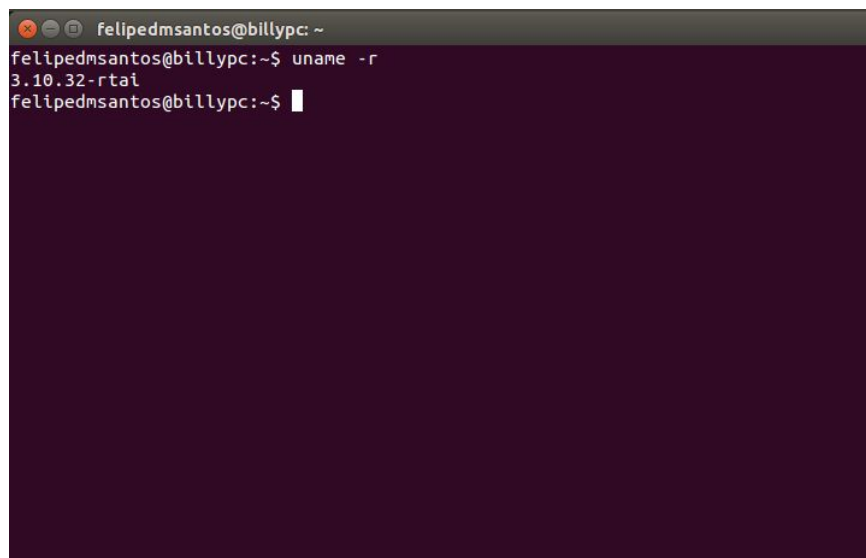


Figura 5: Máquina usada para a execução com kernel do linux 3.10.32 rodando RTAI 4.1

```

felipedmsantos@billypc:~/Área de Trabalho/Execicio4/exerc4$ clear
felipedmsantos@billypc:~/Área de Trabalho/Execicio4/exerc4$ make
gcc -static main.c -L../lib -I../include -lsistema -o exerc4 -lm
felipedmsantos@billypc:~/Área de Trabalho/Execicio4/exerc4$ ./exerc4
Digite o nome do arquivo: teste1
Digite a extensao do arquivo (Exemplo '.txt', '.doc'): .dat
felipedmsantos@billypc:~/Área de Trabalho/Execicio4/exerc4$

```

Figura 6: Como deve ser executado o projeto.

teste.dat x								
0	1.000000	0.628319	0.000000	0.000000	0.000000	98	0	0
1	1.000000	0.628319	-0.809017	0.587785	0.628319	95	-3	-33
2	1.000000	0.628319	-0.618034	1.902113	2.513274	92	-3	-33
3	1.000000	0.628319	0.927051	2.853170	5.654867	93	1	-29
4	1.000000	0.628319	3.236068	2.351141	10.053096	95	2	-28
5	1.000000	0.628319	5.000000	0.000000	15.707963	122	27	-3
6	1.000000	0.628319	4.854102	-3.526712	22.619467	98	-24	-54
7	1.000000	0.628319	2.163119	-6.657396	30.787608	93	-5	-35
8	1.000000	0.628319	-2.472136	-7.608452	40.212386	94	1	-29
9	1.000000	0.628319	-7.281153	-5.290067	50.893801	93	-1	-31
10	1.000000	-0.628319	-10.000000	0.000000	-62.831853	97	4	-26
11	1.000000	-0.628319	-8.899187	-6.465638	-76.026542	94	-3	-33
12	1.000000	-0.628319	-3.708204	-11.412678	-90.477868	93	-1	-31
13	1.000000	-0.628319	4.017221	-12.363735	-106.185832	92	-1	-31
14	1.000000	-0.628319	11.326238	-8.228994	-123.150432	92	0	-30
15	1.000000	-0.628319	15.000000	-0.000000	-141.371669	94	2	-28
16	1.000000	-0.628319	12.944272	9.404564	-160.849544	100	6	-24
17	1.000000	-0.628319	5.253289	16.167961	-181.584055	93	-7	-37
18	1.000000	-0.628319	-5.562306	17.119017	-203.575204	93	0	-30
19	1.000000	-0.628319	-15.371323	11.167920	-226.822990	94	1	-29
20	1.000000	-0.628319	-20.000000	0.000000	-251.327412	95	1	-29

Figura 7: Exemplo de arquivo gerado, onde a sétima coluna é o tempo de resposta de cada iteração em milisegundos, este teste foi feito no modo não RT

teste1.dat x								
0	1.000000	0.628319	0.000000	0.000000	0.000000	98	0	0
1	1.000000	0.628319	-0.809017	0.587785	0.628319	90	-8	-38
2	1.000000	0.628319	-0.618034	1.902113	2.513274	87	-3	-33
3	1.000000	0.628319	0.927051	2.853170	5.654867	89	2	-28
4	1.000000	0.628319	3.236068	2.351141	10.053096	90	1	-29
5	1.000000	0.628319	5.000000	0.000000	15.707963	92	2	-28
6	1.000000	0.628319	4.854102	-3.526712	22.619467	88	-4	-34
7	1.000000	0.628319	2.163119	-6.657396	30.787608	88	0	-30
8	1.000000	0.628319	-2.472136	-7.608452	40.212386	87	-1	-31
9	1.000000	0.628319	-7.281153	-5.290067	50.893801	87	0	-30
10	1.000000	-0.628319	-10.000000	0.000000	-62.831853	88	1	-29
11	1.000000	-0.628319	-8.899187	-6.465638	-76.026542	87	-1	-31
12	1.000000	-0.628319	-3.708204	-11.412678	-90.477868	87	0	-30
13	1.000000	-0.628319	4.017221	-12.363735	-106.185832	87	0	-30
14	1.000000	-0.628319	11.326238	-8.228994	-123.150432	88	1	-29
15	1.000000	-0.628319	15.000000	-0.000000	-141.371669	89	1	-29
16	1.000000	-0.628319	12.944272	9.404564	-160.849544	87	-2	-32
17	1.000000	-0.628319	5.253289	16.167961	-181.584055	88	1	-29
18	1.000000	-0.628319	-5.562306	17.119017	-203.575204	87	-1	-31
19	1.000000	-0.628319	-15.371323	11.167920	-226.822990	87	0	-30
20	1.000000	-0.628319	-20.000000	0.000000	-251.327412	88	1	-29

Figura 8: Teste realizado no modo RT

No arquivo sistema.c temos as funções:

- **double x3 (double \*u, int t )** : Função que calcula o ângulo que será usado nas funções trigonométrica na matriz do problema, tendo como parâmetros os valores do vetor de entrada.
- **double \*ut(int k)**: Retorna o valor do vetor u de acordo com os intervalos especificados no problema.
- **double \*xt(double \*u, int t)**: Retorna o vetor resultante da integral da multiplicação entre a matriz do problema e a entrada u(t);
- **double \*yt(double \*u, int t)**: Retorna o valor da matriz identidade multiplicando o vetor x(t).

No arquivo simulacao.c temos as funções:

- **char \*nomeArquivo()**: Armazena o nome do arquivo que o usuário digitar;
- **FILE \*abreArquivo()**: Retorna o ponteiro que aponta para o endereço do arquivo onde serão gravados os resultados do problema;
- **void simulacao(int t, double \*u, double \*y)**: Recebe como parâmetros o tempo limite da simulação, onde o problema especificou  $t \in [0, 20]$ s além dos endereços de memória alocados para a entrada e a saída do sistema.

No arquivo libIn.c temos a função:

- **double \*dxt(double \*x, double \*u)**: Recebe como parâmetros o vetor x e o vetor u e retorna a função que é derivada de x.

No arquivo libOut.c temos a função:

- **double \*Yt(double \*x)**: Recebe como parâmetro o vetor x a saída y, isto é: o vetor x multiplicado pela matriz identidade.

No arquivo libX.c temos a função:

- **double \*Xt(double \*x, double \*dxt)**: Recebe como parâmetros o vetor x e o vetor dxt e retorna a nova saída de x.

## GRÁFICOS DE SAÍDA GERADOS

Para a geração dos gráficos foi utilizado o software GnuPlot, onde são passadas as colunas do arquivo que serão os eixos:

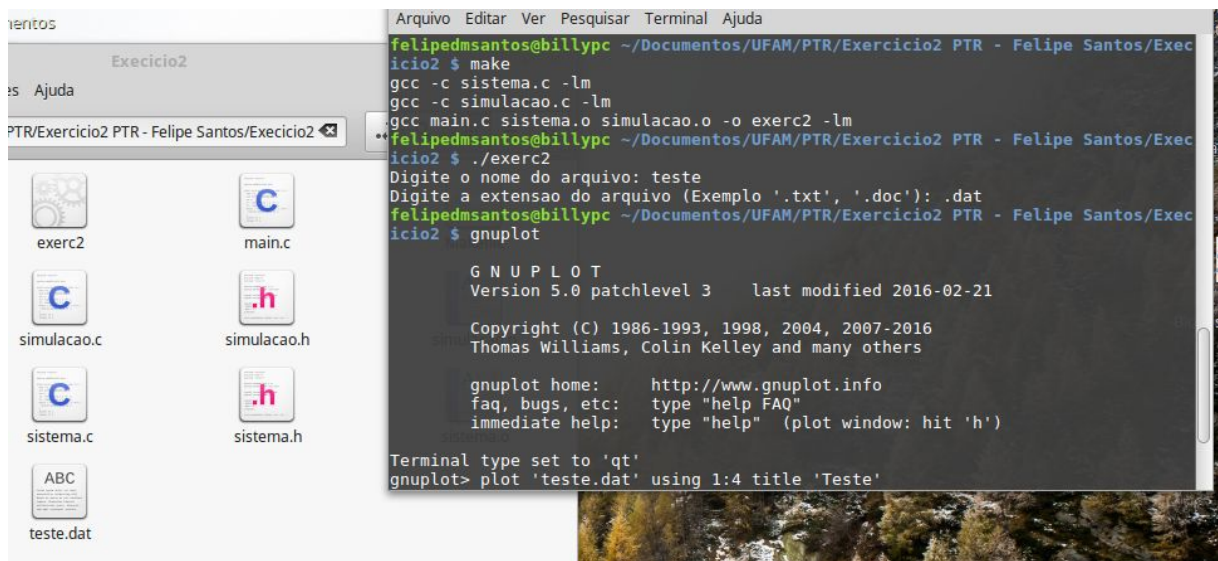


Figura 9: Software GnuPlot iniciando.

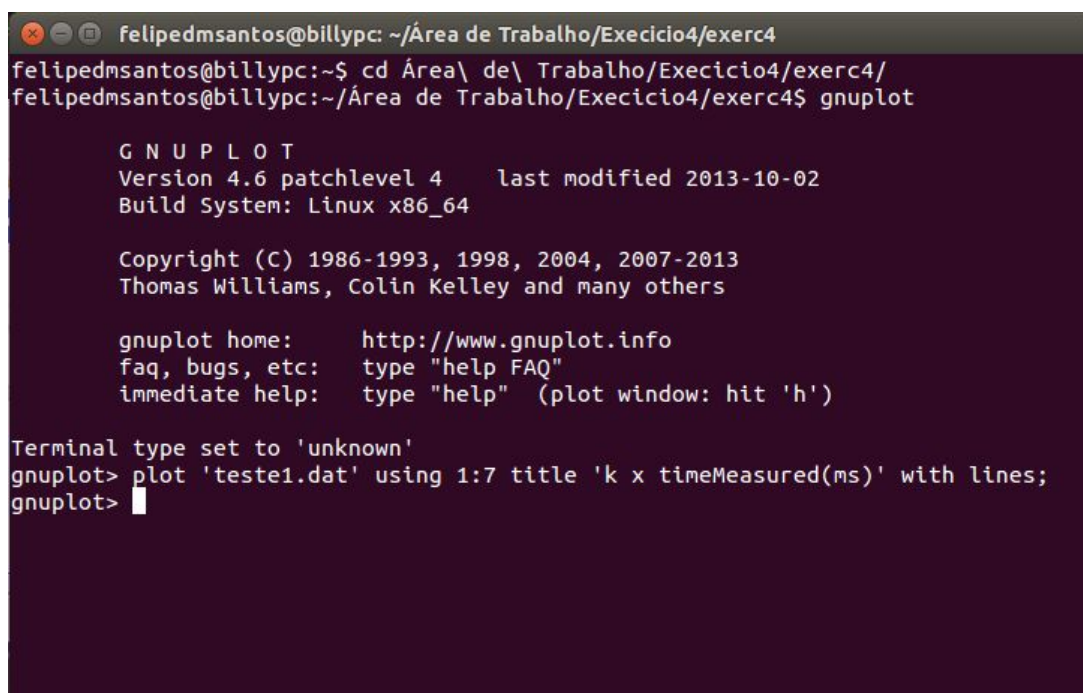


Figura 10: Comando executado para gerar os gráfico dos tempos de resposta.



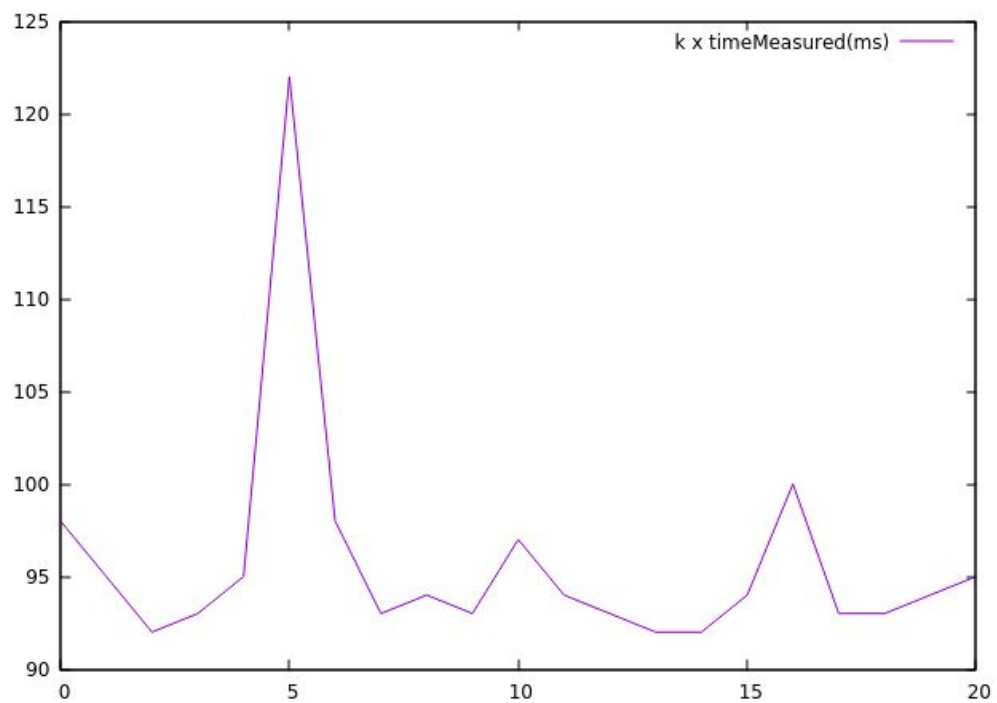


Figura 11: Gráfico do instante do tempo medido em função do número da iteração ( $k$ ) correspondente dentro de  $0 < k < 20$ . Pode-se notar que os valores mais comuns para o tempo de iteração do programa giram no intervalo de 90 ms até 100 ms (Modo não RT)

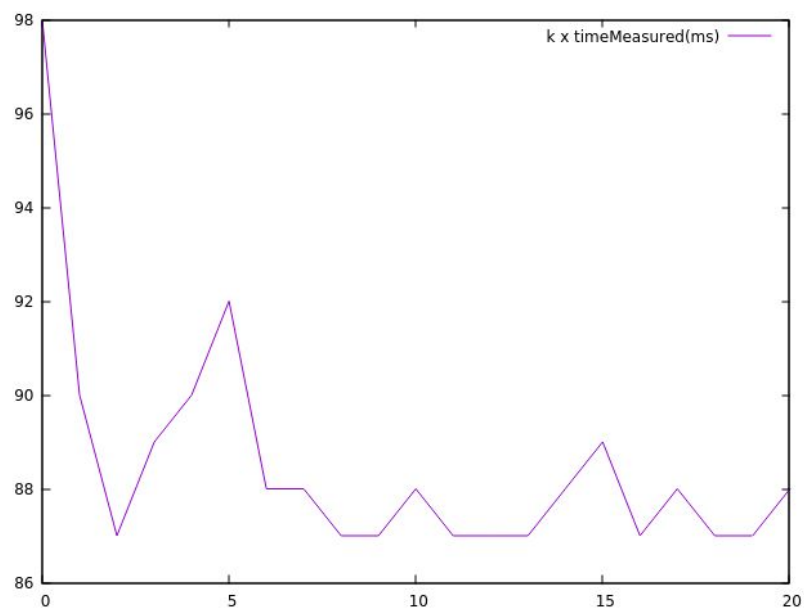


Figura 12: Gráfico do instante do tempo medido em função do número da iteração ( $k$ ) correspondente dentro de  $0 < k < 20$ . (Modo RT)

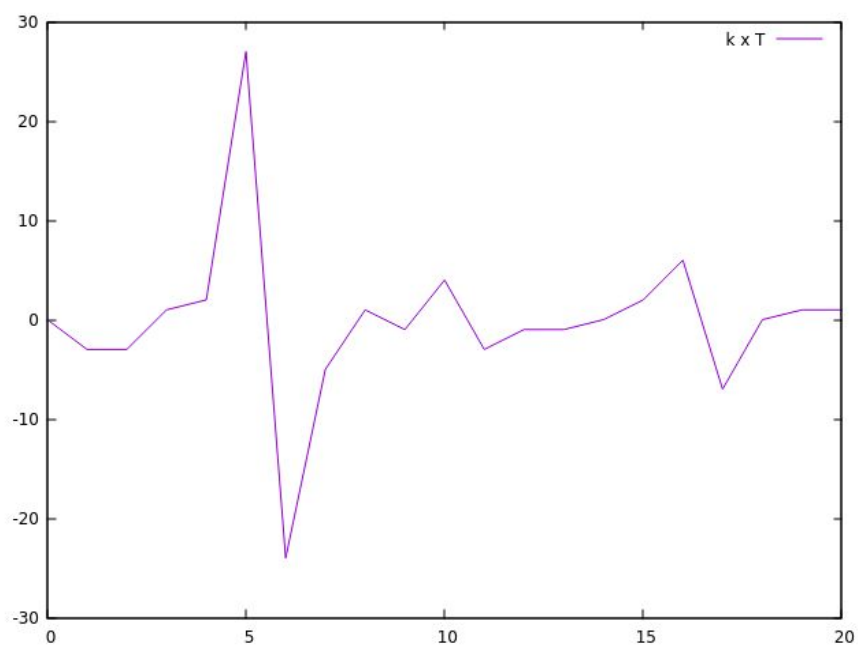


Figura 13: Gráfico do período de amostragem pedido  $T(k)$ , onde  $T(k) = t(k) - t(k - 1)$  e  $0 < k < 20$ . (Modo não RT)

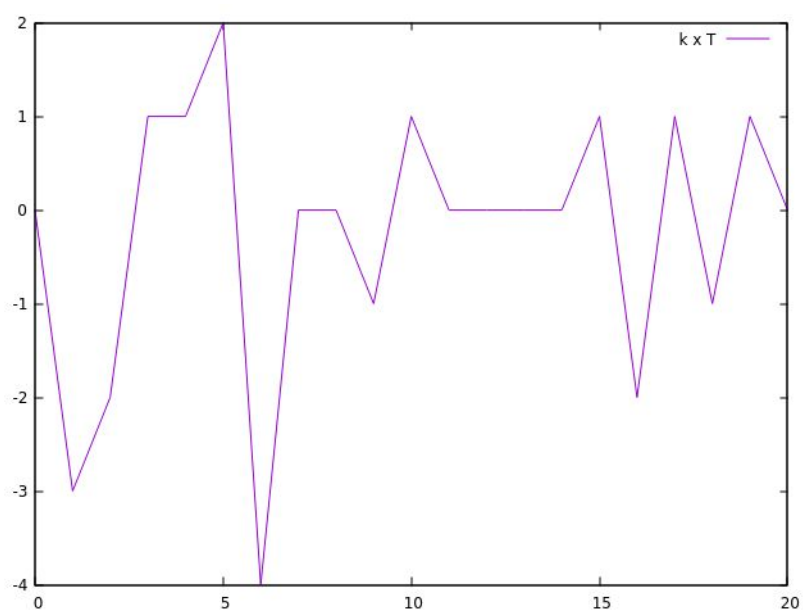


Figura 14: Gráfico do período de amostragem  $T(k)$ . (Modo RT)

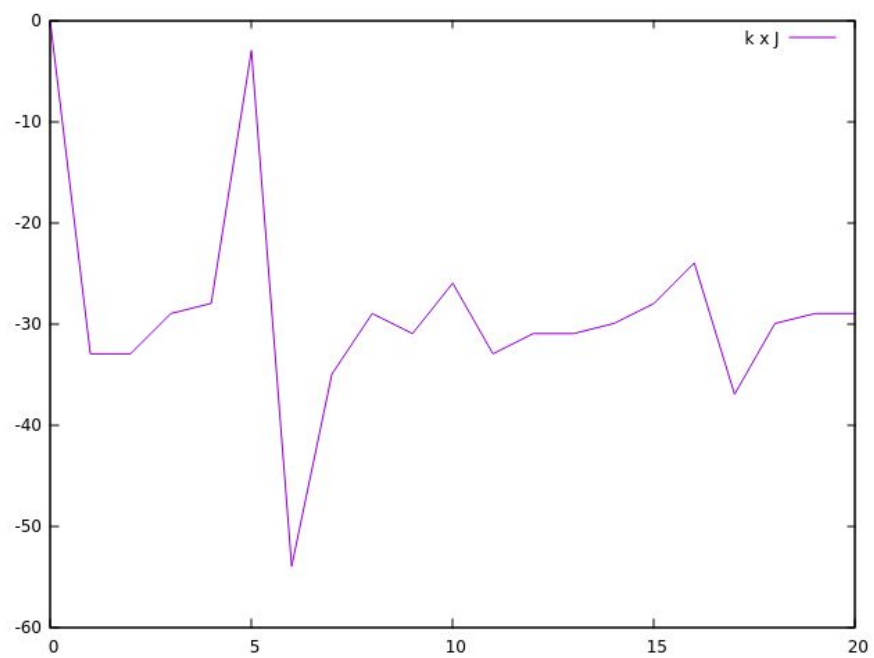


Figura 15: Gráfico de jitter do período de amostragem medido em função de k.  
(Modo não RT)

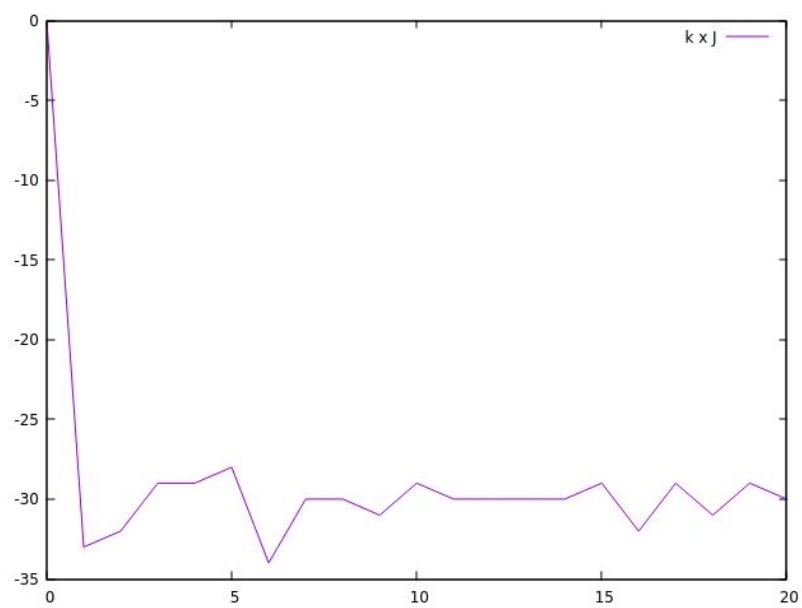
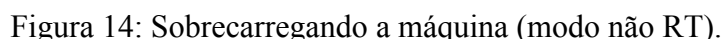


Figura 16: Gráfico de jitter do período de amostragem medido em função de k. (Modo RT)

As tabelas a seguir se tratam de análises estatísticas dos períodos de amostragem onde são calculadas a média, variância, desvio padrão e valores máximos e mínimos em dois casos: no primeiro o computador estava rodando apenas a simulação e no outro sobrecarregamos a máquina rodando várias aplicações ao mesmo tempo além de mandar listar todos os arquivos do disco rígido com o comando “ls -lR /”.





	Média ( $\mu$ )	Variância( $\sigma^2$ )	Desvio Padrão ( $\sigma$ )	Valor Máx	Valor Min
T(k)	-0,14	73,62	8,58	27	-24
J(k)	-28,71	116,91	10,81	0	-54

Tabela 1: Estatística para o caso em que o computador está rodando apenas a simulação.  
(Modo não RT)

	Média ( $\mu$ )	Variância( $\sigma^2$ )	Desvio Padrão ( $\sigma$ )	Valor Máx	Valor Min
T(k)	33,66	7686,03	87,67	417	-20
J(k)	5,09	7627,89	87,33	294	-50

Tabela 2: Estatística para o caso em que o computador está sobrecarregado. (Modo não RT)

	Média ( $\mu$ )	Variância( $\sigma^2$ )	Desvio Padrão ( $\sigma$ )	Valor Máx	Valor Min
T(k)	-0,47	5,26	2,29	2	-8
J(k)	-29,04	49,54	7,03	0	-38

Tabela 3: Estatística para o caso em que o computador está rodando apenas a simulação.  
(Modo RT).

	Média ( $\mu$ )	Variância( $\sigma^2$ )	Desvio Padrão ( $\sigma$ )	Valor Máx	Valor Min
T(k)	-0,23	2,19	1,48	2	-3
J(k)	-28,80	45,76	6,76	0	-33

Tabela 4: Estatística para o caso em que o computador está sobrecarregado. (Modo RT)

Percebe-se que quando o computador, no modo não RT, está sobrecarregado há uma variância, e consequentemente, um desvio padrão significativamente maior nos resultados esperados, além de serem obtidos alguns valores cerca de 5 vezes maiores do que o esperado em algumas iterações na amostra obtida para este relatório. Já no modo RT a diferença de

desempenho com o computador sobrecarregado ou operando em condições normais foi mínima.

## **CONCLUSÃO**

Para a validação de soluções na área da engenharia, principalmente em sistemas de tempo real, as ferramentas estatísticas são de extrema importância, pois elas norteiam a margem de confiabilidade do sistema montado. Desta forma são necessários testes em diversas situações, como foi realizado neste exercício. Foi notado que em uma plataforma onde há sobrecarga de processamento o sistema apresenta grandes variações em seu tempo de resposta, algo que não ocorreu significativamente quando o teste foi realizado em uma máquina com uma interface de tempo real RTAI 4.1 pois a mesma garantiu a estabilidade da resposta do sistema.

## REFERÊNCIAS

- [1] FARINES, J. M.; FRAGA, J. S; OLIVEIRA R. S. Sistemas de Tempo Real. 1. ed. Florianópolis, 2000.
  
- [2] IDAGAWA, H. S. A Importância do Engenheiro no Desenvolvimento de Simulações Computacionais. Revista Intellectus. São Paulo, Ano IX, nº 23.
  
- [3] RTAI Installation for Ubuntu 14.04 - 2017 -  
<<https://gist.github.com/gaoyifan/c881aa36cd02fb5c1c20>> Acesso em 15 out. 2017
  
- [4] How To Port your C++ GNU/Linux application to RTAI/LXRT - 2004 -  
<[http://home.isr.uc.pt/~rui/str/rtai\\_porting.pdf](http://home.isr.uc.pt/~rui/str/rtai_porting.pdf)> Acesso em: 27 out. 2017