

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

PROGRAMAÇÃO EM TEMPO REAL

SIMULAÇÃO DE SISTEMAS

**Manaus – AM
2017**

FELIPE DE MENEZES SANTOS

AMBIENTE DE PROGRAMAÇÃO

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

PROFESSOR: ANDRÉ CAVALCANTE

**Manaus – AM
2017**

SUMÁRIO

OBJETIVOS	4
INTRODUÇÃO TEÓRICA	5
RESULTADOS	6
CONCLUSÃO	9
REFERÊNCIAS	10

OBJETIVOS

A atividade tem como o propósito a criação de uma simulação de um sistema simples aproveitando a estrutura do exercício anterior com a criação de múltiplas funções para a resolução do problema proposto. Além da criação manual de um arquivo makefile que direciona a criação do executável da aplicação.

FUNDAMENTAÇÃO TEÓRICA

“Define-se sistema como a representação de qualquer fenômeno natural, físico ou químico, sujeito a estímulos externos (ditos sinais de entrada ou variáveis de controle) e produzindo respostas a tais estímulos. De modo geral, um sistema é dotado da capacidade de gerar uma saída específica a uma entrada conhecida, de maneira que a integração de suas partes conduz a um resultado combinado dos efeitos individuais de cada porção.” [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 θ a sua orientação $u(t) = [v \ \omega]^T$ é a entrada do sistema, sendo v a velocidade linear e ω 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

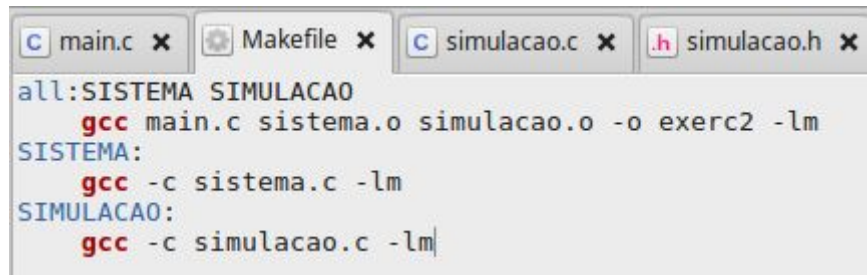
O ambiente possui 5 arquivos fontes: `main.c`, `sistema.c`, `sistema.h`, `simulacao.c`, `simulacao.h`. A `main.c` é o arquivo principal do projeto e nele estão incluídas as bibliotecas `simulacao.h` e `sistema.h`. O `sistema.c` é o arquivo onde estão descritas as funções relacionadas aos cálculos da solução problema em si, o `sistema.h` é onde estão as assinaturas das mesmas. O `simulacao.c` trata-se das funções relacionadas a geração do arquivo que `.txt` servirá de base para a plotagem do gráfico dos resultados e seu respectivo `.h` contém as assinaturas dessas funções. Além disso, o `Makefile` gera os arquivos de compilação `.o` de cada um dos arquivos fontes.



Figura 3: Todos os arquivos dentro do diretório da aplicação.

ARQUIVOS FONTES

No arquivo `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 do diretório do projeto. O nome do executável é “`exerc2`” e pode ser aberto como mostra nas figuras que seguem.



```
all:SISTEMA SIMULACAO
    gcc main.c sistema.o simulacao.o -o exerc2 -lm
SISTEMA:
    gcc -c sistema.c -lm
SIMULACAO:
    gcc -c simulacao.c -lm
```

Figura 4: Arquivo Makefile criado.

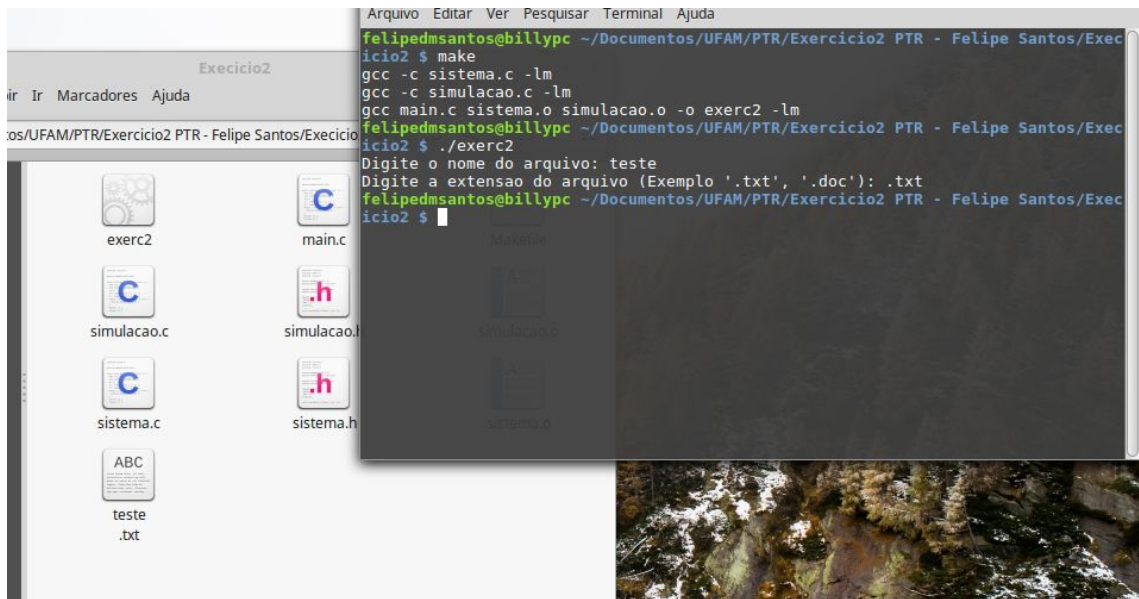
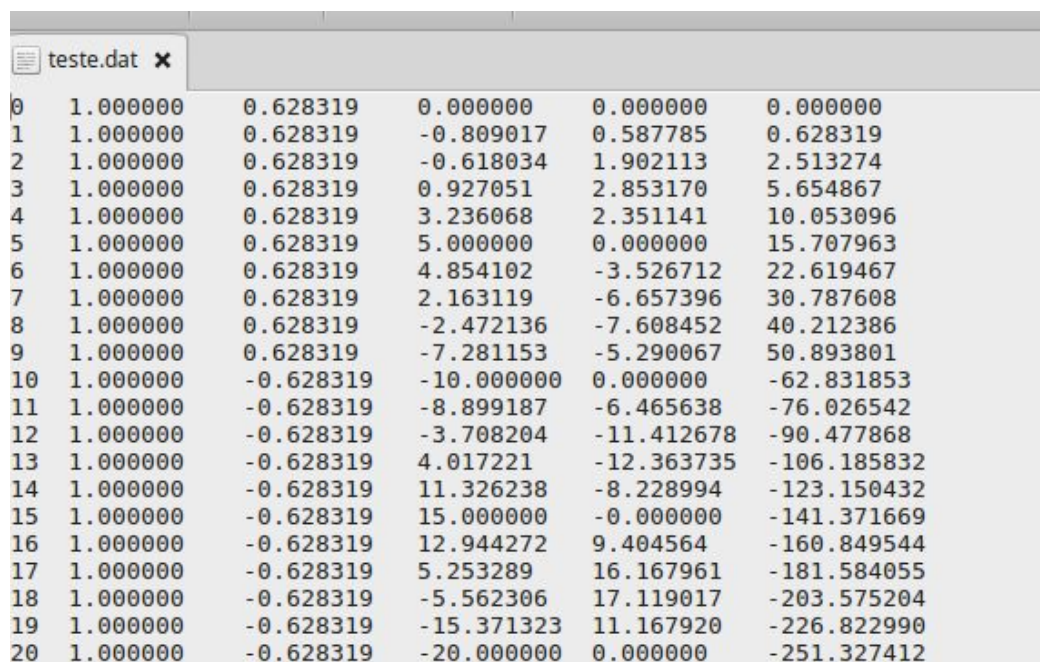


Figura 5: Como deve ser executado o projeto.



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

Figura 6: Exemplo de arquivo gerado (obs: O usuário escolhe o formato que desejar)

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.

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:

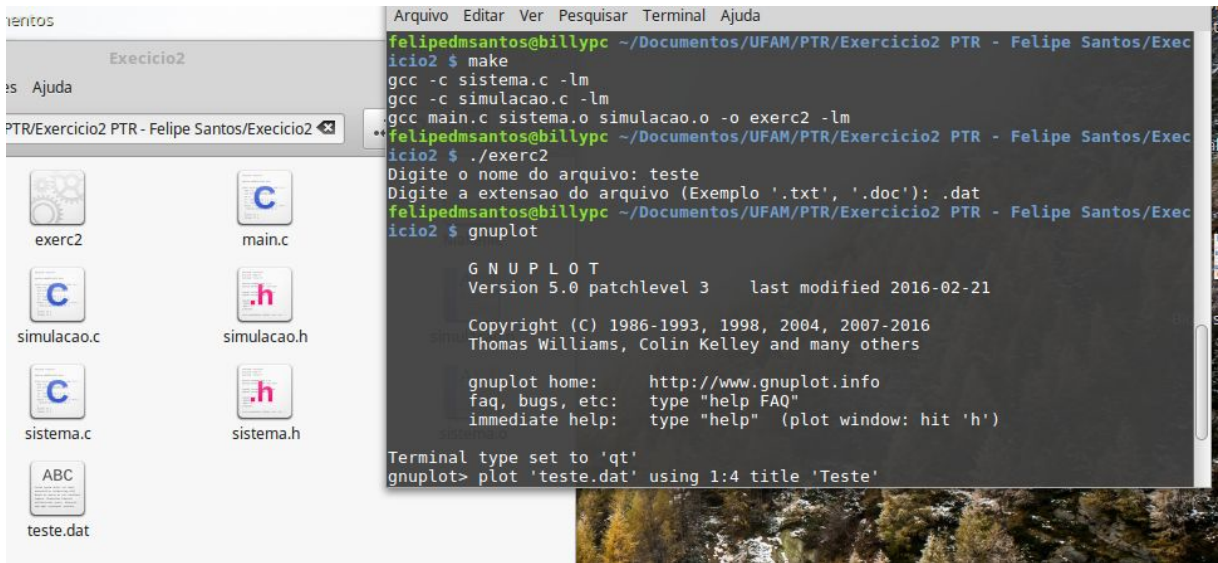


Figura 7: Software Gnuplot iniciando.

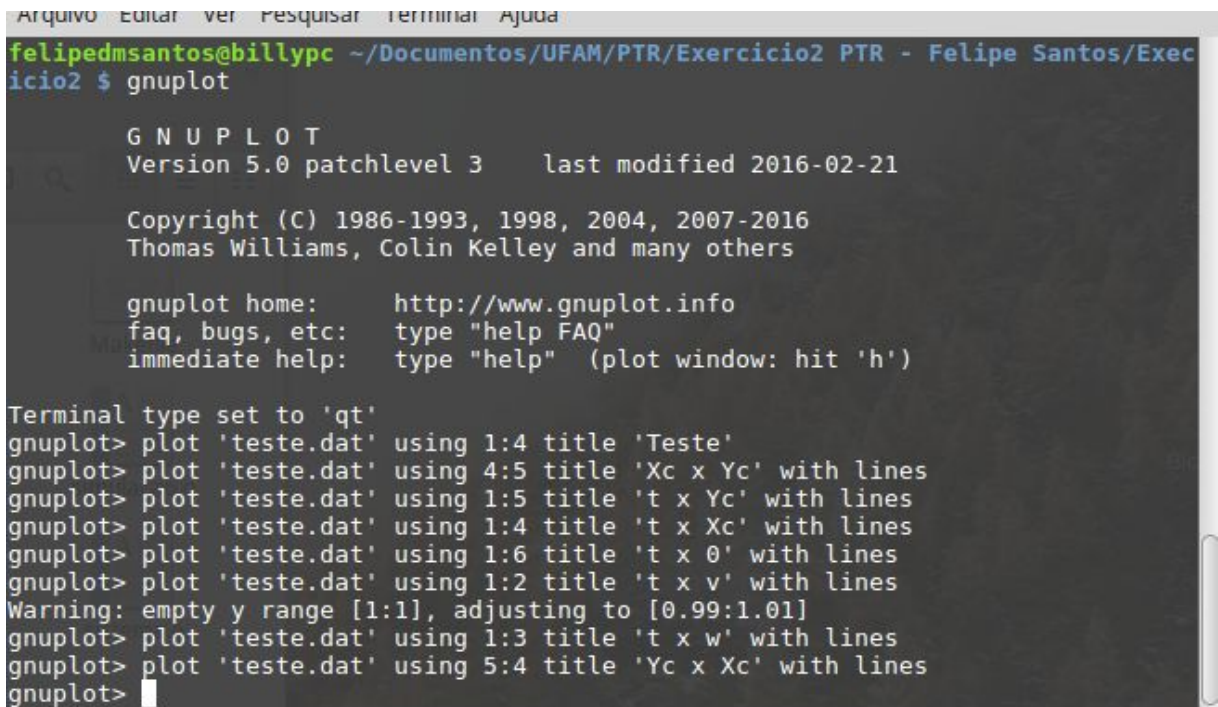


Figura 8: Comandos executados para gerar os gráficos solicitados.

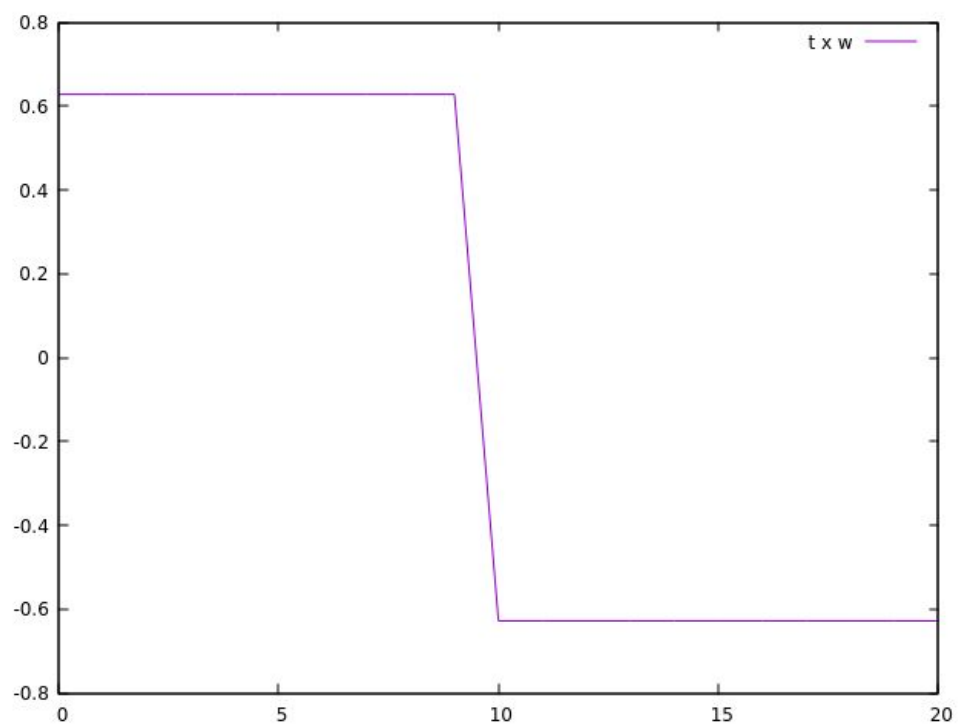


Figura 9: Gráfico $u(t)$ - t x ω

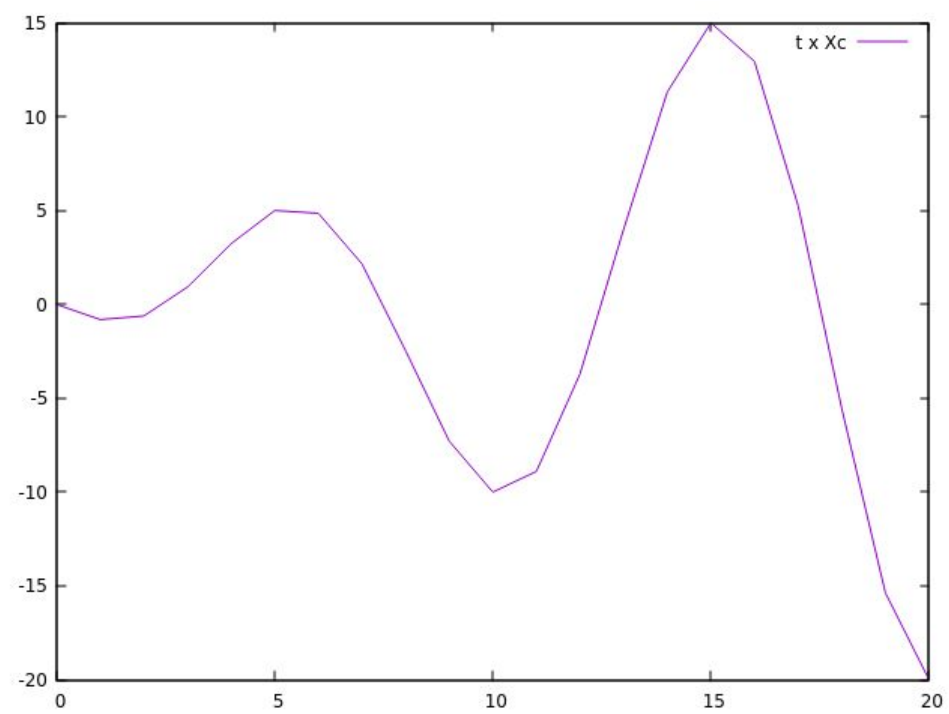


Figura 10: Gráfico $y(t)$ - t x X_c

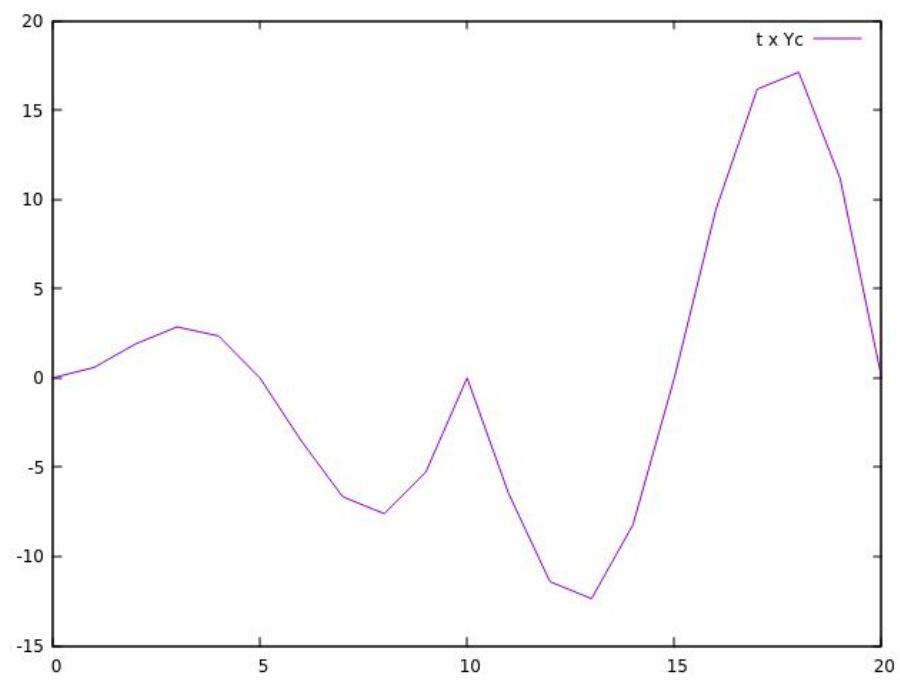


Figura 11: Gráfico $y(t)$ - $t \times Y_c$

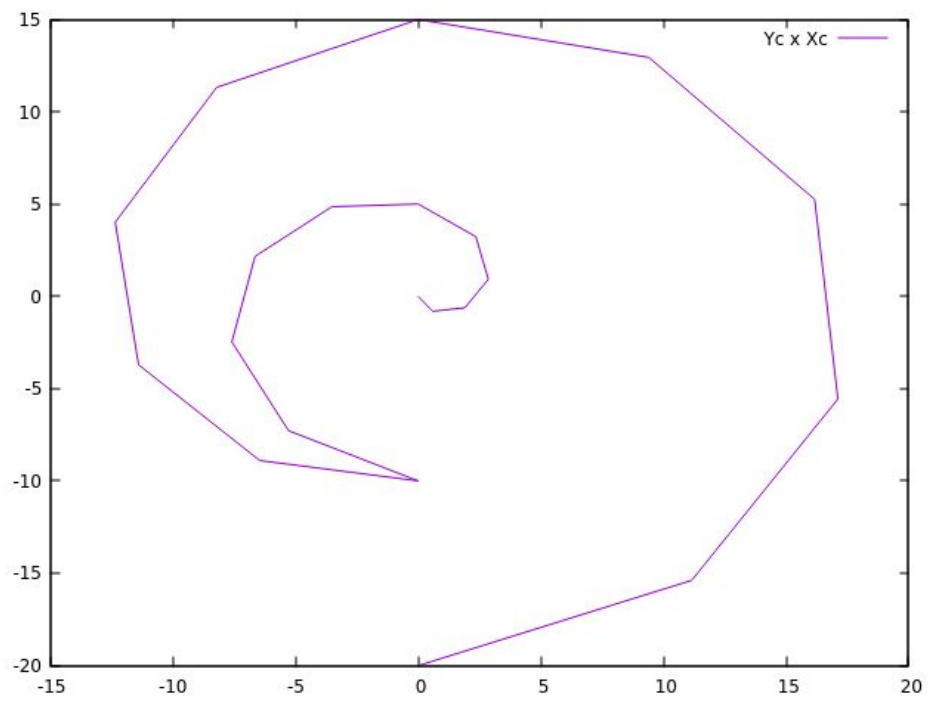


Figura 12: Gráfico $Y_c(t) \times X_c(t)$

CONCLUSÃO

Na engenharia podemos nos deparar com modelos matemático de vários níveis de complexidade, e a computação nos ajuda a fazer uso de modelos cada vez mais completos e eficientes na medida que os softwares de simulação vão se tornando cada vez mais complexos.

No problema do exercício proposto nos deparamos com o modelo cinemático de movimentação de um robô e apenas nos utilizando de ferramentas da linguagem de programação C foi possível realizar a simulação plotando gráficos e chegando a resultados satisfatórios quanto a solução da problemática proposta.

REFERÊNCIAS

- [1] PENEDO, S. R. M. Sistemas de controle: Matemática Aplicada a Projetos. 1. ed. São Paulo: Érica, 2014.

- [2] IDAGAWA, H. S. A Importância do Engenheiro no Desenvolvimento de Simulações Computacionais. Revista Intellectus. São Paulo, Ano IX, nº 23.