Solução dos Exercícios de Programação e MATLAB Capítulo 2

Alisson Jaques*. Matheus Nogueira*. Sillas Francisco*.

**Universidade do Estado de Minas Gerais*

Abstract: In this article you will find resolutions of the programming and MATLAB exercises proposed in chapter 2 of the book Robotics by author John J. Craig.

Resumo: Neste artigo encontra-se as resoluções dos exercícios de programação e MATLAB propostos no capítulo 2 do livro Robótica do autor John J. Craig.

Palavras-chaves: MATLAB; vetores; matriz rotacional; ângulos Z-Y-Z; algoritmos.

1. INTRODUÇÃO

Este artigo fornece explicações sobre os algoritmos e subrotinas, criados em MATLAB, que consistem em soluções para os exercícios do capítulo 2 do livro Robótica do autor John J. Craig (os algoritmos estão anexados numa pasta juntos com este artigo, aconselhamos a leitura em conjunto dos mesmos com este texto, para melhor compreensão).

2. EXERCÍCIOS DE PROGRAMAÇÃO PARTE 2

2.1 SOLUÇÃO

Como as sub-rotinas foram escritas em MATLAB e a função atan2 faz parte de sua biblioteca padrão, não existe a necessidade de criarmos uma função para o cálculo de atan2.

2.2 SOLUÇÃO

Criamos uma sub-rotina em MATLAB que consiste em uma função que recebe um vetor preenchido pelo usuário (vetor = [x, y, θ] e retorna uma matriz correspondente ao sistema de referência definido pelo vetor. Dentro do escopo da função são declaradas variáveis auxiliares que farão conversões em de medidas de ângulos e guardarão o valor da coordenada x e y fornecidos pelo usuário. Em seguida é retornada a matriz correspondente ao sistema de referência do vetor.

2.3 SOLUÇÃO

Analisando o procedure vemos que precisamos criar uma função que multiplique juntas duas matrizes que são definidas por brela (sistemas de coordenadas {B} rotacionado em {A}) e *crelb* (sistemas de coordenadas {C} rotacionado em {B}). Para tanto, foi desenvolvido em MATLAB uma sub-rotina que recebe duas matrizes e retorna uma matriz que é o produto das mesmas. Caso o usuário digite uma matriz inválida para o produto, no prompt de comando do MATLAB surgirá um erro explicando porque não foi possível realizar a multiplicação.

2.4 SOLUÇÃO

Como descrito no procedure, temos como entrada um sistema de coordenadas {B} rotacionado em {A} e retornamos em nossa sub-rotina o sistema de coordenadas {A} rotacionado em {B} (processo inverso). No escopo da função criada temos a declaração de uma variável auxiliar que recebe a transposta da matriz passada como argumento (brela). Em seguida criamos dois for aninhados, que terão como tarefa realizar o processo inverso. Finalmente, associamos a variável auxiliar à variável de retorno de nossa função.

3. EXERCÍCIOS PARA MATLAB 2A

3.1 a) SOLUÇÃO

O algoritmo criado consiste em uma função, que recebe um vetor de ângulos Z-Y-X (α-β -γ) de Euler, como argumento, e retorna a matriz rotacional do sistema de referência. No escopo da função é declarada uma variável auxiliar para conversão do ângulo em radianos bem como variáveis auxiliares que recebem o conteúdo de funções trigonométricas, essenciais para a solução do problema. Por fim é retornada uma matriz de rotação cujos termos são definidos pelas variáveis auxiliares criadas anteriormente.

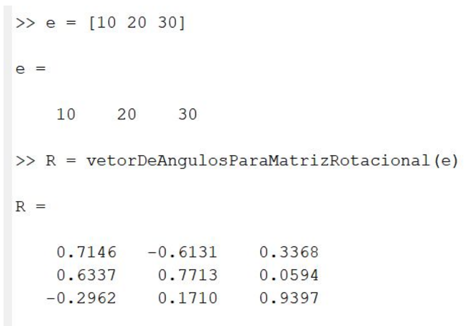
3.1.1 i) $\alpha = 10°$, $\beta = 20°$ e $\gamma = 30°$ - SOLUÇÃO

Uma vez que o algoritmo desenvolvido seja executado, inserimos no terminal do MATLAB os seguintes comandos:

\>> e = [10 20 30];

\>> R = vetorDeAngulosParaMatrizRotacional(e)

Obtivemos como resposta a matriz rotacional descrita na FIGURA 1.



```
>> e = [10 20 30]

e =

    10    20    30

>> R = vetorDeAngulosParaMatrizRotacional(e)

R =

    0.7146   -0.6131    0.3368
    0.6337    0.7713    0.0594
   -0.2962    0.1710    0.9397
```

Figura 1: Matriz rotacional R, obtida pela execução do comando.

Para as seis restrições para matrizes rotacionais ortonormais unitárias foram executados os seguintes comandos no terminal:

>> X = R(:,1); Y = R(:,2); Z = R(:,3);   (1)

>> norm(X), norm(Y), norm(Z)

Em (1) X, Y e Z são definidos como colunas de R

A FIGURA 2 mostra o resultado obtido na execução dos comandos acima:

```
>>  X = R(:,1); Y = R(:,2); Z = R(:,3);
>> norm(X), norm(Y), norm(Z)

ans =

    1.0000


ans =

     1


ans =

     1
```

Figura 2: Magnitude das colunas

Cada coluna possui magnitude 1 o que implica 3 restrições. Calculamos os três produtos escalares com o seguinte comando no terminal:

>> dot(X,Y), dot(X,Z), dot(Y,Z)

O resultado é mostrado na FIGURA 3, onde verificamos que os três produtos escalares é 0, o que implica em mais 3 restrições.

```
>> dot(X,Y), dot(X,Z), dot(Y,Z)

ans =

  -2.7756e-17


ans =

     0


ans =

     0
```

FIGURA 3: Resultado dos produtos escalares

Sendo assim, dados 9 elementos em R e 6 restrições, existem apenas 3 quantidades independentes resultantes.

3.1.2 ii) $\alpha = 30°$, $\beta = 90°$ e $\gamma = -55°$ - SOLUÇÃO

Uma vez que o programa criado tenha sido executado, inserimos no terminal do MATLAB os seguintes comandos:

>> e = [30 90 -55];

>> R = vetorDeAngulosParaMatrizRotacional(e)

Obtivemos como resposta a matriz rotacional descrita na FIGURA 4.

```
>> e = [30 90 -55]

e =

    30    90   -55

>> R = vetorDeAngulosParaMatrizRotacional(e)

R =

    0.4096   -0.2868    0.8660
   -0.7094    0.4967    0.5000
   -0.5736   -0.8192    0.0000
```

FIGURA 4: Matriz rotacional obtida pela execução do comando

3.2 b) SOLUÇÃO

O programa desenvolvido consiste em uma função que recebe uma matriz rotacional e retorna dois vetores de ângulos Z-Y-Z de Euler. Sendo um processo inverso ao da questão que foi solucionada no tópico *3.2 a)* . No escopo da função são declaradas várias variáveis auxiliares, estas recebem expressões que manipulam propriedades trigonométricas, incluindo o arco-tangente de dois argumentos. O intuito desse cálculo é de se obter os ângulos Z-Y-Z de Euler que, uma vez obtidos, são atribuídos nos índices de dois vetores, sendo estes retornados pela função.

3.2.2 DEMONSTRANDO A SOLUÇÃO INVERSA PARA i

Primeiramente vamos entrar com os ângulos de Euler no código criado no tópico *3.1 a)*, fazemos isso informando o conteúdo de *e* no terminal:

>> e = [10 20 30]

Depois, passamos *e* como argumento para a função vetorDeAngulosParaMatrizRotacional e atribuímos o seu conteúdo (uma matriz rotacional) à variável R:

>> R = vetorDeAngulosParaMatrizRotacional(e)

Agora atribuímos a matriz rotacional, obtida no comando acima, ao programa desenvolvido neste tópico:

>> [e1,e2] = matrizRotacionalParaVetorDeAngulos(R)

Vamos obter dois vetores com ângulos de Euler, conforme FIGURA 5.

```
>> e = [10 20 30]

e =

    10    20    30

>> R = vetorDeAngulosParaMatrizRotacional(e)

R =

    0.7146   -0.6131    0.3368
    0.6337    0.7713    0.0594
   -0.2962    0.1710    0.9397

>> [e1,e2] = matrizRotacionalParaVetorDeAngulos(R)

e1 =

   10.0000   20.0000   30.0000


e2 =

  -170   -20  -150
```

FIGURA 5: Execução dos comandos e obtenção dos vetores com ângulos de Euler

Na FIGURA 5 temos que *e1* é igual ao vetor *e* de entrada para o primeiro programa, sendo a solução que já tínhamos conhecimento. Já *e2* é nossa segunda solução. Para verificarmos se *e2* é uma solução válida, basta colocar o mesmo como argumento para o algoritmo desenvolvido no tópico *3.1 a)*. A FIGURA 6 mostra que a matriz rotacional é a mesma o que implica que ambos os vetores de ângulos de Euler são equivalentes.

```
>> R = vetorDeAngulosParaMatrizRotacional(e2)

R =

    0.7146   -0.6131    0.3368
    0.6337    0.7713    0.0594
   -0.2962    0.1710    0.9397
```

FIGURA 6: Matriz rotacional obtida com o vetor de ângulos e2

3.2.2 DEMONSTRANDO A SOLUÇÃO INVERSA PARA ii

Primeiramente vamos entrar com os ângulos de Euler no código criado no tópico *3.1 a)*, fazemos isso informando o conteúdo de *e* no terminal:

>> e = [30 90 -55]

Depois, passamos *e* como argumento para a função vetorDeAngulosParaMatrizRotacional e atribuímos o seu conteúdo (uma matriz rotacional) à variável R:

>> R = vetorDeAngulosParaMatrizRotacional(e)

Agora atribuímos a matriz rotacional, obtida no comando acima, ao programa desenvolvido neste tópico:

>> [e1,e2] = matrizRotacionalParaVetorDeAngulos(R)

Vamos obter dois vetores com ângulos de Euler, conforme FIGURA 7.

```
>> e = [30 90 -55]

e =

    30    90   -55

>> R = vetorDeAngulosParaMatrizRotacional(e)

R =

    0.4096   -0.2868    0.8660
   -0.7094    0.4967    0.5000
   -0.5736   -0.8192    0.0000

>> [e1,e2] = matrizRotacionalParaVetorDeAngulos(R)

e1 =

   30.0000   90.0000  -55.0000


e2 =

 -150.0000  -90.0000  125.0000
```

FIGURA 7: Execução dos comandos e obtenção dos vetores com ângulos de Euler

Na FIGURA 7 temos que *e1* é igual ao vetor *e* de entrada para o primeiro programa, sendo a solução que já tínhamos conhecimento. Já *e2* é nossa segunda solução. Para verificarmos se *e2* é uma solução válida basta colocar o mesmo como argumento para o algoritmo desenvolvido no tópico *3.1 a)*. A FIGURA 8 mostra que a matriz rotacional é a mesma o que implica que ambos os vetores de ângulos de Euler são equivalentes.

```
>> R = vetorDeAngulosParaMatrizRotacional(e2)

R =

    0.4096   -0.2868    0.8660
   -0.7094    0.4967    0.5000
   -0.5736   -0.8192    0.0000
```

FIGURA 8: Matriz rotacional obtida com o vetor de ângulos e2

3.3 c) SOLUÇÃO

Iniciamos com o sistema de coordenadas {B} e o giramos 20° em torno do eixo *y*, assim definiremos no terminal a seguinte matriz rotacional:

>> R = [0.9397 0 0.3420;0 1 0;-0.3420 0 0.9397]

Agora inserimos no terminal o vetor bP:

>> bP = [1 0 1]’

Em seguida obtemos o vetor aP, que é a solução da questão:

>> aP = R*bP

A FIGURA 9 mostra o resultado dos comandos acima e o valor de aP encontrado.

```
Command Window

>> R = [0.9397 0 0.3420;0 1 0;-0.3420 0 0.9397]

R =

    0.9397         0    0.3420
         0    1.0000         0
   -0.3420         0    0.9397

>> bP = [1 0 1]'

bP =

     1
     0
     1

>> aP = R*bP

aP =

    1.2817
         0
    0.5977
```

FIGURA 9: Obtenção do vetor P no sistemas de coordenadas {A}

Na FIGURA 10 temos um desenho que demonstra a veracidade dos dados obtidos acima:

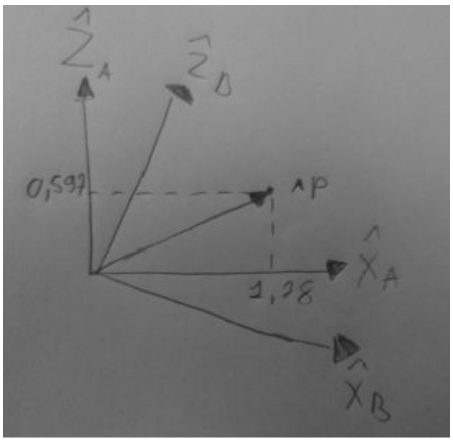


FIGURA 10: Representação gráfica da obtenção do vetor aP

5. CONCLUSÃO

Neste trabalho fica evidente a grande utilidade do ambiente de programação MATLAB, pois todos os algoritmos e sub-rotinas foram desenvolvidos (de maneira simples) com essa ferramenta. Podemos também compreender melhor a relação entre a álgebra, no que diz respeito à cinemática dos robôs, e os programas de computadores que, juntos, controlam todo o tipo de movimentação do robô no espaço tridimensional. Foi apresentado aqui as respostas dos exercícios de programação e MATLAB do capítulo 2 do livro Robótica do Craig, o que nos forneceu grande embasamento sobre os sistemas de coordenadas, e suas transformações, para a manipulação de robôs industriais.

AGRADECIMENTOS

Agradecemos ao professor Ângelo pelas aulas ministradas, que foram fundamentais para a compreensão do conteúdo, bem como sua compreensão (no que diz respeito ao tempo que levaria para terminarmos essa atividade).

REFERÊNCIAS

Craig, John J. "Robótica. 3ª edição." (2012).

MathWorks. Disponível em: <www.mathworks.com>. Acesso em: 22 nov. 2020.

MATLAB Avançado. Disponível em: <http://mtm.ufsc.br/~melissa/arquivos/matlabpet/aula_01.pdf>. Acesso em: 22 nov. 2020.

Apostila MATLAB para engenharia Disponível em: <http://ftp.demec.ufpr.br/disciplinas/TMEC078/curso%20Matlab/referencia/6_MatLab_para_Engenharia.pdf> Acesso em: 22 nov. 2020.

MATLAB_12-Funções. Disponível em: <https://edisciplinas.usp.br/pluginfile.php/95376/mod_resource/content/1/MATLAB_12-Funcoes.pdf> Acesso em: 22 nov. 2020.

atan2(funções MATLAB). Disponível em: <http://www.ece.northwestern.edu/local-apps/matlabhelp/techdoc/ref/atan2.html> Acesso em: 23 nov. 2020.