FUNDAÇÃO UNIVERSIDADE FEDERAL DO ABC

Determinação de Atitude Estática

DAESTS005-17SB: Dinâmica e Controle de Veículos Espaciais
Prof.º Dr. Luiz de Siqueira Martins Filho

Allan Moreira de Carvalho
Rodrigo Vidal Cabral

9 de agosto de 2018

PROBLEMA

O controle de atitude de veículos espaciais depende primordialmente da determinação da atitude. A determinação da atitude pode ser entendida como a determinação da posição do sistema de referências fixa ao corpo do veículo espacial em relação à um sistema de referência externo. Dessa forma o problema da determinação de atitude resume-se à determinação de um tensor R que relaciona dois sistemas de coordenadas, tal tensor, conhecido como matriz de rotação pode ser obtido por diversos métodos que utilizam dois ou mais vetores medidos em diferentes referenciais. No presente trabalho, serão comparados o desempenho dos métodos TRIAD e método q.

Dados os vetores observados por quatro sensores distintos e referenciados no sitema $\{\hat{b}\}$ fixo ao corpo de um hipotético veículo espacial

$$\vec{v}_{1b} = \begin{bmatrix} 0.8273 \\ 0.5541 \\ -0.0920 \end{bmatrix} \quad \vec{v}_{2b} = \begin{bmatrix} -0.8285 \\ 0.5522 \\ -0.0955 \end{bmatrix} \quad \vec{v}_{3b} = \begin{bmatrix} 0.2155 \\ 0.5522 \\ 0.8022 \end{bmatrix} \quad \vec{v}_{4b} = \begin{bmatrix} 0.5570 \\ -0.7442 \\ -0.2882 \end{bmatrix} \tag{0.1}$$

e os vetores $\vec{v}_{*i}$ resultantes dos modelos matemáticos e referenciados no sitema $\{\hat{\imath}\}$ inercial

$$\vec{v}_{1i} = \begin{bmatrix} -0.1517 \\ -0.9669 \\ 0.2050 \end{bmatrix} \quad \vec{v}_{2i} = \begin{bmatrix} -0.8393 \\ 0.4494 \\ -0.3044 \end{bmatrix} \quad \vec{v}_{3i} = \begin{bmatrix} -0.0886 \\ -0.5856 \\ -0.8000 \end{bmatrix} \quad \vec{v}_{4i} = \begin{bmatrix} 0.8814 \\ -0.0303 \\ 0.5202 \end{bmatrix} \tag{0.2}$$

a matriz de rotação R pode ser determinada pelos métodos TRIAD e q utilizando a ferramenta MATLAB.

1 Algorítmo TRIAD

O algorítmo TRIAD destaca-se pela sua simplicidade, para a determinação determinística da atitude estática são considerados apenas dois vetores diretores. Parte da informação contida nesses dois vetores é discartada pelo método, dessa maneira a sobredeterminação do método é eliminada de maneira simples. Além disso, ao descartar parte da informação, um dos sensores é escolhido como o melhor dentre àqueles disponíveis. Assumindo-se portanto que esse possua maior acurácia, menos ruído ou qualquer outra melhor característica de mérito.

O método consiste na construção de duas triades, dái o nome, que representam um novo referencial $\{\hat{t}\}$ em termos dos referenciais $\{\hat{b}\}$ e $\{\hat{\imath}\}$. A construção das duas tríades utiliza um par de vetores medidos em diferentes referenciais.

1.1 Utilizando sensores 1 e 2

Assume-se que o vetor $\vec{v}_1$ seja proveniente do melhor sensor disponível. Faz-se o primeiro versor da nova base $\{\hat{t}\}$ igual ao versor $\vec{v}_1$

$$\begin{aligned} \hat{t}_{1b} &= \hat{v}_{1b} \\ \hat{t}_{1i} &= \hat{v}_{1i} \end{aligned} \tag{1.1}$$

O segundo versor da base $\{\hat{t}\}$ é definido como perpedicular aos versores $\vec{v}_1$ e $\vec{v}_2$

$$\begin{aligned} \hat{t}_{2b} &= \frac{\hat{v}_{1b} \times \hat{v}_{2b}}{|\hat{v}_{1b} \times \hat{v}_{2b}|} \\ \hat{t}_{2i} &= \frac{\hat{v}_{1i} \times \hat{v}_{2i}}{|\hat{v}_{1i} \times \hat{v}_{2i}|} \end{aligned} \tag{1.2}$$

A base $\{\hat{t}\}$ é ortonormal e portando o terceiro versor é ortogonal aos anteriores

$$\begin{aligned} \hat{t}_{3b} &= \frac{\hat{t}_{1b} \times \hat{t}_{2i}}{\left|\hat{t}_{1b} \times \hat{t}_{2i}\right|} \\ \hat{t}_{3i} &= \frac{\hat{t}_{1i} \times \hat{t}_{2i}}{\left|\hat{t}_{1i} \times \hat{t}_{2i}\right|} \end{aligned} \tag{1.3}$$

Duas matrizes de rotação são montadas com os vetores obtidos

$$\begin{aligned} R^{bt} &= \begin{bmatrix} \hat{t}_{1b} & \hat{t}_{2b} & \hat{t}_{3b} \end{bmatrix} \\ R^{it} &= \begin{bmatrix} \hat{t}_{1i} & \hat{t}_{2i} & \hat{t}_{3i} \end{bmatrix} \end{aligned} \tag{1.4}$$

Finalmente a matriz de rotação que leva de referencial inercial para o referencial do corpo é simplesmente

$$
\begin{aligned}
R^{bi} &= R^{bt} R^{ti} = R^{bt} {R^{it}}^T \\
R^{bi} &= \begin{bmatrix} \hat{t}_{1b} & \hat{t}_{2b} & \hat{t}_{3b} \end{bmatrix} \begin{bmatrix} \hat{t}_{1i} \\ \hat{t}_{2i} \\ \hat{t}_{3i} \end{bmatrix}
\end{aligned} \tag{1.5}
$$

Utilizando os vetores $\vec{v}_{1b}$, $\vec{v}_{11}$, $\vec{v}_{2b}$ e $\vec{v}_{2i}$ dados temos que

$$
R_{12}^{bi} = \begin{bmatrix} 0.4156 & -0.8551 & 0.3100 \\ -0.8339 & -0.4943 & -0.2455 \\ 0.3631 & -0.1566 & -0.9185 \end{bmatrix} \tag{1.6}
$$

Cujo quatérnio associado pode ser calculado como

$$
\overline{q} = \begin{bmatrix} \vec{q} \\ q_4 \end{bmatrix} \tag{1.7}
$$

tal que

$$
\vec{q} = \frac{1}{4q_4} \begin{bmatrix} R_{23} - R_{32} \\ R_{31} - R_{13} \\ R_{12} - R_{21} \end{bmatrix} \qquad q_4 = \pm \frac{1}{2} \sqrt{1 + tr\,(R)} \tag{1.8}
$$

assim sendo

$$
\overline{q}_{12} = \begin{bmatrix} -0.8409 \\ 0.5022 \\ -0.2001 \\ \pm 0.0264 \end{bmatrix} \tag{1.9}
$$

a matriz de rotação em termos de quatérnios é

$$
R = \left(q_4^2 - \vec{q}^{\,T} \vec{q}\right) I + 2\vec{q}\vec{q}^{\,T} - 2q_4 \left[\vec{q}^{\,\times}\right] \tag{1.10}
$$

Logo,

$$
R_{12_q}^{bi} = \begin{bmatrix} 0.4156 & -0.8551 & 0.3100 \\ -0.8339 & -0.4943 & -0.2455 \\ 0.3631 & -0.1566 & -0.9185 \end{bmatrix} \tag{1.11}
$$

que é a mesma matriz de rotação encontrada anteriormente como esperado. A partir da função custo

$$
J = \frac{1}{2} \sum_{k=1}^{N} w_k \left(v_{kb} - R_{bi} \cdot v_{ki}\right)^2 \tag{1.12}
$$

é possível obter uma medida da performance do método e da acurácia da matriz de rotação obtida. Utilizando-se os pares de vetores utilizados e impondo pesos w_k iguais para ambos, temos

$$J_{12} = 1.8298 \times 10^{-7} \tag{1.13}$$

1.2 Utilizando os sensores 1 e 3

De maneira análoga ao procedimento adotado anteriormente, o método TRIAD pode ser aplicado ao par de vetores (v_{1*}, v_{3*}), como resultados temos a matriz de rotação

$$R_{13}^{bi} = \begin{bmatrix} 0.4167 & -0.8563 & 0.3051 \\ -0.8370 & -0.4924 & -0.2387 \\ 0.3546 & -0.1559 & -0.9219 \end{bmatrix} \tag{1.14}$$

, o quatérnio

$$\overline{q}_{13} = \begin{bmatrix} -0.8413 \\ 0.5032 \\ -0.1961 \\ \pm 0.0246 \end{bmatrix} \tag{1.15}$$

, a matriz de rotação calculada pelos termos do quatérnio

$$R_{13_q}^{bi} = \begin{bmatrix} 0.4167 & -0.8563 & 0.3051 \\ -0.8370 & -0.4924 & -0.2387 \\ 0.3546 & -0.1559 & -0.9219 \end{bmatrix} \tag{1.16}$$

Finalmente o custo

$$J_{13} = 1.1146e \times 10^{-5} \tag{1.17}$$

1.3 Utilizando os sensores 1 e 4

Para os sensores 1 e 4, obteve-se a matriz de rotação

$$R_{14}^{bi} = \begin{bmatrix} 0.4279 & -0.8736 & 0.2317 \\ -0.8750 & -0.4646 & -0.1359 \\ 0.2263 & -0.1446 & -0.9633 \end{bmatrix} \tag{1.18}$$

, o quatérnio

$$\overline{q}_{14} = \begin{bmatrix} 0.8449 - 0.51740.1355 \pm 0.0026 \end{bmatrix} \tag{1.19}$$

, a matriz de rotação calculada pelos termos do quatérnio

$$R_{14_q}^{bi} = \begin{bmatrix} 0.4279 & -0.8736 & 0.2317 \\ -0.8750 & -0.4646 & -0.1359 \\ 0.2263 & -0.1446 & -0.9633 \end{bmatrix} \tag{1.20}$$

Com custo

$$J_{14} = 0.0014 \tag{1.21}$$

1.4 Resultado do algorítmo TRIAD

Como resultado constata-se que o par de sensores 1 e 2 apresenta o melhor resultado, indicando que possivelmente esses sejam os sensores mais precisos e acurados.

Finalmente o custo

$$J_{12} = 1.8298 \times 10^{-7} \tag{1.22}$$

2 Método q

O método q é ideal para sistemas sobredeterminados pois utiliza toda a informação disponível. É possível mostrar que minimizar a função custo (1.12) é analogo à maximizar a função

$$g(R) = \sum_{k=1}^{N} w_k \left(v_{kb}^T R^{bi} v_{ki} \right) \tag{2.1}$$

Adotando os quatérnios como parâmetros, pode-se reescrever $g(R)$ como

$$g(\overline{q}) = \overline{q}^T K \overline{q} \tag{2.2}$$

onde a matriz K é definida como

$$K = \begin{bmatrix} S - \sigma I & Z \\ Z^T & \sigma \end{bmatrix} \tag{2.3}$$

com

$$\begin{aligned} B &= \sum_{k=1}^{N} w_k (v_{kb} v k i^T) \\ S &= B + B^T \\ \sigma &= tr(B) \\ Z &= \begin{bmatrix} B_{23} - B_{32} \\ B_{31} - B_{13} \\ B_{12} - B_{21} \end{bmatrix} \end{aligned} \tag{2.4}$$

diferenciando a equação (2.2) e usando multiplicadores de Lagrange

$$g'(\overline{q}) = \overline{q}^T K \overline{q} - \lambda \overline{q}^T \overline{q} \tag{2.5}$$

o valor de lamba que máximixa a função $g'(R)$ é o maior autovalor de

$$K\overline{q} = \lambda\overline{q} \tag{2.6}$$

e o quatérnio é o autovalor associado a esse autovalor. Assim, utilizando a equação (1.10) obtem-se a matriz de rotação.

2.1 Resultado do Método Q

O método q, quando aplicado a todos os vetores disponíveis e utilizando um peso médio distribuido igualmente, tais que

$$w_1 = 0.25 \qquad w_2 = 0.25 \qquad w_2 = 0.25 \qquad w_2 = 0.25 \tag{2.7}$$

nos fornece um quatérnio

$$\overline{q}_q = \begin{bmatrix} -0.8498 \\ 0.4975 \\ -0.1741 \\ 0.0060 \end{bmatrix} \tag{2.8}$$

e a matriz de rotação

$$R_q = \begin{bmatrix} 0.4443 & -0.8477 & 0.2899 \\ -0.8435 & -0.5048 & -0.1834 \\ 0.3018 & -0.1630 & -0.9393 \end{bmatrix} \tag{2.9}$$

para essa matriz de rotação a função custo é

$$J_q = 0.0019 \tag{2.10}$$

O resultado foi surpreendente, do ponto de vista conceitual, esperava-se que o custo seria menor, uma vez que o método utiliza toda informação disponível dos quatro sensores. Na tentativa de compreender melhor os resultados obtidos foi desenvolvido uma rotina que procura pela combinação de pesos que minimize o custo, e assim, com uma precisao de duas casas os pesos que minimizam a função custo são

$$w_1 = 0.97 \qquad w_2 = 0.01 \qquad w_2 = 0.01 \qquad w_2 = 0.01 \tag{2.11}$$

e o custo nesse caso é

$$J_q = 8.9985 \times 10^{-}5 \tag{2.12}$$

que ainda é um custo maior do que àquele do método TRIAD utilizando os sensores $1,2$ e $1,3$. Portanto o sensor 1 deve ser muito mais preciso que os demais, utilizando pesos

$$w_1 = 1 \qquad w_2 = 0 \qquad w_2 = 0 \qquad w_2 = 0 \tag{2.13}$$

obteve-se um custo

$$J_q = 5.0845 \times 10^{-32} \tag{2.14}$$

um numero tão pequeno quanto a precisão disponível computacionalmente, o que nos leva a crer que o sensor 1 é um caso ideal de sensor que não possui erro e a partir dele seria hipotéticamente possível construir uma matriz de rotação perfeita com erro tentando a zero.

3 Resultados

Saída do código em MATLAB usando o algorítmo TRIAD.

```
M todo TRIAD para determina  o de atitude
----------------------------------------------------
Metodo TRIAD para vetores 1 e 2
matriz de rota  o R_12
    0.4156   -0.8551    0.3100
   -0.8339   -0.4943   -0.2455
    0.3631   -0.1566   -0.9185

tempo de execu  o
    0.0100

quat rnio q_12
   -0.8409
    0.5022
   -0.2001
    0.0264

matriz de rota  o R_q_12
    0.4156   -0.8551    0.3100
   -0.8339   -0.4943   -0.2455
    0.3631   -0.1566   -0.9185

custo J_12 com pesos w1=0.5 e w2=0.5
   1.8298e-07

----------------------------------------------------
Metodo TRIAD para vetores 1 e 3
matriz de rota  o R_13
    0.4167   -0.8563    0.3051
   -0.8370   -0.4924   -0.2387
    0.3546   -0.1559   -0.9219

tempo de execu  o
```

```
     0

quat rnio q_13
   -0.8413
    0.5032
   -0.1961
    0.0246

matriz de rota  o R_q_13
    0.4167   -0.8563    0.3051
   -0.8370   -0.4924   -0.2387
    0.3546   -0.1559   -0.9219

custo J_13 com pesos w1=0.5 e w3=0.5
   1.1146e-05

--------------------------------------------------
Metodo TRIAD para vetores 1 e 4
matriz de rota  o R_14
    0.4279   -0.8736    0.2317
   -0.8750   -0.4646   -0.1359
    0.2263   -0.1446   -0.9633

tempo de execu  o
     0

quat rnio q_14
    0.8449
   -0.5174
    0.1355
    0.0026

matriz de rota  o R_q_14
    0.4279   -0.8736    0.2317
   -0.8750   -0.4646   -0.1359
    0.2263   -0.1446   -0.9633

custo J_14 com pesos w1=0.5 e w4=0.5
    0.0014
```

Saída do código em MATLAD utilizando o método q

```
M todo q para determina  o de atitude
--------------------------------------------------
matriz de rotacao
```

```
   0.4443   -0.8477    0.2899
  -0.8435   -0.5048   -0.1834
   0.3018   -0.1630   -0.9393

quaternio
  -0.8498
   0.4975
  -0.1741
   0.0060

tempo de execu  o
   0.0300

--------------------------------------------------
pesos w1, w2, w3 e w4
   0.2500    0.2500    0.2500    0.2500

custo
   0.0019

--------------------------------------------------
pesos w1, w2, w3 e w4 que minimizam o custo
   0.9700    0.0100    0.0100    0.0100

custo
  8.9985e-05

--------------------------------------------------
peso m ximo para o sensor 1
     1     0     0     0

custo
  5.0845e-32
```

Nota-se que, embora o algoritmo TRIAD tenha apresentado o menor valor de custo, muito abaixo da media, deve-se ao fato dos custos do método q não estarem ajustados de maneira ótima.

O tempo de execução do algoritmo TRIAD medidos em um processador i5-2520M rodando o software MATLAB sobre o sistema operacional GNU/Linux com versão de kernel 4.9.65 foi de 0.01s enquanto que nas mesmas, o método q levou 0.03s.

Assim, à menos que hajam restrições quanto ao custo ou tempo de processamento, o algoritmo do método q seria o mais adequado, por apresentar valores médios de ambas variáveis, sem descartar nenhum vetor em seu método, sendo assim mais preciso, desde que tenham sensores confiáveis e os pesos sejam ajustados para levar em conta as diferenças de precisão e acurácia dos sensores.

4 Códigos em MATLAB

4.1 TRIAD

Código principal que resolve o problema utilizando o método TRIAD

```
clc;
clear all;
close all;

% vetores no referencial do corpo
v1b = [0.8273; 0.5541; -0.0920];
v2b = [-0.8285; 0.5522; -0.0955];
v3b = [0.2155; 0.5522; 0.8022];
v4b = [0.5570; -0.7442; -0.2884];

% vetores no referencial inercial
v1i = [-0.1517; -0.9669; 0.2050];
v2i = [-0.8393; 0.4494; -0.3044];
v3i = [-0.0886; -0.5856; -0.8000];
v4i = [0.8814; -0.0303; 0.5202];

vkb = normaliza([v1b v2b v3b v4b]);
vki = normaliza([v1i v2i v3i v4i]);

% pesos
w = [0.5 0.5];

disp('M todo TRIAD para determina o de atitude')

for k=2: length(vkb)
    % roda o algoritmo triad
    disp('-----------------------------------------------------------')
    disp(['Metodo TRIAD para vetores 1 e ', num2str(k)])

    % matriz de rotacao
    ti = cputime;
    R = triad(vkb(:,1), vki(:,1), vkb(:,k), vki(:,k));
    tf = cputime - ti;

    disp(['matriz de rota  o R_1',num2str(k)])
    R = triad(vkb(:,1), vki(:,1), vkb(:,k), vki(:,k));
```

```
    disp(R)

    disp('tempo de execu  o')
    disp(tf)



    % quaternio
    q = quaternio(R);
    disp(['quat rnio q_1',num2str(k)])
    disp(q)

    % matriz de rotacao a partir do quaternio
    R_q = R_from_q(q);
    disp(['matriz de rota   o R_q_1',num2str(k)])
    disp(R_q)

    % custo
    J = custo(R, w, [vkb(:,1) vkb(:,k)], [vki(:,1) vki(:,k)]);
    disp(['custo J_1',num2str(k),' com pesos w1=', num2str(w(1)), ' e w'
        , num2str(k), '=', num2str(w(2))])
    disp(J)
end
```

4.2 Método q

Código principal que resolve o problema utilizando o método q

```
clc;
clear all;
close all;

% determinacao da matriz de atitude pelo metodo q

% vetores no referencial do corpo
v1b = [0.8273; 0.5541; -0.0920];
v2b = [-0.8285; 0.5522; -0.0955];
v3b = [0.2155; 0.5522; 0.8022];
v4b = [0.5570; -0.7442; -0.2884];

% vetores no referencial inercial
v1i = [-0.1517; -0.9669; 0.2050];
v2i = [-0.8393; 0.4494; -0.3044];
v3i = [-0.0886; -0.5856; -0.8000];
```

```
v4i = [0.8814; -0.0303; 0.5202];

vkb = normaliza([v1b v2b v3b v4b]);
vki = normaliza([v1i v2i v3i v4i]);

% pesos w1, w2, w3 e w4
w = [0.25 0.25 0.25 0.25];

% determinacao de atitude utilizando metodo q
disp('M todo q para determina  o de atitude')
disp('-----------------------------------------------------------')

% calcula a matriz de rotacao R e o quaternio pelo metodo q
ti = cputime;
[R,q] =  metodo_q(w, vkb, vki);
tf = cputime-ti;

disp('matriz de rotacao')
disp(R)

disp('quaternio')
disp(q)

disp('tempo de execu  o')
disp(tf)

disp('-----------------------------------------------------------')
% pesos
disp('pesos w1, w2, w3 e w4')
disp(w)

% calculo do custo
J = custo(R, w, vkb, vki);
disp('custo')
disp(J)

disp('-----------------------------------------------------------')
% procura pesos que minimezem o custo
w = find_w_opt(R, vkb, vki);
disp('pesos w1, w2, w3 e w4 que minimizam o custo')
disp(w')

% recalcula a matrix de roatacao
[R,q] = metodo_q(w, vkb, vki);
```

```

% calculo do custo
J = custo(R, w, vkb, vki);
disp('custo')
disp(J)

disp('------------------------------------------------------------')
% peso maximo para o sensor 1
w = [1 0 0 0 ];
disp('peso m ximo para o sensor 1')
disp(w)

% recalcula a matriz de rotacao
[R,q] = metodo_q(w, vkb, vki);

% calculo do custo
J = custo(R, w, vkb, vki);
disp('custo')
disp(J)
```

4.3 Subrotinas utilizadas nos códigos principais

Subrotina que calcula a matriz de rigidez usando o método TRIAD dado os versores

```
% determinacao da matriz de atitude R utilizando o algoritmo TRIAD
function [R_bi] = triad(s_b,s_i,m_b,m_i)
% considerando os vetores s e m

% normalizacao
s_b = s_b/norm(s_b);
s_i = s_i/norm(s_i);
m_b = m_b/norm(m_b);
m_i = m_i/norm(m_i);

% o primeiro vetor t1 considera o melhor sensor s
t1_b = s_b;
t1_i = s_i;

% o segundo vetor t2 e definido perpendicular aos vetores s e m
t2_b = cross(s_b,m_b)/norm(cross(s_b,m_b));
t2_i = cross(s_i,m_i)/norm(cross(s_i,m_i));

% o terceiro vetor e ortonormal aos anteriores (t1 e t2)
t3_b = cross(t1_b,t2_b)/norm(cross(t1_b,t2_b));
```

```
t3_i = cross(t1_i,t2_i)/norm(cross(t1_i,t2_i));

% matrizes de  r o t a o  usando os vetores t
R_bt = [t1_b t2_b t3_b];
R_it = [t1_i t2_i t3_i];

% matriz de rotacao final
R_bi = R_bt*(R_it.');



end
```

subtorina que retorna o quatérnio dado uma matriz de rotação

```
function [q] = quaternio(R)
q_4 = (1/2)*sqrt(1+trace(R));

q_vec =(1/(4*q_4))*[R(2,3) - R(3,2);
                    R(3,1) - R(1,3);
                    R(1,2) - R(2,1)];

q = [q_vec; q_4];
end
```

subrotina que normaliza uma lista de vetores

```
function [v_k] = normaliza(v_k)
    % normalizacao dos vetores
    for i=1:length(v_k(1,:))
        v_k(:,i) = v_k(:,i)/norm(v_k(:,i));
    end
end
```

subrotina que retorna a matriz de rotação dado o quatérnio

```
function [R] = R_from_q(q)
q_vec = q(1:3);
R = (q(4)^2 - q_vec.'*q_vec)*eye(3) + 2*q_vec*q_vec.' - 2*q(4)*[0 -q_vec(3) q_vec(2); q_vec(3) 0 -q_vec(1); -q_vec(2) q_vec(1) 0];
end
```

subrotina que retorna o vetor de pesos que minimize a função custo

```
function [cost_vec] = find_w_opt(R, vkb, vki)
    idx = 0;
    cost_ = 1;
    for i=0.01:0.01:0.99
```

```
        for j=0.01:0.01:0.99
            for k=0.01:0.01:0.99
                for l=0.01:0.01:0.99
                    if (i + j + k +l) == 1
                        idx = idx + 1;
                        opt(1,idx)=i;
                        opt(2,idx)=j;
                        opt(3,idx)=k;
                        opt(4,idx)=l;
                        cost = custo(R, opt(:,idx)', vkb, vki);
                        if cost < cost_
                            cost_ = cost;
                            cost_vec = opt(:,idx);
                        end
                    end
                end
            end
        end
    end
end
```

subrotina que calcula o custo

```
function [J] = custo(R_bi, w, vkb, vki)
J = 0;
for k = 1:length(w)
    J = J + w(k)*norm( vkb(:,k) - R_bi*vki(:,k) )^2;
end
J = J/2;
end
```