Cinemática inversa de robôs baseado em modelo NN

Ricardo Coke (61368), Verónica Kehel (64384)

***Abstract*—In this article we describe the fundamentals of direct and inverse kinematics related to a two-axis robot arm and the usage of neural networks to learn which internal angles correspond to every external coordinates of the extreme point of the robot arm in the working space. The neural network will be based on a feed-forward method with added backpropagation method in which the external coordinates of the extreme point will be inserted as an input and the output will be the internal angles of the robotic arm. The information obtained from the neural network will be saved and later acessed when the robot needs to reach a certain position. We will also insert ostacles in the working space and make criterias in which the robotic arm will display if it's possible to reach that position without collision.**

***Index Terms*—Robot Kinematics, Neural Networks, Backpropagation, MATLAB**

I. Introdução

O objectivo deste trabalho é definir um algoritmo que movimente um braço robótico composto por dois braços articulados num espaço de trabalho definido, evitando obstáculos ou regiões que impliquem colisão entre os braços articulados e o obstáculo através da cinemática directa e inversa. Posteriormente toda a informação de espaço obtida, relativamente às coordenadas externas, ângulos internos e a sua exequibilidade será introduzida sob a forma de input e valor desejado numa rede neural cujo objectivo será aprender de forma autónoma o espaço de trabalho ajustando os pesos das camadas de forma a obter uma resposta com um valor de erro zero (igual à saída desejada). A implementação deste projecto será feita no programa MATLAB.

A. Cinemática na Robótica

Uma ferramenta fundamental na cinemática robô são as equações das cadeias cinemáticas que formam o robô. Estas equações não lineares são usadas para mapear os parâmetros articulares para a configuração do sistema robô. As equações cinemáticas também são usadas na biomecânica do esqueleto e na animação computorizada de personagens articulados. [1]

B. Cinemática Directa

Os robôs manipuladores, frequentemente são estruturas constituída por elos rígidos, interligados por juntas contendo uma extremidade fixa . Num 'braço humano', os robôs manipuladores industriais usam, normalmente, as primeiras juntas para posicionar a estrutura formada pelas demais juntas, as quais são utilizadas para orientar o elemento terminal. As juntas usadas para o posicionamento formam a estrutura designada braço. A cinemática direta do robô possibilita obter a posição do elemento no extremo do braço (coordenadas externas) a partir das posições angulares de cada uma das juntas (coordenadas internas).

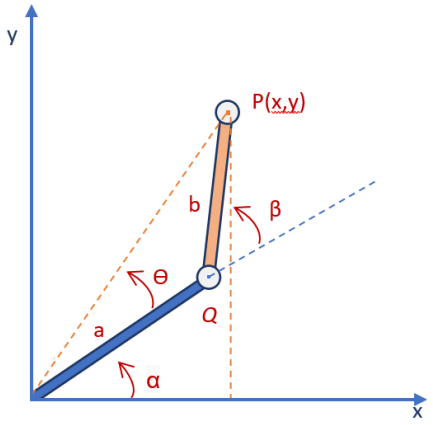

Fig. 1: Braço robótico

C. Cinemática Inversa

Na robótica a cinemática inversa descreve o processo matemático de cálculo de parâmetros articulares (ângulos internos dos braços robóticos) a partir do ponto extremo do braço robótico numa dada posição ou coordenadas (especificada pelo utilizador) em relação à origem do braço robótico. Determinamos portanto as coordenadas internas de um sistema a partir das coordenadas externas. O método como é feito este cálculo pode se basear em dois métodos: [2]

- Soluções analíticas, em que apenas estão presentes nos casos em que existem menos de 7 graus de liberdade, caso contrário, vão existir infinitas soluções para o problema da cinemática inversa e uma solução analítica não vai existir.
- Soluções numéricas, os métodos mais flexíveis dependem tipicamente da otimização iterativa para procurar uma solução aproximada, devido à dificuldade de inverter a equação cinemática directa e à possibilidade de um espaço de solução vazio.

D. Redes Neuronais

Uma Rede Neuronal Artificial (Artificial Neural Network) é baseada num agrupamento de unidades ou nós interligados chamados neurónios artificiais, que modelam, de forma simplista, os neurónios num cérebro biológico. [3] Cada ligação, como as sinapses num cérebro biológico, pode transmitir um

sinal a outros neurónios. Um neurónio artificial que recebe um sinal processa-o e pode sinalizar neurónios ligados a ele. Nas implementações da ANN, o ”sinal” numa ligação é o input dado pelo utilizador, e a saída de cada neurónio é calculada por alguma função não linear da soma das suas entradas. As ligações são chamadas Biases. Os neurónios e as Biases normalmente têm um peso que se ajusta à medida que a aprendizagem prossegue. Tipicamente, os neurónios são agregados em camadas em que diferentes camadas podem realizar diferentes transformações nas suas entradas. Os sinais viajam desde a primeira camada (a camada de entrada), até à última camada (a camada de saída) e este método é denominado de FeedForward. O peso aumenta ou diminui a força do sinal numa ligação de modo a que a resposta final seja ajustada de modo a corresponder ao output desejado, no entanto, este método é aplicado através da chamada retro propagação (Back Propagation) em que o sinal de entrada viaja até à camada final (Feed Forward), compara o resultado final obtido com a resposta esperada e caso não correspondam, de acordo com algum critério ou valor de erro, propaga de volta (da saída para a entrada) o sinal de saída de modo a ajustar os pesos das ligações entre camadas interiores, de modo a que na próxima iteração a resposta de saída seja mais semelhante à resposta esperada.

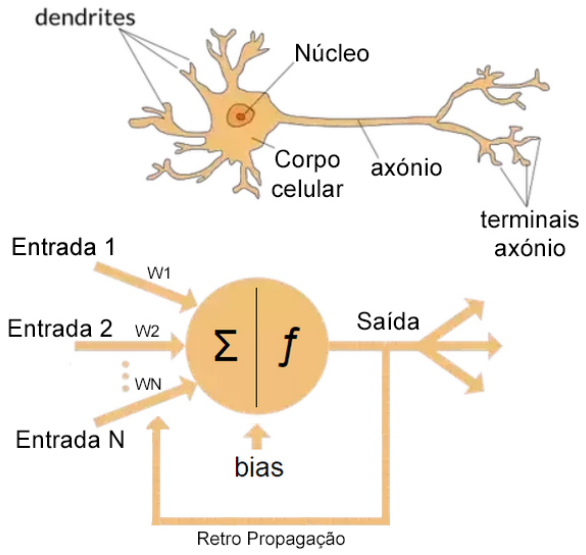

Fig. 2: Esquema de neurónio real e computarizado

II. Desenvolvimento

A. Algoritmo MATLAB

Agora que temos todas as funções necessárias para os cálculos que precisamos de fazer vamos avançar com a implementação em conjunto de todas estas ferramentas com o objectivo de indicar ao robô a aprendizagem do espaço todo de operação para posterior aplicação desse conhecimento. De modo a organizar o raciocínio seguido na elaboração do algoritmo vamos dividir a tarefa em questão em pequenos passos:

- Começando pela observação do espaço é necessário saber quantos pontos (coordenadas externas) vão haver no total e qual o espaçamento entre eles ao longo do espaço. Deste modo decidimos criar um espaço simétrico entre x e y, com os valores de -5 a 5 em ambos os paramêtros com um espaçamento unitário entre eles, na direção horizontal e vertical.
- Com a dimensão do espaço devemos agora calcular qual deve ser o tamanho de cada braço do robô de modo a ele conseguir chegar a todos os pontos do espaço, tal vai ser calculado através da distância Euclidiana aos ponto mais extremos do espaço que se encontram nos cantos.
- O próximo passo lógico é formular uma maneira do braço percorrer todos os pontos presentes agora no espaço de forma metódica de modo a posteriormente guardar as informações obtidas. Para isto foi utilizado dois ciclos for, que percorrem as linhas todas e as colunas todas presentes no espaço de trabalho.
- Colocando agora as equações correspondentes à cinemática directa e inversa em prática vamos, para cada ponto que é indicado ao robô, obter os valores das coordenadas exteriores e dos ângulos internos e armazenar essa informação numa matriz Mt.
- Também vão ser calculadas as coordenadas exteriores e ângulos correspondentes ao modelo simétrico do robô, cujas informações vão ser armazenadas numa matriz Mts.
- Agora que temos todas as informações sobre o espaço necessárias vamos introduzir as coordenadas externas como entrada na rede neuronal de modo a obter os valores dos ângulos internos como saída da rede neuronal (objectivo).
- A rede neuronal irá agora aprender todos os ângulos internos que correspondem às coordenadas externas, através do modelo FeedForward em que a resposta é encaminhada da entrada para a saída e através do modelo de retro propagação vai corrigir o valor de saída de modo a corresponder ao valor desejado após algumas iterações.

B. Dados de espaço de trabalho

Atribuímos ao espaço trabalho dimensões iguais de x e y de modo a obter um espaço de trabalho de formato quadrado em que cada lado corresponde a 11 pontos (5 para ambos os lados e a origem). De modo a poder percorrer todos os pontos temos de nos certificar que os braços têm comprimento (em conjunto) suficiente para chegar aos pontos mais distantes do espaço de trabalho que são os cantos do quadrado. Usando a fórmula da distância Euclidiana verificamos que seria $sqrt(50) = 7.07$,podendo então fazer corresponder a cada braço ,do braço robótico completo, metade do valor da distância euclidiana resultando em 3.55 aproximademente.Vamos também introduzir um obstáculo no percurso de modo a forçar o robô, com a aprendizagem feita do espaço, a se certificar se as coordenadas externas do ponto pedido podem ser permitidas ou não, sendo impossível atingir esse ponto caso ele se encontre dentro das coordenadas do obstáculo ou se algum dos braços o intersecte.

Listagem 1: Dimensionamento espaço trabalho.

```
%Distancia dos bracos do robo
%Precisam ser >= maior distancia do
    espaco
a=3.55;
b=3.55;
braco=[0;a;a+b];
origem=[0 0];

%Nr pontos espaco
dx=11; %vai de -5 a 5 (a origem tambem
    conta)
dy=11; %vai de -5 a 5

%Visual (Iniciar figura)
x=0;
y=0;
figure(1);
plot(braco,[0;0;0],'b','LineWidth',3);
h=plot(braco,[0;0;0],'k','LineWidth',3);
axis([-10 10 -10 10]);
grid on;
hold on;
%obstaculos (quadrado)
rectangle('Position',[1 1 1 1],'FaceColor
    ','b');
%Plots para intersecao
xi=[1 2];
xi2=[2 1];
plot(xi);
plot(xi2);
```

O método recursivo usado para percorrer o espaço todo vai-se basear no salto entre colunas desde o ponto mais extremo na esquerda para o ponto mais extremo na direita, saltando para a linha seguinte uma vez que se atinga o ponto mais extremo direito. Este método irá se repetir até serem percorridas as 11 linhas do espaço de trabalho e as suas 11 colunas correspondentes. Serão também usadas as equações de cinemática directas e inversas para a posição normal e simétrica do robô que se vai analisar posteriormente.

Listagem 2: Posicionamente dos braços do robô.

```
x=-5; %canto sup esquerdo
y=6;  %canto sup esquerdo
%Matriz posicao e angulos correspondentes
Mt=[];
% Matriz posicao e angulos
    correspondentes (simetrica)
Mts=[];
 for i=1:dy
     y=y-1;
     x=-5;
     for j=1:dx
% Cinematica Inversa
         aux=(x.^2+y.^2-a*a-b*b)/(2*a*b);
          if abs(aux)<=1
           beta=acos(aux);
           betas=-acos(aux);
          end
beta=acos(aux);
betas=-acos(aux);
alfa=atan2(y,x)-atan2(b*sin(beta),a+b*cos
    (beta));
alfas=atan2(y,x)-atan2(b*sin(betas),a+b*
    cos(betas));

%Cinematica directa
% Coordenadas  ponto Q
Qx=a*cos(alfa)
Qy=a*sin(alfa)
Qxs=a*cos(alfas);
Qys=a*sin(alfas);

% Coordenadas ponto P
Px=round(Qx+b*cos(alfa+beta))
Py=round(Qy+b*sin(alfa+beta))
Pxs=round(Qxs+b*cos(alfas+betas));
Pys=round(Qys+b*sin(alfas+betas));

%nova posicao (visual)
set(h(1),'XData',[0;Qx;Px],'YData',[0;Qy;
    Py]);
plot(Px,Py,'r.');
%Incremento proxima coluna
x=x+1;
%Actualizacao matrizes com dados
Mt=[Mt;Px,Py,alfa,beta];
Mts=[Mts;Pxs,Pys,alfas,betas];
     end
end
```

C. *Equações cinemática directa*

Tendo em conta a figura 1, vamos deduzir as coordenadas externas(x,y) da articulação Q e do ponto extremo P do robô a partir dos ângulos internos.

$$Q = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a * cos(\alpha) \\ a * sin(\alpha) \end{bmatrix}$$

$$P = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} a * cos(\alpha) + b * cos(\alpha + \beta) \\ a * sin(\alpha) + b * sin(\alpha + \beta) \end{bmatrix}$$

Listagem 3: Cinemática directa Matlab.

```
 % Coordenadas do ponto Q
Qx=a*cos(alfa)
Qy=a*sin(alfa)
 % Coordenadas do ponto P
Px=round(Qx+b*cos(alfa+beta))
Py=round(Qy+b*sin(alfa+beta))
%Actualizacao Matriz informacao
Mt=[Mt;Px,Py,alfa,beta];
```

D. Cinemática Inversa

Vamos agora deduzir as equações correspondentes ao modelo da cinemática inversa de modo a poder calcular os ângulos internos a partir das coordenadas externas (x,y). Deve ser tomada uma medida de precaução para que o valor obtido pelo $\cos^{-1}$ esteja compreendido entre 0 e 1, caso contrário deve ocorrer um erro no código.

$$\vec{OP} = x^2 + y^2 \tag{1}$$

$$\vec{OA} = \vec{OQ} + \vec{QA} = \alpha + b * cos(\beta) \tag{2}$$

$$\vec{AP} = b * sin(\beta) \tag{3}$$

$$\vec{OP}^2 = \vec{OA}^2 + \vec{AP}^2 \tag{4}$$

$$\begin{aligned} x^2 + y^2 = (\alpha + b * cos(\beta))^2 + b^2 * sin^2(\beta) <=> \\ <=> \beta = cos^{-1}\left(\frac{x^2 + y^2 - a^2 - b^2}{2ab}\right) \end{aligned} \tag{5}$$

$$\begin{aligned} \alpha + \theta = tg^{-1}\frac{y}{x} \\ \theta = tg^{-1}\frac{(b * sin\beta)}{(a + b * cos\beta)} \end{aligned} \tag{6}$$

Fazendo a substituição de θ na equação 6 e organizando a equação em ordem a α:

$$\alpha = tg^{-1}\frac{y}{x} - tg^{-1}\frac{(b * sin\beta)}{(a + b * cos\beta)} \tag{7}$$

Transpondo estas equações para o código MATLAB:

Listagem 4: Cinemática inversa Matlab.

```
aux=(x.^2+y.^2-a*a-b*b)/(2*a*b);
beta=acos(aux);
alfa=atan2(y,x)-atan2(b*sin(beta),a+b*cos
    (beta));
```

E. Posições Simétricas

Analisando a cinemática novamente verifica-se que de modo a obter um resultado espelhado da posição normal é necessário fazer algumas alterações nas variáveis internas (ângulos internos), não esquecendo que a posição do ponto extremo P terá de ser a mesma. Começando pelo $cos^{-1}(\beta)$ que pode tomar um valor negativo ou positivo influenciando a posição do ponto Q, resultando num ângulo côncavo entre primeiro e segundo braço constituintes do braço robótico se o resultado for positivo ou resultando num ângulo convexo caso o valor seja negativo. Desde modo para obtermos a posição simétrica da qual consideramos a posição normal (quando o ângulo no ponto Q é concâvo e o valor do $cos^{-1}(\beta)$ é positivo) além de actualizarmos o sinal do resultado para o simétrico (negativo) temos de substituir na equação correspodente a α o novo ângulo β obtendo assim a posição espelhada da posição normal. Foi feita também uma matriz Mts que vai armazenar toda a informação relativa às coordenadas externas e ângulos internos para este modelo simétrico do braço robótico.

Listagem 5: Posição simétrica do braço robótico.

```
aux=(x.^2+y.^2-a*a-b*b)/(2*a*b);
betas=-acos(aux);
alfas=atan2(y,x)-atan2(b*sin(betas),a+b*
    cos(betas));
% Coordenadas do ponto Q simetrico
Qxs=a*cos(alfas);
Qys=a*sin(alfas);
% Coordenadas do ponto P
Pxs=round(Qxs+b*cos(alfas+betas));
Pys=round(Qys+b*sin(alfas+betas));
%Actualizacao Matriz simetrica
Mts=[Mts;Pxs,Pys,alfas,betas];
```

F. Obstáculo e colisões

Introduzindo um obstáculo no espaço de trabalho do robô vão surgir alguns problemas:

- As coordenadas externas pretendidas podem estar dentro do obstáculo não sendo possível ao robô de alcançar essa zona.
- Os braços do robô ao ajustarem-se para chegar a uma posição específica podem embater no obstáculo, ficando impossibilitado, portanto, de atingir as coordenadas externas pretendidas.

Sendo assim devemos impossibilitar do robô durante a sua aprendizagem de armazenar informação relativa a zonas em que não é possível chegar, colocando o valor dos ângulos internos para essas coordenadas externas com um valor infinito de modo a ficarem inutilizáveis posteriormente. No entanto o robô pode ser capaz de atingir uma dada coordenada externa pela sua posição simétrica e pela posição normal ser incapaz, tendo assim que organizar a informação acerca do espaço de trabalho em duas matrizes distintas, uma para coordenadas externas e ângulos internos correspondentes da posição normal e outra para a posição simétrica. Em termos práticos para conseguirmos identificar se o braço robótico está de facto a colidir com o objecto foram introduzidas dois segmentos de recta cruzados dentro do obstáculo de modo a conseguirmos fazer a intersecção entre o braço do robô e estes segmentos. Caso ambos os segmentos de recta de intersectem, o cálculo da distância da intersecção entre os dois segmentos de recta vai ser zero significando que há uma colisão. Caso não se intersectem o resultado do cálculo da distância entre os dois segmentos será diferente de zero significando que existe uma distância entre o braço robótico e o obstáculo. Para fazer este cálculo da distância entre dois segmentos de recta usamos uma função desenvolvida na aula com o professor, que vai ser analisado na secção seguinte.

Analisando o nosso código temos os inputs 'sr' que correspondem aos segmentos de recta que constituem o objecto, xi e xi2 que corresponde aos segmentos de recta dentro do obstáculo e as distâncias correspondentes da origem à articulação Q, da articulação Q até ao ponto extremo P e as distâncias das suas posições simétricas aos segmentos de recta presentes dentro do obstáculo (xi e xi2). Por fim impondo algumas regras à impressão das coordenadas externas

e ângulos internos para a posição correspodente na matriz Mt e Mts, vamos ter uma impressão sem alterações quando as coordenadas externas são atingidas sem dificuldades mas quando isso não é possível devido às implicações previamente impostas, imprimimos por cima da informação que tinha sido colocada sobre essas coordenadas externas novas informações com o valor dos ângulos internos como infinito. Usando um plot com cor diferente e forma de cruz (X), podemos visualizar na figura resultante 3 as coordendas externas do nosso espaço de trabalho nas quais o braço do robô não consegue alcançar sem colisão , estando representadas a verde as colisões da posição normal e a azul as da posição simétrica do braço do robô.

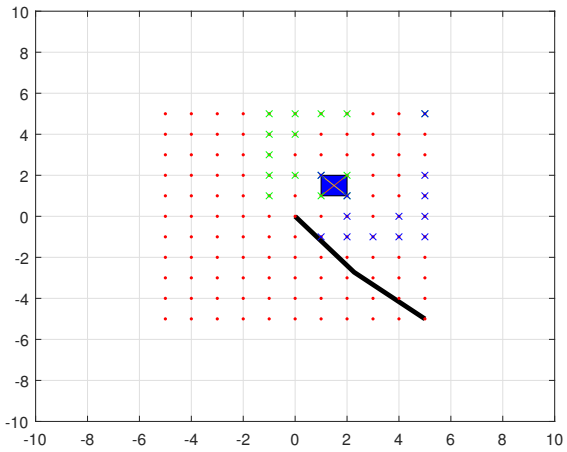

Fig. 3: Coordendas externas alcançáveis

Listagem 6: Algoritmo que lida com obstáculo.

```
%Colisao
coli=1; %incremento matriz colisoes
nrs=0.001; %tender para 0
sr_ia=[1 1]; %segmento recta intersecao
sr_ib=[2 2];
sr_ic=[2 1];
sr_id=[1 2];
Qb=[Qx,Qy]; %ponto Q braco
Qs=[Qxs,Qys];
Pb=[Px,Py];
Ps=[Pxs,Pys];

dist_OQ_sri = DistBetween2Segment(origem,
    Qb,sr_ia,sr_ib);
dist_OQ_srii = DistBetween2Segment(origem
    ,Qb,sr_ic,sr_id);
dist_PQ_sri = DistBetween2Segment(Qb,Pb,
    sr_ia,sr_ib);
dist_PQ_srii = DistBetween2Segment(Qb,Pb,
    sr_ic,sr_id);

dist_OQs_sri = DistBetween2Segment(origem
    ,Qs,sr_ia,sr_ib);
dist_OQs_srii = DistBetween2Segment(
    origem,Qs,sr_ic,sr_id);
dist_PQs_sri = DistBetween2Segment(Qs,Ps,
    sr_ia,sr_ib);
dist_PQs_srii = DistBetween2Segment(Qs,Ps
    ,sr_ic,sr_id);

   if dist_OQ_sri<nrs || dist_OQ_srii<nrs
     || dist_PQ_sri<nrs||dist_PQ_srii<nrs
    ...
       Px==1 && Py==1 || Px==1 && Py==2 ||
     Px==2 && Py==1 || Px==2 && Py==2;

          Mt(coli,1:4)=[Px,Py,Inf,Inf];
          plot(Px,Py,'gx');
   end

   if dist_OQs_sri<nrs || dist_OQs_srii<
    nrs || dist_PQs_sri<nrs ||
    dist_PQs_srii<nrs ...
       Pxs==1 && Pys==1 || Pxs==1 && Pys
    ==2 || Pxs==2 && Pys==1 || Pxs==2 &&
    Pys==2;

          Mts(coli,1:4)=[Pxs,Pys,Inf,Inf];
          plot(Px,Py,'bx');
   end
coli=coli+1; %incremento actualiz matriz
```

G. *Distância entre dois segmentos de recta*

No âmbito da distância entre segmentos vamos nos deparar com três situações:

- Segmentos de recta paralelos.
- Segmentos de recta intersectam-se.
- Segmentos de recta com um dos extremos muito mais próximo do outro segmento.

Para o cálculo da distância entre dois segmentos de recta vamos considerar dois segmentos AB e CD com um valor alfa e beta correspondente que nos vai ser útil para identificar qual o ponto mais próximo de um dos segmentos ao outro segmento de recta e em que posição do segmento se encontra esse ponto,seja isto numa intersecção ou quando existe uma distância entre os segmentos. De modo a determinar a correspondência do alfa e beta vamos derivar M (que corresponde à distância entre o ponto P e Q).

$$M = PQ^T.PQ = (CA+\alpha\vec{AB}-\beta\vec{CD})^T(CA+\alpha\vec{AB}-\beta\vec{CD}) \quad (8)$$

$$\alpha = \frac{-(\vec{CD}^T\vec{CD})(\vec{AB}^T\vec{CA}) + (\vec{AB}^T\vec{CD})(\vec{CD}^TCA)}{den} \quad (9)$$

$$\beta = \frac{-(\vec{AB}^T\vec{AB})(\vec{CD}^T\vec{CA}) - (\vec{CD}^T\vec{AB})(\vec{AB}^TCA)}{den} \quad (10)$$

$$den = ad-bc = (\vec{AB}^T\vec{AB})(\vec{CD}^T\vec{CD})-(\vec{AB}^T\vec{CD})(\vec{CD}^T CA) \quad (11)$$

Deste modo quando alfa ou beta são 0, vai significar que o ponto se encontra no início do segmento correspondente, neste caso em A ou C. Por outro lado quando alfa e beta são iguais ao valor unitário significa que o ponto se encontra no outro extremo do segmento em B ou D, respectivamente. Vamos introduzir dois pontos novos variáveis, P e Q, que se vão encontrar dentro do segmento de recta na posição mais conveniente conforme os valores que alfa e beta tomar.

1) Caso 1 - Intersecção de segmentos: Neste caso alfa e beta têm um resultado que se encontra dentro do intervalo [0,1] e o resultado do cálculo da distância entre os dois segmentos deve ser igual a 0. Caso o alfa e beta não se encontrem dentro do intervalo referido é calculado o valor da multiplicação do módulo de alfa pelo módulo de AB e da multiplicação do módulo de beta pelo módulo do segmento CD. Estes valores serão necessários para diferenciar os próximos casos nas etapas seguintes.

Listagem 7: Caso 1.

```
if ~(alfa>=0 && beta >=0 && alfa<=1 &&
    beta<=1)
    if alfa>1
        a=(alfa-1)*sqrt(m2AB);
else
    if alfa<0
        a=alfa*sqrt(m2AB);
    else
        a=0;
    end
end
if beta>1
    b=(beta-1)*sqrt(m2CD);
else
if beta<0
    b=beta*sqrt(m2CD);
else
    b=0;
end
end
```

2) Caso 2A - Beta como factor limitante: Neste segundo caso temos a situação em que o valor absoluto da multiplicação de beta pelo segmento correspondente (CD) é maior que a multiplicação de alfa pelo segmento AD. Nesta situação beta é o factor limitante e portanto vai ser o primeiro valor a ser actualizado, caso seja superior a 1 é actualizado para 1, implicando que o ponto mais próximo do outro segmento (AB) é Q que se encontra na posição D (extremo) do segmento original. Por outro lado se for inferior a 0, é actualizado para o valor 0 significando que o ponto Q se encontra no ponto C do segmento CD. É posteriormente calculado o alfa resultante através da expressão:

$$\alpha = \frac{\beta\vec{AB}^T\vec{CD} - \vec{AB}^T CA}{\vec{AB}^T\vec{AB}} \quad (12)$$

Agora é necessário verificar em que região do segmento se encontra o ponto P para poder calcular a distância entre pontos.

- Se $0 \leq \alpha \leq 1, P = A + \alpha\vec{AB}$.
- $\alpha < 0, P = A$.
- $\alpha > 1, P = B$.

Por fim é calculado o valor do ponto P e Q e a distância entre eles segundo as seguintes expressões:

$$P = A + \alpha.AB \quad (13)$$

$$Q = C + \beta.CD \quad (14)$$

$$dist = \sqrt{((Q-P)^T.(Q-P))} \quad (15)$$

3) Caso 2B - Alfa como factor limitante: Se $|\beta|\left|\vec{CD}\right| < |\alpha|\left|\vec{AD}\right|$ significa que alfa é o factor limitante e como tal deve ser ajustado o ponto P de acordo.
Se $\alpha > 1, \alpha = 1$ e $P = B$, mas caso $\alpha < 0, \alpha = 0$ e $P = A$.
A expressão usada para calcular beta:

$$\beta = \frac{\vec{CD}^T CA + \alpha\vec{CD}^T\vec{AB}}{\vec{CD}^T\vec{CD}} \quad (16)$$

- Se $0 \leq \beta \leq 1, Q = C + \beta\vec{CD}$.
- $\beta < 0 => Q = C$.
- $\beta > 1 => Q = D$.

Por fim os pontos P e Q são calculados através das expressões 13 e 14, e a distância entre eles calculada pela expressão 15.

4) Caso 3 - Segmentos de recta paralelos: No caso dos segmentos paralelos estes podem não estar exactamente um por cima do outro o que vai dificultar a escolha do extremo do segmento a ser escolhido para calcular a distância ao segmento oposto. De modo a contornar este problema vai ser calculado a distância de cada ponto extremo ao segmento oposto e vai ser escolhida a distância mínima resultante desses quatro cálculos como resultado final representante da distância entre segmentos paralelos.

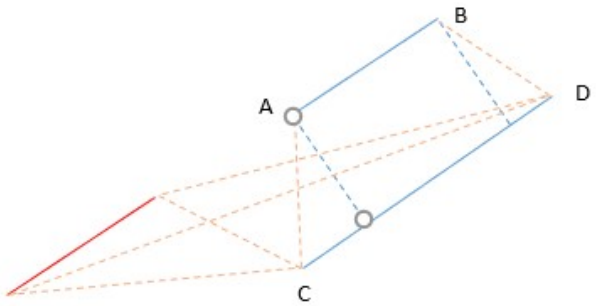

Fig. 4: Segmentos de recta paralelos

Como nesta situação é preciso fazer um cálculo de distância entre um ponto e segmento de recta deriva-se a expressão:

$$M = PQ^T \cdot PQ = \left(PA + \alpha\vec{AB}\right)^T \left(PA + \alpha\vec{AB}\right) \quad (17)$$

$$\alpha = -\frac{\vec{AB}^T \vec{PA}}{\vec{AB}^T \vec{AB}} \quad (18)$$

Se alfa for superior a 1 é actualizado para alfa=1.Caso alfa¡0 é actualizado para alfa=0. Por fim é calculado o ponto correspondente ao segmento AB que se encontra mais próximo do ponto P e é calculada a distância através da expressão:

$$Q = A + \alpha.AB \quad (19)$$

$$D = \sqrt{((Q-P)^T.(Q-P))} \quad (20)$$

H. Redes Neuronais

Esta seção do código foi aproveitada uma função presente numa toolbox do MATLAB (Deep Learning Toolbox) [5] para tratar da estrutura e método de funcionamento de uma rede neural feedforward com retro propagação. Assim a entrada será as coordenadas externas de um ponto e a saída dessa rede neural será os valores dos ângulos internos, foram usadas 3 camadas no total havendo uma camada intermédia entre a entrada e saída. O método de retro propagação usado foi o de Levenberg-Marquardt [6] e o método para calcular o erro foi o do erro quadrático.

Listagem 8: Rede neuronal feedforward com retro propagação.

```
%Matriz das aprendizagens
Mnn=[];
%nr linhas da matriz Mt
sMt=size(Mt,1);
  for inn=1:sMt

     x=Mt(inn,1:2); %entrada NN
     y=Mt(inn,3:4); %saida NN

    np=size(x,2);
    net = newff(x,y,4);
    net = train(net,x,y);
    yN = net(x); %output da rede neuronal
    errors = yN - y; %erro entre saida NN
    e valor desejado
    E2=sum(errors.^2)/np; %E^2

%Corrigir valor para i,j igual
     if x(1,1)==x(1,2)
       yN=yN-errors;
     end
%Matriz aprendizagem NN
Mnn=[Mnn;Mt(inn,1),Mt(inn,2),yN(1,1),yN
    (1,2)];
end
```

III. Conclusões

A solução através da cinemática inversa usando redes neurais para um robô de duas articulações é apresentada num espaço de trabalho com obstáculos. A rede neuronal é treinada até o erro ser aceitável, correspondendo na sua saída aos valores obtidos dos ângulos internos das articulações para uma dada coordenada externa que é verificada na primeira experiência prática de reconhecimento do espaço em que o robô alcança todos os pontos do espaço. Os valores de saída da rede neural são ajustados com base no método de retro propagação de modo a atingirem resultados desejáveis. O procedimento usado para evitar obstáculos foi com base na posição do obstáculo e na colisão entre o braço robótico e o obstáculo que pode acontecer ao alcançar algumas coordenadas externas. Foram usadas regras de cálculo de distância de segmentos de recta para auxiliar neste processo de colisão com obstáculos, colocando dois segmentos de recta cruzados entre os vértices do obstáculo de modo a posteriormente podermos utilizar a distância desses segmentos ao segmento do braço robótico (ao braço que começa na origem e atinge a articulação Q, e ao braço que vai da articulação Q ao ponto extremo P). Um aspecto interessante a desenvolver no seguimento deste projecto seria usar um espaço de trabalho em que os obstáculos mudavam de posição com o tempo e o objectivo seria alcançar pontos específicos do espaço como se tratasse de um tapete rolante de uma fábrica onde circulam peças que o robô deve alcançar e objectos que deve evitar.

Referências bilbiográficas

[1] Wikipedia (2020), *LaTeX: Robotic Kinematics.* Available at: *https://en.wikipedia.org/wiki/Robot_kinematics.*

[2] Wikipedia (2020), *LaTeX: Inverse Kinematics.* Available at: *https://en.wikipedia.org/wiki/Inverse_kinematics*

[3] Wikipedia (2020), *LaTeX: Artificial neural network.* Available at: *https://en.wikipedia.org/wiki/Artificial_neural_network*

[4] MathWorks (2020), *LaTeX: Shortest distance between two line segments.* Available at: *https://www.mathworks.com/matlabcentral/fileexchange/32487-shortest-distance-between-two-line-segments*

[5] MathWorks (2020), *LaTeX: Deep Learning Toolbox.* Available at: *https://www.mathworks.com/products/deep-learning.html*

[6] MathWorks (2020), *LaTeX: Levenberg-Marquardt Algorithm.* Available at: *https://www.mathworks.com/help/deeplearning/ref/trainlm.html*