

MSC HUMAN AND BIOLOGICAL ROBOTICS

Author:

Edward McLAUGHLIN

Course Co-ordinator:

Prof. Etienne Burdet

Human Neuro-Mechanical Control and Learning: Tutorial 2: Kinematics and Redundancy

**Imperial College
London**

DEPARTMENT OF BIOENGINEERING

February 7, 2018

1 Question 1

1.1 Part a

Direct Kinematics

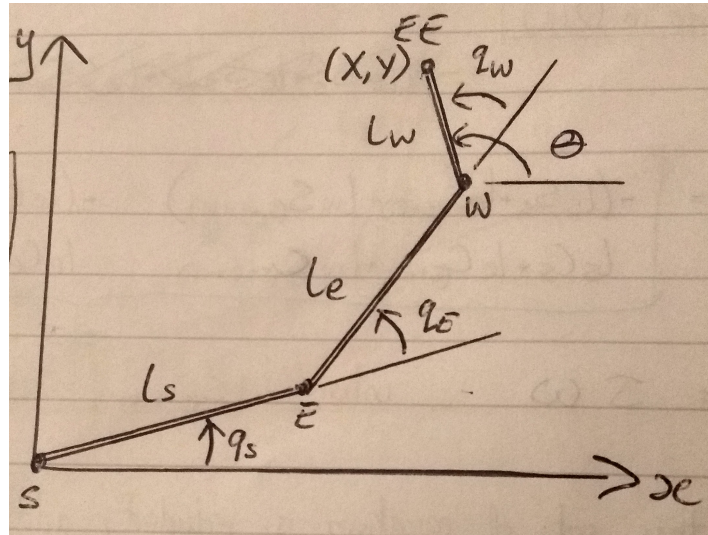


Figure 1

In direct kinematics, the Cartesian coordinates of the end-effector are found as a function of joint space angles. The three bar system shown in 1 represents the human arm from shoulder to end-effector (i.e. hand).

Defining:

$$P_S = \begin{bmatrix} X_S \\ Y_S \end{bmatrix} \quad ; \quad P_E = \begin{bmatrix} X_E \\ Y_E \end{bmatrix} \quad ; \quad P_W = \begin{bmatrix} X_W \\ Y_W \end{bmatrix} \quad ; \quad P_{EE} = \begin{bmatrix} X_{EE} \\ Y_{EE} \end{bmatrix} \quad (1.1)$$

Let the shoulder position represent the origin, i.e.

$$P_S = \begin{bmatrix} X_S \\ Y_S \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (1.2)$$

Therefore,

$$P_E = \begin{bmatrix} X_E \\ Y_E \end{bmatrix} = \begin{bmatrix} l_s \cos(q_s) \\ l_s \sin(q_s) \end{bmatrix} \quad (1.3)$$

$$P_W = \begin{bmatrix} X_W \\ Y_W \end{bmatrix} = \begin{bmatrix} l_s \cos(q_s) + l_e \cos(q_s + q_e) \\ l_s \sin(q_s) + l_e \sin(q_s + q_e) \end{bmatrix} \quad (1.4)$$

$$P_{EE} = \begin{bmatrix} X_{EE} \\ Y_{EE} \end{bmatrix} = \begin{bmatrix} l_s \cos(q_s) + l_e \cos(q_s + q_e) + l_w \cos(q_s + q_e + q_w) \\ l_s \sin(q_s) + l_e \sin(q_s + q_e) + l_w \sin(q_s + q_e + q_w) \end{bmatrix} \quad (1.5)$$

Equation 1.5 gives the direct kinematics between the shoulder and the end-effector.

Inverse Kinematics

The inverse kinematics of the system can be found either geometrically or through differential kinematics. The former will be dealt with here, the latter will be considered in part 1.2.

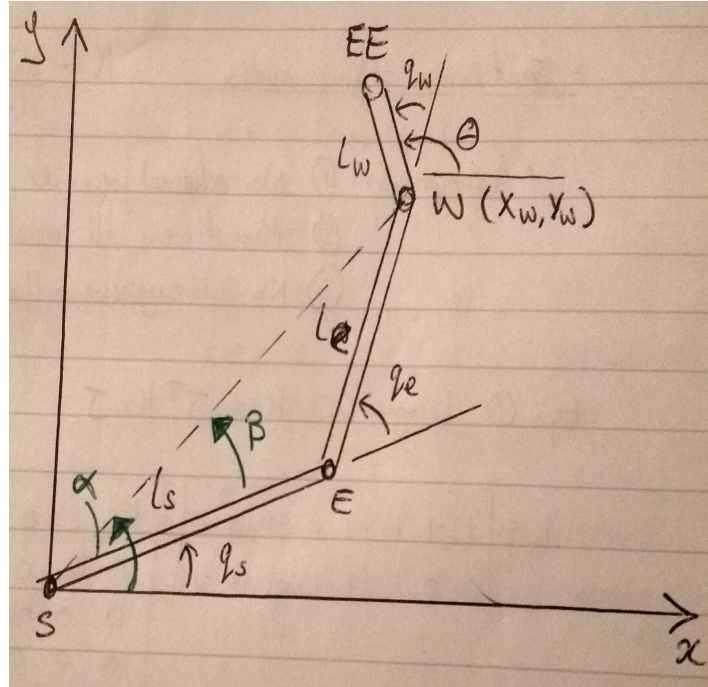


Figure 2

Using the cosine rule,

$$a^2 = b^2 + c^2 - 2ab \cos(\alpha) \quad (1.6)$$

it can be shown that

$$X_w^2 + Y_w^2 = l_s^2 + l_e^2 - 2 l_s l_e \cos(\pi - q_e) \quad (1.7)$$

$$\& \quad l_e^2 = X_w^2 + Y_w^2 + l_s^2 - 2 l_s \sqrt{X_w^2 + Y_w^2} \cos(\beta) \quad (1.8)$$

From equations 1.7-1.8, q_e and β can be expressed as

$$q_e = \cos^{-1} \left(\frac{X_w^2 + Y_w^2 - l_s^2 - l_e^2}{2 l_s l_e} \right) \quad (1.9)$$

$$\& \quad \beta = \cos^{-1} \left(\frac{X_w^2 + Y_w^2 + l_s^2 - l_e^2}{2 l_s \sqrt{X_w^2 + Y_w^2}} \right) \quad (1.10)$$

It can also be shown that

$$\alpha = \arctan2\left(\frac{Y_w}{X_w}\right) \quad (1.11)$$

$$q_s = \alpha - \beta \quad (1.12)$$

$$= \arctan2\left(\frac{Y_w}{X_w}\right) - \cos^{-1}\left(\frac{X_w^2 + Y_w^2 + l_s^2 - l_e^2}{2 l_s \sqrt{X_w^2 + Y_w^2}}\right) \quad (1.13)$$

Adding a constraint on the wrist orientation such that $\Theta = q_s + q_e + q_w$, therefore

$$q_w = \Theta - q_s - q_e \quad (1.14)$$

Thus the inverse kinematics have been determined in equations 1.9, 1.13 and 1.14, which relate the angles to desired X and Y position of the wrist. Since the wrist angle is predetermined, these equations also given the desired angles for a given position of the end-effector with an offset of $-l_w \cos(\Theta)$ in the x-direction and $l_w \sin(\Theta)$ in the y-direction.

1.2 Part b

Differential Kinematics and Joint Space Jacobian

Knowing P_{EE} from part 1.1, the velocity profile, V_{EE} can be calculated as

$$V_{EE} = \frac{dP_{EE}}{dt} = \begin{bmatrix} -l_s \sin(q_s) \dot{q}_s - l_e \sin(q_s + q_e) (\dot{q}_s + \dot{q}_e) - l_w \sin(q_s + q_e + q_w) (\dot{q}_s + \dot{q}_e + \dot{q}_w) \\ l_s \cos(q_s) \dot{q}_s + l_e \cos(q_s + q_e) (\dot{q}_s + \dot{q}_e) + l_w \cos(q_s + q_e + q_w) (\dot{q}_s + \dot{q}_e + \dot{q}_w) \end{bmatrix} \quad (1.15)$$

$$\begin{bmatrix} V_{XEE} \\ V_{YEE} \end{bmatrix} = J \omega \quad (1.16)$$

where

$$J = \begin{bmatrix} -l_s \sin(q_s) - l_e \sin(q_s + q_e) - l_w \sin(q_s + q_e + q_w), \\ l_s \cos(q_s) + l_e \cos(q_s + q_e) + l_w \cos(q_s + q_e + q_w), \\ -l_e \sin(q_s + q_e) - l_w \sin(q_s + q_e + q_w), & -l_w \sin(q_s + q_e + q_w) \\ l_e \sin(q_s + q_e) + l_w \cos(q_s + q_e + q_w), & l_w \cos(q_s + q_e + q_w) \end{bmatrix} \quad (1.17)$$

$$\& \quad \omega = \begin{bmatrix} \dot{q}_s \\ \dot{q}_e \\ \dot{q}_w \end{bmatrix} \quad (1.18)$$

In this case, as the jacobian matrix is not square, there is redundancy in the system and thus adding the constrain in equation 1.14, we can find the inverse of the jacobian in order to implement inverse differential kinematics. In this case,

$$J = \begin{bmatrix} -l_s \sin(q_s) - l_e \sin(q_s + q_e) - l_w \sin(q_s + q_e + q_w), \\ l_s \cos(q_s) + l_e \cos(q_s + q_e) + l_w \cos(q_s + q_e + q_w), \\ 1, \\ -l_e \sin(q_s + q_e) - l_w \sin(q_s + q_e + q_w), & -l_w \sin(q_s + q_e + q_w) \\ l_e \sin(q_s + q_e) + l_w \cos(q_s + q_e + q_w), & l_w \cos(q_s + q_e + q_w) \\ 1, & 1 \end{bmatrix} \quad (1.19)$$

$$V'_{EE} = \begin{bmatrix} V_{XEE} \\ V_{YEE} \\ \Theta \end{bmatrix} \quad (1.20)$$

Alternatively, when dealing with a non-square jacobian, we can obtain the penrose-moore pseudo inverse, J^\dagger , such that

$$J^\dagger = (J^T J)^{-1} J^T. \quad (1.21)$$

Muscle Space Jacobian, J_μ

The muscle space Jacobian for the 3 link system including the shoulder, biarticular, elbow and wrist muscles is given as

$$J_\mu = \begin{bmatrix} \rho_{s+}, & 0, & 0 \\ -\rho_{s-}, & 0, & 0 \\ \rho_{bs+}, & \rho_{bs+}, & 0 \\ -\rho_{bs-}, & -\rho_{bs-}, & 0 \\ 0, & \rho_{e+}, & 0 \\ 0, & -\rho_{e-}, & 0 \\ 0, & 0, & \rho_{w+} \\ 0, & 0, & -\rho_{w-} \end{bmatrix} \quad (1.22)$$

Joint Space Stiffness

The muscle stiffness matrix, K , is given by

$$K = \frac{dJ_\mu^T}{dq} F + J_\mu^T K_x J_\mu \quad (1.23)$$

Assuming that:

- there is no external forces: $F = 0$;
- Moment arms are all equal: $\rho_{s+} = \rho_{s-} = \rho_{bs+} = \rho_{bs-} = \rho_{e+} = \rho_{e-} = \rho_{w+} = \rho_{w-} = \rho_m$;
- No heteronymous reflexes: stiffness matrix, K_x is diagonal.

equation 1.23 reduces to

$$K = J_\mu^T K_x J \quad (1.24)$$

$$= \rho_m \begin{bmatrix} 1, & 0, & 0 \\ -1, & 0, & 0 \\ 1, & 1, & 0 \\ -1, & -1, & 0 \\ 0, & 1, & 0 \\ 0, & -1, & 0 \\ 0, & 0, & 1 \\ 0, & 0, & -1 \end{bmatrix}^T \begin{bmatrix} K_{s+}, & 0, & 0, & 0, & 0, & 0, & 0, & 0 \\ 0, & K_{s-}, & 0, & 0, & 0, & 0, & 0, & 0 \\ 0, & 0, & K_{b+}, & 0, & 0, & 0, & 0, & 0 \\ 0, & 0, & 0, & K_{b-}, & 0, & 0, & 0, & 0 \\ 0, & 0, & 0, & 0, & K_{e+}, & 0, & 0, & 0 \\ 0, & 0, & 0, & 0, & 0, & K_{e-}, & 0, & 0 \\ 0, & 0, & 0, & 0, & 0, & 0, & K_{w+}, & 0 \\ 0, & 0, & 0, & 0, & 0, & 0, & 0, & K_{w-} \end{bmatrix} \rho_m \begin{bmatrix} 1, & 0, & 0 \\ -1, & 0, & 0 \\ 1, & 1, & 0 \\ -1, & -1, & 0 \\ 0, & 1, & 0 \\ 0, & -1, & 0 \\ 0, & 0, & 1 \\ 0, & 0, & -1 \end{bmatrix} \quad (1.25)$$

$$= \rho_m^2 \begin{bmatrix} K_s + K_b, & K_b, & 0 \\ K_b, & K_b + K_e, & 0 \\ 0, & 0, & K_w \end{bmatrix} \quad (1.26)$$

Equation 1.26 gives the muscle stiffness matrix.

1.3 Part c

The process for calculating and plotting the position and velocity of the end-effector as well as the joint angles and angular velocities is as follows:

- Define Time parameters
- Define physiological parameters
- Calculate the position and velocity profiles in x and y
- Iteratively calculate the angular speed and hence angles using
 - Evaluate inverse Jacobian based on current angles and moment arms
 - Calculate angular velocity : $\text{angularVelocity} = \text{inverseJacobian} * \text{velocity}$
 - Calculate angles: $\text{Angles} = \text{oldAngles} + \text{angularVelocity} * \text{timestep}$
- Plot position, velocity, angles and angular velocity.

The plots are shown below. The matlab code is given in the appendix.

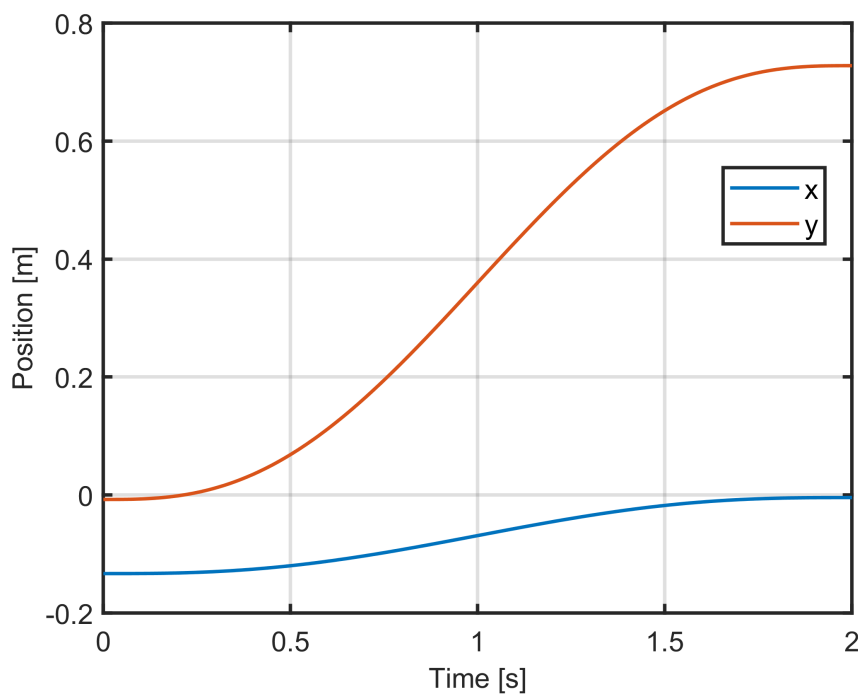


Figure 3: Position profile of the end-effector.

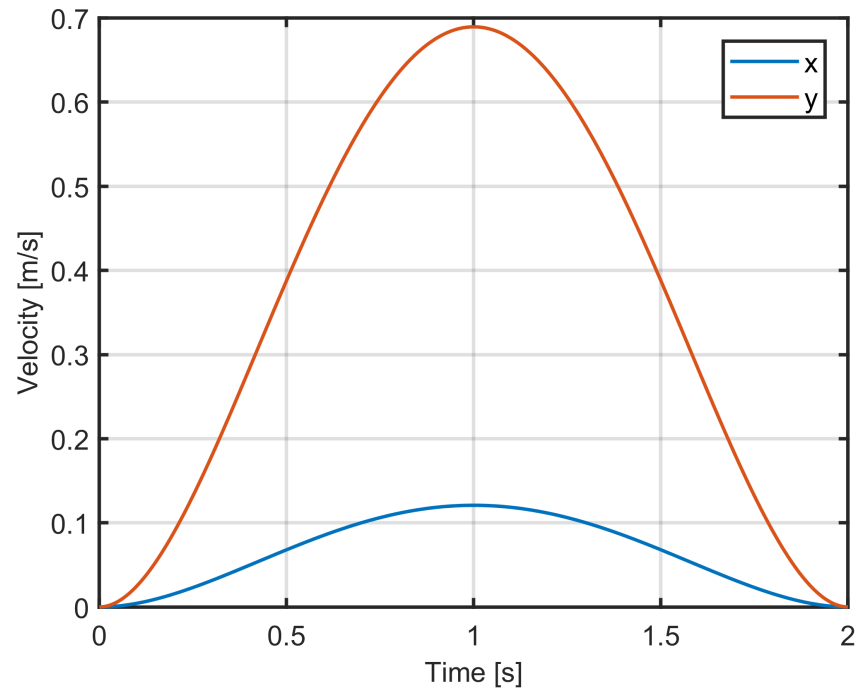


Figure 4: Velocity profile of the end-effector.

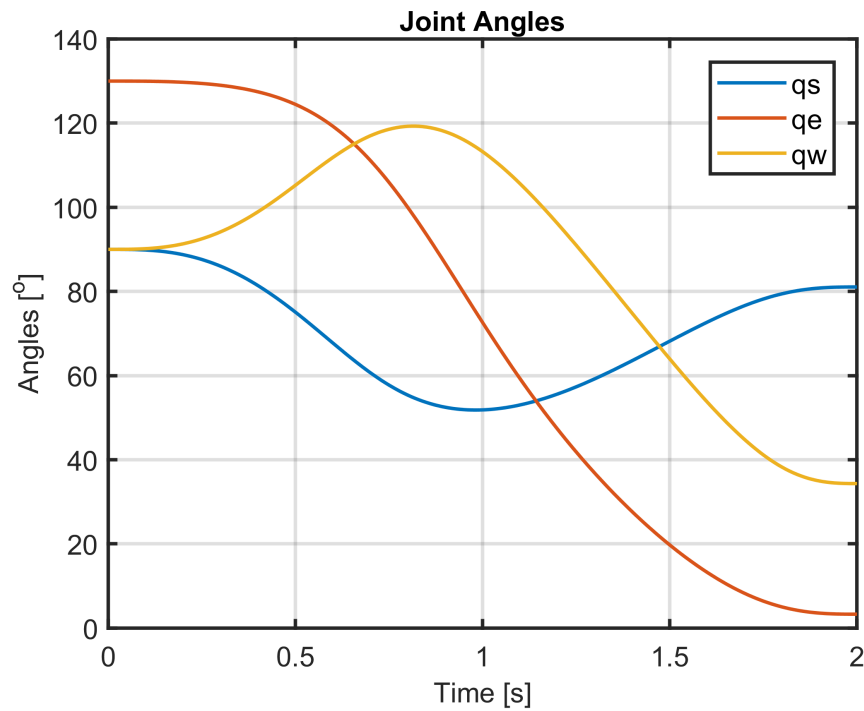


Figure 5: Angles of system to achieve the desired profile.

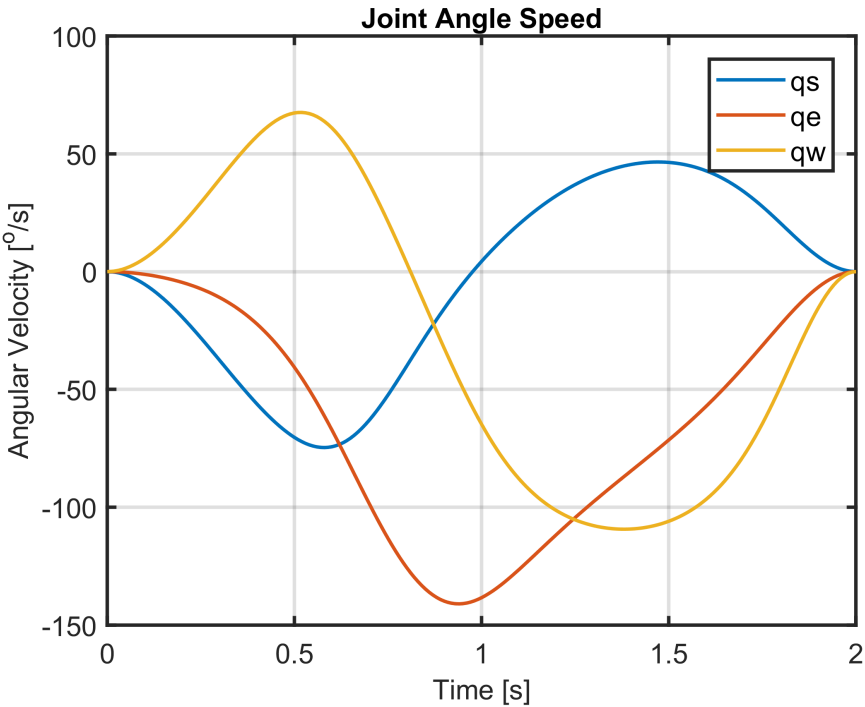


Figure 6: Angular velocities of the system to achieve the desired profile.

2 appendix

```
1
2 %Time properties
3 t = 0:0.01:2;
4 T = 2;
5 tn = t/T;
6
7 %Physiology properties
8 qs = deg2rad(90);
9 qe = deg2rad(130);
10 qw = deg2rad(90);
11 Q(:,1) = [qs;...
12           qe;...
13           qw];
14 ls = 0.3;
15 le = 0.3;
16 lw = 0.15;
17
18 %Position profile as given in the question
19 x_tn = 0.1289*tn.^3.*(6*tn.^2 15*tn+10) 0.1334;
20 y_tn = 0.7355*tn.^3.*(6*tn.^2 15*tn+10) 0.0077;
21
22 %Velocity profile found by derivative of position w.r.t t
23 Vx = 0.1289*(30*tn.^4 60*tn.^3+30*tn.^2)/T;
24 Vy = 0.7355*(30*tn.^4 60*tn.^3+30*tn.^2)/T;
25
26 for i = 1:length(t)
27     %Evaluate inverse jacobian
28     JinvLoop = updateJinv(ls, le, lw, Q(1,i), Q(2,i), Q(3,i));
29
30     %Calculate maximum value in jacobian    mainly for debugging
31     %singularities
32     MaxJ(i) = max(max(JinvLoop));
33
34     %Calculate angular speed
35     omega(:,i) = JinvLoop*[Vx(i); Vy(i)];
36
37     %Calculate new angles
38     Q(:,i+1) = Q(:,i) + omega(:,i)*0.01;
39 end
```