Robot manipulator: project-part 3

Instructor: Prof. Patel

Peyman Yadmellat

1. Computing the Cartesian velocity of end-effector

As mentioned in the book, regarding the mechanical type of joints, two sets of formulas are existed to compute the Cartesian velocity of joints. The sets of formulas are given below

$$\begin{aligned} {}^{i+1}\omega_{i+1} &= {}^{i+1}_{i}R\;{}^{i}\omega_i + \theta_{i+1}\,{}^{i+1}Z_{i+1} \\ {}^{i+1}v_{i+1} &= {}^{i+1}_{i}R\;\;{}^{i}v_i + {}^{i}\omega_i \times {}^{i}P_{i+1} \end{aligned} \tag{1}$$

$$\begin{aligned} {}^{i+1}\omega_{i+1} &= {}^{i+1}_{i}R\;{}^{i}\omega_i \\ {}^{i+1}v_{i+1} &= {}^{i+1}_{i}R\;\;{}^{i}v_i + {}^{i}\omega_i \times {}^{i}P_{i+1} + d_{i+1}\,{}^{i+1}Z_{i+1} \end{aligned} \tag{2}$$

which first one can be used to deal with revolute joints, and second one is to deal with prismatic joints.

Using these equations, we can compute the end-effector velocities, i.e. ${}^{N}\omega_N$ and ${}^{N}v_N$, expressed in terms of end-effector frame N. The velocities can also be rotated into base coordinates by multiplying them with ${}^{0}_{N}R$ if the velocities are desired to be in terms of base frame. Thus, we have

$${}^{0}\omega_N = {}^{0}_{N}R\;{}^{N}\omega_N$$

$${}^{0}v_N = {}^{0}_{N}R\;{}^{N}v_N$$

Two mentioned sets of formulas can be rewritten together by considering a switching parameter γ as follows

$$\begin{aligned} {}^{i+1}\omega_{i+1} &= {}^{i+1}_{i}R\;{}^{i}\omega_i + \gamma\,\theta_{i+1}\,{}^{i+1}Z_{i+1} \\ {}^{i+1}v_{i+1} &= {}^{i+1}_{i}R\;\;{}^{i}v_i + {}^{i}\omega_i \times {}^{i}P_{i+1} + \;\; 1-\gamma\,d_{i+1}\,{}^{i+1}Z_{i+1} \end{aligned} \tag{3}$$

where

$$\begin{aligned} \gamma = 1,&\;\; \text{if joint } i \text{ is revolute} \\ \gamma = 0,&\;\; \text{if joint } i \text{ is prismatic} \end{aligned}$$

2. Problem formulation

In this section, the velocities' equations for two-link and Stanford manipulators will be given.

2.1. Two-link manipulator

Considering Example 5.3, following equations have been rewritten as follow

$$ {}^{0}v_3 = \begin{matrix} -l_1 s_1 \theta_1 - l_2 s_{12} \ \theta_1 + \theta_2 \\ l_1 c_1 \theta_1 + l_2 c_{12} \ \theta_1 + \theta_2 \\ 0 \end{matrix} $$

$$ {}^{0}\omega_3 = \begin{matrix} 0 \\ 0 \\ \theta_1 + \theta_2 \end{matrix} $$

2.2. Stanford manipulator

Considering the manipulator's configuration of Stanford manipulator, we can calculate velocities of each frame by using (1) and (2) sequentially from link to link. Regarding the mechanical type of each joint, we will use equation (1) to compute velocities of two first revolute joints of Stanford manipulator, and equation (2) will be used to compute the last prismatic joint of this manipulator. Considering fixed base frame, i.e. ${}^{N}\omega_N = 0$ and ${}^{N}v_N = 0$, then we have

$$ {}^{1}\omega_1 = {}^{1}_{0}R \ {}^{0}\omega_0 + \theta_1 \ {}^{1}Z_1 = \begin{matrix} 0 & 0 & \theta_1 \end{matrix}^T $$
$$ {}^{1}v_1 = {}^{1}_{0}R \ \ {}^{0}v_0 + {}^{0}\omega_0 \times {}^{0}P_1 \ = \begin{matrix} 0 & 0 & 0 \end{matrix}^T $$

$$ {}^{2}\omega_2 = {}^{2}_{1}R \ {}^{1}\omega_1 + \theta_2 \ {}^{2}Z_2 = \begin{matrix} c\theta_2 & 0 & -s\theta_2 \\ -s\theta_2 & 0 & -c\theta_2 \\ 0 & 1 & 0 \end{matrix} \begin{matrix} 0 \\ 0 \\ \theta_1 \end{matrix} + \begin{matrix} 0 \\ 0 \\ \theta_1 \end{matrix} = \begin{matrix} -\theta_1 s\theta_2 \\ -\theta_1 c\theta_2 \\ \theta_2 \end{matrix} $$

$$ {}^{2}v_2 = {}^{2}_{1}R \ \ {}^{1}v_1 + {}^{1}\omega_1 \times {}^{1}P_2 \ = \begin{matrix} c\theta_2 & 0 & -s\theta_2 \\ -s\theta_2 & 0 & -c\theta_2 \\ 0 & 1 & 0 \end{matrix} \ \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} + \begin{matrix} 0 \\ 0 \\ \theta_1 \end{matrix} \times \begin{matrix} 0 \\ 0.1 \\ 0 \end{matrix} \ = \begin{matrix} -0.1\theta_1 c\theta_2 \\ 0.1\theta_1 s\theta_2 \\ 0 \end{matrix} $$

$$ {}^{3}\omega_3 = {}^{3}_{2}R \ {}^{2}\omega_2 = \begin{matrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{matrix} \begin{matrix} -\theta_1 s\theta_2 \\ -\theta_1 c\theta_2 \\ \theta_2 \end{matrix} = \begin{matrix} -\theta_1 s\theta_2 \\ \theta_2 \\ \theta_1 c\theta_2 \end{matrix} $$

$$ {}^{3}v_3 = {}^{3}_{2}R \ \ {}^{2}v_2 + {}^{2}\omega_2 \times {}^{2}P_3 + d_3 \ {}^{3}Z_3 = \begin{matrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{matrix} \ \begin{matrix} -0.1\theta_1 c\theta_2 \\ 0.1\theta_1 s\theta_2 \\ 0 \end{matrix} + \begin{matrix} -\theta_1 s\theta_2 \\ -\theta_1 c\theta_2 \\ \theta_2 \end{matrix} \times \begin{matrix} 0 \\ -f \\ 0 \end{matrix} $$

$$\begin{array}{c} \\ + \\ \\ \end{array}\begin{array}{c} 0 \\ 0 \\ f \end{array} = \begin{array}{c} -0.1\theta_1 c\theta_2 + f\theta_2 \\ 0.1\theta_1 s\theta_2 \\ -0.1\theta_1 s\theta_2 + f \end{array}$$

Also, the velocities of end-effector can be rotated into terms of base frame as follows

$$\begin{array}{c} {}^0\omega_3 \\ --- \\ {}^0v_3 \end{array} = \begin{array}{cccc} {}^0_3R & | & 0 & {}^0\omega_3 \\ -- & - & -- & --- \\ 0 & | & {}^0_3R & {}^0v_3 \end{array}$$

Now, by substituting numerical values of Stanford manipulator's parameter as given in project definition, we calculate

$$\begin{array}{c} {}^0\omega_3 \\ --- \\ {}^0v_3 \end{array} = \begin{array}{c} -3.2045\,^\circ/s \\ 19.7416\,^\circ/s \\ 35.000\,^\circ/s \\ --- \\ -0.0539m/s \\ 0.1894m/s \\ -0.01542m/s \end{array}$$

3. Matlab numerical results

A Matlab function entitled "Velocity.m" has been written to compute the Cartesian velocity of the end-effector (the last frame). The program is written in a way to be able to deal with all general cases. In addition, a special case is considered in program to compute the Cartesian velocities of Stanford manipulator's end-effector without entering default values of its D-H parameters given in project definition. Syntax of this function is considered as follow

```
[v w]=Velocity
[v w]=Velocity(JointVelocities)
[v w]=Velocity(JointVelocities,alpha,a,d,theta)
[v w]=Velocity(JointVelocities,alpha,a,d,theta,TypeOfJoints)
```

3.1. Description

`[v w]=Velocity` computes linear and angular velocities of Stanford manipulator's end-effector considering all default values given in project definition, i.e. `JointVelocities=[35;20;0.1]`, `alpha=[0;-90;90]`, `a=[0;0;0]`, `d=[0.4;0.1;0.3536]`, `theta=[9.22;115.1;0]`, and `TypeOfJoints = 'RRP'`.

`[v w]=Velocity(JointVelocities)` computes above problem for Stanford manipulator given joint velocities defined by user. In this case default values of D-H parameters assumed to be as given in project definition.

`[v w]=Velocity(JointVelocities,alpha,a,d,theta)` computes linear and angular velocities of the end-effector using joint velocities and D-H parameter defined by user. As default, the number of joint is assumed to be three and mechanical type of joints are considered as `TypeOfJoints = 'RRP'`.

`[v w]=Velocity(JointVelocities,alpha,a,d,theta,TypeOfJoints)` is general case which computes linear and angular velocity of the end-effector given joint velocities, D-H parameters, and mechanical type of each joint.

3.2. Input arguments

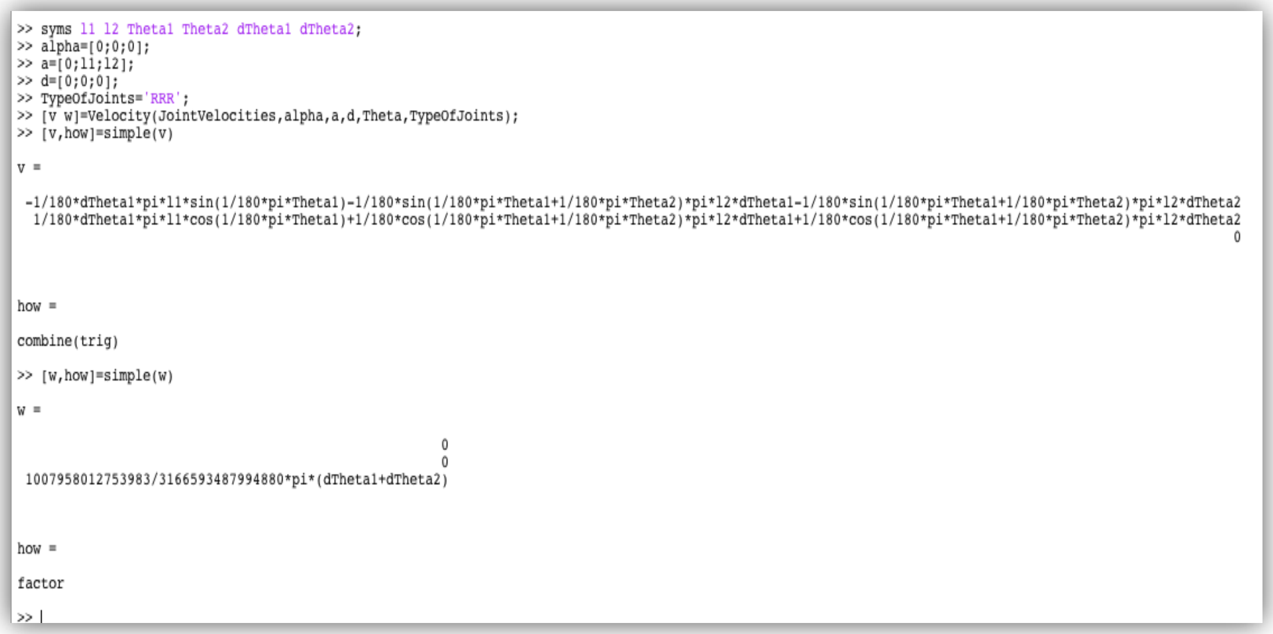
`JointVelocities`	is an either column or row vector of velocity. Default value is set as `JointVelocities=[35;20;0.1].`
`alpha`	is a column vector of α_{i-1}, $i = 1,2,\cdots,n$ where n is the number of joints. Default value is set as `alpha=[0;-90;90].`
`a`	is a column vector of a_{i-1}, $i = 1,2,\cdots,n$ where n is the number of joints. Default value is set as `a=[0;0;0].`
`d`	is a column vector of d_i, $i = 1,2,\cdots,n$ where n is the number of joints. Default value is set as `d=[0.4;0.1;0.3536].`
`theta`	is a column vector of θ_i, $i = 1,2,\cdots,n$ where n is the number of joints. Default value is set as `theta=[9.22;115.1;0].`
`TypeOfJoints`	is a string containing the mechanical type of each joint. Prismatic joint is represented by `'P',` and `'R'` stands for revolute joint. Mechanical type of joints should be written in order. For example, `'RPRR'` (equivalently `['R','P','R','R']`) means that first joint is revolute, second one is prismatic, and the other two are revolute.

3.3. Output arguments

`v`	returns the column vector of linear velocities of end-effector or the last frame of manipulator in meter/second
`w`	returns the column vector of angular velocity of end-effector or the last frame of manipulator in degree/second

3.4. Example

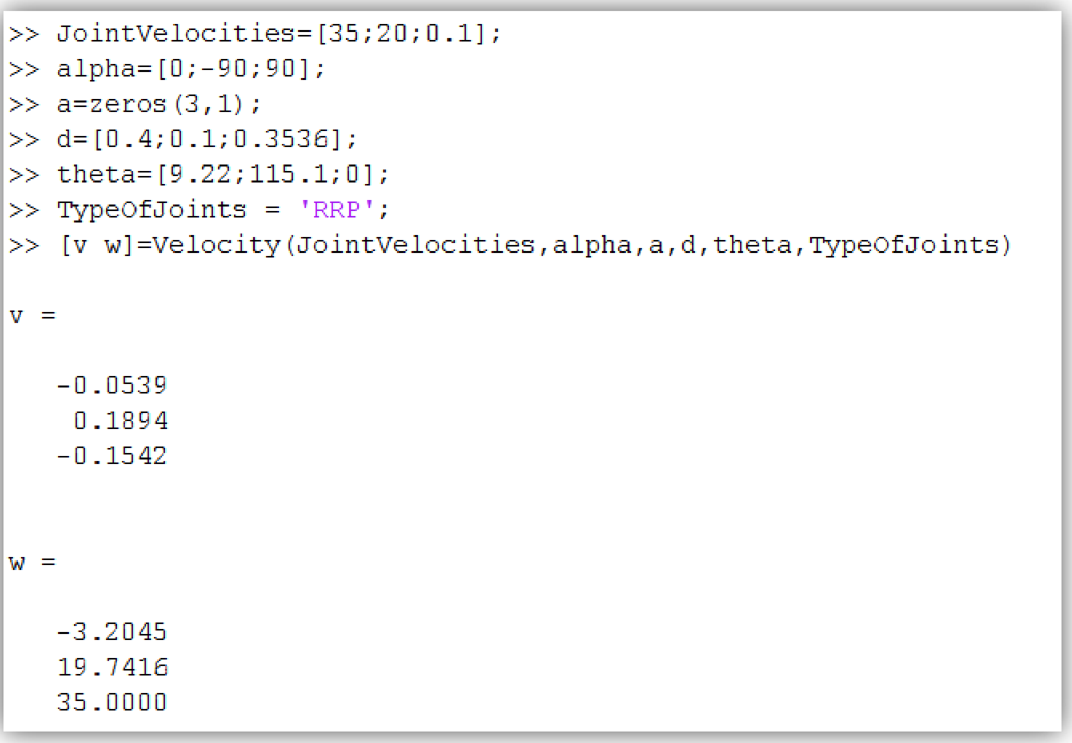
1. Following commands can be used to evaluate the Matlab function for two-link manipulator (Example 5.3)



```
>> syms l1 l2 Theta1 Theta2 dTheta1 dTheta2;
>> alpha=[0;0;0];
>> a=[0;l1;l2];
>> d=[0;0;0];
>> TypeOfJoints='RRR';
>> [v w]=Velocity(JointVelocities,alpha,a,d,Theta,TypeOfJoints);
>> [v,how]=simple(v)

v =

 -1/180*dTheta1*pi*l1*sin(1/180*pi*Theta1)-1/180*sin(1/180*pi*Theta1+1/180*pi*Theta2)*pi*l2*dTheta1-1/180*sin(1/180*pi*Theta1+1/180*pi*Theta2)*pi*l2*dTheta2
  1/180*dTheta1*pi*l1*cos(1/180*pi*Theta1)+1/180*cos(1/180*pi*Theta1+1/180*pi*Theta2)*pi*l2*dTheta1+1/180*cos(1/180*pi*Theta1+1/180*pi*Theta2)*pi*l2*dTheta2
                                                                                                                                                           0


how =

combine(trig)

>> [w,how]=simple(w)

w =

                                                     0
                                                     0
 1007958012753983/316659348799488O*pi*(dTheta1+dTheta2)


how =

factor

>> |
```

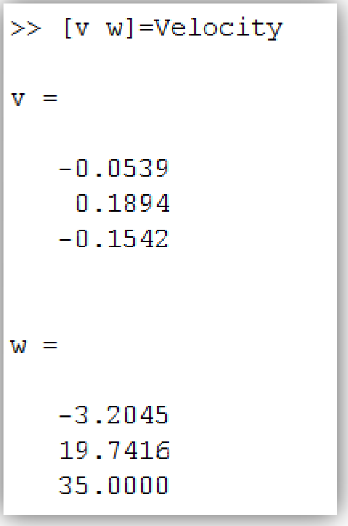
As you can see, the results are the same as the theoretical results which derived in Section 2.1. Notice that the above results are abstained in degree unit instead of radian.

2. The Cartesian velocity of the origin of {4} of the Stanford manipulator can be obtained similarly by using following commands



```
>> JointVelocities=[35;20;0.1];
>> alpha=[0;-90;90];
>> a=zeros(3,1);
>> d=[0.4;0.1;0.3536];
>> theta=[9.22;115.1;0];
>> TypeOfJoints = 'RRP';
>> [v w]=Velocity(JointVelocities,alpha,a,d,theta,TypeOfJoints)

v =

   -0.0539
    0.1894
   -0.1542


w =

   -3.2045
   19.7416
   35.0000
```

Or simply using following command as default values of input arguments are predefined in program based on given D-H parameters and joint velocities of the Stanford manipulator



```
>>  [v w]=Velocity

v =

   -0.0539
    0.1894
   -0.1542


w =

   -3.2045
   19.7416
   35.0000
```

Appendix 1. Matlab code of “Velocity” function:

```
function [v w]=Velocity(JointVelocities,alpha,a,d,theta,TypeOfJoints)
% Robotic manipulators - Project part 3
%
% This function computes the Cartesian velocity of the end-effector, given
% a set of velocities and manipulator's configuration
%
% According to mechanical type of each joint, Two sets of equations
% can be used to compute the Cartesian velocities of end-effector:
%
% In the case of Revolute Joints, link to link velocities can be calculated
% as given below:
%
%         i+1       i+1  i       .      i+1^
%            w     =   R  w + theta      Z                       (R-1)
%             i+1    i    i   i+1     i+1
%---------------------------------------------------
%
%         i+1       i+1   i    i     i
%            v     =   R ( v +  w  x  P   )                      (R-2)
%             i+1    i     i    i      i+1
%
% And in case of Prismatic Joints, link to link velocities can be
% calculated as given below:
%
%         i+1       i+1  i
%            w     =   R  w                                       (P-1)
%             i+1    i    i
%---------------------------------------------------
%
```

```
%        i+1      i+1   i    i     i       .   i+1^
%           v    =   R ( v +  w  x  P   )+ d      Z             (P-2)
%            i+1    i     i    i     i+1      i+1    i+1
%
%                                           N       N
% Also to rotate resulting velocities, i.e.  v  and  w, into base
%                                           N       N
% coordinates, following equation should be used:
%
%           0     0  N
%            v  =  R  v
%             N   N    N
%     ------------------------
%           0     0  N
%            w  =  R  w
%             N   N    N
%
%--------------------------------------------------------------------------
% Input arguments:
%
%   JointVelocities is an either column or row vector of velocity.
%   Default value is set as JointVelocities=[35;20;0.1].
%
%   alpha,a,d,and theta are column vectors corresponding D-H parameters
%   Default values of D-H parameters are set as
%   alpha=[0;-90;90], a=[0;0;0], d=[0.4;0.1;0.3536], and
%   theta=[9.22;115.1;0].
%
%   TypeOfJoints is a string containing the mechanical type of each joint.
%   Prismatic joint is represented by 'P', and 'R' stands for revolute
%   joint. Mechanical type of joints should be written in order. For
%   example, 'RPRR' (equivalently ['R','P','R','R']) means that first
%   joint is revolute, second one is prismatic, and the other two are
%   revolute.
%
% Note that the default values will be considered only in a case of missing
% argument(s). In other word, the default values will be considered
% automatically if those arguments are not entered.
%--------------------------------------------------------------------------
% Output arguments:
% v returns the column vector of linear velocities of end-effector
%   or the last frame of manipulator in meter/second
% w returns the column vector of angular velocity of end-effector
%   or the last frame of manipulator in degree/second
%--------------------------------------------------------------------------
%
% Peyman Yadmellat

% March 09, 2010
% ------------------------------
% ------------------------------

% default type of joints is set for Stanford manipulator
defaultTypeOfJoints = 'RRP';

%default DH parameters considered as DH parameters of Stanford manipulator
```

```
default_alpha=[0;-90;90];
default_a=zeros(3,1);
default_d=[0.4;0.1;0.3536];
default_theta=[9.22;115.1;0];

%default joint velocities are set as given in project definition
defaultJointVelocities=[35;20;0.1];

% Handle missing arguments and set each missed argument as it should be
% for Standford manipulator
if nargin<6,TypeOfJoints = defaultTypeOfJoints;
    if nargin<5,theta=default_theta;
        if nargin<4,d=default_d;
            if nargin<3,a=default_a;
                if nargin<2,alpha=default_alpha;
                    if nargin<1,JointVelocities=defaultJointVelocities;
                    end,end,end,end,end,end

TypeOfJoints=upper(TypeOfJoints); %Set to uppercase

n=length(TypeOfJoints); %number of joints

%Initializing velocities
v=zeros(3,1);
w=zeros(3,1);

% Computing linear and angular velocities of end effector
% respect to end effector frame
for i=1:n
    T=Forward_kin(alpha,a,d,theta,i-1,i);

    % Extracting the rotation matrix of joint i respect to its description
    % in frame {i-1}
    R=T(1:3,1:3).';

    % Extracting the translation matrix of joint i-1 respect to its
    % description in frame {i}
    P=T(1:3,4);

    % this set of formulas computes linear and angular velocities of each
    % joint regarding the equation (3) in report in which these formulas
    % switch between two possible cases according to the mechanical type
    % of each joint. The switching parameter is directly assigned by
    % comparing the mechanical type of each joint to possible Prismatic and
    % Revolute cases by using "isequal" command
    v=R*(v+cross(w,P))+...
        isequal(TypeOfJoints(i),'P')*[0;0;JointVelocities(i)];
    w=R*w+isequal(TypeOfJoints(i),'R')*[0;0;JointVelocities(i)*pi/180];

end


T=Forward_kin(alpha,a,d,theta,0,n);
```

```
%Computing the rotation matrix to rotate velocities of the end-effector
%respect to base coordinates
R=T(1:3,1:3);

% Returns linear velocity respect to base frame in m/s
v=R*v;

% Returns angular velocity respect to base frame in degree/sec
w=180/pi*R*w;
```