NEW MATLAB FUNCTIONS APPLIED TO MECHATRONIC SYSTEMS

Cătălin-Iulian CHIVU, Catrina CHIVU
Transilvania University of Brasov, Romania

Abstract. In the study of mechatronic systems, in particularly of robotics, the constant concern is the location of objects in three-dimensional space. These objects are the links of the manipulator, the parts and tools with which it deals, and other objects in the manipulator's environment. At a crude but important level, these objects are described by just two attributes: position and orientation. Naturally, one topic of immediate interest is the manner in which we represent these quantities and manipulate them mathematically. Matlab program offers some toolboxes that may help but, almost all of them implies to create a geometric model and to generate some input matrixes. All these toolboxes can be used only after many hours of learning, even if the user has very good Matlab knowledge. This was the based idea of the present paper. In this paper the authors present some functions created in Matlab and oriented on mechatronic systems, particularly on robots.

Keywords: robotics, Matlab, SCARA robot, mathematical model, simulation

1. Introduction

In all robot applications, completion of a generic task requires the execution of a specific motion prescribed to the robot. The correct execution of such motion is entrusted to the control system which should provide the robot's actuators with the commands consistent with the desired motion. Motion control demands an accurate analysis of the characteristics of the mechanical structure, actuators, and sensors. The goal of such analysis is the derivation of the mathematical models describing the input/output relationship characterizing the robot components. Modelling a robot manipulator is therefore a necessary premise to finding motion control strategies [4].

During time there were many specialists that tried to develop a robotics toolbox for Matlab [1, 2, 3], but, generally there are not friendly and oriented on some specific applications.

This is the reason why the authors of this paper tried to develop a set of functions for Matlab very easy to use and generally applicable in robotics. To be able to model the dynamics of a robot or manipulator it is mandatory to start with its kinematics and thus to create some functions that allows to model the links and joint of the robot structure.

This paper presents, briefly, the theory based on which the functions were developed and the corresponding Matlab functions. It also includes some results obtained for different kinematic structure of t robots, especially the SCARA robot. SCARA geometry can be realized by disposing two revolute joints and one prismatic joint in such a way that all the axes of motion are parallel. The acronym SCARA stands for *Selective Compliance Assembly Robot Arm* and characterizes the mechanical features of a structure offering high stiffness to vertical loads and compliance to horizontal loads [5]. As such, the SCARA structure is well-suited to vertical assembly tasks. The correspondence between the DOFs and Cartesian space variables is maintained only for the vertical component of a task described in Cartesian coordinates. Wrist positioning accuracy decreases as the distance of the wrist from the first joint axis increases.

The new set of functions contain a global function that allows the user to define the geometric structure of the robot, then are implemented the functions that characterize the translation or rotation joint, then the inertial functions, the Denavit-Hartenberg kinematic transformation, the differential kinematics (or inverse kinematics) and finally the dynamic functions.

Thus, in the beginning of developing a new simulation, any user of the new set of functions should implement so called *GlobalDeclaration* function and so it should declare the global variable of his robot:

```
global X Y Z U Xaxis Yaxis Zaxis ORIGIN R T M F
global Xaxis_n Yaxis_n Zaxis_n Row Col
global ma r l1 l2 M1 M2 F3
global mC1 m12 mC21 m31 mC2 m23
```

2. Kinematics functions

Kinematic analysis of the mechanical structure of a robot concerns the description of the motion with respect to a fixed reference Cartesian frame by ignoring the forces and moments that cause motion of the structure. It is meaningful to distinguish between kinematics and differential kinematics. With reference to a robot manipulator, *kinematics* describes the analytical relationship between the joint positions and the end-effector position and orientation. *Differential kinematics* describes the analytical relationship between the joint motion and the end-effector motion in terms of velocities, through the manipulator Jacobian.

The formulation of the kinematics relationship allows the study of two key problems of robotics, namely, the direct kinematics problem and the inverse kinematics problem. The former concerns the determination of a systematic, general method to describe the end-effector motion as a function of the joint motion by means of linear algebra tools. The latter concerns the inverse problem; its solution is of fundamental importance to transform the desired motion, naturally prescribed to the end-effector in the workspace, into the corresponding joint motion. The availability of a manipulator's kinematic model is also useful to determine the relationship between the forces and torques applied to the joints and the forces and moments applied to the end-effector in *static* equilibrium configurations.

Kinematics of a manipulator represents the basis of a systematic, general derivation of its *dynamics*, i.e., the equations of motion of the manipulator as a function of the forces and moments acting on it. The availability of the dynamic model is very useful for mechanical design of the structure, choice of actuators, determination of control strategies, and computer simulation of manipulator motion.

2.1. Direct kinematics functions

As it was said above the kinematics of a manipulator describes the direct relation between joint positions and end-effector position and orientation. There are many methods used to determine this dependence. The new Matlab functions developed by the authors of this paper are based on joint definition and homogenous operators. Thus there is one function called *Rotation* that builds a position matrix of a frame whose origin is initially stored in point P and rotated of angle q about axis a:

```
function  m=rotation(a,q,P)
% This function builds a position matrix
% of a frame with origin in point P and rotated
% of angle q about axis a.
%____________________________________________
GlobalDeclaration
if ( a<X | a>U )
    error('Illegal rotation axis ');
elseif P(U)~=1
    error('Illegal point "O" ');
end
if (a==U)
    R=eye(3);
else
    x=rem(a,3)+1;    y=rem(a+1,3)+1;    z=a;
    c=cos(q);      s=sin(q);
    R(x,x)= c;     R(x,y)=-s;     R(x,z)= 0;
    R(y,x)= s;     R(y,y)= c;     R(y,z)= 0;
    R(z,x)= 0;     R(z,y)= 0;     R(z,z)= 1;
end
m=R;
m(U,X:Z)=[0 0 0];
m(:,U)=P(:);
R=m(1:3,1:3);
Or=R*P(X:Z);
m(X:Z,U)=Or;
m(U,X:U)=[0 0 0 1];
```

Using this function, the user will be able to plot the workspace of his robot or kinematic chain. For example, for a SCARA robot (figure 1) with the following input data:

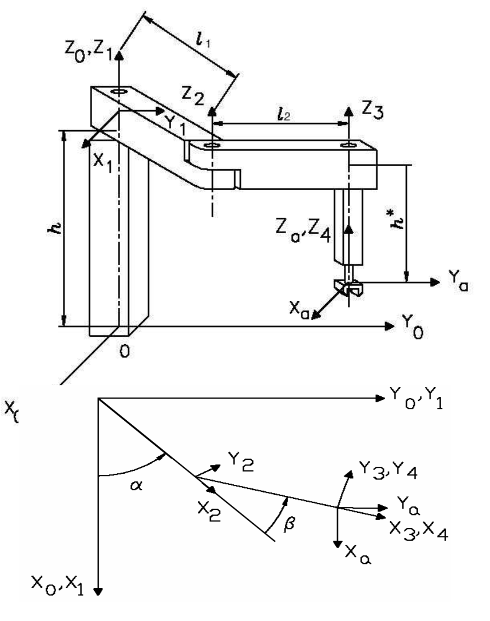


Figure 1. SCARA robot kinematic chain

$$\begin{array}{ll} q_1 = \left[-110^o,\, 110^o\right]; & q_2 = \left[-150^o,\, 150^o\right] \\ q_3 = [0,\ 0.15]; & L_1 = 0.35 \\ L_2 = 0.25; & H = 0.36 \end{array} \quad (1)$$

Thus, the workspace of the above kinematic chain is presented in figure 2.

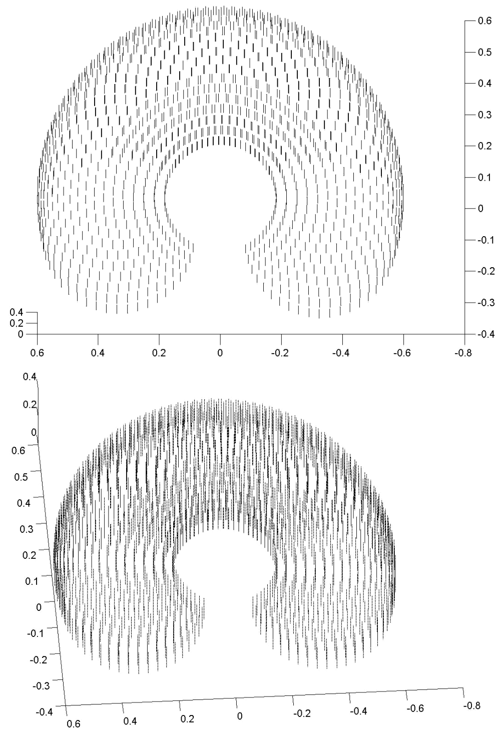


Figure 2. Workspace of SCARA robot

2.2. Inverse kinematic functions

The inverse instantaneous kinematics problem for a serial chain manipulator is: given the positions of all members of the chain and the total velocity of the end-effector, find the rates of motion of all joints.

To be able to determine these motions it has to be obtained, starting from the geometric data and kinematic chain structure the Jacobian vectors of the robot. The Jacobian constitutes a fundamental tool to characterize a manipulator, since it allows the determination of singular configurations, an analysis of redundancy and the expression of the relationship between forces and moments applied to the end-effector and the resulting joint torques at equilibrium configurations.

The general form of the Jacobian matrix, equation (2), was particularized for the two elementary movements: rotation and translation and were generated the two corresponding Matlab functions.

$$J = \begin{array}{c} \begin{array}{cccccc} Ox & Oy & Oz & \mathrm{Pr}.Ox & \mathrm{Pr}.Oy & \mathrm{Pr}.Oz \end{array} \\ \begin{bmatrix} n_x & n_y & n_z & (\overline{p}\times\overline{n}) & (\overline{p}\times\overline{n}) & (\overline{p}\times\overline{n}) \\ o_x & o_y & o_z & (\overline{p}\times\overline{o}) & (\overline{p}\times\overline{o}) & (\overline{p}\times\overline{o}) \\ a_x & a_y & a_z & (\overline{p}\times\overline{a}) & (\overline{p}\times\overline{a}) & (\overline{p}\times\overline{a}) \\ 0 & 0 & 0 & n_x & n_y & n_z \\ 0 & 0 & 0 & o_x & o_y & o_z \\ 0 & 0 & 0 & a_x & a_y & a_z \end{bmatrix} \end{array} \quad (2)$$

where $\overline{n}, \overline{o}, \overline{a}, \overline{p}$ vectors that characterize the attached reference system related to the fix one.

The corresponding Matlab functions are:

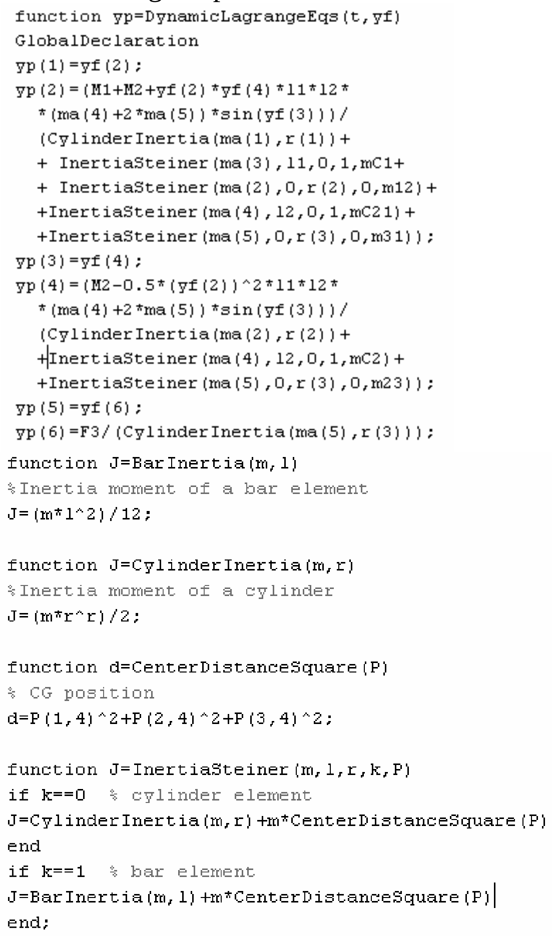
```
function m=jacobian_R(P)
% Jacobian for Rotation joint
m=zeros(6,1);
nx=P(1,1); ny=P(2,1); nz=P(3,1);
ox=P(1,2); oy=P(2,2); oz=P(3,2);
ax=P(1,3); ay=P(2,3); az=P(3,3);
px=P(1,4); py=P(2,4); pz=P(3,4);
dx=px*ny-py*nx;
dy=px*oy-py*ox;
dz=px*ay-py*ax;
deltax=nz;
deltay=oz;
deltaz=az;
m(:,1)=[dx dy dz deltax deltay deltaz]';

function m=jacobian_T(P)
% Jacobian for Prismatic joint
m=zeros(6,1);
nx=P(1,1); ny=P(2,1); nz=P(3,1);
ox=P(1,2); oy=P(2,2); oz=P(3,2);
ax=P(1,3); ay=P(2,3); az=P(3,3);
px=P(1,4); py=P(2,4); pz=P(3,4);
dx=nz;
dy=oz;
dz=az;
deltax=0;
deltay=0;
deltaz=0;
m(:,1)=[dx dy dz deltax deltay deltaz]';
```

3. Dynamics functions

Dynamics of a robot/manipulator is given by the motion equation as a function of the forces and moments acting on robot/manipulator.

There are many methods used to model mathematically the dynamic of a robot. The Matlab functions created by the author are based on Lagrange equations. To be able to write the dynamic equations, there are needed the inertia moments of the kinematic elements of the system, the position of the gravity centre and a method for solving the differential equation. Thus, at the beginning there were implemented: two functions for computing the inertia moment of a bar and of a cylinder element; a function for position of the centre of gravity and a function for Steiner inertia. In the following are presented these functions:



```
function yp=DynamicLagrangeEqs(t,yf)
GlobalDeclaration
yp(1)=yf(2);
yp(2)=(M1+M2+yf(2)*yf(4)*l1*l2*
   *(ma(4)+2*ma(5))*sin(yf(3)))/
   (CylinderInertia(ma(1),r(1))+
   + InertiaSteiner(ma(3),l1,0,1,mC1+
   + InertiaSteiner(ma(2),0,r(2),0,m12)+
   +InertiaSteiner(ma(4),l2,0,1,mC21)+
   +InertiaSteiner(ma(5),0,r(3),0,m31));
yp(3)=yf(4);
yp(4)=(M2-0.5*(yf(2))^2*l1*l2*
   *(ma(4)+2*ma(5))*sin(yf(3)))/
   (CylinderInertia(ma(2),r(2))+
   +InertiaSteiner(ma(4),l2,0,1,mC2)+
   +InertiaSteiner(ma(5),0,r(3),0,m23));
yp(5)=yf(6);
yp(6)=F3/(CylinderInertia(ma(5),r(3)));
function J=BarInertia(m,l)
%Inertia moment of a bar element
J=(m*l^2)/12;

function J=CylinderInertia(m,r)
%Inertia moment of a cylinder
J=(m*r^r)/2;

function d=CenterDistanceSquare(P)
% CG position
d=P(1,4)^2+P(2,4)^2+P(3,4)^2;

function J=InertiaSteiner(m,l,r,k,P)
if k==0  % cylinder element
J=CylinderInertia(m,r)+m*CenterDistanceSquare(P)
end
if k==1  % bar element
J=BarInertia(m,l)+m*CenterDistanceSquare(P)
end;
```

These functions were applied to SCARA robot and were potted the results of the end-effector motions, depending on the three form of the input signal: step, pulse and sinusoidal. Some of these results were presented in the below figures (when the input data was a step signal).

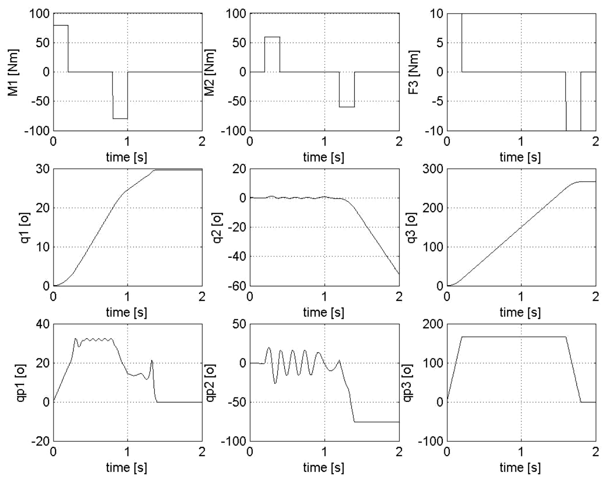


Figure 3. Torques, force and positions of the three joint of SCARA robot resulted for step input

Using direct Matlab editor it is very easy to visualise, on the same figure, all the desired simulations, just using the *plot* and *subplot* functions of Matlab. In this way it is very easy to compare and to analyse if the results of the simulations are correct.

4. Conclusions

Matlab Program was designed to be simultaneously user–friendly and powerful in tackling efficiently the most demanding problems of engineering and sciences. Besides that, almost all the input data in robotics are numerically given or determined.

These are the most important reasons why the author of the present paper decided to develop a set of functions that can be used by the mechatronics engineer and not only.

As it was said in the beginning of the paper there are some toolboxes of Matlab that resolve the problems of mechatronic systems or robotics systems, but they imply that the user is not a beginner in Matlab programming. The functions presented here and, in generally, all the set of functions developed by the authors of the present paper are easy to be applied, even for a beginner in Matlab programming.

The functions were tested for different kinematics structure and were verified using some oriented robotic program.

As future work, the authors will develop a set of functions for dynamics that includes: friction forces in joints, elasticity of joints, functions for transmissions or actuation.

References

1. Corke, P.: *MATLAB toolboxes: robotics and vision for students and teachers*. IEEE Robotics and Automation Magazine, Volume 14(4), December 2007, p. 16-17, ISSN: 1070-9932, Enschede, The Netherlands
2. Corke, P.: *Robotics Toolbox for Matlab*. Available at: http://www.petercorke.com/Robotics_Toolbox.html. Accessed: 2009–08–20
3. Kalechman, M.: *Practical Matlab applications for engineers*. CRC Press Publishing House, ISBN 978-1-4200-4776-9, USA, 2009
4. Siciliano, B., Khatib, O.: *Springer Handbook of Robotics*. Springer-Verlag Publishing House, e-ISBN: 978-3-540-30301-5, Berlin, 2008
5. Siciliano, B., Sciavicco, L., Villani, L., Oriolo, G.: *Robotics. Modelling, planning and control*. Springer-Verlag Publishing House, e-ISBN 978-1-84628-642-1, London, 2009