EEC 693 – Special Topic
Robot Modeling and Control

Homework 3

REZA SHISHEIE
2708062

Due: 3/7/2019

PROBLEM 1:

- **PART 1.1**

Starting from the initial equation:

$$116\,x_0^2 + 56\,y_0^2 + 44z_0^2 + 64x_0y_0 + 16x_0z_0 - 8y_0z_0 = 9$$

$$[x \quad y \quad z]\begin{bmatrix} a & b & c \\ b & d & e \\ c & e & f \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix} = 9$$

$$ax_0^2 + d\,y_0^2 + ez_0^2 + 2bx_0y_0 + 2cx_0z_0 + 2ey_0z_0 = 9$$

Comparing the two equations:

$$a = 116$$

$$b = \frac{64}{2} = 32$$

$$c = \frac{16}{2} = 8$$

$$d = 56$$

$$e = -\frac{8}{2} = -4$$

$$f = 44$$

Thus, Q would be:

$$Q = \begin{bmatrix} 116 & 32 & 8 \\ 32 & 56 & -4 \\ 8 & -4 & 44 \end{bmatrix}/9$$

$$P = Q^{-\frac{1}{2}}$$

P would be calculated from svd function in MATLAB. It gives matrix U and S. U columns represent the principal coordinates of the new coordinate system. Each element on the diagonal of S represents length of each principal axis.

$$\sigma = S = \begin{bmatrix} \sigma_x & 0 & 0 \\ 0 & \sigma_y & 0 \\ 0 & 0 & \sigma_z \end{bmatrix}$$

```
%% PART 1:Find the principal axis lengths
% extracting Q from the given elipsoid
fprintf("\n---PROBLEM1 - PART 1---\n")

Q = [116 32 8;
     32 56 -4;
     8 -4 44]/9;
```

```
% initializing P and getting U, S and V
P = Q^(-1/2);
[U,S,V] = svd(P)
sigma = [S(1,1), S(2,2), S(3,3)]
% sigma represent length of each principal axis X,Y,Z
```

```
---PROBLEM1 - PART 1---
U =
   -0.3333   -0.2166   -0.9176
    0.6667    0.6340   -0.3919
    0.6667   -0.7423   -0.0669
S =
    0.5000         0         0
         0    0.4253         0
         0         0    0.2629
V =
   -0.3333   -0.2166   -0.9176
    0.6667    0.6340   -0.3919
    0.6667   -0.7423   -0.0669
sigma =
    0.5000    0.4253    0.2629
```

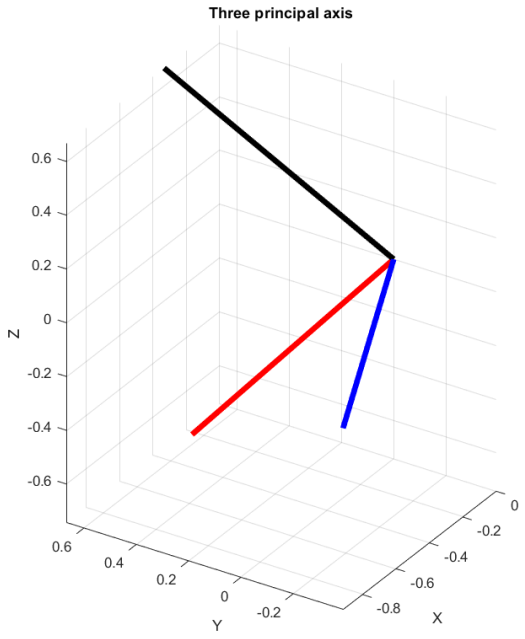
- **PART 1.2**

```
%% PART 2:
% Find a new coordinate frame 1 where the equation of the ellipsoid appears
% in the canonical form.
% Give the rotation matrix between frames 0 and 1 and the Euler angles
% defining the orientation of the ellipsoid in frame 0. Use this information
% to make a hand sketch of the ellipsoid.
fprintf("\n---PROBLEM1 - PART 2---\n")

% defining old and new coordinate frames
x0 = [1,0,0];
y0 = [0,1,0];
z0 = [0,0,1];

x1 = U(:,1);
y1 = U(:,2);
z1 = U(:,3);

% defining rotation matrix
R01 = [dot(x1,x0)   dot(y1,x0)  dot(z1,x0);
       dot(x1,y0)   dot(y1,y0)  dot(z1,y0);
       dot(x1,z0)   dot(y1,z0)  dot(z1,z0)];

% extracting euler angles in degrees
euler_angles_degrees = tr2eul(R01)*180/pi

% getting equation of elipsizin new coodinate frame
syms w1 w2 w3
X = U*[w1;w2;w3];
svm_result = vpa(simplify(X.'*Q*X),3)
```

```
%checking the coefficients of x1^2 ,y1^2 , and z1^2. They should match with
%the results of vpa above
W = [1/sigma(1)^2 1/sigma(2)^2 1/sigma(3)^2];
fprintf("direct_results=\n%.2fw1^2 + %.2fw2^2 + %.2fw3^2 = 1\n", W(1), W(2),
W(3))

figure
plot3([0 U(1,1)], [0 U(2,1)], [0 U(3,1)],'k','LineWidth',4)
hold on
plot3([0 U(1,2)], [0 U(2,2)], [0 U(3,2)],'r','LineWidth',4)
hold on
plot3([0 U(1,3)], [0 U(2,3)], [0 U(3,3)],'b','LineWidth',4)
grid on
axis equal
title("Three principal axis")
xlabel("X")
ylabel("Y")
zlabel("Z")
```

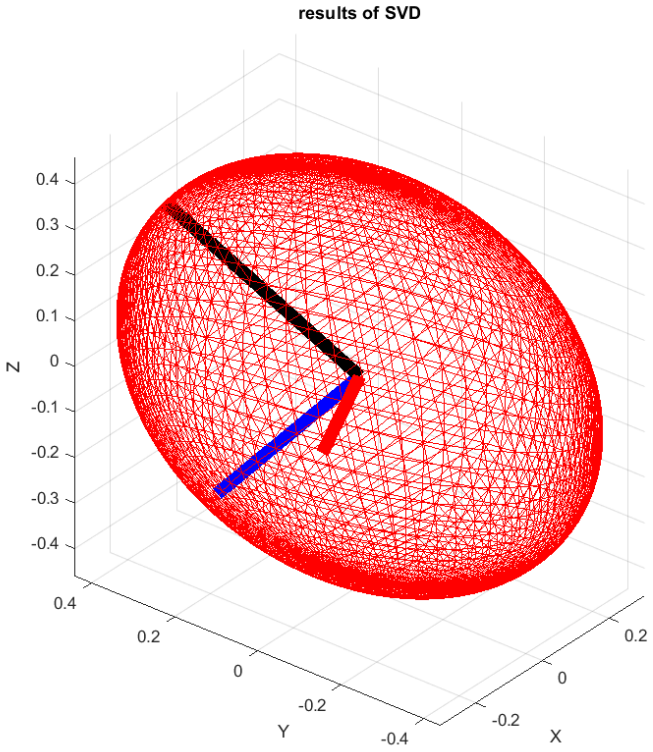
```
---PROBLEM1 - PART 2---
euler_angles_degrees =
 -156.8750   93.8381 -131.9256
svm_result =
4.0*w1^2 + 1.05e-14*w1*w2 - 1.11e-15*w1*w3 + 5.53*w2^2 + 5.97e-15*w2*w3 +
14.5*w3^2
direct_results=
4.00w1^2 + 5.53w2^2 + 14.47w3^2 = 1
```

As you see results of canonical form from svd is the same as the direct method. Coefficients of xy, xz, and yz are almost close to zero which is the result of computation error. Here is the plot of principal axis of the new coordinate system based on single value decomposition:



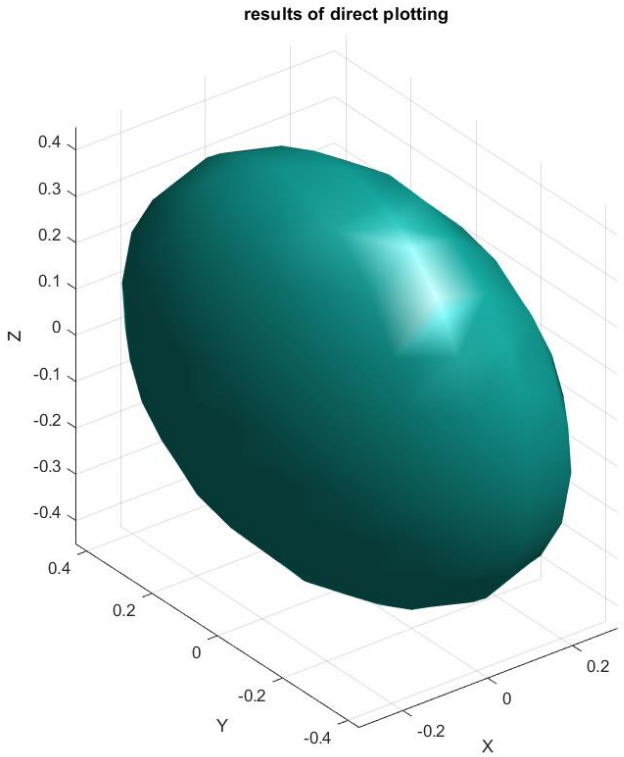
- **PART 1.3**

```
%% PART 3:
% Download the Matlab ellipsoidal toolbox by Alex Kurzhanskiy, available
% through the Matlab File Exchange. Use it to obtain a 3D plot of the
% ellipsoid.
fprintf("\n---PROBLEM1 - PART 3---\n")

% sketching elipsid using Matlab ellipsoidal toolbox by Alex Kurzhanskiy
figure
E = ellipsoid(inv(Q));
plot3(E)
hold on
plot3([0 U(1,1)*S(1,1)],[0 U(2,1)*S(1,1)],[0
U(3,1)*S(1,1)],'k','LineWidth',8)
hold on
plot3([0 U(1,2)*S(2,2)],[0 U(2,2)*S(2,2)],[0
U(3,2)*S(2,2)],'b','LineWidth',8)
hold on
plot3([0 U(1,3)*S(3,3)],[0 U(2,3)*S(3,3)],[0
U(3,3)*S(3,3)],'r','LineWidth',8)
grid on
axis equal
title("results of SVD")
xlabel("X")
ylabel("Y")
zlabel("Z")
```



- **PART 1.4**

```
%% PART 4:
% Verify your calculations by plotting the three principal axis as obtained
% from the U and Îf matrix of the singular value decomposition, overlaying
% the plot of the ellipsoid. Show one 3D view and three plan views where
% the axes can be easily compared to the ellipsoidâ€™s boundary.
fprintf("\n---PROBLEM1 - PART 4---\n")

% checking results with the actual plot
figure
[x,y,z]= meshgrid(-2:0.1:2);
v= 116*x.^2 + 56*y.^2 + 44*z.^2 + 64*x.*y + 16*x.*z- 8*y.*z - 9;
isosurface(x,y,z,v,0);
%axis tight
axis equal
grid on
title("results of direct plotting")
xlabel("X")
ylabel("Y")
zlabel("Z")
```



As you can see both plots – one from direct plotting and one from results of svd function – are identical.

PROBLEM 2:

- **PART 2.1**

```
%% PART 1:
fprintf("\n---PROBLEM2 - PART 1---\n")
% Obtain all transformations necessary to compute the Jacobian. As a
verification
% reference, set q 1 = 1, q 2 = â^'1, q 3 = Ï€/2, q 4 = Ï€/2, q 5 = 0.

syms q1 q2 q3 q4 q5

H10 = Rotz(pi/2)*Transz(q1)*Transx(0)*Rotx(pi/2);
H21 = Rotz(-pi/2)*Transz(q2)*Transx(0)*Rotx(-pi/2);
H32 = Rotz(q3)*Transz(1)*Transx(0)*Rotx(-pi/2);
H43 = Rotz(q4)*Transz(0)*Transx(0)*Rotx(pi/2);
H54 = Rotz(q5)*Transz(0)*Transx(-1)*Rotx(0);

H20 = H10*H21;
H30 = H20*H32;
H40 = H30*H43;
H50 = H40*H54;

% Computing H50 with initial conditions
q1 = 1; q2 = -1; q3 = pi/2; q4 = pi/2; q5 = 0;
H_eval_check = eval(H50)
```

```
---PROBLEM2 - PART 1---
H_eval_check =
   -0.0000   -0.0000   -1.0000   -1.0000
   -1.0000         0    0.0000    2.0000
   -0.0000    1.0000   -0.0000    1.0000
         0         0         0    1.0000
```

H50 evaluation matches with the one provided by homework assignment.

- **PART 2.2**

```
%% PART 2: Finding the velocity Jacobian Jv using symbolic processing.
fprintf("\n---PROBLEM2 - PART 2---\n")

% finding center of each coordinate system
o1 = H10(1:3,4);
o2 = H20(1:3,4);
o3 = H30(1:3,4);
o4 = H40(1:3,4);
o5 = H50(1:3,4);

% finding zeros
z0 = [0,0,1];
z1 = H10(1:3,3);
z2 = H20(1:3,3);
z3 = H30(1:3,3);
z4 = H40(1:3,3);
z5 = H50(1:3,3);
```

```
% getting jacobian
Jv1 = z0';
Jv2 = z1;
Jv3 = cross(z2,(o5-o2));
Jv4 = cross(z3,(o5-o3));
Jv5 = cross(z4,(o5-o4));
Jw1 = [0;0;0];
Jw2 = [0;0;0];
Jw3 = z2;
Jw4 = z3;
Jw5 = z4;

Jv = [Jv1, Jv2, Jv3, Jv4, Jv5];
Jw = [Jw1, Jw2, Jw3, Jw4, Jw5];
J = [Jv; Jw]
```

Actual Jacobian can be calculated by running the code. It is too long to be printed here.

- **PART 2.3**

```
%% PART 3:
% As a cautionary note, try the rank command on the symbolic J v , with
% unspecified values for q. Then evaluate J v using the test values of q
above,
% entering sym(pi) for Ï€. Comment on the results.
fprintf("\n---PROBLEM2 - PART 3---\n")

% here is the rank of Symbolic Jv. It is 3 which means it is full rank as
% far as linear velocity
rank_jv_sym = rank(Jv)

% Here is rank of Jv if it is initialized. Rank is 2 which means that it has
% lost one degree of freedom in Jacobian and linear velocity is limited in
% one direction.
q1 = 1; q2 = -1; q3 = pi/2; q4 = pi/2; q5 = 0;
rank_jv_eval = rank(eval(Jv))
```

```
---PROBLEM2 - PART 3---
rank_jv_sym =
     3
rank_jv_eval =
     2
```

As you see rank of symbolic Jacobian is 3 which is full rank but in a particular orientation it is 2 which means robot can't provide velocity in one of the directions.

- **PART 2.4**

```
%% PART 4: Based on the above, can J v be singular at certain q values? For
%  example which ones?

% Jv can be singular at certain qs. One of them is the one mentioned above.
```

- **PART 2.5**

```
%% PART 5: Select a q that produces singularity and provide a
physical/geometric
%  explanation of how this occurs.
fprintf("\n---PROBLEM2 - PART 5---\n")

% To have an idea what the physical position of the end effector is lets
% find p once all angles are zero and q1=q2=1
q1 = 1;
q2 = 1;
q3 = 0;
q4 = 0;
q5 = 0;
p = eval(H50*[0 0 0 1]');
p_nonsing = p(1:3,1)'
rank_jv_nonsing_eval = rank(eval(Jv))
% p yields [1 1 2] which is an upright position
% this is also a full rank situation

q1 = 1;
q2 = 1;
q3 = 0;
q4 = pi/2;
q5 = 0;
p = eval(H50*[0 0 0 1]');
p_sing = p(1:3,1)'
rank_jv_sing_eval = rank(eval(Jv))
% in thhis orientation p yields [1 2 1]. Rank of Jacobian in this position
% is 2 which is not full rank. q1 and q2 can generate any planar motion in
% the xz plane. Singularity happens when the rod p is in such an orientation
% that the motion of q3 q4 a5 only generate velocity in direction of q1 and
% q2. that is an example of a situation when singularity happens. In this
% example q4 turns 90 degrees such that the rod p is pointing up toward y0
% and in this orientation, it can only provide velocity in the xz plane and
% not in y0 direction. This would be a singularity situation where one
% degree of freedom in velocity is lost.
```

```
---PROBLEM2 - PART 5---
p_nonsing =
    1.0000    1.0000    2.0000
rank_jv_nonsing_eval =
     3
p_sing =
    1.0000    2.0000    1.0000
rank_jv_sing_eval =
     2
```

As you see, rank of the robot in the first orientation is 3 which means it is full rank, However, rank in the second orientation is 2 which is not full rank. Please read comments in the code for explanation.

- **PART 2.6**

```
%% PART 6: Compute Yoshikawaâ€™s manipulability measure Î¼(q) using symbolic
%  processing. Provide a mathematical and/or physical explanation for the
result.
fprintf("\n---PROBLEM2 - PART 6---\n")

% mu_sym = sqrt(det(J.'*J));

% non singular mu
q1 = 1;
q2 = 1;
q3 = 0;
q4 = 0;
q5 = 0;
J_eval = eval(Jv);
mu_nonsing = sqrt(det(J_eval*J_eval.'))
rank_Jv_nosing = rank(J_eval)

% singular mu
q1 = 1;
q2 = 1;
q3 = 0;
q4 = pi/2;
q5 = 0;
J_eval = eval(Jv);
mu_sing = sqrt(det(J_eval*J_eval.'))
rank_Jv_sing = rank(J_eval)
% as you see the determinant of Yoshikawa manipulability is 0 for singular
% orientation and non-zero for non-singular orientation. The singular and
% non-singular orientation was were already mentioned above
```

```
---PROBLEM2 - PART 6---
mu_nonsing =
    1.7321
rank_Jv_nosing =
     3
mu_sing =
   2.4493e-16
rank_Jv_sing =
     2
```

The first orientation is non singular which has rank of 3 and Yoshikawa manipulability of 1.73 which is not 0. This means Jacobian is not singular. The second one; however, is rank 2 and has Yoshikawa manipulability of 0 and rank of 2. This one is singular.

- **PART 2.7**

For this section some qdots are set and them linear velocity Jacobian is computed. If Jacobian is singular no initial value for qdots can make the orientation have velocity for all axis.

```
%% PART 7: For the above test values of q, describe the set of all joint
velocities that
%  result in an instantaneous zero velocity for P . Give an example of such
%  a joint velocity vector (other than the zero vector) and verify it with a
%  forward calculation.
 fprintf("\n---PROBLEM2 - PART 7---\n")

% lets compute for a non singular situation.
q1 = 1;
q2 = 1;
q3 = 0;
q4 = 0;
q5 = 0;
q1dot = 1;
q2dot = 2;
q3dot = 3;
q4dot = 4;
q5dot = 5;
p = eval(H50*[0 0 0 1]');
p_nonsing = p(1:3,1)'
Jv_eval = eval(Jv);
% forward kinematics
xdot_nonsing = (Jv_eval*[q1dot, q2dot, q3dot, q4dot, q5dot]')'
% as you see all values of xdot is non zero

% lets compute for a singular situation now
q1 = 1;
q2 = 1;
q3 = 0;
q4 = -pi/2;
q5 = 0;
q1dot = 1;
q2dot = 2;
q3dot = 3;
q4dot = 4;
q5dot = 5;
p = eval(H50*[0 0 0 1]');
p_sing = p(1:3,1)'
Jv_eval = eval(Jv);
% forward kinematics
xdot_sing = (Jv_eval*[q1dot, q2dot, q3dot, q4dot, q5dot]')'
% The orientation of rod is pointing down and thus the location of p is at
% [1,0,1]. In this orientation rod is in direction of y and our expectation
% is that the velocity in y direction should be zero. As you see all values
% of xdot is non zero except the y element as we predicted
```

```
---PROBLEM2 - PART 7---
p_nonsing =
    1.0000    1.0000    2.0000
xdot_nonsing =
   10.0000    4.0000    1.0000
p_sing =
```

```
    1.0000         0    1.0000
xdot_sing =
    7.0000   -0.0000    5.0000
```

- **PART 2.8-9**

```
%% PART 8-9:
% â€¢ Calculate the angular velocity Jacobian J w and exclude the first two
% columns. This is equivalent to considering the wrist only, and only ori-
% entation kinematics.
% â€¢ For the above reduced angular velocity Jacobian, compute Yoshikawaâ€™s
% manipulability measure in symbolic form. It predicts singularity for cer-
% tain values of q 4 . Interpret geometrically.
fprintf("\n---PROBLEM2 - PART 8-9---\n")

Jw_wrist = Jw(:,3:5);
q4 = 0;
det_sing_0 = det(eval(Jw_wrist))

Jw_wrist = Jw(:,3:5);
q4 = pi;
det_sing_pi = det(eval(Jw_wrist))

% determinant is zero which implies singularity when
% q4 = n*pi , n = 0,1,2,3,...
% this is called gimbal lock and this phenomena happens when two rotational
% axis are aligned to each other. When q4 = 0 or pi q3 and q5 are aligned
% and thus one degree of freedom is lost.
```

```
---PROBLEM2 - PART 8-9---
det_sing_0 =
   1.5095e-48
det_sing_pi =
  -1.2246e-16
```