Robot manipulability ellipsoid (SVD): Matlab example, redundant manipulator

© 2024, Antonio Sala Piqueras, Universitat Politecnica de Valencia. All rights reserved.

Presentations in video:

http://personales.upv.es/asala/YT/V/elipm1EN.html , http://personales.upv.es/asala/YT/V/elipm2EN.html , http://personales.upv.es/asala/YT/V/elipm3EN.html .

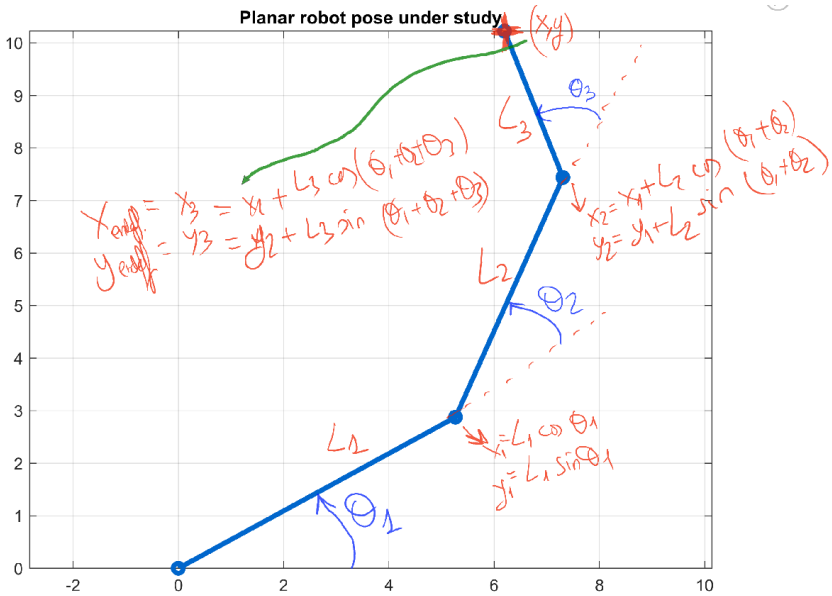
*The code ran in Matlab **R2022b** (Linux)

Objectives: illustrate the manipulability ellipsoid in a robotic manipulator. Planar and no gravity for simplicity.

Table of Contents

Modelling

```
syms theta1 theta2 theta3 real
q=[theta1; theta2; theta3];
```




```
L1=3; L2=2.4; L3=1.5;  %Constant parameters
```

```
%Model equations (no dynamics):
x1=L1*cos(theta1);
y1=L1*sin(theta1);
x2=x1+L2*cos(theta1+theta2);
y2=y1+L2*sin(theta1+theta2);
x3=x2+L3*cos(theta1+theta2+theta3);
y3=y2+L3*sin(theta1+theta2+theta3);
```

We'll lump together the model that gives all articulation positions from angles as:

```
allartics=[x1; y1; x2; y2; x3; y3]
```

```
allartics =
```

$$\begin{pmatrix} 3\cos(\theta_1) \\ 3\sin(\theta_1) \\ \sigma_2 + 3\cos(\theta_1) \\ \sigma_1 + 3\sin(\theta_1) \\ \frac{3\cos(\theta_1+\theta_2+\theta_3)}{2} + \sigma_2 + 3\cos(\theta_1) \\ \frac{3\sin(\theta_1+\theta_2+\theta_3)}{2} + \sigma_1 + 3\sin(\theta_1) \end{pmatrix}$$

where

$$\sigma_1 = \frac{12\sin(\theta_1+\theta_2)}{5}$$

$$\sigma_2 = \frac{12\cos(\theta_1+\theta_2)}{5}$$

```
allnum=matlabFunction([0 0; x1 y1; x2 y2; x3 y3]) %Compiled, for later simulations
```

```
allnum = function_handle with value:
    @(theta1,theta2,theta3)reshape([0.0,cos(theta1).*3.0,cos(theta1+theta2).*(1.2e+1./5.0)+cos(theta1).*3.
```

```
r=[x3;y3]; %end effector position
```

Let us analyse a given configuration (pose) point:

```
%th1_0=.6;th2_0=1.75;th3_0=1.98;% well-conditioned ellipse
th1_0=.5;th2_0=0.65;th3_0=0.8;% medium-conditioned ellipse
%th1_0=1.25;th2_0=.27;th3_0=-.3;% ILL-CONDITIONED ellipse
%th1_0=1.0;th2_0=.0002;th3_0=-0.0002;% singularity (almost)
```

Cartesian positions of articulations at that pose are:

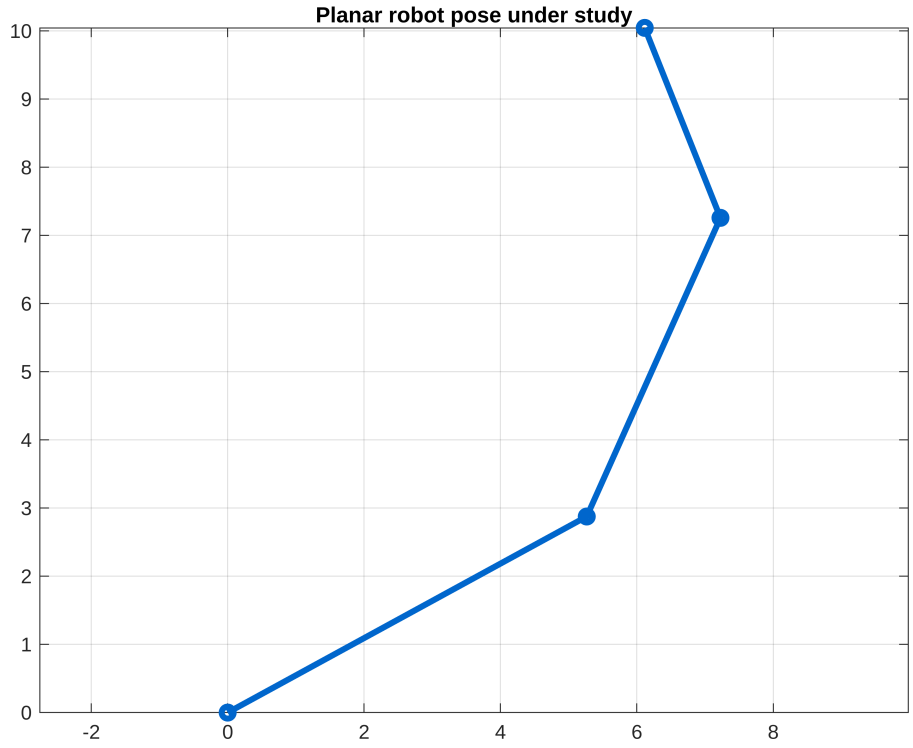
```
all_artic_pos=allnum(th1_0,th2_0,th3_0) %positions
```

```
all_artic_pos = 4×2
```

```
         0         0
    2.6327    1.4383
    3.6131    3.6289
    3.0578    5.0223
```

Let us plot the pose:

```
allop2=all_artic_pos*2; %we scale the robot twice for pictures
plot(allop2(:,1), allop2(:,2),'-o', LineWidth=3,Color=[0 .4 .8])
grid on, axis equal, title("Planar robot pose under study") %plots the robot
```



Manipulability analysis

Chain rule in $x = f(q(t))$ if x and q were single real variables would be written as $\frac{dx}{dt} = \frac{df}{dq} \cdot \frac{dq}{dt}$. But, if q is a vector of parameters, we need to change to partial derivatives, etc. as follows (multivariable chain rule):

From $x = f(q(t))$, $\frac{dx}{dt} = \sum_1^3 \frac{\partial f}{\partial q_i} \cdot \frac{dq_i}{dt} = \frac{\partial f}{\partial q}_{1\times 3} \cdot \frac{dq}{dt}_{3\times 1}$ and, likewise, an analogous expression for y, we will compute end effector speed as follows.

End effector velocity

```
Jac_r=simplify(jacobian(r,q)); %symbolic robot Jacobian
```

$$v_{end} = \begin{pmatrix} \dot{x}_{end} \\ \dot{y}_{end} \end{pmatrix} = J(q)_{2\times 3} \cdot \dot{q}_{3\times 1} = \frac{\partial r}{\partial q}\dot{q}$$

```
J_r_num=matlabFunction(Jac_r); %Jacobian as a Matlab numeric function
Jop=J_r_num(th1_0,th2_0,th3_0) %Jacobian at the chosen pose (numeric)
```

```
Jop = 2×3
   -5.0223   -3.5841   -1.3934
    3.0578    0.4251   -0.5553
```

Scaling: We need to scale so that maximum angular speed is "1" in scaled units, in order to allow for different joint characteristics.

So, $\dot{q} := E_q \dot{q}_n$

```
Eq=diag([1 1 1.5]);
```

SVD and its interpretation:

```
ScaledJacobian=Jop*Eq;
[U,S,V]=svd(ScaledJacobian)
```

```
U = 2×2
    0.9297    0.3684
   -0.3684    0.9297
S = 2×3
    6.9597         0         0
         0    2.0557         0
V = 3×3
   -0.8327    0.4830    0.2708
   -0.5013   -0.4500   -0.7391
   -0.2351   -0.7512    0.6168
```

```
cond(ScaledJacobian)
```

```
ans = 3.3855
```

Principal maneuvers: U has dimensions of "cartesian end-effector speed", S has dimensions of "speed gain" (gear ratio, sort of), and V has dimensions of "scaled angular joint speed ($\dot{q}_n$)". So, SVD describes the Jacobian as two "independent" gears.

Inverse of speed gain will be later on related to "force ellipsoids".

All joint Jacobian

```
Jnumall=matlabFunction(jacobian(allartics,q)); %all articulation positions jacobian
```

```
Jopall=Jnumall(th1_0,th2_0,th3_0)*Eq %Jacobian of all joints (scaled)
```

```
Jopall = 6×3
   -1.4383         0         0
    2.6327         0         0
   -3.6289   -2.1906         0
    3.6131    0.9804         0
   -5.0223   -3.5841   -2.0902
    3.0578    0.4251   -0.8329
```

```
ttt=Jopall*V %speed xy at all joint ends in principal end-effector maneouvers
```

```
ttt = 6×3
    1.1977   -0.6946   -0.3894
   -2.1924    1.2715    0.7128
    4.1200   -0.7669    0.6366
   -3.5002    1.3038    0.2537
    6.4703    0.7572    0.0000
   -2.5636    1.9112   -0.0000
```

"ttt" has as output cartesian articulation speed in principal maneuvers, input "principal maneuvres scaled angular speed". Note the [0;0] at the right-most column, bottom 2 rows: nullspace of end-effector maneuvre.

Graphical representation: manipulability ellipsoid

The ellipse, centered at the origin, image of the unit circle in $\dot{q}$ with the map $v = J\dot{q}$, $\|\dot{q}\| \leq 1$, with full row rank J, is $v^T(JJ^T)^{-1}v \leq 1$.

Indeed, minimum joint speed $\dot{q}$ to achieve v is $\widetilde{\dot{q}} = pinv(J) \cdot v = J^T(JJ^T)^{-1}v$; hence $\widetilde{\dot{q}}^T\widetilde{\dot{q}} = v^T(JJ^T)^{-1}J \cdot J^T(JJ^T)^{-1}v = v^T(JJ^T)^{-1}v \leq 1$, so these are the end-effector speeds achievable with unit joint speed norm. If $J = USV^T$ then $JJ^T = USV^T \cdot VSU^T = USS^TU^T = UDU^T$: square root of eigenvalues of JJ^T are the singular values of J; the eigenvectors of JJ^T are the columns of U.

```
rop=allop2(4,:)'; %center of the plotted ellipse: end effector
plot(allop2(:,1), allop2(:,2),'-o', LineWidth=3,Color=[0.2 .5 .9]), grid on %plots the
hold on
syms v [2 1] real;
```

We change JJ^T by the scaled version $J_{scaled}J_{scaled}^T = JE_qE_q^TJ^T$ to get the end-effector speeds achievable with unit "*scaled*" joint speed norm.

```
M1=(Jop*Eq*Eq'*Jop')
```

```
M1 = 2×2
   42.4383  -15.1402
  -15.1402   10.2249
```

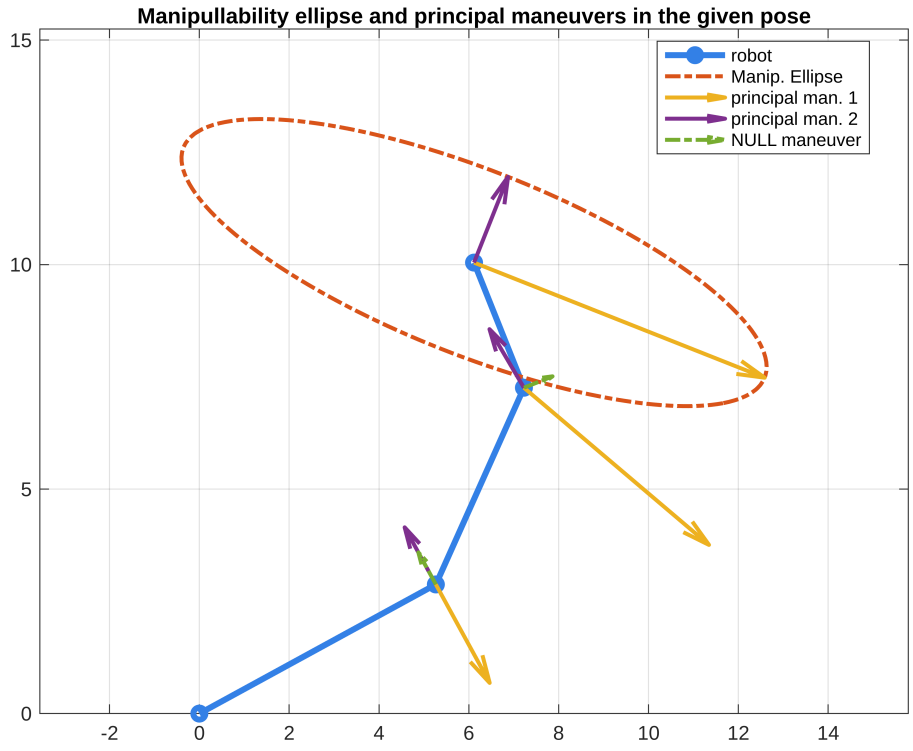
```
eM=eig(M1)'
```

```
eM = 1×2
    4.2261   48.4371
```

```
sqrt(eM) %these are the Singular Values of the Jacobian (scaled)
```

```
ans = 1×2
    2.0557    6.9597
```

```
fimplicit((v-rop)'*inv(M1)*(v-rop)-1,'-.',LineWidth=2) %we draw it centered at the end-
quiver(allop2(2:end,1),allop2(2:end,2),ttt(1:2:end,1),ttt(2:2:end,1),0,LineWidth=2)
quiver(allop2(2:end,1),allop2(2:end,2),ttt(1:2:end,2),ttt(2:2:end,2),0,LineWidth=2)
quiver(allop2(2:end,1),allop2(2:end,2),ttt(1:2:end,3),ttt(2:2:end,3),0,'-.',LineWidth=2
hold off, axis([-6 11 0 15.25]),
axis equal
legend("robot","Manip. Ellipse","principal man. 1","principal man. 2","NULL maneuver",I
title("Manipullability ellipse and principal maneuvers in the given pose")
```



Note that arrows and ellipse have units of "cartesian speed".

Conservatism of ellipsoid vs polyhedron geometry

If the "scaled" normalised to 1 joint speed was limited by 1 in magnitude, not in "Euclidean norm", the actually achievable end-effector speeds would be:

```
dotqnVert=[1 1 1;1 1 -1;1 -1 1;1 -1 -1;-1 1 1;-1 1 -1;-1 -1 1;-1 -1 -1]';
dotqVert=Eq*dotqnVert
```

```
dotqVert = 3×8
    1.0000    1.0000    1.0000    1.0000   -1.0000   -1.0000   -1.0000   -1.0000
    1.0000    1.0000   -1.0000   -1.0000    1.0000    1.0000   -1.0000   -1.0000
    1.5000   -1.5000    1.5000   -1.5000    1.5000   -1.5000    1.5000   -1.5000
```

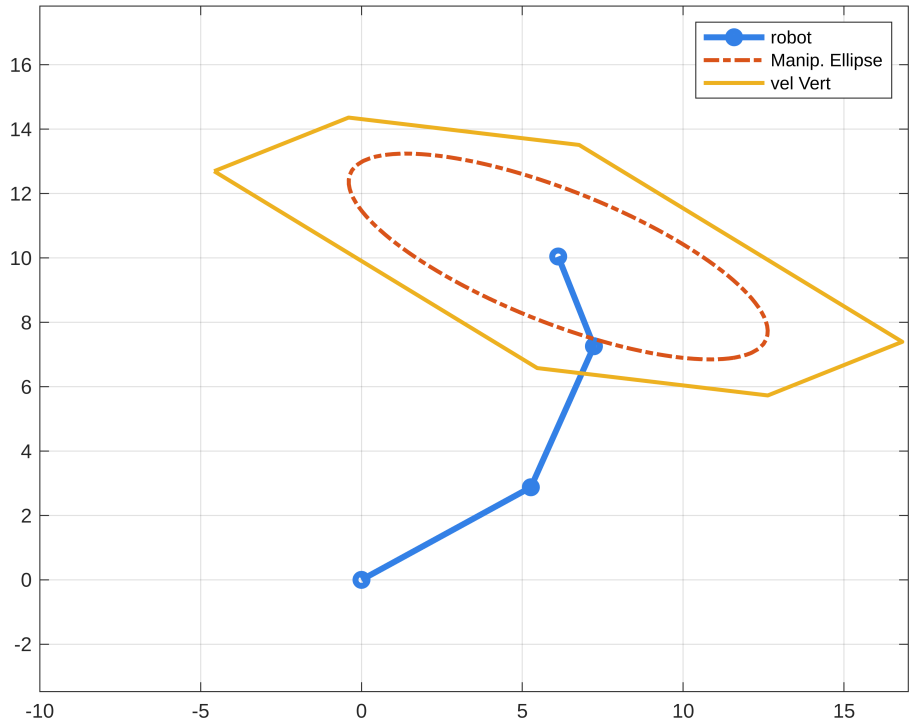
```
VelEndEffVert=Jop*dotqVert
```

```
VelEndEffVert = 2×8
  -10.6966   -6.5163   -3.5284    0.6519   -0.6519    3.5284    6.5163   10.6966
    2.6500    4.3159    1.7998    3.4657   -3.4657   -1.7998   -4.3159   -2.6500
```

```
plot(allop2(:,1), allop2(:,2),'-o', LineWidth=3,Color=[0.2 .5 .9]), grid on %plots the
hold on
fimplicit((v-rop)'*inv(M1)*(v-rop)-1,'-.',LineWidth=2) %we draw it centered at the end-

k=convhull(VelEndEffVert(1,:),VelEndEffVert(2,:));
plot(VelEndEffVert(1,k)+rop(1),VelEndEffVert(2,k)+rop(2),LineWIdth=2)

hold off,  axis([-10 17 0 16.5]), axis equal
legend("robot","Manip. Ellipse","vel Vert")
```



However, the ellipse gives us an idea of the "shape" of the polyhedron, it is easy to calculate, and it is the result that we would get if we used "pseudoinverse of J" to calculate joint velocity from "end effector velocity reference", which might be the case in closed-loop control.