```
%% Animation of principal
maneuvres of a planar robot
% (c) 2024, Antonio Sala Piqueras.
Universitat Politecnica de
Valencia, Spain. All rights
reserved.

% Video presentation:
% English: http://personales.upv.
es/asala/YT/V/elipm3EN.html
% Spanish: http://personales.upv.
es/asala/YT/V/elipm3.html


%run("ElipsRobot1Es.mlx")% si
seguiste la versión en español.
run("ElipsRobot1ENG.mlx")% if you
followed English version
ManeuverIDX=3; %which SVD maneuver
are we exploiting.

th0=[th1_0;th2_0;th3_0];
Ts=0.015;
```

```
Trange=0:Ts:15;
Nsamples=length(Trange);
thplot=zeros(3,Nsamples);
q=th0;
dotq_old=[1;0;0];
Ellip_pl=cell(2,Nsamples);
for i=1:Nsamples
    thplot(:,i)=q;
    JacobianEndEff=J_r_num(q(1),q↙
(2),q(3));
↙
JacobianEndEffScaled=JacobianEndEf↙
f*Eq;
    Ellip_pl{1,i}=inv↙
(JacobianEndEffScaled*JacobianEndE↙
ffScaled');
    [U,S,V]=svd↙
(JacobianEndEffScaled);

    %Sign may change, so we'll↙
ensure maneuver doesn't go back-↙
and-forth
    % we'll choose the sign of↙
```

```
manouver closest to the previous↙
one...
    e1=norm(Eq*V(:,ManeuverIDX)-↙
dotq_old);
    e2=norm(-Eq*V(:,ManeuverIDX)-↙
dotq_old);
    if(e2<e1)
        V=-V;
        U=-U;
    end

    if ManeuverIDX<3
        Ellip_pl{2,i}=U(:,↙
ManeuverIDX)*S(ManeuverIDX,↙
ManeuverIDX);
    else
        Ellip_pl{2,i}=[0;0];
    end
    dotq=Eq*V(:,ManeuverIDX); %↙
redo scaling
    if(ManeuverIDX<3)
        dotq=dotq/2; %smaller↙
movements
```

```
    end
    dotq_old=dotq;
    q=q+dotq*Ts; %Accumulate↙
angular speed to angular position
end
%for debug purposes, not too↙
"beautiful"
%figure(1)
%plot(Trange,thplot'), grid on
%figure(2)
%plot(Trange(1:(end-1)),diff↙
(thplot')), grid on


%% animation
figure(3)
for i=1:2:Nsamples

    all_artic_pos=allnum(thplot(1,↙
i),thplot(2,i),thplot(3,i)); %↙
positions all artics.
    plot(all_artic_pos(:,1),↙
all_artic_pos(:,2),'-o',↙
```

```
LineWidth=4,Color=[0 .4 .8])
    hold on
    fimplicit((v-all_artic_pos↙
(end,:)')'*Ellip_pl{1,i}*(v-↙
all_artic_pos(end,:)')-0.25,'-.',↙
LineWidth=3) %scaled down
    myarrow=Ellip_pl{2,i}/2;
    quiver(all_artic_pos(end,1),↙
all_artic_pos(end,2),myarrow(1),↙
myarrow(2),LineWidth=3.5,↙
AutoScale="off")
    hold off
    axis equal, grid on
    axis([-5 7 -6 8])
    drawnow
    if(i==1)
        pause
    end
end
```