Assignment 5

Vinay K.
2016MEZ8353

May 10, 2017

Ans 1: Leg lengths for Initial platform pose are listed in Table 1

Table 1: Leg Lengths for initial platform configuration

Leg Lengths	mm
L_1	304.1381
L_2	304.0954
L_3	304.0954
L_4	304.1381
L_5	304.0954
L_6	304.0954

Ans 2: The expression for Force Jacobian is derived by force balance equation along the end effector and is given by

$$[J_F] * [\tau] = F$$

where τ is the vector of actuation forces and F is the wrench acting on the end effector. Force Jacobian consists of screws corresponding to each leg stacked vertically, and for initial platform pose is given by:

$$J_F = \begin{bmatrix} -0.1644 & -0.0822 & 0.1414 & 0.1644 & 0.0822 & -0.1414 \\ 0 & -0.1414 & -0.0822 & 0 & 0.1414 & 0.0822 \\ 0.9864 & 0.9865 & 0.9865 & 0.9864 & 0.9865 & 0.9865 \\ 0 & 85.8283 & 49.3266 & 0 & -85.8283 & -49.3266 \\ -98.6394 & -49.3266 & 85.8283 & 98.6394 & 49.3266 & -85.8283 \\ 0 & 0.0822 & 0.0822 & 0 & 0.0822 & 0.0822 \end{bmatrix}$$

Kinematic Jacobian is given by expressions

$$J_K * \dot{L} = \dot{X}$$

$$J_K = {J_F}^{T^{-1}}$$

where $\dot{L}$ is the vector of actuation velocities and $\dot{X}$ is end effector twist. For the initial configuration of platform, kinematic Jacobian do not exists, as the Force Jacobian in singular.

Ans 3: As kinematic Jacobian do not exists, its determinant cannot be computed, however the determinant of its inverse is zero i.e.

$$det({J_F}^T) = 0$$

.

Ans 4: Initial platform pose given in the problem is singular, and remove this singularity a slightly perturbed platform pose was used. Platform pose and leg lengths obtained from **newton raphson** (tolerance 1e-4) for 0%, 1%, 2%, 3%, 4% & 5% increase in leg lengths (leg2, leg3 & leg6)are given in Tables 2-7.

Ans 5: Initial platform pose given in the problem is singular, and remove this singularity a slightly perturbed platform pose was used. Platform pose and leg lengths obtained from **jacobian** (using steps 0.1%) for 0%, 1%, 2%, 3%, 4% & 5% increase in leg lengths (leg2, leg3 & leg6)are given in Tables 8-14.

$$\begin{bmatrix} 0.9397 & 0.1170 & 0.3214 & 0 \\ 0 & 0.9397 & -0.3420 & 0 \\ -0.3420 & 0.3214 & 0.8830 & 300.0000 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

Symbol	Value (mm)	Symbol	Value(mm)
L_1	271.6395	L_4	338.8664
L_2	315.0874	L_5	293.6797
L_3	351.1502	L_6	261.3731

Table 2: Platform pose and Leg Lengths in the initial configuration

$$\begin{bmatrix} 0.9246 & 0.2289 & 0.3046 & -16.7505 \\ -0.0199 & 0.8274 & -0.5613 & 65.2909 \\ -0.3805 & 0.5129 & 0.7695 & 291.5515 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

Symbol	Value (mm)	Symbol	Value(mm)
L_1	271.6395	L_4	338.8664
L_2	318.2382	L_5	293.6797
L_3	354.6617	L_6	263.9868

Table 3: Platform pose and Leg Lengths for 1% increase in leg lengths

$$\begin{bmatrix} 0.9121 & 0.2479 & 0.3265 & -12.4800 \\ 0.0539 & 0.7171 & -0.6949 & 94.3650 \\ -0.4064 & 0.6514 & 0.6407 & 283.0403 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

Symbol	Value (mm)	Symbol	Value(mm)
L_1	271.6395	L_4	338.8664
L_2	321.3891	L_5	293.6797
L_3	358.1732	L_6	266.6006

Table 4: Platform pose and Leg Lengths for 2% increase in leg lengths

$$\begin{bmatrix} 0.8979 & 0.2607 & 0.3546 & 1.1437 \\ 0.1347 & 0.6042 & -0.7854 & 107.5242 \\ -0.4190 & 0.7530 & 0.5074 & 277.8214 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

Symbol	Value (mm)	Symbol	Value(mm)
L_1	271.6395	L_4	338.8664
L_2	324.5400	L_5	293.6797
L_3	361.6847	L_6	269.2143

Table 5: Platform pose and Leg Lengths for 3% increase in leg lengths

$$\begin{bmatrix} 0.8820 & 0.2731 & 0.3842 & 19.7441 \\ 0.2169 & 0.4885 & -0.8452 & 112.3835 \\ -0.4184 & 0.8288 & 0.3716 & 274.3136 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

Symbol	Value (mm)	Symbol	Value(mm)
L_1	271.6395	L_4	338.8664
L_2	327.6909	L_5	293.6797
L_3	365.1962	L_6	271.8280

Table 6: Platform pose and Leg Lengths for 4% increase in leg lengths

$$\begin{bmatrix} 0.8640 & 0.2858 & 0.4146 & 41.6056 \\ 0.2997 & 0.3698 & -0.8795 & 111.9282 \\ -0.4047 & 0.8841 & 0.2339 & 271.0524 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

Symbol	Value (mm)	Symbol	Value(mm)
L_1	271.6395	L_4	338.8664
L_2	330.8417	L_5	293.6797
L_3	368.7077	L_6	274.4418

Table 7: Platform pose and Leg Lengths for 5% increase in leg lengths

$$\begin{bmatrix} 0.9397 & 0.1170 & 0.3214 & 0 \\ 0 & 0.9397 & -0.3420 & 0 \\ -0.3420 & 0.3214 & 0.8830 & 300.0000 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

Symbol	Value (mm)	Symbol	Value(mm)
L_1	271.6395	L_4	338.8664
L_2	315.0874	L_5	293.6797
L_3	351.1502	L_6	261.3731

Table 8: Platform pose and Leg Lengths in the initial configuration

$$\begin{bmatrix} 0.9246 & 0.2291 & 0.3045 & -16.8387 \\ -0.0200 & 0.8272 & -0.5615 & 65.4841 \\ -0.3805 & 0.5131 & 0.7694 & 291.5388 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

Symbol	Value (mm)	Symbol	Value(mm)
L_1	271.6966	L_4	338.8812
L_2	318.2533	L_5	293.7445
L_3	354.6646	L_6	264.0679

Table 9: Platform pose and Leg Lengths for 1% increase in leg lengths

$$\begin{bmatrix} 0.9121 & 0.2479 & 0.3265 & -12.5145 \\ 0.0539 & 0.7170 & -0.6950 & 94.4465 \\ -0.4064 & 0.6515 & 0.6406 & 283.0216 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

Symbol	Value (mm)	Symbol	Value(mm)
L_1	271.6635	L_4	338.8634
L_2	321.3879	L_5	293.7054
L_3	358.1639	L_6	266.6381

Table 10: Platform pose and Leg Lengths for 2% increase in leg lengths

$$\begin{bmatrix} 0.8979 & 0.2607 & 0.3546 & 1.1263 \\ 0.1347 & 0.6041 & -0.7854 & 107.5674 \\ -0.4190 & 0.7530 & 0.5073 & 277.8118 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

Symbol	Value (mm)	Symbol	Value(mm)
L_1	271.6554	L_4	338.8649
L_2	324.5390	L_5	293.6971
L_3	361.6790	L_6	269.2389

Table 11: Platform pose and Leg Lengths for 3% increase in leg lengths

$$\begin{bmatrix} 0.8820 & 0.2731 & 0.3842 & 19.7332 \\ 0.2169 & 0.4884 & -0.8452 & 112.4111 \\ -0.4184 & 0.8288 & 0.3715 & 274.3113 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

Symbol	Value (mm)	Symbol	Value(mm)
L_1	271.6532	L_4	338.8687
L_2	327.6925	L_5	293.6953
L_3	365.1950	L_6	271.8477

Table 12: Platform pose and Leg Lengths for 4% increase in leg lengths

$$\begin{bmatrix} 0.8640 & 0.2858 & 0.4146 & 41.5972 \\ 0.2997 & 0.3698 & -0.8795 & 111.9487 \\ -0.4047 & 0.8841 & 0.2338 & 271.0561 \\ 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

Symbol	Value (mm)	Symbol	Value(mm)
L_1	271.6529	L_4	338.8726
L_2	330.8462	L_5	293.6957
L_3	368.7108	L_6	274.4592

Table 13: Platform pose and Leg Lengths for 5% increase in leg lengths

```matlab
clear all;
close all;
clc;

%%
% leg end point coordinated [px; py; pz; bx, by, bz]   p: platform  b: base
legs = [100,    50,     -87,    -100,   -50,    87;
        0,      87,     50,     0,      -87,    -50;
        0,      0,      0,      0,      0,      0;
        150,    75,     -130,   -150,   -75,    130;
        0,      130,    75,     0,      -130,   -75;
        0,      0,      0,      0,      0,      0];

% Pf = [Yaw, Pitch, Roll, X, Y, Z]  platform fixed frame information
Pf = [degtorad(0); degtorad(0); degtorad(0); 0; 0; 300];

%%
% Transformation matrix representing platform orientation wrt base
% Following the convention yaw pitch roll
thz = Pf(1);    thy = Pf(3);    thx = Pf(2);
Rz = [cos(thz)      -sin(thz)    0;
      sin(thz)       cos(thz)    0;
      0              0           1];

Ry = [cos(thy)      0    sin(thy);
      0             1           0;
      -sin(thy)     0    cos(thy)];

Rx = [1      0           0;
      0      cos(thx)    -sin(thx);
      0      sin(thx)    cos(thx)];

Rot = Rz*Ry*Rx;

PlatTran = [Rot, [Pf(4); Pf(5); Pf(6)];
            zeros(1,3), 1];
%%
LegLengths = zeros(6,1);
jac = zeros(6,6);   % Kinematic Jacobian about the origin of platform
% As the actuators are applying only forces P*, Q*, R* simplifies to P, Q, R

for i = 1:6
    PlatPt = PlatTran*[legs(1:3,i);1];          r = PlatPt(1:3) - Pf(4:6);
    BasePt = legs(4:6,i);

    L = PlatPt(1) - BasePt(1);  M = PlatPt(2) - BasePt(2);  N = PlatPt(3) - BasePt(3) ;
    LegLengths(i) = sqrt(L^2+M^2+N^2);

    jac(i,:) = [L,M,N,cross(r,[L;M;N])']/LegLengths(i);
end
```

Listing 1: Matalb Code for Problems 1–3

```
close all;
clear all;
% clc;

% This script computes the forward dynamics using newton raphson method

%%
% InPfOri = [Roll, Pitch, Yaw, X, Y, Z] initial platform
% position

InPfOri = [degtorad(0); degtorad(20); degtorad(20); 0; 0; 300];

% Platform and base end of legs, in plarform and base coordinates
% respectivelty   [leg1, leg2 ..... leg6]

legs = [100,    50,     -87,    -100,   -50,    87;
        0,      87,     50,     0,      -87,    -50;
        0,      0,      0,      0,      0,      0;
        150,    75,     -130,   -150,   -75,    130;
        0,      130,    75,     0,      -130,   -75;
        0,      0,      0,      0,      0,      0];
%%
thz = InPfOri(1);    thy = InPfOri(2);    thx = InPfOri(3);
Rz = [cos(thz)      -sin(thz)    0;
      sin(thz)       cos(thz)    0;
      0              0           1];

Ry = [cos(thy)      0    sin(thy);
      0             1           0;
      -sin(thy)     0    cos(thy)];

Rx = [1     0          0;
      0     cos(thx)   -sin(thx);
      0     sin(thx)   cos(thx)];

Rot = Rz*Ry*Rx;

% InPlatPose is platform pose matrix
InPlatPose = [Rot, [InPfOri(4); InPfOri(5); InPfOri(6)];
              zeros(1,3), 1];
%%
InLegLn = zeros(6,1);
InJac = zeros(6,6);
% Kinematic Jacobian about the origin of platform
% As the actuators are applying only forces P*, Q*, R* simplifies to P, Q, R

% PlatPt is coordinate of leg end attached to platform, represented
% in base frame

% r is the vector joining platform origin to PlatPt represented in
% base frame
for i = 1:6
    PlatPt = InPlatPose*[legs(1:3,i);1];
    BasePt = legs(4:6,i);
    r = PlatPt(1:3) - InPlatPose(1:3,4);

    L = PlatPt(1) - BasePt(1);  M = PlatPt(2) - BasePt(2);  N = PlatPt(3) - BasePt(3) ;
    InLegLn(i) = sqrt(L^2+M^2+N^2);
    InJac(i,:) = [L,M,N,cross(r,[L;M;N])']/InLegLn(i);
end

if rank(InJac)<6 ; disp('Intial configuration of platform was singular'); return; end

clearvars -except legs InLegLn InPlatPose

%%

GuessPlatPose = InPlatPose;
```

```matlab
%%
disp(GuessPlatPose)
for multiplier = 1.01:0.01:1.05
    % AcLegLn = Actual Leg Lengths [L1,L2,L3,L4,L5,L6]
    AcLegLn = diag([1,multiplier,multiplier,1,1,multiplier])*InLegLn;
    for i = 1:100                           % newton raphson method
        ThLegLn = zeros(6,1);
        jac = zeros(6,6);                   % with screws stacked horizontally
        for ji = 1:6
            PlatPt = GuessPlatPose*[legs(1:3,ji);1];            r = PlatPt(1:3) - GuessPlatPose(1:3,4);
            BasePt = legs(4:6,ji);
            L = PlatPt(1) - BasePt(1);  M = PlatPt(2) - BasePt(2);  N = PlatPt(3) - BasePt(3) ;
            ThLegLn(ji) = sqrt(L^2+M^2+N^2);
            jac(ji,:) = [L,M,N,cross(r,[L;M;N])']/ThLegLn(ji);
        end

        error = AcLegLn - ThLegLn;
        if norm(error) < 1e-4
            break
        end

        if rank(jac) < 6
            disp('Jacobian turned out to be singular....  Stopping')
            break
        end

        dlta = jac\error;
        GuessPlatPose = NewOri(dlta, GuessPlatPose);
    end
    disp(GuessPlatPose)
end
```

Listing 2: Matalb Code for Forward Dynamics using Newton Raphson

```
close all;
clear all;
% clc;

% This script does the forward kinematic by jacobian approximation and
% error correction. We don't have a control on the amount of error

%%
% InPfOri = [Roll, Pitch, Yaw, X, Y, Z] initial platform
% position

InPfOri = [degtorad(0); degtorad(20); degtorad(20); 0; 0; 300];

% Platform and base end of legs, in plarform and base coordinates
% respectivelty   [leg1, leg2 ..... leg6]

legs = [100,    50,     -87,    -100,   -50,    87;
        0,      87,     50,     0,      -87,    -50;
        0,      0,      0,      0,      0,      0;
        150,    75,     -130,   -150,   -75,    130;
        0,      130,    75,     0,      -130,   -75;
        0,      0,      0,      0,      0,      0];
%%
thz = InPfOri(1);    thy = InPfOri(2);    thx = InPfOri(3);
Rz = [cos(thz)      -sin(thz)   0;
      sin(thz)      cos(thz)    0;
      0             0           1];

Ry = [cos(thy)      0   sin(thy);
      0             1           0;
      -sin(thy)     0   cos(thy)];

Rx = [1     0           0;
      0     cos(thx)    -sin(thx);
      0     sin(thx)    cos(thx)];

Rot = Rz*Ry*Rx;

% InPlatPose is platform pose matrix
InPlatPose = [Rot, [InPfOri(4); InPfOri(5); InPfOri(6)];
            zeros(1,3), 1];
%%
InLegLn = zeros(6,1);
InJac = zeros(6,6);
% Kinematic Jacobian about the origin of platform
% As the actuators are applying only forces P*, Q*, R* simplifies to P, Q, R

% PlatPt is coordinate of leg end attached to platform, represented
% in base frame

% r is the vector joining platform origin to PlatPt represented in
% base frame
for i = 1:6
    PlatPt = InPlatPose*[legs(1:3,i);1];
    BasePt = legs(4:6,i);
    r = PlatPt(1:3) - InPlatPose(1:3,4);

    L = PlatPt(1) - BasePt(1);  M = PlatPt(2) - BasePt(2);  N = PlatPt(3) - BasePt(3) ;
    InLegLn(i) = sqrt(L^2+M^2+N^2);
    InJac(i,:) = [L,M,N,cross(r,[L;M;N])']/InLegLn(i);
end

if rank(InJac)<6 ; disp('Intial configuration of platform was singular'); return; end

clearvars -except legs InLegLn InPlatPose InJac
%%
ThPlatPose = InPlatPose;
ThLegLn = InLegLn;
```

```
ThJac = InJac;
for i = 0.1:0.1:5.1                    % sought of jacobian control with error correction
    if rem(round((i - 0.1)*10)/10,1) == 0
%           disp(i)
        disp(ThPlatPose)
    end
    multiplier = 1+i/100;
    DesiredLegLn = diag([1,multiplier,multiplier,1,1,multiplier])*InLegLn;

    error = (DesiredLegLn - ThLegLn);
    dlta = ThJac\error;
    ThPlatPose = NewOri(dlta, ThPlatPose);


    ThLegLn = zeros(6,1);
    ThJac = zeros(6,6);

    for ji = 1:6
        PlatPt = ThPlatPose*[legs(1:3,ji);1];           r = PlatPt(1:3) - ThPlatPose(1:3,4);
        BasePt = legs(4:6,ji);
        L = PlatPt(1) - BasePt(1);  M = PlatPt(2) - BasePt(2);  N = PlatPt(3) - BasePt(3)  ;
        ThLegLn(ji) = sqrt(L^2+M^2+N^2);
        ThJac(ji,:) = [L,M,N,cross(r,[L;M;N])']/ThLegLn(ji);
    end

    if rank(ThJac)<6 ; disp('Platform configuration turned out to be singular'); return; end

end
```

Listing 3: Matalb Code for Forward Dynamics using Newton Raphson

```
function [ Pfnew ] = NewOri( dlta, Pfold )
% takes delta (dx,dx,dz,dthx,dthy,dthz) and Pfold (platform old position)
% and returns updated position of platform Pfnew

th = norm(dlta(4:6));                          % amount of rotation

kBase = dlta(4:6)/th;                          % vector of direction cosines represented in
    base frame
kx = kBase(1); ky = kBase(2); kz = kBase(3);   % direction cosines
%%
% kNew = Pfold(1:3,1:3)*kBase;
% kx = kNew(1); ky = kNew(2); kz = kNew(3);   % direction cosines
%%

c = cos(th); s = sin(th); v = 1-c;

R = [kx^2*v+c       kx*ky*v-kz*s     kx*kz*v+ky*s    ;
     kx*ky*v+kz*s   ky^2*v+c         ky*kz*v-kx*s    ;
     kx*kz*v-ky*s   ky*kz*v+kx*s     kz^2*v+c     ];

Rot = R*Pfold(1:3,1:3);

Pfnew = [Rot,             Pfold(1:3,4)+dlta(1:3);
         zeros(1,3),      1];
end
```

Listing 4: Matalb Code for Problems 4 & 5