Term Project- Computer Programming of Robot kinematics

Submitted by- Chaitanya Kabade(csk420 / N19500276)

Programming Exercise- Part 2

Q1. If your function library does not include an Atan2 function subroutine, write one.

```
x=input('Enter X=');
y=input('Enter Y=');
theta=tan_inv(x,y);
disp('Theta is')
disp(theta)
```

```
function [t]=tan_inv(x,y)
    t=atan2d(y,x);
end
```

(On command window)

```
>> Atan2_fun_subroutine
Enter X=5
Enter Y=5
Theta is
     45
```

Q2. To make a friendly user interface, we wish to describe orientations in the planar world by a single angle, θ, instead of by a 2 x 2 rotation matrix. The user will always communicate in terms of angle θ, but internally we will need the rotation-matrix form. For the position-vector part of a frame, the user will specify an x and a y value. So, we want to allow the user to specify a frame as a 3-tuple: (x, y, θ). Internally, we wish to use a 2 x 1 position vector and a 2 x 2 rotation matrix, so we need conversion routines. Write a subroutine whose Pascal definition would begin:

Procedure UTOI (VAR uform: vec3; VAR iform: frame);

where "UTOI" stands for "User form To Internal form." The first argument is the 3-tuple (x, y, θ), and the second argument is of type "frame," consists of a (2 x 1) position vector and a (2 x 2) rotation matrix. If you wish, you may represent the frame with a (3 x 3) homogeneous transform in which the third row is [0 0 1]. The inverse routine will also be necessary:

Procedure ITOU (VAR iform: frame; VAR uform: vec3);

```
x = input('Enter x position=');
y = input('Enter y position=');
z = input('Enter theta=');
A = Procedure_UTOI(x,y,z);
disp('The required Rotation Matrix is:')
disp(A)
[a,b,c] = Procedure_ITOU(A);
disp('The required (x, y, theta) is:')
disp([a,b,c])
```

```
function [R] = Procedure_UTOI(x,y,z)
    R = [ cosd(z) -sind(z) x; sind(z) cosd(z) y;0 0 1];
end
function [a,b,c] = Procedure_ITOU(A)
    a = A(1,3);
    b = A(2,3);
    c = acosd(A(1,1));
end
```

(On command window)

```
Enter x position=5
Enter y position=7
Enter theta=65

The required Rotation Matrix in UTOI is:
    0.4226   -0.9063    5.0000
    0.9063    0.4226    7.0000
         0         0    1.0000
The required (x, y, theta) in ITOU form is:
     5     7    65
```

Q3. Write a subroutine to multiply two transforms together. Use the following procedure heading:

Procedure TMULT (VAR brela, crelb, crela: frame);

The first two arguments are inputs, and the third is an output. Note that the names of the arguments document what the program does.

```
x1= input('Enter a value of the x1 coordinate:');

y1= input('Enter a value of the y1 coordinate:');

t1= input('Enter a value of theta1:');

x2= input('Enter a value of the x2 coordinate:');

y2= input('Enter a value of the y2 coordinate:');

t2= input('Enter a value of theta2:');

brela = Procedure_UTOI(x1,y1,t1)

crelb = Procedure_UTOI(x2,y2,t2)

crela=Procedure_TMULT(brela,crelb);

disp('Matrix B relative to A=')

disp(brela)

disp('Matrix C relative to B=')

disp(crelb)

disp('Matrix C relative to A=')

disp(crela)

function [R] = Procedure_UTOI(x,y,z)

    R = [ cosd(z) -sind(z) 0 x; sind(z) cosd(z) 0 y; 0 0 1 0; 0 0 0 1];

end

function [crela]= Procedure_TMULT(brela,crelb)

    crela=brela*crelb;

end
```

(On command window)

```
Enter a value of the x2 coordinate:2
Enter a value of the y2 coordinate:5
Enter a value of theta1:50
Enter a value of the x2 coordinate:6
Enter a value of the y2 coordinate:3
Enter a value of theta2:30
```

```
Matrix B relative to A=
    0.6428   -0.7660         0    2.0000
    0.7660    0.6428         0    5.0000
         0         0    1.0000         0
         0         0         0    1.0000
Matrix C relative to B=
    0.8660   -0.5000         0    6.0000
    0.5000    0.8660         0    3.0000
         0         0    1.0000         0
         0         0         0    1.0000
Matrix C relative to A=
    0.1736   -0.9848         0    3.5586
    0.9848    0.1736         0   11.5246
         0         0    1.0000         0
         0         0         0    1.0000
```

Q4. Write a subroutine to invert a transform. Use the following procedure heading:

Procedure TINVERT (VAR brela, arelb: frame);

The first argument is the input, the second the output. Note that the names of the arguments document what the program does.

```
x= input('Enter a value of the x coordinate:');
y= input('Enter a value of the y coordinate:');
t= input('Enter a value of theta:');
brela = Procedure_UTOI(x,y,z)
arelb = Procedure_TINVERT(brela)
disp('Matrix B relative to A=')
disp(brela)
disp('Matrix A relative to B=')
disp(output_arelb)
```

```
function [R] = Procedure_UTOI(x,y,z)
    R = [ cosd(z) -sind(z) 0 x; sind(z) cosd(z) 0 y; 0 0 1 0; 0 0 0 1];
end
function [arelb]=Procedure_TINVERT(brela)
    arelb= inv(brela);
end
```

(On command window)

Enter a value of the x coordinate:5
Enter a value of the y coordinate:6
Enter a value of theta:50

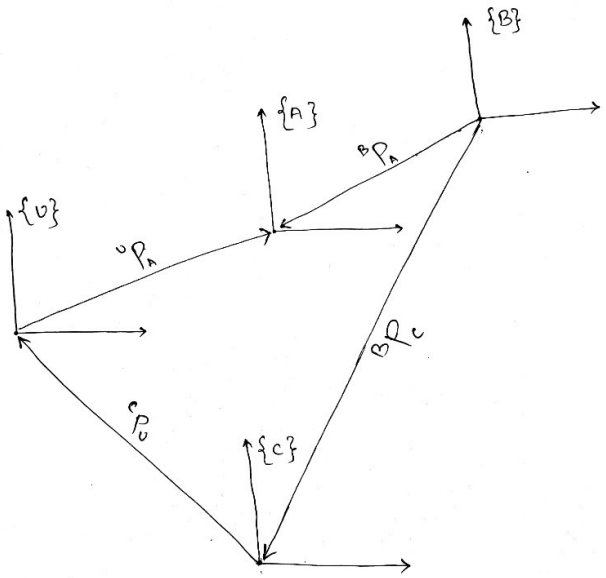
```
Matrix B relative to A=
    0.6428   -0.7660         0    5.0000
    0.7660    0.6428         0    6.0000
         0         0    1.0000         0
         0         0         0    1.0000
Matrix A relative to B=
    0.6428    0.7660         0   -7.8102
   -0.7660    0.6428         0   -0.0265
         0         0    1.0000         0
         0         0         0    1.0000
```

Q5. The following frame definitions are given as known:

arelu = [11.0 1.0 30.0],
arelb = [0.07.0 45.0],
urelc = [-3.0 -3.0 -30.0].

These frames are input in the user representation [x, y, θ] (where θ is in degrees). Draw a frame diagram (like Fig. 2.15, only in 2-D) that qualitatively shows their arrangement. Write a program that calls TMULT and TINVERT (defined in programming exercises 3 and 4) as many times as needed to solve for CrelB. Then print out CrelB in both internal and user representation.

Diagram:




```
x1 = input('Enter x position=');
y1 = input('Enter y position=');
z1 = input('Enter theta=');
x2 = input('Enter x position=');
y2 = input('Enter y position=');
z2 = input('Enter theta=');
x3 = input('Enter x position=');
y3 = input('Enter y position=');
z3 = input('Enter theta=');
ArelU=UTOI(x1,y1,z1)
ArelB=UTOI(x2,y2,z2)
UrelC=UTOI(x3,y3,z3)
UrelA=TINVERT(ArelU)
CrelU=TINVERT(UrelC)
UrelB=TMULT(ArelB,UrelA)
CrelB=TMULT(UrelB,CrelU)
disp('User form of Matrix C relative to B is')
[x0,y0,z0]=ITOU(CrelB)
```

(On command window)

Enter x position=11
Enter y position=-1
Enter theta=30
Enter x position=0
Enter y position=7
Enter theta=45
Enter x position=-3

```
Enter y position=-3
Enter theta=-30
```

```
ArelU = 3×3
    0.8660   -0.5000   11.0000
    0.5000    0.8660   -1.0000
         0         0    1.0000
ArelB = 3×3
    0.7071   -0.7071         0
    0.7071    0.7071    7.0000
         0         0    1.0000
UrelC = 3×3
    0.8660    0.5000   -3.0000
   -0.5000    0.8660   -3.0000
         0         0    1.0000
UrelA = 3×3
    0.8660    0.5000   -9.0263
   -0.5000    0.8660    6.3660
         0         0    1.0000
CrelU = 3×3
    0.8660   -0.5000    1.0981
    0.5000    0.8660    4.0981
         0         0    1.0000
UrelB = 3×3
    0.9659   -0.2588  -10.8840
    0.2588    0.9659    5.1189
         0         0    1.0000
CrelB = 3×3
    0.7071   -0.7071  -10.8840
    0.7071    0.7071    9.3616
         0         0    1.0000
User form of Matrix C relative to B is
x0 = -10.8840
y0 = 9.3616
z0 = 45
```

Programming Exercise- Part 3

Q1. Write a subroutine to compute the kinematics of the planar 3R robot in Example 3.3—that is, a routine with the joint angles' values as input, and a frame (the wrist frame relative to the base frame) as output. Use the procedure heading (or equivalent in C):

Procedure KIN(VAR theta: vec3; VAR wrelb: frame);

where "wrelb" is the wrist frame relative to the base frame, the type "frame" consists of a 3 x 3 rotation matrix and a 3 x 1 position vector. If desired, you may represent the frame with a 4 x 4 homogeneous transform in which the fourth row is [0 0 0 1]. (The manipulator data are l1 = l2 = l3 = 0.5 meters.)

```
t1=input('Input theta1=');
t2=input('Input theta2=');
t3=input('Input theta3=');
L=0.5; %considering length of all the links are same i.e. L1=L2=L3=0.5m
WrelB= Procedure_Kin(t1,t2,t3,L);
disp('W relative to B is given by=')
disp(WrelB)
```

```
function [WrelB]= Procedure_Kin(t1,t2,t3,L)
 R1= [ cosd(t1) -sind(t1) 0 L*cosd(t1); sind(t1) cosd(t1) 0 L*sind(t1); 0 0 1 0; 0 0 0 1];
 R2= [ cosd(t2) -sind(t2) 0 L*cosd(t2); sind(t2) cosd(t2) 0 L*sind(t2); 0 0 1 0; 0 0 0 1];
 R3= [ cosd(t3) -sind(t3) 0 L*cosd(t3); sind(t3) cosd(t3) 0 L*sind(t3); 0 0 1 0; 0 0 0 1];
    WrelB=R1*R2*R3 ;
```

```
end
```

(On command window)

Input theta1=20
Input theta2=45
Input theta3=65

```
W relative to B is given by=
   -0.6428   -0.7660         0    0.3598
    0.7660   -0.6428         0    1.0072
         0         0    1.0000         0
         0         0         0    1.0000
```

Q2. Write a routine that calculates where the tool is, relative to the station frame. The input to the routine is a vector of joint angles:

Procedure WHERE(VAR theta: vec3; VAR TrelS: frame);

Obviously, WHERE must make use of descriptions of the tool frame and the robot base frame in order to compute the location of the tool relative to the station frame. The values of TrelW and BrelS should be stored in global memory (or, as a second choice, you may pass them as arguments in 'WHERE).

```
t1=input('Input theta1=');
t2=input('Input theta2=');
t3=input('Input theta3=');
L=0.5; %considering length of all the links are same i.e. L1=L2=L3=0.5m
x1=input('Input x position of TrelW=');
y1=input('Input y position of TrelW=');
z1=input('Input theta of TrelW=');
x2=input('Input x position of BrelS=');
y2=input('Input y position of BrelS=');
z2=input('Input theta of BrelS=');
TrelW= UTOI(x1,y1,z1);
BrelS=UTOI(x2,y2,z2);
TrelS=Procedure_WHERE(t1,t2,t3,L,TrelW,BrelS)

function [R] = Procedure_WHERE(t1,t2,t3,L,TrelW,BrelS)
WrelB = KIN(t1,t2,t3,L);
R=BrelS*WrelB*TrelW;
end
```

(On command window)

Input theta1=0
Input theta2=30
Input theta3=60
Input x position of TrelW=2
Input y position of TrelW=5
Input theta of TrelW=60
Input x position of BrelS=5
Input y position of BrelS=6
Input theta of BrelS=30

```
TrelS = 4×4
   -1.0000         0         0    0.1029
         0   -1.0000         0    6.3481
         0         0    1.0000         0
         0         0         0    1.0000
```

Q3. A tool frame and a station frame for a certain task are defined as follows by the user: trelw = [0.1 0.2 30.0], srelb = [—0.1 0.3 0.0]. Calculate the position and orientation of the tool relative to the station frame for the following three configurations (in units of degrees) of the arm:

$[\theta_1, \theta_2, \theta_3] = [0.0, 90.0, —90.0]$; $[\theta_1, \theta_2, \theta_3] = [—23.6, —30.3, 48.0]$;
$[\theta_1, \theta_2, \theta_3] = [130.0, 40.0, 12.0]$.

```
t1=input('Enter theta1=');
t2=input('Enter theta2=');
t3=input('Enter theta3=');
L=0.5;
TrelW=UTOI(0.1,0.2,30);
SrelB=UTOI(-0.1,0.3,0);
BrelS=TINVERT(SrelB);
WrelB = KIN(t1,t2,t3,L);
TrelS = BrelS*WrelB*TrelW;
[a,b,c]=ITOU(TrelS);
disp('Position and Orientation of tool relative to station frame')
disp([a,b,c])
```

(On command window)
Enter theta1=0
Enter theta2=90
Enter theta3=-90

```
Position and Orientation of tool relative to station frame
    1.2000    0.4000   30.0000
```

Enter theta1=-23.6
Enter theta2=-30.3
Enter theta3=48

```
Position and Orientation of tool relative to station frame
    1.4702   -0.7669   24.1000
```

Enter theta1=130
Enter theta2=40
Enter theta3=12

```
Position and Orientation of tool relative to station frame
   -1.3065   -0.0510  148.0000
```

Programming Exercise- Part 4

Q1. Write a subroutine to calculate the inverse kinematics for the three-link manipulator of Section 4.4. The routine should pass arguments in the form

Procedure INVKIN(VAR wrelb: frame; VAR current, near, far: vec3; VAR sol: boolean);

where "wrelb," an input, is the wrist frame specified relative to the base frame; "current," an input, is the current position of the robot (given as a vector of joint angles); "near" is the nearest solution; "far" is the second solution; and "sol" is a flag that indicates whether solutions were found. (sol = FALSE if no solutions were found). The link lengths (meters) are $l1 = l2 = l3 = 0.5$. The joint ranges of motion are $—170° \leq \theta \leq 170°$. Test your routine by calling it back-to-back with KIN to demonstrate that they are indeed inverses of one another.

```
L1 = input('Enter a value of L1=');
L2 = input('Enter a value of L2=');
L3 = input('Enter a value of L3=');
x = input('Enter a value of x=');
y = input('Enter a value of y=');
t = input('Enter a value of theta=');
```

```
wrelb = UTOI(x,y,t);
[near,far,solution] = INVERSEKIN(wrelb,L1,L2,L3);
```

```
function [near,far,solution]= INVERSEKIN(wrelb,L1,L2,L3)
Px=wrelb(1,3)-(L3*wrelb(1,1));
Py=wrelb(2,3)-(L3*wrelb(2,1));

c2=(Px^2+Py^2-L1^2-L2^2)/(2*L1*L2);

if abs(c2)<1
    solution=true;
%calculating theta2 for closer position
    s2=sqrt(1-c2^2);
    near(1,2)=atan2d(s2,c2);
 %calculating theta1 for closer position
    c1=((L1+L2*c2)*Px+(L2*s2)*Py)/((L1+L2*c2)^2+(L2*s2)^2);
    s1=(-L2*Px*s2+(L1+L2*c2)*Py)/((L1+L2*c2)^2+(L2*s2)^2);
   near(1,1)=atan2d(s1,c1);
%calculating theta3 for closer position
    near(1,3)=atan2d(wrelb(2,1),wrelb(1,1)-near(1,1)-near(1,2));
%calculating theta2 for farther position
    s2f=-s2;
    far(1,2)=atan2d(s2f,c2);
%calculating theta1 for farther position
    c1f=((L1+L2*c2)*Px+(L2*s2f)*Py)/((L1+L2*c2)^2+(L2*s2f)^2);
    s1f=(-L2*s2f*Px+(L1+L2*c2)*Py)/((L1+L2*c2)^2+(L2*s2f)^2);
    far(1,1)=atan2d(s1f,c1f);
%calculating theta3 for farther position
    far(1,3)=atan2d(wrelb(2,1),wrelb(1,1))-far(1,1)-far(1,2);
%%
disp('Near solutions')
disp(near)
disp('Far solutions')
disp(far)
%%
%setting the limits as -170<[theta1 theta2 theta3]<170
    if (near(1,1)>170 || near(1,2)>170 || near(1,3)>170 ||  near(1,1)<-170 || near(1,2)<-
170 || near(1,3)<-170 );
        disp('Invalid solution because near angles are greater than 170 or less than -
170');
    elseif (far(1,1)>170 || far(1,2)>170 || near(1,3)>170 || far(1,1)<-170 || far(1,2)<-
170 || near(1,3)<-170);
        disp('Invalid solution because far angles are greater than 170 or less than -170')
    end
else
    solution=false;
    disp('No solution exists');
end
end
```

(On command window)

```
>> INVKIN
Enter a value of L1=0.5
Enter a value of L2=0.5
Enter a value of L3=0.5
Enter a value of x=5
Enter a value of y=6
Enter a value of theta=35
Near solutions
```

155.0000 120.0000 179.8801
Far solutions
-85.0000 -120.0000 240.0000

“Invalid solution because near angles are greater than 170 or less than -170”
>> INVKIN
Enter a value of L1=0.5
Enter a value of L2=0.5
Enter a value of L3=0.5
Enter a value of x=5
Enter a value of y=6
Enter a value of theta=105
Near solutions
-135.0000 120.0000 3.7490

Far solutions
-15.0000 -120.0000 240.0000

Q2. A tool is attached to link 3 of the manipulator. This tool is described by the tool frame relative to the wrist frame. Also, a user has described his work area, the station frame relative to the base of the robot. Write the subroutine

Procedure SOLVE(VAR trels: frame; VAR current, near, far: vec3; VAR sol: boolean);

where "trels" is the {T} frame specified relative to the {S} frame. Other parameters are exactly as in the INVKIN subroutine. The definitions of {T} and {S} should be globally defined variables or constants. SOLVE should use calls to TMULT, TINVERT, and INVKIN.

```
L1 = input('Enter a value of L1=');
L2 = input('Enter a value of L2=');
L3 = input('Enter a value of L3=');
x1 = input('Enter a value of x1=');
y1 = input('Enter a value of y1=');
t1 = input('Enter a value of theta1=');
x2 = input('Enter a value of x2=');
y2 = input('Enter a value of y2=');
t2 = input('Enter a value of theta2=');
x3 = input('Enter a value of x3=');
y3 = input('Enter a value of y3=');
t3 = input('Enter a value of theta3=');
trels = UTOI(x1,y1,t1);
trelw = UTOI(x2,y2,t2);
srelb = UTOI(x3,y3,t3);
[near,far,solution] = SOLVE1(trels,trelw,srelb,L1,L2,L3);
```

```
function [near,far,sol]=SOLVE1 (trels,trelw,srelb,L1,L2,L3)
wrelt= TINVERT(trelw);
wrels= TMULT (trels,wrelt);
wrelb= TMULT (srelb,wrels);
[near,far,sol]= INVKIN(wrelb,L1,L2,L3);
end
```

(On command window)

Enter a value of L1=5
Enter a value of L2=5
Enter a value of L3=5
Enter a value of x1=2
Enter a value of y1=2
Enter a value of theta1=30

Enter a value of x2=5
Enter a value of y2=6
Enter a value of theta2=30
Enter a value of x3=9
Enter a value of y3=9
Enter a value of theta3=45

```
Near solutions
  -46.9674  103.4699  179.2739

Far solutions
   56.5025 -103.4699   91.9674
```

Q3. Write a main program that accepts a goal frame specified in terms of x, y, and this goal specification is {T} relative to {S}, which is the way the user wants to specify goals. The robot is using the same tool in the same working area as in Programming Exercise (Part 2), so {T} and {S} are defined as

TrelW= [x y θ] = [0.1 0.2 30.0],
SrelB= [x y θ] = [—0.1 0.3 0.0].

Calculate the joint angles for each of the following three goal frames:

$$[x_1\ y_1\ \phi_1] = [0.0\ 0.0\ -90.0],$$
$$[x_2\ y_2\ \phi_2] = [0.6\ -0.3\ 45.0],$$
$$[x_3\ y_3\ \phi_3] = [-0.4\ 0.3\ 120.0],$$
$$[x_4\ y_4\ \phi_4] = [0.8\ 1.4\ 30.0].$$

Assume that the robot will start with all angles equal to 0.0 and move to these three goals in sequence. The program should find the nearest solution with respect to the previous goal point. You should call SOLVE and WHERE back-to-back to make sure they are truly inverse functions.

```
L1=0.5;
L2=0.5;
L3=0.5;
%initializing closer angles and farther angles
near=[0 0 0];
far= [0 0 0];
trelw= UTOI(0.1,0.2,30);
srelb= UTOI(-0.1,0.3,0);
% user will provide the input for the position and orientation of goal so
% that we can decide about the existence of solution
x= input('Enter x position of goal matrix=');
y= input('Enter y position of goal matrix=');
phi= input('Enter rotation phi of goal matrix=');
trels= UTOI(x,y,phi);
[near,far,solution]=SOLVE(trels,trelw,srelb,L1,L2,L3);
```

(On command window)

```
Enter x position of goal matrix=0
Enter y position of goal matrix=0
Enter rotation phi of goal matrix=90
Near solutions
 -180.0000  120.0000    0.8201

Far solutions
  -60.0000 -120.0000  240.0000


Enter x position of goal matrix=0.6
```

```
Enter y position of goal matrix=-0.3
Enter rotation phi of goal matrix=45
Near solutions
  135.0000  120.0000  179.9416

Far solutions
 -105.0000 -120.0000  240.0000
```

Enter x position of goal matrix=-0.4
Enter y position of goal matrix=0.3
Enter rotation phi of goal matrix=120

```
Near solutions
 -150.0000  120.0000    1.9092

Far solutions
  -30.0000 -120.0000  240.0000
```

Enter x position of goal matrix=0.8
Enter y position of goal matrix=1.4
Enter rotation phi of goal matrix=30

```
Near solutions
  120.0000  120.0000  180.0000

Far solutions
 -120.0000 -120.0000  240.0000
```