Gröbner Basis

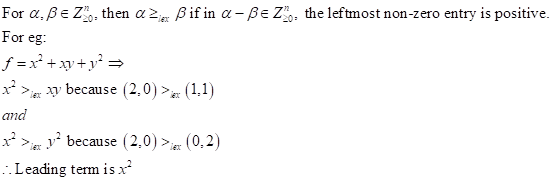
Table of Contents

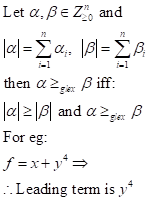
[Polynomial Division](#MW_H_C74485F7)   
[Monomial Orderings](#MW_H_1C14F314)   
[Multi-variate Polynomial Division](#MW_H_7F77CDC2)   
[Rings](#MW_H_1E7F7A07)   
[Ideals & Fields](#MW_H_603D9D76)   
[Gröbner Basis](#MW_H_6214345D)   
[Buchberger's Algorithm](#MW_H_B400D701)   
[Unique Remainders](#MW_H_13D9203C)   
[Applications in Inverse Kinematics](#MW_H_E8FEF898)   
 [Consider a robot with 2 revolute joints.](#MW_H_B7433057)   
 [Now considering a 3-DOF 5R Robot.](#MW_H_CD0572C9)

## Polynomial Division

* Leading monomial: Monomial of the polynomial with the highest degree.
* Leading coefficient: Coefficient of the leading monomial.
* Leading term: Product of *a* and *b*.
* Degree of polynomial: Degree of the leading term.
* *Degree of remainder always lesser than that of the divisor.*

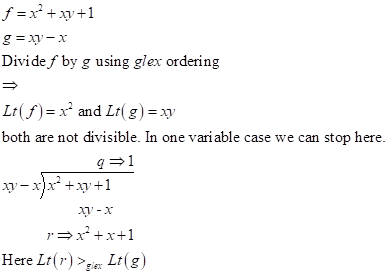
## Monomial Orderings

* Lexicographic ordering or Lex ordering:
* Graded lexicographic ordering:



## Multi-variate Polynomial Division

* Dividing 2 polynomials:



clear all;clc

syms x y

f = x^2+x\*y+1;

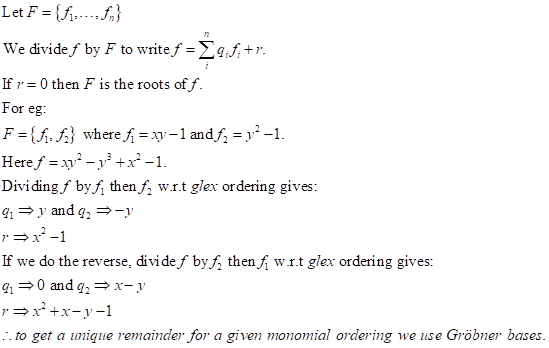
d = x\*y-x;

[r,q] = polynomialReduce(f,d)

r = 

q =

1

* Dividing by a system of polynomials:

syms x y

f = x\*y^2-y^3+x^2-1;

d1 = [x\*y - 1, y^2 - 1];

d2 = [y^2 - 1, x\*y - 1];

[r1,q1] = polynomialReduce(f,d1)

r1 = 

q1 = 

[r2,q2] = polynomialReduce(f,d2)

r2 = 

q2 = 

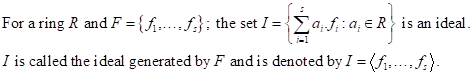
## Rings

A ring is a non-empty set equipped with 2 operations that satisfy the following axioms:

* It is closed under addition: 
* Associative
* Commutative
* Should have additive identity: 
* Should have additive inverse.
* Closed under multiplication:
* Distributive and assosiative.
* *A set of* ***Z****,* ***Q*** *and* ***C*** *are rings. Set of even numbers form a ring but set of odd nos don’t.*

## Ideals & Fields

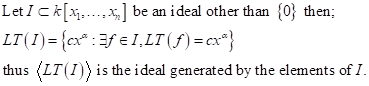
* A subring *I* of a ring *R* is an **ideal** *iff* when *r ∈ R* and*a ∈ I,* then *r.a ∈ I* and *a.r ∈ I*. For eg. a set of even integers is an ideal of the ring ***Z*** and  and ***R*** are ideals for every ring ***R***.

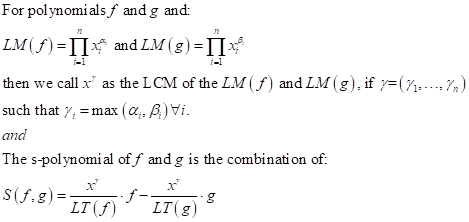


* The zero ideal is generated by a single element set; *I=<**>={**} ∀* ring ***R***.
* Ideals can have different set of generators.
* Noetherian Rings: A ring ***R*** is a Noetherian ring if every ideal *I*of ***R*** is finitely generated. And, according to Hilbert’s Basis Theorem if *R* is a Noetherian ring then so is the polynomial ring *R[x]*.

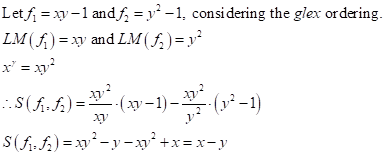
## Gröbner Basis

(LM(*f*) denotes fixed monomial ordering and LT(*f*) denotes the leading term.)

* A **Gröbner basis** is a set of multivariate nonlinear polynomials enjoying certain properties that allow simple algorithmic solutions for many fundamental problems in mathematics and natural and technical sciences.
* An arbitrary set of polynomials is in general not a Gröbner basis. However, for every basis  there exists a Gröbner basis  which is equivalent to  in the sense that the linear combinations of elements of  are precisely the same as the linear combinations of elements of . For  and  equivalent in this sense, the congruence of polynomials modulo  agrees with the congruence modulo .
* Given an arbitrary basis , an equivalent Gröbner basis  can be found constructively by Buchberger's Algorithm.
* A Gröbner basis of an ideal *I* is a set of generators of *I*.
* For a fixed monomial ordering a finite subset *G=*of an ideal *I* is said to be a Gröbner basis if:
* Reduced Gröbner basis: A RGb for a set of polynomials *F* is a Gröbner basis *G* of *F* such that: 1. LC is 1  and for all , none of the terms of *p* is divisible by the . That is a basis is a Gröbner basis if and only if all S-polynomials of any two basis elements reduce to zero.
* For a linear system of equations Grobner Basis is basically the Row Echelon form. For a univariate system of equations it is the GCD(). In general for an ideal **I** the Grobner basis consists of all the "smallest polynomials" in **I** for a given monomial ordering.
* S-polynomials:



* For example:



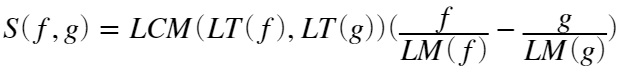
## Buchberger's Algorithm

A variety of frequently arising questions about sets of polynomial equations can be answered easily when the sets are "Gröbner bases" while they are not easy to answer for an arbitrary set of polynomials. Buchberger's algorithm provides a means to transform an arbitrary set of polynomials into an equivalent Gröbner basis. Therefore, questions about some set of polynomials arising in an application can be answered by first using Buchberger's algorithm to compute an equivalent Gröbner basis and then answering the question for the Gröbner basis.

Gröbner basis is a reduction step of a polynomial *p* with respect to a fixed set *B* of polynomials consists of subtracting some multiple of an element of *B* from *p* in such a way that some power product in *p* is replaced by power products which are smaller (w.r.t. a fixed order imposed on the power products).

Repeated reduction steps, after finitely many steps, will always lead to some polynomial which cannot be reduced any further. Such a polynomial is called a *reduced form* of *p* with respect to *B* . Reduced forms are in general not unique. However, one can give an algorithm which for given *p* and *B* computes *some* reduced form of *p* with respect to *B* . By RED we denote such a function, i.e., RED(*p*,*B*) is a reduced form of *p* with respect to *B* obtained by repeated reduction steps.

We also recall from the Gröbner basis article that the S-polynomial of two polynomials *p* and *q* is defined as:

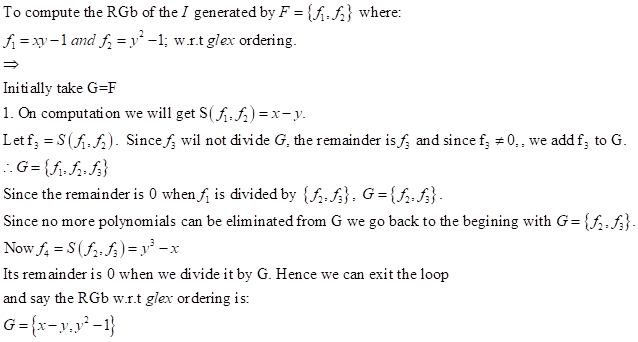


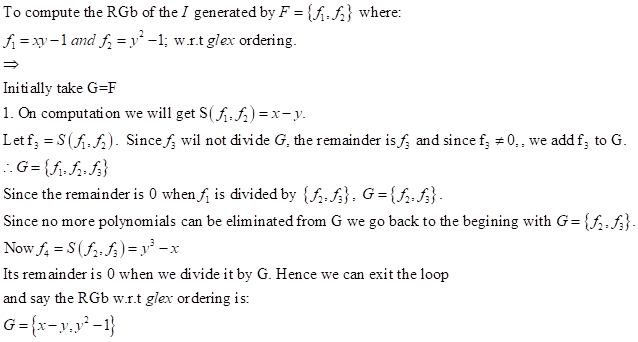
where the symbols LT and LM refer to the leading power product and the leading monomial (w.r.t. a fixed order imposed on the power products), and LCM to the least common multiple.

**Termination** of the algorithm is shown by considering the leading power products of the polynomials in *G* . A leading power product of some new polynomial *h* cannot be a multiple of a leading power product of a polynomial in *G* , because the polynomials *h* are completely reduced with respect to *G* . If on some input the algorithm did not terminate, this would give rise to an infinite sequence of power products in which none is a multiple of any power product appearing earlier in the sequence.

Rough algorithm:

* Chose the monomial ordering.
* Start with *G := F*.
* Repeat G’ := G until G = G’

For example: 



syms x y

r = [x\*y-1, y^2-1];

G = gbasis(r)

G = 

Proper algorithm:

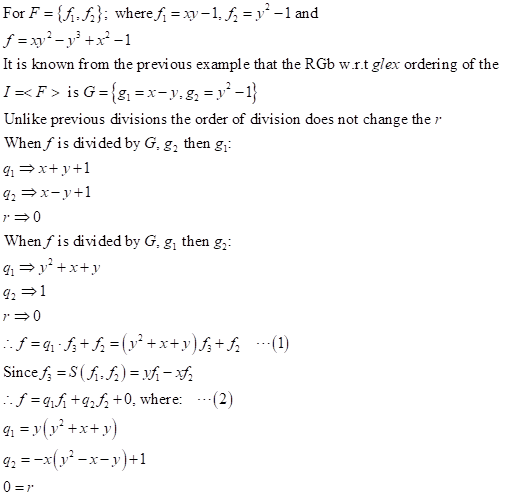
Input: A finite set *F* of polynomials.

Output: A finite Gröbner basis *G* equivalent to *F*

1. 
2. 
3. while  do
4. choose a pair  from 
5. 
6. 
7. if  then
8. 
9. 
10. return 

## Unique Remainders

* For example:



syms x y

f = x\*y^2-y^3+x^2-1;

d = [x\*y-1,y^2-1];

G = gbasis(d);

Gd = [G(1,2) G(1,1)];

[r1,q1] = polynomialReduce(f,G)

r1 =

0

q1 = 

[r2,q2] = polynomialReduce(f,Gd)

r2 =

0

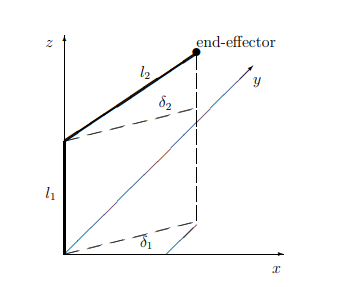
q2 = 

A zero remainder implies that the solutions of *F* are the roots of *f*. In conclusion, to divide a polynomial *f* by a set of polynomials *F* to get a unique remainder the following steps are to be followed:

* Compute Gröbner basis of the ideal *I=<F>.*
* Divide *f* by *G* to get a unique remainder *r*.
* Trace the quotientsfrom the S-polynomials to write 

## Applications in Inverse Kinematics

### Consider a robot with 2 revolute joints.



here we have 9 parameters;  where  are the link lengths;  are the end-effector x-, y- and z- coordinates;  are the angles describing the revolutions of the revolute joints; and  are the sines and cosines of .

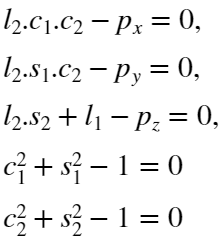
So with these we consider a system of equations:

given 

solve for 

syms l1 l2 px pz s1 s2 c1 c2 py

Using these we can get 5 equations:



F = [l2\*c1\*c2 - px, l2\*s1\*c2-py, l2\*s2+l1-pz, c1^2+s1^2-1, c2^2+s2^2-1];

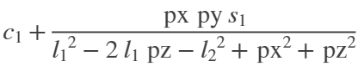
G = gbasis(F,[c1 c2 s1 s2 py],'MonomialOrder','lexicographic');

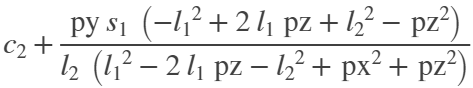
siz = size(G);

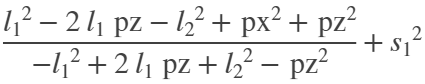
for i=1:siz(2)

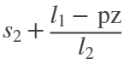
disp(G(i))

end





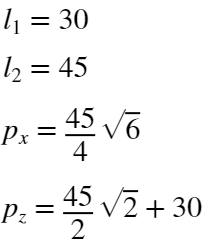






These can be transformed into a Grobner Basis in the polynomial ring :

This gives us 5 equations and 5 unknowns and thus solves the problem. Let:



L1 = 30;L2 = 45; PX = 45\*sqrt(6)/4; PZ = 45\*sqrt(2)/2+30;

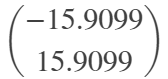
subs1 = subs(G(5), [l1, l2, px, pz], [L1, L2, PX, PZ]);

sol1 = solve(subs1 , py,'ReturnConditions',true);

disp('py=')

py=

disp(vpa(sol1.py,6))



siz = size(sol1.py);

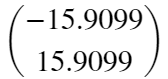
if(siz(1) > 1)

PY = vpa(sol1.py(2),6);

else

PY = vpa(sol1.py,6);

end

Here we get py=. We take py=15.9099

subs2 = subs(G(4), [l1, l2, pz], [L1, L2, PZ]);

sol2 = solve(subs2 , s2,'ReturnConditions',true);

disp('s2=')

s2=

disp(vpa(sol2.s2,6))

0.707107

siz = size(sol2.s2);

if(siz(1) > 1)

S2\_1 = vpa(sol2.s2(2),6);

else

S2\_1 = vpa(sol2.s2,6);

end

s2=

subs3 = subs(G(3), [l1, l2, px, pz], [L1, L2, PX, PZ]);

sol3 = solve(subs3 , s1,'ReturnConditions',true);

disp('s1=')

s1=

disp(vpa(sol3.s1,6))



siz = size(sol3.s1);

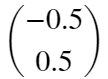
if(siz(1) > 1)

S1\_1 = vpa(sol3.s1(2),6);

else

S1\_1 = vpa(sol3.s1,6);

end

s1=. We take s1=0.5

subs4 = subs(G(2), [s1, l1, l2, px, py, pz], [S1\_1, L1, L2, PX, PY, PZ]);

sol4 = solve(subs4 , c2,'ReturnConditions',true);

disp('c2=')

c2=

disp(vpa(sol4.c2,6))

0.707107

siz = size(sol4.c2);

if(siz(1) > 1)

C2\_1 = vpa(sol4.c2(2),6);

else

C2\_1 = vpa(sol4.c2,6);

end

c2=

subs5 = subs(G(1), [s1, l1, l2, px, py, pz], [S1\_1, L1, L2, PX, PY, PZ]);

sol5 = solve(subs5 , c1,'ReturnConditions',true);

disp('c1=')

c1=

disp(vpa(sol5.c1,6))

0.866025

siz = size(sol5.c1);

if(siz(1) > 1)

C1\_1 = vpa(sol5.c1(2),6);

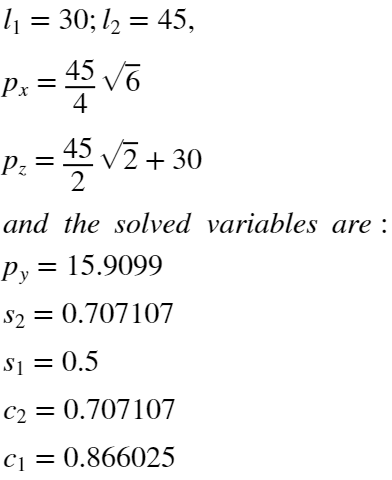
else

C1\_1 = vpa(sol5.c1,6);

end

and c1=

Thus the problem is solved with:



and the angles are:

del1 = vpa(rad2deg(acos(C1\_1)),6);

del2 = vpa(rad2deg(acos(C2\_1)),6);

disp('𝛿1=')

𝛿1=

disp(del1);

30.0

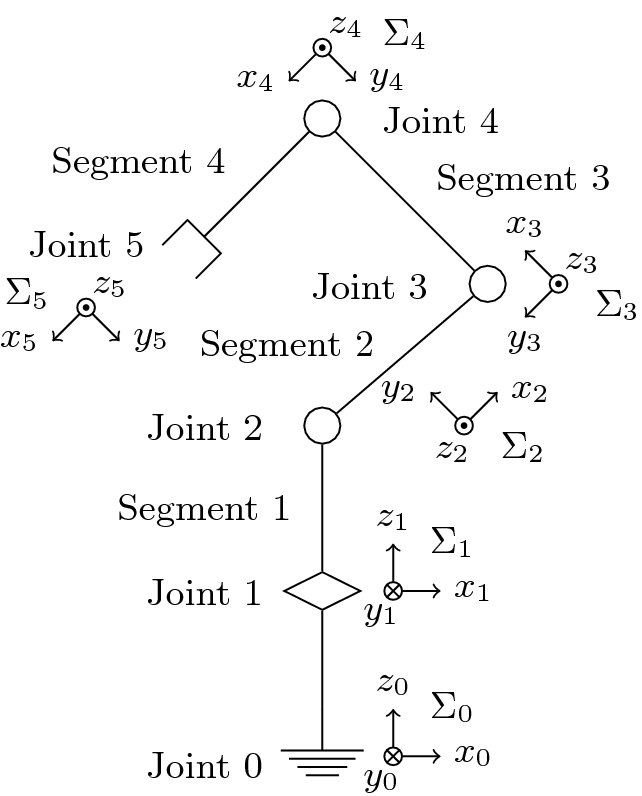
disp('𝛿2=')

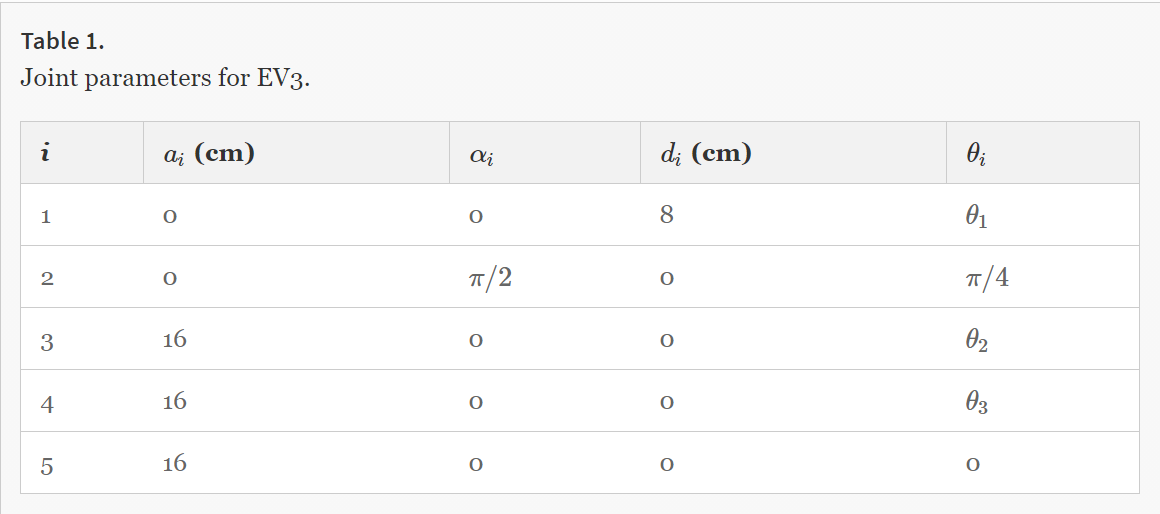
𝛿2=

disp(del2);

45.0

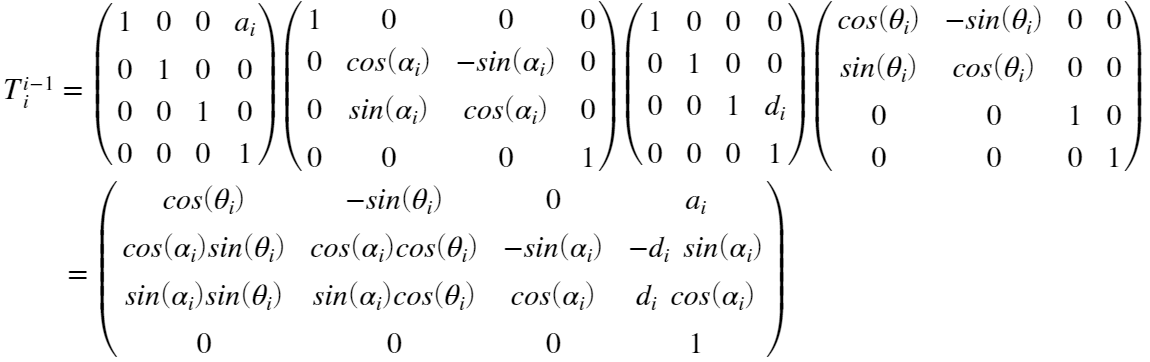
### Now considering a 3-DOF 5R Robot.





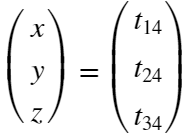
- where EV3 is the manipulator being used in this example.

DH convention to express transformation from  system to  system.



where  is the distance beween  and ,  is the angle between axes  and  with respect to  axis,  is the distance between  and , and  the angle between axes  and  with respect to  axis.

Thus we can get  and the end effector position is the translation part of the matrix.



syms th1 th2 th3

the = [th1, pi/4, th2, th3, 0];

d = [8, 0, 0, 0, 0];

alp = [0, pi/2, 0, 0, 0];

a = [0, 0, 16, 16, 16];

T0\_1 = getMat(a(1), alp(1),d(1),the(1));

T1\_2 = getMat(a(2), alp(2),d(2),the(2));

T2\_3 = getMat(a(3), alp(3),d(3),the(3));

T3\_4 = getMat(a(4), alp(4),d(4),the(4));

T4\_5 = getMat(a(5), alp(5),d(5),the(5));

T0\_5 = T0\_1\*T1\_2\*T2\_3\*T3\_4\*T4\_5;

% simplify(vpa(T0\_5,2))

x = 12;

y = 9;

z = 4;

EP = [x ; y ; z]

EP = 3×1

12

9

4

syms c1 s1 c2 s2 c3 s3

f1 = 8\*sqrt(2)\*c1\*(c2+c3\*(c2-s2)-s2-s3\*(c2+s2)+1) - x;

f2 = 8\*sqrt(2)\*s1\*(c2+c3\*(c2-s2)-s2-s3\*(c2+s2)+1) - y;

f3 = 8\*sqrt(2)\*(c2+c3\*(c2+s2)+s2+s3\*(c2-s2)+1) + 8 - z;

f4 = s1^2 + c1^2 - 1;

f5 = s2^2 + c2^2 - 1;

f6 = s3^2 + c3^2 - 1;

F = [f1,f2,f3,f4,f5,f6];

G = gbasis(F,[c1 c2 s1 s2 c3 s3],'MonomialOrder','lexicographic');

[a1,a2,a3,a4,a5,a6] = solve(G,[c1 s1 c2 s2 c3 s3]);

C1\_1 = double(vpa(a1(1),3));

S1\_1 = double(vpa(a2(1),3));

C2\_1 = double(vpa(a3(1),3));

S2\_1 = double(vpa(a4(1),3));

C3\_1 = double(vpa(a5(1),3));

S3\_1 = double(vpa(a6(1),3));

C1\_2 = double(vpa(a1(2),3));

S1\_2 = double(vpa(a2(2),3));

C2\_2 = double(vpa(a3(2),3));

S2\_2 = double(vpa(a4(2),3));

C3\_2 = double(vpa(a5(2),3));

S3\_2 = double(vpa(a6(2),3));

theta1\_1 = rad2deg(acos(C1\_1));

theta2\_1 = rad2deg(acos(C2\_1));

theta3\_1 = rad2deg(acos(C3\_1));

theta1\_2 = rad2deg(acos(C1\_2));

theta2\_2 = rad2deg(acos(C2\_2));

theta3\_2 = rad2deg(acos(C3\_2));

disp("Orientation 1:")

Orientation 1:

disp("θ1="+theta1\_1)

θ1=143.1301

disp("θ2="+theta2\_1)

θ2=147.2651

disp("θ3="+theta3\_1)

θ3=35.8657

disp("Orientation 2:")

Orientation 2:

disp("θ1="+theta1\_2)

θ1=143.1301

disp("θ2="+theta2\_2)

θ2=176.8692

disp("θ3="+theta3\_2)

θ3=35.8657

function T = getMat(a, alp, d, the)

T = [cos(the), -sin(the), 0, a;

cos(alp)\*sin(the), cos(alp)\*cos(the), -sin(alp), -d\*sin(alp);

sin(alp)\*sin(the), sin(alp)\*cos(the), cos(alp), d\*cos(alp);

0, 0, 0, 1];

end

x-x-x