# Robot Groebner Basis Sage code

June 17, 2022

[22]:
```
#Determine the Groebner basis if we assume a=0
R.<c1,s1,c2,s2,c3,s3,l4,b,c,s> = PolynomialRing(RR, order = 'lex')
# s is sine of theta
# c is cosine of theta
# b is vertical y-coordinate given a=0

J = ideal(c1*c2*c3-s1*s2*c3-s1*c2*s3-c1*s2*s3-c,\
          s1*c2*c3+c1*s2*c3+c1*c2*s3-s1*s2*s3-s,\
          c1^2+s1^2-1, c2^2+s2^2-1, c3^2+s3^2-1,\
          l4*(s1*c2*c3+c1*s2*c3+c1*c2*s3-s1*s2*s3)+c1*s2+s1*c2+s1,\
          l4*(c1*c2*c3-c1*s2*s3-s1*c2*s3-s1*s2*c3)+c1*c2-s1*s2+c1-b)

C = J.groebner_basis()
print("If a=0 the Groebner basis is equal to")
for x in C:
    print(x)
    print(" ")
```

```
If a=0 the Groebner basis is equal to
c1 + c3*c + s3*s + l4*c - b

s1 + c3*s - s3*c + l4*s

c2 - 0.500000000000000*l4^2 + l4*b*c - 0.500000000000000*b^2 + 1.00000000000000

s2 - c3*b*s - s3*l4 + s3*b*c

c3^2 + s3^2 - 1.00000000000000

c3*s3*b*s - 0.500000000000000*c3*b^2*s^2 + s3^2*l4 - s3^2*b*c -
0.500000000000000*s3*l4*b*s + 0.500000000000000*s3*b^2*c*s +
0.250000000000000*l4^3 - 0.750000000000000*l4^2*b*c -
0.500000000000000*l4*b^2*s^2 + 0.750000000000000*l4*b^2 - l4 -
0.250000000000000*b^3*c + b*c

c3*l4 - c3*b*c - s3*b*s + 0.500000000000000*l4^2 - l4*b*c +
0.500000000000000*b^2
```

```
s3^2*l4^2 - 2.00000000000000*s3^2*l4*b*c + s3^2*b^2 - s3*l4^2*b*s +
2.00000000000000*s3*l4*b^2*c*s - s3*b^3*s + 0.250000000000000*l4^4 - l4^3*b*c -
l4^2*b^2*s^2 + 1.50000000000000*l4^2*b^2 - l4^2 - l4*b^3*c +
2.00000000000000*l4*b*c + 0.250000000000000*b^4 + b^2*s^2 - b^2

c^2 + s^2 - 1.00000000000000
```

[22]:
```
s3^2*l4^2 - 2.00000000000000*s3^2*l4*b*c + s3^2*b^2 - s3*l4^2*b*s +
2.00000000000000*s3*l4*b^2*c*s - s3*b^3*s + 0.250000000000000*l4^4 - l4^3*b*c -
l4^2*b^2*s^2 + 1.50000000000000*l4^2*b^2 - l4^2 - l4*b^3*c +
2.00000000000000*l4*b*c + 0.250000000000000*b^4 + b^2*s^2 - b^2
```

[28]:
```
#Express c2^2+s2^2=1 as a combination of polynomials in our Groebner Basis
x = C[5]
y = C[6]
z = C[7]

R.<c1,s1,c3,s3,l4,a,b,c,s> = PolynomialRing(RR, order = 'lex')
c2 = 0.5*l4^2-l4*b*c+0.5*b^2-1
s2 = c3*b*s+s3*l4-s3*b*c

(c2^2␣
 ↪+s2^2-1)-(c3^2+s3^2-1)*b^2*s^2+x*(-2*l4+2*b*c)-y*(b^2*s^2)+z+(c^2+s^2-1)*(s3^2*b^2-s3*b^3*s+0
 ↪5*l4^2*b^2+0.5*b^4-2*b^2)
```

[28]:
```
0
```

[26]:
```
#Express c1^2+s2^1=1 as a combination of polynomials in our Groebner Basis
R.<c1,s1,c2,s2,c3,s3,l4,a,b,c,s> = PolynomialRing(RR, order = 'lex')
c1 = -c3*c-s3*s-l4*c+b
s1 = -c3*s+s3*c-l4*s

(c1^2+s1^2-1)-(c^2+s^2)*(c3^2+s3^2-1)-(c^2+s^2-1)*(2*c3*l4+l4^2+1)-2*y
```

[26]:
```
0
```

[41]:
```
#Determine the Discriminant of the last polynomial in our Groebner Basis
R.<s3,l4,b,c,s> = PolynomialRing(RR, order = 'lex')
s3= polygen(R)
z = s3^2*l4^2 - 2.00000000000000*s3^2*l4*b*c + s3^2*b^2 - s3*l4^2*b*s + 2.
 ↪00000000000000*s3*l4*b^2*c*s - s3*b^3*s + 0.250000000000000*l4^4 - l4^3*b*c -␣
 ↪l4^2*b^2*s^2 + 1.50000000000000*l4^2*b^2 - l4^2 - l4*b^3*c + 2.
 ↪00000000000000*l4*b*c + 0.250000000000000*b^4 + b^2*s^2 - b^2
z.discriminant()
```

[41]:
```
-l4^6 + 6.00000000000000*l4^5*b*c - 8.00000000000000*l4^4*b^2*c^2 +
5.00000000000000*l4^4*b^2*s^2 - 7.00000000000000*l4^4*b^2 +
```

```
4.00000000000000*l4^4 - 12.0000000000000*l4^3*b^3*c*s^2 +
20.0000000000000*l4^3*b^3*c - 16.0000000000000*l4^3*b*c +
4.00000000000000*l4^2*b^4*c^2*s^2 - 8.00000000000000*l4^2*b^4*c^2 +
6.00000000000000*l4^2*b^4*s^2 - 7.00000000000000*l4^2*b^4 +
16.0000000000000*l4^2*b^2*c^2 - 4.00000000000000*l4^2*b^2*s^2 +
8.00000000000000*l4^2*b^2 - 4.00000000000000*l4*b^5*c*s^2 +
6.00000000000000*l4*b^5*c + 8.00000000000000*l4*b^3*c*s^2 -
16.0000000000000*l4*b^3*c + b^6*s^2 - b^6 - 4.00000000000000*b^4*s^2 +
4.00000000000000*b^4
```

[45]:
```
#Determine the Groebner Basis in the case that theta=theta_1+theta_2+theta_3=0
R.<c1,s1,c2,s2,c3,s3,l4,a,b> = PolynomialRing(RR, order = 'lex')
# s is sine of theta
# c is cosine of theta
# b is vertical y-coordinate given a=0

J = ideal(c1*c2*c3-s1*s2*c3-s1*c2*s3-c1*s2*s3-1,\
          s1*c2*c3+c1*s2*c3+c1*c2*s3-s1*s2*s3-0,\
          c1^2+s1^2-1, c2^2+s2^2-1, c3^2+s3^2-1,\
          l4*(s1*c2*c3+c1*s2*c3+c1*c2*s3-s1*s2*s3)+c1*s2+s1*c2+s1+a,\
          l4*(c1*c2*c3-c1*s2*s3-s1*c2*s3-s1*s2*c3)+c1*c2-s1*s2+c1-b)

C = J.groebner_basis()
print("If theta_1+theta_2+theta_3 = 0, Groebner basis is equal to")
for x in C:
    print(x)
    print(" ")
```

```
If theta_1+theta_2+theta_3 = 0, Groebner basis is equal to
c1 + c3 + l4 - b

s1 - s3 + a

c2 - 0.500000000000000*l4^2 + l4*b - 0.500000000000000*a^2 -
0.500000000000000*b^2 + 1.00000000000000

s2 - c3*a - s3*l4 + s3*b

c3^2 + s3^2 - 1.00000000000000

c3*s3*a - 0.500000000000000*c3*a^2 + s3^2*l4 - s3^2*b -
0.500000000000000*s3*l4*a + 0.500000000000000*s3*a*b + 0.250000000000000*l4^3 -
0.750000000000000*l4^2*b + 0.250000000000000*l4*a^2 + 0.750000000000000*l4*b^2 -
l4 - 0.250000000000000*a^2*b - 0.250000000000000*b^3 + b

c3*l4 - c3*b - s3*a + 0.500000000000000*l4^2 - l4*b + 0.500000000000000*a^2 +
0.500000000000000*b^2
```

```
s3^2*l4^2 - 2.00000000000000*s3^2*l4*b + s3^2*a^2 + s3^2*b^2 - s3*l4^2*a +
2.00000000000000*s3*l4*a*b - s3*a^3 - s3*a*b^2 + 0.250000000000000*l4^4 - l4^3*b
+ 0.500000000000000*l4^2*a^2 + 1.50000000000000*l4^2*b^2 - l4^2 - l4*a^2*b -
l4*b^3 + 2.00000000000000*l4*b + 0.250000000000000*a^4 +
0.500000000000000*a^2*b^2 + 0.250000000000000*b^4 - b^2
```

[44]:
```
#Determine the discriminant of the last polynomial which is a quadratic equation␣
↪is s3
R.<s3,l4,a,b> = PolynomialRing(RR, order = 'lex')
s3 = polygen(R)
y = s3^2*l4^2 - 2.00000000000000*s3^2*l4*b + s3^2*a^2 + s3^2*b^2 - s3*l4^2*a + 2.
↪00000000000000*s3*l4*a*b - s3*a^3 - s3*a*b^2 + 0.250000000000000*l4^4 - l4^3*b␣
↪+ 0.500000000000000*l4^2*a^2 + 1.50000000000000*l4^2*b^2 - l4^2 - l4*a^2*b -␣
↪l4*b^3 + 2.00000000000000*l4*b + 0.250000000000000*a^4 + 0.
↪500000000000000*a^2*b^2 + 0.250000000000000*b^4 - b^2
f = y.discriminant()
#Trying to factor the discriminant.
g= a^2+(l4-b)^2-4
f.quo_rem(g)
```

[44]:
```
(-l4^4 + 4.00000000000000*l4^3*b - l4^2*a^2 - 6.00000000000000*l4^2*b^2 +
2.00000000000000*l4*a^2*b + 4.00000000000000*l4*b^3 - a^2*b^2 - b^4,
 0)
```

[113]:
```
#Factorising the discriminant even further
h = -l4^4 + 4.00000000000000*l4^3*b - l4^2*a^2 - 6.00000000000000*l4^2*b^2 + 2.
↪00000000000000*l4*a^2*b + 4.00000000000000*l4*b^3 - a^2*b^2 - b^4
p = a^2+(l4-b)^2
h.quo_rem(p)
```

[113]:
```
(-l4^2 + 2.00000000000000*l4*b - b^2, 0)
```

[46]:
```
#What happens if all the lengths are variable?
R.<c1,s1,c2,s2,c3,s3,l3,l2,l4,a,b,c,s> = PolynomialRing(RR, order = 'lex')
# s is sine of theta
# c is cosine of theta
# b is vertical y-coordinate given a=0

J = ideal(c1*c2*c3-s1*s2*c3-s1*c2*s3-c1*s2*s3-1,\
          s1*c2*c3+c1*s2*c3+c1*c2*s3-s1*s2*s3-0,\
          c1^2+s1^2-1, c2^2+s2^2-1, c3^2+s3^2-1,\
          l4*(s1*c2*c3+c1*s2*c3+c1*c2*s3-s1*s2*s3)+l3*(c1*s2+s1*c2)+l2*s1+a,\
          l4*(c1*c2*c3-c1*s2*s3-s1*c2*s3-s1*s2*c3)+l3*(c1*c2-s1*s2)+l2*c1-b)

C = J.groebner_basis()
for x in C:
```

```
    print(x)
    print(" ")
```

```
c1 - c2*c3 + s2*s3

s1 + c2*s3 + s2*c3

c2^2 + s2^2 - 1.00000000000000

c2*c3*l3 + c2*l4 - c2*b + s2*s3*l3 - s2*a + c3*l2

c2*c3*l4 - c2*c3*b - c2*s3*a + c2*l3 - s2*c3*a - s2*s3*l4 + s2*s3*b + l2

c2*c3*a + c2*s3*l4 - c2*s3*b + s2*c3*l4 - s2*c3*b - s2*s3*a + s2*l3

c2*s3*l3 - c2*a - s2*c3*l3 - s2*l4 + s2*b + s3*l2

c2*s3*l4^2 - 2.00000000000000*c2*s3*l4*b + c2*s3*a^2 + c2*s3*b^2 - c2*l3*a +
s2*c3*l4^2 - 2.00000000000000*s2*c3*l4*b + s2*c3*a^2 + s2*c3*b^2 + s2*l3*l4 -
s2*l3*b - l2*a

c2*l3^2 - c2*l4^2 + 2.00000000000000*c2*l4*b - c2*a^2 - c2*b^2 -
2.00000000000000*s2*c3*l3*a - 2.00000000000000*s2*s3*l3*l4 +
2.00000000000000*s2*s3*l3*b - c3*l2*l4 + c3*l2*b + s3*l2*a + l3*l2

c2*l2 + c3*l4 - c3*b - s3*a + l3

s2*c3*l4^3 - 3.00000000000000*s2*c3*l4^2*b + s2*c3*l4*a^2 +
3.00000000000000*s2*c3*l4*b^2 - s2*c3*a^2*b - s2*c3*b^3 -
2.00000000000000*s2*s3^2*l3*l4^2 + 4.00000000000000*s2*s3^2*l3*l4*b -
2.00000000000000*s2*s3^2*l3*a^2 - 2.00000000000000*s2*s3^2*l3*b^2 +
2.00000000000000*s2*s3*l3^2*a + s2*s3*l4^2*a - 2.00000000000000*s2*s3*l4*a*b +
s2*s3*a^3 + s2*s3*a*b^2 - 0.500000000000000*s2*l3^3 +
1.50000000000000*s2*l3*l4^2 - 3.00000000000000*s2*l3*l4*b -
0.500000000000000*s2*l3*a^2 + 1.50000000000000*s2*l3*b^2 - c3*s3*l2*l4^2 +
2.00000000000000*c3*s3*l2*l4*b - c3*s3*l2*a^2 - c3*s3*l2*b^2 +
0.500000000000000*c3*l3*l2*a + 0.500000000000000*s3*l3*l2*l4 -
0.500000000000000*s3*l3*l2*b - l2*l4*a + l2*a*b

s2*l2 - c3*a - s3*l4 + s3*b

c3^2 + s3^2 - 1.00000000000000

c3*s3*l3*a - 0.500000000000000*c3*l3^2 + 0.500000000000000*c3*l2^2 -
0.500000000000000*c3*l4^2 + c3*l4*b - 0.500000000000000*c3*a^2 -
0.500000000000000*c3*b^2 + s3^2*l3*l4 - s3^2*l3*b - l3*l4 + l3*b

c3*l3*l4 - c3*l3*b - s3*l3*a + 0.500000000000000*l3^2 - 0.500000000000000*l2^2 +
```

```
0.500000000000000*l4^2 - l4*b + 0.500000000000000*a^2 + 0.500000000000000*b^2

c3*l2^2*l4 - c3*l2^2*b - c3*l4^3 + 3.00000000000000*c3*l4^2*b - c3*l4*a^2 -
3.00000000000000*c3*l4*b^2 + c3*a^2*b + c3*b^3 + 2.00000000000000*s3^2*l3*l4^2 -
4.00000000000000*s3^2*l3*l4*b + 2.00000000000000*s3^2*l3*a^2 +
2.00000000000000*s3^2*l3*b^2 - 2.00000000000000*s3*l3^2*a + s3*l2^2*a -
s3*l4^2*a + 2.00000000000000*s3*l4*a*b - s3*a^3 - s3*a*b^2 +
0.500000000000000*l3^3 - 0.500000000000000*l3*l2^2 - 1.50000000000000*l3*l4^2 +
3.00000000000000*l3*l4*b + 0.500000000000000*l3*a^2 - 1.50000000000000*l3*b^2

s3^2*l3^2*l4^2 - 2.00000000000000*s3^2*l3^2*l4*b + s3^2*l3^2*a^2 + s3^2*l3^2*b^2
- s3*l3^3*a + s3*l3*l2^2*a - s3*l3*l4^2*a + 2.00000000000000*s3*l3*l4*a*b -
s3*l3*a^3 - s3*l3*a*b^2 + 0.250000000000000*l3^4 - 0.500000000000000*l3^2*l2^2 -
0.500000000000000*l3^2*l4^2 + l3^2*l4*b + 0.500000000000000*l3^2*a^2 -
0.500000000000000*l3^2*b^2 + 0.250000000000000*l2^4 -
0.500000000000000*l2^2*l4^2 + l2^2*l4*b - 0.500000000000000*l2^2*a^2 -
0.500000000000000*l2^2*b^2 + 0.250000000000000*l4^4 - l4^3*b +
0.500000000000000*l4^2*a^2 + 1.50000000000000*l4^2*b^2 - l4*a^2*b - l4*b^3 +
0.250000000000000*a^4 + 0.500000000000000*a^2*b^2 + 0.250000000000000*b^4
```

[49]:
```
R.<s3,l3,l2,l4,a,b,c,s> = PolynomialRing(RR, order = 'lex')
s3 = polygen(R)
y = s3^2*l3^2*l4^2 - 2.00000000000000*s3^2*l3^2*l4*b + s3^2*l3^2*a^2 + s3^2*l3^2*b^2 - s3*l3^3*a + s3*l3*l2^2*a - s3*l3*l4^2*a + 2.00000000000000*s3*l3*l4*a*b - s3*l3*a^3 - s3*l3*a*b^2 + 0.250000000000000*l3^4 - 0.500000000000000*l3^2*l2^2 - 0.500000000000000*l3^2*l4^2 + l3^2*l4*b + 0.500000000000000*l3^2*a^2 - 0.500000000000000*l3^2*b^2 + 0.250000000000000*l2^4 - 0.500000000000000*l2^2*l4^2 + l2^2*l4*b - 0.500000000000000*l2^2*a^2 - 0.500000000000000*l2^2*b^2 + 0.250000000000000*l4^4 - l4^3*b + 0.500000000000000*l4^2*a^2 + 1.50000000000000*l4^2*b^2 - l4*a^2*b - l4*b^3 + 0.250000000000000*a^4 + 0.500000000000000*a^2*b^2 + 0.250000000000000*b^4
y.discriminant()
```

[49]:
```
-l3^6*l4^2 + 2.00000000000000*l3^6*l4*b - l3^6*b^2 +
2.00000000000000*l3^4*l2^2*l4^2 - 4.00000000000000*l3^4*l2^2*l4*b +
2.00000000000000*l3^4*l2^2*b^2 + 2.00000000000000*l3^4*l4^4 -
8.00000000000000*l3^4*l4^3*b + 2.00000000000000*l3^4*l4^2*a^2 +
12.0000000000000*l3^4*l4^2*b^2 - 4.00000000000000*l3^4*l4*a^2*b -
8.00000000000000*l3^4*l4*b^3 + 2.00000000000000*l3^4*a^2*b^2 +
2.00000000000000*l3^4*b^4 - l3^2*l2^4*l4^2 + 2.00000000000000*l3^2*l2^4*l4*b -
l3^2*l2^4*b^2 + 2.00000000000000*l3^2*l2^2*l4^4 -
8.00000000000000*l3^2*l2^2*l4^3*b + 2.00000000000000*l3^2*l2^2*l4^2*a^2 +
12.0000000000000*l3^2*l2^2*l4^2*b^2 - 4.00000000000000*l3^2*l2^2*l4*a^2*b -
8.00000000000000*l3^2*l2^2*l4*b^3 + 2.00000000000000*l3^2*l2^2*a^2*b^2 +
2.00000000000000*l3^2*l2^2*b^4 - l3^2*l4^6 + 6.00000000000000*l3^2*l4^5*b -
2.00000000000000*l3^2*l4^4*a^2 - 15.0000000000000*l3^2*l4^4*b^2 +
```

```
8.00000000000000*l3^2*l4^3*a^2*b + 20.0000000000000*l3^2*l4^3*b^3 -
l3^2*l4^2*a^4 - 12.0000000000000*l3^2*l4^2*a^2*b^2 -
15.0000000000000*l3^2*l4^2*b^4 + 2.00000000000000*l3^2*l4*a^4*b +
8.00000000000000*l3^2*l4*a^2*b^3 + 6.00000000000000*l3^2*l4*b^5 - l3^2*a^4*b^2 -
2.00000000000000*l3^2*a^2*b^4 - l3^2*b^6
```