NB.07.F8

July 23, 2024

1 Examples of each configuration of alignments

[12]:
```
load("basic_functions.sage")
```

2 Configuration $(C__1)

2.1 Three random aligned points:

[13]:
```
P1 = vector(S, (A1, B1, C1))
P2 = vector(S, (A2, B2, C2))
P3 = u1*P1 + u2*P2

rnd_exmp = {S(a): small_random() for a in [A1, B1, C1, A2, B2, C2, u1, u2]}

p1 = P1.subs(rnd_exmp)
p2 = P2.subs(rnd_exmp)
p3 = P3.subs(rnd_exmp)
```

The condition matrix has rank 6

[14]:
```
M = condition_matrix([p1, p2, p3], S, standard="all")
assert(M.rank() == 6)
```

[15]:
```
M = M.matrix_from_rows([0, 1, 3, 4, 6, 7])
assert(M.rank() == 6)
```

Construction of a random cubic with p1, p2, p3 eigenpoints:

[16]:
```
M = M.stack(vector(S, [small_random() for _ in range(10)]))
M = M.stack(vector(S, [small_random() for _ in range(10)]))
M = M.stack(vector(S, [small_random() for _ in range(10)]))
```

[17]:
```
assert(M.rank() == 9)

M = M.stack(vector(S, mon))
```

The cubic with eigenpoints p_1, p_2, p_3:

[18]:
```
cb = M.det()
```

[19]:
```
Ecb = eig(cb)
```

[20]:
```
assert(Ecb.subs(substitution(p1)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p2)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p3)) == vector(S, (0, 0, 0)))
```

2.2 Three points on a line tangent to Ciso in P_1

[21]:
```
P1 = vector(S, (1, ii, 0))
P2 = vector(S, (A2, ii*A2, C2))
P3 = u1*P1 + u2*P2

rnd_exmp = {S(a): small_random() for a in [A2, C2, u1, u2]}

p1 = P1.subs(rnd_exmp)
p2 = P2.subs(rnd_exmp)
p3 = P3.subs(rnd_exmp)

M = condition_matrix([p1, p2, p3], S, standard="all")
assert(M.rank() == 5)
```

[22]:
```
M = M.matrix_from_rows([0, 1, 3, 4, 7])
assert(M.rank() == 5)
```

Construction of a generic cubic with p_1, p_2, p_3 eigenpoints:

[23]:
```
M = M.stack(vector(S, [small_random() for _ in range(10)]))
M = M.stack(vector(S, [small_random() for _ in range(10)]))
M = M.stack(vector(S, [small_random() for _ in range(10)]))
M = M.stack(vector(S, [small_random() for _ in range(10)]))
```

[24]:
```
assert(M.rank() == 9)

M = M.stack(vector(S, mon))
```

[25]:
```
cb = M.det()
Ecb = eig(cb)
```

[26]:
```
assert(Ecb.subs(substitution(p1)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p2)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p3)) == vector(S, (0, 0, 0)))
```

3 Configuration (C_2)

3.1 Case $\delta_1(P_1, P_2, P_4) = 0$ and rank condition matrix 9:

Construction of five random points

[27]:
```
P1 = vector(S, (A1, B1, C1))
P2 = vector(S, (A2, B2, C2))

Q = scalar_product(P1, P1)*P2-scalar_product(P1, P2)*P1
P4 = vector(S, (A4*Q[2], B4*Q[2], -Q[0]*A4-Q[1]*B4))

assert(scalar_product(Q, P4) == 0)

assert(delta1(P1, P2, P4) == 0)

P3 = u1*P1 + u2*P2
P5 = v1*P1 + v2*P4

rnd_exam = {S(a): small_random() for a in [A1, B1, C1, A2, B2, C2, A4, B4, u1,␣
↪u2, v1, v2]}

p1 = P1.subs(rnd_exam)
p2 = P2.subs(rnd_exam)
p3 = P3.subs(rnd_exam)
p4 = P4.subs(rnd_exam)
p5 = P5.subs(rnd_exam)
```

The condition matrix has rank 9:

[28]:
```
M = condition_matrix([p1, p2, p3, p4, p5], S, standard = "all")
assert(M.rank() == 9)
```

Construction of a random cubic

[ ]:
```
M = M.matrix_from_rows([0, 1, 3, 4, 6, 7, 9, 10, 12])
assert(M.rank() == 9)

M = M.stack(vector(S, mon))

cb = M.det()
```

the random cubic cb has the expected eigenpoints

[237]:
```
Ecb = eig(cb)
```

[238]:
```
assert(Ecb.subs(substitution(p1)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p2)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p3)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p4)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p5)) == vector(S, (0, 0, 0)))
```

3.1.1 The seven eigenpoints:

[226]:
```
pd = S.ideal(list(Ecb)).radical().primary_decomposition()
```

3.2 Case condition matrix of rank 8

Hence $\delta_1(P_1, P_2, P_4) = 0$, $\overline{\delta}_1(P_1, P_2, P_3) = 0$, $\overline{\delta}_1(P_1, P_4, P_5) = 0$

[227]:
```
d1 = delta1b(P1, P2, P3)
P3 = P3.subs({u1: d1.coefficient(u2), u2: -d1.coefficient(u1)})

assert(delta1b(P1, P2, P3) == 0)
```

[ ]:
```
d1 = delta1b(P1, P4, P5)
P5 = P5.subs({v1: d1.coefficient(v2), v2: -d1.coefficient(v1)})

assert(delta1b(P1, P4, P5) == 0)
```

Construction of 5 points which satisfy zthe three deltas

[ ]:
```
rnd_exam = {S(a): small_random() for a in [A1, B1, C1, A2, B2, C2, A4, B4]}

p1 = P1.subs(rnd_exam)
p2 = P2.subs(rnd_exam)
p3 = P3.subs(rnd_exam)
p4 = P4.subs(rnd_exam)
p5 = P5.subs(rnd_exam)
```

The condition matrix (of rank 8)

[ ]:
```
M = condition_matrix([p1, p2, p3, p4, p5], S, standard= "all")

assert(M.rank() == 8)
```

Construction of a random cubic

[ ]:
```
M = M.matrix_from_rows([0, 1, 3, 4, 6, 7, 9, 10])
assert(M.rank() == 8)

M = M.stack(vector(S, [small_random() for _ in range(10)]))
M = M.stack(vector(S, mon))

cb = M.det()
```

The cubic cb has the expected eigenpoints

[234]:
```
Ecb = eig(cb)
```

[235]:
```
assert(Ecb.subs(substitution(p1)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p2)) == vector(S, (0, 0, 0)))
```

```
assert(Ecb.subs(substitution(p3)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p4)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p5)) == vector(S, (0, 0, 0)))
```

[236]:
```
pd = S.ideal(list(Ecb)).radical().primary_decomposition()
```

4 Configuration (C_3)

4.1 General case:

4.2 $P_3 = (s_{14}s_{15}s_{22} - s_{12}^2 s_{45})P_1 + s_{12}(s_{11}s_{45} - s_{14}s_{15})P_2$

[247]:
```
P1 = vector(S, (A1, B1, C1))
P2 = vector(S, (A2, B2, C2))
P4 = vector(S, (A4, B4, C4))
P5 = v1*P1+v2*P4

U1 = scalar_product(P1, P4)*scalar_product(P1, P5)*scalar_product(P2,␣
↪P2)-scalar_product(P1, P2)^2*scalar_product(P4, P5)
U2 = scalar_product(P1, P2)*(scalar_product(P1, P1)*scalar_product(P4,␣
↪P5)-scalar_product(P1, P4)*scalar_product(P1, P5))

P3 = U1*P1+U2*P2

assert(delta2(P1, P2, P3, P4, P5) == 0)

rnd_exam = {S(a): small_random() for a in [A1, B1, C1, A2, B2, C2, A4, B4, C4,␣
↪v1, v2]}

p1 = P1.subs(rnd_exam)
p2 = P2.subs(rnd_exam)
p3 = P3.subs(rnd_exam)
p4 = P4.subs(rnd_exam)
p5 = P5.subs(rnd_exam)
```

The conditioni matrix (of rank 9)

[248]:
```
M = condition_matrix([p1, p2, p3, p4, p5], S, standard= "all")

assert(M.rank() == 9)
```

Construction of a random cubic

[250]:
```
M = M.matrix_from_rows([0, 1, 3, 4, 6, 7, 9, 10, 12])
assert(M.rank() == 9)

M = M.stack(vector(S, mon))
```

```
cb = M.det()
```

[251]:
```
Ecb = eig(cb)
```

The cubic has the expected eigenpoints

[252]:
```
assert(Ecb.subs(substitution(p1)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p2)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p3)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p4)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p5)) == vector(S, (0, 0, 0)))
```

[268]:
```
pd = S.ideal(list(Ecb)).radical().primary_decomposition()
```

pd should be given by 6 ideals, the first (i.e. pd[0]) should be the ideal of two points (p6, p7)

[269]:
```
assert(len(pd) == 6)
```

[272]:
```
# pd[0] has a polynomial in x, y, z of degree 2
assert(2 in [pd[0].gens()[1].degree(x), pd[0].gens()[1].degree(y), pd[0].
 ↪gens()[1].degree(z)])
```

The points p6 and p7 are aligned with p1. We see this since the first generator of pd[0] is a line passing through p1

[273]:
```
assert(pd[0].gens()[0].subs(substitution(p1)) == 0)
```

4.3 Case $s_{12} = 0$, $s_{14} = 0$

[329]:
```
P1 = vector(S, (A1, B1, C1))
P2 = vector(S, (A2*C1, B2*C1, -A1*A2-B1*B2))
P4 = vector(S, (A4*C1, B4*C1, -A1*A4-B1*B4))

P3 = u1*P1+u2*P2
P5 = v1*P1+v2*P4

assert(scalar_product(P1, P2) == 0)
assert(scalar_product(P1, P4) == 0)

assert(matrix([P1, wedge_product(P2, P4)]).minors(2) == [0, 0, 0])

assert(delta2(P1, P2, P3, P4, P5) == 0)

rnd_exam = {S(a): small_random() for a in [A1, B1, C1, A2, B2, A4, B4, u1, u2,␣
 ↪v1, v2]}

p1 = P1.subs(rnd_exam)
p2 = P2.subs(rnd_exam)
p3 = P3.subs(rnd_exam)
```

```
p4 = P4.subs(rnd_exam)
p5 = P5.subs(rnd_exam)
```

The condition matrix has rank 9

```
[330]: M = condition_matrix([p1, p2, p3, p4, p5], S, standard= "all")

assert(M.rank() == 9)
```

construction of the cubic

```
[331]: M = M.matrix_from_rows([0, 1, 3, 4, 6, 7, 9, 10, 12])
assert(M.rank() == 9)

M = M.stack(vector(S, mon))

cb = M.det()
```

```
[332]: Ecb = eig(cb)
```

the five points are eigenpoints

```
[333]: assert(Ecb.subs(substitution(p1)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p2)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p3)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p4)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p5)) == vector(S, (0, 0, 0)))
```

```
[334]: pd = S.ideal(list(Ecb)).radical().primary_decomposition()
```

pd should be given by 7 ideals. In addition to $p_1, \dots, p_5$ we have p_6 and p_7

```
[335]: assert(len(pd) == 7)
```

We select the points p_6 and p_7 and we verify that p_1, p_6, p_7 are aligned:

```
[336]: J = S.ideal(list(Ecb))
J = J.saturation(S.ideal(matrix([(x, y, z), p1]).minors(2)))[0]
J = J.saturation(S.ideal(matrix([(x, y, z), p2]).minors(2)))[0]
J = J.saturation(S.ideal(matrix([(x, y, z), p3]).minors(2)))[0]
J = J.saturation(S.ideal(matrix([(x, y, z), p4]).minors(2)))[0]
J = J.saturation(S.ideal(matrix([(x, y, z), p5]).minors(2)))[0]

pd67 = J.radical().primary_decomposition()
assert(len(pd67) == 2)

p6 = vector(S, (pd67[0].reduce(x), pd67[0].reduce(y), pd67[0].reduce(z))).
  ↪subs({x:1, y:1, z:1})
p7 = vector(S, (pd67[1].reduce(x), pd67[1].reduce(y), pd67[1].reduce(z))).
  ↪subs({x:1, y:1, z:1})
```

```
assert(matrix([p1, p6, p7]).det() == 0)
```

Here we have four collinearities:

$$[(1,2,3),(1,4,5),(1,6,7),(2,4,6)] \quad \text{or} \quad [(1,2,3),(1,4,5),(1,6,7),(2,4,7)]$$

[31]:
```
assert(
    alignments([p1, p2, p3, p4, p5, p6, p7]) in
    [
        [(1, 2, 3), (1, 4, 5), (1, 6, 7), (2, 4, 6)],
        [(1, 2, 3), (1, 4, 5), (1, 6, 7), (2, 4, 7)]
    ]
)
```

The configuration is (C_5) as should be, since $p_1 = p_2 \times p_4$

4.4 Case $s_{12} = 0$, $s_{22} = 0$

[358]:
```
P2 = vector(S, (1, ii, 0))
P1 = vector(S, (A1, ii*A1, C1))

P4 = vector(S, (A4, B4, C4))
P3 = u1*P1 + u2*P2
P5 = v1*P1+v2*P4

assert(scalar_product(P1, P2) == 0)
assert(scalar_product(P2, P2) == 0)

assert(delta2(P1, P2, P3, P4, P5) == 0)

rnd_exam = {S(a): small_random() for a in [A1, C1, A4, B4, C4, u1, u2, v1, v2]}

p1 = P1.subs(rnd_exam)
p2 = P2.subs(rnd_exam)
p3 = P3.subs(rnd_exam)
p4 = P4.subs(rnd_exam)
p5 = P5.subs(rnd_exam)
```

The condition matrix has rank 9

[359]:
```
M = condition_matrix([p1, p2, p3, p4, p5], S, standard= "all")

assert(M.rank() == 9)
```

construction of the cubic:

[360]:
```
M = M.matrix_from_rows([0, 1, 4, 6, 7, 9, 10, 12, 13])
assert(M.rank() == 9)
```

```
M = M.stack(vector(S, mon))

cb = M.det()
```

The five points are eigenpoints

[366]:
```
Ecb = eig(cb)
```

[367]:
```
assert(Ecb.subs(substitution(p1)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p2)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p3)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p4)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p5)) == vector(S, (0, 0, 0)))
```

[368]:
```
pd = S.ideal(list(Ecb)).radical().primary_decomposition()
```

pd should be given by 6 ideals, the first (i.e. pd[0]) should be the ideal of two points (p6, p7)

[372]:
```
assert(len(pd) == 6)

# pd[0] has a polynomial in x, y, z of degree 2
assert(2 in [pd[0].gens()[1].degree(x), pd[0].gens()[1].degree(y), pd[0].
 ↪gens()[1].degree(z)])

# the two points given by pd[0] are aligned with p1 (the first generator of␣
 ↪pd[0] is a line thorugh p1)

assert(pd[0].gens()[0].subs(substitution(p1)) == 0)
```

No other collinearities among the seven points are possible (since pd[0] is a prime ideal)

4.5 Case $\sigma(P_1, P_2) = 0$ and $\sigma(P_1, P_4) = 0$

[379]:
```
P1 = vector(S, (1, 0, 0))
Qa = vector(S, (0, 1, ii))
Qb = vector(S, (0, 1, -ii))

P2 = m1*P1 + m2*Qa
P4 = l1*P1 + l2*Qb

assert(sigma(P1, P2) == 0)
assert(sigma(P1, P4) == 0)

P3 = u1*P1 + u2*P2
P5 = v1*P1 + v2*P4

assert(delta2(P1, P2, P3, P4, P5) == 0)
```

```
rnd_exam = {S(a): small_random() for a in [m1, m2, l1, l2, u1, u2, v1, v2]}

p1 = P1.subs(rnd_exam)
p2 = P2.subs(rnd_exam)
p3 = P3.subs(rnd_exam)
p4 = P4.subs(rnd_exam)
p5 = P5.subs(rnd_exam)
```

Construction of the condition matrix

[380]:
```
M = condition_matrix([p1, p2, p3, p4, p5], S, standard= "all")

assert(M.rank() == 9)
```

Construction of the cubic

[382]:
```
M = M.matrix_from_rows([0, 1, 3, 4, 6, 7, 9, 10, 12])
assert(M.rank() == 9)

M = M.stack(vector(S, mon))

cb = M.det()
```

In this case the lines $p_1 \vee p_2$ $(y+iz)$ and $p_1 + p_4$ $(y-iz)$ are lines of eigenpoints

[395]:
```
Ecb = eig(cb)

assert(Ecb.subs(y = -ii*z)==0)
assert(Ecb.subs(y = ii*z)==0)
```

This is a case in which δ_2 is zero, but we do not have seven eigenpoints

5 Configuration (C_5)

We define $P_1 = P_2 \times P_4$ and * $P_6 = (s_{15}s_{24}s_{34}+s_{15}s_{23}s_{44} -s_{13}s_{25}s_{44} -s_{13}s_{24}s_{45}) , P_2 + (s_{13}s_{24}s_{25}-2s_{15}s_{22}s_{34}+s_{13}s_{22} s_{45}) , P_4 $ * $P_7 = (s_{16}(s_{26}s_{45}+s_{24}s_{56}) - s_{26}s_{15}s_{46} - s_{24}s_{15}s_{66})P_1 - (s_{11}(s_{26}s_{45}+s_{24}s_{56}) - s_{26}s_{15}s_{14} - s_{24}s_{15}s_{16})P_6$

[30]:
```
P2 = vector(S, (A2, B2, C2))
P4 = vector(S, (A4, B4, C4))

P1 = wedge_product(P2, P4)

P3 = u1*P1 + u2*P2
P5 = v1*P1 + v2*P4

assert(scalar_product(P1, P2) == 0)
```

```
assert(scalar_product(P1, P4) == 0)

L1 = (
    scalar_product(P1, P5)*scalar_product(P2, P4)*scalar_product(P3, P4)
    + scalar_product(P1, P5)*scalar_product(P2, P3)*scalar_product(P4, P4)
    - scalar_product(P1, P3)*scalar_product(P2, P5)*scalar_product(P4, P4)
    - scalar_product(P1, P3)*scalar_product(P2, P4)*scalar_product(P4, P5)
)

L2 = (
    scalar_product(P1, P3)*scalar_product(P2, P4)*scalar_product(P2, P5)
    - 2*scalar_product(P1, P5)*scalar_product(P2, P2)*scalar_product(P3, P4)
    + scalar_product(P1, P3)*scalar_product(P2, P2)*scalar_product(P4, P5)
)


P6 = L1*P2 + L2*P4

N1 = (
    scalar_product(P1, P6)*(scalar_product(P2, P6)*scalar_product(P4, P5)
    + scalar_product(P2, P4)*scalar_product(P5, P6))
    - scalar_product(P2, P6)*scalar_product(P1, P5)*scalar_product(P4, P6)
    - scalar_product(P2, P4)*scalar_product(P1, P5)*scalar_product(P6, P6)
)

N2 = (
    scalar_product(P1, P1)*(scalar_product(P2, P6)*scalar_product(P4, P5)
    + scalar_product(P2, P4)*scalar_product(P5, P6))
    - scalar_product(P2, P6)*scalar_product(P1, P5)*scalar_product(P1, P4)
    - scalar_product(P2, P4)*scalar_product(P1, P5)*scalar_product(P1, P6)
)

P7 = N1*P1 - N2*P6

rnd_exam = {S(a): small_random() for a in [A2, B2, C2, A4, B4, C4, u1, u2, v1, v2]}

p1 = P1.subs(rnd_exam)
p2 = P2.subs(rnd_exam)
p3 = P3.subs(rnd_exam)
p4 = P4.subs(rnd_exam)
p5 = P5.subs(rnd_exam)
p6 = P6.subs(rnd_exam)
p7 = P7.subs(rnd_exam)
```

The seven points are in a (C_5) configuration.

[6]:
```
assert(alignments([p1, p2, p3, p4, p5, p6, p7]) == [(1, 2, 3), (1, 4, 5), (1,␣
↪6, 7), (2, 4, 6)])
```

[7]:
```
M = condition_matrix([p1, p2, p3, p4, p5, p6, p7], S, standard= "all")

assert(M.rank() == 9)
```

construction of the cubic

[8]:
```
M = M.matrix_from_rows([0, 1, 3, 4, 6, 7, 9, 10, 12])

assert(M.rank() == 9)

M = M.stack(vector(S, mon))

cb = M.det()
```

[9]:
```
Ecb = eig(cb)
```

The seven points are eigenpoints

[10]:
```
assert(Ecb.subs(substitution(p1)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p2)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p3)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p4)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p5)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p6)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p7)) == vector(S, (0, 0, 0)))
```

[11]:
```
pd = S.ideal(list(Ecb)).radical().primary_decomposition()
```

[12]:
```
assert(len(pd) == 7)
```

6 Configuration (C_8)

[ ]:
```
P1 = vector(S, (A1, B1, C1))
P2 = vector(S, (A2, B2, C2))
P4 = vector(S, (A4, B4, C4))

P7 = (
    wedge_product(P1, P2)*scalar_product(P1, P4)*scalar_product(P2, P4)
    - scalar_product(P1, P2)*wedge_product(P1, P4)*scalar_product(P2, P4)
    + scalar_product(P1, P2)*scalar_product(P1, P4)*wedge_product(P2, P4)
)

P3 = intersection_lines(P1, P2, P4, P7)
P5 = intersection_lines(P1, P4, P2, P7)
P6 = intersection_lines(P1, P7, P2, P4)
```

```
rnd_exam = {S(a): small_random() for a in [A1, B1, C1, A2, B2, C2, A4, B4, C4]}

p1 = P1.subs(rnd_exam)
p2 = P2.subs(rnd_exam)
p3 = P3.subs(rnd_exam)
p4 = P4.subs(rnd_exam)
p5 = P5.subs(rnd_exam)
p6 = P6.subs(rnd_exam)
p7 = P7.subs(rnd_exam)
```

The seven points are in a (C_8) configuration

[17]:
```
assert(
    alignments([p1, p2, p3, p4, p5, p6, p7]) ==
    [(1, 2, 3), (1, 4, 5), (1, 6, 7), (2, 4, 6), (2, 5, 7), (3, 4, 7)]
)
```

The condition matrix has rank 9

[18]:
```
M = condition_matrix([p1, p2, p3, p4, p5, p6, p7], S, standard= "all")

assert(M.rank() == 9)
```

Construction of the cubic

[20]:
```
M = M.matrix_from_rows([0, 1, 3, 4, 6, 7, 9, 10, 12])
assert(M.rank() == 9)

M = M.stack(vector(S, mon))

cb = M.det()
```

[22]:
```
Ecb = eig(cb)
```

[23]:
```
assert(Ecb.subs(substitution(p1)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p2)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p3)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p4)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p5)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p6)) == vector(S, (0, 0, 0)))
assert(Ecb.subs(substitution(p7)) == vector(S, (0, 0, 0)))
```

[24]:
```
pd = S.ideal(list(Ecb)).radical().primary_decomposition()
```

[25]:
```
assert(len(pd) == 7)
```