NB.07.F3

July 23, 2024

1 Configurations (C_5)

[5]:
```
load("basic_functions.sage")
```

[6]:
```
do_long_computations = False
```

In the above computations, we always assume that, if $P_1, \dots, P_5$ are eigenpoints in a V-configuration, then $\Phi(P_1, \dots, P_5)$ has rank 9 and not 8 (the case rank 8 is studied elsewhere).

In order to study a (C_5) configuration, alignments:

$(1, 2, 3), (1, 4, 5), (1, 6, 7), (2, 4, 6)$

we define 6 points such that P_3 is collinear with P_1 and P_2, P_5 is collinear with P_1 and P_4, and P_6 is collinear with P_2 and P_4. These points must be eigenpoints, so $\delta_1(P_2, P_1, P_4) = 0$, $\delta_1(P_4, P_1, P_2)$, $\delta_1(P_6, P_1, P_2) = 0$, $\delta_2(P_1, P_2, P_3, P_4, P_5) = 0$. Moreover, we can divide $\delta_2(P_6, P_1, P_2)$ by w_2.

We construct the ideal generated by these conditions and we saturate it w.r.t. conditions that do not give restrictions.

We get that $s_{14} = 0$, $s_{12} = 0$, $s_{16} = 0$, hence $P_1 = P_2 \times P_4$. We redefine therefore the points.

Below the computations:

[7]:
```
P1 = vector(S, (A1, B1, C1))
P2 = vector(S, (A2, B2, C2))
P3 = u1*P1+u2*P2
P4 = vector(S, (A4, B4, C4))
P5 = v1*P1+v2*P4
P6 = w1*P2+w2*P4

## indeed, delta1(P6, P1, P2) is divisible by w2:
assert(delta1(P6, P1, P2).quo_rem(w2)[1] == S(0))

## hence we can consider the following ideal:
J = S.ideal(delta1(P2, P1, P4), delta1(P4, P1, P2), \
delta1(P6, P1, P2).quo_rem(w2)[0], delta2(P1, P2, P3, P4, P5))

## We saturate J and we get that J is the ideal generated
## by (P1|P2) and (P1|P4):
```

```
J = J.saturation(matrix([P1, P2, P4]).det())[0]
assert(J == S.ideal(scalar_product(P1, P4), scalar_product(P1, P2)))

## Moreover, we have that s16 is in J:

assert(scalar_product(P1, P6) in J)

## So we define P1 in this way and we re-write the points:

P2 = vector(S, (A2, B2, C2))
P4 = vector(S, (A4, B4, C4))
P1 = vector(S, list(wedge_product(P2, P4)))
P3 = u1*P1+u2*P2
P5 = v1*P1+v2*P4
P6 = w1*P2+w2*P4
```

With the above points the ideal J below is zero.

We have several orthogonalities among the lines: * $P_1 \vee P_2$ orthogonal $P_4 \vee P_6$ * $P_1 \vee P_6$ orthogonal $P_2 \vee P_4$ * $P_1 \vee P_4$ orthogonal $P_2 \vee P_6$

Hence the line $P_2 \vee P_4 \vee P_6$ is orthogonal to $P_1 \vee P_2$, to $P_1 \vee P_4$ and to $P_1 \vee P_6$

[8]:
```
J = S.ideal(
    delta1(P2, P1, P4),
    delta1(P4, P1, P2),
    delta1(P6, P1, P2),
    delta2(P1, P2, P3, P4, P5)
)

assert(J == S.ideal(S.zero()))

## orthogonalities among the lines:
assert(scalar_product(wedge_product(P1, P2), wedge_product(P4, P6)) == S(0))
assert(scalar_product(wedge_product(P1, P6), wedge_product(P2, P4)) == S(0))
assert(scalar_product(wedge_product(P1, P4), wedge_product(P2, P6)) == S(0))
```

The matrix M below must have rank ≤ 9. We select (in a suitable way) one particular order 10 minor of M called Nx and we compute its determinant. If `do_long_computations` is `True`, the next block requires 8 minutes.

[9]:
```
M = condition_matrix([P1, P2, P3, P4, P5, P6], S, standard="all")

Nx = M.matrix_from_rows([0, 1, 3, 4, 6, 7, 9, 10, 12, 15])

dn_old = expand(
    (-27) * A4 * A2 * w2 * w1 * v2 * v1 * u2^2 * u1^2 * (-C2*B4 + B2*C4) *␣
 ↪(-B2*A4 + A2*B4)
```

```
    * (A2^2 + B2^2 + C2^2) * (-u1*C2*B4 + u1*B2*C4 + u2*A2) * (-C2*A4^2 - C2*B4^2 + A2*A4*C4 + B2*B4*C4)
    * (B2^2*A4^2 + C2^2*A4^2 - 2*A2*B2*A4*B4 + A2^2*B4^2 + C2^2*B4^2 - 2*A2*C2*A4*C4 - 2*B2*C2*B4*C4 + A2^2*C4^2 + B2^2*C4^2)^6
    * (-2*u2*v1*w1*A2^3*A4 - 2*u2*v1*w1*A2*B2^2*A4 - 2*u2*v1*w1*A2*C2^2*A4 + 2*u1*v2*w1*A2^2*A4^2 - 2*u2*v1*w2*A2^2*A4^2
    + u1*v2*w1*B2^2*A4^2 - u2*v1*w2*B2^2*A4^2 + u1*v2*w1*C2^2*A4^2 - u2*v1*w2*C2^2*A4^2 + 2*u1*v2*w2*A2*A4^3
    - 2*u2*v1*w1*A2^2*B2*B4 - 2*u2*v1*w1*B2^3*B4 - 2*u2*v1*w1*B2*C2^2*B4 + 2*u1*v2*w1*A2*B2*A4*B4 - 2*u2*v1*w2*A2*B2*A4*B4
    + 2*u1*v2*w2*B2*A4^2*B4 + u1*v2*w1*A2^2*B4^2 - u2*v1*w2*A2^2*B4^2 + 2*u1*v2*w1*B2^2*B4^2 - 2*u2*v1*w2*B2^2*B4^2
    + u1*v2*w1*C2^2*B4^2 - u2*v1*w2*C2^2*B4^2 + 2*u1*v2*w2*A2*A4*B4^2 + 2*u1*v2*w2*B2*B4^3 - 2*u2*v1*w1*A2^2*C2*C4
    - 2*u2*v1*w1*B2^2*C2*C4 - 2*u2*v1*w1*C2^3*C4 + 2*u1*v2*w1*A2*C2*A4*C4 - 2*u2*v1*w2*A2*C2*A4*C4 + 2*u1*v2*w2*C2*A4^2*C4
    + 2*u1*v2*w1*B2*C2*B4*C4 - 2*u2*v1*w2*B2*C2*B4*C4 + 2*u1*v2*w2*C2*B4^2*C4 + u1*v2*w1*A2^2*C4^2 - u2*v1*w2*A2^2*C4^2
    + u1*v2*w1*B2^2*C4^2 - u2*v1*w2*B2^2*C4^2 + 2*u1*v2*w1*C2^2*C4^2 - 2*u2*v1*w2*C2^2*C4^2 + 2*u1*v2*w2*A2*A4*C4^2
    + 2*u1*v2*w2*B2*B4*C4^2 + 2*u1*v2*w2*C2*C4^3)
)

if do_long_computations:
    dn = Nx.det()
else:
    dn = dn_old

assert(dn == dn_old)
```

Some factors of dn are specific of the choice of the minor of M. We consider only the last factor. In the next block we will see that it is enough.

[10]:
```
fdn = dn.factor()
ftC = fdn[-1][0]
```

In the computations below we shall show that the

rank of M is ≤ 9 iff the polynomial `ftC` is zero.

One way to see this, is to consider the ideal of all the order 10 minors of M, but this computation requires too much time. Hence we assume that the point P_1 is $(1:0:0)$ or $(1:i:0)$. In this case the computation of the ideal of all the order 10 minors of the corresponding matrix M is easy to manipulate.

1.1 Case $P_1 = (1:0:0)$

Since $s_{12} = 0$, $s_{14} = 0$, we redefine the points and we re-define the matrix M (i.e. $A_2 = 0$, $A_4 = 0$)

[11]:
```
P1 = vector(S, (1, 0, 0))
P2 = vector(S, (0, B2, C2))
P3 = u1*P1+u2*P2
P4 = vector(S, (0, B4, C4))
P5 = v1*P1+v2*P4
P6 = w1*P2+w2*P4

M1 = condition_matrix([P1, P2, P3, P4, P5, P6], S, standard="all")
```

The matrix M1 has the 0th row equals to (0,1,0,…,0) the 1st row equals to (0,0,0,0,1,0,…,0) and the 2nd row given by: (0, 0,…, 0). Hence we can extract from M1 a matrix N1 which does not have the rows 0, 1, 2 and does not have the columns 1 and 4. All the order 10 minors of M1 are 0 iff all the order 8 minors of N1 are 0.

[12]:
```
N1 = M1.matrix_from_rows_and_columns(
    [3,4,5,6,7,8,9,10,11,12,13,14,15,16,17],
    [0,2,3,5,6,7,8,9]
)
```

We see that the following rows of N1 are linearly dependent: * row 1 and row 2 * row 7 and row 8 * row 13 and row 14 * row 4 and row 5 and row 6 * row 10 and row 11 and row 12 * row 1 and row 13 and row 14

Hence, in order to compute all the order 8 minors of N1, we can skip several submatrices.

We construct therefore the list `LL1` of all the rows of 8 elements which have to be considered and we get that `LL1` contains 1362 elements:

[13]:
```
assert(N1.matrix_from_rows([0, 1]).rank() == 1)

assert(N1.matrix_from_rows([6, 7]).rank() == 1)
assert(N1.matrix_from_rows([12, 13]).rank() == 1)
assert(N1.matrix_from_rows([3, 4, 5]).rank() == 2)
assert(N1.matrix_from_rows([9, 10, 11]).rank() == 2)
assert(N1.matrix_from_rows([0, 12, 13]).rank() == 2)
```

[14]:
```
## Here we construct the list of all the rows of 8 elements
## which have to be considered:

L1 = list(Combinations(15,8))

LL1 = []
for lx in L1:
    ll = Set(lx)
    if Set([0,1]).issubset(ll) or Set([6, 7]).issubset(ll) or Set([12, 13]).
 ↪issubset(ll) or \
    Set([3, 4, 5]).issubset(ll) or Set([9, 10, 11]).issubset(ll) or Set([0, 12,␣
 ↪13]).issubset(ll):
        continue
```

```
    else:
        LL1.append(lx)

## LL1 contains 1362 elements:
assert(len(LL1) == 1362)
```

The polynomial `ftC` constructed above should appear in the factors of the determinant of the order 8 minors of N_1 (when specialized with the condition $A_2 = 0$, $A_4 = 0$) We verify this and we collect the order 8 minors of N1 divided by the polynomial `ftC` specialized (called `ftCs`). About 16 seconds of computations.

[15]:
```
ftCs = ftC.subs({A2:0, A4:0})

JJ = []
for nr in LL1:
    NN = N1.matrix_from_rows(nr)
    dt = NN.det()
    dvs = dt.quo_rem(ftCs)
    if dvs[1] != 0:
        print("Unexpacted situation. The minor is not a multiple of ftCs")
        print("Do not trust to the next computations, something went wrong!")
    else:
        JJ.append(dvs[0])
```

We define the ideal generated by JJ and we saturate it. We get that $\mathrm{JJ} = (1)$, so the order 8 minors of N1 are 0 iff ftCs is zero.

[16]:
```
JJ = S.ideal(JJ)
JJ = JJ.saturation(u1*u2*v1*v2*w1*w2)[0]
JJ = JJ.saturation(S.ideal(matrix([P2, P4]).minors(2)))[0]

assert(JJ == S.ideal(S(1)))
```

Hence, in case $P_1 = (1 : 0 : 0)$, the matrix M has rank ≤ 9 iff $\mathrm{tfC} = 0$. Now the other case

1.2 Case $P_1 = (1 : i : 0)$

Then we have to consider the case in which $P_1 = (1 : i : 0)$.

But in this case we use the following result:

If Q2, Q4 are points of the plane, if Q1 = wedge_product(Q2, Q4) and if Q1 is on the isotropic conic, i.e. scalar_product(Q1,Q1) = 0, then Q1, Q2, Q4 are aligned:

[13]:
```
Q4 = vector(S, (A4, B4, C4))
Q2 = vector(S, (A2, B2, C2))
Q1 = wedge_product(Q2, Q4)
assert(det(matrix([Q1, Q2, Q4])) == scalar_product(Q1, Q1))
```

From this we get that the case $P_1 = (1 : i : 0)$ does not need to be considered.

First conclusion: configuration (C_5) is possible iff the polynomial `ftC` is zero.

1.3 Construction of the point P_6

We go back to the general case. We redefine the points

From the above computations we have that P_1 cannot be a point on the isotropic conic (indeed we sow that if P_1 is on the isotropic conic, then P_1, P_2, P_4 are collinear). Hence s_{11} is not zero.

We have that the condition `ftC`, which is the condition that implies that $P_1, P_2, P_3, P_4, P_5, P_6$ in configuration (C_5) are eigenpoints, can be expressed by:

$$\Big(\langle P_2,P_6\rangle\left(\langle P_4,P_5\rangle\ \langle P_1,P_3\rangle-\langle P_4,P_3\rangle\ \langle P_1,P_5\rangle\right)+\langle P_4,P_6\rangle\left(\langle P_2,P_5\rangle\ \langle P_1,P_3\rangle-\langle P_2,P_3\rangle\ \langle P_1,P_5\rangle\right)\Big)/\left\langle P_1,P_1\right\rangle$$

[17]:
```
P2 = vector(S, (A2, B2, C2))
P4 = vector(S, (A4, B4, C4))
P1 = vector(S, list(wedge_product(P2, P4)))
P3 = u1*P1+u2*P2
P5 = v1*P1+v2*P4
P6 = w1*P2+w2*P4
```

[18]:
```
ftC1 = (
    scalar_product(P2,P6)*(
        scalar_product(P4,P5)*scalar_product(P1, P3)
        - scalar_product(P4,P3)*scalar_product(P1, P5)
    ) + scalar_product(P4,P6)*(
        scalar_product(P2,P5)*scalar_product(P1, P3)
        - scalar_product(P2,P3)*scalar_product(P1, P5)
    )
)

assert(ftC1 == -2*scalar_product(P1, P1)*ftC)
```

since s_{11} is never zero, we have that $\mathrm{ftC} = 0$ iff $\mathrm{ftC1} = 0$.

Hence $P_1, \dots, P_6$ in config (5) are eigenpoints iff

$s_{26}(s_{45}s_{13} - s_{34}s_{15}) + s_{46}(s_{25}s_{13} - s_{23}s_{15}) = 0$

This proves the formula of configuration C5 (probably (25))

If we substitute in the expression $s_{26}(s_{45}s_{13} - s_{34}s_{15}) + s_{46}(s_{25}s_{13} - s_{23}s_{15}) = 0$ in place of P_3 the expression $u_1P_1+u_2P_2$, we get a new equation, linear in u_1 and u_2 which is equal to $u_1U_2+u_2U_1 = 0$, where $U1$ and U_2 are defined as follows:

$U_1 = s_{12}(s_{26}s_{45} + s_{46}s_{25}) - s_{26}s_{15}s_{24} - s_{46}s_{15}s_{22}$

$U_2 = s_{11}(s_{26}s_{45} + s_{46}s_{25}) - s_{26}s_{15}s_{14} - s_{46}s_{15}s_{12}$

Hence

ftC is zero iff $u_1 = U_1, u_2 = -U_2$:

[19]:
```
U1 = (
    scalar_product(P1, P2)*(
        scalar_product(P2, P6)*scalar_product(P4, P5)
        + scalar_product(P4, P6)*scalar_product(P2,P5)
    ) - scalar_product(P2, P6)*scalar_product(P1, P5)*scalar_product(P2, P4)
    - scalar_product(P4, P6)*scalar_product(P1, P5)*scalar_product(P2, P2)
)

U2 = (
    scalar_product(P1, P1)*(
        scalar_product(P2, P6)*scalar_product(P4, P5)
        + scalar_product(P4, P6)*scalar_product(P2,P5)
    ) - scalar_product(P2, P6)*scalar_product(P1, P5)*scalar_product(P1, P4)
    - scalar_product(P4, P6)*scalar_product(P1, P5)*scalar_product(P1, P2)
)

assert(ftC.subs({u1:U1, u2:-U2}) == S(0))
```

Here we see that it is not possible that U1 and U2 are zero:

[20]:
```
assert(
    S.ideal(U1, U2).saturation(matrix([P1, P2, P4]).det())[0].
  ↪saturation(u1*u2*v1*v2*w1*w2)[0] ==
    S.ideal(S.one())
)
```

The condition `ftC1 = 0` gives that in order to have 6 points in configuration (C_5) we can choose P_2 and P_4 in an arbitrary way, $P_1 = P_2 \times P_4$, P_3 on the line $P_1 \vee P_2$, P_5 on the line $P_1 vee P_4$. Then P_6 is a point on the line $P_2 \vee P_4$ determined by a linear equation in w_1 and w_2 given by `ftC1 = 0`.

In order to determine P_6, we need to find w_1 and w_2. We observe that `ftC1` is linear in w_1 and w_2

[21]:
```
assert(ftC1.degree(w1) == 1)
assert(ftC1.degree(w2) == 1)
```

and we have that the coefficient of `ftC1` w.r.t. w_1 is

$s_{13}s_{24}s_{25} - 2s_{15}s_{22}s_{34} + s_{13}s_{22}s_{45}$

and we have that the coefficient of `ftC1` w.r.t. w_2 is

$-(s_{15}s_{24}s_{34} + s_{15}s_{23}s_{44} - s_{13}s_{25}s_{44} - s_{13}s_{24}s_{45})$

To verify this, we define a substitution which sends s_{ij} to the scalar prod. of P_i and P_j

[22]:
```
sst_5 = {
    s11:scalar_product(P1, P1),
    s12:scalar_product(P1, P2),
    s22:scalar_product(P2, P2),
```

```
    s14:scalar_product(P1, P4),
    s24:scalar_product(P2, P4),
    s44:scalar_product(P4, P4),
    s13:scalar_product(P1, P3),
    s23:scalar_product(P2, P3),
    s34:scalar_product(P3, P4),
    s33:scalar_product(P3, P3),
    s45:scalar_product(P4, P5),
    s15:scalar_product(P1, P5),
    s25:scalar_product(P2, P5)
}

## Then we have:
assert(
    ftC1.coefficient(w1) ==
    (s13*s24*s25 - 2*s15*s22*s34 + s13*s22*s45).subs(sst_5)
)

assert(
    ftC1.coefficient(w2) ==
    -(s15*s24*s34 + s15*s23*s44 - s13*s25*s44 - s13*s24*s45).subs(sst_5)
)
```

Hence P_6 is:

[ ]:
```
P6 = (
    (s15*s24*s34 + s15*s23*s44 - s13*s25*s44 - s13*s24*s45)*P2
    + (s13*s24*s25 - 2*s15*s22*s34 + s13*s22*s45)*P4
)

P6 = P6.subs(sst_5)
```

Here is the formula forP_6:

$$P_6 = (s_{15}s_{24}s_{34} + s_{15}s_{23}s_{44} - s_{13}s_{25}s_{44} - s_{13}s_{24}s_{45})P_2 + (s_{13}s_{24}s_{25} - 2s_{15}s_{22}s_{34} + s_{13}s_{22}s_{45})P_4$$

and this is formula (26) (probably)

At this point we know that the matrix $\Phi(P_1, P_2, P_3, P_4, P_5, P_6)$ has rank 9, so the points $P_1, \dots, P_6$ are eigenpoints.

From the definition of P_6 we have that it holds:

$$s_{16} = 0$$

[26]:
```
assert(scalar_product(P1, P6) == 0)
```

1.4 The point P_6 is always defined.

It is not possible that the three cordinates of P_6 are all zero:

[24]:
```
assert(S.ideal(list(P6)).saturation(u1*u2*v1*v2)[0].
 ↪saturation(scalar_product(P1, P1))[0] == S.ideal(1))
```

1.5 Construction of the point P_7

In order to find P_7 we observe that it is symmetric to P_3. Hence, if in the above formula, where we define U_1 and U_2 we exchange P_2 with P_6 we get P_7:

[ ]:
```
L1 = (
    scalar_product(P1, P6)*(
        scalar_product(P6, P2)*scalar_product(P4, P5)
        + scalar_product(P4, P2)*scalar_product(P6,P5)
    ) - scalar_product(P6, P2)*scalar_product(P1, P5)*scalar_product(P6, P4)
    - scalar_product(P4, P2)*scalar_product(P1, P5)*scalar_product(P6, P6)
)

L2 = (
    scalar_product(P1, P1)*(
        scalar_product(P6, P2)*scalar_product(P4, P5)
        + scalar_product(P4, P2)*scalar_product(P6,P5)
    ) - scalar_product(P6, P2)*scalar_product(P1, P5)*scalar_product(P1, P4)
    - scalar_product(P4, P2)*scalar_product(P1, P5)*scalar_product(P1, P6)
)
```

But, since we know that $s_{1,6} = 0$ and $s_{14} = 0$, we redefine L_1, L_2 and then we define P_7

[29]:
```
L1 = (
    - scalar_product(P6, P2)*scalar_product(P1, P5)*scalar_product(P6, P4)
    - scalar_product(P4, P2)*scalar_product(P1, P5)*scalar_product(P6, P6)
)

L2 = (
    scalar_product(P1, P1)*(
        scalar_product(P6, P2)*scalar_product(P4, P5)
        + scalar_product(P4, P2)*scalar_product(P6,P5)
    )
)
```

The point P_7 is therefore defined by the formula:

$$P_7 = (s_{26}s_{15}s_{46} + s_{24}s_{15}s_{66})P_1 + s_{11}(s_{26}s_{45} + s_{24}s_{56})P_6$$

This is formula (27) (probably).

[35]:
```
P7 = -L1*P1 + L2*P6
```

P_7 can be simplified, since the components have a common factor which is never zero. Hence we redefine P_7

[36]:
```
assert(gcd(list(P7)) == scalar_product(P1, P1)^4*v1)
## Since we know that s11 is not 0, we redefine P7:
P7 = vector(S, [pp.quo_rem(gcd(list(P7)))[0] for pp in P7])
```

Partial conclusion: we have that P_7 is given by the formula (30).

1.6 P_7 is an eigenpoint of the cubic obtained from $\Phi(P_1, \dots, P_5)$

P_7 is an eigenpoints. This can be seen as follows: the rank of $\Phi(P_1, \dots, P_5)$ is 9, so $\Lambda(\Phi(P_1, \dots, P_5))$ is a unique cubic C, this cubic also has P_6 as an eigenpoint, so the rank of $\Phi(P_1, P_2, P_6, P_4, P_5)$ is 9, so from $\Lambda(P_1, P_2, P_6, P_4, P_5))$ we get a unique cubic which is again C and this cubic has the eigenpoint P_7. In this proof there are however some points that require attention (it could happen that the rank is not 9 but 8 …). In order to avoid the study of the exceptions, we give a direct proof that P_7 is an eigenpoint.

To directly prove that P_7 is an eigenpoint, we need some further computations.

In order to simplify the computations, we redefine the point with the condition that $P_2 = (1 : 0 : 0)$ or $P_2 = (1 : i : 0)$.

1.6.1 A lemma

First of all, we show that P_2 cannot be the point $(1 : i : 0)$. We define $P_2 = (1 : i : 0)$ and, since we know that $s_{12} = 0$, we define P_1 generic, orthogonal to P_2

[37]:
```
P2 = vector(S, (1, ii, 0))
P1 = vector(S, (-ii*B1, B1, C1))
assert(scalar_product(P1, P2) == 0)
```

In the next computation, we verify that $P_1 \vee P_2$ is tangent to Ciso in P_2, since the intersection of $P_1 \vee P_2$ with Ciso is the double point given by the ideal $(x + iy, z^2)$.

[38]:
```
assert(
    S.ideal(matrix([P1, P2, (x, y, z)]).det(), Ciso).saturation(S.
↪ideal(matrix([P1, P2]).minors(2)))[0] ==
    S.ideal(x+ii*y, z^2)
)
```

Hence the martix $\Phi(P_1, \dots, P_5)$ has rank 8, but this condition is excluded by our hypothesis. So the only possibility is that P_2 can be chosen as the point $(1 : 0 : 0)$.

1.6.2 We redefine the points $P_1, \dots, P_6$:

We redefine $P_1, \dots, P_5$ with the condition that $P_2 = (1 : i : 0)$and we redefine P_6 from the above formula, evaluated on the new points:

[40]:
```
P2 = vector(S, (1, 0, 0))
P4 = vector(S, (A4, B4, C4))
P1 = wedge_product(P2, P4)
P3 = u1*P1+u2*P2
P5 = v1*P1+v2*P4
```

```
P6 = (
    (s15*s24*s34 + s15*s23*s44 - s13*s25*s44 - s13*s24*s45)*P2
    + (s13*s24*s25 - 2*s15*s22*s34 + s13*s22*s45)*P4
)

sst_5b = {
    s11:scalar_product(P1, P1),
    s12:scalar_product(P1, P2),
    s22:scalar_product(P2, P2),
    s14:scalar_product(P1, P4),
    s24:scalar_product(P2, P4),
    s44:scalar_product(P4, P4),
    s13:scalar_product(P1, P3),
    s23:scalar_product(P2, P3),
    s34:scalar_product(P3, P4),
    s33:scalar_product(P3, P3),
    s45:scalar_product(P4, P5),
    s15:scalar_product(P1, P5),
    s25:scalar_product(P2, P5)
}

P6 = P6.subs(sst_5b)
```

We define the condition matrix of $P_1, \ldots, P_6$ and we verify that it has rank 9. First we extract a suitable submatrix from it which has rank 9 and does not have rows obtained from $\phi(P_6)$:

[41]:
```
M = condition_matrix([P1, P2, P3, P4, P5, P6], S, standard="all")
## The nine rows 0, 2, 3, 4, 6, 7, 9, 10, 12 of M are linearly independent:
M9 = M.matrix_from_rows([0, 2, 3, 4, 6, 7, 9, 10, 12])
assert(M9.rank() == 9)
```

Now we add to M_9 the three rows $\Phi(P_6)$, one by one and we verify that they are lin. dep. from the 9 rows of M_9 (3 seconds of computations):

[42]:
```
dtA1 = M.matrix_from_rows([0, 2, 3, 4, 6, 7, 9, 10, 12, 15]).det()
dtA2 = M.matrix_from_rows([0, 2, 3, 4, 6, 7, 9, 10, 12, 16]).det()
dtA3 = M.matrix_from_rows([0, 2, 3, 4, 6, 7, 9, 10, 12, 17]).det()

assert((dtA1, dtA2, dtA2) == (0, 0, 0))
```

1.6.3 We define P_7 in this case

We take the above cofmula for P_7 and we evaluate it on the points here defined:

[46]:
```
# P_7 = (s_{26}s_{15}s_{46}+s_{24}s_{15}s_{66})P_1 +␣
 ↪s_{11}(s_{26}s_{45}+s_{24}s_{56})P_6
```

[ ]:
```
P7 = (
    (
        scalar_product(P6, P2)*scalar_product(P1, P5)*scalar_product(P6, P4)
        +scalar_product(P4, P2)*scalar_product(P1, P5)*scalar_product(P6, P6)
    )*P1
    + scalar_product(P1, P1)*(scalar_product(P6, P2)*scalar_product(P4, P5)
    + scalar_product(P4, P2)*scalar_product(P6,P5))*P6
)
```

[49]:
```
assert(gcd(list(P7)) == scalar_product(P1, P1)^4*v1)
```

[50]:
```
## Since we know that s11 is not 0, we redefine P7:
P7 = vector(S, [p7.quo_rem(gcd(list(P7)))[0] for p7 in P7])
```

Finally, we show that the three rows of the matrix $\Phi(P_7)$ are linearly dependent of the 9 rows of M_9, hence P_7 is an eigenpoint of the unique cubic defined by $\Lambda(\Phi(P_1, \dots, P_5))$.

These computations require seconds about 180 seconds.

[51]:
```
ttA = cputime()
Phi_of_P7 = condition_matrix([P7], S, standard="all")
assert(det(M9.stack(Phi_of_P7[0])) == 0)
assert(det(M9.stack(Phi_of_P7[1])) == 0)
assert(det(M9.stack(Phi_of_P7[2])) == 0)
print("time of computation: "+str(cputime()-ttA))
```

```
time of computation: 178.981875
```

1.6.4 A final computation

Here we see that if we impose that (P_2, P_5, P_7) are aligned, then we get a (C8) configuration:

[52]:
```
#we define a list of triplets, each is given by three points that
# must not be aligned:
impossible_collin = [
    [P1, P2, P4], [P1, P2, P6], [P1, P4, P6],
    [P2, P3, P4], [P2, P4, P5], [P2, P6, P7],
    [P4, P6, P7], [P1, P2, P7], [P2, P3, P7]
]
```

[ ]:
```
## We define the condition (P2, P5, P7) aligned and we simplify it,
## erasing factors that are surely not zero:

J = S.ideal(matrix([P2, P5, P7]).det()).saturation(u1*u2*v1*v2)[0]
for tr in impossible_collin:
    J = J.saturation(matrix(tr).det())[0]
```

J is a principal ideal. Its generator has two factors. The first implies that P_3, P_4, P_7 are aligned, so we have the alignments:

(1, 2, 3), (1, 4, 5), (2, 4, 6), (2, 5, 7), (3, 4, 7)

The second factor implies the alignments:

(1, 2, 3), (1, 4, 5), (2, 4, 6), (2, 5, 7), (3, 5, 6):

[ ]:
```
pdJ = J.primary_decomposition()
assert(len(pdJ) == 2)
assert(pdJ[0].reduce(matrix([P3, P4, P7]).det()) == 0)
assert(pdJ[1].reduce(matrix([P3, P5, P6]).det()) == 0)
```

Similar computations can be done for the condition (P3, P4, P7) aligned or (P3, P5, P6) aligned.