NB.03.F6

July 23, 2024

1 Theorem

Let $P_1, \ldots, P_5$ be a V-configuration of points. Then $\operatorname{rk}\Phi(P_1, \ldots, P_5) = 8$ if and only if one of the following two conditions is satisfied: * $\delta_1(P_1, P_2, P_4) = 0$, $\overline{\delta}_1(P_1, P_2, P_3) = 0$, $\overline{\delta}_1(P_1, P_4, P_5) = 0$,; * the line $P_1 \vee P_2$ is tangent to the isotropic conic in P_2 or P_3 and the line $P_1 \vee P_4$ is tangent to the isotropic conic in P_4 or P_5; moreover, in this case we have $\delta_1(P_1, P_2, P_4) \neq 0$.

[1]:
```
load("basic_functions.sage")
```

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

We define five points, so that P_1, P_2, and P_3 are aligned and P_1, P_4, and P_5 are aligned.

[2]:
```
P1 = vector((1, 0, 0))
P2 = vector(S, (A2, B2, C2))
P4 = vector(S, (A4, B4, C4))
P3 = u1*P1+u2*P2
P5 = v1*P1+v2*P4
```

We define the matrix of conditions of $P_1, \ldots, P_5$.

[3]:
```
M = condition_matrix([P1, P2, P3, P4, P5], S, standard="all")
```

Since the first three rows of M are, respectively, $(0, 1, 0, 0, 0, 0, 0, 0, 0, 0)$, $(0, 0, 0, 0, 1, 0, 0, 0, 0, 0)$, and $(0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$, in order to compute the minors of order 9 of M, we can compute the minors of order 7 of the matrix obtained from the rows $3, 4, \ldots, 14$ of M and all the columns of M except columns 1 and 4.

[4]:
```
assert(M[0] == vector(S, (0, 1, 0, 0, 0, 0, 0, 0, 0, 0)))
assert(M[1] == vector(S, (0, 0, 0, 0, 1, 0, 0, 0, 0, 0)))
assert(M[2] == vector(S, (0, 0, 0, 0, 0, 0, 0, 0, 0, 0)))
N = M.matrix_from_rows_and_columns(range(3, 15), [0, 2, 3, 5, 6, 7, 8, 9])
```

Since we have

$$P_{i,z}N_{3(i-2)+1)} - P_{i,y}N_{(3(i-2)+2)} + P_{i,x}N_{(3(i-2)+3)}$$

for $i \in \{2, 3, 4, 5\}$ where $N_{(k)}$ is the k-th row of N:

[5]:
```
assert(tuple(P2[2]*N[0]-P2[1]*N[1]+P2[0]*N[2]) == (0, 0, 0, 0, 0, 0, 0, 0))
assert(tuple(P3[2]*N[3]-P3[1]*N[4]+P3[0]*N[5]) == (0, 0, 0, 0, 0, 0, 0, 0))
assert(tuple(P4[2]*N[6]-P4[1]*N[7]+P4[0]*N[8]) == (0, 0, 0, 0, 0, 0, 0, 0))
```

```
assert(tuple(P5[2]*N[9]-P5[1]*N[10]+P5[0]*N[11]) == (0, 0, 0, 0, 0, 0, 0, 0))
```

a square submatrix of order 7 of N has surely determinant zero if it contains the three rows 0, 1, 2 or the three rows 3, 4, 5 or the three rows 6, 7, 8 or the three rows 9, 10, 11.

Hence we construct all the submatrices of N of order 7 thatdo not contain these these triplets of rows.

[6]:
```
def is_min_sure_zero(st):
    '''
    Given a list of rows, check if it contains the triplet
    0, 1, 2 or 3, 4, 5 or 6, 7, 8 or 9, 10, 11.
    '''
    return(
        Set([0, 1, 2]).issubset(Set(st))
        or Set([3, 4, 5]).issubset(Set(st))
        or Set([6, 7, 8]).issubset(Set(st))
        or Set([9, 10, 11]).issubset(Set(st))
    )

## select the "good" rows
good_rows = filter(lambda u: not is_min_sure_zero(u), Combinations(12, 7))

## select the "good" columns
good_cols = Combinations(8, 7).list()
```

Computation of minors of order 7 (it may take 30')

[7]:
```
m7 = [N.matrix_from_rows_and_columns(rr, cc).det() for rr in good_rows for cc␣
 ↪in good_cols]
m7 = [p for p in m7 if not(p.is_zero())]
```

Computation of squarefree polynomials (it may take 30')

[8]:
```
m7s = [get_sqrfree(p) for p in m7]
```

Saturation of the polynomials w.r.t. u and v

[9]:
```
m7ss = [clear_uv(p) for p in m7s]
```

Saturation with respect to the condition that the points are distinct

[10]:
```
J = S.ideal(m7ss)
points = [P1, P2, P3, P4, P5]
pairs = Combinations(points, 2)
for pair in pairs:
    to_sat = S.ideal(matrix([pair[0], pair[1]]).minors(2))
    J = J.saturation(to_sat)[0]
```

Saturation with respect to the condition that P_1, P_2, and P_4 are aligned

[11]:
```
J = J.saturation(matrix([P1, P2, P4]).det())[0]
```

Primary decomposition of the ideal J

[13]:
```
PD = J.radical().primary_decomposition()
```

We get that the primary decomposition is constituted of 9 ideals:

[14]:
```
assert(len(PD) == 9)
```

Of these ideals, 8 can be written explicitly: they correspond to the case in which the lines $P_1 \vee P_2$ and $P_1 \vee P_4$ are tangent to the isotropic conic in, respectively, (P_2 or P_3) and (P_4 or P_5).

[15]:
```
for P in [P2, P3]:
    for Q in [P4, P5]:
        for I in S.ideal(
            scalar_product(P, P),
            sigma(P1, P2),
            scalar_product(Q, Q),
            sigma(P1, P4)
        ).saturation(
            S.ideal(matrix([P, Q]).minors(2))
        )[0].radical().primary_decomposition():
            assert(I in PD)
```

There is a final ideal in the primary decomposition which is the ideal generated by

$$\delta_1(P_1, P_2, P_4), \bar{\delta}_1(P_1, P_2, P_3), \bar{\delta}_1(P_1, P_4, P_5)$$

[16]:
```
I = J
for P in [P2, P3]:
    for Q in [P4, P5]:
        I = I.saturation(
            S.ideal(scalar_product(P, P), sigma(P1, P2), scalar_product(Q, Q),␣
 ↪sigma(P1, P4)).saturation(
                S.ideal(matrix([P, Q]).minors(2))
            )[0].radical()
        )[0]
```

[17]:
```
assert(I == S.ideal(delta1(P1, P2, P4), delta1b(P1, P2, P3), delta1b(P1, P4,␣
 ↪P5)))
```

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

We define five points, so that P_1, P_2, and P_3 are aligned and P_1, P_4, and P_5 are aligned.

[18]:
```
P1 = vector((1, ii, 0))
P2 = vector(S, (A2, B2, C2))
P4 = vector(S, (A4, B4, C4))
P3 = u1*P1+u2*P2
```

```
P5 = v1*P1+v2*P4
```

We define the matrix of conditions of $P_1, \dots, P_5$.

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

Manipulation of M with elementary rows and columns operations, in order to have a simpler matrix.

We use the fact that M has the following first three rows: $(-3i, 3, 3i, -3, 0, 0, 0, 0, 0, 0)$, $(0, 0, 0, 0, 1, i, -1, 0, 0, 0)$, $(0, 0, 0, 0, i, -1, -i, 0, 0, 0)$.

The second and third row are linearly independent.

```
[20]: assert(M[0] == vector(S, ((-3*ii), 3, (3*ii), -3, 0, 0, 0, 0, 0, 0)))
      assert(M[1] == vector(S, (0, 0, 0, 0, 1, ii, -1, 0, 0, 0)))
      assert(M[2] == vector(S, (0, 0, 0, 0, ii, -1, (-ii), 0, 0, 0)))
```

```
[21]: M.rescale_row(0, 1/3)
```

```
[22]: for j in range(3, 15):
          M.add_multiple_of_row(j, 0, -M[j, 1])

      assert([M[j, 1] for j in range(3, 15)] == [0 for j in range(3, 15)])
```

```
[23]: for j in range(3, 15):
          M.add_multiple_of_row(j, 1, -M[j, 4])

      assert([M[j, 4] for j in range(3, 15)] == [0 for j in range(3, 15)])
```

In order to compute the minors of order 9 of M, now, we can compute the minors of order 7 of the matrix obtained from the rows $3, 4, \dots, 14$ of M and all of its columns, except columns 0 and 4.

```
[24]: N = M.matrix_from_rows_and_columns(range(3, 15), [0, 2, 3, 5, 6, 7, 8, 9])
```

Since we have

$$P_{i,z}N_{3(i-2)+1)} - P_{i,y}N_{(3(i-2)+2)} + P_{i,x}N_{(3(i-2)+3)}$$

for $i \in \{2, 3, 4, 5\}$ where $N_{(k)}$ is the k-th row of N:

```
[25]: assert(tuple(P2[2]*N[0]-P2[1]*N[1]+P2[0]*N[2]) == (0, 0, 0, 0, 0, 0, 0, 0))
      assert(tuple(P3[2]*N[3]-P3[1]*N[4]+P3[0]*N[5]) == (0, 0, 0, 0, 0, 0, 0, 0))
      assert(tuple(P4[2]*N[6]-P4[1]*N[7]+P4[0]*N[8]) == (0, 0, 0, 0, 0, 0, 0, 0))
      assert(tuple(P5[2]*N[9]-P5[1]*N[10]+P5[0]*N[11]) == (0, 0, 0, 0, 0, 0, 0, 0))
```

a square submatrix of order 7 of N has surely determinant zero if it contains the three rows 0, 1, 2 or the three rows 3, 4, 5 or the three rows 6, 7, 8 or the three rows 9, 10, 11.

Hence we construct all the submatrices of N of order 7 thatdo not contain these these triplets of rows.

```
[26]: def is_min_sure_zero(st):
          '''
```

```
    Given a list of rows, check if it contains the triplet
    0, 1, 2 or 3, 4, 5 or 6, 7, 8 or 9, 10, 11.
    '''
    return(
        Set([0, 1, 2]).issubset(Set(st))
        or Set([3, 4, 5]).issubset(Set(st))
        or Set([6, 7, 8]).issubset(Set(st))
        or Set([9, 10, 11]).issubset(Set(st))
    )

## select the "good" rows
good_rows = filter(lambda u: not is_min_sure_zero(u), Combinations(12, 7))

## select the "good" columns
good_cols = Combinations(8, 7).list()
```

Computation of minors of order 7 (it may take 6m)

[27]:
```
m7 = [N.matrix_from_rows_and_columns(rr, cc).det() for rr in good_rows for cc␣
  ↪in good_cols]
m7 = [p for p in m7 if not(p.is_zero())]
```

Some preprocessing. We divide each element of m7 by u_1, u_2, v_1, v_2, and the condition that P_1, P_2, and P_4 are aligned as much as possible In this way the elements of m7 become simpler (it may take 4m).

[28]:
```
dt = matrix([P1, P2, P4]).det()
```

[29]:
```
m7 = [poly_saturate(p, u1) for p in m7]
m7 = [poly_saturate(p, u2) for p in m7]
m7 = [poly_saturate(p, v1) for p in m7]
m7 = [poly_saturate(p, v2) for p in m7]
m7 = [poly_saturate(p, dt) for p in m7]
```

[30]:
```
len(m7)
```

[30]: 2592

[33]:
```
m7[0].variables()
```

[33]: (u1, u2, v1, v2, A2, B2, C2, A4, B4, C4)

[39]:
```
sst = {A2:7, B2:-4, A4:3, B4:B4}
m7n = [mm.subs(sst) for mm in m7]
```

[40]:
```
m7ns = [get_sqrfree(p) for p in m7n]
```

[41]:
```
m7nss = [clear_uv(p) for p in m7ns]
```

[ ]:
```
Jn = S.ideal(m7nss)
Jn = Jn.saturation(u1)[0]
dtn = dt.subs(sst)
Jn = Jn.saturation(dtn)[0]
```

[ ]:
```
Jn.groebner_basis()
```

Computation of squarefree polynomials (it may take 1h)

[29]:
```
m7s = [get_sqrfree(p) for p in m7]
```

Saturation of the polynomials w.r.t. u and v (it may take 1h)

[30]:
```
m7ss = [clear_uv(p) for p in m7s]
```

Saturation with respect to the condition that the points are distinct

[41]:
```
J = S.ideal(m7ss)
```

[ ]:
```
J = J.saturation(u1)[0]
```

[ ]:
```
J = J.saturation(dt)[0]
```

[36]:
```
points = [P1, P2, P3, P4, P5]
pairs = Combinations(points, 2)
for pair in pairs:
    to_sat = S.ideal(matrix([pair[0], pair[1]]).minors(2))
    J = J.saturation(to_sat)[0]
```

Saturation with respect to the condition that P_1, P_2, and P_4 are aligned (it may take 80')

[ ]:
```
assert(J == S.ideal(S.one()))
```