NB.04.F2

July 23, 2024

1 Proposition {proposition:P1_sing}

[16]:
```
load("basic_functions.sage")
```

We can split the problem into two cases: $P_1 = (1 : i : 0)$ and $P_1 = (1 : 0 : 0)$.

In both cases, we construct the matrix $\Phi(P_1, \dots, P_5)$ for * generic P_2, P_4, * P_3 aligned with P_1 and P_2, * P_5 aligned with P_1 and P_4.

We show that, actually, the case $P_1 = (1 : i : 0)$ cannot happen.

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

The conclusion of the computations in this case is that P_1 cannot be singular.

We define the points $P_1, \dots, P_5$

[29]:
```
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
```

Construction of the matrix of all the linear conditions.

[30]:
```
M1 = condition_matrix([P1, P2, P3, P4, P5], S, standard="all")
```

Since the first three rows of M_1 are, respectively, $(-3i, 3, 3i, -3, 0, 0, 0, 0, 0, 0)$, $(0, 0, 0, 0, 1, i, -1, 0, 0, 0)$, $(0, 0, 0, 0, i, -1, -i, 0, 0, 0)$ in order to compute the minors of order 10 of M_1, we can compute the minors of order 8 of the matrix obtained from the rows 3, 4, …, 14 of M_1 and all the columns of M_1 except columns 1 and 4. If we assume that P_1 is singular, then we have the further condition: $(0, 1, 2i, -3, 0, 0, 0, 0, 0, 0)$ to add to M_1, hence we have to manipulate M_1.

[31]:
```
rw = vector(S, (0, 1, (2*ii), -3, 0, 0, 0, 0, 0, 0))
for i in range(10):
    M1[1, i] = rw[i]
## Now we can use the first three rows of M1 to simplify M1 with elementary
## row operations

M1.rescale_row(0, 1/(-3*ii))
```

```
M1.rescale_row(2, -ii)
M1.add_multiple_of_row(0, 1, -ii)

for j in range(3, 15):
    M1.add_multiple_of_row(j, 0, -M1[j, 0])

## We check that the first column has all the elements of position 3, 4, ..
## equal to zero:

assert([M1[j, 0] for j in range(3, 15)] == [0 for j in range(3, 15)])

for j in range(3, 15):
    M1.add_multiple_of_row(j, 1, -M1[j, 1])

## We check that the second column has all the elements of position 3, 4, ..
## equal to zero:

assert([M1[j, 1] for j in range(3, 15)] == [0 for j in range(3, 15)])

for j in range(3, 15):
    M1.add_multiple_of_row(j, 2, -M1[j, 4])

## We check that the fifth column has all the elements of position 3, 4, ..
## equal to zero:

assert([M1[j, 4] for j in range(3, 15)] == [0 for j in range(3, 15)])
```

Now we can consider the matrix MM given by the rows 3, 4, ..., 14 and all the columns, except 0, 1, 4

[32]:
```
MM = M1.matrix_from_rows_and_columns(range(3, 15), [2, 3, 5, 6, 7, 8, 9])

## Since we have:
assert(tuple(P2[2]*MM[0]-P2[1]*MM[1]+P2[0]*MM[2]) == (0, 0, 0, 0, 0, 0, 0))
assert(tuple(P3[2]*MM[3]-P3[1]*MM[4]+P3[0]*MM[5]) == (0, 0, 0, 0, 0, 0, 0))
assert(tuple(P4[2]*MM[6]-P4[1]*MM[7]+P4[0]*MM[8]) == (0, 0, 0, 0, 0, 0, 0))
assert(tuple(P5[2]*MM[9]-P5[1]*MM[10]+P5[0]*MM[11]) == (0, 0, 0, 0, 0, 0, 0))
## in the computation of the order 10 minors of M1 and hence of order 7 minors
## of MM, we can erase many matrices:

rg = Combinations(12, 7)

## given a list of (seven) rows st, the method checks if it
## contains the triplet [0, 1, 2] or [3, 4, 5] or ...
def is_min_sure_zero(st):
    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

rg1 = filter(lambda uu: not is_min_sure_zero(uu), rg)
```

First 'long' computation: computation of minors of order 7 (about 38 sec):

[33]:
```
ttA = cputime()
min7 = [MM.matrix_from_rows(rr).det() for rr in rg1]

print(cputime()-ttA)
```

```
36.945547000000005
```

We manipulate the minors so that ...

[34]:
```
## division by u1
dt = matrix([P1, P2, P4]).det()

print("some divisions. Can take some time (8 sec).")
ttA = cputime()

min7 = filter(lambda uu: uu != 0, min7)

min7 = [qr_gener(mm, u1) for mm in min7]
min7 = [qr_gener(mm, u2) for mm in min7]
min7 = [qr_gener(mm, v1) for mm in min7]
min7 = [qr_gener(mm, v2) for mm in min7]
min7 = [qr_gener(mm, dt) for mm in min7]
print(cputime()-ttA)
```

```
some divisions. Can take some time (8 sec).
5.802280999999994
```

A long conmputation of a Groebner basis of the ideal of order 7 minors

[35]:
```
J7 = S.ideal(min7)
ttA = cputime()
gJ7 = J7.groebner_basis()
print(cputime()-ttA)
```

```
0.06705599999999379
```

Compute some partial saturations

[36]:
```
## division by u1 etc
ttA = cputime()

gJ7 = [poly_saturate(mm, u1) for mm in gJ7]
```

```
gJ7 = [poly_saturate(mm, u2) for mm in gJ7]
gJ7 = [poly_saturate(mm, v1) for mm in gJ7]
gJ7 = [poly_saturate(mm, v2) for mm in gJ7]
gJ7 = [poly_saturate(mm, dt) for mm in gJ7]
print(cputime()-ttA)
```

```
0.03796800000000644
```

Computation of squarefree polynomials. About 2,5 sec.

[37]:
```
ttA = cputime()
gJ7 = [get_sqrfree(mm) for mm in gJ7]
print(cputime()-ttA)

sgJ7 = S.ideal(gJ7).saturation(u1*u2*v1*v2)[0]
sgJ7 = S.ideal(gJ7).saturation(dt)[0]
```

```
1.6758489999999995
```

The final ideal is (1), so P_1 cannot be singular.

[38]:
```
assert(sgJ7 == S.ideal(S.one()))
```

Conclusion: in a V-configuration, the point P_1 cannot be both on the isotropic conic and singular for the cubic.

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

We define the points $P_1, \dots, P_5$

[17]:
```
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
```

Construction of the matrix $\Phi(P_1, \dots, P_5)$

[18]:
```
M1 = condition_matrix([P1, P2, P3, P4, P5], S, standard="all")
```

Since the first three rows of M_1 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 10 of M_1, we can compute the minors of order 8 of the matrix obtained from the rows 3, 4, …, 14 of M_1 and all the columns of M_1 except columns 1 and 4. If we assume that P_1 is singular, then we have the further condition: $(1,0,0,0,0,0,0,0,0,0)$ to add to M_1, hence we can extract from M_1 the matrix without the first three rows and without the columns 0, 1, 4.

[19]:
```
MM = M1.matrix_from_rows_and_columns(range(3, 15), [2, 3, 5, 6, 7, 8, 9])
```

Since we have:

[20]:
```
assert(tuple(P2[2]*MM[0]-P2[1]*MM[1]+P2[0]*MM[2]) == (0, 0, 0, 0, 0, 0, 0))
assert(tuple(P3[2]*MM[3]-P3[1]*MM[4]+P3[0]*MM[5]) == (0, 0, 0, 0, 0, 0, 0))
assert(tuple(P4[2]*MM[6]-P4[1]*MM[7]+P4[0]*MM[8]) == (0, 0, 0, 0, 0, 0, 0))
assert(tuple(P5[2]*MM[9]-P5[1]*MM[10]+P5[0]*MM[11]) == (0, 0, 0, 0, 0, 0, 0))
```

in the computation of the order 10 minors of M_1 and hence of order 7 minors of MM, we can erase many matrices:

[21]:
```
rg = Combinations(12, 7)

## given a list of (seven) rows st, the method checks if it
## contains the triplet [0, 1, 2] or [3, 4, 5] or ...
def is_min_sure_zero(st):
    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

rg1 = filter(lambda uu: not is_min_sure_zero(uu), rg)
```

First 'long' computation: computation of minors of order 7:

[22]:
```
ttA = cputime()
min7 = [MM.matrix_from_rows(rr).det() for rr in rg1]
print(cputime()-ttA)
```

```
1.9884460000000006
```

Some saturations. Can take some time

[24]:
```
ttA = cputime()

dt = matrix([P1, P2, P4]).det()
min7 = filter(lambda uu: uu != 0, min7)

min7 = [poly_saturate(mm, u1) for mm in min7]
min7 = [poly_saturate(mm, u2) for mm in min7]
min7 = [poly_saturate(mm, v1) for mm in min7]
min7 = [poly_saturate(mm, v2) for mm in min7]
min7 = [poly_saturate(mm, dt) for mm in min7]
print(cputime()-ttA)
```

```
0.7965390000000001
```

Computation of a Groebner basis of the ideal of order 7 minors

[25]:
```
J7 = S.ideal(min7)
ttA = cputime()
```

```
gJ7 = J7.groebner_basis()
print(cputime()-ttA)

print("Groebner basis computed")
```

```
0.05365900000000057
Groebner basis computed
```

Computed some partial saturations

[26]:
```
## division by u1 (1 minute)
ttA = cputime()

gJ7 = [poly_saturate(mm, u1) for mm in gJ7]
gJ7 = [poly_saturate(mm, u2) for mm in gJ7]
gJ7 = [poly_saturate(mm, v1) for mm in gJ7]
gJ7 = [poly_saturate(mm, v2) for mm in gJ7]
gJ7 = [poly_saturate(mm, dt) for mm in gJ7]
print(cputime()-ttA)
```

```
0.03507700000000025
```

Computation of the primary decomposition of the radical of the ideal `gJ7`

[27]:
```
sgJ7 = S.ideal(gJ7).saturation(u1*u2*v1*v2)[0]
sgJ7 = S.ideal(gJ7).saturation(dt)[0]

PD = sgJ7.radical().primary_decomposition()
```

We get three ideals, which are: * $(\delta_1(P_1,P_2,P_4),\overline{\delta}_1(P_1,P_4,P_5))$, * $(\delta_1(P_1,P_2,P_4),\overline{\delta}_1(P_1,P_2,P_3))$, * $(\overline{\delta}_1(P_1,P_2,P_3),\overline{\delta}_1(P_1,P_4,P_5))$

[28]:
```
assert(len(PD) == 3)

assert(PD[0] == S.ideal(delta1(P1, P2, P4), delta1b(P1, P4, P5)))
assert(PD[1] == S.ideal(delta1(P1, P2, P4), delta1b(P1, P2, P3)))
assert(PD[2] == S.ideal(delta1b(P1, P2, P3), delta1b(P1, P4, P5)))
```

Conclusion: in a V-configuration, where the point P_1 is not on the isotropic conic, then P_1 is singular for the cubic if and only if * $\delta_1(P_1,P_2,P_4)=0$ and $\overline{\delta}_1(P_1,P_4,P_5)=0$, * $\delta_1(P_1,P_2,P_4)=0$ and $\overline{\delta}_1(P_1,P_2,P_3)=0$, * $\overline{\delta}_1(P_1,P_2,P_3)=0$ and $\overline{\delta}_1(P_1,P_4,P_5)=0$.