geometrie_molecule

March 21, 2025

Nous ne traiterons que le cas du cyclohexane. Quelques notations :

[1]:
```
R = RealField(300)
Ruvw.<u,v,w> = PolynomialRing(R)
Ru.<u> = PolynomialRing(R)
Rv.<v> = PolynomialRing(R)

L = 1; phi = 109*2*pi/360; alpha = pi/3
HB = L * cos(phi/2)
AH = sqrt(L^2 - HB^2)
AC = 2 * AH

def f(u): # paramétrisation rationnelle du cercle, rayon HB centré en H
    return vector([AH, HB*(1-u^2)/(1+u^2), HB*2*u/(1+u^2)])

def rot(a): # rotation d'angle a, axe k
    return matrix([[cos(a), -sin(a), 0], [sin(a), cos(a), 0], [0,0,1]])
```

[2]:
```
AB = f(u) #coordonnées de B dans (A,i,j,k)
AF = rot(alpha) * f(1/v) # coordonnées de F
```

[3]:
```
P = Ruvw(numerator(AB.dot_product(AF) - L^2*cos(phi))); P
```

[3]:
```
-0.66098174906670123213697405645739535144110018222245336519002271941593156653868
2888511221660*u^2*v^2 + 1.651136308914313337428017871589443143597703213518246214
430445589893203361056768358856353 20*u^2 + 2.697727382171373325143964256821113712
804593572963507571139108820213593277886463282287293 62*u*v + 1.651136308914313337
428017871589443143597703213518246214430445589893203361056768358856353 20*v^2 + 2.
614390675809641244421027671225724782234209822777192008481359489095541649708987 96
508028124
```

[4]:
```
sol = Ru(P(v=u)).roots()  # racines de P(u,u)
sol
```

[4]:
```
[(-
3.0812363914163384385181470820080173906364669953938216473175048442356269740663 10
21150875908,
```

```
  1)),
 (3.0812363914163384385181470820080173906364669953938216473175048442356269740663
1021150875908,
  1))]
```

Les deux racines réelles de $P(u, u)$ sont $\pm r$

[5]:
```
r = sol[1][0]; r
```

[5]:
```
3.08123639141633843851814708200801739063646699539382164731750484423562697406631021150875908
```

[6]:
```
P(u = r, v = r) # proche de 0
```

[6]:
```
1.178182431671454372742985269196706154314051722999867895626771788124905261133211
80555042349e-89
```

1) Le cas $u = v = w$

[7]:
```
def Triangle(A,B,C, c): # triangle (ABC) de couleur 'c'
    return line3d([A,B],color=c,size=2) + line3d([B,C],color=c,size=2) +␣
 ↪line3d([C,A],color=c,size=2)

def Affiche(U, V, W):
    # On calcule les coordonnées des 6 points A, ..., F dans (A,i,j,k)
    A = (0,0,0)
    C = (AC,0,0)
    E = rot(alpha) * vector(C)
    B = AB(u = U)
    F = AF(v = V)
    AD = rot(-alpha) * (f(1/w) - vector(C)) + vector(C) # rotation de centre C
    D = AD(w = W)

    P = Graphics() + text3d("A",(-0.1,0,0)) + text3d("C", (AC-0.1,0,0)) +␣
 ↪text3d("E", (E[0] - 0.1,E[1],E[2]))
    P += Triangle(A,B,C, 'green')
    P += Triangle(C,D,E, 'red')
    P += Triangle(E,F,A, 'violet')
    P += Triangle(A,C,E, 'blue')
    for p in [A,B,C,D,E,F]:
        P += point3d(p,color='black',size=15)
        # print(numerical_approx(vector(p)))
    return (P)

Affiche(r, r, r) # Fixez le quadrilatère (BCEF) et visualisez la "chaise"
```

[7]:
```
Graphics3d Object
```

[8]:
```
Affiche(-r, -r, -r) # résultat analogue à symétrie près, encore une "chaise"
```

[8]:
```
Graphics3d Object
```

2) Le cas $u = v = r$, w quelconque

[9]:
```
factor(P(u = r))
```

[9]:
```
(-
4.624236115962029139129983856093676990077606791481399159240936618618269855190902
49473514971) * (v - 3.081236391416338438518147082008017390636466995393821647317 5
0484423562697406631021150875908) * (v + 1.283677708075303232522630388989170702 40
468336315597934351096113397774015339896986547736026)
```

[10]:
```
sol = Rv(P(u = r)).roots(); sol
```

[10]:
```
[(-
1.283677708075303232522630388989170702404683363155979343510961133977740153398969
86547736026,
  1),
 (3.08123639141633843851814708200801739063646669953938216473175048442356269740663
1021150875908,
  1)]
```

[11]:
```
s = sol[0][0]; s
```

[11]:
```
-1.28367770807530323252263038898917070240468336315597934351096113397774015339896
986547736026
```

[12]:
```
Affiche(r, r, s) # fixer (BCEF) et visualiser le "bateau"
```

[12]:
```
Graphics3d Object
```

3. Cas général. Si P(u,v) = P(w,u) = 0, alors Q(v,w) = 0.

[13]:
```
Q = P(u=w,v=u).resultant(P,u); Q # resultat manifestement faux !!!
```

[13]:
```
-7.94400000000000000000000000000000000000000000000000000000000000000000000000000
000000000000*v^4*w^2 + 15.8900000000000000000000000000000000000000000000000000
0000000000000000000000000000000000*v^3*w^3 - 7.944000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000*v^2*w^4 + 19.8400000000
000000000000000000000000000000000000000000000000000000000000000000000000000000*v
^4 - 7.2640000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000*v^3*w - 25.1400000000000000000000000000000000000000000000000000
000000000000000000000000000000000000*v^2*w^2 - 7.2640000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000*v*w^3 + 19.8400000000
000000000000000000000000000000000000000000000000000000000000000000000000000000*w
^4 + 31.4100000000000000000000000000000000000000000000000000000000000000000000000
0000000000000000*v^2 - 62.8300000000000000000000000000000000000000000000000000
```

```
0000000000000000000000000000000000*v*w + 31.410000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000*w^2
```

Calcul en multipliant par 10^20 et en arrondissant, pour espérer une vingtaine de chiffres significatifs:

[14]:
```
Zuvw.<u,v,w> = PolynomialRing(ZZ)
P = Zuvw(pari.round(10^20 * P))
```

[15]:
```
Q = P.resultant(P(v=w),u)
R = Q.resultant(P(u=w),w)
```

Si $Q(v,w) = P(v,w) = 0$, alors $R(w) = 0$. On retrouve les valeurs connues:

[16]:
```
Rv(R).roots()
```

[16]:
```
[(-
3.081236391416338438511365227368963613015103971449029569056236565532201466805538
92757325755,
  1),
 (-
1.283677708075303232522105100438348827468409772817992981481595778660442490963371
69927531456,
  1),
 (1.28367770807530323252210510043834882746840977281799298148159577866044249096337
169927531456,
  1),
 (3.08123639141633843851136522736896361301510397144902956905623656553220146680553
892757325755,
  1)]
```

[ ]:

[ ]:

[ ]: