4-brachistochrone

June 13, 2023

[1]:
```
from sympy import *
from DifferentialAlgebra import *
init_printing ()
```

0.1 Computation of the brachistochrone equation

[2]:
```
x,g,c,D = var('x,g,c,D')
L,S_L,y = function('L,S_L,y')
```

[3]:
```
params = [g,c,D]
R = DifferentialRing (derivations=[x], blocks=[L,S_L,y,params],␣
 ↪parameters=params)
R
```

[3]:
```
differential_ring
```

[4]:
```
syst = [ Eq(L(x)**2, (1+Derivative(y(x),x)**2)/(2*g*y(x))),
         Eq(2*L(x)*S_L(x), Derivative(y(x),x)/(g*y(x))),
         Eq(L(x) - Derivative(y(x),x)*S_L(x), c),
         Eq(D, 1/(2*g*c**2))]
syst
```

[4]:
$$\left[L^2(x) = \frac{\left(\frac{d}{dx}y(x)\right)^2 + 1}{2gy(x)},\ 2L(x)\,\mathrm{S_L}\,(x) = \frac{\frac{d}{dx}y(x)}{gy(x)},\ L(x) - \mathrm{S_L}\,(x)\frac{d}{dx}y(x) = c,\ D = \frac{1}{2c^2g}\right]$$

[5]:
```
ideal = R.RosenfeldGroebner (syst)
ideal
```

[5]:
```
[regular_differential_chain, regular_differential_chain]
```

[6]:
```
[ C.equations(solved=True) for C in ideal ]
```

[6]:
$$\left[\left[g = \frac{1}{2Dc^2},\ \left(\frac{d}{dx}y(x)\right)^2 = \frac{D - y(x)}{y(x)},\ \mathrm{S_L}\,(x) = c\frac{d}{dx}y(x),\ L(x) = \frac{Dc}{y(x)}\right],\ \left[g = \frac{1}{2Dc^2},\ y(x) = D,\ \mathrm{S_L}\,(x) = 0,\ L(\right.\right.$$

[7]:
```
leader = var('leader')
derivative = function('derivative')
```

```
C = ideal[0]
edo = C.equations(selection=Eq(leader,derivative(y(x))))[0]
edo
```

[7]:
$$-D + y(x)\left(\frac{d}{dx}y(x)\right)^2 + y(x)$$

0.2 Use of a Puiseux series to perform the first numerical integration step

[8]:
```
a0,a1,a2 = var('a0,a1,a2')
solp = a0*x**(2/Integer(3)) + a1*x**(4/Integer(3)) + a2*x**(6/Integer(3))
solp
```

[8]:
$$a_0 x^{\frac{2}{3}} + a_1 x^{\frac{4}{3}} + a_2 x^2$$

[9]:
```
values = {}
values[a1] = -9/(20*a0)
values[a2] = -243/(2800*a0**3)
values[a0] = (9*D/4)**(1/Integer(3))
y_at_solp = R.evaluate (edo, {y(x):solp})
```

[10]:
```
y_at_solp.subs(values).subs(values).doit().expand()
```

[10]:
$$\frac{207x^2}{3500D} - \frac{19683x^4}{85750000D^3} - \frac{1809\sqrt[3]{2}\cdot 3^{\frac{2}{3}}x^{\frac{8}{3}}}{245000D^{\frac{5}{3}}} - \frac{729\cdot 2^{\frac{2}{3}}\sqrt[3]{3}x^{\frac{10}{3}}}{350000D^{\frac{7}{3}}}$$

[11]:
```
soln = R.evaluate (R.evaluate (solp, values), values)
soln
```

[11]:
$$\frac{\sqrt[3]{2}\cdot 3^{\frac{2}{3}}\sqrt[3]{D}x^{\frac{2}{3}}}{2} - \frac{27x^2}{700D} - \frac{3\cdot 2^{\frac{2}{3}}\sqrt[3]{3}x^{\frac{4}{3}}}{20\sqrt[3]{D}}$$

[12]:
```
values[D] = 2
soln = R.evaluate (soln, values)
```

[13]:
```
theta = var('theta')
def true_sol (x) :
    D_val = D.subs(values)
    theta_val = nsolve ((D_val/2) * (theta-sin(theta)) - x, theta, .1)
    return (D_val/2) * (1 - cos(theta_val))
```

At x0, the value of the ordinate of the point on the cycloid is compared to that of the point on the Puiseux curve

Both are quite close

[14]:
```
x0 = .01
xend = 1.
true_sol (x0), true_sol (xend)
```

[14]:
(0.0760417718282471, 1.35579714038883)

[15]:
```
soln.subs({x:x0}).evalf(), soln.subs({x:xend}).evalf ()
```

[15]:
(0.0760417845217514, 1.35910982123678)

0.3 Numerical integration over a prolongated system to pass the second point

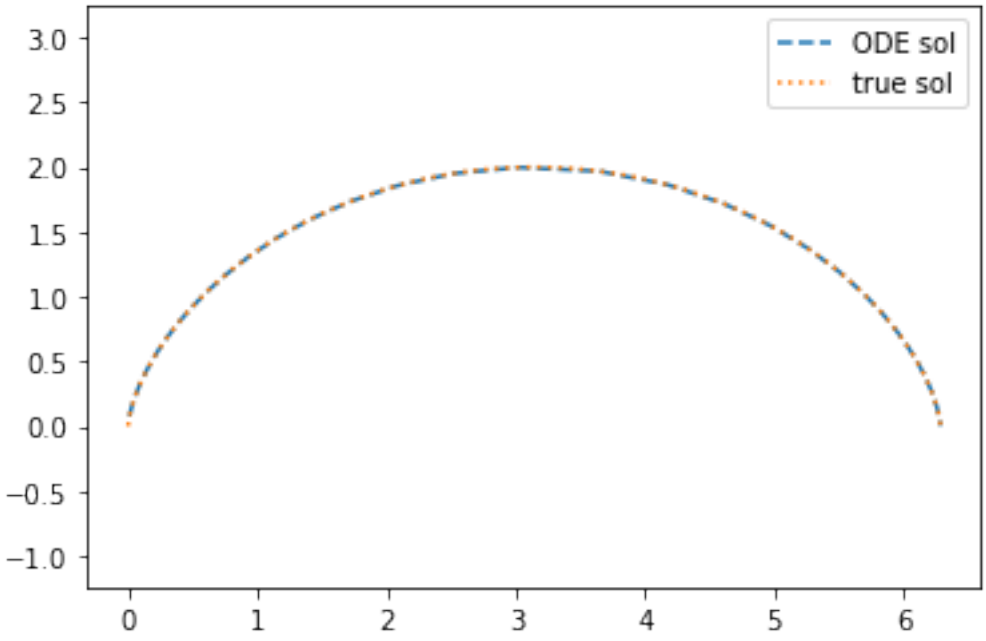
Having passed the first point using the Puiseux series, we can integrate the brachistochrone ODE

In order to pass the second point, a prolongated equation is used

[16]:
```
import math
import numpy as np
from scipy.integrate import solve_ivp
def f(x,y) :
    return np.array ([y[1], - (y[1]**2 + 1)/(2*y[0])], dtype=np.float64)

xend = 2*math.pi
y0 = soln.subs({x:x0}).evalf()
yp0 = math.sqrt(D.subs(values)/y0 - 1)
ode_sol = solve_ivp (f, [x0,xend], np.array([y0,yp0],dtype=np.float64),␣
 ↪rtol=1e-8)
```

[17]:
```
import matplotlib.pyplot as plt
plt.axis('equal')
plt.plot (ode_sol.t, ode_sol.y[0], linestyle='dashed', label='ODE sol')
tplot = np.linspace (0, 2*math.pi, 100)
xplot = [ D.subs(values)/2*(theta - math.sin(theta)) for theta in tplot ]
yplot = [ D.subs(values)/2*(1 - math.cos(theta)) for theta in tplot ]
plt.plot (xplot, yplot, linestyle='dotted', label='true sol')
plt.legend ()
plt.show ()
```


ODE sol
true sol
3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
0
1
2
3
4
5
6

[ ]: