Способи моделювання кінематики і динаміки механізмів мовами Python та Modelica

Перший спосіб:
використання мови загального призначення Python та математичної бібліотеки SciPy

Код програми

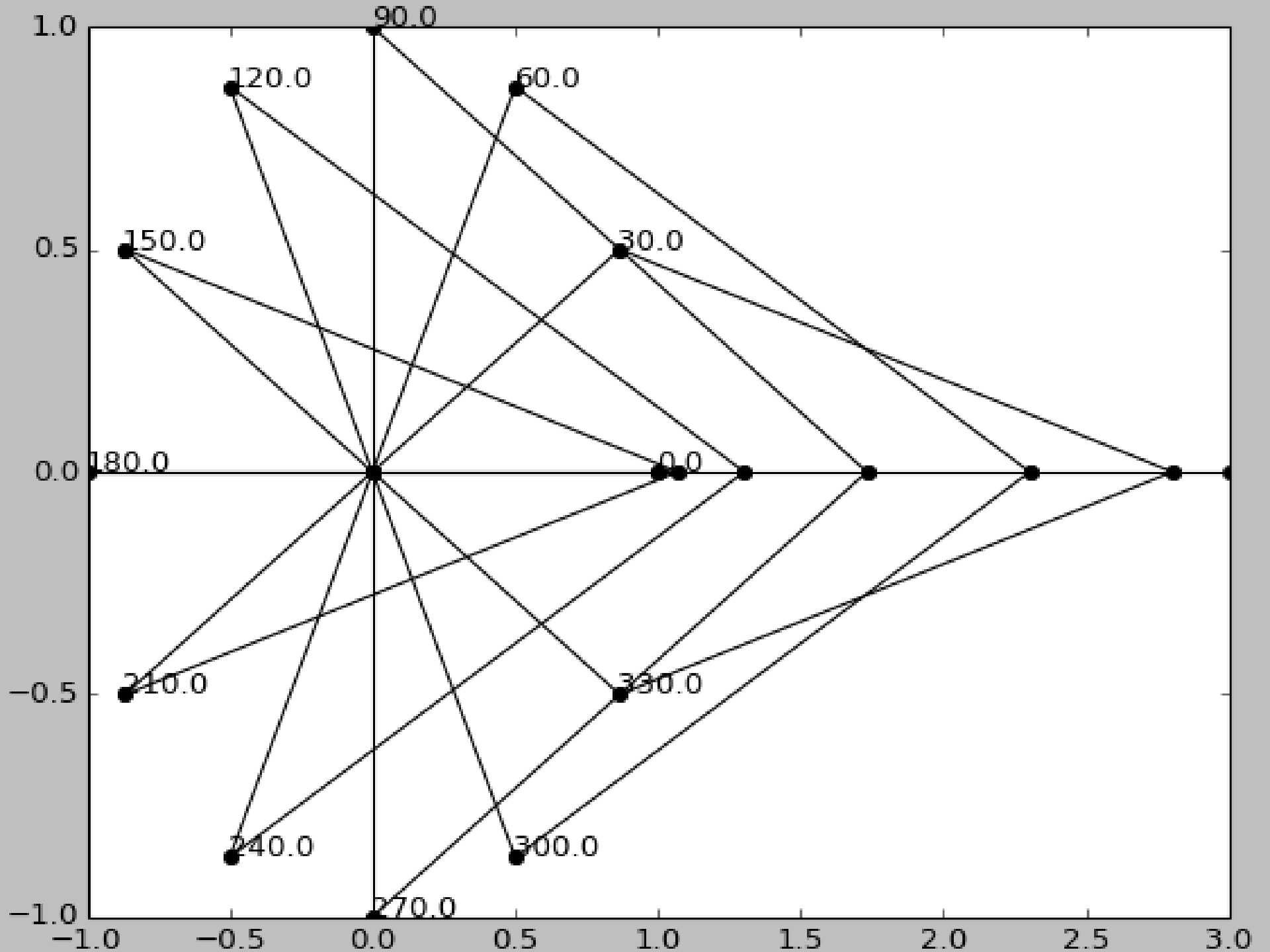
```
from __future__ import division
import matplotlib.pyplot as plt
from scipy.optimize import root
from math import pi,sin,cos,tan,degrees

def f(X, L1,L2,x0,y0,x1,y1,y2): # векторна функція
    x2,=X # невідомі (зверніть увагу X - вектор!)
    eqs=[0] # бо кількість рівнянь повинне бути >1

    # наступне рівняння не обов'язкове
    #eqs+= [(x1-x0)**2 + (y1-y0)**2 - L1**2] # незмінна довжина L кривошипа dx**2+dy**2=L**2
    eqs+= [(x2-x1)**2 + (y2-y1)**2 - L2**2] # незмінна довжина L шатуна dx**2+dy**2=L**2
    return eqs
```

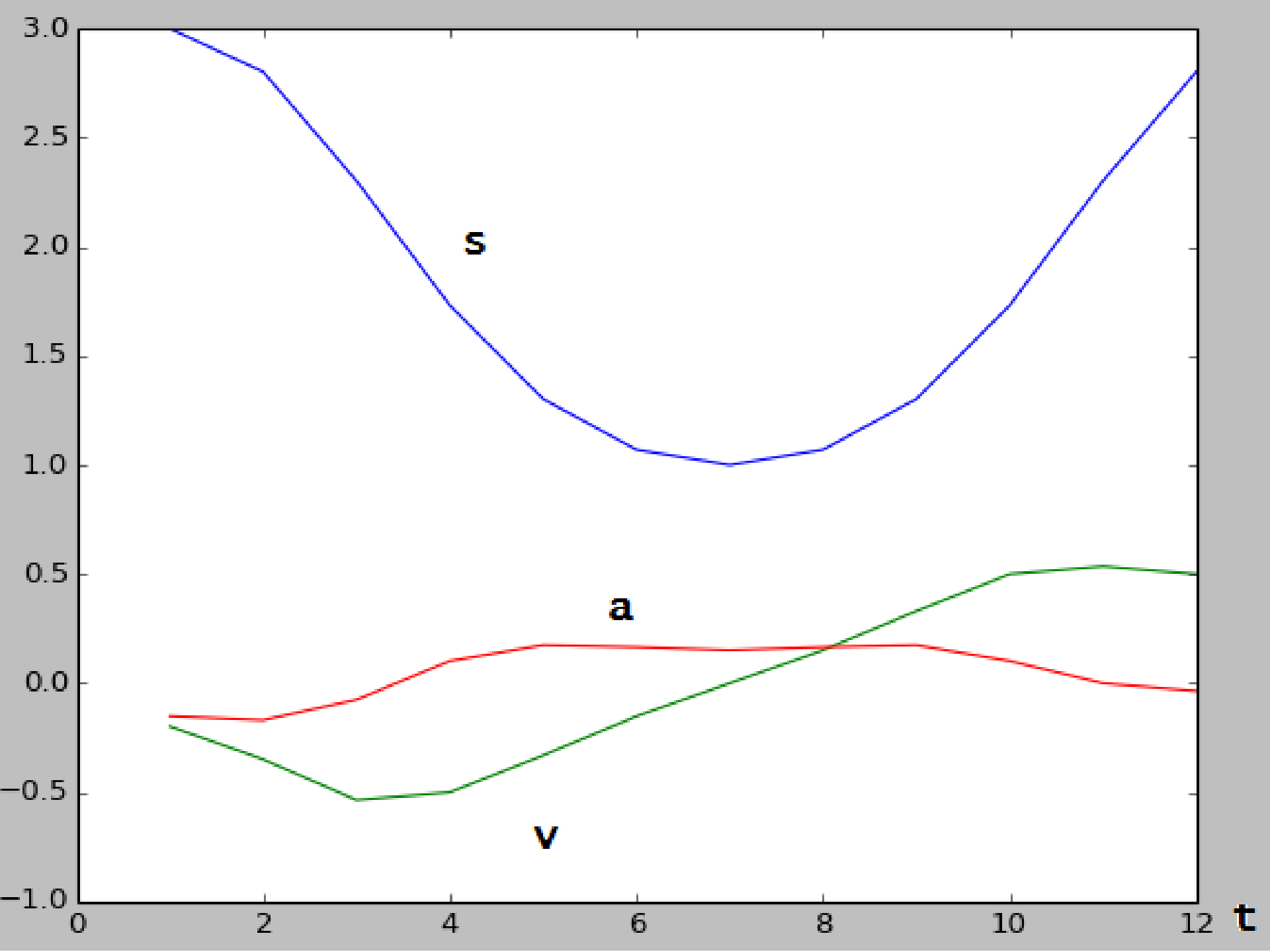
Код програми (продовження)

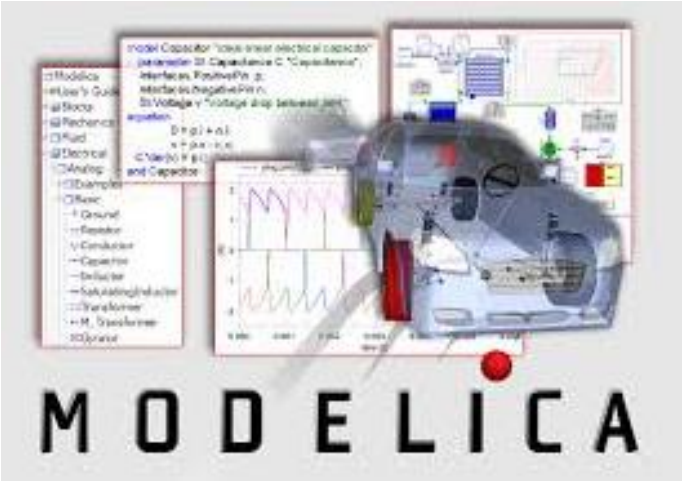
```
L1,L2,x0,y0,y2,a=1,2,0,0,0,0 # відомі
x2=L1+L2 # невідомі (початкове значення)
t=0
T,X2=[],[]
while a<2*pi:
    x1=L1*cos(a)+x0 # координата точки кривошипа dx/L=cos(a)
    y1=L1*sin(a)+y0 # координата точки кривошипа dy/L=sin(a)
    # у звязку з тим що рівняння можуть мати кілька коренів потрібно функції root
    # передавати наближені значення невідомих, знайдені на попередніх кроках
    sol = root(f, [x2], args=(L1,L2,x0,y0,x1,y1,y2), method='lm') # розв'язати систему
    x2=sol.x[0] # корені
    t+=1
    T.append(t)
    X2.append(x2)
    # нарисувати положення ланок механізму
    plt.plot([x0,x1],[y0,y1],'ko-')
    plt.text(x1, y1, round(degrees(a)))
    plt.plot([x1,x2],[y1,y2],'ko-')
    a+=pi/6 # збільшити кут на крок
plt.show()
```

Код програми (продовження)

```
import scipy
#V2=scipy.diff(X2) # швидкість повзуна (forward differences)
# але краще
dt=T[1]-T[0]
V2=scipy.gradient(X2, dt) # швидкість повзуна (central differences)
A2=scipy.gradient(V2, dt) # прискорення повзуна (central differences)
plt.plot(T,X2)
plt.plot(T,V2)
plt.plot(T,A2)
plt.show()
```


90.0
120.0
60.0
150.0
30.0
180.0
0.0
210.0
330.0
240.0
300.0
270.0
1.0
0.5
0.0
−0.5
−1.0
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5
3.0

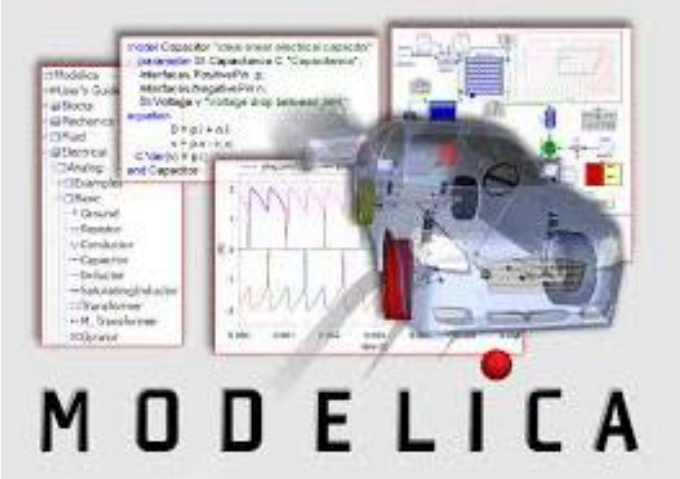

3.0
2.5
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
0
2
4
6
8
10
12
t
s
a
v



Другий спосіб:
використання мови моделювання Modelica

Код програми

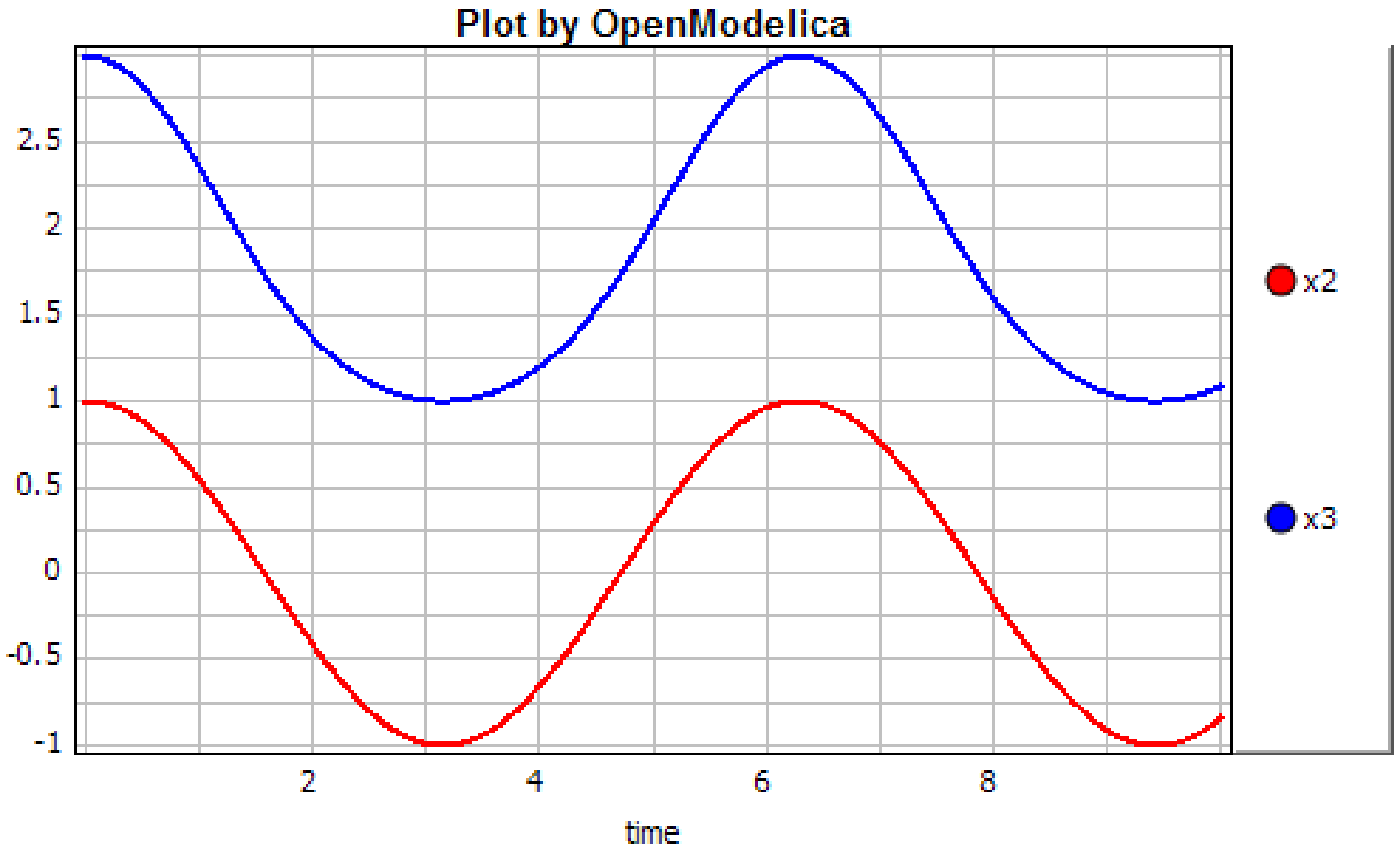
```
model SliderCrank
  parameter Real L1=1, L2=2;
  Real x1(start=0),x2(start=1),x3(start=3),
  y1(start=0),y2(start=0),y3(start=0);
  Real a(start=0);
equation
  der(a)=1;
  x1=0;
  y1=0;
  y3=0;
  x2=L1*cos(a)+x1;
  (x1-x2)^2+(y1-y2)^2=L1^2;
  (x2-x3)^2+(y2-y3)^2=L2^2;
end SliderCrank;
```



Третій спосіб:
використання мови моделювання Modelica та компонентно-орієнтованого підходу

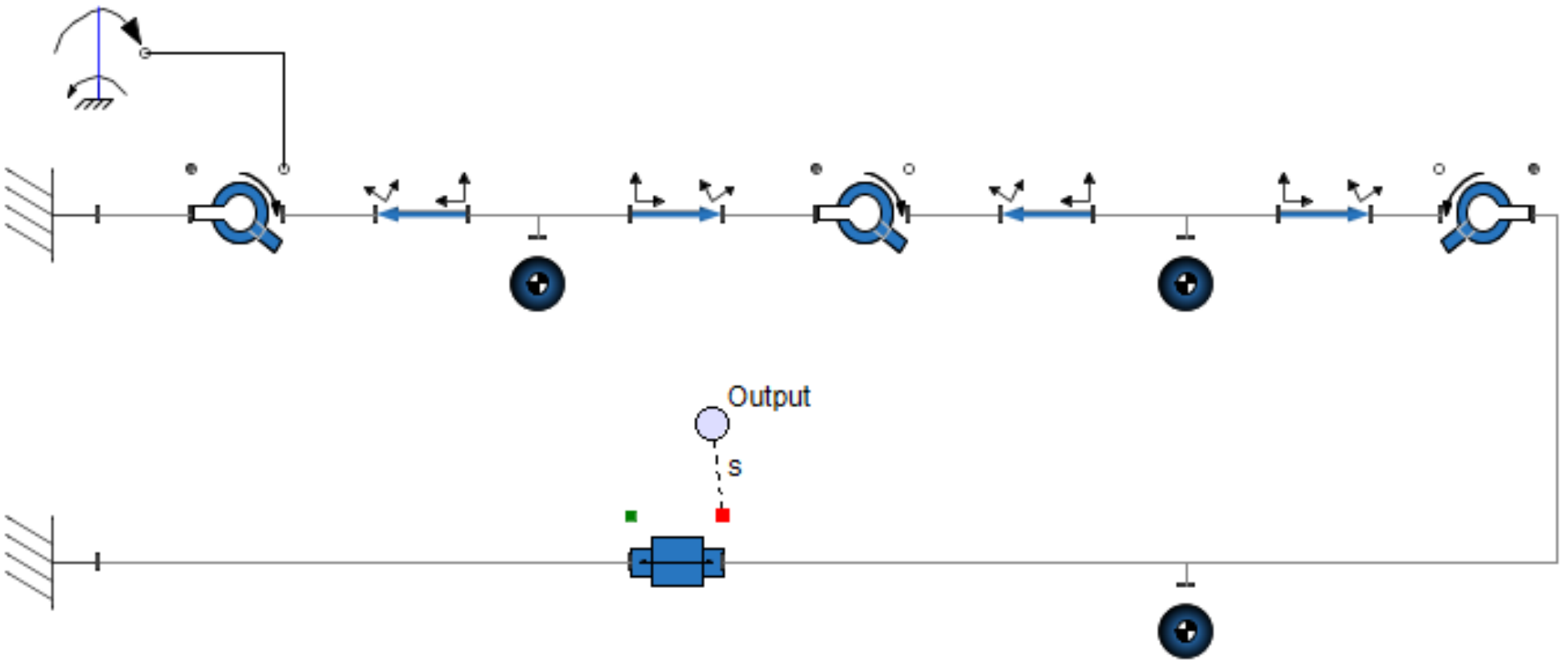
Код програми

```
connector Flange
  Real x,y;
end Flange;

model Frame2D
  parameter Real L=1;
  Flange flange_a;
  Flange flange_b;
equation
  (flange_a.x-flange_b.x)^2+(flange_a.y-flange_b.y)^2=L^2;
end Frame2D;

model SliderCrank
  Real a(start=0);
  Frame2D frame1(L=1);
  Frame2D frame2(L=2);
equation
  der(a)=1;
  frame1.flange_a.x=0;
  frame1.flange_a.y=0;
  frame1.flange_b.x=frame1.L*cos(a)+frame1.flange_a.x;
  //frame1.flange_b.y=frame1.L*sin(a)+frame1.flange_a.y;
  frame2.flange_b.y=0;
  connect(frame1.flange_b, frame2.flange_a);
end SliderCrank;
```


Plot by OpenModelica
2.5
2
1.5
1
0.5
0
-0.5
-1
2
4
6
8
time
x2
x3

Четвертий спосіб:
використання мови моделювання Modelica та компонентів її стандартної бібліотеки за допомогою середовища Maplesim.

Блок-схема моделі в Maplesim 7



Результати симуляції