УДК 622.276.054

КОМПОНЕНТНО-ОРІЄНТОВАНЕ МОДЕЛЮВАННЯ КІНЕМАТИКИ МЕХАНІЗМІВ МОВОЮ PYTHON НА ПРИКЛАДІ МЕХАНІЗМУ ВЕРСТАТА-ГОЙДАЛКИ

В. Б. Копей
ІФНТУНГ, вул. Карпатська, 15, м. Івано-Франківськ, Україна, 76019
e-mail: vkopey@gmail.com

Комп'ютерне імітаційне моделювання кінематики і динаміки механізмів дозволяє суттєво підвищити ефективність їх проектування. Для спрощення побудови складних моделей доцільно використовувати прості компоненти, тобто застосовувати компонентно-орієнтований підхід. Найбільш відомим сучасним засобом компонентно-орієнтованого моделювання мультидоменних динамічних систем є мова Modelica [1, 2]. Її недоліки випливають з її спеціалізованого характеру - мову важко швидко освоїти програмістам, що володіють якою-небудь імперативною мовою програмування.

В даній роботі поставлено завдання реалізувати компонентно-орієнтований підхід для моделювання кінематики механізмів універсальною імперативною мовою програмування Python. Опишемо принципи побудови програми для моделювання кінематики механізмів на прикладі механізму верстата-гойдалки СКД8-3-4000.

Наведений нижче клас **Frame** описує множину однотипних компонентів-ланок механізмів, крайні точки яких задані координатами **x1**, **y1**, **x2**, **y2**, а довжина - змінною **L**. Функція **eqs()** повертає систему рівнянь, яка описує поведінку компонента. В даному випадку система містить тільки одне рівняння, яке описує незмінну відстань **L** між точками. Функція **plot()** рисує компонент у вигляді лінії за допомогою графічної бібліотеки matplotlib.

```
class Frame:
    def __init__(self,x1,y1,x2,y2,L):
        self.x1,self.y1,self.x2,self.y2,self.L=x1,y1,x2,y2,L
    def eqs(self):
        eqs=[]
        eqs+=[(self.x2-self.x1)**2+(self.y2-self.y1)**2-self.L**2]
        return eqs
    def plot(self):
        plt.plot([self.x1,self.x2],[self.y1,self.y2],'ko-')
```

Наступний клас **Connector** описує множину однотипних компонентів, які являють собою шарнірне з'єднання двох ланок. Тут **e1,e2** - дві ланки, які з'єднуються точками 2 і 1 відповідно. Функція **eqs()** повертає систему рівнянь, яка описує рівність координат точок з'єднання (e1.x2=e2.x1, e1.y2=e2.y1). Функція **plot()** рисує компонент у вигляді червоної точки.

```
class Connector:
    def __init__(self,e1,e2):
        self.e1,self.e2=e1,e2
    def eqs(self):
        eqs=[]
        eqs+= [self.e1.x2-self.e2.x1]
        eqs+= [self.e1.y2-self.e2.y1]
        return eqs
    def plot(self):
        plt.plot([self.e1.x2],[self.e1.y2],'ro')
```

Аналогічно можна розробити клас **Connector2**, який описує нерухоме з'єднання двох ланок. Його функція **eqs()** крім рівнянь, які наведені у класі **Connector**, повинна повертати рівняння, яке описує рівність кутів повороту ланок (tg a1 = tg a2).

Наступний клас **System** описує систему компонентів - плоский механізм. Тут **e** - список компонентів механізму. Функція **eqs()** повертає систему усіх рівнянь, які описують поведінку механізму. В цю систему рівнянь автоматично додаються рівняння кожного компонента системи. Функція **plot()** рисує механізм шляхом виклику функції **plot()** усіх компонентів системи.

```
class System:
    def __init__(self,e):
        self.e=e
    def eqs(self):
```

```
        eqs=[]
        for ei in self.e:
            eqs+= ei.eqs()
        return eqs
    def plot(self):
        for ei in self.e:
            ei.plot()
```

На основі розроблених класів створимо об’єкти: кривошип (**fr0**), шатун (**fr1**), заднє плече балансира (**fr2**), переднє плече балансира (**fr3**), шарнір між кривошипом і шатуном (**con0**), шарнір між шатуном і балансиром (**con1**), нерухоме з'єднання плечей балансира (**con2**), механізм верстата-гойдалки (**s**).

```
fr0=Frame(x1=0.0,y1=0.0, x2=0.81371,y2=0, L=0.81371)
fr1=Frame(x1=0.81371,y1=0, x2=0.65,y2=3, L=3.0)
fr2=Frame(x1=0.65,y1=3, x2=-1.345,y2=3.01195, L=2.0)
fr3=Frame(x1=-1.345,y1=3.01195, x2=-3.6,y2=3, L=2.29)
con0=Connector(fr0,fr1)
con1=Connector(fr1,fr2)
con2=Connector2(fr2,fr3)
s=System([fr0, fr1, fr2, fr3, con0, con1, con2])
```

Для симуляції руху верстата-гойдалки можна поступово змінювати кут повороту кривошипа від початкового значення до кінцевого з невеликим кроком, наприклад у циклі **while**. Тіло цього циклу повинно містити команди обчислення координат точки кривошипа за кутом його повороту, виклик функції розв’язування системи рівнянь об’єкта **s**, запису або візуалізації результатів (**s.plot()**) та збільшення кута на крок. Для розв’язування системи рівнянь можна використати функцію **root()** з пакету scipy.optimize [3], яка розв’язує систему чисельним методом Левенберга-Марквардта. Функція **root()** потребує наближених значень коренів, які можна взяти з попередньої ітерації. Результатом симуляції будуть значення невідомих координат точок механізму в різні моменти часу. Після цього можна розрахувати швидкості і прискорення точок шляхом диференціювання результатів функціями **diff()** або **gradient()** з пакету scipy [3].

Запропоновані принципи можуть бути розвинені у повноцінну систему моделювання кінематики і динаміки складних механізмів будь-якого типу без необхідності застосування спеціалізованих засобів моделювання. Для цього необхідно розробити компоненти, які описують кінематичні пари різного типу. Розроблена програма може бути використана для експрес-аналізу кінематики механізмів верстатів-гойдалок нового типу.

Список використаних джерел

1. Modelica and the Modelica Association [Electronic resource]. – Mode of access: https://modelica.org/
2. Fritzson, Peter A. Principles of object oriented modeling and simulation with Modelica 3.3: a cyber-physical approach / Peter Fritzson. - 2nd edition. - Wiley-IEEE Press, 2014. - 1256 p.
3. SciPy v0.18.1 Reference Guide [Electronic resource]. – Mode of access: https://docs.scipy.org/doc/scipy/reference/