Implementation of an Open Source Planar Linkage Mechanism Simulation and Dimensional Synthesis System

Yuan Chang[a], Chiaming Yen[a]
Department of Mechanical Design Engineering, National Formosa University [a]

***Abstract* — In this paper, an open source cross-platform single-degree-of-freedom planar linkage mechanism simulation and dimensional synthesis system, called Pyslvs, is presented. The geometric constraint solver of SolveSpace is used as one of the kernels in this Python 3 and PyQt5 based software. Another kernel is developed through the symbolic derivation by using the SymPy module.**

Given a series of tracking points and the type of the planar four-bar linkage, the software can be used to calculate feasible linkage topologies and associated dimensions by using the real-coded genetic algorithm, firefly genetic algorithm or differential evolution methods. At the end of this paper, the typical crank-rocker and Jansen's linkage that demonstrate the operation and simulation process are tested in the Windows and Ubuntu operating systems.

Keyword: Open source software; Planar linkage mechanism simulation; Dimensional synthesis.

INTRODUCTION

Toward the start of the advancement of numerous mechanical products, it is generally required to simulate or synthesize planar linkage mechanisms. Therefore, the related computer aided simulation or synthesis software is eventually an indispensable tool in the product development process. The majority of the commercial computer-aided mechanical design software can be utilized to analyze or simulate the linkage mechanism, however without the dimensional synthesis capabilities.

The motivation behind this study is to build up an open source planar linkage mechanism simulation and four-bar linkage dimensional synthesis package, called Pyslvs, would like to wind up plainly a course instructing tool. The simulation function is to extend the Solvespace Geometric Constraint Solver (SGCS) and include a PyQt5 based Graphics User Interface to enable user to simulate various planar linkage system under the Windows and Ubuntu operating systems [1].

In this paper, a set of triangular symbolic and numerical formulation is also created to allow us to add dimensional synthesis function to the package and can be utilized to verify the corresponding simulation results from the SGCS as well.

1.1 Related Works

The Geometric Constraint Solver (GCS) is usually the main core of a parametric computer aided design software package [2].

GeoSolver [3] and Solvespace [4] are the often mentioned open source GCS written in C++. The open source JSketcher [5] GCS is developed in JavaScript.

The Working Model [6] and Simulation and Analysis of Mechanisms [7] are the most popular commercial software for planar mechanism analysis while Planar Mechanism Kinematic Simulator [8] 、Design Analysis and Synthesis of compliant mechanisms [9] and Mechanism Designer and Simulator [10] are similar open source packages.

The Real-coded Genetic Algorithm, RGA [11][12], Firefly Genetic Algorithm [13][14][15] and Differential Evolution, DE [16] methods are often used for the synthesis and optimization of planar mechanisms.

CONCEPT & PLATFORM ARCHITECTURE

The architecture of Pyslvs software is shown in Fig. 1. The core of this linkage simulation is a cross platform Simplified Wrapper and Interface Generator (SWIG) oriented SGCS shared libraries. The triangle constraint solver formulation derived from the SymPy module is provided as the base of fitness equations in the dimensional synthesis module. The graphic user interface of Pyslvs is written in PyQt5 and the program is developed on Eric6 integrated environment.

In the dimensional synthesis module, RGA, Firefly and DE evolutional computation methods are used. All the algorithms are firstly written in Python3 and augmented with the Cython static type declarations to generate the associated shared library for Python3 main program.

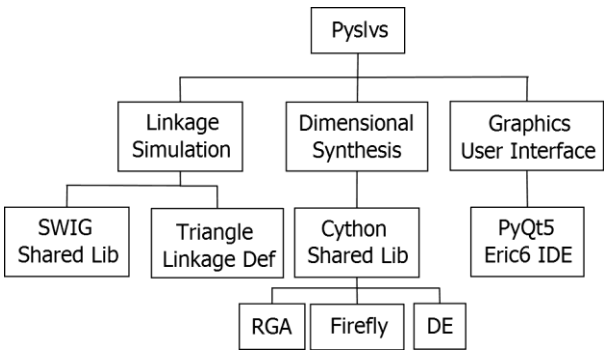

Fig.1 Pyslvs System Architecture

2.1 Solvespace Geometric Constraint Solver

In this section, the application of SGCS is presented to show the basic concept of linkage simulation module in Pyslvs.

The Fig. 2 shows a Crank-Rocker type four-bar linkage with known link lengths, Point1 and Point2 coordinates. We use the aforementioned SGCS library to track the path of Point3 coordinate.

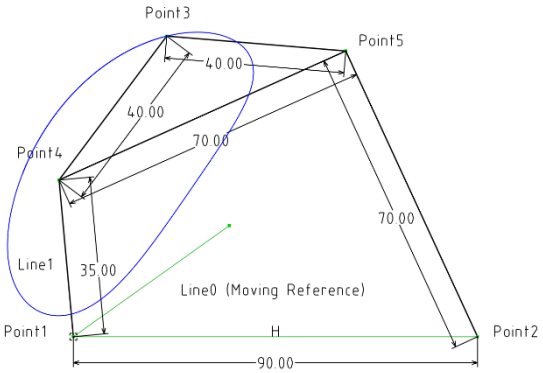

Fig.2 Four-bar Linkage Path Tracking

The pseudo source code for the linkage simulation analysis of Pyslvs is as per the following:

```
Import slvs module
def crank_rocker(rot_angle, input_point_coordinate):
    create slvs system
    setup solver group
    add origin Point
    add normal vector on Sketch Plane
    add Workplane
    add Point1 and Fixed Constraint
    add Point2 and Fixed Constraint
    add Moving Reference Point and Fixed Constraint
    add Rotation Reference Line0
    add Track Point3
    add Point4, Point5 and Triangle Length Constraint
    add Rotation Link Line1
    add Angle Constraint between Line0 and Line1
    Solve all Constraints
    return Track Point3 Coordinate
```

The full source code of this program is shown on (SGCS 2_1 program, 2017), and the execution result is as follows:

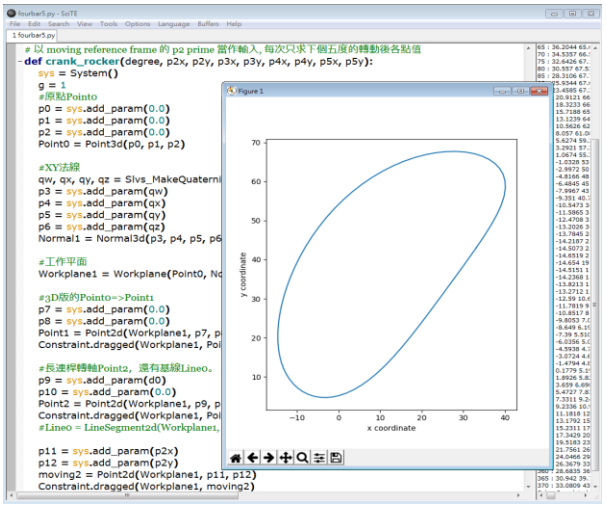

Fig.3 Slvs library Python3 Path Tracking Application

2.2 Triangle Constraint Solver Formulation

In order to derivate the path tracking fitness equations for the dimensional synthesis module of Pyslvs, the SymPy module is utilized. In the accompanying, two triangle based constraint solver equations are displayed.

2.2.1 PLAP Solver Derivation

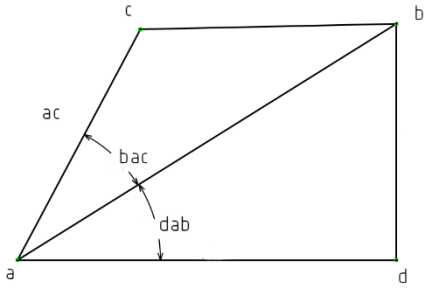

Fig.4 PLAP and PLLP Solver Constraints

PLAP: As appeared in Fig. 4, the Point a coordinates, ac link Length, rotational Angle bac and the Point b coordinates are known. By using the SymPy symbolic module, Point c coordinates can be derived as takes after:

```
#PLAP
from sympy import symbols, sqrt, solve, cos, sin, Abs

# inputs
ax, ay, bx, by, bac, ac = symbols('ax ay bx by bac ac')
# intermediate variables
ab, dab = symbols('ab dab')
ad, bd = symbols('ad bd')
# outputs
cx, cy = symbols('cx cy')
# from a, b coordinates find ab, ad and bd
ab = sqrt((ax-bx)**2+(ay-by)**2)
ad = Abs(bx-ax)
bd = Abs(by-ay)
data = solve(-bd**2+ad**2+ab**2-2*ad*ab*cos(dab
), dab)
# first set of solution
dab = data[0]
cx = ax+ac*cos(dab+bac)
cy = ay+ac*sin(dab+bac)
print("cx=", cx, "cy=", cy)
# second set of solution
dab = data[1]
cx = ax+ac*cos(dab+bac)
cy = ay+ac*sin(dab+bac)
print("cx=", cx, "cy=", cy)
```

2.2.3 Application of Triangle Formulation

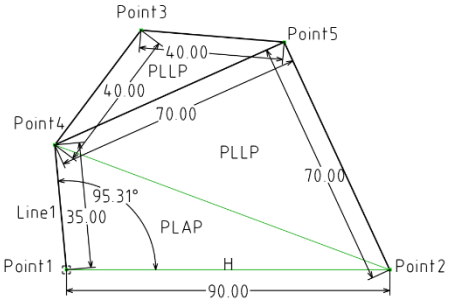

Fig.5 Four-bar Linkage Triangle Decomposition

This section describes how to utilize the PLAP and PLLP definitions to create a Crank-Rocker type of four-bar linkage program as appeared in Fig. 5.

```
def crank_rocker(angle, p1x, p1y, p2x, p2y, len1, len2, len3, len4, len5):
    p4x, p4y = plap(p1x, p1y, len1, angle, p2x, p2y, 0)
    p5x, p5y = pllp(p4x, p4y, len2, len3, p2x, p2y, 0)
    p3x, p3y = pllp(p4x, p4y, len4, len5, p5x, p5y, 0)
    return p3x, p3y
```

The full source code of this program is shown on (Triangle 2_1 program, 2017), and the execution result is as follows:

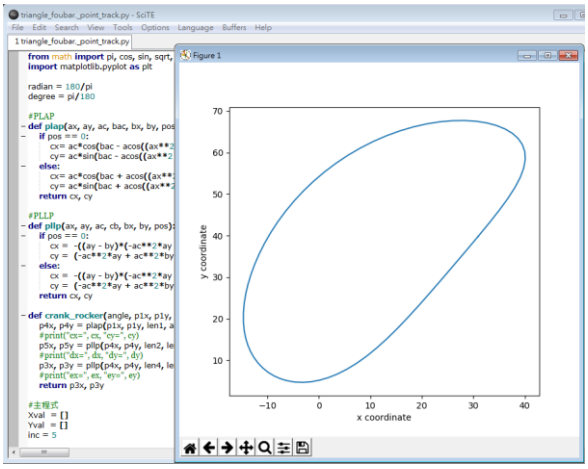

Fig.6 Path Tracking Using Triangular Formulations.

2.2.3 Application of Triangle Formulation

In this section, the evolution design process of the four-bar linkage dimensional synthesis function is introduced.

The RGA, DE and Firefly evolutionary design process can be characterized as follows:

1. Generate the initial population.
2. Evaluate the fitness of each member.
3. Crossover, recombination or move fly.
4. Mutation, generate random vectors or random values.
5. Selection and check whether the termination condition is met.
6. If the condition is not reached, generate new group and got to step 2 until reach the condition.

The UML diagram of these three synthesis pro-grams is indicated in Fig. 7. Although different method has different reproduction, selection and randomness strategies, main basic procedure to converge to best combination of design variables are actually quite similar.

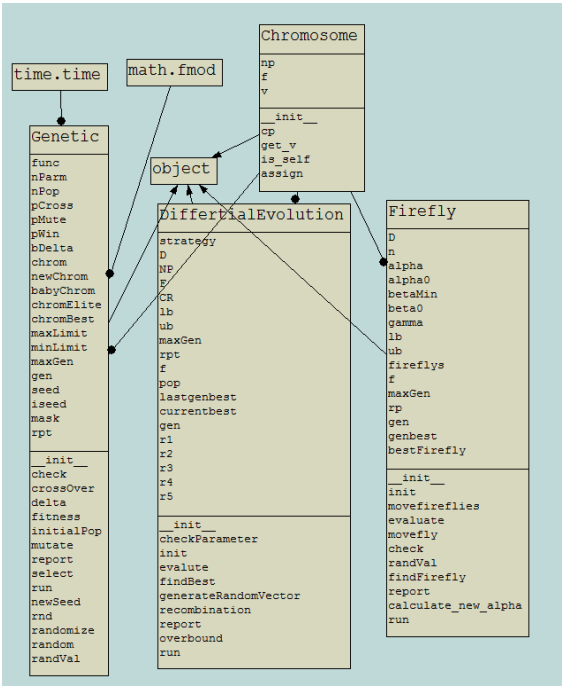

Fig.7 UML of RGA, Firefly and DE programs.

After three evolution synthesis Python3 pro-grams were created, the Cython technology is utilized to compile these programs into shared library to speed up the execution.

EXAMPLES OF PYSLVS

In this section, Pyslvs examples are presented which include the Jansen's walking mechanism and four-bar linkage dimensional synthesis.

3.1 Planar Linkage Simulation

The following describes the basic flow of planar link mechanism simulation using Pyslvs as shown in (https://vimeo.com/218628564):

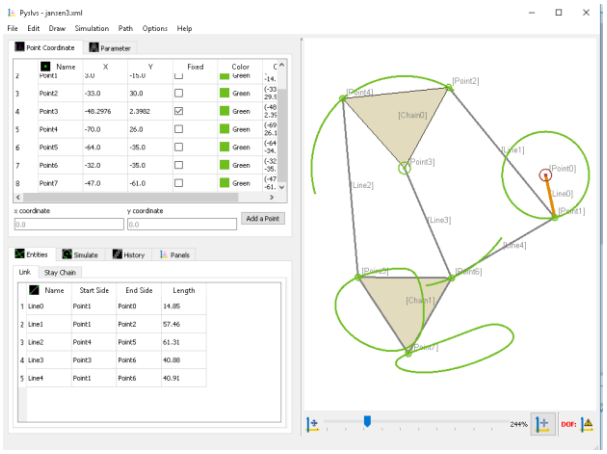

Fig.8 Jansen's Linkage on Pyslvs

1. Use File pull-down menu to open a "New Workbook".
2. Press mouse right button on drawing area to add seven points as shown on Fig. 8.
3. Use "Draw -> New Linkage" to add link connects associated points and "Draw -> New Stay Chain" to add two set of three point links.
4. Under "Simulate" tab press mouse right button toto add a "driver shaft".
5. Under "Point Coordinate" tab, use mouse right button to select Point3 and edit the attribute of Point3 to be fixed.
6. Under "Panel" tab, press "Drive Shaft" and "Play" to activate the drive shaft to simulation.

3.2 Six-Point Four-bar Linkage Synthesis

Next the dimensional synthesis process of four-bar linkage passing six-point is presented as shown in (https://vimeo.com/218428757). The operations are as follows:

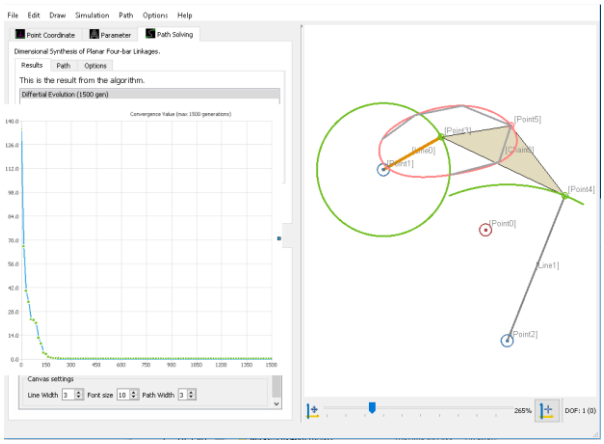

Fig.9 Six-Points Dimensional Synthesis

1. Use "File" pull-down menu to open a "New Workbook".
2. Use left mouse button click "Panels" tab.
3. Click "Path Solving" button.
4. Press right mouse button on drawing area to add six "path points" as shown on Fig. 9.
5. Click "generate" button under "Path Solving" tab to bring up the "Dimensional Synthesis" window.
6. Click "Start" to do synthesis computation.
7. When computation reach 1500 generations, click "Chart" to exam error.
8. Click "merge" button to open synthesis result on the drawing area.

3.3 Ten-Point Four-bar Linkage Synthesis

When the number of passing point increases to 10 points, after the DE evolutionary operation of 1500 generations, from the error value screen, the obvious error value remained, i.e. there are two input points are not closely passed as shows in Figure 10. In other words, as the number of passing points increase, longer computation time is

needed. The operation process is displayed in (https://vimeo.com/218428847):

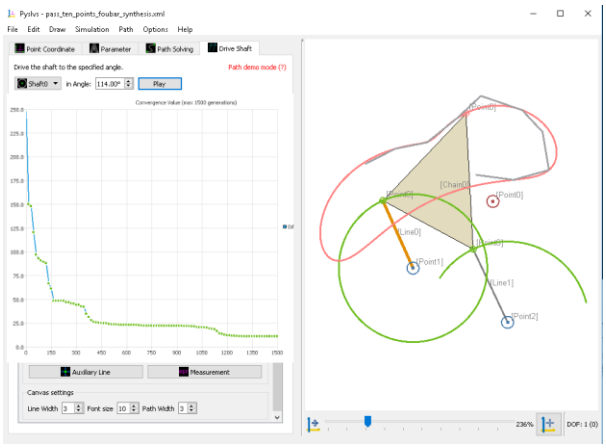

Fig.10 Ten-Points Dimensional Synthesis

CONCLUSIONS

In this study, an open-source planar linkage mechanism simulation and four-bar dimensional synthesis system, i.e. Pyslvs, is developed. The executable files for Windows and Ubuntu operating system as well as source codes can be downloaded from (http://pyslvs.com).

This program has the following characteristics:

1. The Solvespace GCS library developed in this research can be utilized independently with other Python3 program application.
2. The triangle geometric solver formulation can be extended to dynamic analysis of the mechanism, in addition to the dimensional synthesis.
3. Affirmed that the combination of SWIG and Cython technology let C + + and Python3 program development advancement with great performance.

REFERENCES

[1] Pyslvs, May 2017, http://pyslvs.com
[2] Bouma, W. et al., 1995. A Geometric Constraint Solver. Computer Aided Design. 27, 6 (June), 487-501.
[3] GeoSolver, May 2017, http://geosolver.sourceforge.net
[4] Solvespace, April 2017, http://solvespace.com
[5] JSketcher, April 2017, https://github.com/xibyte/jsketcher
[6] Working Model 2D, April 2017, https://goo.gl/EyHa2q
[7] Sam, April 2017, http://www.artas.nl/en/sam
[8] PMKS, April 2017, http://designengrlab.github.io/PMKS
[9] Das-2d, April 2017, https://goo.gl/xPQc1u
[10] Linkage, April 2017, https://goo.gl/3RnNLP
[11] Erbatur F. et al., Optimal design of planar and space structures with genetic algorithms, Computers and Structures, 75 (2000) 209-224.
[12] Omer A. Shpli, Jordan, Genetic Algorithms in Synthesis of Path Generator Four-bar Mechanism with Maximum Mechanical Advantage, Proceeding, The 16th IASTED International Conference on Applied Simulation and Modelling, 154-161, 2007.
[13] Amir H. G. et al., Mixed variable structural optimization using Firefly Algorithm, Computers and Structures 89 (2011) 2325–2336.
[14] Mohammad K. et al., Firefly-inspired Algorithm for Discrete Optimization Problems: An Application to Manufacturing cell Formation, Journal of Manufacturing Systems, 32 (2013) 78–84.
[15] Srivatsava P.R. et al., Optimal Test Sequence Generation using Firefly Algorithm, Swarm and Evolutionary Computation, 8 (2013) 44–53.
[16] Sanjay B. Matekar a, Gunesh R. Gogate, Optimum Synthesis of Path Generating Four-bar Mechanisms using Differential Evolution and a Modified Error Function, Mechanism and Machine Theory, 52 (2012) 158–179.
[17] Arnaud Fabre , Pascal Schreck, Combining Symbolic and Numerical Solvers to Simplify Indecomposable Systems Solving, Proceedings of the 2008 ACM symposium on Applied computing, March 16-20, 2008, Fortaleza, Ceara, Brazil.
[18] Christophe Jermann, Gilles Trombettoni, Bertrand Neveu, Pascal Mathis. Decomposition of Geometric Constraint Systems: a Survey, International Journal of Computational Geometry and Applications, 2006, 16 (5-6), 379-414.
[19] Christoph M. H. and Robert J. A., Symbolic Constraints in Constructive Geometric Constraint Solving, Journal of Symbolic Computation, 23 (2-3), 287-299, 1997.
[20] Crank Rocker Simulation, https://vimeo.com/218413184
[21] Duanling Li, Zhonghai Zhang and J. Michael McCarthy, A constraint graph representation of metamorphic linkages, Mechanism and Machine Theory, 46 (2011) 228–238.
[22] Glenn A. Kramer, Solving Geometric Constraint Systems, AAAI-90 Proceedings, 1990.
[23] Hadizadeh S. Kafash, A. Nahvi, Optimal Synthesis of Four-bar Path Generator Linkages using Circular Proximity Function, Mechanism and Machine Theory, 115 (2017) 18–34.
[24] Jansen Linkage Simulation, https://vimeo.com/218628564
[25] Joan Arinyo R., Soto A., A Correct Rule-based Geometric Constraint Solver, Computers and Graphics, v.21 n.5, p.599-609, September, 1997.
[26] Juan-Arinyo, R. and Soto, A. 1995. A rule-constructive geometric constraint solver. Tech. Rep. LSI-95-25-R, Univ. Politecnica de Catalunya.
[27] Lucas Weihmann et al., Modified Differential Evolution Approach for Optimization of Planar Parallel Manipulators Force Capabilities, Expert Systems with Applications, 39 (2012) 6150–6156.
[28] MechAnalyzer, April 2017, https://goo.gl/f8bYXh
[29] Radovan R. Bulatovic and Stevan R. Dordevic, On the Optimum Synthesis of a Four-bar Linkage using Differential Evolution and Method of Variable Controlled Deviations, Mechanism and Machine Theory, 44 (2009) 235–246.
[30] Reyes Pavón et al., An Adjustment Model in a Geometric Constraint Solving Problem, Proceedings of the 2006 ACM symposium on Applied computing, April 23-27, 2006, Dijon, France.
[31] SGCS 2_1 program, May 2017, https://goo.gl/Weaa1D
[32] Triangle 2_1 program, May 2017, https://goo.gl/AThAH6