Software Optimization Problem Solver for Automated Linkage Design

Kirill Kuprianoff[1(✉)], Christina Shutova[1], and Andrei Vukolov[2]

[1] Energetic Engineering Faculty, Bauman Moscow State Technical University, Moscow, Russian Federation
bdf-1@mail.ru

[2] ELETTRA Sincrotrone Trieste S.C.p.A., AREA Science Park Basovizza, Trieste, Italy
andrei.vukolov@gmail.com

Abstract. The selection and design of the kinematic diagram of the mechanism is the first stage in the creation of a technical device. In the design process, an optimization task inevitably arises, during the solution of which it is necessary to choose the best one from the proposed options according to certain criteria. This article proposes a software solution to automate this process. On the example of the synthesis of a planar mechanism by a given function of the output joint, the advantages of this automated approach are shown. When solving, the two methods are used: enumerating various geometric parameters for a given structural diagram of the mechanism and solving the forward kinematic problem, followed by finding the minimum of the target function, as well as solving the inverse kinematic problem for a given structural diagram of the mechanism.

Keywords: Optimization · Target function · Automation · Design · Planar mechanisms · Linkages

1 Introduction

At the moment, there are a number of specialized software products that allow the analysis of kinematic chains, but at the same time solving only the specific direct kinematics problem, without the possibility of solving the optimization problem and enumerating many solutions [1,3,13,16]. Such packages and software systems include commercial software package ADAMS, which provides the ability to design and optimize kinematic chains. ADAMS is a powerful tool, but requires a high level of knowledge and experience with such programs. In addition, this program is proprietary [10,11,17,18] and has strong licensing constraints. Also CATIA has the kinematic analysis tools which enable 3D modelling of compounds with any degree of freedom and the output of the motion curves of the mechanism parts, but they do not have the ability to optimize generated solutions [12]. There is a modern JavaScript library for modelling, visualization and

D. Pisla et al. (Eds.): EuCoMeS 2020, MMS 89, pp. 451–458, 2020.
https://doi.org/10.1007/978-3-030-55061-5_51

analysis of linkages called `mec2` [9]. It provides the main advantage—simplicity of code visualizing the linkage. But this solution has no possibility to multiply variants of calculation for given structural schema [9]. The another linkage design tool—`WinMeCC` [2,5] is also proprietary. It provides user-friendly interface (Fig. 1[1]) but very limited possibilities to build linkage libraries, open stocks and optimized collections.

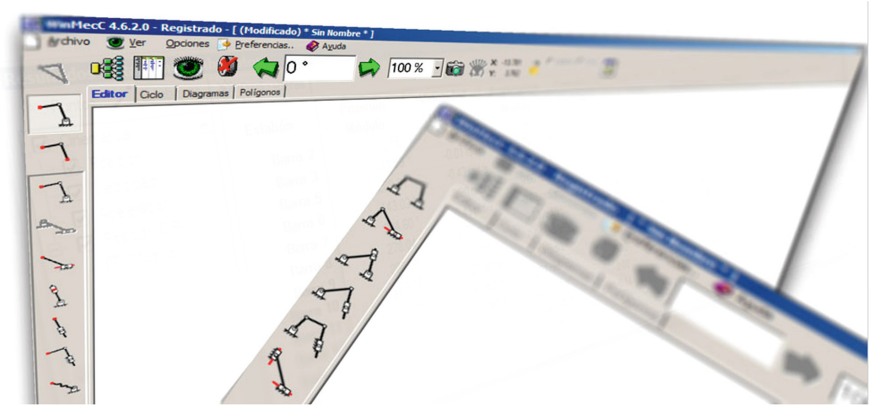

Fig. 1. WinMeCC—main window (advertisement from official website)

The program developed and implemented by the authors which is written in Python programming language is the open source solution aimed primarily to efficient enumeration of many variants of kinematic schemes, which allows, as the first approximation to obtain the semi-optimal or optimal solution. A library of mechanisms has also been created as a set of structural schemes with parametrically defined geometric parameters. Thus, this program gives user the freedom of choice both in the structure of the mechanism limited only by the size of the library, and in the selection of parameters with respect to which there will be a search for an optimal solution. The output of the program is a set of calculated parameters that provide minimized target function. This set of geometric snapshots can then be converted to several formats and visualized as motile mechanism using any of the above mentioned toolchains.

2 Advantages of Computer-Aided Design of Technological Systems

The main idea in the implementation of the program code was the principle of maximally free assignment of the base parameters and the whole model as it should be [6]. At the same time, it is obvious that excessive freedom of input of the initial data, meaning the creation of directly structural schemes from basic

[1] http://en.winmecc.com/.

elements, such as points and vectors, only leads to a complication of the initial data. The limitation in the form of a library of mechanisms makes it possible to speed up the process of formulating parameters for the program and at the same time avoids wasting time on the calculation of solutions in which it is impossible to obtain a satisfactory result. In addition, this approach reduces the number of errors that can be made when setting conditions in the program.

As a result, the user gets the opportunity to quickly and easily obtain an initial approximate solution for his task, with a minimum of effort spent directly on the formulation of the problem and the methods for solving it. This provides several advantages:

1. Significantly accelerate the process of obtaining a preliminary design, which can then be refined using more accurate software systems;
2. Elimination of deliberately unsatisfactory options, which also saves time spent on design and avoids further problems when working out a specific solution.

The resulting program has the ability to integrate with other programs. The output format of the program is simple to understand and easy to change, which allows the user to transfer geometric parameters to another program to visualize the solution, or to carry out further refinement calculations.

3 Synthesis of a Mechanism for a Given Function of the Output Joint: An Example of a Solution

Setting the initial data: Using the built-in library, the user chose several structural schemes for which the calculation will be done, and also sets the necessary output motion curve $\phi_{out} = f(\phi_1)$ for each scheme having one degree of freedom. The selected mechanisms with the names assigned to them are displayed in the left window. At the next stage, the parameters that will change during the calculation together with their range of variation, and the constants are set in the right window (Fig. 2).

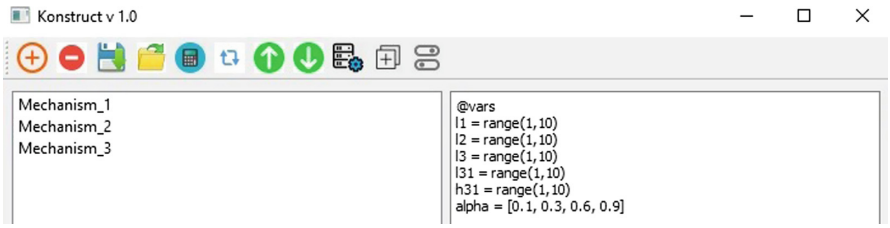

Fig. 2. Main window of the program

Examples of selection are shown in (Fig. 3).

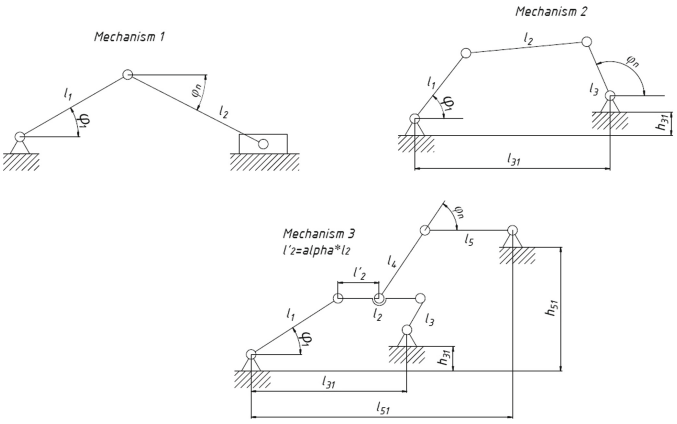

Fig. 3. Selection

3.1 Calculation of the Direct Kinematic Problem

The program performs optimization dimensional synthesis for each mechanism selected from the library, thus allowing to compare the results between different structural schemes to find the better solution. In the program code, equations for the selected mechanisms are automatically generated and solved, taking into account the selected variable parameters. The `result.txt` file is created at the output with all satisfactory parameters for which the task has a solution and the corresponding values of the target function. In our example, we need to find a mechanism for which the motion curve of the output joint is as close as possible to the required one and the target function is defined as

$$\Delta = \int_{\phi} (\phi_{out}^{calculated} - \phi_{out}^{given})^2$$

Figure 4 represents the obtained result.

The most optimal solution here is found by the minimum value of the target function and the motion curves are displayed for it (Fig. 6), where the red line is the user-defined output law for changing the angle, and the blue one is the obtained law by solving systems of equations.

During the calculation process, all internal parameters can be used to iteratively calculate not only the target function, but also any other functionality that the user sets manually, for example, the sum of the lengths of the links. The resulting functionalities are also added to the `result.txt` file. The architectural solution of the program implies the creation in the global memory area of all objects of the selected classes of mechanisms, as well as current variable parameters. Below is the part of the code that describes the function that starts the optimization calculation (Fig. 5).

```
l1: 4 l2: 8 l3: 5 l31: 6 h31: 9 ; Delta:928597.34
l1: 4 l2: 8 l3: 5 l31: 7 h31: 1 ; Delta:417189.43
l1: 4 l2: 8 l3: 5 l31: 7 h31: 2 ; Delta:1291859.53
l1: 4 l2: 8 l3: 5 l31: 7 h31: 3 ; Delta:869880.21
l1: 4 l2: 8 l3: 5 l31: 7 h31: 4 ; Delta:1538204.59
l1: 4 l2: 8 l3: 5 l31: 7 h31: 5 ; Delta:1294309.70
l1: 4 l2: 8 l3: 5 l31: 7 h31: 6 ; Delta:1202785.13
l1: 4 l2: 8 l3: 5 l31: 7 h31: 7 ; Delta:1211973.13
l1: 4 l2: 8 l3: 5 l31: 7 h31: 8 ; Delta:1294276.20
l1: 4 l2: 8 l3: 5 l31: 7 h31: 9 ; Delta:1437601.24
l1: 4 l2: 8 l3: 5 l31: 8 h31: 1 ; Delta:1065312.52
l1: 4 l2: 8 l3: 5 l31: 8 h31: 2 ; Delta:1802396.81
l1: 4 l2: 8 l3: 5 l31: 8 h31: 3 ; Delta:1695311.18
l1: 4 l2: 8 l3: 5 l31: 8 h31: 4 ; Delta:2480455.42
l1: 4 l2: 8 l3: 5 l31: 8 h31: 5 ; Delta:2137663.10
l1: 4 l2: 8 l3: 5 l31: 8 h31: 6 ; Delta:1957415.55
l1: 4 l2: 8 l3: 5 l31: 8 h31: 7 ; Delta:1897919.13
l1: 4 l2: 8 l3: 5 l31: 8 h31: 8 ; Delta:1931839.98
```

Fig. 4. Resulting dataset

```
1    def calcModel(self):
2        const_text, var_text = Calc.texthandler(self.code.toPlainText()) #
             Parsing code in right                window
3        Objects = deepcopy(self.Objects) # creating the copy of objects in left
              window
4        Variable_data = Calc.data_init(Objects, const_text) # Initialization of
              variables
5        Elements = Calc.elem_builder(Objects) # Building elements based on
             variables
6        Calc.elements_reInit(Elements) # Building elements based on variables
7        calc_text = Calc.var_texthandler(var_text)
8        exec(calc_text) # Run calculation for the generated task
```

Fig. 5. CalcModel function

The equations to be solved are compiled with respect to the variable parameters. The use of the global memory region allows sequential calculations, that is, to use the results obtained in one object, as source data for subsequent objects.

3.2 Visualization

To visualize the optimized kinematic scheme, a written script is used that converts the format of the program results to the JSON format which allows further work with the obtained linkage. For example constructing of the motion law of any joint. Choosing a specific kinematic scheme and link sizes from the result file, the user can convert this data and transfer it to the program for visualization. Converted input file obtained using the script, the constructed and visualized linkage are shown in (Fig. 7).

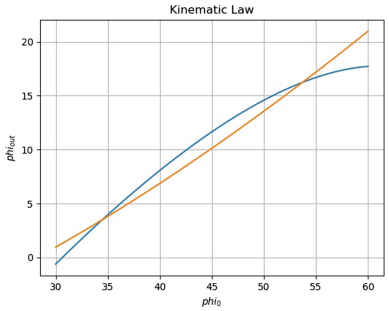

Fig. 6. Comparison of kinematic functions

```
model = {
  id: '4bar',
  gravity: false,
  nodes: [
      { id:"A0",x: 0+dx,y:0+dy,base:true },
      { id:"A", x:-57+dx,y:23+dy },
      { id:"B", x:-40+dx,y:80+dy },
      { id:"B0",x:10+dx,y:90+dy,base:true }
  ],
  constraints: [
      { id:"a",p1:"A0",p2:"A",len:{ type:"const" } },
      { id:"b",p1:"A", p2:"B",len:{ type:"const" } },
      { id:"c",p1:"B0",p2:"B",len:{ type:"const" } }
  ]
};
```

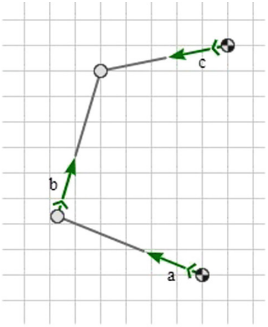

Fig. 7. Converted input file and graphic representation of the mechanism (screenshot of `mec2` web canvas)

4 Development Prospects: Training

In addition to the direct use of the written program for designing linkages, when used, it becomes possible to process and analyse the obtained array of results, which allows to trace the influence of individual geometric parameters on the kinematics of the system. In addition, the use of structure libraries over time will develop the user's intuitive thinking when choosing a kinematic scheme and variable parameters. Thus, the program will increase understanding of design issues among inexperienced designers, and for professionals it will become an additional tool for quick verification and analysis of their own ideas. Developing the idea of learning, the next step will be the implementation of full automation, in which the design experience is somehow saved in the form of a database and used in machine learning [14,15]. In this case, a trained ML system will be able to offer initial structural schemes itself, thereby leaving human user input only for the stage of formulating the problem. This approach can also stimulate development of special mechanics and assistance tools such as mechanisms with flexible parts and limited joint movement [4,7,19].

5 Conclusion

Optimization problems in the design of mechanisms were successfully solved in the implemented program code. On the example of solving the optimization problem of the synthesis of a planar mechanism according to the known motion curve of the output joint, it was shown:

- The ability to automate the design process, increasing the speed of development and the quality of the result;
- The ability to process a dataset for AI training based on the analysis of an existing array of solutions obtained using the program.

Thus, the written program is the first step to fully automate the design process [8]. The development of such programs is a promising direction, allowing us to take the class of optimization tasks to a new level. The next step for development of this program is to make it open source to increase possibilities for other developers to improve the code and to develop their own tools based on the free software model [17].

References

1. CATIA version 5-6 release 2015. Documentation in HTML format
2. WinMeCC - manual de usuario. Online User Manual. http://winmecc.uma.es/guia-usuario-WinMecC.pdf. Accessed 20 Jan 2020
3. ADAMS MSC team et al.: ADAMS User's Manual. MacNeal-Schwendler, Inc., Oakdale (2001)
4. Baryshnikova, O.: The creation of clean robots on the basis of a flexible elastic thin-walled elements. Int. J. Mech. Eng. Robot. Res. **8**(5), 759–763 (2019). https://doi.org/10.18178/ijmerr.8.5.759-763. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85071775978&doi=10.18178%2fijmerr.8.5.759-763&partnerID=40&md5=5b3c791bb6c0aae1e8bda9d4957e6559
5. Bataller, A., Ortiz, A., Cabrera, J.A., Nadal, F.: WinMecC: software for the analysis and synthesis of planar mechanisms. In: Wenger, P., Flores, P. (eds.) New Trends in Mechanism and Machine Science, pp. 233–242. Springer, Cham (2017)
6. Belonozhko, P.P.: Assembly and service robotic space module. Mathematical model of the reduced system, pp. 337–347. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-32579-4_27
7. Chernaya, L., Safnonoff, I., Vukolov, A.: Design of cam mechanisms with swinging roller follower: the modern algorithm-based approach. In: Castejón, C., García-Prada, J.C., et al. (eds.) Proceedings of ISEMMS 2017 the 2nd International Symposium on the Education in Mechanism and Machine Science. Universidad Carlos III de Madrid, Springer, Madrid (2018). Preprint edn
8. Ghernik, A., Smirnov, N.: Development of the software for the automatization of kinematic calculation for closed kinematic scheme mechanisms, vol. 14 (2010). (in Russian)
9. Gössner, S.: Fundamentals for web-based analysis and simulation of planar mechanisms. In: Proceedings of the 7th European Conference on Mechanism Science, pp. 215–222 (2019). https://doi.org/10.1007/978-3-319-98020-1_25

10. Klein Breteler, A.: Kinematic optimization of mechanisms: a finite element approach (1987). http://resolver.tudelft.nl/uuid:eca45c68-9353-4a64-8882-ab0b935eedc8
11. Kovalenko, V.A.: State of deployment analysis of free software in institutions of the educational system in Russia (in Russian). Pedagogical Educ. Russia **6**, 188–192 (2013)
12. Krokhmal, N.N., Krokhmal, O.: General method of optimization kinematic synthesis of planar lever mechanisms based on its structural properties by example of the eight-link mechanism. In: Proceedings of the 15th National Conference on Machines and Mechanisms (iNaCoMM 2011) (2011). https://www.researchgate.net/publication/320023939_General_method_of_optimization_kinematic_synthesis_of_planar_lever_mechanisms_based_on_its_structural_properties_by_example_of_the_eight-link_mechanism
13. Pucheta, M., Cardona, A., Cugnon, F.: Kinematic optimization of planar linkages using Samcef BOSS-Quattro software. In: 5th Asian Conference on Multibody Dynamics 2010, ACMD 2010, vol. 2, pp. 800–807 (2014)
14. Sarkar, D., Bali, R., Sharma, T.: The Python Machine Learning Ecosystem, pp. 67–118. Apress, Berkeley (2018). https://doi.org/10.1007/978-1-4842-3207-1_2
15. Semeraro, G., Esposito, F., Malerba, D., Fanizzi, N., Ferilli, S.: Machine learning + on-line libraries = IDL. In: Peters, C., Thanos, C. (eds.) Research and Advanced Technology for Digital Libraries, pp. 195–214. Springer, Heidelberg (1997)
16. Suyatinov, S.I.: Conceptual Approach to Building a Digital Twin of the Production System, pp. 279–290. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-32579-4_22
17. Titov, A., Vukolov, A.: Free and open source software for technical texts editing, its advantages and experience of usage on TMM training in Bauman University. In: Castejón, C., García-Prada, J.C., et al. (eds.) Proceedings of ISEMMS 2017 the 2nd International Symposium on the Education in Mechanism and Machine Science. Universidad Carlos III de Madrid, Springer, Madrid (2018). Preprint edn
18. Vukolov, A.: Free and open source software applications for education of TMM discipline in Bauman University. In: Wenger, P., Flores, P. (eds.) New Trends in Mechanism and Machine Science: Theory and Industrial Applications, pp. 253–260. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-44156-6_26. http://dx.doi.org/10.1007/978-3-319-44156-6_26
19. Zudina, O.V.: Process-oriented approach into Rao X simulation modeling system. In: Uhl, T. (ed.) Advances in Mechanism and Machine Science, pp. 531–536. Springer, Cham (2019)