International Journal
Information Technology Applications (ITA)

Volume 8, Number 1, July 2019

COMPUTATIONAL MECHANICS AND OPTIMAL PROGRAMMING PARADIGM

Michal Kráčalík

Abstract:

The aim of the paper is to find an optimal programming paradigm in context of the computational mechanics that is mainly focused on the numerical solution of the partial differential equations. Two paradigms - procedural and objected oriented (OOP) are compared along with a simple software development representing by a GUI. Objected oriented programming can be viewed as a good compromise among the time required for the production of the code and numerical tests procedures in the computational mechanics. Procedural programming can be suitable for a quick testing of the numerical solution of partial differential equation. If the project is extended, the code can be for reusability rebuild to classes in objected oriented programming. One computational example is shown in the paper and limitations of the selected programming paradigm are described.

Keywords:

Partial differential equations, objected oriented programming, creep curve, contact.

ACM Computing Classification System:

Applied computing → Physics

Introduction

Computational mechanics is mainly focused on the finding the approximate solution of the partial differential equations (PDE) [1]. PDEs describe natural laws mathematically and these equations are usually solved by numerical method like finite element method (FEM), finite difference method (FDM), boundary element method (BEM) etc. [1]. There is plenty of commercial and open source specialized "graphical" software like Ansys, Abaqus, Simflow, OpenFOAM, CalculiX and many more dealing with the numerical solution of PDEs (especially in the field of the continuum mechanics). Mathematical software like Mathematica, Maple or Matlab are able to handle with numerical solutions of PDEs as well. Free packages like FEniCS [2] or Code_Aster [3] are in the development and used in the various field of the continuums mechanics. Software/Toolboxes based on Matlab have also been developed [4], [5]. Once the solutions of PDE is known, the results are normally visualized in 2-D or 3-D graphs. Standardly is tested the influence of the input variables on the results. Therefore, procedural and functional programming are probably the most used in the computational mechanics. Object-oriented programming (OOP) finds in context of the computational mechanics mainly place in software development [2], [4].

According to author´s experience, it cannot be predicted whether the piece of "PDE-Testing code" will serve as a basis for the new software development or not. Hence, choosing the appropriate programming paradigm can be crucial.

The aim of the paper is to compare two programming paradigm and simple software development in the framework of Computational mechanics and in prospect of future software development. Procedural programming and OOP paradigm will be tested on the computational example; the simple software development will be expressed through the simple GUI. No exception handling or unit testing are included in the example and the Python programming language (Ver. 3.7.0) will be used.

1 Computational Example

The creep curve represents the computational example. The creep curve relates the longitudinal creepage (slip) with the tangential forces in the rolling/sliding contact [6]. Such contact situation can be found for instance between the rail and the train wheels [7], [8]. The computation is based on the equations from [6] and the input variables are listed in (Tab.1) (The underlying PDEs can be found in [9]).

Table 1. The input variables

Variable	Unit	Notice
Normal force per unit thickness (F_N)	N/mm	-
Friction coefficient (μ)	-	0.5 for steel/steel contact
Young's modulus (E)	MPa	Material constants
Poissons's ratio (ν)	-	
Maximum contact pressure (P_{max})	MPa	-
Longitudinal slip (γ)	-	%/100

The full-slip state is defined as:

$$\gamma^{\mathrm{fs}} = 2(1-\nu)\mu P_{max}/G, \tag{1}$$

where G is the shear modulus defined as:

$$G = E/2(1-\nu). \tag{2}$$

The tangential force T is defined as:

$$T = \begin{cases} \mu F_N\left(1-\left(1-\left|\frac{\gamma}{\gamma^{\mathrm{fs}}}\right|\right)^2\right) & |\gamma| < \gamma^{\mathrm{fs}} \\ \mu F_N & |\gamma| \geq \gamma^{\mathrm{fs}} \end{cases} \tag{3}$$

The code was written in the Scientific Python Development Environment Spyder (Ver. 3.3.1) using Python libraries numpy (Ver. 1.15.4) for numerical computation, matplotlib (Ver. 3.0.0) for graph visualisation and tkinter (Ver. 8.6) for GUI. The GUI with the given input variables is depicted in (Fig.1) and the creepage curve in (Fig.2).

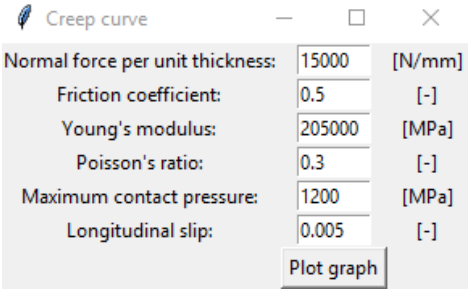

Fig.1. Simple GUI with input variables.

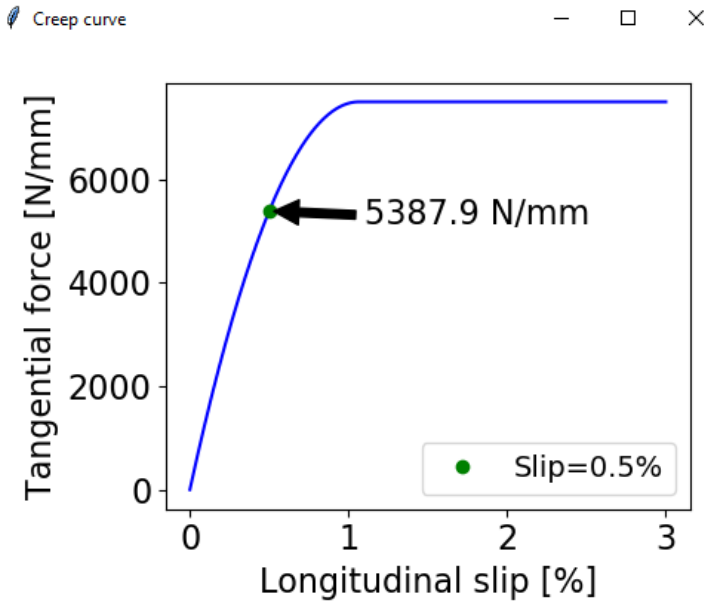

Fig.2. The creep curve – the relationship between longitudinal creepage (slip) and tangential force in rolling/sliding contact.

2 Results and Discussion

The main results are summarised in (Tab.2). Simply assuming the linear dependency between the size of the programming code and the programming time, the simple GUI requires a much more programming time then an OOP. On the other hand, the difference between the OOP and procedural programming is not pronounced.

Here has to be said that a simple GUI can probably be programmed in the more efficient way through the loop. Currently, the one entry requires a one manually written bunch of code. Nevertheless, the GUI creation generally slows down the numerical testing of the physical problem and should be left to software engineers in the next phase of the projects. OOP can be viewed in this context as a compromise between the procedural programming and possible future software development but it is not recommended for all applications.

According to author´s scientific experience, OOP comes on the scene, if the developed procedure turns into a replicable entity. Otherwise, procedural programming is suitable as well. Once the piece of code is suitable for an OOP, it is still not time consuming to create a new class. Implementing known physical problem, OOP can be used from the beginning.

Table 2. Assumed linear dependency between the size of the programming code and time

Paradigm	Size of the programming code (in lines)	Programming time (ratio to Procedural programming)
Procedural	52	1
OOP	61	1.17
OOP+GUI	157	3.02

The variety of a computational mechanics problem cannot be fully covered by two programming paradigm. Recursive programming is used as well [10], [11]. However, recursive programming technics can be implemented into the classes in the multi-paradigm language as for instance in the paper used Python.

Conclusion

This paper tests two programming paradigm (procedural and objected oriented) along with a simple software development at one computational example in order to find an optimal paradigm suitable for an usage in the computational mechanics. The code written using objected oriented paradigm needs only a few more line than procedurally written code. The GUI slows a numerical testing of the underlying partial differential equations and should be left to the software developer in case of turning the code into a real software development. Procedural programming is suitable for a quick testing of the ideas without requirements on the reusability of the code. These two programming paradigm and functional programming are probably most used in the computational mechanics and other programming paradigms are excluded from the paper.

References

[1] Oishi, A., Yagawa, G. (2017). Computational mechanics enhanced by deep learning. *Computer Methods in Applied Mechanics and Engineering*, 327, 327-351, ISSN 0045-7825, doi:https://doi.org/10.1016/j.cma.2017.08.040

[2] Rodriguez, M. A., Augustin, C. M., Shadden, S. C. (2019). FEniCS mechanics: A package for continuum mechanics simulations. *SoftwareX*, 9, 107-111, ISSN 2352-7110, doi:https://doi.org/10.1016/j.softx.2018.10.005

[3] Antonutti, R., Peyrard, C., Incecik, A., Ingram, D., Johanning, L. (2018). Dynamic mooring simulation with Code_Aster with application to a floating wind turbine. *Ocean Engineering*, 151, 366-377, ISSN 0029-8018, doi:https://doi.org/10.1016/j.oceaneng.2017.11.018

[4] Richard, B., Rastiello, G., Giry, C., Riccardi, F., Paredes, R., Zafati, E. Lejouad, C. (2019). CastLab: an object-oriented finite element toolbox within the Matlab environment for educational and research purposes in computational solid mechanics. *Advances in Engineering Software*, 128, 136-151, ISSN 0965-9978, doi:https://doi.org/10.1016/j.advengsoft.2018.08.016

[5] Dallemule, M. (2013). Buckling resistance of arch structures. Ph.D. dissertation, *Slovak University of Technology in Bratislava*

[6] Brouzoulis, J. (2014). Wear impact on rolling contact fatigue crack growth in rails. *Wear*, 314, 13-19, ISSN 0043-1648, doi:https://doi.org/10.1016/j.wear.2013.12.009

[7] Wiedorn, J., Daves, W., Ossberger, U., Ossberger, H., Pletz, M. (2018). Numerical assessment of materials used in railway crossings by predicting damage initiation – Validation and application. *Wear*, 414-415, 136-150, ISSN 0043-1648, doi:https://doi.org/10.1016/j.wear.2018.08.011

[8] Mai, S. H., Gravouil, A., Nguyen-Tajan, M. L., Trollé, B. (2017). Numerical simulation of rolling contact fatigue crack growth in rails with the rail bending and the frictional contact. *Engineering Fracture Mechanics*, 174, 196-206, ISSN 0013-7944 doi:https://doi.org/10.1016/j.engfracmech.2016.12.019

[9] Johnson, K. L. (1985). *Contact Mechanics*. Cambridge University Press. doi:10.1017/CBO9781139171731

[10] Sulejmanpasic, T., Ünsal, M. (2018). Aspects of perturbation theory in quantum mechanics: The BenderWu Mathematica ® package. *Computer Physics Communications*, 228, 273-289, ISSN: 0010-4655, doi:https://doi.org/10.1016/j.cpc.2017.11.018

[11] Gil, A., Segura, J., Temme, N. M. (2017). Efficient computation of Laguerre polynomials. *Computer Physics Communications*, 210, 124-131, ISSN 0010-4655 doi:https://doi.org/10.1016/j.cpc.2016.09.002

Author

Dr. Michal Kráčalík
Zurndorf, Austria
michal.kracalik@gmail.com

Dr. Michal Kráčalík (1985) finished his PhD. study at University of Leoben in 2015 and Master study at Faculty of Mechanical Engineering, Slovak University of Technology (STU) in Bratislava in 2012. He worked as a researcher in Leoben and he currently works as a CAE specialist in the industry. His specialization is investigation of the mechanical behaviour of metallic materials using numerical simulations. Dr. Kráčalík is the co-author of several CC articles and he would like to supervise bachelor and master thesis at universities.