The Journal of Open Source Education

Python-based Lagrange analytical mechanics course

Víctor A. Bettachini[1,2], **Mariano A. Real**[3,4], **and Edgardo Palazzo**[5]

1 Universidad Nacional de La Matanza - UNLaM, Buenos Aires, Argentina. **2** Instituto Geográfico Nacional - IGN, Buenos Aires, Argentina. **3** Instituto Nacional de Tecnología Industrial - INTI, Buenos Aires, Argentina. **4** INCALIN, Universidad de San Martín - UNSAM, Buenos Aires, Argentina. **5** Universidad Tecnológica Nacional - UTN, Buenos Aires, Argentina.

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Submitted: 03 September 2024
Published: unpublished

Summary

We present a code-based undergraduate course on analytical mechanics for engineering students with little to no prior programming knowledge. This 16-week flipped classroom (**?**) course provides skills to calculate dynamics and strains of simple mechanical devices, modelled as rigid bodies by solving Euler-Lagrange equations. Each example and practice exercise is solved using computer-based analytical and numerical calculations focusing students' attention on physics modelling and not on repetitive mathematical tasks. This approach also aims to improve creativity, the students have to solve problems by trial and error (Hoffmann et al., 2021).

The course addresses specific regional issues faced by third-year Latin American students (mid-career), that by then have learned how to solve ordinary differential equations. Theory and examples exercises, along with the *Python* code that solves them are presented in *Jupyter notebooks* run online to avoid installation and hardware requirement issues. Currently, the material is available in a GitHub repository in Spanish and has only been partially translated into English.

Statement of need

Latin American public universities face two simultaneous constrains: tight budgets and the need to accommodate their classes' schedules to day-working students (Vallejo et al., 2022). These cash-stripped universities seldom avail computing resources for courses that are not directly related to computer science or programming. Also, as undergraduate programs on engineering at Latin American universities are usually longer than the three-year bachelor's degrees at their Anglo-Saxon counterparts, it is quite common for students to already be part of the labour market while studying. As a result, they have tight schedules and are often unable to attend to university during daytime hours.

The course presented addresses those issues by providing a free, online, and asynchronous learning environment allowing students to study at their own pace through the flipped classroom approach (Moraros et al., 2015). In advance to weekly meetings, students are required to study the theory and examples provided in the notebooks, as well as to initiate solving the accompanying exercises. During those evening meetings, whether online or in person, students are encouraged to ask questions and discuss the problems they could not solve with the teaching staff.

Basis for the syllabus

Traditionally, systems addressed in analytical mechanics courses are kept as simple as possible, to limit the extent of the mathematical work required. So, modelling of multiple machine parts is seldom undertaken, as that would lead to a level of complexity sometimes untenable for students and teaching staff working on the blackboard or paper. This course aims to avoid this pitfall by taking advantage of the relative simple syntax of modern programming languages to tackle mathematical problems. In this way it is possible to rapidly introduce life-like problems avoiding oversimplifications to the students.

The required modelling as well as algebraic and calculus operations to generate the Euler-Lagrange differential equations are performed using *physics.mechanics*, the symbolic dynamics sub-package of the *SymPy* library (Meurer et al., 2017). Its code was ported from the *PyDy* library, a replacement of *Autolev* (Levinson & Kane, 1990),a commercial software that instrumentalised the Kane's method (Kane & Levinson, 1985). As stated in the online textbook for the Multibody Dynamics course at TU Delft, a successor to the one *PyDy* was developed for, this method avoids accounting for non-conservative forces with Lagrange's multipliers, but it requires modelling forces in the system (Jason K. Moore, 2024). Our choice was instead to make students model systems solely by their energy, a more traditional approach, in order to immerse them into a radically different way of solving mechanical problems in their first contact with analytical mechanics. We think that when facing problems requiring a more efficient method, they will be able to apply such other less abstract methods.

Although *physics.mechanics* provides functionality for deriving equations of motion using Lagrange's method, this course aims for the student to follow the standard mathematical notation and procedures, as they would have done on paper. The idea is to ensure that students can verify each step of the process and only later rely on functions built around these steps, avoiding any *black box*.

We would like to emphasise that the course is not about teaching programming, nor about high-performance modelling of mechanical systems. The aim of employing the computer is to free-up students from the repetitive nature of the calculations, so they can focus on the physical aspects of the problems. The deliberate decision that everything get solved by code, even the earliest examples, aims to reinforce the advice given to students to avoid solving the initial problem sets on paper. Some students did so at earlier editions of the course, only to got stuck later while solving more complex problems without the computer help. By slight modifications over the Python code presented by the teaching staff, students build their own library of solutions to address mechanical modelling challenges. Once the students generate the Euler-Lagrange equations, their numerical solutions are obtained using the *Scipy* library (Virtanen et al., 2020), and plotted using *Matplotlib* (Hunter, 2007) to better understand the physical implications of the solutions.

Overview, Content, and Structure

Full course material is available in a GitHub repository in Spanish, with an ongoing translation to English. The first twelve folders contain the course material, each one corresponding to a unit: 1. Course methodology, Newtonian physics and Sympy introduction. 2. Degrees of freedom, generalized coordinates and energy. 3. Euler-Lagrange mechanics, Euler-Lagrange equations. 4. Constraints as a function of coordinates. 5. Numerical solving of Euler-Lagrange equations. 6. Constraint reactions and Lagrange multipliers. 7. Non-conservative forces in the Euler-Lagrange framework. 8. Rigid-body and inertia tensor. 9. Rigid-body, Euler equations. 10. Oscillations in single degree of freedom (SDoF) systems, forced oscillations and discrete systems. 11. Oscillations multiple degrees of freedom (MDoF) systems. Normal modes of discrete systems.

Each folder contains Jupyter notebooks with the required theory for the unit subject alongside the code that solves example exercises. The students only need to modify that code to solve the exercises proposed at the accompanying problem sets. It is worth mentioning that many problems are modifications of problems presented in the course bibiography, and that they are cited, to help the students follow possible issues and to induce them to further use the textbooks. The problem sets are provided in PDF format alongside their LaTeX source and figure files, allowing their customisation. The number of exercises in each problem set, while still being illustrative of the variety of the unit subject applications, is kept small in order to make their solving mandatory on a weekly basis. Those of units 8, 9 and 11 are exceptions, requiring two weeks each, as they deal with subjects that had shown to be somewhat more demanding to students.

Two further weeks complete a 16-week schedule. These are reserved not only for the students to submit overdue exercises but, mainly, to perform an oral presentation on how they solved a final project. Its aim is to calculate torques and forces that the motors of a simplified factory robotic arm should apply to make it perform a sequence of movements. As it requires the student to master the skills acquired during the first nine units, its statement is presented at the second week for that unit. This arrangement gives enough time for the students to consult on its difficulties and prepare the presentation. The oral examination is intended to gauge the students' learning, not only on the physics and computational skill required to solve this kind of problems, but also on how to provide a well planned oral presentation.

Implementation

The *Google Colaboratory* service allows students to read and execute Jupyter notebooks, as it currently demands no payment and can be accessed from any internet browser. At *UNLaM*, the university where the course is taught, *SageMaker StudioLab*, *GitHub Codespaces*, *Cocalc* or indeed *Kaggle* had also been tested for this purpose. Nevertheless, Colab, as it is commonly known, is currently used because it provides a useful feature for students to pose questions via side-notes to each cell of the notebooks. Teaching staff can reply them individually, and students can re-reply, thus providing an asynchronous interaction channel in between the weekly synchronic meetings.

Students are required to submit their solution to the complete course's problem sets. *MS Teams* is used to assign and keep track of student's work, but any LMS, such as the open source *Moodle*, can fulfil this task. Teaching staff check the submissions and, if required, returns them with comments to correct them. This way, students are encouraged to solve all exercises, as they are mandatory to pass the course, and to ask for help when they are stuck.

Conclusions

The mechanical engineering programme is relatively new at UNLaM, so the number of students per class is still low, around eight, thus allowing personalised tracking of student's progress. Larger audiences will provide a challenge, probably requiring to include new teaching assistants as well as introducing automatic grading, to somewhat keep the current methodology.
For the time being, feedback from students consistently indicates a high level of satisfaction with this course, especially with its code-driven aspect. Additionally, students express interest in the final examination as it provides an opportunity to apply both their presentation skills and the knowledge acquired throughout the course. In relation to the flipped classroom model, students acknowledge that it requires a grater effort, but a majority of them agree that it is a positive and beneficial implementation. This is in line with previous research on the flipped classroom model for advances mechanical engineering courses (**?**).

The authors are confident that the methodology employed in this course offers greater practical utility to students in subsequent subjects and their professional lives, surpassing the benefits of a traditional course.

The Journal of Open Source Education

Aknowledgments

The authors would like to thank DIIT-UNLaM for its support and grant C2-ING-109 (2023-2024). **Probablemente Edgardo quiera sumar algo acá de UTN?** # References

Hoffmann, A. F., Vigh, C., & Fernández-Liporace, M. (2021). Creatividad y enfoques de aprendizaje en estudiantes universitarios: Creativity and learning approaches in college students. *Psicogente*, *24*(46), 1–17. https://doi.org/10.17081/psico.24.46.4492

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Jason K. Moore. (2024). *Learn multibody dynamics* (0.2.dev0+f440663 ed.). https://moorepants.github.io/learn-multibody-dynamics/index.html

Kane, T. R., & Levinson, D. A. (1985). *Dynamics, theory and applications*. McGraw Hill. ISBN: 978-0-07-037846-9

Levinson, D. A., & Kane, T. R. (1990). AUTOLEV — a new approach to multibody dynamics. In W. Schiehlen (Ed.), *Multibody systems handbook* (pp. 81–102). Springer. https://doi.org/10.1007/978-3-642-50995-7_7

Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller, R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., … Scopatz, A. (2017). SymPy: Symbolic computing in python. *PeerJ Computer Science*, *3*, e103. https://doi.org/10.7717/peerj-cs.103

Moraros, J., Islam, A., Yu, S., Banow, R., & Schindelka, B. (2015). Flipping for success: Evaluating the effectiveness of a novel teaching approach in a graduate level setting. *BMC Medical Education*, *15*(1), 1–10. https://doi.org/10.1186/s12909-015-0317-2

Vallejo, W., Díaz-Uribe, C., & Fajardo, C. (2022). Google colab and virtual simulations: Practical e-learning tools to support the teaching of thermodynamics and to introduce coding to students. *ACS Omega*, *7*(8), 7421–7429. https://doi.org/10.1021/acsomega.2c00362

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2