

# Python-based Lagrange analytical mechanics course

Víctor A. Bettachini <sup>1,2</sup>, Mariano A. Real <sup>3,4</sup>, and Edgardo Palazzo <sup>5</sup>

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

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

## Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Submitted: 01 January 1970

Published: unpublished

## License

Authors of papers retain copyright<sup>13</sup> and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

## 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 of third-year Latin American students (mid-career), that by then have learned how to solve ordinary differential equations. Theory and examples exercises alongside the *Python* code that solves them are presented in *Jupyter notebooks*, that are 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 constraints: 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 as simple as possible in order 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

42 untenable for students and teaching staff working on the blackboard or paper. This course  
43 aims to avoid this pitfall by taking advantage of the relative simple syntax of modern  
44 programming languages to tackle mathematical problems. In this way it is possible to  
45 rapidly introduce life-like problems avoiding oversimplifications to the students.

46 The required modelling as well as algebraic and calculus operations to generate the  
47 Euler-Lagrange differential equations are performed using *physics.mechanics*, the symbolic  
48 dynamics sub-package of the *SymPy* library (Meurer et al., 2017). Its code was ported  
49 from the *PyDy* library, a replacement of *Autolev* (Levinson & Kane, 1990), a commercial  
50 software that instrumentalised the Kane's method (Kane & Levinson, 1985). As stated in  
51 the online textbook for the *Multibody Dynamics course at TU Delft*, a successor to the one  
52 *PyDy* was developed for, this method avoids accounting for non-conservative forces with  
53 Lagrange's multipliers, but it requires modelling forces in the system (Jason K. Moore,  
54 2024). Our choice was instead to make students model systems solely by their energy,  
55 a more traditional approach, in order to immerse them into a radically different way of  
56 solving mechanical problems in their first contact with the subject of analytical mechanics.  
57 We think that when facing problems requiring a more efficient method, they will be able  
58 to apply such other less abstract methods.

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

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

## 76 Overview, Content, and Structure

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

87 Each folder contains Jupyter notebooks with the required theory for the unit subject  
88 alongside the code that solves example exercises. The students only need to modify that  
89 code to solve the exercises proposed at the accompanying problem sets. These are provided  
90 in PDF format alongside their LaTeX source and figure files allowing their customisation.  
91 The number of exercises in each problem set, while still being illustrative of the variety of  
92 the unit subject applications, is kept small in order to make their solving mandatory on a

93 weekly basis. Those of units 8, 9 and 11 are exceptions, requiring two weeks each, as they  
94 deal with subjects that had shown to be somewhat more demanding to students.

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

## 105 Implementation

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

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

## 120 Conclusions

121 The mechanical engineering programme is relatively new at UNLaM, so the number of  
122 students per class is still low, around eight, thus still allowing this personalised tracking  
123 of student's progress. Larger audiences will provide a challenge, probably requiring to  
124 include new teaching assistants as well as introducing automatic grading, to somewhat  
125 stick to this methodology.

126 For the time being, feedback from students consistently indicates a high level of satisfaction  
127 with this course, especially with its code-driven aspect. Additionally, students express  
128 interest in the final examination as it provides an opportunity to apply both their presen-  
129 tation skills and the knowledge acquired throughout the course. In relation to the flipped  
130 classroom model, students acknowledge that it requires a grater effort, but a majority of  
131 them agree that it is a positive and beneficial implementation. This is in line with previous  
132 research on the flipped classroom model for advances mechanical engineering courses (?).

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

## 136 References

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

- 140 Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science &*  
141 *Engineering*, 9(3), 90–95. <https://doi.org/10.1109/MCSE.2007.55>
- 142 Jason K. Moore. (2024). *Learn multibody dynamics* (0.2.dev0+f440663 ed.). [https:](https://moorepants.github.io/learn-multibody-dynamics/index.html)  
143 [//moorepants.github.io/learn-multibody-dynamics/index.html](https://moorepants.github.io/learn-multibody-dynamics/index.html)
- 144 Kane, T. R., & Levinson, D. A. (1985). *Dynamics, theory and applications*. McGraw Hill.  
145 ISBN: 978-0-07-037846-9
- 146 Levinson, D. A., & Kane, T. R. (1990). AUTOLEV — a new approach to multibody  
147 dynamics. In W. Schiehlen (Ed.), *Multibody systems handbook* (pp. 81–102). Springer.  
148 [https://doi.org/10.1007/978-3-642-50995-7\\_7](https://doi.org/10.1007/978-3-642-50995-7_7)
- 149 Meurer, A., Smith, C. P., Paprocki, M., Čertík, O., Kirpichev, S. B., Rocklin, M., Kumar,  
150 A., Ivanov, S., Moore, J. K., Singh, S., Rathnayake, T., Vig, S., Granger, B. E., Muller,  
151 R. P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., ... Scopatz, A.  
152 (2017). SymPy: Symbolic computing in python. *PeerJ Computer Science*, 3, e103.  
153 <https://doi.org/10.7717/peerj-cs.103>
- 154 Moraros, J., Islam, A., Yu, S., Banow, R., & Schindelka, B. (2015). Flipping for success:  
155 Evaluating the effectiveness of a novel teaching approach in a graduate level setting.  
156 *BMC Medical Education*, 15(1), 1–10. <https://doi.org/10.1186/s12909-015-0317-2>
- 157 Vallejo, W., Díaz-Uribe, C., & Fajardo, C. (2022). Google colab and virtual simulations:  
158 Practical e-learning tools to support the teaching of thermodynamics and to introduce  
159 coding to students. *ACS Omega*, 7(8), 7421–7429. [https://doi.org/10.1021/acsomega.](https://doi.org/10.1021/acsomega.2c00362)  
160 [2c00362](https://doi.org/10.1021/acsomega.2c00362)
- 161 Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau,  
162 D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett,  
163 M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R.,  
164 Larson, E., ... SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for  
165 Scientific Computing in Python. *Nature Methods*, 17, 261–272. [https://doi.org/10.](https://doi.org/10.1038/s41592-019-0686-2)  
166 [1038/s41592-019-0686-2](https://doi.org/10.1038/s41592-019-0686-2)