Experiences on a Computational Analitical Mechanics course based on code and inverted classroom

Víctor A. Bettachini[1][0000−0001−7485−8884],
Mariano A. Real[1,2][0000−0003−3022−7516], and Edgardo Palazzo[3][NNNNNN]

[1] Universidad Nacional de La Matanza - UNLAM, Buenos Aires, Argentina
vbettachini@unlam.edu.ar
https://ingenieria.unlam.edu.ar/
[2] Instituto Nacional de Tecnología Industrial - INTI, Buenos Aires, Argentina
[3] Universidad Tecnológica Nacional - UTN, Buenos Aires, Argentina

Abstract. In this paper, we present different technichs applied to a course on Computational Analytical Mechanics for Mechanical Engineering students. The course is designed to teach students Analythical Mechanics and how to take advantage of programming languages as a tool to solve problems in physics. The course is produced using online free tools such as Python, Markdown and LaTeX by using Jpyther notebooks in Google Colab. The course is hosted on Github and Microsoft Teams, which provide an excellent way to manage and arrange the information given and also keep track of the work required to the students. Students are not required to have prior knowledge of programming languages, but they are expected to modify the code given by the professor to solve the syllabus problems. The course was initially taught completely online since started during the SARS-CoV-2 pandemic, but now it is given in the classroom. During the return to the classroom it was further improved by appling an inverted classroom technique, where students are given all the material for the class beforehand. They must read the theory and start working on problems to then use the synchronous time with the teachers to ask questions on the theory and finishing solving the problems. We find that reducing the ammount of homework but seting all problems as mandatory, greatly improved the students response. The course is finished by solving a final complex problem, where the students solve forces and torques on a mechanical robotic arm. They are required to defend it in a short presentation to teachers, improving their presentation and sythesis skills.

Keywords: Mechanics · Code · Jupyter · Python · Inverted Classroom

1 Resume

"Numerical Analysis" and "Programming Fundamentals" are mentioned as "Knowledge Descriptors" for a Mechanical Engineer in the "Proposal of standards for

the second generation for engineering degrees accreditation in the Argentine Republic" approved by the "Federal council of engineering deans" in 2018, and best known as "Libro Rojo de CONFEDI"[1]. Regrettably after theory and use of these tools are learnt by students they are not fully exploited in courses at later years.

In this work the experience had at the subject Mecánica General (General Mechanics) of the UNLaM's mechanical engineering degree third-year is described. Traditionally modeled systems are limited to those analytically solvable working on blackboard or paper. In this course the students solve their problem sets using Python language code, applying this century tools, such as library functions for symbolic and numerical analysis, plotting, etc.

All classes were conducted in full using Jupyter notebooks as a platform. In those notebooks code is interbowen with graphical information and text including clear mathematical notation with LaTeX symbols. Students can re-use this same code to solve course's problem sets as well as in future courses and their professional life.

The notebooks mentioned above run on free software. Students operate on them with free to use web platforms that allow concurrent commenting and editing among them and/or their professor.

The pandemic pushed us all to teach through a computer. After an initial adaptation period the students recognized the virtues of this methodology. Even assessments were more enriching than in a conventional course as it reached the complexity of simulating industrial-like mechanical systems.

2 Introduction

The last three quarters, the SARS-CoV-2 pandemic forced the courses for Mechanical Engineering students at the National University of La Matanza (UNLaM) to be taught remotely. The fact that students were behind a computer during class was used to impose a pedagogical methodology that omitted traditional chalkboard, paper, and non-interactive computer presentations in favor of presenting theoretical concepts and practicing their use in activities on an interactive computer platform based on code written in the Python language.

The complexity of the code increases as new aspects affecting a mechanical system are contemplated from class to class. In a chalkboard and paper-based course, similar calculations are repeated in each new activity, while in a code-based course, it presents the advantage of its reuse. By making modifications to the code tested in previous classes with simple mechanical systems, the analysis capacity is expanded without the loss of time that would be required for writing from scratch the long set of calculations that an analysis in the Euler-Lagrange scheme of a more complex mechanical system requires.

In the post-pandemic return to the classroom the proposed methodology was further improved, now using an inverted classroom aproach incluir CITA. The students are presented to online theory and examples to be read and physics problems to be started at home that must be finished during their time in the

classroom with the help of teachers. In this way the theory and initial stages of the problem-solving are asynchronous, while they take advantage of the classroom time to ask issues regarding the problems and theory.

3 About Computational Analytical Mechanics course in UNLaM

In the curriculum of the Mechanical Engineering degree program at the Department of Engineering and Technological Research (DIIT) of UNLaM, this subject is the link between the first ones specific to the specialty with the basic ones in which tools of algebra, mathematical analysis, numerical calculus, and Newtonian mechanics are taught. The scheme of immediate correlations to the General Mechanics subject, which shows Figure 1, makes it clear that this subject must have among its objectives to show the student how these tools have an application in their area of interest.

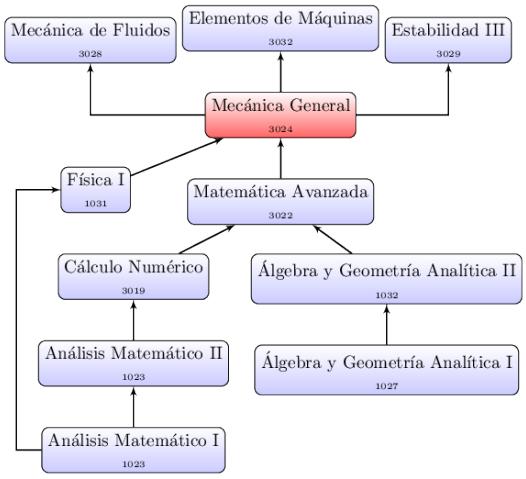


Fig. 1. Preceded by subjects of algebra, analysis, and physics, General Mechanics is the first in which such knowledge is applied to mechanical engineering.

The subject trains students in the ability to model the physics of simple mechanical systems. Modeling is understood as a series of procedures with which a simplified scheme of physics is constructed based on a semi-quantitative evaluation of the forces and fields that act on the system as well as the constraints that limit its degrees of freedom. With such information, some of these are prioritized and others are discarded to arrive at the aforementioned scheme. Having such a model allows:

- to choose generalized coordinates to describe the relevant degrees of freedom,
- to write mathematical relationships between them that account for constraints,
- to describe generalized forces that are not the effect of fields (gravitational, electromagnetic, etc.),
- and to describe the potential and kinetic energy of the system as a whole.

After performing the above, the Euler-Lagrange formalism is demonstrated and put into practice in the course to obtain a set of differential equations that describe the dynamics of the system and/or the mechanical stresses that each of its components must withstand at each instant of time.

From what has been exposed in the preceding paragraphs, it is evident that the subject matter of the course object of this work is circumscribed to the conventional one of rational mechanics courses as detailed in its canonical reference literature [2]. It is not in the subject matter but in its didactic methodology where innovation was made.

3.1 The blackboard, the only didactic tool

Traditionally, the systems worked on in rational mechanics courses are relatively simple to limit the time and/or difficulty of mathematical analysis and/or algebra calculations required by the steps commented in the previous paragraph. But this extreme simplification leads to a noticeable jump in the complexity required to model mechanical devices, fluid dynamics, or rigid structures, the respective subjects of the courses subsequent to General Mechanics (see Figure 1).

The limitation to complexity is imposed by what the teacher can calculate on the blackboard during a class. As they are erased successively, they not only prevent the student from using a reference of something that is no longer in sight, but also impose on them to devote a good part of their attention to not making mistakes when transcribing what is written there into their notebook. This paper support, in turn, limits the length and complexity of the problems that can be proposed to students to exercise what they have learned. What summarizes the preceding paragraph is nothing more than the procedure in the teaching of science or engineering courses at the university level that has been almost unchanged since the 19th century to the present day.

3.2 Computer-based didactic tools

The steps before and after obtaining a system of differential equations that describe the dynamics and stresses for a complex mechanical model can be performed with computer-based tools, but they are different for each case. To solve and analyze the result of the equations, the tools learned in the Numerical Calculus subject, previously taken before General Mechanics, and ubiquitous graphing tools to visualize the temporal evolution of different magnitudes are applied. Although it is true that numerical calculus is occasionally used in exercises [3, 4], it

is rarely used by the teacher during class. This is a missed opportunity to better exemplify and deepen the analysis of the behavior of the modeled systems.

But if the application of numerical calculus during classes is rare, the use of computer algebra systems (CAS) is even rarer. These systems allow automating all the mathematical procedures required for modeling: from defining degrees of freedom, coordinate systems, fields, and external forces to the model to building the system of differential equations for the dynamics. Using them to solve linear algebra and mathematical analysis calculations allows to remove the emphasis on these and focus the attention of the teacher and students on the subject matter of the course.

However, the daily reality of our classrooms is that these calculations continue to be done manually on the blackboard or on paper during classes, ignoring the use of computers. Insisting on this procedure at the university level would be analogous to depriving students of the use of pocket calculators to perform arithmetic procedures learned at the primary level. It is an anachronism in the third decade of the 21st century.

3.3 Code recycling

The above can be misinterpreted as a mere call to use computers as an analog of the pocket calculator, replacing the results of the calculations required to solve exercises on paper. This would be an underutilization of such a resource in the class, repeating the pattern followed by most computer users who ignore a fundamental aspect of computing in their daily use.

The digital computer was invented during World War II to perform numerical calculations that were manually defined in each execution, operating as a faster calculator with the ability to automate some of the numerical manipulation processes. But since the middle of the last century, it has acquired the ability to operate under a series of instructions stored in its memory on how to process both numerical and other information. Such instructions are called programs, and they are written in a code that respects the syntax of a certain language.

Modern high-level languages allow writing a single code incorporating all the procedures required for the resolution and analysis of a rational mechanics problem. This includes from the assumptions made to simplify the physics of the problem to the analysis with graphs of its dynamics and mechanical stresses, passing through all the intermediate algebraic and numerical calculations.

Starting from the objective that students exploit this tool, the course had as its methodology to avoid paper work and instead develop the ability to write in a single code the set of operations required to solve exercises. In class, the teacher explained synchronously examples of codes that perform all the procedures required to model a mechanical system. In each class, a guide to solvable exercises is provided by making small modifications to the code.

4 Tools used on this curse

4.1 Python, Sympy, Numpy, Scipy and Matplotlib

The Python programming language is by default devoid of scientific and engineering calculation capabilities. This is a design decision to make such functionalities be added by specialized libraries. The effect of this decision is that the development of these libraries is carried out by users who apply them in various fields of scientific-technological development rather than by computer professionals.

The functions of symbolic calculation are provided by the Sympy library. Its Mechanics module is particularly useful for generating equations for the dynamics of rigid body systems with multiple degrees of freedom and in various reference systems [5].

Differential equation systems are solved by numerical methods supported by functions for the manipulation of algebraic elements of the Numpy library [6] and the numerical optimization and integration algorithms of Scipy [7].

Engineering analysis of numerical results is usually interpreted with graphical representations. This capability is provided by the functions of the Matplotilb library [8].

4.2 Jupyter Notebooks

The environment used in the course to run code is the web-based application of the Jupyter Project called JupyterLab, whose document format is the Jupyter notebook [9]. This alternates independent sections called cells. Input cells are code (in various languages, Python is just one of the possibilities) or annotations, as shown in Figure 2. This latter variant of cells is written in the Markdown markup language [10] that allows you to embed: text and/or mathematical expressions in LaTeXformat interspersed, and multimedia content: web links, images, video and/or sound players.

4.3 Running Jupyter Online

Students are not required to install any software on their computer to take the course. They only need to use a standard web browser to use one of the services that run Jupyter notebooks online. This can be an installation of JupyterHub from the Jupyter Project on university-owned servers or commercial clouds, or alternatively one of the services that offer even free alternatives such as CoCalc, IBM Watson or Google Colaboratory. Of these, the latter has been used in the latest editions of the course after Microsoft closed its free Azure Notebooks service.

The Google Colaboratory service, colloquially only Colab, presents as a convenience the ability to run notebooks hosted in a Git repository managed by the online service GitHub. A modification in the URL is enough to point a notebook to a web browser to run it in Colab [11]. Working with Jupyter notebooks in

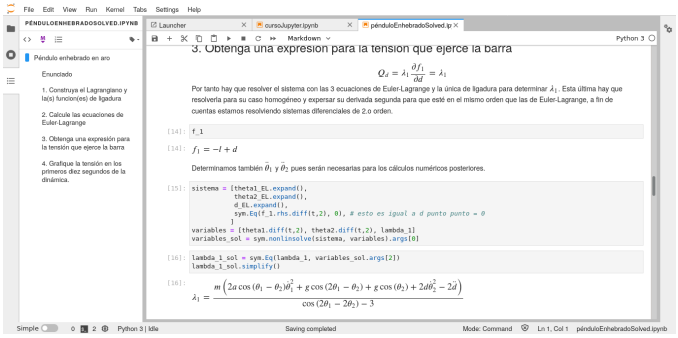


Fig. 2. A Jupyter notebook is a set of cells. These are in Markdown or executable code format. The former can contain text, mathematical expressions or multimedia content. The latter are lines of code in various programming languages. Interspersing titles in Markdown cells generates the index (on the left) that facilitates location within the document.

this service can be done concurrently by several students and/or teachers. Comments can also be included, whose update is reported by email, which is useful for correcting exercises since the location of errors in the code can be indicated, as shown in Figure 3.

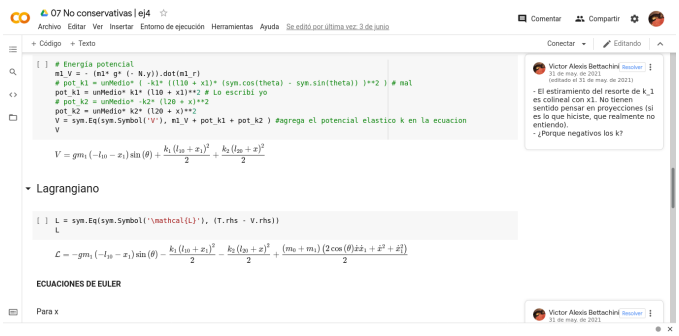


Fig. 3. The Google Colaboratoy website allows Jupyter notebooks to be edited and run concurrently between students and teachers, as well as including comments. The latter feature is useful for corrections.

4.4 Git Repository

The aforementioned repository on GitHub is organized into separate folders for each class of the course, as shown in Figure 4. Each of these hosts the corresponding theoretical material and exercises in the format of Jupyter notebooks,

as well as exercise guides and occasional notes in the portable document format known by its acronym in English PDF. This arrangement facilitates both the teacher and the students an overview of the material of each topic as well as verifying any updates to it. In this way, the course material is publicly accessible, making it available for use by interested parties [12] as long as they cite its origin and do not use it commercially as indicated by its Creative Commons CC-BY-NC-SA license under which it is published [13].

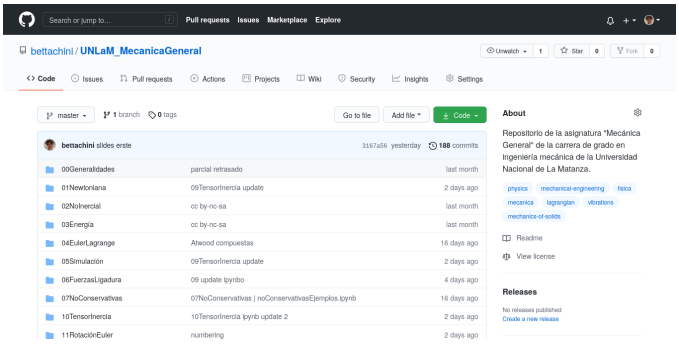


Fig. 4. The students find the material organized in separate directories per class.

4.5 Learning Management System

At UNLaM, the Microsoft Teams business communications platform is used to replace classroom interaction with students with videoconferencing. After each class, the videos are saved to Microsoft OneDrive online storage. Links to these and to the class materials hosted in the Git repository are compartmentalized into what the system calls channels, respecting the same numbering and naming as in the Git repository. Figure 5 shows the contents that head the ones displayed for the tenth class.

Each channel includes links to:

- practical exercise guide
- any occasional notes in PDF
- both ways to view Jupyter notebooks, interactive in Colab or static in nbviewer
- invitation to the videoconference or its video once it is over

Microsoft Teams also provides the rudiments of a Learning Management System (LMS) by allowing tasks to be assigned to students with acceptance deadlines by the system. Students can upload a link to their Colab notebook or the same in .ipynb format if modification is not allowed after a certain date for evaluation purposes.

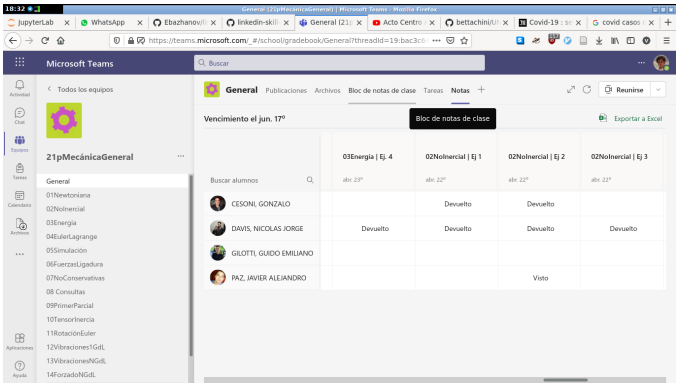


Fig. 5. Separate channels per class present links to their material.

5 Cruse chronology

Of the 16 weekly online meetings throughout the semester, 13 of them present new topics. Some of these topics were selected to illustrate the progression of the course.

Class 1. Review of Newtonian mechanics, analysis, and algebra required to obtain the ideal pendulum dynamics revisiting the approximations and calculations seen in Physics 1. This theory material is distributed in this and subsequent classes in a Jupyter notebook. Students are encouraged to review the LaTeX notation with which the teacher writes mathematical formulas such as those shown in Figure 6.

Considero que el potencial V es nulo en el origen de coordenadas, es decir que donde se encuentra su mínimo $\varphi = 0, V(\varphi = 0) = -mg\ell$ y por tanto

$$V(\varphi) = mg\,(-\ell\cos\varphi) = -mg\ell\cos\varphi,$$

Como vemos la aproximación funciona bastante bien. Conformes con ella calculamos la fuerza

$$\vec{F} = -\vec{\nabla}V = -\left(\frac{\partial}{\partial r}, \frac{1}{r}\frac{\partial}{\partial\varphi}, \frac{\partial}{\partial z}\right)V(\varphi)$$

Pero solo nos interesa expresar la 2.a ley de Newton para lo que pasa en $\hat{\varphi}$

$$m\ddot{\vec{r}}\cdot\hat{\varphi} = -\frac{1}{r}\frac{\partial}{\partial\varphi}V(\varphi)$$

En el lado izquierdo de la expresión de la aceleración en cilíndricas $\ddot{\vec{r}} = (\ddot{r} - r\dot{\varphi}^2)\hat{r} + (\dot{r}\dot{\varphi}^2 + r\ddot{\varphi})\hat{\varphi} + \ddot{z}\hat{z}$, nos quedamos solo con la componente en $\hat{\phi}$,

$$\ddot{\vec{r}}\cdot\hat{\varphi} = \dot{r}\dot{\varphi}^2 + r\ddot{\varphi}$$

y como el hilo del péndulo es rígido e inextensible $r \equiv \ell$ solo queda de esto

$$\ddot{\vec{r}}\cdot\hat{\varphi} = \ell\ddot{\varphi}$$

En el lado derecho la derivada del potencial respecto a φ es

$$\frac{\partial}{\partial\varphi}V(\varphi) = mg\ell\sin(\varphi)$$

Fig. 6. The theory is presented in Jupyter notebooks. All mathematical formulas are expressed in standardized LaTeXnotation that students can edit or copy for their own purposes.

In addition to the reiteration of what has already been seen in previous courses, in this first class, we already advance in the use of code to analyze results. Figure **??** shows instructions for the Matplotlib library to graph the solution for the ideal pendulum dynamics.

Class 3. Starting from this class, the Sympy library is applied in class for automatic symbolic calculation. Figure 7 shows how to calculate the kinetic energy of a system with two generalized coordinates differentiated in its reference system.

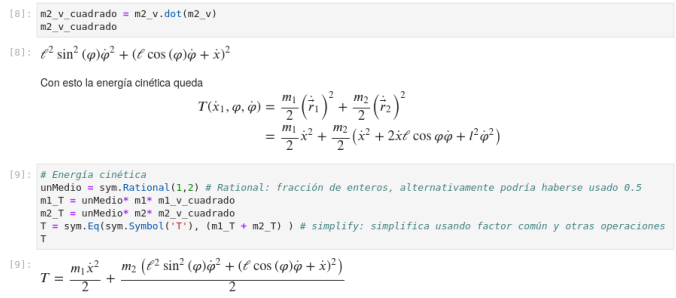


Fig. 7. First symbolic calculations using the SymPy library.

Class 4. The Euler-Lagrange equations allow students to obtain the equations that describe the dynamics of a system. Figure 8 shows how functions of the Sympy library facilitate obtaining such equations for a system with two degrees of freedom.

Fig. 8. Application of SymPy functions to generate the Euler-Lagrange differential equations that describe the dynamics of a system.

So far, code has been used to perform the same steps that are solved on a blackboard or paper in a conventional rational mechanics course to arrive at differential equations that are only solved for trivial cases. In contrast, using Sympy quickly solves complex systems for accelerations as a function of generalized coordinates and velocities as shown in Figure 9. Performing such a task manually would require a non-negligible amount of time and effort even for this system with only two degrees of freedom.

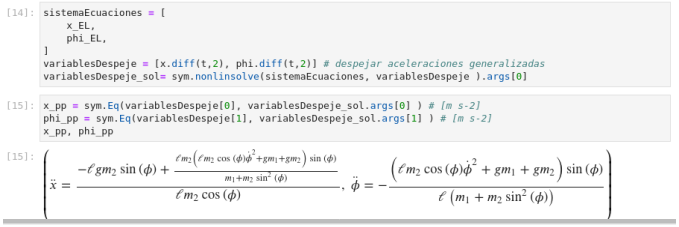


Fig. 9. The resolution of systems of differential equations of certain complexity is avoided in conventional courses. In this course, it only takes a couple of lines of code with functions of the Sympy library.

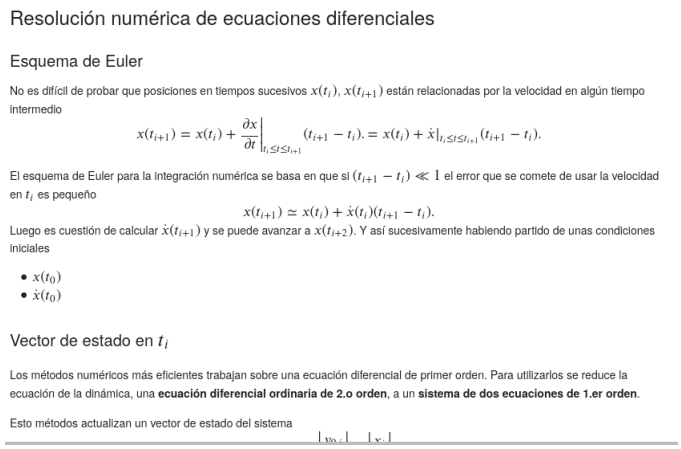
Class 5. Students passed a numerical calculus course to enroll in this course where such knowledge will be used. In class, the fundamentals of numerical resolution methods for differential equations are reviewed and how they would be implemented in a state vector notation suitable for efficient processing. Such a review is presented to students with the same methodology as for other topics, in Jupyter notebooks that students can edit, as shown in Figure 10.

Immediately after the review of fundamentals, the functions of the scientific calculation library Scipy are shown in action to efficiently obtain solutions for the dynamics of a two-degree-of-freedom system as illustrated in Figure 11.

The generalized positions and velocities obtained numerically in the range of times of interest are graphically represented. Figure **??** shows such a representation that serves to discuss with the students whether the behavior of the system is consistent with what can be predicted from a qualitative analysis of this simple system. Confirming that the symbolic and numerical calculation tools used obtain correct results gives confidence in them in view of applying them to more complex systems.

5.1 Class 7.

Non-conservative forces are incorporated into the codes, which ultimately are the majority of those that can affect an industrial mechanical device. As a first example, the analogy of pendulum oscillations is extended to a damped system shown in Figure 12 to analyze non-conservative forces. In this system, a linear damper acts with velocity. This non-conservative force could not be analyzed with the code of previous classes.



Resolución numérica de ecuaciones diferenciales

Esquema de Euler

No es difícil de probar que posiciones en tiempos sucesivos $x(t_i)$, $x(t_{i+1})$ están relacionadas por la velocidad en algún tiempo intermedio

$$x(t_{i+1}) = x(t_i) + \left.\frac{\partial x}{\partial t}\right|_{t_i \le t \le t_{i+1}} (t_{i+1} - t_i). = x(t_i) + \dot{x}|_{t_i \le t \le t_{i+1}} (t_{i+1} - t_i).$$

El esquema de Euler para la integración numérica se basa en que si $(t_{i+1} - t_i) \ll 1$ el error que se comete de usar la velocidad en t_i es pequeño

$$x(t_{i+1}) \simeq x(t_i) + \dot{x}(t_i)(t_{i+1} - t_i).$$

Luego es cuestión de calcular $\dot{x}(t_{i+1})$ y se puede avanzar a $x(t_{i+2})$. Y así sucesivamente hablendo partido de unas condiciones iniciales

- $x(t_0)$
- $\dot{x}(t_0)$

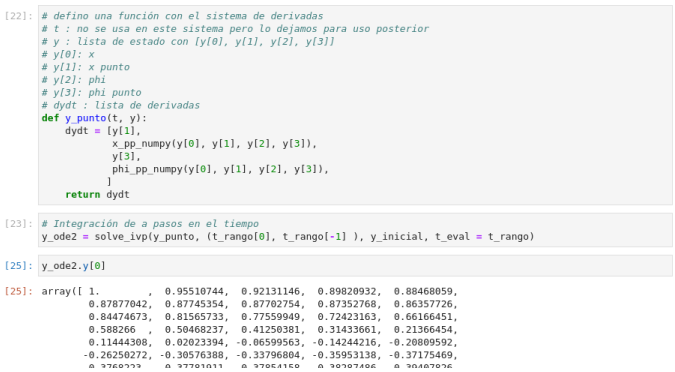
Vector de estado en t_i

Los métodos numéricos más eficientes trabajan sobre una ecuación diferencial de primer orden. Para utilizarlos se reduce la ecuación de la dinámica, una **ecuación diferencial ordinaria de 2.o orden**, a un **sistema de dos ecuaciones de 1.er orden**.

Esto métodos actualizan un vector de estado del sistema

$$\begin{bmatrix} v_{0,i} \\ x_i \end{bmatrix} \quad \begin{bmatrix} \\ x_i \end{bmatrix}$$

Fig. 10. Prior to proceeding with numerical resolution of differential equations, a review of its fundamentals is presented in Jupyter notebooks.



```
[22]: # defino una función con el sistema de derivadas
      # t : no se usa en este sistema pero lo dejamos para uso posterior
      # y : lista de estado con [y[0], y[1], y[2], y[3]]
      # y[0]: x
      # y[1]: x punto
      # y[2]: phi
      # y[3]: phi punto
      # dydt : lista de derivadas
      def y_punto(t, y):
          dydt = [y[1],
                  x_pp_numpy(y[0], y[1], y[2], y[3]),
                  y[3],
                  phi_pp_numpy(y[0], y[1], y[2], y[3]),
                 ]
          return dydt

[23]: # Integración de a pasos en el tiempo
      y_ode2 = solve_ivp(y_punto, (t_rango[0], t_rango[-1] ), y_inicial, t_eval = t_rango)

[25]: y_ode2.y[0]

[25]: array([ 1.        ,  0.95510744,  0.92131146,  0.89820932,  0.88468059,
              0.87877042,  0.87745354,  0.87702754,  0.87352768,  0.86357726,
              0.84474673,  0.81565733,  0.77559949,  0.72423163,  0.66166451,
              0.588266  ,  0.50468237,  0.41250381,  0.31433661,  0.21366454,
              0.11444308,  0.02023394, -0.06599563, -0.14244216, -0.20809592,
             -0.26250272, -0.30576388, -0.33796804, -0.35953138, -0.37175469,
```

Fig. 11. The system of equations for the dynamics of a two-degree-of-freedom system is numerically solved with functions of the SciPy library.

The figure 13 shows the graph that allows analyzing the dynamics calculated with the same procedure and code that has been used in the previous classes.

Next classes. The usual syllabus of a course in rational mechanics is completed by focusing on extensive systems analyzed within the framework of the rigid body and the analysis of forced oscillations in systems with multiple degrees of freedom. Towards the end of the course, students have already developed the ability to autonomously analyze "realistic" systems in terms of being more similar to mechanical devices existing in the industry. To capture this, they are proposed to calculate the torques that the motors of a highly simplified industrial robotic arm must perform so that it performs a sequence of movements. Examples of the result of students' work in response to this proposal are shown in figure 14.

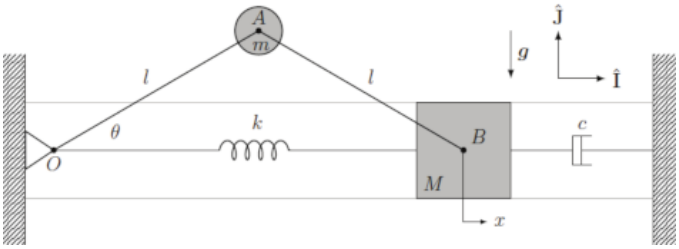


Fig. 12. The range of analysable factors expanpands gradually. In the system shown a damper provides a linear resistance to velocity. This non-conservative force could not be analysed with the previous classes code.

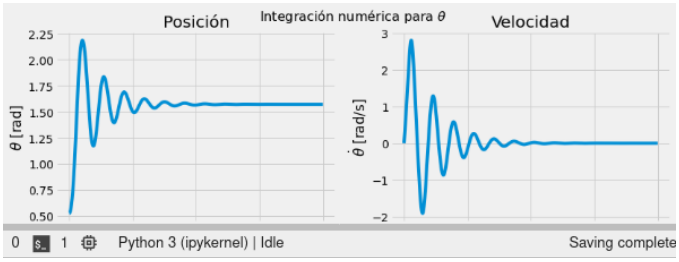


Fig. 13. The range of analyzable factors is gradually extended. In this system, a linear damper with velocity acts. This non-conservative force could not be analyzed with the code of previous classes.

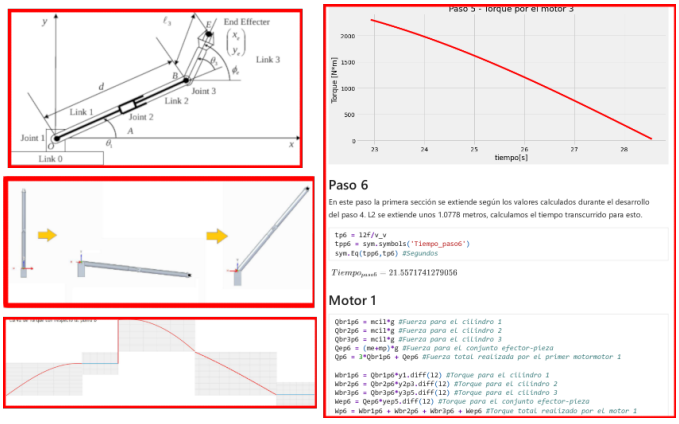


Fig. 14. For an industrial mechanical arm to perform even a simple movement, its motors must apply a sequence of torques. Students calculate them in a work that reflects their mastery of analytical and computer tools whose use was learned in the course.

6 Results and objectives

Coloque aquí los resultados alcanzados y los objetivos en curso o futuros del proyecto.

7 Discusssion

Desarrollo del texto.

8 End discussion

La exposición de teoría y la ejercitación práctica tomaron distintas ventajas de la metodología de este curso.

Clases de teoría:

- La exposición en pizarrón o en una presentación en que los alumnos están concentrados en transcribir lo allí escrito se reemplazó por código que pueden reutilizar para resolver sus ejercicios.
- En las clases en línea cada palabra del docente durante la clase queda registrada en video liberando al alumno de la toma de notas.
- El docente puede cambiar el código durante la clase para corregir un error o graficar otro aspecto de la temática.

Ejercicios de práctica:

- En papel una variación sobre un ejercicio resuelto anteriormente obliga a reiterar tediosos cálculos similares. Con código basta con modificar ligeramente el mismo para atender al nuevo caso.
- En forma remota varios alumnos pueden trabajar concurrentemente en la resolución en un mismo ejercicio.
- Los alumnos pueden alertar al docente a toda hora vía el LMS de un inconveniente que enfrenten en la resolución de un ejercicio. El docente puede dedicarle tiempo y detenimiento en el momento que encuentre propicio a diferencia del acotado tiempo de consultas del que se dispone en el aula.
- Los docentes pueden comentar y corregir el mismo código sobre el que está trabajando el alumno inclusive en tiempo real.

9 Conclussions

El uso de herramientas informáticas durante la clase para la resolución de ejercicios de mecánica racional aliviaría, en especial al alumnado, de tediosos cálculos en pizarrón o papel que distraen del temario propio de la asignatura. El no hacerlo se justificaba por cuestiones de disponibilidad de equipamiento, en particular en las universidades del tercer mundo. La pandemia de SARS-CoV-2 forzó a realizar cursos a través de internet que demostraron donde tales recursos estaban disponibles: en los hogares y lugares de trabajo de alumnos y docentes.

El curso objeto de este trabajo instrumentó una metodología para presentar los conceptos de teoría y su puesta en práctica en ejercicios a través de la escritura de código que capitaliza conocimientos de los alumnos adquiridos en materias previamente cursadas: "Fundamentos de programación" y "Cálculo numérico". Los términos entre paréntesis son "descriptores de conocimiento" para un Ingeniero Mecánico en el "Libro Rojo de CONFEDI" [1]. No hacer uso de los mismos en materias posteriores a aquellas en las que se aprendieron es un desperdicio del esfuerzo de sus docentes y alumnos. Docentes en universidades reconocidas a nivel mundial también lo han entendido así al iniciar en esta última década cursos de ingeniería con idénticas metodología y herramientas que las presentadas en este trabajo [14] [4,17,18].

Es el convencimiento de los autores que la metodología de este curso presenta una mayor utilidad al estudiante en vistas a próximas asignaturas y su vida profesional respecto a la que pueden brindar clases presenciales. Las ventajas citadas en este trabajo de un curso basado en código por sobre una cursada presencial les lleva a recomendar la continuidad de esta metodología independientemente de si las condiciones sanitarias vuelven a permitir cursos con la metodología convencional basados en tecnología del siglo XIX.

Acknowledgments. The authors would like to thank the coordination of the Mechanical Engineering career at DIIT-UNLaM for accompanying the development and implementation of this course.

Disclosure of Interests. The authors declare no competing interests.

Bibliography

[1] Consejo Federal de Decanos de Ingeniería - CONFEDI. *Propuesta de estándares de segunda generación para la acreditación de carreras de ingeniería en la República Argentina*. Universidad FASTA Ediciones, 2018.

[2] L.D. Landau and E.M. Lifshitz. *Mecánica*. 1st ed. Ciudad de México, México: Reverté, 1994.

[3] A Mirasso, S. Raichman, and E. Totter. "Articulación de estratégias y recursos para el aprendizaje de métodos numéricos en ingeniería". In: *Mecánica Computacional Vol XXXIII*. XXI Congreso sobre Métodos Numéricos y sus Aplicaciones. Bariloche, Argentina, 2014, pp. 2099–2109.

[4] Marta Caligaris et al. "Desarrollo de habilidades matemáticas durante la resolución numérica de problemas de valor inicial usando recursos tecnológicos". In: 14 (Feb. 2018), pp. 30–40.

[5] Aaron Meurer et al. "SymPy: symbolic computing in Python". In: *PeerJ Computer Science* 3 (Jan. 2017), e103. URL: https://doi.org/10.7717/peerj-cs.103.

[6] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. URL: https://doi.org/10.1038/s41586-020-2649-2.

[7] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272.

[8] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95.

[9] Thomas Kluyver et al. "Jupyter Notebooks – a publishing format for reproducible computational workflows". In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. Ed. by F. Loizides and B. Schmidt. IOS Press. 2016, pp. 87–90.

[10] J. Gruber. *Daring Fireball*. Sept. 28, 2023. URL: https://daringfireball.net/projects/markdown/.

[11] Google LLC. *Using Google Colab with GitHub*. URL: https://colab.research.google.com/github/googlecolab/colabtools/blob/master/notebooks/colab-github-demo.ipynb.

[12] V. A. Bettachini. *Repositorio de la asignatura Mecánica General*. URL: https://github.com/bettachini/UNLaM_MecanicaGeneral/.

[13] Creative Commons. *Atribución-NoComercial-CompartirIgual 4.0 Internacional*. URL: https://creativecommons.org/licenses/by-nc-sa/4.0/deed.es.

[14] Lorena A. Barba. *Teaching and Learning with Jupyter*. Medium. Jan. 7, 2019. URL: https://blog.jupyter.org/teaching-and-learning-with-jupyter-c1d965f7b93a.