

# A rational mechanics course where everything is made with Python code

Bettachini, Víctor A.<sup>1</sup> Real, Mariano A.<sup>1</sup> Palazzo, Edgardo<sup>2</sup>



<sup>1</sup> DIIT, Universidad Nacional de La Matanza, <sup>2</sup> FRA, Universidad Tecnológica Nacional, ARGENTINA

New Media Pedagogy 23: research trends,  
methodological challenges and succesful  
implementations

21-24 november 2023



JAGIELLONIAN UNIVERSITY  
IN KRAKÓW



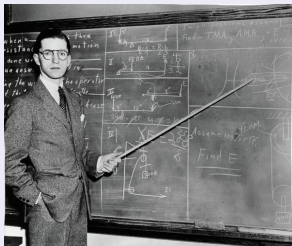
INNOEDUCA  
GRUPO DE INVESTIGACIÓN

## Value students and professors time

Licklider (1957): 85 % of “thinking” are actually mundane task (calculations, drawings, etc.)

# Value students and professors time

Licklider (1957): 85 % of “thinking” are actually mundane task (calculations, drawings, etc.)

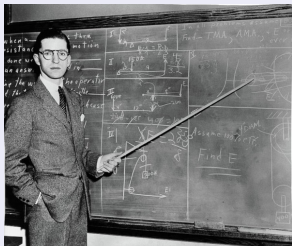


Most current classrooms: an exercise on transcription

- Professor: prepares lessons  $\xrightarrow{\text{by heart}}$  blackboard/slides

# Value students and professors time

Licklider (1957): 85 % of “thinking” are actually mundane task (calculations, drawings, etc.)

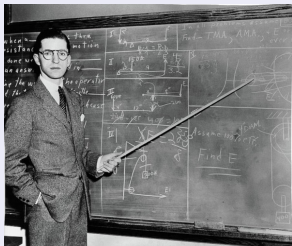


Most current classrooms: an exercise on transcription

- Professor: prepares lessons  $\xrightarrow{\text{by heart}}$  blackboard/slides
- Student: blackboard/slides  $\xrightarrow{\text{transcribes}}$  own notebooks

# Value students and professors time

Licklider (1957): 85 % of “thinking” are actually mundane task (calculations, drawings, etc.)

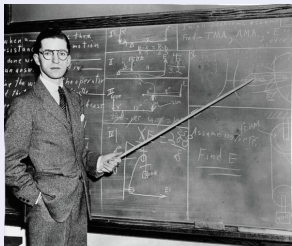


Most current classrooms: an exercise on transcription

- Professor: prepares lessons  $\xrightarrow{\text{by heart}}$  blackboard/slides
- Student: blackboard/slides  $\xrightarrow{\text{transcribes}}$  own notebooks
- Practice **reiterate** diagrams, calculations, etc.

# Value students and professors time

Licklider (1957): 85 % of “thinking” are actually mundane task (calculations, drawings, etc.)

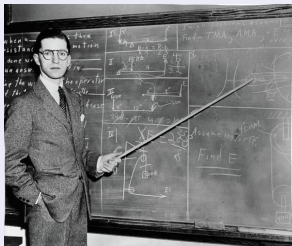


Most current classrooms: an exercise on transcription

- Professor: prepares lessons  $\xrightarrow{\text{by heart}}$  blackboard/slides
- Student: blackboard/slides  $\xrightarrow{\text{transcribes}}$  own notebooks
- Practice **reiterate** diagrams, calculations, etc.
- Boredom  $\rightarrow$  loss of concentration on the subject

# Value students and professors time

Licklider (1957): 85 % of “thinking” are actually mundane task (calculations, drawings, etc.)

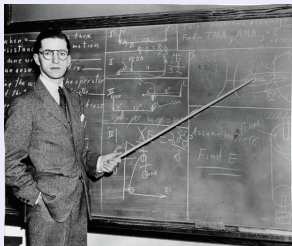


Most current classrooms: an exercise on transcription

- Professor: prepares lessons  $\xrightarrow{\text{by heart}}$  blackboard/slides
- Student: blackboard/slides  $\xrightarrow{\text{transcribes}}$  own notebooks
- Practice **reiterate** diagrams, calculations, etc.
- Boredom  $\rightarrow$  loss of concentration on the subject

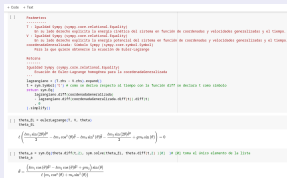
# Value students and professors time

Licklider (1957): 85 % of “thinking” are actually mundane task (calculations, drawings, etc.)



Most current classrooms: an exercise on transcription

- Professor: prepares lessons  $\xrightarrow{\text{by heart}}$  blackboard/slides
- Student: blackboard/slides  $\xrightarrow{\text{transcribes}}$  own notebooks
- Practice **reiterate** diagrams, calculations, etc.
- Boredom  $\rightarrow$  loss of concentration on the subject



Digital course material with embedded code

- Professor: ideas  $\xrightarrow{\text{updates}}$  code/theory in repository



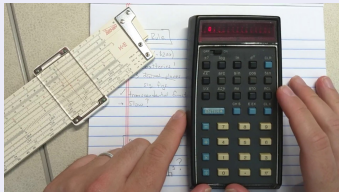




Engineering students can take advantage of code at every single lecture

# Engineering students can take advantage of code at every single lecture

- Currently they use a pocket calculator **after they learnt** learning arithmetics at school



# Engineering students can take advantage of code at every single lecture

- Currently they use a pocket calculator **after they learnt** learning arithmetics at school
- Now they'll employ computational algebra **after they learnt** algebra and calculus

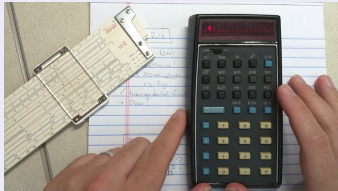


```
[14]: sistemaEcuaciones = [  
      x_EL,  
      phi_EL,  
    ]  
variablesDespeje = [x.diff(t,2), phi.diff(t,2)] # despejar aceleraciones generalizadas  
variablesDespeje_sol= sym.nonlinsolve(sistemaEcuaciones, variablesDespeje ).args[0]  
  
[15]: x_pp = sym.Eq(variablesDespeje[0], variablesDespeje_sol.args[0]) # [m s-2]  
      phi_pp = sym.Eq(variablesDespeje[1], variablesDespeje_sol.args[1]) # [m s-2]  
      x_pp, phi_pp
```

$$[15]: \left( \begin{array}{l} \ddot{x} = \frac{-\ell g m_2 \sin(\phi) + \frac{\ell m_2 (\ell m_2 \cos(\phi) \dot{\phi}^2 + g m_1 + g m_2) \sin(\phi)}{m_1 + m_2 \sin^2(\phi)}}{\ell m_2 \cos(\phi)}, \ddot{\phi} = -\frac{(\ell m_2 \cos(\phi) \dot{\phi}^2 + g m_1 + g m_2) \sin(\phi)}{\ell (m_1 + m_2 \sin^2(\phi))} \end{array} \right)$$

# Engineering students can take advantage of code at every single lecture

- Currently they use a pocket calculator **after they learnt** learning arithmetics at school
- Now they'll employ computational algebra **after they learnt** algebra and calculus
  - ▶ Focus on new skills, not in automatable calculations



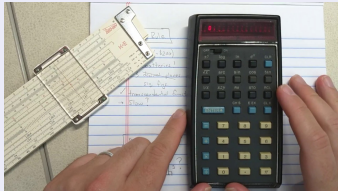
```
[14]: sistemaEcuaciones = [  
    x_EL,  
    phi_EL,  
]  
variablesDespeje = [x.diff(t,2), phi.diff(t,2)] # despejar aceleraciones generalizadas  
variablesDespeje_sol= sym.nonlinsolve(sistemaEcuaciones, variablesDespeje ).args[0]
```

```
[15]: x_pp = sym.Eq(variablesDespeje[0], variablesDespeje_sol.args[0]) # [m s-2]  
phi_pp = sym.Eq(variablesDespeje[1], variablesDespeje_sol.args[1]) # [m s-2]  
x_pp, phi_pp
```

$$\ddot{x} = \frac{-\ell g m_2 \sin(\phi) + \frac{\ell m_2 (\ell m_2 \cos(\phi) \dot{\phi}^2 + g m_1 + g m_2) \sin(\phi)}{m_1 + m_2 \sin^2(\phi)}}{\ell m_2 \cos(\phi)}, \quad \ddot{\phi} = -\frac{(\ell m_2 \cos(\phi) \dot{\phi}^2 + g m_1 + g m_2) \sin(\phi)}{\ell (m_1 + m_2 \sin^2(\phi))}$$

# Engineering students can take advantage of code at every single lecture

- Currently they use a pocket calculator **after they learnt** learning arithmetics at school
- Now they'll employ computational algebra **after they learnt** algebra and calculus
  - ▶ Focus on new skills, not in automatable calculations
  - ▶ Employing numerical calculus they solve what is impossible in a blackboard/paper

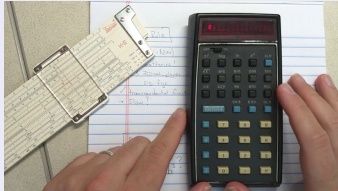


```
[14]: sistemaEcuaciones = [  
    x_EL,  
    phi_EL,  
]  
variablesDespeje = [x.diff(t,2), phi.diff(t,2)] # despejar aceleraciones generalizadas  
variablesDespeje_sol= sym.nonlinsolve(sistemaEcuaciones, variablesDespeje ).args[0]  
  
[15]: x_pp = sym.Eq(variablesDespeje[0], variablesDespeje_sol.args[0]) # [m s-2]  
phi_pp = sym.Eq(variablesDespeje[1], variablesDespeje_sol.args[1]) # [m s-2]  
x_pp, phi_pp
```

$$\ddot{x} = \frac{-\ell g m_2 \sin(\phi) + \frac{\ell m_2 (\ell m_2 \cos(\phi) \dot{\phi}^2 + g m_1 + g m_2) \sin(\phi)}{m_1 + m_2 \sin^2(\phi)}}{\ell m_2 \cos(\phi)}, \quad \ddot{\phi} = -\frac{(\ell m_2 \cos(\phi) \dot{\phi}^2 + g m_1 + g m_2) \sin(\phi)}{\ell (m_1 + m_2 \sin^2(\phi))}$$

# Engineering students can take advantage of code at every single lecture

- Currently they use a pocket calculator **after they learnt** learning arithmetics at school
- Now they'll employ computational algebra **after they learnt** algebra and calculus
  - ▶ Focus on new skills, not in automatable calculations
  - ▶ Employing numerical calculus they solve what is impossible in a blackboard/paper



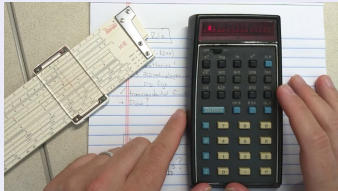
```
[14]: sistemaEcuaciones = [  
    x_EL,  
    phi_EL,  
]  
variablesDespeje = [x.diff(t,2), phi.diff(t,2)] # despejar aceleraciones generalizadas  
variablesDespeje_sol= sym.nonlinsolve(sistemaEcuaciones, variablesDespeje ).args[0]  
  
[15]: x_pp = sym.Eq(variablesDespeje[0], variablesDespeje_sol.args[0]) # [m s-2]  
phi_pp = sym.Eq(variablesDespeje[1], variablesDespeje_sol.args[1]) # [m s-2]  
x_pp, phi_pp
```

$$\ddot{x} = \frac{-\ell g m_2 \sin(\phi) + \frac{\ell m_2 (\ell m_2 \cos(\phi) \dot{\phi}^2 + g m_1 + g m_2) \sin(\phi)}{m_1 + m_2 \sin^2(\phi)}}{\ell m_2 \cos(\phi)}, \quad \ddot{\phi} = -\frac{(\ell m_2 \cos(\phi) \dot{\phi}^2 + g m_1 + g m_2) \sin(\phi)}{\ell (m_1 + m_2 \sin^2(\phi))}$$



# Engineering students can take advantage of code at every single lecture

- Currently they use a pocket calculator **after they learnt** learning arithmetics at school
- Now they'll employ computational algebra **after they learnt** algebra and calculus
  - ▶ Focus on new skills, not in automatable calculations
  - ▶ Employing numerical calculus they solve what is impossible in a blackboard/paper



```
[14]: sistemaEcuaciones = [  
    x_EL,  
    phi_EL,  
]  
variablesDespeje = [x.diff(t,2), phi.diff(t,2)] # despejar aceleraciones generalizadas  
variablesDespeje_sol= sym.nonlinsolve(sistemaEcuaciones, variablesDespeje ).args[0]  
  
[15]: x_pp = sym.Eq(variablesDespeje[0], variablesDespeje_sol.args[0]) # [m s-2]  
phi_pp = sym.Eq(variablesDespeje[1], variablesDespeje_sol.args[1]) # [m s-2]  
x_pp, phi_pp
```

$$[15]: \left( \begin{array}{l} \ddot{x} = \frac{-\ell g m_2 \sin(\phi) + \frac{\ell m_2 (\ell m_2 \cos(\phi) \dot{\phi}^2 + g m_1 + g m_2) \sin(\phi)}{m_1 + m_2 \sin^2(\phi)}}{\ell m_2 \cos(\phi)}, \quad \ddot{\phi} = -\frac{(\ell m_2 \cos(\phi) \dot{\phi}^2 + g m_1 + g m_2) \sin(\phi)}{\ell (m_1 + m_2 \sin^2(\phi))} \end{array} \right)$$

Papert (1980) “...the best learning takes place when the learner takes charge”

- An example problem is solved by the professor provided code

# Engineering students can take advantage of code at every single lecture

- Currently they use a pocket calculator **after they learnt** learning arithmetics at school
- Now they'll employ computational algebra **after they learnt** algebra and calculus
  - ▶ Focus on new skills, not in automatable calculations
  - ▶ Employing numerical calculus they solve what is impossible in a blackboard/paper



```
[14]: sistemaEcuaciones = [  
      x_EL,  
      phi_EL,  
      ]  
variablesDespeje = [x.diff(t,2), phi.diff(t,2)] # despejar aceleraciones generalizadas  
variablesDespeje_sol= sym.nonlinsolve(sistemaEcuaciones, variablesDespeje ).args[0]  
  
[15]: x_pp = sym.Eq(variablesDespeje[0], variablesDespeje_sol.args[0]) # [m s-2]  
      phi_pp = sym.Eq(variablesDespeje[1], variablesDespeje_sol.args[1]) # [m s-2]  
      x_pp, phi_pp
```

$$[15]: \left( \begin{array}{l} \ddot{x} = \frac{-\ell g m_2 \sin(\phi) + \frac{\ell m_2 (\ell m_2 \cos(\phi) \dot{\phi}^2 + g m_1 + g m_2) \sin(\phi)}{m_1 + m_2 \sin^2(\phi)}}{\ell m_2 \cos(\phi)}, \quad \ddot{\phi} = -\frac{(\ell m_2 \cos(\phi) \dot{\phi}^2 + g m_1 + g m_2) \sin(\phi)}{\ell (m_1 + m_2 \sin^2(\phi))} \end{array} \right)$$

Papert (1980) “...the best learning takes place when the learner takes charge”

- An example problem is solved by the professor provided code
- The student **recycles** it to solve other related problems

# Engineering students can take advantage of code at every single lecture

- Currently they use a pocket calculator **after they learnt** learning arithmetics at school
- Now they'll employ computational algebra **after they learnt** algebra and calculus
  - ▶ Focus on new skills, not in automatable calculations
  - ▶ Employing numerical calculus they solve what is impossible in a blackboard/paper



```
[14]: sistemaEcuaciones = [  
      x_EL,  
      phi_EL,  
    ]  
variablesDespeje = [x.diff(t,2), phi.diff(t,2)] # despejar aceleraciones generalizadas  
variablesDespeje_sol= sym.nonlinsolve(sistemaEcuaciones, variablesDespeje ).args[0]  
  
[15]: x_pp = sym.Eq(variablesDespeje[0], variablesDespeje_sol.args[0]) # [m s-2]  
      phi_pp = sym.Eq(variablesDespeje[1], variablesDespeje_sol.args[1]) # [m s-2]  
      x_pp, phi_pp
```

$$[15]: \left( \begin{array}{l} \ddot{x} = \frac{-\ell g m_2 \sin(\phi) + \frac{\ell m_2 (\ell m_2 \cos(\phi) \dot{\phi}^2 + g m_1 + g m_2) \sin(\phi)}{m_1 + m_2 \sin^2(\phi)}}{\ell m_2 \cos(\phi)}, \ddot{\phi} = -\frac{(\ell m_2 \cos(\phi) \dot{\phi}^2 + g m_1 + g m_2) \sin(\phi)}{\ell (m_1 + m_2 \sin^2(\phi))} \end{array} \right)$$

Papert (1980) “...the best learning takes place when the learner takes charge”

- An example problem is solved by the professor provided code
- The student **recycles** it to solve other related problems
- Gradually becoming autonomous by reusing not the provided but his own code

# Tools — Jupyter notebook: text + equations + executable code

The screenshot shows a Jupyter Notebook window with the title "PÉNDULO ENHEBRADO SOLVED.IPYNB". The left sidebar contains a table of contents with the following items:

- Péndulo enhebrado en aro
- Enunciado
- 1. Construya el Lagrangiano y la(s) función(es) de ligadura
- 2. Calcule las ecuaciones de Euler-Lagrange
- 3. Obtenga una expresión para la tensión que ejerce la barra
- 4. Grafique la tensión en los primeros diez segundos de la dinámica.

The main content area displays the following text and code:

### 3. Obtenga una expresión para la tensión que ejerce la barra

$$Q_d = \lambda_1 \frac{\partial f_1}{\partial d} = \lambda_1$$

Por tanto hay que resolver el sistema con las 3 ecuaciones de Euler-Lagrange y la única de ligadura para determinar  $\lambda_1$ . Esta última hay que resolverla para su caso homogéneo y expresar su derivada segunda para que esté en el mismo orden que las de Euler-Lagrange, a fin de cuentas estamos resolviendo sistemas diferenciales de 2.º orden.

```
[14]: f_1
```

$$f_1 = -l + d$$

Determinamos también  $\ddot{\theta}_1$  y  $\ddot{\theta}_2$  pues serán necesarias para los cálculos numéricos posteriores.

```
[15]: sistema = [theta1_EL.expand(),
               theta2_EL.expand(),
               d_EL.expand(),
               sym.Eq(f_1.rhs.diff(t,2), 0), # esto es igual a d punto punto = 0
               ]
variables = [theta1.diff(t,2), theta2.diff(t,2), lambda_1]
variables_sol = sym.nonlinsolve(sistema, variables).args[0]
```

```
[16]: lambda_1_sol = sym.Eq(lambda_1, variables_sol.args[2])
      lambda_1_sol.simplify()
```

```
[16]:
```

$$\lambda_1 = \frac{m \left( 2a \cos(\theta_1 - \theta_2) \dot{\theta}_1^2 + g \cos(2\theta_1 - \theta_2) + g \cos(\theta_2) + 2d\ddot{\theta}_2 - 2\ddot{d} \right)}{\cos(2\theta_1 - 2\theta_2) - 3}$$

# Tools — Google Colaboratory runs Jupyter on-line

- The platform allows professors to edit and comment on the work of students.
- Students can collaborate remotely, working on the same Jupyter notebook.

The screenshot displays a Google Colaboratory notebook titled "07 No conservativas | ej4". The interface includes a top menu bar with options like "Archivo", "Editar", "Ver", "Insertar", "Entorno de ejecución", "Herramientas", and "Ayuda". On the right, there are buttons for "Comentar", "Compartir", and a settings icon. The notebook content is divided into two main sections: a code cell and a text cell.

**Code Cell:** The code defines a potential energy function  $V$  using SymPy. It includes comments in Spanish explaining the steps. Below the code, the resulting mathematical expression for  $V$  is shown:

$$V = gm_1(-l_{10} - x_1) \sin(\theta) + \frac{k_1(l_{10} + x_1)^2}{2} + \frac{k_2(l_{20} + x)^2}{2}$$

**Text Cell:** The text "Lagrangiano" is followed by a code cell defining the Lagrangian  $L$  as  $L = T - V$ . Below this, the full Lagrangian expression is displayed:

$$\mathcal{L} = -gm_1(-l_{10} - x_1) \sin(\theta) - \frac{k_1(l_{10} + x_1)^2}{2} - \frac{k_2(l_{20} + x)^2}{2} + \frac{(m_0 + m_1)(2 \cos(\theta) \dot{x}_1 \dot{x} + \dot{x}^2 + \dot{x}_1^2)}{2}$$

Below the Lagrangian, the text "ECUACIONES DE EULER" is visible, followed by a code cell starting with "Para x".

**Comments:** On the right side of the notebook, there are two comments from "Victor Alexis Bettachini" dated "31 de may. de 2021". The first comment discusses the collinearity of the spring force  $k_1$  and the coordinate  $x_1$ , and the second comment asks why the spring constants  $k$  are negative.

# Tools — All material in a GitHub repository

- Clearly organised, freely accesible, and easy to update.
- Google Colaboratory loads Jupyter notebooks directly from GitHub.

The screenshot shows the GitHub interface for the repository 'bettachini / UNLaM\_MecanicaGeneral'. The repository has 1 pull request, 1 star, and 0 forks. The 'Code' tab is selected, showing a file tree with 11 folders. The 'About' section on the right describes the repository as a collection of materials for the 'Mecánica General' course at the Universidad Nacional de La Matanza.

Repository: bettachini / UNLaM\_MecanicaGeneral

Unwatch 1 Star 0 Fork 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master 1 branch 0 tags

Go to file Add file Code

File/Folder	Description	Last Commit
<b>bettachini slides erste</b> 3167a56 yesterday 188 commits		
00Generalidades	parcial retrasado	last month
01Newtoniana	09TensorInercia update	2 days ago
02NoInercial	cc by-nc-sa	last month
03Energia	cc by-nc-sa	last month
04EulerLagrange	Atwood compuestas	16 days ago
05Simulación	09TensorInercia update	2 days ago
06FuerzasLigadura	09 update lpybno	4 days ago
07NoConservativas	07NoConservativas   noConservativasEjemplos.lpynb	16 days ago
10TensorInercia	10TensorInercia lpynb update 2	2 days ago
11RotaciónEuler	numbering	2 days ago

**About**

Repositorio de la asignatura "Mecánica General" de la carrera de grado en Ingeniería mecánica de la Universidad Nacional de La Matanza.

physics mechanical-engineering fisica  
mecanica lagrangian vibrations  
mechanics-of-solids

Readme View license

**Releases**

No releases published  
[Create a new release](#)

# Tools — From the 1st class — L<sup>A</sup>T<sub>E</sub>X: concise mathematical notation

- L<sup>A</sup>T<sub>E</sub>X typesetting follows strictly the standards of the American Mathematical Society

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  $\ddot{\vec{r}} = (\ddot{r} - r\dot{\varphi}^2)\hat{r} + (r\dot{\varphi}^2 + r\ddot{\varphi})\hat{\varphi} + \ddot{z}\hat{z}$ , nos quedamos solo con la componente en  $\hat{\varphi}$ ,

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

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  $\varphi$  es

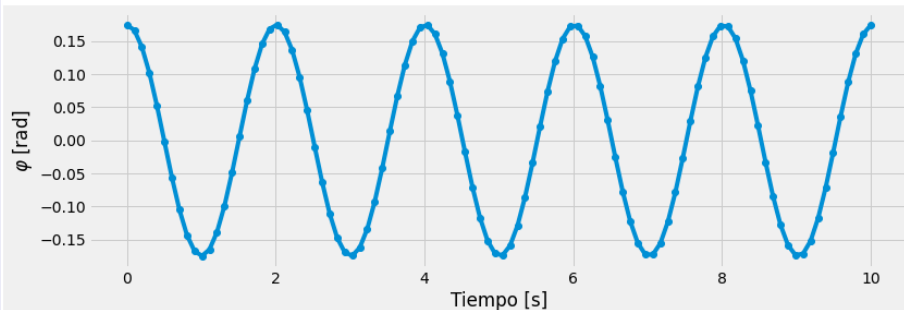
$$\frac{\partial}{\partial \varphi}V(\varphi) = mg\ell \sin(\varphi)$$

# Tools — 1st class — Matplotlib: precise and reproducible graphics

- Explicit Python code for producing graphics. Students are enticed to play with it.

```
# graficación
fig, ax = plt.subplots(figsize=(12, 4))
ax.plot(tiempos, phi(tiempos), 'o-')
ax.set_xlabel('Tiempo [s]')
ax.set_ylabel(r'$\varphi$ [rad]')
```

```
Text(0, 0.5, '$\varphi$ [rad]')
```





## Tools — 3rd class — SymPy: symbolic calculations

- Students effort centred on new subjects by freeing them of calculus and algebraic tasks.

```
[8]: m2_v_cuadrado = m2_v.dot(m2_v)
     m2_v_cuadrado
```

```
[8]:  $\ell^2 \sin^2(\varphi) \dot{\varphi}^2 + (\ell \cos(\varphi) \dot{\varphi} + \dot{x})^2$ 
```

Con esto la energía cinética queda

$$\begin{aligned} T(\dot{x}_1, \varphi, \dot{\varphi}) &= \frac{m_1}{2} (\dot{r}_1)^2 + \frac{m_2}{2} (\dot{r}_2)^2 \\ &= \frac{m_1}{2} \dot{x}^2 + \frac{m_2}{2} (\dot{x}^2 + 2\dot{x}\ell \cos \varphi \dot{\varphi} + \ell^2 \dot{\varphi}^2) \end{aligned}$$

```
[9]: # Energía cinética
unMedio = sym.Rational(1,2) # Rational: fracción de enteros, alternatively podría haberse usado 0.5
m1_T = unMedio*m1*m1_v_cuadrado
m2_T = unMedio*m2*m2_v_cuadrado
T = sym.Eq(sym.Symbol('T'), (m1_T + m2_T) ) # simplify: simplifica usando factor común y otras operaciones
T
```

```
[9]: 
$$T = \frac{m_1 \dot{x}^2}{2} + \frac{m_2 (\ell^2 \sin^2(\varphi) \dot{\varphi}^2 + (\ell \cos(\varphi) \dot{\varphi} + \dot{x})^2)}{2}$$

```

# Tools — 4th class — Equations for Lagrangian dynamics

## Ecuaciones de Euler-Lagrange

Para  $x$

```
[8]: x_EL = sym.Eq(L.rhs.diff(x) - L.rhs.diff(x.diff(t)).diff(t), 0).simplify() # ecuación igualando a cero
      x_EL
```

$$[8]: m_1 \ddot{x} + m_2 \left( -\ell \sin(\phi) \dot{\phi}^2 + \ell \cos(\phi) \ddot{\phi} + \ddot{x} \right) = 0$$

Esta es una ecuación diferencial lineal de segundo orden homogénea. De aquí podría despejarse  $\ddot{x}$

```
[9]: sym.Eq(x.diff(t,2),
           list( sym.solve(x_EL, x.diff(t,2) ) ) [0] # solveset devuelve un set, que convertimos a lista
           ) # aceleración = x punto punto [m s-2]
```

$$[9]: \ddot{x} = \frac{\ell m_2 \left( \sin(\phi) \dot{\phi}^2 - \cos(\phi) \ddot{\phi} \right)}{m_1 + m_2}$$

Pero queda en función de otra aceleración  $\ddot{\phi}$ .

Para  $\phi$

```
[10]: phi_EL = sym.Eq(L.rhs.diff(phi) - L.rhs.diff(phi.diff(t)).diff(t), 0).simplify() # ecuación igualando a cero
       phi_EL
```

## Tools — 4th class — Automatisation of resolutions

- Mathematical complexity doesn't limit the scope of tackled mechanical problems.

```
[14]: sistemaEcuaciones = [  
        x_EL,  
        phi_EL,  
    ]  
variablesDespeje = [x.diff(t,2), phi.diff(t,2)] # despejar aceleraciones generalizadas  
variablesDespeje_sol = sym.nonlinsolve(sistemaEcuaciones, variablesDespeje ).args[0]
```

```
[15]: x_pp = sym.Eq(variablesDespeje[0], variablesDespeje_sol.args[0] ) # [m s-2]  
phi_pp = sym.Eq(variablesDespeje[1], variablesDespeje_sol.args[1] ) # [m s-2]  
x_pp, phi_pp
```

$$[15]: \left( \ddot{x} = \frac{-\ell g m_2 \sin(\phi) + \frac{\ell m_2 (\ell m_2 \cos(\phi) \dot{\phi}^2 + g m_1 + g m_2) \sin(\phi)}{m_1 + m_2 \sin^2(\phi)}}{\ell m_2 \cos(\phi)}, \ddot{\phi} = -\frac{(\ell m_2 \cos(\phi) \dot{\phi}^2 + g m_1 + g m_2) \sin(\phi)}{\ell (m_1 + m_2 \sin^2(\phi))} \right)$$

## Tools — 5th class — SciPy: numerical computation of results

```
[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]

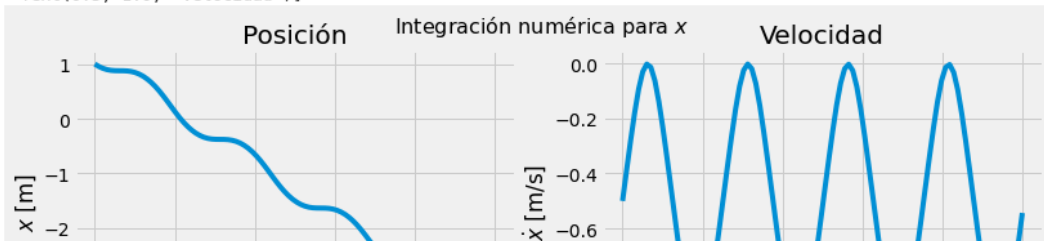
[25]: array([ 1.          ,  0.95510744,  0.92131146,  0.89820932,  0.88468059,
            0.87877042,  0.87745354,  0.87702754,  0.87352768,  0.86357726,
            0.84474673,  0.81565733,  0.77559949,  0.72423163,  0.66166451,
```

# Tools — 5th class — Graphical analysis of numerical results

```
[26]: solucion = y_ode2
      nombreCoordenada = 'x'

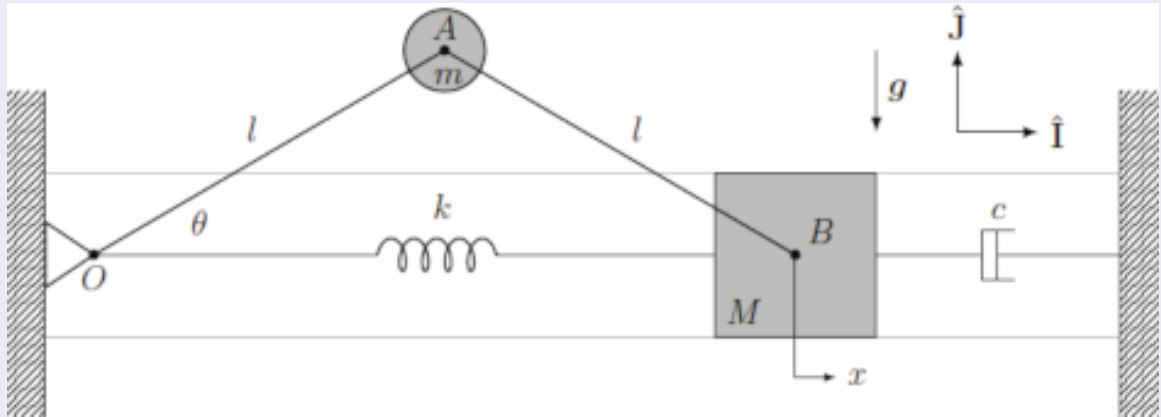
      fig, ax = plt.subplots(nrows= 1, ncols= 2, squeeze=False, figsize=(12, 4)) # dos figuras en la misma fila
      fig.suptitle('Integración numérica para $'+ nombreCoordenada + '$', fontsize=16)
      ax[0,0].plot(solucion.t, solucion.y[0]) # posición x
      ax[0,0].set(xlabel='t [s]', ylabel='$' + nombreCoordenada + '$ [m]', title='Posición')
      ax[0,1].plot(solucion.t, solucion.y[1]) # velocidad x
      ax[0,1].set(xlabel='t [s]', ylabel='$\dot{x}$ [m/s]', title='Velocidad')
```

```
[26]: [Text(0.5, 0, 't [s]'),
      Text(0, 0.5, '$\dot{x}$ [m/s]'),
      Text(0.5, 1.0, 'Velocidad')]
```



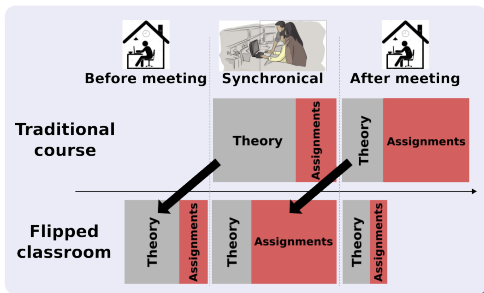
## Tools — 7th class — Adding complexity

- Code from previous classes is **recycled** to model more realistic devices.



# Methodology — Weekly cycle of our flipped classroom

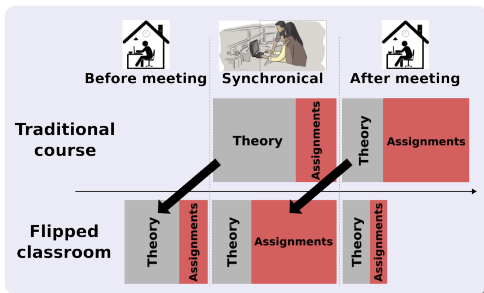
- New theory (Jupyter notebooks and videos) and assignments published on-line



	Synchronic	Theory	Assignments
Before		Read and apply	Start them
During		Consultations	Complete them
After		Additional consultations	TA's corrections

# Methodology — Weekly cycle of our flipped classroom

- New theory (Jupyter notebooks and videos) and assignments published on-line
- Previous week assignments **must** be turned-in for scoring

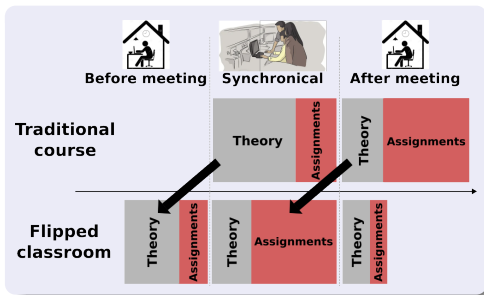


	Synchronic	Theory	Assignments
Before		Read and apply	Start them
During		Consultations	Complete them
After		Additional consultations	TA's corrections



# Methodology — Weekly cycle of our flipped classroom

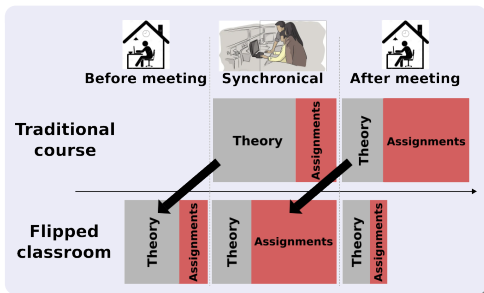
- New theory (Jupyter notebooks and videos) and assignments published on-line
- Previous week assignments **must** be turned-in for scoring
- On-line 24/7 **asynchronous** consultations that are **public** to other students



Synchronic	Theory	Assignments
Before	Read and apply	Start them
During	Consultations	Complete them
After	Additional consultations	TA's corrections


# Methodology — Weekly cycle of our flipped classroom

- New theory (Jupyter notebooks and videos) and assignments published on-line
- Previous week assignments **must** be turned-in for scoring
- On-line 24/7 **asynchronous** consultations that are **public** to other students
- **synchronous** meetings with TA's to finish assignments



	Synchronic	Theory	Assignments
Before		Read and apply	Start them
During		Consultations	Complete them
After		Additional consultations	TA's corrections

# Tools — Google Colaboratory: asynchronic remote assistance

 07 No conservativas | ej4 ☆

Archivo Editar Ver Insertar Entorno de ejecución Herramientas Ayuda [Se editó por última vez: 3 de junio](#)

Comentar Compartir

+ Código + Texto

Conectar Editando

```
[ ] # Energía potencial
m1_V = - (m1* g* (- N.y)).dot(m1_r)
# pot_k1 = unMedio* ( -k1* ((l10 + x1)* (sym.cos(theta) - sym.sin(theta)) )**2 ) # mal
pot_k1 = unMedio* k1* (l10 + x1)**2 # Lo escribí yo
# pot_k2 = unMedio* -k2* (l20 + x)**2
pot_k2 = unMedio* k2* (l20 + x)**2
V = sym.Eq(sym.Symbol('V'), m1_V + pot_k1 + pot_k2 ) #agrega el potencial elastico k en la ecuacion
V
```

$$V = gm_1(-l_{10} - x_1)\sin(\theta) + \frac{k_1(l_{10} + x_1)^2}{2} + \frac{k_2(l_{20} + x)^2}{2}$$


### ▼ Lagrangiano

```
[ ] L = sym.Eq(sym.Symbol('\mathcal{L}'), (T.rhs - V.rhs))
L
```

$$\mathcal{L} = -gm_1(-l_{10} - x_1)\sin(\theta) - \frac{k_1(l_{10} + x_1)^2}{2} - \frac{k_2(l_{20} + x)^2}{2} + \frac{(m_0 + m_1)(2\cos(\theta)\dot{x}\dot{x}_1 + \dot{x}^2 + \dot{x}_1^2)}{2}$$

### ECUACIONES DE EULER


Para x

 Victor Alexis Bettachini [Resolver](#)

31 de may. de 2021  
(editado el 31 de may. de 2021)

- El estiramiento del resorte de k\_1 es colineal con x1. No tienen sentido pensar en proyecciones (si es lo que hiciste, que realmente no entiendo).

- ¿Porque negativos los k?

 Victor Alexis Bettachini [Resolver](#)

31 de may. de 2021

vbettachini@unlam.edu.ar

A code centred mechanics subject

2023-11-23

16 / 19



# Course — Summary of methodology

## A course centred on code

- Theory: text + equations + executable code in digital notebooks.

## Flipped classroom

# Course — Summary of methodology

## A course centred on code

- Theory: text + equations + executable code in digital notebooks.
- Reinforced by: suggested bibliography and short professor's videos.

## Flipped classroom

# Course — Summary of methodology

## A course centred on code

- Theory: text + equations + executable code in digital notebooks.
- Reinforced by: suggested bibliography and short professor's videos.
- Assignments: professor's code recycling.

## Flipped classroom

# Course — Summary of methodology

## A course centred on code

- Theory: text + equations + executable code in digital notebooks.
- Reinforced by: suggested bibliography and short professor's videos.
- Assignments: professor's code recycling.
- On-line:

## Flipped classroom



# Course — Summary of methodology

## A course centred on code

- Theory: text + equations + executable code in digital notebooks.
- Reinforced by: suggested bibliography and short professor's videos.
- Assignments: professor's code recycling.
- On-line:
  - ▶ Remote collaboration and correction

## Flipped classroom

# Course — Summary of methodology

## A course centred on code

- Theory: text + equations + executable code in digital notebooks.
- Reinforced by: suggested bibliography and short professor's videos.
- Assignments: professor's code recycling.
- On-line:
  - ▶ Remote collaboration and correction
  - ▶ Doesn't require powerful nor at campus computers

## Flipped classroom

# Course — Summary of methodology

## A course centred on code

- Theory: text + equations + executable code in digital notebooks.
- Reinforced by: suggested bibliography and short professor's videos.
- Assignments: professor's code recycling.
- On-line:
  - ▶ Remote collaboration and correction
  - ▶ Doesn't require powerful nor at campus computers
  - ▶ A dated record of each student's work

## Flipped classroom

# Course — Summary of methodology

## A course centred on code

- Theory: text + equations + executable code in digital notebooks.
- Reinforced by: suggested bibliography and short professor's videos.
- Assignments: professor's code recycling.
- On-line:
  - ▶ Remote collaboration and correction
  - ▶ Doesn't require powerful nor at campus computers
  - ▶ A dated record of each student's work

## Flipped classroom

- Theory: emphasis on student's autonomus reading

# Course — Summary of methodology

## A course centred on code

- Theory: text + equations + executable code in digital notebooks.
- Reinforced by: suggested bibliography and short professor's videos.
- Assignments: professor's code recycling.
- On-line:
  - ▶ Remote collaboration and correction
  - ▶ Doesn't require powerful nor at campus computers
  - ▶ A dated record of each student's work

## Flipped classroom

- Theory: emphasis on student's autonomus reading
- Consultations: mostly on-line asynchronical and publicly accessible

# Course — Summary of methodology

## A course centred on code

- Theory: text + equations + executable code in digital notebooks.
- Reinforced by: suggested bibliography and short professor's videos.
- Assignments: professor's code recycling.
- On-line:
  - ▶ Remote collaboration and correction
  - ▶ Doesn't require powerful nor at campus computers
  - ▶ A dated record of each student's work

## Flipped classroom

- Theory: emphasis on student's autonomus reading
- Consultations: mostly on-line asynchronical and publicly accessible
- Synchronical mettings: TA's personal assistance for completing assignments

# Course — Latest developments

2023: student's feedback improved:

- Repository material: theory notes and code revised from class to class
- Evaluation: grading each assignment led to a higher student's performance

2024:

- AI: students to code in *GitHub Codespaces* with *GitHub Copilot*
- Translation: repository content to English (collaborations welcome!)

## GitHub Copilot AI assists with coding

```
lagrangiano = (T.rhs - V.rhs).expand()
t = sym.Symbol('t') # como se deriva respecto al tiempo con la función diff se declara t como símbolo
return sym.Eq(
    lagrangiano.diff(coordenadaGeneralizada)
    - lagrangiano.diff(coordenadaGeneralizada.diff(t)).diff(t)
    , 0
).simplify()
```

[21]

```
x1_EL = eulerLagrange(T, V, x1)
x1_EL
```

[22]

...  $\frac{\pi^2 M \ddot{x}_1}{2} - g m_1 + g m_2 + m_1 \ddot{x}_1 + m_2 \ddot{x}_1 = 0$

Esta es una ecuación diferencial lineal de segundo orden homogénea. De aquí se puede despejar  $\ddot{x}$

▷

```
#Despejar x1PuntoPunto
k1PuntoPunto = sym.solve(x1_EL, x1.diff(t, t)).args[0]
```