

# 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

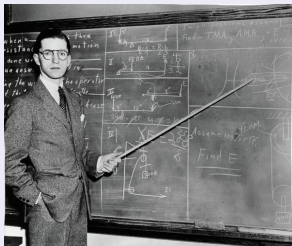


## Value student's and professor's time

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

# Value student's and professor's time

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

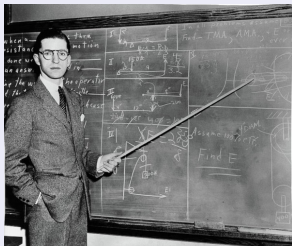


Most current classrooms: an exercise on transcription

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

# Value student's and professor's time

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

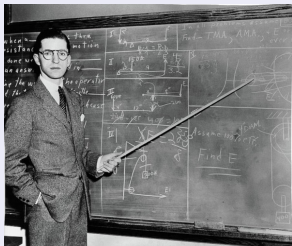


Most current classrooms: an exercise on transcription

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

# Value student's and professor's time

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

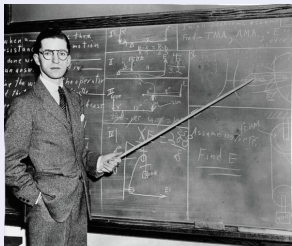


Most current classrooms: an exercise on transcription

- Professor: prepares lessons  $\xrightarrow{\text{by heart}}$  blackboard/slides
- Student: blackboard/slides  $\xrightarrow{\text{transcribes}}$  own notebooks
- Assignments: do **again** same calculations, drawings,...

# Value student's and professor's time

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

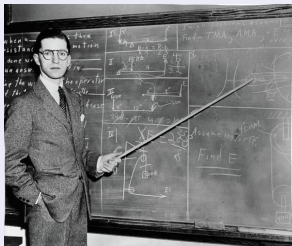


Most current classrooms: an exercise on transcription

- Professor: prepares lessons  $\xrightarrow{\text{by heart}}$  blackboard/slides
- Student: blackboard/slides  $\xrightarrow{\text{transcribes}}$  own notebooks
- Assignments: do **again** same calculations, drawings,...
- Boredom  $\rightarrow$  loss of concentration on the subject

# Value student's and professor's time

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

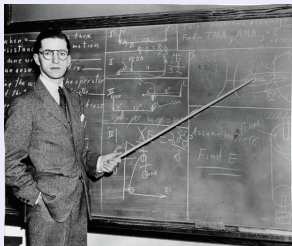


Most current classrooms: an exercise on transcription

- Professor: prepares lessons  $\xrightarrow{\text{by heart}}$  blackboard/slides
- Student: blackboard/slides  $\xrightarrow{\text{transcribes}}$  own notebooks
- Assignments: do **again** same calculations, drawings,...
- Boredom  $\rightarrow$  loss of concentration on the subject

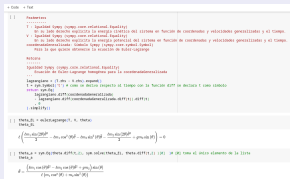
# Value student's and professor's time

Licklider (1957): 85 % of “thinking” are actually mundane tasks (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
- Assignments: do **again** same calculations, drawings,...
- Boredom  $\rightarrow$  loss of concentration on the subject



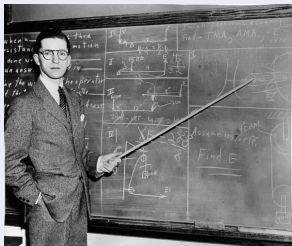
Digital course material with embedded code

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



# Value student's and professor's time

Licklider (1957): 85 % of “thinking” are actually mundane tasks (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
- Assignments: do **again** same calculations, drawings,...
- Boredom  $\rightarrow$  loss of concentration on the subject

```
1 // Parameters
2 // ...
3 // ...
4 // ...
5 // ...
6 // ...
7 // ...
8 // ...
9 // ...
10 // ...
11 // ...
12 // ...
13 // ...
14 // ...
15 // ...
16 // ...
17 // ...
18 // ...
19 // ...
20 // ...
21 // ...
22 // ...
23 // ...
24 // ...
25 // ...
26 // ...
27 // ...
28 // ...
29 // ...
30 // ...
31 // ...
32 // ...
33 // ...
34 // ...
35 // ...
36 // ...
37 // ...
38 // ...
39 // ...
40 // ...
41 // ...
42 // ...
43 // ...
44 // ...
45 // ...
46 // ...
47 // ...
48 // ...
49 // ...
50 // ...
51 // ...
52 // ...
53 // ...
54 // ...
55 // ...
56 // ...
57 // ...
58 // ...
59 // ...
60 // ...
61 // ...
62 // ...
63 // ...
64 // ...
65 // ...
66 // ...
67 // ...
68 // ...
69 // ...
70 // ...
71 // ...
72 // ...
73 // ...
74 // ...
75 // ...
76 // ...
77 // ...
78 // ...
79 // ...
80 // ...
81 // ...
82 // ...
83 // ...
84 // ...
85 // ...
86 // ...
87 // ...
88 // ...
89 // ...
90 // ...
91 // ...
92 // ...
93 // ...
94 // ...
95 // ...
96 // ...
97 // ...
98 // ...
99 // ...
100 // ...
```

Digital course material with embedded code

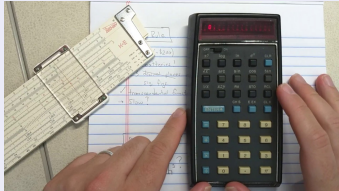
- Professor: ideas  $\xrightarrow{\text{updates}}$  code/theory in repository
- Student: course repository  $\xrightarrow{\text{clones}}$  its own modifiable one



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** arithmetics at school



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

- Currently they use a pocket calculator **after they learnt** 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** 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
```

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

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

- Currently they use a pocket calculator **after they learnt** 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
```

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

Engineering students can take advantage of code at every single lecture



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

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

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

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 notebooks 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 Jupyter notebook. The top bar includes the Google Colab logo, the notebook title "07 No conservativas | ej4", and navigation links like "Archivo", "Editar", "Ver", "Insertar", "Entorno de ejecución", "Herramientas", and "Ayuda". On the right, there are buttons for "Comentar", "Compartir", and a user profile icon.

The notebook content is divided into two main sections. The first section, titled "+ Código", contains a Jupyter code cell with Python code using SymPy for symbolic computation. The code defines variables for a physics problem involving a mass-spring system and calculates the potential energy  $V$ . Below the code, the resulting mathematical expression for  $V$  is shown:

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

The second section, titled "Lagrangiano", contains another code cell defining the Lagrangian  $L$  as the difference between kinetic energy  $T$  and potential energy  $V$ . Below the code, the resulting expression for  $L$  is displayed:

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

Below the Lagrangian section, the text "ECUACIONES DE EULER" is visible. At the bottom, there is a section titled "Para x".

On the right side of the notebook, there is a comment thread. The first comment is from Victor Alexis Bettachini, dated 31 de may. de 2021, discussing the physical interpretation of the spring constants  $k_1$  and  $k_2$ . The comment text is: "El estiramiento del resorte de  $k_1$  es colineal con  $x_1$ . No tienen sentido pensar en proyecciones (si es lo que hiciste, que realmente no entiendo). ¿Porque negativos los  $k$ ?"

# Tools — All course material in a GitHub repository

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

The screenshot shows the GitHub repository page for 'MecanicaAnaliticaComputacional' by user 'bettachini'. The repository is public and has 3 stars, 2 forks, and 3 watchers. It is currently on the 'master' branch with 2 branches in total. A notification states 'Your master branch isn't protected'. The repository contains a list of files and folders, including '00Generalidades', '01Vectorial', '03Energía', '04EulerLagrange', '05Ligaduras', '06Simulación', '07FuerzasLigadura', '08NoConservativas', '09aNoInercial', and '09bTensorInercia'. The 'About' section describes it as a repository for the 'Mecánica Analítica Computacional' course. The 'Releases' section shows the latest release 'v2023p' from August 10.

Repository: **MecanicaAnaliticaComputacional** (Public)

Tags: master (2 branches), 1 tag

Actions: Go to file, Add file, Code

Notification: Your master branch isn't protected. Protect this branch.

Repository Description: Repositorio de la asignatura "Mecánica Analítica Computacional" de la carrera de grado en ingeniería mecánica de la Universidad Nacional de La Matanza.

Topics: python, physics, mechanical-engineering, ingeniería, física, mecánica, lagrangian, vibrations, mechanics-of-solids

Files and Folders:

File/Folder	Description	Updated
00Generalidades	Correcciones LaTeX	3 months ago
01Vectorial	re-escritura para que LaTeX en línea	3 months ago
03Energía	Correcciones LaTeX	3 months ago
04EulerLagrange	Modificada la definición de T y V para que devuelva ecuaciones y entr...	2 months ago
05Ligaduras	ligadura   atwood   replace 2	3 months ago
06Simulación	06Simulación solved	last month
07FuerzasLigadura	Atwood compuesta: coordenadas poleas en figura	5 months ago
08NoConservativas	no conservativas   cilindros solidarios	last month
09aNoInercial	copias araña	last month
09bTensorInercia	tensorInercia 2	3 weeks ago

Releases: v2023p (Latest) on Aug 10

# 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

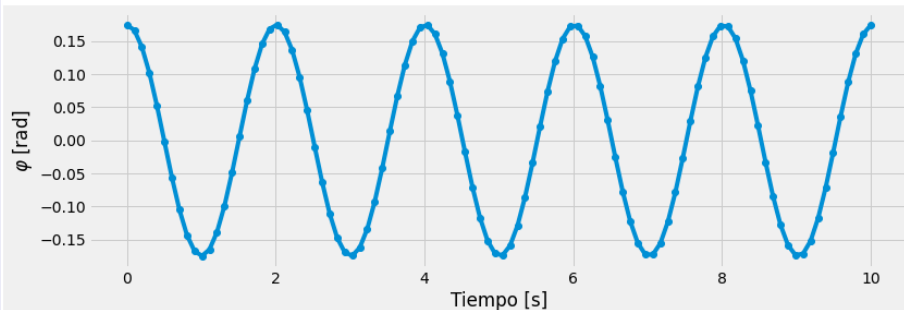
$$\frac{\partial}{\partial \varphi}V(\varphi) = mg\ell \sin(\varphi)$$

# 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

- Student's 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
```

- 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]

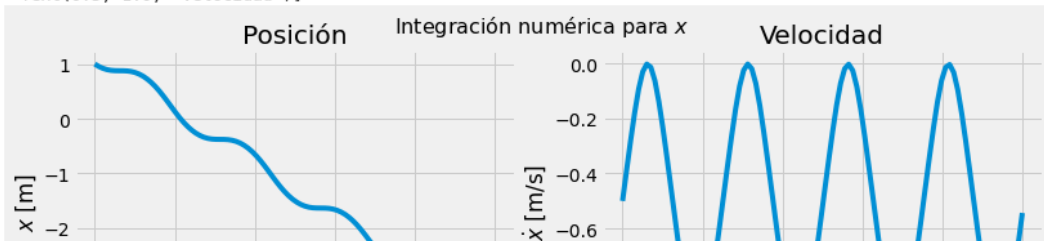
[25]: array([ 1.          ,  0.95510744,  0.92131146,  0.89820932,  0.88468059,
            0.87877042,  0.87745354,  0.87702754,  0.87352768,  0.86357726,
            0.84474673,  0.81565733,  0.77559949,  0.72423163,  0.66166451,
```

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

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

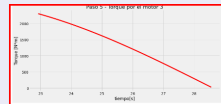
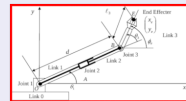
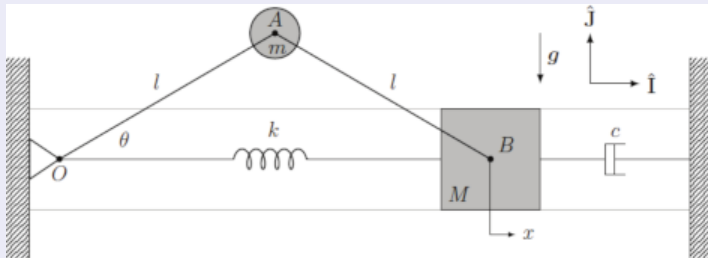
      fig, ax = plt.subplots(nrows= 1, ncols= 2, squeeze=False, figsize=(12, 4)) # dos figuras en la misma fila
      fig.suptitle('Integración numérica para '$'+ nombreCoordenada + '$', fontsize=16)
      ax[0,0].plot(solucion.t, solucion.y[0]) # posición x
      ax[0,0].set(xlabel='t [s]', ylabel='$'+ nombreCoordenada+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



# Tools — 8th class — GitHub Copilot: AI assistance for coding

- Now that the students grasp the basics they're invited to take advantage of AI
- After commenting in plain text what they need code is suggested

```
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]

$$\dots \quad \frac{\pi^2 M \ddot{x}_1}{2} - gm_1 + gm_2 + m_1 \ddot{x}_1 + m_2 \ddot{x}_1 = 0$$

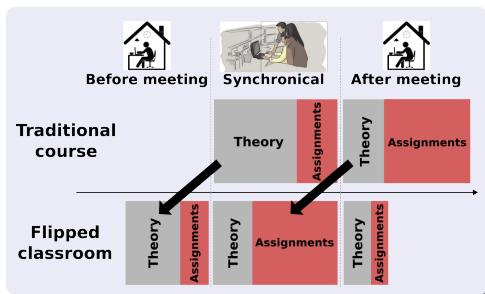
Esta es una ecuación diferencial lineal de segundo orden homogénea. De aquí se puede despejar  $\ddot{x}$

▷ ▾

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

# Methodology — Weekly cycle of our flipped classroom

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

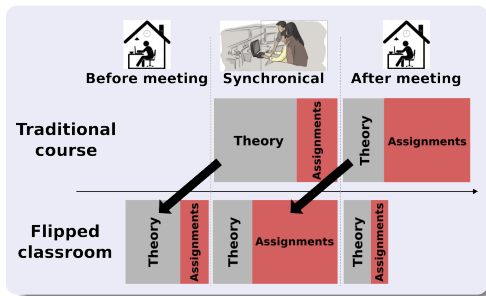


	Synchronic	Theory	Assignments
Before		Read and apply	Start them
During		Consultations	Complete them
After		Additional consultations	TA's corrections



# Methodology — Weekly cycle of our flipped classroom

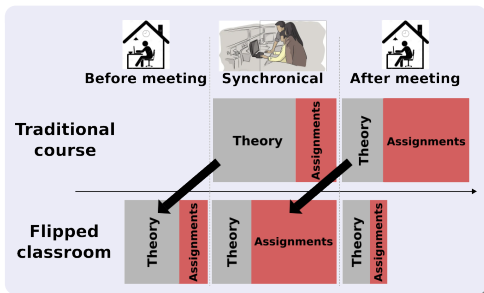
- New theory (Jupyter notebooks and videos) and assignments published on-line
- Previous week assignments **must** be turned-in for scoring



	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

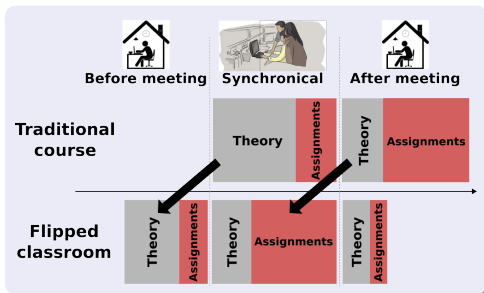
- New theory (Jupyter notebooks and videos) and assignments published on-line
- Previous week assignments **must** be turned-in for scoring
- On-line 24/7 **asynchronous** consultations that are **public** to other students



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

# Methodology — 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](#)

+ 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  $x_1$ . 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

18 / 21

## Methodology — Microsoft Teams: individualized student follow-up

A record of the weekly turn-up of assignments

Calificaciones							
Vencimiento el 28 sept	<div> <div>Exportar a Excel</div> <div></div> </div>						
g06e03	g06e04	g06e05	g05e01a	g05e01c	g05e02	g05e03	
28 sept	28 sept	28 sept	14 sept	14 sept	14 sept	14 sept	
<b>Promedio de clase</b>							
Alumno 1	Visto		Visto	Devuelto	Entregado	Entregado	Entregado
Alumno 2				Devuelto	Entregado	Entregado	Entregado
Alumno 3				Entregado	Entregado	Entregado	Entregado
Alumno 4	Visto	Visto	Visto	Entregado	Entregado	Entregado	Entregado
Alumno 5				Entregado	Entregado		Entregado
Alumno 6	Visto	Visto	Visto	Entregado	Entregado	Visto	Entregado

# Thank You!

# Summary

## A course centred on code

- Material: text + equations + executable code in digital notebooks in a repository
- Assignments: professor's code recycling
- On-line:
  - ▶ Remote collaboration and correction
  - ▶ Doesn't require powerful nor at campus computers
  - ▶ AI assistance for coding
  - ▶ Dated record of each student's work

## Flipped classroom

- Theory: emphasis on student's autonomous reading
- Reinforced by: suggested bibliography and short professor's videos
- Consultations: mostly on-line asynchronous and publicly accessible
- Synchronic meetings: TA's personal assistance for completing assignments