

HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY – VNU HCMC  
OFFICE FOR INTERNATIONAL STUDY PROGRAM  
FACULTY OF APPLIED SCIENCE

————— \* —————



## GENERAL PHYSICS 1 MATLAB PROJECT REPORT

Lecturer : **Dr. Nguyễn Trung Hậu**  
**Dr. Nguyễn Xuân Thanh Trâm**  
Subject : **General Physics 1**  
Class : **CC03**  
Group : **02**  
Members : **Huỳnh Dương Gia Bảo – 2252063**  
**Lương Triển Đạt – 2252144**  
**Phạm Hoàng Quân – 2252687**  
**Võ Minh Thăng – 2252762**  
Completion date : **April 6<sup>th</sup>, 2023**

Ho Chi Minh City, May 2<sup>nd</sup>, 2023

# Contents

<b>I.</b>	<b>Introduction</b>	2
<b>II.</b>	<b>Theory</b>	3
<b>III.</b>	<b>MATLAB Code and Explanation</b>	4
1.	Application user interface (UI) <code>app.mlapp</code>	4
2.	Matlab script <code>ComputerEnergy.m</code>	6
<b>IV.</b>	<b>General Information</b>	9
1.	Requirements	9
2.	Installation	9
3.	Features	9
<b>V.</b>	<b>Results and Discussion</b>	10
<b>VI.</b>	<b>Conclusion</b>	12
<b>VII.</b>	<b>References</b>	13

# I Introduction

The objective of this MATLAB project is to calculate and graphically represent the kinetic and potential energies of a particle moving under the influence of a conservative force. A conservative force is defined as a force whose work done on an object is independent of the path taken by the object, and depends only on its initial and final positions. In this project, we will consider a specific conservative force given by the expression:

$$F(x) = kx - 4qx^3$$

where  $k$  and  $q$  are parameters

To calculate the potential energy of the particle, we will integrate the force function with respect to position in order to obtain its value at any point along the particle's trajectory, i.e.,

$$U(x) = - \int F(x)dx$$

We will also calculate the kinetic energy of the particle, which is a function of its velocity, and hence of time. By combining the expressions for potential and kinetic energy, we can obtain the total mechanical energy of the particle, which is conserved for a conservative force.

Finally, we will use MATLAB to plot the functions for kinetic and potential energy as functions of time, along with the total mechanical energy. This will provide us with a graphical representation of the motion of the particle, which can help us to gain insights into its behavior under the influence of the conservative force.

## II Theory

In this MATLAB project, we will use the Euler-Cromer method[**euler**] to numerically compute the motion of a mass acted on by a conservative force. In the context of energy conservation, the total energy in the system (sum of kinetic and potential energies) is conserved for this type of motion. Specifically, the kinetic energy  $K$  of the mass is given by the expression:

$$K = \frac{1}{2}mv^2$$

where  $m$  is the mass,  $v$  is the velocity

Specifically, we will consider a potential energy function which has a corresponding conservative force given by the expression:

$$F(x) = kx - 4qx^3 \\ \Rightarrow U(x) = - \int F(x)dx = -\frac{k}{2}x^2 + qx^4$$

Using Newton's second law, we can compute the motion, including the acceleration  $a$ , of the mass under the influence of a given force. To numerically solve for the position and velocity of the mass, we will use the Euler-Cromer method, which is a variant of the leap-frog method. This method involves solving two ordinary differential equations that describe the motion of the mass:

$$\frac{dv}{dt} = a = \frac{F}{m} ; \frac{dx}{dt} = v$$

In the Euler-Cromer method, we use right derivative approximations to obtain iterative equations for the final values of velocity and position, where the subscripts  $i$  and  $f$  refer to the initial (time  $t$ ) and final (time  $t + \Delta t$ ) values. This allows us to compute the position of the mass with greater accuracy than the Euler method. Specifically, we use the new velocity value to compute the new position value as:

$$v_f = v_i + \frac{F_i}{m}\Delta t \Rightarrow x_f = x_i + v_i\Delta t$$

Of course, the approximation is only accurate when  $\Delta t$  is small. Solving each equation for the final values of velocity and position, we can obtain the numerical solution for the motion of the mass under the given force.

Using MATLAB, we will implement the Euler-Cromer method to compute the motion of the mass and demonstrate that the total energy of the system  $E = K + U$  is conserved. By plotting the position, velocity, kinetic energy, potential energy, and total energy as functions of time, we can gain insights into the behavior of the mass under the influence of the conservative force.

### III MATLAB Code and Explanation

The project consists of two parts: the application UI and the Matlab script.

#### 1. Application user interface (UI) `app.mlapp`<sup>1</sup>

This portion of the MATLAB code contains six methods that handle component events for the graphical user interface (GUI) including:

- **Compute energy on button press:**

```
39 function GraphButtonPushed(app, event)
40     ComputeEnergy(app);
41 end
```

It is a callback function for the **Graph** button that triggers the computation of the energy for the system and plots it on the axes.

- **Check and validate input value:**

```
44 function mEditFieldValueChanged(app, event)
45     if app.mEditField.Value <= 0
46         app.mEditField.Value = 1;
47         errordlg('Input must be greater than zero!', 'Invalid
↪ Input', 'modal');
48     end
49 end
50
52 function tEditFieldValueChanged(app, event)
53     if app.tEditField.Value <= 0
54         app.tEditField.Value = 10;
55         errordlg('Input must be greater than zero!', 'Invalid
↪ Input', 'modal');
56     end
57 end
```

The `mEditFieldValueChanged` and `tEditFieldValueChanged` methods are callback functions for the input field of the parameters  $m$  and  $t$ , respectively. Both methods check whether the input is greater than zero, and if it is not, they set the value to 1 (for `mEditField`) or 10 (for `tEditField`) and display an error message.

---

<sup>1</sup>Source code of `app.mlapp`: <https://github.com/datdadev/MATLAB-Conservative-Force-Energy-Analysis/blob/main/app.mlapp>.

- Toggle visibility of plot objects based on checkbox values:

```
61 function KECheckBoxValueChanged(app, event)
62     if ~app.KECheckBox.Value && ~app.PECheckBox.Value
63         app.KECheckBox.Value = true;
64         errordlg('At least one of the two checkboxes should be
↪ activated!', 'Warning', 'modal');
65     elseif length(app.UIAxes.Children) == 2
66         if app.KECheckBox.Value
67             set(app.UIAxes.Children(2), 'Visible', 'on');
68         else
69             set(app.UIAxes.Children(2), 'Visible', 'off');
70         end
71     end
72 end
73
75 function PECCheckBoxValueChanged(app, event)
76     if ~app.PECheckBox.Value && ~app.KECheckBox.Value
77         app.PECheckBox.Value = true;
78         errordlg('At least one of the two checkboxes should be
↪ activated!', 'Warning', 'modal');
79     elseif length(app.UIAxes.Children) == 2
80         if app.PECheckBox.Value
81             set(app.UIAxes.Children(1), 'Visible', 'on');
82         else
83             set(app.UIAxes.Children(1), 'Visible', 'off');
84         end
85     end
86 end
```

The callback functions `KECheckBoxValueChanged` and `PECCheckBoxValueChanged` are associated with the kinetic energy checkbox and potential energy checkbox, respectively. These functions check whether at least one of the checkboxes is activated. If neither checkbox is activated, the function activates one of them and displays a warning message. Otherwise, the function adjusts the visibility of the plot based on the checkbox value.

- About button for project info:

```
88 function AboutButtonPushed(app, event)
89     web('https://github.com/datdadev/
    ↪ MATLAB-Conservative-Force-Energy-Analysis', '-browser');
90 end
```

The `AboutButtonPushed` method handles the event of the user clicking the `About` button. It opens a web page about the project on GitHub in the default browser.

## 2. Matlab script `ComputerEnergy.m`<sup>1</sup>

This MATLAB program calculates the kinetic and potential energy of a particle subject to a conservative force. It retrieves input values from the application UI `app.mlapp` and generates plots of energy over time directly within the application.

The following are the different parts of the code with detailed explanations:

- **Function declaration and input parameter:**

```
6 function ComputeEnergy(app)
7     % This function computes the energy of the system
8     ...
9     % Some other code here
72 end
```

This line declares a function called `ComputeEnergy` that takes an input parameter `app` in order to retrieve necessary parameter values for the calculation.

- **Initialization:**

```
9     k = app.kEditField.Value;
10    q = app.qEditField.Value;
11    m = app.mEditField.Value;
12    v = app.vEditField.Value;
13    t = app.tEditField.Value;
14    x = app.xEditField.Value;
```

These lines initialize the parameters and variables  $k$ ,  $q$ ,  $m$ ,  $v$ ,  $t$ , and  $x$  with the values obtained from the application UI.

---

<sup>1</sup>Source code of `ComputerEnergy.m`: <https://github.com/datdadev/MATLAB-Conservative-Force-Energy-Analysis/blob/main/ComputeEnergy.m>.

– **Reading checkbox values:**

```
18     KEVisibility = app.KECheckBox.Value;
19     PEVisibility = app.PECheckBox.Value;
```

These lines read the checkbox values from the application UI to determine if the kinetic energy and potential energy plots should be displayed.

– **Setting up the plot:**

```
22     axes = app.UIAxes;
23     cla(axes);
24     xlabel(axes, 'Time (s)');
25     ylabel(axes, 'Energy (J)');
26     grid(axes, 'on');
27     hold(axes, 'on');
```

These lines access the UIAxes object from the application UI, clear the axes, set labels for the  $x$  and  $y$  axes, display grid lines, and hold the axes on the screen while plotting new data.

– **Setting up time step variables and arrays:**

```
30     nStep = 300;
31     dt = t/(nStep-1);
35     T = 0:dt:t;
36     KE = zeros(1, nStep);
37     PE = zeros(1, nStep);
```

These lines initialize variables and arrays for the computation and visualization of kinetic and potential energy over time. `nStep` is the number of time steps to be computed, set to 300. `dt` is the time step size, calculated as `t` divided by `nStep-1` to ensure even spacing of time values in the `T` array. `T` stores time values at which energy is computed, while `KE` and `PE` are initialized as empty arrays with `nStep` elements for storing energy values for each time step. These arrays are used to plot energy over time, with the time values on the  $x$ -axis and the energy values on the  $y$ -axis, based on their visibility status from the checkboxes.



- Computing energy, force, and acceleration in a loop:

```
40     for i = 1:nStep
41         KEnergy = 0.5 * m * v ^ 2;
42         PEnergy = -0.5 * k * x ^ 2 + q * x ^ 4;
43         KE(i) = KEnergy;
44         PE(i) = PEnergy;
45         F = k * x - 4 * q * x ^ 3;
46         a = F / m;
47         v = v + a * dt;
48         x = x + v * dt;
49     end
```

These lines calculate the kinetic and potential energy values for each time step and store them in their respective arrays KE and PE. Additionally, the force and acceleration are calculated to update the velocity and position using the Euler-Cromer method.

- Checking visibility of kinetic and potential energy plots:

```
60     if KEVisibility
61         plot(axes, T, KE, 'Color', 'red', 'LineWidth', 1.5,
62             ↪ 'DisplayName', 'KE');
63     end
64
65     if PEVisibility
66         plot(axes, T, PE, 'Color', 'blue', 'LineWidth', 1.5,
67             ↪ 'DisplayName', 'PE');
68     end
```

The code checks if the visibility of kinetic and potential energy plots is set to true. If it is, the respective plots are created using the plot function.

- Drawing the plot(s) and adding legend:

```
69     drawnow;
70     legend(axes, 'Location', 'northeast');
71     hold(axes, 'off');
```

The `drawnow` function is used to display the created plot(s). The `legend` function is used to add a legend to the plot(s), and the `hold` function, which sets the hold state to off, is used to release the hold on the plot.

# IV General Information

## 1. Requirements

- MATLAB R2019b or later<sup>1</sup>
- MATLAB Symbolic Math Toolbox<sup>2</sup>

## 2. Installation

1. Visit the project's GitHub repository<sup>3</sup> and clone it to your local machine (if necessary).
2. Manually run the `app.mlapp` GUI file by clicking the **Run** button in the App Designer toolbar.
3. In the GUI window, enter the initial position, velocity, mass, and conservative force parameters.
4. Click the **Graph** button in the GUI window to compute the kinetic and potential energies of the particle and plot them on the GUI.
5. Experiment with different parameter values by adjusting the sliders and clicking the **Graph** button again.

## 3. Features

- **Enhanced User Interface:** The application provides a user-friendly interface, which eliminates the need to manually input commands into the `.m` script. The application UI streamlines the process of energy analysis, enabling users to easily select options and input data.
- **Instant Graphing:** The energy analyst application allows users to graph their data instantly, without having to regenerate new axes. This means that users can quickly visualize their data as soon as they change the parameter values and press the **Graph** button, without any delay or interruption in the graphing process.
- **Energy Visibility Controls:** The app includes checkboxes for controlling the visibility of kinetic and potential energy data. This allows users to focus on specific energy types or combinations and switch between energy visualizations for a deeper understanding of the data.

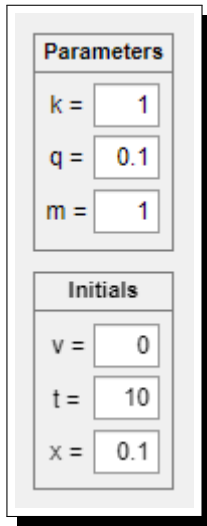
---

<sup>1</sup>MATLAB: <https://www.mathworks.com/products/matlab.html>.

<sup>2</sup>MATLAB Symbolic Math Toolbox: <https://www.mathworks.com/products/symbolic.html>.

<sup>3</sup>GitHub repository link: <https://github.com/datdadev/MATLAB-Conservative-Force-Energy-Analysis>

# V Results and Discussion



Parameters

k = 1

q = 0.1

m = 1

Initials

v = 0

t = 10

x = 0.1

Figure 1: Input variables

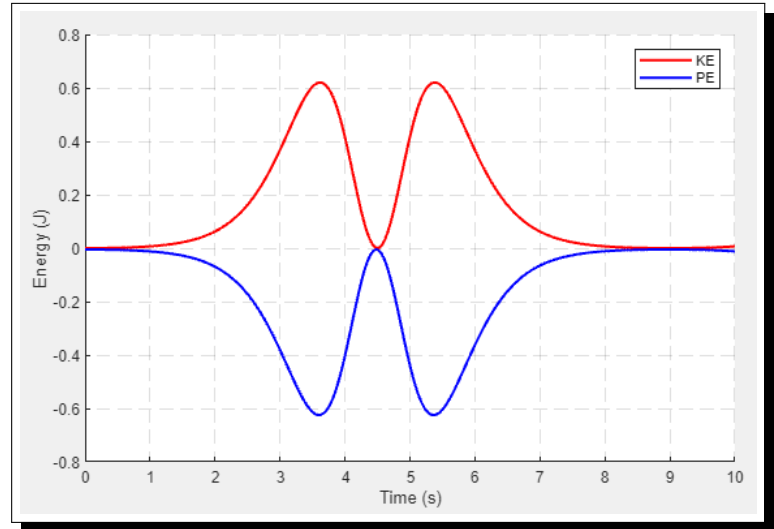


Figure 2: Output the result

With the given initial parameters, the MATLAB simulation successfully calculated the kinetic and potential energies of the particle moving under the influence of the conservative force expressed as  $F(x) = kx - 4qx^3$ . The simulation results matched the theoretical values computed manually, confirming the accuracy of the simulation. Furthermore, we can also replace appropriately many others values of quantities to study other special cases.

This is the user interface for the energy graphing simulation application:

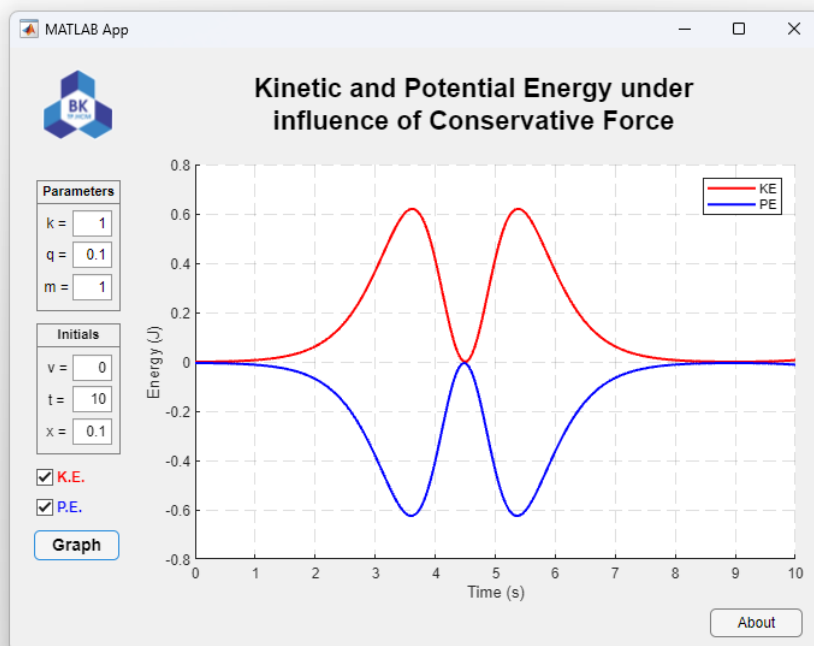


Figure 3: App-based energy graphing simulation

By replacing the other initial parameters, we can analyze the state of energy at various points during the motion of an object. For instance, let's change those values by inputting them directly into the app's input fields:  $k = 2 \text{ N/m}$ ,  $q = 0.2 \text{ N/m}^3$ ,  $m = 2 \text{ kg}$ ,  $v = 1 \text{ m/s}$ ,  $t = 10 \text{ s}$ , and  $x = 0.5 \text{ m}$ . Then, click the **Graph** button to observe the plot for the purpose of studying and analyzing it. The resulting plot should resemble the figure below:

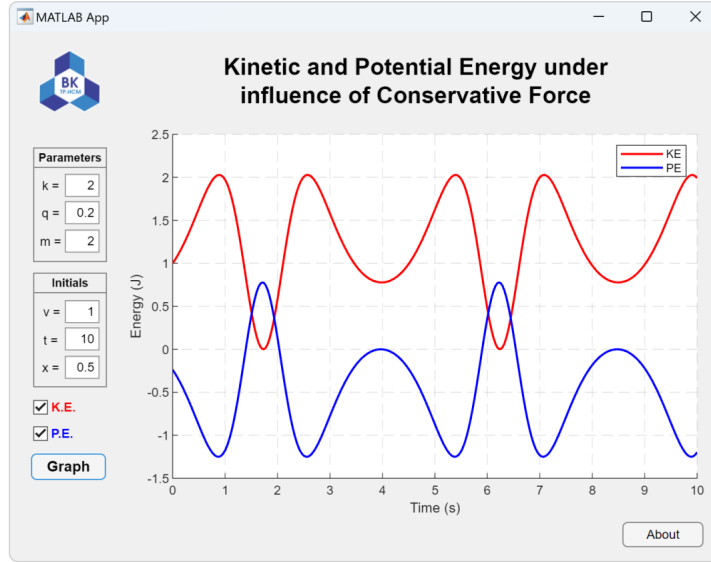


Figure 4: Another example and the result

Trying out different variables, specifically,  $k = 3 \text{ N/m}$ ,  $q = 0.35 \text{ N/m}^3$ ,  $m = 2 \text{ kg}$ ,  $v = 5 \text{ m/s}$ ,  $t = 5 \text{ s}$ , and  $x = 5 \text{ m}$ . The resulting plot may present some difficulty as two wavy plots will stack on top of each other. To address this, users can utilize the K.E. and P.E. checkboxes to control the visibility of kinetic and potential energy, facilitating analysis of the desired plot. The figures below demonstrate the different energy plot visibilities:

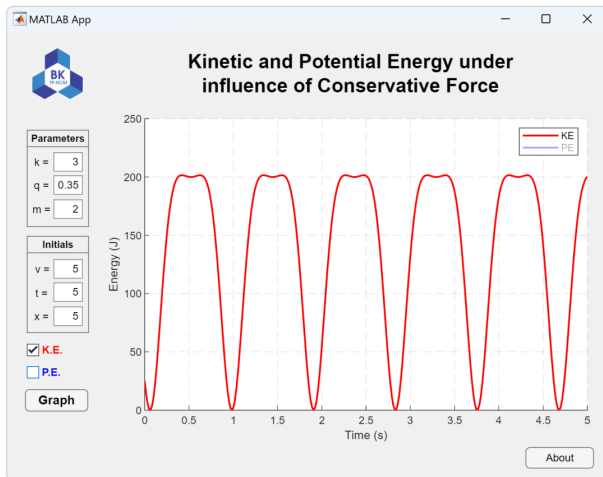


Figure 5: Example with kinetic energy visibility

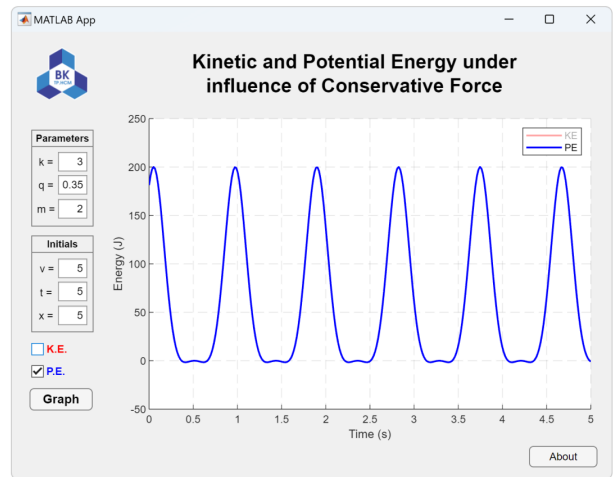


Figure 6: Example with potential energy visibility

## VI Conclusion

In conclusion, this MATLAB project demonstrated how to numerically calculate the kinetic and potential energies of a particle moving under the influence of a conservative force. Using the Euler-Cromer method, we were able to accurately compute the motion of the particle and observe that the total energy in the system, which is the sum of kinetic and potential energies, is conserved. The project also allowed us to visualize the changes in kinetic and potential energies over time using MATLAB plots. By adjusting the initial parameters, we were able to explore other special cases and observe how the system behaves under different conditions. Overall, this project provided a valuable insight into numerical methods and their application in solving physical problems.

## VII References

- [1] Alejandro Garcia. *Projects for Scientists and Engineers*. URL: <https://www.mathworks.com/matlabcentral/fileexchange/2268-projects-for-scientists-and-engineers>. (retrieved April 11, 2023).
- [2] A. L. Garcia and C. Penland. *MATLAB Projects for Scientists and Engineers*. URL: <http://www.algarcia.org/fishbane/fishbane.html>. (1996).
- [3] MathWorks. *MATLAB Documentation*. URL: <https://www.mathworks.com/help/matlab/>. (2023).
- [4] Dr. Nguyễn Xuân Thanh Trâm. *Lecture 4 - WORK and ENERGY Lecture note*. (2023).
- [5] Wikipedia. *Euler-Cromer method*. URL: [https://en.wikipedia.org/wiki/Semi-implicit\\_Euler\\_method](https://en.wikipedia.org/wiki/Semi-implicit_Euler_method). (edited: December 03, 2022).