

USING GEOGEBRA AND VPYTHON SOFTWARE FOR TEACHING MOTION IN A UNIFORM GRAVITATIONAL FIELD

DALY MARCIUC^{1,2}, CRISTINA MIRON^{1*}, E.S. BARNA¹

¹University of Bucharest, Faculty of Physics, Bucharest-Măgurele, Romania

²“Mihai Eminescu” National College of Satu Mare, Satu Mare, Romania

*Corresponding author: cmiron_2001@yahoo.com

Received July 6, 2015

Abstract. This article presents an interdisciplinary approach of teaching at high-school level some topics related to the movement in a uniform gravitational field. Our approach is based on the methodology of learning Physics by modelling. We create the computer models of the studied phenomena, using free but very versatile software, like GeoGebra for analytical modelling, respectively the VPython programming language, for numerical modelling. The built models are related to themes with a major role in the history of Physics: projectile motion and gravitational pendulum. Firstly, we construct and interpret the mathematical models for the studied systems, and then based on these we achieve the models on the computer. The equation of the safety parabola for the motion of a projectile is inferred and is plotted, both in the case when only the gravitational force is acting on the projectile, and also in the case of an additional horizontal force. An application of the model of the gravitational pendulum is the construction of the mathematical and computer models of the transversal wave generated by a set of uncoupled pendulums of different lengths. In GeoGebra we realized a two-dimensional model of this wave, and by using the VPython programming language we built a three-dimensional model of the system, based on the method of numerical modelling. The activities described in this article are the result of a research project of the authors in the field of Physics Education and this result was applied to lessons of students of the interdisciplinary study group focused on mathematical and computational modelling in County Centre of Excellence of Satu Mare.

Key words: Physics Education, computational models, projectile motion, gravitational pendulum, interdisciplinary learning, GeoGebra, VPython.

1. INTRODUCTION

Modeling based learning is a relatively new approach in Physics education, which is proving effective. D. Hestenes founded this innovative method in teaching Physics, proposing the organization of learning contents and activities around the

study of some fundamental physical models. Thus, learning is no longer focused on solving problems, but on the construction and use of models. In fact, the construction of mathematical models includes solving problems but the applicability of the obtained models is broader: one and the same model can be used to solve a wide range of problems [1].

The affordable technologies of nowadays allow interweaving mathematical modeling activities with the computer modeling activities. Mathematical softwares, like Mathematica and GeoGebra, allow dynamic graphical representations and support in-depth understanding of abstract mathematical concepts [2–3]. The VPython programming language is increasingly used in Physics education, especially for numerical modeling of physical phenomena [4–5]. In this article we present some ways in which high school students can implement on the computer two significant mathematical models used in Physics: the projectile and the gravitational pendulum.

2. COMPUTER MODELS FOR PROJECTILE MOTION

2.1. AN INTERACTIVE MODEL IN GEOGEBRA FOR THE SIMPLE PROJECTILE

Involving students in building models supports the understanding of how complex physical phenomena can be studied using simple models. We begin with the construction of a model for the projectile which is acted only by the gravitational force. We consider a projectile thrown with an initial velocity v , forming the angle α with the horizontal. Choosing a reference system with the origin at the point of launch, the coordinates of the projectile at time t are:

$$\begin{cases} x = v \cos \alpha \cdot t \\ y = v \sin \alpha \cdot t - \frac{1}{2} g t^2 \end{cases} \quad (1)$$

With the GeoGebra software we can visualise the projectile's motion resulting from the composition of the uniform motion with constant velocity $v \cos \alpha$ along the x -axis, and uniformly accelerated motion with acceleration g and initial velocity $v \sin \alpha$, in the direction of the y -axis. From the condition $y > 0$ we obtain the range of time t until landing:

$$t \in \left[0, \frac{2v \sin \alpha}{g} \right]. \quad (2)$$

Based on relations (1) and (2) we build the model of the projectile, by typing in the input bar of GeoGebra the command presented in Table 1.

Table 1

Commands for building a projectile model in GeoGebra

Command	Result
$v = 2; \alpha = 30^\circ; t = 0; g = 9.8$	Defining variables
$M_1 = (v \cos(\alpha) t, 0)$	Projectile projection on x axis
$M_2 = (0, v \sin(\alpha) t - 1/2 g t^2)$	Projectile projection on y axis
$M = (v \cos(\alpha) t, v \sin(\alpha) t - 1/2 g t^2)$	The position of the projectile at time t
$T_{\{fin\}} = 2 v \sin(\alpha)/g$	Defining landing time

In the *Properties* dialog box associated with the variable t , we set its extreme values to 0, respectively T_{fin} , as defined by condition (2). By starting the animation of the variable t is simulated the movement of the projectile, represented by the point M , and the movement of its projections on the axes, represented by the points M_1 and M_2 . The values of the parameters v and α can be adjusted using the sliders that are displayed on the graphical Panel (Fig. 1).

GeoGebra allows drawing traces of the points in motion, option that helps us highlighting the types of motion of the projectile's projections along the two axes. Students can see that the projection of the projectile on x -axis runs equal distances in equal periods of time. At the same time, the projection of the projectile on y -axis runs on smaller and smaller distances in equal periods of time, until achieving the maximum height. After this moment, the projection of the projectile on y -axis runs on bigger and bigger distances in equal periods of time. In this way, the students can visualise the composition of the two types of motion along the axes, uniform and respectively uniformly accelerated, resulting the projectile's movement on a parabola (Fig. 1).

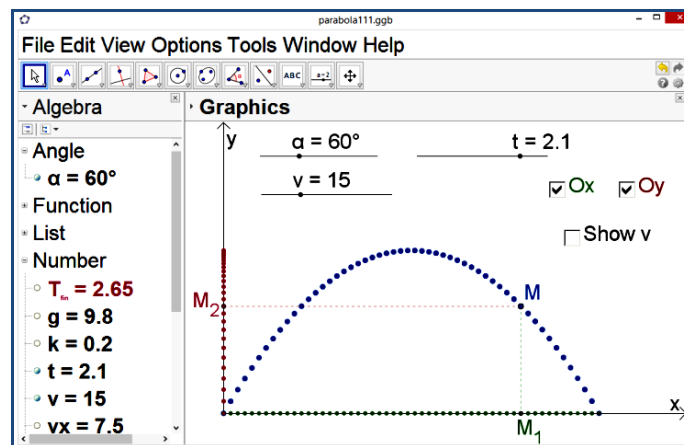


Fig. 1 – Decomposition of projectile motion.

The colored versions can be accessed at <http://www.infim.ro/rfp/>.

By eliminating the variable t between the relationships (1), we find the equation of the projectile trajectory:

$$y = -\frac{g}{2v^2}(1 + \tan^2(\alpha))x^2 + \tan(\alpha)x. \quad (3)$$

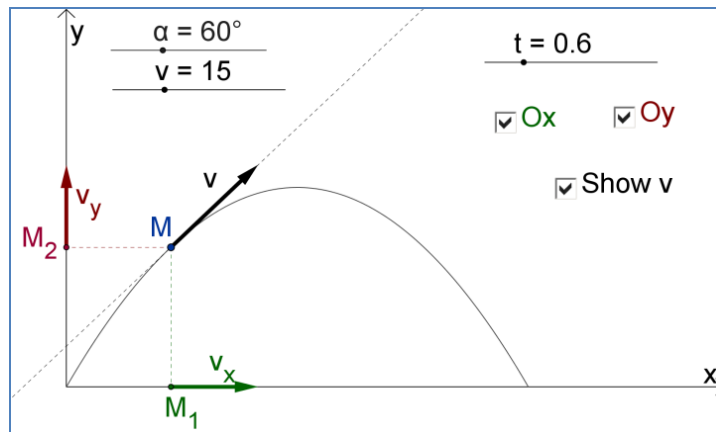


Fig. 2 – The velocity of the projectile and its components with GeoGebra.
The colored versions can be accessed at <http://www.infim.ro/rtp/>.

To draw in GeoGebra the projectile trajectory we can use the parametric equations (1) or we can define a function, according to the equation (3). To represent velocity and its components on axes, we use the following relations:

$$\begin{cases} v_x = v \cos \alpha \\ v_y = v \sin \alpha - gt \end{cases} \quad (4)$$

Table 2 presents the main commands typed in GeoGebra to obtain the dynamic representation captured in Fig. 2.

Table 2

The GeoGebra commands used for a dynamical representation of projectile velocity

Command	Result
$v_x = v \cos(\alpha);$ $v_y = v \sin(\alpha) - gt$	Velocity components according to relation (4)
$u_x = (v_x, 0); u_y = (0, v_y);$ $u = (v_x, v_y)$	Representing the velocity and its projections on axes
$c = \text{Curve}[v \cos(\alpha) t, v \sin(\alpha) t - 1/2 g t^2, t, 0, T_{\text{fin}}]$	Representation of trajectory through parametric equations, according to relations (1) and (2)
$f(x) = (-1) / 2 g / v^2 (1 + \tan(\alpha)^2) x^2 + \tan(\alpha) x$	Define trajectory based on relationship (3)

Changing the angle α by using the associated cursor allows us to visualise various possible trajectories of the projectile (Fig. 3). To determine which positions can be attained by the projectile for a fixed speed v when the α angle is changing, we rewrite equation (3) as an equation of second degree in $\tan(\alpha)$. By imposing the condition that the discriminant of this equation must be positive we obtain:

$$y \leq -\frac{g}{2v^2} x^2 + \frac{v^2}{2g}. \quad (5)$$

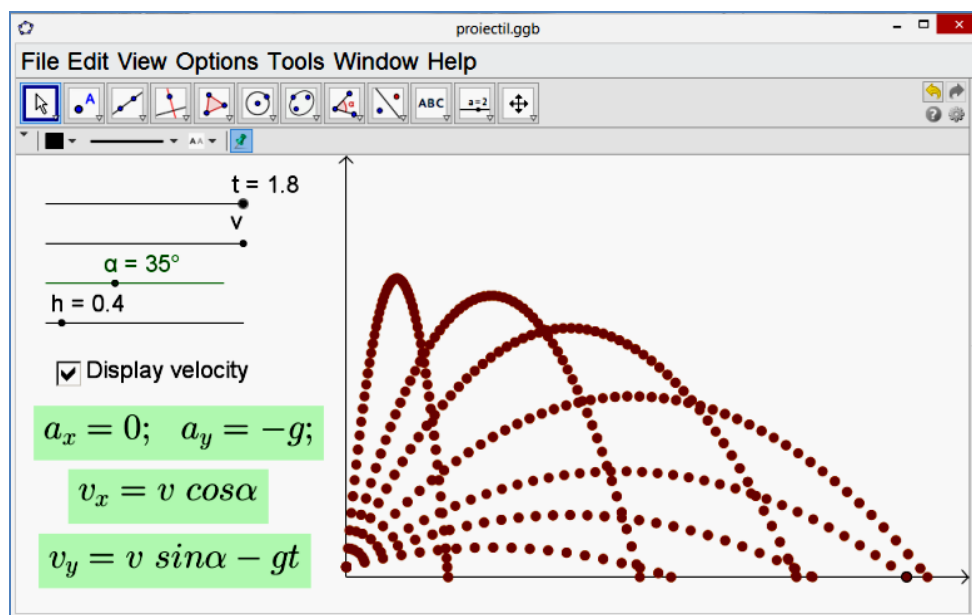


Fig. 3 – Trajectories obtained by modification of the launch angle.

The colored versions can be accessed at
<http://www.infim.ro/rfp/>.

Condition (5) allows defining the safety parabola for a given value of the projectile speed. The consistency between relations (1) and (5) was verified by plotting a class of trajectories, exploiting the facility of GeoGebra to work with lists of geometric objects. With a single command, $Sequence[Curve[v \cos(q) t, v \sin(q) t - g t^2 / 2, t, 0, 2v \sin(q) / g], q, 1^\circ, 90^\circ, 5^\circ]$, we display all the trajectories seen in Fig. 4. By representing the function given by the right side of relationship (5) we can view that this curve represents the envelope of the possible trajectories (Fig. 4).

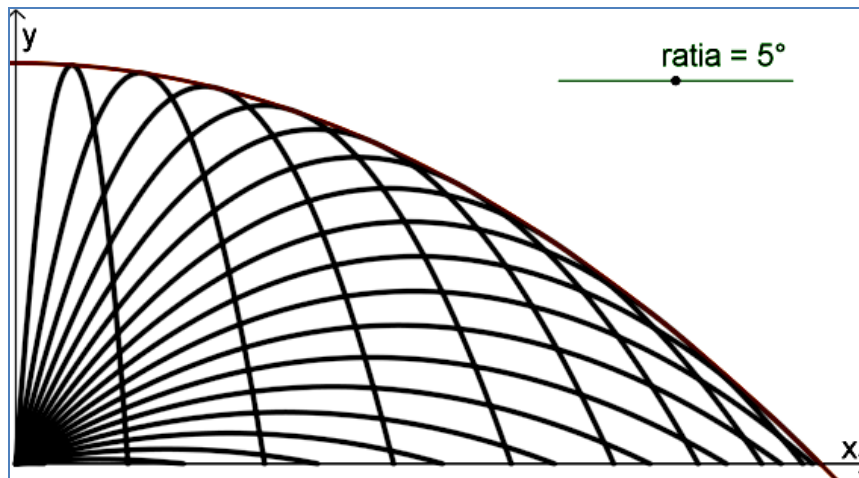


Fig. 4 – Trajectories represented in GeoGebra and their envelope.
The colored versions can be accessed at <http://www.infim.ro/rfp/>.

2.2. THE PROJECTILE'S MOTION UNDER THE ACTION OF AN ADDITIONAL HORIZONTAL FORCE

The next model refers to the motion of a projectile upon which acts a constant force \vec{F} parallel to the ground, in addition to gravity \vec{G} . Denoting by k the ratio between the magnitudes of the two forces, so that $|\vec{F}| = k|\vec{G}|$, the motion equations of the projectile are:

$$\begin{cases} x = v \cos \alpha \cdot t - \frac{1}{2} k g t^2 \\ y = v \sin \alpha \cdot t - \frac{1}{2} g t^2 \end{cases} \quad (6)$$

A first sketch of the trajectory is obtained after comparing the values of the variable t in three particular situations: at landing, at reaching the x maximum and at reaching the maximum height y . From relations (6) we deduce:

$$\begin{cases} t_{xmax} = \frac{v \cos \alpha}{kg} \\ t_{ymax} = \frac{v \sin \alpha}{g} \end{cases} \quad (7)$$

By imposing the condition $y = 0$, from the relation (6) we deduce the value of time t at landing:

$$t_{fin} = \frac{2v \sin \alpha}{g} \quad (8)$$

Since it is obvious that $t_{ymax} < t_{fin}$, three situations are possible:

- a) $t_{y\max} < t_{fin} < t_{x\max} \Leftrightarrow k \tan(\alpha) < \frac{1}{2}$,
 b) $t_{y\max} < t_{x\max} < t_{fin} \Leftrightarrow \frac{1}{2} < k \tan(\alpha) < 1$,
 c) $t_{x\max} < t_{y\max} < t_{fin} \Leftrightarrow k \tan(\alpha) > 1$.

In Fig. 5 the projectile trajectories are sketched in the three cases.

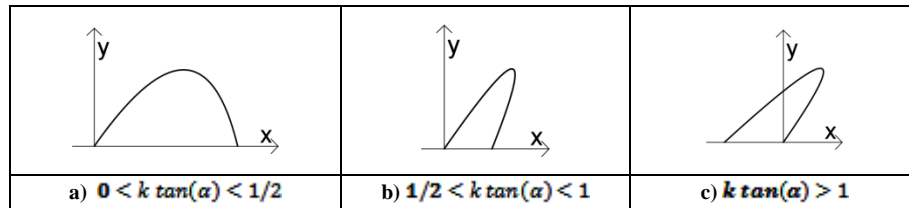


Fig. 5 – Three categories of possible trajectories.

By eliminating the variable t from equations (6) we find the trajectory equation:

$$(k \tan(\alpha) - 1)(x \tan(\alpha) - y) = -\frac{g}{2v^2} (1 + \tan^2(\alpha))(x - ky)^2. \quad (9)$$

By calculating the invariants of this conic, we can verify that equation (9) represents a parabola. We distinguish two limit cases, illustrated by Fig. 6: the case when the trajectory is straight and the projectile comes back to the launch point (Fig. 6a)), and the case when the tangent to the parabola at the point of landing is vertical (Fig. 6b)).

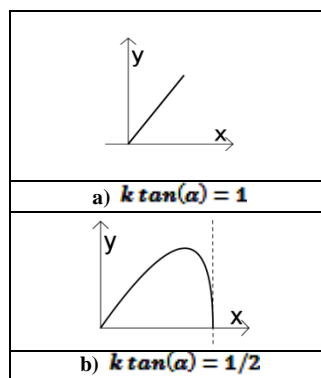


Fig. 6 – Limit cases of the trajectories.

Actually, we can justify based on physical considerations that the trajectory of the projectile is in all cases a parabola whose axis of symmetry forms the angle

$r = \arctan(k)$ with the y -axis. Indeed, the resultant force $\vec{F} = -mg\vec{j} - kmg\vec{i}$ forms with the y -axis the angle r and has the magnitude $g\sqrt{1+k^2}$. Therefore, in the system of axes $x'Oy'$ resulted by the rotation with angle $-r$ of the original system xOy (Fig. 7), the trajectory equation is obtained from equation (3), by replacing the acceleration g with $g\sqrt{1+k^2}$, and by replacing the launch angle α with $\alpha + r$:

$$y' = -\frac{g\sqrt{1+k^2}}{2v^2}(1 + \tan^2(\alpha + r))x'^2 + \tan(\alpha + r)x'. \quad (10)$$

Starting from equation (10) we will find again equation (9), by applying the rotation of centre O and angle r , *i.e.* through the transformation:

$$\begin{cases} x' = x \cos r - y \sin r \\ y' = x \sin r + y \cos r \end{cases} \Leftrightarrow \begin{cases} x' = \frac{x - ky}{\sqrt{1+k^2}} \\ y' = \frac{xk + y}{\sqrt{1+k^2}} \end{cases}. \quad (11)$$

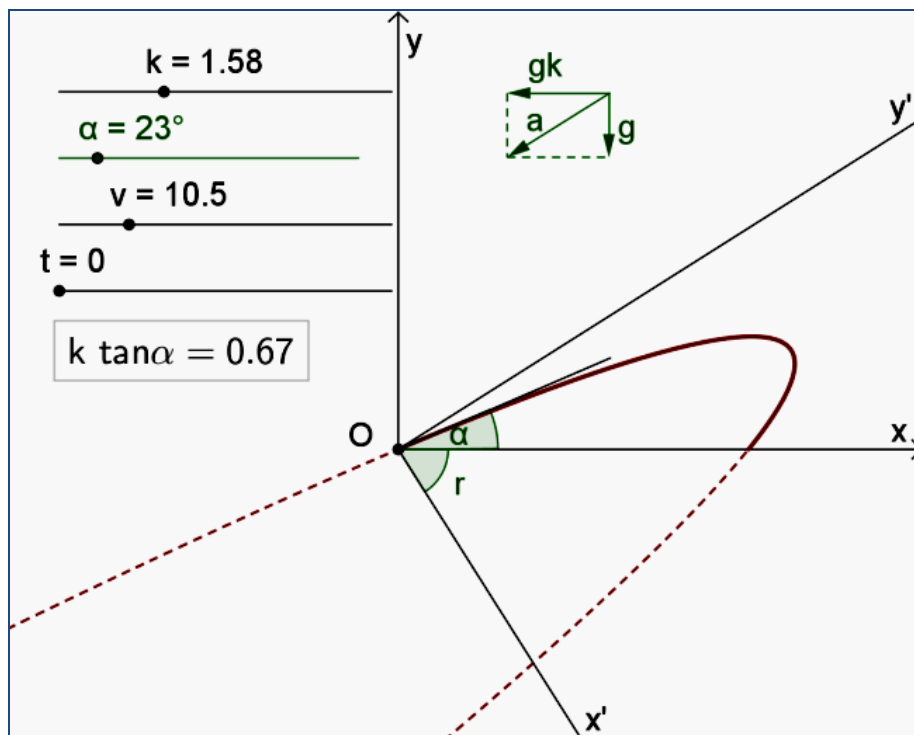


Fig. 7 – Changing the reference system.

The colored versions can be accessed at <http://www.infim.ro/trp/>.

By applying transformation (11) to condition (5), we deduce relation (12) that corresponds to the equation of the parabola safety.

$$xk + y \leq -\frac{g}{2v^2}(x - ky)^2 + \frac{v^2}{2g}. \quad (12)$$

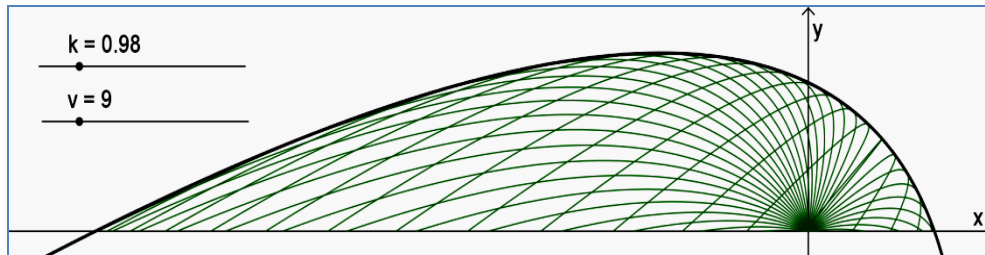


Fig. 8 – The envelope of the projectile trajectories.
The colored versions can be accessed at <http://www.infim.ro/trp/>.

In Fig. 8 we captured the GeoGebra representation of the safety parabola together with a set of possible trajectories.

2.3. NUMERICAL MODELING OF THE PROJECTILE'S MOTION USING VPYTHON PROGRAMMING

Numerical modeling of the projectile's motion, achieved using the VPython programming language, is based on the cyclical determination of the position and speed of the object at a given moment, knowing both the position and velocity at a previous time and also the forces acting on the object. Thus, if at time t the position vector is $\mathbf{r} = (x, y, z)$, and the velocity is $\mathbf{v} = (v_x, v_y, v_z)$, at a later time $t + \Delta t$, the position vector \mathbf{r}' and the velocity \mathbf{v}' are evaluated based on the relations:

$$\begin{cases} \mathbf{r}' = \mathbf{r} + \mathbf{v} \cdot \Delta t \\ \mathbf{v}' = \mathbf{v} + \mathbf{a} \cdot \Delta t \end{cases} \quad (13)$$

where \mathbf{a} is the acceleration vector, obtained by dividing the resultant force by the mass of the object.

The VPython program presented below simulates the motion of the projectile in six distinct cases, applying iteratively the relations (13).

Figure 9 captures the landing of the projectile in our simulations, the velocities being represented by arrows. The last two lines of the program trigger the display, in the VPython console, of landing speed of the projectile in the six simulated cases, that is for $k = 0$, $k = 0.2$, $k = 0.5$, $k = 0.7$, $k = 1$ and $k = 1.2$. In all these cases, the launch angle is 45° , and the components of launch velocity are $(15, 15, 0)$. We mention that VPython works by default with the international system of units, thus the speeds are expressed in m/s.

```

from visual import *
dt = 0.01; g = 9.8; vscale = 0.3; k = [0,0.2,0.5,0.7,1,1.2]
d = [ ]
for i in range(6):
    d.append(display(x=600*(i//3), y=30+200*(i % 3 ), width=600, height=200,
                    center=(0,7,0), background=(0,0,0), title="k="+str(k[i])))
floors=[ ];balls=[ ];trails=[ ];varr=[ ]
for i in range(6):
    floors.append(box(pos=(1,0,0), length=70, height=0.5, width=20, color=color.blue,
                      display=d[i]))
    balls.append(sphere(pos=(-15,1,0), color=color.red,display=d[i],a=vector(-k[i]*g,-
                                         g,0)))

    trails.append(curve(display=d[i]))
for ball in balls:
    ball.velocity = vector(15,15,0)
for i in range(6):
    varr.append(arrow(pos=balls[i].pos, axis=vscale*balls[i].velocity,
                     display=d[i],color=color.yellow))
while balls[0].y > 0.9:
    rate(100)
    for i in range(6):
        balls[i].pos = balls[i].pos + balls[i].velocity*dt
        balls[i].velocity = balls[i].velocity + balls[i].a*dt
        varr[i].pos = balls[i].pos
        varr[i].axis = vscale*balls[i].velocity
        trails[i].append(pos=balls[i].pos)
for i in range(6):
    print(i, mag(balls[i].velocity))

```

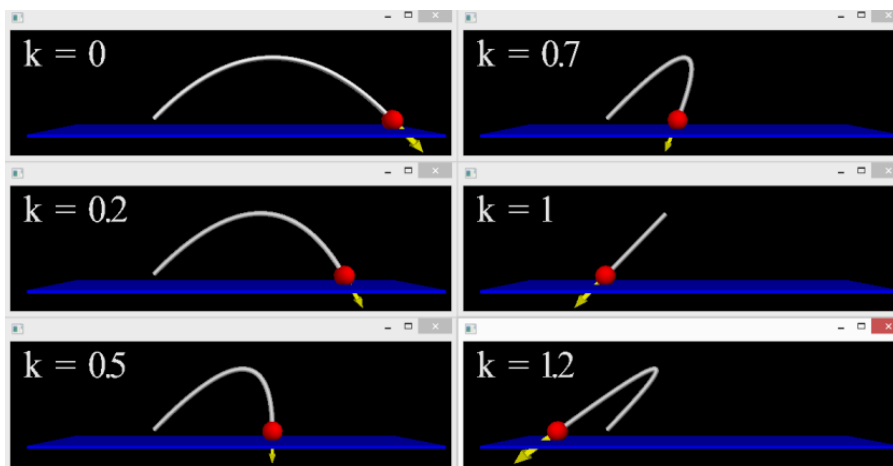


Fig. 9 – Comparative models using VPython.
 The colored versions can be accessed at <http://www.infim.ro/rtp/>.

3. COMPUTER MODELS FOR THE PENDULUM MOTION

Many physical phenomena can be described by analogy with the pendulum motion, which constitutes the paradigm of the linear oscillator [6]. The construction of computer models of the pendulum allows students to test different hypotheses, to visualise the system evolution for different values of the parameters, favoring a deep understanding of the used physical and mathematical concepts.

3.1. MODELING THE PENDULUM WITH GEOGEBRA

The equation of the motion of a pendulum with length l is a non linear second order differential equation:

$$l\ddot{\varphi} = -g\sin(\varphi), \quad (14)$$

where φ is the angle formed by the pendulum with the vertical. To model the pendulum with GeoGebra we consider the approximation $\sin(\varphi) \cong \varphi$ convenient for small values of the angle φ . The solution of the resulted linear differential equation is:

$$\varphi = \varphi_{max}\cos(\omega t), \quad (15)$$

where φ_{max} is the amplitude of the oscillation and ω is the angular speed, given by

$$\omega = \sqrt{\frac{g}{l}}. \quad (16)$$

Table 3 lists the main commands for building the GeoGebra simulation.

Table 3

GeoGebra commands for building the pendulum model

Command	Result
$m = 2; t = 0; l = 6; g = 9.8; \varphi_{max} = 0.1$	Defining the system variables
$\omega = \text{sqrt}(g/l)$	Defining the angular speed
$\varphi = \varphi_{max}\cos(\omega t)$	Defining the deviation angle at time t
$M(l \sin \varphi, l \cos \varphi)$	The position of the pendulum at time t
$\text{period} = 2\pi/\omega$	Defining the period of the oscillation

The simulation captured in Fig. 10 allows changing the length of the pendulum and the maximum angle of deviation φ_{max} using the displayed sliders. Our application allows the representation of the forces acting on the body and the representation of the velocity at every moment. The animation is triggered using variable t , for which we set in the *Properties* dialog box the extreme values to 0 and respectively, *period*, and the animation style was set to be *increasing*.

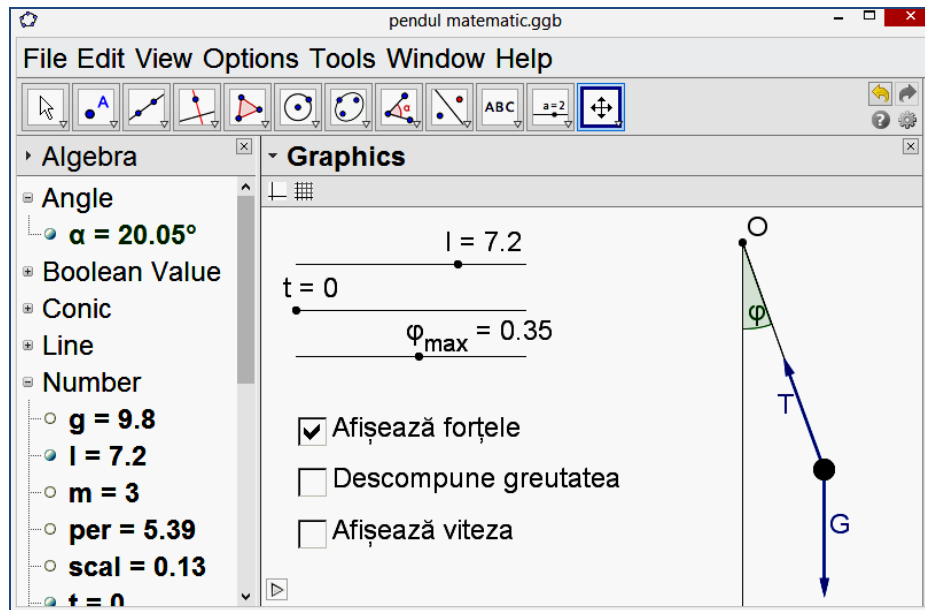


Fig. 10 – A model of the pendulum with GeoGebra.
The colored versions can be accessed at <http://www.infim.ro/rfp/>.

3.2. THE VELOCITY AND ACCELERATION HODOGRAPH OF THE PENDULUM

For an exact representation of the velocity and acceleration in the case of large amplitudes, we determine the pendulum's speed from the energy conservation law. If v_0 is the maximum speed of the pendulum, achieved when it runs through the position of equilibrium, then at time t , when the deviation angle is φ , the speed is given by the relation

$$v^2 = v_0^2 - 2gl(1 - \cos\varphi). \quad (17)$$

We also know tangential and radial components of the acceleration:

$$\begin{cases} a_r = \frac{v^2}{l} \\ a_t = g \sin\varphi \end{cases}. \quad (18)$$

Applying again the energy conservation law we infer the amplitude of the oscillation:

$$\varphi_{max} = \arccos\left(1 - \frac{v_0^2}{2gl}\right). \quad (19)$$

Based on these relationships, the new model is constructed by entering the GeoGebra commands shown by Table 4.

Table 4

The GeoGebra commands used for the representation of pendulum velocity and acceleration

Command	Result
$t = 0; l = 3; g = 9.8;$ $v_0 = 3$	Defining the system parameters
$\omega = \sqrt{g/l}; \text{per} =$ $2\pi/\omega$	Defining the angular speed and period
$f = v_0^2/(g l);$ $\varphi_{\max} = \arccos(1-f/2)$	Defining the amplitude based on relation (19)
$\varphi = \varphi_{\max} \cos(\omega t)$	Defining the deviation angle at time t
$x_M = l \sin \varphi;$ $y_M = l \cos \varphi$	The position of the pendulum at time t
$v = \text{If}[t < \text{per}/2,$ $v_0^2 - 2gl(1 - \cos(\varphi)),$ $-v_0^2 + 2gl(1 -$ $\cos(\varphi))]$	Defining the speed according to the relation (17) and considering the movement sense
velocity = $(-v \cos \varphi,$ $-v \sin \varphi)$	Defining the velocity at the time t
$ar = v^2/l, at = g \sin \varphi$	Defining the acceleration components
$a = (-ar \sin \varphi$ $- at \cos \varphi,$ $ar \cos \varphi - at \sin \varphi)$	Defining the acceleration vector at the time t

Using the GeoGebra facility to mark the traces of the geometric objects in motion, we can visualise the hodograph of the velocity and of the acceleration [7]. In Fig. 11 the velocities at every point of the trajectory are represented and the hodograph of velocity is traced for half of the period of oscillation. For the other half of the period, the hodograph is obtained by symmetry relative to the y -axis.

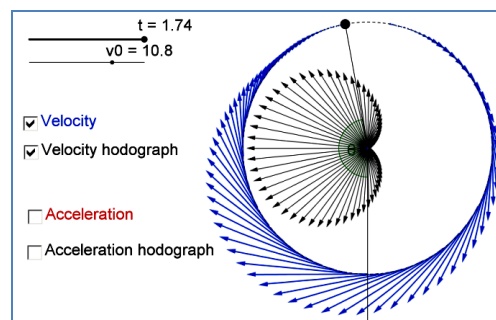


Fig. 11 – The hodograph of velocity.

The colored versions can be accessed at <http://www.infim.ro/rp/>.

The mathematical form of the velocity components shows that the velocity hodograph is a Pascal limaçon curve. If the equality $v_0^2 = 4gl$ is hold, *i.e.* the maximum deviation angle is π , this curve is even a cardioid:

$$\begin{cases} v_x = -v \cos \varphi = (-v_0^2 + 2gl) \cos \varphi - 2gl \cos^2 \varphi \\ v_y = -v \sin \varphi = (-v_0^2 + 2gl) \sin \varphi - 2gl \cos \varphi \sin \varphi \end{cases} \quad (20)$$

The acceleration hodograph is also a Pascal limaçon, which become a cardioid when $v_0^2 = 5gl$ (Fig. 12). If we represent the acceleration in each point of the trajectory, the extremity of this vector describes also a Pascal limaçon, or even a cardioid if a convenient scaling factor is chosen [8]: $s = \frac{r}{f+1}$, where $f = \frac{v_0^2}{gr}$.

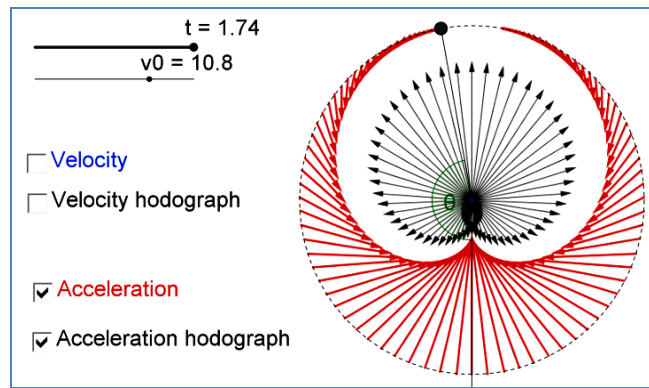


Fig. 12 – The acceleration hodograph.

The colored versions can be accessed at <http://www.infim.ro/rfp/>.

3.3. THE SIMULATION OF THE PENDULUM'S MOTION WITH VPYTHON

The differential equation (14) describing the pendulum's motion is equivalent to the following two relations:

$$\begin{cases} \dot{\varphi} = \theta \\ \dot{\theta} = -\frac{g}{l} \sin(\varphi) \end{cases} \quad (21)$$

where θ is the angular speed. If dt is a very small interval of time in which the angular velocity varies very little, then relations (21) lead to

$$\begin{cases} \varphi_f = \varphi_i + \theta_i \cdot dt \\ \theta_f = \theta_i - \frac{g}{l} \sin(\varphi_f) \cdot dt \end{cases} \quad (22)$$

where φ_i, θ_i are the deviation and the angular speed at time t , and φ_f, θ_f are the deviation and the angular speed at time $t+dt$. Using iteratively relations (22), the VPython program presented below simulates the movement of the pendulum.

```

from visual import *
dt = 0.01
g=9.8
phi=0.3
l=0.7
teta=0
display(x = 600, y = 100 , width = 300, height = 400,
        center=(0,-0.6*1,0), background=(0,0,0),title="Pendulum")
ceil = box(length=1, height=0.01, width=0.5, color=color.blue)
ball = sphere(pos=(1 *sin(phi),-1*cos(phi),0),
              color=color.red,radius=0.05)
rod = cylinder(pos=(0,0,0),axis=(ball.pos.x,ball.pos.y,0), radius=0.005)
while 1:
    rate(100)
    phi=phi+teta*dt
    teta=teta-g/l *sin(phi)*dt
    ball.pos = (1 *sin(phi),-1*cos(phi),0)
    rod.axis = (ball.pos.x,ball.pos.y,0)

```

3.4. SIMULATION OF PENDULUM WAVE

An interesting application of the pendulum model is the construction of a series of k pendulums of different lengths, so that the numbers of oscillations performed by them in a given interval T form an arithmetic progression with the ratio 1. If the pendulum i performs $n + i$ oscillations in the interval T , then its length is

$$l_i = \frac{gT^2}{(n+i)^2}, i \in \{1, 2, \dots, k\}. \quad (23)$$

The GeoGebra simulation was achieved using the commands listed in Table 5.

Table 5

GeoGebra commands used for a pendulum wave simulation	
Command	Result
$T = 60; n = 8; k = 18; g = 9.8; t = 0; \varphi_0 = 0.4$	Defining the system parameters
$d = 0.1$	Distance between two pendulums
$c = g T^2 / (4\pi^2)$	Defining the system constant
$L = \text{Sequence}[c / (n + i)^2, i, 1, k]$	Calculating the pendulums' lengths
$F = \text{Sequence}[\sqrt{g / \text{Element}[L, i]}, i, 1, k]$	Calculation the angular frequencies
$\text{balls} = \text{Sequence}[\text{Element}[L, i] * \sin(2 * \varphi_0 * \cos(t * \text{Element}[F, i])), d * i], i, 1, k]$	Calculation the pendulums' positions at a time t
$\text{rods} = \text{Sequence}[\text{Segment}[\text{Element}[\text{balls}, i + 1], (0, d i)], i, 0, k]$	Representing the pendulums' rods

The sliders included in application's graphic panel shown in Fig. 13 allow easy modification of the total period T , of the amplitude φ_0 , of the number k of pendulums, and of the number $(n + i)$ of oscillations made by each pendulum during the period T . The motion of the pendulum in our simulation is triggered by turning on the animation of the time variable t , in *increasing* style in range $[0, T]$.

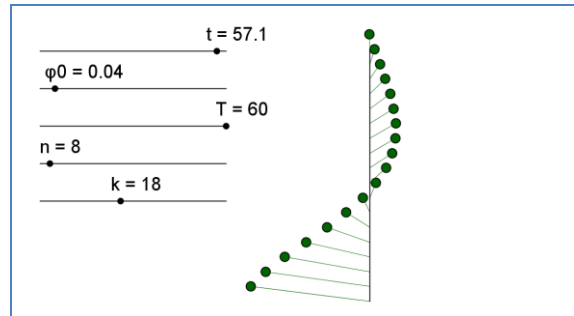


Fig. 13 – Simulation of pendulum wave with GeoGebra.
The colored versions can be accessed at <http://www.infim.ro/trp/>.

A three-dimensional simulation of the pendulum wave was achieved through numerical modeling by the following VPython program:

```

from visual import *
scene=display(x = 60, y = 30 , width=500, height=500,
             center=(0.2,0,0), background=(0,0,0))
dt = 0.01; g=9.8; pi=3.14;per=60;k=15;n=45;dist=0.05
teta=[];anglspeed=[];l=[];balls=[];rods=[]
bar=cylinder(pos=(0,0,0),axis=(0,0,dist*(k-1)), radius=0.005,color=color.blue)
for i in range (k):
    l.append(g*per*per/(4*pi*pi*(n+i)*(n+i)))
    teta.append(0.3)
    anglspeed.append(0)
    balls.append(sphere(pos=(l[i] *cos(teta[i]-pi/2),l[i]*sin(teta[i]-pi/2),-i*dist),
                        color=color.red,radius=0.02))
    rods.append(cylinder(pos=(0,0,i*dist),
                        axis=(l[i] *cos(teta[i]-pi/2),l[i]*sin(teta[i]-pi/2),0),
                        radius=0.003))

while 1:
    rate(100)
    for i in range (k):
        teta[i]=teta[i]+anglspeed[i]*dt
        anglspeed[i]=anglspeed[i]-g/l[i] *sin(teta[i])*dt
        balls[i].pos = (l[i] *cos(teta[i]-pi/2),l[i]*sin(teta[i]-pi/2),i*dist)
        rods[i].axis = (balls[i].pos.x,balls[i].pos.y,0)

```


The graphical scene provided by programming with VPython offers the possibility of a three-dimensional visualization of the objects. The angle under which the objects are viewed can be changed by acting simultaneously the *Ctrl* key and the mouse, and the *zoom-in* and *zoom-out* functions are provided by acting simultaneously the *Alt* key and the mouse. The values of the parameters can be modified by program in the initialization variables section. The modeling was achieved based on relationships (22), applied to each pendulum of the system.

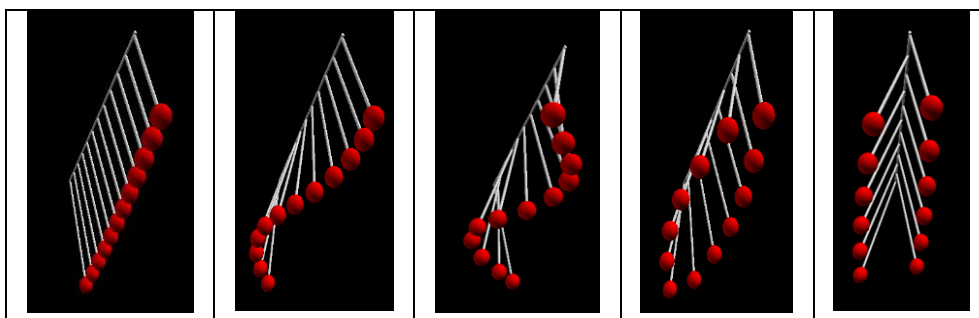


Fig. 14 – Simulation of pendulum wave with VPython.
The colored versions can be accessed at <http://www.infim.ro/rfp/>.

4. CONCLUSIONS

This article presents some ways of using two affordable informatic tools for modeling activities with students: the mathematical software GeoGebra and the programming language VPython. Each of these presents specific advantages. Models are created easily with GeoGebra, based on the analytical equations. Modeling with GeoGebra fosters the development of students' mathematical competencies. The simultaneous displaying of algebraic and graphic panels of the GeoGebra application allows clear visualization of the effect of changing the equations of the model upon the computer simulation. On the other hand, the VPython programming allows numerical modeling, based on the differential equations of the mathematical model, so that the computational and programming skills of the students are developed. In both cases, the students have the opportunity to understand in depth the studied physical phenomena, by emphasizing the connections between the mathematical model and the corresponding physical phenomenon.

In this article we focused on two physical models that played a major role in the history of Newtonian mechanics: the projectile and the pendulum. The proposed approach is an alternative of teaching these phenomena using computers, in addition to others already known [9–11]. Building computer models helps students to understand some abstract concepts, like force, speed or acceleration,

and optimizes students' motivation for learning [12–13]. The students are involved into complex activities, similar to those encountered in scientific research [14–15]. The interaction between the students and the computer models that they build favours deepening students' knowledge and helps them discover the physical significances of the used abstract mathematical equations.

REFERENCES

1. D. Hestenes, *Modeling methodology for physics teachers*, AIP Conference Proceedings, IOP Institute of Physics Publishing LTD, 1997, 935–958.
2. J. Hohenwarter, M. Hohenwarter, *Journal of Computers in Mathematics and Science Teaching* **28**, 2, 135–146 (2009).
3. M. Aktumen, M. Bulut, *Anthropologist* **16**, 1–2, 167–176 (2013).
4. R. Chabay, B. Sherwood, *Am. J. Phys.* **76**, 4, 307–313 (2008).
5. M.D. Caballero, M.A. Kohlmyer, M.F. Schatz, *Phys. Rev. ST Phys. Educ. Res.* **8**, 2, 020106 (2012).
6. R. Newburgh, *The pendulum: A paradigm for the linear oscillator*, *Science & Education* **13**, 4–5, 297–307 (2004).
7. D. Marciuc, C. Miron, *Technology Integration of Geogebra Software in Interdisciplinary Teaching*, Proceedings of the 10th International Scientific Conference *eLearning and Software for Education* (Bucharest), **3**, 280–287 (2014).
8. M. Lieberherr, *The Physics Teacher* **49**, 9, 576–577 (2011).
9. C.M. Ezrailson, G.D. Allen, C. Loving, *Sci. & Educ.* **13**, 437–457 (2004).
10. M. Fowler, *Sci. & Educ.* **13**, 791–796 (2004).
11. A. Jimoyiannis, V. Komis, *Comput. Educ.* **36**, 2, 183–204 (2001).
12. L. Dinescu, C. Miron, E.S. Barna, *Rom. Rep. Phys.* **63**, 2, 557–566 (2011).
13. C. Kuncser, A. Kuncser, G. Maftai, S. Antohe, *Rom. Rep. Phys.* **64**, 4, 1119–1130 (2012).
14. C. Miron, I. Staicu, *Rom. Rep. Phys.* **62**, 4, 906–917 (2010).
15. J.K. Gilbert, *Int. J. Sci. Math. Educ.* **2**, 115–130 (2004).