http://dx.doi.org/10.21172/ijiet.94.20

A Visual Analysis of the 2D Projectile Motion by Computable Document Files

C. Deniz[1*], Y. Cerci[2]

[1]*Department of Electrical and Electronics Engineering, Communication Engineering, Faculty of Engineering, Adnan Menderes University, Aydin, Turkey*

**Corrosponding Author*

[2]*Department of Mechanical Engineering, Faculty of Engineering, Adnan Menderes University, Aydin, Turkey*

Abstract-In this work, we present our design regarding a visual analysis of the 2D motion under gravity running under the Wolfram's CDF (Computable Document Format) player. Mathematica software presents interesting visual panels like a user console to achieve animations and manipulations. Wolfram's CDF also enables users to run them interactively in a real time computation. Once it is coded on Mathematica software, it is easily converted to the CDF file to be run on the Wolfram's CDF player, which can be easily and freely downloaded from the Wolfram's internet pages. In our design, users can control the kinematics parameters involving gravitational acceleration, initial position, projection angle and velocity values of the point mass objects as well as animation parameters such as animation speed to visualize the real-time-computed position and velocity values as a function of time. By setting appropriate combinations, users can study one or two-object cases under the selected gravitational acceleration value with the desired animation parameters. Our design is thought to be useful for studying Newtonian kinematics maintaining the required specifications given in the literature for teaching as well as professional applications in science and engineering.

Keywords – Projectile motion, Mathematica, Computable Document Format, Wolfram's CDF technologies

I. INTRODUCTION

Motion in 2D under gravity is an introductory physics subject in Newtonian mechanics involving various kinematical studies of motion such as free-fall, projectile motion, vertical motion, horizontal motion, and coincidence problems for one or more object cases [1, 2]. Each of these motions, can be studied kinematically (and also dynamically) by some applets involving animations and simulations designed in various software such as Matlab, Excel, Ansys, Java, Phyton, Mathematica, etc. [3−13]. These applets are very important in teaching as well as professional applications in science and engineering and their quality can be evaluated by some certain criteria as described in [14−18]. Namely, for such applets, it is desired that users can choose the parameters freely and various physical situations of the sub-subjects, as much as possible, should be studied in a combined single applet with a very simple user panel. Results are also desired to be reasonably accurate. Mathematica presents interesting visual panels like a user console to achieve animations and manipulations in a real time computation as in [13, 19−25] to maintain these criteria. It is also desired that users can interactively use these codes in a user-friendly environment without purchasing a complex software. Wolfram's CDF (Computable Document Format) enables users to run these codes interactively and it seems very useful for designing such applets in the CDF to maintain these criteria. We suggest that various motion types regarding the 2D motion under gravity can be studied kinematically by using Wolfram's CDF effectively.

We present our design involving two-point mass objects with the related pseudocodes in Mathematica to study these motions unified in a single applet. Desired motion type such as free fall, trajectory motion (with or without ramp), one or two-point masses, etc. can be studied by the user selected physical and animation parameters from the control panel. One of the main advantages in using it is its usability without requiring an extra software other than the Wolfram's CDF player, which is entirely free. Moreover, once it is coded on the Mathematica software, it is easily converted to the CDF file and users can run it interactively [13, 19−25].

In our design, users can control both kinematics parameters and animation parameters for studying motion of one or two objects by using sliders (or entering the desired values) among the labeled menu in the user control panel designed at the top of the CDF window. Kinematics parameters involve: horizontal and vertical positions of two ramps (ramp A for mass A and ramp B for mass B), projectile angles and projectile (initial) velocities (of mass A and mass B), and gravitational acceleration. As to the animation parameters, we have: update button, animation repeat and animation speed parameters. The kinematics and animation parameters in our design are listed in Table 1 below. The produced CDF file presented here can be freely downloaded from the web page of our institution given in [26] at the moment and readers are encouraged to use it.

Table -1 User controlled parameters defined.

user control		Parameter	symbol	default val.	min. val.	max. val.	incr.
kinematics parameters	Mass A	ramp hor. pos. (m)	x_{0A}	10	-1000	1000	1
		ramp vert. pos. (m)	y_{0A}	150	0	1000	1
		projectile angle (°)	α_A	30	0	360	1
		initial velocity (m/s)	v_{0A}	10	0	1000	1
	Mass B	ramp hor. pos. (m)	x_{0B}	10	-1000	1000	1
		ramp vert. pos. (m)	y_{0B}	150	0	1000	1
		projectile angle (°)	α_B	45	0	360	1
		initial velocity (m/s)	v_{0B}	20	0	1000	1
	Gravity	gravitational accel.	$g\ (m/s^2)$	9.78	0	100	1
Animation parameters		update	$update$	0	0	1	flip-flop
		anim. repeat	$repeat$	3	1	10	1
		anim. speed (frame/s)	$speed$	25	10	250	discrete

II. DESIGN AND PROPOSED ALGORITHM

Our design enabling study of one and two-point objects involves parameters whose symbols, default values, minimum/maximum values, and user-controlled increment values are listed in Table 1. A general view of our design while running in the "parameter-select mode" and in the "run mode" are given in Figure 1 and figure 2, respectively.
These modes are sequentially activated by pushing the update button. Kinematics parameters can also be changed by the user. There are also extra options such as forward/backward, slow, fast, stop, etc. and they are opened by clicking the "+" symbols at the right of the parameters in the user console. General view of the user console with these options opened are given in Figure 3. Although kinematics parameters can always be entered by the user, they are activated in the run mode in our design.
These modes are sequentially activated by pushing the update button. Kinematics parameters can also be changed by the user. There are also extra options such as forward/backward, slow, fast, stop, etc. and they are opened by clicking the "+" symbols at the right of the parameters in the user console. General view of the user console with these options opened are given in Figure 3. Although kinematics parameters can always be entered by the user, they are activated in the run mode in our design.

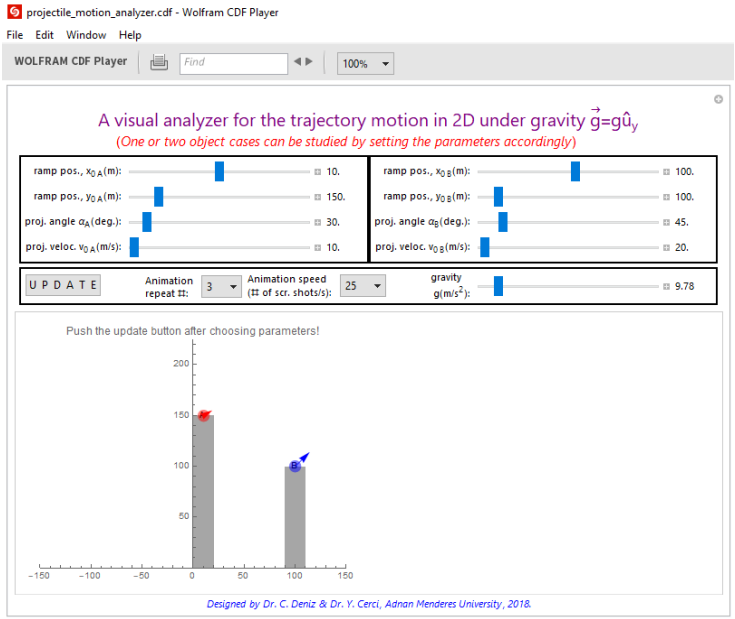


Figure 1 General view of our design while running in the "parameter-select mode" at the default parameter values.

http://dx.doi.org/10.21172/ijiet.94.20

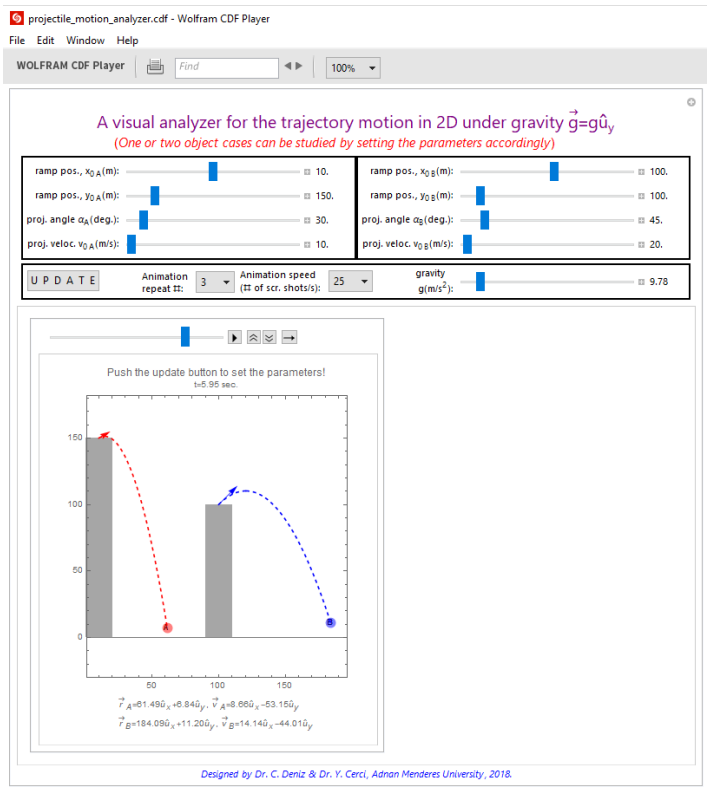


Figure 2 General view of our design while running in the "run mode" at the default parameter values.

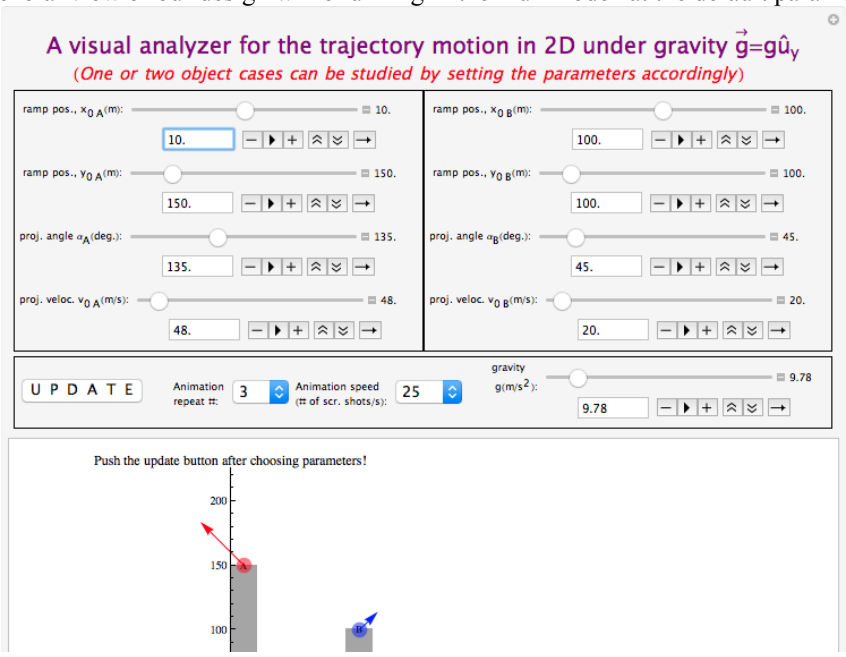


Figure 3 A view of the user console of our design with the extra options are opened in the user console.

A. General Pseudocodes (Main Module)

General pseudocodes used in this design are as follows:

Step 1: initialize

(*position x of balls A&B:*) x0A=100m; x0B=150m;

(*position y of balls A&B:*) y0A=150m; y0B=100m;

(*projectile angle of balls A&B:*) αA=30°; αB=45°;

(*projectile speed of balls A&B:*) v0A=10m/s; v0B=20m/s; (*grav. accel.:*) g=9.78m/s^2;
(*anim. speed*) speed=25 screen shot/s;
(*anim. Repeat*) repeat=3; (*update*) update=0
Step 2: (*message to the user*) message="Push the update button after choosing parameters!"
Step 3: Follow inputs from user console: (*Ball A:*) x0A, y0A, αA, v0A, (*Ball B:*) x0B, y0B, αB, v0B, (*anim. parameters:*) speed, repeat, update button, (*gravity:*) g
Step 4: Plot the picture of the system: {ramp A&B, ball A&B at the initial points (x0A, y0A)&(x0B, y0B), initial velocity vectors (v0A&v0B), message}, Plot range: -1.5Max[Abs(x0A), Abs(x0B)]≤x≤1.5Max[Abs(x0A), Abs(x0B)] and 0≤y≤1.5Max(y0A,y0B)
Step 5: If update==0, go to step 2
Else Run animation module
Step 6: End (* by exiting the CDF *)

In Step 1 here, the default values of the kinematics parameters (x0A, x0B, y0A, y0B, αA, αB, v0A, v0B) and animation parameters (speed, repeat, update) are defined. In order that user can set these parameters, message is defined as "Push the update button after choosing parameters!" in Step 2. Then it follows the inputs from user console in Step 3 and plots the picture of the system in Step 4. Plot range of the system in this picture is preferred to be:

$$-1.5\,Max\,[|x_{0A}|,|x_{0B}|] \le x \le 1.5\,Max\,[|x_{0A}|,|x_{0B}|]$$
$$0 \le y \le 1.5Max\,(y_{0A},y_{0B})$$

for a better view. As the user changes parameters, this plot range preference is always conserved enabling a visual usefulness to the user in our design, though other plot range values greater than 100% (rather than 150% as preferred here) might have been chosen but it is essential that it should be always locked to these values in order that user can see the whole physical system as the parameters are changed. If the plot range values were fixed to some certain values, some region of the physical system might have been out of view in the picture. Then, in Step 4, it is checked if the user pushed the update button or not. The default value of the update button is 0 as defined in Step 1 and it runs as a flip-flop operation as given in Table 1, namely:

$$update = 1 - update$$

As the update button is pressed its status always changes from 0 (default value) in the order: 1,0,1,0,1, etc. If it is 0, this means user continues changing parameters, which means the physical system along with the animation parameters have not been selected by the user yet. As the user changes the parameters, the picture of the system appears with the message: "Push the update button after choosing parameters!". Now, if the update button is pressed by the user, then it becomes 1 and the comparison in step 5 yields FALSE which means that the parameters have been selected by the user, so run the animation module.

B. Pseudocodes for the Animation Module

Pseudocodes for the animation module are as follows:
Step 1: (*message to the user*) message="Push the update button to set the parameters!";
TrackedSymbols :> update, x0A, x0B, hA, hB, αA, αB, v0A, v0B
Step 2: Define functions:
(*position x of balls A&B:*) xA(t, other parameters); xB(t, other parameters);
(*position y of balls A&B:*) yA(t, other parameters); yB(t, other parameters);
(*x comp. of velocity of balls A&B:*) vxA(t,other parameters); vxB(t,other parameters);
(*y comp. of velocity of balls A&B:*) vyA(t,other parameters); vyB(t,other parameters);
Step 3: Evaluate :
(*flight time of balls A&B:*) tflightA; tflightB
(*flight time of the system:*) Tsys=If tflightA> tflightB then tflightA Else tflightB
(*max. height of balls A&B:*) HA; HB
(*max. height of the system:*) Hsys=If HA> HB then HA Else HB
(*range of balls A&B:*) RA; RB
(*range of the system:*) Rsys=If RA> RB then RA Else RB
Step 4: i=0; For t=0 to 1.2*Tsys step 1.2*Tsys/speed do; (* we set the scan time up to 120% of Tsys for better view*); i=i+1; (*Plot trajectory A:*); trajA=parametric plot [xA(t, other parameters), yA(t, other parameters)] (*dashed plot *); (*Plot trajectory B:*); trajB=parametric plot [xB(t, other parameters), yB(t, other parameters)]; (*dashed plot *);

(*scr. shot matrix*) Anim[[i]]=Graphics[message, rampA, rampB, position of ball A, position of ball B, trajA, trajB, Print[$\vec{r}_A, \vec{v}_A, \vec{r}_B, \vec{v}_B$] (*we set the plot range by 110% of Rsys, and 120% of Hsys for better view*)
Step 5*:* If update==0, go to step 2 of the Main Module
Step 6: End (* by exiting the CDF *)
(Note that, comments are given here between "(*" and "*)" as in Mathematica's comment style)
Since update=1, it is now in the animation mode. So, the message is now changed to "Push the update button to set the parameters!" in Step 1 and the common kinematics formulas for the physical quantities under search (as given in [1,2]) are defined in Step 2. The physical quantities to determine the maximum values of the system (Tsys, Hsys, Rsys) for the user selected parameters are calculated in Step 3. Formulas for definitions and calculations in Step 2&3 are summarized in Table 2.

Table -2 Calculations in Step 2 of the animation mode.

physical quantities to be calculated	variable name	formula
flight time of ball A and B	tflightA; tflightB	$t_{flight_{A,B}} =: T_{A,B} = T_{1_{A,B}} + T_{2_{A,B}}$ $T_{1_{A,B}} = \frac{v_{0_{A,B}} \sin(\alpha_{A,B})}{g}, T_{2_{A,B}} = \sqrt{\frac{2H_{A,B}}{g}}$
max. height of ball A and B	HA; HB	$H_{A,B} = h_{A,B} + \frac{v_{0_{A,B}}^2 sin^2 \alpha_{A,B}}{2g}$ $R_{A,B} = T_{A,B} \cos \alpha_{A,B}$
x coordinate of particle A and B	xA; xB	$x_{A,B} = x_{0_{A,B}} + v_{0_{A,B}} \cos(\alpha_{A,B})\, t$
y coordinate of particle A and B	yA; yB	$y_{A,B} = \begin{cases} h_{A,B} + v_{0_{A,B}} \sin(\alpha_{A,B})\, t - \frac{1}{2} g t^2, if\ 0 \le t < T_{1_{A,B}} \\ H_{A,B} - \frac{g(t - T_{1_{A,B}})^2}{2}, if\ T_{1_{A,B}} \le t \le T_{A,B} \\ 0, if\ T_{A,B} < t \end{cases}$
x and y components of velocity of balls A&B	vxA; vxB; vyA; vyB	$v_{x_{A,B}} = \frac{dx_{A,B}}{dt} = v_{0_{A,B}} \cos(\alpha_{A,B}), v_{y_{A,B}} = \frac{dy_{A,B}}{dt}$
Max. values	{Tsys, Hsys, Rsys}	$\left\{ \begin{matrix} T_{sys} = Max(t_{flight_A}, t_{flight_B}), Hsys = Max(H_A, H_B) \\ , Rsys = Max(R_A, R_B) \end{matrix} \right\}$

In Step 4, index i which will be used for animation matrix is set to zero and Do-Loop starts to scan the time. We scan the time domain from t=0 up to t=1.2*Tsys. Here the 120% factor is chosen for a better view to see the complete picture of the trajectory appearing longer time even after the motion ends (before the animation repetition starts). The speed parameter was chosen by the user in the main module with the default value of 25 frame/sec. Then the trajectories of ball A and B at time t (trajA&trajB) are plotted parametrically and animation pictures are assigned to the Animation matrix as in [24−25]. Animation matrix involves: "t (*at the top*), message, rampA, rampB, position of ball A, position of ball B, trajA, trajB, Print[$\vec{r}_A, \vec{v}_A, \vec{r}_B, \vec{v}_B$] (*at the bottom*)" where the numerical values of positions and velocities of balls are numerically calculated for the present t value (according to the loop) by
$\vec{r}_{A,B}(t) = x_{A,B}(t)\hat{u}_x + y_{A,B}(t)\hat{u}_y, \vec{v}_{A,B}(t) = v_{A,B}\hat{u}_x + v_{A,B}(t)\hat{u}_y$
In our ith Animation matrix element, the numerical value of time (t) appears at the top and the numerical values of the positions and velocities of the balls at that time t appears at the bottom of the graphs. It is clear that the greater the preferred speed parameter, the larger the dimension of the animation matrix (and hence slower animation). So, although, its default value is 25 frames/sec here, elements of the Animation matrix for speed=10 frames/sec in order to save space in this text (with a smaller matrix dimension) but all the other default values are kept the same is given in Figure 4.

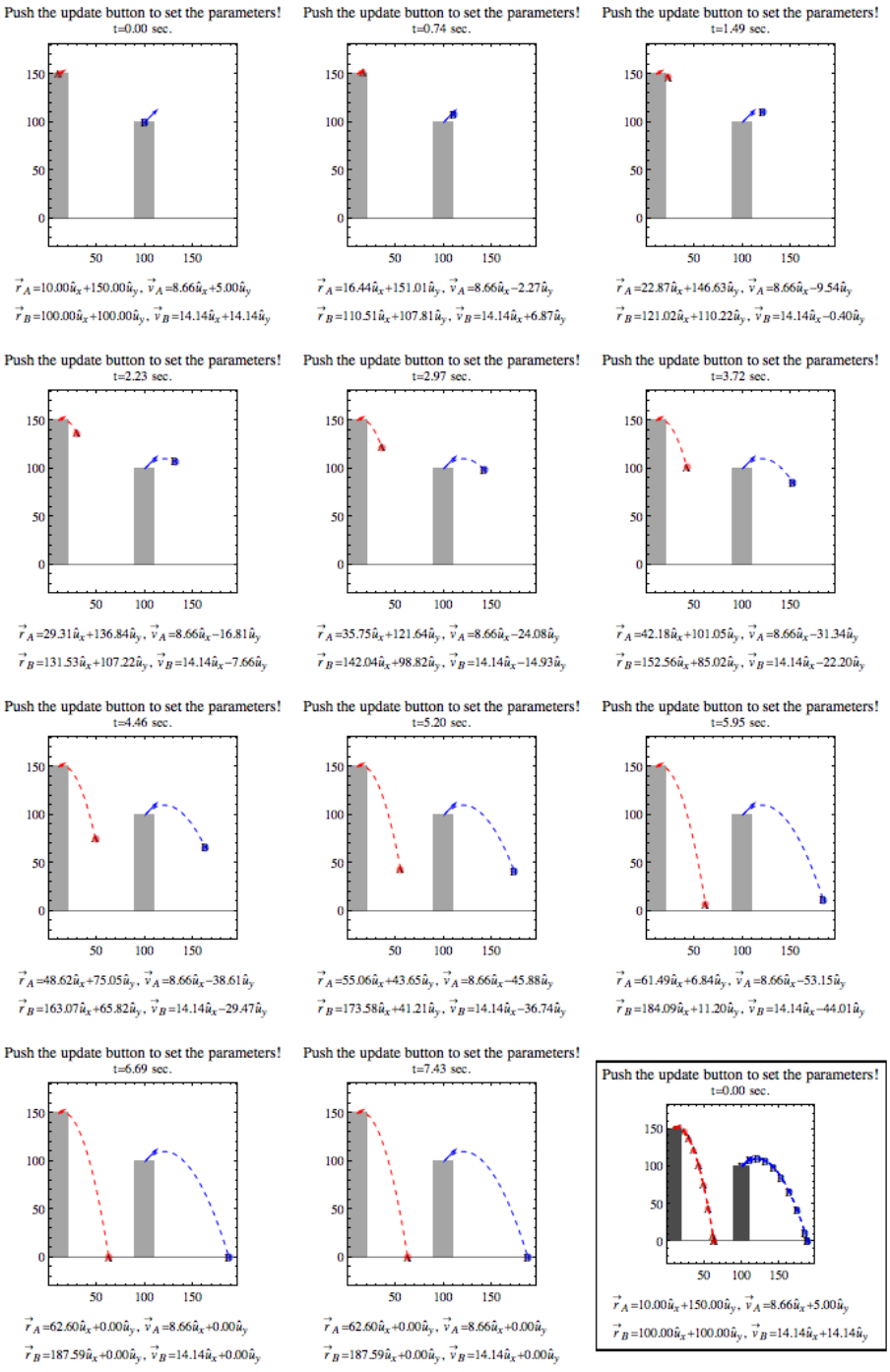


Figure 4 Elements of animation matrix: Anim[i] (last figure in box shows superposition of all matrix elements)

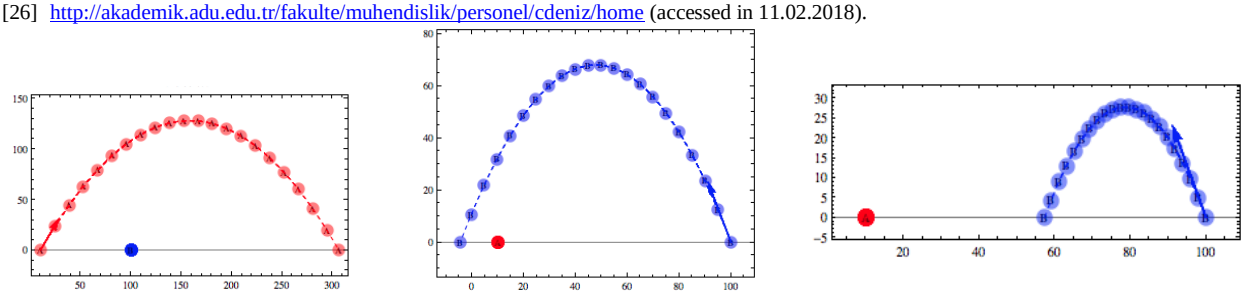
III. CONCLUSION

We used here the parametric plot of Mathematica (by the the "ParametricPlot[…]" command) for 2D graphs to study the projectile motion up to two particles. Each parametric plot for each value of time parameter is assigned to an animation matrix and they are animated with the "ListAnimate[…]" command of Mathematica as in [24−25]. Kinematics parameters involving initial positions of balls, initial projectile velocities and angles, and value of the gravitational acceleration are user controlled by sliding the related sliders or directly entering the numerical values in the control panel. Animation parameters involving update button, animation repeat number and animation speed values are also user controllable as desired. All these controllable parameters designed to be placed at the user control panel via the hereby presented pseudocodes enables users to study various types of 2D motion such as, free fall,

horizontal and vertical throw with/without ramp, motion on other planets (by changing gravity), etc. by appropriately selected parameters, some of which are as shown in Figure 5 as examples. Once such a design is coded in Mathematica, it is easily converted to the Mathematica's Computable Document Format (CDF) enabling users to use this coded software interactively without having a necessity to own any special software other than the Mathematica's free CDF player [13, 19−25]. Consequently, design presented here can be used as a useful tool in teaching physics and engineering by enabling the desired properties given in [14−18]. Moreover, Mathematica's CDF seems promising in designing such applets. Our design presented here can be downloaded from the web page of our institution given in [26] at the moment and readers are encouraged to use it as well as the comments which will be welcomed.

IV. REFERENCE

[1] B. Corona, M. Nakano, H. Pérez, "Adaptive Watermarking Algorithm for Binary Image Watermarks", *Lecture Notes in Computer Science, Springer, pp. 207-215, 2004*. R. A. Serway and J. W. Jewett, "Physics for Scientists and Engineers", 9th ed., Brooks/Cole Chengage Learning, Massachusetts, pp. 84–91, 2014.
[2] H. D. Young and R. A. Freedman, "University Physics with Modern Physics", 14th ed., Pearson, Harlow, pp. 99–106, 2016.
[3] Y. Ding and Y. Song, "The Research on Application of Computer Technology in Physics Education", *Second Int. Work. Educ. Technol. Comput. Sci.*, pp. 346–349, 2010.
[4] W. Christian and F. Esquembre, "Modeling physics with EasyJava Simulations", The Physics Teacher, vol. 45, pp. 475–480, 2007.
[5] J. R. Levenick, "Simply Java: An Introduction to Java Programming", Charles River, Massachusetts, pp. 199–220, 2006.
[6] M. A. Garcia-March et al., "Teaching classical mechanics using an applied example: Modelling and Software", *Modelling in Science Education and Learning*, vol. 2, 35–43 (2009).
[7] http://www.ae.msstate.edu/vlsm/ (accessed in 11.11.2017).
[8] http://www.engapplets.vt.edu (accessed in 11.11.2017).
[9] A. Bäcker, "Computational Physics Education with Python", Computing in Science and Engineering, vol. 9, issue 3, pp. 30–33, 2007.
[10] T.-S. Weng, M.-H. Hsu, and D.-C. Yang, "Dynamic Teaching of Kinematics of Particles and Python", *International Journal of e-Education, e-Business, e-Management and e-Learning*, vol. **3** issue 4, pp. 318–321, 2013.
[11] R. Kusak, "Applications of Wolfram Mathematica in the Theoretical Mechanics", *21st Annual Conference of Doctoral Students - WDS 2012*, WDS'12 Proceedings of Contributed Papers, Part III, pp. 88–92, 2012.
[12] S. Mangana, *Mathematica Cook Book*, O'Reilly, USA, pp. 524–52, 2010.
[13] N. E. Sanders, C. Faesi, A. A. Goodman, "A New Approach to Developing Interactive Software Modules Through Graduate Education", Journal of Science Education and Technology, vol. 23, issue 3, 431–440, 2014. (url: http://nrs.harvard.edu/urn-3:HUL.InstRepos:12336377).
[14] R. J. Beichner, "The effect of simultaneous motion presentation and graph generation in a kinematics lab", Journal of Research in Science Teaching, vol. 27, issue 8, pp. 803–815, 1990.
[15] R. J. Beichner, "The impact of video motion analysis on kinematics graph interpretation skills", Am. J. Phys., vol. 64, issue 10 , pp. 1272–1277, 1996.
[16] J. R. Lux and B. D. Davidson, "Guidelines for the Development of Computer-based Instruction Modules for Science and Engineering", Educational Technology & Society, vol 6 issue 4, pp. 125–133, 2003. (url: http://ifets.ieee.org/periodical/6_4/12.pdf)
[17] J. Bryan, "Technology for Physics Instruction", Contemporary Issues in Technology and Teacher Education, vol. **6**, issue 2, pp. 230–245, 2006.
[18] A. Pejuan, X. Bohigas, X. Jaén, "An evaluation tool for physics applets", Journal of Technology and Science Education, vol. 6, issue 1, pp. 36–51, 2015.
[19] Mathematica CDF player, Wolfram, https://www.wolfram.com/cdf-player/ (accessed in 11.11.2017).
[20] J. R. Beaulieu, *"A Dynamic, Interactive Approach to Learning Engineering and Mathematics"*, MSc Thesis, Virginia Polytechnic Institute and State University, 2012.
[21] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevA.91.013615 for two files in Wolfram's Computable Document Format (Related paper: R. S. Tasgal and Y. B. Band, Phys. Rev. A vol. 91, 013615, 2015).
[22] D. A. Russell, "Creating interactive acoustics animations using Mathematica's Computable Document Format", The Journal of the Acoustical Society of America vol. 133, issue 5, 3319, 2013, (url: https://doi.org/10.1121/1.4805539).
[23] C. Deniz,"A Fast Newton-Raphson Based Roots Finding Algorithm Design and its Applications to Circular Waveguides", El-Cezeri El-Cezeri Journal of Science and Engineering, vol. 4, issue 1, pp. 32–45, 2017. (url: http://dergipark.gov.tr/download/article-file/273099)
[24] P. T. Tam, *A Physisist's Guide to Mathematica 2nd ed.*, Elsevier, London, pp. 144–189, 2008.
[25] S. Mangana, *Mathematica Cook Book*, O'Reilly, USA., pp. 249–251, 2010.
[26] http://akademik.adu.edu.tr/fakulte/muhendislik/personel/cdeniz/home (accessed in 11.02.2018).



(a) (b) (c)

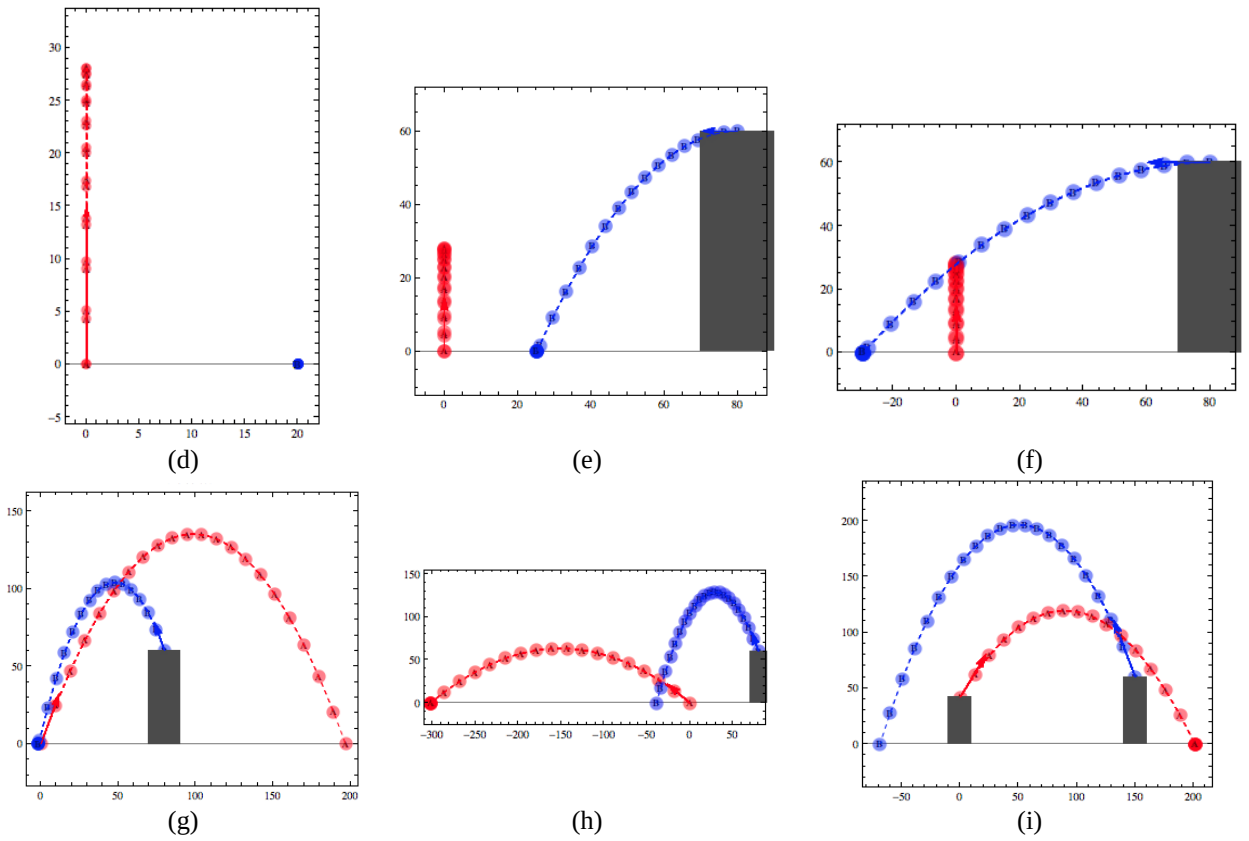


Figure 5 Elements of animation matrix: Anim[i] (last figure in box shows superposition of all matrix elements)