Getting Started

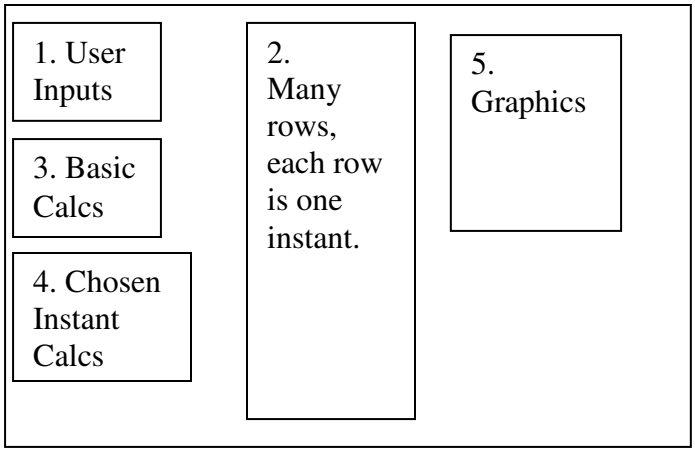
In this document, "I" means you. As we create this demo, we need to sometimes exchange two "hats" for ourselves: most of the time, we are the "creator", but at other times, we need to think like the "user". In the long term, there's only one creator, but probably multiple users, including yourself and your students.

When I think about the properties of a trajectory motion that a user might control, they include many things: what is the initial angle, and the initial speed? But there are some other properties that are less obvious. I'd like my users to be able to change "g", the gravitational acceleration on earth, to see what might happen on other planets. Also, I'd like the user to be able to select a specific time during the motion to examine. Finally, I'd like the user to be able to visually scale the display at some level, so I want them to be able to select a "magnification".

So, generally speaking, I divide my spreadsheet into 5 regions:

1. User inputs
2. A set of calculated rows, each representing one possible time during the flight
3. Basic calculations that describe the *overall* motion
4. A set of results about the specific time chosen to be investigated by the user
5. A place for computing "graphics" information.

In my spreadsheet, I usually organize these sections as seen in the diagram to the right:



User Inputs

Please open the accompanying file "ProjectileTemplate.xlsm" to begin to build your own simulation. You have to "enable content" at the top of the workbook to use all the features.

I never use column A (I keep it as a margin), so I'm going to make a list of my user inputs and place their names into column B. In column C, I list the values of these inputs, and in column D, I place their units. I'm using 5 user inputs in this version: Initial speed, initial angle θ, g, a "time selector" that I call "index", and a magnification. I can easily populate this list with numerical values, knowing that my user will probably make different choices for these things than I will. To start, I'll set the initial speed to 50, the initial angle to 60 (degrees), g to 9.8, the index to some random integer (how about 8), and the magnification to 1 (since I don't yet even know what this number means).

Basic Calculations

Since my user thinks in degrees, but Excel thinks in radians, I should make a new version of the angle that is in radians. I put this in Cell C10, using the "Radians" function. I also used some physics knowledge to calculate the expected range for this projectile motion. From the equation

$R = \frac{v_0^2}{g}\sin(2\theta)$, I can compute the range immediately from the user inputs. I can also compute the total time of flight, using this physics equation: $R = v_0\cos(\theta)\cdot\Delta t$, so $t_{\text{duration}} = \frac{R}{v_0\cos(\theta)}$. I placed these formulas into cells C11 and C12.

Next, I need to divide the motion into multiple small steps. The exact number doesn't matter at all, but in the attached spreadsheet, I'm using 201 steps that I'll number from 0 to 200. Since the time of flight is already known, I can calculate the amount of time between successive steps $dt = \frac{t_{\text{duration}}}{200}$. For my starting numbers, this came out to be near 0.02 seconds, but of course it will change as the user changes the inputs.

Just for fun, I am going to pre-calculate the maximum height of the projectile, and put that value into cell C15. Later, I'll compare this to the "measured" maximum height observed to see whether they agree: $h_{\max} = \frac{(v_0\cos(\theta))^2}{2g}$.

The Main Calculations for all possible times

Next, I need to create a set of columns for which each row represents an instant in the flight of the projectile. I start by making a numbered list of values from 0 to 200 that I call "index" number, or simply "row number". As I start to type in the list (0, then 1, then 2, etc.) I quickly realize that I don't want to spend forever typing in integers. If I highlight the first few numbers (even just the 0 and the 1 in cells F3 and F4 are enough), I can grab-and-drag the tiny little dark box seen in the bottom right corner of the selection, and pull it downwards until I reach the end of my desired list (200, in this case).

The next column over (G) is going to represent "time", in seconds. I want the time to move forwards from $t = 0$ in the first row, to some t_{max} in the 200th row. Again, I don't want to type in 200 times, nor do I even want to type in 200 equations for time.

So in cell G3, I type in the following: "=F3*C13", which is (index #)×(time step size).
Since this cell is row zero, the result is that the time is also zero.

Next, I want to auto-fill all the other time rows. To do that, I can select call G3, which has the equation for the first time already typed into it, and then double-click the dark dot in the bottom right corner of the cell (instead of dragging it downwards as we did before). When we double click, Excel notices that the column to the left (column F) is already full, so it decides to auto-fill this new column (G) to the same length as column F.

Sadly, I see that the results are mostly errors. The time has not, in fact, increased from 0 to t_{max} through this column. When I click on some of the cells in column G, I notice that Excel has decided, correctly, to use the adjacent index # for each of these calculations, by adjusting "F3" to "F4", then "F5", etc. for each subsequent cell. That's great. But, it also did the same thing with my *dt* value in cell C13, sliding it to "C14", then "C15", etc. I want to "lock down" cell C13 so that all

the calculations use this same *dt* value, while allowing Excel to continue adjusting the references to column F. We do this with the dollar sign.

Every cell reference is comprised of a letter and a number (such as "C13"). When Excel auto-fills downward, it always chooses to adjust the number portion of the cell reference as it goes. To "lock in" the number part, so that Excel won't adjust it during an auto-fill, we place a dollar sign in front of the portion of the cell reference we want to lock. So, in this case, we want "C$13" in our calculation in cell G3, instead of just "C13". You can also lock the letter portion ("$C13"), or both parts ("$C$13"), but we don't need to do that in this case.

So, I'll fix cell G3 with a dollar sign, and then double-click the corner to re-do the autofill.

In the next column, I'll create values for the *x* position as a function of time.
I'll use the physics equation $x = v_0\cos(\theta)\cdot\Delta t$.

Therefore, cell H3 becomes "=C$2*COS(C$10)*G3". Note that I used dollar signs twice for each of the column C references. I'll also calculate a column for the *y* positions in column I using the physics equation $y = v_0\sin(\theta)\cdot\Delta t - \frac{1}{2}g\Delta t^2$. In columns J and K, I'll also calculate the *x*-component of velocity using $v_x = v_0\cos(\theta)$, and the *y*-component of velocity using $v_y = v_0\sin(\theta) - g\Delta t$. Be sure to use dollar signs in front of the numeric part of any cell reference to cells in column C. For each new column, I'll double-click the top cell (or, you can select all of cells H3 through K3 and double-click the entire collection all at the same time) to build all the rows in the main calculations section.

At this point, we can check our work so far by making a plot of columns *x* and *y* and verifying that it is a proper parabola. Insert a "scatterplot" having dots connected by smooth curves. Sometimes, Excel makes bad decisions about the range it chooses for x and y on plots. We really want 10 horizontal meters to look the same as 10 vertical meters. In other words, we want to control the "aspect ratio" for this plot. But that never happens by default. Worse, it's very hard to fix it unless we restrict the user in some way. For example, suppose that one user chooses an 87° launch angle, with a speed of 10 m/s, but another user wants to have the same angle but with a 50 m/s launch. In the first case, the range is about 1 m, but in the second case, it's about 250 m. in the first case, the height of the launch is 5 m, but in the second case, it is 65 m. It is difficult to decide in advance what the total horizontal range of the plot should be, if the plots can vary in content by that much!

It is possible to write a macro to control both the aspect ratio and scaling for a scatter plot, but that's too hard for us today. Instead, let's just assume that the user will only ever choose some "reasonable" speeds and angles:

Assumption 1: $10° \le \theta \le 70°$
Assumption 2: $10 \text{ m/s} \le v_0 \le 50 \text{ m/s}$

It is also possible to enforce these assumptions, but I'll leave that for later, too.

For this range of options, the range reached might be as small as 3.5 m, and as large as 255 m.
The maximum height reached might be as small as 0.15 m, or as large as 110m.
Let's adjust the plot so that the largest of each of these numbers is always visible, *and* so that the aspect ratio is correct. But when we do this, shorter launches will look pretty small on our plot.

I'll double click on some of the horizontal axis numbers to bring up some axis formatting options. The little green "bar chart" that appears on the right is a submenu that lets us enforce our preferences for axis formatting. Change the Min to 0, the Max to 260, and the Major to 40. Then, do something similar for the vertical axis: set the Min to 0, the Max to 120, and the Major to 40. Make sure that for all 6 numbers, it no longer says "auto" next to the numerical choice. If any of these 6 properties was already correct by default, then we need to first get it out of "automatic" mode. If any of them still has a small box called "auto" next to it, then change the number to some other random number, and then back to the desired value.

Finally, go to the plot itself and grab one of the edges. Drag the edge to rescale the plot until the boxes formed by the major grid lines appear to be squares rather than rectangles. Now, this plot won't resize itself anymore when the user changes some of the parameters. Of course, now, when the user picks small initial speeds, the plot will appear pretty tiny.

Calculations that are specific to a chosen time step

The user needs to be able to investigate what's happening at any *specific* time. They do this by using the "index" input that we created in cell C5. When the user specifies that index is "8" (as we have done so far), then what we mean is that the user wants to get some information about row #8. I can easily imagine that the user might want to ask about at least 5 things. Specifically, I mean cells C18 through C24: every row in the "big" calculation columns (like t, x, y, v_x, and v_y).

Let's extract these for the user. To do this, we'll use a new function called "VLOOKUP" in excel. The "V" stands for "vertical". The job of this function is to take a row number (like row #8), and scan vertically down a column until it finds that number, and then look horizontally at one of the adjacent rows. You have to give this function 3 pieces of information: what row you want, the location of the table that contains all the info, and which column value you want to extract. There is also an extra required input which doesn't matter to us today; we'll just set that to "1". It gives instructions to Excel about what to do if it can't find the row that we asked for.

The syntax is VLOOKUP(index #, table location, column #,1)

For example, for the data seen to the right, the bold numbers refer to Excel cell names. So, Cell B4 is "15".

The function VLOOKUP(44, A2:C7, 2, 1) will tell me "31.9".
The function VLOOKUP(41, A2:C7, 1, 1) will tell me "10".

	A	B	C
1	index	time	Temp.
2	40	5	25.1
3	41	10	25.3
4	42	15	26.2
5	43	30	25.7
6	44	60	28.6
7	45	120	31.9

In our own projectiles spreadsheet, =VLOOKUP(C$5,F$3:K$335,2,1) will tell me the time of the index number that the user selected in cell C5. Notice that I used several dollar signs, because I plan to copy this function down into my next 4 cells to get current values of x, y, v_x, and v_y. For each copied cell, I have to manually adjust the desired column number in the VLOOKUP function (while time is column 2 of the selected table, x is column 3, y is column 4, etc.).

There are other questions that the user might ask that aren't directly listed in our tall table. For example, the projectile speed and angle of flight aren't columns. I can calculate these using basic physics equations using the v_x and v_y that I already obtained from VLOOKUP:

C23: Angle = atan(v_y / v_x), from basic trig, and
C24: Speed = sqrt(vx^2 + vy^2), from the Pythagorean theorem.

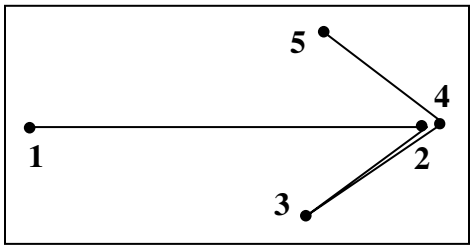
The resulting angle is in radians, but I'm fine with that.

Graphics

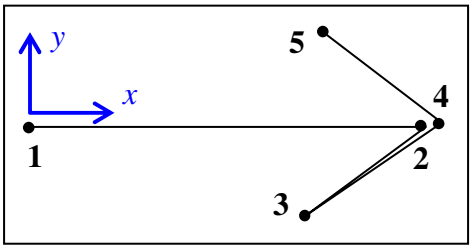
We've completed regions 1, 2, 3, and 4 of our spreadsheet as outlined on page 1. All that's left is to make stuff pretty: the graphics

One of my goals is to eventually draw an arrow representing the velocity on top of the trajectory for the user to see. The arrow should point in the direction of motion, and proportionally increase in length as the projectile increases in speed. To be able to do this, I first need to be able to at least draw some kind of an arrow.

An arrow can be drawn in Excel as a scatter plot by connecting some points with straight lines. Here's the simplest arrow I can think of: It's drawn by tracing your pen from point 1 to point 5 without picking the pen up. Here, points 2 and 4 should ideally be at the same location.

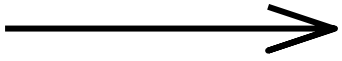


To draw this in a scatter plot, I need to create some coordinates for these 5 points. I'll refer to the points with respect to a coordinate system, shown in blue. Points to the right of the origin have positive x values, and points above the origin have positive y values. I'll (arbitrarily) choose to make this arrow "1" unit wide for now. My list of points might look something like those seen in this table (but there are many other options, depending on your view of what a "good" arrow should look like:



Point #	x	y
1	0	0
2	1	0
3	.8	-0.07
4	1	0
5	.8	0.07

To test my arrow, I'll copy these values into Excel in my "Graphics" section, and then plot it. The results look strange because Excel has set the aspect ratio to something silly again. When drawn with a consistent aspect ratio, this arrow looks something like this one:



Next, I remember that if this arrow represents velocity, it will not always be oriented to the right as seen here. It will change direction, length, and position as the object moves. I'll take care of each of these separately, and then combine them.

Rotation is the hardest. Since I already know the angle of the velocity in cell C21, I can use the trigonometry of rotation to compute new rotated coordinates for this arrow. For each of the five (x, y) points that comprise the arrow, I can compute new (x,y) pairs using:

$$x_{new} = x_{old} \cdot \cos(\theta) - y_{old} \cdot \sin(\theta)$$
$$y_{new} = x_{old} \cdot \sin(\theta) + y_{old} \cdot \cos(\theta)$$

Using the appropriate cell references, I typed this formula into cells M13 and N13, respectively. Note that, as usual, I used dollar signs when referring to column C. Then, I dragged these formulae down so that I have all 5 new points.

I'm going to combine the last two operations into one final arrow representation. I am going to Rescale the x and y coordinates to make them larger with larger speed, and I am also going to rescale them using my so-far-unused user input called "magnification". Additionally, I am going to allow this arrow to *move* with the projectile by adding the position of the ball, too. As equations, this looks like:

$$x_{final} = (x_{rotated})(\text{magnification})(\text{speed}) + x_{index}$$
$$y_{final} = (y_{rotated})(\text{magnification})(\text{speed}) + y_{index}$$

I type these equations into cells M22 and N22, respectively, then dragged them down so that I have 5 total points.

Adding the Arrow to the Plot

I want my final scatter plot to include 3 total things:

1. The trajectory (which we already have), consisting of smooth lines without dots.
2. The arrow, consisting of straight lines without dots.
3. The projectile itself, consisting of a single dot without lines.

To add the second two to the existing plot, I right click anywhere on the plot, and choose "Select Data". This brings up a sub-menu. I'll click on "add" to draw the arrow. I chose "velocity" as my new series name. The "series x values" are cells M22 through M26, which I selected with the mouse (or, you can type "=Sheet1!$M$22:$M$26"), and the "series y values" are cells N22 through N26.

I'll also "add" my final plot, which I'll call "Ball". The "series x values" is just a single cell, C19, and the "series y values" is just a cell C20.

When I add these plots, they are immediately visible on the plot, but they are ugly. I'll right-click on any of the points comprising the arrow, and choose "format data series". Using the green "paint bucket" submenu, then the "line" submenu, I chose "Solid Line", "Red", Width = 2, and way down at the bottom, I unchecked the "smooth line". Then, near the top of this menu, I went to the "Marker" submenu, then opened up the "Marker Options" Triangle, and changed the marker style to "none.

Next, I right-clicked on the ball itself on the plot. It is a little gray dot, and hard to see. As before, I went to "format data series", then the green paint bucket, then "marker", and the "marker options" triangle. I changed the Marker style to "built in", and made the size larger (9). I also changed the opened the fill triangle, and changed the fill option to "solid", and then picked purple. Since I was already here, I also went to the "border" tab, and changed the border to black.

Finally, I went to the "format data series" for the trajectory itself, and turned off all the markers.

The plot looks pretty good right now. I can pretend to be the "user", and start changing values for the initial angle, speed, index number, or even change the gravity. As I look at the results, if I think the red arrow is too long or too short, as the user, I can change the magnification of the arrow in cell C6. Be sure to try typing in different numbers into cell C% ("index"). Don't type anything not between 0 and 200.

Adding a Scroll Bar

In the nicer user-controlled animation, the user doesn't have to type any numbers in. Instead, they use a graphical control called a "scroll bar". We should probably add separate scroll bars for every user input, but for now, we'll just add the most important one: index number. That way, we can watch the ball fly without a bunch of numbers manually.

To add a scroll bar, we have to turn on the Developer Tab of the ribbon. Find a blankish part of the ribbon, then right-click and choose "Customize the Ribbon". Then, "choose commands from" "Main Tabs", then highlight the "Developer Tab" and add it to your ribbon using the arrow in the center of the panel. Finalize this by clicking the "OK" button at the bottom.

Now, we can use the Developer Tab. Click on the "insert" tool to get a submenu of choices, then pick the "scroll bar" (2nd row, 3rd column). This gives a cursor to draw a rectangle somewhere on your spreadsheet. Draw a rectangle that a skinny horizontal bar. I drew mine so that it more-or-less covered cells A8 through E8. The scrollbar isn't actually in these cells, it's just laying on top of them.

Now right-click the scroll bar and go to "format control". There are 4 properties that matter. Since this scrollbar will be used instead of the user typing in numbers for "index" in cell C5, I want to set the following:

Min = 0
Max = 200
Increment = 1
Linked Cell = C5

These values mean that using the scroll bar, the user can select a number from 0 to 200 inclusive, and that value will be immediately copied into cell C5. If you want the user to only be able to select values like 5, 10, 15, 20, etc. , then you'd set the increment to 5 instead of 1.

Adding a "RUN" button.

The run button is a macro written in Visual Basic that, in this case, adjusts the slide bar from 0 to 200 automatically for the user, in steps of 1.

The macro I'm going to write here is very primitive, and makes no attempt to control the rate at which the slide bar is manipulated. On a fast gaming computer, the manipulation might happen so fast that the user can't see it. On an older, slower computer, the motion might try your patience. On my laptop (a midrange 2013 hp), it's pretty reasonable.

In the Developer Tab, add a "Command Button" (This time, it's an "Active-X Control", 1st row, 1st column). This again changes you to a draw cursor, so draw a button sized rectangle somewhere on your spreadsheet.

Now that we added this command button, the file format "projectiles.xlsx" is no longer a valid file format, because ".xlsx" files may not include macros. We should use the "save as" menu to resave our file as a ".xlsm", or "macro-enabled" Excel file.

Next, right-click this new button, and select "Properties" (not "format control").

About halfway down the list, change the "Caption" to "Run!" (or something similarly exciting). Then, close this list, and right-click on the button a second time. This time, choose "view code". This will open a Visual Basic editor window.

In this macro editor, here is the code I entered. Note that the first and last lines are already pre-entered for you by Excel, so you don't need to type them or change them.

```
Private Sub CommandButton1_Click()
'
' StartButtonClick Macro
'
Dim istep As Integer
    For istep = 0 To 200
      Range("C5").Select
      ActiveCell.FormulaR1C1 = istep
      Range("D5").Select
    Next istep
End Sub
```

The first line is the name of the program for this macro
The next three lines are comments, so they do nothing. Comments always start with apostrophes.
The next line ("Dim") reserves memory for a variable I'm creating called istep.
The next line starts a "For" loop that counts from 0 to 200.
The "Range("C5").Select" causes Excel to pretend to be the user, and clicks on Cell C5 for us.
The "ActiveCell" line causes Excel to pretend to be the user, and types a new number into the selected cell (in this case, cell C5, which is our "index". The number typed is the iteration number of the for loop (0, for now).
The Next Range("D5").Select causes Excel to "finalize" the previous line by "entering" it, by clicking on a new cell.
The "Next" line instructs the For loop to increment it's variable to the next value (1), and repeat everything after the "For" line. If the variable istep is already 200, then the program discontinues returning to the "For" line.

Using the Command Button

You can close the Visual Basic/Macro editor at any time. The code you entered is saved when you save the entire workbook. However, even when you close the Macro editor, Excel might still be in

"Designer" mode. On the Developer Tab of the ribbon, look at this icon. If it's highlighted as seen here, click it once so that it's no longer green. When it's green, the workbook is in programmer mode, but when it's white, it's in user mode.

Now you can actually press the command button to see what it does.