Projectile Motion Tutorial #5

- a 2D projectile motion model using numerical analysis of projectile dynamics (including aerodynamic drag)

<excelunusual.com>

by George Lungu

An outline of the formulas derived in the previous part:

$$v_{current\_x} = v_{previous\_x} \cdot \left(1 - \frac{\rho \cdot A \cdot Cx \cdot \sqrt{v_{previous\_x}^2 + v_{previous\_y}^2}}{2 \cdot m} \cdot \Delta t\right)$$

$$v_{current\_y} = v_{previous\_y} \cdot \left(1 - \frac{\rho \cdot A \cdot Cx \cdot \sqrt{v_{previous\_x}^2 + v_{previous\_y}^2}}{2 \cdot m} \cdot \Delta t\right) - g \cdot \Delta t$$

$$x_{current} = x_{previous} + v_{current\_x} \cdot \Delta t$$

$$y_{current} = y_{previous} + v_{current\_y} \cdot \Delta t$$

$$x_0, y_0, v_{x\_0}, v_{y\_0}, \rho, A, Cx, m, g, \Delta t$$

As a conclusion, to run this model for as long as we want, all we need is the above four formulas plus the above ten constants. We choose ∆t so as to get a good compromise speed-precision (a smaller Dt results in a better precision but a slower model).

Let's implement this in Excel:

-Copy the "Tutorial_3" worksheet and name the new worksheet "Tutorial_4-5"

- Cut C25:G2200 and paste it one column to the right (to D25:H2200)

- Replace G36 with H36 in the macro "Fire"

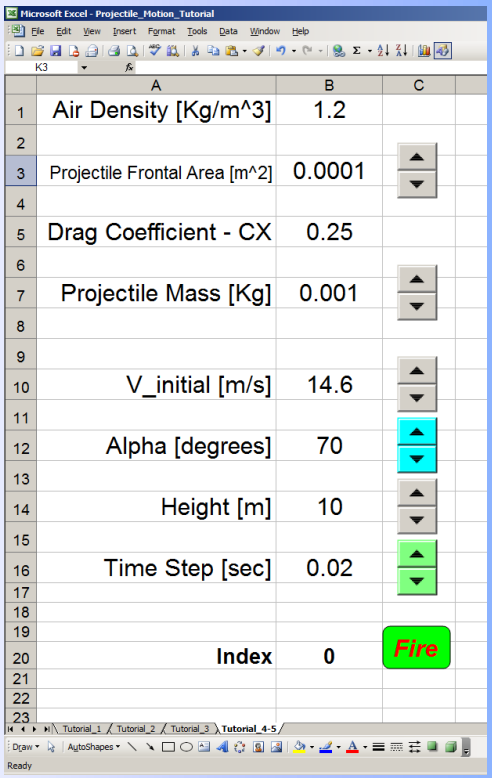
- Hit the "Fire" button and verify that the macro and the animation are still working properly

```
Sub Fire()
[B20] = 0
Do While [G36] > 0 And [B20] < 2000
DoEvents
[B20] = [B20] + 1
Loop
Sleep 1000
[B20] = 0
Exit Sub
End Sub
```

Add the following entries to the input data area:

-Air Density (cell B1), Projectile Frontal Area (cell B3), Drag Coefficient CX (cell B5) and Projectile Mass (cell B7)

- The buttons for Projectile Frontal Area and Projectile Mass can go between Min=0 and Max=18




```
Private Sub Frontal_Area_Change()
arrScale = Array(1, 2, 5)
[B3] = arrScale(Frontal_Area.Value Mod 3) * 10 ^ Int(Frontal_Area.Value / 3) / 1000000
End Sub
----------------------------------------------------------------------------------------------------------------
Private Sub Mass_Change()
arrScale = Array(1, 2, 5)
[B7] = arrScale(Mass.Value Mod 3) * 10 ^ Int(Mass.Value / 3) / 100000
End Sub
```

-The fist macro changes the Projectile Frontal Area by the following series (oscilloscope style):

0.000001, 0.000002, 0.000005. 0.00001, 0.00002…….0.02, 0.05, 0.1, 0.2, 0.5, 1

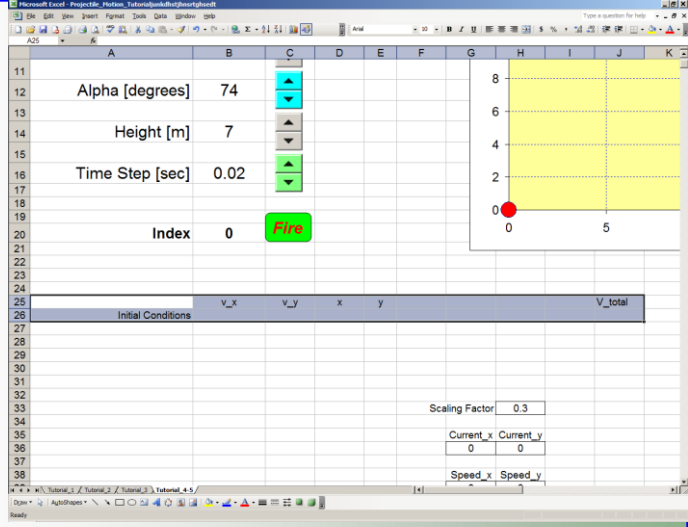
- The second macro changes the Projectile Mass by the following series (oscilloscope style):

0.00001, 0.00002, 0.00005. 0.0001, 0.0002……. 0.2, 0.5, 1, 2, 5, 10

- This type of macro uses two VBA functions, “Mod” and “Int” to crate an “oscilloscope style” almost exponential variable setting and it will be explained in a future post

The initial conditions:

- The cells you can see to the right with text in the highlighted areas are just labels



- After filling in for labels let's input the initial conditions:
- Cell B26: "=B10*COS(RADIANS(B12))"
- Cell C26: "=B10*SIN(RADIANS(B12))"
- Cell D26: "=0", Cell E26: "=B14"

Here are the formulas (from page 2 of the current presentation):

$$v_{current\_x} = v_{previous\_x} \cdot \left(1 - \frac{\rho \cdot A \cdot Cx \cdot \sqrt{v^2_{previous\_x} + v^2_{previous\_y}}}{2 \cdot m} \cdot \Delta t\right)$$

$$v_{current\_y} = v_{previous\_y} \cdot \left(1 - \frac{\rho \cdot A \cdot Cx \cdot \sqrt{v^2_{previous\_x} + v^2_{previous\_y}}}{2 \cdot m} \cdot \Delta t\right) - g \cdot \Delta t$$

$$x_{current} = x_{previous} + v_{current\_x} \cdot \Delta t$$

$$y_{current} = y_{previous} + v_{current\_y} \cdot \Delta t$$

-Cell B27: "=B26*(1-B$1*B$3*B$5 *SQRT(B26^2+C26^2)*B$16/(2*B$7))"

- Cell C27: "=C26*(1-B$1*B$3*B$5* SQRT(B26^2+ C26^2)*B$16/(2*B$7))-9.81*B$16"

- Cell D27: "=D26+B27*B16"

- Cell E27: "=E26+C27*B16"

- After entering the above formulas copy down the range B27:E27 to row 2100

We need to add a column with the total speed data:

- As opposed to the previous model, in this one, the total speed drops in flight due to the air friction. It would be nice to plot the total projectile speed along the trajectory
- We can find the total speed from the X and Y speed components using Pythagoras theorem
- The data for the total speed will be placed in the range J26:J2100
- To the right there is a snapshoot of the finalized calculation area of the spreadsheet
- We will add a new scatter chart (green) displaying the total speed on top of the existing trajectory chart. The X data for this chart is taken from range D26:D2026 and the Y data is taken from J26:J2026

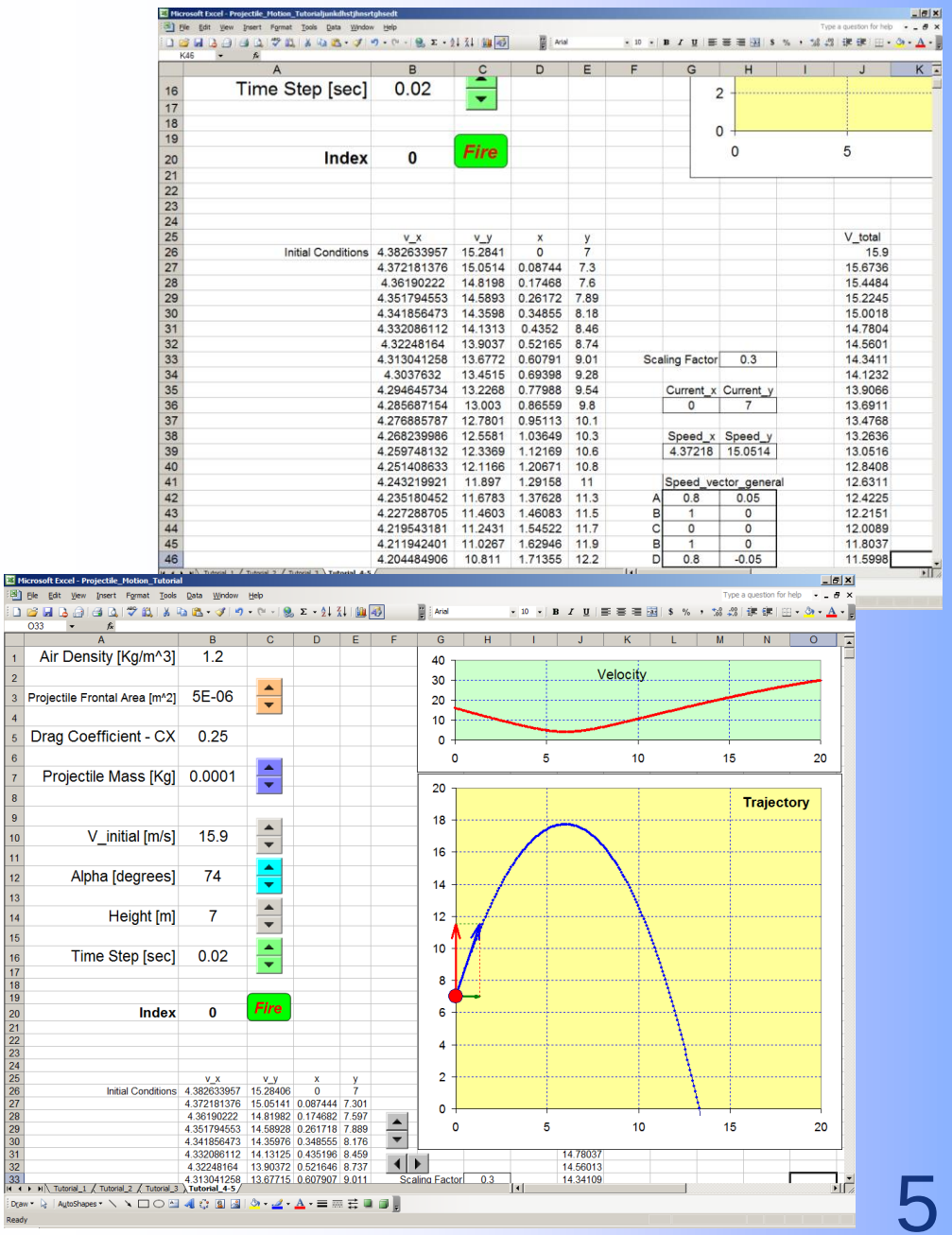


Chart renaming macros:

- We use these macros one by one to set the chart names to “Chart_1” and “Chart_2” respectively
- Select the lower chart and run “Rename1” from the VBA editor, Repeat with the upper chart and “Rename2”

```
Sub Rename1()
    ActiveChart.Parent.Name = "Chart_1"
End Sub
Sub Rename2()
    ActiveChart.Parent.Name = "Chart_2"
End Sub
```

Chart scale adjustment macros:

- The upper macro is used to adjust the X axes of both charts to a value between 1 and 1 million (same value using an “oscilloscope style” pseudo exponential rule: 1, 2, 5, 10, 20, 50, …)
- The lower macro is used to adjust the Y axis of the trajectory chart from 1 to 1 million (the upper chart Y scale is set on “Auto”)
- On the sheet there are two buttons associated to these macros

```
Private Sub Scale_X_Change()
Dim aX As Long
arrScale = Array(1, 2, 5)
aX = arrScale(Scale_X.Value Mod 3) * 10 ^ Int(Scale_X.Value / 3)
ActiveSheet.ChartObjects("Chart_1").Chart.Axes(xlCategory).MaximumScale = aX
ActiveSheet.ChartObjects("Chart_2").Chart.Axes(xlCategory).MaximumScale = aX
End Sub

Private Sub Scale_Y_Change()
Dim aX As Long
arrScale = Array(1, 2, 5)
aY = arrScale(Scale_Y.Value Mod 3) * 10 ^ Int(Scale_Y.Value / 3)
With ActiveSheet.ChartObjects("Chart_1").Chart
.Axes(xlValue).MinimumScale = 0
.Axes(xlValue).MaximumScale = aY
End With
End Sub
```

To be continued...