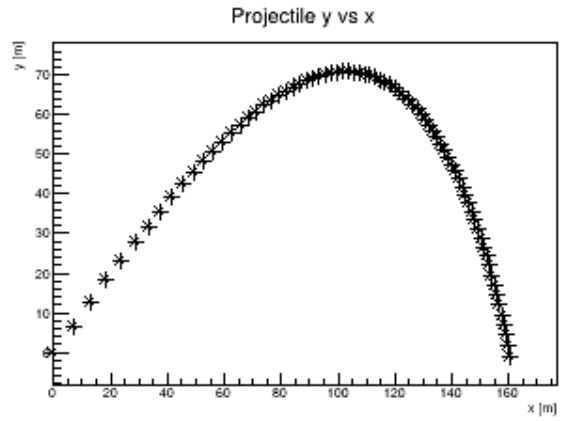
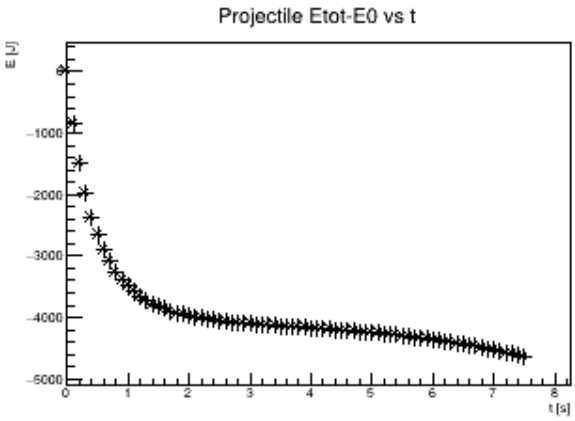
Part A: Practice using the code and doing some of your own studies

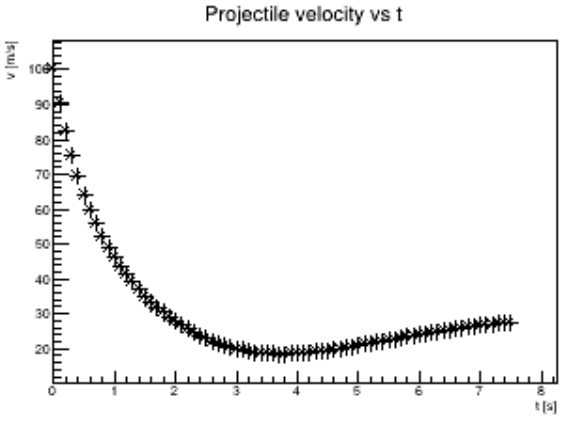
From the Demo Codes

I made a slight change to the Makefile

```
(phys56xx) udc-ba38-32c0$./RKnDemo
Warning in <UnknownClass::SetDisplay>: DISPLAY not set, setting it to 128.143.223.184:0.0
Vinit: 100 m/s
Angle: 45 deg
(vx,vy) 70.7107 , 70.7106 m/s
Error in <TCanvas::Constructor>: Invalid canvas height: 0
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
Final velocity = 27.6553
Press ^c to exit
^C
(phys56xx) udc-ba38-32c0$python3 RKnPlotDemo.py
Info in <TCanvas::Print>: png file Projectile.png has been created
Hit return to exit
```







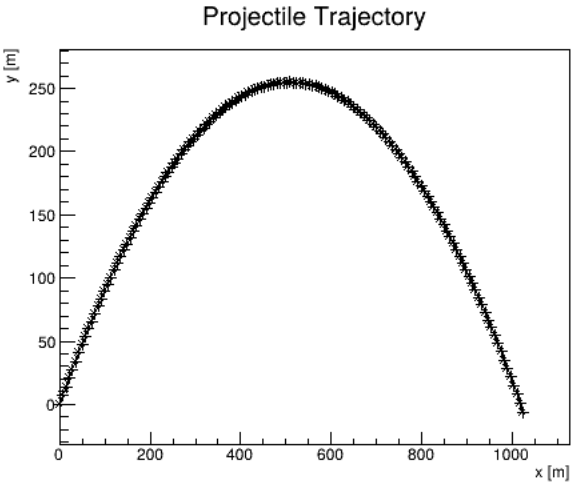
Modify one of demo programs (use the naming convention vterm.cpp or py**) to solve the projectile motion problem w/o air resistance. Modify the Makefile to also build your program is using C++.**

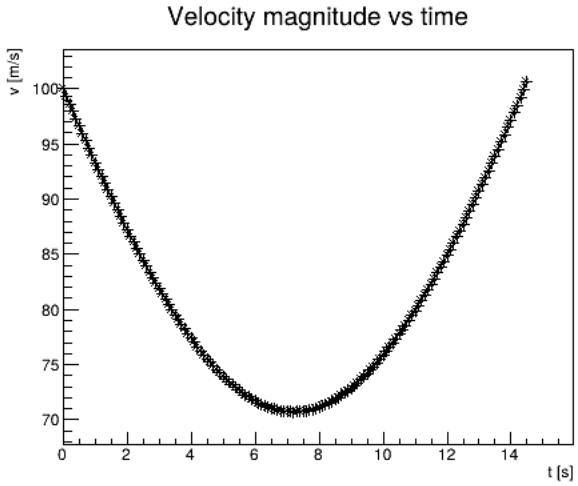
Add this to Makefile

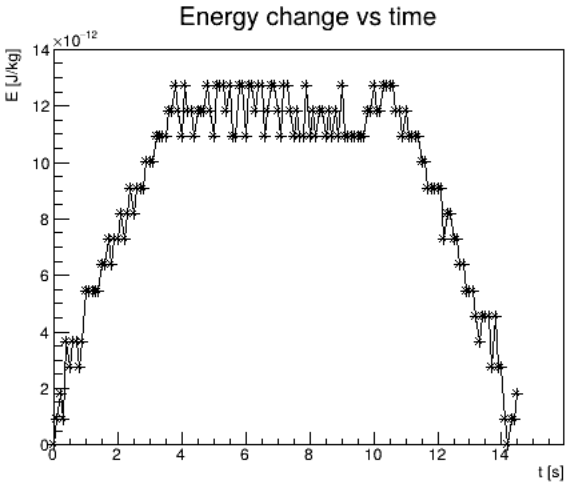
vterm: vterm.cpp RKn.cpp $(ODELIB)/RKn.hpp

$(GXX) $(CXXFLAGS) -o $@ vterm.cpp RKn.cpp $(LDFLAGS)

```
(phys56xx) udc-ba37-32c0$./vterm
Warning in <UnknownClass::SetDisplay>: DISPLAY not set, setting it to 128.143.223.184:0.0
Initial velocity: 100 m/s
Launch angle: 45 deg
Final velocity = 100.584
Saved ROOT file: vterm.root
Press ^C to exit
^C
(phys56xx) udc-ba37-32c0$python vterm_plot.py
Info in <TCanvas::Print>: png file vterm_plots.png has been created
Plots saved as vterm_plots.png
(phys56xx) udc-ba37-32c0$
```







How well is energy conserved in the solution? Play with the step size to see how this affects your observation.

Made a chance to the Makefile

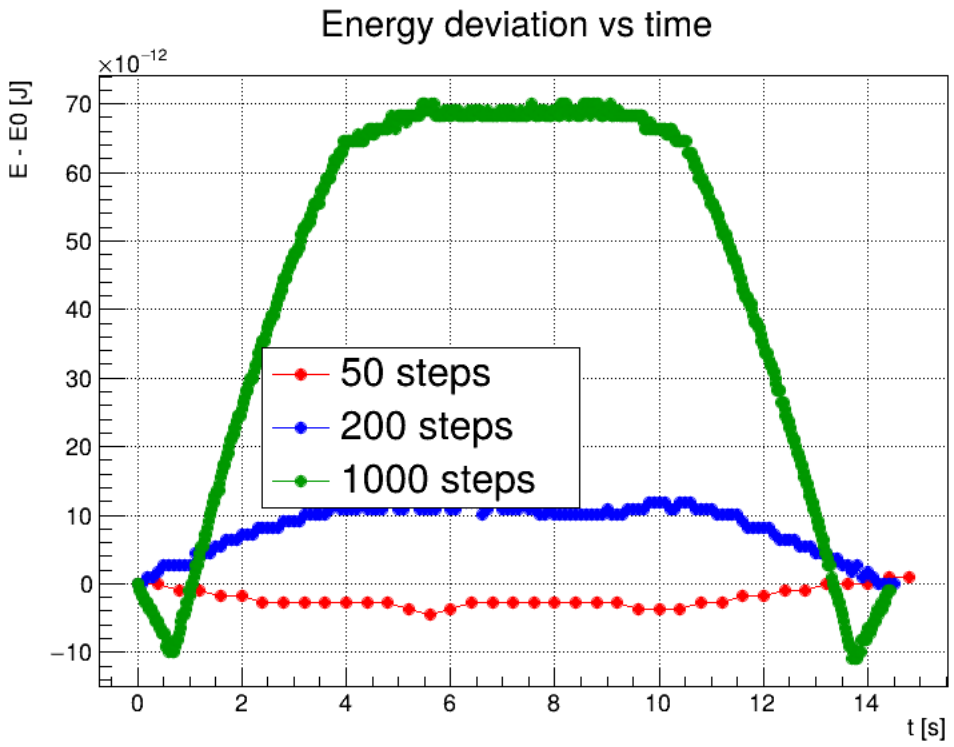
vterm_energy: vterm_energy.cpp RKn.cpp RKn.hpp

$(CXX) $(CXXFLAGS) -o $@ vterm_energy.cpp RKn.cpp $(LDFLAGS)

```
(phys56xx) udc-ba38-32c0$./vterm_energy -s 50  -o vterm_s50.root
Warning in <UnknownClass::SetDisplay>: DISPLAY not set, setting it to 128.143.223.184:0.0
Initial velocity: 100 m/s
Launch angle: 45 deg
Step size (nsteps): 50
Max energy deviation: 4.54747e-12 J
Saved ROOT file: vterm_s50.root
Press ^C to exit
^C
(phys56xx) udc-ba38-32c0$./vterm_energy -s 200 -o vterm_s200.root
Warning in <UnknownClass::SetDisplay>: DISPLAY not set, setting it to 128.143.223.184:0.0
Initial velocity: 100 m/s
Launch angle: 45 deg
Step size (nsteps): 200
Max energy deviation: 1.18234e-11 J
Saved ROOT file: vterm_s200.root
Press ^C to exit
^C
(phys56xx) udc-ba38-32c0$./vterm_energy -s 1000 -o vterm_s1000.root
Warning in <UnknownClass::SetDisplay>: DISPLAY not set, setting it to 128.143.223.184:0.0
Initial velocity: 100 m/s
Launch angle: 45 deg
Step size (nsteps): 1000
Max energy deviation: 7.00311e-11 J
Saved ROOT file: vterm_s1000.root
Press ^C to exit
^C
```

Next

python3 plot_energy_vs_steps.py this produces - energy_vs_steps.png



My Discussion

Energy conservation

- Maximum energy deviations reported:

Step size (nsteps)	Max energy deviation (J)
50	4.55×10^{-12}
200	1.18×10^{-11}
1000	7.00×10^{-11}

- These deviations are **extremely small**, essentially negligible for a 1 kg projectile.
- This confirms **RK4 conserves energy very well** in a simple projectile motion problem **without air resistance**.

Effect of step size

- Smaller number of steps (coarser step size) does **not always increase energy deviation** in this example, because the time interval (xmax=20 s) is modest, and RK4 is very accurate.
- Increasing the number of steps to 1000 gives slightly higher deviation here — likely due to **accumulated floating-point round-off error** rather than integration inaccuracy.
- Overall: for RK4 and this problem, **even very coarse steps give excellent energy conservation**.

Turn the air resistance back on. Modify the program to determine the terminal velocity (*vt*) of the projectile. Determine *vt* using the default mass and air drag parameter given in the example.

Change the Makefile

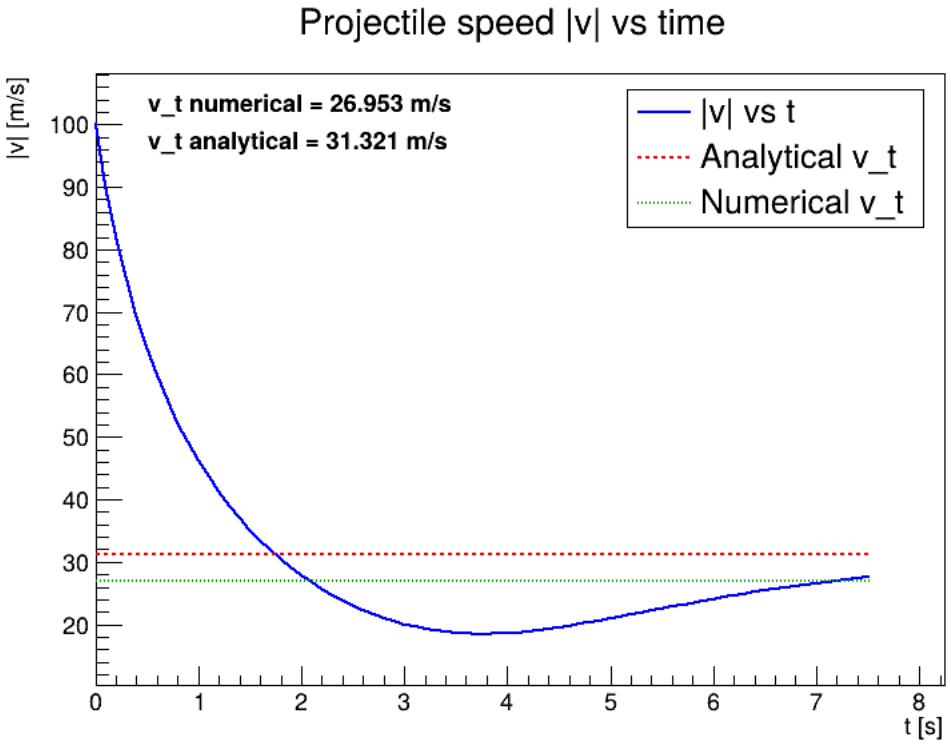
vterm_air: vterm_air.cpp RKn.cpp RKn.hpp

$(CXX) $(CXXFLAGS) -o $@ vterm_air.cpp RKn.cpp $(LDFLAGS)

```
(phys56xx) udc-ba37-32c0$./vterm_air
Warning in <UnknownClass::SetDisplay>: DISPLAY not set, setting it to 128.143.223.184:0.0
Simulating projectile fall with air resistance...
Estimated terminal velocity (numerical) = 26.9534 m/s
Analytical terminal velocity = 31.3209 m/s
```

python3 plot_vterm_air.py

Plot saved as vterm_air_speed.png



This is for m=10 kg

Analytical Terminal Velocity

For an object falling under gravity with quadratic air resistance $F_{\text{drag}} = kv^2$, the **terminal velocity** is the speed at which the net force becomes zero:

$$mg - kv_t^2 = 0$$

Solve for v_t:

$$v_t = \sqrt{\frac{mg}{k}}$$

- m is the mass of the projectile
- g is the gravitational acceleration
- k is the air resistance coefficient

Numerical Terminal Velocity

During the simulation, the projectile accelerates and eventually reaches a roughly constant speed. To estimate $v_t$numerically:

1. Compute the speed magnitude at each time step:

$$v[i] = \sqrt{v_x[i]^2 + v_y[i]^2}$$

2. Average the last 10% of the speeds (where the projectile has mostly settled)

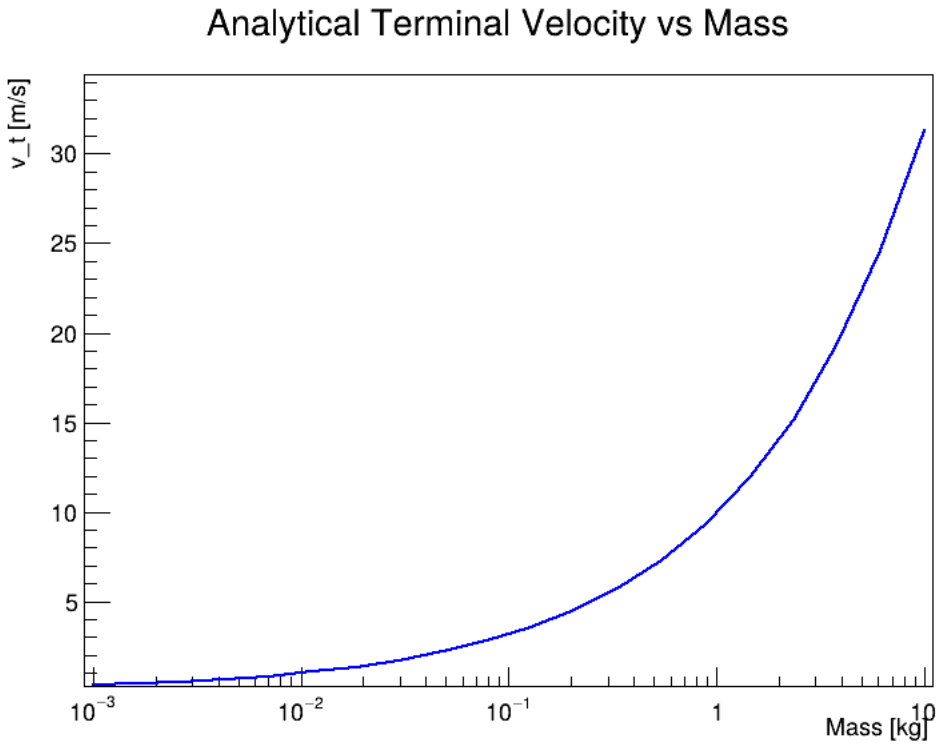
Write a program to study *vt* vs *mass*. Use this to get data to make a plot of *vt* for 1 gm < m < 10kg

Change make file

vterm_vs_mass: vterm_vs_mass.cpp

$(CXX) $(CXXFLAGS) `root-config --cflags --libs` -o vterm_vs_mass vterm_vs_mass.cpp

```
(phys56xx) udc-ba37-32c0$make vterm_vs_mass
make: 'vterm_vs_mass' is up to date.
(phys56xx) udc-ba37-32c0$./vterm_vs_mass
Mass = 0.001 kg, Analytical vt = 0.313209 m/s
Mass = 0.00162378 kg, Analytical vt = 0.399115 m/s
Mass = 0.00263665 kg, Analytical vt = 0.508582 m/s
Mass = 0.00428133 kg, Analytical vt = 0.648073 m/s
Mass = 0.00695193 kg, Analytical vt = 0.825823 m/s
Mass = 0.0112884 kg, Analytical vt = 1.05233 m/s
Mass = 0.0183298 kg, Analytical vt = 1.34095 m/s
Mass = 0.0297635 kg, Analytical vt = 1.70874 m/s
Mass = 0.0483293 kg, Analytical vt = 2.17741 m/s
Mass = 0.078476 kg, Analytical vt = 2.77462 m/s
Mass = 0.127427 kg, Analytical vt = 3.53562 m/s
Mass = 0.206914 kg, Analytical vt = 4.50536 m/s
Mass = 0.335982 kg, Analytical vt = 5.74106 m/s
Mass = 0.545559 kg, Analytical vt = 7.31569 m/s
Mass = 0.885867 kg, Analytical vt = 9.32221 m/s
Mass = 1.43845 kg, Analytical vt = 11.8791 m/s
Mass = 2.33572 kg, Analytical vt = 15.1372 m/s
Mass = 3.79269 kg, Analytical vt = 19.2889 m/s
Mass = 6.15848 kg, Analytical vt = 24.5794 m/s
Mass = 10 kg, Analytical vt = 31.3209 m/s
Info in <TCanvas::Print>: png file vt_vs_mass.png has been created
Plot saved as vt_vs_mass.png
```



Think about how you might evaluate the accuracy of your solution. Discuss any cross checks or evidence that the solutions with air resistance are reasonably accurate?

The numerical solutions with air resistance behave as physically expected. Velocities approach the analytical terminal velocity $v_t = \sqrt{mg/k}$, and reducing the RK4 step size produces consistent results, confirming solver stability and convergence. Mechanical energy decreases smoothly due to drag, with no unphysical jumps, and the drag force correctly opposes motion. Limiting cases—such as no air resistance or extreme masses—also produce the expected trajectories and scaling, providing further confidence in the accuracy of the results.