

VPython coding challenges for calculus-based physics

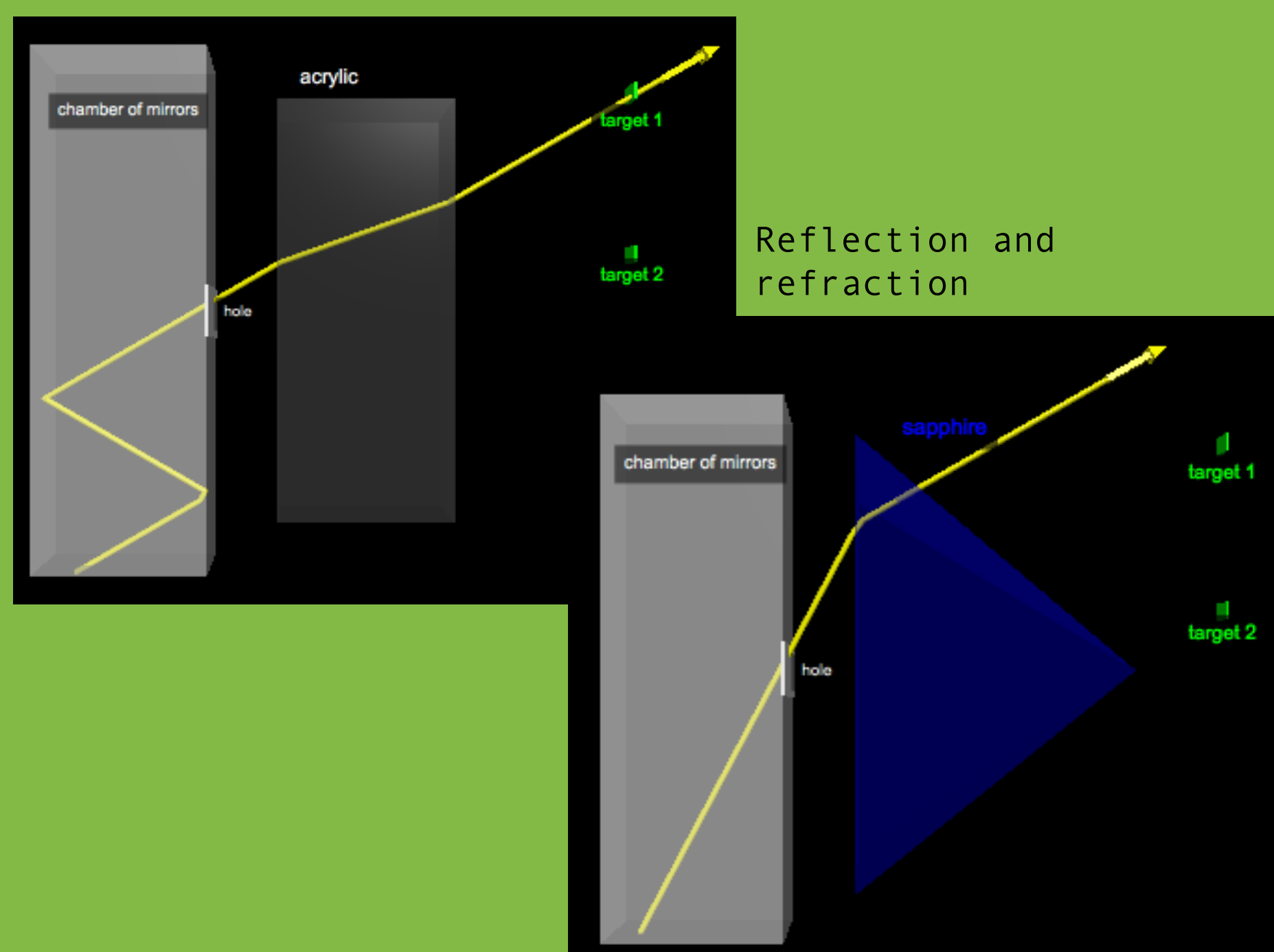


Glenda Denicolo
Suffolk County Community College, NY
denicog@sunysuffolk.edu

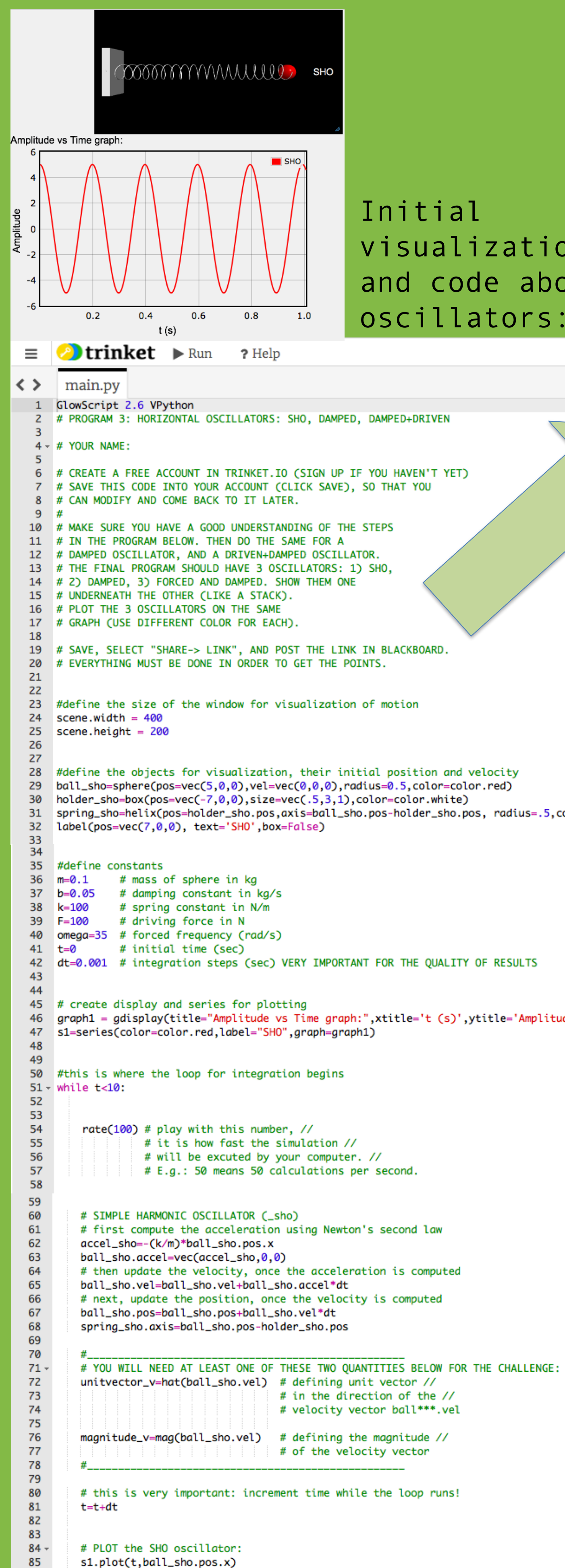
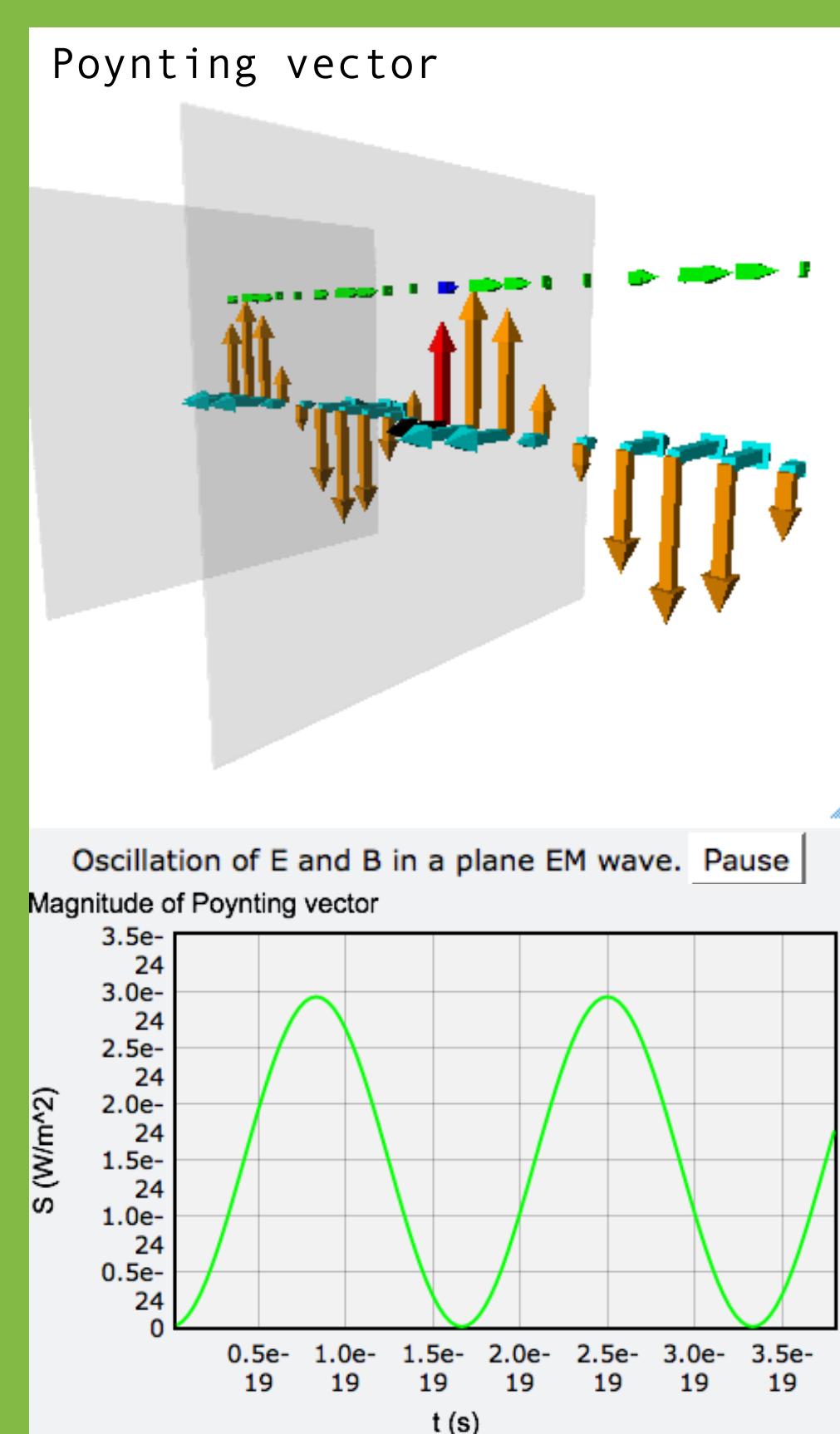
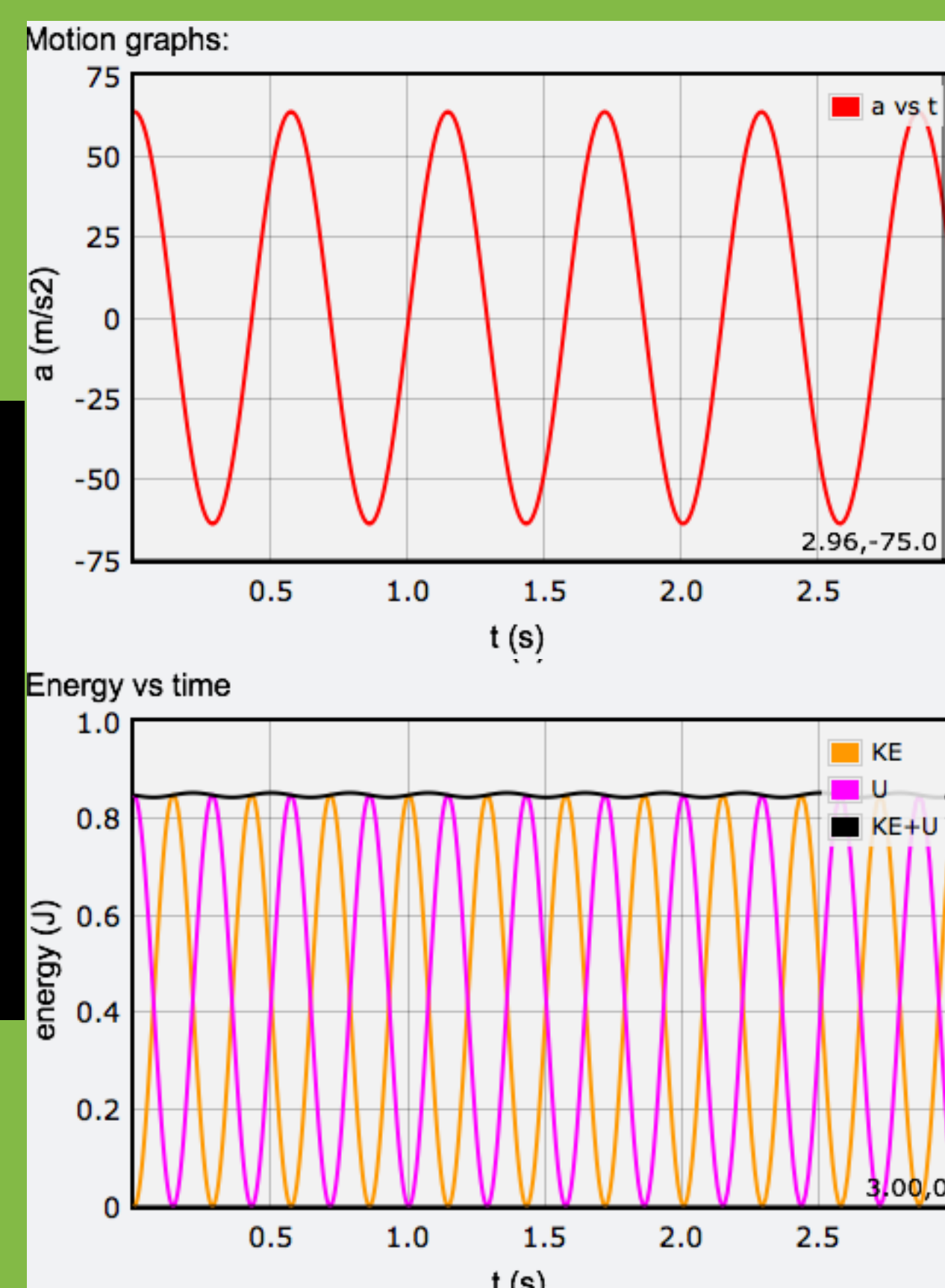
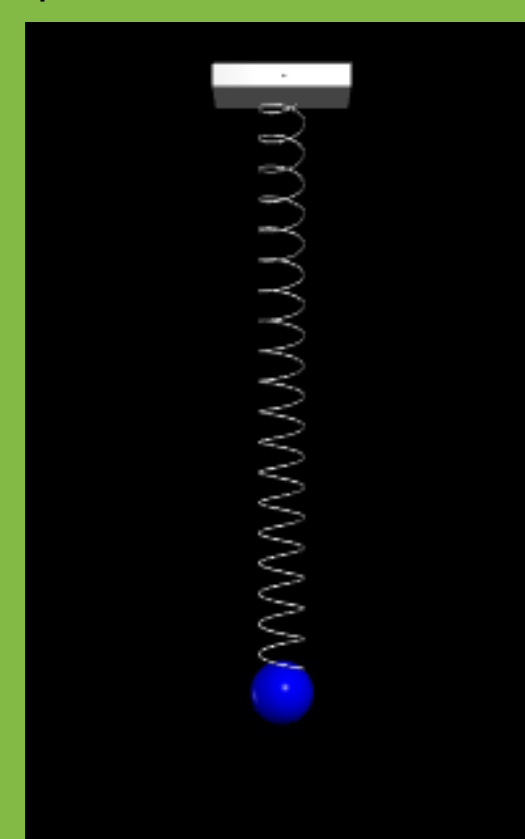
- The VPython coding challenges I use in my calculus-based physics classes are [available for public use](#).
- Codes are [minimal working examples](#) mostly written by me, but some are modified from Tom O'kuma, Dwain Desbien, Ruth Chabay, and Bruce Sherwood.
- The codes are deployed with [trinket.io](#) (no installation requirements), and are offered for extra credit. Little time is spent in class for instruction to these coding challenges, as they are designed to be self-explanatory. Selected online resources and documentation about VPython and Glowscript are provided to students.
- Sometimes only a few changes to the initial code are necessary. When more is required, students are guided by comments contained in the codes, which also have questions to be answered.
- Videos are provided to show a possible solution to each challenge.
- The **topics** are mostly in mechanics and include, so far:

- one-dimensional motion
- projectile motion at complementary angles
- Newton's second law with air drag
- examining elliptical versus circular orbits
- energy and momentum of a bouncing ball
- two-dimensional collision
- center of mass motion in a collision
- angular momentum conservation in orbital motion
- simple harmonic oscillator
- damped and driven oscillators
- superposition of waves

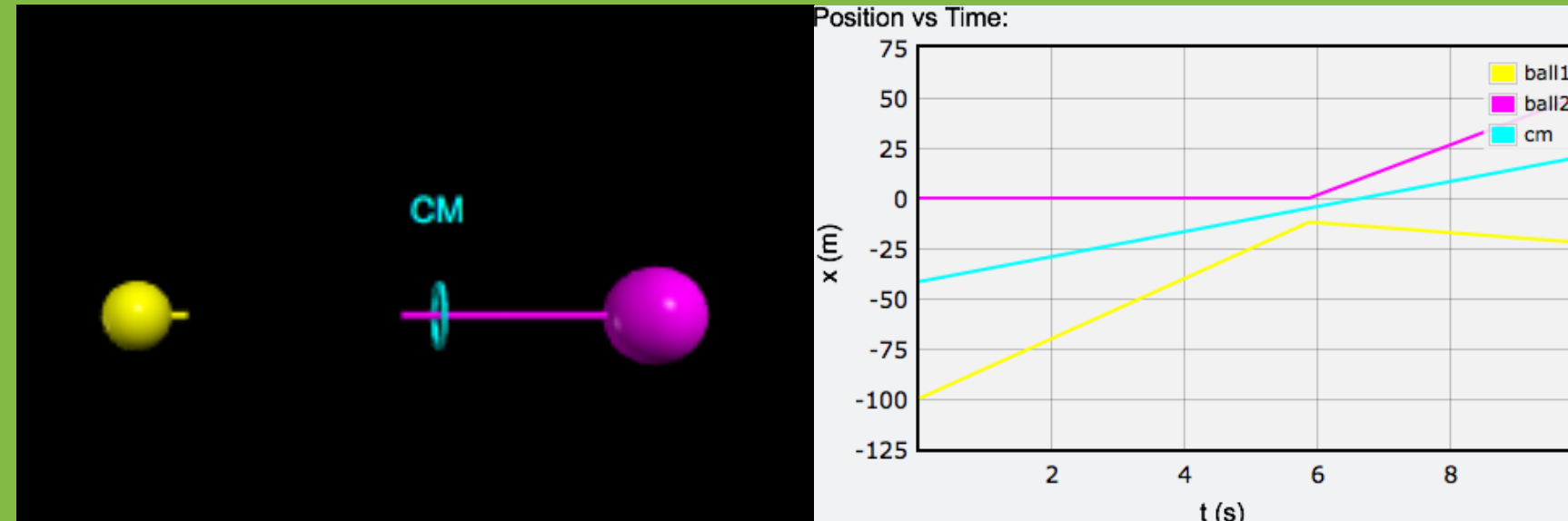
- The coding challenges are available at:
<http://gdenicolo.net/vpython.html>



Simple linear pendulum



Center of mass



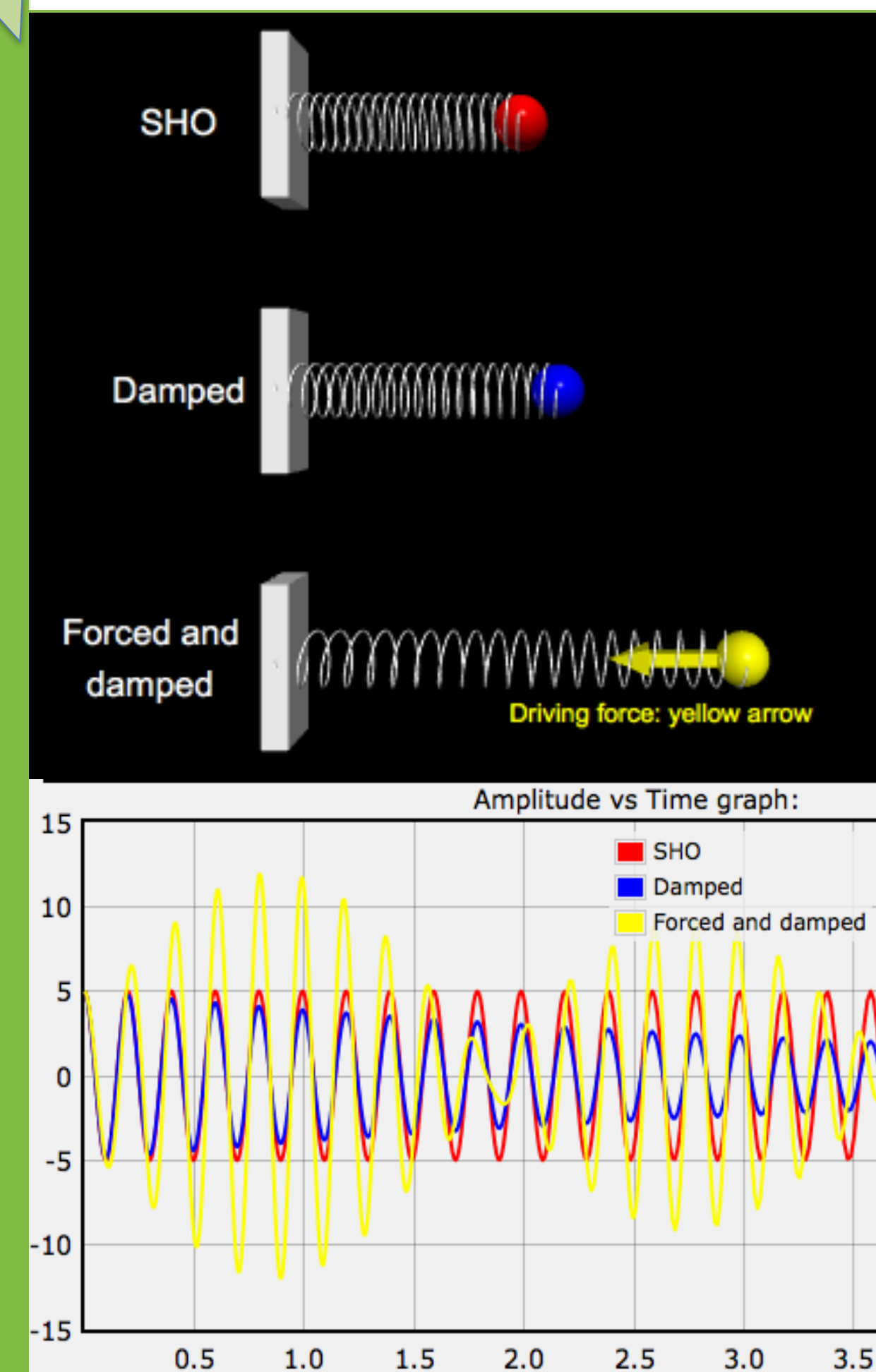
Example of final desired plot and visualization, and part of final code:

```
# SIMPLE HARMONIC OSCILLATOR (.sho)
unitvector_v=hat(ball_sho.vel) # defining unit vector //
                                # in the direction of the //
                                # velocity vector ball***.vel
magnitude_v=mag(ball_sho.vel) # defining the magnitude //
                                # of the velocity vector

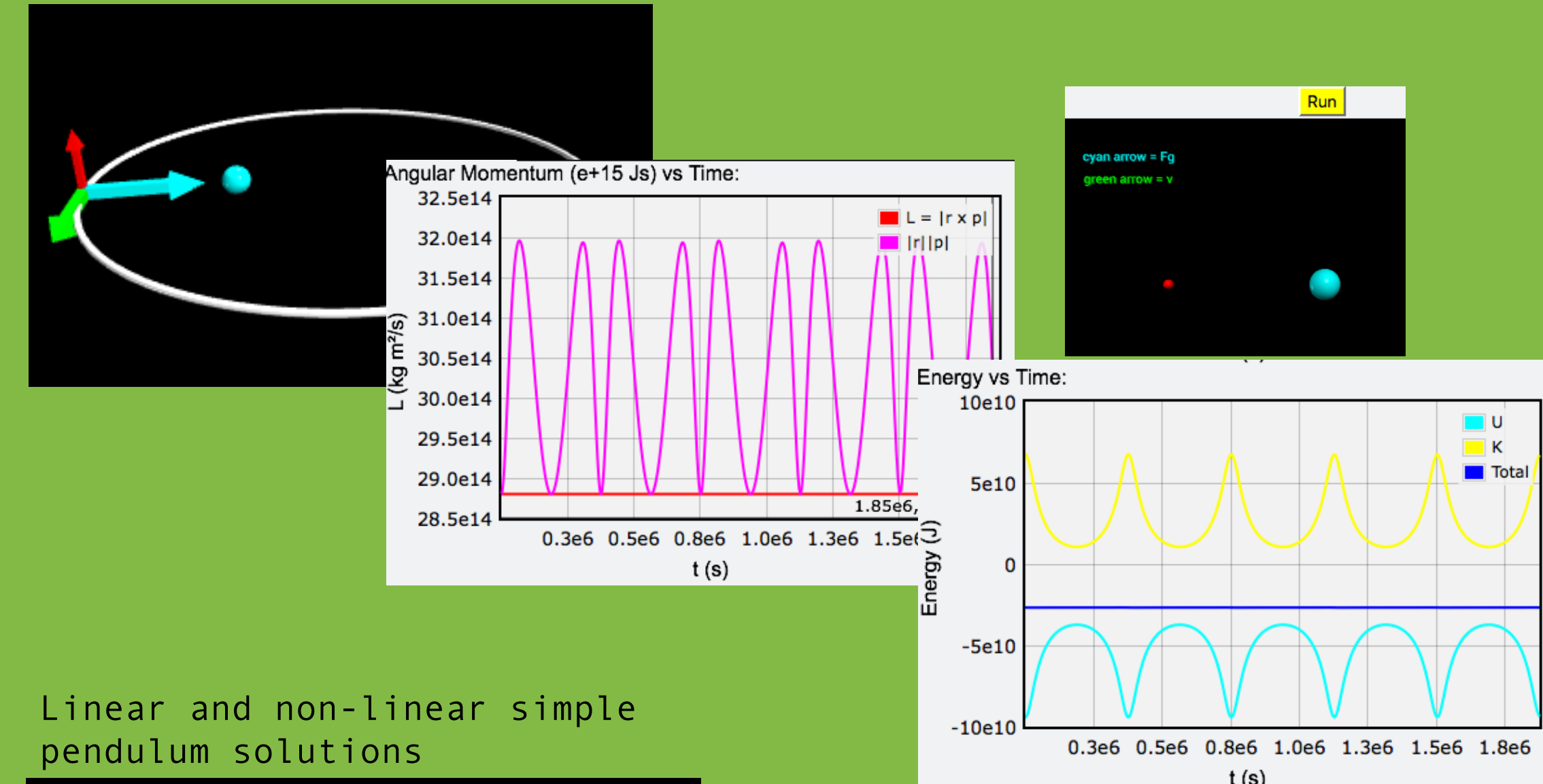
# first compute the acceleration using Newton's second law
accel_sho=(k/m)*ball_sho.pos.x
ball_sho.accel=vec(accel_sho,0,0)
# then update the velocity, once the acceleration is computed
ball_sho.vel=ball_sho.vel+ball_sho.accel*dt
# next, update the position, once the velocity is computed
ball_sho.pos=ball_sho.pos+ball_sho.vel*dt
spring.sho.axis=ball_sho.pos-holder_sho.pos

# DAMPED OSCILLATOR (.d)
unitvector_v=hat(ball_d.vel) # defining unit vector //
                                # in the direction of the //
                                # velocity vector ball***.vel
magnitude_v=mag(ball_d.vel) # defining the magnitude //
                                # of the velocity vector

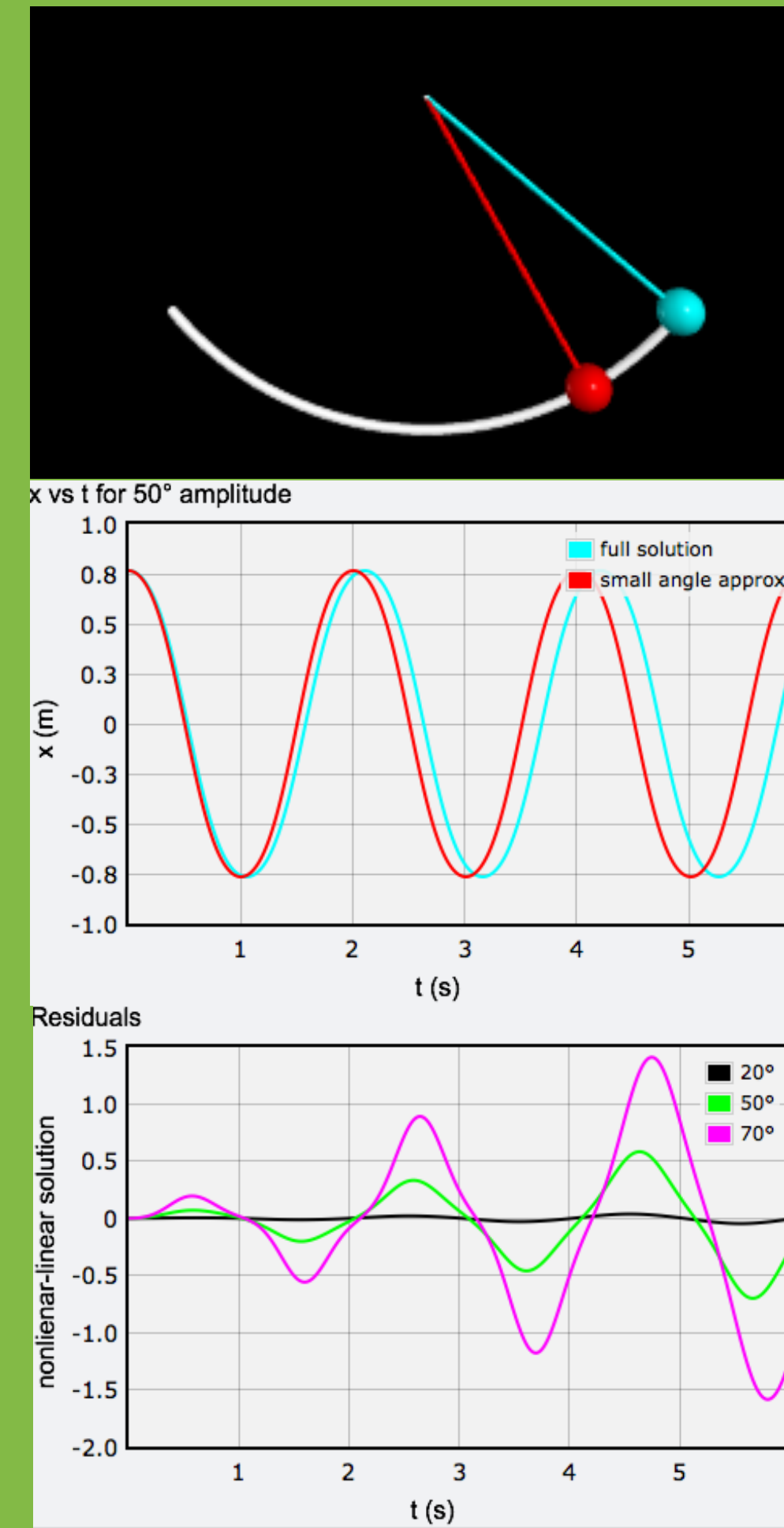
# first compute the acceleration using Newton's second law
accel_d=(b/m)*ball_d.vel.x+(k/m)*ball_d.pos.x
ball_d.accel=vec(accel_d,0,0)
# then update the velocity, once the acceleration is computed
ball_d.vel=ball_d.vel+ball_d.accel*dt
# next, update the position, once the velocity is computed
ball_d.pos=ball_d.pos+ball_d.vel*dt
spring.d.axis=ball_d.pos-holder_d.pos
```



Angular momentum



Linear and non-linear simple pendulum solutions



Wave superposition

