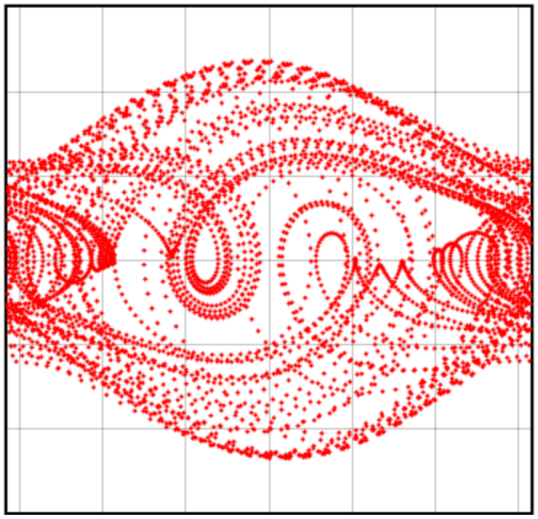
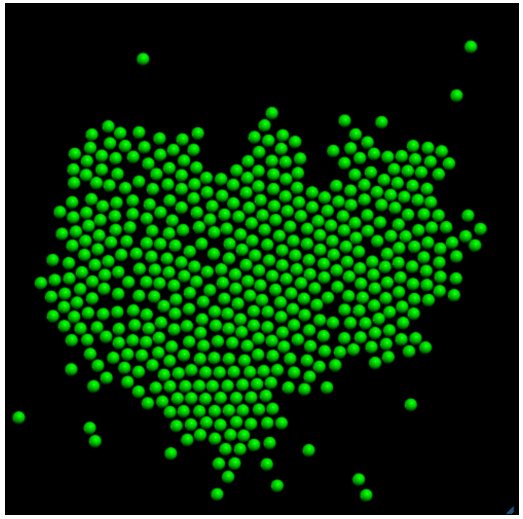
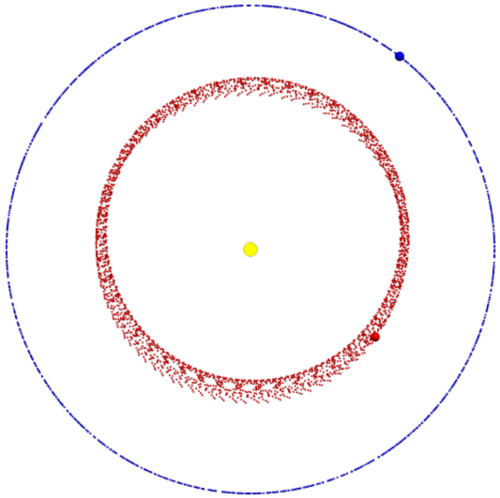
Scientific Computing for Physical Systems

Spring semester, 2018

Course Goals

- Learn a programming language (Python)
- Learn some numerical algorithms (e.g., for solving differential equations)
- Explore some interesting physics: nonlinear dynamics, chaos, celestial mechanics, many-particle systems, phase transformations
- Have fun

Course Format

- Lectures / textbook / homework / exams

Course Format

- ~~Lectures / textbook / homework / exams~~

- Labs / projects / mini-lectures / quizzes
  - 6 “canned” projects (66%)
  - Independent final project (20%)
  - Quizzes, attendance, etc. (14%)

- No required textbook (see reference list)

Projects

- Making shapes (VPython graphics)
- Projectile motion (1D and 2D)
- Pendulum (chaos!)
- Orbits (planets and asteroids)
- Molecular dynamics (500 particles at once)
- Random processes (statistical data)
- Independent project (with paper and talk)

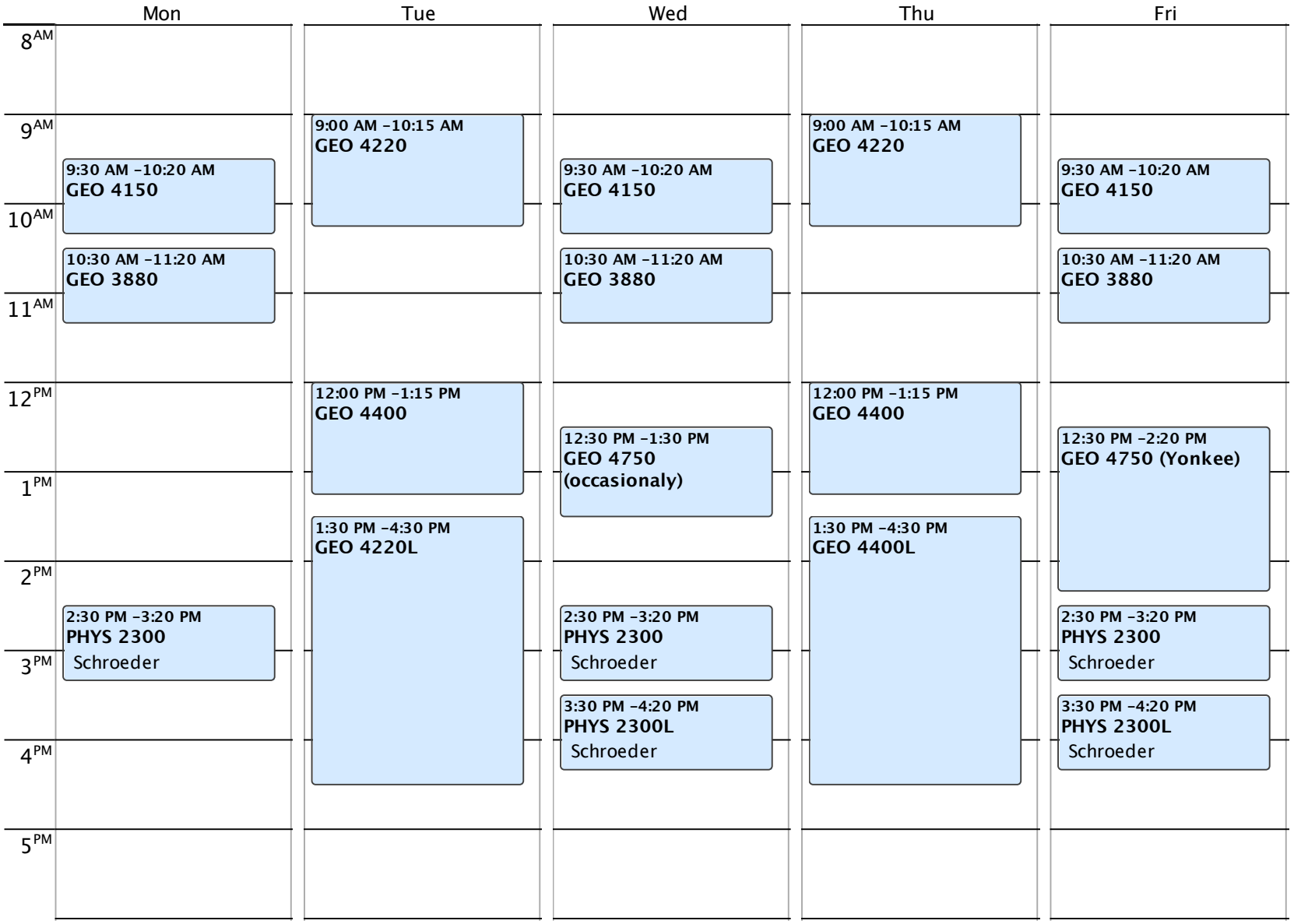
Remember: This is a lab course.

- Plan on spending plenty of time in this room.
- Coordinate work times with your lab partner.
- You're graded on what you *accomplish*—not on how smart you are.
- Procrastination isn't an option.

Upcoming events

- Fill out questionnaire before you leave today
- Continue work on Project 1 this Wednesday and Friday
- Project 1 due Wednesday, Jan. 17, at beginning of class (but finish by Friday if you can!)
- First quiz will be Wednesday, Jan. 17, during class (find errors/bugs in a short program)

TY 127 Schedule, Spring 2018



January 10

- My office: TY 322, up two flights and through Physics Dept. door, then to your right
- Course web page:

  `physics.weber.edu/schroeder/scicomp`
- Questions on course policies?
- Rules for collaboration and getting outside help
- Contact me privately if you have concerns about current or future lab partners.
- Questions about Project 1, GlowScript, VPython?
- Project 1 due 1/17, but try to finish by Friday

Project 1: Making Shapes

- print function
- box, sphere, cylinder
- Vectors
- Colors
- Variables
- Arithmetic
- while loop
- Animation
- Graphs
- Comments

Project 1: Making Shapes

- print function
- box, sphere, cylinder
- Vectors
- Colors
- Variables
- Arithmetic
- while loop
- Animation
- Graphs
- Comments

Features of the Python language itself

Project 1: Making Shapes

- print function
- box, sphere, cylinder
- Vectors
- Colors
- Variables
- Arithmetic
- while loop
- Animation
- Graphs
- Comments

Features of VPython

Project 1: Making Shapes

- print function
- box, sphere, cylinder
- Vectors
- Colors
- Variables
- Arithmetic
- while loop
- Animation
- Graphs
- Comments

Features of VPython

What about cos, sin, pi?

Project 2: Projectile Motion

- Your first simulation project!
- Solving differential equations (Newton's second law) using the Euler and Euler-Richardson algorithms
- Dealing with inaccuracies in calculations
- Physics: Air resistance, terminal speed, range
- if statements
- GUI controls ("widgets")
- Defining your own functions
- Boolean variables and constants
- Projectile1 program due Monday, January 22, 4:30 pm.
  Rest of project due Monday, January 29, 2:30 pm

Project 2: Projectile Motion

- Congratulations!
- Solving differential equations (Newton's second law) using the Euler and Euler-Richardson algorithms

```
# Euler algorithm:
ax = fx / m                 # fx could depend on x, vx, t
x += vx * dt                # Use old vx to increment x
vx += ax * dt               # Use old ax to increment vx
t += dt

# Euler-Richardson algorithm:
ax = fx / m
xmid = x + 0.5*vx*dt
vxmid = vx + 0.5*ax*dt
axmid = fxmid / m           # fxmid depends on xmid, vxmid, tmid
x += vxmid * dt             # Use vxmid to increment x
vx += axmid * dt            # Use axmid to increment vx
t += dt
```

Project 2: Projectile Motion

- Congratulations!
- Solving differential equations (Newton's second law) using the Euler and Euler-Richardson algorithms
- Interpolation

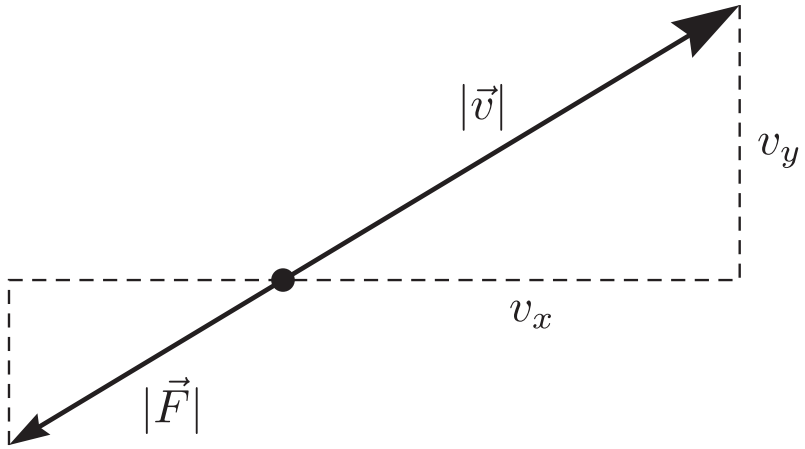
```
timeSinceLanding = y / vy
flightTime = t - timeSinceLanding
range = x - vx*timeSinceLanding
```

Project 2: Projectile Motion

- Congratulations!
- Solving differential equations (Newton's second law) using the Euler and Euler-Richardson algorithms
- Interpolation
- Truncation errors — 3 ways to estimate:
  - Compare to an exact calculation (when you can do one!)
  - Make dt smaller and see how much the results change
  - Monitor a conserved quantity (in next project)

Project 2: Projectile Motion

- Congratulations!
- Solving differential equations (Newton's second law) using the Euler and Euler-Richardson algorithms
- Interpolation
- Truncation errors
- Relating vector magnitudes to components

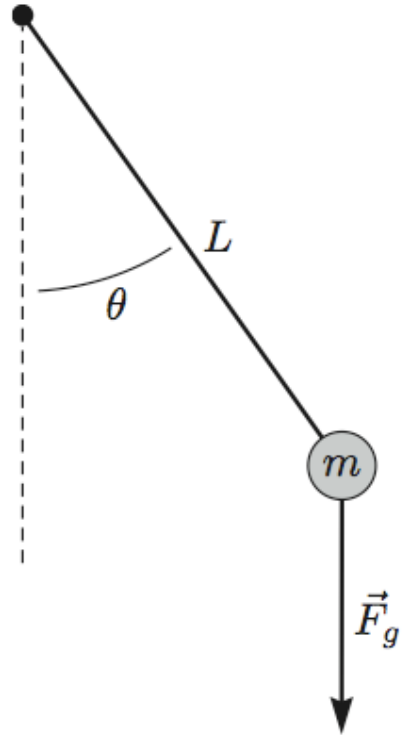


Project 2: Projectile Motion

- Congratulations!
- Solving differential equations (Newton's second law) using the Euler and Euler-Richardson algorithms
- Interpolation
- Truncation errors
- Relating vector magnitudes to components
- Physics results: Air resistance, terminal speed, range
- if statements, boolean variables and constants
- GUI controls (button, slider, wtext)
- Defining your own functions

Project 3: Pendulum

- More practice with Euler and Euler-Richardson algorithms
- This time the force depends on position, velocity, and time
- Angular variables: theta, omega, alpha
- Natural units ($m = g = L = 1$)
- Chaos!
- Pendulum1 program due next Monday, February 5, 2:30 pm.
- Finished project due Monday, February 12, 2:30 pm.
- Work with lab partners, but write your own separate code and turn it in separately.



Debugging Tips

- Everyone makes mistakes! Relax and don't feel guilty.
- Test your program as frequently as possible.
- Don't take error messages literally, but do check line number.
- Errors in a bound function don't generate messages! (Actually they do, but only in JavaScript console; try control-shift-J.)
- To diagnose logical errors, insert print() functions to display values of variables.

An ounce of prevention…

- Keep code clean and organized.
- Use comments as notes to yourself.
- Break up large tasks into smaller ones, each in its own function.

Project 3: Pendulum

- Pendulum1 modeled a freely swinging pendulum, measured period vs. amplitude.
- Now add “damping” and “driving” torques.
- Chaos!
- Pendulum2 will model *two* pendulums with slightly different starting conditions.
- Pendulum3 will let you vary the drive amplitude and observe phase space plots.
- Finished project due Monday, February 12, 2:30 pm.
- Work with lab partners, but write your own separate code and turn it in separately.