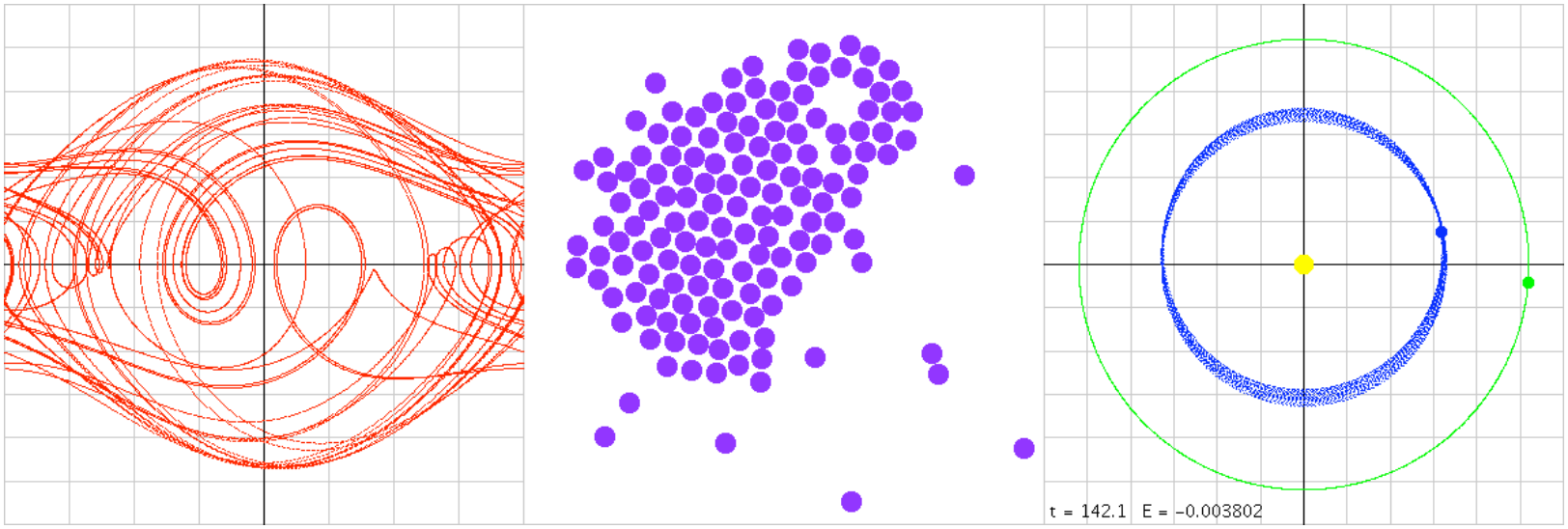
Computer Simulations of Physical Systems



Spring semester, 2006

Course Goals

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

Course Format

- Lectures / textbook / homework / exams

Course Format

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

- Labs / projects / mini-lectures / quizzes
  - 8 “canned” projects (70%)
  - Independent final project (20%)
  - Quizzes, attendance, etc. (10%)

Projects

- Hello, world!
- Range of a projectile (numbers and plots)
- Simulating projectile motion
- Pendulum motion and chaos
- Orbits
- Molecular dynamics
- Random processes
- Ising model of a ferromagnet
- Independent project

What You Need

- Lab manual and 3-ring binder
- USB backup device
- A computer; either…
  - Your own
  - SL 221 when intro labs not in session
  - SL 222, 225 when not in use
  - Physics majors room (2)
  - Science computer lab (8 in back, 1 Mac)

Warning!

This is an “experimental course”; I’ve never taught it before. You are guinea pigs! Some parts of the course might not go smoothly. I’m depending on you to tell me what works and what doesn’t.

By Thursday you should:

- Read lab manual preface
- Read course policies
- If using your own computer, get the necessary software installed and working
- Finish the “Hello, world!” project
- Prepare for our first quiz (find and correct bugs in a program similar to “Hello, world!”)

Let’s get started…

Comments on Quiz 1, Project 1

- Quiz: Good work!
- Be careful with capitalization (Hello, not hello)
- Use spaces, tabs, and new lines consistently
- Learn terminology (study glossary)

For Thursday, Jan. 19:

- Finish Project 2 (email code before class if possible)
- In your email, (1) tell me how much time you spent on the project and (2) describe your previous programming experience (if any), including courses, languages, work experience.
- Bring questions for beginning of class
- Turn in lab report before quiz, including printouts
- **Quiz:** Write a short program from scratch using a `main` method, constructor method, `System.out.println()`, math operations, trig functions, `if-else`, `while`, `for` structures
- No need to memorize `DecimalFormat`, parsing command-line arguments, plotting

Variable Scope

- **Local variables** are declared within a method, and available only within that method. They exist only as long as the method is being executed.

- **Instance variables** are declared at the class level (not within a method), and are accessible to any nonstatic method within the class (and sometimes outside the class). They are created when the class is instantiated (using `new`), and can hold different values for different instances of the class.

- **Class variables** are declared as `static` at the class level, and are accessible to all methods within the class (and sometimes outside the class). They are created only once, when the class is read into the computer's memory, and hold only a single value that is shared by all instances of the class.

Quiz Thursday, Jan. 26:

- Write a short program from scratch using a `main` method, constructor method, `System.out.println()`, math operations, trig functions, `if-else`, `while`, `for` structures, and plotting using the `Plot` class.
- No need to memorize `DecimalFormat`, parsing command-line arguments, Euler algorithm, etc.

Debugging Tips

- Test your program as frequently as possible.
- Write “dummy code” as a quick placeholder for testing, to be replaced later with something more complicated.
- When you get a compiler error, start with the first message and look mainly at the line number.
- To diagnose run-time errors, add println() statements to display additional information.

An ounce of prevention…

- Use consistent spacing and indentation.
- Type } immediately after typing {, then fill in what goes between.
- Use comments as notes to yourself.
- Use local variables when possible, instance/class variables when necessary.
- Break up large tasks into smaller ones, each in its own method.

Quiz Tuesday, Feb. 7:

- Write a short simulation program from scratch using the Euler algorithm, with text and/or graphical output.

- Physics scholarship applications are due today (application form in office).

- By today you should be about half way through Project 4. Due date will be Valentine's Day (next Tuesday).

- Next quiz (also on Feb. 14) will ask you to correctly recognize and identify Java language words, built-in class names, methods, local variables, instance variables, and class variables.

Truncation Error

- Euler algorithm is “linear” in *dt*; error in each step is proportional to $(dt)^2$; *total* error in a simulation is ordinarily proportional to *dt*.
- Euler-Richardson algorithm is accurate to order $(dt)^2$; error in each step is proportional to $(dt)^3$; *total* error in a simulation is ordinarily proportional to $(dt)^2$.
- With any algorithm, an accurate *interpolation* is often needed when the point of interest is between two of the calculated points.

More Java Features

- Integer (int, long) and boolean data types
- 3 kinds of methods:
  - void (main, paint, doStep)
  - non-void (Math.sin, DecimalFormat.format, torque)
  - constructors (Plot, Orbit2)
- Extending a class such as Canvas or Plot: all superclass methods are inherited, but can be overridden.
- Implementing an interface: requires that you provide certain methods (such as run for Runnable)
- Graphics primitives (drawLine, fillOval, drawString)
- Threads

Molecular Dynamics

- Force law (from Lennard-Jones potential)
- Natural units (molecular diameter, mass, interaction energy)
- Arrays
  - `double[] x; // declares an array of doubles`
  - `x = new double[N];  // creates the array`
  - `x[0]  // first element`
  - `x[N-1]  // last element`
  - `for (int i=0; i<N; i++) { x[i] = i*i; }`
    `// typical loop`
- Quiz Tuesday (3/7) : Write a short program to create and manipulate arrays. To prepare, you should have worked through page 77.

Extra “Lab” Hours:
Friday and Monday, 1:00 - 4:00

Molecular Dynamics

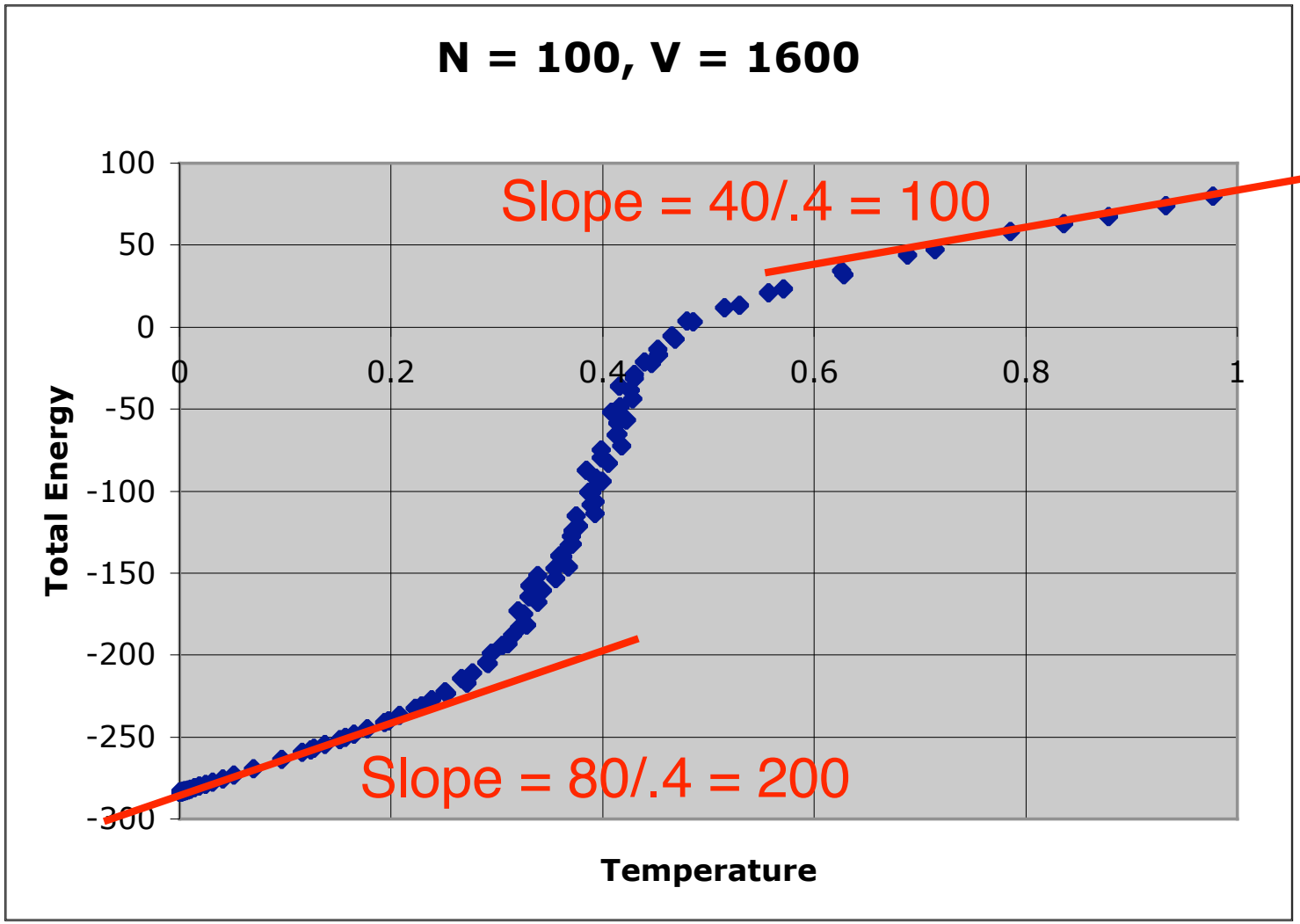
- Force law
- Natural units
- Arrays

- "Top-Down" design:
  - extends Canvas, implement Runnable
  - Write the constructor, run, paint, and main methods.
  - Fill in details later (doStep, computeAccelerations, GUI, data output).
  - Test at every opportunity, even if it means temporarily adding "dummy" code to be removed later.
  - Don't try to optimize performance until after the code is working.
  - Doesn't use the Plot class, but you will plot data in a spreadsheet.

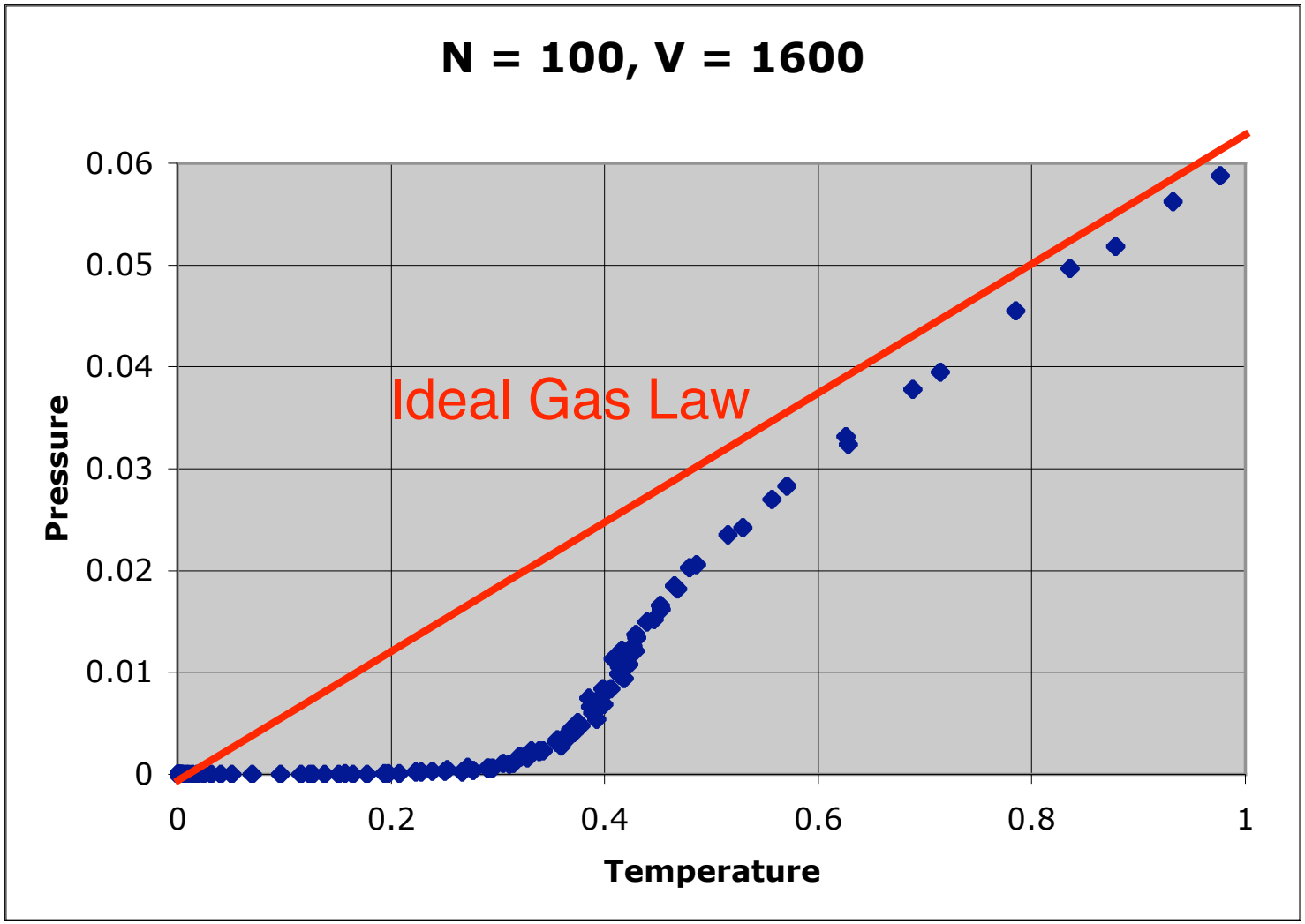
Extra “Lab” Hours:
Friday and Monday, 1:00 - 4:00

MD project is due Tuesday,
March 21, 5:00 pm

MD Data


N = 100, V = 1600
Slope = 40/.4 = 100
Slope = 80/.4 = 200
Total Energy
Temperature
100
50
0
-50
-100
-150
-200
-250
-300
0
0.2
0.4
0.6
0.8
1

MD Data


N = 100, V = 1600
Ideal Gas Law
Pressure
Temperature
0.06
0.05
0.04
0.03
0.02
0.01
0
0
0.2
0.4
0.6
0.8
1

Project 7: Random Processes

- Math.random() returns “random” double between 0 and 1.
- To get a random integer from 0 to n–1, multiply by n and round down.
- First program: RandomTest (prints and plots random numbers)
- Second program: TwoBoxes (randomly “moves” molecules from one box to another and back)
  - n = total number of molecules
  - nLeft = number in left-hand box
  - n – nLeft = number in right-hand box
  - Algorithm: Generate a random integer from 0 to n–1. If it’s less than nLeft, decrease nLeft by 1; otherwise increase nLeft by 1. Repeat and plot nLeft vs. “time”.

So you like computational physics? Then consider a long-term project…

- Beishline Fellowship
  - $12,000
  - Full-time summer job plus part-time job during academic year
  - Awarded for a computational science project with faculty supervisor
  - Applications due April 7
- Projects for pay, funded through faculty grants
- Projects for academic credit (Physics 4800, 4830)

Now that this course is ending, What next?

- Computer languages
- Numerical methods
- Physics

Computer Languages: More Java

- Language features: do-while, switch-case, break, continue, %, <<, >>, &, |, ?:, byte, short, float, char, …
- 3000 more built-in classes: more GUI features, file input/output, database interaction, arbitrary-precision arithmetic, other random number generators, image processing, applets, networking
- Add-on packages such as Open Source Physics

Other Computer Languages

- C, C++, Fortran
- Basic
- Python
- Mathematica, Maple, Matlab
- Scripting languages
- Graphics and web languages

Numerical methods

- This course
  - Solving systems of ordinary differential equations
  - Monte Carlo
  - linear interpolation

- All of these algorithms can be enhanced, extended, et cetera. (But fancier isn't always more accurate!)

- Other numerical tasks
  - Evaluating special functions
  - Root finding
  - Maximization / minimization
  - Numerical integration
  - Curve fitting
  - Matrix manipulation, diagonalization
  - Partial differential equations, Fourier transforms

Computational Physics

- This course
  - Mechanical systems
  - Chaotic dynamics
  - Many-particle systems
  - Phase transformations

- You can explore all these subjects much further, with or without computers!

Computational Physics

- Computation is now part of every subfield of physics:
  - Electromagnetic fields
  - Wave phenomena
  - Quantum mechanics (e.g., Schrödinger equation)
  - Solid state physics
  - Nuclear physics
  - Astrophysics (stellar structure, galactic dynamics, gravitational waves)
  - Elementary particle physics (Monte Carlo prediction of event rates at colliders, lattice gauge theory)