$1/R^2$ Force Netlogo Simulation – Scott Mayle

Purpose

The $1/R^2$ force law is prevalent throughout physics and is responsible for many of the structures that make up the universe (eg atoms and solar systems). Unfortunately the mathematics required to show students how these structures form from the force equations is often too complicated. An alternative to using a mathematical derivation is a computer simulation. By having students program a simulation of particles simply interacting with a $1/R^2$ force, they can see the conditions required to form orbits without the differential equations that are usually necessary. This also allow students to see the benefits and drawbacks of using a computer to model real world systems.

Overview

This lesson is broken into four parts (note: depending on the students prior familiarity with netlogo, the first part of the lesson is not necessary).

The first part of the lesson is to talk about the simulation program/language netlogo. I designed a simple program that demonstrates different relationships between the code structure of netlogo and resulting simulation. Using this demo program, I give a brief introduction in front of the whole class as to how the simulator works, how the language and code structure works, and some basic commands required to program in netlogo.

The second part of the lesson is to get the students working with the simulator and to get the students to read and start understanding the code of a simulator. To do this students are provided with a simplified version of a simulator they will edit (so they do not need to start programming from scratch). The students then must figure out what the initial simulator does, and then answer questions from a worksheet about the code they are provided with.

The third part of the lesson is where students will turn the simulation they are given into a simulation that can produce orbits and bound structures such as atoms or planetary system. The students will follow instructions to edit the code of the simulation they are given to turn it into a useful simulator.

Finally, in the fourth part of the lesson the students will use the simulation they created to answer questions about the physics of bound orbits.

Student Outcomes

Students should know what netlogo is and how to use simulations that are programmed in netlogo.

Students should be able to read and understand basic netlogo code, understanding the importance of the architecture of the code and the basic commands in the netlogo language.

Students should be able to edit the code of a simulation in netlogo and run the code to determine if the desired changes were produced.

Students should then be able to use the netlogo simulation to see how the force law affects the dynamics of the simulated particles.

Students should understand how to use simulations in science to explore systems and changes that would not be easily done in the real world.

Illinois state standards lesson applies to:
IL SS 11.B.5a -11.B.5f
IL SS 12.D.5a &12.D.5b
IL SS 13.A.5b & 13.B.5c

Time
2 class periods of approximately 45 minutes.

Level
Senior or Junior level high school students in physics who have completed sections either on the Coulomb force or the universal gravitational force.

Materials and Tools
- Computers
- Netlogo (Free simulation software)
- Simulations for students to edit

Preparation
Make sure netlogo is installed and runs properly on the computer the students will be work on. Also distribute the simulator the students will be editing on each computer as well, or have it accessible in some central location students can download from.

Prerequisites
N/A

Background
Students should have background knowledge of the $1/R^2$ force they will be simulating and knowledge of the systems these forces produce.

Teaching Notes
This lesson was designed to combine a common topic in high school physics with the computational thinking skills gained from working with simulations and programming. The first part of the lesson is to get the students familiar with the computer program netlogo. Netlogo is a

modeling program and language used primarily for educational purposes. To give students with no previous exposure to netlogo an introduction, I show them a demonstration program. I start by showing students the basic mechanisms needed to start and run the simulators in netlogo. After the students have seen how to run a simulator in netlogo, I then show them how to access the code of the simulators. Finally I talk the students through the code of my demo simulator and show them how each piece of code corresponds to a specific action in the simulator and that a change in the code subsequently changes the actions of the simulator. At this point hopefully the students can sit at a computer and use a netlogo simulation and also be able to recognize the necessary structure to the code and some basic commands in the code.

The next part is with the students in front of computers. This allows the students to use the knowledge they have hopefully gained to start up and use the simulation they are provided with. The simulation they are provided with is a simple program that incorporates the $1/R^2$ force between two objects and then calculates and updates the movement of these particles due to the force. The two particles that interact are initially assigned random positions and no initial velocities. This will cause the two particles to simply start moving towards each other from rest and collide. The students then must go through the code of this simulation and answer questions provided to them via a worksheet. If the students cannot figure out what a particular command in netlogo does, it is best to refer them to the code dictionary on the netlogo webpage and have them look up the command.

Once students have played with the simulator, read through the code, and answered the appropriate questions, they will start to edit the code. Students will follow the instructions (also on the worksheet) to edit the code of the simulator they are given through a series of steps to eventually make the simulator produce and represent orbits. Since some students might have previous coding experience it is best to have some challenges available to keep all student busy.

Finally, having followed the instructions to edit the code, the students should have a working simulator that allows a user to specify the initial positions, charges, and velocities of the particles. This allows students to play with the input parameters and create a variety of different behaviors including orbits. The students will then answer question about which initial conditions create what behaviors and in general play with a dynamic system that cannot mathematically be understood without differential equations. As an added exercise, you can also make the exponent on the force law, a user input value. This will allow students to play with the system if we didn't have a $1/R^2$ law, but other exponential dependences. This allows us to see how the universe behaves by changing something that cannot be done experimentally.

Assessment

During the first portion of the activity, students will be assessed informally. As this is just the introduction material, students should be asked questions along side with the lecture. These questions again should be aimed to see if they understand how netlogo works (Which button do I always need to press before running my simulation?), if they understand the code infrastructure (If I move this line of code from the *to go* procedure to the *to set-up* procedure how will this change how the simulator acts?), and if they understand some basic commands (How do I change

the shape of particle one to a square?). The students will also be formally assessed on what they pick up in this section in the next section.

In the second part of the lesson, assessment is primarily taken care of on the attached worksheet. This sheet asks students to go through the code of the provided simulator and answer questions regarding the codes structure and commands. It is also helpful to informally assess students as they go through the worksheet. This allows you to gauge what they learned in the first part of the lesson, questions they had for you, and allows you to gauge how well they are going through the code (you can ask them additional questions about the code).

In the third part, formal assessment will primarily be the final simulator they create. This program can be saved and submitted to a common location, or the code for the simulators they create can be printed out and evaluated.

Finally the fourth part of the lesson will again be formally assessed using the worksheet. The worksheet prompts students to think about the physics of the simulators and allows students to use the simulator they created to find answers to questions that would normally require differential equations.

Additional Information

All attached documents have been customized to work with the universal law of gravitation. This lesson can be repeated for the Coulomb force or any other $1/R^2$ force.
First attached is the worksheet that guides the activity and poses questions.
Second is the code for the netlogo simulation that the students receive.
Last is an example code for the how the simulation should look after the activity.

Intro to Netlogo and a Gravity Simulation

Name:

Find code dictionary here: http://ccl.northwestern.edu/netlogo/docs/

Part 1: Understanding the Code
How many types of agents are there?

What are they?

Including GO and Setup how many procedures are there?

In what procedure does the force between the two masses get calculated?

What approximation is made when moving the particles?

In the procedure “update-force” explain what the command “face one-of centers” does.

The variable r is defined as a global variable, what procedures is it used in and how is it calculated?

Part 2: Editing the code

Position of moving particle

Instead of the test (movable) mass being generated at a random location in the “world”, we now want to be able to specify where it starts. To begin, we must first delete the command that creates the particle at a random location find the line with the following code and delete it:

```
fd random-float (max-pxcor - 6)
```

Since netlogo is primarily a visual modeling language, the x y locations of agents is already a well defined variable built into the language. We can edit the pre existing variables by using the command: “setxy x y”. This command sets the location of the agent, an example of this is we wanted to set the particle to the position of (-15 , 12) we would use

```
setxy -15 12
```

We cannot simply use this piece of code where ever we want. Since we are setting the position of the movable mass (or what this simulation calls a particle, not a center), we need to make sure we are setting its position and nothing else. Luckily, since this is an agent based language this is rather easy. We could simply just ask the particle to move to our desired location:

```
ask particles [
setxy -15 12
]
```

With this in mind we need to be conscientious of where in the program we are asking the particle to set its location and where this would be best served. Do we want this location to be set once at the start, or do we want the location to be set to the specified location over and over again?

If placed correctly in the code, asking the particle to set its location will do the job, but I think an easier and cleaner way to write this into the code is to set the particle’s location when the particle is created.

If you find where the particle is created you can simply add the

```
setxy -15 12
```

command to the list of other properties that are set at this time.

Now that we know how to change the location of the particle by editing the code, we want to be able to do this without having to recode our simulator every time. This means that we want to create a user defined variable in the simulator. On the Interface tab we can right click in any of the empty white space to bring a list of user input options. I would recommend using either a slider or an input option.

A slider defines a variable that is set by the value the slider is slide to. To use a slider you must define a variable, a maximum value, a minimum value, and an increment.

Now that we can have the user define the value of a variable using a slider we can use the variable to set the x and y position of the particle. This can be done as the arguments of the setxy command do not need to be numbers but can also be variables. For example if we had a user define two variables “favoritenumber” and “age” we could set the particle’s location to:

setxy favoritenumber age

So in my case this would set the particles position to (24,26). But someone else will have a different favorite number and age, setting their particle to a different location without having to adjust the code.

Please note you do not need to call your variables favoritenumber and age, it is usually best to use something that represents the variable you are controlling, like x-particle for the x position of the particle.

Masses

Thus far we have changed the location of the particles. In the original simulator we have a slider to determine the values of both mass simultaneously. The next improvement to the simulator should be allowing the two masses to be set independently of each other. No special commands should be necessary, however one must set another new user defined variable and look at the code carefully to make the necessary adjustments.

Initial Velocities

Now that we can control the positions and masses of each of the agents we want to make things more interesting than having the two masses fly towards each other. To do this we will need to give the movable mass some initial velocity. Note there are already variables that exist for the x and y velocity of the particle (vx and vy), we just need to figure out how to set their initial values. To do this you will need to create two new user input variables (initial x velocity and intial y velocity) and set the particles velocity (note only at the start) to these user input variables. Be very careful where you set the velocities equal to the initial velocities, as we only want to change the initial velocity once at the start and not affect how to velocity is calculated later in the code.

Exponential Law's

Last but not least, to get to some really interesting physics, we want to be able to change the type of force experience by the two masses. It is well known that the Gravity Force is a 1/r^2 force, but what if it was not? What if it was 1/r or 1/r^3? Usually these are very complicated problems requiring differential equations to solve, however instead of solving the problem lets model it!

Part 3: Using the Simulator

Find 2 configurations that makes the moving mass orbit the stationary mass. List all used parameters and describe the shape of the orbit and if the orbit is stable (the orbit stays in the same place) or unstable (the orbit itself precesses around the stationary mass).

Find 2 configurations that give you a nearly perfect circular orbit. Again list all used parameters. Can you get a circular orbits in both stable and unstable configurations?

Repeat the previous 4 configurations now using a 1/r law instead of a 1/r^2 law. Describe how each of these differed from the original. Can we make any generalizations about the relative strengths between a 1/r versus 1/r^2 law?

Repeat the previous 4 configurations now using a 1/r^3 law instead of a 1/r^2 law. Describe how each of these differed from the original. Can we make any generalizations about the relative strengths between a 1/r^3 versus 1/r^2 law?

Code for simulation given to Students

```
breed [centers center]
breed [particles particle]

particles-own [
  fx     ;; x-component of force vector
  fy     ;; y-component of force vector
  vx     ;; x-component of velocity vector
  vy     ;; y-component of velocity vector
]

globals [
  r         ;; distance between particle and center
  potential  ;; potential energy of particle
  force      ;; modulus of force vector
  previous-permittivity ;; the last recorded value of permittivity
  c-mass   ;; mass of the center particle
]

;;;;;;;;;;;;;;;;;;;;;;;;
;;; Setup Procedures ;;;
;;;;;;;;;;;;;;;;;;;;;;;;

to setup
  clear-all
  set-default-shape turtles "circle"

  ask patches [ set pcolor 77 ]

  set c-mass mass

  create-centers 1 [
    set size 10
    set color gray
    set label c-mass
    setxy 0 0
  ]

  create-particles 1 [
    set color blue
    set size 10
```

```
    set label mass
    fd random-float (max-pxcor - 6)
  ]

  reset-ticks

end

;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Runtime Procedures ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;

to go

    every 0.1
    [
      ask particles [
        update-force
        move
      ]
      ask particles [ calculate-potential-energy ]

      tick
    ]

end

to update-force  ;; particle procedure
  set r distance one-of centers
  ;; if r = 0, then it's at the mass, and then the model stops;
  ;; prevents divide by zero
  if r = 0 [ die ]
  ;; Calculate force using inverse square law
  set force ( mass) * c-mass / (r ^ 2)
  ;; Separate force into x and y components
  face one-of centers
  set fx force * dx
  set fy force * dy
end

to move  ;; particle procedure
  ;; update the particle's velocity, by taking the old velocity and
  ;; adding the force to it.
  set vx vx + (fx / mass)
  set vy vy + (fy / mass)
  ;; disappear if we reached the edge
10
```

```
  if patch-at vx vy = nobody [ die ]
  ;; update the particle's position
  setxy (xcor + vx) (ycor + vy)
  ;; leave a trail
  set pcolor 72
end

to calculate-potential-energy  ;; particle procedure
  set r distance one-of centers
  set potential mass / r
end

;;;;;;;;;;;;;;;;;;;;;;;;
;;; Visual Procedures ;;;
;;;;;;;;;;;;;;;;;;;;;;;;

to-report background
  report 77
end



; Copyright 2005 Pratim Sengupta and Uri Wilensky.
; See Info tab for full copyright and license.
```

Example Code for End result

```
breed [centers center]
breed [particles particle]
breed [particles2 particle2]

particles-own [
  fx     ;; x-component of force vector
  fy     ;; y-component of force vector
  vx     ;; x-component of velocity vector
  vy     ;; y-component of velocity vector
]

globals [
  r         ;; distance between particle and center
  potential  ;; potential energy of particle
  force      ;; modulus of force vector

]

;;;;;;;;;;;;;;;;;;;;;;;;
;;; Setup Procedures ;;;
;;;;;;;;;;;;;;;;;;;;;;;;

to setup
  clear-all
  set-default-shape turtles "circle"
  ask patches [ set pcolor background ]
  create-centers 1 [
    set size 10
    set color gray
  ]

  create-particles 1 [
    set color blue
    set size 10
    ;; Set initial velocity of Mass 2
    set vx Vix
    set vy Viy
    ;; Set Position of Mass 2
    setxy x1 y1
  ]
```

```
 ;; label the particles with the appropriate Mass
 ask centers
    [
     setxy x2 y2
     set label mass2
    ]



 ask particles [ set label mass1
  ]

 reset-ticks
end

;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Runtime Procedures ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;

to go


  ask centers [ show-turtle ]
  every 0.1
  [

    ask particles [
     update-force
     move
    ]
    ask particles [ calculate-potential-energy ]

    tick
  ]

end

to update-force  ;; particle procedure
 set r distance one-of centers
 ;; if r = 0, then it's at the mass, and then the model stops;
 ;; prevents divide by zero
 if r = 0 [ die ]
 ;; Calculate force using inverse power law
 set force  ( mass1) * mass2 / (r ^ exponent)
 ;; Separate force into x and y components
```

```
  face one-of centers
  set fx force * dx
  set fy force * dy
end

to move  ;; particle procedure
  ;; update the particle's velocity, by taking the old velocity and
  ;; adding the force to it.
  set vx vx + (fx / mass1)
  set vy vy + (fy / mass1)
  ;; disappear if we reached the edge
  if patch-at vx vy = nobody [ die ]
  ;; update the particle's position
  setxy (xcor + vx) (ycor + vy)
  ;; leave a trail
  set pcolor background - 5
end

to calculate-potential-energy  ;; particle procedure
  set r distance one-of centers
  set potential mass1 / r
end

;;;;;;;;;;;;;;;;;;;;;;;;
;;; Visual Procedures ;;;
;;;;;;;;;;;;;;;;;;;;;;;;

to-report background
  report 77
end



; Copyright 2005 Pratim Sengupta and Uri Wilensky.
; See Info tab for full copyright and license.
```