Concerning the Expansion of the Universe

New Mexico
Supercomputing Challenge
Final Report
April 2, 2014

Team #16
Centennial High School

Team Members:

Devon Miller

Bryce Austin

Teacher:

Ms. Hagaman

Project Mentor:

Dr. Charles Miller PhD

EXECUTIVE SUMMARY

With the universe expanding at a precariously fast rate and with galaxies apparently flying at extreme speeds away from us because of red shifts, it seems unsurprising that the topic is receiving lots of wide-spread acclaim. Astronomers and cosmologists, all over the world, scramble to decipher the meticulous details of our ever-growing universe. For example, N-Body simulations have been invented in order to simulate the universe and how it ages over very extensive periods of time. Many simulate the clumping of galaxies and super clusters as they drift around the universe due to gravity, others simulate how bodies orbit one another in various types of scenarios, and some simulate the collisions of galaxies, showing all of the various particles swirling and melding with one another. Our interest is in the simulations that show the expansion of the universe as particles are sent outwards from a single origin point. Plenty of these simulations exist, however would it be possible to code a homemade model of this simulation. In a simple program like NetLogo code can be extensive but fairly manageable and much lower-grade than cosmological simulators.

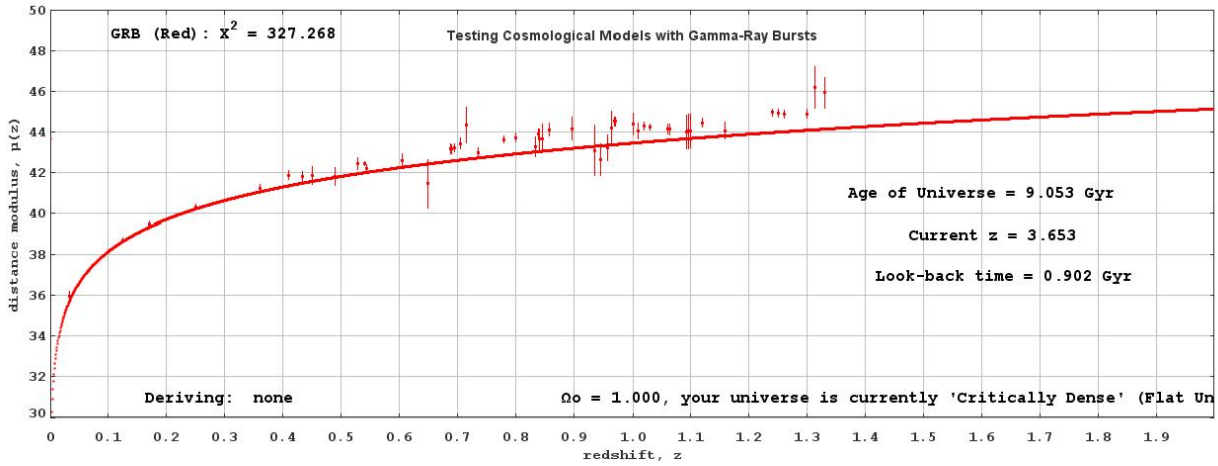
But what if a complex N-Body simulation could be translated and converted into NetLogo in a very simplified state. Not only would this mean that many people could learn easily about the aspects of the expanding universe but also it would demonstrate to many how the mechanics of universal expansion behave. My partner and I went out to attempt this and we spent many weeks beforehand performing extensive research on cosmological topics like Hubble's Law. We then coded our variables, like gravity and force, into NetLogo by building off of similar simulations previously made by Dr. Charles Miller. This step proved to be the most tedious; my partner and I found it very overwhelming and difficult to code the variable of gravity and make the simulation run seamlessly. After plenty of tweaking our simulation emerged intact and efficient. Our results proved to be fairly accurate compared with other high-grade simulations and it appeared that our ultimate goal of simulating a basic N-Body model with simplified command structure proved to be successful.

Our goals for the future of this project are to hopefully increase the accuracy of the simulation by including more variables and, if all goes well, to hopefully translate this astronomical model in NetLogo 3D. By doing this we could greatly improve the realism of the simulation and we could perceive a much more in-depth and meticulous reflection of universal expansion. Obviously coding into three dimensions would be much trickier than two but if we were to succeed we could have access to complex cosmological data with little trouble.

INTRODUCTION

The main goal my partner and I had for finding a realistically applicable scenario capable of being computationally modeled was to find a topic dealing with astrophysics. We were both interested in tackling a unique and complicated problem and having Dr. Charles Miller, PhD in astrophysics and father of group member Devon Miller, for assistance proved to be a great advantage for research material and coding. Originally our topic was very broad, being vaguely expressed as "determining the expansion of the universe." After a long period of time involving research on the topic and planning on how to map this problem we eventually came across the idea of "N-Body Simulations" after discovering various examples of these computational models. Our goal was then to see if we could somehow map a basic N-Body Simulation into a very simple program like NetLogo and to figure out if we could initialize the variables of velocity, gravity, mass, and distance in order to calculate similar results to that of real N-Body simulations with a focus on universal expansion. We decided, building off the topic of universal expansion, to instead map a simulation that showed the slowing down of acceleration between celestial bodies over time from the Big Bang rather than map the classical models of galactic, planetary, and interstellar interaction and collision.

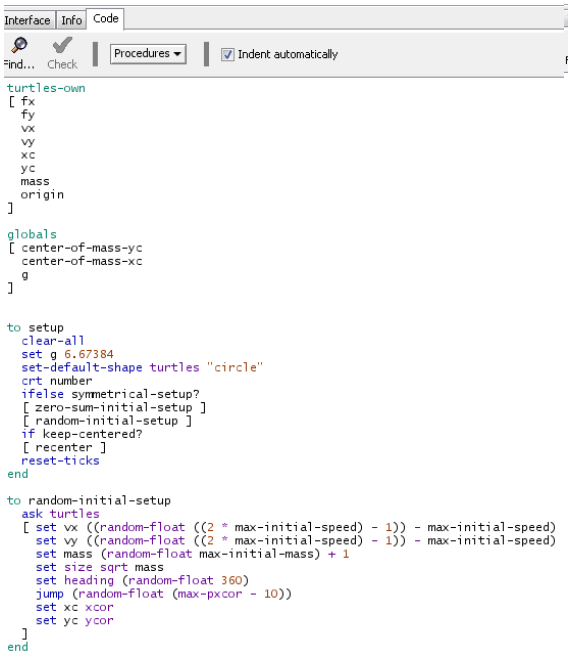
So my partner and I then began to research background information such as redshifts, the Hubble Constant, the Doppler Effect, etc. This research was mostly conducted before the N-body simulation problem was pursued. At that point our research focused mainly on the coding behind classical N-body simulation models and understanding the relationships and initialization of the gravity and velocity variables. The challenging part was incorporating a way in which the agents in NetLogo could understand and calculated the relationships between time and velocity or gravity. Using another program known as CosmoEJS we calculated and found the graphs illustrating the velocity curve of objects moving outwards from space. We wanted to also see if we could recreate very similar curves in NetLogo by plotting distance from origin over time.



Graph from CosmoEJS depicting universal expansion using Gamma-Ray bursts.

DESCRIPTION

In order to construct our N-Body simulation we first coded gravity and velocity into our program. The visual construction was very simple with a set amount of particles moving outwards from one another equally while the formula and equation conversions into NetLogo proved to be significantly more difficult. First we initialized velocity and what the turtles interpreted it as far as individual speed and movement goes. In order to calculate gravity between the different bodies we needed to establish the gravitational constant as a global and tell the agents to update their velocity in terms of their respective forces, a separate variable. In our experiment we raised the gravitational constant to the 11^{th} power making it 6.67384 as opposed to 0.000000000067384. The reasoning for this was because the gravitational constant was so small we would need a simulation size and capacity of up to 5,000 by 5,000 and about 500,000 ticks or more of time before any significant behavior would appear. Even at the fastest speed this would take a long period of time for each individual experiment.



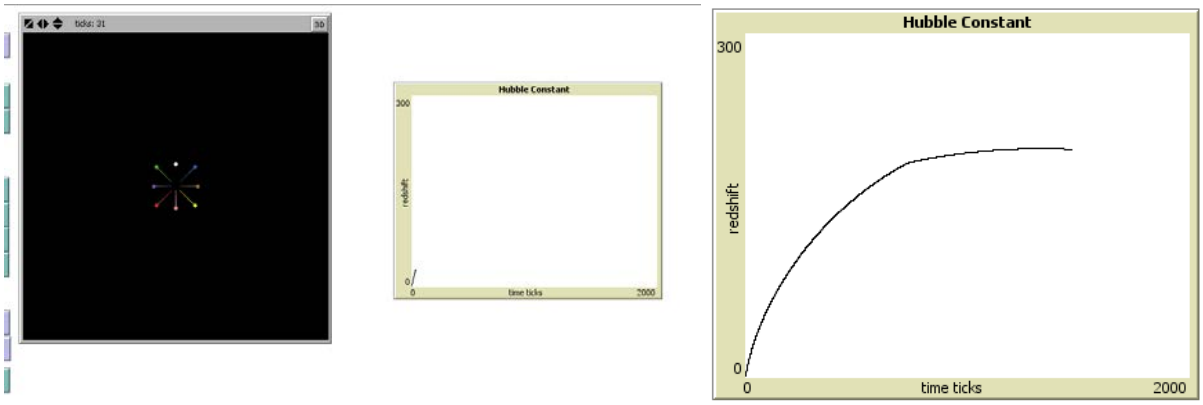
Setup procedure

Go procedure

Now that the variables were initialized we tested out the simulation repeatedly with only two bodies to ensure that the velocities and forces were operating appropriately and calculated correctly. As each experiment turned out ideally with the forces acting between each of the bodies and their velocities gradually slowing down, we added more and more bodies to the simulation. We needed to ensure the simulation was completely even and symmetrical so that bodies wouldn't conglomerate into one spot and since the headings could not be calculated symmetrically when adding bodies on a slider, we were forced to manually add new bodies into the code. Eventually we settled with simply eight bodies in order to conserve time. Now that our simulation was finalized and our experiments were conducted we proceeded to the next step.

RESULTS

The result of our experiments appeared to be that the same curve associated with the Hubble Constant emerged from our calculations. A lot of the times the bodies would eventually start receding back to their condensed due to their forces and their proximal distance to one another. Eventually we were able to make the graph reach a constant progression by giving a velocity that allowed each body to counteract the force of gravity, that is, achieve its escape velocity.



The settings that worked the best for achieving curves like the one above were applying an initial velocity of 1.65 and a mass of 0.1 for all the bodies. Adding more mass only made the bodies recede inward after a certain period of time.

CONCLUSIONS

The conclusion from the given data is that it is indeed possible to code an N-Body simulation into NetLogo and that the Hubble Constant can be visualized using this simulation. With this data we can further implement more variables into the simulation to enhance it's authenticity and accuracy so it can better reflect the expansion of the universe. It would seem from our experiments that our universe is very fortunate for having a very constant and perpetual rate of expansion as our model showed that if that weren't the case our universe would be violently collapsing in the future.

We managed to solve the initial problem of trying to see if it is possible to code an N-Body simulation and visualize universal expansion in NetLogo however now my partner and I are eager to take it even further because of the fact that we ran out of most of our time trying to code in the gravity, force, and velocity factors in addition to our research.

RECOMMENDATIONS

The main thing my partner and I would want to do with this model is extend its accuracy even further by incorporating more variables into the simulation and its calculations. We would likely incorporate more sophisticated elements like dark matter density, and dark matter, which all slightly affect the universal expansion. The random distribution of bodies would also greatly increase the efficiency of this model because at the moment all the bodies are set up perfectly even and symmetrical to simplify matters. With the bodies being randomly distributed within a certain margin of space we would be able to see the bodies clump together and form nebulas, galaxies, and super clusters which would make the simulation much more accurate and complex. If we were to master all of these extra steps the next big course of action would be to attempt to recreate this same simulation in NetLogo 3D. Making use of three dimensions would greatly improve realism/accuracy.

ACKNOWLEDGEMENTS

My partner and I would like to thank Dr. Charles Miller who provided a great deal of assistance and who helped get our project on track and organized. We would also like to thank Judy Miller who assisted with the actual coding part of the experiment. Lastly we would also like to thank our teacher, Ms. Hagaman, who supported us along the way as well as all of you who help to keep the Super Computing Challenge going for providing us with an extensive and entertaining opportunity to experiment and learn about computer science.

REFERENCES

http://en.wikipedia.org/wiki/Hubble's_law (Wikipedia, 2008)

http://csep10.phys.utk.edu/astr162/lect/cosmology/hubble_constant.html (Center for Science and Engineering, 2009)

http://en.wikipedia.org/wiki/Doppler_effect (Wikipedia, 2010)

http://archive.ncsa.illinois.edu/Cyberia/Bima/doppler.html (National Center for Supercomputing Applications, 1995)

http://en.wikipedia.org/wiki/Redshift (Wikipedia, 2011)

http://www.universetoday.com/30565/astronomers-closing-in-on-dark-energy-with-refined-hubble-constant/hubble-constant-2/ (Atkinson, 2009)

http://www.astronomy.ohio-state.edu/~pogge/Ast162/Unit5/expand.html (Pogge, 2006)

http://physics.drexel.edu/~steve/n-body.html (McMillan, 2001)

APPENDIX

Entire Code:

```
turtles-own
[ fx
  fy
  vx
  vy
  xc
  yc
  mass
  origin
]

globals
[ center-of-mass-yc
```

```
  center-of-mass-xc
  g
]


to setup
  clear-all
  set g 6.67384
  set-default-shape turtles "circle"
  crt number
  reset-ticks
end

to random-initial-setup
  ask turtles
  [ set vx ((random-float ((2 * max-initial-speed) - 1)) - max-initial-speed)
    set vy ((random-float ((2 * max-initial-speed) - 1)) - max-initial-speed)
    set mass (random-float max-initial-mass) + 1
    set size sqrt mass
    set heading (random-float 360)
    jump (random-float (max-pxcor - 10))
    set xc xcor
    set yc ycor
  ]
end

to zero-sum-initial-setup
  ask turtles with [who < (number / 2)]
  [ set vx (random-float (((2 * max-initial-speed) - 1)) - max-initial-speed)
    set vy (random-float (((2 * max-initial-speed) - 1)) - max-initial-speed)
    setxy random-xcor random-ycor
    set xc xcor
    set yc ycor
    set mass (random-float max-initial-mass) + 1
    set size sqrt mass
  ]

  ask turtles with [who >= (number / 2)]
  [ set vx (- ([vx] of turtle (who - (number / 2))))
    set vy (- ([vy] of turtle (who - (number / 2))))
    set xc (- ([xc] of turtle (who - (number / 2))))
    set yc (- ([yc] of turtle (who - (number / 2))))
    setxy xc yc
    set mass [mass] of turtle (who - (number / 2))
    set size sqrt mass
  ]
  set center-of-mass-xc 0
  set center-of-mass-yc 0
```

```
end

to setup-two-planet
 set number 0
 setup
 crt 1
 [ set color blue
  set mass initial-mass
  set size 5
  set xc 5
  set yc 5
  setxy xc yc
  set vx (((2 * max-initial-speed) - 1) - max-initial-speed)
  set vy (((2 * max-initial-speed) - 1) - max-initial-speed)
  set heading 45
  set origin patch-here
 ]
 crt 1
 [ set color red
  set mass initial-mass
  set size 5
  set xc -5
  set yc -5
  setxy xc yc
  set vx  ( - (((2 * max-initial-speed) - 1) - max-initial-speed))
  set vy  ( - (((2 * max-initial-speed) - 1) - max-initial-speed))
  set heading 225
  set origin patch-here
 ]
  crt 1
 [ set color green
  set mass initial-mass
  set size 5
  set xc -5
  set yc 5
  setxy xc yc
  set vx  ( - (((2 * max-initial-speed) - 1) - max-initial-speed))
  set vy  (((2 * max-initial-speed) - 1) - max-initial-speed)
  set heading 0
  set origin patch-here
 ]
  crt 1
 [ set color yellow
  set mass initial-mass
  set size 5
  set xc 5
  set yc -5
  setxy xc yc
```

```
  set vx  (((2 * max-initial-speed) - 1) - max-initial-speed)
  set vy  ( - (((2 * max-initial-speed) - 1) - max-initial-speed))
  set heading 180
  set origin patch-here
]
crt 1
[ set color pink
  set mass initial-mass
  set size 5
  set xc 0
  set yc -5
  setxy xc yc
  set vx  0
  set vy  ( - (((2 * max-initial-speed) - 1) - max-initial-speed))
  set heading 180
  set origin patch-here
]
crt 1
[ set color white
  set mass initial-mass
  set size 5
  set xc 0
  set yc 5
  setxy xc yc
  set vx  0
  set vy  (((2 * max-initial-speed) - 1) - max-initial-speed)
  set heading 180
  set origin patch-here
]
  crt 1
[ set color brown
  set mass initial-mass
  set size 5
  set xc 5
  set yc 0
  setxy xc yc
  set vx  (((2 * max-initial-speed) - 1) - max-initial-speed)
  set vy  0
  set heading 180
  set origin patch-here
]
crt 1
[ set color violet
  set mass initial-mass
  set size 5
  set xc -5
  set yc 0
  setxy xc yc
```

```
   set vx  ( - (((2 * max-initial-speed) - 1) - max-initial-speed))
   set vy  0
   set heading 180
   set origin patch-here
  ]
end

to create-two-particle-expansion
  crt 1
  [    setxy 0 5
       set vx initial-velocity-x
       set vy initial-velocity-y
       set mass initial-mass
       set size sqrt mass
  ]
  crt 1
  [
       setxy 0 -5
       set vx initial-velocity-x
       set vy initial-velocity-y
       set mass initial-mass
       set size sqrt mass
  ]
end

to go
  ask turtles
  [ set fx 0
    set fy 0
  ]
  ask turtles [ check-for-collisions ]
  ask turtles [ update-force ]
  ask turtles [ update-velocity ]
  ask turtles [ update-position ]
  fade-patches
  tick
end

to check-for-collisions
  if any? other turtles-here
  [
    ask other turtles-here
    [
      set vx vx + [vx] of myself
      set vy vy + [vy] of myself
      set mass mass + [mass] of myself
      set size sqrt mass
    ]
```

```
    die
  ]
end

to update-force
  ask other turtles [ sum-its-force-on-me myself ]
end

to sum-its-force-on-me [it]
  let xd xc - [xc] of it
  let yd yc - [yc] of it
  let d sqrt ((xd * xd) + (yd * yd))
  set fx fx + (cos (atan (- yd) (- xd))) * ([mass] of it * mass) / (d * d)
  set fy fy + (sin (atan (- yd) (- xd))) * ([mass] of it * mass) / (d * d)
end

to update-velocity
  set vx (vx + (fx * g / mass))
  set vy (vy + (fy * g / mass))
end

to update-position
  set xc (xc + vx)
  set yc (yc + vy)
  adjust-position
end

to adjust-position
  ifelse patch-at (xc - xcor) (yc - ycor) != nobody
  [ setxy xc yc
    show-turtle
    if (fade-rate != 100)
    [ set pcolor color + 3 ]
  ]
  [ hide-turtle ]
end

to recenter
  find-center-of-mass
  ask turtles
  [ set xc (xc - center-of-mass-xc)
    set yc (yc - center-of-mass-yc)
    adjust-position
  ]
end

to find-center-of-mass
  if any? turtles
```

```
  [ set center-of-mass-xc sum [mass * xc] of turtles / sum [mass] of turtles
    set center-of-mass-yc sum [mass * yc] of turtles / sum [mass] of turtles
  ]
end

to fade-patches
  ask patches with [pcolor != black]
  [ ifelse (fade-rate = 100)
    [ set pcolor black ]
    [ if (fade-rate != 0)
      [ fade ]
    ]
  ]
end

to fade
  let new-color pcolor - 8 * fade-rate / 100
  ifelse (shade-of? pcolor new-color)
  [ set pcolor new-color ]
  [ set pcolor black ]
end

to plot-rule-mean
  if any? (turtles with [mass > 0])
  [ plot mean [distance origin] of (turtles with [mass > 0]) ]
end
```