FINDING THE PERFECT PANEL

New Mexico Supercomputing Challenge

Final Report

Thursday, March 27, 2014

Team 49

Desert Academy

Team Members

Damian Browne

Isaac Fischer

Cameron Mathis

Project Mentors

Jeffery Mathis

Jocelyne Comstock

Executive Summary

Solar power is widely recognized as an efficient and renewable form of alternative energy. Sunlight is sustainable; the sun is not expected to go out any time soon, and there is enough hitting the surface of the earth to power the planet many times over. However, current technology cannot realistically harvest sunlight at an efficiency greater than 15%, and so the solar panels must be positioned carefully in order to maximize efficiency.

For our project, we used NetLogo to create a three-dimensional model of a solar panel. The model featured a sun casting light on a small solar panel. We added functions to change the inclination of the panel – in theory to tip it more towards the sun – and to change the "latitude" at which the experiment is taking place – by moving the sun accordingly. Once we set out these parameters, we elected to find a method of determining the ideal inclination for a solar panel at a given latitude.

We then recorded the energy at latitudes ranging from 0° to 60° for solar panels at inclinations ranging from 0° to 80°, and for each latitude found the inclination that optimized the total energy collection. Surprisingly, at every latitude this turned out to be 20° of inclination. This leads us to the conclusion that 20° is the ideal inclination for a solar panel no matter the latitude.

The Problem

Solar panels are very fickle in relationship to variables such as latitude and tilt of the panel. It is a complicated issue to determine how to position solar panels to maximize efficiency, and as a result, energy is often wasted. Our real-world problem that we are trying to solve is as follows: How can we position a solar panel at a given latitude to maximize its energy efficiency?

The Method

Our model was made entirely in NetLogo, because we decided to use an agent-based approach to the problem rather than a purely mathematical approach. The source code for our project is available in Appendix B at the end of this report.

To accurately simulate a solar panel in NetLogo, we had to first alter the two-dimensional coordinate system to support the three-dimensional nature of our model. To do this, we added

two new "turtles-own" variables, "zcor" and "zheading." The former represents the turtle's position above or below the XY plane, while the latter represents the angle that the turtle's movement vector makes with the XY plane. To simulate movement in three dimensions, the procedure "move" was defined, which involved moving forward and altering zcor based on the cosine and sine, respectively, of the heading.

Once the three-dimensional coordinate system was in place, we needed to add a light source – the sun. Since the world in NetLogo cannot be programmed to move, and since an agent cannot be located outside the world, we had to settle for a small orbital sun, with a special movement function governed by a parametric equation. It provided light in the form of "photon" agents, which would be fired in the general direction of the location where the solar panel would be. This way, the amount of incoming solar radiation would vary depending on the position of the sun.

The solar panel itself was not represented by an agent; rather, it was represented by a rectangle 41 patches wide. The patches were overseen by one agent which would tell them when to call the procedure, which would simply determine whether the photons directly above or below them are in the range where they would logically collide with the panel. If so, the photon would die, and an electron would be hatched. The electrons would then travel from the panel to a lightbulb, which would light up at varying brightness depending on the number of electrons coming in at any given time.

While the use of electrons in this method would seem unnecessary, they serve an additional purpose: the instant count of electrons is used to depict the energy entering the bulb. The number of electrons that were created on any given day represents the total energy that was produced by the panel on that day. With this to measure, it was then possible to set up variables to change in order to conduct experiments. While many experiments were possible, we decided to find the ideal inclination for a solar panel at many different latitudes.

To conduct this experiment, it was necessary to be able to alter the latitude and inclination of the solar panel. Altering the latitude was easy; we multiplied the sine of the latitude by the maximum y-coordinate and made it negative, and set the y-coordinate of the sun to the resulting value. Altering the inclination of the solar panel proved more difficult; the final solution was to find the equation for the plane that the panel would lie in, and have the patches check whether the

photons intersected that plane rather than the XY plane. We then created sliders to control these attributes, making it possible to conduct our experiment.

Verification and Validation

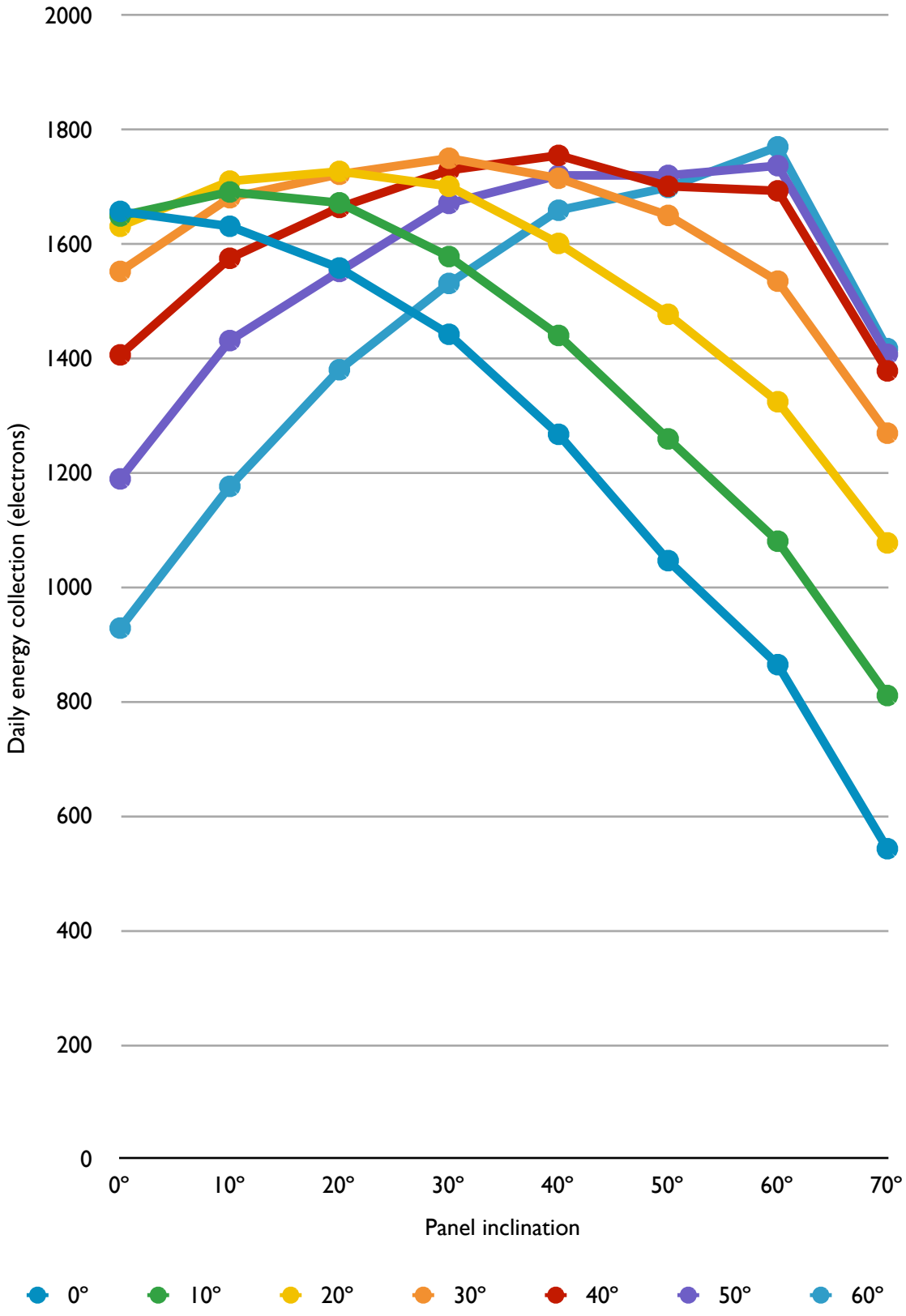
In order to confirm the scientific accuracy of our model, we did a good deal of research before we began the code. Solar panels work by displacing atoms so that when they come in contact with a photon an electron is bumped out of orbit around the nucleus (Toothman/Aldous). The electron is replaced by the photon and becomes free to be used as power. Our model accurately portrays latitude, as we used a trigonometric functions for determining the sun's position in the XY plane and on the Z axis. We also made our solar panel tilt, as tilting a solar panel can help optimize its efficiency. That is what ensures the validity of our model.

Results

Table 1: Latitude (top) by inclination (left): total daily energy collection

Inclination \ Latitude	0°	10°	20°	30°	40°	50°	60°
0°	1658	1650	1632	1553	1407	1190	929
10°	1632	1692	1711	1682	1576	1432	1177
20°	1559	1673	1728	1723	1665	1553	1381
30°	1443	1579	1702	1751	1730	1672	1532
40°	1268	1441	1602	1716	1756	1721	1660
50°	1047	1260	1478	1651	1702	1721	1700
60°	865	1081	1325	1536	1694	1738	1771
70°	543	811	1078	1270	1379	1408	1418

Graph 1: Daily energy collection at various inclinations by latitude



Analysis and Conclusion

The data we collected show a very distinct pattern, with the optimal inclination increasing as latitude increases, and the maximum efficiency occurring when latitude and inclination are equal. The only outlier from this set was at 50° of latitude, when the maximum fell at 60° inclination. However, this may be due to experimental error.

These results correspond with the hypothesis we obtained by using geometry to solve the same issue. According to geometry, the angle between the panel and the incoming sunlight should be the only determining factor in the amount of received energy; latitude and inclination *on their own* would have no effect. Any variations from this prediction are possibly due to the variability of the results of the experiment. The results imply our hypothesis to be correct.

These results open up the window for future study. One possibility would be the exploration of arranging multiple solar panels with the same total surface area; these panels would need to tilt independently of the others and would need to reflect a certain amount of light. Since a mixture of tilt angles could account for sunlight coming from a variety of directions, this could help improve the efficiency of the model. Another option would be to incorporate more real-world variables such as cloud cover and objects that cast shadows on the panel for part of the day. This model only serves as the basics, opening up a world of opportunity.

Most Significant Achievement

Although there were many successes in our project, such as finding the right function to set the x and y coordinates for latitude, probably the biggest triumph was correctly creating the tilt factor on our panel. We used many ineffective methods for determining whether photons actually collide with the panel when tilted, such as counting any photon underneath the panel as having passed through it, which created problems in the early morning and late evening. We then pondered determining the intersection by the photon's heading relative to the headings from its origin to the four corners of the panel, but this method was even less effective. We discovered our final method by consulting with Santa Fe Institute intern John Balwit, who proposed the current method of having the patches that make up the panel's area determine whether photons directly above them intersected the panel.

Acknowledgements

Our project was a team effort between all three of us. There were, however people that we could not have done without. We have to acknowledge our mentors at school, Jocelyne Comstock and Jeffery Mathis. They have helped us with motivation, as well as advising the way we went about coding our model. Additionally, we may not have completed our model without the advice of John Balwit, an intern at the Santa Fe Institute. He said we might try asking patches under our panel to tell if the photon landed there or not. We also acknowledge Bandit Gangwere, a very helpful individual that we met at the kickoff who gave us advice at the start, when we needed it most. We also thank Thomas Laub from Sandia National Labs, who us some critical and very helpful feedback on our Interim Report. Thanks to these people, we were able to get solid opinions and feedback. We couldn't have done it without them.

Appendix A: References

Jessika Toothman and Scott Aldous. *How Solar Cells Work*. Retrieved from http://science.howstuffworks.com/environmental/energy/solar-cell.htm

PNM. *Solar and Wind Energy*. Retrieved from https://www.pnm.com/solar-and-wind-energy

Appendix B: Code

```
breed [photons photon]
breed [electrons electron]
breed [bulb]
breed [sun]

globals [currentenergy cumulativeenergy energylist]
turtles-own [zcor zheading]
electrons-own [wire]
sun-own [angle optheading optzheading]
bulb-own [energy]

to setup
  clear-all
  reset-ticks
  set currentenergy 0
  set cumulativeenergy 0
  set energylist []
  ask patches [
    set pcolor gray + 3
  ]
  ask patches with [abs(pxcor) <= 40 and abs(pycor) <= 40 * cos inclination
                  and pycor != 0] [set pcolor black]
  create-bulb 1 [
    set shape "bulb"
    set size 60
    set color black
    set heading 0
    set ycor 100
    set zcor 0
    set energy 0
  ]
  create-sun 1 [
    set xcor 160
    set ycor 0
    set zcor 0
    set color pink
```

```
    set size 120
    set shape "circle"
    set angle 0
  ]
end

to move [far]
  repeat far [
    set zcor zcor + sin (0 - zheading)
    jump cos zheading
  ]
end

to illuminate ; sun
  if zcor >= 0 [
    hatch-photons luminosity [
      set hidden? false
      set color yellow
      set size 7.5
      set heading [optheading] of turtle 1
      set zheading [optzheading] of turtle 1
      set xcor xcor + random 120 - 60
      set ycor ycor + random 120 - 60
      set zcor zcor + random 120 - 60
      move 1
    ]
  ]
end

to orbit ; sun
  let radius (cos latitude) * 160
  set optheading (towards patch 0 0)
  set optzheading (atan zcor distance patch 0 0)
  set angle angle + 1
  setxy 160 * cos angle (- sin latitude * sin angle) * 160
  set zcor radius * sin angle
  if zcor >= 2 [
    set color yellow
    if angle mod 360 = 14 [ask patches with [abs(pxcor) > 40 or abs(pycor) >
                  40] [set pcolor white]]
  ]
  if zcor <= -2 [
    set color gray - 1
    if angle mod 360 = 194 [ask patches with [abs(pxcor) > 40 or abs(pycor) >
                  40] [set pcolor gray]]
  ]
```

```
  if zcor < 2 and zcor > -2 [
    set color pink
    if angle mod 360 = 167 [ask patches with [abs(pxcor) > 40 or abs(pycor) >
                  40] [set pcolor gray + 3]]
    if angle mod 360 = 347 [ask patches with [abs(pxcor) > 40 or abs(pycor) >
                  40] [set pcolor gray + 3]]
  ]
  if zcor = 0 - radius [
    set energylist lput cumulativeenergy energylist
    set cumulativeenergy 0
  ]
  set size 120 + zcor / 5
end

to photomove ; photons
  repeat 10 [
    move 1
    if abs(pxcor) < 40 and abs(pycor) < 40 * cos inclination [
      if zcor <= ycor * tan inclination + 1 and zcor >= ycor * tan
                  inclination - 1 [
        hatch-electrons 1 [
          set color red
          set size 7.5
          set wire 0
          set zheading 0
          set currentenergy currentenergy + 1
          set cumulativeenergy cumulativeenergy + 1
        ]
        die
      ]
    ]
  ]
  if ycor >= 160 - 1 and (heading > 270 or heading < 90) [die]
  if ycor <= min-pycor + 1 and (heading < 270 and heading > 90) [die]
  if xcor >= 160 - 1 and (heading < 180) [die]
  if xcor <= min-pxcor + 1 and (heading > 180) [die]
  if zcor > 1.5 * 160 or zcor < -40 [die]
  if zcor < 0 [set color violet]
end

to electromove ; electrons
  if wire = 0 [
    setxy 0 random 20
    set wire 1
  ]
  if wire = 1 [
```

```
    set heading 0
    move 20
    if ycor >= 100 [set wire 2]
  ]
  if wire = 2 [
    ask bulb [set energy energy + 1]
    set currentenergy currentenergy - 1
    die
  ]
end

to happybulb ; bulb
  set color 30 + 20 * (1 / 2) ^ (1 / (energy + 1))
  set energy 0
end

to go
  ask photons [
    photomove
  ]
  ask electrons [
    electromove
  ]
  ask sun [
    orbit
    illuminate
  ]
  ask bulb [
    happybulb
  ]
  tick
end

to experiment
  let trial 0
  repeat 7 [
    set latitude 10 * trial
    set inclination 0
    setup
    repeat 9 [
      repeat 10 [repeat 360 [go]]
      let average (item (length energylist - 1) energylist + item (length
                  energylist - 2) energylist + item (length energylist - 3)
                  energylist + item (length energylist - 4) energylist + item
                  (length energylist - 5) energylist + item (length
                  energylist - 6) energylist + item (length energylist - 7)
```

```
                energylist + item (length energylist - 8) energylist + item
                (length energylist - 9) energylist + item (length
                energylist - 10) energylist) / 10
      repeat 10 [set energylist but-last energylist]
      set energylist lput round(average) energylist
      set inclination inclination + 10
    ]
    show latitude
    show energylist
    set trial trial + 1
  ]
end
```