Daisy World

Background

The climate of Earth is a very complex system, with many interacting parts, making prediction and understanding rather difficult. But using simplified models, we can come to understand the chaos behind the climate. One such model is commonly called Daisyworld. In the Daisyworld model, there is a very simple earth that has only two species on it: black daisies and white daisies. These daisies prefer to grow at a certain temperature and when the planetary temperature deviates from that temperature, the daisies will begin to die. And if the planetary temperature moves back towards the temperature that the daisies prefer, the population of the daisies increases. The twist is that black daisies increase the amount of solar energy absorbed, thus increasing the temperature of the planet, whereas the white daisies tend to do the reverse. So as the daisy populations increase, the temperature may increase (or decrease, depending on the daisy mix), and result in the daisies dying off. But when they die off, the temperature balance changes again, resulting in yet another change in daisy population. This continues ad nauseam.

To simulate these daisies, we must figure out how to determine how many daisies of each type there are (represented by the coverage of the daisies over the planet) and how the daisies affect temperature. For simplicity's sake, all the black daisies must grow together in one patch and all the white daisies must also grow together in one patch. The remaining ground is bare, having a neutral albedo. We calculate the temperature over each patch using a simplified set of equations based on the size of the patch and the solar forcing. Using that temperature, we can calculate how the daisy populations will respond. If the temperature is too low or too high, the daisies start dying. After we use the temperature to determine the growth rate of the daisy patches, we use the new size of the daisy patches as input into the whole calculation for the next time step. This continues on forever, with the daisy patches growing and shrinking according to the temperature.

Octave is a great tool to look at the Daisyworld scenario. It allows one to easily calculate the changes in daisy population and plot these results. Next, we'll explain how to use Octave to visualize the Daisyworld scenario.

Math

Simulating the daisy populations requires a fairly decent set of formulas. First, let's define some constants that will be used throughout:

$A_W = 0.75$ [albedo of white daisies]
$A_B = 0.25$ [albedo of black daisies]
$A_G = 0.50$ [albedo of bare ground]
$D = 0.3$ [death rate of daisies]
$Tr = 0.6$ [horizontal temperature transport parameter]
$q = L \cdot S / 4 \cdot \sigma$ (where L = 1.2, S = solar constant = 1368 W/m^2 and sigma is the Stefan-Boltzmann constant 5.67×10^{-8} $Wm^{-2}K^{-2}$)
$b = 0.003265$ [controls growth rate]
$T_0 = 295.5$ K [optimal temperature for daisy growth]

All of these, save, perhaps, q, can be modified to change the results of the simulation.

Now, we have two variables, C_W and C_B, which give the fraction of the planet covered in white and black daisies respectively. The amount of bare ground, C_G, is defined to be whatever land isn't covered by daisies and is calculated by $C_G = 1 - C_W - C_B$. Using these fractions, we can determine the total albedo of the planet, A, using the following formula:

$$A = C_G \cdot A_G + C_B \cdot A_B + C_W \cdot A_W$$

The surface temperature above each daisy patch is given by the following two formulas (one for each type of daisies). [perhaps should explain the physics behind these equations?]:

$$T_B^4 = (1 - Tr) \cdot q \cdot (A - A_B) + 2 \cdot q \cdot (1 - A)$$
$$T_W^4 = (1 - Tr) \cdot q \cdot (A - A_W) + 2 \cdot q \cdot (1 - A)$$

Using these formulas, the next step is to determine the growth rates, β_B and β_W, for the black and white daisies respectively. The growth rate is controlled, essentially, by the difference between the actual temperature and the optimal temperature. The closer the temperature of the planet is to the optimal temperature, the greater the growth rate will be.

$$\beta_B = 1 - b \cdot (T_0 - T_B)^2$$
$$\beta_W = 1 - b \cdot (T_0 - T_W)^2$$

So, now that we have the growth rate, we can finally create the equations that determine the change in daisy populations. These equations are the core of the simulation.

$$\Delta C_B = \Delta t \cdot C_B \cdot (C_G \cdot \beta_B - D)$$
$$\Delta C_W = \Delta t \cdot C_W \cdot (C_G \cdot \beta_W - D)$$

<also put in some websites>

Objectives

- Use Octave to calculate daisy populations and solve related differential equations
- See how initial conditions and simulation parameters affect the daisy populations

Learning Objectives

1. Octave:
   a) Creating functions
   b) Using global variables
   c) Solving differential equations
2. General
   a) Modelling the world through differential equations
   b) Chaos, equilibrium and such

Step-by-Step

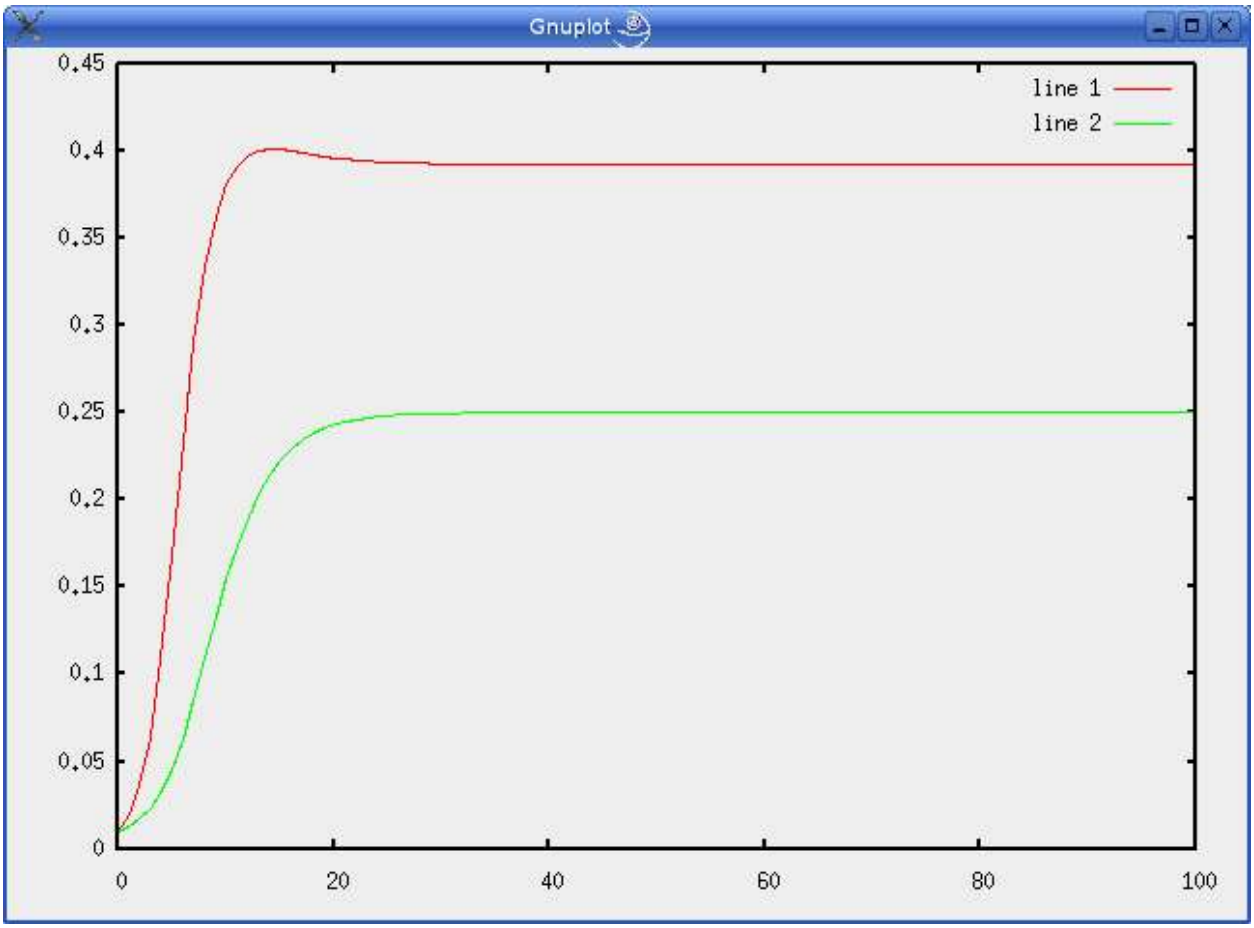
1. Build an Octave script file which will solve the differential equations used in the Daisyworld simulation.

a) Put global variable definitions at the top of the script file. These global variables hold the constants that control the simulation.
b) Create a function, or set of functions, which can find the size of the daisy populations given the current size of those populations.
c) Add the code to solve the differential equations and plot the results

2. Experiment with different constants and initial conditions to see how they affect the size of the daisy populations and how soon (if ever), the populations stabilize.

Analysis

Using the default values, the graph looks like this:



By changing, for example, the albedo of the daisies, or the death rate of the daisies, the amount of time it takes for the system to reach equilibrium changes. Also, the equilibrium fraction of each type of daisy changes. Since there are many constants to play with, there are many possibilities for modifying the results of the simulation.

Below is the script file that was used when making the graph above:

```
# Daisyworld Octave simulation script
#

#
# Global constants used in the simulation formulas
#
global Ab = 0.25;
global Aw = 0.75;
global Ag = 0.50;
global L = 1.2;
global S = 1368;
global sigma = 5.67e-8;
global q = (L * S) / (4 * sigma);
global Tr = 0.6;
global D = 0.3;
global b = 0.003265;
global Tnought = 295.5;

#
# Utility functions that calculate variables needed by the main equations
#
function Cg = calc_Cg (Cb,Cw)
        Cg = 1 - Cb - Cw;
endfunction

function A = calc_A (Cg,Cb,Cw)
        global Aw;
        global Ab;
        global Ag;
        A = Cg.*Ag + Cb.*Ab + Cw.*Aw;
endfunction

function Tb = calc_Tb (A)
        global Ab;
        global q;
        global Tr;
        Tb = ((1 - Tr) * q * (A - Ab) + (2 * q * (1 - A))) ** 0.25;
endfunction

function Tw = calc_Tw (A,Ts)
        global Aw;
        global q;
        global Tr;
        Tw = ((1 - Tr) * q * (A - Aw) + (2 * q * (1 - A))) ** 0.25;
endfunction

function Betab = calc_Betab(Tb)
        global b;
        global Tnought;
        Betab = 1 - b * ((Tnought - Tb) ** 2);
endfunction

function Betaw = calc_Betaw(Tw)
        global b;
        global Tnought;
        Betaw = 1 - b * ((Tnought - Tw) ** 2);
endfunction
```

```
function Cb_new = calc_del_Cb (Cb,Cg,Betab,del_T)
        D = 0.3;
        if (Cb == 0)
                Cb_new = 0.01;
        else
                Cb_new = Cb * del_T * (Cg * Betab - D);
        endif
endfunction

function Cw_new = calc_del_Cw (Cw,Cg,Betaw,del_T)
        D = 0.3;
        if (Cw == 0)
                Cw_new = 0.01;
        else
                Cw_new = Cw * del_T * (Cg * Betaw - D);
        endif
endfunction

#
# Main set of equations
#
function ret = population (x, t)
        Cg = calc_Cg(x(1),x(2));
        A = calc_A(Cg,x(1),x(2));
        Tb = calc_Tb(A);
        Tw = calc_Tw(A);
        Betab = calc_Betab(Tb);
        Betaw = calc_Betaw(Tw);
        ret(1) = calc_del_Cb(x(1),Cg,Betab,1);
        ret(2) = calc_del_Cw(x(2),Cg,Betaw,1);
endfunction

# set initial conditions (fraction of planet covered by black and white
daisies)
x0 = [0.01; 0.01];
# create range of time values
t = linspace(0,100,100);
# solve the differential equations
x = lsode("population",x0,t);
# plot the results
plot (t,x);
```