Saturation Equilibrium of Craters

Jashan Chopra*

ASTR 3750

University of Colorado Boulder

November 15th, 2018

1 Introduction

This project's goal is to simulate the cratering process on a typical planetary surface. As we can see in the case of our Moon, a planetary surface will eventually reach saturation of the amount of craters it can have. After time, the density of craters reaches this maximum saturation and will remain relatively constant over time. This numerical test simulation will be carried out with MATLAB. Craters will form in a 500 kilometer region, obliterating other craters that they land on. The simulation stops once we have achieved an appropriate saturation. Graphs and plots will be included to illustrate the cratering process.

2 Assumptions

Real life cratering scenarios are extremely involved. A wide variety of occurrences can affect a terrestrial surface. Wind and water erosion, covering lava flows, secondary craters, and more can affect the saturation of craters on the surface. In addition, impacting objects come in all sizes and forms, creating a variety of ejecta blankets and crater sizes. To standardize this problem, we will make a broad range of assumptions. We consider only a 500 kilometer region, and assume that every impact creates a 50 kilometer diameter crater, with a 60 kilometer diameter ejecta blanket. We assume that a single crater is formed every 1000 years. Furthermore, if the ejecta blanket of one crater covers the center of another crater, that covered crater is effectively obliterated. Ignoring the aforementioned effects, like erosion, we will consider our surface saturated when the average number of craters in the test region changes by less than 5 percent after a doubling of time.

*107689146

3 Methods

Simulating Crater Impacts

The MATLAB script is broken up into two main sections. To start off, we preallocate a wide variety of variables for use in the analysis and simulation. These variables will be introduced where appropriate. First we introduce a number of iterations, each iteration is 1000 years, and thus one crater hits per iteration. Our entire simulation process occurs over a FOR loop, which iterates through 1000 iterations. We then assign an x and y location for the crater's center at random.

Once the crater's center is assigned, we need to make sure that the ejecta blanket does not cover another crater center. To do this, we iterate a FOR loop for each crater that currently exists. We then find the vector that traverses our new crater's center to each other crater's center. If the magnitude of this vector is between -30 kilometer to 30 kilometer, we can tell the old crater center was covered by the new ejecta blanket. When this occurs, we delete that old crater's data entries from our ongoing matrix of crater location values.

After the new crater is created, and the location is checked to other craters, we then check our saturation condition. During each crater impact, the current number of craters is compared to the number of craters at half of the current time step. The number of craters is calculated from the number of nonzero entries in the position matrix. If the difference between the number of craters at the doubled time and the number of craters at the previous time is less than 5 percent, then we exit the loop.

The final part of the loop calculates our saturation. We take the current area of all our ejecta blankets, calculated from the 60 kilometer diameter, and divide it by our 500 kilometer region. This result is the saturation of our surface, and the script prints the current saturation to the screen after each crater hits.

Plots

After our FOR loop runs the course, we now have filled out matrices for use in plotting. We have a matrix for the x and y locations, the saturation at each point, the total number of craters, and each timestamp.

To plot the simulation, we iterate a FOR loop for each crater. We plot the center of each crater. This is done after each crater location is found, so that we can delete the old craters by simply removing their locations. The axes align with the x and y locations respectively. We plot the region slightly over the 500 kilometer area so that we can see the full extent of every crater. After plotting the center, we use the *viscircles* function in MATLAB to create two circles around each center. The first circle defines the extent of the crater, and the second circle shows the extent of the ejecta blanket. Using the *linkdata* command in MATLAB, we can have our plot update every time a new crater appears. Thus, we create an animation of the crater impacts. Below we see four plots of different simulations at indicated time stamps.

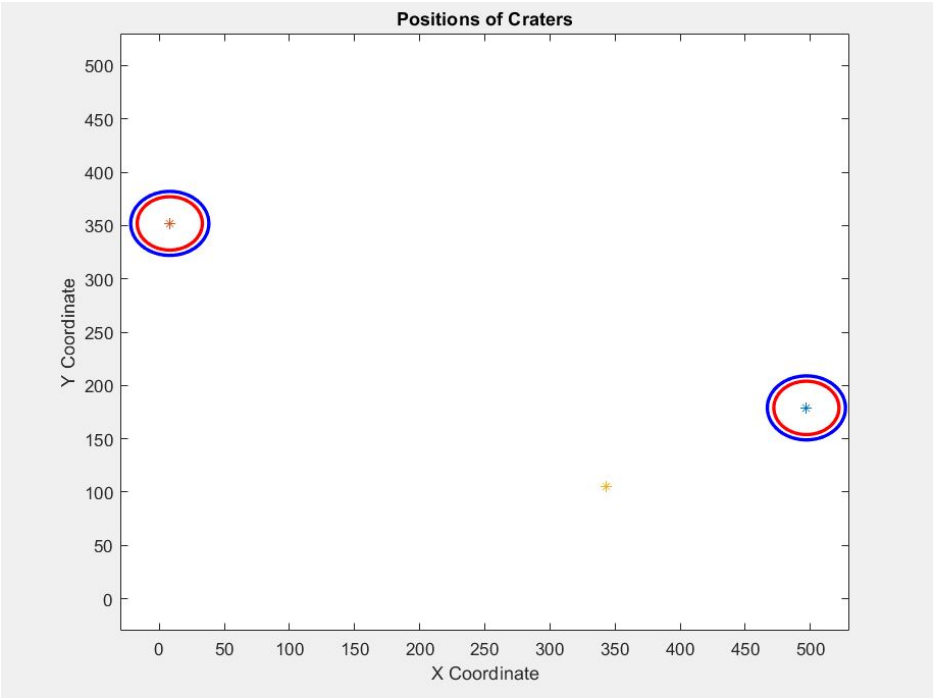


Figure 1: 2 Craters after 2,000 years

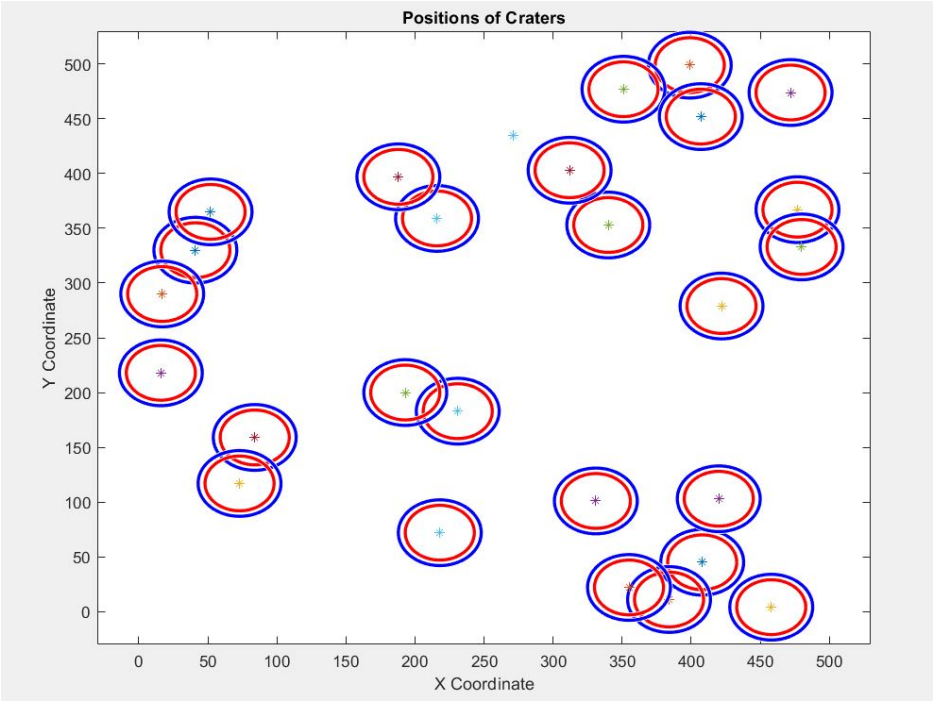


Figure 2: 26 Craters after 26,000 years

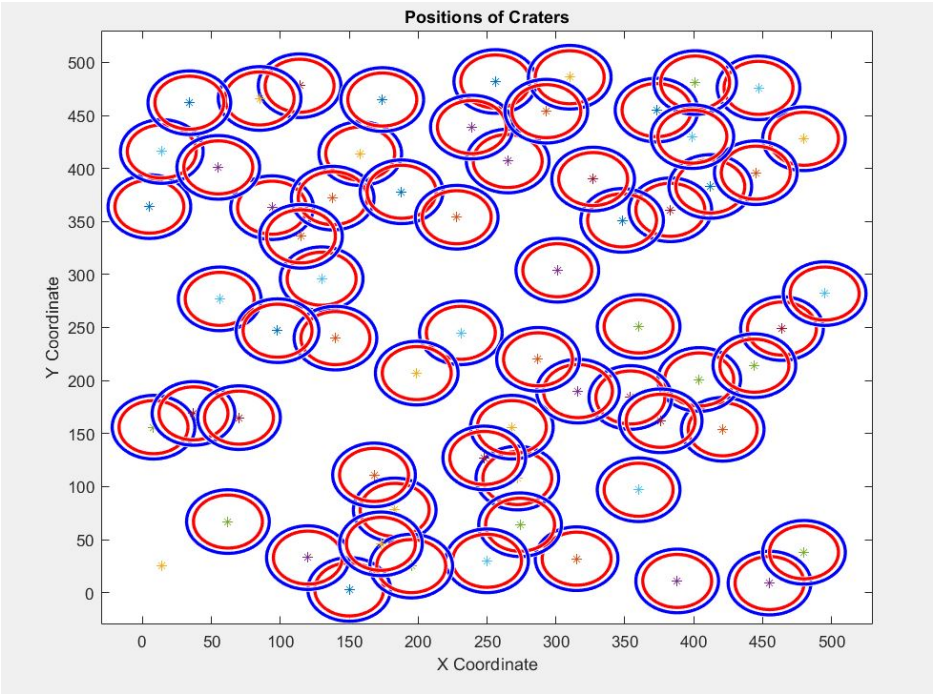


Figure 3: 66 Craters after 99,000 years

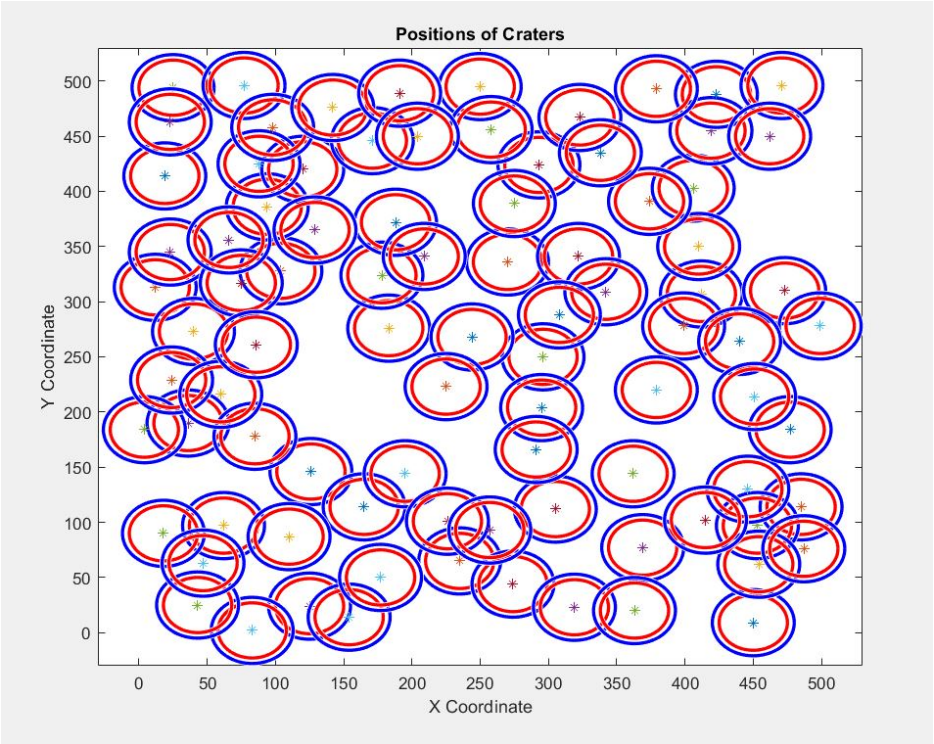


Figure 4: 101 Craters after 495,000 years

4 Results

The final section of our MATLAB code outputs tables of our crucial values. For the sake of clarity, the large tables of individual crater location and saturation at each time step are omitted here, as they would be upwards of 400 columns long. Our final important values are as follows:

Table 1: Critical Values from Crater Simulation

Variable	Value
Number of Craters	91 Craters
Final Saturation	82.046 Percent
Final Time	449,000 Years

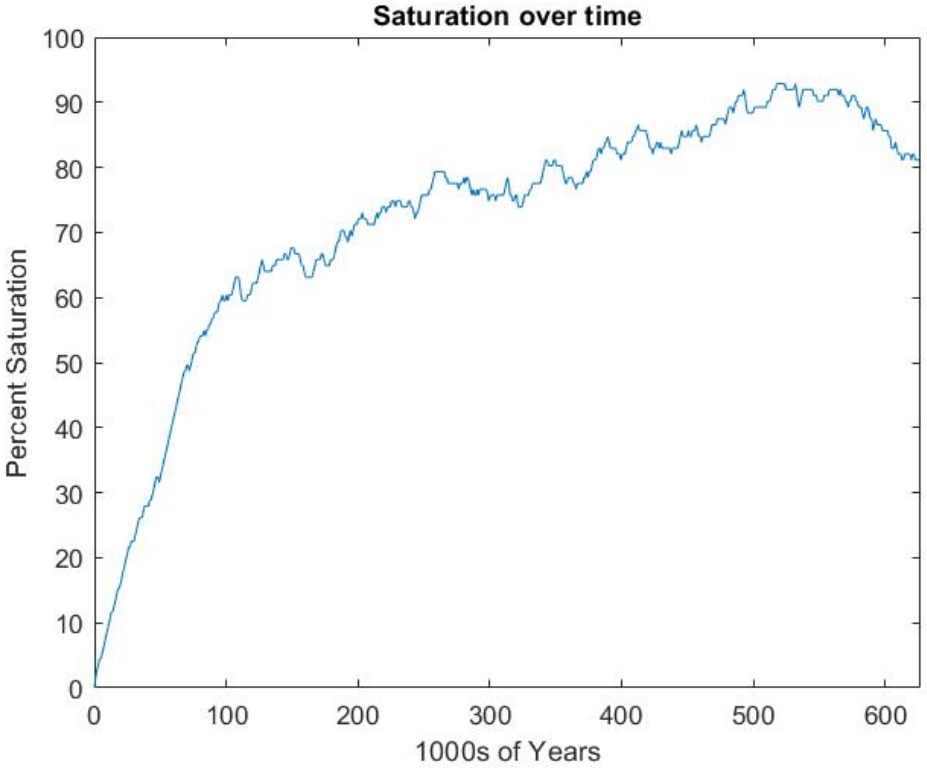
Note that this is only a single simulation. Each simulation returns different results, so to get a more accurate result we perform multiple simulations and average the numbers. We simulated the crating scenario ten times to get a better sense of our results. For clarity, the tables for each simulation are excluded, but the uncertainty calculations are as follows:

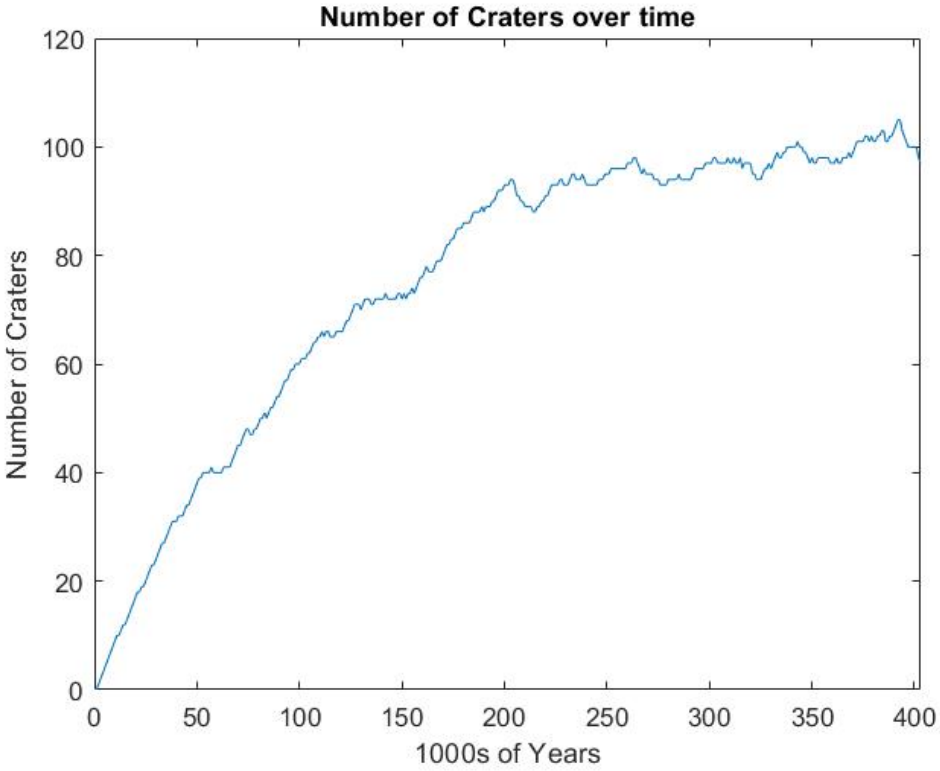
Table 2: Averages of 10 simulations

Variable	Value	Uncertainty
Number of Craters	90 Craters	± 4 Craters
Final Saturation	82.4065 Percent	± 4 Percent
Final Time	$447,900$ Years	± 96000 Years

By performing 10 simulations and finding the mean and standard deviation of the data, we get a better sense of the number of craters and the saturation. We can see that the uncertainty in the number of craters and the final saturation of the simulation is fairly small. This makes sense because the number of craters is quite controlled, and each simulation has a regularly defined stopping point. The saturation number is directly related to the number of impacts, so it also makes sense that the uncertainty would be low. We do notice that the uncertainty in the final time is large. However, this too makes sense. In the later stages of the simulation, a single crater can obliterate a fair amount of previous craters. These small spikes in crater number can push the stopping condition back far.

Finally, instead of a long table of values, we plot the number of craters and the saturation over time. Note that these two plots are for separate test cases, to give a sense of the variety in the simulation. Two of these plots for the same test case would look very similar, as the saturation is a function of the number of craters. We can see from the plots that the number of craters rises nearly linearly, then starts to plateau as some craters are destroyed each time a new one lands.


Saturation over time
Percent Saturation
1000s of Years
0
10
20
30
40
50
60
70
80
90
100
0
100
200
300
400
500
600


Number of Craters over time
Number of Craters
1000s of Years
0
20
40
60
80
100
120
0
50
100
150
200
250
300
350
400

5 Discussion and Conclusion

The main purpose of this project is to show that the crater saturation of a planetary surface will eventually plateau. Through a constrained simulation we've shown that the surface will linearly approach a certain saturation value. After this, the saturation will form a rough plateau and continue to perform small rises and falls over the course of time. Our simulation shows that the linear behavior stops at around 80 percent saturation. From here we do see small rises, occasionally approaching the high $80's$ and sometimes low $90's$.

This plateau occurs because when the ejecta blanket of one crater overlaps the center of another crater, the older crater is effectively destroyed. Once we reach our peak saturation values, it becomes extremely likely that a new crater will overlap an older one. We also notice dips in the saturation, because new craters can sometimes obliterate multiple older craters. We've seen that due to our stopping condition, we usually end the simulation at around $450,000$ years. If we ran the program indefinitely, we would see the same plateau behavior. The saturation will never approach 100 percent because of the circular nature of the ejecta blankets.

In reality, a large variety of factors can affect a terrestrial surface. Lava flows, wind and water erosion, tectonics, planetary tilt, and more can all affect the surface crater saturation. Different sized craters and non-constant impacts would greatly affect the linear behavior of our graphs. Ignoring these effects, we've seen computational proof that surface saturation does eventually reach a limit.

6 Appendix

```
%% Jashan Chopra - 10/25/2018
% This script completes various tasks associated with the ASTR3750 Project

clc;
clear all;
clf;

%% Identifying Given Variables

% We want to represent a 500 km^2 region
rows = 500; % Each matrix element will be a 1km^2 region

% Crater Function
cdiameter = 50; % Diameter
cradius = cdiameter / 2; % Radius

% Ejecta Blanket Function
ebdiameter = 50*1.2; % Diameter [km]
ebradius = ebdiameter / 2; % Radius [km]
ebarea = pi * ebradius^2; % Area of Ejecta Blanket [km^2]

%% Preallocating Variables
iterations = 10000; % Num iterations

t = ones(iterations,1); % [1000 years = t]
num = ones(length(t),1); %[1 crater every 1000 years]
x = zeros(length(t),1); % X coordinate position
y = zeros(length(t),1); % Y coordinate position
sat = zeros(length(t),1); % Current time stamp saturation of surface
v = zeros(length(t),2); % Position vector
dist = zeros(length(t),1); % Distance vector

% Creates vector of doubled time values
tspan = 2 * ones(10,1);
for i = 1:10
```

```
36      tspan(i+1) = tspan(i) * 2;
37  end
38
39  %% Simulating Crater Impacts
40
41  % Every year 1 crater impacts, with a center (x,y) and an eject blanket
42  % diamter of 60km
43  for i = 1:length(t)
44
45      % Iterate time
46      t(i+1) = t(i) + 1;
47
48      % Computes the number of craters
49      num(i) = nnz(y); % Number of nonzero elements of y
50
51      % We want to find the x and y coordinates of an impact at random
52      x(i) = randi(500,1,1);
53      y(i) = randi(500,1,1);
54
55      % Construct a vector from the new crater x & y coordinates to all other
56      % crater x & y coordinates
57      for j = 1:(i-1)
58          % Resets vector value for next crater
59          v(i,1) = 0;
60          v(i,2) = 0;
61
62          % Find the distance between each crater center
63          v(i,1) = x(i) - x(j); % distance in x
64          v(i,2) = y(i) - y(j); % distance in y
65          dist(i) = sqrt((v(i,1)^2) + (v(i,2)^2)); % sum distance in quadrature
66
67          % If the distance is between -30 to 30km (radius of ejecta) to
68          % another crater, delete that craters coordinates
69          if dist(i) < 30 && dist(i) > -30
70              x(j) = 0; % Sets that crater x coordinate to 0
71              y(j) = 0; % Sets that crater y coordinate to 0
72          end
73      end
```

```
74
75      % total area
76      totarea = ebarea * sum(num(i));
77
78      % Crater saturation is taking into effect portions of craters outside
79      % of the 500km space
80      % crater saturation
81      sat(i) = totarea / (rows + ebdiameter)^2;
82      sat(i) = sat(i) * 100; % Convert to percent
83      fprintf('Saturation of surface is %f percent\n',sat(i))
84
85      % If the change in num of craters was less than 5% after time double
86      if ((num(i) - num(round(i/2))) / num(i)) < .05
87          tfin = t(i);
88          break
89      end
90  end
91
92  x(x==0) = []; % Delete all 0 values from x
93  y(y==0) = []; % Delete all 0 values from y
94  sat(sat==0) = []; % Delete all 0 values from saturation
95      sat(1) = 0; % Make the first value 0
96
97  %% Plots each center, its crater, and its ejecta blanket
98  for i = 1:(length(x))
99      % Plot the center point
100     plot(x(i),y(i),'*')
101     % linkdata on % Updates every time a new crater is made
102         xlabel('X Coordinate') % Gives x label
103         ylabel('Y Coordinate') % Gives y label
104         title('Positions of Craters') % Creates title
105         axis([-30 530 -30 530]) % Axis shows wider view
106         hold on
107
108        % Plot the crater
109         center = [x(i),y(i)];
110         viscircles(center,25);
111
```

```
112         % Plot the ejecta blanket
113          viscircles(center,30,'Color','b');
114 
115 end
116 %% Create a table of results
117 
118 % Create table with x values, y values for each crater
119 locations = table(x,y);
120 display(locations); % Outputs table to window
121 
122 % Table with final results
123 results = table(num(tfin), sat(tfin-2),tfin*1000);
124 results.Properties.VariableNames = ...
125     {'Craters','Saturation','Time'}; % Renames Variables
126 display(results); % Outputs table to window
127 
128 %% Displays the number of comets & End
129 
130 fprintf('Number of craters is %d\n',num(tfin))
131 fprintf('Final Saturation is %f percent\n',sat(tfin-2))
132 fprintf('Final Time is %d Years\n',tfin * 1000)
133 
134 %% Plot Saturation and Craters over time
135 
136 hold off;
137 plot(linspace(0,length(sat),length(sat)),sat)
138 title('Saturation over time')
139 xlabel('1000s of Years')
140 ylabel('Percent Saturation')
141 xlim([0 tfin])
142 %{
143 plot(linspace(1,tfin,tfin),num(1:tfin))
144 title('Number of Craters over time')
145 xlabel('1000s of Years')
146 ylabel('Number of Craters')
147 xlim([0 tfin])
148 %}
```