For loops and arrays

% Construct the first 25 numbers in the
% Fibonacci sequence and store them
% in the array fib

fib = zeros(1,25);
fib(1) = 1;
fib(2) = 1;

for i=3:25
    fib(i) = fib(i-1) + fib(i-2);
end

% Display the sequence
for i=1:25
    disp(fib(i))
end

// Construct the first 25 numbers in the
// Fibonacci sequence and store them
// in the array fib

#include <iostream>
#include <fstream>
#include <cmath>
using namespace std;

main() {
    int fib[25];
fib[0] = 1;
fib[1] = 1;

    int i;
    for (i=2; i<25; i++) {
        fib[i] = fib[i-1] + fib[i-2];
    }

    // Display the sequence
    for (i=0; i<25; i++) {
        cout << fib[i] << endl;
    }
}
Model rocket

% Model rocket simulation

%% The thrust function was defined in the file thrust_c11.m Here is its definition:
% function [thrust] = thrust_c11(t)
% %
% if t < .3
%    thrust = 22/.3*t;
% elseif t < .4
%    thrust = 22-10/.1*(t-.3);
% elseif t < .7
%    thrust = 12-2/.3*(t-.4);
% elseif t < .8
%    thrust = 10-10/.1*(t-.7);
% else
%    thrust = 0;
% end

// Model rocket simulation

#include <iostream>
#include <fstream>
#include <cmath>
using namespace std;

double thrust_c11(double t) {
    double thrust;
    if (t < .3) {
        thrust = 22/.3*t;
    } else if (t < .4) {
        thrust = 22-10/.1*(t-.3);
    } else if (t < .7) {
        thrust = 12-2/.3*(t-.4);
    } else if (t < .8) {
        thrust = 10-10/.1*(t-.7);
    } else {
        thrust = 0;
    }

    return thrust;
}
Model rocket, continued

% The mass function was defined in the file mass_c11.m Here is its definition:
% function \[ mass \] = mass_c11(\( t \))
% if \( t \leq 0 \)
% \( mass = 30.4; \)
% else if \( t \geq .8 \)
% \( mass = 18.9; \)
% else
% \( times = [0:.001:t]; \)
% \( thrust = 8.8; \)
% for \( time = 1:\text{round}(t/.001)+1; \)
% \( thrust = thrust - .001*\text{mass}_c11(times(time)); \)
% end
% \( mass = 18.9 + (30.4-18.9)*\text{thrust}/8.8; \)
% end
% \( mass = mass/1000; \)

dt = .001;
g = 9.8;
rho = 1.22;
mass_rocket = .02835;
radius_rocket = .041;
radius_parachute = .1;
coef_drag_rocket = .75;
coef_drag_parachute = .85;

t = zeros(1,1000);
h = zeros(1,1000);
v = zeros(1,1000);

double mass_c11(double t) {
    double mass, thrust, time;
    if (t <= 0) {
        mass = 30.4;
    }
    else if (t >= .8) {
        mass = 18.9;
    }
    else {
        thrust = 8.8;
        for (time = 0; time <= t; time += .001) {
            thrust = thrust - .001*mass_c11(time);
        }
        mass = 18.9 + (30.4-18.9)*thrust/8.8;
    }
    mass = mass/1000;
    return mass;
}

main() {
    double dt = .001;
    double g = 9.8;
    double rho = 1.22;
    double mass_rocket = .02835;
    double radius_rocket = .041;
    double radius_parachute = .1;
    double coef_drag_rocket = .75;
    double coef_drag_parachute = .85;
    // Create arrays that are big enough
    // to store everything
    double t[100000];
    double h[100000];
    double v[100000];
Model rocket, continued

\[ t(1) = 0; \]
\[ h(1) = 0; \]
\[ v(1) = 0; \]
\[ i = 1; \]

while \((h(i) > 0) \mid (v(i)>0)\)
  \[ i = i + 1; \]
  \[ t(i) = t(i-1) + dt; \]
  \[ mass = mass\_rocket + mass\_c11(t(i)); \]
  if \(v(i-1) > 0\)
    \[ area = \pi*radius\_rocket^2; \]
    \[ drag = .5*rho*coef\_drag\_rocket*area*v(i-1)^2; \]
    \[ force = -g*mass + thrust\_c11(t(i)) - drag; \]
  else
    \[ area = \pi*radius\_parachute^2; \]
    \[ drag = .5*rho*coef\_drag\_parachute*area*v(i-1)^2; \]
    \[ force = -g*mass + thrust\_c11(t(i)) + drag; \]
end

acceleration = force/mass;

if \((h(i-1) == 0) \&\& (acceleration < 0)\)
  acceleration = 0;
end

\[ v(i) = v(i-1) + acceleration*dt; \]
\[ h(i) = h(i-1) + v(i-1)*dt; \]
end

t[0] = 0;
h[0] = 0;
v[0] = 0;
int i = 0;
double mass, area, drag, force, acceleration;

while \(((h[i] >= 0) || (v[i] >= 0)) \&\& (i<100000)\) { 
  i = i+1;
  t[i] = t[i-1] + dt;
  mass = mass\_rocket + mass\_c11(t[i]);
  if \(v[i-1] > 0\) {
    area = 3.14159265*pow(radius\_rocket, 2);
    drag = .5*rho*coef\_drag\_rocket*area*pow(v[i-1], 2);
    force = -g*mass + thrust\_c11(t[i]) - drag;
  } else {
    area = 3.14159265*pow(radius\_parachute, 2);
    drag = .5*rho*coef\_drag\_parachute*area*pow(v[i-1], 2);
    force = -g*mass + thrust\_c11(t[i]) + drag;
  }
  acceleration = force/mass;
  if \((h[i-1] == 0) \&\& (acceleration < 0)\) {
    acceleration = 0;
  }
  v[i] = v[i-1] + acceleration*dt;
  h[i] = h[i-1] + v[i-1]*dt; 
}
Model rocket, continued

```cpp
% Plot the height of the rocket
plot(t, h)
```

```
// Save the data to a file so we can analyze it
// in Matlab
ofstream oFile("rocket.dat");
int j;
for (j=0; j<i; j++) {
    oFile << t[j] << " " << h[j] << endl;
}
oFile.close();

// Plot the height of the rocket
// Execute the following in Matlab to load
// data and plot it.

// load rocket.dat
// plot(rocket(:,1), rocket(:,2))
```