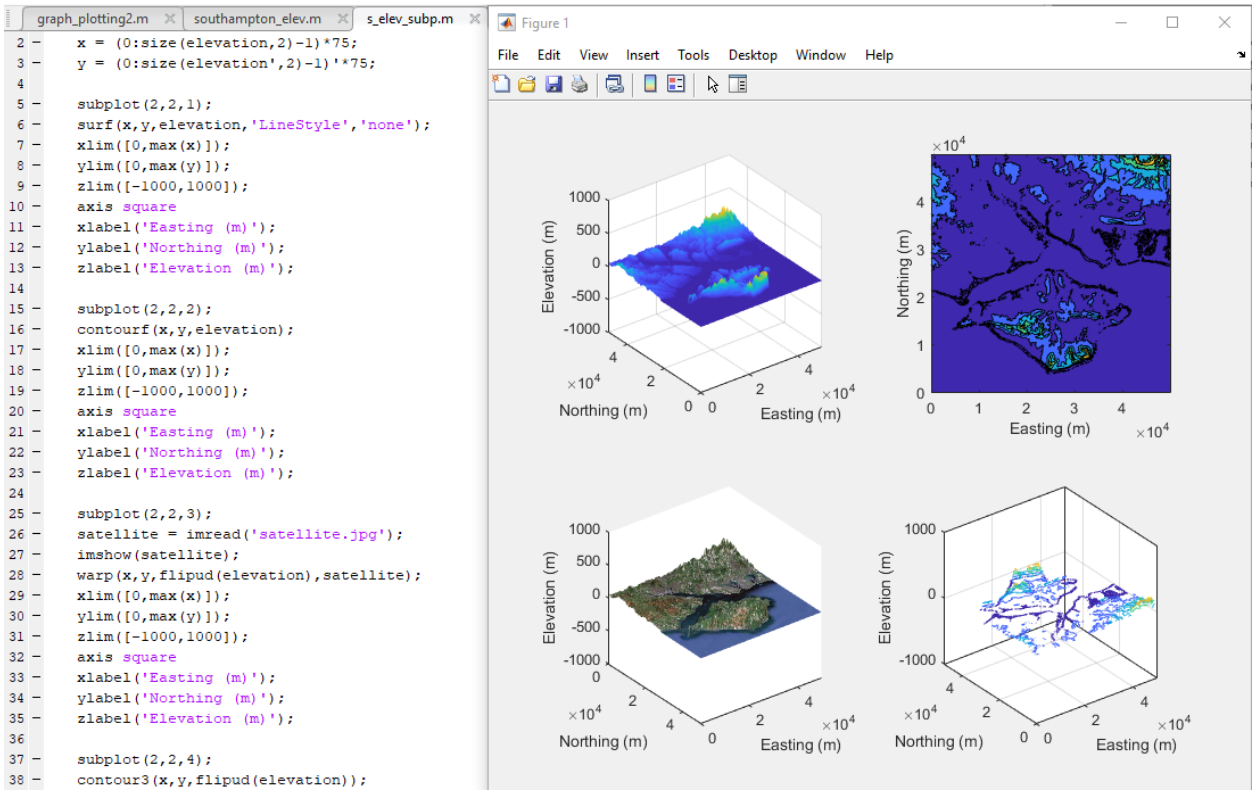
Including subplots and neatly lying out my code I was able to get all of my graphs into one figure and to fix the amplified scaling of the previous example. My data is all clearly displayed and understandable, making it easy to interpret.



Operators and Arrays
https://secure.ecs.soton.ac.uk/notes/ellabs/1/c3/c3.pdf
https://en.wikipedia.org/wiki/Floating-point_arithmetic
https://www.tutorialspoint.com/cprogramming/c_arrays.htm
https://stackoverflow.com/questions/4955198/what-does-dereferencing-a-pointer-mean/4955297
https://www.tutorialspoint.com/cprogramming/c_arrays.htm
https://stackoverflow.com/questions/4955198/what-does-dereferencing-a-pointer-mean/4955297

Inside rnd() the integer RAND_MAX must be turned into a float because the function returns a float.

The simulated evolution should stop when the best solution found yields a y-value no more than EPSILON away from the target value, or the maximum number of generations (MAX_GEN) has been reached. Write down the condition in the while-loop you will need for this.

best_ifit > EPSILON && gen < MAX_GEN

What part(s) of Darwin's algorithm (Reproduction, Variation, Selection) is happening in the for-loop inside main()? I Write down the condition needed in the if-statement inside main().

Selection.

Code for initpop and offspring, filled in the skeleton code.

```
86  // Returns a random value between 0.0 and 1.0
87  float rnd()
88  {
89      return rand() / (float)RAND_MAX;
90  }
91
92  //Sets up an a array with all posiitions a random number
93  void initpop(float *pop, int size)
94  {
95      for (int i = 0; i < size; i++)
96      {
97          *(pop + i) = rnd();
98      }
99  }
100
101 //repopulates the array with values close to the best of the last generation
102 void offspring(float parent, float mutst, float *pop, int size)
103 {
104     //parent is best value of last round
105     *pop = parent;
106     //fills the rest of the array with variations on the best value
107     for (int i = 1; i < size; i++)
108     {
109         float rand_mut = (2*rnd() - 1)* mutst;
110         *(pop + i) = parent + rand_mut;
111     }
112 }
```