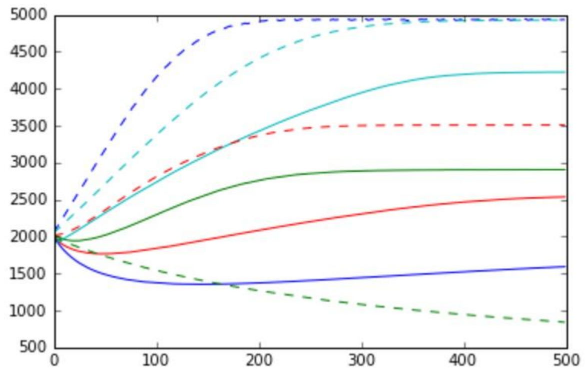
CX4230 Project 3D Summary

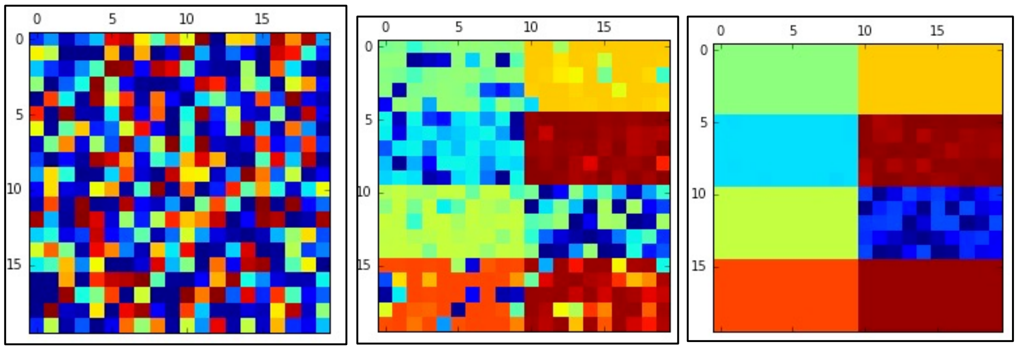
By Kartikeya Kumar and Karan Shah

What we have implemented:

Population change: We implemented the differential equations mentioned in our conceptual model. At every time step, the population change is dependent on the birth and death rates of the country that the cell is in. There is a hard limit according to the max population allowed in the cell. If the population is going to reach that limit, the function just subtracts the number of people died instead of adding more people. Depending on the death rate, the number subtracted from the limit is different for different countries. Because the rates are country dependent (all cells in the country have same rates) and the only difference is the initial populations, all cells in a country reach the same population eventually and the population stops changing. This will not happen once we implement migration functionality.
The rates are initialized in **initialize˙country˙properties()** function. The differential equation is modeled in the **pop˙change()** function. For 3D, we have added a method **showPopChange()** at the end which runs the population change simulation for 500 iterations (can be changed). It shows the state of the world at different points in time and plots the population-time plots for different countries.
Here is a sample population change with 8 countries having different birth and death rates (the rates are assigned randomly, but we can change is the max value of the random number that is generated):

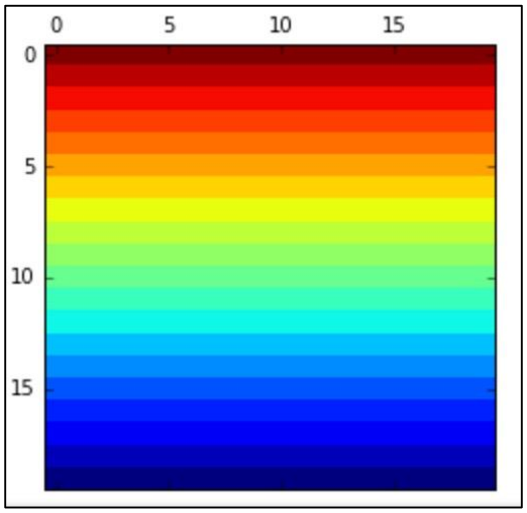


(Population on Y axis, Time steps on X axis)



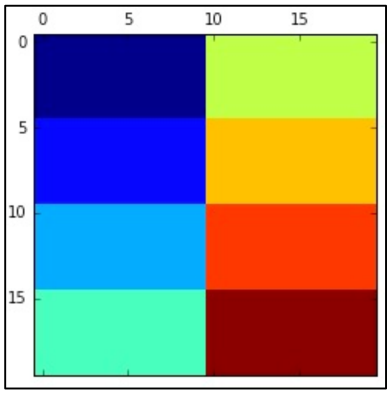
Here, the first image is the population grid at time t = 0, t = 400 and t = 1000 respectively. At t = 0, the cells have a big variance in population. However, as time increases, the populations reach a steady state and we can even infer the shape of countries using population distributions.

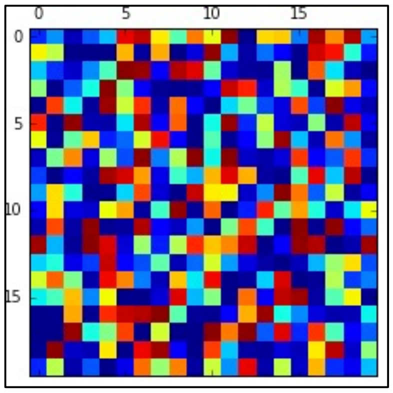
Temperature: In the **distribute˙values()** method, we implemented a small portion of code that assigns a temperature value to each cell. This is a very simple piece of code that assigns the same temperature to each row of the matrix and decreases as we go down the rows. It will be used in the utility function later.

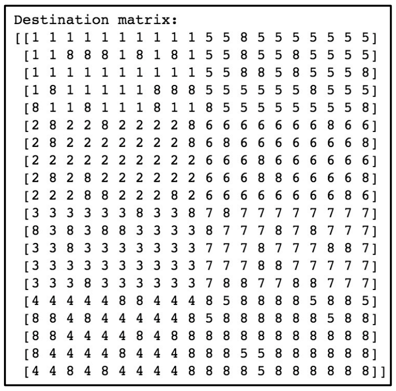


Temperature distribution for a 20X20 world. Red denotes 0 C, blue denotes 30 C. Each row is like a latitude and the climate is a linear function of latitude for this model.

Towards migration:
We implemented a method called **findDestMatrix()** that determines which country is most desirable for each people in each cell of the grid. It returns a grid of the same size consisting of country numbers which are most desirable for people in that cell(in our world grid) to move to. To determine the most desirable country, this method uses criteria such as difference in wealth and ideal population density between the cell and the destination country by assigning weights to both these criteria each. The higher the aggregate score, the more 'desirable' the country. The weights to both the criteria each are passed in as parameters and can be changed by the user. Refer to conceptual model to see how we use wealth and population to calculate utility.







```
Destination matrix:
[[1 1 1 1 1 1 1 1 1 1 5 5 8 5 5 5 5 5 5 5]
 [1 1 8 8 8 1 8 1 8 1 5 5 8 5 5 8 5 5 5 5]
 [1 1 1 1 1 1 1 1 1 1 5 5 8 8 5 8 5 5 5 8]
 [1 8 1 1 1 1 1 8 8 8 5 5 5 5 5 5 8 5 5 5]
 [8 1 1 8 1 1 1 8 1 1 8 5 5 5 5 5 5 5 5 8]
 [2 8 2 2 8 2 2 2 2 8 6 6 6 6 6 6 6 8 6 6]
 [2 8 2 2 2 2 2 2 2 8 6 8 6 6 6 6 6 6 6 8]
 [2 2 2 2 2 2 2 2 2 2 6 6 6 8 6 6 6 6 6 6]
 [2 8 2 8 2 2 2 2 2 2 6 6 6 8 8 6 6 6 6 8]
 [2 2 2 8 8 2 2 2 8 2 6 6 6 6 6 6 6 6 8 6]
 [3 3 3 3 3 3 8 3 3 8 7 8 7 7 7 7 7 7 7 7]
 [8 3 8 3 8 8 3 3 3 3 8 7 7 7 8 7 8 7 7 7]
 [3 3 8 3 3 3 3 3 3 3 7 7 7 8 7 7 7 8 8 7]
 [3 3 3 3 3 3 3 3 3 3 7 7 7 8 8 7 7 7 7 7]
 [3 3 3 8 3 3 3 3 3 3 7 8 8 7 7 8 8 7 7 7]
 [4 4 4 4 4 8 8 4 4 4 8 5 8 8 8 8 5 8 8 5]
 [8 8 4 8 4 4 4 4 4 8 5 8 8 8 8 8 8 5 8 8]
 [8 8 4 4 4 4 8 4 8 8 8 8 8 8 8 8 8 8 8 8]
 [8 4 4 4 4 8 4 4 4 8 8 8 5 5 8 8 8 8 8 8]
 [4 4 8 4 8 4 4 4 4 8 8 8 8 5 8 8 8 8 8 8]]
```

The first figure shows 8 countries on a grid of size 20x20. The second figure shows population distribution of each of the 400 cells. The third image shows the destination matrix, a 20x20 grid of country numbers that people of corresponding cells find the most desirable based on the utility function.

Finally, we also implemented a method called **globalDesirability()** that returns a list of the top 3 locations in each country by wealth on the grid. This will be important for implementing the migration part in the next milestone where when a person wants to move to another country, this method can be called to find the list of most desirable locations in that country so the person can move to it. The method, currently, only uses wealth to find the 3 most desirable locations in each country. However, this can be easily modified to include other criteria as well.

So, both the above mentioned methods will be used to find locations in other countries to move to. We will move a small proportion of people from each cell to one of the three wealthiest cities in their desired countries. This is not ideal because if we keep moving people to 3 major cities per country, they will reach their population capacity quickly and migration will stop.

Implementation Problems

We have figured out how to model people migrating to other countries. This was achieved by making big simplifications (people will only move to the 3 wealthiest regions in their country of choice).
However, to model migration within the country, we have to check the utility of each cell with respect to every other cell. This makes it very intensive computationally. We haven't been able to model this yet.

Ideally, there should be a distance term in the utility function that decreases utility as distance between the source cell and destination cell increases. For the whole grid, we want to see the relation of each cell with difference to each other cell. We are looking into better ways to implement this. We might reduce the number of variables and distributions that have made this project complex and vague in order to focus on coming up with a good algorithm to calculate all the utility scores.