Lab Notebook: Range shifts and environmental gradients

**May 26, 2017**

Choosing a model for the genetic architecture

Options

1. Individuals are haploid and asexual. Individuals have a value for each trait which is inherited in offspring with some normally distributed, small mutation effect.
   1. This approach is used by ecological theorists it seems and ignores genetic architecture per se, but is still a useful simplification.
2. Individuals reproduce via sexual reproduction, but have no underlying genetic architecture. Instead, each individual is defined by a breeding value for each trait which is then translated into a phenotype according to an environmental variance parameter. Mid-parent breeding values are inherited according to local genetic variability, which is related to heritability.
   1. This approach would be equivalent to the model for habitat heterogeneity.
3. Individuals are monoecious (hermaphroditic) and diploid with both sexual and asexual (self-fertilization) reproduction possible. Individuals are defined by *kT* quantitative trait loci for each trait *T*. Allele values at loci are real numbers and the total trait value is an additive combination from all the loci under some relevant transformation (stabilizing selection for fitness or simply a logit-like transformation as I used for the diffusion coefficient in the heterogeneity model). For simplicity, all loci segregate independently so offspring inherit one random copy from each parent. This assumption is fine as long as *kT* doesn’t get too large at which point it might be unrealistic (I’m not exactly sure what “too large” really is, but I’m thinking somewhere on the order of 10-100 loci makes for a reasonable assumption).
   1. I think this approach is the way to go for several reasons
      1. This approach is more realistic than the first two in describing the underlying genetic processes without adding *too* many additional parameters.
      2. It allows for a relatively easy extension to consider gene surfing (i.e. either add in mutations to the fitness trait directly as in Henry et al. 2015 or include an additional set of loci subject to deleterious mutations and make fitness a product of the effects in both set of loci as in Gilbert et al. 2017).
   2. For this approach, I will need to read closer about how Gilbert et al. 2017 handled environmental and mutational variance as well as setting things like heritability (if they did; I don’t remember).

How to model environmental gradients with a quantitative genetic fitness trait

I came up with an equation based on S shaped (logistic) functions similar to my equation to convert dispersal phenotypes to diffusion coefficients. It’s in the blue notebook. Right now, I am having trouble getting the correct limits and derivatives for this (or I am having trouble graphing it). Next time, solve for the derivative of the equation and the limits of the equation and the derivative as x approaches beta from either side. I want to find the conditions under which the equation value is approximately alpha and the derivative is approximately 0.

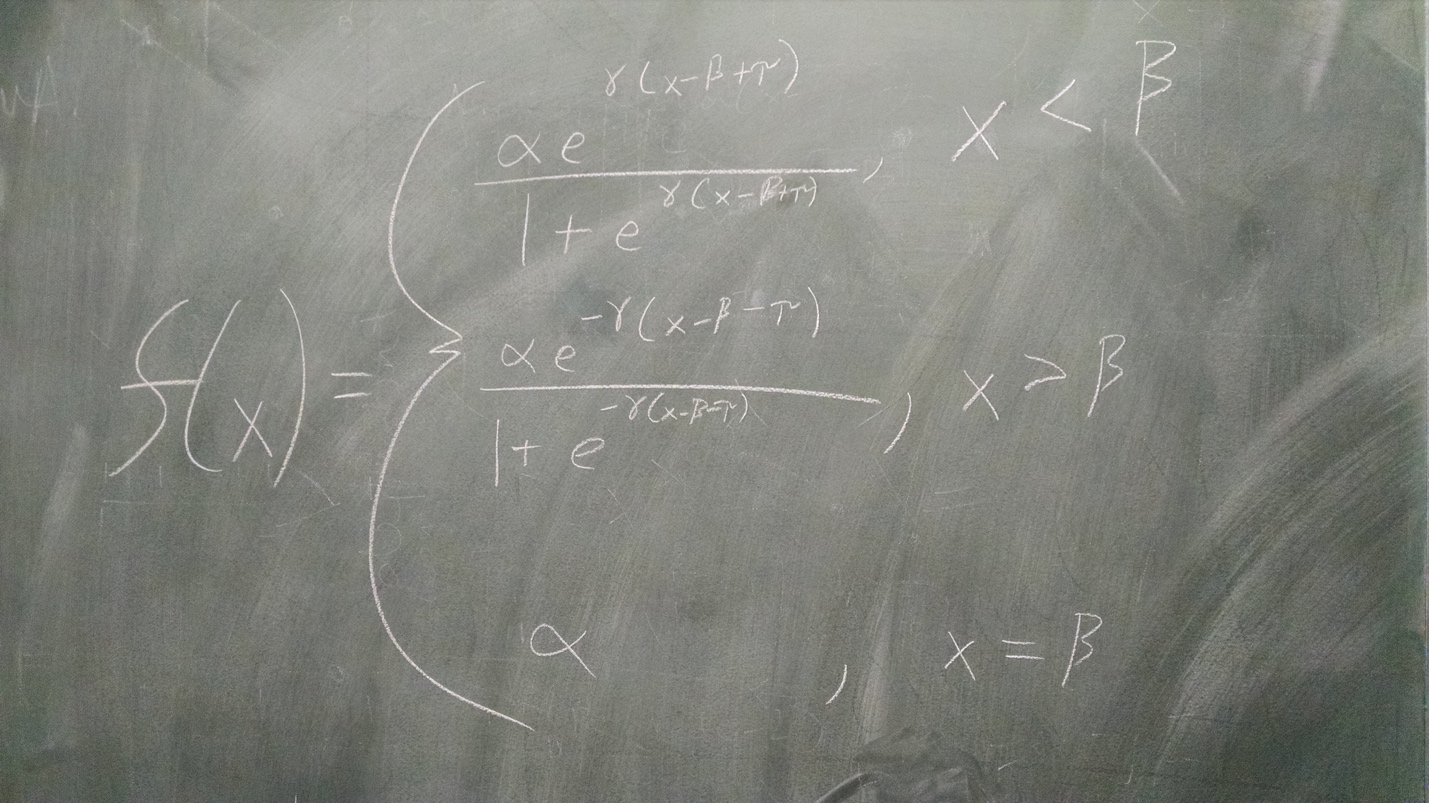
To control for varying range capacities with different environmental gradients, I should also look at the integral of Rmax. The goal of the study should be to vary the slope of the gradient while holding the total range capacity (area under the curve) constant to avoid confounding results of different gradients with a relationship to range capacity.

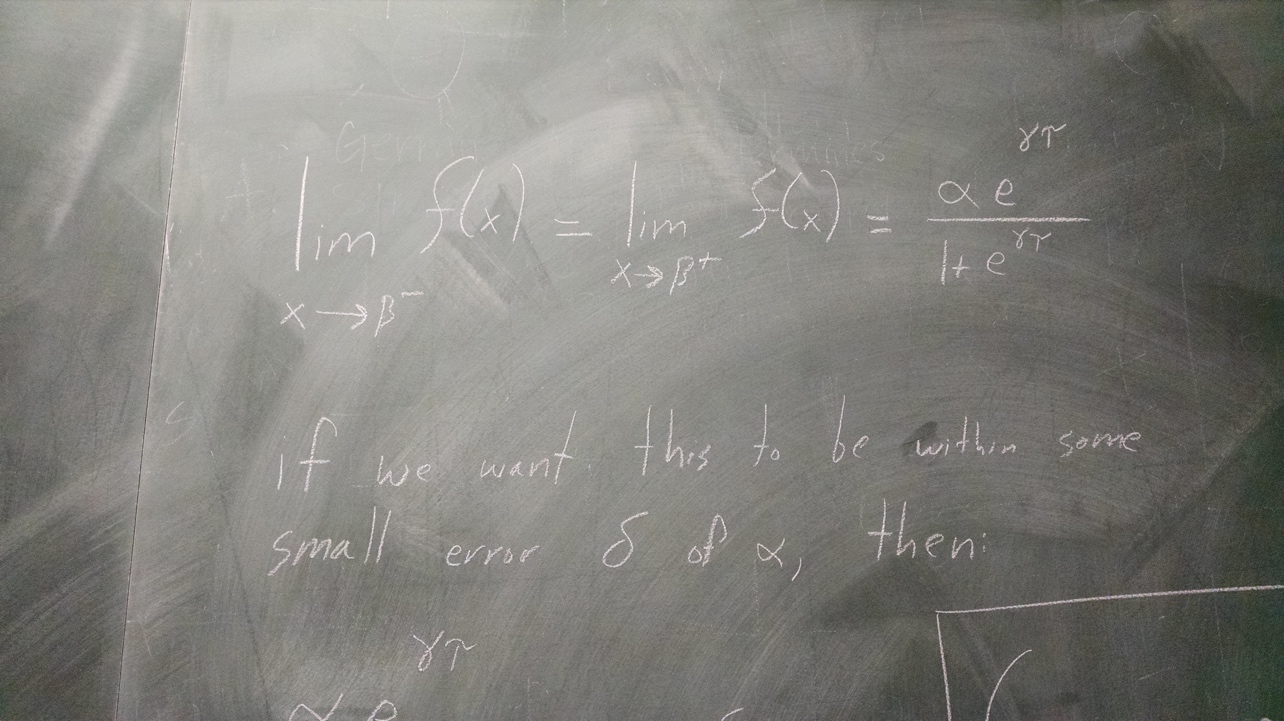
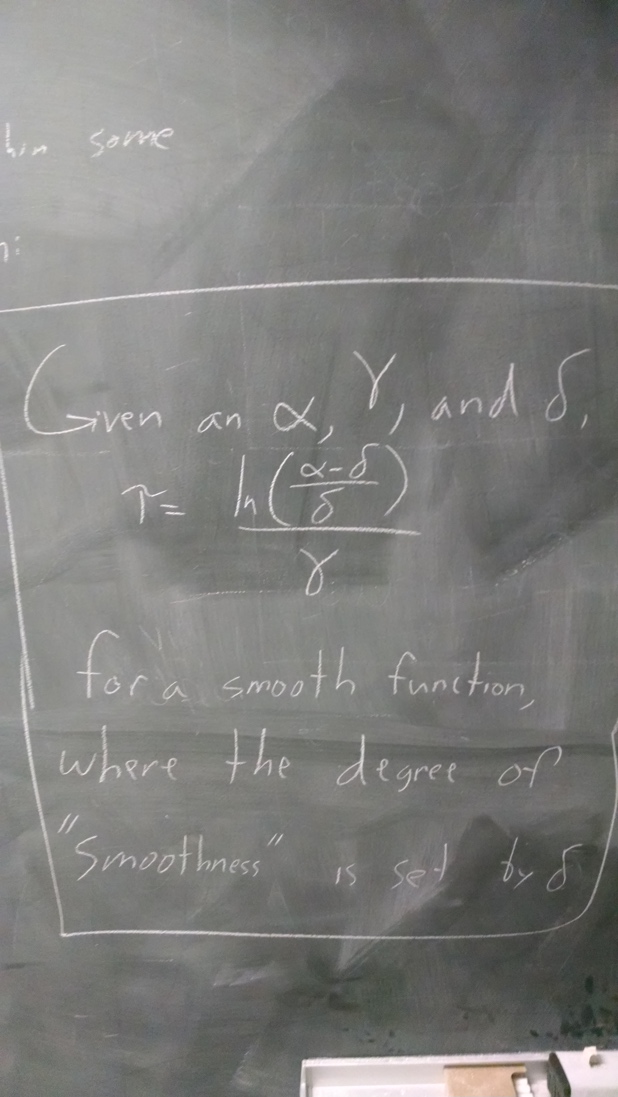
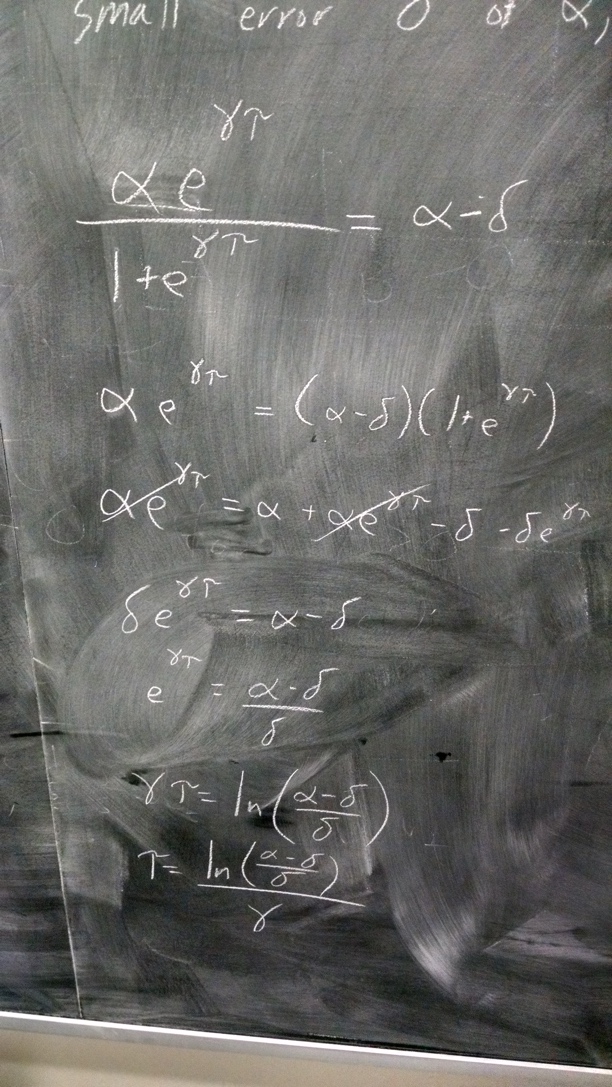
**May 28, 2017**

Deriving and exploring the function for range capacity

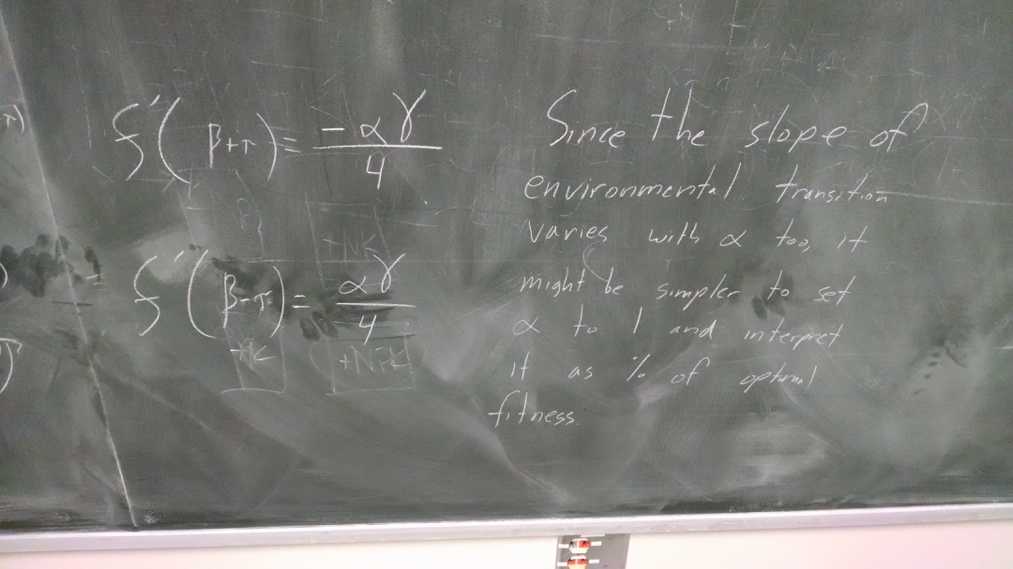
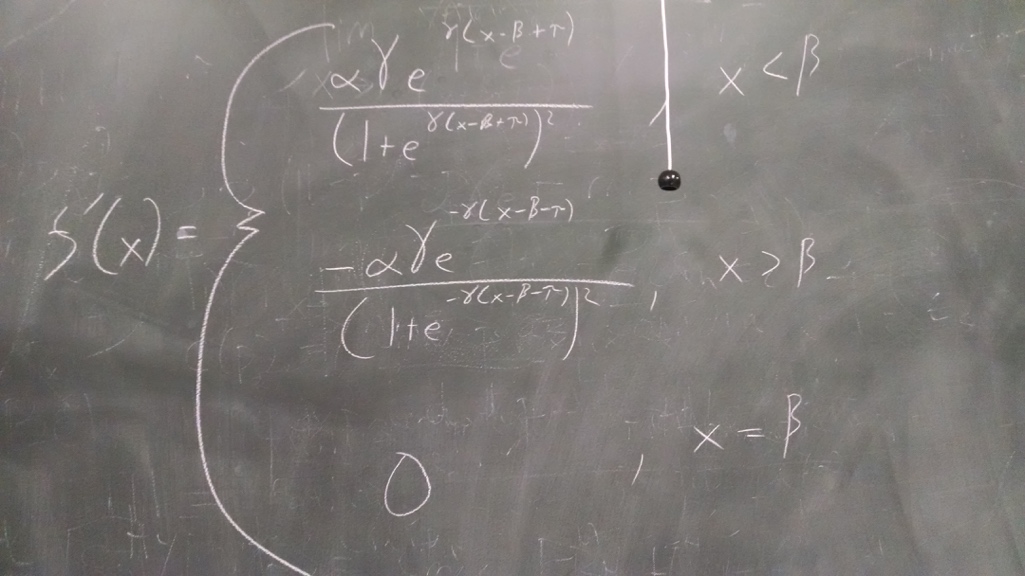
Today I worked on the blackboard primarily to explore the function I came up with for range capacity. Below are photos of what I came up with:

Function definition

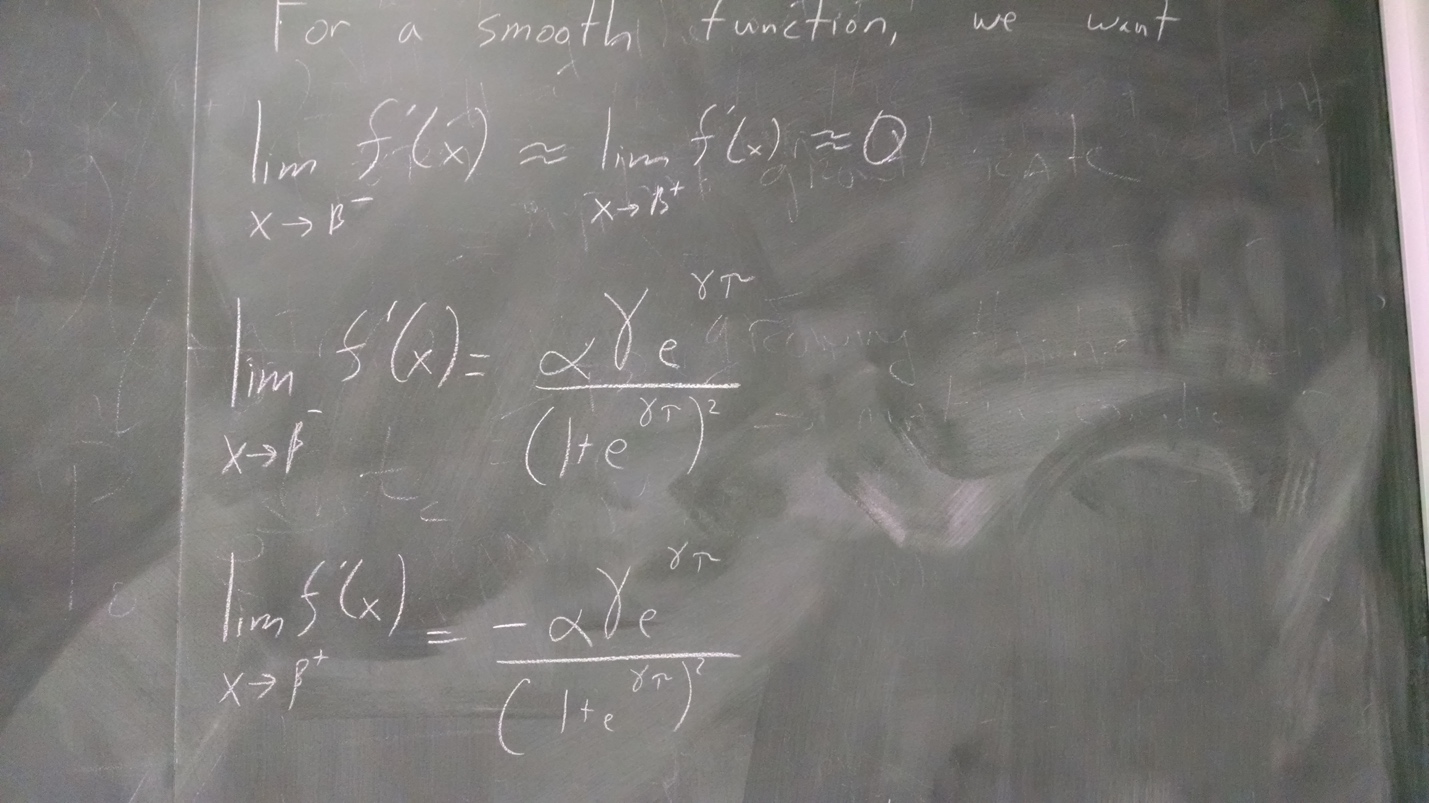


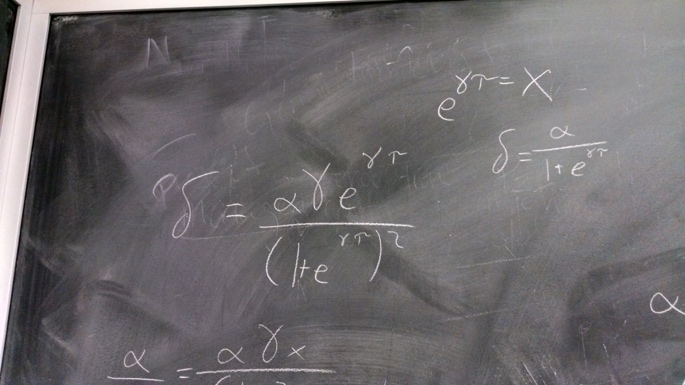
Calculating conditions under which the function would be “smooth” to within some approximation delta: 

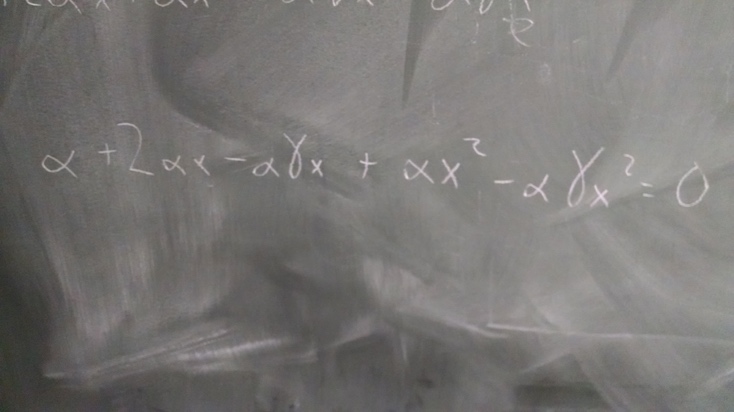
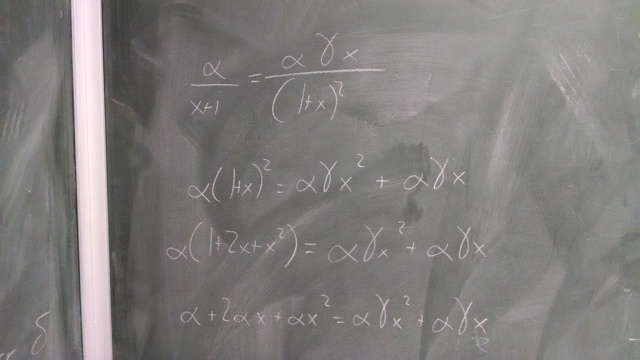
Exploring the derivative of the function:

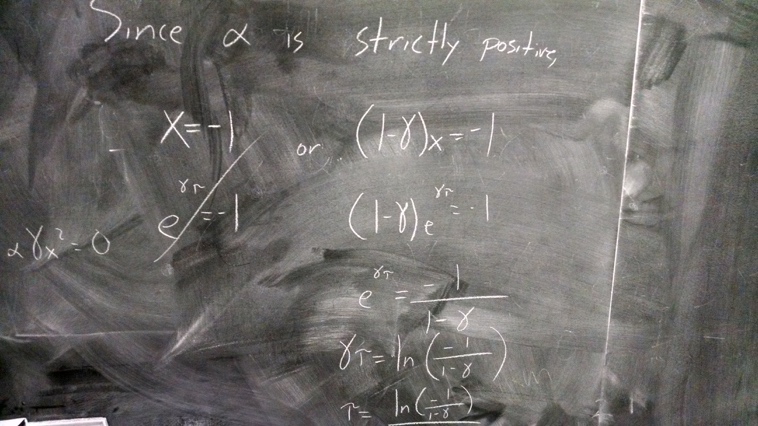
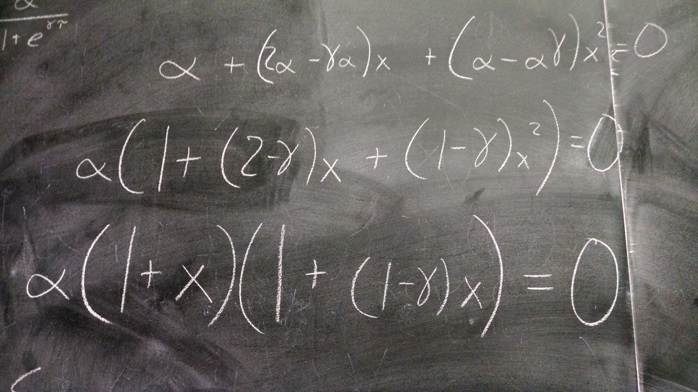


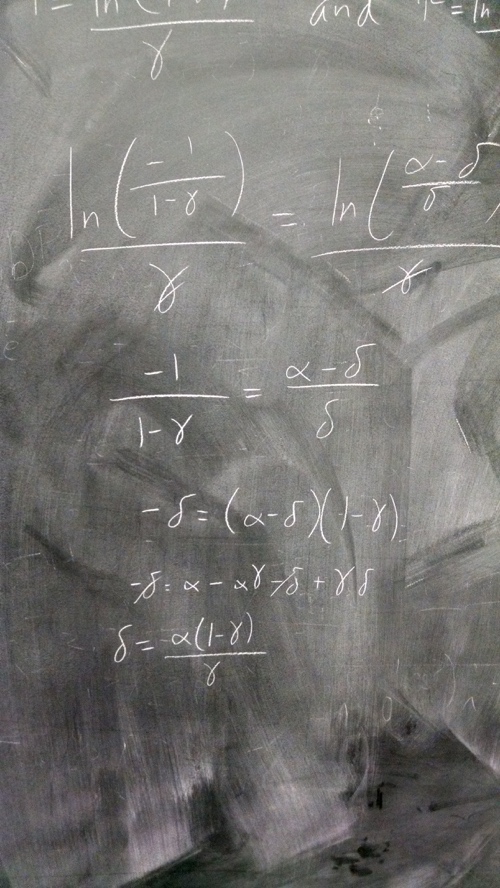
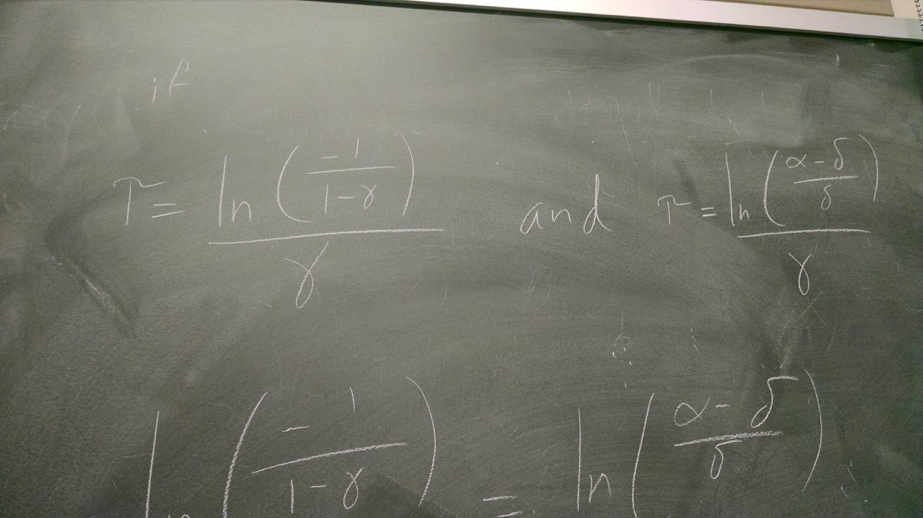
Trying out ideas for smoothness of the function using the derivative:

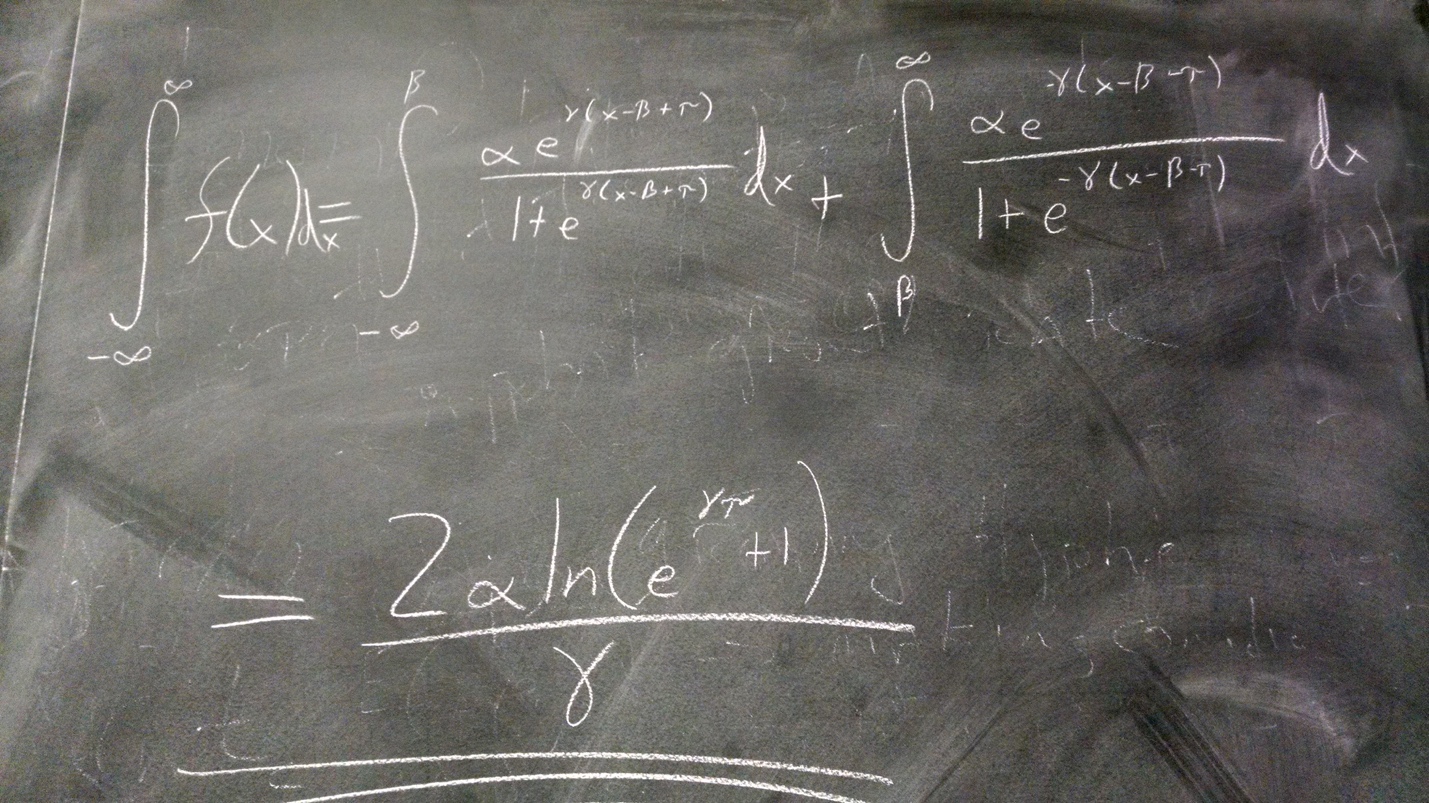


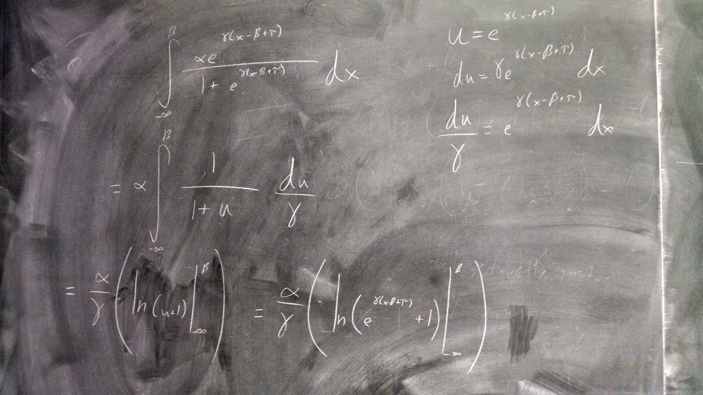


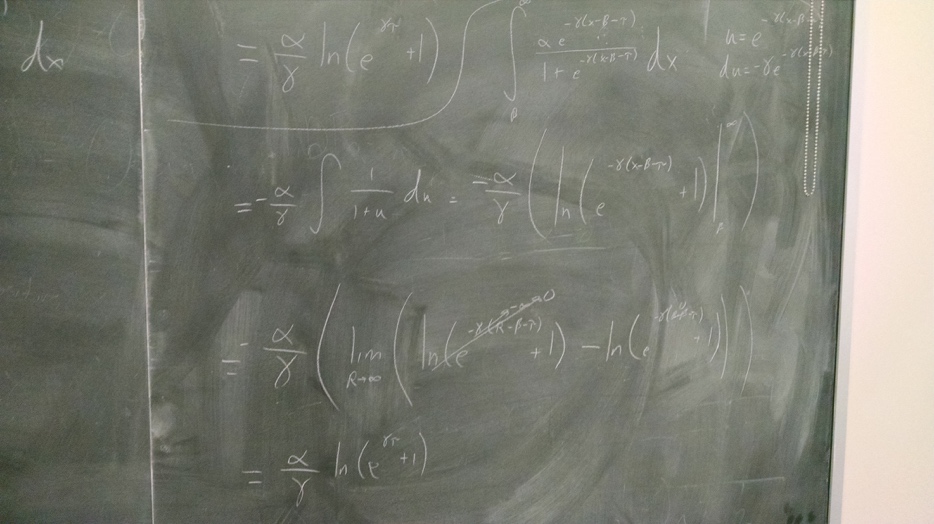




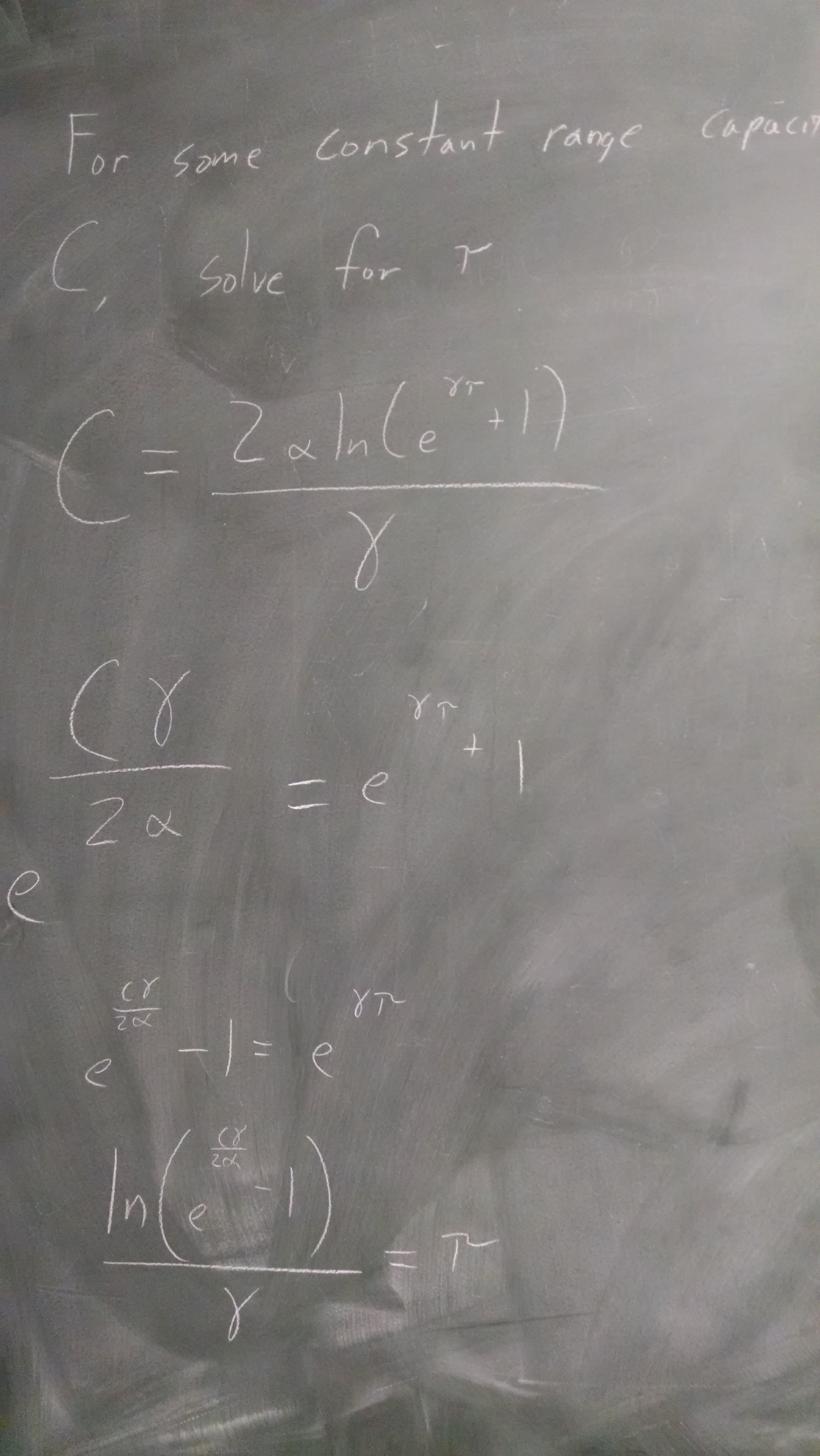


Calculating the integral of the Range Capacity function:  


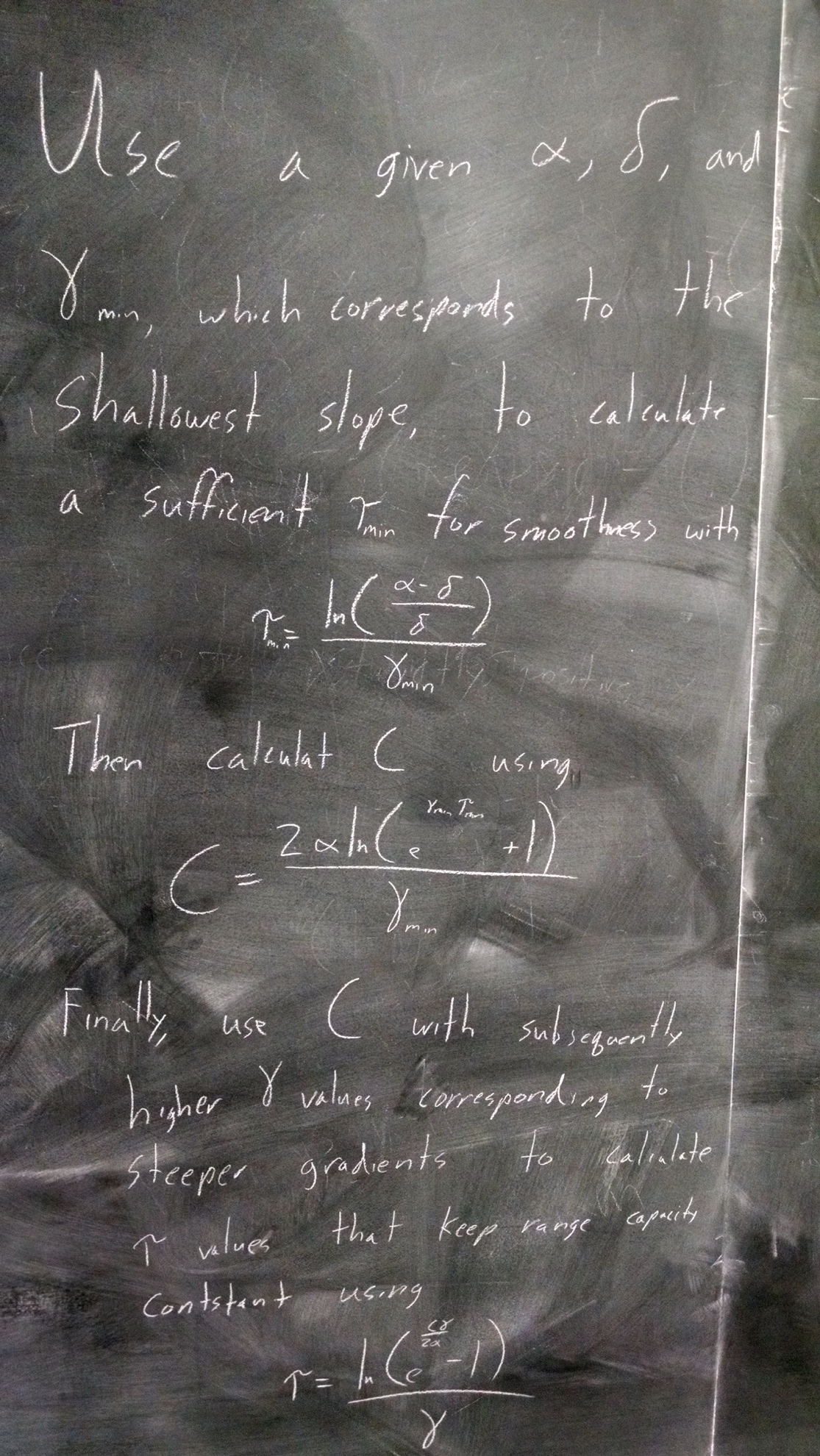




Calculating tau for a constant range capacity:



A proposed workflow for choosing parameter values:



All of the above pictures are saved in the RangeCapacityFunctionNotes folder in the RangeShifts project folder. For my next time, I think a good approach would be to write an R script to implement the above ideas for choosing parameter values and use that to make some preliminary graphs of what the different types of ranges would look like and then implement a changing beta term and make a plot of climate change shifting the range.

**May 31, 2017**

Writing up the details and generating an example plot

Today I wrote up the details described above in the blue notebook to show to Allison. I also created a script to use the equations to find parameter values and illustrate them graphically. The script is called RangeCapacityPlot.R and the figure is the RangeCapacity.pdf figure.

After meeting with Allison, I think the way to start is to treat space as discrete and use the integral of the function to come up with the overall patch value for each lattice point (or I could also simply use the midpoint of the function if the integral adds too much complication…). Also, beta should be set up as a vector with initial and changing periods of time. At first, I can just use a constant rate of change, but leave enough generality that non-constant change could be incorporated (e.g. speed ups and slow downs). I will have to look at the climate modelling literature to determine which of these is more realistic (I’m guessing the speed ups and slow downs, but it would be great to have a citation).

Things to do next time:

* Look for literature on constant vs. not rates of climate change
* Design the matrix I will use to keep track of individuals and their genetics
* Make a function to generate an empty matrix like in my other model
* Make functions to extract trait values for a given individual in the matrix

**June 5, 2017**

Figuring out the structure of the matrix I will use to track individuals

Proposal:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Natal Location (x) | Natal Location (y) | Post Dispersal Location (x) | Post Dispersal Location (y) | Sex (binary) | Fitness locus 1 | … | Fitness locus Nf | Dispersal locus 1 | … | Dispersal locus Nd |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |

I also created a github repository (ShiftingSlopes) for this project and an associated Rstudio project for all the code in the folder “~/Desktop/RangeShifts/ShiftingSlopes/”

In the github repository, there is now a script for all of the simulation functions. Currently it has a function to find safe IDs for folders to save results to and to generate population matrices according to the above format.

Next time I start, I should work on making a function to calculate a dispersal trait value based on the Disp loci and think about the best way to implement it in a discrete space lattice framework.

**June 6, 2017**

I added functions to calculate a dispersal trait value and to implement dispersal according to an unbiased diffusion process implemented via a Poisson process. Next time, I should work on functions to calculate range capacity values along the y axis of the landscape and implement a moving beta for climate change in a very general framework. The next step will then be to look back at the papers that have used multiple types of gradients (e.g. carrying capacity, intrinsic fitness, extinction rate, etc.) and think about which one is best for the current model. Along the same lines, I think I should use the logistic growth equation for this model as it is better known and has nice analytical properties.

**June 7, 2017**

So far today I have added a function to calculate the mean of the range capacity function f(x) over a given interval [a,b] using the Mean Value Theorem and started to work on a function to use that to generate a vector of environmental quality metrics across a range.

Formula for calculating the mean of f(x) over [a,b]: (the proof of this is in the blue notebook for now).

If b <= β:

if a >= β

if b > β & a < β:

I have also thought more about how to treat space in this model. I think I will keep space as discrete patches, but allow for a mapping between discrete and continuous space. Basically, by default, patches will be 1 unit long and wide, and centered at whatever patch number it corresponds to on the x axis for the function f(x). I can then use a constant multiplicand to make patches wider (i.e. taking up more space on the x axis) or smaller. In this way, I can adjust them to be a reasonable size for whatever scale the range ends up taking on the x axis and then also devote an analysis to the effect of discretizing space by allowing this constant to become smaller and smaller and see what it does to my results.

For next time, I will concentrate on finishing the implementation of space in the model and adjust the dispersal function to use a dispersal kernel of some sort (or just use both negative exponential and student’s t distributions or some other two-three distributions proactively). Also, I need to look back at the papers looking at different types of gradients to decide how to implement this one and start to work on the fitness component.

**June 19, 2017**

Today I adjusted the dispersal function to work with dispersal kernels. Currently, it can work with either exponential, normal, or student’s t dispersal kernels. I also finished some of the functions on getting environmental quality values for patches that I started before.

Next time, look through the literature and decide what type(s) of gradients I want to implement (i.e. gradient in carrying capacity, growth rate, extinction probability, etc.). Once I decide, I need to start working on the fitness functions. Also, I should start coming up with unit tests I will use on the individual functions once I get to the quality control part. Finally, I need to email Courtney to confirm that we’re not accidentally working on the same things

**June 21, 2017**

I will implement the range capacity function so that it can apply to either carrying capacity (K) or growth rate (R). The Henry et al. (2013) paper suggests that these should be broadly similar, but could lead to differences in the initial population distribution. Reproduction will occur according to a standard logistic function for a given patch (i.e. the expected number of offspring will follow the logistic distribution and the actual number will be drawn from a Poisson distribution). Mating pairs will be drawn randomly from the patch according to individual fitnesses to produce a single offspring. This process will be repeated until all offspring have been assigned parentage. Individual fitness will be determined according to stabilizing selection acting on the additive fitness loci. The local fitness optimum as it varies with location x (zopt(x)) will be a function of the range capacity as follows:

This ensures a gradient for local adaptation, with equal reproductive potential on either side of the range, but with the local optimum on one end different from the other (modeling the idea of a north/south gradient for local adaptation).

**June 22, 2017**

Today I spent a lot of time putting together some of the remaining functions needed for a simulation (including the skeleton of a main function to run a single, full simulation). I also emailed Courtney for clarification.

Next time, finish the Reproduce function according to the comments in the code. Also, adjust it so that monoecious scenarios allow for self fertilization, but maybe only if there is only a single individual (or regardless? I should think more about this). Make sure I have all the functions I need for the main simulation function and make a list of all the relevant parameters to use with the SaveParams function.

**Summary through July 19, 2017**

I forgot to update this document for a few weeks, so I’m just going to supply a summary of what’s been accomplished and what the next steps are. First, all the functions necessary for the model as initially conceived have been written, though not all have been thoroughly vetted. A description of the model following the ODD protocol has been written and received some preliminary feedback from Allison. Where to go from here can be broken down into immediate, mid-term, and long-term goals as follows.

*Immediate Goals*

* Complete the code for a single, “master function” that will use all the previously made functions to run a single simulation. The skeleton of this function exists, but needs to be updated with the necessary parameters for each function and a mechanism for saving data during the model run.
* Write an example script defining parameters and passing them to the model for some number of simulations.
* Add a subfolder to the Github repository for model testing. This folder will contain a description of all the model tests that will be run, R scripts to perform and record the results of each test, and an Rmarkdown document summarizing the tests and the results of them.

*Mid-term Goals*

* Choose three levels of both γ and λ that will be used in a factorial design to explore the interacting effects of slope of the range boundary and the gradient in local selection within the range on stationary population dynamics. This will be for a gradient in R initially.
* Under these different values, run the model with no climate shifts. First, choose a single parameter combination and run for varying lengths of time to both quantify a reasonable equilibrium state and record a time frame to use for the other parameter combinations.
* Characterize the distinctions in population dynamics at equilibrium for these different scenarios.

*Long-term Goals*

* Adjust the initialization of the model to be able to take in the equilibrium conditions of the different parameter combinations and then use that to explore the dynamics of climate change.
* First, use a single set of parameters to explore the effect of climate speed on population dynamics. Depending on results and the time it takes, choose a single speed, or do that for all parameter combinations.
* Repeat this process for a gradient in carrying capacity; Henry et al. 2013 suggest that gradients in K vs. R should result in different dynamics in the initial, stable range, but qualitatively similar behavior during the range shift.

**July 20, 2017**

I wrote up scripts and successfully tested several of the lower level functions today. Tomorrow, I will work on testing the more complex functions, hopefully being able to move on to full simulation testing over the weekend.

**July 21, 2017**

Idea for dealing with patch width: (1) mean dispersal distance is multiplied by PatchScale so dispersal occurs on the same scale as patches, and (2) carrying capacity is set to some baseline and multiplied by PatchScale (e.g. if the baseline is 1, a PatchScale of 50 would lead to a carrying capacity of 50 individuals per patch and 1 would correspond to a single individual occupying each patch (sort of)).

I adjusted the Reproduce function as stated above to account for PatchScale. As it turns out, I had already had a similar idea for the Disperse function so that should be good to go. I checked more functions today getting through the Disperse function so that only the reproduction functions are left along with the master function and example script before the more full tests.

**July 22, 2017**

One thing I noticed when testing out the functions is that the equation for stabilizing selection and local phenotypic optima have the potential to interact and cause problems in the model. Since the range of phenotypic optima only goes from 0 to 1, the strength of stabilizing selection (inverse of omega) must be set to a large enough value to compensate and still cause significant reductions in fitness for deviating phenotypes. At some values of omega, the distance from 0 to 1 in terms of phenotypic optima is only enough to case relative fitness to decline to 0.95 or higher.

**July 24, 2017**

Should we scale R and K parameters by local average fitness individually, or should we calculate (1 + R(1 – N/K)) \* mean fitness since that represents the realized increase in population size which could be interpreted as the more relevant scale for fitness to act upon (i.e. affecting the number of descendants being produced directly rather than affecting growth rate or carrying capacity…)? Or should it do both as occurs in the Peischl 2015 paper?

Also, save population size within the for loop for reproduction to avoid repetitious function calls. Possibly use some form of lapply prior to the loop? After all my bitching and moaning, it might be necessary to add a check for negative values before the call to rpois…

**July 27, 2017**

Today, I finished all the testing for the individual model functions and made the Rmarkdown write up of the results for the Model Testing folder. Next, I need to finish testing the model by using the full simulation to run the test cases described in the document.

**August 23, 2017**

The full model is working now (I still need to run a few test cases), but the biggest issue is the speed. I need to put it up on the super computer for the test cases and simultaneously, start working on fixes to improve the speed on my own computer.

**September 6, 2017**

I managed to make the model a little over twice as fast by implementing some changes to the structure of the code. It will especially make a big difference for the larger simulations with bigger population sizes. I also caught a small but significant error that caused exponential population growth in some cases. That also reduced the apparent slow down in the old code, but the new, faster code will definitely still pay off. I need to finish a few model tests, but the biggest next step is to choose parameter values and get some simulations running on the super computer to characterize the population distributions expected for different range types under a stationary climate.

Update: All the model tests are complete and passed. Next step is to set up some simulations on the super computer to get the dynamics of a stationary range for various different gradients of local selection and range edges. Next steps:

1. Make sure I am comfortable with parameters from the StationaryRange test
2. Write up a script to run 100 simulations for 1000 generations on the supercomputer
3. Use the local results from the test to estimate file sizes and make sure this is feasible
4. Use the local results to get a graphing script for population abundances, trait values, etc.

**November 1, 2017**

I’ve forgotten to update this document for a while, but here’s a summary of where things are at right now. First of all, I’ve successfully run the model for two range parameter combinations, generating conditions of those ranges under a stable climate. The next step is then to use those results to run the range shifting simulations and take a look at some of those preliminary results. Further, I’ve changed tactics to use a simple linear function to define local adaptation across a range. There’s no need to make this more complicated than it has to be and the simple idea is that conditions limiting a species’ range on one extreme are likely different to those limiting it on the other extreme (e.g. cold limited on one end and heat limited on the other). There’s really no good justification for making this more complicated. Even while the range capacity stays the same, conditions within the range are likely changing and therefore the local phenotypic optimum is still potentially changing. Finally, I’ve adjusted the simulation function to return the mean trait value and the genetic variance as that is what we actually care about. I’m waiting for MSI to finish maintenance so I can make some graphs and then I will work on getting the range shifting simulations running.

**November 5, 2017**

My next thinking deeply task for the model needs to be about space. Do there need to be discrete patches or am I making this unnecessarily complicated? The answer to this question will mostly be determined by how I would define the mating population without discrete patches to do things like calculate average fitness and assign parentage. Secondly, is the model currently treating the transition from Cartesian coordinates to discrete space the way I want it to? The main areas to check here are dispersal and how patch centers and x and y coordinates are defined for the GetEnvQuals function. To truly fix this, it might be necessary to adjust the coding to keep track of Cartesian coordinates more directly or to at least make sure the x and y coordinates are always representing the center of whatever patch the individuals are currently in (I don’t think this is the case currently with dispersal…).