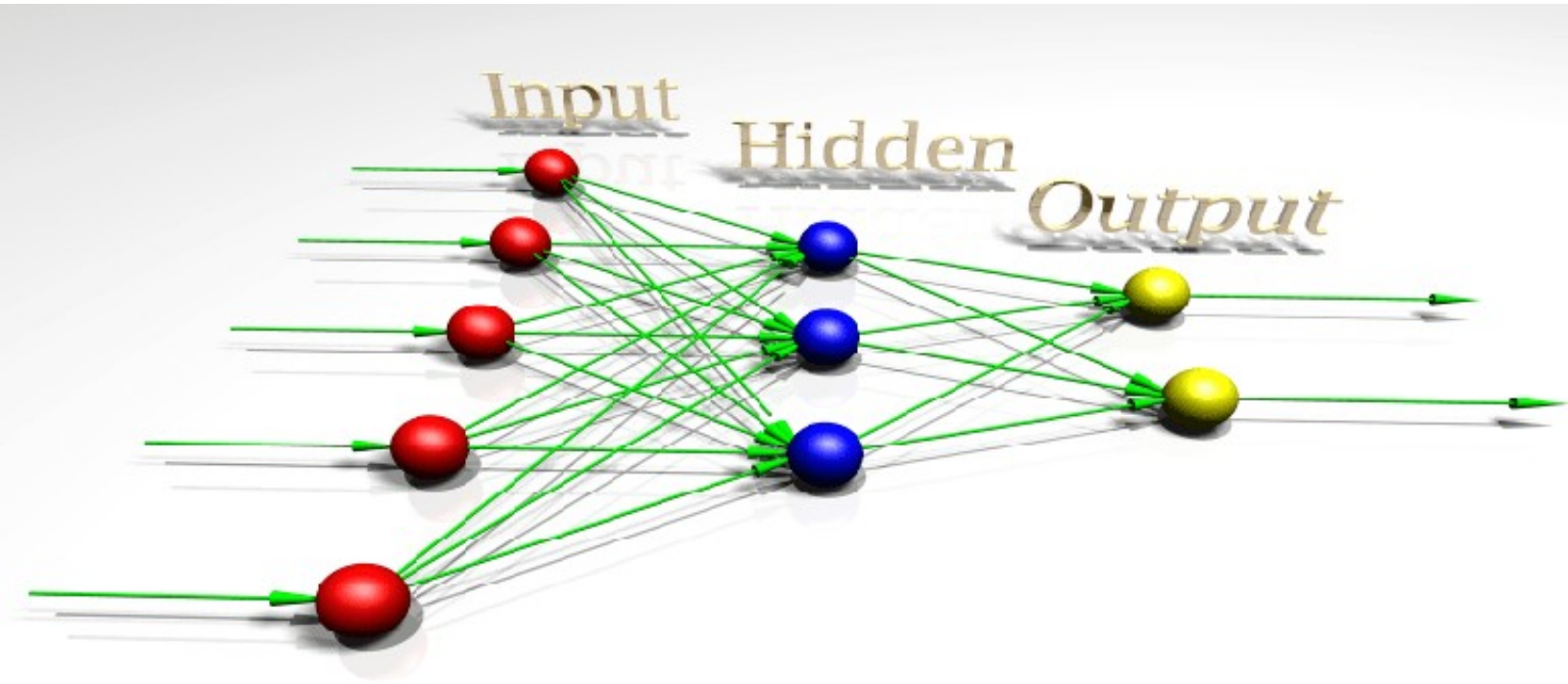

Input
Hidden
Output

Independent AI Research

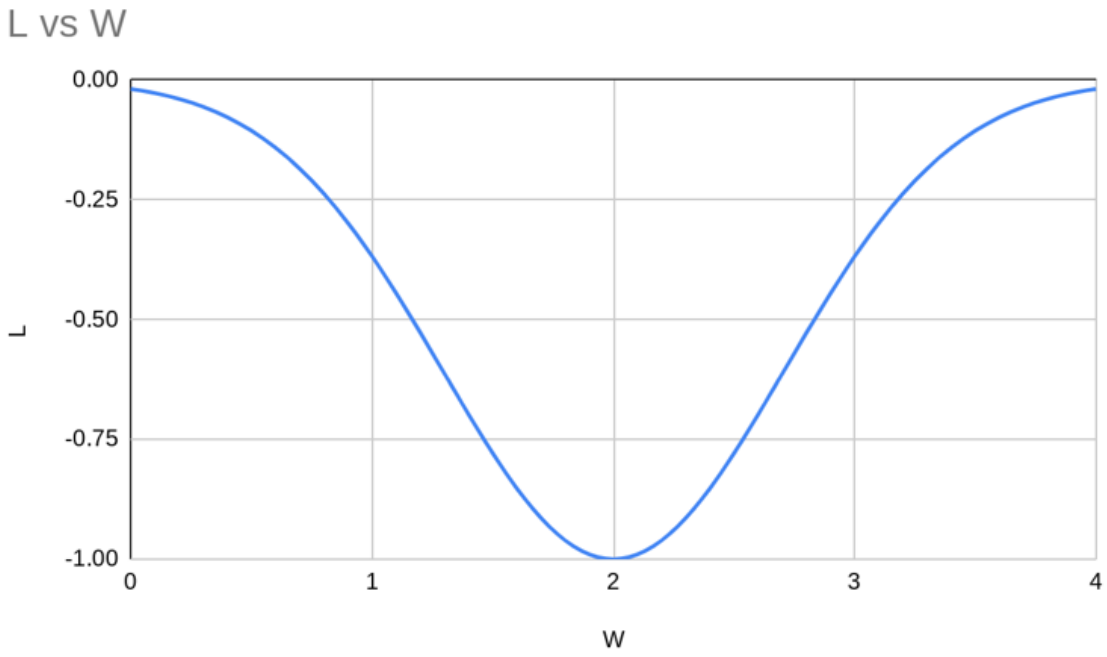
1. Modelling and Understanding SGD
2. SGD Understood through Probability Current
3. Hypotheses about Finding Knowledge and One-Shot Causal Entanglements
4. Knowledge Localization: Tentatively Positive Results on OCR
5. Defining Optimization in a Deeper Way Part 1
6. Defining Optimization in a Deeper Way Part 2
7. Defining Optimization in a Deeper Way Part 3
8. Defining Optimization in a Deeper Way Part 4

Modelling and Understanding SGD

I began this as a way to get a better understanding of the feeling of SGD in generalized models. This doesn't go into detail as to what a loss function actually is, and doesn't even mention neural networks. The loss functions are likely to be totally unrealistic, and these methods may be well out-of-date. Nonetheless I thought this was interesting and worth sharing.

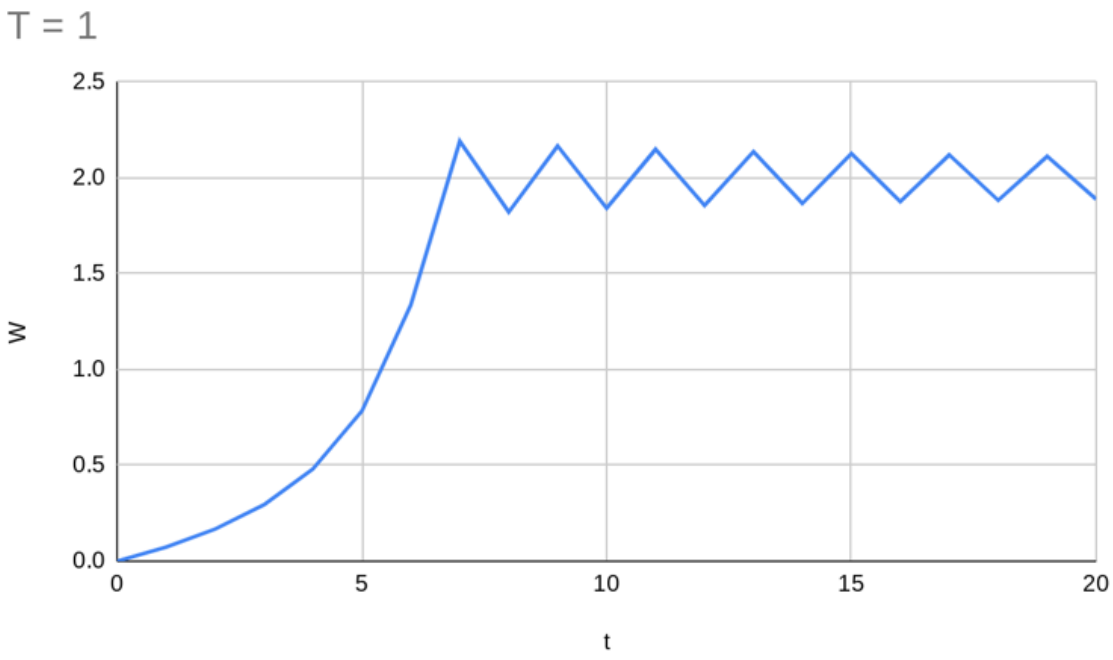
Imagine we have a one-parameter model, fitting to one datapoint. The parameter starts at $W = 0$ and the loss is $L = -\exp(-(W-2)^2)$. The gradient will then be

$\frac{dL}{dW} = 2(W-2)\exp(-(W-2)^2)$.

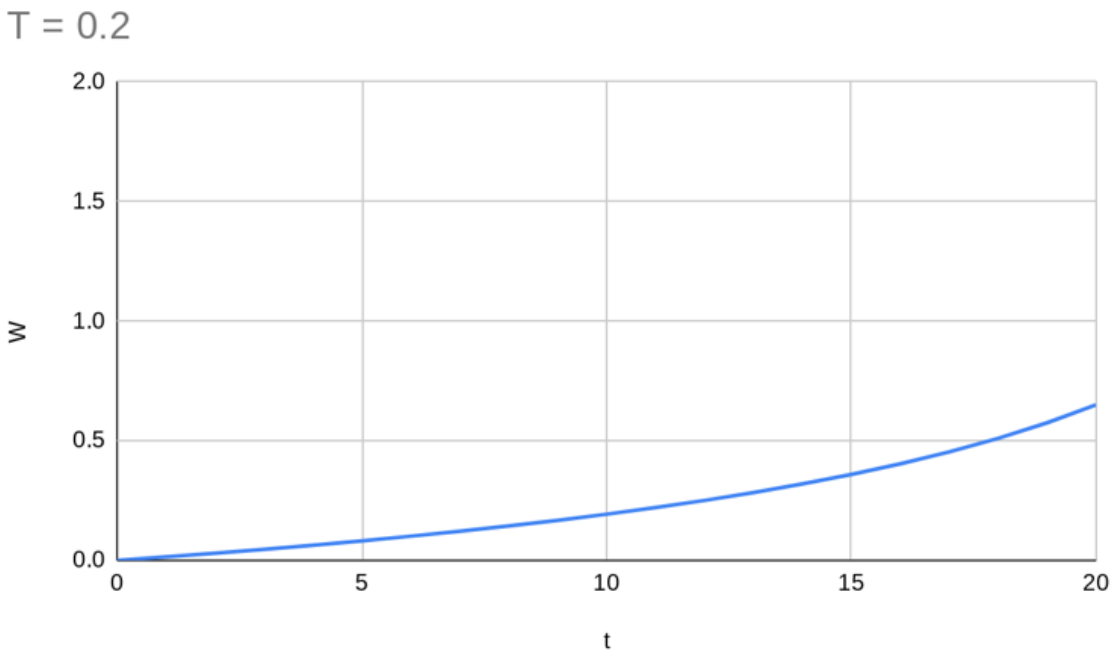


An imaginary continuous gradient descent will smoothly move to the bottom of the well and end up with $W = 2$.

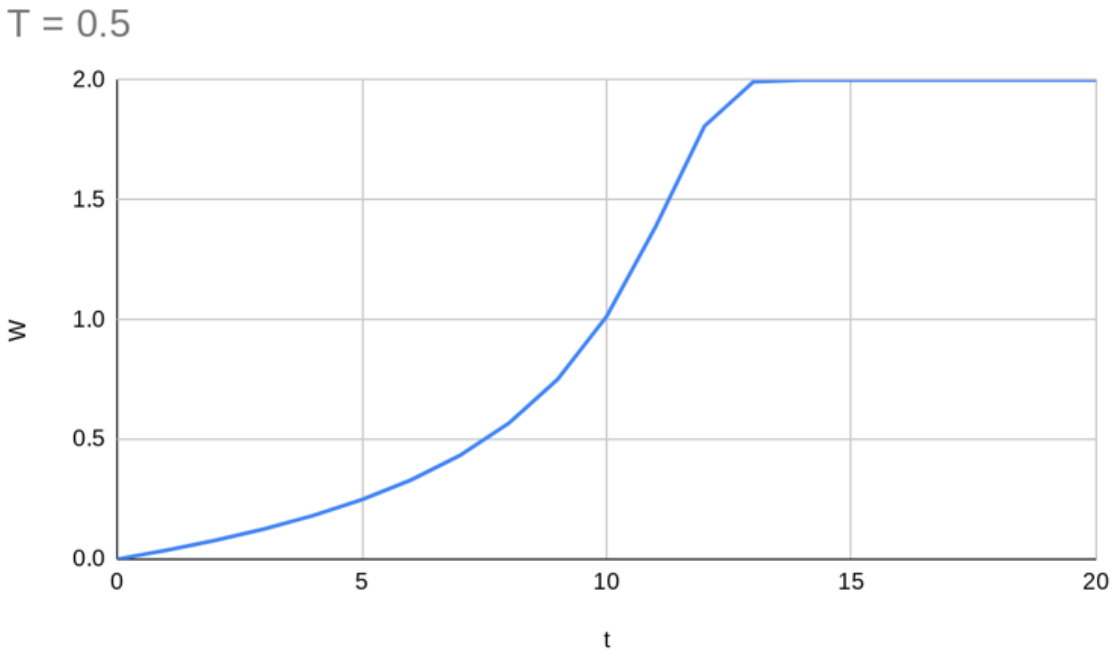
A stepwise gradient descent needs a hyperparameter T telling it how much to move the parameters each step. Let's start with this at 1.



This gives us a zig-zag motion. The system is overshooting. Let's pick a smaller T value.



Hmm. Now our model converges quite slowly. What about 0.5?

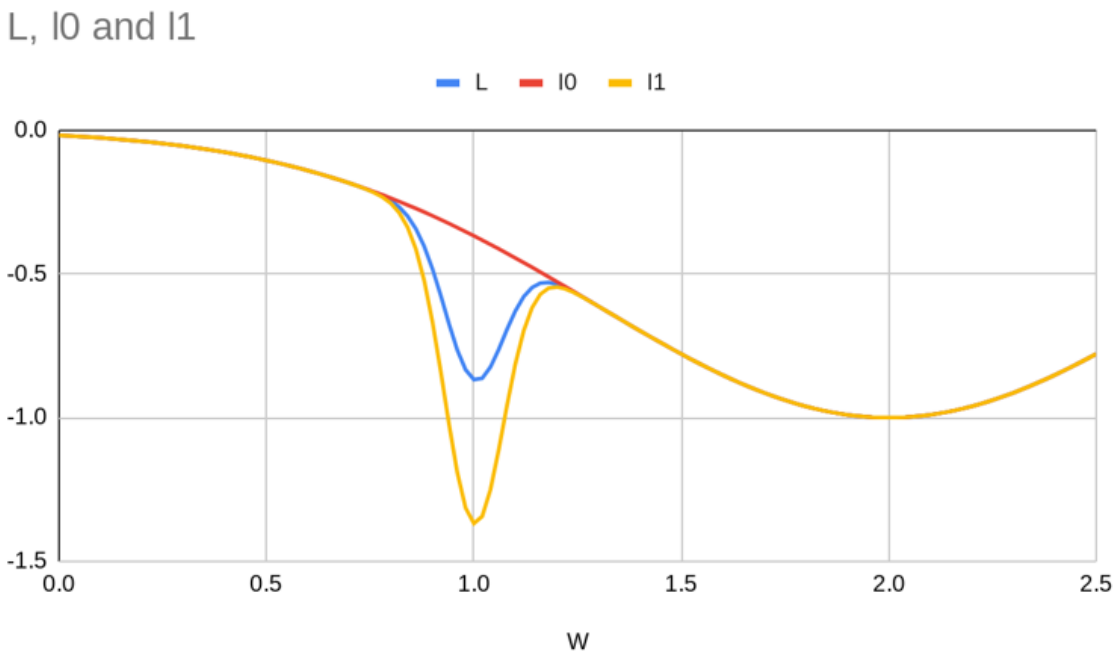


Much better. Seems that there's an optimal value of T, which we will later see depends on the spikiness of the loss function.

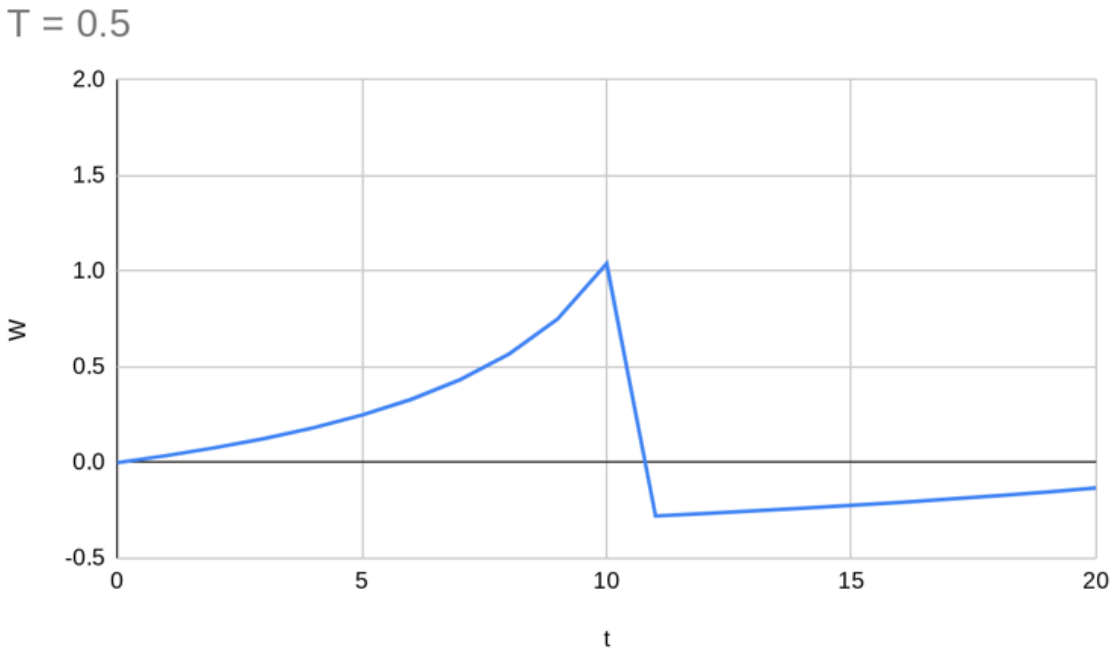
T is not dimensionless and has the dimension of W^2/L, as $\Delta W = -T\frac{dl}{dW}$. Certain function-minimising methods like Newton's method use (for example) $\Delta W = -k\frac{dl}{dW}/\frac{d^2l}{dW^2}$ where k is dimensionless. This is why different loss functions require different T values.

SGD and Local Minima

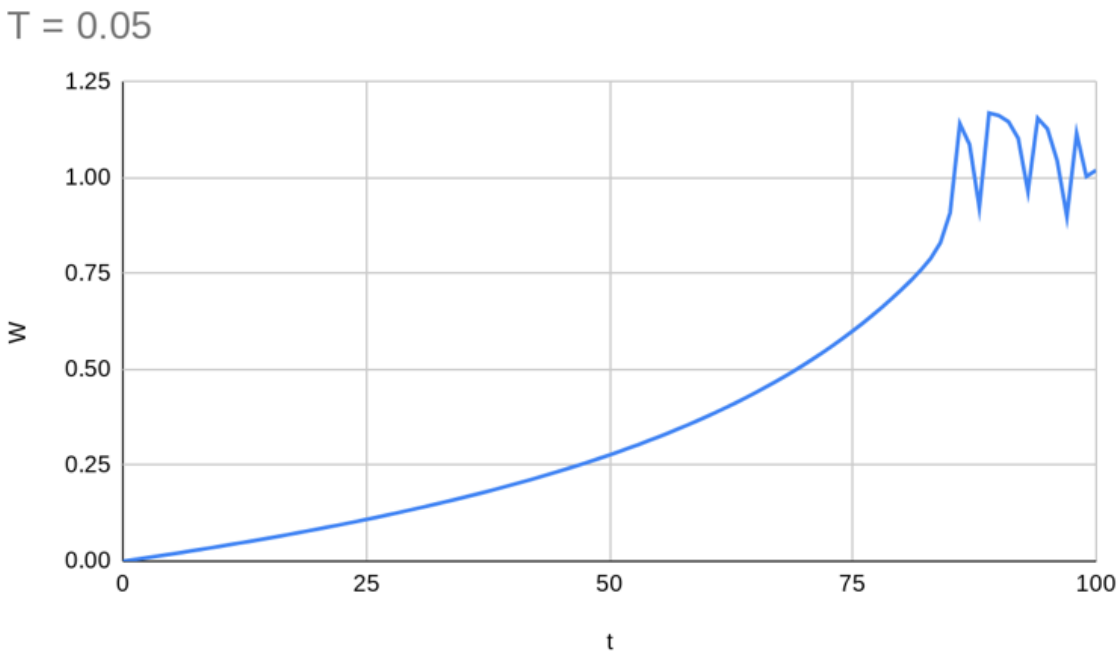
What if we have two datapoints? Now $L = (l_0 + l_1)/2$. Let $l_0 = -\exp(-(W-2)^2)$ as above but $l_1 = -\exp(-(W-2)^2) - \exp(-100(W-1)^2)$.



Now our loss function L has a new local minimum I think of this as "the pit". Where we end up depends on where we start. If we start at W = 4 then we'll clearly end up at W = 2 but if we start at W = 0, then:



Huh, this isn't what I expected at all! The gradient must have been too high around the local minimum. Let's try with T = 0.05, which will be slower but ought to work better.



This is more like what I expected. Now our system is stuck in the local minimum.

But most modern gradient descent algorithms use stochastic gradient descent. We can model this here by randomly picking one of the two datapoints (with associated loss function) to descend by each step.

What happens now? Well now we have a chance to escape the pit. Let's say we're at the minimum of the pit. How many *consecutive* descent steps performed on l_0 will get us out of the pit?

Well by solving for the stationary points of l_1, we get $W = 1.004$ and $W = 1.196$. It turns out 5 steps of descent on l_0 will get us to $W = 1.201$ which is out of the pit. Within 100 steps we have an 81% chance of getting out.

The interesting thing is that this doesn't depend much on the depth of the $W = 1$ pit. If the local pit is twice as deep in l_1, then we only require one more consecutive step to escape it. If this is the case, then $W = 1$ is in fact a *global minimum* in L. But because of SGD, it's not stable, and the only stable point is $p_0 = 2$! Clearly there is something other than overall minimum which affects how the parameter of the model changes over time.

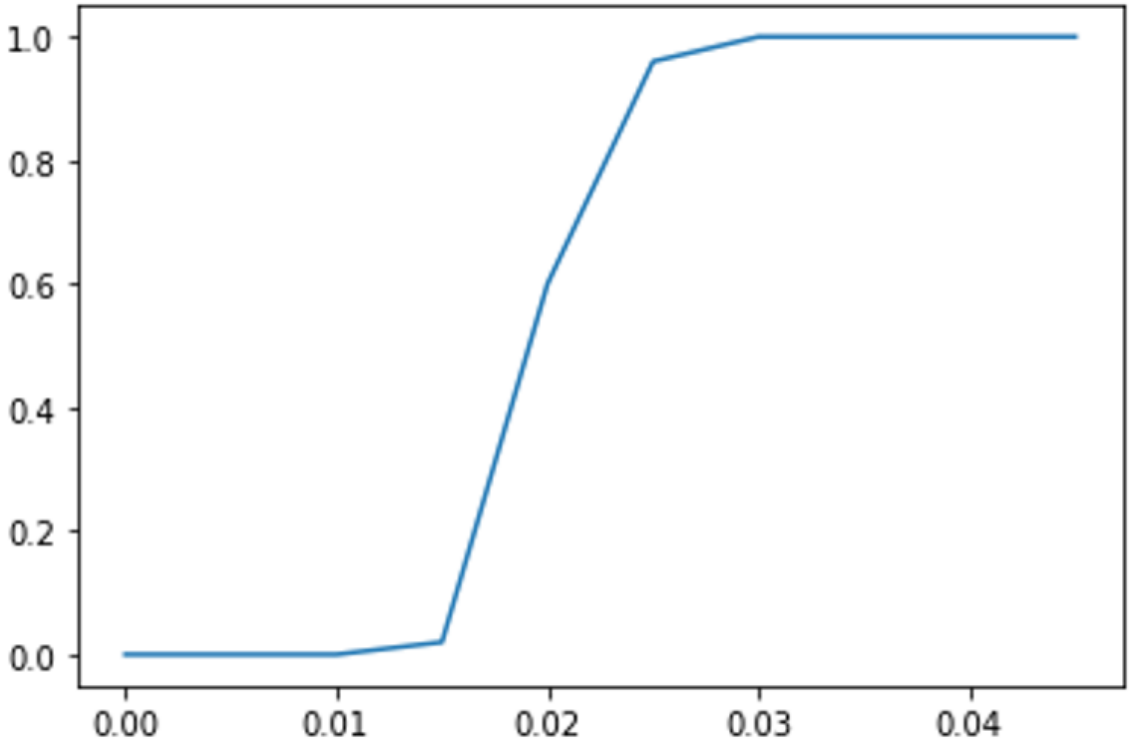
What about smaller values of T? The number of consecutive steps in l_0 needed to escape the pit is inversely proportional to T. In the infinite limit as $T \to \infty$, we just converge on a continuous gradient descent, which will find the first minimum it comes to.

This reminds me of chemistry. It's as if the size of the step has to be big enough for the model to overcome some activation energy to cross the barrier from the $p_0 = 1$ pit to the $p_0 = 2$ one. This is the motivation for the letter T: temperature. The larger T is, the more likely it is that the model will cross over out of the local minimum.

Monte Carlo

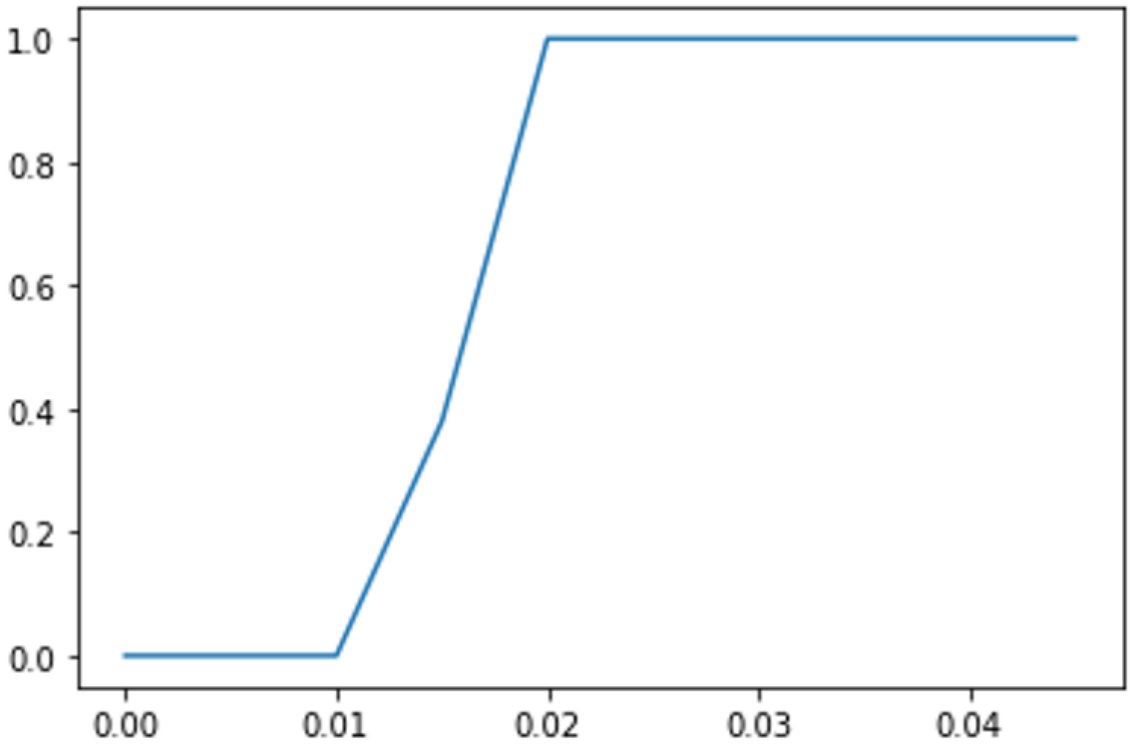
In fact we generally need significantly fewer than this. Starting around 1, one update on l_0 is enough to push us into the high-gradient region of l_1, which means even an update on l_1 will not move us back down into the centre, but rather to the left side of the pit, where a subsequent l_1 update might push us further towards the right.

Let's estimate the probability of escaping the pit in 100 steps as a function of T:



Probability of escaping the pit in 100 steps as a function of T

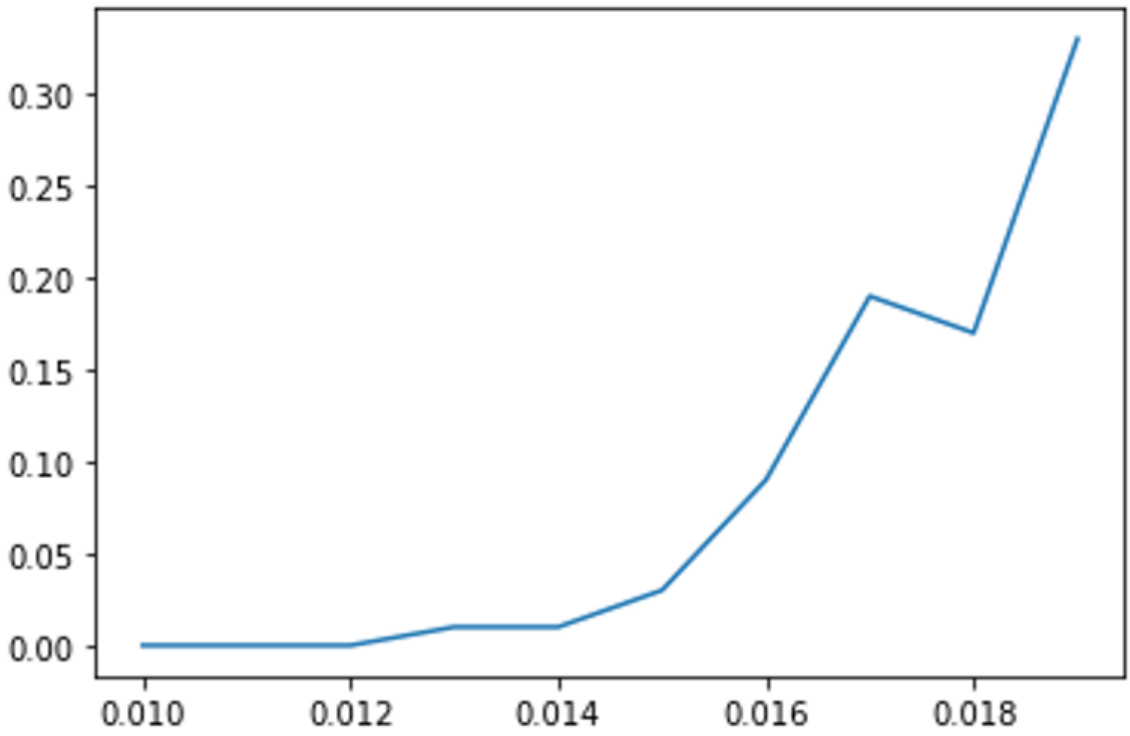
What about 1000?



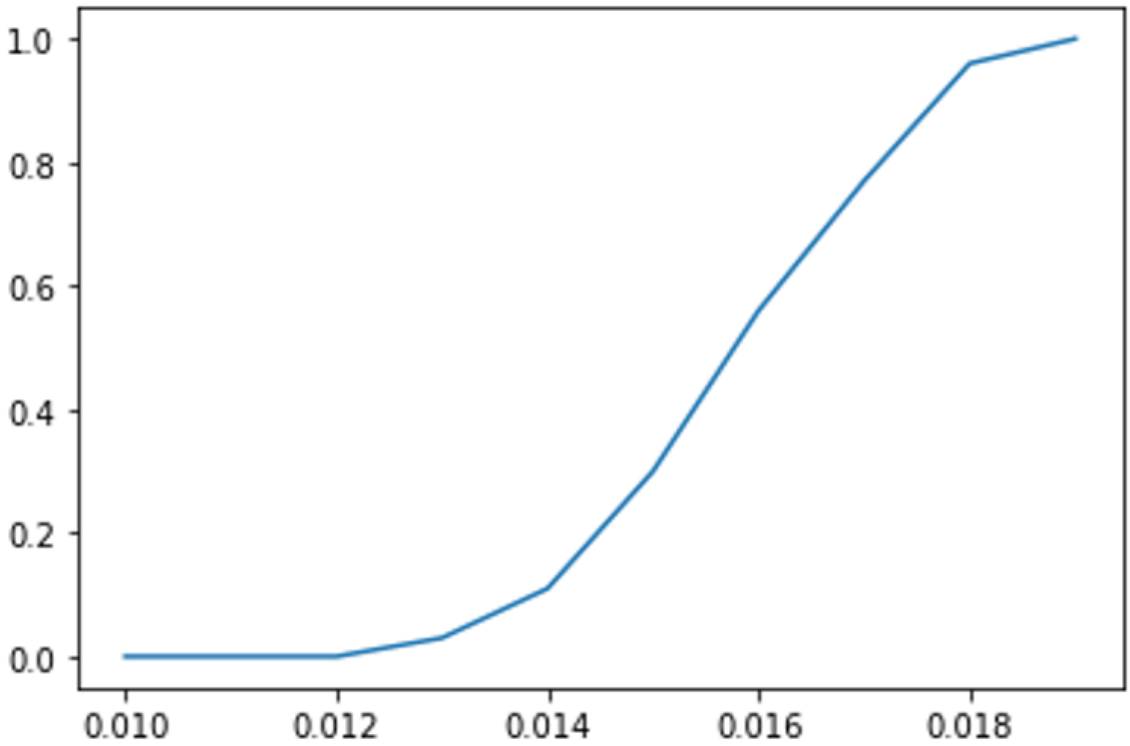
Probability of escaping the pit in 1000 steps as a function of T

As we sort-of expected, it is easier to escape the pit than our original model predicted.

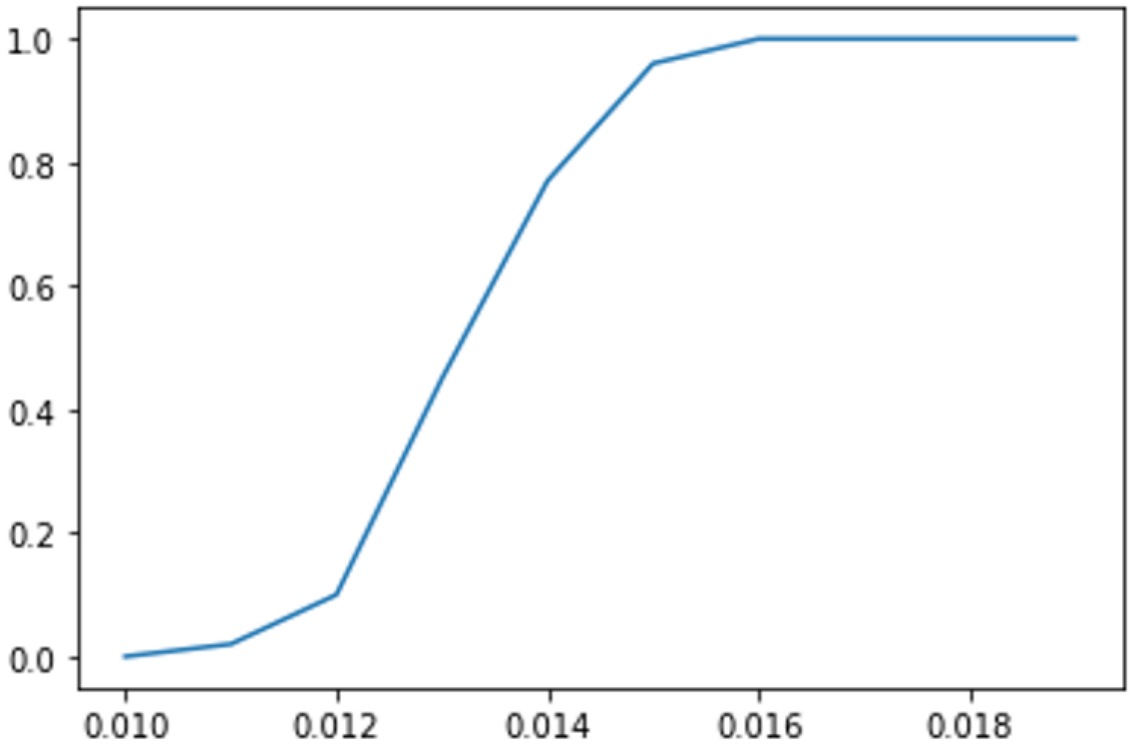
Let's look more closely at the region between 0.01 and 0.02:



Probability of escaping the pit in 100 steps as a function of T (note Y axis)



Probability of escaping the pit in 1000 steps as a function of T



Probability of escaping the pit in 10000 steps as a function of T

It looks roughly like the log of the number of steps required to escape the pit is a function of T. Hopefully the later posts in this series will allow me to understand why.

SGD Understood through Probability Current

My previous post about SGD was an intro to this model. That post concerned a model of a loss landscape on two "datapoints". In this post I attempt to build a new model of SGD and validate it, with mixed success, but it is sort of interesting.

Gradient Variance

We could model this another way. The expected change of W on each step is $-T\frac{dW}{dL}$, but we will also expect variance. W will evolve over time through probability space. There are two competing "forces" here, the "spreading force" created by variance in in $\frac{dW}{dl_j}$ over all datapoints in the model, and the "descent force" being exerted by gradient descent pushing W back into the centre of a given local minimum.

I think it makes sense to introduce some new notation here.

$g_j(W) = \frac{dl_j}{dW}$

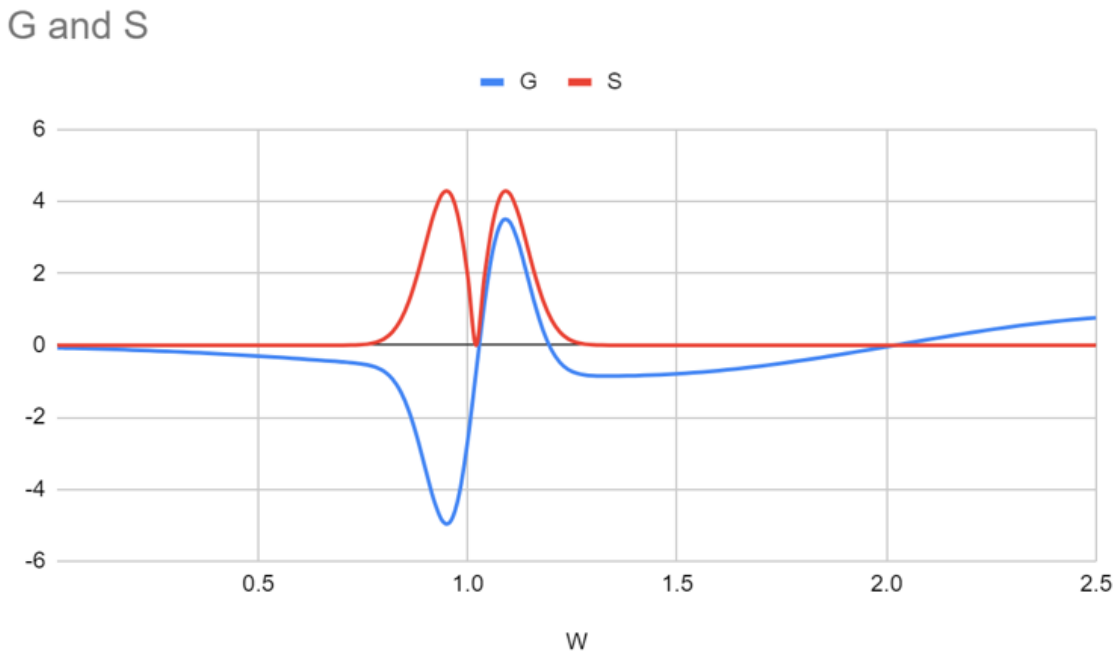
$G(W) = \frac{dL}{dW} = average_{(all\ j)}(g_j(W))$

$S^2(W) = variance_{(all\ j)}(g_j(W))$

$S(W) = +\sqrt{S^2(W)}$

The S^2 notation should be thought of like the $\cos^2(x)$ notation.

Plotting these for our current system:



Places where G is zero *and* the gradient of G is positive are the stable equilibrium points with regards to gradient descent on L (at ~1 and 2). If G and S^2 are both zero at the same place, then this is an equilibrium point with regards to SGD on L (only at 2). The zero points for G and S^2 around the pit at 1 are not quite in the same pace.

It is possible to consider probability mass of W "moving" according to the following rule:

A "point" (dirac δ distribution) of probability at W, between t and $t+1$, changes to a distribution centred at $W - TG(W)$ with a variance of $T^2S^2(W)$.

Now we have abstracted away T from the actual process of discontinuous updates, we can try and factor out the discontinuity entirely. This will make the maths more manageable when it comes to generalizing to larger models. T will likely be much smaller for larger models but as long as S(W) grows larger with the number of datapoints used, this will compensate.

(Point of notation, I will be using d rather than ∂, even though the latter is arguably more correct. As we will never be "mixing" Wand t it won't make a difference to our results)

Instead of probability distribution moving, we might now consider it flowing. This can be described by a probability current density ρ:

Consider a system with $S^2(W) = 0$ everywhere. The probability will just flow down the gradient:

$$\rho(W, t)_G = -T\, G(W)\, P(W, t)$$

Taking $\frac{dP}{dt} = -\frac{dQ}{dW}$ we get (when dependencies are removed for ease of reading):

$$\frac{dP}{dt}_G = T\left[G\frac{dP}{dW} + P\frac{dG}{dW}\right]$$

Now consider a system with $G(W) = 0$ everywhere. Now we effectively have the evolution of a probability distribution via random walk. This gives a "spreading out" effect. With constant S^2 we have the following equation for ρ, borrowed from the heat equation. I will take the central limit theorem and assume that the gradients are normally distributed.

$$\rho(W, t)_S = -\frac{1}{2} T^2 S^2 \frac{dP(W, t)}{dW}$$

Based on the fundamental solution of the heat equation this will increase our variance by T^2S^2 each step of t.

Which gives us:

$$\frac{dP}{dW}_S = \frac{1}{2} T^2 S^2 \frac{d^2P}{dW^2} + \frac{1}{2} T^2 \frac{d(S^2)}{dW^2}\frac{dP}{dW}$$

But the speed of "spreading out" is proportional to $S(W)$ which changes the equation. The slower the "spreading out", the higher the probability of W being there. This makes $S(W)$ act like a "heat capacity" of the location for $P(W, t)$ for which ρ is a conserved current. We might be able to borrow more from heat equations. In this case $P(W, t)S(W)$ acts as the "temperature" of a region.

$$\rho(W, t)_S = -k(W)\frac{d[P(W, t)S(W)]}{dW}$$

$$\rho(W, t)_S = -k(W)\left[S(W)\frac{dP(W, t)}{dW} + P(W, t)\frac{dS(W)}{dW}\right]$$

Calculating k based on our previous equation gives $k(W) = \frac{1}{2}T^2S(W)$, which gives:

$$\frac{dQ}{dW}_S = -\frac{d}{dW}\left[-\frac{1}{2} T^2 S^2(W)\frac{dP(W, t)}{dW} - \frac{1}{2} T^2 S(W) P(W, t)\frac{dS(W)}{dW}\right]$$

This can be reduced to the rather unwieldy equation (removing function dependencies for clarity):

$$\left(\frac{dP}{dt}\right)_S = T^2 \left[\frac{1}{2} S \frac{dG}{dW} \frac{dP}{dW} + \frac{1}{2} S^2 \frac{d^2P}{dW^2} + \frac{1}{2} \left(\frac{dG}{dW}\right)^2 P + \frac{1}{2} S P \frac{d^2S}{dW^2} \right]$$

But these can be expressed in terms of S^2 rather than S, which is good when S is pathological in some way (like when S^2 is zero above, S has a discontinuous derivative). It also makes sense that our equation shouldn't depend on our choosing positive rather than negative S.

$$\rho_S = -T^2 \left[\frac{1}{2} S^2 \frac{dP}{dW} + \frac{1}{4} P \frac{d(S^2)}{dW} \right]$$

$$\left(\frac{dP}{dt}\right)_S = T^2 \left[\frac{1}{4} \frac{d(S^2)}{dW} \frac{dP}{dW} + \frac{1}{2} S^2 \frac{dP^2}{dW} + \frac{1}{4} \frac{d^2(S^2P)}{dW^2} \right]$$

Finally giving our master equations:

$$\rho = -T^2 \left[\frac{1}{2} S^2 \frac{dP}{dW} + \frac{1}{4} \frac{d(S^2)}{dW} P \right] - T \left[G P \right]$$

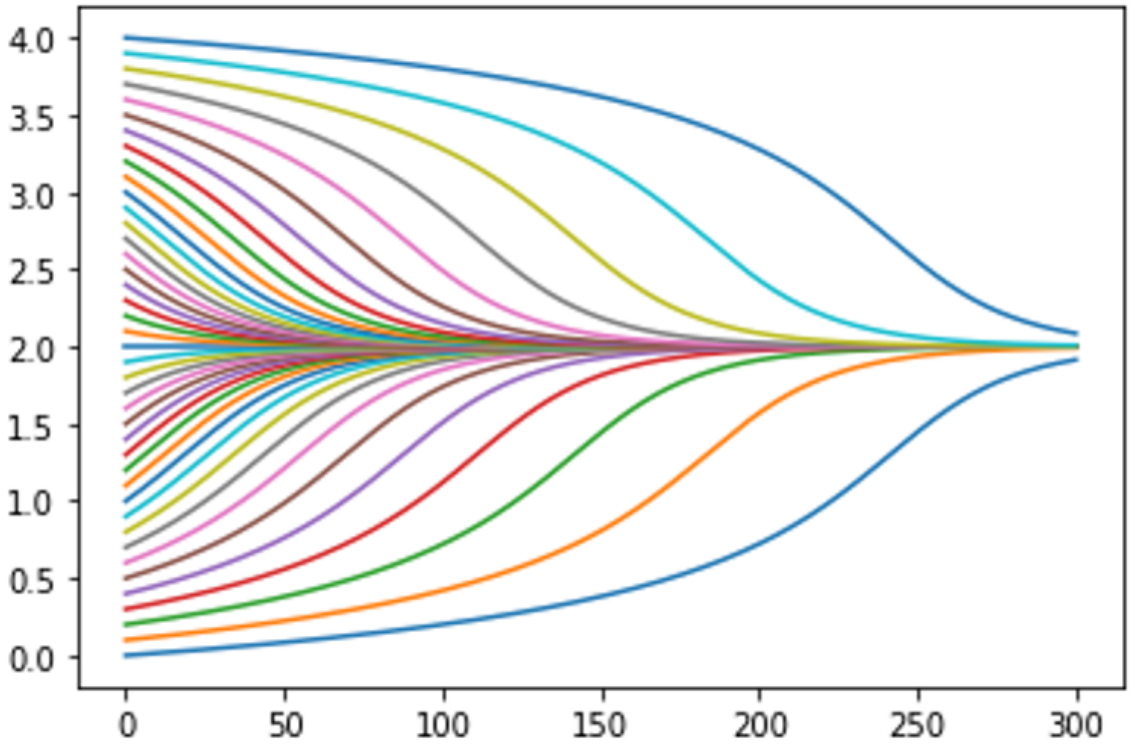
$$\frac{dP}{dt} = T^2 \left[\frac{1}{4} \frac{d(S^2)}{dW} \frac{dP}{dW} + \frac{1}{2} S^2 \frac{dP^2}{dW} + \frac{1}{4} \frac{d^2(S^2P)}{dW^2} \right] + T \left[G \frac{dP}{dW} + P \frac{dG}{dW} \right]$$

Validation of the First Term of the Equations

Let's start with the first equation, and simulate using our G function from before.

T = 0.02, no stochasticity yet.

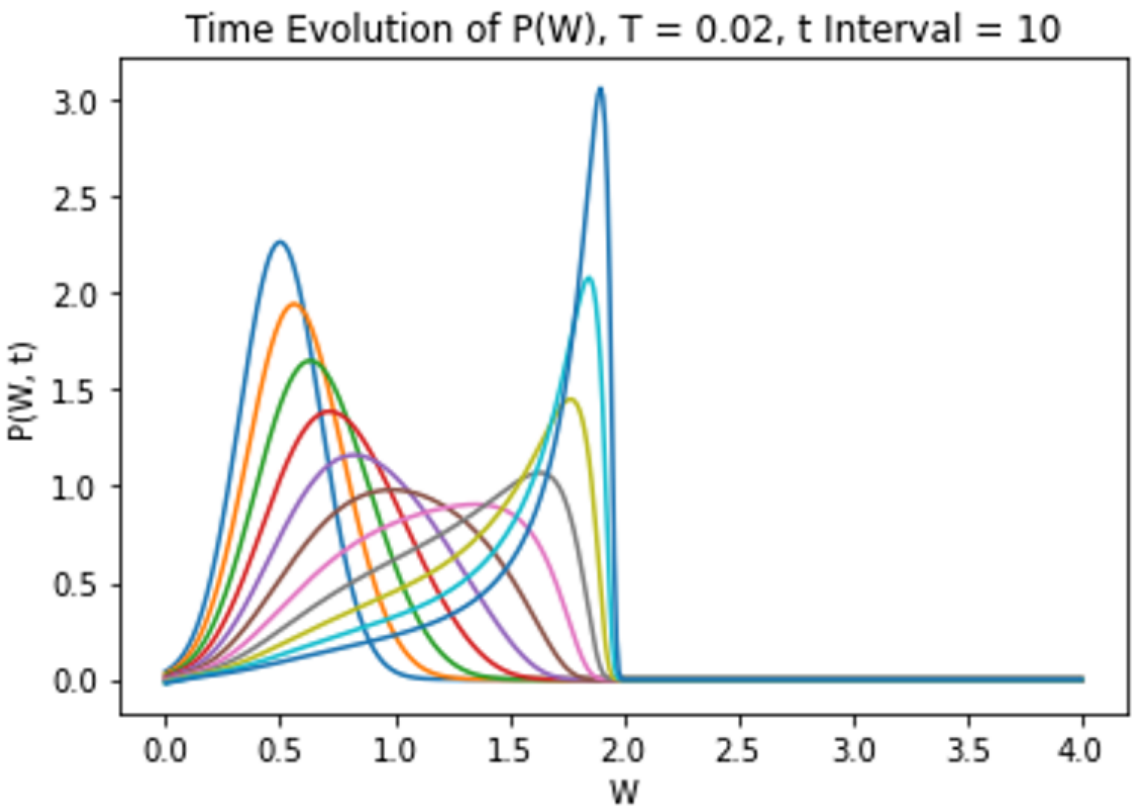
Here's W on the y-axis, and t on the x-axis. This is what the evolution of W looks like for a series of initial W values:



Evolution of W values against t

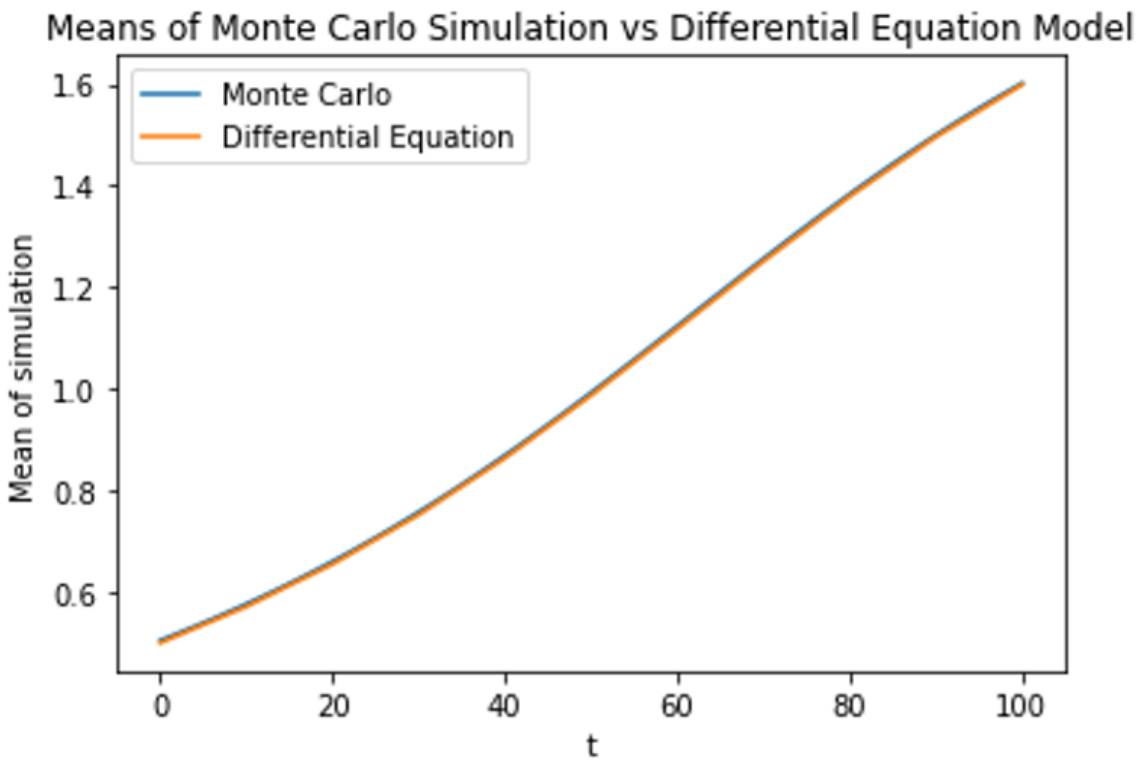
Now let's pick a couple of initial distributions and see how they evolve over time:

Time evolution with steps of $\Delta t = 10$:

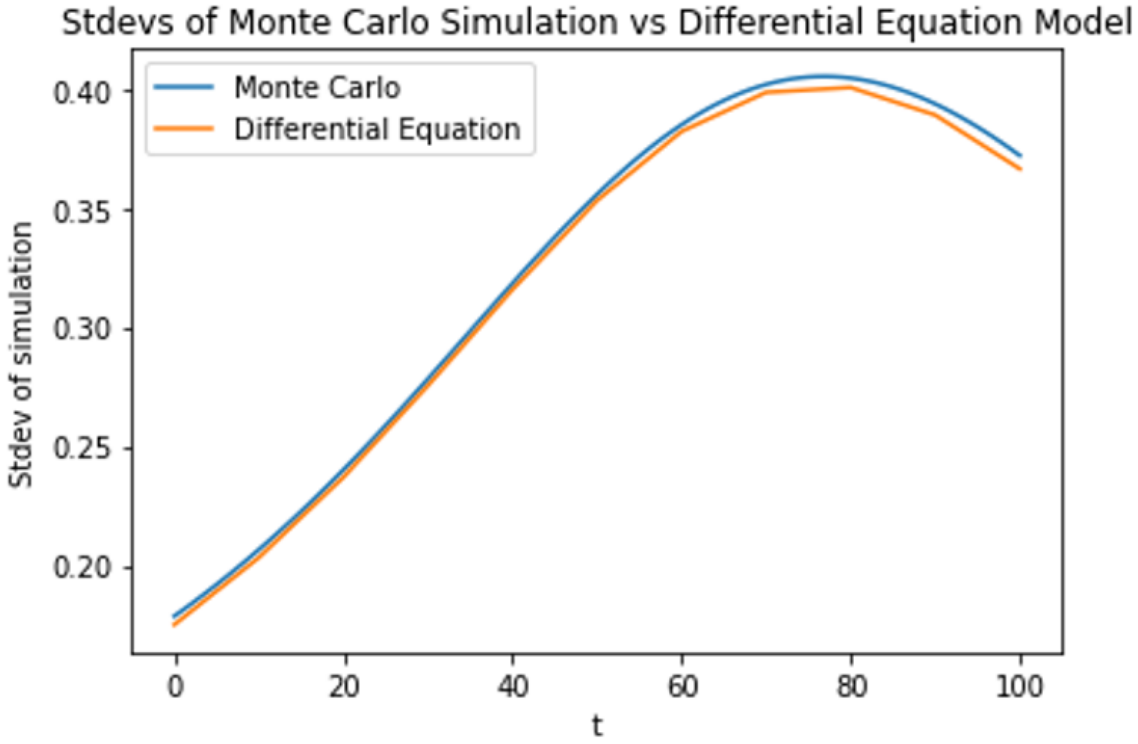


This looks about right!

Now let's plot the mean of this over time, and compare to the mean and standard deviation of a Monte Carlo simulation of gradient descent. The Monte Carlo simulation starts with 1000 W values chosen to form a normal distribution with the roughly same mean and standard deviation (0.5 and 0.175 respectively) as our initial $P(W, t)$ distribution.



Yes there are two lines there.



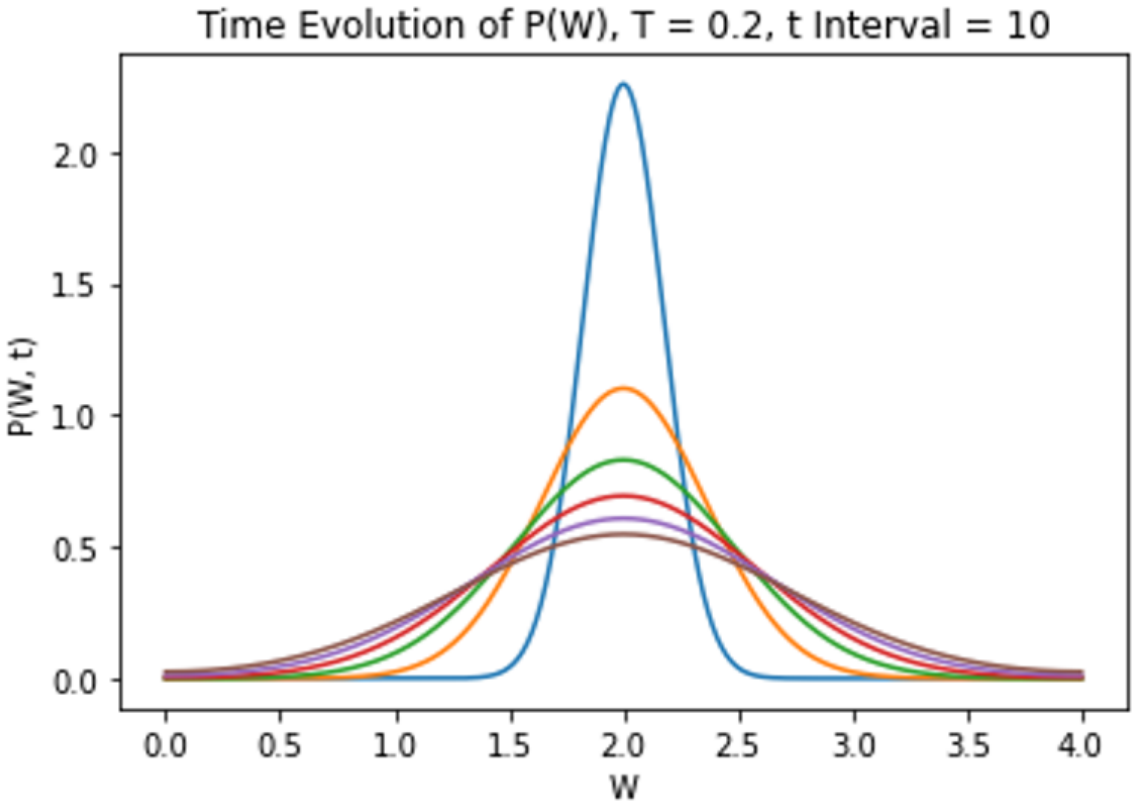
Our first equation is an accurate description of non-stochastic gradient descent. The rest of the difference in the standard deviation is most likely due to imperfect matching of our initial

data (P is a truncated normal distribution but our Monte Carlo uses a normal distribution with matched mean and variance to the truncated P, so some elements are < 0 where the gradient is small).

Validation of the Second Term of the Equations

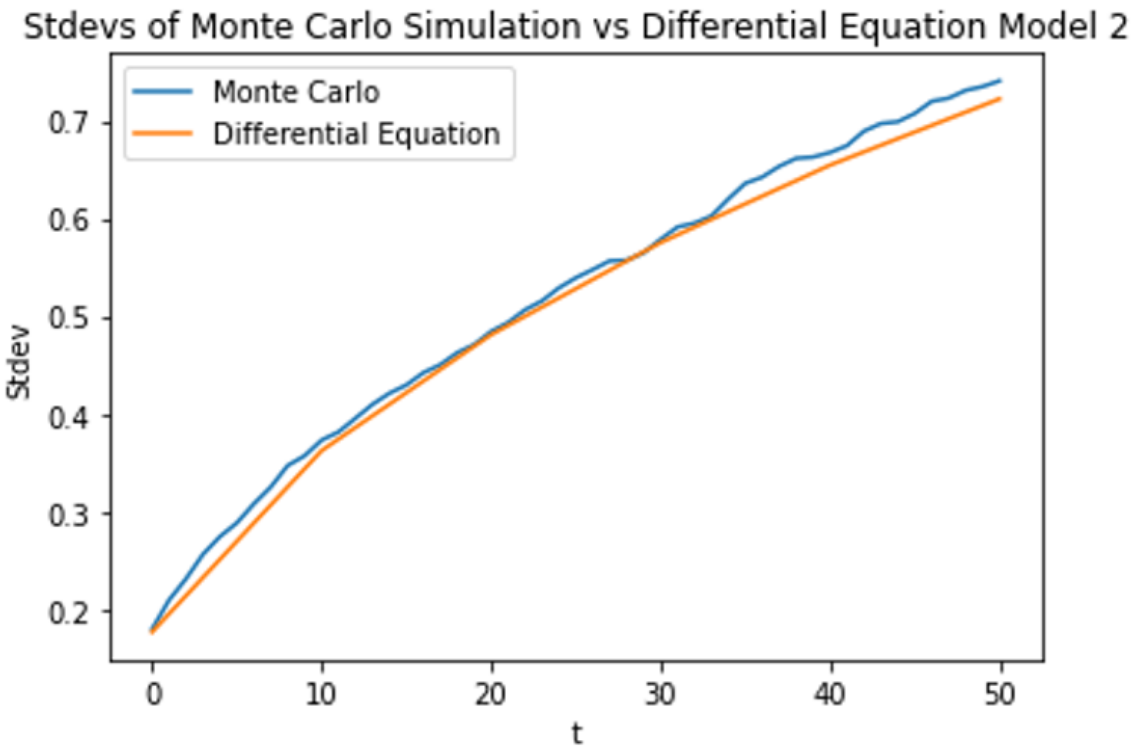
Let's take our first example as a distribution spreading out.

$g_0 = -1$, $g_1 = 1$, $G = 0$, $S^2 = 1$



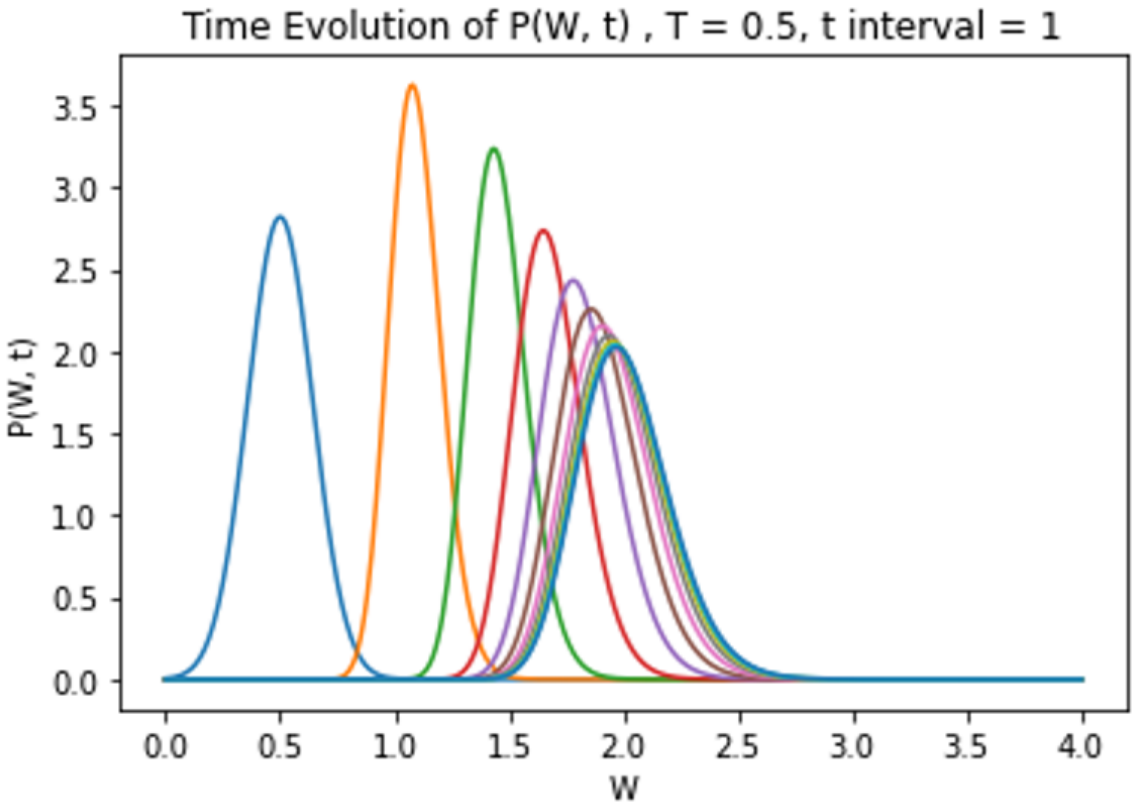
The probability distribution changes from a concentrated one to a broadened one.

And compare standard deviations to our Monte Carlo simulation:

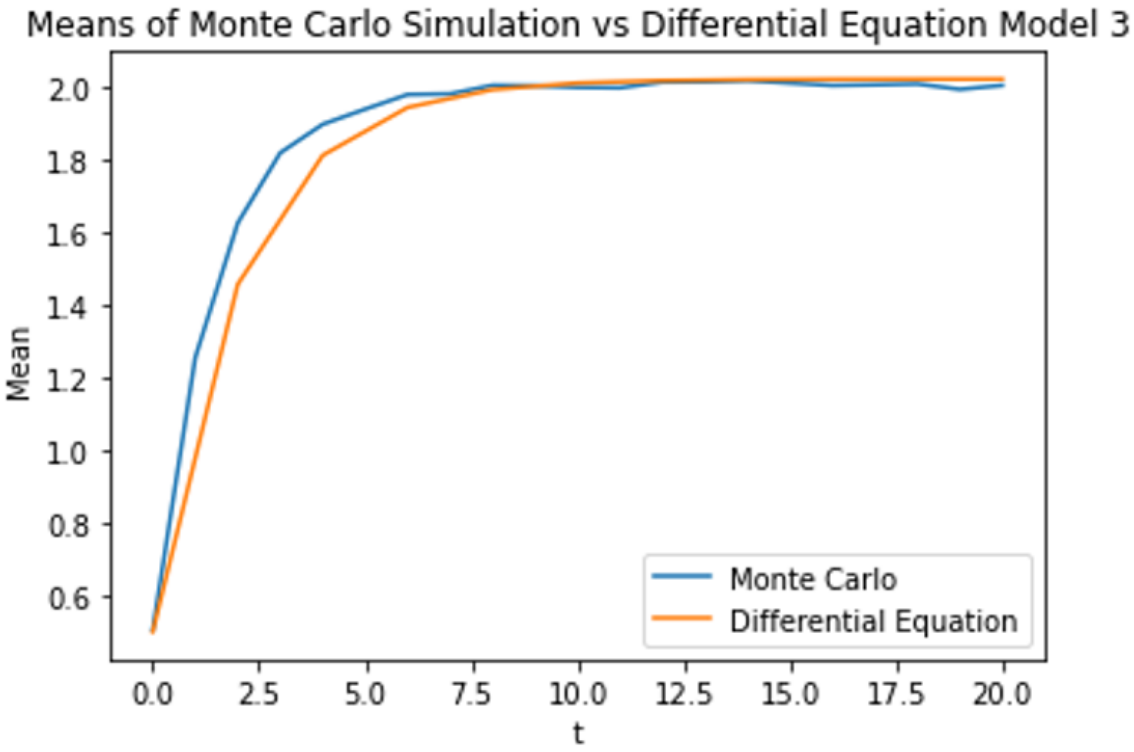


Looking good, errors here may also be due to truncation.

One final validation step: take $g_0 = 1.2W - 2$, $g_1 = 0.8W - 2$, $G = W - 2$, $S^2 = 0.4W^2$, $T = 0.5$. This model will be used to assess a few things: our ability to perform well at higher T, its ability to predict the correct form of the counterbalancing "concentrating" and "spreading" forces of G and S^2, and its ability to predict the concentration of probability mass in regions of lower S^2.



Here the probability distribution moves from the left to the centre but doesn't concentrate due to the gradient variance.



Unfortunately the computational modelling seems to fall apart when applied to the original system. The large first and second derivatives of S^2 lead to a lot of instability. This means I can't validate it much more than this. High values of T also cause the model to break down, as the gradient might change a lot in the span of a step. I think this can be remedied by (for example) picking a g to update on and updating with multiple small steps before changing g.

I'm no master programmer and I don't have much experience working with unstable PDEs. So I can't do much more here.

Solving for End-States

For an end-state, ρ = 0 everywhere. This means:

$$0 = -T^2 \left[\frac{1}{2} S^2 \frac{d\rho}{dW} + \frac{1}{4} \frac{d(S^2)}{dW} P \right] - T [G P]$$

$$T \frac{1}{2} S^2 \frac{d\rho}{dW} = -T \frac{1}{4} \frac{d(S^2)}{dW} P - G P$$

$$\frac{d\rho}{dW} \frac{1}{\rho} = -\frac{1}{2S^2} \frac{d(S^2)}{dW} - \frac{2G}{TS^2}$$

$$\frac{d(\log(\rho))}{dW} - \frac{1}{2S^2} \frac{d(S^2)}{dW} - \frac{2G}{TS^2}$$

$$\frac{d(\log(P))}{dW} - \frac{1}{2} \frac{d(\log(S^2))}{dW} \frac{2G}{TS^2}$$

This shows our problem. When S^2 vanishes, our equations don't work terribly well. We might have to hope that the two opposing S^2 terms cancel out and it works, but who knows. This is probably the source of instability in our equations.

But around some minimum it lets us interpret something. If log(P) is decreasing linearly then P decreases exponentially. Let's consider the $\frac{2G}{TS^2}$ term now. If we have two minima (with a maximum between them) around which the loss landscapes are exactly the same, except one is twice as wide (in all l_i) then the G component will be halved in the wider one, but the S^2 part will be quartered. This means the integral of $\int -\frac{2G}{TS^2} dW$ from the centre of the wider one to the maximum will be four times that of the narrower one. Therefore the probability density at the centre of the wider minimum's basin will be ~~e^4 = 56 times~~ *Edit: a lot* higher.

What's the point?

Reasoning about stochastic processes is difficult. Reasoning about differential equations is also difficult, but the tools to analyse differential equations are different and might be able to solve different problems.

SGD is believed to have certain "bias" towards low-entropy models of the world. Part of this is a preference for "broader" rather than "narrower" minima in L. Now we have some tools which may allow us to understand this. Under this model, SGD is also biased towards regions of low variance in loss function.

Further Investigation

I think there's something like a metric acting on a space here. S^2 looks like a metric, and perhaps it's actually more correct to consider the space of W with the metric such that $S^2 = 1$ everywhere. For higher dimensions we get the following transformations:

$$
\begin{array}{ccc}
 & \longrightarrow & \\
W & \rightarrow & W \\
 & \rightarrow & \\
G & \rightarrow & G \\
S^2 & \rightarrow & S^2 \\
 & \rightarrow \quad \rightarrow &
\end{array}
$$

Now W and G are vectors and S^2 is a matrix. This extends nicely as we can choose our metric such that $S^2 = I$. It might be useful to define some sort of function like an "energy" over the landscape of \(\\)W in terms of G, S, and T alone which describes the final probability distribution. In fact such a function must exist assuming SGD converges, as $\log(P(W, \infty))$ is well-defined. What the actual form of this function is would require to do some working out, and it may not be at all easily described. This whole process is very reminiscent of both chemical dynamical modelling and finding the minimum-energy configuration of a quantum energy landscape, as both consist of a "spreading" term and an "energy" term.

While it is quite interesting, I don't consider this a research priority for myself. About 90% of this post has been sitting in my drafts for the past 3 months. Even if powerful AI is created using SGD, I'm not convinced that this sort of model will be hugely useful. It might be possible to wrangle some selection-theorem-ish-thing out of this but I don't think I'll focus on it.

Hypotheses about Finding Knowledge and One-Shot Causal Entanglements

Crossposted from the AI Alignment Forum. May contain more technical jargon than usual.

Epistemic status: my own thoughts I've thought up in my own time. They may be quite or very wrong! I am likely not the first person to come to these ideas. All of my main points here are just hypotheses which I've come to by the reasoning stated below. Most of it is informal mathematical arguments about likely phenomena and none is rigorous proof. I might investigate them if I had the time/money/programming skills. Lots of my hypotheses are really long and difficult-to-parse sentences.

> What is knowledge?

I think this question is bad.

It's too great of a challenge. It asks us (implicitly) for a mathematically rigorous definition which fits all of our human feelings about a very loaded word. This is often a doomed endeavour from the start, as human intuitions don't neatly map onto logic. Also, humans might disagree on what things count as or do not count as knowledge. So let's attempt to right this wrong question:

> Imagine a given system is described as "knowing" something. What is the process that leads to the accumulation of said knowledge likely to look like?

I think this is much better.

We limit ourselves to systems which can definitely be said to "know" something. This allows us to pick a starting point. This might be a human, GPT-3, or a neural network which can tell apart dogs and fish. In fact this will be my go-to answer for the future. We also don't need to perfectly specify the process which generates knowledge all at once, only comment on its likely properties.

Properties of "Learning"

Say we have a very general system, with parameters θ, with t representing time during learning. Let's say they're initialized as θ_0 according to some random distribution. Now it interacts with the dataset which we will represent with X, taken from some distribution over possible datasets. The learning process will update θ_0, so we can represent the parameters the parameters after some amount of time as $\theta(\theta_0;\ X;\ t)$. This reminds us that the set of parameters depends on three things: the initial parameters, the dataset, and the amount of training.

Consider $\theta(\theta_0;\ X;\ 0)$. This is trivially equal to θ_0, and so it depends only on the choice of θ_0. The dataset has had no chance to affect the parameters in any way.

So what about as $t \to \infty$? We would expect that $\theta_\infty(\theta_0;\ X) = \theta(\theta_0;\ X;\ \infty)$ depends mostly on the choice of X and much less strongly on θ_0. There will presumably be some dependency on initial conditions, especially for very complex models like a big neural network with many local minima. But mostly it's ω which influences θ.

So far this is just writing out basic sequences stuff. To make a map of the city you have to look at it, and to learn your model has to causally entangle itself with the dataset. But let's think about what happens when ω is slightly different.

Changes in the world

So far we've represented the whole dataset with a single letter X, as if it were just a number or something. But in reality it will have many, many independent parts. Most datasets which are used as inputs to learning processes are also highly structured.

Consider the dog-fish discriminator, trained on the dataset $X_{dog/fish}$. The system $\theta_\infty(\theta_0;\ X_{dog/fish})$ could be said to have "knowledge" that "dogs have two eyes". One thing this means if we instead fed it an X which was identical except every dog had three eyes (TED) then the final values of θ would be different. The same is true of facts like "fish have scales", "dogs have one tail". We could express this as follows:

$$\theta_\infty(\theta_0;\ X_{dog/fish} + \Delta X_{TED})$$

Where ΔX_{TED} is the modification of "photoshopping the dogs to have three eyes". We now have:

$$\theta_\infty(\theta_0;\ X_{dog/fish} + \Delta X_{TED}) = \theta_\infty(\theta_0;\ X_{dog/fish}) + \Delta\theta_\infty(\theta_0;\ X_{dog/fish};\ \Delta X_{TED})$$

Now let's consider how $\Delta\theta_\infty(\theta_0;\ X;\ \Delta X)$ behaves. For lots of choices of ΔX it might just be a series of random changes tuning the whole set of θ values. But from my knowledge of neural networks, it might not be. Lots of image recognizing networks have been found to contain neurons with *specific functions* which relate to structures in the data, from simple line detectors, all the way up to "cityscape" detectors.

For this reason I suggest the following hypothesis:

> Structured and localized changes in the dataset that a parameterized learning system is exposed to will cause localized changes in the final values of the parameters.

Impracticalities and Solutions

Now it would be lovely to train all of GPT-3 twice, once with the original dataset, and once in a world where dogs are blue. Then we could see the exact parameters that lead it to return sentences like "the dog had [chocolate rather than azure] fur". Unfortunately rewriting the whole training dataset around this is just not going to happen.

Finding the flow of information, and influence in a system is easy if you have a large distribution of different inputs and outputs (and a good idea of the direction of causality). If you have just a single example, you can't use any statistical tools at all.

So what else can we do? Well we don't just have access to θ_∞. In principle we could look at the course of the entire training process and how θ changes over time. For each timestep, and each element of the dataset X, we could record how much each element of θ is changed. We'll come back to this

Let's consider the dataset as a function of the external world: X(Ω). All the language we've been using about knowledge has previously only applied to the dataset. Now we can describe how it applies to the world as a whole.

For some things the equivalence of knowledge of X and Ω is pretty obvious. If the dataset is being used for a self-driving car and it's just a bunch of pictures and videos then basically anything the resulting parameterised system knows about X it also knows about Ω. But for obscure manufactured datasets like [4000 pictures of dogs photoshopped to have three eyes] then it's really not clear.

Either way, we can think about Ω as having influence over X the same way as we can think about X as having influence over θ_∞. So we might be able to form hypotheses about this whole process. Let's go back to $X_{dog/fish}$. First off imagine a change $\Omega_{new} = \Omega + \Delta\Omega$, such as "dogs have three eyes". This will change some elements of X more than others. Certain angles of dog photos, breeds of dogs, will be changed more. Photos of fish will stay the same!

Now we can imagine a function $\Delta\theta(\theta_0;\ X(\Omega);\ \Delta X(\Omega;\ \Delta\Omega))$. This represents some propagation of influence from $\Omega \to X \to \theta$. Note that the influence of Ω on X is independent of our training process or θ_0. This makes sense because different bits of the training dataset contain information about different bits of the world. How different training methods extract this information might be less obvious.

The Training Process

During training, $\theta(t)$ is exposed to various elements of X and updated. Different elements of X will update $\theta(t)$ by different amounts. Since the learning process is about transferring influence over θ from θ_0 to Ω (acting via X), we might expect that for a given element of X, it has more "influence" over the final values of the elements of θ which were changed the most due to exposure to that particular element of X during training.

This leads us to a second hypothesis:

> The degree to which an element of the dataset causes an element of the parameters to be updated during training is correlated with the degree to which a change to that dataset element would have caused a change in the final value of the parameter.

Which is equivalent to:

> Knowledge of a specific properties of the dataset is disproportionately concentrated in the elements of the final parameters that have been updated the most during training when "exposed" to certain dataset elements that have a lot of mutual information with that property.

For the dog-fish example: elements of parameter space which have updated disproportionately when exposed to photos of dogs that contain the dogs' heads (and therefore show just two eyes), will be more likely to contain "knowledge" of the fact that "dogs have two eyes".

This naturally leads us to a final hypothesis:

> Correlating update-size as a function of dataset-element across two models will allow us to identify subsets of parameters which contain the same knowledge across two very different models.

Therefore

> Access to a simple interpreted model of a system will allow us to rapidly infer information about a much larger model of the same system if they are trained on the same datasets, and we have access to both training histories.

Motivation

I think an AI which takes over the world will have a very accurate model of human morality, it just won't care about it. I think that one way of getting the AI to not kill us is to extract parts of the human utility-function-value-system-decision-making-process-thing from its model and tell the AI to do those. I think that to do this we need to understand more about where exactly the "knowledge" is in an inscrutable model. I also find thinking about this very interesting.

Knowledge Localization: Tentatively Positive Results on OCR

Epistemic Status: Original research, see previous post for more information

The Plan

1. Hard-code a simple ML system in Python (yikes!) so I can see all of the moving parts and understand it better.
2. Give the ML system something to learn (in this case OCR based on the MNIST dataset, because it's easy to get hold of and basically a solved problem in terms of actually classifying characters)
3. Actually test my hypothesis:

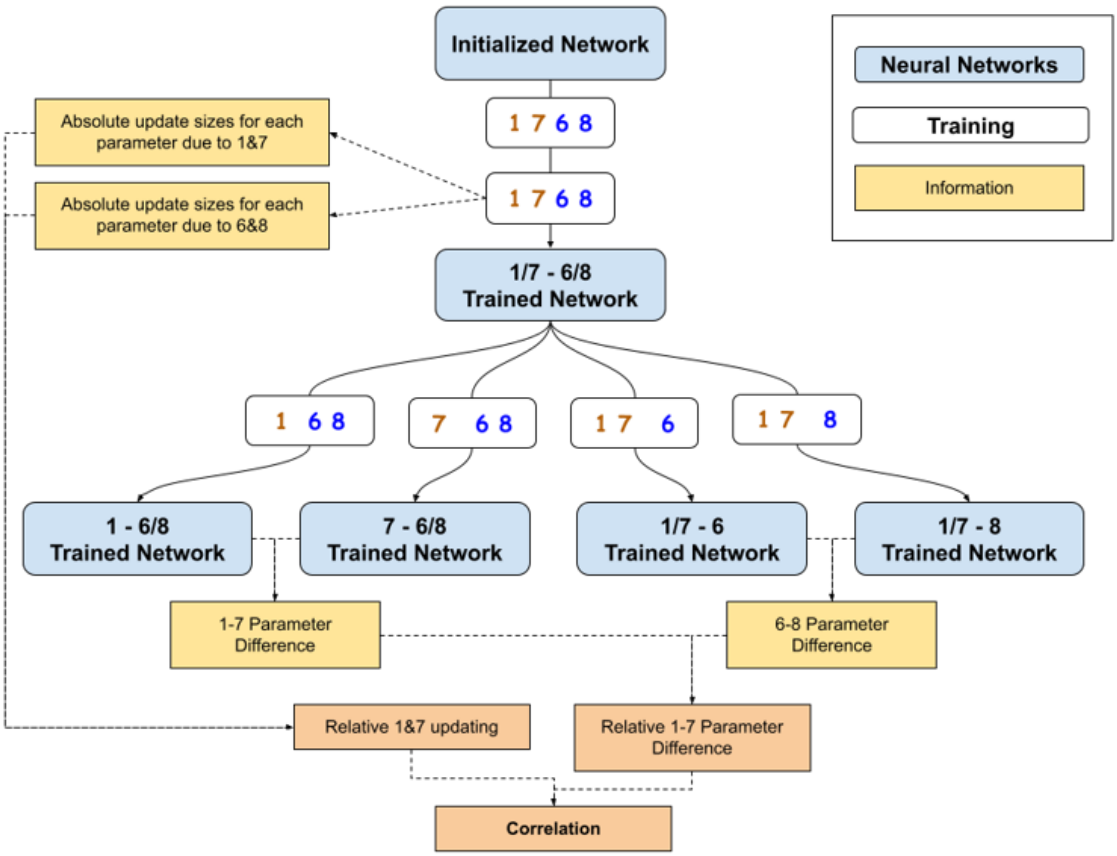
Methods

The neural network used is simple, 784 neurons on the input layer, 100 hidden neurons, 2 output. The activation function used is log(exp(x) + 1)). All weights and biases were initialized with He initialization. For compatibility reasons the data label is either $[1, 0]$ or $[0, 1]$. The input values are normalized to be between 1/256 and 1 rather than 0 and 255.

The learning process involves backpropagating on a single datapoint (one labelled character) at a time. Overall loss appears to reach a minimum early within the first training epoch. For this reason the first 1200 of 60,000 datapoints were used for phase 1. This refers to training a single neural network on a dataset consisting of all of the 6s and 8s; and 1s and 7s, from the first 12000 datapoints (keeping roughly equal numbers of the two types of data helps to keep the biases and weights towards the last layer from having large changes in any one direction).

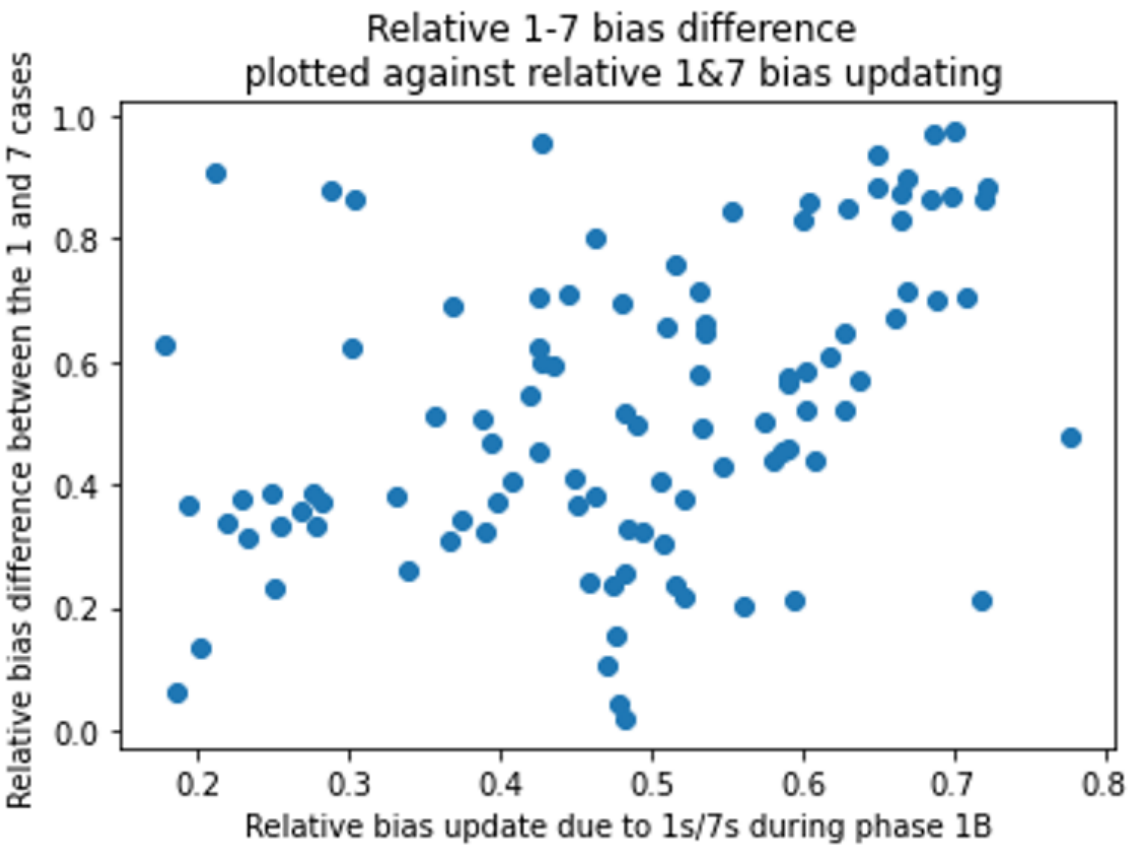
During the second half of phase 1, the absolute value of the change of each parameter is added to one of two running tallies, one for 1s/7s and one for 6s/8s. Then the total updating done to each parameter by each class of datapoint can be found. This amounts to summing the absolute value of the update sizes from each datapoint. The "relative updating due to 1s/7s" variable for each parameter is then a/(a + b) where a is the sum of the updates due to 1s and 7s, and b is the sum of the updates due to 6s and 8s. (The correlation still holds whether we look at all of phase 1 or just the second half, but I've found it to be slightly stronger for the second half of training.

Then four copies of the network are made. Both are trained on the remainder of the MNIST dataset, one on 1s and 6s/8s, one on 7s and 6s/8s; then one on 1s/7s and 6s, and one on 1s/7s and 8s. The difference in the final parameter values between these two pairs of networks is then found. We then take a similar function c/(c + d) where c is the absolute difference between the parameter values of the 1s vs 6s/8s case and the 7s vs 6s/8s case, and d is the difference between the parameter values of the 6s vs 1s/7s case, and the 8s vs 1s/7s case.

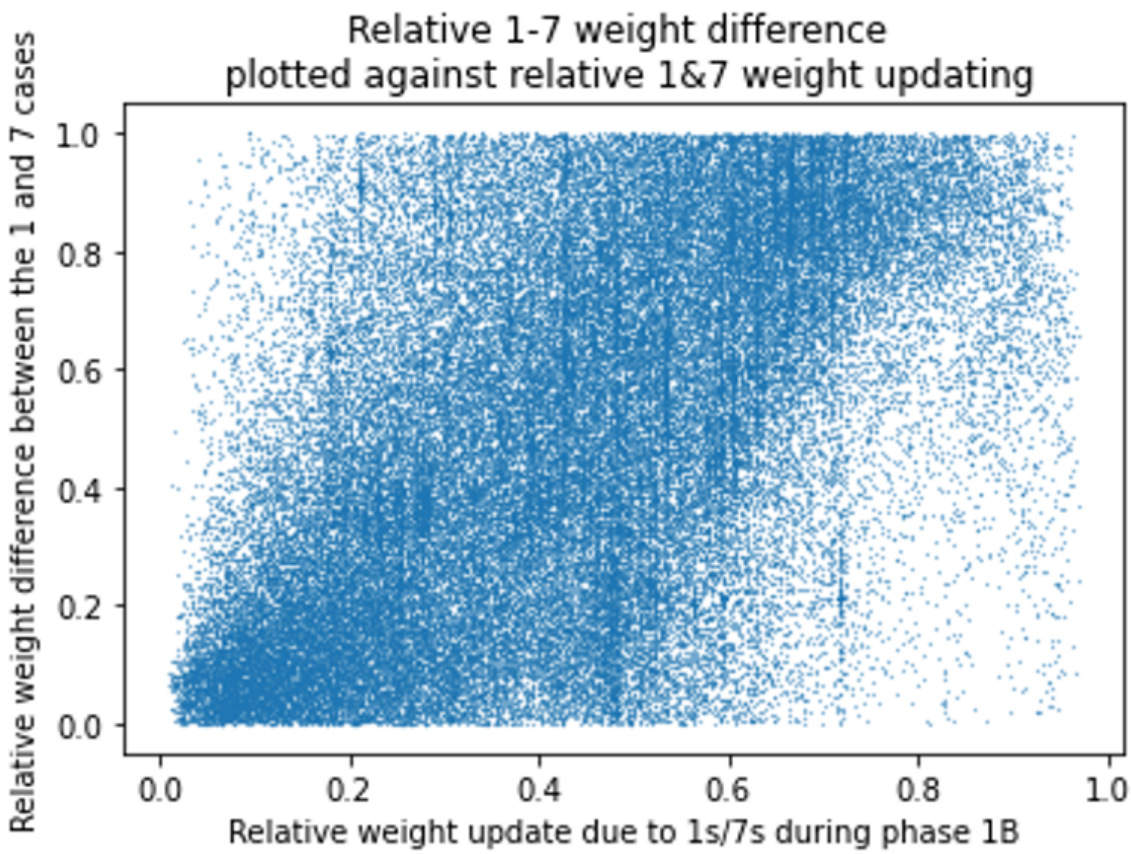


Analysis

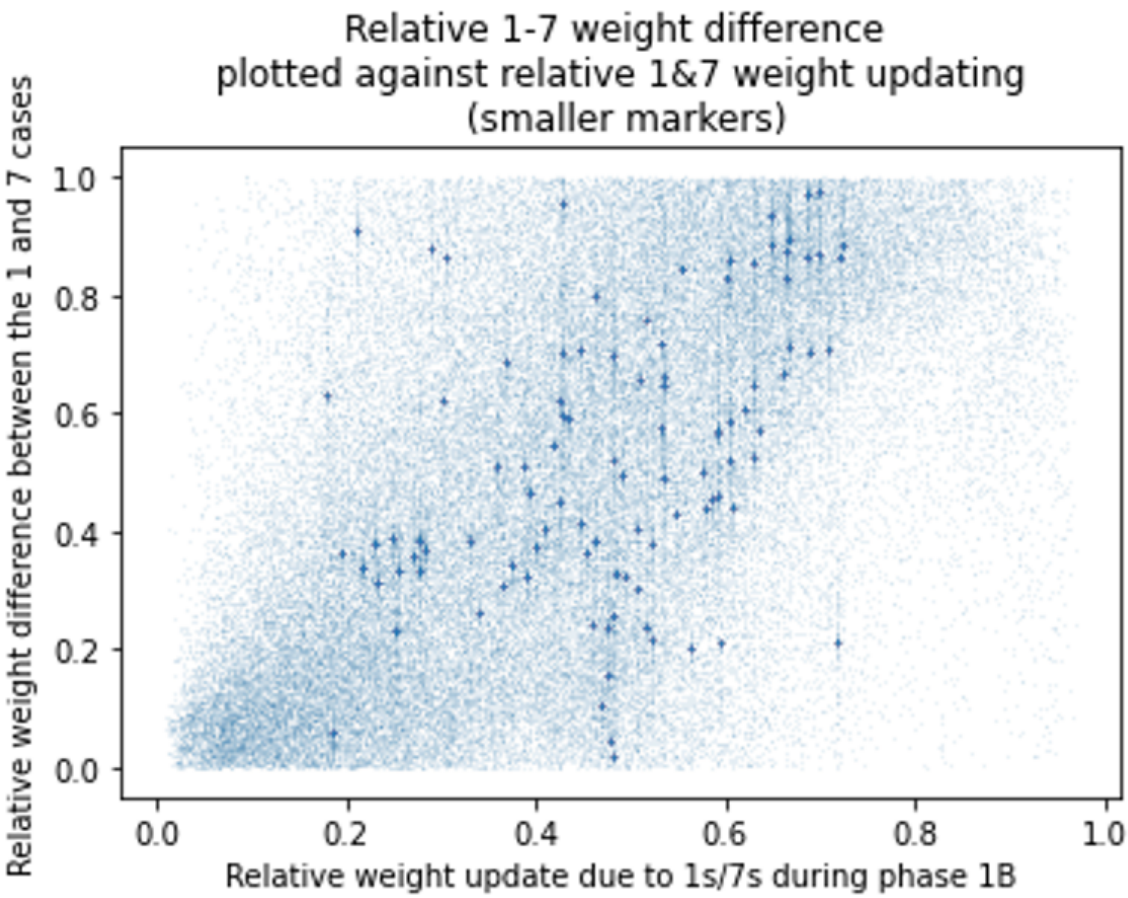
The hypothesis I was testing is about what update sizes during training can tell us about where knowledge is stored in a network. The primary plots are of "relative amount of updating due to 1s/7s" against the relative difference between the 1s vs 6s/8s and 7s vs 6s/8s forks. A plot which was strong evidence in favour of my hypothesis would basically just be a positive correlation. The parameters which were (relatively) more different in the cases of the data being 1s vs 7s would have updated more due to 1s and 7s.



Weak-ish correlation for biases, $R^2 = 0.17$, $p < 0.0001$ by T-test if you're into that.

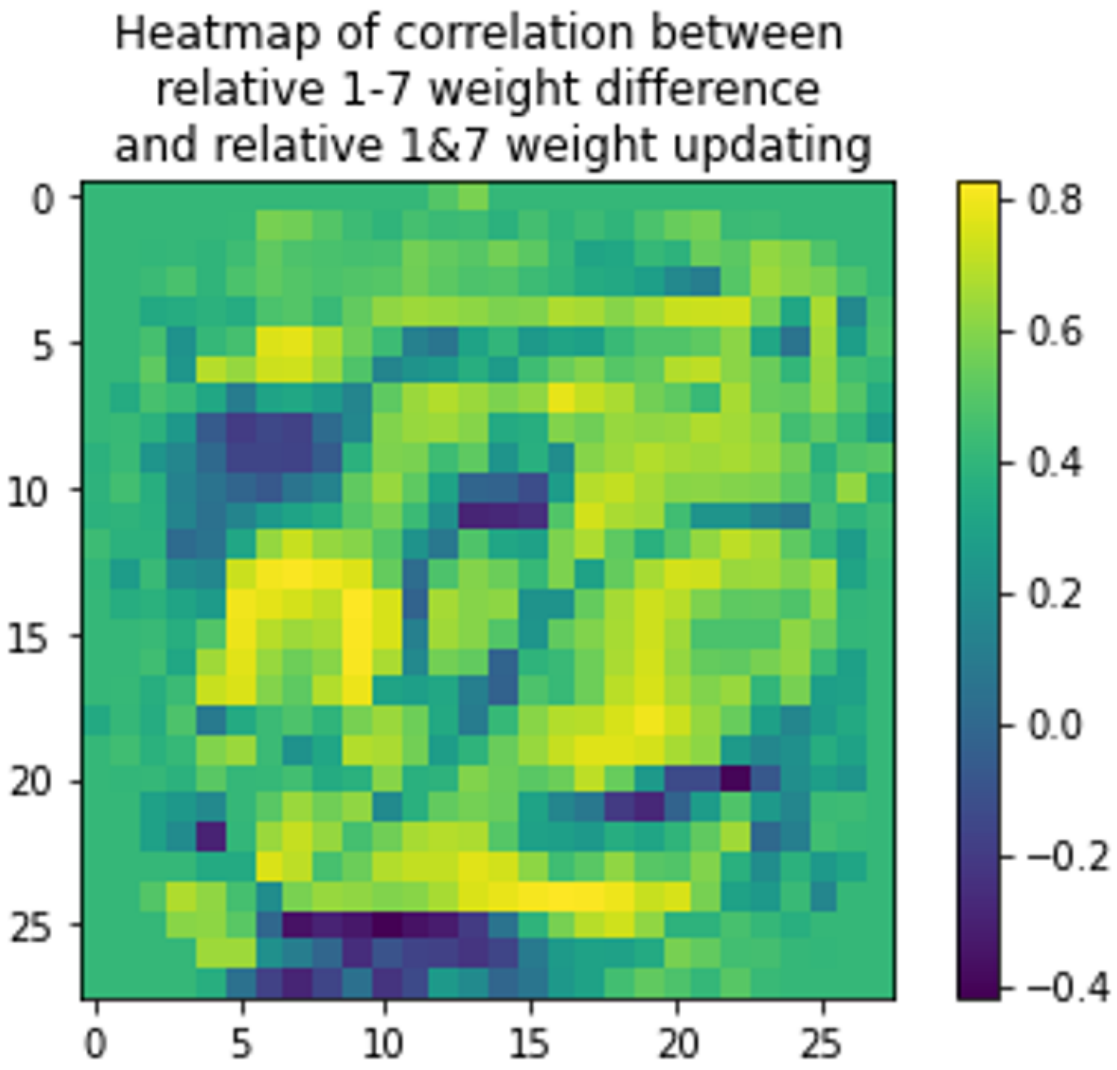


Correlation for weights is very clear. $R^2 = 0.28$. I'm not going to calculate a p-value here because I have eyes.



If we make the markers smaller we can see little clusters. I'm pretty confident each of these clusters is a set of weights going into one of the second layer neurons. Lots of weights will be coming from input neurons which only ever have a value of 1/256, so they're all effectively the same. The pattern of clusters is almost identical to the pattern of bias updates.

More interestingly we can also plot the correlation between our two values for the weights coming *out of* each of the input neurons.



I'm not completely sure how to interpret this. Overall correlation is mostly > 0 but we have some regions of negative correlation. Round the edges the correlation is much weaker as these only ever take 1/256 as input.

Conclusions and Further Plans

So this worked pretty well. This is some weak evidence for my hypothesis. Without a doubt the most interesting thing is that **this only worked for a ReLU-like activation function**. I was originally using sigmoid activation functions and I couldn't get anything like this. That's really weird and other than ReLU being magic, I have no hypotheses.

I want to try this out on other systems. I wonder if a more abstraction-heavy system would make a better candidate. I'd like to find out when these correlations are stronger or weaker. For this I'll need to mess around with some ML libraries to see if I can extract the data I want. Hard-coding a neural network in the slowest language known to man was a bad time.

Defining Optimization in a Deeper Way Part 1

My aim is to define optimization without making reference to the following things:

1. **A "null" action or "nonexistence" of the optimizer**. This is generally poorly defined, and choices of different null actions give different answers.
2. **Repeated action**. An optimizer should still count even if it only does a single action.
3. **Uncertainty**. We should be able to define an optimizer in a fully deterministic universe.
4. **Absolute time at all**. This will be the hardest, but it would be nice to define optimization without reference to the "state" of the universe at "time t".

Attempt one

First let's just eliminate the concept of a null action. Imagine the state of the universe at a time t.

Let's divide the universe into two sections and call these A and B. They have states S^A and S^B. If we want to use continuous states we'll need to have some metric $D(s_1, s_2)$ which applies to these states, so we can calculate things like the variance and entropy of probability distributions over them.

Treat S^A and S^B as part of a Read-Eval-Print-Loop. Each S^A_t produces some output O^A_t which acts like a function mapping $S^B_t \rightarrow S^B_{t+1}$, and vice versa. O^A_t can be thought of as things which cross the Markov blanket.

Sadly we still have to introduce probability distributions. Let's consider a joint probability distribution $P^{AB}_t(s^A, s^B)$, and also the two individual probability distributions $P^A_t(s^A)$ and $P^B_t(s^B)$.

By defining distributions over O outputs based on the distribution P^{AB}_t, we can define $P^{AB}_{t+1}(s^A, s^B)$ in the "normal" way. This looks like integrating over the space of s_A and s_B like so:

$$P^{AB}_{t+1}(s^A_{t+1}, s^B_{t+1}) = \int P^{AB}_t(s^A_t, s^B_t)\ \delta(O^A_t(s^A_t) - s^B_{t+1})\ \delta(O^B_t(s^B_t) - s^A_{t+1})\ ds^A_t\, ds^B_t$$

What this is basically saying is that to define the probability distribution of states s^A_{t+1} and s^B_{t+1}, we integrate over all states s^A_t and s^B_t and sum up the states where the O^A_t corresponding to s^A_t maps s^B_t to the given s^B_{t+1}.

Now lets define an "uncorrelated" version of P^{AB}_{t+1}, which we will refer to as $P^{\prime AB}_{t+1}$.

$$P^{\prime AB}_{t+1}(s^A_{t+1}, s^B_{t+1}) = \int P^A_t(s^A_t)\ P^B_t(s^B_t)\ \delta(O^A_t(s^A_t) - s^B_{t+1})\ \delta(O^B_t(s^B_t) - s^A_{t+1})\ ds^A_t\, ds^B_t$$

This loosely represents what happens if we decorrelate s^A and s^B. In the language of humans, this is like an agent taking a random move from a selection.

We can refer to a probability distribution P^{AB}_t as an "optimizing" probability distribution if $P^{\prime AB}_{t+1}$ is higher entropy than P^{AB}_{t+1}.

For an example, imagine the universe is divided into two parts: a room R and a thermostat T. The room can have states in the set $s^R \in \{\text{hot}, \text{lukewarm}, \text{cold}\}$, and the thermostat can have states $s^T \in \{\text{high}, \text{low}, \text{off}\}$. Imagine that O^R and O^T are defined as follows:

$$O^T[\text{high}]: \begin{cases} \text{hot} \to \text{hot} \\ \text{warm} \to \text{hot} \\ \text{cold} \to \text{warm} \end{cases}$$

$$O^T[\text{low}]: \begin{cases} \text{hot} \to \text{hot} \\ \text{warm} \to \text{warm} \\ \text{cold} \to \text{cold} \end{cases}$$

$$O^T[\text{off}]: \begin{cases} \text{hot} \to \text{warm} \\ \text{warm} \to \text{cold} \\ \text{cold} \to \text{cold} \end{cases}$$

$$O^R[\text{hot}]: \begin{cases} \text{high} \to \text{off} \\ \text{low} \to \text{off} \\ \text{off} \to \text{off} \end{cases}$$

$$O^R[\text{warm}]: \begin{cases} \text{high} \to \text{low} \\ \text{low} \to \text{low} \\ \text{off} \to \text{low} \end{cases}$$

$$O^R[\text{cold}]: \begin{cases} \text{high} \to \text{high} \\ \text{low} \to \text{high} \\ \text{off} \to \text{high} \end{cases}$$

Basically the thermostat decides whether the room gets warmer, stays the same, or gets colder, and the thermostat.

We can also consider the probability mass flowing from each of the nine states to another one:

	hot	warm	cold
high	(hot, off)	(hot, low)	(warm, high)
low	(hot, off)	(warm, low)	(cold, high)
off	(warm, low)	(cold, low)	(cold, high)

Imagine the following $P_0^{TR}(s^T, s^R)$:

	hot	warm	cold
high	0	0	1/3
low	0	1/3	0
off	1/3	0	0

This will give us the following $P_1^{TR}(s^T, s^R)$:

	hot	warm	cold
high	0	1/3	0
low	0	1/3	0
off	0	1/3	0

Which has 1.6 bits of entropy.

And the following $P_1^{'TR}(s^T, s^R)$:

	hot	warm	cold
high	0	1/9	2/9
low	1/9	1/9	1/9
off	2/9	1/9	0

Which has 2.7 bits of entropy.

This means that the joint-ness of the probability distribution P_0^{RT} has removed 1.1 bits of entropy from the system. We say that our choice of P_0^{RT} is optimizing, with an optimizing strength of 1.1 bits.

But what if we consider a "smarter" thermostat, which turns off just *before* the temperature changes.

$$O^R[hot]: \begin{cases} high \to off \\ low \to off \\ off \to low \end{cases}$$

$$O^R[warm]: \begin{cases} high \to low \\ low \to low \\ off \to low \end{cases}$$

$$O^R[cold]: \begin{cases} high \to low \\ low \to high \\ off \to high \end{cases}$$

	hot	warm	cold
high	(hot, off)	(hot, low)	(warm, low)
low	(hot, off)	(warm, low)	(cold, high)
off	(warm, off)	(cold, low)	(cold, high)

With the same choice of $P_0^{TR}(s^T, s^R)$:

	hot	warm	cold
high	0	0	1/3
low	0	1/3	0
off	1/3	0	0

This will give us the following $P_1^{TR}(s^T, s^R)$:

	hot	warm	cold
high	0	0	0
low	0	1	0
off	0	0	0

With an entropy of zero.

And the following $P_1^{TR}(s^T, s^R)$:

	hot	warm	cold
high	0	0	2/9
low	1/9	1/3	1/9
off	2/9	0	0

Which has 2.2 bits of entropy.

In the new system, P_0^{RT} has an optimizing strength of 2.2 bits, approximately twice as much. This indicates that the latter system is "better" at optimizing the distribution P_0^{RT} in some way.

So we have eliminated the idea of needing the optimizer to have clearly-defined existence/nonexistence cases, or needing some "null" action to compare its outputs to. This is good. We have also eliminated the concept of repeated action.

Next I will attempt to eliminate the need to start with a probability distribution. In both of the examples above, our choice of P_0^{RT} was important. I want to find a more "natural" way of defining probability distributions.

Defining Optimization in a Deeper Way Part 2

We have successfully eliminated the concepts of **null actions and nonexistence** from our definition of optimization. We have also eliminated the concept of **repeated action**. We are halfway there, and now have to eliminate **uncertainty** and **absolute time**. Then we will have achieved the goal of being able to wrap a 3D hyperplane boundary around a 4D chunk of relativistic spacetime and ask ourselves "Is this an optimizer?" in a meaningful way.

I'm going to tackle uncertainty next.

TL;DR I have allowed for a mor

We've already defined, for a deterministic system, that a joint probability distribution $P_t^{AB}(s^A, s^B)$ has a numerical optimizing-ness, in terms of entropy. Now I want to extend that to a non-joint probability distribution of the form $P^A(s^A)P^B(s^B)$. We can do this by defining $P_{t-1}^{A}(s^A)$ and $P_{t-1}^{B}(s^B)$ for the *previous* timestep.

We can then define $P_t^{AB}(s^A, s^B)$ as by stepping forwards from $t-1$ to t as before, according to the dynamics of the system.

A question we might want to ask is, for a given $P_{t-1}^{A}(s^A)$ and $P_{t-1}^{B}(s^B)$, how "optimizing" is the distribution $P_t^{AB}(s^A, s^B)$?

The Dumb Thermostat

Lets apply our new idea to the previous models, the two thermostats. Lets begin with uncorrelated, maximum entropy distributions.

For thermostat 1 we have the dynamic matrix:

	hot	warm	cold
high	(hot, off)	(hot, low)	(warm, high)
low	(hot, off)	(warm, low)	(cold, high)
off	(warm, off)	(cold, low)	(cold, high)

(In this matrix, the entry for a cell represents the state at time = t + 1 *given the coordinates of that cell represent the state at* time = t*)*

With the P^{RT}_{t-1} distribution:

	hot	warm	cold
high	1/9	1/9	1/9
low	1/9	1/9	1/9
off	1/9	1/9	1/9

As an aside this has 3.2 bits of entropy.

Leading to the P^{RT}_{t} distribution:

	hot	warm	cold
high	0	1/9	2/9
low	1/9	1/9	1/9
off	2/9	1/9	0

This gives us the "standard" P^{RT}_{t+1} distribution of:

	hot	warm	cold
high	0	2/9	1/9
low	1/9	1/9	1/9
off	1/9	2/9	0

And the "decorrelated" $P^{\prime RT}_{t+1}$ distribution is actually just the same as P^{RT}_{t} ! When we decorrelate the probabilities for s^R and s^T we just get back to the maximum entropy distribution and so $P^{\prime RT}_{t+1} = P^{RT}_{t}$

It's clear by inspection that the distributions P^{RT}_{t+1} and $P^{'RT}_{t+1}$ have the same entropy, so the decorrelated maximum entropy $P^{R}_{t-1}P^{T}_{t-1}$ does *not* produce an "optimizing" distribution at P^{RT}_{t}.

If we actually consider the dynamics of this system, we can see that this makes sense! The temperature actually either stays at (warm, low) or falls into the cycle:

$$(hot, low) \to (hot, off) \to (warm, off) \to (cold, low)$$

$$\to (cold, high) \to (warm, high) \to (hot, low)$$

So there's no compression of futures into a smaller number of trajectories.

The Smart Thermostat

What about our "smarter" thermostat? This one has the dynamic matrix:

	hot	warm	cold
high	(hot, off)	(hot, low)	(warm, low)
low	(hot, off)	(warm, low)	(cold, high)
off	(warm, low)	(cold, low)	(cold, high)

Well now our P^{RT}_{t} distribution looks like this:

	hot	warm	cold
high	0	0	2/9
low	1/9	1/3	1/9
off	2/9	0	0

Giving "standard" a P^{RT}_{t+1} of this:

	hot	warm	cold

high	0	0	1 / 9
low	0	7 / 9	0
off	1 / 9	0	0

And a "decorrelated" $P_t^{\prime RT}$ of:

	hot	warm	cold
high	2 / 27	2 / 27	2 / 27
low	5 / 27	5 / 27	5 / 27
off	2 / 27	2 / 27	2 / 27

Giving the decorrelated $P_{t+1}^{\prime RT}$:

	hot	warm	cold
high	0	0	7 / 27
low	2 / 27	1 / 3	2 / 27
off	7 / 27	0	0

Now in this case, these two *do* have different entropies. P_{t+1}^{RT} has an entropy of 1.0 bits, and $P_{t+1}^{\prime RT}$ has an entropy of 2.1 bits. This gives us a difference of 1.1 bits of entropy. This is the Optimizing-ness we defined in the last post, but I think it's actually somewhat incomplete.

Let's also consider the initial difference between P_t^{RT} and $P_t^{\prime RT}$. Decorrelating P_t^{RT} takes it from 2.2 to 3.0 bits of entropy. So the entropy difference *started off* at 0.8 bits. Therefore the *difference of the difference* in entropy is 0.3 bits.

The value of associated with P_{t-1}^{RT} is equal to $(S[P_{t+1}^{\prime RT}] - S[P_{t+1}^{RT}]) - (S[P_t^{\prime RT}] - S[P_t^{RT}])$. which can also be expressed as $S[P_{t+1}^{\prime RT}] + S[P_t^{RT}] - S[P_{t+1}^{RT}] - S[P_t^{\prime RT}]$. We might call this quantity the *adjusted optimizing-ness.*

Quantitative Data

The motivation for this was that a maximum entropy distribution is "natural" in some sense. This moves us towards not needing uncertainty. If we have a given state of a system, we might be able to "naturally" define a probability distribution around that state. Then we can measure the optimizing-ness of the next step's distribution.

What happens with a different P^{RT}_{t-1} condition? What if we have a distribution like this:

	hot	warm	cold
high	$\epsilon^2/4$	$\epsilon(1-\epsilon)/2$	$\epsilon^2/4$
low	$\epsilon(1-\epsilon)/2$	$(1-\epsilon)^2$	$\epsilon(1-\epsilon)/2$
off	$\epsilon^2/4$	$\epsilon(1-\epsilon)/2$	$\epsilon^2/4$

For some small epsilon in the second situation.

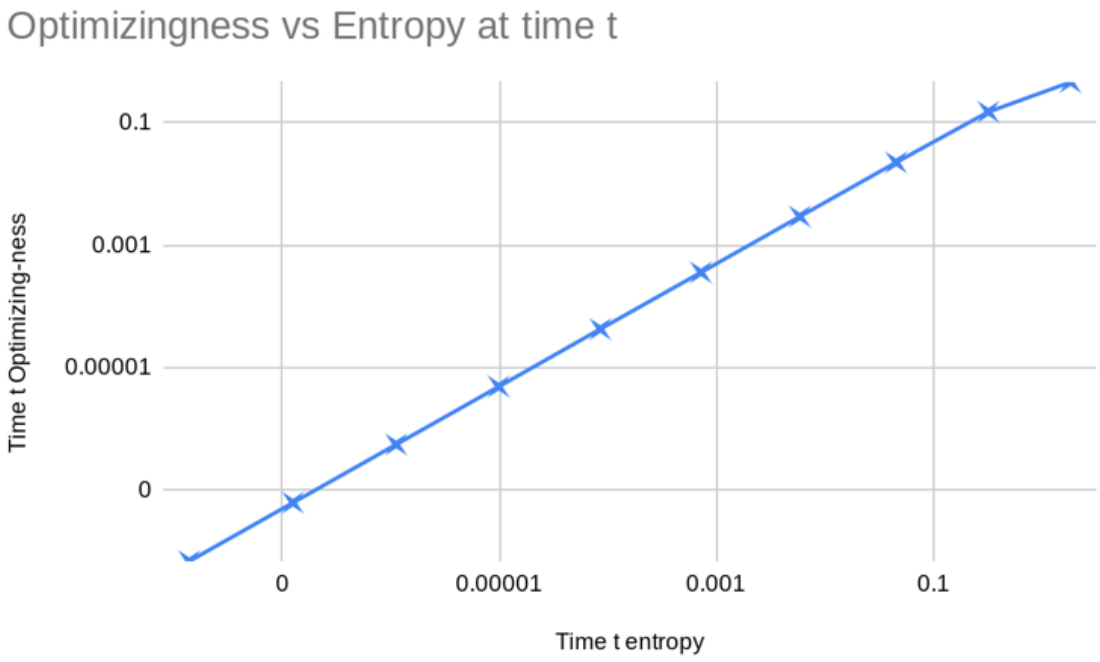
Now P^{RT}_{t} is like this:

	hot	warm	cold
high	0	0	$\epsilon(1-\epsilon)/2 + \epsilon^2/4$
low	$\epsilon(1-\epsilon)/2$	$(1-\epsilon)^2 + \epsilon^2/2$	$\epsilon(1-\epsilon)/2$
off	$\epsilon(1-\epsilon)/2 + \epsilon^2/4$	0	0

So P^{RT}_{t+1} is:

	hot	warm	cold
high	0	0	$\epsilon(1-\epsilon)/2$
low	0	$1-\epsilon(1-\epsilon)$	0
off	$\epsilon(1-\epsilon)/2$	0	0

While it is theoretically possible to decorrelate everything, calculate the next set of things, and keep going, it's a huge mess. Using values for epsilon between 0.1 and 10^{-10} we can make the following plot between the entropy of P^{RT}_{t-1} and our previously defined adjusted optimizing-ness.



It looks linear in the log/log particularly in the region where ϵ is very small. By fitting to the leftmost five points we get a simple linear relation: The adjusted optimizing-ness approaches half of the entropy of P^{RT}_{t-1}.

This is kind of weird. This might not be an optimal system to study, so let's look at another toy example. A more realistic model of a thermostat:

The Continuous Thermostat

The temperature of the room is considered as $S^R \in R$. The activity of the thermostat is considered as $T \in R$. Each timestep, we have the following updates:

$$S^T_{t+1} = S^R_t$$

$$S^R_{t+1} = S^R_t - k \; S^T_t$$

Consider the following distributions:

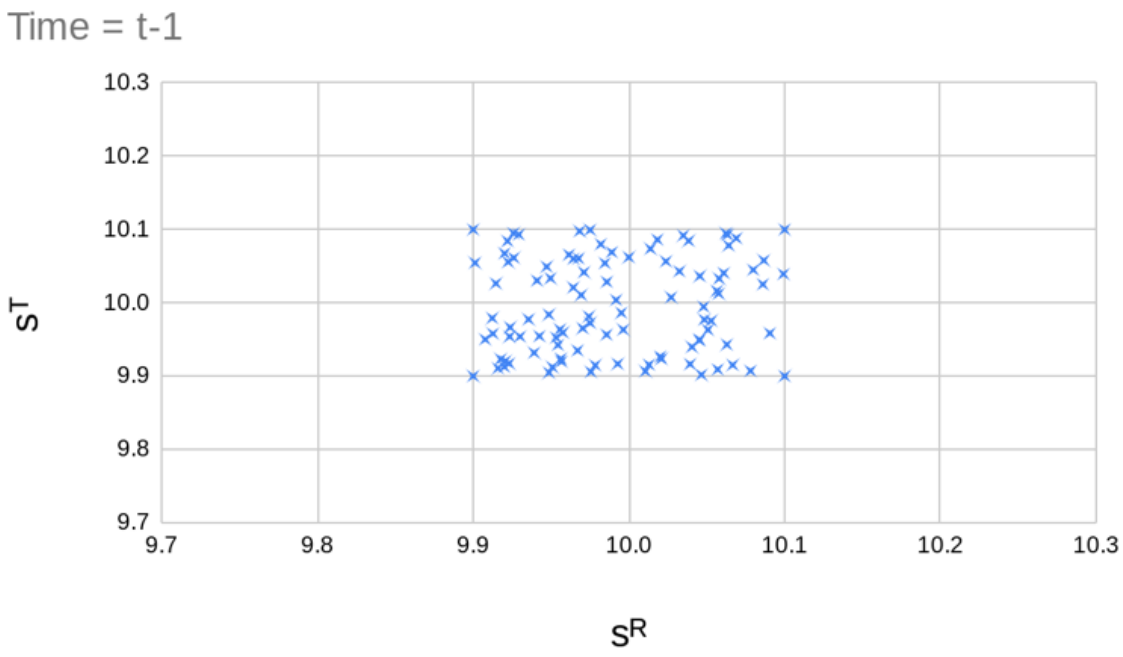
$$P^R_{t-1}(s^R) \sim U(10 - \epsilon/2, \; 10 + \epsilon/2)$$

$$P^{T}_{t-1}(s^{T}) \sim U(10 - \epsilon/2,\ 10 + \epsilon/2)$$

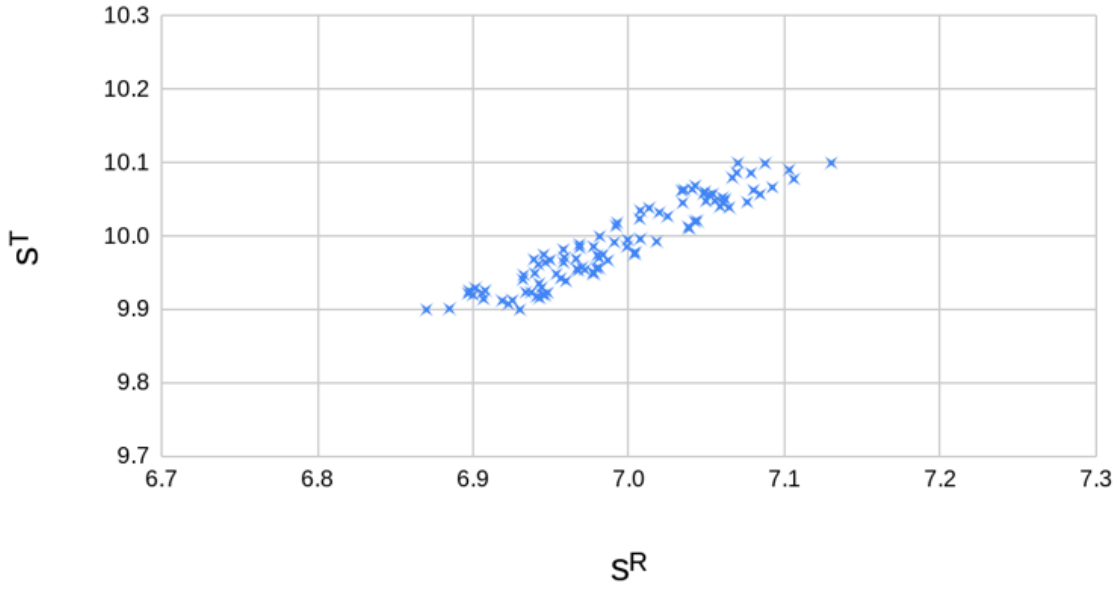
Where $U(a, b)$ refers to a uniform distribution between a and b. P^{RT}_{t-1} can be thought of as a square of side length ϵ centered on the point $(10, 10)$. P^{RT}_{t} turns out to be a rhombus. The corners transform like this:

Time = t − 1	Time = t
$(10 + \epsilon, 10 + \epsilon)$	$(10(1 - k) + \epsilon(1 - k), 10 + \epsilon)$
$(10 + \epsilon, 10 - \epsilon)$	$(10(1 - k) + \epsilon k, 10 + \epsilon)$
$(10 - \epsilon, 10 + \epsilon)$	$(10(1 - k) - \epsilon k, 10 - \epsilon)$
$(10 - \epsilon, 10 - \epsilon)$	$(10(1 - k) - \epsilon(1 - k), 1 - \epsilon)$

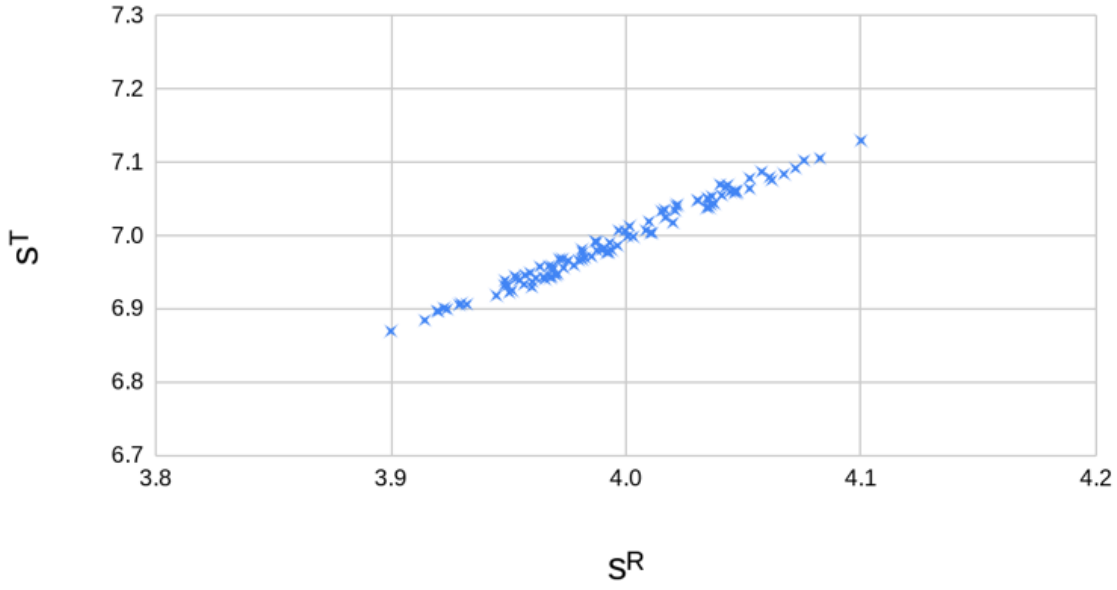
For $\epsilon = 0.1, k = 0.3$ the whole sequence looks like the following:

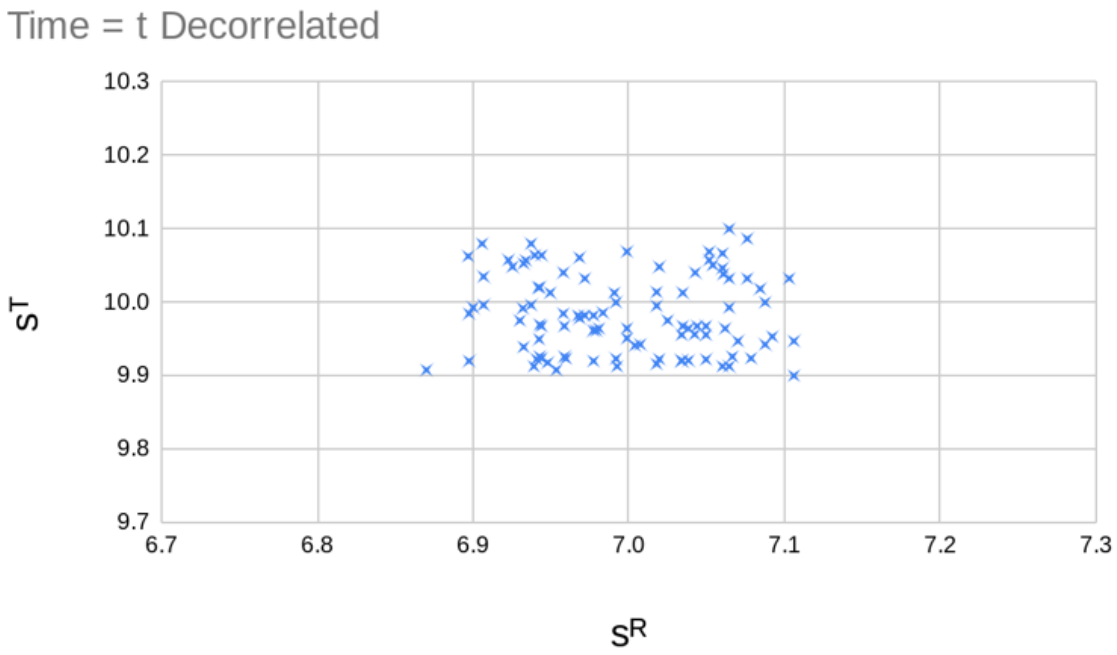


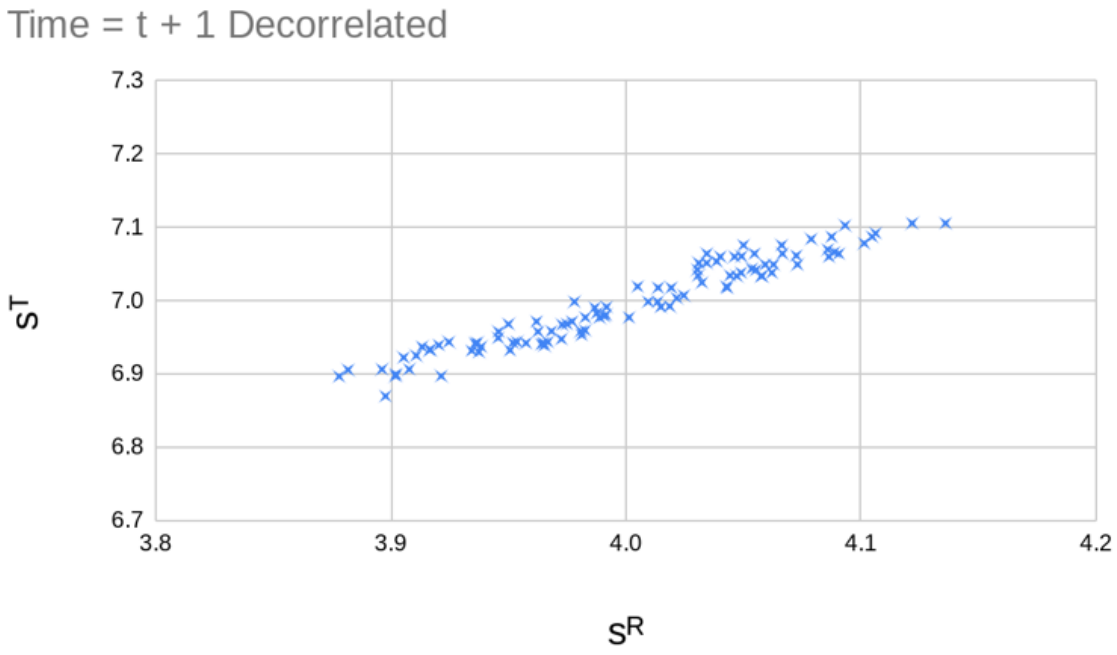
Time = t



Time = t + 1







So we clearly have some sort of optimization going on here. Estimating or calculating the entropy of these distributions is not easy. And when we use the entropy of a continuous distribution, we get results which depend on the choice of coordinates (or alternatively the choice of some weighting function). Entropies of continuous distributions may also be negative, which is quite annoying.

Perhaps calculating the variance will leave us better off? Sadly not. I tried it for gaussians of decreasing variance and didn't get much. The equivalent to our adjusted optimizing-ness

which we might define as $\log(V[P^{\prime AB}_{t+1}]) + \log(V[P^{AB}_{t}]) - \log(V[P^{\prime AB}_{t}]) - \log(V[P^{AB}_{t+1}])$ is always zero for this system. The non-adjusted version $\log(V[P^{\prime AB}_{t+1}]) - \log(V[P^{AB}_{t+1}])$ fluctuates a lot.

Where does this leave us?

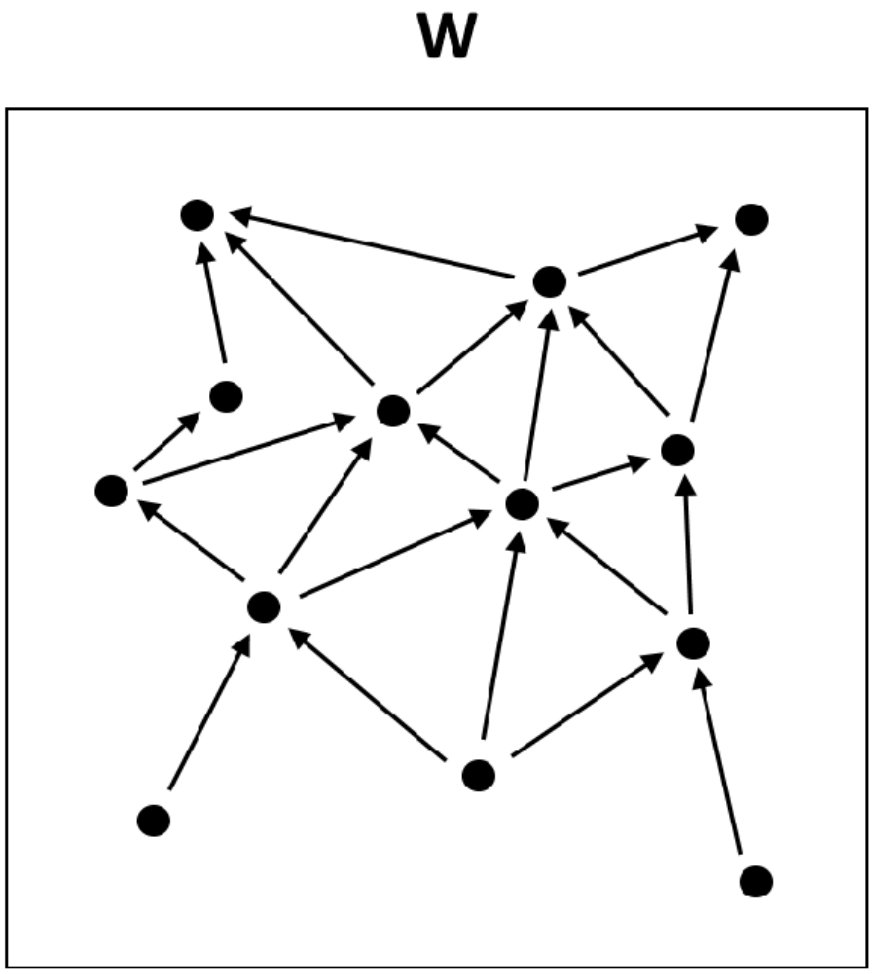
We can define whether something is an optimizer based on a probability distribution which need not be joint over A and B. This means we can define whether something is an optimizer for an arbitrarily narrow probability distribution, meaning we can take the limit as the probability distribution approaches a delta. We found an interesting relation between quantities in our simplified system but failed to extend it to a continuous system.

Defining Optimization in a Deeper Way Part 3

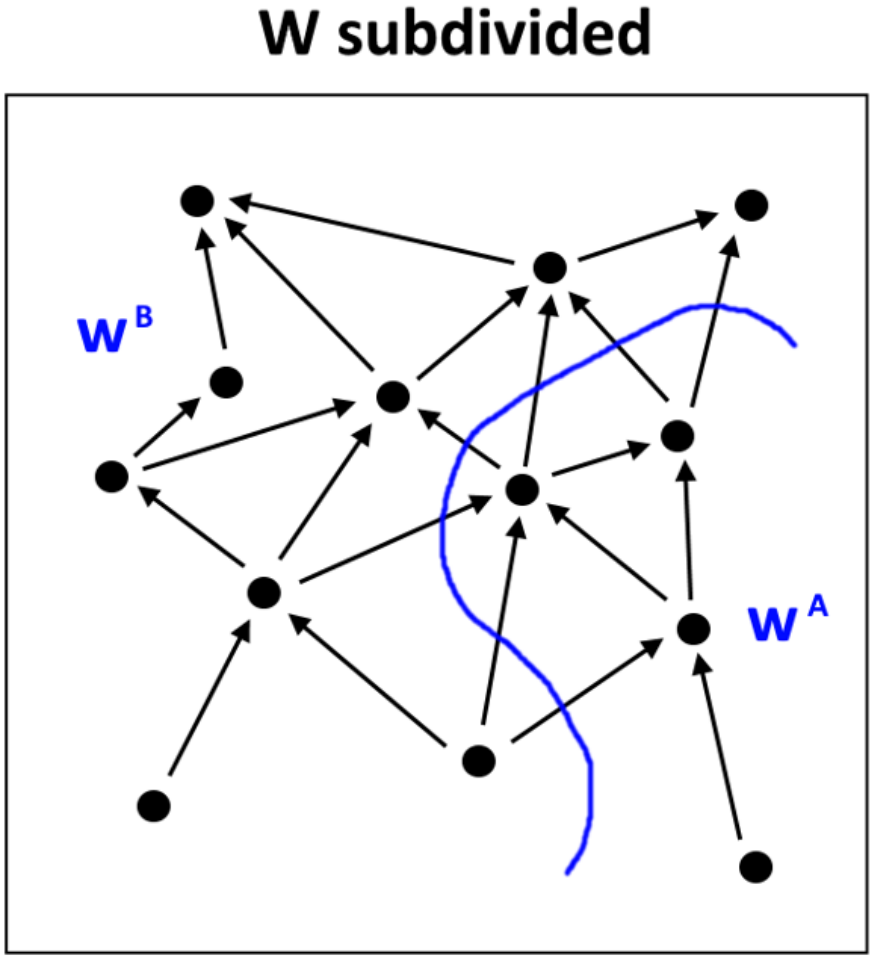
Last time I got stuck trying to remove the need for **uncertainty**. I have done it now! In the process I have also removed the need for **absolute time**.

The Setup

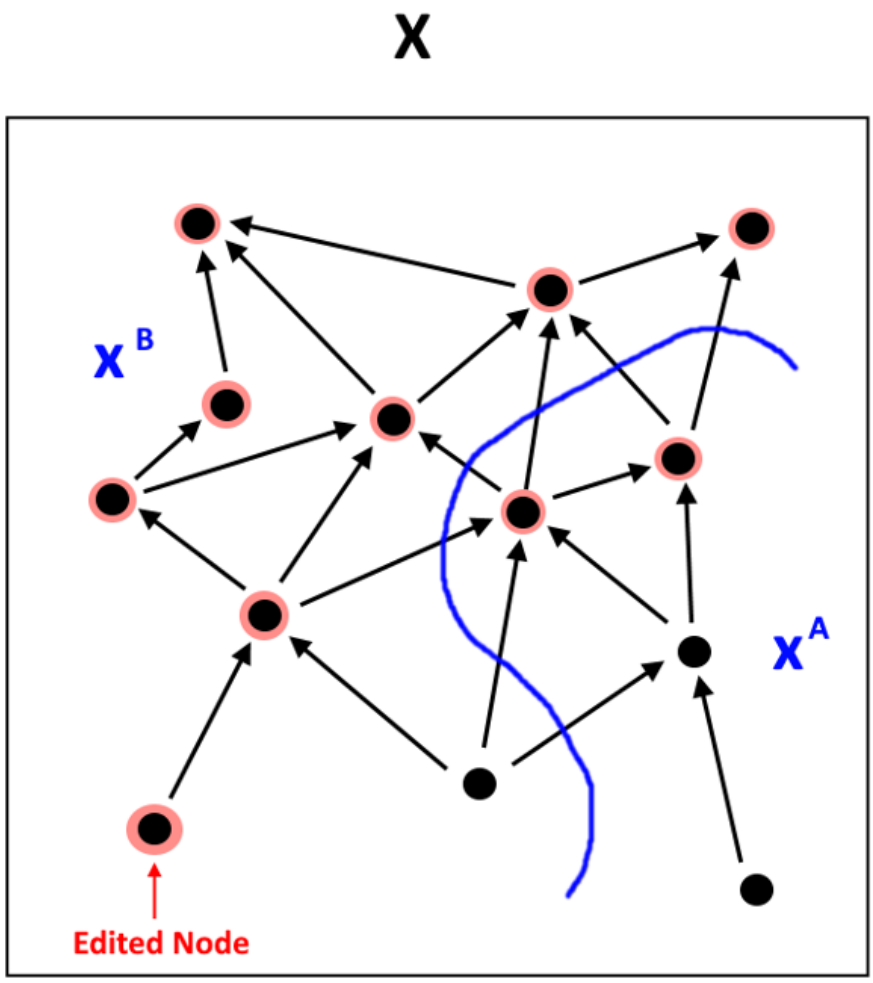
First we need to consider the universe as a causal network. This can be done without any notion of absolute time:



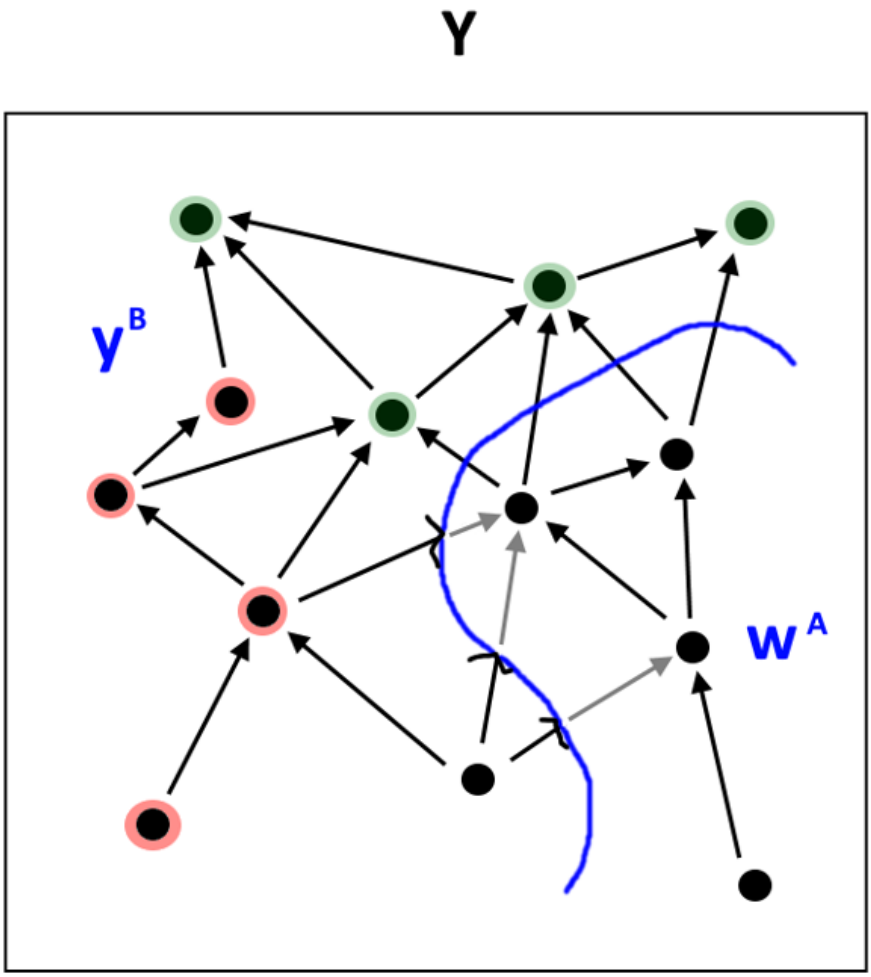
We can imagine dividing the world W into two parts, A and B. We can represent the states of each of these parts with vectors w^A and w^B. We might want this division to be somewhat "natural" in the sense of John Wentworth's work on natural abstractions and natural divisions of a causal network. But it's not particularly necessary.



Now consider what happens if we make a small alteration to one of the variables in w^B. Let's choose one which doesn't depend on any other variables. Everything which is "downstream" of it will be affected, so let's highlight those in red. Let's call this universe X.



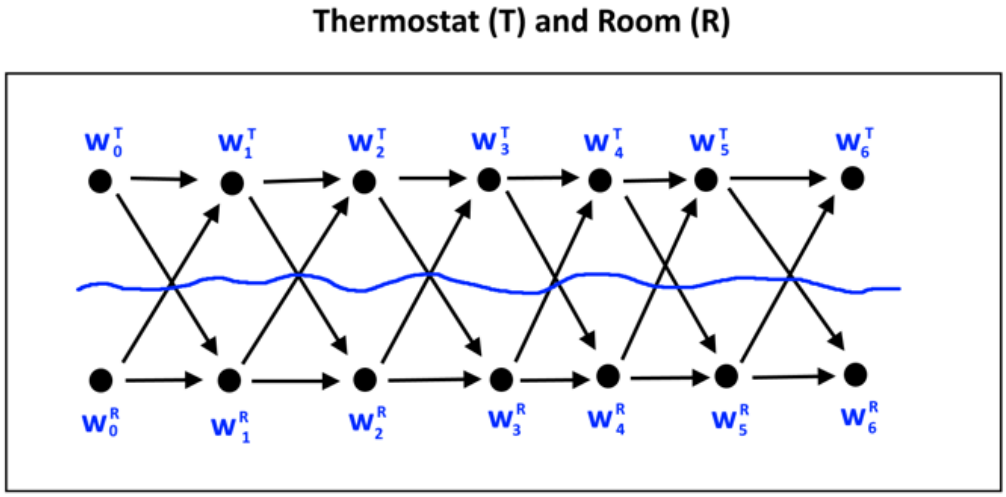
Now let's imagine that we take everything in A, and replace it with the version from W. So all of the influence coming out of A will be the same as it is in W. Consider what happens to B. We'll call this universe Y, and label everything which is different from X in green:



Now if any part of the universe is doing a decent job of optimizing, we would expect the influence of changing that node in X to be eliminated over time. Crucially, if A is optimizing B , then that difference should be eliminated in X, but *not eliminated as much* in Y , since w^A has no information about the fact that y^B is different to w^B.

Example

Imagine our state-continuous thermostat. The causal network looks like this:

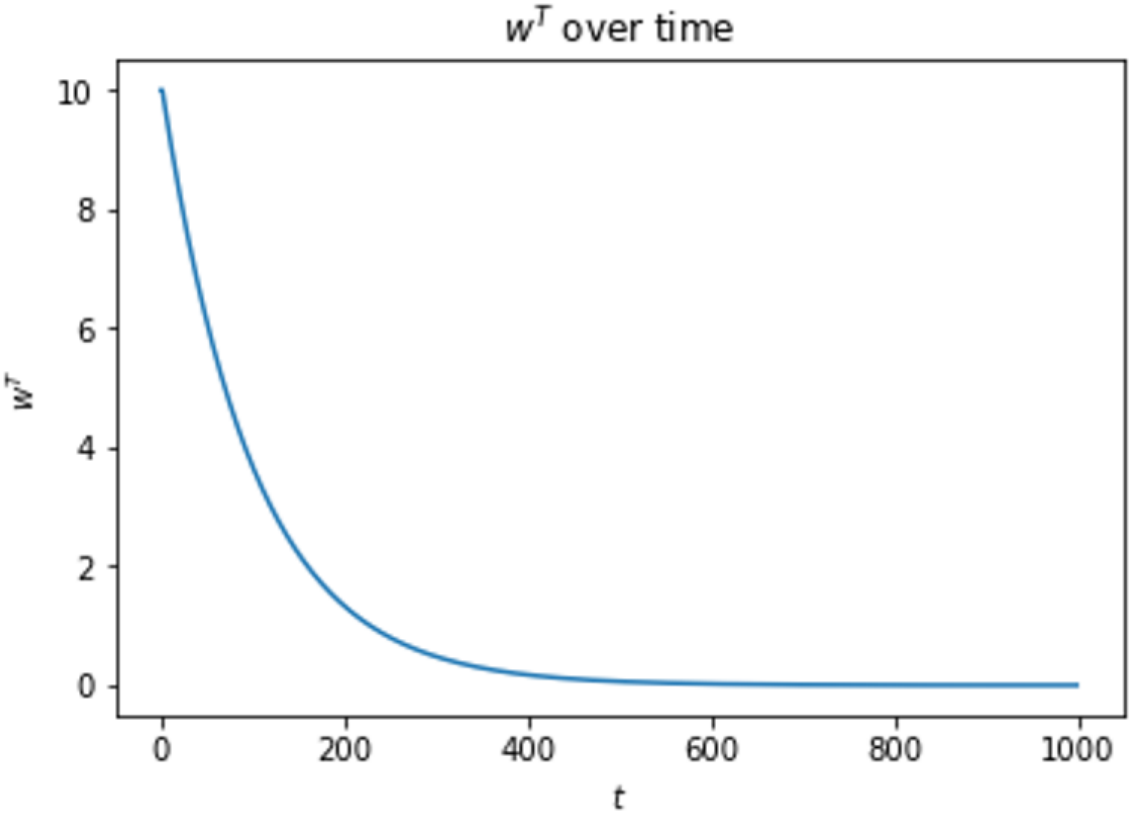


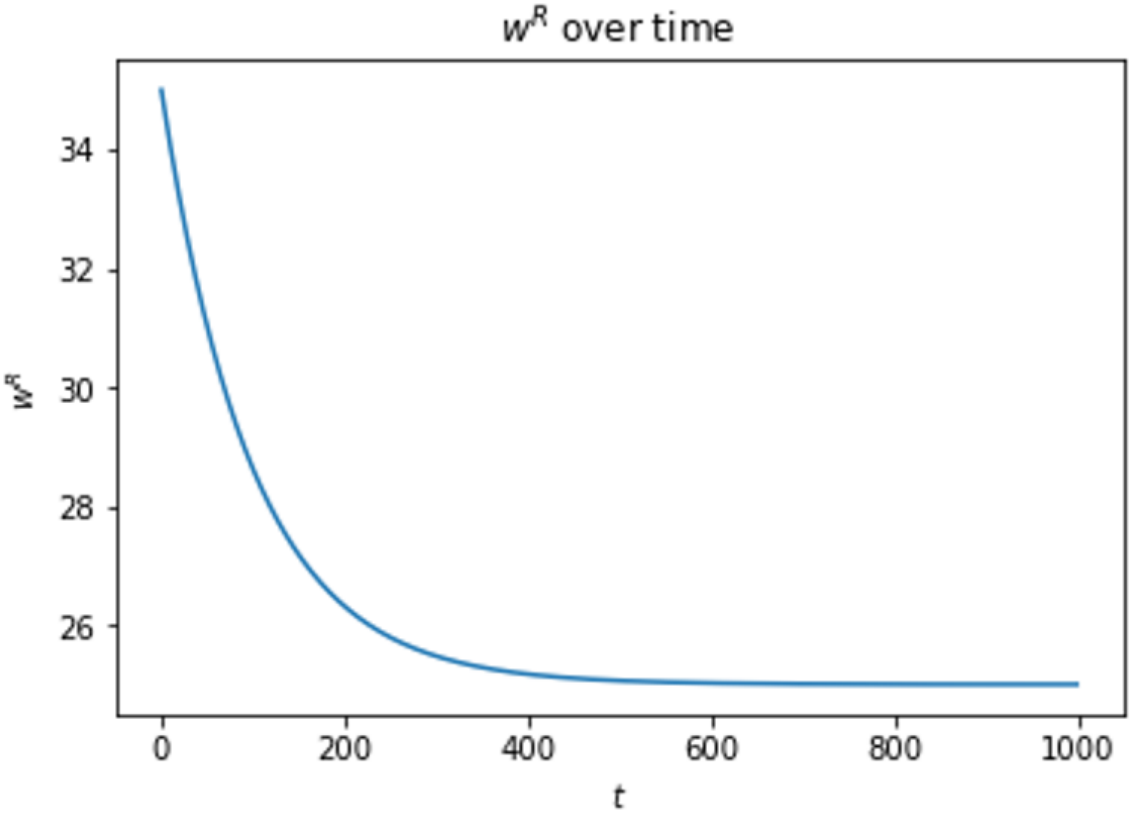
The world is divided into the thermostat T and room R. The dynamics of the system are described by the following equations, in which d is the "desired" temperature, and k represents the strength of the thermostat's ability to change the temperature of the room.

$$w_t^T = w_{t-1}^R - d$$

$$w_t^R = w_{t-1}^R - k \times w_{t-1}^T$$

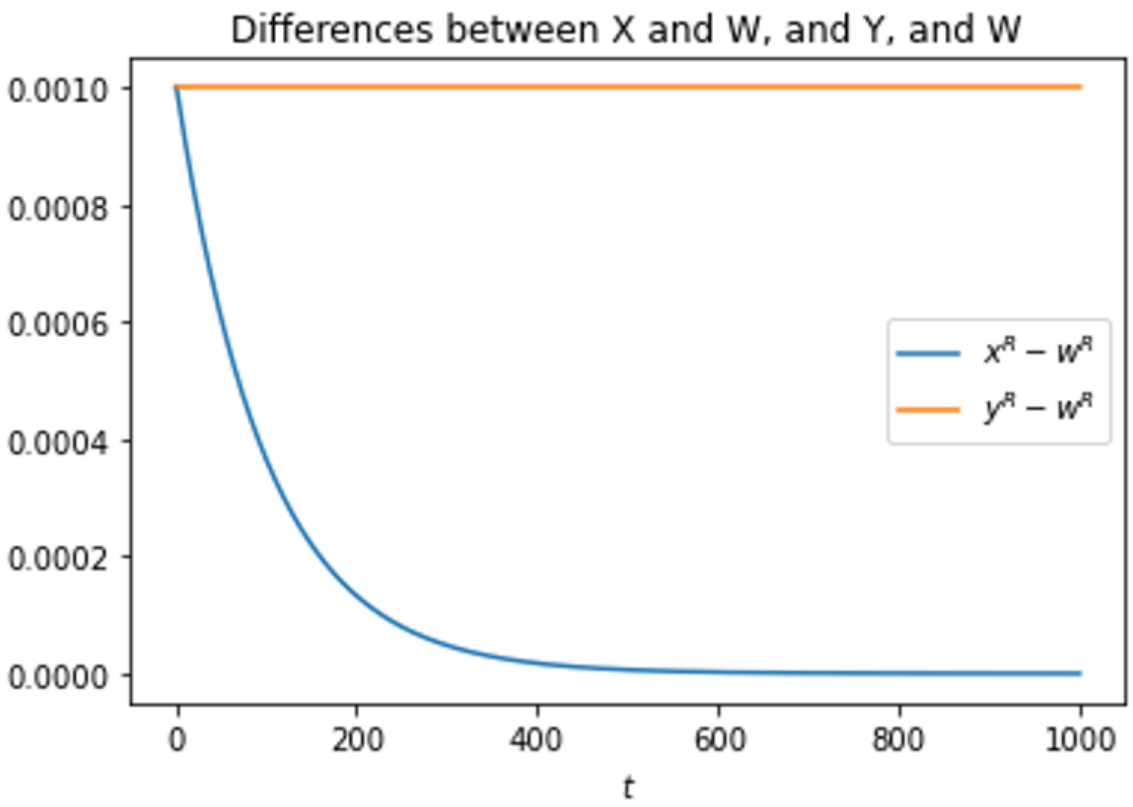
Choosing $w_0^T = 10$, $w_0^R = 35$, k = 0.01, d = 25 and with 1000 timesteps, we can plot the course of W like this:


w^T over time
10
8
6
4
2
0
w^T
0
200
400
600
800
1000
t

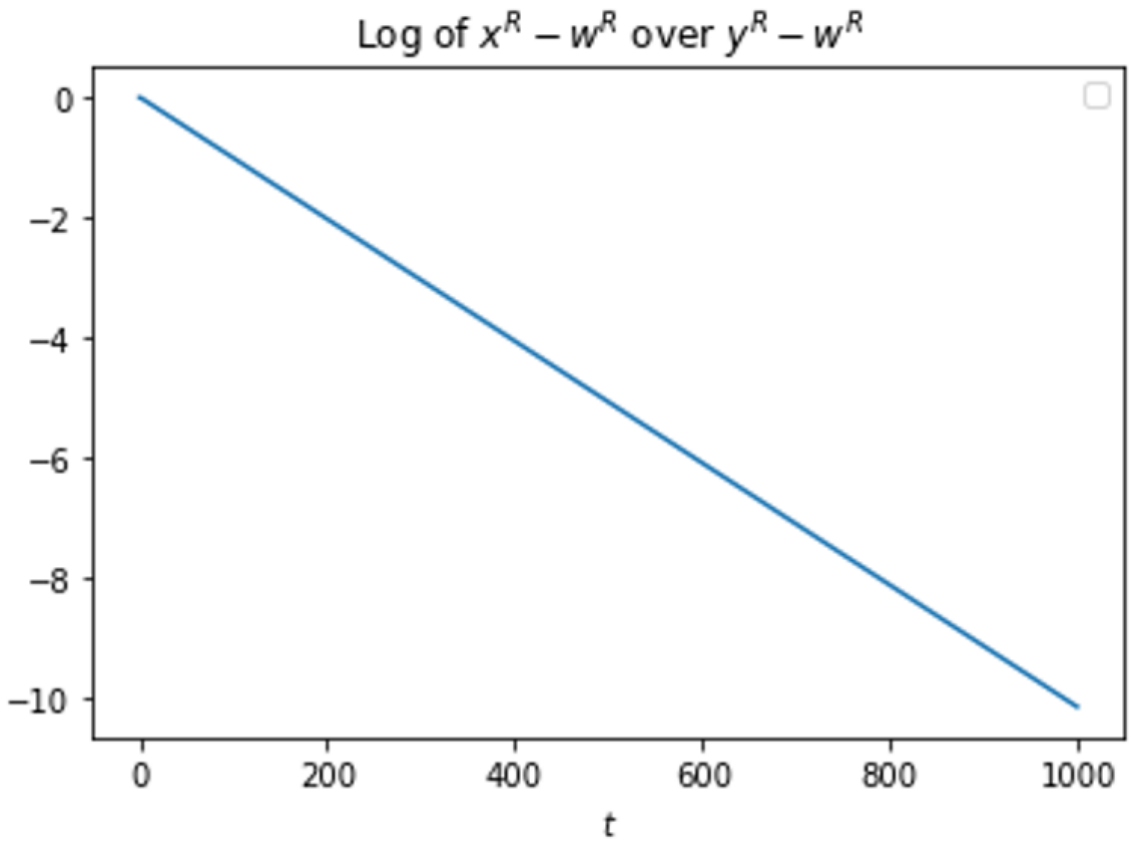


As expected, w^R approaches 25, and w^R approaches 25. If we choose to alter w_0^R by $\delta = 0.001$, we can calculate x_t^R, x_t^T, and y_t^R values. There's not much point plotting these since they're pretty much identical.

What we can plot are the relative values of $x_t^R - w_t^R$ and $y_t^R - w_t^R$:



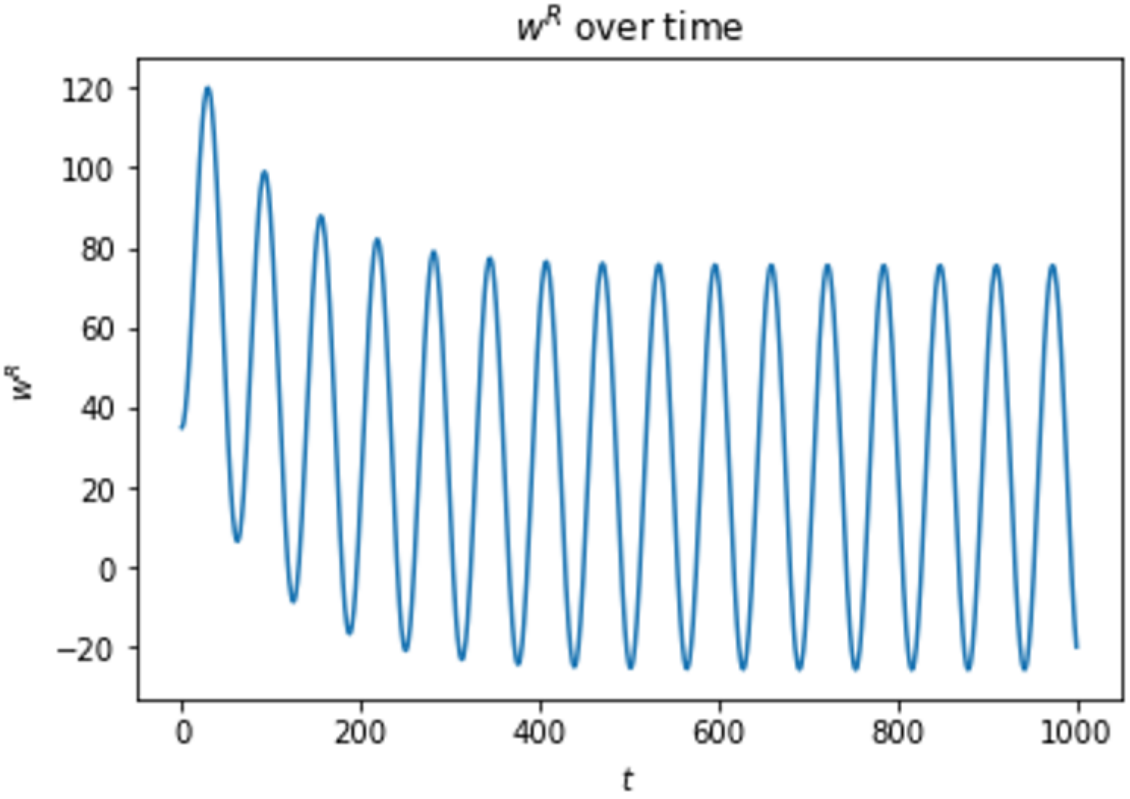
As we can see, x^R converges to the same value as w^R, while y^R remains the same distance away. We can take one final measure, $\log(\frac{x^R}{y^R} = \frac{w^R}{w^R})$ and plot this:

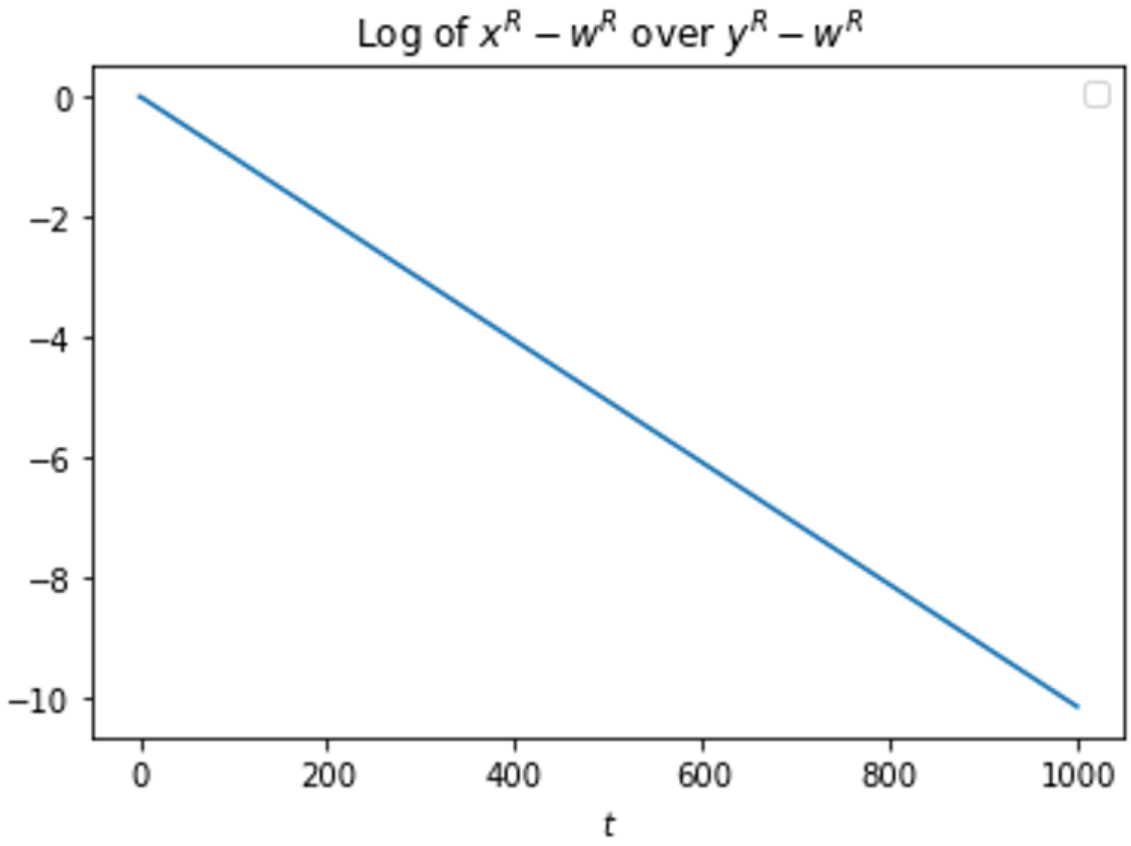


Now this plot has an interesting property in the system we are studying: it doesn't depend on our choice of δ.

This metric can be thought of as a sort of "compressing ability" of T with respect to a change in w_0^R. This optimization is measured with respect to a particular axis of variation.

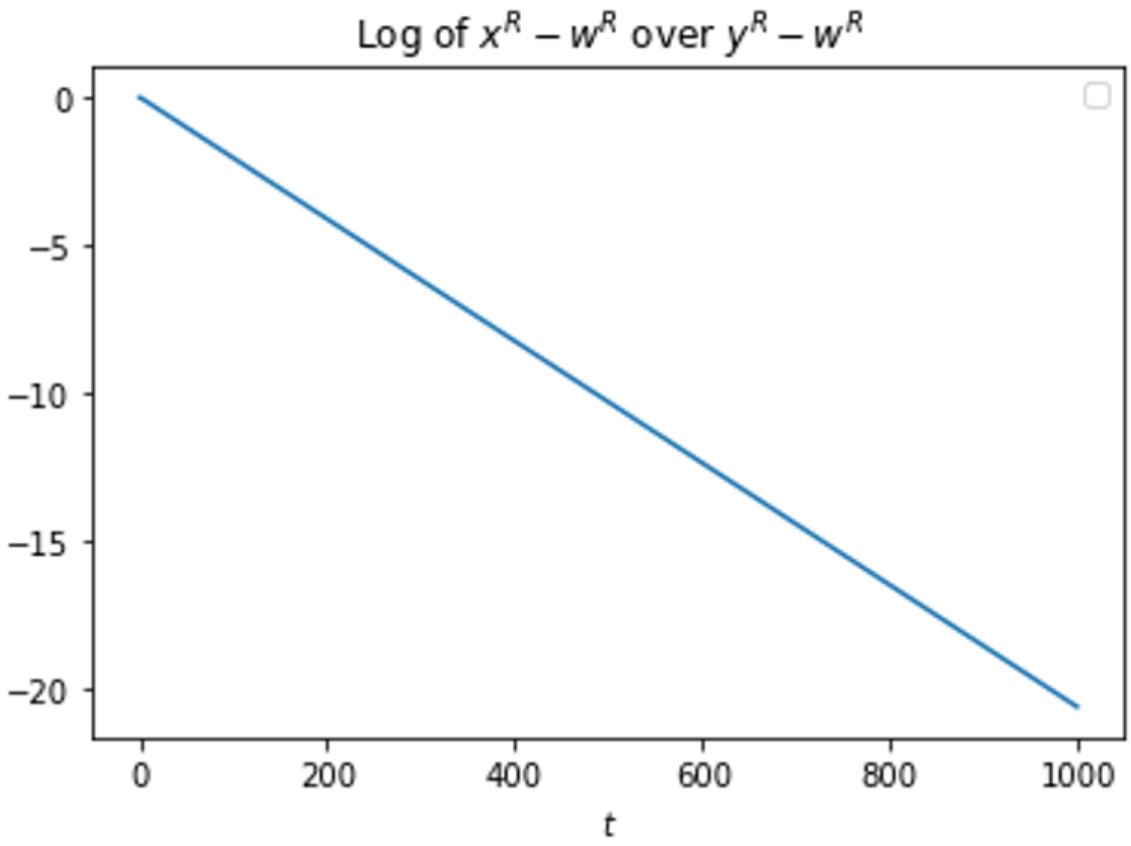
The compressing ability actually doesn't change if we add some noise to the system in the form of a sinusoidal temperature variation. Even if the sinusoidal temperature variation is comically large:


w^R over time
120
100
80
60
40
20
0
−20
w^R
0
200
400
600
800
1000
t



Yes, that is actually the new compressing ability being plotted, I didn't accidentally re-use the same graph! This is almost certainly a result of the fact that our "thermostat" is very mathematically pure.

What it does depend on is k. With a k of 0.02, we get the following:



It's twice as big! This is almost too good to be true! Looks like we might actually be onto something here.

Further Thoughts

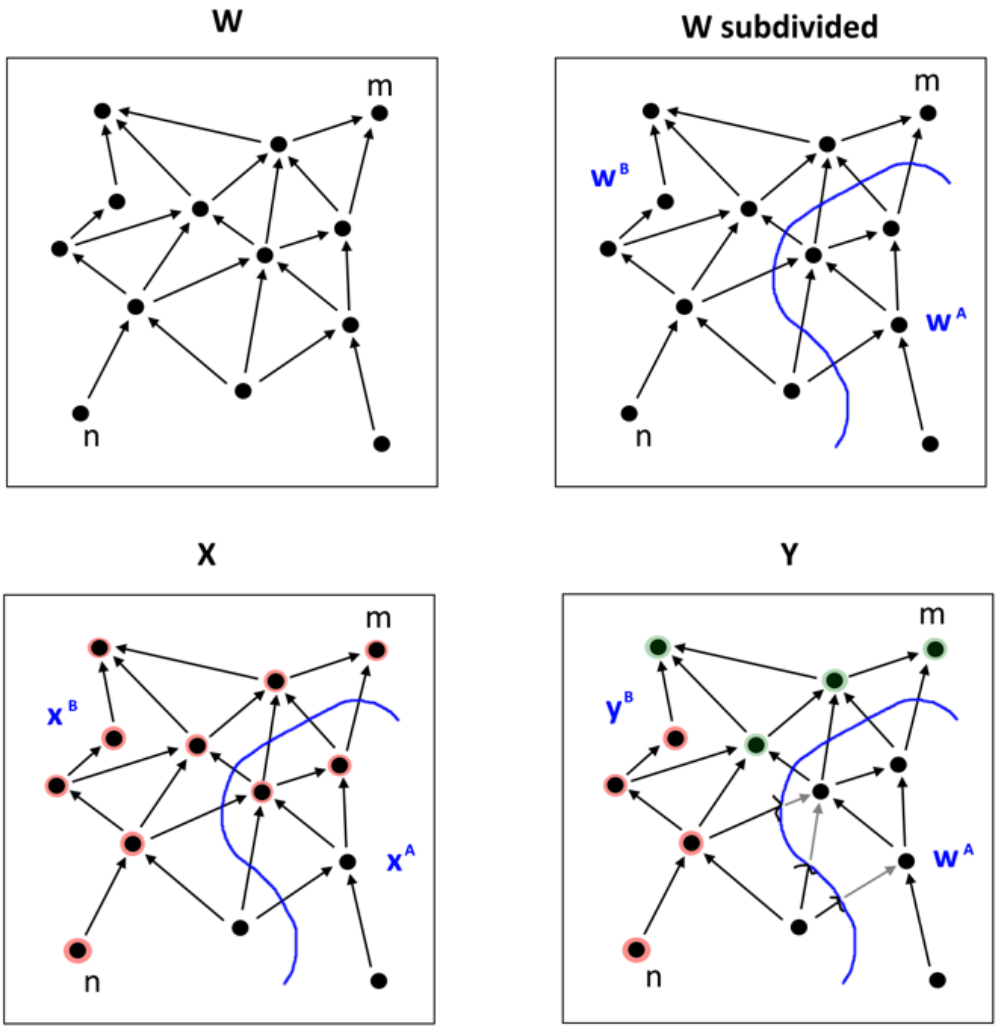
The next step is obvious. We have to measure this in systems which *aren't* optimizers, or are less good optimizers. If it still holds up, we can try and compare the strengths of various optimizers. Then I'd like to explore whether or not this metric is able to find optimization *basins*. Ideally this would involve trying to interpret some neural nets modelling some simple dynamic system.

Defining Optimization in a Deeper Way Part 4

In the last post I introduced a potential measure for optimization, and applied it to a very simple system. In this post I will show how it applies to some more complex systems. My five takeaways so far are:

1. **We can recover an intuitive measure of optimization**
2. **Even around a stable equilibrium,** $Op(A;n,m)$ **can be negative**
3. **Our measures throw up issues in some cases**
4. **Our measures are very messy in chaotic environments**
5. Op **seems to be defined even in chaotic systems**

It's good to be precise with our language, so let's be precise. Remember our model system which looks like this:



In this network, each node is represented by a real number. We'll use superscript notation to notate the value of a node: w^n is the value of node n in the world W.

The heart of this is a quantity I'll call Comp, which is:

$$Comp(A; n, m) = \lim_{x^m \to w^n} [\,]$$

Which is equivalent to.

$$Comp(A : n, m) = \left| \frac{\partial s^m}{\partial s^n} \right|_{A \text{ varies}} / \left| \frac{\partial s^m}{\partial s^n} \right|_{A \text{ constant}}$$

(s^n is the generic version of w^n, x^n, y^n)

Our current measure for optimization is the following value:

$$Op(A; n, m) = \lim_{x^m \to w^n} [-\log |Comp(A : n, m)|]$$

Op is positive when the nodes in A are doing something optimizer-ish towards the node m. This corresponds when Comp is < 1. We can understand this as when A is allowed to vary with respect to changes in s^n, the change that propagates forwards to s^m is smaller.

Op is negative when the nodes in A are doing something like "amplification" of the variance in m. Specifically, we refer to A optimizing m with respect to n around the specific trajectory W, by an amount of nats equal to Op(A; n, m). We'll investigate this measure in a few different systems.

A Better Thermostat Model

Our old thermostat was not a particularly good model of a thermostat. Realistically a thermostat cannot apply infinite heating or cooling to a system. For a better model let's consider the function

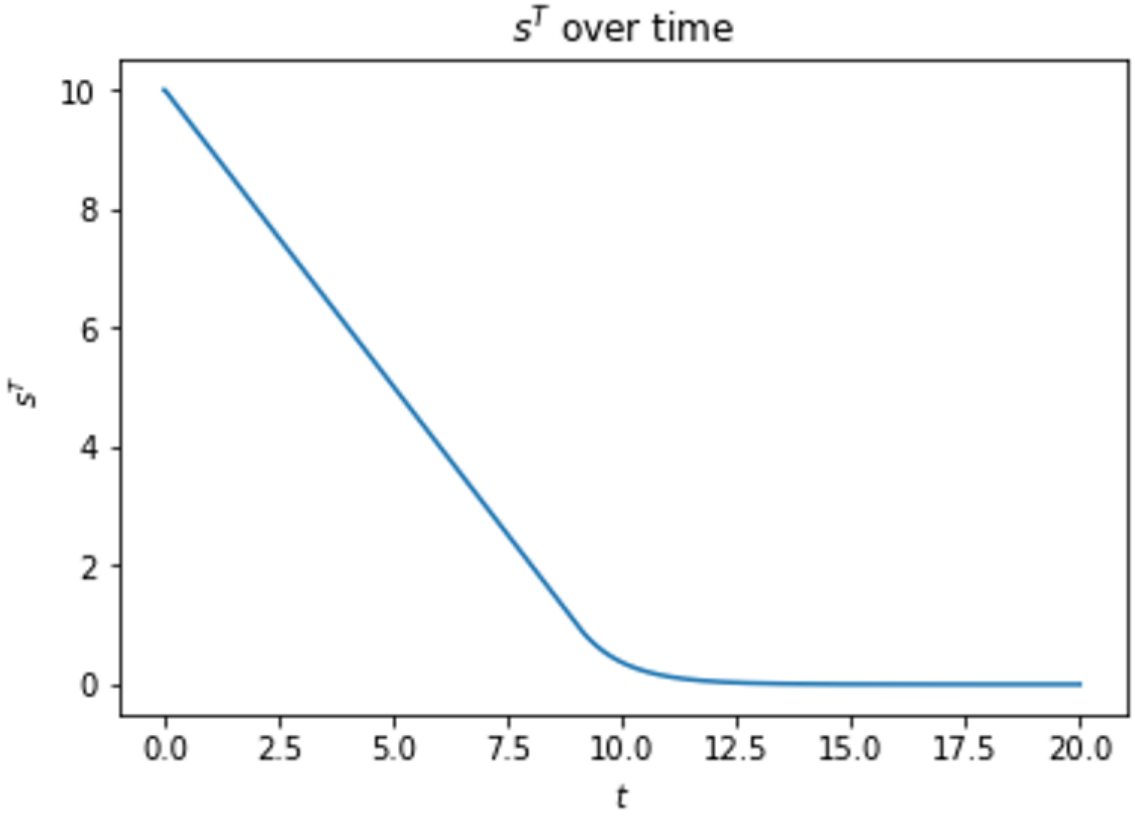
$$Therm(\theta, p; s^T) \begin{cases} p & \theta \leq s^T \\ \theta\, s^T & -\theta < s^T < \theta \\ -p & s^T \leq -\theta \end{cases}$$

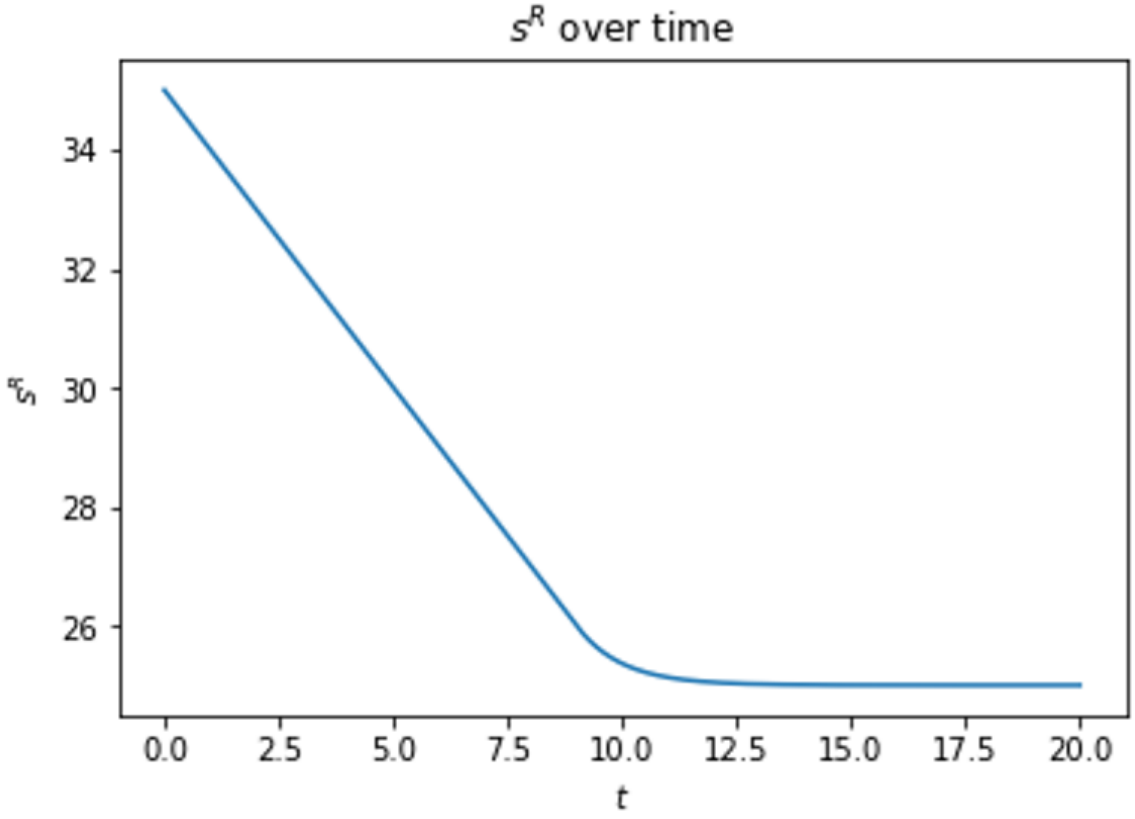
Now imagine we redefine our continuous thermostat like this:

$$s^T_{t+\delta t} = s^R_t - d$$

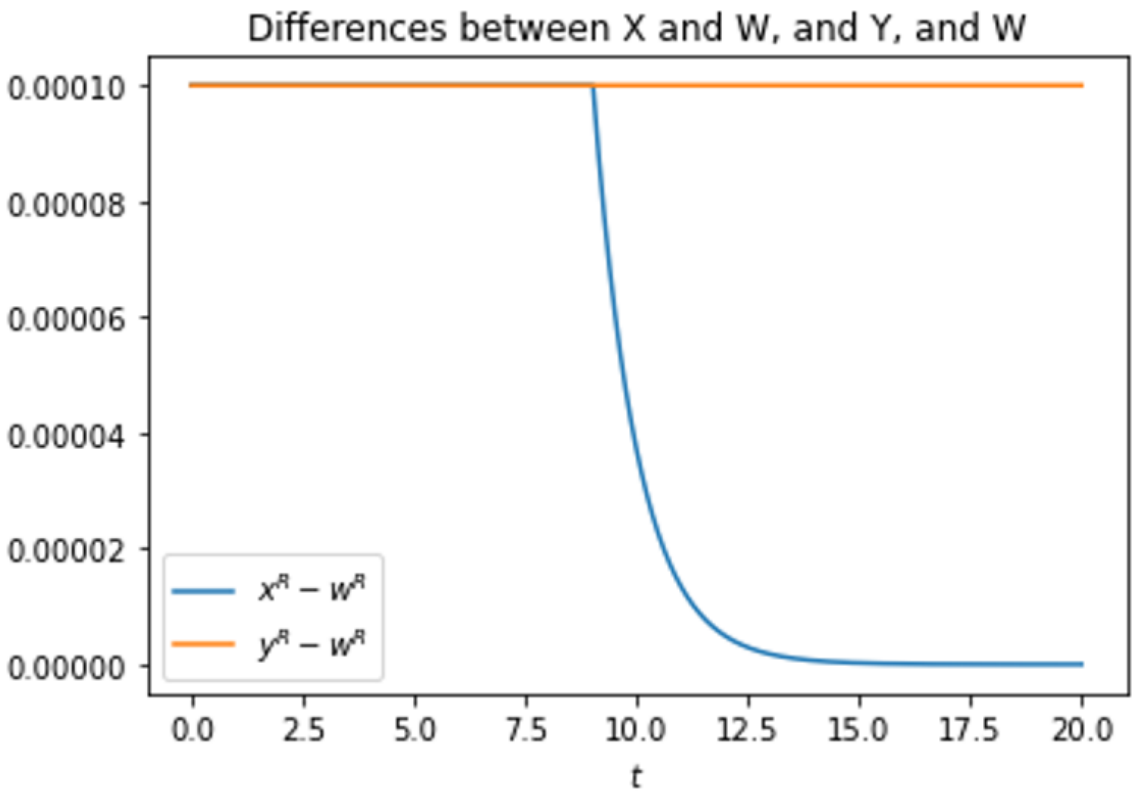
$$s^R_{t+\delta t} = s^R_t - \delta t \times Therm(\theta, p; s^T_t)$$

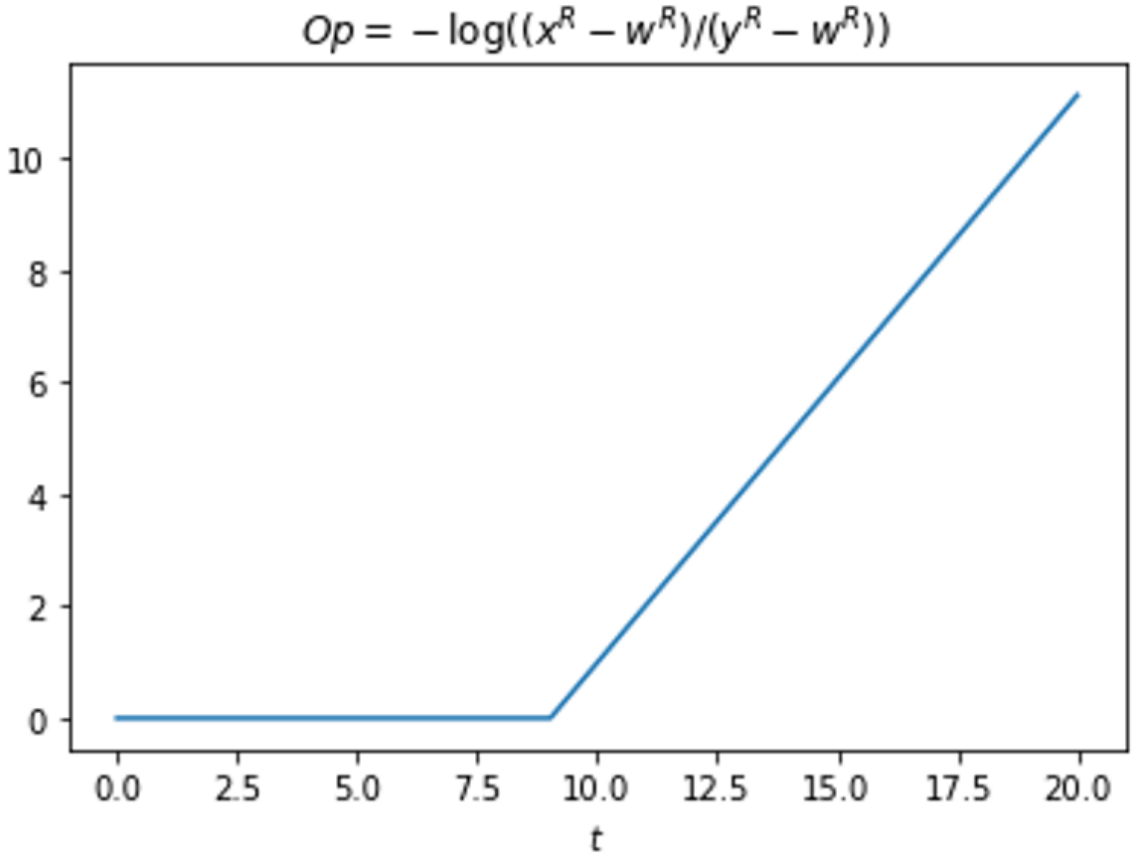
Within the narrow "basin" of $-\theta \leq s^T \leq \theta$, it behaves like before. But outside the change in temperature over time is constant. This looks like the following:


s^T over time
10
8
6
4
2
0
s^T
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
t

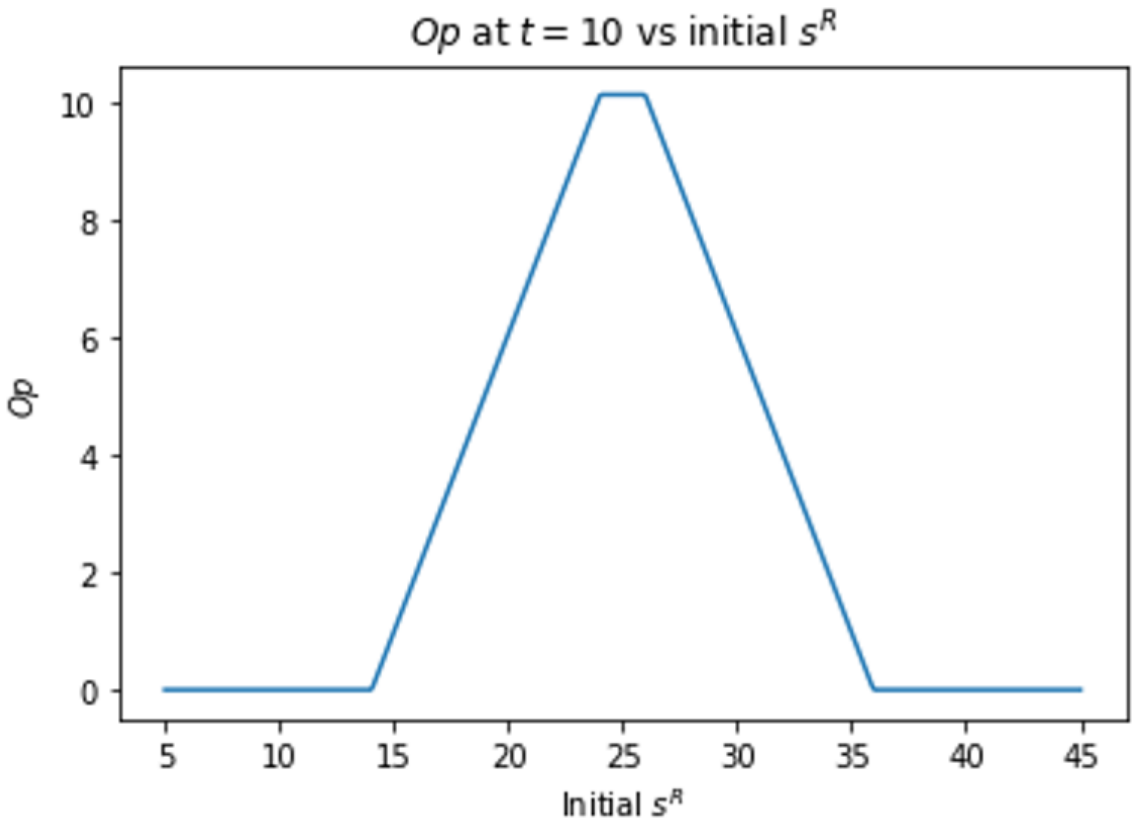


When we look at our optimizing measure, we can see that in while s^R remains in the linear decreasing region, Op = 0. It only increases when s^R reaches the exponentially decreasing region.


Differences between X and W, and Y, and W
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
$x^R - w^R$
$y^R - w^R$
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
t



Now we might want to ask ourselves another question, for what values of s_0^R is $Op(T; s_0^R, s_t^R)$ positive for a given value of t, say t = 10? Let's set p = 1, θ = 1 and the initial $s_0^T = 0$. The graph of this looks like the following:

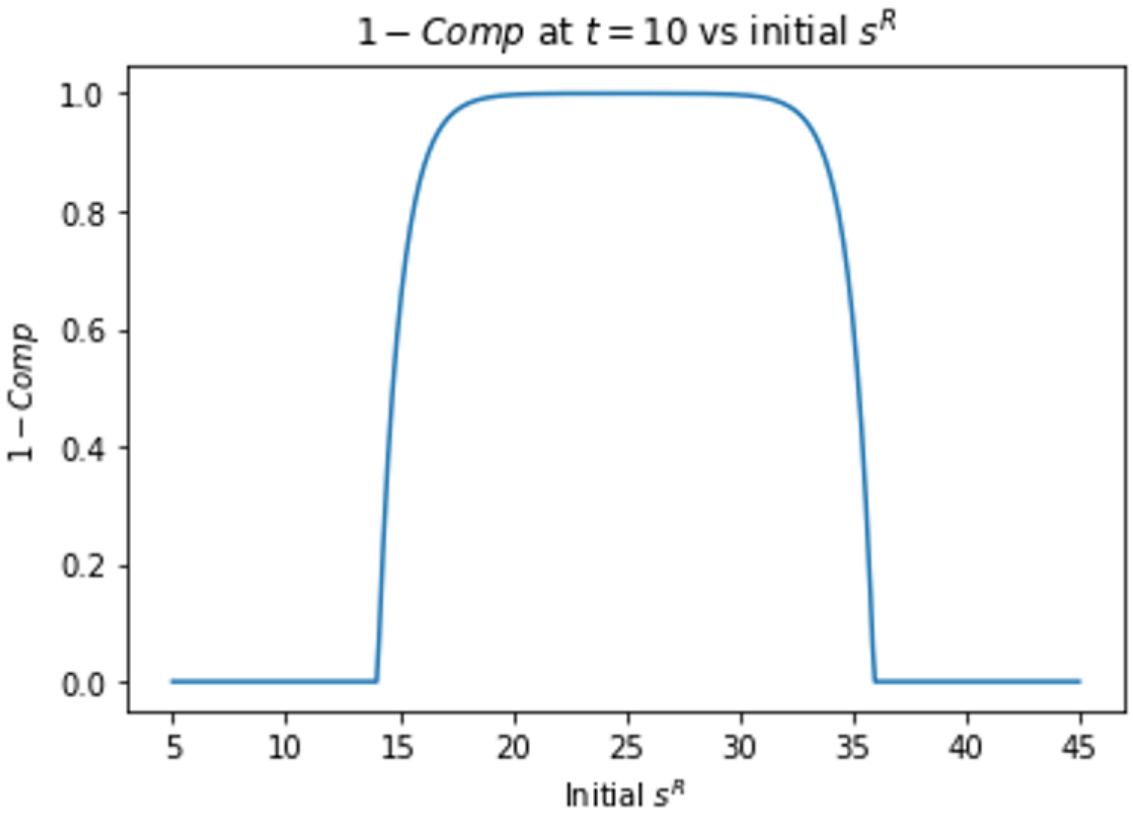


Every initial s^R which has a trajectory which leads into the "optimizing region" between the temperatures of 24 and 26 is optimized a bit. The maximum Op values are trajectories which start in this region.

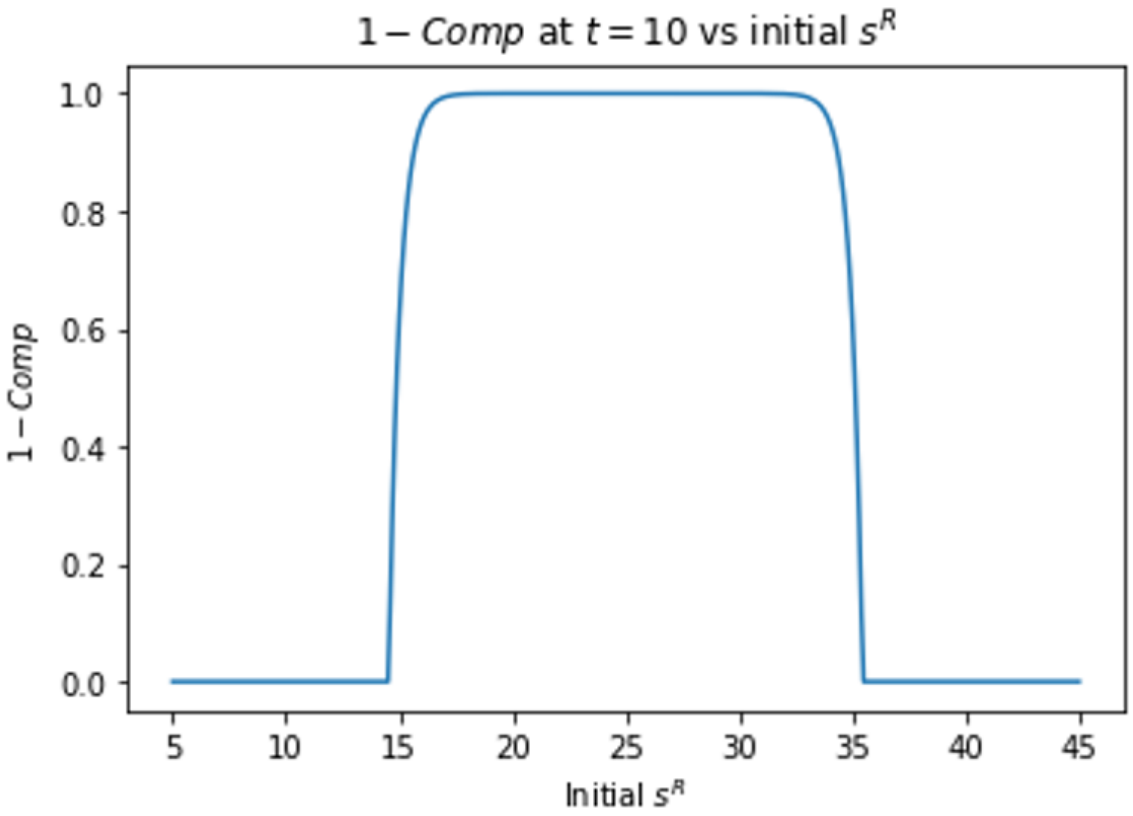
Point 1: We can Recover an Intuitive Measure of Optimization

What we might want to do is measure the "amount" of optimization in this region, between the points $s_0^R = 5$ and $s_0^R = 45$, with respect to s_{10}^R. If we choose this measure to be the integral of $1 - \text{Comp}$, we get some nice properties.

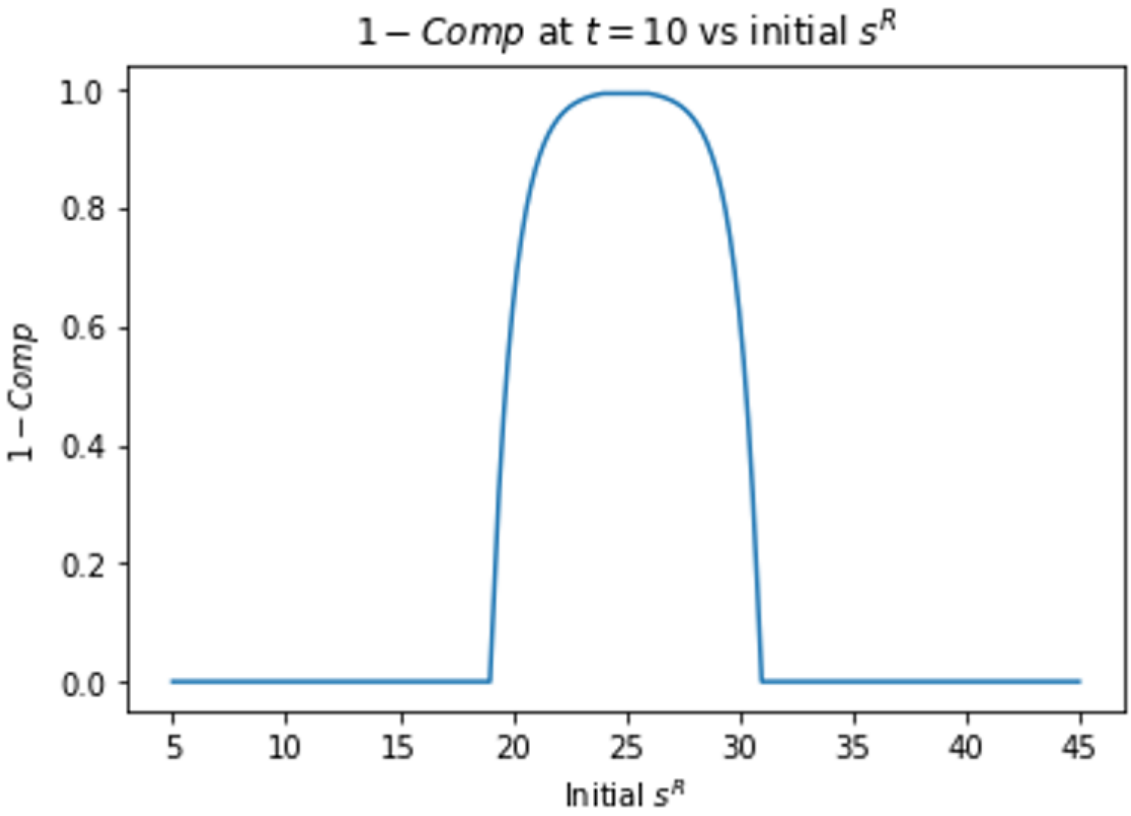
It (almost) no longer depends on θ, but depends linearly on p.



θ = 1, p = 1 gives an integral of 19.961



θ = 0.5, p = 1 gives an integral of 19.611



θ = 1, p = 0.5 gives an integral of 9.980

As θ → 0, our integral remains (pretty much) the same. This is good because it means we can assign some "opimizing power" to a thermostat which acts in the "standard" way, i.e. applying a change of +p each time unit to the temperature if it's below the set point, and a change of −p each time unit if it's above the set point. And it's no coincidence that that power is equal to 2pt.

Let's take a step back to consider what we've done here. If we consider the following differential equation:

$$\frac{dT}{dt} = \begin{cases} -p & T > T_{set} \\ 0 & T = T_{set} \\ p & T < T_{set} \end{cases}$$

It certainly *looks* like T values are being compressed about T_{set} by 2p per time unit, but that requires us to do a somewhat awkward manoeuvre: We have to equivocate our metric of the space of T at t = 10 with our metric of the space of T at t = 0. For temperatures this can be done in a natural way, but this doesn't necessarily extend to other systems. It also doesn't

stack up well with systems which *naturally* compress themselves along some sort of axis, for example water going into a plughole.

We've managed to recreate this using what I consider to be a much more flexible, well-defined, and natural measure. This is a good sign for our measure.

The Lorenz System

This is a famed system defined by the differential equations:

$$\frac{da}{dt} = \sigma (a - b)$$

$$\frac{db}{dt} = a (\rho - c) - b$$

$$\frac{dc}{dt} = a\, b - \beta\, c$$

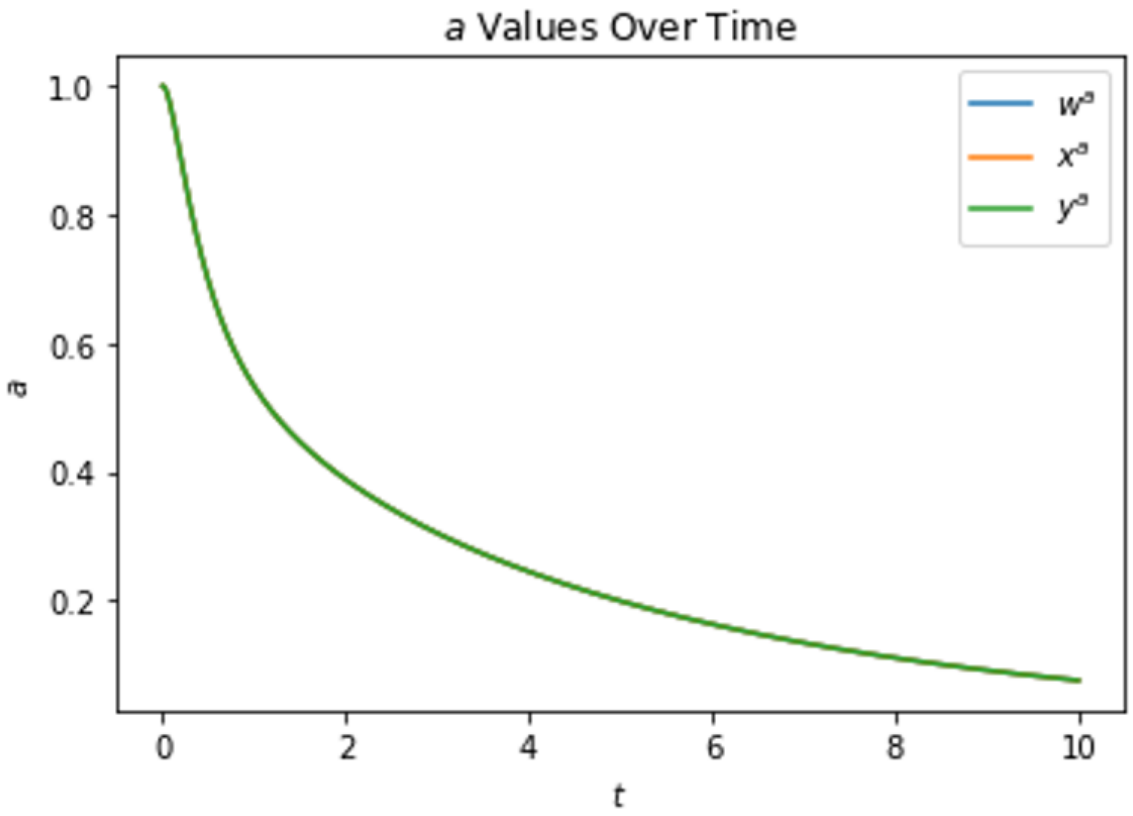
(I have made the notational change from the "standard" $x, y, z \to a, b, c$ in order to avoid collision with my own notation)

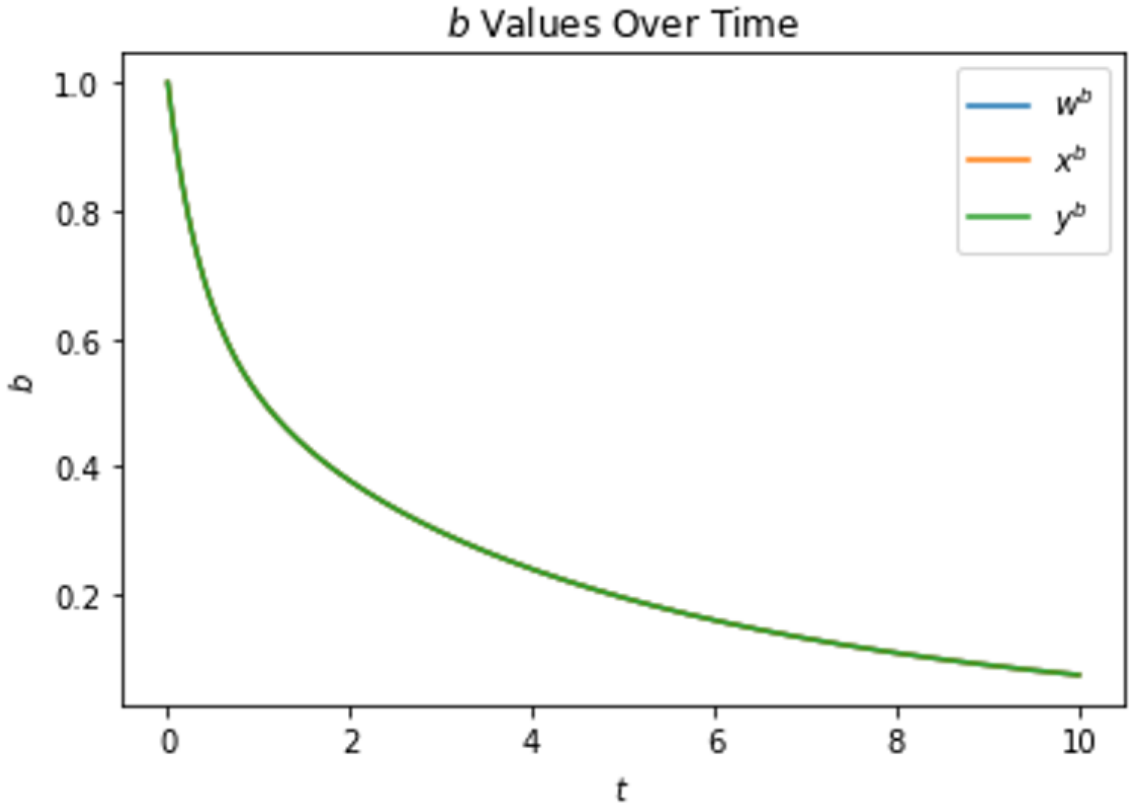
Which can fairly easily and relatively accurately be converted to discreet time. We'll keep $\sigma = 10, \beta = \frac{8}{3}$ as constant values. For values of $\rho < 1$ we have a single stable equilibrium point. For values $1 < \rho < 24.74$ we get three stable equilibria, and for values $\rho > 24.74$ we have a chaotic system. We'll investigate the first and third cases.

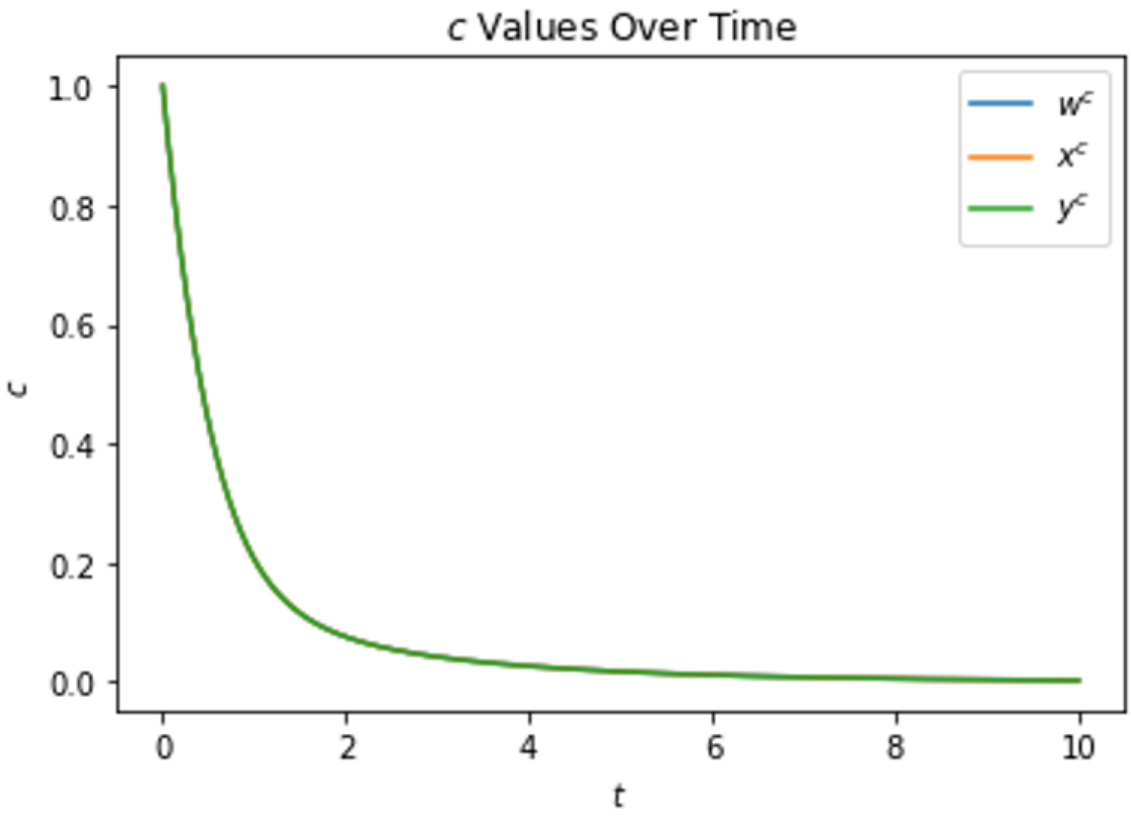
The most natural choices for A are all of any one of the a b or c values. We could also equally validly choose A to be a pair of them, although this might cause some issues. A reasonable choice for n would be the initial value of either of the two a, b, or c which aren't chosen for A.

Point 2: Even Around a Stable Equilibrium, $\mathrm{Op}(A; n, m)$ an be Negative

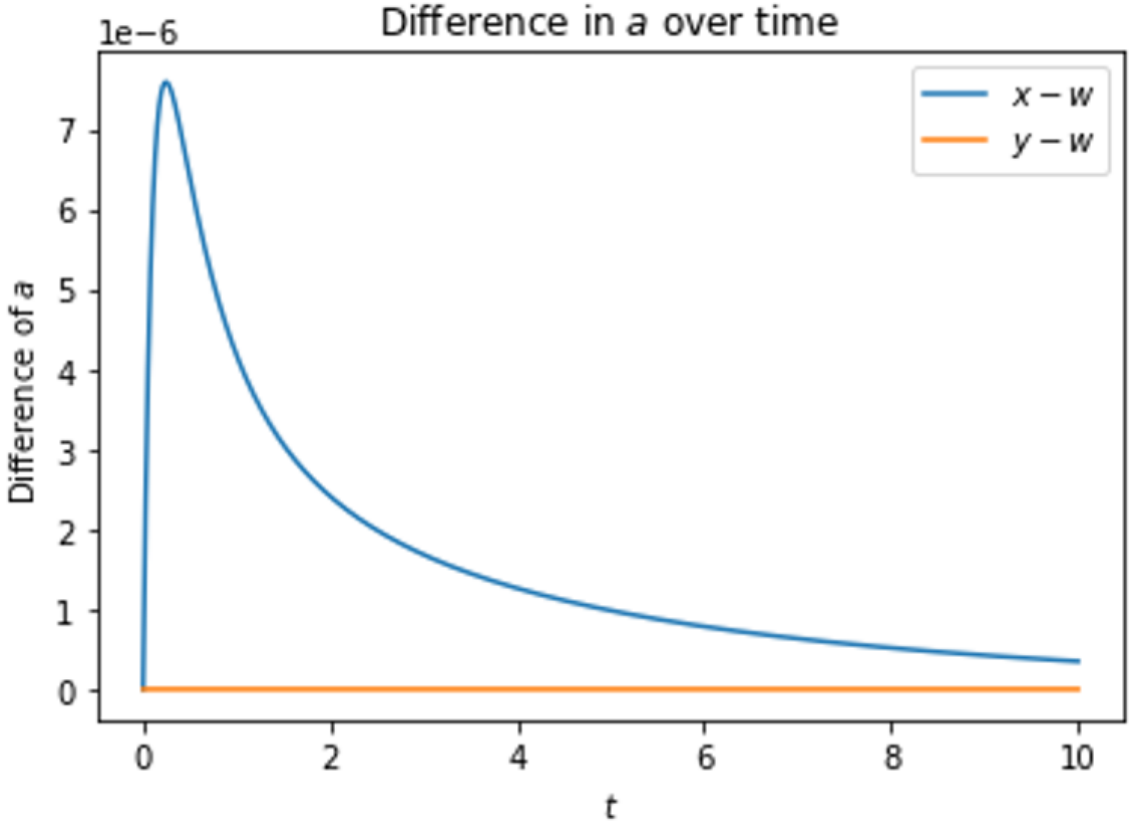
Let's choose $\rho = 0.8$, which means we have a single stable point at $a = 0, b = 0, c = 0$. Here are plots for the choice of a as the set A, and s_0^b as the axis along which to measure optimization. (So we're changing the value of s_0^b and looking at how future values of s_t^b and s_t^c change, depending on whether or not values of s_t^a are allowed to change)

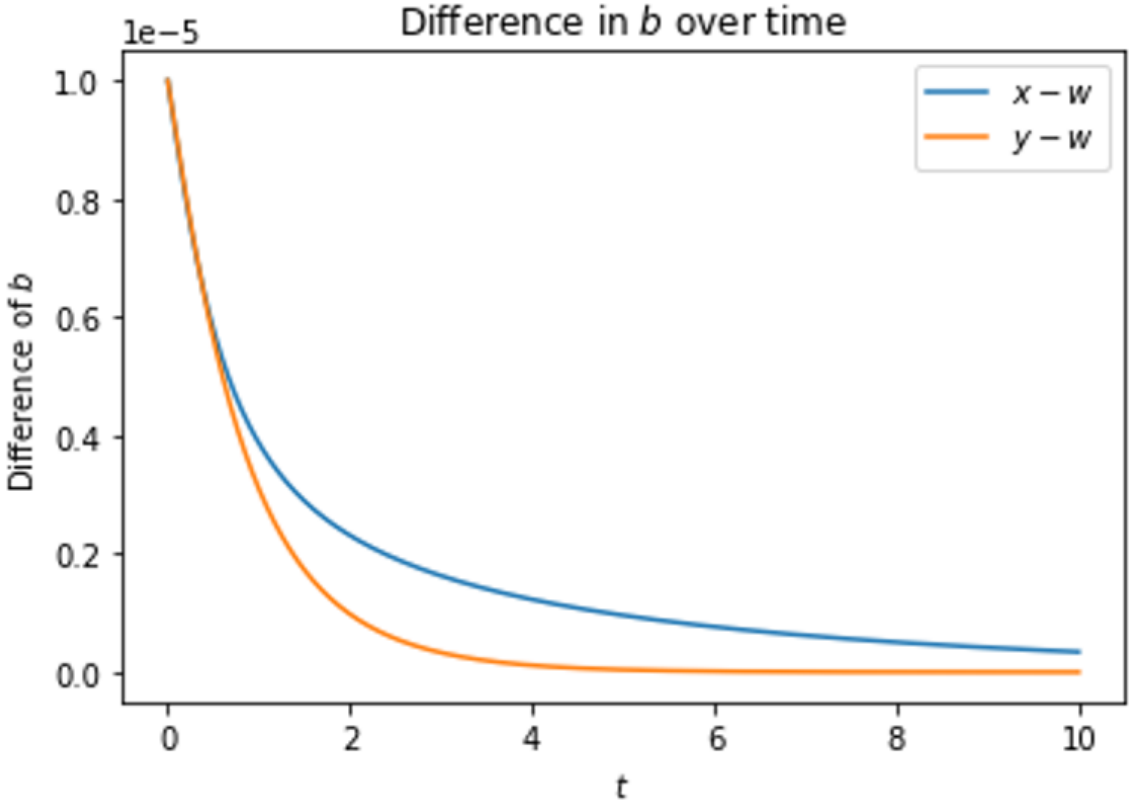

a Values Over Time
1.0
0.8
0.6
0.4
0.2
a
0
2
4
6
8
10
t
w^a
x^a
y^a

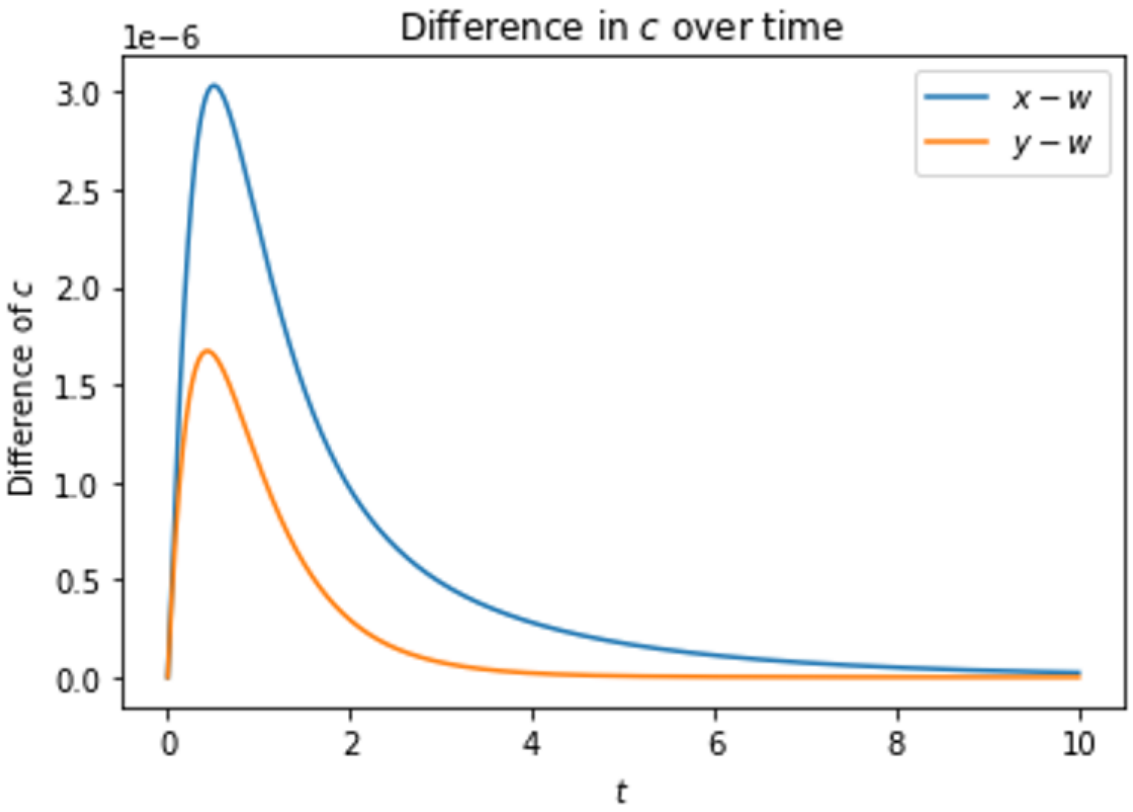

b Values Over Time
1.0
0.8
0.6
0.4
0.2
b
0
2
4
6
8
10
t
w^b
x^b
y^b



Due to my poor matplotlib abilities, those all look like one graph. This indicates that we are not in the chaotic region of the Lorenz system. The variables a, b, and c approach zero in all cases.

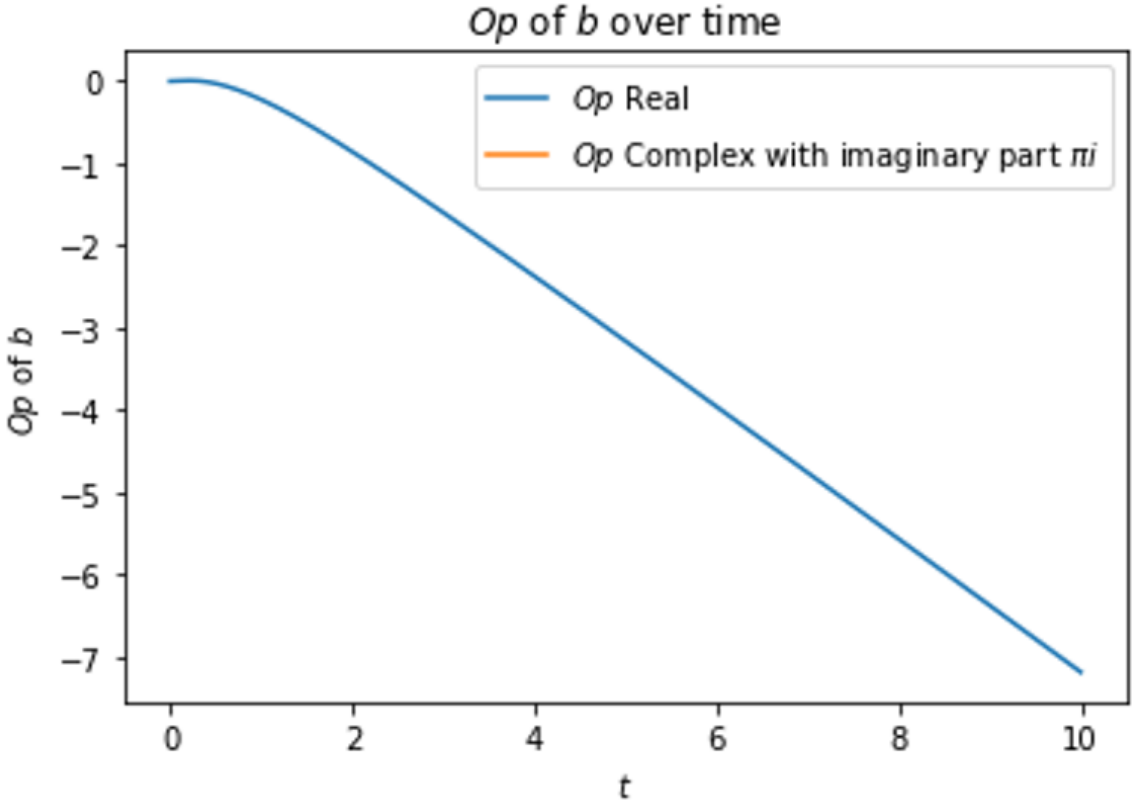

Difference in a over time
1e−6
Difference of a
t
$x - w$
$y - w$
0
1
2
3
4
5
6
7
0
2
4
6
8
10

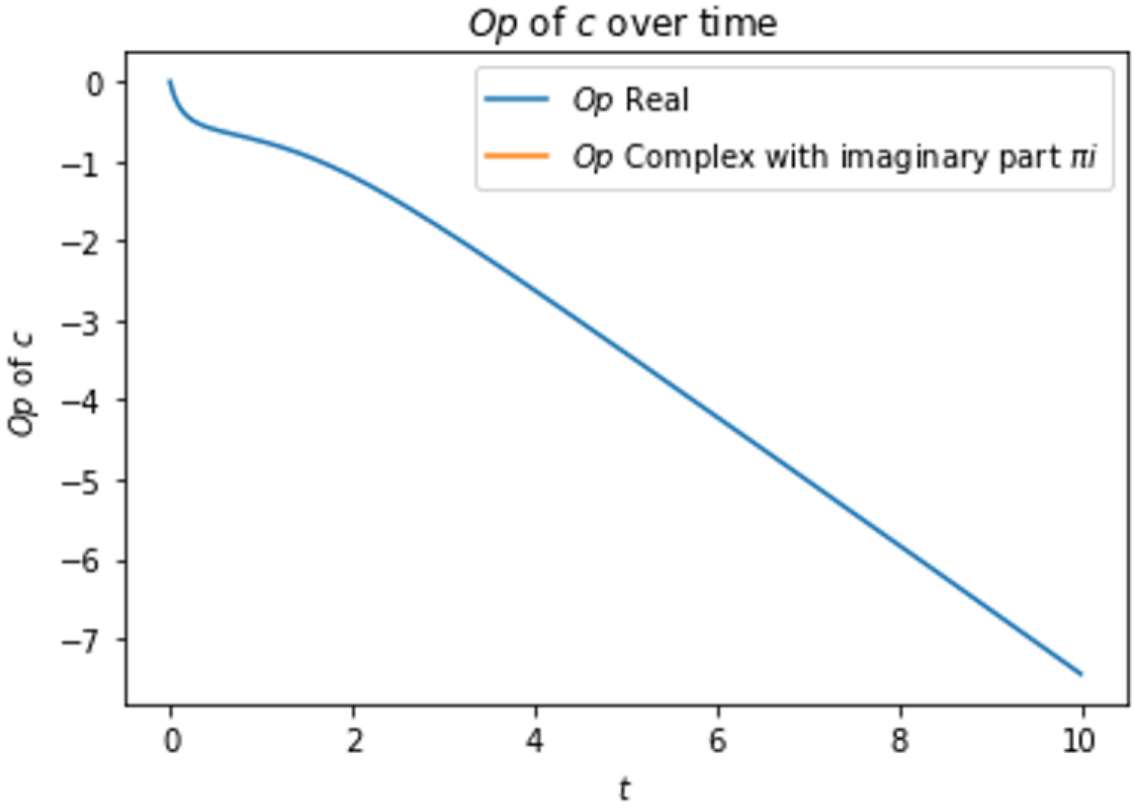

Difference in b over time
1e−5
Difference of b
t
$x - w$
$y - w$
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
10



As we can see, difference $y_t^b - w_t^b$ is *greater* than the difference $x_t^b - w_t^b$. The mathematics of this are difficult to interpret meaningfully, so I'll settle with the idea that changes in a, b, and c in some way compound on one another over time, even as all three approach zero.

When we plot values for Op we get this:

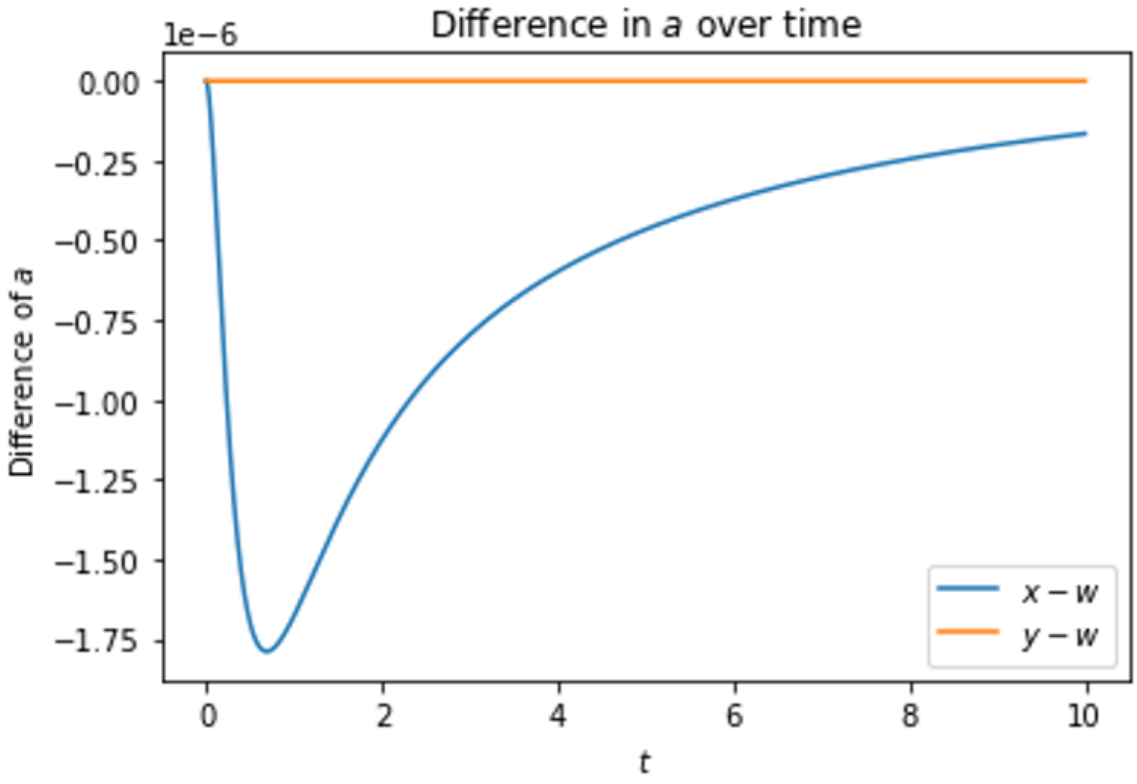

Op of b over time
Op Real
Op Complex with imaginary part πi
Op of b
t
0
−1
−2
−3
−4
−5
−6
−7
0
2
4
6
8
10

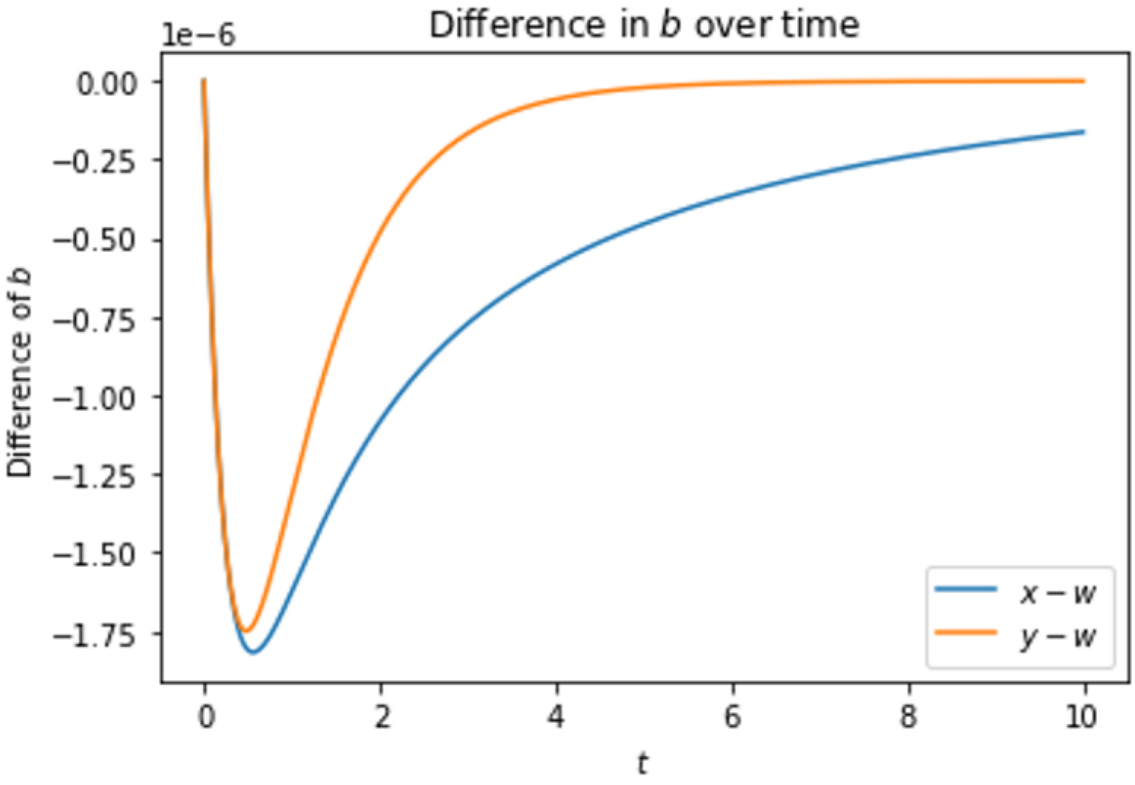

Op of c over time
Op Real
Op Complex with imaginary part πi
Op of c
t
0
−1
−2
−3
−4
−5
−6
−7
0
2
4
6
8
10

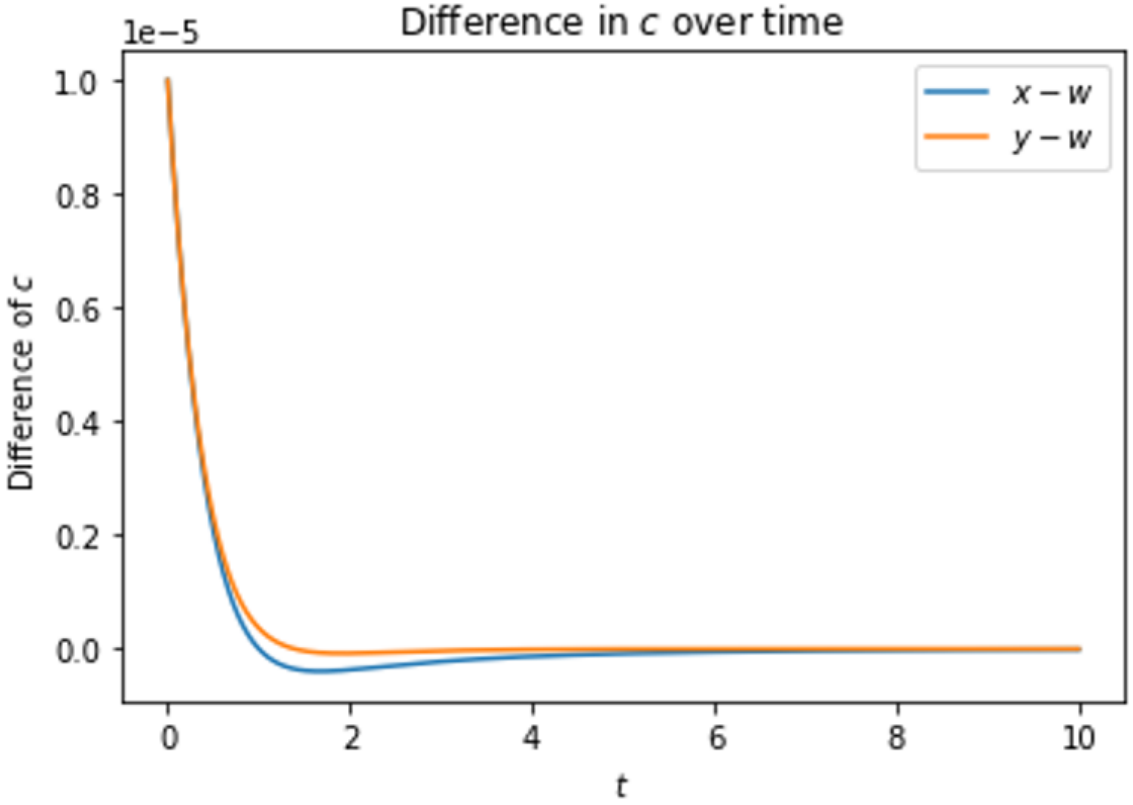
The values for $\mathrm{Op}(a; s_0^b, s^b)$ and $\mathrm{Op}(a; s_0^b, s^c)$ are negative, as expected. This is actually really important! It's important that our measure captures the fact that even though the future is being "compressed" — in the sense that future values of a, b, and c approach zero as $t \to \infty$ — it's not necessarily the case that these variables (which are the only variables in the system) are optimizing each other.

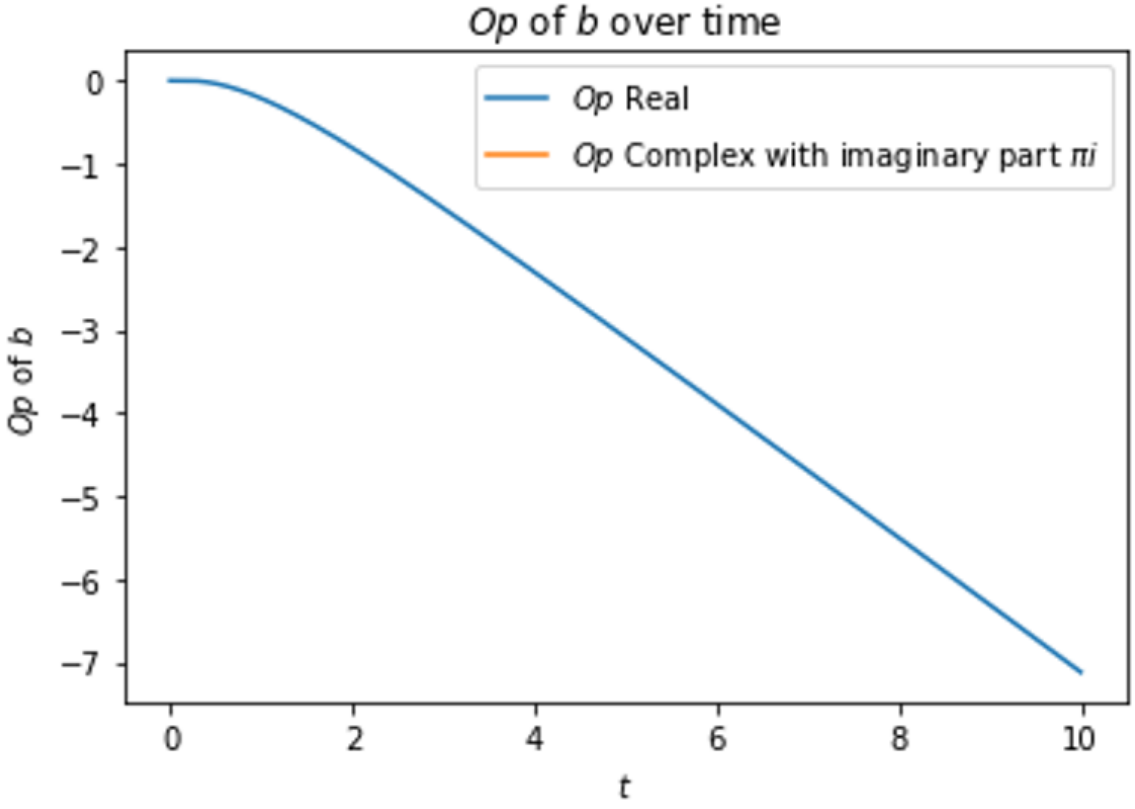
Point 3: Our Measures Throw Up Issues in Some Cases

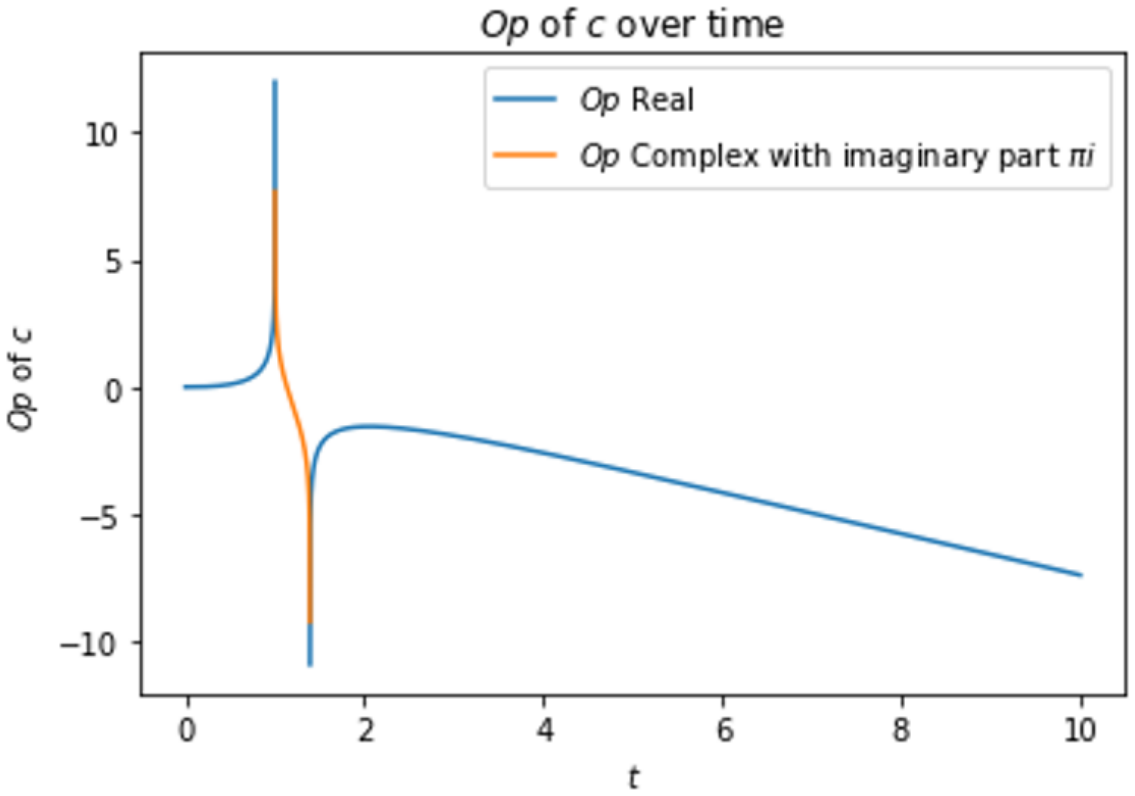
Now what about variation along the axis s_0^c?




Difference in b over time
1e−6
0.00
−0.25
−0.50
−0.75
−1.00
−1.25
−1.50
−1.75
Difference of b
0
2
4
6
8
10
t
$x - w$
$y - w$


Difference in c over time
1e−5
1.0
0.8
0.6
0.4
0.2
0.0
Difference of c
0
2
4
6
8
10
t
$x - w$
$y - w$

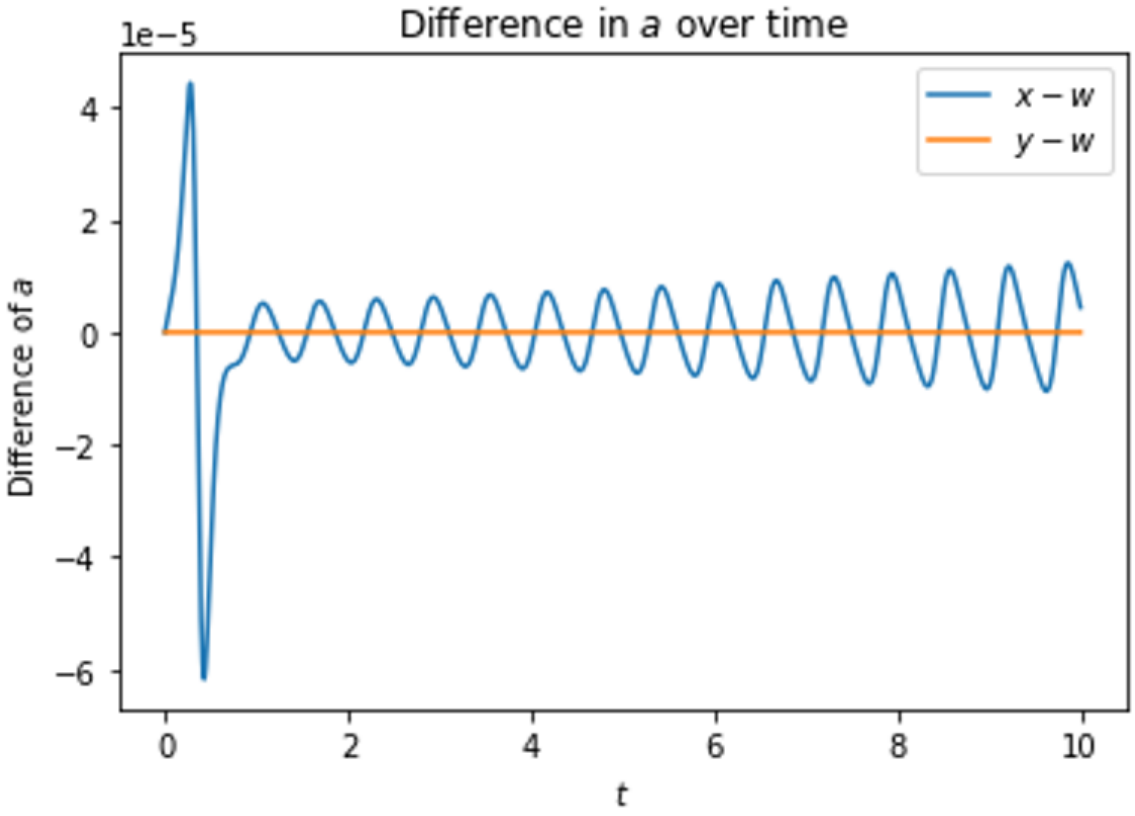

Op of b over time
Op Real
Op Complex with imaginary part πi
Op of b
t
0
−1
−2
−3
−4
−5
−6
−7
0
2
4
6
8
10

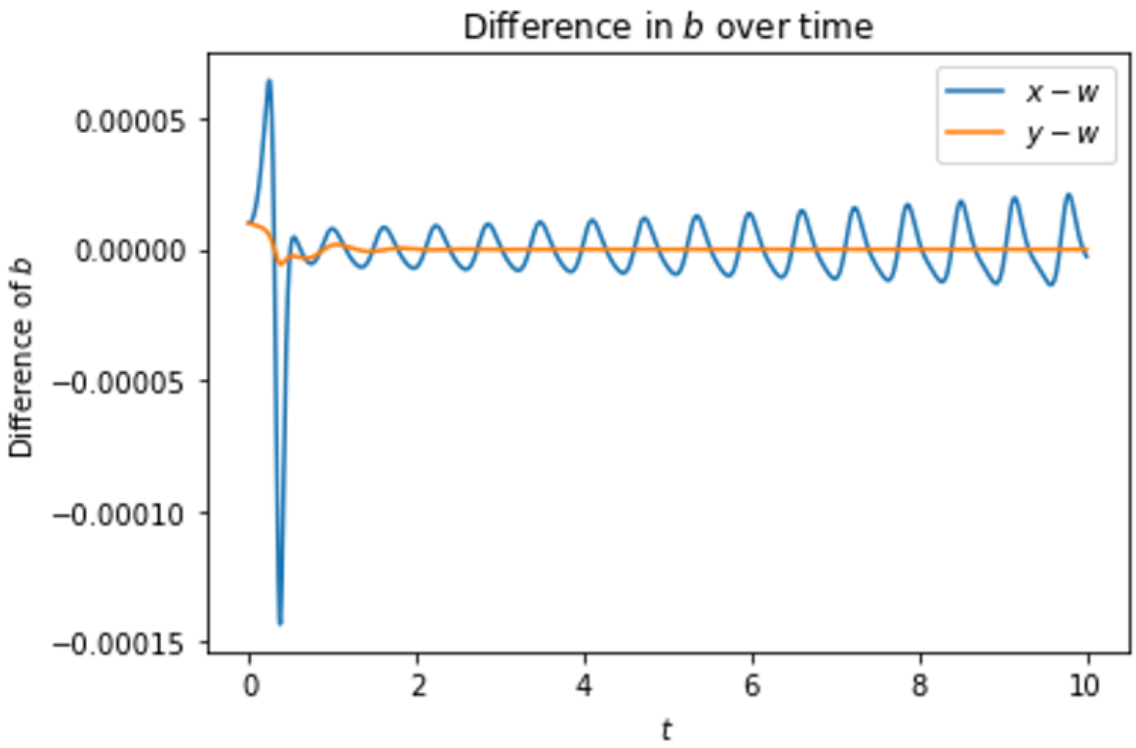

Op of c over time
Op Real
Op Complex with imaginary part πi
Op of c
t
10
5
0
−5
−10
0
2
4
6
8
10

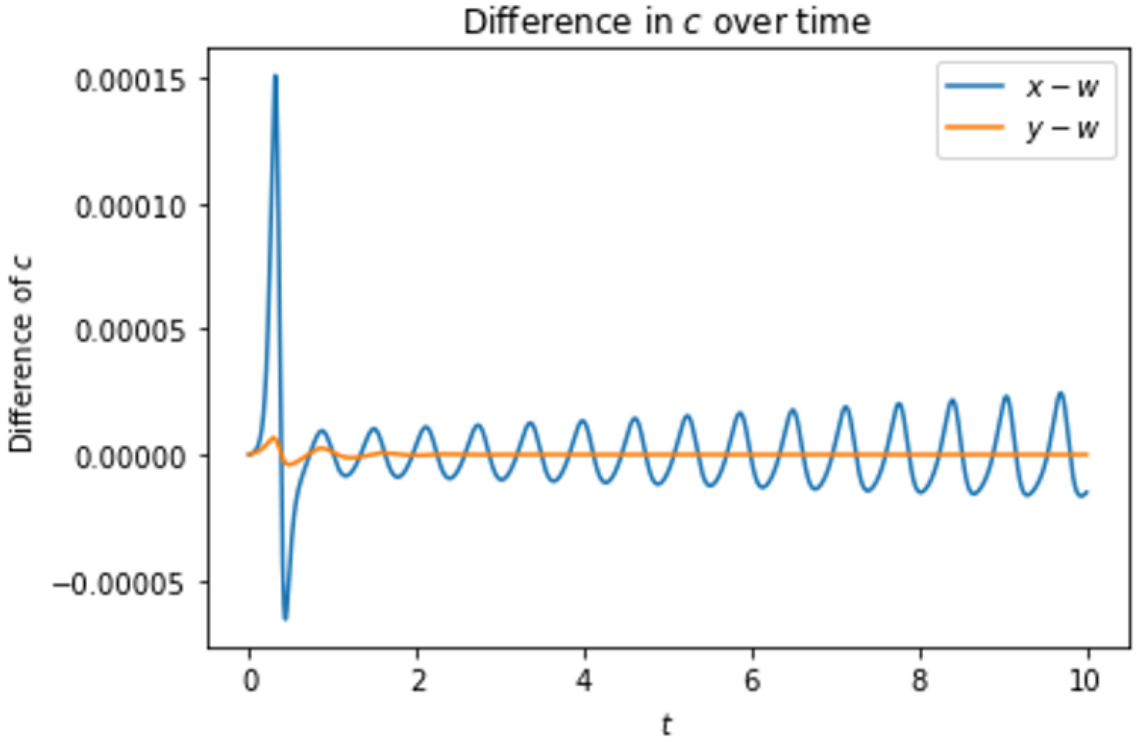
We run into a bit of an issue! For a small chunk of time, the differences $x^c - w^c$ and $y^c - w^c$ have different signs. This causes Op to be complex valued, whoops!

Point 4: Our Measures are Very Messy in Chaotic Environments

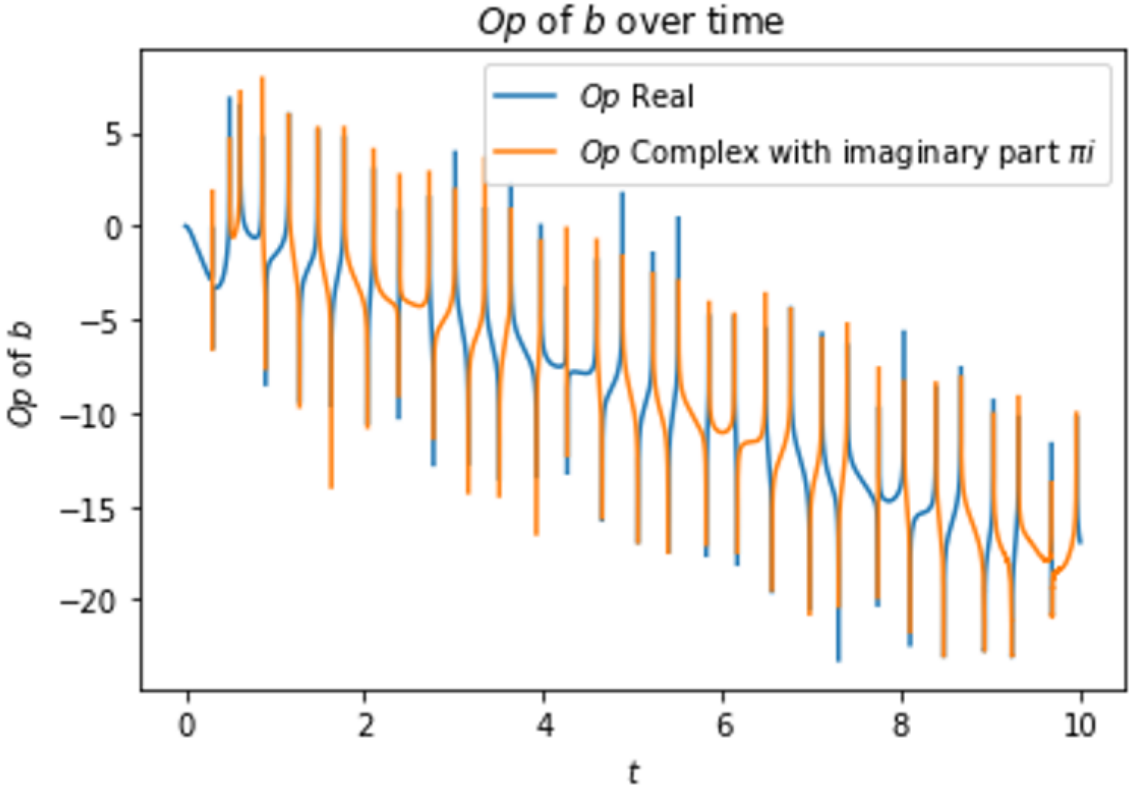
When when choose ρ = 28, it's a different story. Here we are with a as A, s_0^b as the axis of optimization:

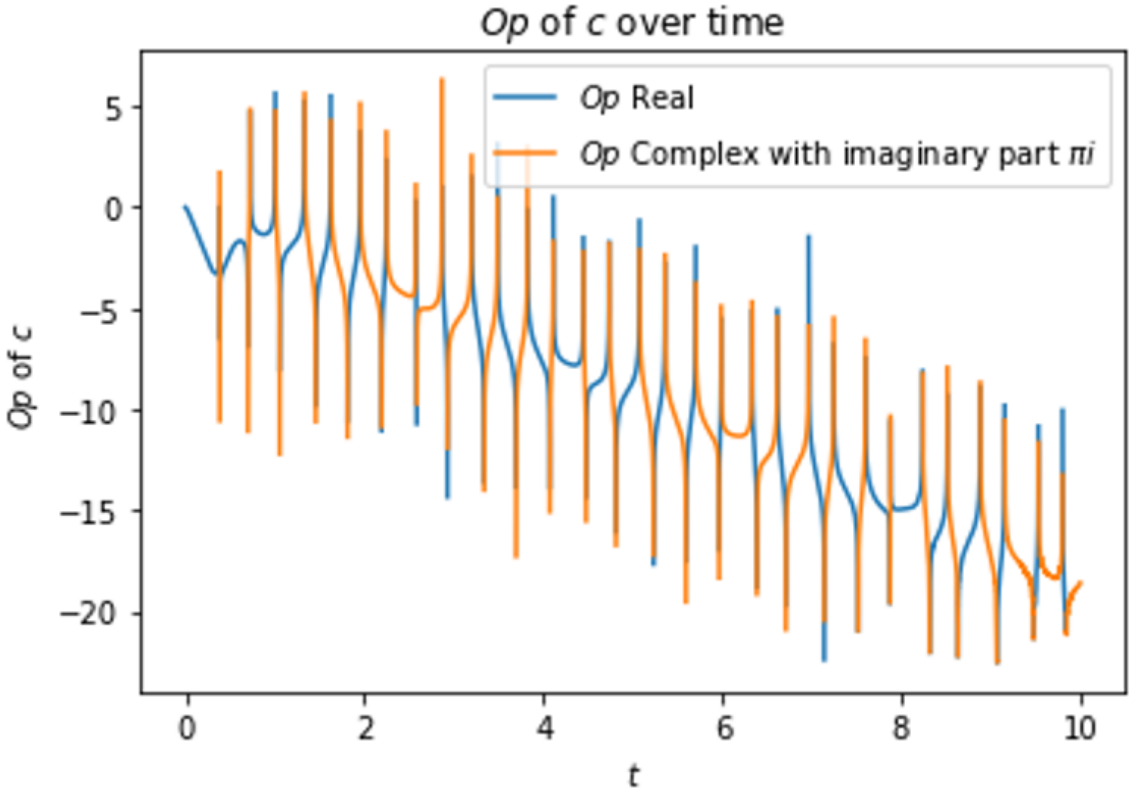






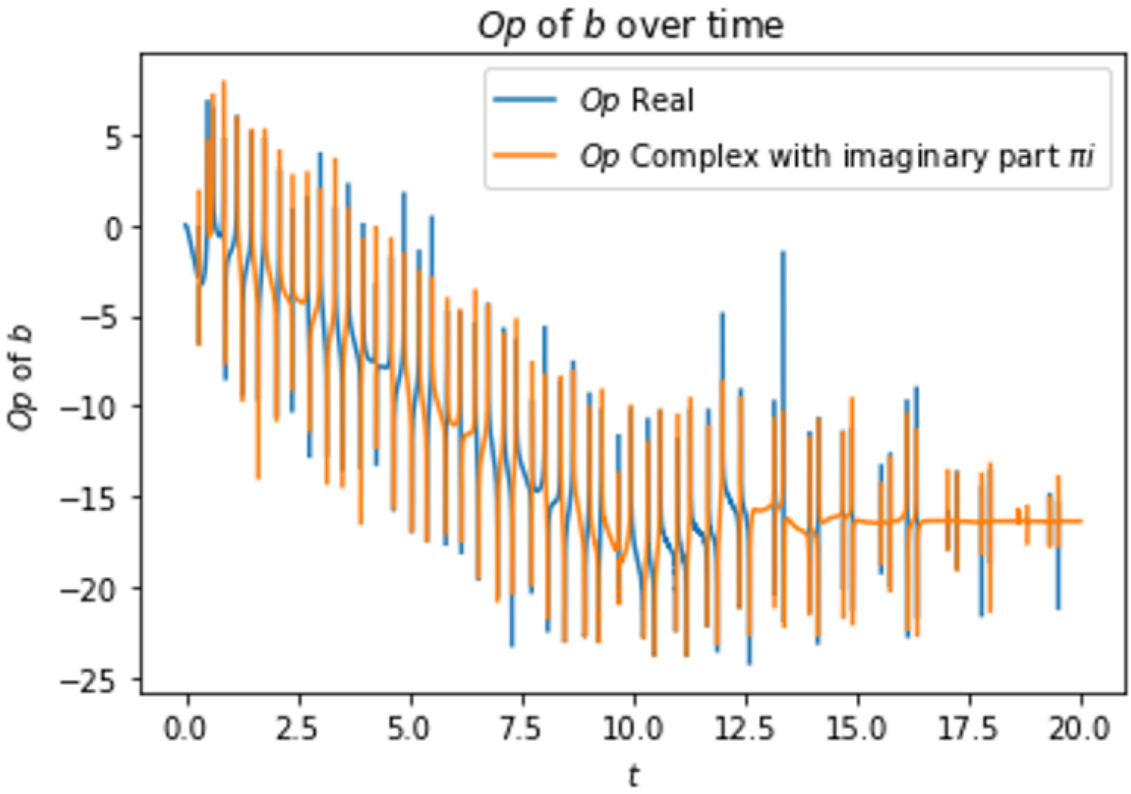
Now the variations are huge! And they're wild and fluctuating.

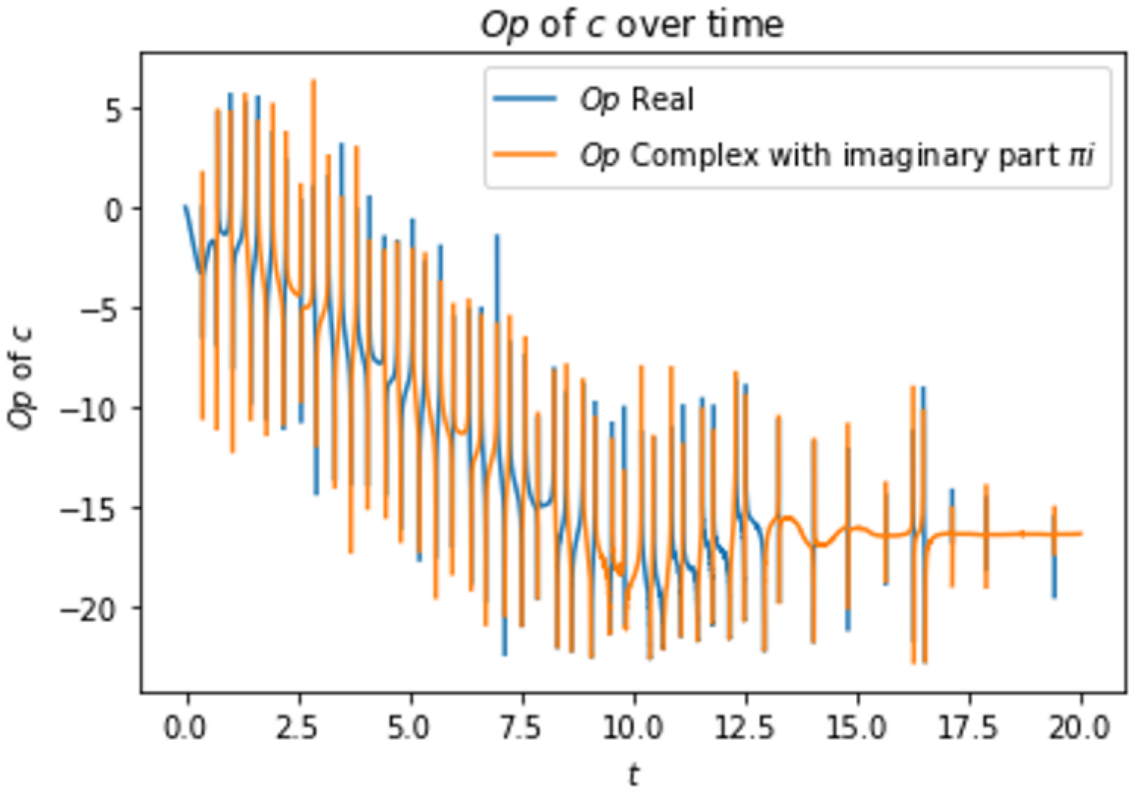

Op of b over time
Op Real
Op Complex with imaginary part πi
Op of b
t
5
0
−5
−10
−15
−20
0
2
4
6
8
10


Op of c over time
Op Real
Op Complex with imaginary part πi
Op of c
t
5
0
−5
−10
−15
−20
0
2
4
6
8
10

Huge variations across everything. This is basically what it means to have a chaotic system. But interestingly there is a trend towards Op becoming negative in most cases, which should tell us something, namely that these things are spreading one another out.

What happens if we define A as $a, t \leq 10$? This means that for s_t^a values with $t > 10$ we allow a difference between the w^a and y^a values. We get graphs that look like this:

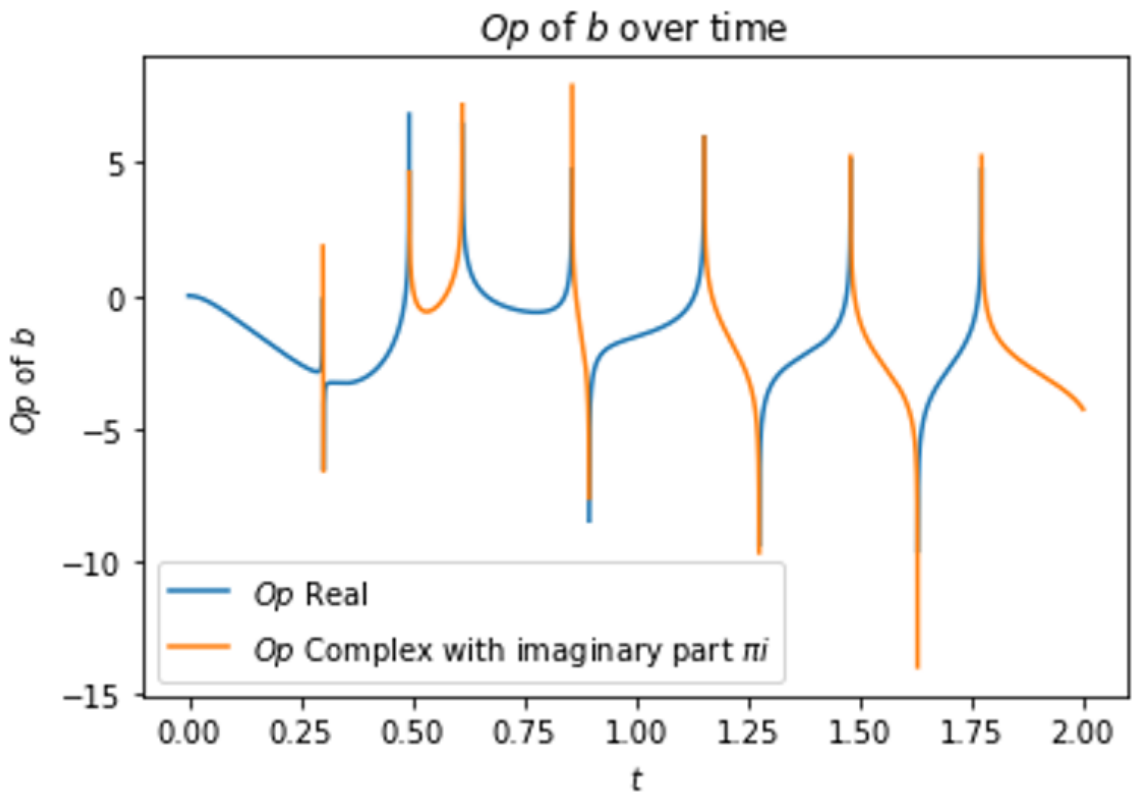




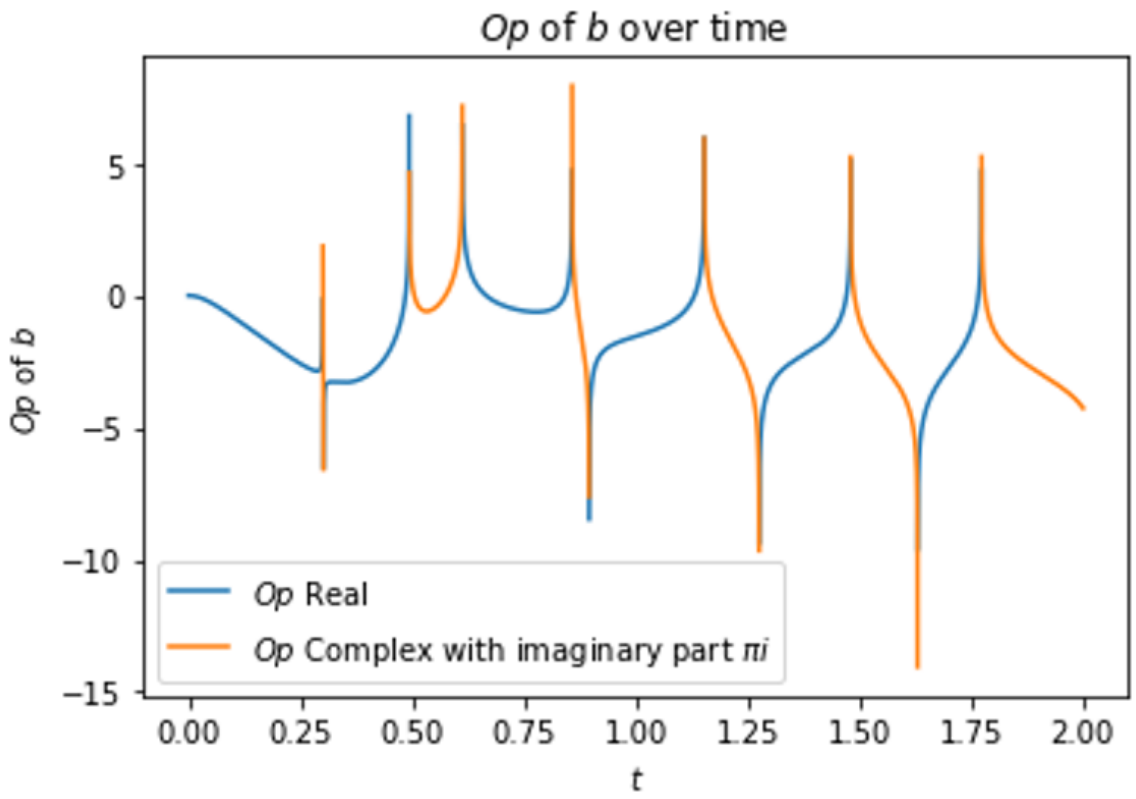
This is actually a good sign. Since A only has a finite amount of influence, we'd expect that it can only de-optimize b and c by a finite degree into the future.

Point 5: Op **Seems to be Defined Even in Chaotic Systems**

It's also worth noting that we're only looking at an *approximation* of Op here. What happens when we reduce the $\delta b = x_0^b - w_0^b$ by some amount? In our other cases we get the same answer. Let's just consider the effect on s^b.

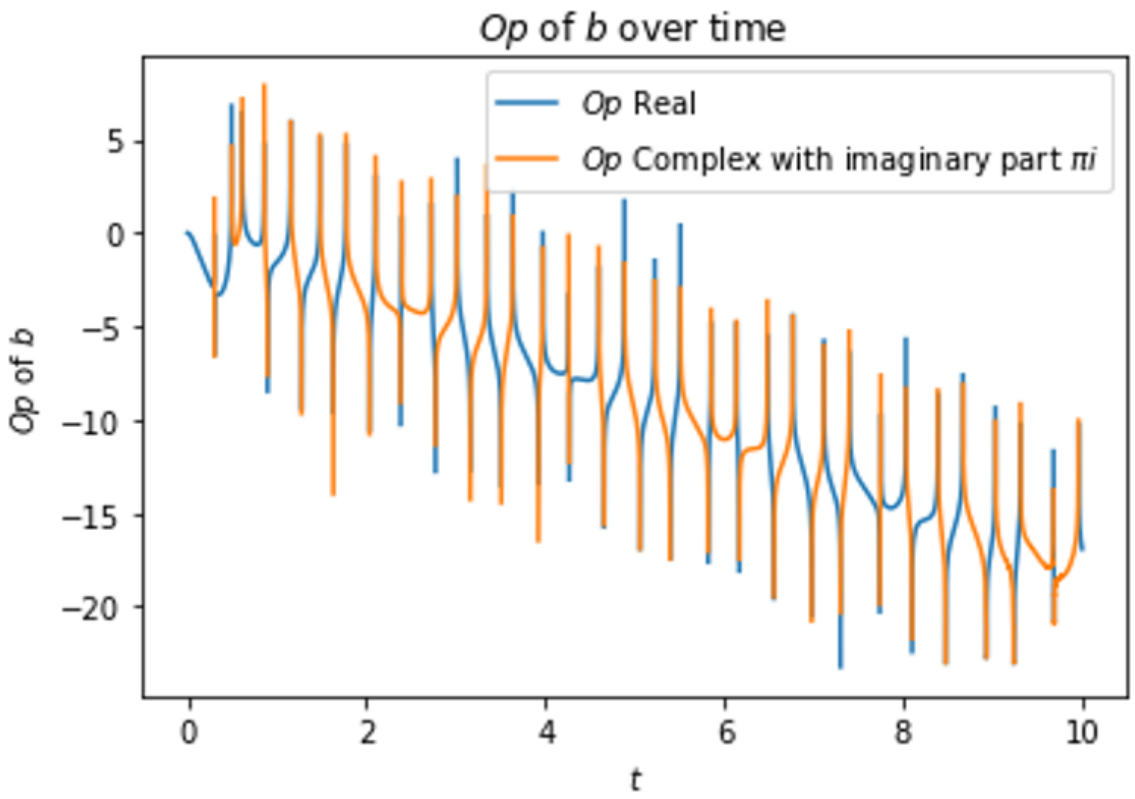


$$\delta s_0^b = 10^{-5}$$

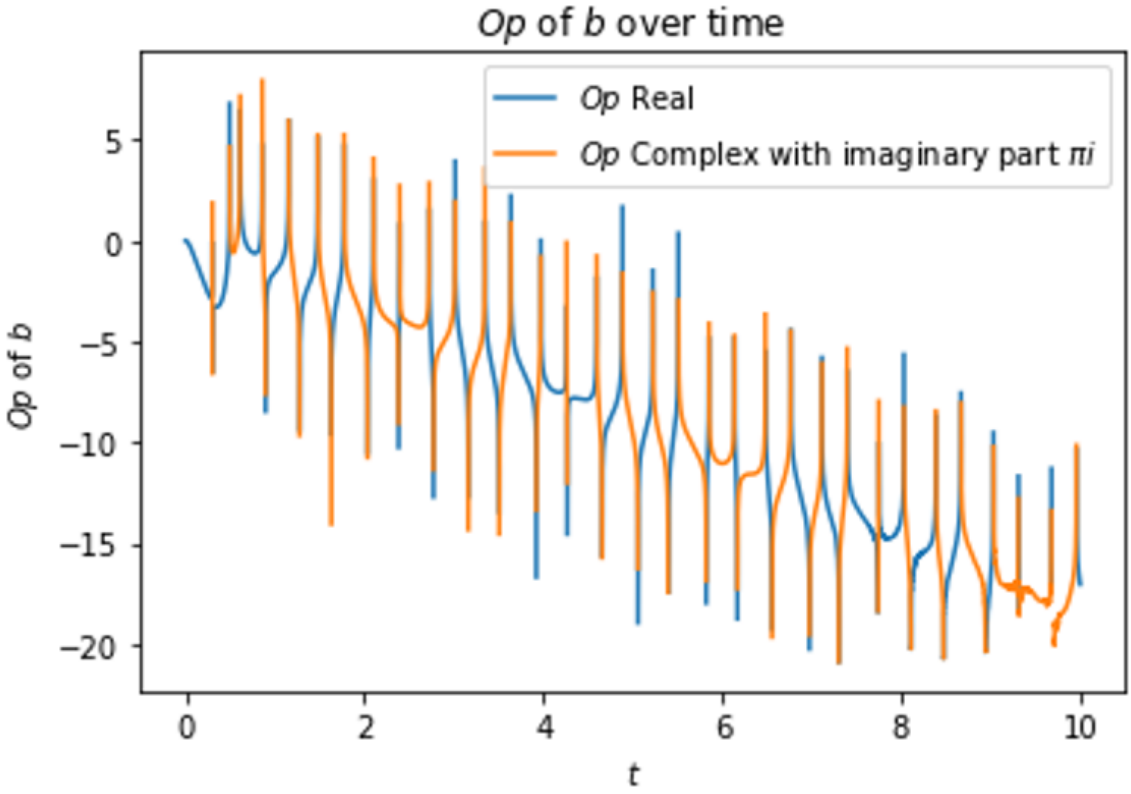


$\delta b = 10^{-6}$

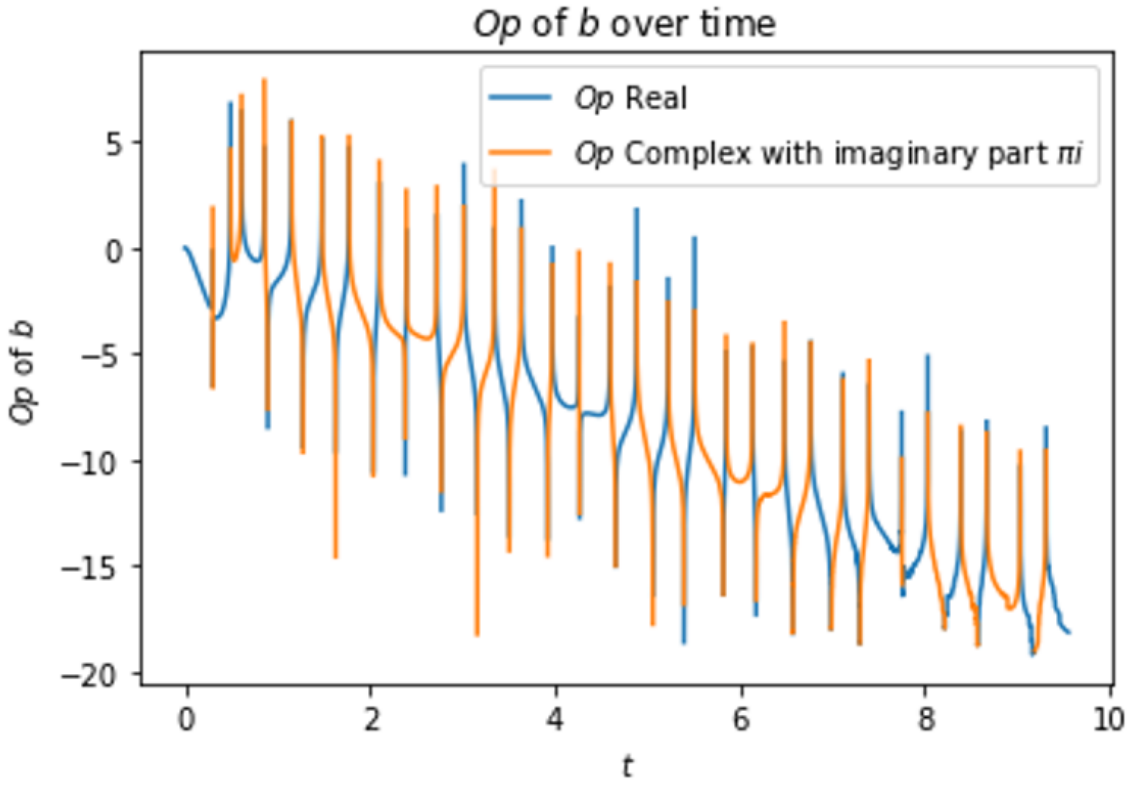
Works for a shorter simulation, what about a longer one?


Op of b over time
Op Real
Op Complex with imaginary part πi
Op of b
t
5
0
−5
−10
−15
−20
0
2
4
6
8
10

$\delta b = 10^{-5}$


Op of b over time
Op Real
Op Complex with imaginary part πi
Op of b
t
5
0
−5
−10
−15
−20
0
2
4
6
8
10

$\delta b = 10^{-6}$



$\delta b = 10^{-7}$

This seems to be working mostly fine.

Conclusions and Next Steps

Looks like our system is working reasonably well. I'd like to apply it to some even more complex models but I don't particularly know which ones to use yet! I'd also like to look at landscapes of Op and Comp values for the Lorenz system, the same way I looked at landscapes of the thermostat system. The aim is to be able to apply this analysis to a neural network.