*Fantastic networks and how to evolve them: understanding polygenic adaptation in the age of gene regulatory networks*

*Part 1: The Negative Autoregulation Motif*

Stella Maria Sophie Knief

The University of Queensland

School of Biological Sciences

Ortiz-Barrientos Laboratory

COE for Plant Success

**Keywords:** I would love a nap,

**Word Count:**. 2323

*Table of Contents*

[Overview 3](#_Toc69917076)

[Step 1: Mutations 5](#_Toc69917077)

[Step 2: Change in motif parameter values 5](#_Toc69917078)

[Step 3: Motif Dynamics (ODE’s) 6](#_Toc69917079)

[*A brief note on R integration* 7](#_Toc69917080)

[Step 4: Motif Dynamics (AUC) 7](#_Toc69917081)

[Step 5: Translation to Phenotype 8](#_Toc69917082)

[Step 6: Selection 8](#_Toc69917083)

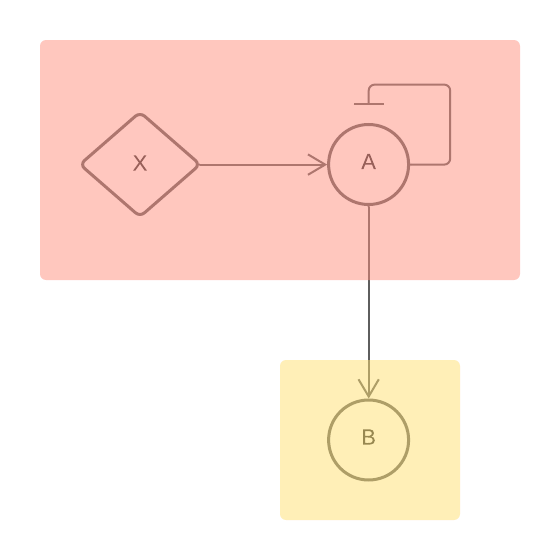
[Step 7: Survival 9](#_Toc69917084)

[Step 8: Offspring production 9](#_Toc69917085)

[SO…. 9](#_Toc69917086)

# Overview

The negative autoregulation motif (NAR) is the simplest motif structure that exists within Pygmalion. It consists of only one node, node A, that is connected to one node of the following motif, node B (Figure 1). Whilst this motif is characterized by only one node, the succeeding node is included in this model as to demonstrate the effect of changes in node A and how these cascade through succeeding nodes. Node A is activated by a signal, here shown as signal X (Figure 1b).

Graphical user interface, application

Description automatically generated

**Figure 1.** A) The complete structure of Pygmalion, the network model to be built within this honours year. Motifs are highlighted with different coloured blocks: Negative autoregulation (red), multi-output module (yellow), incoherent feedforward loop (blue) and bi-fan (purple). B) A close-up of the negative autoregulation motif (shown in red) with its immediate succeeding node (shown in yellow).

In this model, it will be assumed that the activation signal (node X) is an environmental input such as the length of the observed photoperiod, and that node A represents a transcription factor (TF) that directly causes flowering in a hypothetical plant. The start of the activation signal (Xstart) will be a variable to be set before the simulation begins and the end point of that signal (Xstop) will be handled as a constant. The observed phenotype, flowering time, is here defined as the time it takes for the first bud to appear on a plant. This phenotype relies on the amount of TF A that is produced within whilst the signal is present.

This is the first out of four motif models to be developed this year, and the NAR is the perfect place to start developing the foundations that all *in silico* models operate under. These foundations include the generation of mutations, the translation of genotype to motif dynamics and the way in which motif dynamics determine the observed phenotype. This document thus serves as a tentative blueprint for the model. I believe the best way to go through my plan so far is to look at what happens within the model each generation and explain the mechanism behind each step.

**Figure 2.** Flowchart of the processes occurring in each generation. Blue boxes represent steps that need to occur within R, whereas white boxes represent steps that occur in SLiM.

Within each generation, eight processes occur:

1. Generation of mutations
2. Motif parameter values are changed by the new mutations
3. The motif dynamics are modelled in R using differential equations
4. The integral of the motif dynamics is taken
5. This “motif value” is translated into the phenotype
6. This phenotype is subjected to selection
7. Mortality occurs
8. New offspring is generated

This cycle then begins anew.

I want to ultimately create two versions of this model, one with the NAR motif and one as a simple regulation equivalent, where A does not autoregulate. That way, we can compare the effect of the specific motif onto the population (and it should be fairly easy to code)!

# Step 1: Mutations

The generation of mutations is a fairly simple process within SLiM. Three mutation types will be introduced (“m1”, “m2”, “m3”) where one mutation type represents neutral, beneficial and deleterious mutations respectively. The neutral mutations will be drawn form a fixed fitness effect function that is functionally neutral, whereas the beneficial mutations will follow gaussian distributions and the deleterious mutations will follow exponential distributions:

*//Example code*

initializeMutationType("m1", 0.5, "f", 0.0);

initializeMutationType("m2", 0.5, "g", 0.5, 1.0);

initializeMutationType("m3", 0.5, "e", 0.4);

These mutation types will apply to all coding genomic element types and noncoding genomic element types will be affected only by neutral mutations (see Step 2 for an explanation of the genome and genomic element type set-up).

**TL;DR:** Mutations are generated and applied to the relevant genomic units.

# Step 2: Change in motif parameter values

The genome within this simulation may look something like this:

α

β

n

α

β

**Figure 3:** Tentative genome layout. Red squares represent node/TF A parameters and yellow squares represent node/TF B parameters. Not to scale.

All genes are equal in length and may span 1000kb. The is no recombination between the two halves of the chromosome (shown by the black line in Figure 3). Each parameter has an initial value and is then altered by the effect sizes of the incoming mutations.

Parameter = initial value + Σ[effect size of mutations]

So, for example, if the initial parameter value for ß was 10 (arbitrarily chosen number; I have not looked into expected ranges yet) and a deleterious mutation occurs in an individual with an effect size of -1, it follows that the new parameter value is 9 for that individual within that generation, assuming that this is the only mutation affection ß.

**TL;DR:** Motif parameters are encoded by separate loci and mutations change these values.

# Step 3: Motif Dynamics (ODE’s)

Okay, I will be honest: this step and the Step 5 I am still the shakiest on, so please bear with me.

I’ve looked at Jan’s code for the shiny app and I’ve taken the two ODE’s for the NAR system and used them as the basis for this step. We are interested in the production rate of A and production rate of B, which I have understood to be best represented by:

respectively, where

is the maximum promoter activity (driven by mutations)

is the time in which the signal X is present (given as Xstart && Xstop in R)

is the decay rate of the protein (driven by mutations as this is unique for the proteins, I have checked – depends on an ubiquitin constant that is determined by their structure)

is the hill constant and determined the shape of the function (driven by mutations)

The first equation here is taken from Jan’s ODE for the active motif, and the second equation is taken from Jan’s ODE without the NAR motif, so I am assuming that is representative of simple regulation. My main caveat is the dependency of B on A, as we want to represent the cascading effect of these motifs and proteins (I am very happy with ). My intuitive suggestion would be either multiplying the by the concentration of A at time t or perhaps allowing the production of B to only occur in the time points where A is at least ½ of it’s steady state concentration, K. The advantage of that latter approach is that the parameter K can then be included in the model and this may also be driven by mutations (add a new loci then). I did want to ask why K was not included in your model, Jan, so this is a perfect reminder. Like I said, this is where I am shakiest, so I would love it if we could put our heads together on this one in the meeting.

**TL;DR:** The motifs are considered through ODE’s describing their unique dynamics. I don’t know these yet for sure.

## A brief note on R integration

Regardless of what the function looks like, I am anticipating to solve these in R. I have not tried any of these yet (I am planning to), but I have found a few good resources / places to start in terms of R and SLiM integration:

1. The SLiM workshop on animated graphs in R
2. A few GitHub sources that attempt integration
   1. bodkan/slimr
   2. rdinnager/slimrlang
   3. rdinnager/slimr

It may be possible to run and solve these ODE’s in SLiM, but I have a gut feeling R might be quicker and easier (or what do you think?).

# Step 4: Motif Dynamics (AUC)

Once I’ve gotten the ODE’s, I need to be able to return a value that summarizes their dynamics or rather the change therein due to the shift in parameter values. I was thinking that area under the curve (AUC), or rather taking the integral of these ODE’s might be a good method for this?

By taken the AUC, we take into account all changes made by the parameters: changes in maximum promoter activity changes height, the hill constant changes the shape of the upwards slope, and the decay rate affects maximum height too. It seems like an elegant solution, as this area would represent the total activity of the focal transcription factor, which may then be correlated to the observed phenotype.

This would also occur in R the way I am currently planning it, so it depends on the same resources as above.

(is this rubbish?)

**TL;DR:** Return the AUC as a value for SLiM to work with rather than feeding the whole ODE back to SLiM.

# Step 5: Translation to Phenotype // SKIP THIS AND WORK WITH ACTIVITY AS A FITNESS FUNCTION (SEE MEETING PICTURE)

Okay, once we have our AUC values, we have to relate them back to our phenotype that we are working with, which is a very simple flower time. So I am thinking that this quasi genotype-to-phenotype\* translation could look something like this:

(\*Does this qualify as a genotype-phenotype map? Or am I misconnecting two thoughts here?)

**Figure 4:** Tentative look at what a conversion might look like. The axis labels are awful and nonrepresentative. I’ve only done this because it still looks kind of better than me drawing this with a pen.

I’ve had a lightbulb moment when Daniel drew a figure like Figure 4 on the whiteboard in a meeting, and I do think this is the best way of going about it. Essentially, the translation of the activity of A that we have measured depends on a function; once we have our AUC, we can determine the corresponding flowering time. I intuitively think this function should be sigmoidal, but I am still reading more into flowering time papers to figure that out.

My main question here, however, is whether this function is something that I can determine or if that is something I should be able to find in literature? And if so, what specific key words am I looking here for?

( ALSO NOTE: TF B has no impact on the measured phenotype, it is just there as a check if the cascading function works / I want to output the activity of B over time as well in these simulations – just the TF B has no relationship to flowering time quite yet.)

TL;DR: I have an idea, but the nitty gritty isn’t here yet / I am confusion.

# Step 6: Selection

For selection to occur, we need an optima and a shift in optima. I am envisioning that during the burn in period, optimum flowering time could lie at a fixed value opt1, and then the start of the experimental period is characterized by a shift to optimum flowering time opt2 (e.g. opt1 = 4 and opt 2 = 10). This results in directional selection.

Another idea I had was letting the burn-in period determine opt1; so for example, if the populations ends up sitting at a mean flowering time of 4 days after the burn-in period, SLiM could set that as the optimum for the first half of the experiment and then in the later half, the optimum shifts so that it is the opt1 + 5 days, for example. That way we could have directional and stabilizing selection that is observed within one simulation; and I know it is possible to change selection behaviour in SLiM after a given generation, so I reckon it could be feasible.

In terms of the actual code and functions for selection, I will be relying on the SLiM Manual, as it has guidelines and code for both temporally varying fitness functions (CH 10.1) and directional vs stabilising selection (CH 13.6). I have not drafted any specific functions yet, but I am anticipating to run those through both of you before they get into my code. Realistically, the first thing I will do is simple tests of R-SLiM communication and then build the environment and genome – so it should have a few more days time!

**TL;DR:** Selection. Yes.

# Step 7: Survival

I have been thinking about two possible ways of handling mortality: either a fixed percentage dies off after each generation (i.e. the 10% that are the furthest away from the population optima) or a variable amount can die off depending on the deviation within the population (i.e. all individuals with a flowering time smaller than 3 die immediately). Alternatively, a combination of both may be used in a type of “knock-out” mutation scenario that applies in addition to the 10% rule. These knock-out rules can be based on literature or we can put them in to just check functions and constraints.

**TL;DR:** Three ways in which we can model survival, no preference for any of these yet (maybe a mild preference for mixed?)

# Step 8: Offspring production

The population will operate under a carrying capacity and will attempt to reach this capacity after each generation. Offspring production should be an easy final step.

**TL;DR:** should be okay!

# SO….

As you can see, there’s still a few things I am not too sure about / if my understanding here is correct, so I am hoping we can go through all of (or most of) that tomorrow! I hope the little write up helps – I think it makes more sense for me to try and write stuff like this down before the meeting and send it to you two so that we can work through my blockages/ rather than me trying to explain all this with less time and more “uhmm”s as I speak. Hopefully this made some sense and doesn’t sound insane / completely wrong!

:)