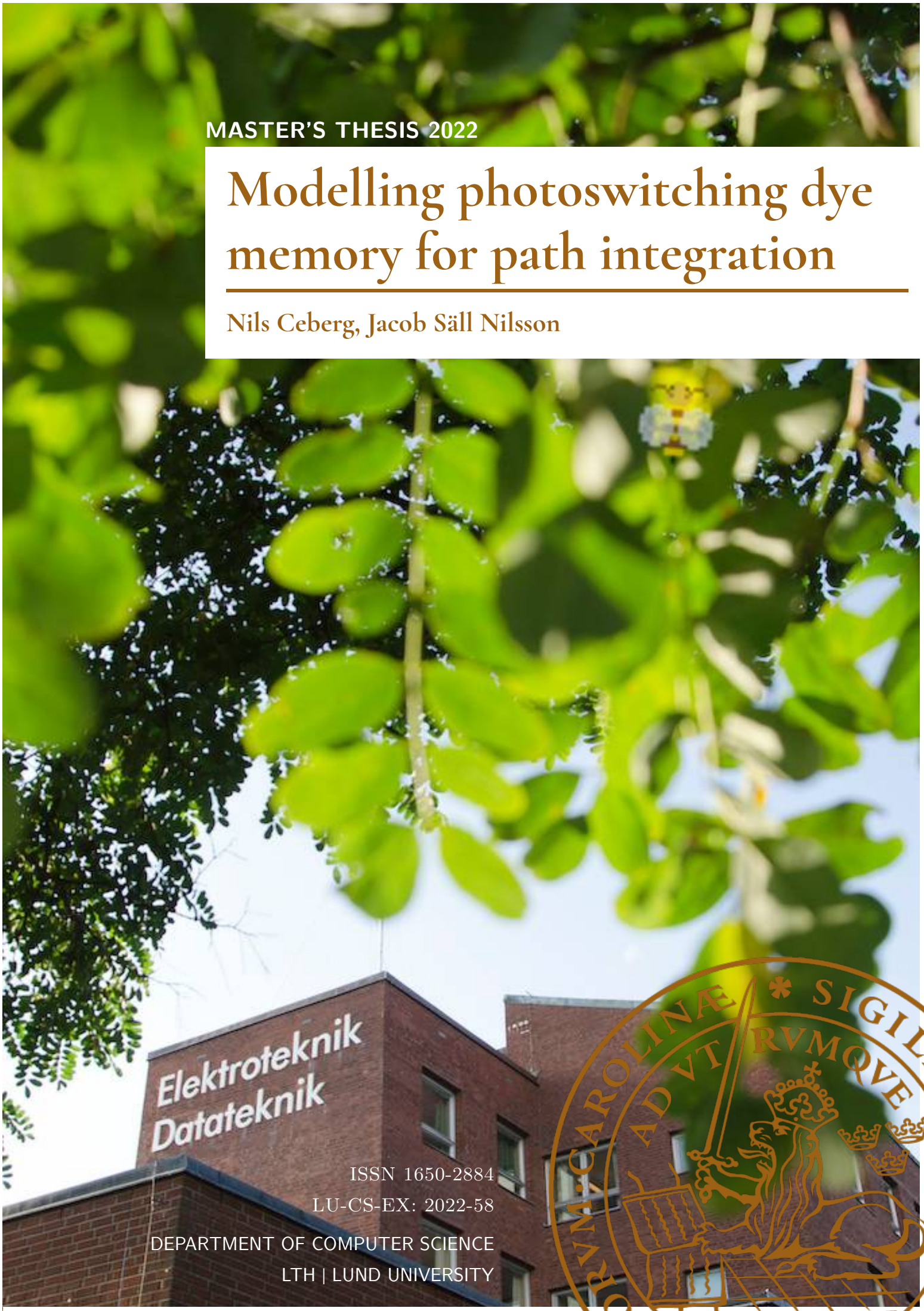


MASTER'S THESIS 2022

Modelling photoswitching dye memory for path integration

Nils Ceberg, Jacob Säll Nilsson

ISSN 1650-2884
LU-CS-EX: 2022-58

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2022-58

Modelling photoswitching dye memory for path integration

Modellering av photoswitch-molekyler som minne för vägintegrering

Nils Ceberg, Jacob Säll Nilsson

Modelling photoswitching dye memory for path integration

Nils Ceberg
ni7228ce-s@student.lu.se

Jacob Säll Nilsson
ja0877ni-s@student.lu.se

November 23, 2022

Master's thesis work carried out at
the Department of Computer Science, Lund University.

Supervisors: Stanley Heinze, stanley.heinze@biol.lu.se
Barbara Webb, bwebb@inf.ed.ac.uk

Examiner: Jacek Malec, jacek.malec@cs.lth.se

Abstract

The central complex region of insect brains contains neural circuitry suitable for performing path integration, used for example by bees in order to find their way home after foraging. An artificial neural network based on this anatomy has previously been successfully modelled, but without considering the memory mechanism in detail.

We explore, using computational modelling, the idea of using photoswitching dye molecules as synaptic weight-based memory in a physical nanowire-based realisation of such a neural network.

With a simplified model of the dye molecule dynamics and minimal changes to the network we perform a brute force parameter optimization and find that we can get a barely functional path integrator within the realistic parameter ranges of the dye molecules. We go on to suggest some additional changes to the network that increase performance to levels comparable to the previous model.

Finally, we also find that the nonlinearity of the dye memory has negative consequences for the prospect of being able to use this circuitry for more general vector-based navigation, and briefly discuss how our work may relate to biological memory mechanisms.

Keywords: memory, path integration, neural networks, nanowire, photoswitches, neuromorphic computing

Acknowledgements

Thank you to Stanley and Barbara for letting us get a glimpse of this fascinating research, for their great input, and for helping us make sense of our confusion every Tuesday.

We are also grateful to Bo for an introduction to the dyes and for feedback on the model, to Roman and David for general input and insights, to Valentin for sharing his precious CPU time, and to Janka and Mathilda for reading and giving us feedback on this report.

Thanks finally to the rest of the BrainInBrain group – especially to Ebba for putting up with us all summer in B226.

Contents

Chapter 1
Introduction

When foraging for nectar, bees travel along convoluted paths that may reach multiple kilometres away from their nest. Despite this, they are able to fly in a straight line back home for their return journey. This is an old observation, and has even given rise to the expression ”beeline” for describing a straight path to a target [1].

There is evidence that they do this by visually estimating self-motion in relation to an external compass based on celestial cues, such as the sky’s polarisation pattern. In the *central complex* (CX), a brain region that is highly conserved among many insect species, it is thought that this self-motion is integrated – in the mathematical sense – to maintain a home-pointing vector, in a process aptly known as *path integration* [9].

A circuit that has the components necessary to be able to perform such path integration has been identified in the bee brain and successfully modelled as an artificial (recurrent) neural network, which we will refer to as the *Stone model* [22]. In this model, however, the actual memory has only been implemented abstractly as a generic integrator, and the underlying mechanism is unknown.

Meanwhile, in a separate field, nanowire-based optoelectronic neural devices (that communicate using light) have been developed whereby artificial neural networks can be implemented physically on micrometer scales and at nanowatt energy use. Previous work has proposed a version of the Stone model where the *ring attractor* subnetwork has been implemented using such devices [23]. This is the first step toward constructing a full path integration circuit in this manner. Still, the memory mechanism remains a missing part of this puzzle.

In yet another area of research, so-called *photo-switching dye molecules* (which we will refer to as simply *dyes*) are being developed [8][14]. A key property of these dyes is that their transparency to light depends on the amount of light they have previously absorbed. In effect, the attenuation of a light signal passed through them will represent a *memory* of previous signals. Because one likely biological memory mechanism is the *plasticity* (i.e. changeability) of the synaptic weights between neurons, and because the nanowire neural devices communicate using light, these dyes are an interesting candidate for implementing such memory.

Thus, in our thesis, we explore the idea of using dye molecules placed as an optical medium between some of the nanowire neurons to act as plastic synaptic weights. There are many challenges in realising this concept in an actual circuit, including the physical geometry of the connections. Our project focuses on the initial plausibility of a dye-based memory by making modifications to the Stone model aimed at replacing the generic integrator with synaptic weights with dynamics that attempt to model the dye behaviour. As criteria for success we consider how well the dye-based model recreates the behaviour of the original Stone model (quantitatively and qualitatively), and whether the required parameter settings fall within realistic ranges for the real dyes.

Finally, we also try to relate the behaviour of the dye-based memory to more detailed proposals of biological memory mechanisms, in the hope that the development of the nanowire implementation and continued biological research might help inform each other.

1.1 Goals & research questions

As described above, the thesis explores the following three research questions:

1. Can the Stone model be implemented using a memory that models candidate photoswitching dye molecules?
2. What modifications to the topology of the network are required?
3. Can our results be related to plausible biological memory mechanisms?

To answer these questions, we set up the following sub-goals:

- Create a framework that allows for implementing and evaluating variations of the Stone model.
- Create a conceptual model of the memory as synaptic weights.
- Implement such a model using a mathematical model of the candidate dye molecules.
- Evaluate performance of models, quantitatively and qualitatively, using the Stone model as a benchmark.
- Compare to biological memory mechanisms whose dynamics can be compared to ours.

1.2 Divison of work

Most parts of the work were produced as a pair. The ideas were produced as a collaborative effort and all parts have been thoroughly discussed. If not otherwise specified, the work for a particular section was divided equally.

Nils had primary responsibilty of the qualitative evaluation of the results, and was the main contributor to the methods, the results up until section 4.2.6, and much of the discussion. Jacob's primary responsibility was the quantitative evaluation, and he also performed the experiments on obstacle avoidance and holonomic movement. He was the main contributor to the background on physics and the results from section 4.2.6 and onwards.

Chapter 2
Background

2.1 Computational neuroscience and biological basis

This first section aims to introduce a simple mathematical model of neurons and neural networks, as well as some general biological background on memory and the central complex brain region.

2.1.1 Neurons

Neurons are complex cells, receiving inputs from many other neurons and providing outputs to many others still. They communicate using *action potentials* – spikes in the cell's electric potential that carry information which is propagated to its post-synaptic partners. While these spikes have precise timings, we employ an abstraction: we deal only with *firing rates*, i.e. the frequency of spikes. Additionally, since neurons have a saturation level in the form of a maximum firing rate, we normalise each neuron's firing rate to a dimensionless number between 0 and 1, representing the degree of activity [3]. Throughout this thesis, we will use the term *activity* to refer to a neuron's normalised firing frequency.

In theoretical neuroscience, as well as in the artificial neural networks used in machine learning, the simplest model of a neuron multiplies each input with a corresponding weight (the strength of the synapse), sums them, and passes the sum through an activation function, which is typically nonlinear.

The sum of the weighted inputs (the *total synaptic current*), I, is thus

$$I = \sum_i w_i u_i = \mathbf{w} \cdot \mathbf{u} \tag{2.1}$$

where $\mathbf{w}$ is the weight vector (synaptic strengths) and $\mathbf{u}$ is the input firing rate vector.

The output firing rate v is then simply

$$v = f(I) \tag{2.2}$$

where f is the activation function (also known as the *f-I curve*, as it relates input current to firing rate). Note that I, $\mathbf{u}$ and v are implicitly functions of time.

Additionally, real brains exhibit significant noise. To reflect this, we can add some randomness to the value of v at each point in time. Exploring the resilience of model circuits in the face of such noisy conditions is an important part of their evaluation.

In the description above, $\mathbf{u}$ and v are used to denote input and output firing rates for a single neuron. When later describing the neural network model, we will usually refer to the (output) firing rate of a layer L as a whole as the vector $\mathbf{r}_{\mathrm{L}}$, and its total synaptic current as $\mathbf{I}_{\mathrm{L}}$. The firing rate of a single neuron i in that layer would be denoted as r_{L_i}.

This is the neural model used in the Stone model of the path integration circuit. In the Stone model, the simulation is stepped forward in discrete time units, and propagation of signals is instant and synchronous. In particular, this means that recurrent connections use the output from the last time step. For example, a neuron's activity might be a function of itself at the previous time step like so:

$$v(t) = f(v(t-1)) \tag{2.3}$$

2.1.2 Population coding

While a single neuron's rate-based activity level is a scalar, populations of multiple neurons can together encode higher-dimensional information, which is known as *population coding*.

Consider for example a vector representing the current head direction. With a population of neurons where each cell is tuned to a certain direction (i.e. it is most active when the head direction matches its *preference angle*), the head direction vector can be represented by its projections onto the unit vectors with those angles [3]. This forms a *population vector code*, and has the advantage of redundancy.

When the activities of an infinite population of neurons are plotted against their preference angles spanning from 0^{o} to 360^{o}, this results in a sinusoid whose amplitude and phase represent the length and angle of the represented vector, respectively (since a firing frequency cannot be negative, the sinusoid is shifted "upward" so that its mean is 0.5). We will thus often refer to the population-encoded vector as a "sinusoidal bump". Figure 2.1 attempts to illustrate the idea.

Since the projection of a vector onto the preference vectors is a linear operation, this encoding has the crucial property that two populations can be summed to represent the sum of the vectors they encode, and in the continuous case, they can be integrated. Viewed as sinusoids, this is equivalent to the fact that adding sinusoids of the same frequency yields a sinusoid with the same frequency, but with a change in phase and amplitude [22].

2.1.3 Memory

One widely hypothesised mechanism by which a neural network can exhibit memory is persistent recurrent activity, i.e. neurons firing in a loop. For example, a simple integrator based

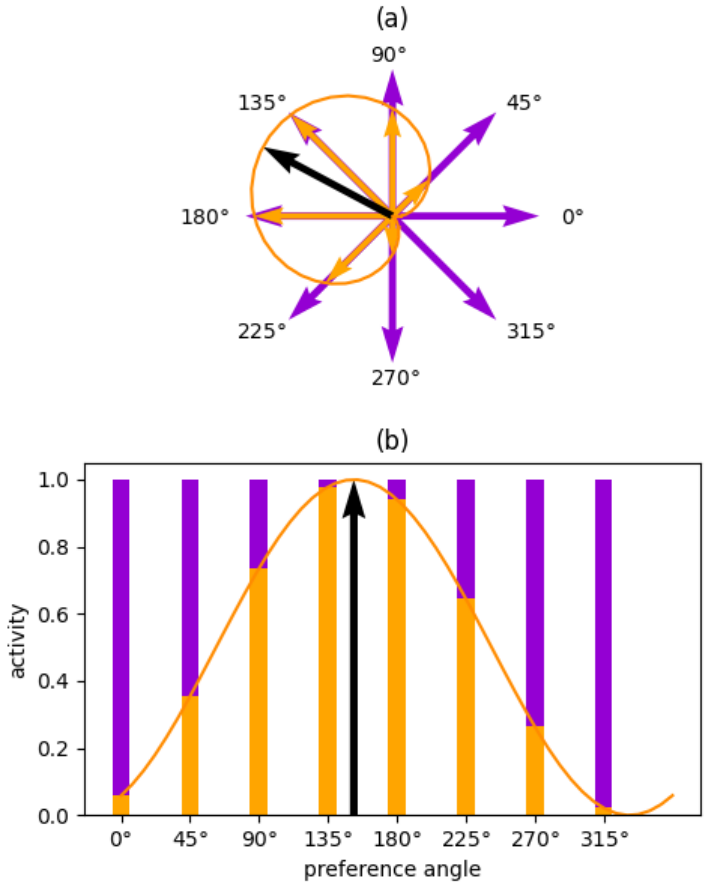


Figure 2.1: An illustration of population vector coding. **(a)** The black vector's representation as its projection (orange) onto the preference vectors of a population (violet). **(b)** The same projection illustrated as a sinusoid.

on a single neuron with a recurrent connection and a single external input can be modelled by

$$\tau \frac{dv}{dt} = hv + u - v \tag{2.4}$$

where hv represents recurrent connections with a net gain of h, and u is an external input firing rate. However, this circuit requires said gain to be very close to 1 in order to constitute a sufficiently stable memory. Note how $h = 1$ corresponds to perfect integration of u: $\tau dv/dt = 1 \cdot v + u - v = u$.

However, if h is even slightly mistuned, the integrator will either rapidly "forget" (if $h < 1$) or accumulate until it reaches saturation ($h > 1$). This may be suitable for short-term memory (perhaps on the scale of seconds to minutes) [3], but as path integration needs to work on timescales from hours to days, other mechanisms may be more likely. Furthermore, placing path-integrating insects under anesthesia that disrupts neural activity does not necessarily cause them to entirely lose the memory of their home vector [20]; instead, recent work has shown that the home vector degrades gradually, again suggesting that there may be another mechanism at play. Interestingly, the length of the vector is lost faster than the direction, which is consistent with the population vector coding.

One such alternative mechanism is memory that is stored in the synaptic weights, i.e. the strength at which a signal is propagated from one neuron to the next. This forms the biological argument for the dye-based memory that we attempt to model. While the Stone model (described below) assumes that the PFN cells have the ability to themselves accumulate activity, we will instead explore the idea that the accumulation happens in the synapses between PFN and their post-synaptic partners.

2.1.4 Central complex

The central complex is a collection of distinct regions in the brain of all insects that has a key role in navigational tasks. The neurons in the central complex are characterized by columnar organization and interhemispheric connections [19]. Sensory information, especially from visual systems, converge in the central complex. Different species can make use of different sensory information, but both external cues, such as visual landmarks, wind direction, polarized light patterns or the earth's magnetic field, and internal cues, for example optic flow or proprioception, have been shown to play a role in navigation. The difference in sensory information presents itself as different navigational strategies, such as straight-line orientation, long-distance migration, landmark based navigation and path integration, which is what this project is focused on [6].

While the central complex is highly conserved across insects species, it has been studied separately in different species which has given rise to some discrepancies in terminology. The Stone paper uses cell type names specific to the *Megalopta*, but as the field is moving towards standardising on using *Drosophila*-based (a common model species in neurobiology) terminology, we have chosen to use the latter as well. To aid in relating our model to previous work, table 2.1 provides a translation key.

Drosophila	*Megalopta*
ring neurons	TL2
EPG	CL1
Δ7	TB1
LNO	TN
PFN	CPU4
PFL	CPU1
hΔ	pontine

Table 2.1: Translation key of cell types (layers) between *Megalopta* and *Drosophila* [9][5].

2.2 The Stone model of path integration

What we refer to as the Stone model [22] is a model of a neural circuit that contains all necessary components to be able to perform path integration and is anatomically constrained by central complex connectomics from *Megalopta genalis* (the sweat bee). The model is a rate-based neural network whose high-level connectivity is outlined in 2.2. It consists of eight layers (translated from the original paper according to the key in table 2.1), listed here and explained in more detail below:

- **Ring neurons** that represent the compass input, which is passed through the **EPG** layer for completeness,
- **Δ7** cells which make up a *ring attractor* that stabilises noisy compass input,
- **LNO** cells that are sensitive to *optic flow* and whose activities reflect the speed of the bee through its environment,
- **PFN** cells where speed input is converted to velocity and integrated (i.e. constitutes the actual memory),
- **hΔ** which is required for memory balance,
- **PFL** which compares the PFN memory with the current head direction,
- and finally the **motor** layer which uses the PFL output to generate a steering signal.

2.2.1 Speed neurons: LNO

A crucial requirement for path integration is the ability to estimate the speed at which you are travelling. In the Stone model, this ability stems from LNO neurons that are sensitive to optic flow in such a way that their activities encode the bee's speed through the environment, thereby acting as a visual odometer.

Optic flow refers to the global motion patterns that appear to your eyes as you move around in space. An example is that as you walk on a path, looking the same direction that you are moving, the environment around you appear to move outward into the periphery of your

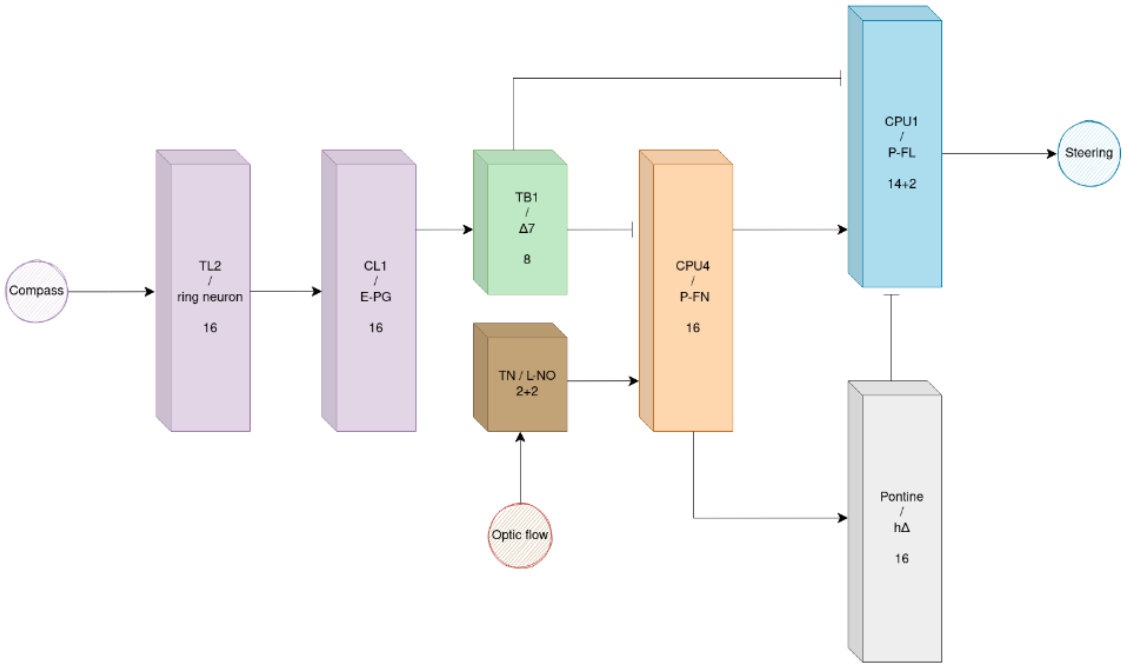


Figure 2.2: A high level schematic of the layers in the Stone model. An arrow represents an excitatory connection, and a flat end represents inhibition.

field of vision. Just as when spaceships move into hyperspace in Star Wars, all objects around you move radially from a point ahead where you are moving (and looking). As you walk, if you would move from side to side, or rotate, this radial pattern would change accordingly. This optic flow can be used by the observer to discern the direction and speed of self motion [18].

There are two types of LNO neurons (called TN1 and TN2 in *Megalopta*) that are sensitive to translational optic flow, that is to say they are sensitive to forwards and backwards motion. TN2 cells were found to be excited by forward motion, while TN1 cells are inhibited by forwards motion, and instead excited by backwards motion. There are one of each type in each hemisphere of the brain. Each LNO neuron has a preferred angle of expansion $[0, 2\pi)$, which is a point of expansion of optic flow that generates the largest response in that cell. In the Stone model, these angles are $\pm\frac{\pi}{4}$ to the left and right of the body axis for forward motion, and $\pm\frac{3\pi}{4}$ for backward motion [22].

Together, a pair of these neurons (one in each hemisphere), represents the bee's velocity through space as Cartesian coordinates. Being able to represent a velocity that is not directly aligned with the head direction is important, as not all movement is strictly in the forward direction; for example, a bee is easily swept sideways by a gust of wind (known as *holonomic* movement). Note, however, how this coordinate system is *egocentric* – the velocity is expressed in terms relative to the bee's orientation. As long as it travels in its forward direction at a constant speed, the LNO cells will represent the same forward vector even when turning, as the coordinate system turns along with it (just like the notions of *left* and *right*).

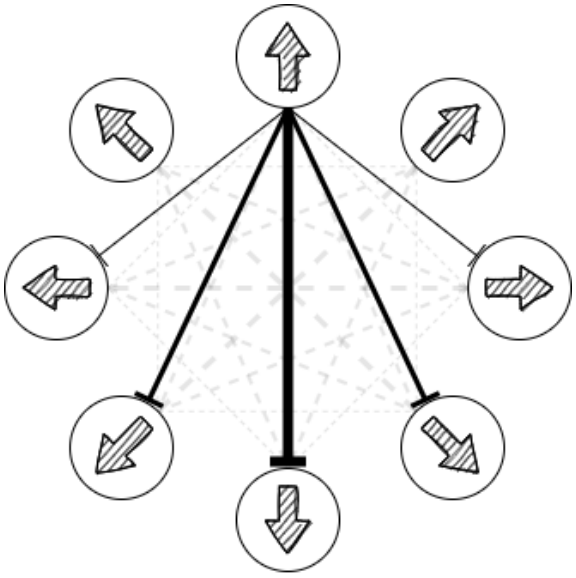


Figure 2.3: $\Delta 7$ neuron ring attractor. The figure illustrates how one neuron inhibits other neurons in the ring. The neuron opposite, which has the opposite preference angle, is the most inhibited, and the inhibition is weaker as the preference angle is closer to that of the neuron itself. All neurons in the ring exhibit this pattern.

2.2.2 Head direction system: ring neurons, EPG, and $\Delta 7$

For path integration, we need to be able to express movements in an *allocentric* coordinate system, i.e. one that is not relative to the current head direction (like the notions of north, east, south and west). The head direction system consists of ring neurons, EPG neurons, and $\Delta 7$ neurons that make up an internal compass, based on celestial cues such as polarized light from the sky. The ring neurons are also called compass neurons in [22], since they provide the actual compass input to the network. In the model, 16 ring neurons (of which there are a lot more in the actual brain) are each sensitive to one of the eight cardinal directions, in pairs of two. The compass input from the ring neurons pass via the EPG neurons and then connect to the eight $\Delta 7$ neurons.

The $\Delta 7$ are assumed to inherit the directional tunings from the EPG neurons, meaning they are sensitive to one direction each, separated by 45º (their preference angles). Apart from having forward connections, they also have recurrent connections between themselves which is visualized in figure 2.3, where the thickness of the line represent how weighted the connection is, the neuron tuned to the opposite direction being the most inhibited. With these properties, the $\Delta 7$ neurons are proposed to form a ring attractor, allowing for noisy compass inputs to generate a stable sinusoid pattern with a bump in the direction of the current head direction. This can be interpreted as a population-encoded unit vector representing the direction in which the head is pointing.

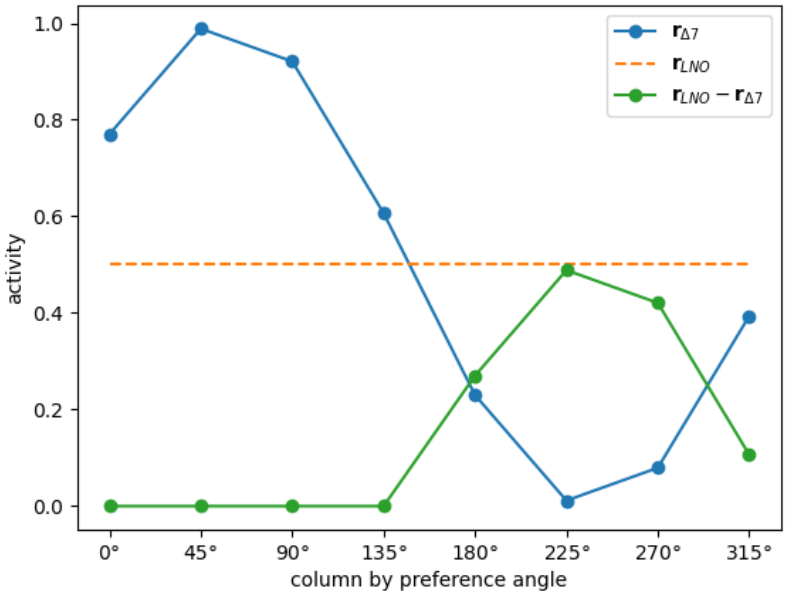


Figure 2.4: The computation performed by the PFN layer in the Stone model.

2.2.3 Memory: PFN

With the speed and the compass inputs, the egocentric motion vector can effectively be reprojected using the head direction system to construct a population-coded vector representing the allocentric velocity.

The PFN layer consists of eight neurons in each hemisphere (which is again a simplification, and each neuron really represents a column of many cells in the real brain), and is excited by the speed-encoding LNO neurons and inhibited by the $\Delta 7$ neurons, which lets them approximate a multiplication: when the speed input is 0 there is nothing to inhibit and the result is 0, and when it is 0.5 the result is an approximated sinusoid (the inverted head direction with its phase shifted 180^{o}) of amplitude 0.5. This happens separately in the different hemispheres: in the left hemisphere, the egocentric motion vector component in the forward-left direction is "multiplied" by the head direction vector to result in a population-coded vector representing the velocity in the forward-left direction, and vice versa in the right hemisphere. This results in a "distributed" representation of the actual velocity vector, which corresponds to the sum of the two hemispheres' vectors. Additionally, because the $\Delta 7$ neurons *inhibit* the PFNs, the vector is really inverted, representing the negative velocity (i.e. it points backwards). This computation is illustrated in figure 2.4, where the green line shows the allocentric population-coded motion vector.

Additionally, these cells are proposed to be the possible integrator cells, meaning that they accumulate the result of this computation and as such constitute the circuit's "memory". The end result is that the memory held in the PFN is a population-coded representation of the home vector, i.e. a vector that points towards the bee's nest, where the outbound journey started.

A figure of the memory representation can be seen in figure 2.5. The horizontal axis is the

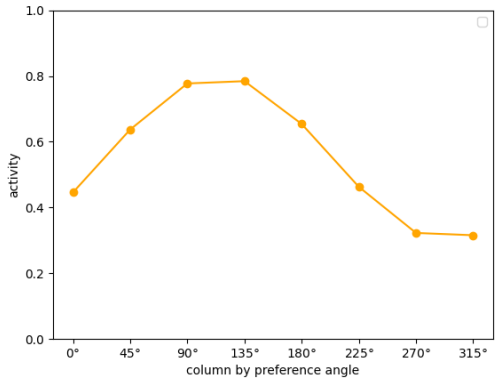


Figure 2.5: Conceptual memory representation. Both sets of 8 PFN cells hold one such representation each, one in each hemisphere.

different columns of PFN and the vertical axis represents activity. The sinusoidal bump will grow in amplitude as the bee forages further away from the nest, and then during homing the memory representation will rebalance and flatten out. Crucially, for our project, the Stone model implements the PFN cells as generic integrators (i.e. activity simply accumulates in a variable for each time step), and does not go into any detail concerning the mechanism by which they integrate their activity as the underlying biological mechanism is unknown.

2.2.4 Steering system: hΔ, PFL, and motor output

The memory of the home vector, i.e. the output from the PFN layer, connects with excitatory synapses to the steering system with a one-column shift (corresponding to 45°) to the left in one hemisphere, and to the right in the other. This input is inhibited by the head direction cells (the Δ7 output), which effectively compares the alignment between the two vectors: if the head direction is aligned with the home vector, the PFL cells in the two hemispheres will be, in total, equally inhibited. However, if the shift to the right results in a better alignment, that side will be more inhibited, resulting in a surplus of activity on the left side (and vice versa). The idea is illustrated in figure 2.6, where the green shaded area represents a surplus in the right hemisphere, driving steering to the right.

The sum of the PFL outputs on either side can thereby be used to generate a steering signal. When homing is activated in the simulation, it uses this steering signal to drive rotation of the agent (i.e. the virtual bee), which will cause it to turn toward the home vector until it aligns with the current head direction. When equally aligned in the left and right hemispheres, the steering system will no longer produce a steering signal. When the agent gets close to the nest, the memory will have seen the opposite activity and have levelled out, no longer having any discernible bump. At that point noise will dominate the system and a search pattern will emerge where the bee circles the nest.

A requirement for being able to perform the above comparison is that the memory bumps in the two hemispheres have the same mean. Since holonomic movement may result in an imbalance in the memory buildup between the hemispheres, this means that they need to be balanced before being compared. This is accomplished using a layer of hΔ pontine cells that shifts the PFN output by four columns (180°) and also inhibits the PFL layer, resulting in a

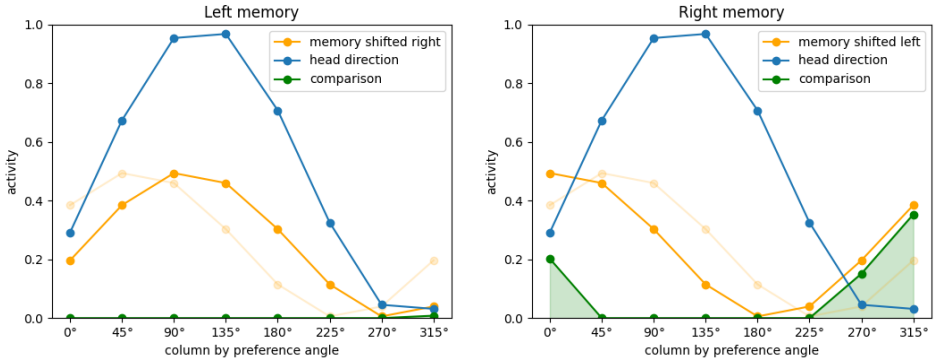


Figure 2.6: Illustration of comparison between home vector and current head direction in order to generate a steering signal.

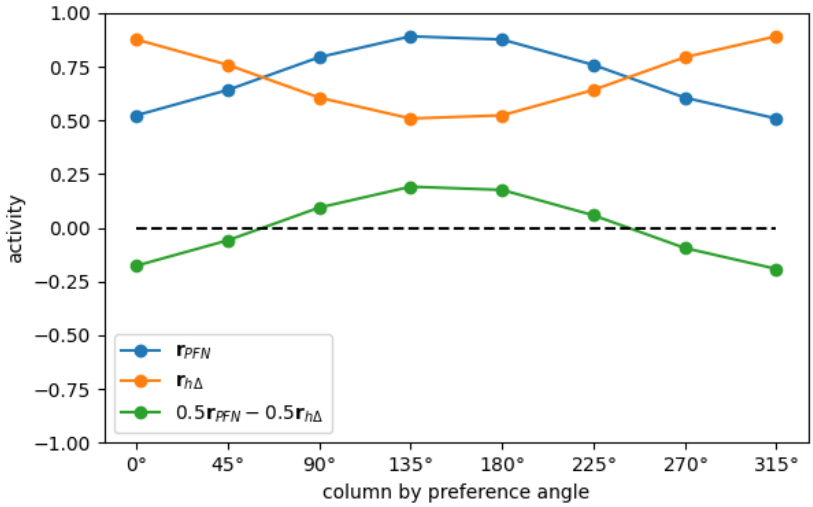


Figure 2.7: Illustration of memory balancing.

memory bump with a mean of zero. The effect of this balancing is illustrated by figure 2.7.

2.3 Physics

2.3.1 Nanowires

A nanowire is, as the name suggests, a structure in the size range of nanometers. Thanks to their small dimensions, and other preferential properties, semiconductor nanowires are highly considered for electronic, photonic and opto-electronic applications [11]. III-V nanowires is an especially mature platform of nanowires, that is versatile and can be created with a wide variation of electronic and optical properties [23]. The III-V in the name comes from the fact that they are made from materials found in the 3rd to 5th row in the periodic table [2].

The use of photonics for neural networks is something that has gained interest lately, as communicating with light is both efficient and fast[4]. Nanowires present a way to realize

this idea in a way with low power consumption and an incredibly small footprint, moving significantly closer to the power usage of an actual neuron as compared to traditional computers. Small neural networks such as the Stone model, which is based on an insect brain, were chosen as a good starting point for testing the potential of nanowires in this field. Insect brains are a good choice as they are comparably low in numerical complexity but still perform tasks that are very advanced, using very few neurons and incredibly low amounts of energy [23].

Optoelectronic neural units based on these III-V nanowires have been used to simulate the ring attractor part of the Stone model. The neural units communicate using light, and they have an LED in one end to output signals, and two photo-transistors that absorb light in the other end, one being excitatory and one being inhibitory. The neural units can add and subtract the inputs, process the result through a nonlinear activation function, and then output the appropriate amount of light from the LED. They are then put in a 2-dimensional waveguide where they can communicate without any wiring, further reducing the footprint of the network implementation [23].

2.3.2 Molecular dyes

A photoswitch is a molecule that can change its molecular structure, and in so affect its properties, when exposed to light. These kind of molecules have two different states: a thermodynamically stable state and a metastable photostationary state [8].

The candidate dye molecules for our project are, in their thermodynamically stable state, hereafter referred to as the ON state, opaque and absorb light. When a molecule absorbs a photon, there is a probability ϕ – known as the quantum yield – that the molecule will switch to its metastable photostationary state, the OFF state (otherwise the energy is simply lost as heat). In the OFF state, the molecules are transparent, but will after some exponentially distributed time spontaneously turn back into their ON state, meaning that molecules in their OFF state have a half-life $T_{\frac{1}{2}}$. This process of returning to the ON state is dependent on the temperature, but can also be affected by irradiation by light of a different wavelength [8].

For a long time the research surrounding photoswitches was surrounding molecules that required UV light to switch to the OFF (photostationary) state, which limited the possible applications due to the possible damage and limited penetration depth [7]. Recently more research has been done surrounding molecules sensitive to visible light wavelengths around 450-700nm [8], which also better coincides with the wavelengths of light that the III-V nanowires explored in [23] can operate at.

Chapter 3
Methods

As mentioned, we attempt to answer our primary research question first and foremost by way of simulation. To do so we create a simple framework in which we can easily define a variety of (primarily neural network-based) models of the central complex, make these control a virtual bee that moves according to rudimentary physics, and run trials to test path integration performance. The technical details of this framework are described in the first section below.

To quantify said performance we also need a suite of evaluation measures. The second and third sections describe how we can make sense of the memory and what measures we use, for evaluating individual trials, a model as a whole, and comparisons between models.

The mathematical model of the actual dye mechanics is described in the fourth section. Finally, we briefly discuss performing a grid search to find working parameter settings.

3.1 Framework

To run computational experiments with variations of the Stone model we created a framework in Python 3 with the following features (explained in the following sections):

- Flexible networks
- Simulation using simple physics
- Declarative experimental setups
- Separate data collection and analysis
- Usable both from command line and from Jupyter Notebooks

The main usage pattern is to define experiments using JSON to configure parameters. Such experiments can then be run using a command-line interface. The data collected can

then be analysed in a Jupyter Notebook environment. Notebooks were also used for creating many of the figures for this report.

The full framework code can be found on GitHub[1].

3.1.1 Neural networks

To be able to easily test different kinds of changes to the neural network, we implemented a flexible network model consisting of abstract `Network` and `Layer` classes. These can then be specialised depending on the kind of network. The work we present is mostly based on the `RecurrentForwardNetwork` implementation, which behaves like a feed-forward network but uses values from the last time step for recurrent connections.

The layers are typically `FunctionLayer`s that take some input, compute a weighted sum, and pass that sum through an activation function (as described in equations 2.1 and 2.2). These functional layers are mostly directly adapted from the Stone code[2], and use a sigmoid activation function with parameters a and b for slope and bias respectively. On top of this, we add normally distributed noise to reflect the noisy nature of real brains:

$$f(I) = \frac{1}{1 + \exp(-(aI - b))} + s, \text{ where } s \in \mathcal{N}(0, \text{noise}) \tag{3.1}$$

As firing rates are normalised to the interval $[0, 1]$, the noise can be directly interpreted as the proportion of the maximal (saturation) activity that is made up of noise. For example, a standard deviation of 0.1 represents a noise level of 10 %.

Additional layers that we have introduced are `MemorylessCPU4Layer`, which performs the PFN computation without integration, `PlasticWeightLayer` for the weight-based memory proof of concept, and `DyeLayer` for simulation of the dyes, as described below.

3.1.2 Central complex models

The `CentralComplex` model consists of a neural network as described above. For each time step it computes the inputs to the neural network based on current heading and velocity, in the form of head direction and speed neuron activity. With these inputs the neural network is then stepped forward one time unit. Finally, the motor output computed by the neural network is returned as the output.

3.1.3 Numerical methods

For solving differential equations we use the simple Euler method

$$y(t + \Delta t) = y(t) + \Delta t \frac{dy}{dt}(t, y(t)) \tag{3.2}$$

where Δt is the step size (typically 1 time step in our case) [21]. We found that this was sufficient for accuracy by comparing with SciPy's `solve_ivp`, while also being faster.

[1] Our GitHub repository: `https://github.com/nilsceberg/path-integration-memory`

[2] Stone's Github repository: `https://github.com/InsectRobotics/path-integration`

3.1.4 Simulation

Random outbound paths of length T_{outbound} are generated in the same way as in the Stone implementation: rotation speeds are sampled from a von Mises (circular normal) distribution and low-pass filtered to generate smooth turns. The acceleration is also randomly varied between a number of randomly selected key levels. The rotation speed ω and acceleration a, along with current heading θ and velocity $\mathbf{v}$, are then used by a simple physics simulation to compute the heading and velocity at the next time step like so

$$\theta(t + \Delta t) = \theta(t) + \omega \Delta t \tag{3.3}$$

$$\mathbf{v}(t + \Delta t) = (\mathbf{v}(t) + \begin{bmatrix} \sin\theta \\ \cos\theta \end{bmatrix} a\Delta t)(1 - d)^{\Delta t} \tag{3.4}$$

where d is a drag constant.

Alternatively, hand-crafted paths can also be specified using waypoints that the agent will navigate to in order.

The headings and velocities representing the random outbound path are fed to the central complex model, which steps the neural network as described above. For the outbound journey, the motor signal returned from the central complex is ignored.

For the homing phase, whose duration is T_{inbound}, the central complex is stepped forward in the same way. Acceleration is kept constant, and the motor signal produced by the CX is used as the turn speed (multiplied by some "motor factor") that is passed to the same rudimentary physics simulation as above.

The paths (headings and velocities), and optionally direct "recordings" of neural network layer outputs and internal states (for example the molecule concentrations in the dye layer), can then be saved to disk as a result file, or analysed immediately.

3.1.5 Analysis

For analysis, data is loaded from disk as a `SimulationResults` object. This class provides analysis methods, such as reconstruction of paths from velocities and headings, and error metrics used for evaluation, as will be described below.

3.1.6 Setups

The computational experiments run using this framework are specified declaratively using JSON configuration files. These files allow specifying for example the number of trials, length of the outbound and inbound phases, and what kind of network model to use for the CX, and its parameters.

An important feature is that we can specify ranges for parameters: for example, we might want to evaluate the performance at different noise levels. This can specified as `"noise": { "linspace": [0.1, 0.5, 5] }`, resulting in the same experiment being run for each of the noise levels 0.1, 0.2, 0.3, 0.4, and 0.5. We will use this for performing parameter searches, and the experiments used for our results are listed in Appendix A.

3.2 Constructing and decoding population vectors

A population vector code is constructed by simply projecting a Cartesian vector $\mathbf{v}$ onto the preference vectors $\mathbf{c}_i$. In other words, the value of unit i is

$$m_i = \mathbf{v} \cdot \mathbf{c}_i. \tag{3.5}$$

When $\mathbf{v}$ and $\mathbf{c}_i$ are unit vectors with angles θ and θ_i respectively, this is equivalent to

$$m_i = cos(\theta - \theta_i). \tag{3.6}$$

As firing rates cannot be negative, to capture the entire shape of this sinusoid it can be centered around 0.5:

$$m_i = 0.5 \cos(\theta - \theta_i) + 0.5. \tag{3.7}$$

Although the ring attractor output is a unit vector with the representation above, the computation in the PFN layer results in a sinusoid whose bottom half is clipped, which can be better approximated as[3]

$$m_i = [\cos(\theta - \theta_i)]_+ \tag{3.8}$$

and is the signal that we would like to integrate.

When exploring readout behaviour, we would like be able to give the network arbitrary home vectors. Due to noise and the specific behaviour dynamics of the memory the actual shape will likely depend somewhat on the specific path taken to get to a certain point. However, for this purpose we will be using idealised constructed memory vectors with an angle θ, amplitude A and a constant B like so:

$$m_i = A[cos(\theta - \theta_i)]_+ + B \tag{3.9}$$

These population vectors can be decoded in different ways. In the Stone model paper, it is done using a Fourier transform to get the phase and amplitude of the fundamental frequency. While this is useful, it does throw away higher-frequency information. To take the actual shape of the vector into account, we will use a simpler method known just as the *vector method* [3], whereby the linear combination of column preference vectors $\mathbf{c}_i$ using corresponding memory values m_i as coefficients:

$$\mathbf{v}_{\text{pop}} = \sum_{i=1}^{16} m_i \mathbf{c}_i. \tag{3.10}$$

The decoded vector is parallel to the vector that is represented. Due to the nonlinear memory we will be exploring, we consider the amplitude to be unimportant.

[3] $[x]_+ = \max(0, x)$

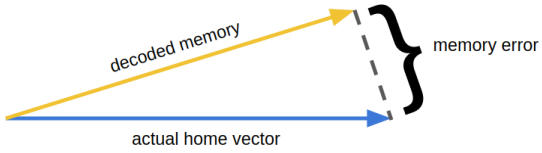


Figure 3.1: An illustration of the *memory error* metric.

3.3 Evaluation

To be able to quantitatively evaluate models and their variations and compare them against eachother, we establish a number of key metrics in this section. Using these, we also compile an evaluation suite which we will use to evaluate all models.

3.3.1 Error metrics

Evaluation of a model as a whole can be done by simply running simulations and examining metrics describing the resulting behaviour of the bee, such as closest distance to home achieved during the homing phase.

However, this will not reflect only the memory performance, but also that of the steering system. While we ultimately do want a memory that works well with the model of the steering system, it is also useful to evaluate the memory itself more directly (such as in section 4.2.3 on nonlinearity).

To this end, we use the vector method in equation 3.10 above to directly decode the memory. Because the steering system does not directly depend on the amplitude of the vector, but only the phase of the sinusoidal bump, we are only interested in the heading it represents. On the other hand, a simple angular error does not give an accurate picture, as the same angular error is much more catastrophic at a large distance from home. Therefore, we use the following metric: given a decoded heading θ_d, how close to home is the closest point along the ray determined by θ_d? This is illustrated geometrically in figure 3.1.

With the true angle to home θ_h and true distance to home d, we let $\alpha = \alpha(\theta_d, \theta_h)$ be the (positive) angular error. The memory error e (i.e. smallest distance to home along the ray) can then be computed as

$$e = \begin{cases} d \sin \alpha & \text{if } \alpha < \frac{\pi}{2} \\ d & \text{if } \alpha > \frac{\pi}{2}. \end{cases}$$

To evaluate the steering system, the quality of the memory readout, we also introduced the heading error. This is measured in the same way as the memory error, except it compares the head direction to the home vector. This metric is only relevant during the inbound path, since the steering is only active when homing.

Another interesting metric is the straightness of homing paths, which is referred to as *tortuosity*. This metric is used by Stone et al. [22], so we borrow their definition to allow comparison between results. We measure this by looking at the deviation from the best possible path when homing, i.e. the straight line path to home. With the straight line distance defined as L, and C as the distance covered toward the nest after L steps, the tortuosity met-

ric can be calculated as $T = \frac{L}{C}$. The optimal path home then has minimal tortuosity, $T = 1$, and paths that do not go as straight has tortuosity of $T > 1$.

From Stone [22] we also borrow a metric for how well aligned the bee is with the actual home vector after homing is engaged. This is done by measuring the angular deviation after crossing a 20 step radius from where the bee was when starting to home.

3.3.2 Evaluation suite

We developed a suite for evaluating a single model at a time, showing a comprehensive view of a few different error metrics. As an example we ran the neural model from Stone et al. [22] through the suite, which is presented in figure 3.2. The metrics that are included are explained earlier in this section. Figure 3.2 **(c)-(f)** show mean ± 1 standard deviation (SD).

3.4 Dyes

To simulate the plastic weights based on dye molecule dynamics, we use a simplified mathematical model in the form of a differential equation to describe how the concentration of the dye molecules in the two states changes, together with the light transmittance as a function of this concentration. The transmittance is the fraction of light that passes through the sample; i.e. given an incident (pre-synaptic in our terms) intensity I_0 and the attenuated (post-synaptic) intensity I, the transmittance T is defined by

$$I = T \cdot I_0 \tag{3.11}$$

and as such represents our weight $w = T$. In subsequent parts of this report, we will mostly stick to the notation w, except when talking about the transmittance as a function of concentration.

As described in 2.3.2, the dye molecules have two states which we call ON and OFF. The transmittance T is a function of the concentration of molecules in these states. The molecules that are not in the ON state are of course in the OFF state, and vice versa, i.e.

$$c_{\text{ON}} = c_{\text{tot}} - c_{\text{OFF}}, \tag{3.12}$$

where c_{tot} is the total concentration of dye molecules.

For brevity, we take a non-subscript c to mean the OFF-state concentration c_{OFF}. The change in this concentration is the result of two effects: the spontaneous back-reaction into the ON state and the switching to OFF in response to absorbed light:

$$\frac{dc}{dt} = -kc + u\phi(1 - T) \tag{3.13}$$

where

- $-kc$ represents the first-order back-reaction, and k is its rate coefficient (related to the half-life as $k = \frac{\log 2}{T_{\frac{1}{2}}}$),
- u is the PFN output, i.e. its normalised activity,

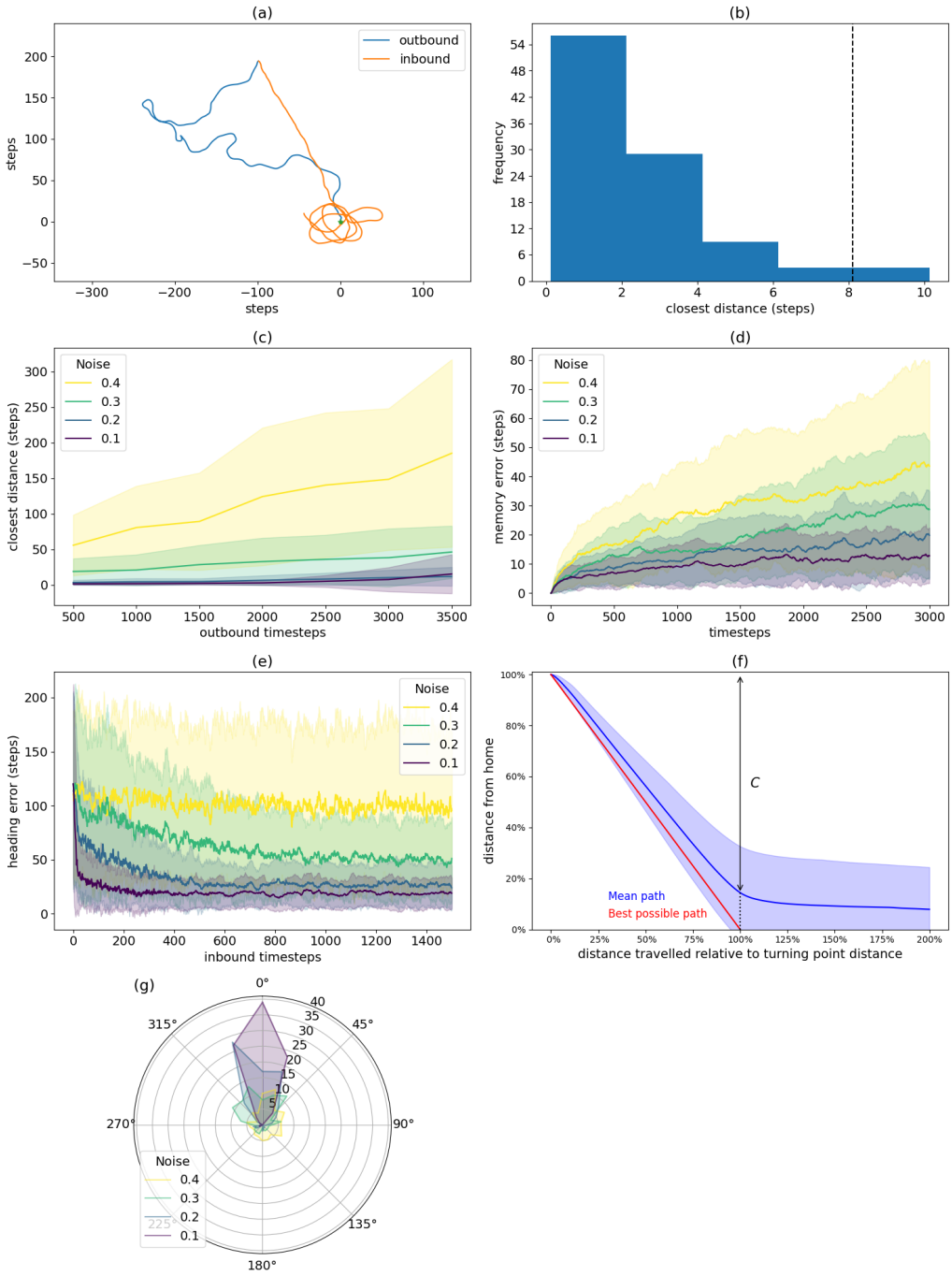


Figure 3.2: Evaluation suite for Stone model. **(c)-(f)** show mean ± 1 SD. **(a)** Example path. Outbound/inbound time = 1500 steps each. Noise=10%. **(b)** Closest distance histogram. N=100 runs. Outbound/inbound time = 1500 steps each. Noise=10%. **(c)** Closest distance for differing outbound length, at 4 noise levels. N=100 runs for each outbound time and noise level. **(d)** Memory error during outbound and inbound journey at 4 noise levels. Avg. over N=100 runs. **(e)** Heading error during inbound journey at 4 noise levels. Avg. over N=100 runs. **(f)** Deviation from best possible route for N=100 runs. Outbound/inbound time = 1500 steps each. Noise=10%. **(g)** Head direction deviation from actual home vector after crossing 20 step radius, after homing is engaged. Distribution over N=100 runs each for 4 noise levels.

- ϕ is the proportion of the absorbed light that leads to switching,
- T is the transmittance.

Note that u here is still dimensionless, and that ϕ is therefore not really the quantum yield (but we have co-opted the notation); it also has a built-in conversion factor from normalised light intensity to photons per second. Leaving u dimensionless means we generalise our results for varying time and power scales.

The transmittance T, in turn, is a function of the concentration of absorbent ON-state molecules c_{ON} as per the Beer-Lambert law. The absorbance A is

$$A = \epsilon l c_{\text{ON}} = \epsilon l (c_{\text{tot}} - c) \tag{3.14}$$

where

- ϵ is the molar absorption coefficient,
- l is the optical path length through the sample,

and finally the transmittance is expressed as

$$T = 10^{-A}. \tag{3.15}$$

This can further be combined into a single equation expressed as $\frac{dT}{dt}$, which we will do in the results to more directly relate the dye mechanics to weight memory. However, for numerical reasons the simulation is expressed in OFF-state dye concentration as its rate of change does not vary as much.

3.5 Parameter search

With the model above, we have five parameters, and in the results we will introduce a sixth: k, ϕ, ϵ, l, c_{tot}, and β (background activity).

Luckily, ϵ and l only ever occur as their product ϵl, which can thus be considered a single parameter. For the models where $k = 0$, we can disregard that as well. On the other hand, the optimal setting will also depend on parameters external to the model, such as outbound path length T_{outbound}. With this, we end up with a total of five parameters that are of interest to map out: ϕ, ϵl, c_{tot}, β, and T_{outbound}.

Ultimately, the purpose of the path integration circuit, when used for homing, is to help navigate home. With this in mind we mainly use closest distance to home as the performance metric for the search for optimal settings.

To perform the search, we run a brute force grid search using experiment setups using parameter ranges, with N trials (whose performance metrics are averaged, as there is stochasticity in both outbound paths and in noise in the network) for each combination of parameter settings. For parameters that span several orders of magnitude, such as ϕ, k and ϵl, we typically use logarithmic ranges for the parameters; the rest are linear ranges.

When plotting the closest distance metric with varying T_{outbound}, we normalise it by the "turning point distance" (i.e. the distance from the point where homing was activated to

home) which is on average proportional to the outbound path length, to avoid a bias in favor of shorter outbound paths.

Since there are more parameters than spatial dimensions, visualising them all in one plot is difficult; however, as we gain a good understanding of roughly how each parameter affects the others, we do not find that strictly necessary. To fit as many parameters into a single plot as possible, we use 2- or 3-dimensional heat maps with the color representing performance, and the spatial axes the parameters.

Chapter 4
Results

In this chapter we present our results, beginning with the conceptual changes whereby we move the memory from integrator neurons to the synaptic weights interconnecting them, and then going on to implementing this with the dye-based dynamics, including insights regarding further improvements and the consequences for behaviour. These first two parts employ mostly qualitative reasoning and reflect our process in reaching these results. The last section finally presents a quantitative comparison between the Stone model and our models.

4.1 Memory in plastic synaptic weights

In the Stone model, the PFN layer plays two roles: firstly, it "reprojects" the egocentric Cartesian self-motion inputs onto an allocentric population-coded vector, using the head-direction input state from the $\Delta 7$ ring attractor. Secondly, it integrates this information and thus also functions as the memory. Figure 4.1 zooms in on the PFN layer in the Stone model.

As a first step in the process of implementing the memory using dye molecules, we would like to separate these roles. The PFN layer is thus only tasked with performing the reprojection. The second role is instead filled by a second layer whose output is strictly its inputs multiplied by a set of weight – however, the weights themselves are also changed in the process. The dynamics of how these weights change depend on the type of memory.

One important consequence of making this change is that the activation function on the PFN output in the Stone model is now applied to the *integrand*, as opposed to the integral. This activation function approximates a rectified linear activation function by having a slope of $\mathbf{5}$ and a bias of $\mathbf{2.5}$, as well as being noisy. When applied to the integral as in Stone, this error does not accumulate, but as we apply it before the integration, it does.

Figure 4.2 demonstrates the conceptual design we have concentrated on, where the integration is done by plastic weights on the output of the PFN layer.

The dye mechanics, along with the fact that we would like to remain somewhat consistent with anatomy if possible, place the following constraints on how this may work:

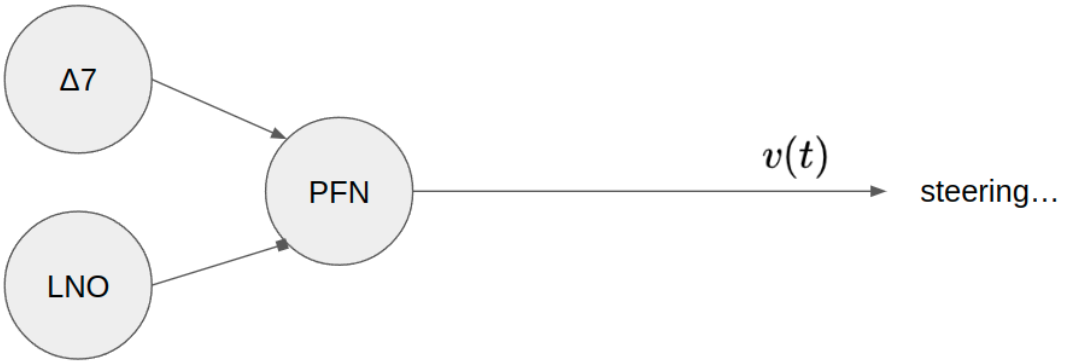


Figure 4.1: The location of the PFN cells in the Stone model.

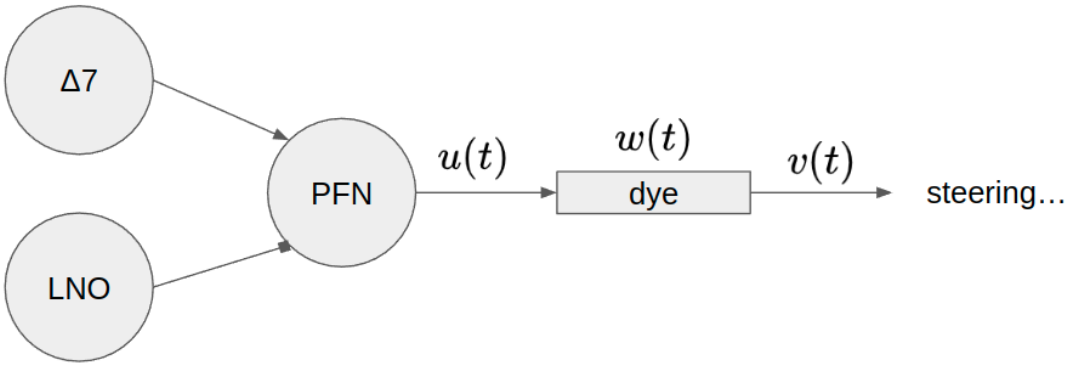


Figure 4.2: Variation of the Stone model with plastic weights between the PFN and PFL/hΔ layers ("steering").

1. The weights can only attenuate the signal, i.e. each weight must be between 0 and 1.
2. The weights must be "self-modulating", i.e. the signal passed through a weight must also be the one used to modulate it.

Constraint 2 in particular poses a fundamental problem to consider: the same signal must be used for both modulation and readout. Specifically, if $u_i(t)$ is the pre-synaptic activity for column i (i.e. the firing rate of one PFN neuron, $u_i(t) = r_{\mathrm{PFN}_i}(t)$), then the corresponding weight $w_i(t)$ changes as a function of $u_i(t)$ and its current value:

$$\frac{dw_i}{dt} = f(u_i(t), w_i(t)) \tag{4.1}$$

The synaptic current contribution to downstream neurons $v_i(t)$ (PFL and hΔ input) is then the product of $u_i(t)$ and $w_i(t)$:

$$v_i(t) = w_i(t)u_i(t) \tag{4.2}$$

In other words, since it is the weight that represents the memory, we cannot read the memory directly – we can only see the result of the multiplication of the signal by the weight, but we also cannot choose the signal freely.

In previous work, a weight-based memory in a Stone-inspired model has been accomplished by modulating the weights between the head direction layer and the steering layer with a one-column shift to the left and right (depending on hemisphere), which results in the weighted signal effectively being a measure of whether a left or right turn would result in better alignment with the home vector [5]. This cannot be done when adhering to this second constraint, and thus a somewhat different approach is required.

4.1.1 Readout

As the buildup of the dye transmittance is a form of integration, our focus is on how to get a working readout without destroying the memory. This section will outline a few different ideas on how this might be accomplished, leading onto the next section (4.1.2) where we construct a proof of concept of weight-based memory with linear dynamics.

Compensation

An initial, simple idea would be to use some kind of compensating process to account for the varying u: since $v = wu$, $\hat{v} = \frac{v}{u} = w$ would be a perfect readout of the weights.

This would require some kind of computational unit that takes u and v as separate inputs and effectively computes their quotient. The idealised model of a neuron from 2.1.1 cannot do this, as its inputs are simply summed. However, in the work implementing the ring attractor subnetwork using a model of nanowire units [23], the excitatory and inhibitory inputs to a neuron are really two separate input channels. While they approximate the model neuron, the total synaptic current does not strictly depend only on the sum of the inputs, but on the absolute values of the excitatory and inhibitory inputs, resulting in a 2-dimensional function I of the inputs (figure S9 in supplemental information for [23]).

It might be possible to tune this function differently in order to achieve the required compensation, which would look something like figure 4.3. However, since it is not clear

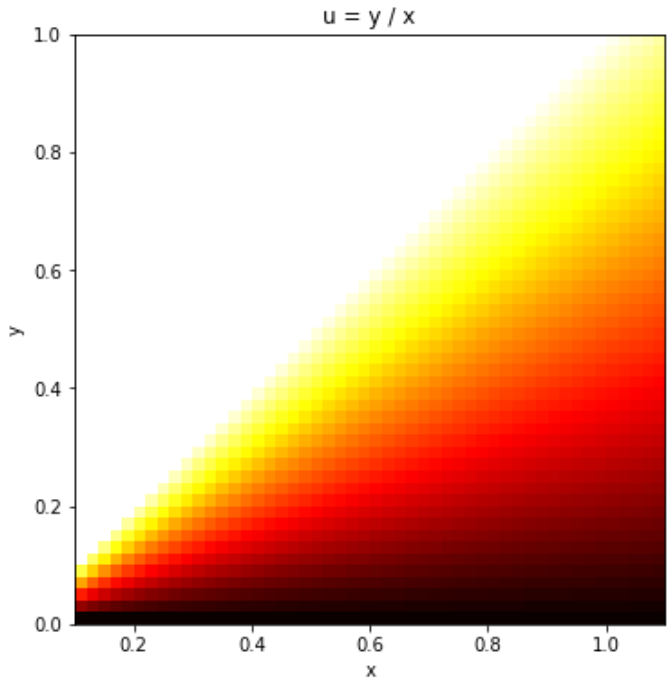


Figure 4.3: A two-dimensional activation function depending on two inputs, x and y, computing their quotient.

whether or not this can actually be done, we have left this idea here and not explored it further.

Alignment

While we cannot get a readout that itself represents whether a left or right turn would result in a better alignment as in [5], the output *is* in some sense a measure of alignment. An instantaneous readout is difficult to make sense of, as there is nothing to compare it to (the two hemispheres perform essentially the same calculation).

However, the alignment might in a sense be compared over time, by letting high alignment of the backwards-pointing PFN vector and the home vector result in faster turning. This should presumably result in a tendency to move towards the nest, although it seems unlikely to produce any sort of straight paths. An initial attempt at implementing this solution (figure 4.4) is in line with this expectation.

While this would make the phase comparison system accomplished by column offsets and the PFL layer in the Stone model superfluous, it does not strictly speaking violate its anatomical constraints as it simply does away with a part of the network. Still, our primary concern is modelling memory that could work as input to that steering system, so this idea was also left without further consideration.

Background activity

With r_{PFN_i} denoting the PFN output activity (that we would like to integrate), we can introduce $u_i = r_{\mathrm{PFN}_i} + \beta$, where u_i is the signal that is actually passed through the weights and

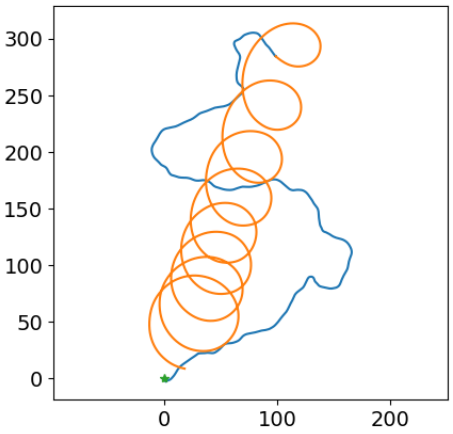


Figure 4.4: Resulting homing when using only the sum of weighted PFN outputs as the motor output.

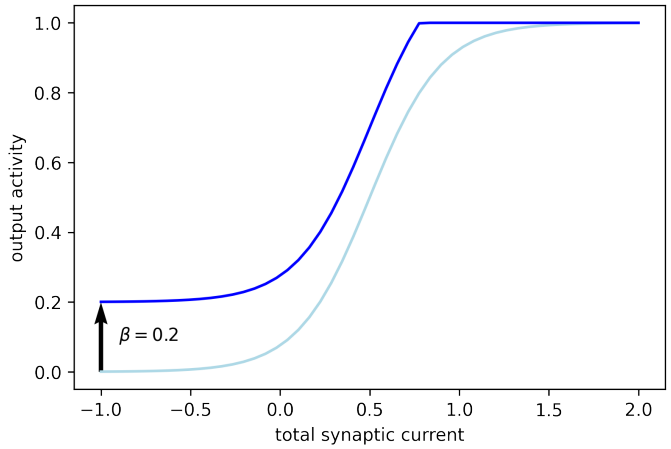


Figure 4.5: Activation function for PFN layer with background activity.

includes some *background activity* β.

If $\beta \approx 1$, i.e. completely dominates the signal, then $v_i = w_i u_i \approx w_i$. In general,

$$v_i = w_i u_i = w_i \beta + w_i r_{\mathrm{PFN}_i}, \tag{4.3}$$

meaning that a portion of the readout signal is proportional to the weights, with a known proportionality constant (β). The other part of the signal, $w_i r_{\mathrm{PFN}_i}$, would, in this view, constitute noise.

This sort of background activity corresponds to shifting the whole activation function for the PFN layer upwards, as is illustrated in figure 4.5. We still clip the output to between **0** and **1**, meaning that if the background activity is too high in relation to the actual signal, information will be lost. This may or may not be realistic based on how the background activity is actually implemented, but should be the most conservative assumption.

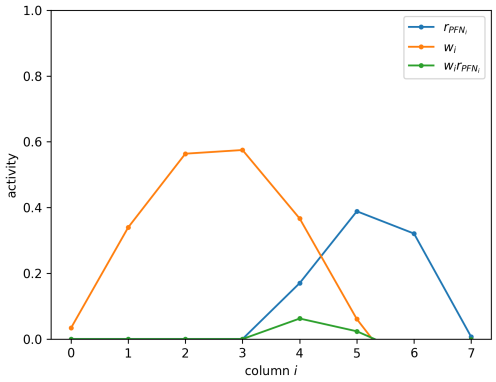


Figure 4.6: Output phase-shifted toward home vector. Note that there are only 8 data points; the linear interpolation between the points is only to better illustrate the sinusoid shape.

Phase shift

Consider that since all that is really needed in order to turn in the right direction is a motor signal of the correct sign, it should in theory be enough to get a memory readout that is shifted from the current head direction *toward* the home vector. In terms of sinusoids, the only important property is the phase of the memory bump's fundamental frequency, not its exact shape or amplitude.

When the PFN output is multiplied by the synaptic weight, the weighted activity of each column depends on how aligned the (inverted) head-direction bump and the memory bump are. If they are close to perfectly aligned (or exactly opposite), the resulting bump is at the same location. If they are misaligned, though, the greatest activity will be in the columns with the greatest overlap, essentially resulting in a shift in the phase towards the phase of the home vector. Figure 4.6 illustrates this idea with a simplified model of the memory and head direction bumps.

However, as is evident in the figure, the output signal is often very weak. Additionally, when close to aligned, there is already a lot of overlap, and the shift is less pronounced. By giving the system an arbitrary home vector, such as the most pronounced one possible constructed using equation 3.9 (i.e. $A = 1, B = 0$), we can plot the motor output as a function of the misalignment, which is done in figure 4.7. Indeed, this figure seems to even suggest that the motor output has entirely the wrong sign at some misalignments, and plateaus when close to being aligned. We shall see soon that this readout does in fact result in a tendency to travel towards home, but not very reliably.

4.1.2 Proof of concept with linear dynamics

To begin evaluating the readout, we consider a memory that accumulates in constant proportion ϕ to its input u, and also has a constant decay of $-k\phi$ (i.e. an appropriate function f for equation 4.1):

$$\frac{dw_i}{dt} = \phi(-k + u_i(t)) \tag{4.4}$$

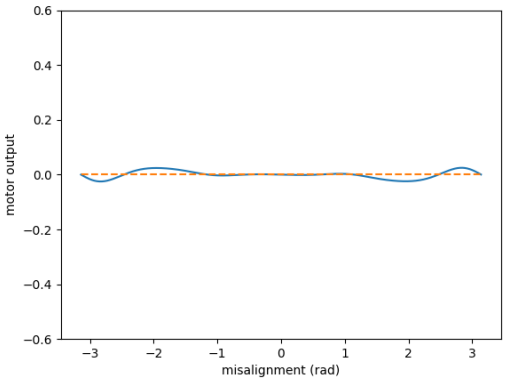


Figure 4.7: Plot of motor output as a function of misalignment.

To achieve memory dynamics equivalent to those in the original model, we simply let ϕ be equal to the memory gain of 0.0025, and k be the constant decay factor of 0.125 – numbers that will, roughly, keep the memory balanced around the starting point, which we set to 0.5 (the middle of the range).

By making no other modifications to the network than separating the roles of the PFN layer as described above, we can test the performance of the readout based only on the phase shift effect described above. This results in an example path that is visualised in figure 4.8. The path is not very straight, but it does very clearly go in the direction of home, which is a promising first result.

While relying only on the phase shift may not result in an entirely satisfactory readout, it does mean that the $w_i r_{\text{PFN}_i}$ portion is not merely useless noise and thus an approach using background activity does not actually require that β dominate the PFN output. Instead, we can choose a background activity that results in a better readout, but does not result in the actual information needing to be exceedingly small in amplitude. In the same way that a background activity-less readout would result in a bump close to the PFN bump but slightly shifted towards the memory bump, a readout with background activity can be understood as resulting in a bump close to the memory bump but shifted slightly towards the PFN bump.

With a background activity, even if small, the signal is a lot less weak, as it does not depend solely on the overlap of the two bumps. In effect, background activity should be better for the readout, as is illustrated figure 4.9 as compared to figure 4.6. Again, using the same arbitrary home vector as earlier, we can plot motor output as a function of misalignment, but also include the effects of β, as in figure 4.10 which suggests that the most pronounced motor signal is achieved around $\beta = 0.3$.

Tweaking the PFN activation function as in figure 4.5 with $\beta = 0.5$ and balancing k so that $-k + \beta = -0.125$, are minor changes that result in homing that looks like 4.11.

This is a great improvement and looks comparable to the paths produced by the Stone model (in section 4.3 we will compare the performances quantitatively). Finding that it is possible to get a readout that is usable for steering by using background activity, it is evident that a plasticity-based memory can be used in the existing model without any other modifications to the network. As such, this is what we consider our "baseline" model going into the next section: dye-based dynamics.

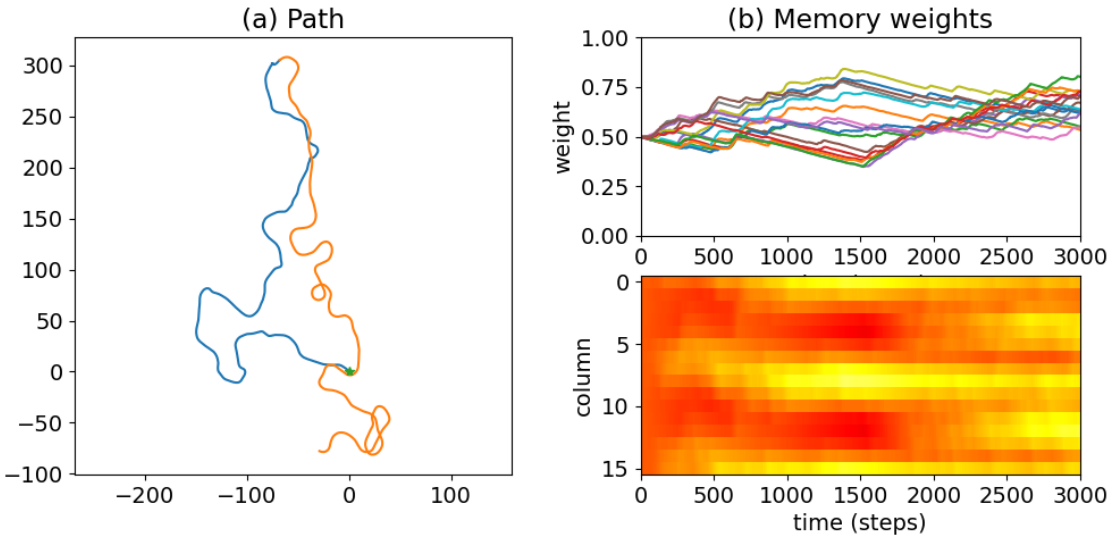


Figure 4.8: **(a)** Pathing with only phase shift readout. The blue line is the outbound path, and the orange line is the inbound path. The green star represents home. **(b)** Recorded memory weights over time. In the upper plot, each colored line represents one column. The lower plot shows the same weights but as a heat map where the columns are laid out along the vertical axis. Note how the points where the agent is close to home can be identified by the smaller differences in activity between columns (such as around $t \approx 0$ and $t \approx 2400$).

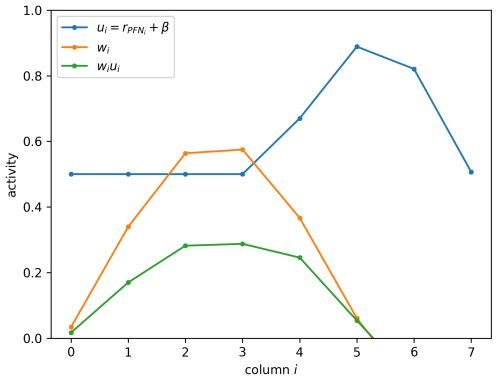


Figure 4.9: Readout with background activity $\beta = 0.5$; cf. figure 4.6.

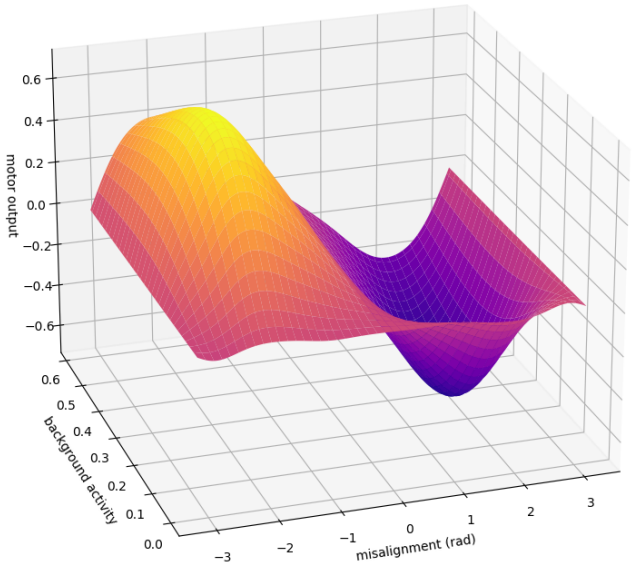


Figure 4.10: Motor output as function of misalignment and background activity.

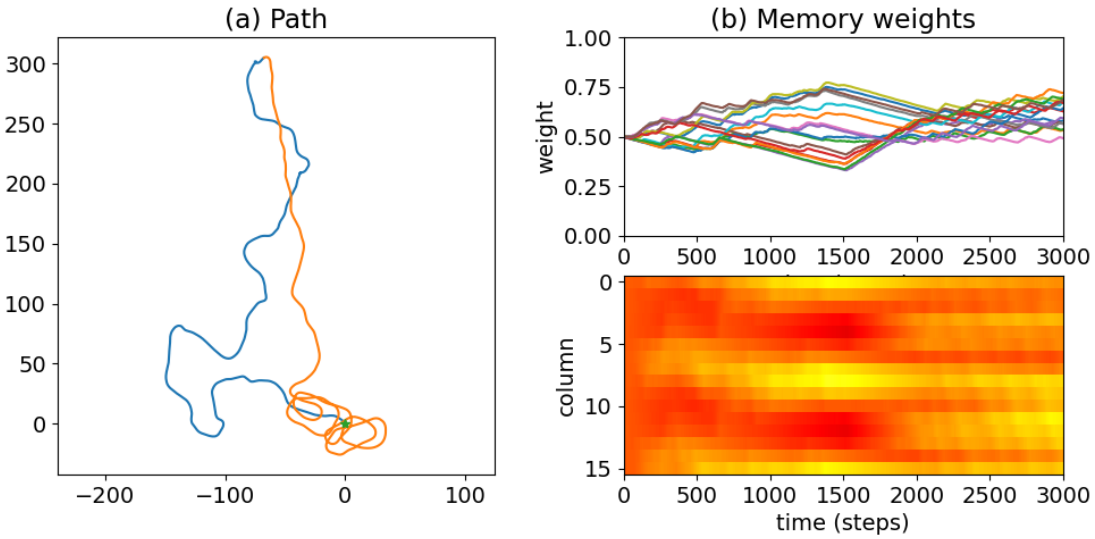


Figure 4.11: **(a)** Pathing with background activity. **(b)** Recorded memory weights. See figure 4.8 for elaborated figure text.

4.2 Dye-based plastic synaptic weights

The full dye model described in 3.4 can be rewritten on the same form as equation 4.1 (again, $w = T$):

$$\frac{dw_i}{dt} = \ln(10) \cdot \epsilon \cdot l \cdot w_i \cdot (-k \cdot c(w_i) + \phi \cdot u_i \cdot (1 - w_i)) \tag{4.5}$$

With a weight layer that is based on these mechanics, we are presented with some additional constraints.

First, the memory decay is no longer constant. In the dye-based model, the decay of the weight corresponds to the decreased transmittance as molecules that had been switched to their transparent OFF state spontaneously switch back to their opaque ON state. Since this back-reaction is stochastic with a constant probability of switching back per time unit, the decrease in OFF-state concentration is proportional to that same concentration (i.e. exponential decay). In effect, the memory of movements that occurred earlier is weaker than memory of movements that occurred later.

Second, the memory also no longer accumulates in constant proportion to the input. The accumulation corresponds to the increase in OFF-state concentration due to photons being absorbed by the ON-state molecules. Thus, as more molecules become transparent, less are available to absorb a photon and switch into the OFF-state; in other words, the accumulation of memory becomes slower and slower.

Both of these facts have the consequence that we can no longer balance the accumulation against the decay across the whole range of concentrations. For this reason, we have focused on the case where $k = 0$, i.e. the molecules switch to their OFF state semi-permanently (requiring some external reset, for example by increasing the surrounding temperature to increase the decay rate). This comes with the obvious drawback that the memory will eventually reach saturation, as it only ever accumulates. The PFN background activity, which is all but required for a decent readout, is especially problematic in this regard, as it means that there is some max time period after which the memory will have become saturated from β alone, even if the bee stood still the entire time.

Furthermore, even after disregarding k, the fact remains that the change in concentration and transmittance is a nonlinear process. The transmittance changes very little in the beginning but soon goes faster and faster until a plateau is reached near 100% transmittance. Which behaviour dominates depends on the parameters, especially ϵl; typically, transmittance over time (i.e. the solution to equation 4.5 above) takes the shape of a sigmoid: it remains low until the concentration hits a soft threshold value where it starts to increase rapidly until most molecules are transparent and a plateau (saturation) is reached.

While the concentration begins to change immediately, the memory it represents is essentially hidden and cannot be read out until it reaches the soft threshold where the transmittance actually starts to increase. Consequently, there is a window between the point where transmittance starts to noticeable increase and the point of saturation where homing actually works.

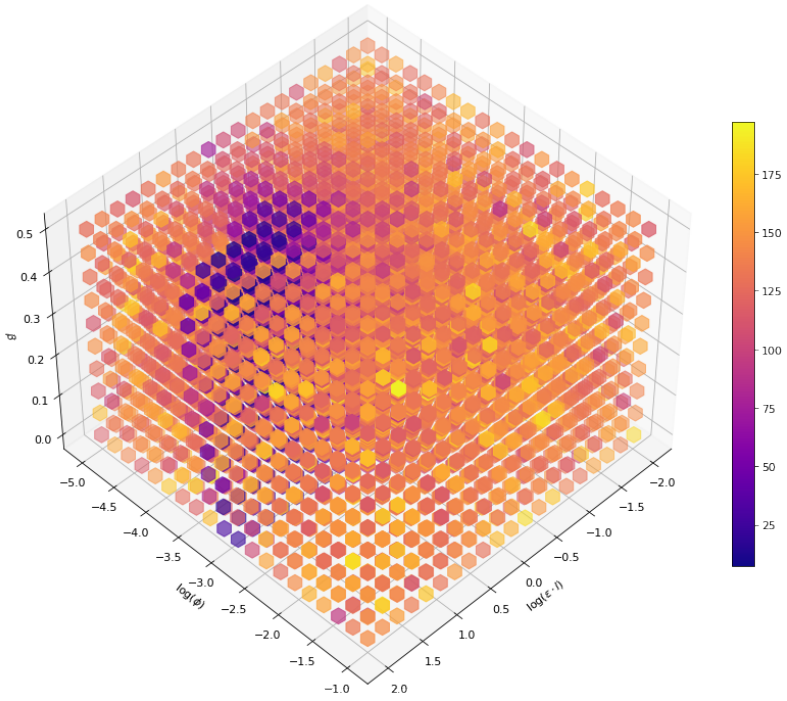


Figure 4.12: Closest distance to home as function of $(\epsilon l, \phi, \beta)$ (smaller is better).

4.2.1 Parameter search

With the same idea as in section 4.1.2, we can run simulations to map out the performance across the parameter space using a grid search. Fixing $k = 0$, $c_{\text{tot}} = 0.3$ (the maximum realistic value), and $T_{\text{outbound}} = 1500$, we map in figure 4.12 the performance as a function of $(\beta, \phi, \epsilon l)$.

This map reveals a sliver of deep violet where the agent manages to get home. The minimum point in this search was approximately $(\epsilon l, \phi, \beta) \approx (15, 0.0003, 0.3)$ resulting in a closest distance to home of about 7.2 distance units. The clearest relationship is that between ϕ and β: a higher ϕ is good for maximising utilisation of the memory capacity, while a higher β is better for the readout. However, both contribute to saturation of the memory. Maximizing performance at only one T_{outbound} thus runs the risk of "overfitting" to that path length, by resulting in the maximum $\beta\phi$ that does not saturate too quickly, thus not working well for longer paths.

By instead varying T_{outbound} we can compare the effects of different levels of background activity on the path lengths that can be handled; this is plotted in figure 4.13. For a small β the readout is worse and it takes longer before the readout window is reached and performance is therefore bad for short paths, but saturation is not a problem. A larger β has the opposite issues.

Using these parameters we get a dye model that does manage to get back home, with an example trial in figure 4.14. Again, a quantitative comparison will be done in section 4.3, but we consider this model to *barely* work: its paths are not very straight, it is sensitive to parameter changes (the sliver of violet is thin), it is near the saturation limit at the end of the example trial, and it is not very good close to the edges of the readout window.

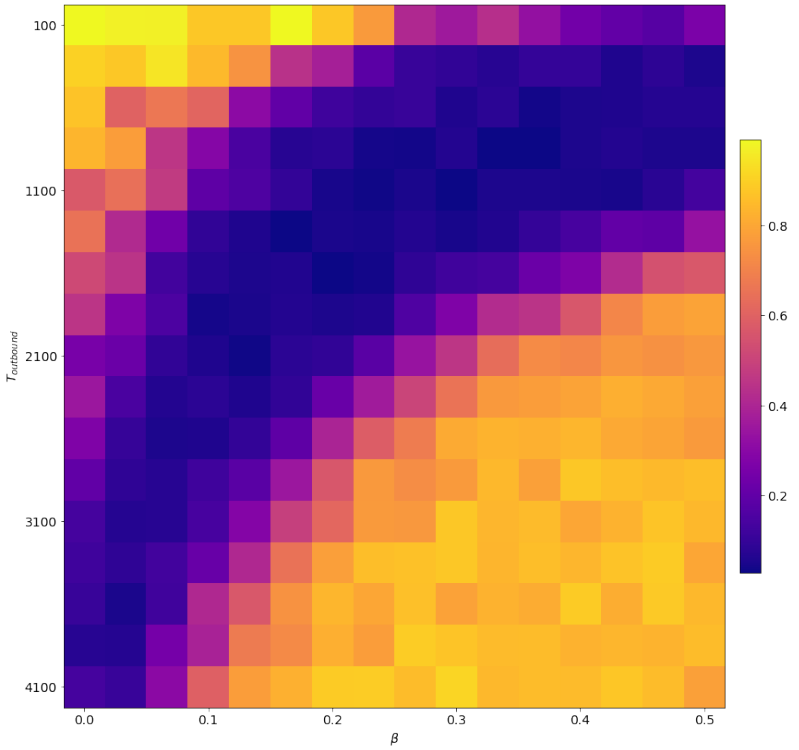


Figure 4.13: Normalised closest distance to home as a function of $(\beta, T_{\text{outbound}})$.

To be able to also handle longer paths and have a wider readout window, we can manually adjust these parameters by lowering the background activity and ϵl and increasing ϕ. With $(\epsilon l, \phi, \beta) = (10, 0.00045, 0.1)$ we remain in the violet area, and get an example trial depicted in figure 4.15. Note how the memory is not as close to saturation.

4.2.2 Further improvements

Variable background activity

The background activity that is used to get a decent readout above is, as mentioned, problematic since it contributes to quicker saturation of the memory. However, since it is really only required for the homing phase, it can be disabled during the outbound journey without affecting the readout during homing. In fact, this instead leaves us with more room to use a larger β during the homing phase, improving performance (see figure 4.16).

While this is conceptually fairly trivial, it does place some additional constraints on how the background activity can be implemented (i.e. it needs to be dynamically controllable). More generally, background activity could be activated only periodically (e.g. a sinusoidal function of time) to reduce its accumulating effect while still resulting in net homeward turns.

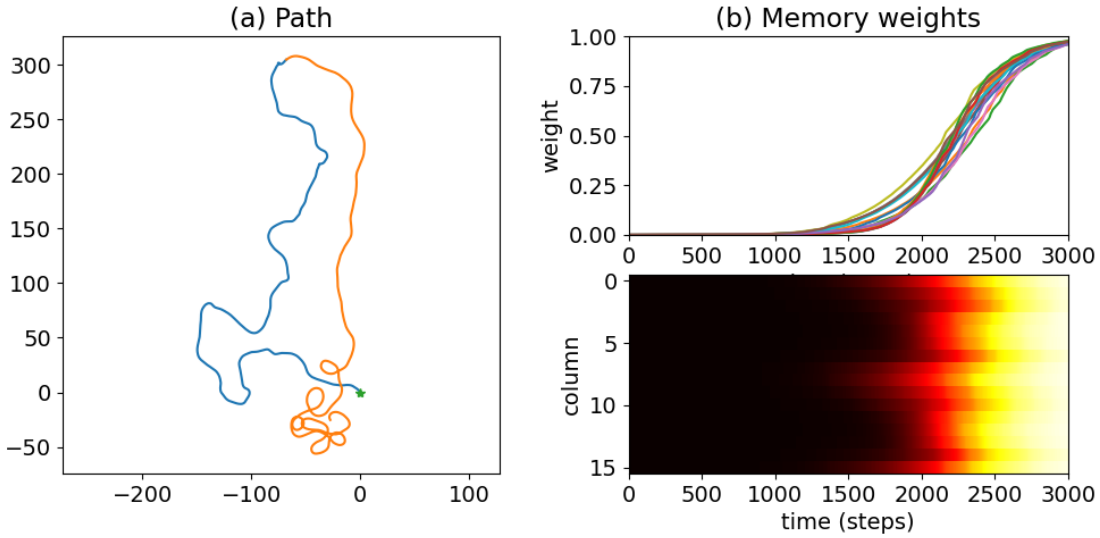


Figure 4.14: **(a)** Dye model pathing with grid search parameters $(\epsilon l, \phi, \beta) = (15, 0.0003, 0.3)$. **(b)** Recorded memory weights (transmittances). See figure 4.8 for elaborated figure text.

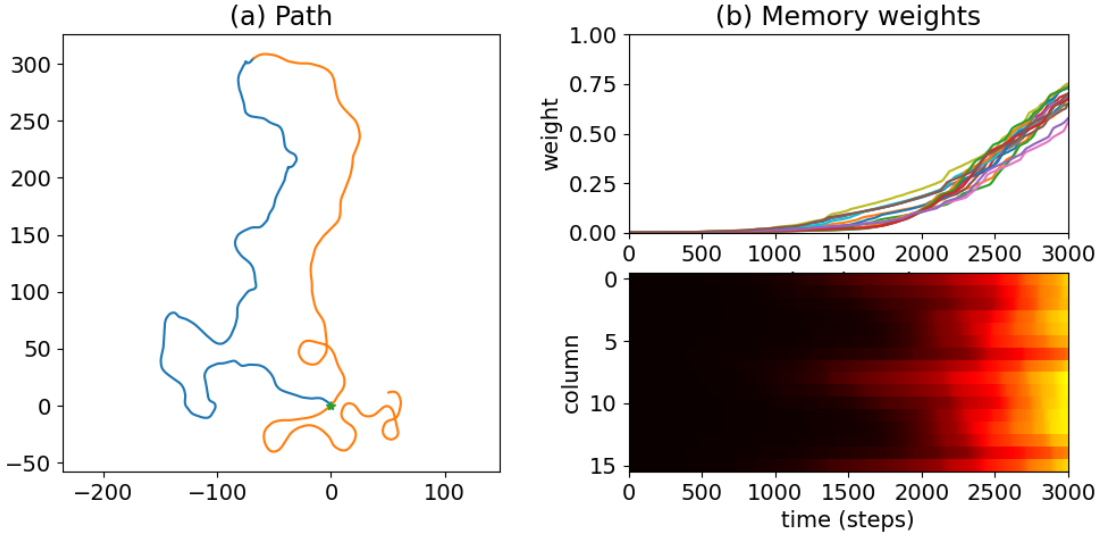


Figure 4.15: **(a)** Dye model pathing with adjusted parameters $(\epsilon l, \phi, \beta) = (10, 0.00045, 0.1)$. **(b)** Recorded memory weights (transmittances). See figure 4.8 for elaborated figure text.

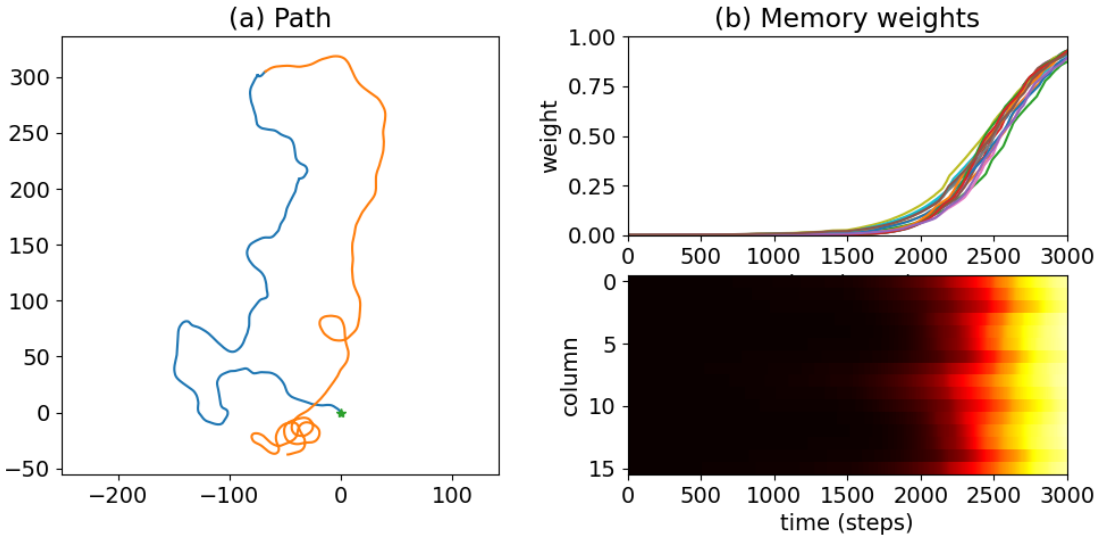


Figure 4.16: (a) A dye model trial with beta activated at $T = 1500$ layer. **(b)** Recorded memory weights. See figure 4.8 for elaborated figure text.

Amplification layer

Even with the PFN background activity, the readout signal can be quite weak, in the sense that the differences between peaks and troughs are small enough to make the bump indistinguishable. A sufficiently steep activation function with a bias corresponding to the mean of the sinusoidal bump could be used to amplify those differences. Unfortunately, that mean is not constant, and depends on the current state of the memory.

However, the PFL layer in the original Stone model computes $\mathbf{r}_{\text{PFN}} - \mathbf{r}_{\text{h}\Delta}$, before this result is compared to the current head direction. This computation results in the average being shifted to 0, in order to balance the memory bumps between the two hemispheres so that they can be compared, as described in section 2.2.4. This would be the perfect place to do the amplification of the difference between peaks and troughs, as this balancing eliminates the need for a bias: it simply needs to amplify positive values and silence negative ones. The amplification is illustrated in figure 4.19.

The full PFL layer computation is

$$\mathbf{r}_{\text{PFL}} = f(0.5W_{\text{PFN-PFL}}\mathbf{r}_{\text{PFN}} - 0.5W_{\text{h}\Delta\text{-PFL}}\mathbf{r}_{\text{h}\Delta} - W_{\Delta7\text{-PFL}}\mathbf{r}_{\Delta7}). \tag{4.6}$$

The amplification described above can be accomplished by adding a amplification layer between the h$\Delta$and PFL layers, with a very steep activation function g like so:

$$\mathbf{r}_{\text{amp}} = g(0.5W_{\text{PFN-amp}}\mathbf{r}_{\text{PFN}} - 0.5W_{\text{h}\Delta\text{-amp}}\mathbf{r}_{\text{h}\Delta}) \tag{4.7}$$

$$\mathbf{r}_{\text{PFL}} = f(W_{\text{amp-PFL}}\mathbf{r}_{\text{amp}} - W_{\Delta7\text{-PFL}}\mathbf{r}_{\Delta7}) \tag{4.8}$$

Adding such an amplifying layer significantly improves performance, as can be seen in the trial in figure 4.17. The paths are straightened and the window is widened, as will be seen in the quantitative evaluation, and the parameters are less sensitive, which is illustrated

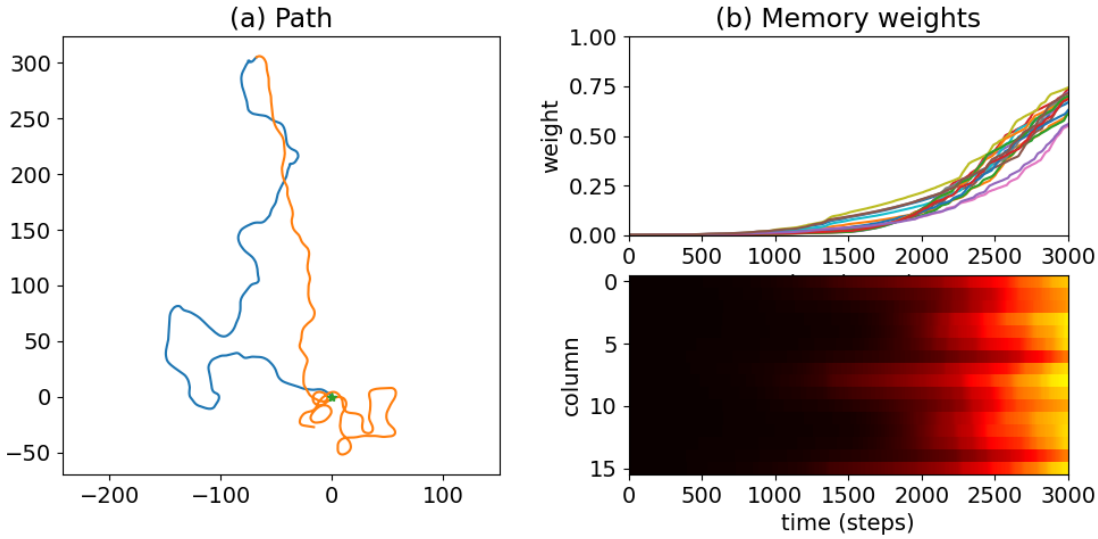


Figure 4.17: (a) A dye model trial with the pontine amplification layer. **(b)** Recorded memory weights. See figure 4.8 for elaborated figure text.

by the increased volume of the violet area of high performance in figure 4.18 compared with figure 4.12 from earlier.

4.2.3 Non-linearity

A classical example from control theory involves regulating the water level in a tank using a pump [12]. If the tank has vertical sides, then pumping in some fixed amount of water will have the same effect on the water level regardless of the starting point. This is a linear system. If, instead, the tank is in the shape of an inverted cone, the same amount of water will have a very large effect on the water level if the tank is nearly empty, but a very small effect if the tank is close to full – thus, the system is nonlinear. The transmittance of our memory units works similarly, and can be compared to the water level in an hourglass-shaped tank.

Figure 4.20 illustrates the memory dynamics (OFF-state concentration and transmittance) at typical (constant) PFN activities, and figure 4.17 shows the transmittances of each dye unit during an actual simulation.

A key property of linear systems is *additivity*. If we denote the memory of a vector $\mathbf{a}$ by $M(\mathbf{a})$, then a linear memory would, by additivity, mean that the memory of location $\mathbf{a}$ and the memory of location $\mathbf{b}$ can be summed to represent the memory of location $\mathbf{a} + \mathbf{b}$:

$$M(\mathbf{a}) + M(\mathbf{b}) = M(\mathbf{a} + \mathbf{b}). \tag{4.9}$$

This *seems* like a very important property for performing path integration. For example, moving θ degrees along the arc of a circle with radius r_1 needs to have the same effect on the home vector phase as moving the same angular distance along a circle of a different radius r_2.

Still, when running simulations with the described nonlinear memory dynamics, the integration does seem to be able to capture these movements correctly. After all, for the memory to become rebalanced, all columns need to see the same total activity. As such, it appears that

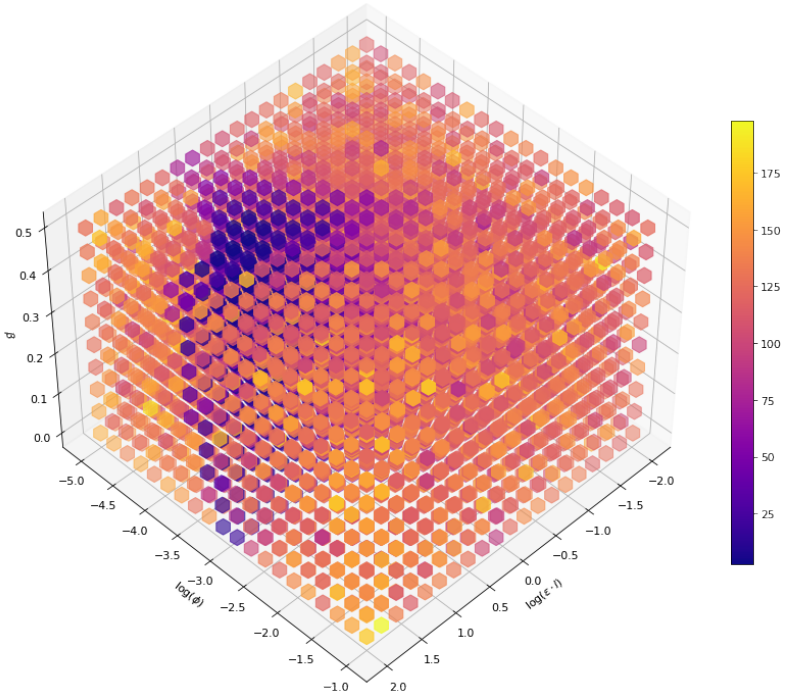


Figure 4.18: Closest distance to home as function of $(\epsilon l, \phi, \beta)$ using the amplification layer (smaller is better).

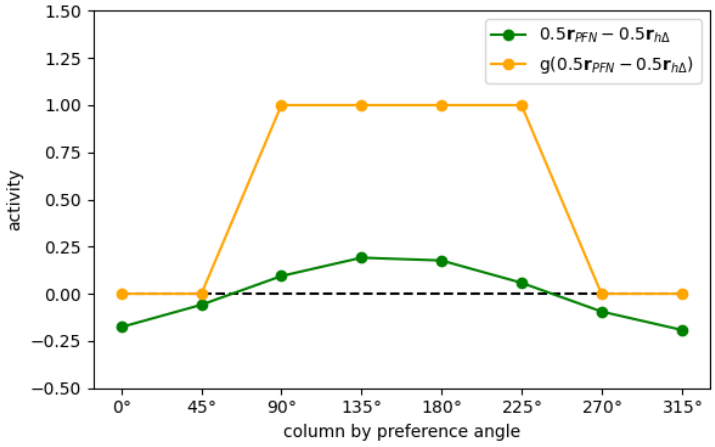


Figure 4.19: Illustration of amplification layer effect on the memory bump (after normalisation, c.f. figure 2.7).

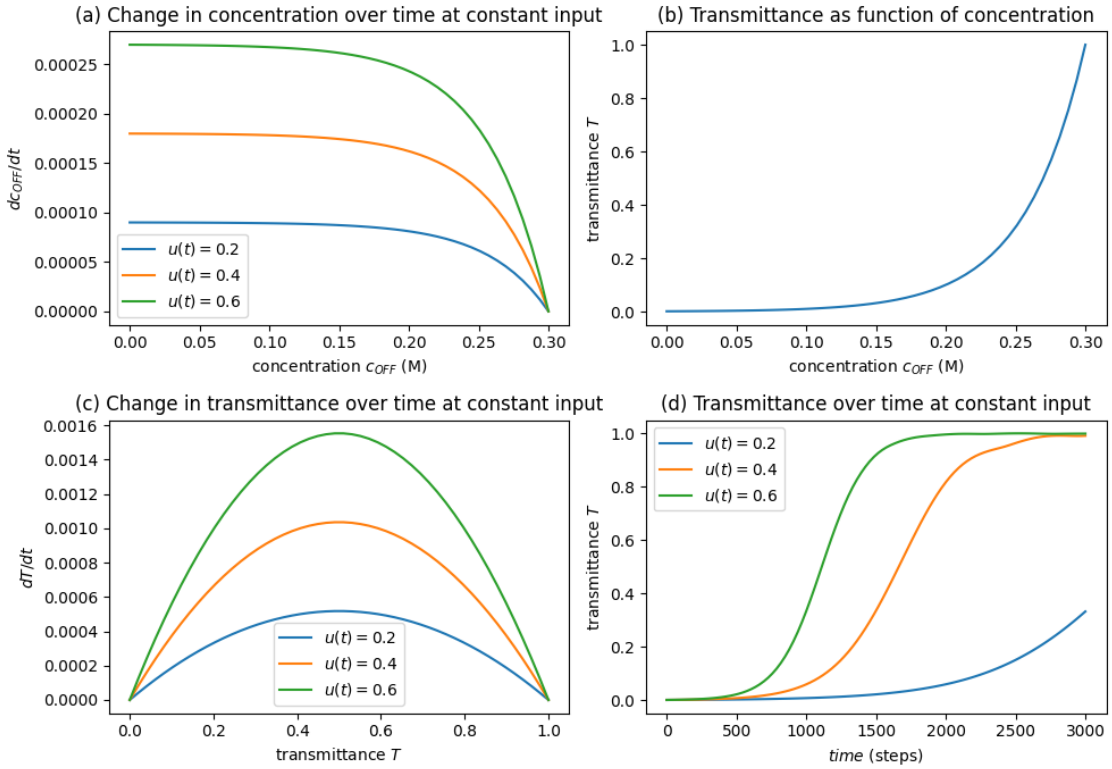


Figure 4.20: How the concentration and transmittance change over time with a constant input $u(t)$ ($\phi = 0.00045$, $\epsilon l = 10$, $c_{\text{tot}} = 0.3$). **(a)** The OFF-state concentration changes in varying proportion to u. **(b)** Transmittance, or weight, as a function of OFF-state concentration. **(c)** The derivative of the transmittance with respect to time. **(d)** The weights over time.

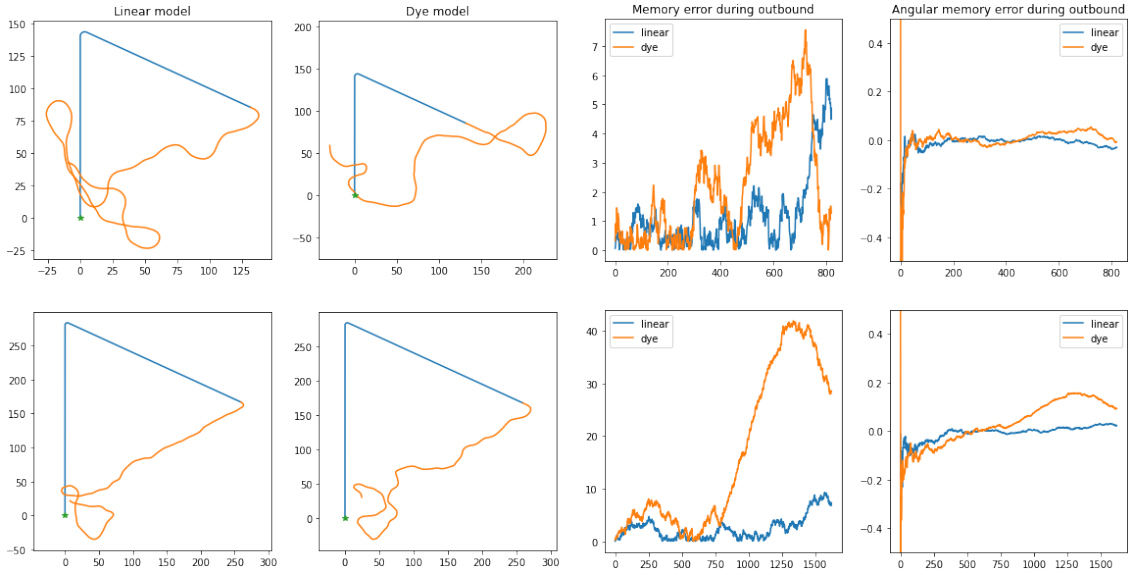


Figure 4.21: Homing for outbound paths with sharp turns to test integration of two distinct movement vectors at different distances from home.

the reason the homing works is that the weights *are* a function of the linear integral of the PFN outputs. To return to the water tank analogy, as long as no water leaks from the tank (in our case, $k = 0$), the total amount of water pumped in can be inferred from the water level.

To more qualitatively explore whether the memory captures the movements correctly, figure 4.21 shows a outbound paths meant to test how well the integration deals with specific situations of summing movement along two distinct vectors. We can look both at the actual routes taken during homing, but also decode the memory and plot the errors over time. We expect to see that even if the decoded memory error is greater at times, the final path still eventually reaches home, even if the path there is not entirely straight. This does appear to be the case: while the error is on the whole comparable between the different models, the nonlinear one (dye) exhibits a more pronounced spike in the memory error metric after turning, but the agent still homes successfully.

4.2.4 Background activity and readout window

As mentioned above, β – while it helps with readout – has the side effect of being constantly accumulated in memory, even when the agent does not move at all. Saturation at some point is of course unavoidable (there is a limit to the distance that can be represented by the memory), but a constant β without the ability to balance it out results in saturation after a certain *time*, in addition to just distance.

As a consequence, the choice of β depends on the length of the outbound path (T_{outbound}). For longer paths, a smaller β may be required in order to not reach saturation too early. Conversely, on shorter paths, there is more room for a larger β.

We must also consider its effects on the readout window: the time at which it is reached depends both on how far from home the agent has travelled, as well as on β, which can to some extent be used to control *when* the readout window is reached. For short paths,

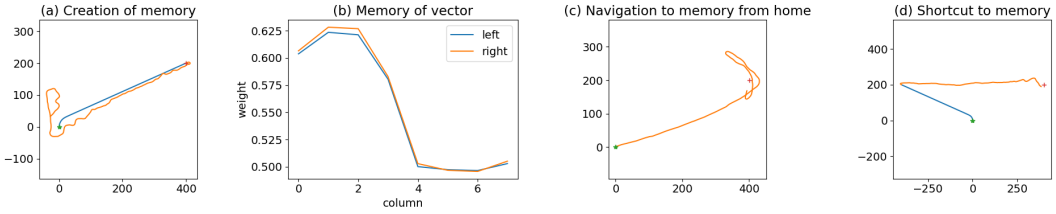


Figure 4.22: Vector-based navigation with linear dynamics. **(a)** First, the agent integrates the forced outbound (blue) path. **(b)** The memory that is generated of the location marked with a "+" is saved. **(c)** The agent is then given the vector memory and navigates to it. **(d)** Similarly, if made to move to another location, the agent takes a "shortcut" to the remembered location, demonstrating vector addition.

the performance improvement where the background activity is disabled for the outbound phase may not be desirable (unless ϕ can be increased instead), as that may delay the readout window by too much.

4.2.5 Consequences for vector-based navigation

Another recently suggested [15] extension of the path integration circuit is the addition of arbitrary vector memories, such as those of previously visited feeding locations. To recall these memories, populations of cells representing them are made to inhibit the output from the PFN cells, changing the effective vector that is passed as input to the steering system.

This can be replicated by making some smaller changes to the network. When the agent is at a location it should remember, it reads the home vector weights (for example by a "flash" of maximal activity on all PFN cells). We add an additional weight layer that stores this vector memory, getting a readout in the same manner as the readout of the integrator, i.e. by weighting the PFN output. This is then passed as an additional inhibitory input to the amplification layer and hΔlayer.

With linear memory dynamics this works well: the memory of location a, $M(a)$, can be summed with the memory of location b, $M(b)$ to represent the memory of location $a + b$, $M(a + b)$. Figure 4.22 illustrates the idea using the linear memory model from 4.1.2.

When attempting to do the same with the dye memory, however, the fact that it is not additive becomes very clear, demonstrated by figure 4.23. In (c), the agent *starts* to move toward the remembered location, but after a while it turns back home. Similarly, in (d), it starts off heading towards the remembered location, but gradually deviates to the right, eventually returning home.

The explanation for this ought to be that the amplitude of the vector memory is static, while the home vector grows nonlinearly. The scales of the two vectors are therefore not comparable, and the vector memory will quickly become very small in relation to the home vector.

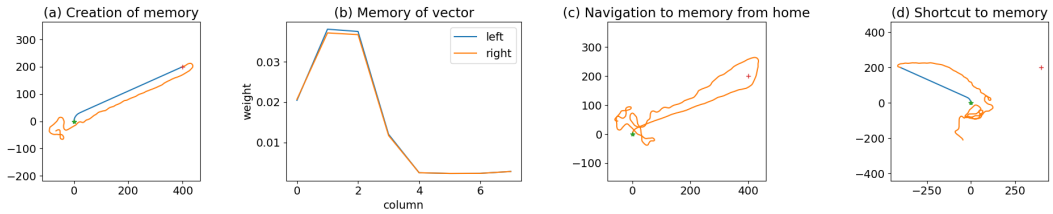


Figure 4.23: Unsuccessful vector-based navigation with dye dynamics.

4.2.6 Obstacle avoidance

We tested how our models would react if there were obstacles on the way home, which they would need to go around to get to the nest. Obstacles are added as an option to the framework, defined as a line segment between two points. During simulation, collision detection is done between the agent and each obstacle. This is done by checking if the simulated agent would cross the obstacle between the last timestep and the current, if movement was simulated normally. It will then instead receive a heading perpendicular to the obstacle, and a velocity pointing in that direction for that timestep. This makes the agent move along the obstacle until it can go around it. In figure 4.24 we can see the result of this test, comparing the basic version of our dye model to the benchmark model from [22]. It appears to work as well as expected of the basic model: the path is not very straight, which we attribute to the nonlinear memory, but it does eventually home in on a point close to the home.

4.2.7 Holonomic movement

Figure S7 (D-F) in the supplemental information for Stone [22], show some test for holonomic movement, i.e. moving in one direction while facing another. We performed similar testing to our dye model, with the same additions as the fully holonomic model in **F** from Stone. We chose the dye model with the added amplification layer, as it performs the best of our dye models for an outbound time of 1500 steps. The reason we did not use variable background activity for this example is that those models suffer from not being in the read-out window when beginning to home, resulting in the agent not turning very fast toward the home vector as homing is engaged. The result in figure 4.25 shows that our model is successful in performing path integration even when moving holonomically on the outbound path, performing about the same as the model in [22] did.

4.3 Quantitative performance

In this section we compare the different models that we have developed. One of the models that we are comparing is our proof-of-concept synaptic weight model, nicknamed *weights*. We also have our dye model at its most basic, nicknamed *dye basic*, as well as the dye model with background activity deactivated on the outbound journey, nicknamed *dye var beta*, and our dye model with the added amplification layer, nicknamed *dye amp*. In addition to these, we have one model that combines the variable background activity and the amplification

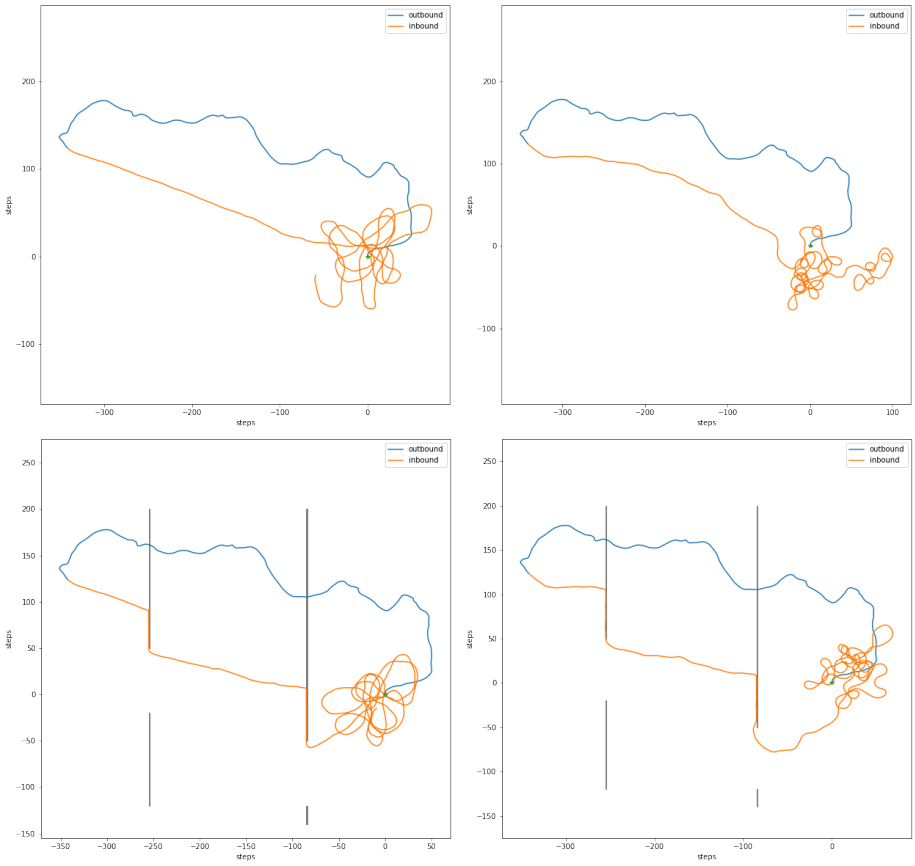


Figure 4.24: Simulation with, and without, obstacles put in the path of the agent while homing. On the left, benchmark model from [22]. On the right, our dye model without further improvements.

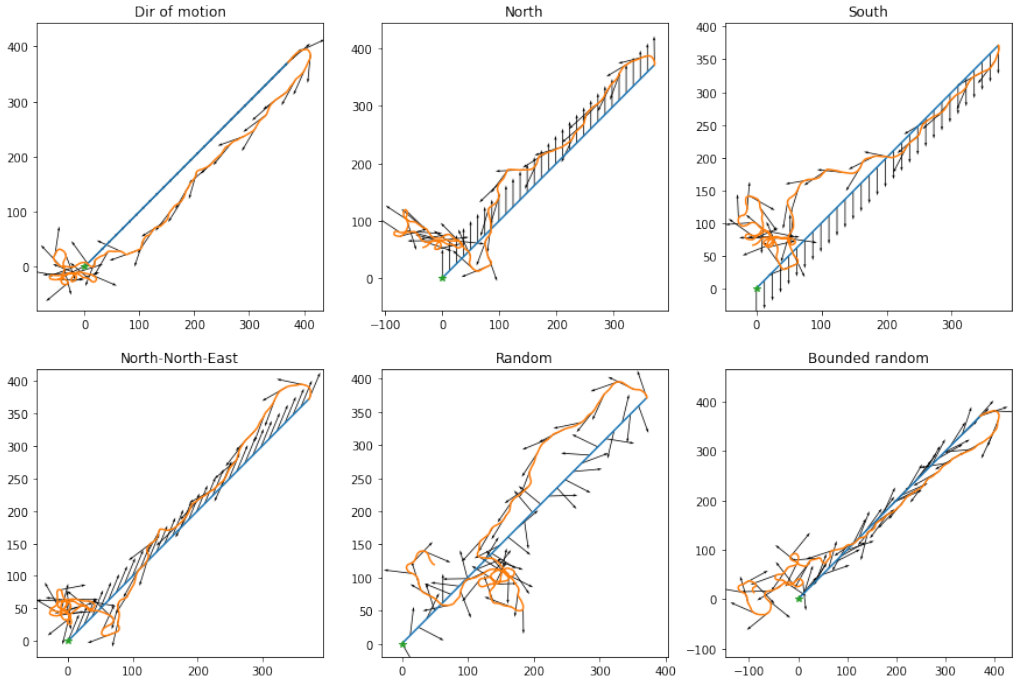


Figure 4.25: The *dye amp* model testing holonomic movement, using six different approaches during the outbound journey. In the first plot the agent is moving in the direction that it is facing, which would be the regular behaviour. In the second plot, the agent is always facing north while moving north-east. The arrows in the plots represent the current heading of the agent at that step.

layer, nicknamed *dye var beta + amp*. To compare against previous work, we have included the neural model that was shown in Stone et al. [22], nicknamed *stone*, as a benchmark.

4.3.1 Parameter settings

As per the parameter search in section 4.2.1, we chose the parameters that yielded the closest distance to home to be as fair as possible to the *dye basic* model, with which the parameter search was performed, even though they may be somewhat "overfit". For the other dye models, we based them upon these parameters with a few changes. The *dye amp* model used the same parameters as the dye basic model. The *dye var beta* model was given an increased background activity, since the perk of disabling β on the outbound path is the ability to use higher β while homing to improve readout. We also increased the ϕ, to increase accumulation of memory, since the model didn't seem to reach the readout window very fast. For the *dye var beta + amp* model, we increased only the ϕ. Finally, it showed that these models gained performance from lowering epsilon, so that was decreased.

We recognize that especially the *dye basic* model has been overfitted to a certain path length due to our parameter search focusing on 1500 outbound and inbound steps each. To achieve performance that is better across many outbound path lengths, one could balance ϕ and β differently; it is a question of resolution versus capacity. Lowering ϕ and β can improve performance on longer paths, but will decrease the performance for shorter paths. The models using variable background activity addresses this problem by a change in the network instead of varying parameters.

The setup for each model, showing all the parameters, is available in Appendix A.

4.3.2 Cross-model comparison

Our main quantitative result can be seen in figure 4.26, where we compare the four main metrics we have chosen for $N = 1000$ simulations of each model.

We can see that our *weights* model is very close to the performance of the *stone* model in all metrics, showing that moving the location of the memory (from PFN to synaptic weights) works without really worsening the performance of the path integration. The combination model *dye var beta + amp* shows essentially the same performance as the *dye amp* at all metrics, or a little worse. What the *dye var beta + amp* model offers in improvement is discussed further in section 4.3.3, where we look deeper into each model separately.

Closest distance

The closest distance is our most important metric, and shows the biggest differences between our models. The *weights* model is comparable to the benchmark *stone* model in this metric, showing that moving the memory to synaptic weights works without really worsening the performance of the path integration The difference is small, but statistically significant ($p = 6.33 \cdot 10^{-11}$). Our *dye basic* is the worst, with the widest distribution and more extreme outliers. Our *dye var beta* model seems to be performing a bit worse than *dye basic*, but a t-test suggests that the difference is not statistically significant ($p = 0.072$). The *dye amp* model shows that the amplification layer is our most effective addition to the model, showing no

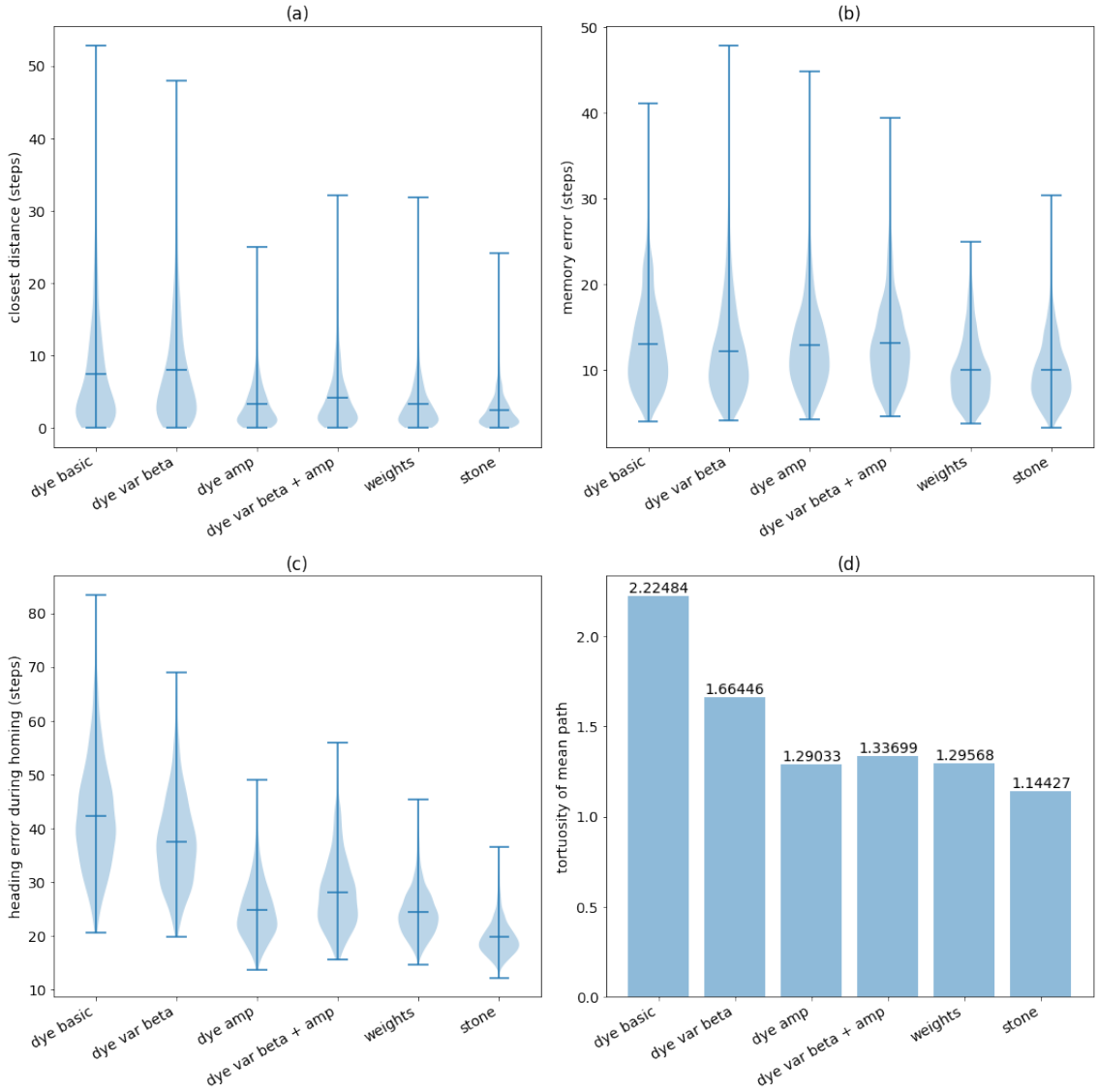


Figure 4.26: Comparison for the 4 main metrics between our dye model variations, our conceptual weights model and one of the neural models from [22]. N=1000 runs for each model. Noise=10%. Outbound/inbound time = 1500 steps each.

significant difference compared to the *weights* model ($p = 0.627$). The difference between *dye amp* and *dye var beta + amp* is also small, but in this case it is significant ($p = 9.91 \cdot 10^{-6}$).

Memory error

The memory error metric shows the quality of the internal memory representation. There is almost no difference in our four versions of the dye model, meaning that the performance gain in closest distance from our additions to the *dye basic* model does not stem from any improvement of the memory.

Not being restricted by the nonlinear dye mechanics, the *weights* model is visibly better than the dye based models. There is in fact no statistically significant difference to the benchmark *stone* model ($p = 0.633$). Between the linear and nonlinear models, such as *weights* and *dye basic*, there is a significant difference ($p = 3.56 \cdot 10^{-25}$).

Heading error

While the memory error is very similar, the heading error shows that our additions to the dye model result in a better readout of the memory. A better memory readout means that our steering system performs better, and the agent will follow the home vector better. It is then no surprise that we see a correlation between this and the closest distance metric. Worth noting is that there is no statistically significant difference between *dye amp* and *weights* ($p = 0.166$) in this metric.

Tortuosity

The fourth plot shows the tortuosity of the mean path for the 1000 simulations. The tortuosity plot shows us that the *stone* model was quite close to optimal performance, a tortuosity score of 1. This metric shows us the weakness of our *dye basic* and *dye var beta* model, and again highlights the amplification layer as a very performance enhancing addition. Due to the readout window and the readout quality just not being good enough, our dye model simply does not make very straight homing paths without the added amplification layer.

4.3.3 Single model evaluation

To add to the cross-model comparison, we have run each of the models through the evaluation suite described in section 3.3.2. The evaluation suite for our dye models and conceptual weight model can be found in figure 4.27-4.31, while the benchmark *stone* model can be found in figure 3.2.

The *dye basic* model (4.28) has the worst performance in subfigure **(c)**, as it does not enter the readout window for shorter paths and its memory is saturated for longer paths.

Looking at figure 4.29 and 4.31 **(c)**, we can see where only having background activity while homing is the most impactful. As the outbound time was increased, the models closest distance didn't increase nearly as much as it did for other models. This is where the combination model *dye var beta + amp* shows that it does offer something compared to just the *dye amp* model, as it essentially has the performance of *dye amp* but with the extra benefit of managing to return home for longer outbound time. The *dye amp* model struggles with

longer paths due to the saturation of memory. The *dye amp*, *dye var beta + amp* and the *weights* model are all comparable to the benchmark stone model in how quickly the heading error decreases (subfigure **(e)**), and they do measure up well in the tortuosity and deviation after 20 step radius too (**(f)** and **(g)**).

4.3.4 Parameter noise

We tested how our dye model would work if we applied some noise to the parameters that control the dye memory. The noise was applied to a parameter p like

$$\hat{p} = p + ps, \text{ where } s \in \mathcal{N}(0, noise)$$

This noise was added to the β, ϕ, ϵ, l and k parameters. Since we use $k = 0$ for all our models, this parameter is not influenced. The result for our *dye basic* model can be seen in figure 4.32. A modest noise level of 1-2% seem to work fine, while increasing it to 10% results in complete loss of path integration ability. This is not a surprise, as when we performed a parameter search we saw that we are working with very small ranges of values for certain parameters to achieve working path integration, especially for ϕ.

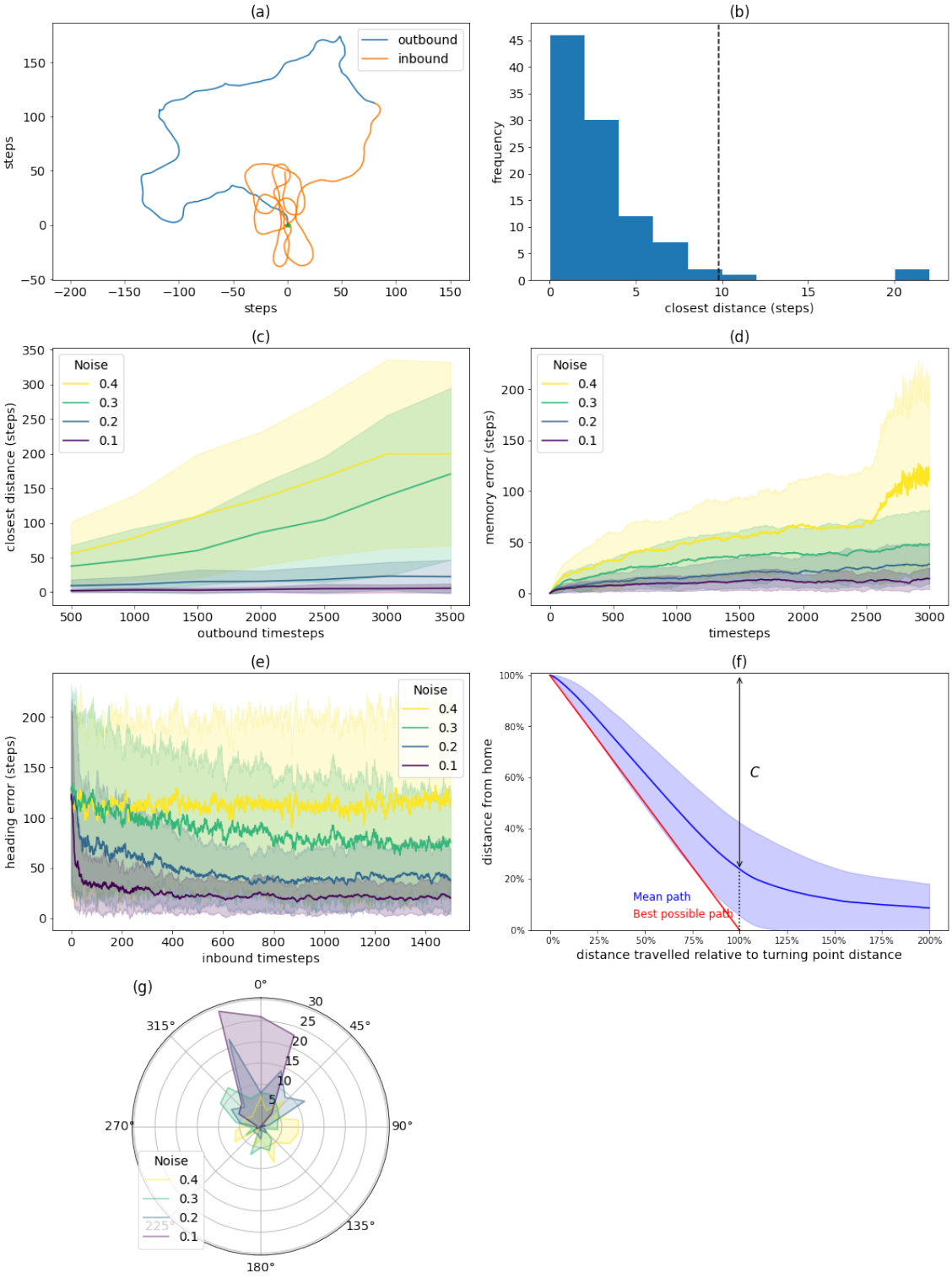


Figure 4.27: Evaluation suite for the *weights* model, our linear weight model. Detailed explanation for subfigures **(a)**-**(g)** can be found in figure 3.2.

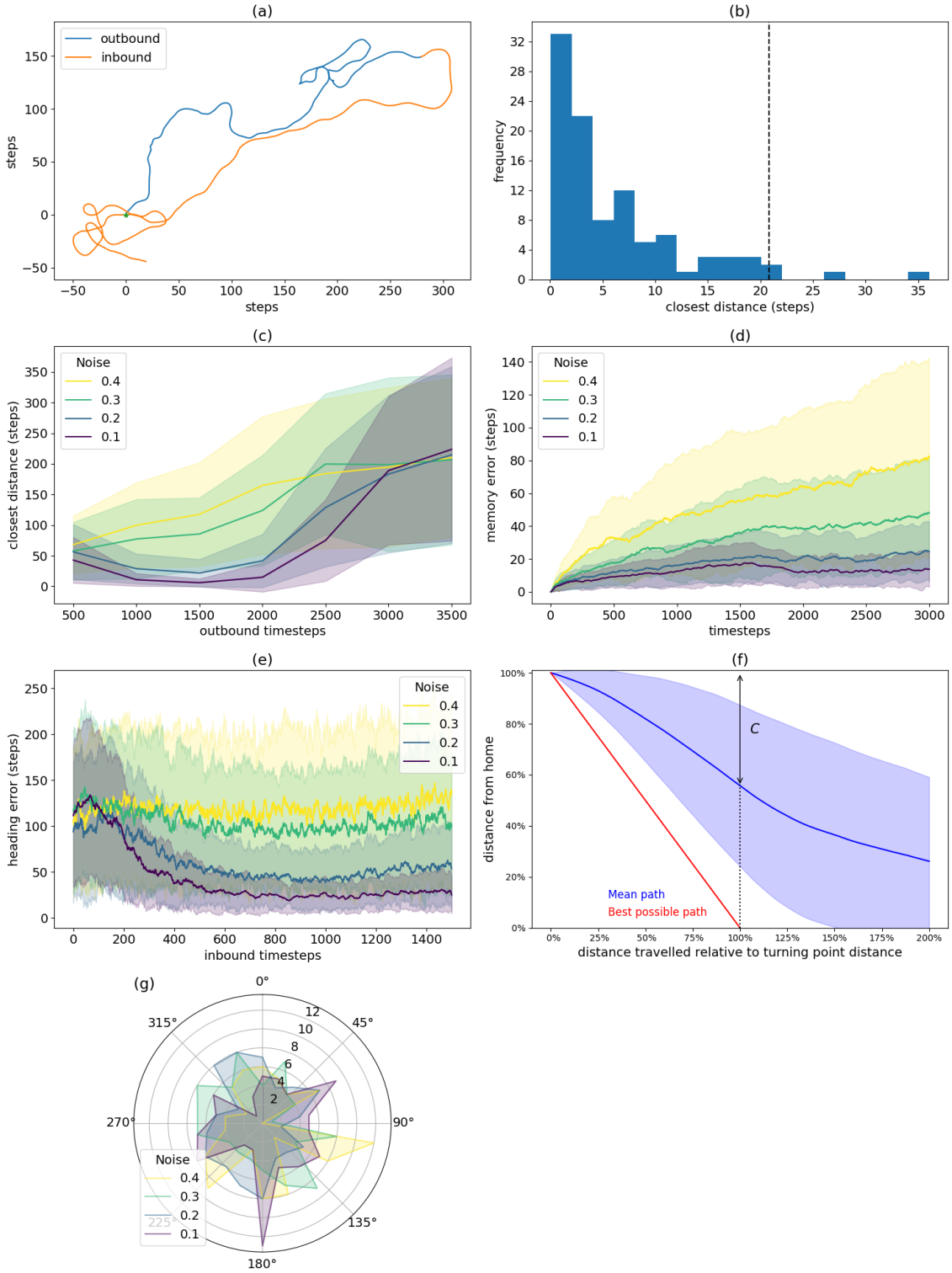


Figure 4.28: Evaluation suite for the *dye basic* model, our dye model without any network changes.

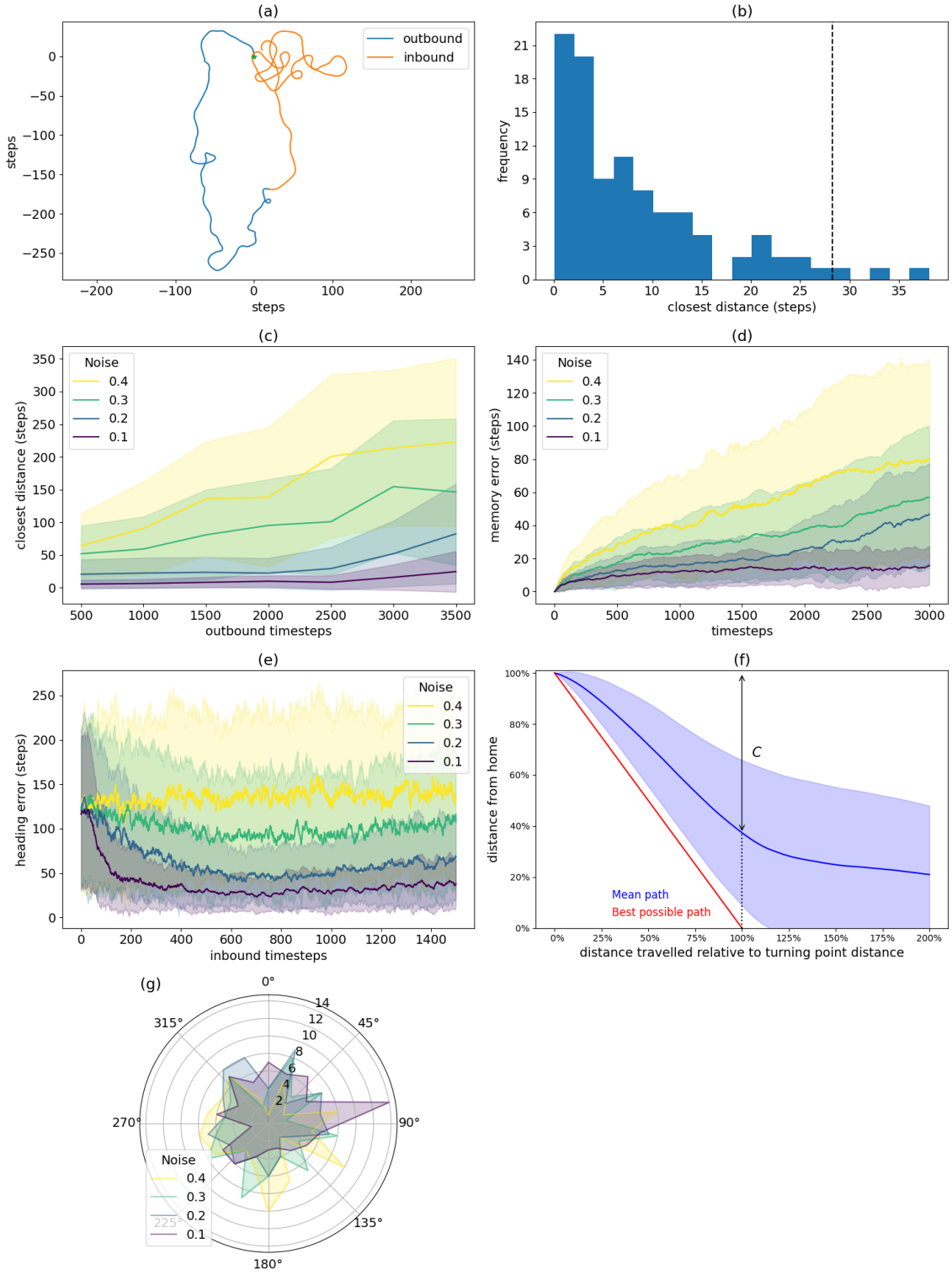


Figure 4.29: Evaluation suite for the *dye var beta* model, our dye model with background activity deactivated during outbound journey.

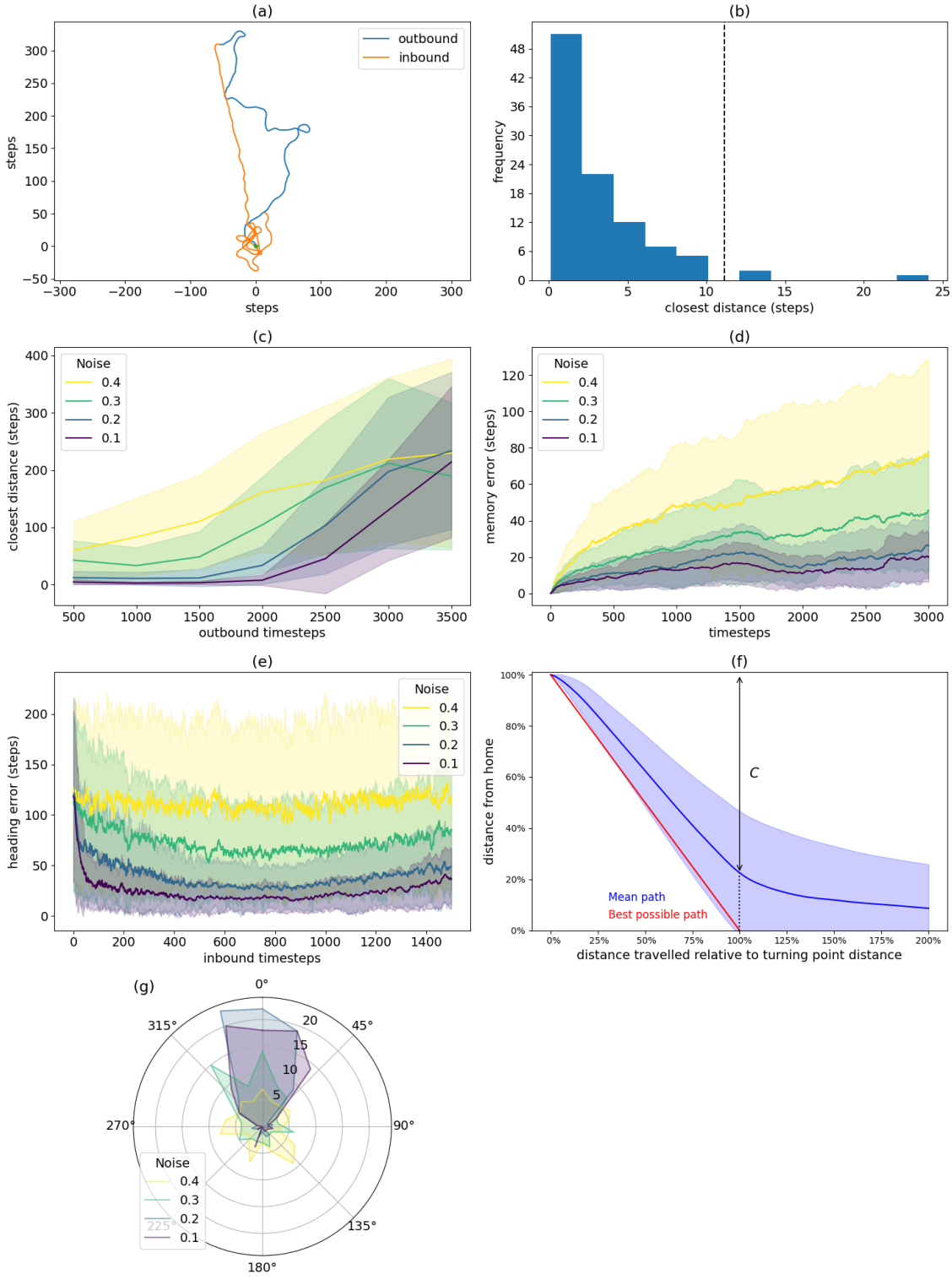


Figure 4.30: Evaluation suite for the *dye amp* model, dye model with added amplification layer.

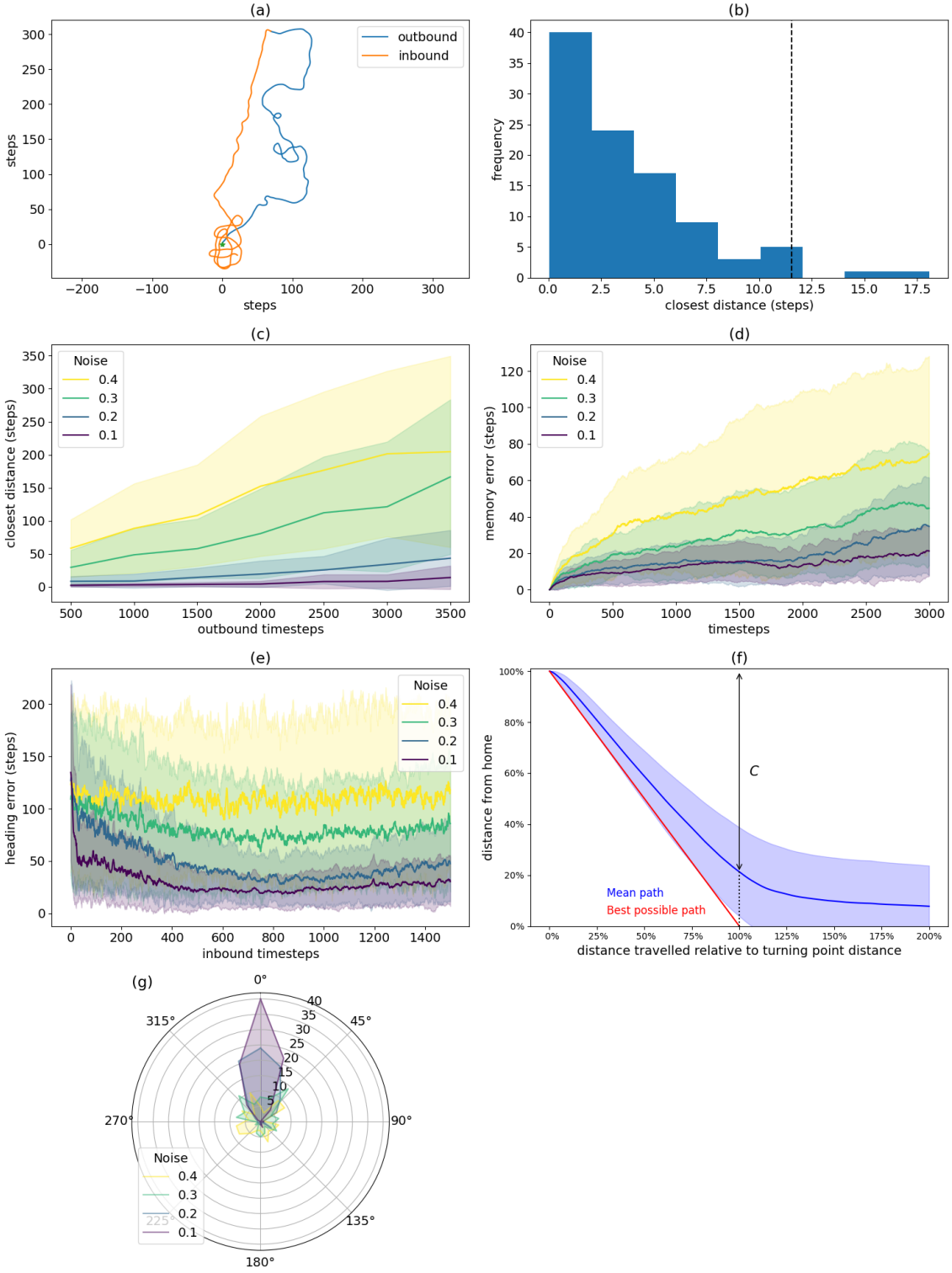


Figure 4.31: Evaluation suite for the *dye var beta + amp* model, dye model with both background activity deactivated during outbound and amplification layer.

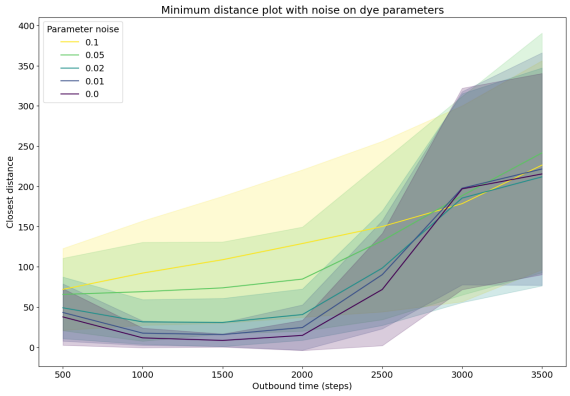


Figure 4.32: The *dye basic* model with noise applied to dye parameters.

Chapter 5
Discussion

Our results suggest that memory based on photoswitching dyes is plausible. With the memory moved to the synaptic weights, and using a constant background activity β for the readout, behaviour close to the original Stone model is achievable with a linear memory process by balancing β with the memory decay. With the nonlinear dye mechanics, we can no longer balance β against the decay. This results in a contradiction: we need β to be small in order to not saturate the memory, but we need it to be large in order to get a good readout. The most basic model, whose readout is based on a small background activity, works, but further improvements are possible.

We have not found any clearly identifable effects of nonlinearity on the agent's ability to reach home. By directly decoding the population vector stored in the synaptic weights, peaks in memory error (i.e. how far from home the decoded vector is pointing) can be seen after turning in figure 4.21, which is consistent with expectations as the initial movement along the second vector should not be reflected as strongly in the memory as movement along the first. However, this does not appear to noticeably affect the location to which the agent navigates; as mentioned in the results, each synapse needs to see the same total activity for the memory bump to flatten out again. Regardless, while these effects could be analysed with more rigorous mathematics, the quantitative comparisons suggest that the effects seem to not be catastrophic for homing performance when quantified by the closest distance to home that was reached by the agent during simulations. In fact, while the memory error is higher for nonlinear memory, we found no statistically significant differences between the *dye basic* and *weights* models in the closest distance or heading error metrics. This suggests that the differences are not caused by the nonlinearity. It should also be noted that even the peak memory error of around 40 distance units in figure 4.21 is within the radius of around 50 distance units for a typical search pattern judging by examining trials.

In some of the example paths, there is an apparent offset between the point around which the search pattern is centered and the actual home. While we found that there is a difference in the memory error metric between the linear and nonlinear models, the linear model also sometimes exhibits this offset (such as in figure 5.1), suggesting this offset is not caused by the

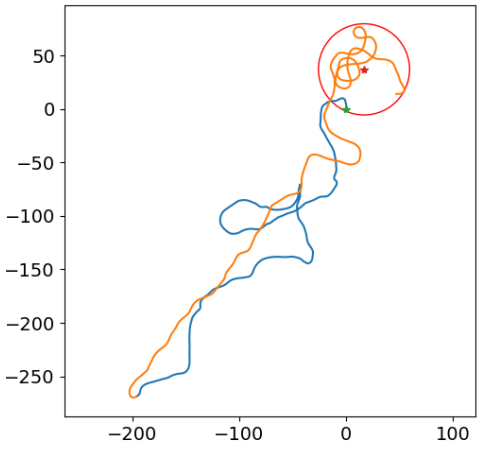


Figure 5.1: A *weights* model trial that shows an estimated search pattern (red circle) that is offset form the actual home.

nonlinearity. Rather, we believe it is a consequence of noise accumulation (due to applying PFN's noisy sigmoid before integration), holonomic movements due to tight turns (which are not captured perfectly with only the forward-sensitive speed neurons), and the poorer readout when compared to the original model.

The main effect of the nonlinearity therefore seems to be less straight paths and the existence of a readout window. An interesting consequence of this window is that when the concentration is below the soft threshold value (below which the transmittance barely reflects the memory at all), the steering system does not get any usable signal. In effect, there is an innate threshold at which homing is "activated". Before this, the behaviour of the steering system is dominated by noise, which results in the agent walking randomly, generating its own exploratory outbound paths.

5.1 Biological plausibility

The main assumption we make is that it is plausible to move the integration of the PFN output to its downstream synapses, and that we can use some kind of background activity to get a usable readout. In figure 4.9 the background activity is illustrated as a change in the PFN layer's activation function (or f-I curve). An alternative mechanism could be that the background activity comes from another neuron that excites the PFN outputs (comparable in nanowire terms to a second light source in close proximity to each PFN neural unit). Moreover, it may be feasible for such a neuron to connect to all PFL/hΔcells through synapses whose strengths are modulated by PFN activity, without the PFN activity directly affecting the readout. The activity of this neuron would then control whether or not homing is active. It may even be feasible that a regulator circuit could vary this activity based on the memory readout in order to compensate for the nonlinear relationship between the total input and the weights.

For the readout of the memory in our model to work better we introduced a new layer

of neurons between the PFN and hΔ/PFL layers. While not present in any of the previous model, *Drosophila* connectomics suggest that there are actually very few direct connections between the PFN and PFL neurons [10]. It is therefore not unimaginable that one the layers of pontine cells between them may have such an amplifying effect. We propose that a type of neuron could exist between the hΔ and PFL layers, that is excited by PFN and inhibited by the hΔ. There would be 16 columns, having the same connectivity pattern as the PFL previously had, and a sharp sigmoidal activation function. Its output would excite the PFLs.

As for plausible biological memory mechanisms, for long-term potentiation (LTP) of synapses (i.e. persistent strengthening) one molecule of interest is Ca^{2+}/calmodulin-dependent protein kinase type II (CaMKII) [17][16]. Such molecules could act as bistable switches that become activated by synaptic input and increase the synaptic weight when active, similarly to dye molecules. However, their active state is maintained by an autophosphorylation reaction, which would be consistent with cold-induced anesthesia experiments on real insects that found gradual loss of the home vector [20] as a lower temperature may reduce the rate of such a reaction, and thereby increase the rate of decay.

Using only semi-permanent photoswitches as we have done in this project (i.e. $k \approx 0$) means that the memory has to be reset by some process external to the model. With the photoswitches, the rate of the back-reaction is dependent on temperature: a higher temperature increases the rate and thereby the memory decay, and a lower temperature would make memory last longer. Interestingly, such a process is in contrast to the results in [20], which is also noted therein. That study found that a proportional reduction of the memory was most consistent with the results, where the length of the home vector degraded faster than the direction. As the concentration of transparent (OFF-state) dye molecules decay exponentially with rate k, it would be subject to such proportional reduction. However, as the accumulation of memory is not linear, and the transmittance is not a linear function of concentration, this does not necessarily correspond to a proportional reduction of the transmittance and of the distance the agent actually homes.

For homing purposes it seems that any mechanism that can store graded information that reflects the total amount of past activity, and that works on appropriate time scales, is a candidate for path integration memory, as it appears to work quite well even with nonlinear processes. The outlook for using this circuitry for more general vector-based navigation [15] seems grim for any mechanisms that are not at least approximately linear, however. Perhaps it may still be plausible with somewhat nonlinear memory, for example with the caveat that it only works close to or far from the nest. Also, as we have mentioned, the dye memory does represent a direct function of the total amount (i.e. the actual integral) of PFN activity it has seen, even if the relationship is not linear. With downstream neurons that perform another nonlinear computation that represents the inverse of this function, it would be feasible to ”decode” the memory so that it can be used for vector-based navigation.

5.2 Nanotechnical plausibility

While the anatomy of the amplification layer is unclear, that is not an obstacle to implementing such a layer in hardware. Rather, it is a question of whether the steepness of the activation function is attainable using nanowires.

The parameters we have found to be required for our model appear to fall inside ranges

that are plausible for the physical dyes. Determining actual quantum yield depends on the intensity of the PFN layer light sources, which could also be adjusted in order to achieve the appropriate memory gain. This also depends on the timescales at which the circuit is to operate. The background activity is expressed as a proportion of the max intensity, and is thus also generalisable to specific light source intensities. Settings where $\epsilon l < 1$ always have a transmittance of more than 50% ($10^{-0.3} \approx 0.5$), meaning that they never come close to 100% attenuation and work with a much smaller span of attenuations. Smaller differences in attenuation between columns results in a poorer readout as the memory bump is not as distinguished. With $c_{\text{tot}} = 0.3$, our parameter searches suggest that performance is best around the order of $\epsilon l = 10$. At an optical path length of $l = 10^{-3}\,\text{cm} = 10\,\mu\text{m}$ (which is on the order of magnitude of the ring attractor network previously modelled using nanowires [23]), this would entail a molar absorption coefficient of $\epsilon = 10^4\,\text{M}^{-1}\,\text{cm}^{-1}$. The upper limit of ϵ is in the range of $10^4\,\text{M}^{-1}\,\text{cm}^{-1}$ to $10^5\,\text{M}^{-1}\,\text{cm}^{-1}$ [14], meaning there may be some room for shorter optical path lengths or for varying ϵl. We have found no reason to decrease c_{tot} from its maximum of 0.3 M; to do so ϵl would also need to be adjusted to compensate.

The geometry of the circuit will likely be a challenge when implementing it in its entirety. Use of dye deposits for memory may complicate this further; for example, extra care has to be taken to ensure that crosstalk from other columns or layers does not affect the memory, even if that signal would not otherwise directly affect the receiver.

Again, using photoswitches with a very slow back reaction, the external reset of the home vector memory by way of increased temperature would need to be implemented somehow. It seems reasonable to do this when homing is successful, i.e. in the "nest". Something along the lines of a higher temperature in the nest could be plausible. Ultimately, this would probably depend on the actual application of the circuit.

5.3 Validity

Again, the dye model is simplified, and how well it captures the actual behaviour of these dyes has not been verified. It was deemed a reasonable approximation [14], but even regardless of its accuracy it is remains interesting to explore the effects of such dynamics.

Specifically, the simplified model of the dyes that we employ makes the following assumptions:

1. that the light behaves like a collimated beam with no fall-off, and
2. that the concentration of OFF-state molecules remains homogeneous throughout the dye.

Neither of these can be expected to be exactly true in a physical circuit. More accurate simulation would have to take into account the geometry of the dye and of the light source, and in general account for more complex interactions between light and the environment; this additional complexity was determined to be outside the scope of this project and a simplified model was deemed sufficient for exploring the behaviour.

The memory error metric (decoded memory) did not show very clear correlations with the closest distance to home metric. We expected to see a larger memory error with accumulating noise (as is an important difference between the original Stone model and our models),

but there was no significant difference in this metric between the *stone* and *weights* models ($p = 0.63$). As the memory error metric does not take the amplitude of the home vector into account, it is likely not very useful in capturing small offsets from home.

5.4 Other ideas

As has been explained, we focused on the case of $k = 0$ for this project, as it is not possible to balance the back-reaction decay with the input signal from the PFN layer so as to make the memory stable across the whole range of concentrations. They could be balanced to result in one fixed point, say at 0.5, towards which the memory would decay. If balanced against the mean of the PFN signal, this would result in negative modulation for PFN columns whose activity is below that mean, and positive for those that are above, thereby keeping the memory sinusoid centered around the fixed point. After some cursory investigation, this was only achieved with sufficiently long memory by working with very small differences in transmittance, something that makes readout impossible. It was therefore put aside.

There were some other ideas that were discussed but never explored, due to time constraints or because they required greater changes to the network. For example, as described in 4.1.1, it may be possible to do away with the steering system and use the weighted PFN signal more directly, or it may be possible to compensate for the weight multiplication in order to get a perfect readout.

Variable background activity can be taken further than was done in this project: disabling it for most of the time, but periodically ”flashing” a very high activity (as in the case where $\beta = 1$, resulting in a perfect readout), could conceivably result in a much better readout while simultaneously affecting the memory less. If this periodic readout is much stronger than the readout during the rest of the time, it may be enough to create a tendency to follow the home vector. If the steering system operates only periodically, synchronously with the flash, it could disregard the signal the rest of the time entirely (perhaps with the trade-off of making fewer but sharper turns).

Finally, constraint number two in section 4.1 could be alleviated by exploiting dye molecules' different responses to different wavelengths, such as the fact that ϵ is a function of wavelength [13]. This could allow for separate ”reading” and ”writing” wavelengths, which would be similar to the idea discussed in section 5.1 of having synapses that are modulated by the PFN activity but read out by another type of cell.

5.5 Future work

This work has only begun to explore the usage of photo-switching dyes as memory in the path integration circuit, and leaves a lot of room for future work. Firstly, the model needs to be verified using more detailed simulation of light signals, taking into account the geometry of the circuit and the dye deposits. The specific geometry may alter the dynamics of the plastic weights as well as introduce issues such as crosstalk between columns and layers.

Simulating memory is but one part of the much greater process that will hopefully lead to simulation of the entire path integration circuit implemented using nanowires, and eventually the actual construction of physical devices.

Further, the alternative ideas above and those discarded in the results could be interesting to explore. As for biological models, it would be interesting to actually implement computational models of the biological memory mechanisms discussed to determine whether they can actually function in a similar manner. Similarly, more rigorously identifying the required memory characteristics would be useful.

The extra layer of pontine cells makes for an interesting hypothesis regarding the connectivity between PFN and PFL cells, which can be studied in greater detail as new connectomics data becomes available.

Chapter 6
Conclusions

With minimal changes to the neural network layout we achieve a functioning path integration circuit where the memory is implemented as plastic synaptic weights using a simple model of the candidate dye molecules, suggesting that this is a promising way forward and as such constituting a positive answer to RQ1. Specifically, the parameters settings required for this to work appear to fall within realistic ranges for the actual dyes. The nonlinear dye dynamics do not appear to be an issue for homing, but do limit the ability for arbitrary vector-based navigation.

As for RQ2, we find that with no other changes to the network than moving the memory to dye-based synaptic weights we get a functioning path integrator, but with relatively poor performance. Background activity helps, especially if it can be varied during operation, and an additional pontine amplification layer makes a significant further difference. Such a layer does not seem entirely implausible from a biological point of view and could constitute a hypothesis about the pontine cells' function, which is not thoroughly understood.

In the discussion we briefly consider RQ3, and compare the dye memory to CaMKII-based mechanisms of plasticity. While they bear an interesting resemblance, they do differ in the way which memory decays and the dye memory seems inconsistent with decay observed in actual insects. Due to time constraints we did not explore biological models in more depth.

We also briefly suggest a few possible alternative implementations that could be explored that make use of more complex dye behaviours and require more significant changes to the network (likely further complicating the circuit geometry).

References

[1] Definition of BEELINE. `https://www.merriam-webster.com/dictionary/beeline`. Accessed: 2022-10-15.

[2] Enrique Barrigón, Magnus Heurlin, Zhaoxia Bi, Bo Monemar, and Lars Samuelson. Synthesis and applications of iii–v nanowires. *Chemical Reviews*, 119(15):9170–9220, 2019. PMID: 31385696.

[3] Peter Dayan and Laurence F. Abbott. *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*. Computational Neuroscience Series. Massachusetts Institute of Technology Press, 2001.

[4] Thomas Ferreira de Lima, Bhavin J. Shastri, Alexander N. Tait, Mitchell A. Nahmias, and Paul R. Prucnal. Progress in neuromorphic photonics. *Nanophotonics*, 6(3):577–599, 2017.

[5] Roman Goulard, Cornelia Buehlmann, Jeremy E. Niven, Paul Graham, and Barbara Webb. A unified mechanism for innate and learned visual landmark guidance in the insect central complex. *PLoS Computational Biology*, 17, 9 2021.

[6] Stanley Heinze. Unraveling the neural basis of insect navigation. *Current Opinion in Insect Science*, 24:58–67, December 2017.

[7] Sameh Helmy, Frank A Leibfarth, Saemi Oh, Justin E Poelma, Craig J Hawker, and Javier Read de Alaniz. Photoswitching using visible light: a new class of organic photochromic molecules. 2014.

[8] James R. Hemmer, Saemi O. Poelma, Nicolas Treat, Zachariah A. Page, Neil D. Dolinski, Yvonne J. Diaz, Warren Tomlinson, Kyle D. Clark, Joseph P. Hooper, Craig Hawker, and Javier Read de Alaniz. Tunable visible and near infrared photoswitches. *Journal of the American Chemical Society*, 138(42):13960–13966, 2016. PMID: 27700083.

[9] Anna Honkanen, Andrea Adden, Josiane Da Silva Freitas, and Stanley Heinze. The insect central complex and the neural basis of navigational strategies. *Journal of Experimental Biology*, 222, 2 2019.

[10] Brad Hulse, Hannah Haberkern, Romain Franconville, Daniel Turner-Evans, Shinya Takemura, Tanya Wolff, Marcella Noorman, Marisa Dreher, Chuntao Dan, Ruchi Parekh, Ann Hermundstad, Gerald Rubin, and Vivek Jayaraman. A connectome of the drosophila central complex reveals network motifs suitable for flexible navigation and context-dependent action selection. *eLife*, 10, 10 2021.

[11] Jerome K. Hyun, Shixiong Zhang, and Lincoln J. Lauhon. Nanowire heterostructures. *Annual Review of Materials Research*, 43(1):451–479, 2013.

[12] Tore Hägglund. Reglerteknik AK Föreläsningar. `https://www.control.lth.se/fileadmin/control/Education/EngineeringProgram/FRTF05/forel.pdf`.

[13] Kristin Klaue, Wenjie Han, Pauline Liesfeld, Fabian Berger, Yves Garmshausen, and Stefan Hecht. Donor–acceptor dihydropyrenes switchable with near-infrared light. *Journal of the American Chemical Society*, 142(27):11857–11864, 2020. PMID: 32476422.

[14] Bo W. Laursen. personal communication.

[15] Florent Le Moël, Thomas Stone, Mathieu Lihoreau, Antoine Wystrach, and Barbara Webb. The central complex as a potential substrate for vector based navigation. *Frontiers in Psychology*, 10, 2019.

[16] John Lisman and MA Goldring. Feasibility of long-term storage of graded information by the ca2+/calmodulin-dependent protein kinase molecules of the postsynaptic density. *Proceedings of the National Academy of Sciences of the United States of America*, 85:5320–4, 08 1988.

[17] John Lisman, Ryohei Yasuda, and Sridhar Raghavachari. Mechanisms of camkii action in long-term potentiation. *Nature Reviews Neuroscience*, 13(3):169 – 182, 2012.

[18] Diederick C. Niehorster. Optic flow: A history. *i-Perception*, 12(6), 2021.

[19] Keram Pfeiffer and Uwe Homberg. Organization and functional roles of the central complex in the insect brain. *Annual Review of Entomology*, 59(1):165–184, 2014.

[20] Ioannis Pisokas, Wolfgang Rössler, Barbara Webb, Jochen Zeil, and Ajay Narendra. Anesthesia disrupts distance, but not direction, of path integration memory. *Current Biology*, 32(2):445–452.e4, 2022.

[21] Timothy Sauer. *Numerical Analysis: Pearson New International Edition*. Pearson Education, 2013.

[22] Thomas Stone, Barbara Webb, Andrea Adden, Nicolai Weddig, Anna Honkanen, Rachel Templin, William Wcislo, Luca Scimeca, Eric Warrant, and Stanley Heinze. An anatomically constrained model for path integration in the bee brain. *Current Biology*, 27, 10 2017.

[23] David Winge, Steven Limpert, Heiner Linke, Magnus Borgström, Barbara Webb, Stanley Heinze, and Anders Mikkelsen. Implementing an insect brain computational circuit using iii-v nanowire components in a single shared waveguide optical network. *ACS Photonics*, 7, 10 2020.

Appendices

Appendix A

Setups

```
"dye basic": {
    "comment": "basic dye model",
    "N": 100,
    "type": "simulation",
    "T_outbound": {"list": [500,1000,1500,2000,2500,3000,3500]},
    "T_inbound": 1500,
    "min_homing_distance": 0,
    "motor_factor": 0.25,
    "record": ["memory"],
    "cx": {
        "type": "dye",
        "output_layer": "motor",
        "params": {
            "noise": {"list": [0.1,0.2,0.3,0.4]},
            "phi": 0.0003,
            "beta": 0.3,
            "k": 0,
            "epsilon": 1.5e4,
            "length": 1e-3,
            "c_tot": 0.3,
            "cheat": false,
            "start_at_stable": false,
            "disable_beta_on_outbound": false
        }
    }
}
```

```
"dye var beta": {
    "comment": "dye model with activatable beta",
    "N": 100,
    "type": "simulation",
    "T_outbound": {"list": [500,1000,1500,2000,2500,3000,3500]},
    "T_inbound": 1500,
    "min_homing_distance": 0,
    "motor_factor": 0.25,
    "record": ["memory"],
    "cx": {
        "type": "dye",
        "output_layer": "motor",
        "params": {
            "noise": {"list": [0.1,0.2,0.3,0.4]},
            "phi": 0.0005,
            "beta": 0.5,
            "k": 0,
            "epsilon": 0.5e4,
            "length": 1e-3,
            "c_tot": 0.3,
            "cheat": false,
            "start_at_stable": false,
            "disable_beta_on_outbound": true
        }
    }
}
```

```
"dye amp": {
    "comment": "dye model with amplification layer",
    "N": 100,
    "type": "simulation",
    "T_outbound": {"list": [500,1000,1500,2000,2500,3000,3500]},
    "T_inbound": 1500,
    "min_homing_distance": 0,
    "motor_factor": -0.25,
    "record": ["memory"],
    "cx": {
        "type": "dye",
        "output_layer": "motor",
        "params": {
            "noise": {"list": [0.1,0.2,0.3,0.4]},
            "phi": 0.0003,
            "beta": 0.3,
            "k": 0,
            "epsilon": 1.5e4,
```

```
            "length": 1e-3,
            "c_tot": 0.3,
            "cheat": true,
            "start_at_stable": false,
            "disable_beta_on_outbound": false
        }
    }
}
```

```
"dye var beta + amp": {
    "comment": "dye model with activatable beta and amplification layer",
    "N": 100,
    "type": "simulation",
    "T_outbound": {"list": [500,1000,1500,2000,2500,3000,3500]},
    "T_inbound": 1500,
    "min_homing_distance": 0,
    "motor_factor": -0.25,
    "record": ["memory"],
    "cx": {
        "type": "dye",
        "output_layer": "motor",
        "params": {
            "noise": {"list": [0.1,0.2,0.3,0.4]},
            "phi": 0.0005,
            "beta": 0.3,
            "k": 0,
            "epsilon": 0.5e4,
            "length": 1e-3,
            "c_tot": 0.3,
            "cheat": true,
            "start_at_stable": false,
            "disable_beta_on_outbound": true
        }
    }
}
```

```
"weights": {
    "comment": "proof-of-concept weight model",
    "N": 100,
    "type": "simulation",
    "T_outbound": {"list": [500,1000,1500,2000,2500,3000,3500]},
    "T_inbound": 1500,
    "min_homing_distance": 0,
    "motor_factor": 0.25,
```

```
    "record": ["memory"],
    "cx": {
        "type": "weights",
        "output_layer": "motor",
        "params": {
            "noise": {"list": [0.1,0.2,0.3,0.4]},
            "beta": 0.5,
            "mem_gain": 0.0025,
            "mem_fade": 0.625
        }
    }
}
```

```
"stone": {
    "comment": "stone's rate pontine model",
    "N": 1000,
    "type": "simulation",
    "T_outbound": {"list": [500,1000,1500,2000,2500,3000,3500]},
    "T_inbound": 1500,
    "min_homing_distance": 0,
    "record": ["memory"],
    "cx": {
        "type": "pontine",
        "params": {
            "noise": {"list": [0.1,0.2,0.3,0.4]}
        }
    }
}
```

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2022-10-25

EXAMENSARBETE Modelling photoswitching dye memory for path integration
STUDENT Nils Ceberg, Jacob Säll Nilsson
HANDLEDARE Stanley Heinze (Lunds universitet), Barbara Webb (University of Edinburgh)
EXAMINATOR Jacek Malec (LTH)

Färgämnen som minns vägen hem

POPULÄRVETENSKAPLIG SAMMANFATTNING **Nils Ceberg, Jacob Säll Nilsson**

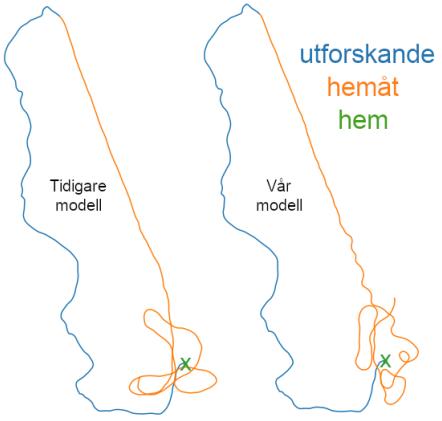
Artificiella neurala nätverk som efterliknar insekter skulle kunna vara grunden för framtidens navigationsteknologier. Vi har undersökt om färgämnen med varierbar genomskinlighet kan utgöra minnet i sådana nätverk.

Inom många områden, däribland artificiell intelligens, är det ofta ett användbart tillvägagångssätt att låta sig inspireras av naturens lösningar. En sådan är de neurala kretsar i hjärnan hos ett bi som låter det hitta hem efter att ha varit på jakt efter nektar. Man tror att de bland annat använder synintryck för att hålla koll på sina rörelser och därmed alltid vet vilket håll som är hemåt, i en process som kallas för *vägintegration*. En krets som skulle kunna ligga till grund för detta har tidigare rekonstruerats i en dator, i form av ett artificiellt neuralt nätverk.

Samtidigt har framsteg inom nanoteknik lett till att man tagit fram koncept för att utav så kallade *nanotrådar* konstruera artificiella neuroner som kommunicerar med hjälp av ljus. Därför undersöker man nu möjligheten att konstruera en fysisk version av en sådan vägintegreringskrets, på skalor som mäts i mikrometer – så litet att det enkelt skulle få plats i en riktig bihjärna!

Det saknas dock en viktig pusselbit: den mekanism som ligger bakom kretsens faktiska minne. Inom biologin finns ett antal sådana möjliga mekanismer, men det verkar troligt att det här handlar om synapser – det vill säga kopplingar mellan hjärncellerna – som kan variera i styrka. Eftersom nanotrådneuronerna kommunicerar med hjälp av ljus skulle sådana synapser kunna motsvaras av att materialet mellan dem kan göras mer eller mindre genomskinligt.

I detta examensarbete undersökte vi med hjälp av datorsimulering om det skulle kunna gå att använda en typ av färgämne som har egenskapen att dess genomskinlighet beror på hur mycket ljus det tidigare har absorberat. Man skulle alltså kunna säga att ämnet ”minns” hur mycket ljus det har utsatts för. Vi utvecklade en modell för att simulera beteendet hos kretsen om vissa av dess synapser ersattes med denna typ av ämne, och utförde sedan experiment då vi lät ett virtuellt bi försöka hitta hem.



Resultaten tyder på att den här typen av minne tycks fungera. Vår modell uppvisar nästintill samma förmåga att hitta hem som den tidigare modellen, vilket innebär att ämnet vi modellerat är en lovande väg framåt för att konstruera en fysisk vägintegrationskrets.