An Evolved Agent Performing Efficient Path Integration Based Homing and Search

R.J. Vickerstaff and E.A. Di Paolo

Centre for Computational Neuroscience and Robotics,
University of Sussex, Brighton, BN1 9QG, United Kingdom
robertvi@sussex.ac.uk
http://www.informatics.sussex.ac.uk/ccnr/people/RobertVickerstaff.html

Abstract. This paper presents analysis and follow up experiments based on previous work where a neurally controlled simulated agent was evolved to navigate using path integration (PI). Specifically, we focus on one agent, the best one produced, and investigate two interesting features. Firstly, the agent stores its current coordinates in two leaky integrators, whose leakage is partially compensated for by a normalisation mechanism. We use a comparison between four network topologies to test if this normalised leakage mechanism is adaptive for the agent. Secondly, the controller generates efficient searching behaviour in the vicinity of its final goal. We begin an analysis of the dynamical system (DS) responsible for this, starting from a simple three variable system.

1 Introduction

Path integration (PI) is a navigational method available in the absence of landmarks, and requires a compass and odometer. Information from these two sources must be continuously integrated during a journey to maintain a running estimate of the current position relative to some fixed reference point. In animal navigation studies the current estimated position is referred to as the home vector (HV), and is subject to the accumulation of error.

Several equational [1],[2],[3] and neural [4],[5],[6] models of PI exist. In our previous work [7] we presented a new neural model of PI in the desert ant *Cataglyphis fortis*, whose PI behaviour has been extensively studied [8]. We showed that it is possible, using a novel class of neural controller, to use a genetic algorithm (GA) to produce a PI system without imposing the neural mechanism for storing the HV. We also showed that using a complete model of the animal and its environment allowed the model to produce a search behaviour once the agent had returned to the vicinity of the nest. *C. fortis* also uses a searching behaviour to locate its nest, thus allowing it to home in spite of the navigation errors inherent to PI. Upon analysis we found our neural controller was very similar to Mittelstaedt's [1],[2] equational model. In this paper we present follow up work aimed at reaching a better understanding of our evolved model. We test four different network topologies to determine which one can produce the best PI system, and examine a series of equational models intermediate between Mittelstaedt's and our own.
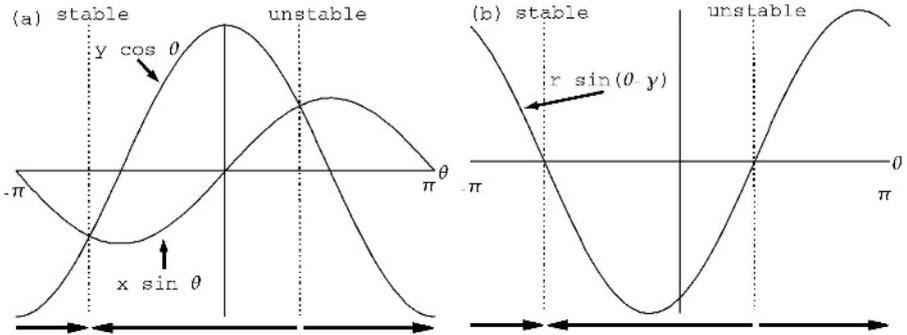
Fig. 1. Dynamics of homing using (a) equation 2 and (b) equation 3. The horizontal axis shows the animal's orientation θ, the vertical axis shows the animals rate of turn $\frac{d\theta}{dt}$ where (x, y) is the geocentric rectangular HV and (r, γ) is the geocentric polar HV. This example shows the case where (a) $x > 0, y = 2x$ and in equivalence (b) $\gamma = tan^{-1}2 = 1.107$. Both schemes lead to one stable and one unstable equilibrium heading with the same respective values.

1.1   Mittelstaedt's Bicomponent Model

PI in two dimensions can be expressed in terms of the operations required to update the HV [9], [10]. Mittelstaedt's bicomponent model [2]:

$$\frac{dx}{dt} = s\cos\theta, \frac{dy}{dt} = s\sin\theta \tag{1}$$

uses rectangular geocentric coordinates where (x, y) is the HV, s is the animal's speed and θ its compass heading. We can express the same system in polar geocentric coordinates:

$$\frac{dr}{dt} = s\cos(\theta - \gamma), \frac{d\gamma}{dt} = \frac{s}{r}\sin(\theta - \gamma)$$

where r is the animal's distance and γ its bearing from the origin. We can describe the process of homing using an equation for the animal's rate and direction of rotation as a function of the HV. Mittelstaedt's [1],[2] bicomponent model uses:

$$\frac{d\theta}{dt} = x\sin\theta - y\cos\theta \tag{2}$$

This causes the agent to always turn towards the origin $(0, 0)$. Converting this into polar form using the relations $x = r\cos\gamma, y = r\sin\gamma$ we obtain:

$$\frac{d\theta}{dt} = r\sin(\theta - \gamma) \tag{3}$$

During foraging the PI system passively updates the HV in response to the animal's movements, but when homing begins the system also directs movement,

forming a feedback system. To produce a complete model of PI navigation we therefore need to include a model of the animal's motion within its environment. Since the Mittelstaedt model updates the HV using geometrically correct formulae, in the absence of noise the HV values will also be the animal's true location. Therefore the three equations provide a simple, complete model of PI, including feedback.

2   Methods and Results

2.1   The Agent and PI Task

Our simulated agent has two compass sensors, C_L, C_R, outputting $\cos\theta, \sin\theta$ respectively (where θ is the agent's current heading), and a speed sensor, S, outputting a value between 0 and 1 indicating the agent's normalised speed. The agent's motion is controlled by an output, F, controlling the forward speed and two opposing outputs R_L, R_R controlling rotation (using $\frac{d\theta}{dt} = 150(R_L - R_R)$). To force the agent to travel at varying speeds, 70 percent noise is applied to the forward speed, and 10 percent noise to rotation. Sensor noise is 1 percent to ensure PI is feasible. To generate an initial outward excursion before homing begins the agent also has two beacon sensors B_L, B_R for phototaxis.

For each trial the agent started at the nest with a random orientation and was presented with a series of one to three randomly placed beacons which it was required to visit. Each beacon was removed when the agent reached it, and the next one activated. After the last beacon the agent's orientation was randomised, and it was held stationary for a short random time and then allowed to home using PI.

The fitness function used was such that the fittest possible agent would visit all beacons and return to the nest using direct paths at full speed, and would also search efficiently for the nest after reaching its estimate of the nest location to compensate for cumulative navigation errors. See [7] for full details.

2.2   ModCTRNN Neural Controller

Our previous work compared two types of neural controller on the PI task, the Continuous Time Recurrent Neural Network (CTRNN) [11], and a new controller the Modified CTRNN (ModCTRNN). Our results showed [7] that the ModC-TRNN outperforms the CTRNN under the conditions tested, and was able to evolve a much better PI system.

A ModCTRNN network consists of ordinary CTRNN neurons, governed by the standard leaky integration equation. Weights can link from one neuron to another as normal, but can also link from a neuron to a weight. The value of a weight is variable, and performs a leaky integration of its inputs using the same form of equation as the neurons. The state equation for ModCTRNN neurons is:

$$\tau_i \frac{dv_i}{dt} = -v_i + \sum_j w_j z_j \qquad (4)$$

where i indexes all neurons, j indexes all weights inputting to neuron i (if any), τ_i is a time constant, v_i is the neuron state, w_j is a weight and z_j is the activation of the sensor or neuron attached by weight j. For a neuron $z_j = \frac{1}{1+e^{-(v_j+b_j)}}$ where b_j is a bias parameter. For a sensor z_j is the current activation. The state equation for ModCTRNN weights is:

$$\alpha_i \frac{dw_i}{dt} = -w_i + \beta_i + \sum_j w_j z_j \tag{5}$$

where i indexes all network weights, j indexes all weights inputting to weight i (if any), α is a time constant, β is a bias term, w_j is a weight and z_j is the output of the neuron or sensor attached by weight j. All weights w_i are initialised to β_i, therefore any weights which receive no inputs remain constant at this value. A weight acting to modify another weight can itself be the target of modification, and so on, allowing an arbitrary degree of higher order weight change to take place.

2.3   Genetic Algorithm

An asexual GA with a population of 30 was used. Each genotype was evaluated in 10 trials per generation. Each trial the agent was tested on the PI task and assigned a fitness value. The genotype's overall fitness was the mean of 10 trials. The fittest 5 genotypes were retained unmodified in the population each generation. Each was copied 5 times to produced 25 new genotypes which were mutated and used to replace the 25 least fit genotypes. As well as mutating network parameter values, the GA also changed the number of neurons and weights present in the networks and the topology. Bilateral symmetry was imposed. Potential and weight biases were mutated within the range $[-100, 100]$. Potential and weight time constants were encoded using values between $[-2, 3]$ and mapped to their final values using $y = 10^x$, giving a range of $[0.01, 1000]$. For further details see [7].

2.4   Evolved ModCTRNN PI System

The topology of the best evolved network is shown (Fig. 2). If weights w_{L5}, w_{R5} and w_{L6}, w_{R6} are ignored, and the network can be seen to approximate Mittelstaedt's model. Weights w_{L3}, w_{R3} have a low time constant and small bias, and so they are largely at equilibrium values of $w_{L4}S, w_{R4}S$. Input to weights w_{L2}, w_{R2} are therefore $w_{R4}SC_R, w_{L4}SC_L$ respectively. Weights w_{L2}, w_{R2} have large enough time constants to remain far from equilibrium during a simulated journey, and therefore act to integrate their inputs, approximating Eqns.1 to act as the rectangular HV. Eqn.2 is implemented since the inputs to R_L, R_R are $w_{L2}C_L, w_{R2}C_R$ and since the two output neurons act in opposition to cause rotation.

Here we take two differing methodologies to analyse the PI system. Firstly, we investigate the effect of network topology on fitness, by constructing four
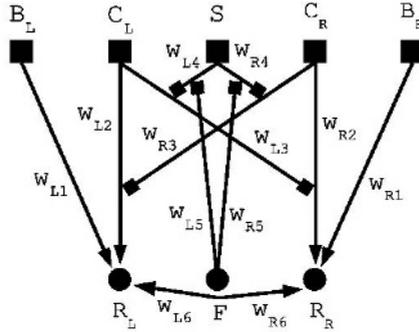
Fig. 2. The original ModCTRNN PI network. $B_{L/R}$, left/right beacon sensor, $C_{L/R}$, left/right compass sensor, $R_{L/R}$, left/right rotation motor neuron, S, speed sensor and F, forward motor neuron. Arrows are weights. Lines ending in small squares are weights which modify other weights. $w_{L1/R1} = 12.0720$, $w_{L2/R2} : \alpha = 8.4355, \beta = 0.0001$, $w_{L3/R3} : \alpha = 0.0123, \beta = 2.0477$, $w_{L4/R4} : \alpha = 5.1753, \beta = -98.7613$, $w_{L5/R5} = 65.9304$, $w_{L6/R6} = -3.5159$, $F : \tau = 0.0489, b = 42.8689$, $R_{L/R} : \tau = 0.0106, b = 0.2994$.

network topologies derived from the original, and evolving multiple populations for each. Secondly, we construct simplified DS models of the original in a noise-less, accurate numerical integration simulator and try to classify the types of behaviour we observe.

2.5   Evolving with Four Network Topology Classes

Four network topologies were constructed: *class 1* the original network topology, *class 2* the original minus weights $w_{L5/R5}$, *class 3* the original minus weights $w_{L6/R6}$ and *class 4* the original minus weights $w_{L5/R5}$ and weights $w_{L6/R6}$. Thus class 4 contains only what is neccessary to implement Mittelstaedt's model, and classes 2 and 3 are the two possible intermediate topologies between it and the evolved network. These were used to re-evolve PI behaviour starting from weight values of zero in GA runs where the network topology was not allowed to mutate. Eight runs of 1500 generations were performed for each. The best agent in each of the final populations was tested in 1000 trials and scored for the number of times it reached the nest within the time limit. The best agent in each class was then evolved for a further 20000 generations, and tested against the agent from 1500 generations. Only the class 4 agent showed an improvement, and was used in place of the generation 1500 agent. The generation 1500 agent was retained in the other classes. The four agents were then ranked for fitness. Each was tested six times for 1000 trials against the agents immediately above and below it, to determine if the ranking was statistically significant. The ranking order is class 1 > class 2 > class 3 > class 4. All differences were significant at $p < 0.01$ (Mann
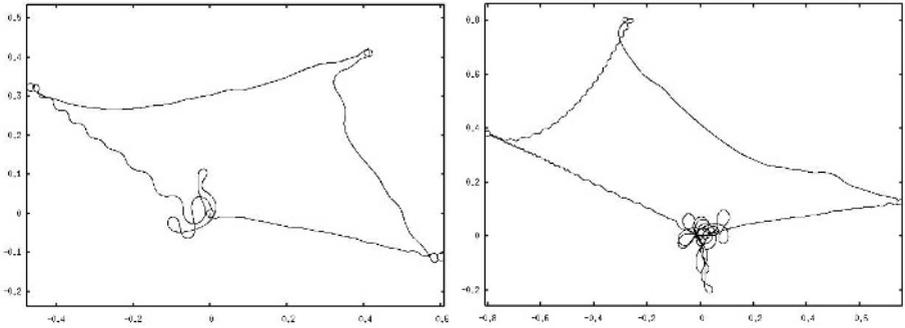
Fig. 3. Behaviour of the full evolved model. Left: the agent visits three beacons, starting from the nest (bottom centre) travelling anticlockwise. The beginning of a search pattern is visible before it reaches the nest. Right: a similar anticlockwise journey, but here the nest has been removed so that the agent searches until the trial times out.
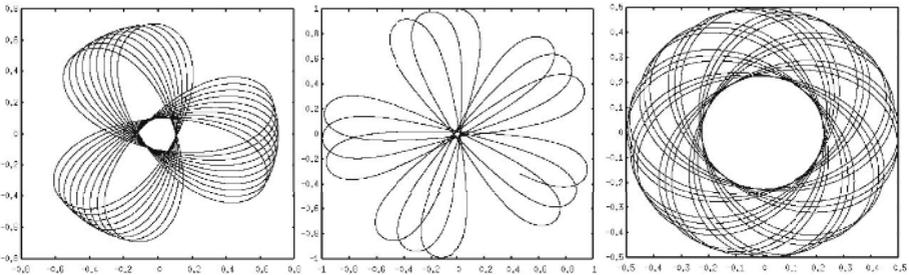

Fig. 4. Agent trajectories plotted in x, y space using the basic Mittelstaedt model with maximum turning rate parameter $k = 8$, initial conditions $x_0 = 0, \theta_0 = 0$ and, moving from left to right $y_0 = 0.1, y_0 = 0.5, y_0 = 1.0$. Clearly the initial condition influences the pattern.

Whitney U test). Percentage success rates for classes 1 to 4 were 97.4, 89.1, 81.1 and 69.5 respectively over 6000 trials.

2.6   Analysis of Searching Behaviour

The ModCTRNN PI agent shows a searching behaviour when it reaches its estimate of the nest location (Fig.3). The shape of this trajectory is important in determining how likely the agent is to locate its nest within the time limit. Trajectories whose density profile closely matches that of the probable nest location (which is usually not exact where the agent expects it to be due to cumulative PI errors) should be most efficient.

In the absence of noise Mittelstaedt's model can constitute a complete PI system, since the HV is also the agent's true location. We begin with this system, assuming the agent's speed is constant at 1, and build in further features derived
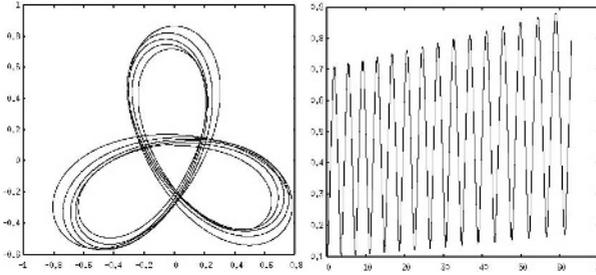
Fig. 5. The basic Mittelstaedt model ($k = 8, x_0 = 0, y_0 = 0.01, \theta_0 = 0$) augmented with a decay term in the turning rate equation, $e^{-\alpha t}$, where $\alpha = 0.01$. Left, the trajectory plotted in x, y space, right, the agent's distance from the origin plotted over time. The pattern clearly gets wider over time.
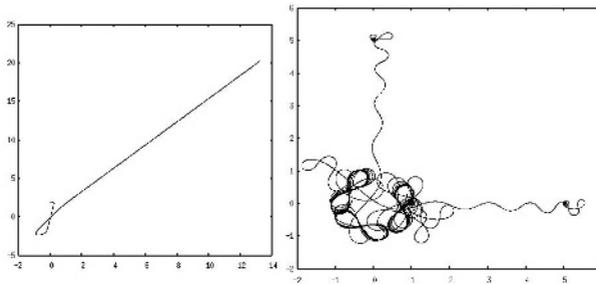

Fig. 6. Left: the addition of sigmoidal functions to the turning rate equation of the basic model ($k = 8, x_0 = 0, y_0 = 2, \theta_0 = 0$) allows the agent to become trapped on an outward diagonal trajectory. Right: the addition of stateful output units ($\tau = 1$) to the basic model ($k = 8, \theta_0 = 0$) causes the trajectories to differentiate into an initial homing phase, followed by a searching phase whose size is independent of the initial release point for the three cases tested here ($x_0 = 0, y_0 = 5$ and $x_0 = 5, y_0 = 0$ and $x_0 = 1, y_0 = 0$).

from the evolved network, in order to build intuition about how the search patterns are generated. We remove all noise from the system to aid analysis, but note that in reality noise would cause uncertainty in the nest location, and could knock the agent out of unstable trajectories. The three state equations to be used are:

$$\frac{dx}{dt} = \cos\theta, \frac{dy}{dt} = \sin\theta, \frac{d\theta}{dt} = k(x\sin\theta - y\cos\theta)$$

where k is the agent's maximum turning rate. Efficiency can obviously increase the faster an agent travels since it covers more ground, but we are interested in the efficiency of the *shape* of the search pattern. Therefore we transfer a degree of freedom from the agent's speed to its turning rate. In polar form:

$$\frac{dr}{dt} = \cos(\theta - \gamma), \frac{d\gamma}{dt} = \frac{1}{r}\sin(\theta - \gamma), \frac{d\theta}{dt} = kr\sin(\theta - \gamma)$$

This system can be derived from the evolved network by assuming the HV is updated without errors, that the output neurons have a linear response and are always at equilibrium and by neglecting w_{L6}, w_{R6}. The rectangular and polar systems were numerically integrated using the Runge-Kutta fourth order method [12] using a step size of 0.00005, and plotted together to check they were identical. The system displays behaviour similar to that of a spirograph [13] (see Fig.4). Trajectories show no division into homing and search phases, unlike the evolved agent. Defining $\delta = \theta - \gamma$, and plotting in (r, δ) phase space (not shown) all initial conditions (except those with $\delta_0 = \pm\pi$ or 0) appear to give closed trajectories in (r, δ) space, with no initial transient. Two fixed points are at $r = \sqrt{\frac{1}{k}}, \delta = \pm\frac{\pi}{2}$, which linearisation shows are neutrally stable centres, corresponding to circles in (r, γ) space. Defining $\phi = \delta \pm \frac{\pi}{2}$ and $\rho = r - \sqrt{\frac{1}{k}}$ we have a reversible system, since $\dot{\rho}(\rho, -\phi) = -\dot{\rho}(\rho, \phi)$ and $\dot{\phi}(\rho, -\phi) = \dot{\phi}(\rho, \phi)$ (where $\dot{\rho}, \dot{\phi}$ indicate derivatives) [14]. It follows that all trajectories sufficiently close to the fixed points are closed, implying that all loops of a given trajectory in (r, γ) space are congruent in this region.

As stated in our previous work [7], the time constants of weights w_{L2}, w_{R2} are small enough to allow the HV to decay significantly during the course of a journey, but this decay is approximately compensated for by the decay of weights w_{L4}, w_{R4}. We refer to this as leakage normalisation, since it scales down the HV integrator inputs to approximately balance the decay of the HV values, thus restoring accurate PI. If we assume this process is perfect we can derive an extension of the above DS which includes the effect of the HV decay process. The HV is still accurate, but is scaled by an exponentially decaying coefficient over the course of the journey. This can be modelled by amending $\frac{d\theta}{dt}$:

$$\frac{d\theta}{dt} = ke^{-\alpha t}(x\sin\theta - y\cos\theta)$$

$\frac{d\theta}{dt}$ will eventually reach zero in this scheme, (which does not happen in the full network model), and the agent will stop turning, but it is sufficient to show that the leakage normalisation mechanism can generate a search pattern which gets wider over time (see Fig.5), as is seen in *C. fortis* [15].

The output units of the ModCTRNN model are sigmoidal, but do not reach saturation during homing and search (data not shown). To see why output saturation might have a deleterious effect on homing and search we can amend $\frac{d\theta}{dt}$:

$$\frac{d\theta}{dt} = k\left(\sigma(x\sin\theta) - \sigma(y\cos\theta)\right)$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$. Saturation causes dead zones to appear around the two equilibrium values of θ (the stable homeward and unstable outward directions), where both outputs are saturated on or off, when x and y have similar magnitudes, thus making the unstable equilibrium neurally stable. This can trap the

agent on an outward trajectory along a diagonal (see Fig.6). Thus we can suggest a good reason why the outputs of R_L, R_R should be unsaturated during homing and search.

The output units in the ModCTRNN are stateful leaky integrators. We have so far treated them as being at equilibrium, but we can include stateful linear output units in the DS model as follows:

$$\frac{d\theta}{dt} = k(R_L - R_R), \frac{dR_L}{dt} = \frac{1}{\tau}(-R_L + x\sin\theta), \frac{dR_R}{dt} = \frac{1}{\tau}(-R_R + y\cos\theta)$$

Now for the first time, we obtain trajectories with distinct homing and search phases (see Fig.6). The example in the figure shows the agent converging onto a figure of eight shape from three initial positions. The figure eight is the same size irrespective of the initial release point. The agent is also seen to oscillate either side of the direct homeward trajectory during homing as the full model does.

3   Discussion

The topology experiment results strongly suggest that the weights in the evolved network which cannot be considered as directly implementing Mittelstaedt's model (weights $w_{L5}, w_{R5}, w_{L6}, w_{R6}$) none the less perform some adaptive role, and that removing w_{L6}, w_{R6} has a greater effect than removing w_{L5}, w_{R5}. Our DS models suggest that leakage normalisation could cause the agent's search pattern to get wider over time, which might increase the search efficiency, but our previous work [7] has only shown weak evidence for this, since individual searches do not get noticeably wider as w_{L4}, w_{R4} decays. We therefore still cannot state the function of the normalisation process. Consideration of linear stateful output neurons shows the agent's trajectory already significantly different from that of Mittelstaedt's model, and considerably more complex (now with 5 variables, not 3). Since w_{L6}, w_{R6} cannot affect HV update, they must function to further modify the agent's search pattern towards that seen in the full model. Further work along similar lines will be needed to finally resolve these issues.

Overall, we conclude that the use of the ModCTRNN model has proved useful for evolving a PI agent, and that whilst the model is slightly more complex than the CTRNN, the resulting PI network is both simple and amenable to considerable analysis and understanding. We hope the ModCTRNN will prove useful for other evolutionary robotics projects in the future.

Acknowledgements

References

1. Mittelstaedt, H.: Control systems of orientation in insects. Annu. Rev. Entomol. **7** (1962) 177–198
2. Mittelstaedt, H.: Analytical cybernetics of spider navigation. In Neurobiology of arachnids (ed. F. Barth) (1985) 298–318 Berlin:Springer.
3. Müller, M. and Wehner, R.: Path integration in desert ants, *Cataglyphis fortis.* Proc. Natl. Acad. Sci. USA **85** (1988) 5287–5290
4. Hartmann, G. and Wehner, R.: The ant's path integration system: a neural architecture. Biol. Cybern. **73** (1995) 483–497
5. Wittmann, T. and Schwegler, H.: Path integration - a neural model. Biol. Cybern. **73** (1995) 569–575
6. DaeEun, K. and Hallam, J.: Neural network approach to path integration for homing navigation. Proceedings of Simulation of Adaptive Behaviour: From Animals to Animats 6. MIT Press (2000)
7. Vickerstaff, R. J. and Di Paolo, E. A: Evolving neural models of path integration. (submitted)
8. Wehner, R.: Desert ant navigation: how miniature brains solve complex task. J. Comp. Physiol. A. **189** 579–588
9. Benhamou, S. and Séguinot, V.: How to find one's way in the labyrinth of path integration models. J. Theor. Biol. **174** (1995) 463–466
10. Maurer, R. and Séguinot, V.: What is modelling for? A critical review of models of path integration. J. Theor. Biol. **175** (1995) 457–475
11. Beer, R. D.: Intelligence as adaptive behaviour: an experiment in computational neuroethology. Academic Press: Boston. (1990)
12. Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P.: Numerical recipes in C: 2nd edition Cambridge University Press
13. Weisstein, E. W.: Spirograph. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/Spirograph.html
14. Strogatz, S. H.: Nonlinear dynamics and chaos. Perseus Group Books (2001)
15. Wehner, R. and Srinivasan, M. V.: Searching behaviour of desert ants, genus *Cataglyphis* (Formicidae, Hymenoptera). J. Comp. Physiol. A. **142** (1981) 315–338