Karbowski model in 2-D

Sebastian James

Adaptive Behaviour Research Group, Department of Psychology, The University of Sheffield, Sheffield, UK

* Correspondence: seb.james@sheffield.ac.uk

1. Introduction

This is a working document starting from the model described in Eqs. 2 and 4 in "Model of the Early Development of Thalamo-Cortical Connections and Area Patterning via Signaling Molecules", Karbowski & Ermentrout, 2004 [2]. I then consider extending this model into two dimensions.

2. One dimensional system

Equations 1-4 of [2] is the full set of differential equations describing their system:

$$\frac{\partial c_i(x,t)}{\partial t} = -\alpha_i c_i(x,t) + \beta_i n(x,t)[a_i(x,t)]^k \tag{1}$$

$$\frac{\partial a_i(x,t)}{\partial t} = \frac{\partial J_i(x,t)}{\partial x} + \alpha_i c_i(x,t) - \beta_i n(x,t)[a_i(x,t)]^k \tag{2}$$

$$n(x,t) + \sum_{i=1}^{N} c_i(x,t) = 1 \tag{3}$$

with the flux current

$$J_i(x,t) = D\frac{\partial a_i(x,t)}{\partial x} - a_i\left(\gamma_{Ai}\frac{\partial \rho_A(x)}{\partial x} + \gamma_{Bi}\frac{\partial \rho_B(x)}{\partial x} + \gamma_{Ci}\frac{\partial \rho_C(x)}{\partial x}\right) \tag{4}$$

Note that (unlike in the original paper) I have been explicit about the space and time dependence of the state variables a_i, c_i, J_i, n, ρ_A, ρ_B and ρ_C. Note also that i *always* refers to the thalamo-cortial connection type; there is one equation system for each connection type. Keep this in mind when reading the sums in the maths below. Note finally that I've not defined all the terms in this working document, but I've used the same variable names as in the paper [2].

By inspection of Eqs. 1 and 2, it can be seen that Eq. 2 can also be written:

$$\frac{\partial a_i(x,t)}{\partial t} = \frac{\partial J_i(x,t)}{\partial x} - \frac{\partial c_i(x,t)}{\partial t} \tag{5}$$

Both forms of the equation are useful; Eq. 2 is closer to the way the numerical solution is found in the code, in which a Runge-Kutta computation is carried out first for a and then for c.

2.1. Boundary condition

Karbowski and Ermentrout apply the 'sealed-end' boundary condition (a term that I've found used in cable theory): $J(0,t) = J(L,t) = 0$. x is in the range $[0, L]$.

$$J_i(x,t)\Big|_{boundary} = D\frac{\partial a_i(x,t)}{\partial x} - a_i\left(\gamma_{Ai}\frac{\partial \rho_A(x)}{\partial x} + \gamma_{Bi}\frac{\partial \rho_B(x)}{\partial x} + \gamma_{Ci}\frac{\partial \rho_C(x)}{\partial x}\right)\Big|_{boundary} = 0 \tag{6}$$

which is a 'Robin boundary condition'; a linear combination of the derivative of a_i and the value a_i. More specifically, it's an 'insulating boundary condition' (because it is set to 0) (from the diffusion-convection equation world).

K & E claim that requiring $J(0) = J(L) = 0$ "implies that the total number of axonal branches and connections of any given type in the system is constant" and that this statement is equivalent to saying that

$$\int_0^L [a_i(x,t) + c_i(x,t)]dx \tag{7}$$

is not a function of time. To show this, start from Eq. 5:

$$\frac{\partial a_i}{\partial t} = \frac{\partial J_i}{\partial x} - \frac{\partial c_i}{\partial t} \tag{8}$$

$$\begin{aligned} \implies \frac{\partial J_i}{\partial x} &= \frac{\partial a_i}{\partial t} + \frac{\partial c_i}{\partial t} \\ &= \frac{\partial}{\partial t}(a_i + c_i) \end{aligned} \tag{9}$$

(because differentiation is linear). So, integrate this system spatially from 0 to L:

$$\begin{aligned} \int_0^L \frac{\partial J_i}{\partial x}dx &= \int_0^L \frac{\partial}{\partial t}(a_i + c_i)dx \\ \implies \int_0^L \partial J_i &= \int_0^L \frac{\partial}{\partial t}(a_i + c_i)dx \\ \implies J(L) - J(0) &= \frac{\partial}{\partial t}\int_0^L (a_i + c_i)dx \end{aligned} \tag{10}$$

because $\int_0^L \partial J_i$ is just the sum of the changes in the value of J, which is equal to the difference between the final value and the initial value. Finally, because our no-flux boundary conditions state that $J(0) = J(L) = 0$, then

$$\frac{\partial}{\partial t}\int_0^L (a_i + c_i)dx = 0 \tag{11}$$

3. Two dimensional system

Extending the system above to two spatial dimensions changes x into a two-dimensional **x** and changes the flux term, $J_i(x,t)$ as follows:

$$\frac{\partial c_i(\mathbf{x},t)}{\partial t} = -\alpha_i c_i(\mathbf{x},t) + \beta_i n(\mathbf{x},t)[a_i(\mathbf{x},t)]^k \tag{12}$$

$$\frac{\partial a_i(\mathbf{x},t)}{\partial t} = \nabla \cdot \mathbf{J}_i(\mathbf{x},t) - \frac{\partial c_i(\mathbf{x},t)}{\partial t} \tag{13}$$

$$n(\mathbf{x},t) + \sum_{i=1}^{N} c_i(\mathbf{x},t) = 1 \tag{14}$$

with the flux current

$$\mathbf{J}_i(\mathbf{x},t) = D\nabla a_i(\mathbf{x},t) - a_i\big(\gamma_{Ai}\nabla\rho_A(\mathbf{x}) + \gamma_{Bi}\nabla\rho_B(\mathbf{x}) + \gamma_{Ci}\nabla\rho_C(\mathbf{x})\big) \tag{15}$$

The boundary condition that I'd like to apply to these equations is:

$$\mathbf{J}_i(\mathbf{x},t)\Big|_{boundary} = 0 \tag{16}$$

to agree with the boundary condition applied in the paper for the 1D system.

3.1. Computing div J: approach 1

In order to compute $\nabla \cdot \mathbf{J}_i(\mathbf{x}, t)$, I need to do a little work. First, I point out that

$$\mathbf{g}(\mathbf{x}) \equiv \left(\gamma_{Ai} \nabla \rho_A(\mathbf{x}) + \gamma_{Bi} \nabla \rho_B(\mathbf{x}) + \gamma_{Ci} \nabla \rho_C(\mathbf{x})\right) \tag{17}$$

is a static vector field; once computed at the start of the simulation, it is time-invariant. $\mathbf{J}_i$ is thus:

$$\mathbf{J}_i(\mathbf{x}, t) = D \nabla a_i(\mathbf{x}, t) - a_i \mathbf{g}(\mathbf{x}) \tag{18}$$

Taking the divergence of Eq. 18:

$$\nabla \cdot \mathbf{J}_i(\mathbf{x}, t) = \nabla \cdot \left(D \nabla a_i(\mathbf{x}, t) - a_i \mathbf{g}(\mathbf{x}) \right) \tag{19}$$

Now simply multiply the current values of a_i and $\mathbf{g}$ and compute an approximation to ∇a_i, resulting in a vector field which is $\mathbf{J}$. The boundary condition can be forced by setting this to 0 for boundary hexes (ugly, and quite possibly completely fallacious). Finally, compute $\nabla \cdot \mathbf{J}_i(\mathbf{x}, t)$ using the result that divergences can be simplified by means of Gauss's Theorem. This states that (in a two-dimensional system) the area integral of the divergence of a vector field $\mathbf{f}$ is equal to the integral around the boundary of the field dotted with the normal to the boundary:

$$\iint_A \nabla \cdot \mathbf{f} \, dx dy = \oint_c \mathbf{f} \cdot d\hat{\mathbf{n}} \tag{20}$$

See, for example [1], Sect. 1.11, p 58.

This can be used to find the average value of the divergence of $\mathbf{f}$ over the area of a hexagon; $\mathbf{f}_0$ is the average value of the vector field at the centre of the hexagon; $\mathbf{f}_1$–$\mathbf{f}_6$ are the average values of the field in the 6 neighbouring hexagons. The contour integral can be discretized for this hexagonal region, which has area Ω and a side length v:

$$\begin{aligned}
\frac{1}{\Omega} \oint_c \mathbf{f} \cdot d\hat{\mathbf{n}} &\approx \frac{1}{\Omega} \sum_{j=1}^{6} \frac{\mathbf{f}_j + \mathbf{f}_0}{2} \cdot \hat{\mathbf{n}} \; v \\
&= \frac{1}{\Omega} \sum_{j=1}^{6} \frac{\mathbf{f}_j^x + \mathbf{f}_0^x}{2} \cdot \hat{\mathbf{n}} \; v + \sum_{j=1}^{6} \frac{\mathbf{f}_j^y + \mathbf{f}_0^y}{2} \cdot \hat{\mathbf{n}} \; v \\
&= \frac{1}{\Omega} \sum_{j=1}^{6} \frac{\mathbf{f}_j^x + \mathbf{f}_0^x}{2} \cos(60 \times (j-1)) \; v + \frac{1}{\Omega} \sum_{j=1}^{6} \frac{\mathbf{f}_j^y + \mathbf{f}_0^y}{2} \sin(60 \times (j-1)) \; v
\end{aligned} \tag{21}$$

An advantage to using this approach is that if it is required that the *divergence* be 0 across the boundary then for those edges of the hex that lie on the boundary $\mathbf{f}_j$ can be set equal to $\mathbf{f}_0$ (this is the ghost cell method), satisfying this boundary condition, whilst permitting flow through the hex parallel to the boundary. However, it is not so easy to choose a ghost cell with properties that will force the *value* of $\mathbf{f}_0$ to 0.

3.2. Computing div J: approach 2

An alternative approach is to apply some vector calculus identities to work out the divergence of $\mathbf{J}_i(\mathbf{x})$. Because the divergence operator is distributive, Eq. 19 can be expanded:

$$\nabla \cdot \mathbf{J}_i(\mathbf{x}) = \nabla \cdot \left(D \nabla a_i(\mathbf{x}, t) \right) - \nabla \cdot \left(a_i \mathbf{g}(\mathbf{x}) \right) \tag{22}$$

D is a constant and applying the vector calculus product rule identity to $\nabla \cdot \big(a_i \mathbf{g}(\mathbf{x})\big)$:

$$\nabla \cdot \mathbf{J}_i(\mathbf{x}) = D\nabla \cdot \nabla a_i(\mathbf{x}, t) - \big(a_i(\mathbf{x}, t)\nabla \cdot \mathbf{g}(\mathbf{x}) + \mathbf{g}(\mathbf{x}) \cdot \nabla a_i(\mathbf{x}, t)\big) \tag{23}$$

(to prove $\nabla \cdot D\nabla a = D\nabla \cdot \nabla a$, apply the product rule and note that $\nabla D = 0$ for constant D). Expanding the brackets we have:

$$\boxed{\nabla \cdot \mathbf{J}_i(\mathbf{x}) = D\nabla \cdot \nabla a_i(\mathbf{x}, t) - a_i(\mathbf{x}, t)\nabla \cdot \mathbf{g}(\mathbf{x}) - \mathbf{g}(\mathbf{x}) \cdot \nabla a_i(\mathbf{x}, t)} \tag{24}$$

which has three elements to compute: the Laplacian of $a_i(\mathbf{x}, t)$; a time-independent modulator of $a_i(\mathbf{x}, t)$ [because $\nabla \cdot \mathbf{g}(\mathbf{x})$ is a time-independent static field]; and the dot product of the static vector field $\mathbf{g}(\mathbf{x})$ and the gradient of $a_i(\mathbf{x}, t)$. These three terms (named, very sensibly term1, term2 and term3) are computed in the compute_divJ() method of both the RD_2D_Karb and RD_James classes.

Each of the divergences can be simplified by means of Gauss's Theorem. Reference [3] presents an application of Gauss's Theorem to the solution of Poisson's equation, which contains a Laplacian term, on an hexagonal grid. The computation of the mean value of our Laplacian, $D\nabla \cdot \nabla a_i(\mathbf{x}, t)$, across the area of one hexagon located at position $\mathbf{p}_0$, with neighbours at positions $\mathbf{p}_1$–$\mathbf{p}_6$ is:

$$\boxed{D\nabla \cdot \nabla a_i(\mathbf{p}_0, t) = D\frac{2}{3d^2}\sum_{j=1}^{6}\big(a_i(\mathbf{p}_j) - a_i(\mathbf{p}_0)\big)} \tag{25}$$

where d is the centre-to-centre distance between hexes in the grid. This is term1. Let's work it through. In the following, v is the length of each of the 6 edges of the hexagon's perimeter and $v = d/\sqrt{3}$. $d\gamma$ is an infinitessimally small distance along the perimeter of the hexagon. The area, Ω, of each hexagon in the lattice is $\frac{\sqrt{3}}{2}d^2$.

$$\begin{aligned}
\frac{1}{\Omega}\iint_A \nabla \cdot \nabla a_i(\mathbf{x})dxdy &= \frac{1}{\Omega}\oint_c \frac{\partial a_i}{\partial \hat{\mathbf{n}}}d\gamma \\
&\approx \frac{1}{\Omega}\sum_{j=1}^{6}\left.\frac{\partial a_i(\mathbf{p}_j)}{\partial \hat{\mathbf{n}}}\right|_{mid} v \\
&= \frac{2}{\sqrt{3}d^2}\sum_{j=1}^{6}\frac{a_i(\mathbf{p}_j) - a_i(\mathbf{p}_0)}{d}\frac{d}{\sqrt{3}} \\
&= \frac{2}{3d^2}\sum_{j=1}^{6}\big(a_i(\mathbf{p}_j) - a_i(\mathbf{p}_0)\big)
\end{aligned} \tag{26}$$

Applying Gauss's Theorem to the second term, the computation of $a_i(\mathbf{p}_0, t)\nabla \cdot \mathbf{g}(\mathbf{p}_0)$ can be written out in a

similar way:

$$\begin{aligned}
\frac{1}{\Omega}\iint_A a_i \nabla\cdot\mathbf{g}\; dxdy &= \frac{a_i(\mathbf{p}_0)}{\Omega}\oint_c \mathbf{g}\cdot d\hat{\mathbf{n}} \\
&\approx \frac{a_i(\mathbf{p}_0,t)}{\Omega}\sum_{j=1}^{6}\frac{\mathbf{g}_j+\mathbf{g}_0}{2}\cdot\hat{\mathbf{n}}\; v \\
&= \frac{2a_i(\mathbf{p}_0,t)v}{\sqrt{3}d^2}\left(\sum_{j=1}^{6}\frac{\mathbf{g}_j^x+\mathbf{g}_0^x}{2}\cdot\hat{\mathbf{n}}+\sum_{j=1}^{6}\frac{\mathbf{g}_j^y+\mathbf{g}_0^y}{2}\cdot\hat{\mathbf{n}}\right) \\
&= \frac{2a_i(\mathbf{p}_0,t)d}{\sqrt{3}d^2\sqrt{3}}\left(\sum_{j=1}^{6}\frac{\mathbf{g}_j^x+\mathbf{g}_0^x}{2}\cdot\hat{\mathbf{n}}+\sum_{j=1}^{6}\frac{\mathbf{g}_j^y+\mathbf{g}_0^y}{2}\cdot\hat{\mathbf{n}}\right) \\
&= \frac{a_i(\mathbf{p}_0,t)}{3d}\left(\sum_{j=1}^{6}\left(\mathbf{g}_j^x+\mathbf{g}_0^x\right)\cos(60\times(j-1))+\sum_{j=1}^{6}\left(\mathbf{g}_j^y+\mathbf{g}_0^y\right)\sin(60\times(j-1))\right)
\end{aligned} \tag{27}$$

$$\boxed{a_i(\mathbf{x},t)\nabla\cdot\mathbf{g}(\mathbf{x}) = \frac{a_i(\mathbf{p}_0,t)}{3d}\left(\sum_{j=1}^{6}\left(\mathbf{g}_j^x+\mathbf{g}_0^x\right)\cos(60\times(j-1))+\sum_{j=1}^{6}\left(\mathbf{g}_j^y+\mathbf{g}_0^y\right)\sin(60\times(j-1))\right)} \tag{28}$$

The final expression is suitable for a numerical computation and can be found in compute_divJ() as term2. Lastly, the final term in Eq. 24 is the scalar product of two vector fields which I'll carry out simply with Cartesian x and y components - so I'll re-use my spacegrad2D function to find ∇a_i. That can be found as term3.

I think that the advantage of separating this computation out is that I can ensure that, at the boundary, **J** resulting from the first term remains 0, by the ghost cell method, then I can tailor $\mathbf{g}(\mathbf{x})$ so that it, and its normal derivative go to 0 at the boundary, ensuring that the second and third terms also contribute nothing to **J**. To do this, I'll apply a fairly sharp logistic function based on distance to the boundary to $\mathbf{g}(\mathbf{x})$.

3.3. Computing a and c

See code in rd_james.h to inspect the Runge-Kutta numerical computation. Note that I include some tests to ensure that a_i does not become negative, c_i does not exceed 1 and n does not become negative (i.e. that $\sum c_i$ does not exceed 1).

4. Signalling

Gao et al. show ephrin-A5 inhibits the growth of axons in neurons of the medial thalamus, while allowing neurons from the lateral thalamus to branch. "The EphA5 receptor and its ligand, ephrin-A5 are expressed in complementary patterns in the thalamus and their cortical targets. *Additionally*, ephrin-A5 specifically inhibits neurite outgrowth of medial thalamic neurons from limbic nuclei and sustains neurite outgrowth of lateral thalamic neurons from primary sensory and motor nuclei. ... Further analysis using hippocampal neurons indicated that ephrin-A5 inhibits primarily the growth of axons detected with antibody against the axon-specific marker τ(60–70% reduction in axonal length), although a mild inhibition of dentritic growth also was observed (≈20% reduction).

5. Two dimensional system with N TCs and M gradients

We're modifying the system above so that we have N TC populations and M curated gradients.

$$\frac{\partial c_i(\mathbf{x},t)}{\partial t} = -\alpha_i c_i(\mathbf{x},t) + \beta_i n(\mathbf{x},t)[a_i(\mathbf{x},t)]^k \tag{29}$$

$$\frac{\partial a_i(\mathbf{x},t)}{\partial t} = \nabla \cdot \mathbf{J}_i(\mathbf{x},t) - \frac{\partial c_i(\mathbf{x},t)}{\partial t} \tag{30}$$

$$n(\mathbf{x},t) + \sum_{i=1}^{N} c_i(\mathbf{x},t) = 1 \tag{31}$$

with the flux current

$$\mathbf{J}_i(\mathbf{x},t) = D\nabla a_i(\mathbf{x},t) - a_i \sum_{j=1}^{M} \left(\gamma_{i,j} \nabla \rho_j(\mathbf{x})\right) \tag{32}$$

The boundary condition that I'd like to apply to these equations is:

$$\mathbf{J}_i(\mathbf{x},t)\Big|_{boundary} = 0 \tag{33}$$

6. Analyse competition in the 2D Karbowski system

First sight of the 2D Karbowski system, as implemented in rd_james.h/james1.cpp with N=2 and M=0 is that competition is very weak indeed. Over the simulation timescale during which the guidance gradients act (about a thousand steps) pretty much nothing happens to the competition between the two TC connection types. If one waits a very long time, then eventually, there is a split between the two, with one winning over the other, but we're talking timescales on the order of 30000 and this effect is entirely dwarfed by the effect of the guidance molecules in the system.

To see if competition is weak across the parameter space, I did a grid search, varying the parameter α [2:0.15:5], β [2:0.15:5] and k [2:0.1:4]. I left D fixed at 0.1 and as there were no guidance molecules, the γs were all 0.

Figure 1: A search of a region of parameter space around k=3, α=3, β=3. The colour of each point represents a normalised scaling of the sum of squared differences between elements of c_0 and c_1 after 5000 iterations of the simulation. The maximum value of this metric was 0.0066; the minimum was 1.7×10^{-5}. For comparison, the value of the metric for a guided system which separated into two clear regions at 5000 steps was 722.

I created two methods to run through the parameter space. One, called analysis/scanspace.py, creates a json file for each parameter set, then calls the program james1c, and finally calls sos_centroid_analyse() to add each result to logs/scanspace.h5. The problem with this method is that it repeatedly creates the morph::HexGrid, adding lots of worthless computation. To fix this, I wrote ps_2N0M.cpp, which uses a single RD_James instance, which it re-sets after each simulation, without having to re-compute the HexGrid. This can evaluate 5000 timesteps of one parameter set with the Hex to Hex spacing set to 0.02 in 1.3 seconds (on threadbeast). ps_2N0M.cpp runs the simulations, but it doesn't apply the final analysis step of computing the sum of squared differences between c0 and c1. This is carried out by analysis/ps_2N0M_postproc.py and plotting by analysis/ps_2N0M_plot.py. See Fig. 1.

7. Two dimensional system with changed competition

The symmetry of the terms for $\pm\alpha_i c_i$ and $\pm\beta_i n a_i$ in Equations 1 & 2 results directly from the simple kinetic reaction:

$$n + a_i \underset{\alpha_i}{\overset{\beta_i}{\rightleftharpoons}} c_i \tag{34}$$

If I want to change the competition between axons from different thalamocortical regions, I need to consder exactly what it is about Eq. 34 that I want to modify, so that the model is changed with some sort of guiding principle.

Here are some modifications to the equations to enable competition between regions dreamed up in just such a flagrant, un-principled way:

In the following, I'll use these definitions:

Let $\hat{a}_i$ be the sum of the branching densities of all axon types except i.

$$\hat{a}_i(\mathbf{x}, t) = \sum_{j \neq i}^{N} a_j(\mathbf{x}, t) \tag{35}$$

Define the static vector field **g** as:

$$\mathbf{g}_i(\mathbf{x}) = \sum_{j=1}^{M} \left(\gamma_{i,j} \nabla \rho_j(\mathbf{x})\right) \tag{36}$$

7.1. Allow the branching rate of all other TC axon types to change $\dot{a}_i$

Add a term to the Karbowski equations which reduces the branching rate change based on the proximity of branching of other TC axon types:

$$\frac{\partial c_i(\mathbf{x}, t)}{\partial t} = -\alpha_i c_i(\mathbf{x}, t) + \beta_i n(\mathbf{x}, t)[a_i(\mathbf{x}, t)]^k \tag{37}$$

$$\frac{\partial a_i(\mathbf{x}, t)}{\partial t} = \nabla \cdot \mathbf{J}_i(\mathbf{x}, t) - \frac{\partial c_i(\mathbf{x}, t)}{\partial t} - \frac{\epsilon_i a_i}{N-1} \sum_{j \neq i}^{N} a_j(\mathbf{x}, t)^l \tag{38}$$

$$n(\mathbf{x}, t) = 1 - \sum_{i=1}^{N} c_i(\mathbf{x}, t) \tag{39}$$

with the flux current as given by Eq. 15 and Neumann boundary conditions as usual. This adds two parameter sets; ϵ and l. The factor $\frac{1}{N-1}$ ensures that the value of ϵ scales with N.

See the binary `james1` for the implementation.

The additional term could result from some sort of axon-axon repulsion as the axons grow into the cortex, or an effect that occurs only once they are growing within the cortical layer.

An implementation of this mechanism in `rd_james_comp1.cpp` does introduce competition quite nicely. Especially evident if β is large.

The problem with using this mechanism on its own, is that the diffusion can easily swamp the ϵ_i attenuation term, and if one level of branching for one TC type runs away, then the attenuation of all the others runs away.

One solution is to pass a_i through a sigmoid, suggesting that there might be a maximal branching density allowable.

7.2. Add an interaction with $\nabla \hat{a}_i$

There could be an additional term to interact with the other branching densities, to prevent the *branching* overlapping (rather than just relying on the competition between connections implied by the definition of $n(\mathbf{x}, t)$ in Eq. 31).

Change the flux current to:

$$\mathbf{J}_i(\mathbf{x}, t) = D \nabla a_i(\mathbf{x}, t) + \frac{\bar{a}_i(\mathbf{x}, t) F}{N-1} \nabla \hat{a}_i(\mathbf{x}, t) - a_i(\mathbf{x}, t)\, \mathbf{g}(\mathbf{x}) \tag{40}$$

where $\bar{a}_i$ is a sigmoid of height h, offset o and sharpness, s:

$$\bar{a}_i = \frac{h}{1 + e^{(o - s\, a_i)}} \tag{41}$$

The use of the sigmoid prevents the flux current from blowing up due to the $\nabla \hat{a}_i$ term. In this term (only), we model the idea that there must be an upper bound on the axonal branching density.

which should be another way to add competition to the system. The idea here is that branching density should flow away from regions where there is a high level of branching of other TC axon types.

Note that the $\nabla \hat{a}_i$ term is positive; if branching is high for the TC type i *and* for all other TC types, then diffusion of branching away from that location should be increased.

Dropping references to space and time arguments:

$$\mathbf{J}_i = D \nabla a_i + \frac{\bar{a}_i F}{N-1} \nabla \hat{a}_i - a_i\, \mathbf{g} \tag{42}$$

The divergence of $\mathbf{J}_i$ is

$$\begin{aligned} \nabla \cdot \mathbf{J}_i &= \nabla \cdot \left[D \nabla a_i + \frac{\bar{a}_i F}{N-1} \nabla \hat{a}_i - a_i \mathbf{g}_i \right] \\ &= D \nabla \cdot \nabla a_i + \frac{F}{N-1} \nabla \cdot \bar{a}_i \nabla \hat{a}_i - \nabla \cdot a_i \mathbf{g}_i \end{aligned} \tag{43}$$

because the divergence operator is distributive (and D & F are scalar constants). Applying the vector calculus product rule identity to $\nabla \cdot \left(\bar{a}_i \nabla \hat{a}_i \right)$ and $\nabla \cdot \left(a_i \mathbf{g}_i \right)$:

$$\nabla \cdot \mathbf{J}_i = D \nabla \cdot \nabla a_i + \frac{F}{N-1} \left(\bar{a}_i \nabla \cdot \nabla \hat{a}_i + \nabla \hat{a}_i \cdot \nabla \bar{a}_i \right) - \left(a_i \nabla \cdot \mathbf{g}_i + \mathbf{g}_i \cdot \nabla a_i \right) \tag{44}$$

Expanding the brackets we have:

$$\boxed{\nabla \cdot \mathbf{J}_i = D \nabla \cdot \nabla a_i + \frac{F}{N-1} \bar{a}_i \nabla \cdot \nabla \hat{a}_i + \frac{F}{N-1} \nabla \hat{a}_i \cdot \nabla \bar{a}_i - a_i \nabla \cdot \mathbf{g}_i - \mathbf{g}_i \cdot \nabla a_i} \tag{45}$$

which gives the five terms to compute in rd_james_comp2.h.

Result: With the sigmoid parameters set to $h = 2, o = 5, s = 0.5$, the range of stable values for F were $[0 - -6]$. At the upper end of this range, the competition effect was very slight.

7.3. Add an interaction with ∇n

Imagine that as connections are made to the cortical dendrites, a molecule is expressed. Axon branching should expand into areas with low connections, so the 'flux of branching' should follow the gradient of this 'connectivity molecule' from low n (region with few connection sites) to high n (region with many connection sites). I've expressed this by adding a new term to Eq. 32:

$$\mathbf{J}_i(\mathbf{x}, t) = D \nabla a_i(\mathbf{x}, t) \boxed{- a_i(\mathbf{x}, t) E \nabla n(\mathbf{x}, t)} - a_i(\mathbf{x}, t)\, \mathbf{g}(\mathbf{x}) \tag{46}$$

With the new term (boxed), it's probably worth going back to look at the method for computing $\nabla \cdot \mathbf{J}_i$. Let's drop references to the space and time arguments ($\mathbf{g}_i(\mathbf{x})$ becomes $\mathbf{g}_i$; $n(\mathbf{x}, t)$ becomes n; and so on). This allows a tidier version of Eq. 46 to be written down:

$$\mathbf{J}_i = D \nabla a_i - a_i E \nabla n - a_i(\mathbf{x}, t) \mathbf{g}_i \tag{47}$$

The divergence of $\mathbf{J}_i$ is

$$\begin{aligned}\nabla \cdot \mathbf{J}_i &= \nabla \cdot \left[D\nabla a_i - a_i E \nabla n - a_i \mathbf{g}_i \right] \\ &= D\nabla \cdot \nabla a_i - E\nabla \cdot a_i \nabla n - \nabla \cdot a_i \mathbf{g}_i\end{aligned} \tag{48}$$

because the divergence operator is distributive (and D & E are scalar constants). Applying the vector calculus product rule identity to $\nabla \cdot \big(a_i \nabla n\big)$ and $\nabla \cdot \big(a_i \mathbf{g}_i\big)$:

$$\nabla \cdot \mathbf{J}_i = D\nabla \cdot \nabla a_i - E\big(a_i \nabla \cdot \nabla n + \nabla n \cdot \nabla a_i\big) - \big(a_i \nabla \cdot \mathbf{g}_i + \mathbf{g}_i \cdot \nabla a_i\big) \tag{49}$$

Expanding the brackets we have:

$$\boxed{\nabla \cdot \mathbf{J}_i = D\nabla \cdot \nabla a_i - E a_i \nabla \cdot \nabla n - E\nabla n \cdot \nabla a_i - a_i \nabla \cdot \mathbf{g}_i - \mathbf{g}_i \cdot \nabla a_i} \tag{50}$$

which gives us 5 terms to compute in `compute_divJ()` in place of the original 3 terms in Eq. 24.

This approach simply increases how fast branching diffuses into free space; the $D\nabla a_i$ term already provides diffusion of branching.

7.4. Interaction with ∇n and $\hat{a}_i$ inhibits $\dot{a}_i$

Combines sections 7.1 and 7.3. Descending the gradient $a_i \nabla n$ expands axon branching into new regions; competition between a_i and $\hat{a}_i$ should prevent excessive branching into neighbouring regions.

$$\frac{\partial c_i}{\partial t} = -\alpha_i c_i + \beta_i n [a_i]^k \tag{51}$$

$$\frac{\partial a_i}{\partial t} = \nabla \cdot \mathbf{J}_i + \alpha_i c_i - \beta_i n [a_i]^k - \frac{\epsilon_i}{N-1} a_i \sum_{j \neq i}^{N} (a_j)^l \tag{52}$$

$$n = 1 - \sum_{i=1}^{N} c_i(\mathbf{x}, t) \tag{53}$$

$$\nabla \cdot \mathbf{J}_i = D\nabla \cdot \nabla a_i - E a_i \nabla \cdot \nabla n - E\nabla n \cdot \nabla a_i - a_i \nabla \cdot \mathbf{g}_i - \mathbf{g}_i \cdot \nabla a_i \tag{54}$$

Note that Eqs. 51–53 are not quite identical those in section 7.1. The difference is the $\frac{1}{N-1}$ factor and the inclusion of a_i in the third term of Eq. 52.

This is very promising! Initial branching conditions produce reliable regions. Initial conditions can be modified to simulate denucleated animal; region becomes smaller, but not non-existent.

7.5. $\hat{a}_i$ inhibits $\dot{a}_i$ more

In the config where I'm trying to make 5 barrel rows, the competition is less competitive than I imagine it could be. The initial condition is a blob of each of the 5 TC types of roughly the same size and I think the problem might be that the ϵ term is inhibiting all 5 TC types more or less equally.

This idea (to avoid attenuating the biggest a) didn't work but lead me to think about bounding a and the problem of a winner-takes-all situation that is easily seen.

7.6. Add interactions with $\nabla a_{p\neq i}$

A scheme similar to comp2 (section 7.2). However, rather than using the gradient $\nabla\hat{a}_i$, use all the gradients of the other TC types, with a ζ matrix to set the strengths of interaction.

Change the flux current to:

$$\mathbf{J}_i(\mathbf{x},t) = D\nabla a_i(\mathbf{x},t) + \bar{a}_i(\mathbf{x},t)\sum_{p\neq i}\zeta_{i,p}\nabla a_p(\mathbf{x},t) - a_i(\mathbf{x},t)\,\mathbf{g}(\mathbf{x}) \tag{55}$$

where $\bar{a}_i$ is a sigmoid of height h, offset o and sharpness, s:

$$\bar{a}_i = \frac{h}{1+e^{(o-s\,a_i)}} \tag{56}$$

used for the same reason as in section 7.2.

Dropping references to space and time arguments:

$$\mathbf{J}_i = D\nabla a_i + \bar{a}_i\sum_{p\neq i}\zeta_{i,p}\nabla a_p - a_i\,\mathbf{g} \tag{57}$$

The divergence of $\mathbf{J}_i$ is

$$\begin{aligned}\nabla\cdot\mathbf{J}_i &= \nabla\cdot\Big[D\nabla a_i + \sum_{p\neq i}\zeta_{i,p}\bar{a}_i\nabla a_p - a_i\,\mathbf{g}\Big]\\ &= D\nabla\cdot\nabla a_i + \sum_{p\neq i}\zeta_{i,p}\nabla\cdot\bar{a}_i\nabla a_p - \nabla\cdot a_i\mathbf{g}_i\end{aligned} \tag{58}$$

because the divergence operator is distributive (and D & ζ are scalar). Applying the vector calculus product rule identity to $\nabla\cdot\big(\bar{a}_i\nabla a_p\big)$ and $\nabla\cdot\big(a_i\mathbf{g}_i\big)$:

$$\boxed{\nabla\cdot\mathbf{J}_i = D\nabla\cdot\nabla a_i + \sum_{p\neq i}\zeta_{i,p}\big(\bar{a}_i\nabla\cdot\nabla a_p + \nabla a_p\cdot\nabla\bar{a}_i\big) - \big(a_i\nabla\cdot\mathbf{g}_i + \mathbf{g}_i\cdot\nabla a_i\big)} \tag{59}$$

which on expansion gives $3+2(N-1)$ terms terms to compute in rd_james_comp6.h (not yet implemented).

7.7. $\hat{a}_i$ *multiplicatively reduces diffusion of* a_i

Change the flux current to:

$$\mathbf{J}_i = D(1-\sigma(\hat{a}_i))\nabla a_i - a_i\,\mathbf{g} \tag{60}$$

where $\sigma(\hat{a}_i)$ denotes a sigmoid function:

$$\sigma(a) = \frac{1}{1+e^{(o-s\,a)}} \tag{61}$$

Let $\lambda = 1-\sigma$. Then $\nabla\cdot\mathbf{J}_i$ is

$$\nabla\cdot\mathbf{J}_i = \nabla\cdot\big(D\lambda\nabla a_i - a_i\,\mathbf{g}\big) \tag{62}$$

Product rules all round to give:

$$\boxed{\nabla \cdot \mathbf{J}_i = D\big(\lambda \nabla \cdot \nabla a_i + \nabla a_i \cdot \nabla \lambda\big) - \big(a_i \nabla . \mathbf{g} + \mathbf{g} \cdot \nabla a_i\big)} \tag{63}$$

Ah, I added in the direct competition mechanism of section 7.1 (though as refined in section 7.4) too (see rd_james_comp7.h).

7.8. Divisive normalization of a_i

Taking inspiration from the divisive normalization idea in neural networks [5], where the overall activity of a group of neurons modulates the output response of an individual member of the group, we will use the overall branching level of all of the axons in a layer to modulate the response, $a_i[j]'$, of the branching level in each element j (using this notation ($a_i[j]$) makes explicit that we're talking about a system containing n_e discrete elements indexed by j).

The response can be written out with the normalization equation [5]:

$$a_i[j]' = h \, \frac{(a_i[j])^q}{\xi^q + \sum_j^h (a_i[j])^q} \tag{64}$$

in which ξ and q are free parameters and n_e is the number of hex elements in the domain of computation (although n_e could also be chosen as a free parameter if required). Here, I'll let $\xi = 0$ and $q = 1$.

$a_i[j]'$, the 'branching response' at element j, then has to be used to drive connections, in practice, it is written back into $a_i[j]$ on every timestep.

That gives the following set of equations to express in the code:

$$\frac{\partial c_i}{\partial t} = -\alpha_i c_i + \beta_i n [a_i]^k \tag{65}$$

$$\frac{\partial a_i}{\partial t} = \nabla \cdot \mathbf{J}_i + \alpha_i c_i - \beta_i n [a_i]^k \tag{66}$$

$$a_i[j]' = n_e \, \frac{a_i[j]}{\sum_j^{n_e} a_i[j]} \tag{67}$$

(a_i' becomes a_i in the next computational step.)

$$n = 1 - \sum_{i=1}^{N} c_i \tag{68}$$

$$\nabla \cdot \mathbf{J}_i(\mathbf{x}) = D \nabla \cdot \nabla a_i - a_i \nabla \cdot \mathbf{g} - \mathbf{g} \cdot \nabla a_i \tag{69}$$

One issue with divisive normalization is that each TC type is normalized to have the same overall branching, and so changed initial conditions, such as reduced initial innervation for one field over another will not result is very different simulations.

An idea to get around this is to introduce an additional multiplier to the normalization, which is the mean initial branching density per element, d_i/n_e, so that Eq. 67 becomes

$$a_i[j]' = d_i \, \frac{a_i[j]}{\sum_j^{n_e} a_i[j]} \tag{70}$$

which is (making time explicit now):

$$a_i[j](t)' = \sum_j^{n_e} a_i[j](0) \, \frac{a_i[j](t)}{\sum_j^{n_e} a_i[j](t)} \tag{71}$$

One further extension is to note that in the K&E simulation, the quantity of axon branching in the system is bounded and the number of connections and branching quantity is related. So we could normalize the 'proportion of a_i which is not participating in connections'.

$$a_i[j](t)' = \sum_j^{n_e} \left[a_i[j](0) - c_i[j](t)\right] \frac{a_i[j](t)}{\sum_j^{n_e} a_i[j](t)} \tag{72}$$

The continuous equivalent, if the surface is called S, would be:

$$\begin{aligned} a_i(\mathbf{x},t) &= \oiint_S \left[a_i(\mathbf{x},0) - c_i(\mathbf{x},t)\right] dS \; \frac{a_i'(\mathbf{x},t)}{\oiint_S a_i'(\mathbf{x},t) dS} \\ &= \left[\oiint_S a_i(\mathbf{x},0) dS - \oiint_S c_i(\mathbf{x},t) dS\right] \frac{a_i'(\mathbf{x},t)}{\oiint_S a_i'(\mathbf{x},t) dS} \end{aligned} \tag{73}$$

which is a little more expensive as $\oiint_S c_i(\mathbf{x},t) dS$ has to be computed on each iteration.

7.9. Divisive normalization of a_i and $\hat{a}_i$ inhibits $\dot{a}_i$

This is a combination of code in section 7.4 (the competition part) and 7.8.

The separate regions compete evenly, as they are all normalized to have the same branching density.

7.10. Piecewise linear transfer function normalizes a_i

$$\frac{\partial c_i}{\partial t} = -\alpha_i c_i + \beta_i n [a_i]^k \tag{74}$$

$$\frac{\partial a_i}{\partial t} = \nabla \cdot \mathbf{J}_i + \alpha_i c_i - \beta_i n [a_i]^k \tag{75}$$

$$a_i' = \begin{cases} 0 & a_i < 0 \\ a_i & a_i \leq a_{max} \\ a_{max} & a_i > a_{max} \end{cases} \tag{76}$$

(a_i' becomes a_i in the next computational step.)

$$n = 1 - \sum_{i=1}^{N} c_i \tag{77}$$

$$\nabla \cdot \mathbf{J}_i(\mathbf{x}) = D \nabla \cdot \nabla a_i - a_i \nabla \cdot \mathbf{g} - \mathbf{g} \cdot \nabla a_i \tag{78}$$

This is implemented in rd_james_comp10.h.

7.11. Piecewise linear transfer function normalizes a_i plus competition

This is implemented in rd_james_comp11.h.

7.12. Divisive normalization of a_i plus competition with extended $\epsilon_{i,j}$

Allow each TC type to interact with the other $N-1$ TC types with varying magnitudes. This is an extension to the competition type in section 7.9. The N sized vector ϵ becomes an $N \times N$ matrix ϵ and Eq. 52 becomes

$$\frac{\partial a_i}{\partial t} = \nabla \cdot \mathbf{J}_i + \alpha_i c_i - \beta_i n [a_i]^k - \frac{1}{N-1} a_i \sum_{p \neq i}^{N} \epsilon_{i,p} \; a_p^l \tag{79}$$

7.13. Divisive normalization, additional competition mechanisms

Because the divisive normalization is so helpful in keeping the N branching variables balanced with respect to each other, I want to reintroduce the additional diffusion ideas developed earlier. First, introduce section 7.2 (flux current affected by gradient of other branching densities).

Note that the mechanism from section 7.2 shouldn't need the sigmoid, as a_i is already divisively normalized in this scheme. The computation of $\mathbf{J}_i$ is thus:

$$\mathbf{J}_i = D\nabla a_i + \frac{a_i F}{N-1}\nabla \hat{a}_i - a_i \mathbf{g} \tag{80}$$

The divergence of $\mathbf{J}_i$ can be computed in the same was as in section 7.2:

$$\nabla \cdot \mathbf{J}_i = D\nabla \cdot \nabla a_i + \frac{F}{N-1} a_i \nabla \cdot \nabla \hat{a}_i + \frac{F}{N-1}\nabla \hat{a}_i \cdot \nabla a_i - a_i \nabla \cdot \mathbf{g}_i - \mathbf{g}_i \cdot \nabla a_i \tag{81}$$

This is implemented in rd_james_comp13. It seems to be very unstable, as it was before.

7.14. Divisive normalization, additional competition mechanisms 2

I've added7.3 (diffusion is increased into regions with low n). This mechanism is successful in keeping branching outside regions where connections have been made.

I'm also going to try to add 7.7 (multiplicative reduction of diffusion of a_i), which I should be able to turn on and off with a parameter.

8. Unimplemented ideas

8.1. Upper bound on a_i (sigmoid) and $\hat{a}_i$ inhibits $\dot{a}_i$

The alternative is to let $n_c \equiv n$ and introduce a new n_a such that

$$n_c = 1 - \sum_i c_i \tag{82}$$

$$n_{a,i} = 1 - \sum_i a_i \tag{83}$$

and then introduce $n_{a,i}$ into the RD equations. The quantity $n_{a,i}$ is then the maximum density of branching permissible for TC type i, due to physical and/or chemical considerations.

9. Initial conditions

Initial reading of the literature suggests that axons grow towards cortex in a 'guideposted manner' and then start branching once within the cortex. This would suggest that initial conditions should be more targeted than in the Karbowski model, with axons of type i arriving in a region expected to be region i, but with the possibility of modifying the model to simulate an enucleated animal by reducing the density of incident visual TC axons, or reducing their branching rate.

I've implemented a scheme, using 2D Gaussians of varying (symmetrical) width, gain and location, to make masks for the initial branching densities for the different thalamocortical axon types.

10. Axon guidance reading

Erzurumlu & Gaspar [4] have a great table of chronological events in the development of the whisker barrels.

Gross topographic order is present in the projections from the trigeminal ganglion to the whisker pad and brainstem even from E12.

Thalamocortical axons are on their way to the neocortex by E13, passing a "corridor" by E13.5.

Layer IV neurons in cortex are born around E14-E15.

Thalamocortical axon pioneers arrive at prospective somatosensory cortex around E15.

At around birth (E19 to P0), barrelette patterns emerge in principle sensory nucleus of the trigeminal nerve.

At P1, axons of the principle sensory nucleus of the trigeminal nerve form bands in the ventro-postero-lateral nucleus the thalamic complex (VPM).

By P3, barreloids are visible in VPM.

Sequential ordering of pattern formation first in brainstem, then in thalamus and finally in cortex is reviewed in Sehara & Kawasaki, 2011.

11. Agent based model

It may be helpful to create a more realistic axon model, where axons grow in a tree manner, with guidance and interactions, from physically plausible locations in a virtual thalamus.

12. Applying the model to whisker barrels

Exploring how the model can speak to Whisker barrels.

Axons on their way from ventrobasal thalamus (VB) to somatosensory cortex arrive in Layer VIb and form a 'mass' of axons, with axons branching in every direction.

I'm going to first try to model the formation of rows of axons within this mass. It's plausible that there is an ordering within this region, because from this region can come both the wildtype barrel cortex *or* the reversed duplicate barrel field.

12.1. Barrel rows: N 1D Gaussian waves

This mechanism makes nice rows. See `configs/whiskers/5N5M_rows.json`. The problem is that it requires bands of expressed guidance molecules and banded expression isn't seen in barrel cortex by the experimentalists.

12.2. Barrel rows: 1 guidance molecule plus competition

With the competition described in section 7.4, it is difficult for separate, well delineated regions to form. I'm trying to think of an idea to make the competition between regions stronger.

One possibility is to have each TC type inhibit only selected other TC types. This might separate the centroids of effect, leading to better row definition.

Another idea is to introduce a chemical generated in proportion to axon branching, which then has some finite decay time.

Or, whichever a is the highest sees no attenuation. (Implemented as `rd_james_comp5.h`).

I think the issue is that whatever inter-TC-type attenuation there is is swamped by the diffusion term. That branching just keeps on spreading out...

I'll re-visit the concept of using the gradient of branching in types $j \neq i$ to influence the branching flux of type i (That's in section 7.2).

12.3. Barrel rows; 1 guidance molecule pushes right, N others push left, giving rows.

The mechanism works quite nicely, and has the advantage of using only graded guidance molecule expression. See configs/whiskers/5N_boundedorder.json

13. Post eLife Review work

One of the reviewers, Bard Ermentrout, suggested an alternative formulation of the branching competition, to avoid the undesirable need to divisively normalize the branching density at each simulation step.

The suggestion was to let the χ term from Eq. 3 of the preprint be given by:

$$\chi_i = \frac{a_i}{N-1}\nabla\hat{a}_i, \tag{84}$$

where $\hat{a}_i$, the sum of branching over all other TC types, is given by Eq. 35. Implementing Eq. 84 gives:

$$\frac{\partial a_i}{\partial t} = \nabla\cdot\left(D\nabla a_i - a_i\sum_{j=1}^{M}\gamma_{i,j}\nabla\rho_j(\mathbf{x})\right) + \frac{a_i}{N-1}\nabla\hat{a}_i - \frac{\partial c_i}{\partial t}. \tag{85}$$

There's a problem here. The problem is that the gradient of $\hat{a}_i$ is a *vector* quantity whereas $\dot{a}_i$ is scalar. So in fact, Bard's suggestion doesn't make good sense as written.

Compare this with Eq. 40 from Section 7.2, which was one of the competition types I tried fairly early on. In fact, it was my second idea for competition. It's similar, but not identical. Written out in the same style as Eq. 85 it is:

$$\frac{\partial a_i}{\partial t} = \nabla\cdot\left(D\nabla a_i - a_i\sum_{j=1}^{M}\gamma_{i,j}\nabla\rho_j(\mathbf{x}) + \frac{\bar{a}_i F}{N-1}\nabla\hat{a}_i\right) - \frac{\partial c_i}{\partial t}. \tag{86}$$

This modifies the axon branching with a similar term to Eq. 84, depending on the gradient of $\hat{a}_i = \sum_{j\neq i} a_j$. However, here, the term is included in the *flux of axonal branching* (i.e. *inside* the brackets) and so its divergence is taken and this contributes a scalar quantity to the change of a_i. Two other points to note about Eq. 86: I included a scalar parameter F to modify the relative contribution of this term; and $\bar{a}_i$ is the result of passing a_i through a sigmoidal transfer function (see Eq. 56) which I found necessary to prevent the term from becoming numerically unstable.

I passed Eq. 86 over for the preprint, because the competition it produced seemed weak, and I was relying on the development of contours and 'well localized regions' to indicate success.

After re-examining the effect of rd_james_comp2, which expresses Eq. 86, I found that it does provide competition enough to generate a barrel-field-like ordering, when the argmaxes of the N c_i values are plotted, as in the preprint.

References

1. George B. Arfken and H. J. Weber. *Mathematical Methods for Physicists*. Academic Press, fourth edition edition, 1995.
2. J. Karbowski and G. Ermentrout. *Journal of Computational Neuroscience*, 17(3):347–363, Nov. 2004.
3. D. Lee, H. Tien, C. Luo, et al. *International Journal of Computer Mathematics*, 91(9):1986–2009, Sept. 2014.
4. R. Erzurumlu and P. Gaspar. *European Journal of Neuroscience*, 35:1540–1553, Feb. 2012.
5. M. Carandini and D. J. Heeger *Nature Reviews Neuroscience*, 13:51–62, Nov. 2011.