Chapter 1

Eigenmorphs? Rational design of pattern and morphology in eigenspace

1.1 Background maths

1.2 Eigenfunctions of the laplacian

Eigenfunctions of the laplacian ∇^2 satisfy

$$\nabla^2 v(\mathbf{x}) + k^2 v(\mathbf{x}) = 0 \tag{1.1}$$

Given some boundary condition, such as $v = 0$ or $\nabla v = 0$ at the boundary of the domain there is a discrete (infinite) set of functions $v_i(\mathbf{x})$, with eigenvalue k_i. For a 1-dimentionsal domain with zero boundary conditions for example, the eignfunctions are sine waves $sin(k\pi/L)$ where L is the length of the domain and $k \in 0, 1, 2, 3....$

On other arbitrary domains we have to compute them numerically. You can see they are waves because the equation says when $v > 0$ the curvature $\nabla^2 v < 0$ and vice versa.

1.3 Eigenfunction expansion or projection

They also have the nice property that they form a complete basis for the space of differentiable functions that satisfy the boundary conditions. That means that any pattern (gene expression, signal concentration, etc.) on a given domain (colony, organism, etc.) can be represented as a series

$$f(\mathbf{x}) = \sum_i w_i v_i(\mathbf{x}) \tag{1.2}$$

Which is exactly the Fourier series for a rectangular or linear domain.

The equivalent of the Fourier transform for arbitrary shaped domains is the eigenspectrum, which gives the amount of each eigenfunction represented in a particular pattern

$$w_i = \langle f, v_i \rangle = \tag{1.3}$$

1.4 Eigenvectors and discrete space

For computation we represent patterns on a regular grid like an image. It turns out convenient to string this out as a 1-dimensional vector of length $w \times h$ for an image of width w and height h. The operator ∇^2 is then a matrix $\mathbf{M}$ which has eigenvectors $\mathbf{v}_i$ such that $\mathbf{A}\mathbf{v}_i = \lambda_i \mathbf{v}_i$. Note that now there are a finite number of eigenvectors equal to the number of grid points or pixels in the image.

The eigenspectrum is just a basis transformation from the grid basis, where each value in the grid (image) is an element of the vector, to the basis of eigenvectors. This is the natural frequency domain of the shape defined by the boundary conditions. Lets call this the eigenspace.

In discrete space we can define a matrix of eigenvectors $\mathbf{W}$ where each column is an eigenvector $\mathbf{v}_i$ such that

$$\mathbf{w} = \mathbf{W}\mathbf{g} \tag{1.4}$$

is the eigenspace transform of the image $\mathbf{g}$, and

$$\mathbf{g} = \mathbf{W}^T \mathbf{w} \tag{1.5}$$

is the inverse transform. This is because the eigenvectors form an orthonormal basis for the space of images (patterns).

1.5 Design of pattern in eigenspace

This means we can compute the eigenspectrum of any pattern, for example one that we might want to design a genetic system to produce. Alternatively we could design the spectrum of shape that we require from our genetic system. It would be easy (relatively) to make a tool that allows you to adjust a curve and see the resulting pattern (in Matlab for example). This curve could be a polynomial for example. It seems reasonable that a useful biological pattern should be localised in eigenspace (frequency space). If all wavelengths are represented equally we have white noise, which possibly is maximal entropy.

What we would like to do is choose a pattern and infer a genetic mechanism coupled to signalling that could generate it. These genetic components would come from a registry of characterised parts. For theoretical work we show the principle for example by randomly generating part parameters to make a library.

1.6 Reacion diffusion systems

In this sense any patterning system can be seen as selecting (weighting) some eigenvectors more than others. Reaction-diffusion systems are an obvious example (there are others, see Murray).

$$u_t = \nabla^2 u + \gamma f(u, v) \tag{1.6}$$
$$v_t = d\nabla^2 v + \gamma g(u, v) \tag{1.7}$$

In general we can't predict the final pattern without solving numerically, but close to an equilibrium (homogeneous or heterogeneous) we can approximate the reaction term as linear in the perturbation $\mathbf{w} = (u - u_0, v - v_0)^T$ where (u_0, v_0) is the equilibrium giving

$$\mathbf{w}_t = \nabla^2 \mathbf{w} + \mathbf{A}\mathbf{w} \tag{1.8}$$

where $\mathbf{A}$ is the Jacobian

$$\mathbf{A} = \begin{bmatrix} f_u & f_v \\ g_u & g_v \end{bmatrix} \tag{1.9}$$

We can use this to do a linear stability analysis and test for patterning. The definition of a Turing system is that it has an unstable homogeneous equilibrium, and a stable heterogeneous equilibrium. Around the homogeneous equilibrium the behaviour can be approximated by the equation above. This is a linear equation and so the solutions can be expanded in eigenvectors to get

$$\mathbf{w} = \sum_{k=0}^{N} c_k \mathbf{W}_k e^{\lambda(k^2)t} \tag{1.10}$$

where $\lambda(k^2)$ is called the *dispersion relation* and tells us how each eigenvector (wave) grows in time. If $\lambda(k^2) > 0$ the wave grows, and this part of the perturbation is propagated. This is how the system effectively selects eigenvectors.

For a two-component reaction-diffusion system as above it can be shown (Murray) that the dispersion relation $\lambda(k^2)$ is a quadratic in k^2. This means it has a single peak and a range $k^2_{min} < k^2 < k^2_{max}$ for which $\lambda > 0$. (Note the use of k^2 is because the sign of the spatial eigenvalue k is irrelevant, bit tedious to explain this but it comes out in the expansion above when you compute the weights c_k).

Three things to note:

1. For a *linear* reaction-diffusion system (not at all realistic but useful to study) we can solve *exactly* the evolution of the system (eigenspace expansion above).

2. In general (non-linear systems) the only way to get the final eigenspectrum of a patterning system is to compute the solution numerically.

3. BUT, if we start from this equilibrium we *can* compute the evolution of *small* perturbations from the eigenspace expansion above. The spectrum says which waves grow.

This last point will be very useful when we consider growing domains below.

1.7 Growing domains

1.8 Coupling morphology to pattern

1.9 Morphology in eigenspace?

1.10