The Simplification of Biological Models using Voronoi Tiling of the Phase Plane

Supervisors: James Hetherington (Dept. of Anatomy and Developmental Biology) Sachie Yamaji (Dept. of Anatomy and Developmental Biology)

Word count: 3,366

Abstract

A novel approach to simplifying biological models has been suggested by Hetherington and Saffrey, involving a Voronoi tiling of the phase plane, then following the flow vector (dx/dt dy/dt, where x and y are the variables undergoing phase plane analysis), to create a directed graph of nodes within the phase plane. Closed loops of the graph will correspond to stable limit cycles. In this way a complicated model may be more simply expressed as the traversal of a set of states in the phase plane of the model. This essay discusses the results of this process as applied to the Morris-Lecar model and a hepatocyte calcium oscillation model, and finds that the phase plane patterns generated by the algorithm closely resemble those found in the literature (obtained by numerical integration methods). The generation of bifurcation diagrams requires a little more tuning to increase reliability, however overall the algorithm seems to work well for the models described.

Introduction

The modelling of biological systems often involves approximating biological processes as systems of non-linear differential equations where the dynamical variables represent concentrations of various chemical or biological entities, proportions of signalling proteins activated, or cellular voltages and currents. Such models often quickly become very complicated with many different equations with large numbers of parameters required to accurately describe the biological system.

One approach in modelling is to incorporate every conceivable variable into the model. This will at least reduce the odds that any important variable has been left out. However even in the unlikely event that the huge all-encompassing model is computationally feasible, the number of parameters available for tweaking may reduce the credibility of the output – it is possible to fit almost any set of results to a very large model just by tweaking parameters. In this case, it is difficult to have confidence that the parameter values reflect relevant biological analogues, and no real insight into the biology of the problem has been gained.

Clearly it is essential to make simplifications and generalisations at every step of the modelling process: one must select the important variables to take into account while neglecting others – a process which requires a great deal of thought since which variables are 'important' depends on the purpose of the model.

Furthermore it is often necessary to use mathematical functions to approximate some biological response curve - for example the use of Hill functions to represent the level of activation of a population of receptor proteins in the presence of an agonist. Even

this common generalisation requires at least two parameters to be set: the Hill exponent n (effectively the sharpness of the response, where the Hill function tends toward the Heavyside step function as n tends to infinity), and the 50% threshold value for the agonist.

Between the initial assumptions of the model and the parameters introduced by the mathematical functions used, clearly the model is going to rapidly diverge from biological reality. A trade-off is required between the simplicity of the model and the complexity required to remain biologically relevant, and at the end of the process the model must actually be computationally feasible.

In [1] the authors argue that the purpose of modelling is to gain insight into the underlying biology, and that absolute simulation of the biological system is not in itself necessarily a useful endeavour. A good model should be 'just complicated enough' and in practical terms if it retains some information about the interaction between the various components of the model (for example stability information, magnitude and thresholds of oscillation of variables) then this can provide useful insight even if the time courses of variables predicted by the model are significantly erroneous.

This essay is concerned with a method for extracting stability information from two well-known biological models without carrying out full numerical integration of the variables. The phase planes of the models are effectively compressed and the stable/unstable features detected. The output of the compressed model is then qualitatively compared with the data available in the literature, in order to assess the impact of the compression.

Compressed phase plane analysis

The 2-D phase plane shows the interaction between two variables in the model while all other variables and parameters are kept constant¹. In a flow vector analysis of the phase plane, the flow vector is defined as the rate of change of the two variables with respect to time – at each point in the plane the vector can be calculated, and points to the next point in the phase plane to which the variables will evolve. It may be possible to easily identify regions of interest in the diagram – e.g. stable/unstable fixed points/limit cycles – assuming there is such an interaction between the two variables chosen for the analysis. By repeating the phase plane analysis for various values of the parameters, one can quickly identify which parameters of the equations have a large effect on the stability of the system, and where the threshold values for those parameters lie. A bifurcation diagram shows the stable (and unstable) points of a variable as a given parameter changes.

The computation and progression of the flow vectors can be simplified by using a method similar to vector quantization [2] of the phase plane. The first step is to carry out a Voronoi tiling of the plane, with each tile containing a node at which point the

¹ This essay only discusses 2D phase plane analysis since 2 dimensions are easy to visualise – correspondingly the models considered in this essay are 2 variable systems. The principles can extend to higher dimensions, but the visualisation and interpretation of results becomes much more difficult to understand.

flow vector is calculated. This vector is taken to be an approximation of the flow vector for the whole of that tile 2 .

A Voronoi tiling pattern is shown in Figure 1.



Figure 1: Voronoi tiling pattern for set of random points within a square (source: Wikipedia)

These nodes can be connected together in some order based upon the flow vectors at each node to give an indication of the flow pattern across the whole of the phase plane. The output is a directed graph where stable limit cycles are represented by strongly connected components of the graph. (Stable fixed points will also appear as small loops in the graph, which must be distinguished from the limit cycles.)

The key problem is to decide how to connect the nodes: the simplest solution is to follow the flow vector from each node until it enters impinges on another Voronoi tile, and to connect the nodes of these tiles together (*repeat for all nodes*). A more sophisticated approach, as implemented by Saffrey and Hetherington is outlined below:

For each Voronoi node X in the phase plane:

- 1. Calculate the vector <u>flow</u> at this node
- 2. Identify the adjacent nodes to X. For each of these adjacent nodes Y:
 - *i.* Calculate the vector <u>spacing</u> from X to Y
 - ii. Calculate the magnitude squared of <u>spacing</u> i.e. $|spacing|^2$
 - iii. Calculate the 'dot' product spacing.flow
 - *iv.* Divide $|spacing|^2$ by spacing.flow call this quantity t
- 3. To identify stable features: connect the current node to the adjacent node which had the smallest positive value of t
- 4. To identify unstable features: connect the current node to the adjacent node which had the smallest positive value of (-t) (i.e. the smallest in magnitude negative value of t).

² The properties of a Voronoi tiling ensure that the Euclidean distance of each point within the tile to the node for that tile is less than the distance to the Voronoi node of any other tile.

Rationale

The flow vector at node **X** probably doesn't point directly at any of the adjacent nodes **Y**, therefore it is inevitable that the decision to assign the flow to any of the adjacent nodes will introduce some error into the flow. Since we are approximating the direction of the flow vector by changing the direction towards a given node, it seems reasonable to adjust the magnitude of the flow vector accordingly: take the component of the true flow vector in the direction of the target node. Mathematically this can be achieved by taking the dot product of the spacing and flow vectors, and dividing by the magnitude of the spacing vector.

|modified flow vector| = spacing.flow / |spacing|

Remembering that the flow vector represents a change of phase variables per unit time, we can estimate a time, t, corresponding to the change from state X to state Y by dividing the distance between X and Y by the magnitude of the modified flow vector:

t = |spacing| / |modified flow vector|

substituting from above:

$t = |spacing|^2 / spacing.flow$

Thus the quantity t being minimised in the algorithm is the transit time of the component of the flow vector travelling between the two nodes.³ The algorithm selects the node to which the flow would travel fastest, taking into account the fact that we are approximating the direction of the flow.

Note that since the algorithm also calculates the transit times for nodes in the opposite direction to the flow vector (i.e. t can be negative), it also works out the closest node from the point of view of the *reverse* flow vector. By building a separate directed graph of the reverse flow, we can detect stable features in the reverse flow, which are in fact the unstable features in the 'actual' forward flow.

How many Voronoi tiles?

Clearly the more tiles that are used to represent the flow across the phase plane, the more accurate will be the directed graph. The minimum number of Voronoi tiles required to cover the phase plane without losing important variation in the flow vectors depends upon the density of important features in the phase plot.

To a certain extent it is necessary to know beforehand where in phase space the interesting regions are. Practically speaking the user must define the extent of the axes of the phase plane, which will then be Voronoi tiled with some number of tiles.

³ Of course there are other geometrically equivalent interpretations of t – one could say it is the time taken for the flow vector to reach the point where a vector from Y drawn perpendicular to <u>spacing</u> crosses the direction of the flow vector (making a right-angled triangle from X to Y to the intersection). Although this is mathematically the same thing, the explanation in the text seems more intuitive to me.

In this essay the number of tiles ranged from 1,000 to 10,000. Bifurcation diagrams were constructed by running the algorithm over the phase plane repeatedly as some parameter or external variable in the model was altered. The degree of fidelity between the bifurcation diagrams produced and the 'textbook' versions was qualitatively measured as the tiling resolution degraded (corresponding to higher compression of the biological model).

Implementation

This scheme of 'simplification by compression' was implemented by Saffrey and Hetherington in a python application which allows the user to define a vector flow field (see Appendices A and B) which will then be 'compressed' using the method described above, giving as an output a directed graph indicating the flow between Voronoi nodes. The program identifies strongly connected components in both the forward and reverse flows, and outputs information about them. The results can be visualised using graphviz and a plotting utility.

Biological Models

At this point it is appropriate to describe the biological models which were used to evaluate the program.

Morris-Lecar

The Morris-Lecar model [3][4] is a well-known simple model to explain electrical behaviour of barnacle muscle fibre. Experiments where a depolarisation current was applied to the muscle fibre resulted in electrical activity in the fibre which were found to arise from voltage gated K+ and Ca2+ channels, as well as a further K+ current activated by intracellular Ca2+. Voltage clamp experiments showed that the action of these channels was not affected in the way predicted by the Hodgkin and Huxley model for the squid giant axon [5], therefore some other model was required.

Morris and Lecar proposed the following model:

$$C\frac{dV}{dt} = -g_{Ca}m_{\infty}(V - V_{CA}) - g_{K}w(V - V_{K}) - g_{L}(V - V_{L}) + I_{app}$$
$$\frac{dw}{dt} = \frac{\Phi(w_{\infty} - w)}{\tau}$$

Electrically speaking, the model assumes the cell to behave like a capacitor which is leaking charge through a variety of conductances which depend upon the capacitor voltage. The biological origins of these charge leakages are both the applied depolarising current I_{app} , as well as a general leakage conductance g_L (with a corresponding voltage-offset of VL), and leakages through the Ca²⁺ and K⁺ channels (with peak conductances g_{Ca} , g_K). V_{CA} , V_K are parameters controlling the voltage characteristics of those channels, while m and w are functions which describe the proportion of open voltage gated Ca²⁺ and K⁺ channels respectively at any given time.

 m_{∞} , w_{∞} and w are described by the following equations:

$$m_{\infty} = 0.5 \left[1 + \tanh\left(\frac{V - v_1}{v_2}\right) \right]$$
$$w_{\infty} = 0.5 \left[1 + \tanh\left(\frac{V - v_3}{v_4}\right) \right]$$
$$\tau = \frac{1}{\cosh\left(\frac{V - v_3}{2v_4}\right)}$$

 $(v_1, v_2, v_3, v_4 \text{ are threshold parameters for the voltage gated ion channels.})$

It is worth noting that in this model w (the proportion of K+ channels open) is a timedependent variable – changes in V alter the value of w with a time lag controlled by τ . τ is itself dependent on V. On the other hand, the fraction of open Ca²⁺ channels, m, has no such complicated time dependency – the value of m depends on V but is independent of time. The assumption made here is that any time lag in m is short enough that it may be neglected and m is assumed to be in steady state. This is useful from the point of view of our phase plane analysis since it means we have only a 2-D system of equations!

This model was implemented by Saffrey and Hetherington and modified slightly for this essay (see Appendix A). Parameter values which produce an interesting phase plane analysis were taken from [4] in order to test the Voronoi compression scheme against a well known system of equations and determine whether the compressed phase plane analysis is capable of retaining useful stability information.

Initially the system was set to oscillate with $I_{app}=150\mu A/cm^2$ – the phase plane analysis of V vs. w should in this situation produce a stable limit cycle [4]. The directed graph of Voronoi nodes produced by the program is shown in Figure 2. The axes are scaled to +/- 75mV on the x-axis and 0 to 1 on the y-axis. The limit cycle is outlined in blue, and the flow pattern seems to generally resemble those in [4].



Figure 2: Phase portrait of Voltage (mv) vs. w (proportion of K⁺ channels open) – 10,000 Voronoi tiles. Limit cycle is the faint red loop (x-axis is scaled –75mV to 75mV, y-axis scaled 0 to 1)

Figure 3 shows a zoomed in region of the phase portrait, showing that the program has also identified some unstable fixed points in the phase plane (the small closed coloured loops near the bottom of the figure).



Figure 3: Zoomed in part of phase plane showing part of a limit cycle (red) and some unstable fixed points (blue, purple, green). Note that the structure is that of a structured graph.

As the number of Voronoi tiles is decreased the resolution of the phase portrait diminishes as expected (Figure 4). However there appears to be little loss of useful information – the limit cycle is still clearly identified and the shape is not very different from the 'high resolution' version, suggesting that the time period and shape of the time course will not be very different between the two levels of compression. Figure 5 shows that this is indeed the case.



Figure 4: 1000 Voronoi tiles - lower resolution, but little loss of information?



Figure 5: Time traces for 'high' (pink) and 'low' (blue) resolution Morris-Lecar model

From the results it is clear that the compressed version of the phase portrait bears remarkable similarity to that derived from full numerical integration. Indeed, the detection of stable limit cycles is quite robust even at high levels of compression (i.e. fewer Voronoi tiles).

However the bifurcation diagrams (Figure 6 and Figure 7) show that there are limitations to the technique. While the detection of stable limit cycles is quite good, the detection and classification of unstable features is a little less reliable.⁴ The quality of the bifurcation diagram is improved for higher values of N, but at the cost of increased 'noise' of spurious unstable features. At low N, the bifurcation diagram shows that the lower resolution version wrongly detects the onset of oscillations. Some further work in this area seems to be required, however the general principle seems sound for this example at least.

⁴ The performance of the fixed point detection was improved by the author for the purpose of this essay, by enforcing the condition that a strongly connected component of the graph corresponds to a fixed point if there are no Voronoi nodes contained within the region bounded by the loop (that are not part of the loop). This is an improvement on the original size-based threshold method where the area of the loop was checked against a fairly arbitrary cutoff value to determine whether it was 'probably' a fixed point rather than a limit cycle.



Bifurcation diagram for Morris-Lecar: lapp vs. Voltage - High Resolution N=10k

Figure 6: Bifurcation diagram with high N - many spurious unstable points, but good overall agreement with literature



Bifurcation diagram for Morris-Lecar: lapp vs. Voltage - Low Resolution N=1000

lapp (uA/cm2)

Figure 7: Bifurcation diagram for lower N

Calcium oscillations in Hepatocytes

The second model explored for this essay was that of calcium oscillations in hepatocytes.



Figure 8: Cultured hepatocytes (rat)⁵

It has been observed experimentally that for cells in general, a release of calcium ions from intracellular compartments into the cytosol can be induced by externally applied stimuli. It is possible to induce oscillatory behaviour of calcium ion concentration in cells by applying suitable stimuli.

In the case of hepatocytes the hormones vasopressin and noradrenalin can activate phopholipase C (PLC) which in turn activate inositol trisphosphate ($InsP_3$) receptors, which give rise to a release of calcium ions from the endoplasmic reticulum (ER) into the cytosol. In addition there is calcium induced calcium release (CICR) of ions from the ER into the cytosol which occurs at moderate cytosolic concentrations of calcium, but which is inhibited by high concentrations of cytosolic calcium.

Meanwhile calcium pumps on the ER membrane known as Sarco/endplasmic reticulum calcium atp-ase (SERCA) transport cytosolic calcium ions back into the ER. At the same time, there is calcium ion exchange to/from the cytosol via the plasma membrane, through plasma membrane calcium atp-ase (PMCA) pumps, and by some level of leakage.

Höfer [6] describes a model to account for these modes of calcium transport within hepatocytes. In addition [6] includes the possibility of inter-cellular signalling and

⁵ Photo taken through a confocal microscope during attempt to observe existence of gap junctions by inter-cellular transport of an injected fluorescent dye. The shadow to the right is a (broken!) glass tip used for the dye injection. Experiments such as these underpin the mathematical models under discussion and must be carried out to both supply parameters to (and validate results from) computational models.

calcium flow through gap junctions between hepatocytes, however for the purpose of this essay gap junction connections are neglected and only single isolated cells are considered.

The model may be expressed as follows (mathematically equivalent to [6] without gap junction contribution, but written in the notation consistent with [1]):

$$\begin{aligned} \frac{dC}{dt} &= J_{ER,in} - J_{ER,out} + J_{PM,in} - J_{PM,out} \\ J_{ER,in} &= k_{EC} (E - C) (l_{EC} + U(P, C)) \\ J_{ER,out} &= k_{EP} \theta_2 (C, c_{EP}) \\ J_{PM,in} &= k_{MC} (l_{MC} + \theta_1 (P, p_{MC})) \\ J_{PM,out} &= k_{MP} \theta_2 (C, c_{MP}) \\ U(P, C) &= [\theta_1 (P, p_{EC}) \theta_1 (C, c_{EC,+}) (1 - \theta_1 (C, \gamma_{EC} (P)))]^3 \\ \gamma_{EC} (P) &= c_{EC,-} \frac{P + d_1}{P + d_3} \end{aligned}$$

 $\theta n(E,e)$ denotes the Hill Function with exponent *n*: $\theta n(E,e) \equiv \frac{1}{1 + (e/E)^n}$

J is the flux of calcium ions for each mode of transport. ER denotes transport in/out of the endoplasmic reticulum. PM denotes transport in/out of the cell via the plasma membrane channels. C is the cytosolic calcium ion concentration. E is the calcium ion concentration in the ER. In fact for the purposes of this essay the phase plane variables were defined as C (cytosolic calcium concentration) vs. *total* calcium concentration Z (Z = C + E/v) to allow comparison with the phase plane analyses in [6]. It follows that:

$$\frac{dZ}{dt} = J_{PM,in} - J_{PM,out}$$

P is the concentration of IP_3 which controls the onset of oscillations. All of the other quantities in the equations are parameters.⁶

[1] discusses the simplification of the model in [6] and carries out an analysis to quantify the impact of the simplification on the output of the model. A similar analysis to that in [1] could be carried out in this case to evaluate the performance and degradation of the full-scale model when analysed using the Voronoi compression scheme. Such a full comparison as described in [1] is beyond the scope of this essay, however in an attempt to make some progress in this direction the above model was implemented (Appendix B) and run through the Voronoi compression program.

⁶ The values of these parameters may be found from the implementation details in Appendix B, and are taken from [5] and [1] which are in turn based on experimental data. The actual values are not included in the body of this text since this essay is not concerned with the details of these models, but rather with the performance of the Voronoi compression scheme in evaluating these models.

Figure 9 is taken from [6] and summarises the output of Höfer's model in [6]. The aim is to recreate these results using the Voronoi compression method. Figures 10-12 give the results for a Voronoi tiling of the phase plane with N=1,000. Superficially the results are quite compelling evidence that the method is not losing too much important information.



Figure 9: Results obtained by Höfer (source: Figure 1 of [6]). This essay aims to recreate figures (a)-(d) of these results, using the Voronoi compression method.



Figure 10: Calcium oscillations for P=2 μ M (1,000 Voronoi nodes) – x axis corresponds to cytosolic concentraion C, y axis is total concentration Z. x-axis scale is from 0 to 0.6 μ M, y-axis scale is from 1.6 to 2.8 μ M. This should resemble Figure 9(c).



Figure 11: time courses for calcium concentration oscillations when $P=2\mu M$ - cytosolic (left) and total (right). These should resemble Figure 9(a) and 9(b).



Bifurcation diagram for Calcium oscillations in hepatocytes (following Hofer): P vs. Cytosolic calcium concentration (N=1000 tiles)

Figure 12:

Bifurcation diagram for calcium oscillations in hepatocytes as constructed using Voronoi compression of phase plane technique. This should resemble 9(d). Clearly some detail and accuracy has been lost in the compression process, and many spurious results appear at high concentrations of P.

Conclusions

This essay set out to describe a novel method of simplification of biological models by a 'vector-quantization like' compression scheme using a Voronoi tiling of the phase plane, implemented by Hetherington and Saffrey. Their code was slightly modified to generate some results of the compression scheme as applied to the wellknown Morris-Lecar model.

Further, a hepatocyte calcium oscillation model suggested by Höfer in [6] (and discussed in the context of 'simplification' by Hetherington et. al. in [1]) was implemented under the framework in order to further test the compression technique against some 'real' biological models.

The technique was found to be very effective at finding stable limit cycles in the phase plane of these models, even at 'high compression' (low number of Voronoi tiles), although there was a tendency to rather a large number of false positives – it appears the algorithm as presented has a little difficulty differentiating fixed points from limit cycles. In addition, the unstable features of the phase plane as identified by this algorithm are currently a little unreliable.

The time course data was essentially accurate in the shapes of waveforms produced, however the time period was elongated slightly, indicating that the limit cycles tend to be over-estimated by this algorithm. This is probably due to the algorithm giving a 'worst-case' flow rate to consecutive nodes in the phase plane.

As a final note on the practical implementation of models under this framework, it seems necessary to know a little beforehand what the interesting regions of the phase plane are to be – the maxima and minima of the axes of the phase plane must be defined in the model.

Appendix A – Morris-Lecar model implementation

This python class implements Morris-Lecar model by returning the flow vector at any given point, scaled for a set of axes spanning -2 to +2 in the x and y directions. (Slightly modified by Fallaize from code supplied by Saffrey and Hetherington).

```
import Numeric as NumPy
# morris-lecar model
class flowcl:
       def __init__(self, axes):
              # parameters
              self.CC = 20e-6
              self.VK = -84e-3
              self.qK = 8e-3
              self.Vca = 120e-3
              self.qCa = 4.4e-3
              self.VL = -60e-3
              self.qL = 2e-3
              self.v1 = -1.2e-3
              self.v2 = 18e-3
              self.v3 = 2e-3
              self.v4 = 30e-3
              self.Phi = 0.04/1e-3
              self.Iapp = 150e-6
               self.defineScaleMatrix(axes)
       def defineScaleMatrix(self, axes):
               [[xmin, ymin], [xmax, ymax]] = axes
               print xmin, ymin, xmax, ymax
              axismaxima = NumPy.array([ 75.0e-3, 1.0 ])
              axisminima = NumPy.array([ -75.0e-3, 0.0 ])
              self.m = (axismaxima - axisminima) / 4.0
              self.c = axisminima
       # flowfn
       def flowfn(self, location):
              def mi(V):
                      return (1 + NumPy.tanh((V-self.v1)/self.v2))/2.0
              def wi(V):
                      return (1 + NumPy.tanh((V-self.v3)/self.v4))/2.0
              def tau(V):
```

```
return 1/(NumPy.cosh((V-self.v3)/(2.0*self.v4)))
```

```
V = location[0]
w = location[1]
Vprime = (-self.gCa*mi(V)*(V-self.Vca) - self.gK*w*(V-self.VK) - self.gL*(V-self.VL) + self.Iapp) / self.CC
wprime = (self.Phi*(wi(V)-w))/tau(V)
return NumPy.array([Vprime, wprime])
def translate(self, location):
    return (location + NumPy.array([2.0,2.0])) * self.m + self.c
def __call__(self, location):
    l = self.translate(location)
    o = self.flowfn(l)
    return o / self.m
```

Appendix B – Calcium oscillations model implementation

This python class implements the hepatocytes calcium flux model by returning the flow vector at any given point, scaled for a set of axes spanning -2 to +2 in the x and y directions.

```
import Numeric as NumPy
# hepatocyte calcium oscillations (Hofer 1999)
class flowcl:
       def __init__(self, axes):
               # parameters
               self.kMC = 0.08e-6
               self.kMP = 0.072e-6
               self.kEC = 1.6
               self.kEP = 0.36e-6
               self.CecOpen = 0.4e-6
               self.CecClose = 0.4e-6
               self.pEC = 0.2e-6
               self.cEP = 0.12e-6
               self.pMC = 4.0e-6
               self.cMP = 0.12e-6
               self.lMC = 0.05
               self.lEC = 0.0005
               self.v = 10.0
               self.d1 = 0.3e-6
               self.d3 = 0.2e-6
               self.P = 2e-6 # driving function IP3 hormone conc
               self.defineScaleMatrix(axes)
       def defineScaleMatrix(self, axes):
               [[xmin, ymin], [xmax, ymax]] = axes
               print xmin, ymin, xmax, ymax
               axismaxima = NumPy.array([ 0.6e-6, 2.8e-6 ])
               axisminima = NumPy.array([ 0.0, 1.6e-6 ])
               self.m = (axismaxima - axisminima) / 4.0
               self.c = axisminima
```

```
# flowfn
def flowfn(self, location):
       def Hill(n, E, e):
              return 1.0 / (1.0 + (e/E)**n )
       def Gamma(P):
              return self.CecClose * ((P + self.d1) / (P + self.d3))
       def U(P, C):
              return (Hill(1,P,self.pEC) * Hill(1,C,self.CecOpen) * (1.0 - Hill(1,C, Gamma(P)) ) )**3
       def JerIN(E, C):
              j = self.kEC * (E - C) * (self.lEC + U(self.P,C))
              return j
       def JerOUT(C):
              j = self.kEP * Hill(2, C, self.cEP)
              return j
       def Jer(E, C):
              j = JerIN(E, C) - JerOUT(C)
              return j
       def JpmIN(E, C):
              j = self.kMC * (self.lMC + Hill(1, self.P, self.pMC))
              return j
       def JpmOUT(C):
              j = self.kMP * Hill(2, C, self.cMP)
              return j
       def Jpm(E, C):
              j = JpmIN(E, C) - JpmOUT(C)
              return j
       C = location[0]
       Z = location[1]
       E = (Z - C) * self.v
       Zprime = Jpm(E, C)
       Cprime = Jer(E, C) + Zprime
       return NumPy.array([Cprime, Zprime])
def translate(self, location):
        return (location + NumPy.array([2.0,2.0])) * self.m + self.c
def __call__(self, location):
       modelflow = self.flowfn(self.translate(location))
       ode2fsflow = modelflow / self.m
       return ode2fsflow
```

References

- [1] Hetherington, J.P.J., Warner, A., & Seymour, R.M. 2005, "Simplification and its consequences in biological modelling: conclusions from a study of calcium oscillations in hepatocytes." *Journal of The Royal Society Interface* 3 (7) 319 331 doi:10.1098/rsif.2005.0101
- [2] <u>http://www.geocities.com/mohamedqasem/vectorquantization/vq.html</u>
- [3] Morris, C., Lecar, H. 1981, "Voltage oscillations in the barnacle giant muscle fiber." *Biophys J* **35** (1) 193-213
- [4] Fall, C.P., Marland, E.S., Wagner, J.M. & Tyson, J.J, 2000, "Interdisciplinary Applied Mathematics Volume 20: Computational Cell Biology" *Springer* pp 34-45
- [5] Hodgkin, A.L., & Huxley, A.F 1952, "A quantitative description of membrane current and its application to conduction and excitation in the nerve" *J. Physiol.* **117**, 500-544
- [6] Höfer, T 1999, "Model of intercellular calcium oscillations in hepatocytes: synchronization of heterogeneous cells." *Biophys. J.* **77**, 1244-1256