The Editor

Journal of the Royal Society Interface

20 April 2011

Dear Tim

Thank you for your strong encouragement to revise and resubmit our manuscript rsif-2011-0068 entitled "*A formal mathematical framework for physiological observations, experiments and analyses*".

We are very pleased that all three referees think that our approach is important and worth publication. We have addressed all of their comments with changes to the manuscript as outlined below.

**Referee 1** is very positive about our approach, but comments on its possible impact. We have provided a theoretical framework and demonstrated that it can be implemented in real-world applications. This is a key first step, as recognised by all three referees, but of course is only a start. As Referee 1 notes, our framework is “easily extensible”, and we fully anticipate that the impact of our work will derive not only from the start we have made here, but also from many extensions to our ideas by us and others across a wide range of disciplines. In the discussion, we now raise the point that programming language features needed for CoPE are currently being implemented in mainstream programming languages.

The referee’s main concern (**comments 1,2 & 4**) relates to our claims about verifiability – particularly as some aspects of an experiment would not usually come under the control of CoPE. We have removed the paragraph dealing with this (last paragraph of the Discussion), and instead suggest possible ways in which our framework can be extended to (a) take advantage of some aspects of mathematical verification such as dimensional unit checking, and (b) incorporate powerful techniques for statistical inference.

The referee’s **third comment** asks whether our framework can deal with stimuli more complex than a cube. It can indeed, and we have now added text to page 9 indicating that natural scenes, for example, can easily be manipulated in CoPE. Given that CoPE is structured around time-varying signals, we anticipate that movies could equally well be implemented with appropriate extensions to our framework.

The referee's **fourth comment** asks what kinds of meta-data can be described with CoPE. We have added text to page 14 to clarify that CoPE is designed to describe all machine-executable meta-data, not everything that could possibly influence the outcome of the experiment.

The referee’s **final comment** asks whether we could implement template matching with dynamically generated templates. We have now added a section to the supplementary information demonstrating simple template matching defined as a single function that transforms each putative spike into a spike with an associated measure of goodness-of-fit to a specified template. Spike identification with dynamically generated templates forms an entire area of research in its own right, and commercial software companies devote huge effort to refining their algorithms for it. We have implemented the skeleton of a working spike detection algorithm that uses dynamically generated templates in CoPE (see appendix to this cover letter) and used it to identify locust DCMD spikes. However, we are reluctant to include this partially developed code in the supplementary material. For example, our implementation relies on a parameter (the threshold goodness-of-fit at which new thresholds are instantiated) to be hand-tuned for successful spike detection. Generalising the algorithm to deal with a wide range of spike shapes from different preparations is really beyond the scope of this theoretical paper, and indeed, would detract from the main message we wish to convey.

**Referee 2** comments that our approach is “original” and “very clear, even to non-experts”, but thinks that our explanation would benefit from an extended description of the relationship between functional programming and C++ programming. We have added text on page 4 and further discussion on page 14 to address this comment.

The referee’s **second comment** is that unfamiliarity with functional programming may be an obstacle to acceptance of CoPE. This criticism could be levelled at any proposed new technique, and of course, initially, there would be a learning curve for users. We anticipate that the theoretical framework we have developed could be implemented in many ways so that users interact with it through much friendlier user interfaces. Developing such an interface was not a goal of this paper, and indeed, would significantly dilute the message we wish to disseminate.

The results section now explicitly states that we have implemented a compiler for CoPE and used this for running the experiments in Examples 1 and 2, including visual stimuli (P 8). As for more complex visual stimuli, we have implemented texture loading in response to the reviewers' comments and describe its use on P 9. Complex shapes can be loaded in a similar manner. We have added text on P 15 of the manuscript to briefly discuss the applicability of our framework in a context that is wider than neurophysiology.

The referee’s **final comment** queries the idea of data provenance. We now identify this as the same issue as experimental description rather than an separate issue and have removed the term “equational formulation”

We have corrected the typo and references.

**Referee 3** comments that our approach is “a worthwhile endeavor that has the potential to guide the community...”, but finds some of our notation difficult. We have now simplified our notation by putting lambdas on the left hand sides of equations, so we go from (for example):

smap = \f -> \s -> {: f <: s :> :}

to

smap f s = {: f <: s :> :}

**Comment 2.** F

We now explicitly state the sampling rate of analog-to-digital and digital-to-analog converter sources. We have altered the text accordingly on pages 8, 10, 12 and 13 and in the supplementary information.

**Comment 3.** We have altered our use of Real numbers to Float, as suggested.

**Comment 4.** We have replaced numeric constants with variables, as suggested by the referee.

**Comment 5.** We now discuss AutoBayes and related statistical programming languages in the revised last paragraph of the Discussion.

**Comment 6.** The referee suggests that we should provide a CoPE description of the additional Methods described in the ‘Materials and Methods’. We now refer to the full CoPE listings for the experiments which are given in the supplementary information. In addition, we have extended our description of the scope of the meta-data that can be described with CoPE. We make it clear that with CoPE our goal is to describe machine-executable meta-data and not all possible information that could possibly influence an experimental outcome – which we think is not feasible.

**Comment 7.** We have corrected the typos.

**Comment 8.** We think that colour is also essential for Figure 1.

**Comment 9.** We have added text on P 16 and 17 to describe point about the animal ethics.

Yours sincerely,

Tom Matheson

Appendix: Dynamic Template Matching

This implementation of spike identification with dynamic template generation iterates through the putative spikes (with a "fold"; Hutton 1999 J Funct Prog 9(4):355-372), maintaining a list of templates by "averaging in" each spike to the template with the lowest root mean square difference. However, if the best template match is greater than the splitThresh parameter, a new template will be created instead. Here is an annotated definition in CoPE:

The threshold parameter is defined as:

splitThresh = 0.2

To access waveforms around a specific timepoint:

aroundTime t = limitSigs (-0.001) 0.001 $ around (ev t) ecVolts

The function rms calculates the r.m.s. difference from a template for a spike occurring at time t.

rms t (templ, \_) = sqrt $ sumSig $ smap (^2) $ templ - aroundTime t

The function minIdx finds the index of the item in a list with the smallest value. But if this is greater than splitThresh, or the list is empty, it returns -1; minimumIx finds the index of the item in a list with the minimum value.

minIdx [] = -1

minIdx xs | (xs!!minimumIx xs)> splitThresh = -1

| otherwise = minimumIx xs

The fold updates a list of templates paired with the spike count for each template (other information could be added as necessary, for instance a list of occurrence times of each template instance). The first line creates a new template and adds it to the list of existing templates, in case no acceptable template has been found. The second line averages the putative spike waveform with the best template match found. The function onIx applies a function to a single item in a list with a specified index.

update t templatesAndCounts -1 = (aroundTime t, 1):templatesAndCounts

update t templatesAndCounts ix =

onIx ix templatesAndCounts $ \(templ, count) ->

(smap (/(count+1)) $ smap (\*count) templ + aroundTime t, count+1)

The fold accumulator is the composition of the functions rms, minIdx and update.

acc templatesAndCounts t =

update t templatesAndCounts $ minIdx $ map (rms t) templatesAndCounts

The templates are calculated with the left-fold, i.e. starting with the earliest spike. The initial list of templates is empty ([]).

templates = foldl acc [] $ map fst putatives

Thus, the variable “templates” denotes the list of dynamically generated spike templates, each paired with the corresponding number of putative matching spikes.