Neuroinform (2012) 10:199–218
DOI 10.1007/s12021-012-9140-7

ORIGINAL ARTICLE

Using Evolutionary Algorithms for Fitting High-Dimensional Models to Neuronal Data

Carl-Magnus Svensson · Stephen Coombes · Jonathan Westley Peirce

Published online: 20 January 2012

Abstract In the study of neurosciences, and of complex biological systems in general, there is frequently a need to fit mathematical models with large numbers of parameters to highly complex datasets. Here we consider algorithms of two different classes, gradient following (GF) methods and evolutionary algorithms (EA) and examine their performance in fitting a 9-parameter model of a filter-based visual neuron to real data recorded from a sample of 107 neurons in macaque primary visual cortex (V1). Although the GF method converged very rapidly on a solution, it was highly susceptible to the effects of local minima in the error surface and produced relatively poor fits unless the initial estimates of the parameters were already very good. Conversely, although the EA required many more iterations of evaluating the model neuron's response to a series of stimuli, it ultimately found better solutions in nearly all cases and its performance was independent of the starting parameters of the model. Thus, although the fitting process was lengthy in terms of processing time, the relative lack of human intervention in the evolutionary algorithm, and its ability ultimately to generate model fits that could be trusted as being close to optimal, made it far superior in this particular application than the gradient following methods. This is likely to be the case in many further complex systems, as are often found in neuroscience.

C.-M. Svensson (✉) · J. W. Peirce
School of Psychology, University Park,
University of Nottingham, NG7 2RD, Nottingham, UK
e-mail: pmxcms1@gmail.com

S. Coombes
School of Mathematical Sciences, University Park,
University of Nottingham, NG7 2RD, Nottingham, UK

Keywords Computational modelling · V1 · Striate cortex · Filter-based · Model fitting · Optimisation methods · Evolutionary algorithms · Pareto optimality

Introduction

Automated Procedures

The study of *complex systems*, which comprise multiple interacting components with an overall behaviour that becomes hard to predict from any of the individual components, presents something of a challenge for many modern scientific disciplines. For example, having a good understanding of the response characteristics of individual neurons and their synapses, does not allow us to predict the behaviour of a network of such neurons. By the fact that these systems' behaviour is hard to predict, our understanding of them typically requires the development of quantitative computational models. Systematic failures of the models to reproduce the behaviours of interest are useful in highlighting shortcomings in our understanding. To find such failures the typical approach is to try to fit a model to data by manipulating its parameters until its outputs are similar to those of the target system.

There are two technical challenges to this endeavour. The first is how to quantify the goodness of a fit, a problem for which there are several possible measures. The second challenge is to find the best fit that the model is able to achieve, by maximising the goodness of fit measure(s). For simple problems, normally defined by convex optimisation in low number of dimensions, the choice of fitting algorithm may not be critical (Box

1966; Nash 1984). For complex systems, however, it is increasingly difficult to know whether the solution arrived at is actually the closest fit to the data that the model can achieve. That is, in cases where a model has failed to provide a good fit to the data it may not be that the model is *unable* to provide a good fit, merely that the fitting procedure has failed to find the optimal parameters. There is, therefore, an increasing need for powerful methods that consistently find the optimal fit to the data. Furthermore, there is also a great need for methods that are able to run with minimal need for manual human input.

Here we consider the advantages and disadvantages of some of these methods as applied to a real-world problem; the fitting of a nonlinear, multi-stage, high-parameter model to somewhat noisy response data. In our case these data came from extracellular in-vivo electrophysiological recordings from primary visual cortex (V1) of the macaque, but the problem is one common to many other areas of neuroscience.

Quantifying Fit Quality

All fitting algorithms essentially come down to an optimisation problem in which the aim is to maximise a goodness-of-fit measure or, equivalently, minimise an error term, E, that quantifies the discrepancy between the model outputs and the data. There are several options to quantify the goodness-of-fit, including;

(i) distance-based measures, e.g. sum of squared residuals (SSQ)
(ii) quantifying the number of model outputs that fulfil a certain criteria, e.g. ratio that fall within the confidence interval (RCI), of the respective data point
(iii) correlation-based measures, e.g. the percentage of the variance in the data that is explained by the model (R^2)

The most commonly used error terms in these optimisation problems are the distance-based measures. Usually these quantify the residual, the difference between the model and the data at each point for which data were collected. One could simply take the sum of the residual, but this would be a poor measure because positive and negative deviations between data and model outputs will cancel. The SSQ gets around the problem of cancellation between positive and negative deviations by squaring all elements of the residual and then summing them. The SSQ can be normalised by the number of data points used to give the mean squared residual (MSQ). This has the advantage of being independent of the number of data points used in fitting, but the fit is functionally equivalent. Similarly the root-mean-squared residual (RMS) can be used which has the advantage that it is then expressed in the same units as the original data.

Although commonly used during optimisation routines, the distance-based measures have disadvantages. One of these is that the final value of the measure does not signal the fit quality in a general case; it is dependent on both the units and the amplitude of the data. By describing the fit quality in terms of the ratio or percentage of points falling within the confidence intervals of the data (RCI), less prior understanding of the data is required to judge the fit quality. For example, finding that the fitting of model neuronal outputs to data resulted in a RMS error of 40 impulses per second (*ips*), is hard to interpret without prior knowledge of typical responses, whereas the fact that 85% of the model responses were within the confidence intervals is more naturally informative. For this reason the RCI may be useful in reporting fit quality. For the purpose of the optimisation procedure the related error term is quantified as $1 - RCI$.

Another natural choice to quantify similarity between data and model outputs is to use the correlation between them. The correlation coefficient squared directly expresses the percentage of the variance in the data that is explained by the model (R^2) (Howell 2004; Kent 1983). The error that needs to be minimised is $1 - R^2$.

In the limit, where the distance-based error (e.g. SSQ) actually approaches zero, the correlation- and criteria-based measures are also naturally optimised, whereas the converse is not necessarily true. However, for noisy systems, the SSQ is not likely to approach zero, and minimising the SSQ might not optimise the other measures. An example of this can be seen in Fig. 1. In this example synthetic data has been created and a pair of candidate model fits are presented. The curve with the lower SSQ (solid line) actually captures the data less well according to the ratio in the confidence interval (RCI) and the variance explained (R^2). An advantage of using correlation-based measures is their strong affinity to the characteristic shape of the data. The disadvantage is that they take no account of the overall amplitude of the data. The RCI suffers from the problem that once a model output falls outside the confidence interval, it no longer matters how far it strays. As a result, some points may end up with extreme errors for single points in order to maximise the number of other points falling within the criterion region.

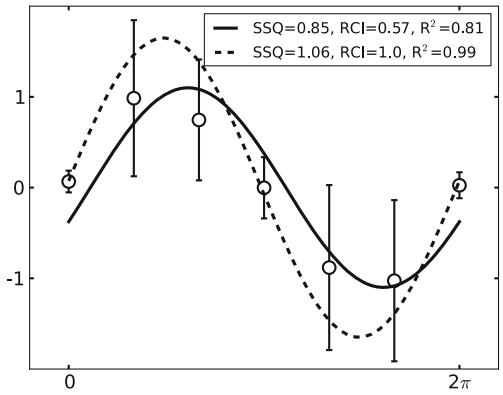

Fig. 1 Example of two candidate model fits to sinusoidal data with noise. All data are synthetic. Note that, according to the SSQ measure of fit quality, the solid line would be considered a better fit, whereas the RCI and R^2 measures would both find the dashed line to be more representative of the data

Gradient Following Algorithms

For non-linear optimisation problems, in our case the minimisation of the *SSQ*-error, Gradient Following (GF) algorithms are the most commonly used. To visualise this optimisation problem we can imagine an error surface as a landscape in which we attempt find the lowest point. The principle of the GF method, is to follow the gradient of the error down towards this minimum. There are numerous practical approaches to accomplish this task.

The Nelder–Mead method is a direct search scheme that performs geometrical manipulations of a simplex placed on the error surface (Heath 2002; Nelder and Mead 1965). This is a method that is effective in low dimensions where it is suitable to fit to non-smooth objective functions (Lagarias et al. 1998). In higher dimensions it is very computationally expensive compared to other GF methods.

The Newton method for optimisation is an alternative and is guaranteed to converge to a minimum, at a quadratic rate (Heath 2002; Nash and Sofer 1996) for points on the error surface sufficiently close to the minimum. For more distant points, however, the convergence will be slower and it may fail altogether. Furthermore, the method requires construction of the Hessian matrix (the second derivative of the error surface) and, as with the Nelder-Mead method, it is computationally expensive for high-dimensional problems. A number of *quasi-Newton* methods have been developed, which approximate the Hessian, making them computationally cheaper, less sensitive to the starting parameters, or both. They typically differ in the way the Hessian approximation is made. The most common quasi-Newton schemes are secant updating, conjugate gradient, nonlinear least squares and truncated Newton methods (Heath 2002; Broyden 1967; Dixon 1972; Nash and Sofer 1996).

One obvious problem with following the gradient of the error surface downwards to a minimum, is that we cannot know if this is the lowest point the surface ever is reached or just a local minimum. The choice of the length of step to be taken may be critical in this, since a large step size may allow local minima to be passed, but may of course also prevent the global minimum from being discovered. Ultimately the search is local and if we choose a starting point far from the global minimum there are no guarantees that this will be found. For a system where the parameters, and their effects on the overall behaviour of the system, are well-understood, finding a starting point close to the global minimum may be achievable. The very nature of complex systems, however, makes this task extremely difficult or impossible. An alternative might be to use multistart methods, whereby a number of GF searches are run in parallel, with different initial conditions (Bolton et al. 2000).

Evolutionary Algorithms

A different approach to optimisation is to use evolutionary algorithms (EAs). These algorithms are inspired by biological evolutionary processes, whereby a population consists of individuals from one or more generations that contribute in some form to future generations in a non-deterministic manner. As of lately this class of methods have become increasingly popular as a tool to fit complex models to neuronal data (Van Geit et al. 2008). Each instantiation of the model can be considered an *individual* whose characteristics (described by the model parameters) can be mutated and propagated to future instantiations. The probability that an individual contributes to future generations depends on its *fitness*. Depending on the EA being used, this contribution may take the form of parameters either being mutated and then passed directly to its 'offspring' (in a manner akin to cell division), or can be mutated and combined with the parameters of another individual (akin to breeding). Fitness of an individual is determined by its quality of fit relative to the rest of the *population*, which comprises other individuals of the current, and possibly previous, generations. The system is non-deterministic in that all mutations and breedings are made on a random basis and even individuals

with poor fitness may have an opportunity to influence future generations.

EAs can differ in; the evolutionary strategy (ES) governing the way in which future generations are formed, the structure of the population (e.g. the size and lifetime of each generation), and the calculation of fitness for each individual. There are numerous ESs of varying complexity which govern how the generations are produced based on the fitness evolution (Hansen et al. 1995; Hansen and Ostermeier 1997). For a given EA there are numerous choices of which ES to implement (Sbalzarini et al. 2000). In the simplest case only mutations occur and these mutations are drawn from a constant distribution. Alternatively the noise distribution from which the mutation is generated could be adaptive. More sophisticated strategies implement a combination of mutation and breeding, where there are also many potential ways to implement 'breeding'. For example, the method might make use of correlations between the fittest individuals in different generations to determine the most effective way to generate following generations. The structure of the population can vary in the lifetime of each generation; in one extreme, individuals are removed from the population as soon as a new generation has been created such that only the new generation can contribute to future generations, whereas in the other extreme, all individuals in the history of the system can contribute to a new generation. In practice, the least fit individuals from past generations may be discarded since they are not likely to contribute anyway. This leads to a population structure containing the current generation and an elite pool of fit individuals from previous generations. The other consideration for the population structure is the size of each generation; if this is too small the parameter space of the model may not be well-explored, whereas if it is very large the computational cost of quantifying fit quality for each individual becomes large.

Quantifying the fitness of an individual in an EA is more flexible than simply measuring the quality of fit through a single error measure, as in GF methods. In particular it is possible to perform multiple objective optimisation (MOO) (Druckmann et al. 2007; Vrugt and Robinson 2007) using EAs. If there are several possible error measures that we wish to minimise, and there may not be a unique solution that minimises all, we can track all these errors simultaneously in the fitness of the individual, which is not possible using GF methods. There are also multiple options to implement this MOO. One commonly-used solution (e.g. see Zitzler and Thiele 1999; Deb 1999; Druckmann et al. 2007) is to determine which individuals are on a *Pareto front* in the error space—the set of individuals that are not improved upon by any other individual on all error terms, they are *non-dominated* (see Fig. 2)—and then calculate fitness based upon the distance of each individual from that Pareto front.

Using the Algorithms in Complex Systems

Due to the probabilistic nature of the EAs, there are no theoretical guarantees that a minimum will be found in finite time. On the other hand they may be less susceptible to converging on local minima, by the fact that the set of solutions that are non-dominated span several locations on the error surface and the fact that they are non-deterministic. In a system about which we have little prior knowledge, and where we expect local minima to be a problem, it may be more important that we are confident the algorithm has found something close to the true global minimum than that it has converged accurately and efficiently.

In our case, we have a 9-parameter nonlinear multi-stage model of a neuron in primary visual cortex that we wished to fit to response data from a large set of recordings made in anaesthetised macaque monkey. By performing manual fitting we found that the GF methods that we initially used in this optimisation problem frequently converged on solutions that were clearly not optimal. This led us to consider an evolutionary approach and to explore its performance relative to GF methods for tackling our real-world optimisation problem. In particular we wished to uncover whether the EA could find a superior solution to that found by the GF method. We would also like to explore the class of multiple objectives best suited to capturing the

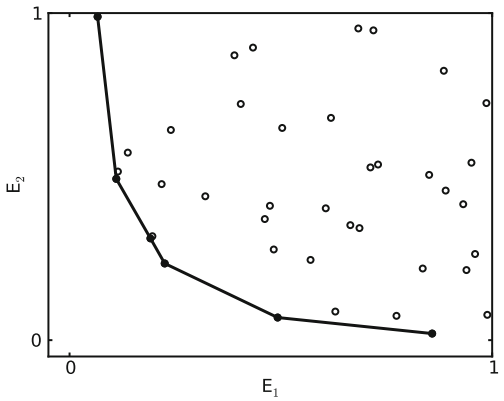

Fig. 2 An example of a two dimensional Pareto front. Each point in the figure is a solution that gives values for two errors, E_1 and E_2. When attempting to mimimise both errors we get a Pareto front. On the Pareto front the solutions are optimal in the sense that there is no other point for which both E_1 and E_2 are smaller, the *filled dots* connected with lines show the Pareto front in this figure

data, and contrast the computational cost of using GFs to EAs. For the sake of this comparison we focus primarily on an implementation of the Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasi-Newton GF method (Broyden 1967) and the Strength Pareto Evolutionary Algorithm (SPEA) (Zitzler and Thiele 1999; Sbalzarini et al. 2000; Zitzler et al. 2003).

We found that, although the BFGS method generated slightly better fits for some cells, the SPEA was more consistent in producing good fits, such that it substantially outperformed the BFGS for some neurons. The BFGS typically took roughly an order of magnitude less computational time to perform its optimisation, but required substantially more human intervention in the choice of good starting parameters. Fitting synthetic cells revealed that the model has a very complex and flat error surface. BFGS turns out to stay in the vicinity of initial parameters while SPEA does a much more efficient global search. We also tested a combined method in which the SPEA was used to generate initial parameters for the BFGS. This memetic approach indicated that the SPEA did typically converge to a point close to the minimum even though this was not guaranteed. The SPEA also had the advantage of providing a family of fits that were, in some sense, equally good rather than giving the illusion of there being a single best fit.

Method

Data and Model to be Fit

Briefly, the data were spike rate measures taken from extracellular recordings of single neurons in primary visual cortex (V1) of paralysed, anaesthetised macaque monkey (Macaca Fascicularis). The data used here were recorded not for this purpose, but to characterise the receptive fields of the neurons as needed for other experiments (Solomon and Lennie 2005; Webb et al. 2005; Peirce et al. 2008). We are grateful to the members of Peter Lennie's lab that allowed the data subsequently to be used in this way. The neurons were stimulated with drifting sinusoidal gratings, which could vary in spatial frequency (SF), orientation, contrast and drift rate (temporal frequency, TF). The gratings were presented in circular apertures that could vary in size. The neurons are tuned in different ways to each of these characteristics; they each have a preferred stimulus orientation, SF, TF and size and also differ in the bandwidth of their tuning to these dimensions. For stimulus contrast the majority of cells have a nonlinear (typically saturating) response function. In Fig. 3 we show sets of tuning curves for four sample neurons to illustrate some of the characteristic behaviours that can be seen.

Our aim was to fit a multi-stage nonlinear model to the data in an attempt to capture all of these tuning characteristics in individual neurons. The dataset consisted of tuning curves for 107 cells, collected from 13 macaques. The data are somewhat noisy for various reasons, both physiological and technical. Neurons themselves can adapt; changing their responses to the same stimuli after prolonged stimulation. The nature of in vivo extracellular recordings means that spikes can be missed as the target neuron moves closer or further from the electrode during recording sessions. Lastly, in the particular recordings made here for which the experimenter primarily wanted to establish the preferred orientation, for example, without necessarily needing a full high-quality characterisation of the entire tuning curve and so there are numerous cells for which the confidence interval for some datapoints is rather large. All of these are common problems when models are being fit to real-world complex data from relatively noisy environments.

The model is based on Gabor filters and has 9 parameters. Earlier filter-based models have captured the behaviour of single neurons to individual tuning dimensions. The main ingredients of the model are a temporal filter, $f(t)$, and a spatial filter, $R(x, y)$. These are combined in the same manner as in Watson and Ahumada (Watson and Ahumada 1985). Importantly we also need a stimulus, $S(x, y, t)$ which we will convolve with the filters. The stimulus is mathematically described in Appendix A. The spatial filter consist of a classical receptive field (CRF), a broadly tuned suppressive field and a contrast normalisation pool. The CRF is a Gabor filter which can be used to model a neuron with tuning to the orientation and spatial frequency of stimulus gratings (Marcelja 1980; Daugman 1985). The broadly tuned suppressive filter is also a Gabor with larger size and lower spatial frequency than the CRF. This filter provides the size tuning of neurons by adding the suppressive in either a divisive or subtractive manner (Simoncelli and Schwartz 1999; Sceniak et al. 2001; Cavanaugh et al. 2002). In our model we will use divisive suppression. The contrast normalisation pool is modeled as a Gaussian window of the same size as the CRF. This filter is not sensitive to orientation, spatial or temporal frequency but only to stimulus contrast. This filter effects the shape of the contrast tuning curve and also works in a divisive way (Heeger 1992; Carandini and Heeger 1994; Carandini et al. 1997). The temporal filter is a low pass filter that gives a temporal frequency tuning (Watson and Ahumada 1985). To achieve a

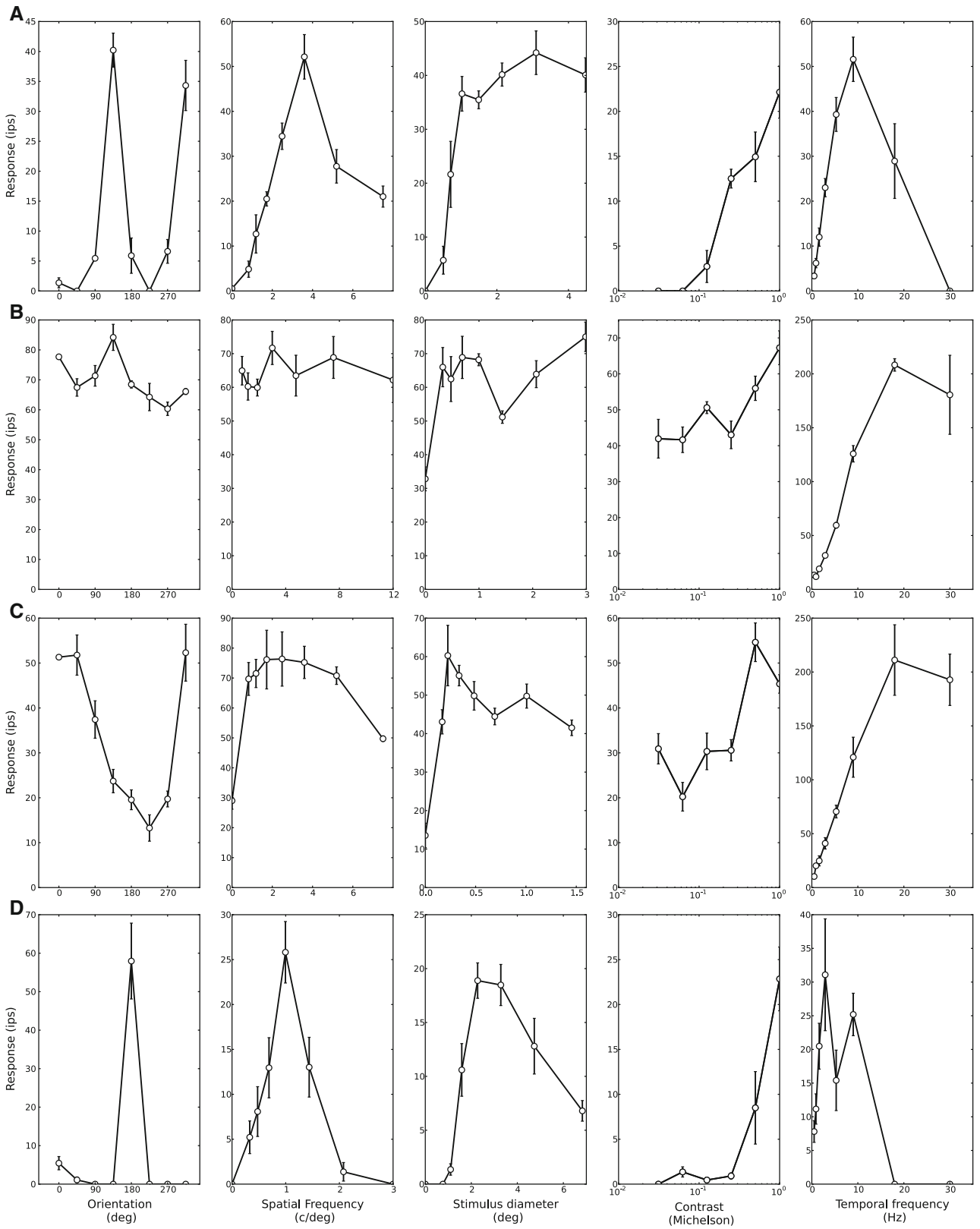

Fig. 3 Four sets of tuning curves that we will attempt to fit using different methods. These cells show typical shapes of the curves and typical firing rates, bars indicate standard errors. We see a wide spread of tuning properties, for example both sharp (**A** and **D**) and flat (**B** and **C**) orientation tuning. In **A** we have almost no direction selectivity while this effect is very strong in **D** although both cells are similar in terms of orientation bandwidth and firing rates. Note the wide range of tuned values for size and spatial frequency tuning across cells

directional selectivity of the model we have to take the Hilbert transform of both the spatial and temporal parts of the model. The Hilbert transform $\mathcal{H}(g)$ transform a function g into a quadrature of it-self (Khvedelidze 2001; Watson and Ahumada 1985). In accordance with Watson and Ahumada the responses from the original filters and the Hilbert transformed filters are then summed. The total response is then calculated as

$$Q(t) = f * \delta * \left[\int_{\mathbb{R}^2} [R \otimes S](x, y)\, \mathrm{d}x\mathrm{d}y \right](t) + \mathcal{H}(f) * \delta * \left[\int_{\mathbb{R}^2} [\mathcal{H}(R) \otimes S](x, y)\, \mathrm{d}x\mathrm{d}y \right](t), \tag{1}$$

where $\otimes$ indicates two dimensional spatial convolution. Temporal convolution is denoted with $*$ and we also introduce a small time delay $\delta = \delta(t - \tau)$ to avoid numerical singularities that can occur when taking the Hilbert transform. The temporal response for each stimulus condition is then denoted $Q(t)$. For a full description of the calculations leading up to Eq. 1 see Appendix B.

The symbolic names of the nine parameters that we fit in the filter model are listed and described in Table 1. There we also state which filter each parameter is associated with. These components of the model are relatively well understood, but when combined nonlinearly, as is the case here, they form a complex system.

Implementation

The stimulus, model and optimisation routines are all written in the Python programming language. The full source code to implement the model is available from http://code.google.com/p/findv1/. For a number of reasons Python is becoming a popular choice of language for solving problems in computational neuroscience and neuroinformatics. In particular it is cross platform, open source, object oriented and many scientific and computational libraries are already implemented, e.g. the *Scipy* library (Jones et al. 2001). A number of recent articles in neuroinformatics journals have recently appeared that further make the case for the use of Python as a tool to unite community activity (Goodman and Brette 2008; Einevoll 2009; Davison et al. 2009; Spacek et al. 2009).

Table 1 Symbolic names and descriptions for the nine parameters of the filter based model of a V1 cell

θ	Orientation of the Gabors, both classical and tuned suppressive receptive fields
k_c	Gain of the CRF
σ_c	Standard deviation of the Gaussian that windows the CRF and normlistation pool
k_s	Gain of the tuned suppressive receptive field
σ_s	Standard deviation of the Gaussian that windows the tuned surround
ω_c	Spatial frequency of the CRF
n_g	Exponent of the contrast gain normalisation that decides the shape of the contrast curve
τ_1	Parameter that determines the time course for the temporal filter
τ_2	Parameter that determines the time course for the temporal filter

Calculation of Error Measures

All error measures are based on the difference of tuning curves between the electrophysiologically recorded cells and model cells. For each of the five tuning dimensions, μ, we calculate a model tuning curve, T^*_μ, that is compared with the corresponding electrophysiological tuning curve, T_μ. The SSQ for tuning dimension μ is then calculated as

$$SSQ_\mu = \sqrt{\sum_i^N \left(T^*_{i,\mu} - T_{i,\mu} \right)}, \tag{2}$$

where i indicates the N recorded points, for example the 8 orientations used to obtain the orientation tuning curve. To get the total SSQ we then sum according to

$$SSQ = \sum_{\mu=1}^{5} SSQ_\mu, \tag{3}$$

where naturally μ are our five tuning dimensions.

For the RCI we calculate the 95% confidence intervals for each recoded point to get an interval $T_\mu \pm c$. The ratio is then calculated as

$$RCI_\mu = \frac{\sum_i^N \Theta\left(T_{i,\mu} + c_i - T^*_i\right) \Theta\left(T^*_{i,\mu} - \left(T_{i,\mu} - c_i\right)\right)}{N}, \tag{4}$$

where $\Theta(x)$ is the Heaviside step-function. The total RCI is then the averaged across tuning dimensions

$$RCI = \frac{\sum_{\mu=1}^{5} RCI_\mu}{5}. \tag{5}$$

In the same fashion we calculate the variance explained using the Pearson's correlation coefficient

$$R^2_\mu = \left(\frac{\sum_{\mu=1}^{N} \left(T_{i,\mu} - \bar{T}_\mu\right)\left(T^*_{i,\mu} - \bar{T}^*_\mu\right)}{\sigma_{T_\mu}\sigma_{T^*_\mu}} \right)^2, \tag{6}$$

where $\bar{T}_\mu$ and $\bar{T}^*_\mu$ are the mean of the tuning curves while σ_{T_μ} and $\sigma_{T^*_\mu}$ are the standard deviations. This is then averaged to give

$$R^2 = \frac{\sum_{\mu=1}^{5} R^2_\mu}{5}. \tag{7}$$

Gradient Following Algorithms

The BFGS method has proven successful in many real world applications in areas such as systems biology, chemistry and nanotechnology (Kim et al. 2007; Pankratov and Uchaeva 2000; Zhao et al. 2002). We chose a version of BFGS with limited memory usage and box constraints, namely the L-BFGS-B method, already implemented in the Scipy library. The limited memory aspect of the implementation means that the whole gradient history is not considered when the Hessian is calculated, thereby saving memory. L-BFGS-B combines the well known computational effectiveness of BFGS (Nocedal 1980; Fiore et al. 2003) with box constraints on the parameters. Such constraints are necessary in our model since some parameters must be positive. To further assess the computational cost and convergence properties of BFGS we compare it with a truncated Newton code (TNC) (Nash 1984; Nash and Sofer 1996; Schlick and Fogelson 1992) with parameter constraints (also implemented in Scipy), which is known to be robust in convergence but computationally costly. We use the SSQ error as the one to be minimised by the GF algorithms, in line with common practice.

When we use the Scipy BFGS implementation we limit the number of function evaluations to 10,000. In practise the run from a single initial condition stops far earlier. The most common reason for termination of the optimisation is that the method converge, i.e. a minima is found. Other reasons can be that the gradient can not be properly evaluated to advance the optimisation, this can be due to saddle points, maxima or other degenerate states in error space. If this would be the case new initial conditions are selected and the method is run again until convergence is reached.

GF algorithms generally converge much faster than EAs, though not necessarily to the global optimum. Convergence to a global optimum is highly dependent on good initial parameters. To find appropriate initial parameters requires a good understanding of the model and the effects of parameters on model output. The interdependencies between different model parameters in our case makes this a challenging task, requiring expert knowledge of the system. For example, the effects of the CRF and suppressive field gain parameters are highly dependent on their size difference. One main method that we have employed to find initial parameters is to utilise known effects that parameters have on things such as maximum values and maximum arguments of the tuning curves. To illustrate this notion consider the time constants for the temporal filter τ_1 and τ_2. By appropriate choices of the time constants $(\tau_1, \tau_2) = (k_1, k_2)/\omega^*_t$, where (k_1, k_2) are constants, then a temporal tuning curve is generated that has a peak at a preferred frequency ω^*_t. The pair (k_1, k_2) can either be fitted numerically or deducted from the model equations that governs the temporal filter. Other parameters that are more interdependent can not naturally be chosen so easily, and we have had to adopt more *ad hoc* approaches. For example, consider the normal pool exponent parameter, n_g, that determines in part the shape of the contrast tuning curve. For $n_g = 1$ the contrast tuning curve is a linearly increasing function of contrast. For high values of n_g the curve is sub-linear whilst for small values of n_g grows faster than linear. However, the exact shape of these curves is further determined by the two gain parameters k_c and k_s. Hence depending on the desired shape of the tuning cure we set $n_g = 1$ (for a near linear tuning curve), $n_g = 0.8$ small (for an accelerating tuning curve) and $n_g = 1.6$ for a sub-linear shape, for fixed values of (k_c, k_s) (which can also be initialised in a manner that makes them consistent with the desired shape of other tuning curves). In Table 2 an

Table 2 Descriptions of how initial parameters for BFGS are chosen

Parameter	Method to acquire starting value
θ	Pick the orientation from the electrophysiological curve with highest firing rate
k_c	This gain is proportional to the maximum value of the size tuning curve
σ_c	The stimulus size when experimentally deciding the temporal tuning curve
k_s	Dependent on the ratio between the maximum response and the response for the largest stimulus in the size tuning curve
σ_s	The size of the biggest stimuli from the size tuning curve
ω_c	If a clear peak occurs in the data then the value is chose at that peak. If not, the value used when collecting later electrophysiological tuning curves is chosen
n_g	If the curve accelerates then initiated as 0.8, nearly linear curve 1.0 and if the contrast saturates the initial value is 1.6
$\tau_{1,2}$	Decided by $(\tau_1, \tau_2) = (k_1, k_2)/\omega^*_t$, for constants (k_1, k_2), where ω^*_t is the desired peak frequency The constants (k_1, k_2) are picked by fitting how choices of the constants give rise to a particular peak frequency

overview is given of how the initial model parameters are chosen. For BFGS we choose initial parameters based on the criteria in Table 2.

Strength Pareto Evolutionary Algorithm

The SPEA is based on the notion of Pareto optimality (Zitzler and Thiele 1999; Sbalzarini et al. 2000; Zitzler et al. 2003). We can define a *Pareto front* as the set of individuals that are not *dominated*, that is to say, not improved upon by any individual in all error measures. The SPEA advances such a front in the MOO error space such that each component error gradually decreases (or remains stationary). The Pareto optimum is the solution for which no further front advances can occur. In the SPEA the population structure consists of the current generation (described in Section "Evolutionary Algorithms") together with an elite population of non-dominated individuals regardless of which generation they were created in. The elite population is in fact the Pareto front. A member of the elite population survives until a new individual that dominates it is created. The new individual then takes the place of the individual(s) it dominates. The size of the elite population is limited and if the number of individuals on the Pareto front exceeds the maximum it is reduced through *clustering*, whereby the individuals lying closest together in the error space are grouped together. The cluster is then replaced by a single point that is the individual closest to the centre of the cluster. The fitness function for an individual in SPEA is based on domination. The individuals that lie on the current Pareto front determine their fitness based on how many individuals in the population they dominate. The fitness for the rest is based on how many other individuals in the population dominate the individual in question. The mating pool is filled by randomly selecting two individuals from the population, the one with best fitness is added to the mating pool. This is then repeated until the pool is filled (to a pre-determined size).

The evolution strategy that we use is the $(\mu/\mu_I, \lambda)$-CMA-ES first introduced by Hansen and Ostermeier (Hansen and Ostermeier 1997). This variant of the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) looks at the covariance between generations and aligns the hyperellipsoid of mutation distributions along the axis of greatest predicted progress. In practice this means that information about the *evolution path*, the direction in which new and improved individuals are generated, is gathered and correlations are recorded. If movement in certain directions in the parameter space tends to improve solutions this is recorded thorough correlations in the evolution path and collected at the mth generation in the correlation matrix $\mathbf{C}(m)$. In our case, this is a 9×9 matrix due to the 9 parameters that are being fitted in the model. Let us denote individual i in the current generation as $x_i(m) \in \mathbb{R}^9$ and members of the mating pool as $\widehat{x}_j(m) \in \mathbb{R}^9$. The mutation step from generation m to $m+1$ is done by

$$x_i(m+1) = \widehat{x}_j(m) + \delta(m)\mathbf{B}(m)\mathbf{D}(m)\mathbf{z}(m), \tag{8}$$

where $\delta(m)$ is a global scaling parameter and $\mathbf{z}(m)$ is a normal distributed vector. The square roots of the eigenvalues of the correlation matrix are arranged in the diagonal matrix $\mathbf{D}(m)$ and the orthogonal matrix $\mathbf{B}(m)$ are such that $\mathbf{C}(m) = \mathbf{B}(m)\mathbf{D}(m)(\mathbf{B}(m)\mathbf{D}(m))^T$. The update rules for all matrices and the global scaling parameters are given in (Hansen and Ostermeier 1997). We consider 32 individuals in each generation and we limit the number of generations to 250. The calculation of the errors for all individuals at a given generation is naturally implemented in a parallel fashion, whilst the fitness calculation and the mating step is performed in a serial manner.

The parameters for the individuals in the initial population are drawn from different random distributions. These distributions are generally uniform between two limits that are chosen to span possible values of the parameter. If the SPEA produces unrealistic parameters, as negative gains or time constants, these individuals are weeded out due to poor fitness.

Results

Computational Cost

All numerical tests have been performed on a 2 × 2.26 GHz Quad-Core Intel Xeon with 6 GB 1066 MHz DDR 3 memory. Each of the 8 cores has has 256 KB L2 cache and each processor has 8 MB L3 cache.

For SPEA the question about how to measure computational costs depends on the decision of how many individuals are included per generation and how many generations to run. Figure 4 shows the evolution of the SSQ for three example cells across 250 generations of a 32-individual SPEA. By 250 generations improvements in fit quality are uncommon and small.

The fitting of cells using TNC (Nash 1984; Nash and Sofer 1996; Schlick and Fogelson 1992) takes approximately 36 h and 30 min per cell while SPEA reaches 250 generations in 14 h. The quickest algorithm by some margin is BFGS that converges in 1 h and 20 min per cell. TNC is exemplary stable and manages to find (potentially local) minima from a wide range of initial data. The quality of the minima found by

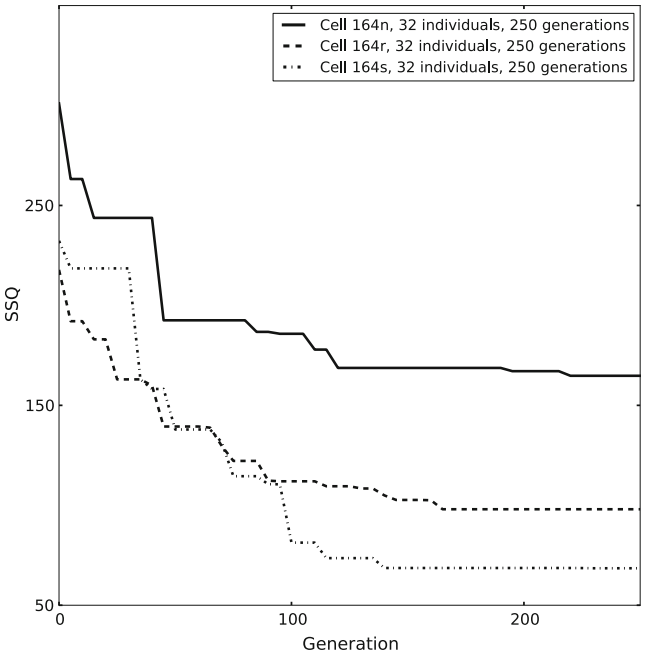

Fig. 4 The evolution of the SSQ over 250 generations for three cells fitted with SPEA

BFGS is, in general, as good as those found by TNC but BFGS is much more sensitive to the choice of initial parameters. If the initial parameters cause very low firing of the model cell BFGS often appears to get stuck in this very undesirable local minimum. This does not generally happen to TNC and it finds a minimum that is reasonable but the time to converge is considerable higher. If we have reliable and good initial conditions the result is that BFGS is much quicker at finding a good fit but TNC is much more stable but requires a great amount of time to converge. This is due to the greater number of function-gradient evaluations that TNC is performing for each iteration (Nash and Nocedal 1991). Hence it might not require more iterations but more function evaluations and is therefore more costly. When comparing BFGS with SPEA we find that it is roughly ten times faster to converge. SPEA is insensitive to local minima.

BFGS and SPEA Fits

More important than computational cost, particularly given the low financial cost of computing, is that we are able to produce fits of high quality. Consider the fits to the tuning curves of Fig. 3. In Fig. 5 we plot the electrophysiological tuning curves but this time with the 95% confidence intervals for each data point. We also plot model tuning curves for fits to each of the four cells for both BFGS (dotted) and SPEA (dashed). Across the set of tuning curves we see that the two methods can find very different solutions. Figure 5A and B show cases where both methods find solutions that look reasonable while in C and D we have cases where BFGS have settled for solutions that are not close to the electrophysiological recordings for at least two of the tuning dimensions. Another common problem for BFGS are that the method gets stuck in a state of no firing and is unable to escape from this. For each BFGS fit we get only one solution (by the very nature of the GF algorithm), whereas for the SPEA this is not the case because it performs MOO. This typically gives a Pareto front of optimal fits rather than a single best solution. The number of solutions on this final front is bounded by the number of allowed individuals in the SPEA elite population but, in our case, commonly comprises 20–30 different individuals. In Fig. 5 we plot the SPEA fit that has the smallest SSQ error along the front. The solutions along the Pareto front can vary fairly strongly (in the shape of their tuning curves) among each other as can be seen in Fig. 6 where five fits from a Pareto front have been plotted. All the fits on the front are viable 'best' solutions for MOO, not being dominated by any individual on all error terms.

When comparing the performance of the methods this spread of solutions along a Pareto front for the SPEA should be kept in mind. If we are only interested in optimising one of the errors we can simply pick the solution from the front that gives the best value for this error. In Fig. 7 we show the average values for min(SSQ), max(RCI) and max(R^2) on the final Pareto fronts together with the BFGS results. In the large majority of cases a solution on the Pareto front can be chosen that outperforms the solution obtained by BFGS (for all choices of our goodness of fit measures). However, it is worth noting that BFGS is able to produce good fits. In Fig. 5 we see that examples A and B are very good fits to the data both in the case of BFGS and SPEA. We have in Table 3 given the means, medians and standard deviations for the fitted cells and in Fig. 8 we have the error distributions for all cells. From the histograms in Fig. 8 the mode is almost unchanged for all errors when comparing SPEA with BFGS. Here, the SPEA results have the lowest errors and the standard deviation is greater for BFGS in all errors. As a consequence many of the BFGS solutions do not provide good fits at all (being far from the mode).

In Fig. 9 we have scatter plots of the error measures for each individual cell. For the RCI and R^2 measures SPEA performs better than, or equal to, BFGS for almost every single cell. There are extremely few examples where BFGS is better than SPEA. It might

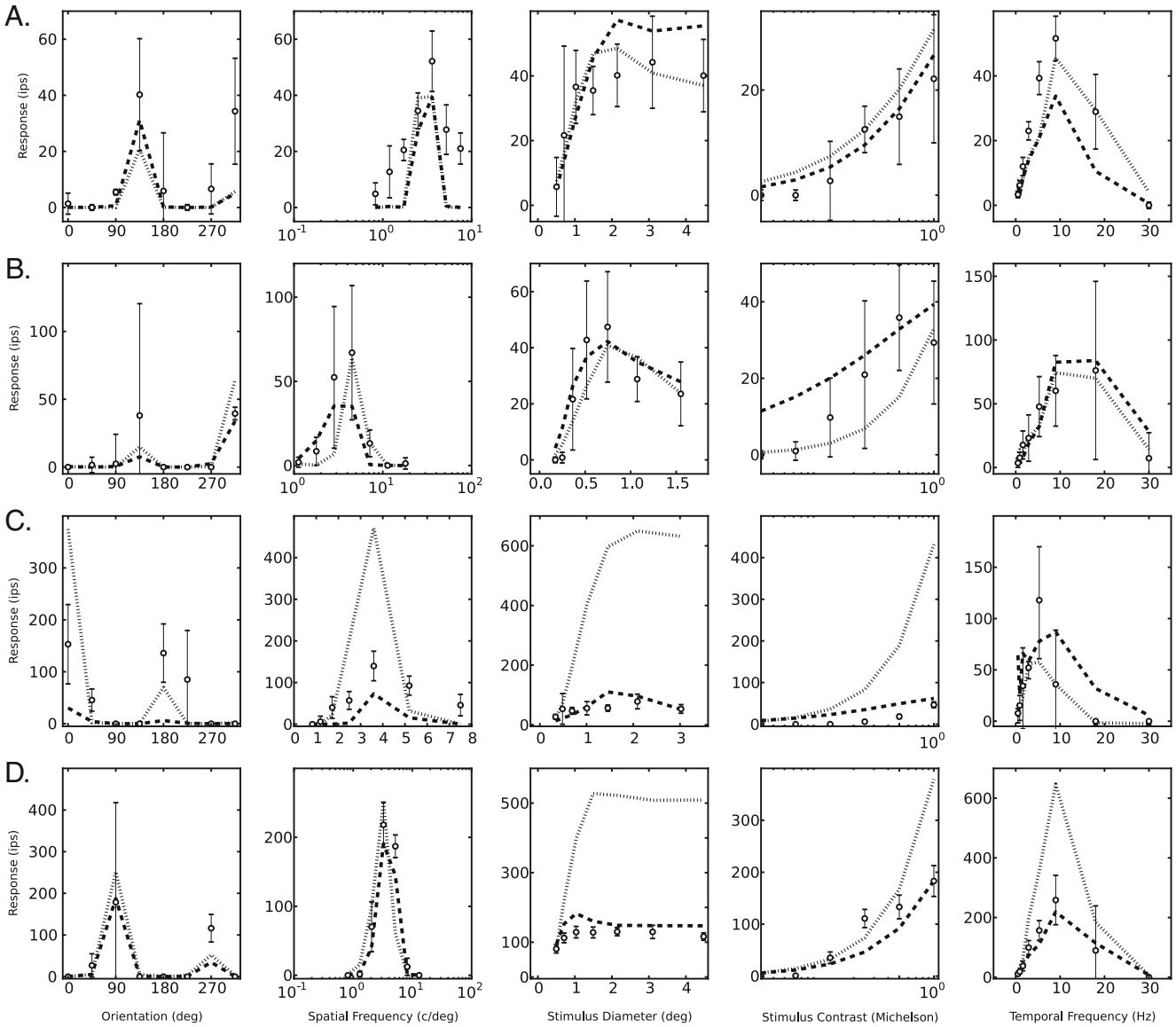

Fig. 5 Examples of fitted model (*curves*) to electrophysiological data (rings with bars indicating 95% confidence intervals). Dashed lines are SPEA fits and dotted are BFGS. In **A** and **B** both BFGS and SPEA do a good job of fitting the data. In **C** the results of both methods are questionable although SPEA lies reasonably close to the recordings. **D** is an example of where SPEA does a good job while BFGS is very far from the recoded data

not be so surprising that SPEA outperforms BFGS on RCI and R^2 but in Fig. 9 C and D it is obvious that SPEA manages to find a better SSQ-solution is almost all of the individual cases. It is actually only 11 of the 107 fitted cells for which BFGS finds a better solution. The mean and median improvement seen in Fig. 8 might seem small but here we see the improvement is incredibly consistent across cells. This tells us that BFGS is generally incapable of finding global minima and that SPEA does a much better job of this. The overall aim of this fitting is to examine how well our model captures electrophysiological data and as SPEA is finding better, or at least equal, fits in almost every single case we can state that the GF method is not appropriate for the task.

Although the vast majority of fits were better when using the SPEA than BFGS we can see in Fig. 9 that this is not always the case. In some cases the SSQ value given by BFGS is lower than by SPEA. To explore the reasons for this we look closer at a few cells with different fitting results. The first cell is *m177ae* with the SSQ values of 323.3 for BFGS and 578.8 for SPEA, this

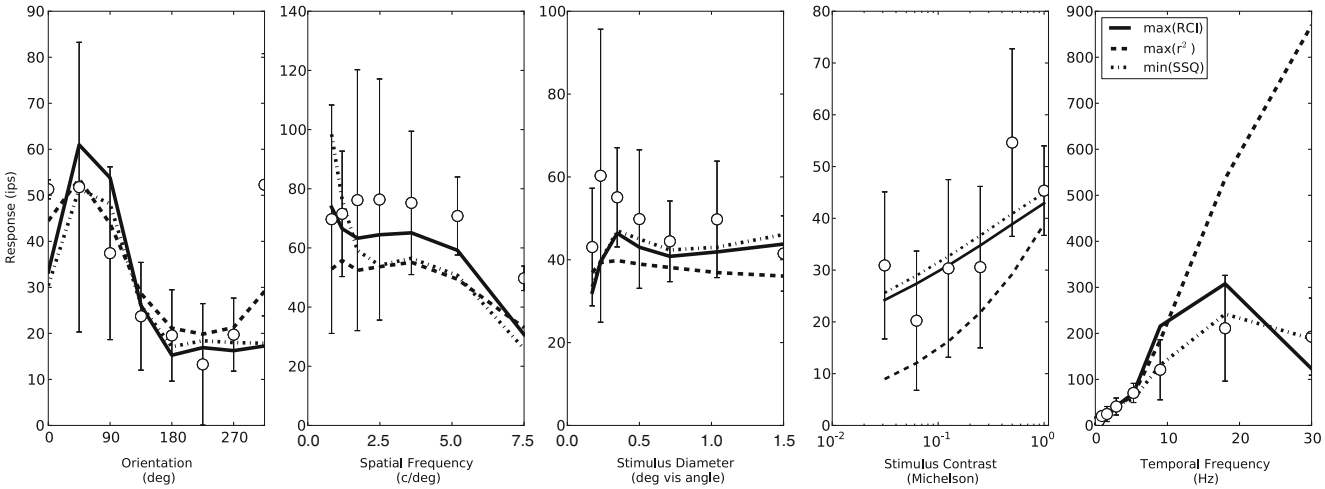

Fig. 6 Five fits from the SPEA to cell A in Fig. 5. We have the min(SSQ) (*dash-dotted*), max(RCI) (*solid*), max(R^2) (*dashed*) and two randomly chosen points (*dotted*)

is actually the cell that is highest above the unity line in Fig. 9C. Due to the stochastic nature of EAs there is a natural risk that the global optimum is not found at each run, and in this case it seems that SPEA settled quite far from the global optimum. To investigate the effects of altering the number of iterations and the size of each generation in the SPEA we reran that fit; 10 times with the original configuration (32 individuals, 250 generations); 10 times with 64 individuals, 250 generations; and 10 times with 32 individuals, 500 generations. The results of these fits are displayed in Fig. 10 (left panel). For this cell the BFGS appears to find a genuinely optimal solution. SPEA normally, but not always, settles close to the BFGS solution for all three choices of individuals and generations. The fact that the first recorded solution had a much worse SPEA result than BFGS can therefore be attributed to the stochastic nature of EAs. Increasing the number of generations or individuals does not improve the results dramatically, although the version using extra generations has no

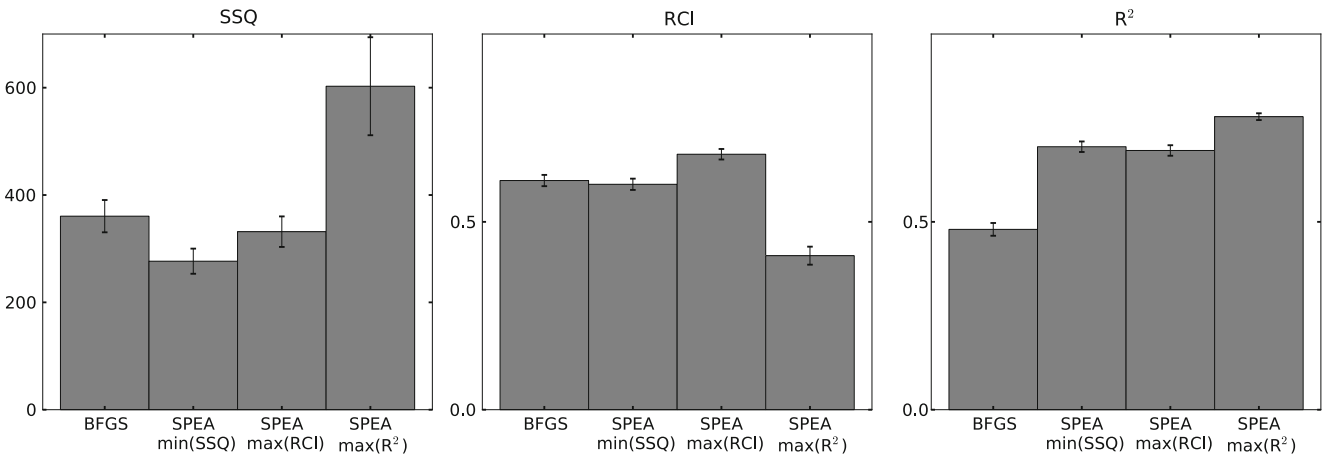

Fig. 7 The average values for SSQ, RCI and R^2 after fitting. Note that low values for SSQ indicate good fits, whereas high values for RCI and R^2 indicate good fits. For SPEA fits three choices from the final Pareto front are made. The mean minimal errors from the fronts are significantly better than those from BFGS, but a fairer comparison to BFGS might be to look at the errors from the min(SSQ) point of the fronts, since that is what the BFGS was optimising. At these points the minimised quantity SSQ is considerably lower in the fits produced by the SPEA which indicates that it performs better at finding global minima than the BFGS. The values of RCI (*centre panel*) are very similar between the two methods, although the SPEA fit with max(RCI) performs poorly here. The SPEA outperforms the BFGS on the (R^2) measure (*right panel*) regardless of which final point on the *pareto front*, indicating that the *entire front* is taking better account of the 'shape' of the tuning functions in generating fits. Error bars indicate standard errors of the mean

Table 3 The mean, median and standard deviations of our fitted cells using SPEA and BFGS

	BFGS			SPEA		
	SSQ	*RCI*	R^2	*SSQ*	*RCI*	R^2
Mean	373.17	0.48	0.61	273.15	0.69	0.79
Median	247.69	0.47	0.63	170.81	0.70	0.80
Standard deviation	340.52	0.19	0.17	241.06	0.15	0.09

The distributions are plotted as histograms in Fig. 8. In the cases of SPEA the best obtained value for each error measure is reported. The values for other points along the final Pareto front are presented in Fig. 7

extreme outliers. We conducted a similar analysis on another neuron (cell *m169x*), for which the original BFGS fit ($SSQ = 534.3$) was considerably worse than the original SPEA fit ($SSQ = 348.8$). As for the previous cell, the SPEA fits show a range of solutions. In this case BFGS clearly does not converge on the global minimum, and the fits of the SPEA are nearly all superior. Still however, the SPEA occasionally settles on a solution that is poor, and the occurrence of this seems largely unrelated to the number of iterations that are made or the number of individuals in the populations.

Fitting Synthetic Cells

We need to know whether the reason for the SPEA algorithms better performance was simply that it was run for longer (roughly ten times the number of evaluations). Obviously simply running the BFGS for longer once it has converged on a minimum does not further reduce the error. Starting multiple BFGS instances with different initial parameters might reduce the error if one of the starting values is in the basin of attraction of the global minimum. In order to know how close our starting values would need to be to the global minimum in a system such as ours, we have created synthetic neurons for which we know what the true parameters are. In these synthetic cells we know that there is at least one global minimum with zero error. By starting BFGS analyses repeatedly from points in the parameter space with known distances from the

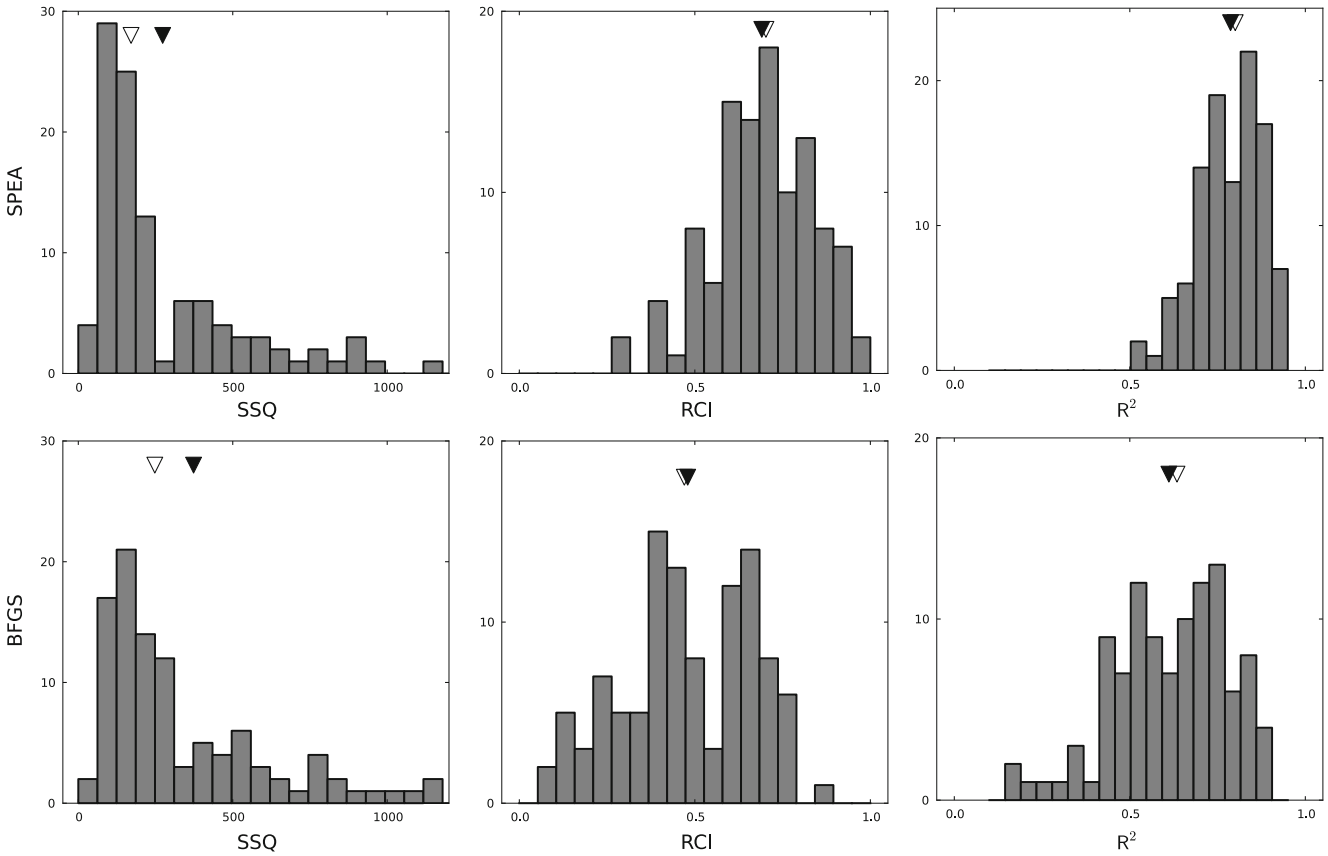

Fig. 8 Histogram showing the goodness of fit values for our fits. The *upper row* shows the SPEA fits and *below* we see the respective BFGS results. The solid triangles show medians and *white* ones show means of the distributions. The BFGS fits have a wider spread for all error measures, indicating more variable fit quality

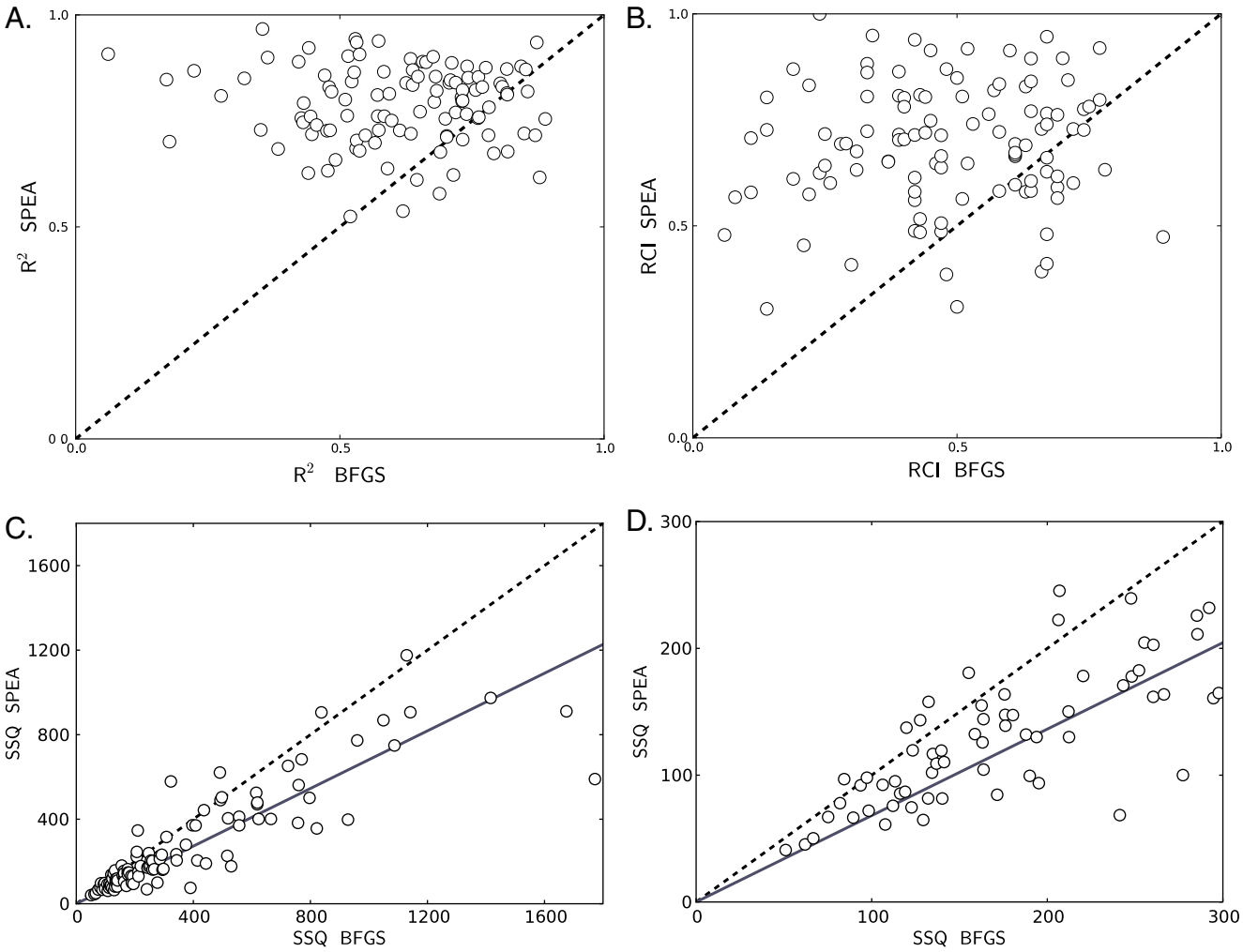

Fig. 9 The goodness of fit measures for the 107 cells as scatter plots.For R^2 values (**A**) and RCI values (**B**) the large majority of points lie above the unity line (*dashed*), thereby indicating that SPEA is performing better than BFGS. Performance on the BFGS appears uncorrelated with the equivalent performance on SPEA. Panels **C** and **D** show the scatter plot for SSQ (**D** is a zoomed view showing the lowest values of panel **C**). The points show a strong correlation between BGFS and SPEA values, indicated by the linear regression line (*solid*)

global minimum and measuring how close it gets in its final parameters, we can better understand the required proximity for the algorithm to find a fit as good as the SPEA.

Besides tracking the errors, as we do for the electrophysiological recordings, we can also track the evaluation of the parameters compared to the true values. The true parameters of the synthetic cell are denoted as $\mathbf{P}^*$. The tuning curves are then fitted with both methods with initial parameters $\mathbf{P^0}$ which are chosen as

$$\mathbf{P^0} = \mathbf{P}^* + \mathcal{N}\left(0, \Sigma^2\right), \tag{9}$$

where $\mathcal{N}(0, \Sigma^2)$ is a normal distributed stochastic vector with the same dimension as $\mathbf{P}^*$. The covariance matrix is defined as $\Sigma = \sigma\widehat{\mathbf{P}}$, where $\widehat{\mathbf{P}}$ is the diagonal matrix with the elements of $\mathbf{P}^*$ on the main diagonal. By varying σ we can move away from the true parameters and see if we relax back to the them.

The parameters of the model have different units and widely varying magnitudes and we need to define a dimensionless distance as,

$$\mathcal{D} = \sqrt{\sum_i \left(1 - \frac{P_i^n}{P_i^*}\right)^2}, \tag{10}$$

where P_i is the ith component of the parameter vector $\mathbf{P}$.

The SPEA we used had a population of 32 individuals and to allow for a fair comparison with the BFGS here we also gave the BFGS 32 start points. The SPEA was run as usual while for the BFGS we ran the 32

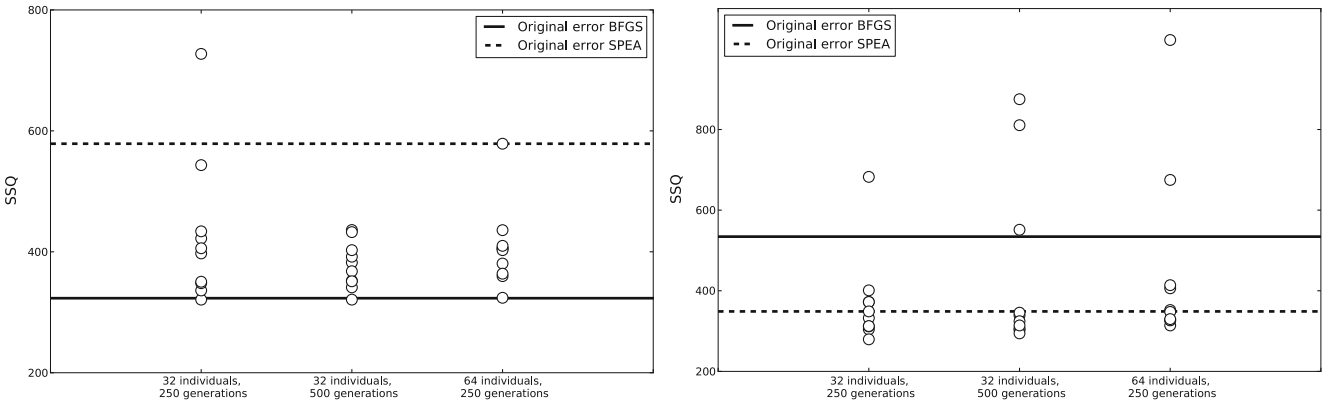

Fig. 10 The stochastic nature of the SPEA causes it to settle on different solutions each run. The lines indicate the level of the original SPEA (*solid*) and BFGS (dashed) results for two cells. In one case (*left*) BFGS performs considerably better than SPEA and in the other (*right*) the performance is opposite. The latter is the the more common case, as presented earlier. By re-running SPEA a number of times (shown by circular points) with different population settings we can see the spread of solutions that are found. In the majority of the cases SPEA settles reasonably close to what is presumably the global minimum. Although there may be a tendency to be more clustered around the global minimum when more generations or individuals are included, the SPEA in all settings occasionally fails to to find the global minimum

instances and selected the one with the lowest SSQ error after convergence.

In Fig. 11 we see the result of fitting from initial parameters chosen according to Eq. 9. In A and B we see the SSQ before and after fitting using both BFGS (dashed) and SPEA (solid) as a function of σ and in C we have the final parameter distances, $\mathcal{D}$. We first notice that even small initial deviations from $\mathbf{P}^*$ result in a non-zero SSQ and $\mathcal{D}$. This very well exemplifies how complex the error surface is as the deterministic BFGS is not able to recover $\mathbf{P}^*$. For $\sigma < 0.5$ the fitting result of both methods are similar but, while BFGS keeps an approximately linear increase of SSQ, SPEA flattens out after $\sigma = 0.75$. From $\mathcal{D}$ it is clear that the

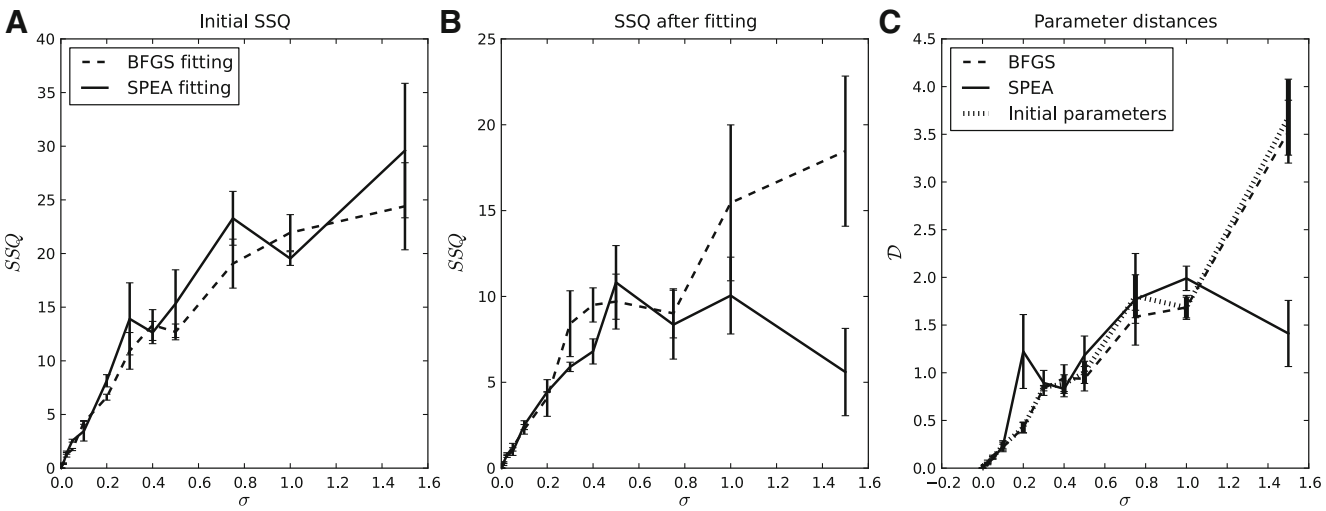

Fig. 11 The fitting results to a synthetic cell as a function of σ using SPEA (solid line) and BFGS (dashed line). Bars indicate standard errors. **A** The SSQ given the initial parameters, a distance metric between the model data and the neuronal data. **B** The equivalent metric for the best final solution found by each of the methods. **C** The distance between the final fitted parameters and the true parameters, created for this synthetic neuron. Beside the distance after SPEA and BFGS we have the parameter distributions for the initial parameter set giving the lowest SSQ averaged over both methods. BFGS is clearly confined to the local area determined by the initial parameters. It is notable that even as we add quite small errors to the parameters we do not retrieve the exact solution with either method. For small σ the two methods give practically the same results but when σ is larger than 1.0 SPEA is considerably better. SPEA is more prone to explore the error space and settle to a solution far from the initial distance. This may be closer to or further away from the real parameters in the case of synthetic cells

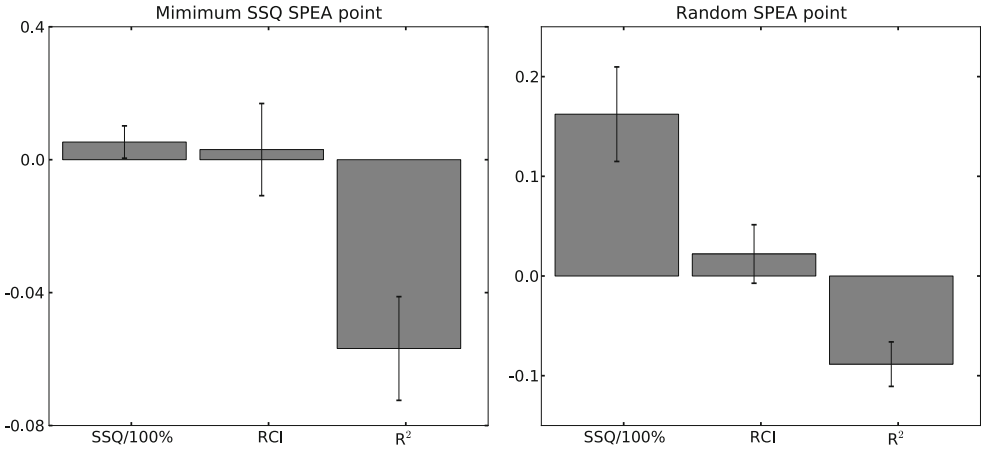

Fig. 12 The effect of using our EA fits as initial data for BFGS. To the right we see the effect as we pick initial data randomly from the final Pareto front. In the *left panel* we pick the point on the Pareto front with min(SSQ) as initial data. Bars show standard errors

BFGS is heavily dependent on the distance of the initial data while SPEA naturally explores different parts of parameter space. It is not uncommon for SPEA to move away from $\mathbf{P}^*$ during fitting where this reduces the SSQ. The dependence of BFGS on its starting parameters is also apparent from the close relationship between initial parameter distance (dotted line) and distance after BFGS fitting (dashed) in the right hand panel. It is therefore unlikely that running 5 or 10 BFGS would greatly improve the performance without being able to pick better initial conditions. This is not surprising but it highlights the fact that for a multi start it is of greater importance to pick many initial conditions than the actual GF-fitting.

These data illustrate the point that for starting parameters even a short distance from the true optimum (as measured by σ) the BFGS fails to compete with SPEA in reducing $\mathcal{D}$. In this synthetic analysis values of σ above 1.0 introduced divergence between the performance of the algorithms in Fig. 11. For comparison we used our original scheme to choose the starting parameters for real cells on this synthetic neuron. That yielded parameter values with a distance $\mathcal{D} = 9.21$ form the true values, considerably larger than the critical distance where the two algorithms perform similarly.

Since our SPEA algorithm took roughly 10 times the number of evaluations to find its solution we might equate the BFGS and SPEA by using 10 start points for the BFGS. The 9-dimensional space of this model means that adding only 10 start points would be unlikely to result in one being close enough to the global minimum to find it. The greater performance of the SPEA did not result from its longer computational time. Furthermore the SPEA algorithm did not need any human input in identifying good choices for starting parameters.

SPEA Fits as Initial Data for BFGS

Having established that SPEA typically outperforms BFGS, due to the problem of finding good starting conditions for the BFGS, it is natural to ask whether further improvement can be made by using the SPEA solution as initial data in a BFGS optimisation. In the previous section we noted that picking the initial condition with lowest SSQ from a population of 32 initial conditions does not give BFGS any clear advantage against SPEA. What we do know is that BFGS converges to a local minima while SPEA might pick points that are not close to a minima. This can be seen as a very elaborate way of picking initial data and we regard this as a type of memetic algorithm (MA) (Vrugt and Robinson 2007; Krasnogor and Smith 2008). The results of this are shown in Fig. 12. When picking the min(SSQ) point from the final Pareto front BFGS does not improve the SSQ very much at all. This indicates that SPEA is typically successful in settling close to a minimum and the use of a GF method does not improve upon that. If picking a point at random from the front then the SSQ is clearly improved upon by BFGS. In both choices of points from the front RCI is only slightly improved and R^2 is made considerably worse. This amounts to yet another evidence that the error space we are faced with is very flat with many local minima and a stochastic approach is more effective than a GF algorithm.

Discussion

We have fitted a multi-stage filter-based model to electrophysiological data from macaque V1, using both traditional GF methods (in particular the BFGS) and

evolutionary methods (SPEA). We have examined the computational cost of each method as well as the quality of solutions that they ultimately found for 107 neurons.

The BFGS certainly has the lowest computational cost of the methods used here; converging roughly an order of magnitude faster than the SPEA. In terms of fit quality, however, it was highly variable. For many cells, fits generated by this algorithm were very good and sometimes slightly better than the SPEA fits, according to the SSQ. For the vast majority of cells SPEA reached solutions with lower SSQ, the improvement was not always very large but in 90% of the cells SPEA was better. In a number of neurons the BFGS fit was extremely poor and in these cases it was dramatically outperformed by the SPEA. Probably these very poor fits were caused by the fact that we were unable, despite a great deal of effort, to provide sufficiently good initial parameters for the model. The quasi-quantative criteria we used to pick initial parameters worked reasonably well for some neurons but in the cases where it did not BFGS was simply unable to recover a reasonable solution. This again highlights the need for expert knowledge of the system and man-hours needed to get any success with GF fitting. With an evolutionary approach this is greatly reduced as we have demonstrated. At points very far from the global minimum, the SSQ is typically very insensitive to changes in parameters; if the model response is extremely poor then small changes to the parameters will not improve it. The TNC is designed to be less sensitive to starting parameters, but carries much higher computational cost. For the dataset and model tested here we found the TNC to take roughly 30 times longer than the BFGS for the subset of neurons on which we tested it. For the SPEA the computation time was slower by roughly a factor 10, but there was much greater consistency in finding solutions that we considered to be plausible.

A multi-start GF, in which several parallel GF searches are conducted from different start points, is one way to alleviate this problem. This has been shown to be superior to EAs in benchmark tests (Bolton et al. 2000) and comparable with EAs in certain real-world applications (Mendes and Kell 1998; Pettinen et al. 2006). This is surely dependent on the number of parameters required for the model being fitted. For a 9-parameter model, as we have here, to span the parameter space evenly, with only 2 points per parameter (which most likely would be insufficient to find the neighbourhood of the global minimum) we would need 2^9 start points, making the algorithm more than 500 times more costly than the original BFGS. That would make it much more expensive than the SPEA which does not require this large number of individuals to cover the parameter space adequately. The results from fitting on synthetic cells and the use of SPEA fits as initial data for BFGS also suggests that for this complex, high-dimensional system the result of a multi-start would be determined to a higher degree on how to pick the initial parameters than the BFGS performance.

To get around the problem that the SPEA is not theoretically guaranteed to converge to a minimum, one could use a memetic approach in which the SPEA is used to find starting parameters for the BFGS. We examined this for a subset of neurons tested here and found relatively modest improvements in terms of the SSQ and RCI, and these often actually decreased the fit quality as measured by the R^2.

An advantage of the SPEA is its ability to optimise multiple objectives simultaneously (MOO). Different error measures emphasise different aspects of fit quality with SSQ, for example, being biased towards overall amplitude of response and R^2 being biased towards the shape of the tuning curves. A method capable of MOO allows the fitting to search for solutions that optimises both shape and amplitude. Methods using single objective optimisation could use a compound error term, such as a weighted average of these errors, to try to optimise both, but would always reveal a single solution and would not care about the respective size of the individual terms. For example a low compound error term might be reached by having a very low SSQ or a very high R^2, or a moderately good fit in both terms. In reality there are likely to be an infinite number of solutions with identical compound errors, trading off one term against another. In SPEA a set of solutions that are, in some sense, equally good is represented by the Pareto front and this makes obvious to the user the reality that there are many good solutions, whereas single objective methods provide the illusion of there being a single 'best' solution to a given optimisation problem. That, of course, leaves the user with the task still of deciding which of the optimal set of solutions they wish to use or report.

The biggest advantage of the method has been the fact that there was a relatively small need for human intervention for each cell. For GF algorithms the need to manually tune initial parameters, or to create sophisticated algorithms to find such parameters, requires substantial man-hours and expert knowledge of the system being modeled. In our case, even with that effort and knowledge, the initial parameters were still often insufficiently close for our GF method to find the global minimum. In contrast, although the SPEA needed a little extra coding effort initially because EAs currently

need custom implementations for each problem. This effort was far outweighed by the much-reduced effort of determining initial parameters for the algorithm.

In this instance the SPEA has been extremely beneficial in fitting the data to our neuron model. We fully expect that the same would be true in numerous other applications of forward models of complex systems in neuroscience. For instance, EAs have already been used to fit conductance-based single neuron models to spiking data (Druckmann et al. 2007), and is likely also to be applicable to computational modeling of EEG, LFP, fMRI and MEG data.

Information Sharing Statement

The source code for the model is publicly available at http://code.google.com/p/findv1/. Redistribution and modification of the code is permitted under the terms of the GNU General Public License http://www.gnu.org/licenses/gpl.html For further information and availability on the electrophysiological data set contact Dr. Jonathan W. Peirce at jonathan.peirce@nottingham.ac.uk.

Acknowledgement This project is supported by Welcome Trust grant number 085444/Z/08/Z.

Appendix A: Stimuli

The stimulus is a drifting grating in a circular aperture described by

$$S(x, y, t) = \sin\left(\omega_{\text{stim}}\xi_{\text{stim}} + vt\right)\Theta\left(r - x^2 - y^2\right);$$
$$\xi_{\text{stim}} = x\cos\left(\theta_{\text{stim}}\right) + y\sin\left(\theta_{\text{stim}}\right), \tag{11}$$

where Θ is the Heaviside function, r is the radius of the aperture, v is the drift velocity of the grating and ω_{stim} is the spatial frequency of the stimulus. The variable ξ_{stim} is a direction orthogonal to the sinusoidal carrier.

Appendix B: Responses

We will work out the final response $Q(t)$ by successively introducing the filters and convolving them with the stimulus or previous responses. The CRF is represented by a Gabor filter

$$F_{\text{CRF}}(x, y) = \exp\left(-\frac{x^2 + y^2}{\sigma_c^2}\right)\sin\left(\omega_c\xi + \phi\right);$$
$$\xi = x\cos(\theta) + y\sin(\theta), \tag{12}$$

where θ is the orientation of the CRF, the width is controlled by σ_c while ω_c is the spatial frequency of the filter and ϕ is the phase of the carrier. The response resulting from this filter is then the spatial convolution of itself with the stimulus:

$$Q_{\text{CRF}}(x, y, t) = k_c\left[S \otimes F_{\text{CRF}}\right](x, y, t), \tag{13}$$

where k_c is the gain of the CRF. In the Watson–Ahumada spatio-temporal filter we also need the response from the Hilbert transform of the CRF. The Gabor function is symmetric except in the direction ξ, this means that the effect of the Hilbert transform is simply that it advances the phase of the carrier by $\pi/2$. This means that we will have

$$\mathcal{H}\left(F_{\text{CRF}}(x, y)\right) = \exp\left(-\frac{x^2 + y^2}{\sigma_c^2}\right) \times \sin\left(\omega_c\xi + \phi + \pi/2\right), \tag{14}$$

The response to the Hilbert transformed CRF is given by

$$Q_{\text{H-CRF}}(x, y, t) = k_c\left[S \otimes \mathcal{H}\left(F_{\text{CRF}}\right)\right](x, y, t). \tag{15}$$

Secondly we introduce the broadly tuned surround filter:

$$F_{\text{tuned}}(x, y) = \exp\left(-\frac{x^2 + y^2}{\sigma_s^2}\right)\sin(\rho\omega_c\xi + \phi). \tag{16}$$

The width is determined by σ_s and the spatial frequency of the carrier is slightly broader than that of the CRF (ω_c) as a result of the scaling factor ρ. As in the CRF ϕ is the phase of the carrier. The response of this filter is

$$Q_{\text{tuned}}(x, y, t) = k_s\left[S \otimes F_{\text{tuned}}\right](x, y, t), \tag{17}$$

where k_s is the gain of the untuned surround. In the same manner as for the CRF we also need the response from the Hilbert transform of the filter given by

$$Q_{\text{H-tuned}}(x, y, t) = k_s\left[S \otimes \mathcal{H}\left(F_{\text{tuned}}\right)\right](x, y, t). \tag{18}$$

The Hilbert transform effects the tuned suppressive filter in the same way as the CRF filter, see Eq. 14.

The last part of the spatial component emulates the effects of a contrast normalisation pool. To model the effects of this pool we multiply the envelope of the CRF

by the stimulus and then take the standard deviation (RMS contrast) of the resulting matrix:

$$Q_{\text{untuned}}(t) = \text{std}_{x,y}\left(\exp\left(-\frac{x^2+y^2}{\sigma_c^2}\right)S(x,y,t)\right), \quad (19)$$

where $\text{std}_{x,y}(X)$ is the spatial standard deviation (RMS contrast) of the object X. The Hilbert transform does not effect the filter so therefore we do not need to calculate this response specifically. The Gaussian mask that is applied to the stimulus has the same width as the CRF envelope. The responses from the three filters are then combined as

$$Q_{\text{spatial}}(x,y,t) = \frac{\lfloor Q_{\text{CRF}}(x,y,t)\rfloor_+^{n_s}}{\lfloor Q_{\text{tuned}}(x,y,t)\rfloor_+^{n_s}\lfloor Q_{\text{untuned}}(t)\rfloor_+^{n_g}}, \quad (20)$$

where $\lfloor Q_{\text{CRF}}(x,y,t)\rfloor_+$ is the rectified response of the CRF (see Eq. 13), $\lfloor Q_{\text{tuned}}(x,y,t)\rfloor_+$ and $\lfloor Q_{\text{untuned}}(t)\rfloor_+$ are the rectified responses from the tuned and untuned respectively (see Eqs. 17 and 19). The exponent n_s represents self excitation and n_g is the contrast normalisation pool exponent. The self excitation exponent is fixed at the value 2.0 while the normal pool exponent is a free parameter. For the Watson–Ahumada method we also need to calculate a response from the Hilbert-transformed version of the spatial filter, which is given by:

$$Q_{\text{H-spatial}}(x,y,t) = \frac{\lfloor Q_{\text{H-CRF}}(x,y,t)\rfloor_+^{n_s}}{\lfloor Q_{\text{H-tuned}}(x,y,t)\rfloor_+^{n_s}\lfloor Q_{\text{untuned}}(t)\rfloor_+^{n_g}}, \quad (21)$$

where $Q_{\text{H-CRF}}(x,y,t)$ and $Q_{\text{H-tuned}}(x,y,t)$ are the responses from Hilbert-transformed versions of the CRF filter and the surround supression (see Eqs. 15 and 18).

With the responses from the spatial filters, $Q_{\text{spatial}}(x,y,t)$ and $Q_{\text{H-spatial}}(x,y,t)$, we can create the full response following the Watson–Ahumada formalism. To do this we need the temporal filter, $f(t)$, with impulse response

$$f(t) = \eta\left[f_1(t) - \zeta f_2(t)\right], \quad (22)$$

$$f_i(t) = \frac{\Theta(t)}{\tau_i(n_i-1)!}(t/\tau_i)^{n_i-1}\text{e}^{-t/\tau_i}, \qquad i \in \{1,2\}. \quad (23)$$

We consider τ_1 and τ_2 to be free parameters while for the rest we have $\eta = 1$, $\zeta = 0.9$, $n_1 = 9$ and $n_2 = 10$.

All the components are temporally convolved and the spatial dimensions are integrated out:

$$Q(t) = \int_{\mathbb{R}^2}\left[R_{\text{spatial}} * f * \delta\right](x,y,t) + \left[R_{\text{H-spatial}} * \mathcal{H}(f) * \delta\right](x,y,t)\,\text{d}x\text{d}y. \quad (24)$$

References

Bolton, H. P. J., Schutte, J. F., & Groenwold, A. A. (2000). Muliple parallel local searches in global optimization. In *Recent advances in parallel virtual machine and message passing interface, Lecture notes in computer science*. New York: Springer.

Box, M. J. (1966). A comparison of several current optimization methods, and the use of transformations in constrained problems. *The Computer Journal, 9*(1), 67–77.

Broyden, C. G. (1967). Quasi-Newton methods and their application to function minimisation. *Mathematics of Computation, 21*(99), 368–381.

Carandini, M., & Heeger, D. J. (1994). Summation and division by neurons in primate visual cortex. *Science, 264*(5163), 1333–1336.

Carandini, M., Heeger, D. J., & Movshon, J. A. (1997). Linearity and normalization in simple cells of the macaque primary visual cortex. *The Journal of Neuroscience, 17*(21), 8621–8644.

Cavanaugh, J. R., Bair, W., & Movshon, J. A. (2002). Nature and interaction of signals from the receptive field center and surround in macaque v1 neurons. *Journal of Neurophysiology, 88*, 2530–2546.

Daugman, J. G. (1985). Uncertainty relation for resolution in space, spatial frequency, and orientation optimized by two-dimensional visual cortical filters. *Journal of the Optical Society of America A, 2*(7), 1160–1169.

Davison, A. P., Brüderle, D., Eppler, J., Kremkow, J., Muller, E., & Pecevski, D. (2009). PyNN: A common interface for neuronal network simulators. *Frontiers in Neuroinformatics, 2*, 1–10.

Deb, K. (1999). Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation, 7*(3), 205–230.

Dixon, L. C. W. (1972). Quasi-Newton algorithms generate identical points. *Mathematical Programming, 2*(1), 383–387.

Druckmann, S., Banitt, Y., Gidon, A. A., Schürmann, F., Markram, H., & Segev, I. (2007). A novel multiple objective optimization framework for constraining conductance-based neuron models by experimental data. *Frontiers in Neuroscience, 1*, 7–18.

Einevoll, G. T. (2009). Sharing with Python. *Frontiers in Neuroscience, 3*, 334–335.

Fiore, C. D., Fanell, S., Lepore, F., & Zellini, P. (2003). Matrix algebras in quasi-Newton methods for unconstrained minimization. *Numerische Mathematik, 94*(3), 479–500.

Goodman, D. F., & Brette, R. (2008). Brian: A simulator for spiking neural networks in Python. *Frontiers in Neuroinformatics, 2*(5). doi:10.3389/neuro.11.005.2008.

Hansen, N., & Ostermeier, A. (1997). Convergence properties of evolution strategies with the derandomized covariance matrix adaptation: The $(\mu/\mu_i,\lambda)$-CMA-ES. In *5th European congress on intelligent techniques and soft computing* (pp. 650–654).

Hansen, N., Ostermeier, A., & Gawelczyk, A. (1995). On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. In L. Eshelman (Ed.), *Proceedings of the sixth international conference on genetic algorithms* (pp. 57–64).

Heath, M. T. (2002). *Scientific computing: An introductory survey* (2nd ed.). New York: McGraw-Hill.

Heeger, D. J. (1992). Normalization of cell responses in cat striate cortex. *Visual Neuroscience, 9*, 181–198.

Howell, D. C. (2004). *Fundamental statistics for the behavioral sciences* (5th ed.). Pacific Grove: Brooks/Cole-Thomson.

Jones, E., Oliphant, T., & Peterson, P. (2001). SciPy: Open source scientific tools for Python.

Kent, J. T. (1983). Information gain and a general measure of correlation. *Biometrika, 70*(1), 163–173.

Khvedelidze, B. V. (2001). Hilbert transform. In M. Hazewinkel (Ed.), *Encyclopaedia of mathematics*. New York: Springer.

Kim, S., Kim, J., & Cho, K.-H. (2007). Inferring gene regulatory networks from temporal expression profiles under time-delay and noise. *Computational Biology and Chemistry, 31*(4), 239–245.

Krasnogor, N., & Smith, J. (2008). Memetic algorithms: The polynomial local search complexity theory perspective. *Journal of Mathematical Modelling and Algorithms, 7*, 3–24.

Lagarias, J. C., Reeds, J. A., Wright, M. H., & Wright, P. E. (1998). Convergence properties of the Nelder–Mead simplex method in low dimensions. *SIAM Journal on Optimization, 9*(1), 112–147.

Marcelja, S. (1980). Mathematical description of the responses of simple cortical cells. *Journal of the Optical Society of America A, 70*(11), 1297–1300.

Mendes, P., & Kell, D. B. (1998). Non-linear optimization of biochemical pathways: Applications to metabolic engineering and parameter estimation. *Bioinformatics, 14*(10), 869–883.

Nash, S. G. (1984). Solving nonlinear programming problems using truncated newton techniques. In P. T. Boggs, R. H. Byrd, & R. B. Schnabel (Eds.), *Numerical optimization* (pp. 119–136). Philadelphia: SIAM.

Nash, S. G., & Nocedal, J. (1991). A numerical study of the limited memory BFGS method and the truncated-Newton method for large scale optimization. *SIAM Journal on Optimization, 1*(3), 358–372.

Nash, S. G., & Sofer, A. (1996). *Linear and nonlinear programming*. New York: McGraw-Hill.

Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The Computer Journal, 7*, 308–313.

Nocedal, J. (1980). Updating quasi-Newton matrices with limited storage. *Mathematics of Computation, 35*(151), 773–782.

Pankratov, A. N., & Uchaeva, I. M. (2000). A semiempirical quantum chemical testing of thermodynamic and molecular properties of arsenic compounds. *Journal of Molecular Structure. Theochem, 498*(1–3), 247–254.

Peirce, J. W., Solomon, S. G., Forte, J. D., & Lennie, P. (2008). Cortical representation of color is binocular. *Journal of Vision, 8*(3), 1–10.

Pettinen, A., Yli-Harja, O., & Linne, M.-L. (2006). Comparison of automated parameter estimation methods for neuronal signaling networks. *Neurocomputing, 69*(10–12), 1371–1374.

Sbalzarini, I. F., Müller, S., & Koumoutsakos, P. (2000). Multiobjective optimization using evolutionary algorithms. In *Center for turbulence research proceedings of the summer program* (pp. 63–74).

Sceniak, M. P., Hawken, M. J., & Shapley, R. (2001). Visual spatial characterization of macaque v1 neurons. *Journal of Neurophysiology, 85*, 1873–1887.

Schlick, T., & Fogelson, A. (1992). TNPACK—a truncated Newton minimization package for large-scale problems: I. Algorithm and usage. *ACM Transactions on Mathematical Software (TOMS), 18*(1), 46–70.

Simoncelli, E. P., & Schwartz, O. (1999). Modeling surround suppression in V1 neurons with a statistically-derived normalization model. *Advances in Neural Information Processing Systems, 11*, 153–159.

Solomon, S. G., & Lennie, P. (2005). Chromatic gain controls in visual cortical neurons. *Journal of Neuroscience, 25*(19), 4779–4792.

Spacek, M., Blanche, T., & Swindale, N. (2009). Python for large-scale electrophysiology. *Frontiers in Neuroinformatics, 2*, 1–10.

Van Geit, W., De Schutter, E., & Achard, P. (2008). Automated neuron model optimization techniques: A review. *Biological Cybernetics, 99*(4), 241–251.

Vrugt, J. A., & Robinson, B. A. (2007). Improved evolutionary optimization from genetically adaptive multimethod search. *Proceedings of the National Academy of Sciences, 104*(3), 708–711.

Watson, A. B., & Ahumada, A. J. (1985). Model of human visual-motion sensing. *Journal of the Optical Society of America A, 2*(2), 322–342.

Webb, B. S., Dhruv, N. T., Solomon, S. G., Tailby, C., & Lennie, P. (2005). Early and late mechanisms of surround suppression in striate cortex of macaque. *The Journal of Neuroscience, 25*, 11666–11675.

Zhao, J., Buldum, A., Han, J., & Lu, J. P. (2002). Gas molecule adsorption in carbon nanotubes and nanotube bundles. *Nanotechnology, 13*(2), 195–200.

Zitzler, E., & Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength pareto evolutionary algorithm. *IEEE Transactions on Evolutionary Computation, 3*(4), 257–271.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., & Grunert da Fonseca, V. (2003). Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation, 7*(2), 117–132.