Working with the eDOG model: Stimulus optimization

Davide Crombie

January 28, 2020

1 Introduction

The extended Difference of Gaussians (eDOG) is a mathematical model designed to capture the stimulus responses of the dorsal lateral geniculate nucleus of the thalamus (dLGN) via linear filtering operations that reflect the corticothalamic circuit structure (Einevoll and Plesser, 2012; Mobarhan et al., 2018). Under this model, the spatiotemporal response of a homogeneous population of neurons is given by the convolution of the input to the neural population (the stimulus, S) with the impulse-response function of the population (G) in the time domain, or a multiplication of the two terms in the frequency domain.

$$R(\mathbf{r}, t) = G(\mathbf{r}, t) * S(\mathbf{r}, t) = \mathcal{F}^{-1}\left[\hat{G}\hat{S}\right] \tag{1}$$

When the input is dependent on the response of a different, connected neural population, the input term is given by the convolution of the upstream population's response with a coupling kernel (K). The model assumes that all such inputs sum linearly to produce a response.

$$\hat{R} = \sum_n \hat{K}_n \hat{R}_n$$

In the case of the dLGN relay neuron population, together with some simplifying assumptions that follow from experimental results, the response to a visual sitmulus is given by

$$\begin{aligned} \hat{R}_r &= \hat{K}_{rg}\hat{R}_g + \hat{K}_{rig}\hat{R}_g + \sum_n \hat{K}_{rc_n}\hat{K}_{c_n r}\hat{R}_r \\ &= \frac{\hat{K}_{rg} + \hat{K}_{ri}}{1 - \sum_n \hat{K}_{rc_n}\hat{K}_{c_n r}}\hat{G}_g\hat{S} \end{aligned} \tag{2}$$

This model can be extended by making changes either to the parameters of the coupling kernels, impulse-response kernels, and/or the structure of the impulse-response term in (2). One might note, however, that the linear filter approach allows for mathematically defined stimuli as well as naturalistic stimuli to be used as input, and it is likely that part of this arbitrarily large stimulus space yeilds almost indistinguishible responses for two different model instances.

> **Goal:** find a stimulus on an arbitrary parameter space which maximally separates the responses of two model configurations.

2 Response function gradient

Let the objective function R be defined as

$$R = (R_r^\alpha - R_r^\beta)^2 \tag{3}$$

Where R_r^α is the spatio-temporal response of the relay cell population in an eDOG model instance α. The full eDOG in (2) can be simplified back to the form of (1)

$$R_r^\alpha(\mathbf{r}, t, \mathbf{x}) = G_r^\alpha(\mathbf{r}, t) * S(\mathbf{r}, t, \mathbf{x})$$

$$\therefore \quad R(\mathbf{r},t,\mathbf{x}) = \mathcal{F}^{-1}\left[\left(\hat{G}^{\alpha}(\mathbf{k},\omega) - \hat{G}^{\beta}(\mathbf{k},\omega)\right)\hat{S}(\mathbf{k},\omega,\mathbf{x})\right]^2 \tag{4}$$

Evaluating the gradient of this function with respect to the stimulus parameters $\mathbf{x}$ — as is necessary for optimization — is intractable. Furthermore, this formulation of the response in terms of functions precludes the use of a "naturalistic stimulus", where the entries in $\mathbf{x}$ would refer to refer to the intensity values of pixels in an image. As in a digital image, impulse-response functions and mathematically-defined stimuli can be evaluated at a discrete set of points in space and time to yield arrays of intensity values. When all terms are treated as arrays in the frequency domain, the objective function becomes

$$\hat{\mathbf{R}}(\mathbf{x}) = \left((\hat{\mathbf{G}}^{\alpha} - \hat{\mathbf{G}}^{\beta}) \circ \mathbf{x}\right)^2 = \left(\hat{\mathbf{G}} \circ \mathbf{x}\right)^2 = \sum_n (\hat{g}_n x_n)^2 \tag{5}$$

where $\hat{\mathbf{R}}(\mathbf{x}) = \mathbf{R}(\mathbf{x})$ by Perseval's theorem. For simplicity, the the complex-valued entries in $\hat{\mathbf{G}}$ may be replaced by their magnitudes. The gradient and the Hessian are then given by

$$\frac{\partial \hat{\mathbf{R}}(\mathbf{x})}{\partial \mathbf{x}} = \begin{pmatrix} 2g_1^2 x_1 & 2g_2^2 x_2 & \cdots & 2g_n^2 x_n \end{pmatrix}^T \tag{6}$$

$$\frac{\partial^2 \hat{\mathbf{R}}(\mathbf{x})}{\partial \mathbf{x}^2} = \begin{pmatrix} 2g_1^2 & 0 & \cdots & 0 \\ 0 & 2g_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 2g_n^2 \end{pmatrix} \tag{7}$$

3 First-order constraints

The necessity for various constrains on the optimization problem arises from the above: it is clear from (5) that the objective function can be arbitrarily large, depending on the amplitude of the stimulus. Thus the stimulus should be normalized for a meaningful result.

The Lagrangian method for constrained optimization seeks to find the maximum value of a function $f(\mathbf{x})$ constrained on the manifold $\{\mathbf{x} \in \mathbb{R}^n \mid c(\mathbf{x}) = a\}$ departing from the observation that at the maximum of $f(\mathbf{x})$ on $c(\mathbf{x}) = a$ the gradients of $f(\mathbf{x})$ and $c(\mathbf{x})$ are paralell (Nocedal and Wright, 2006, ch.12).

$$\nabla f(\mathbf{x}) = \lambda \nabla c(\mathbf{x}), \quad \text{where } c(\mathbf{x}) = a$$

The Lagrangian function is thus constructed as

$$\mathcal{L}(\mathbf{x},\lambda) = f(\mathbf{x}) - \lambda(c(\mathbf{x}) - a) \tag{8}$$

Formally, the stimulus amplitude constraint set is given by the unit n-sphere (sometimes referred to as the L_2 norm of $\mathbf{x}$)

$$c(\mathbf{x}) = ||\mathbf{x}||_2 - 1 = \sum_n x_n^2 - 1 = 0 \tag{9}$$

So the Lagrangian objective function given (5) and (9) is

$$\mathcal{L}_R(\mathbf{x},\lambda) = \hat{\mathbf{R}}(\mathbf{x}) - \lambda c(\mathbf{x}) \tag{10}$$

The gradient is given by

$$\nabla\mathcal{L}(\mathbf{x},\boldsymbol{\lambda}) = \begin{pmatrix} \frac{\partial\mathcal{L}}{\partial\lambda} & \frac{\partial\mathcal{L}}{\partial\mathbf{x}_1} & \frac{\partial\mathcal{L}}{\partial\mathbf{x}_2} & \cdots & \frac{\partial\mathcal{L}}{\partial\mathbf{x}_n} \end{pmatrix}^T \tag{11}$$

$$\nabla\mathcal{L}_R(\mathbf{x},\boldsymbol{\lambda}) = \begin{pmatrix} c(\mathbf{x}) \\ \frac{\partial f(\mathbf{x})}{\partial x_1} - \lambda\frac{\partial c(\mathbf{x})}{\partial x_1} \\ \frac{\partial f(\mathbf{x})}{\partial x_2} - \lambda\frac{\partial c(\mathbf{x})}{\partial x_2} \\ \vdots \\ \frac{\partial f(\mathbf{x})}{\partial x_n} - \lambda\frac{\partial c(\mathbf{x})}{\partial x_n} \end{pmatrix} = \begin{pmatrix} \sum_n x_n^2 - 1 \\ 2g_1^2 x_1 - 2\lambda x_1 \\ 2g_2^2 x_2 - 2\lambda x_2 \\ \vdots \\ 2g_n^2 x_n - 2\lambda x_n \end{pmatrix}$$

The Lagrangian objective has stationary points at $\nabla\mathcal{L}(\mathbf{x},\boldsymbol{\lambda}) = \mathbf{0}$. The bottom n rows of this system of equations form the eigenvalue problem

$$\mathbf{I}\hat{\mathbf{G}}^2\mathbf{x} = \lambda\mathbf{x} \tag{12}$$

The trivial solution to this eigenvalue problem, $\mathbf{x} = \mathbf{0}$ does not satisfy $c(\mathbf{x}) = 0$, and hence is not a solution to the full $\nabla\mathcal{L}(\mathbf{x},\boldsymbol{\lambda}) = \mathbf{0}$ system. The non-trival solutions, given by the elements of $\hat{\mathbf{G}}^2$ and the corresponding standard basis vectors of $\mathbb{R}^n$ (e.g. $\lambda = g_1^2$ and $\mathbf{x} = (1, 0, ..., 0)$), are valid as they do satisfy the constraint. Thus the objective function $\hat{\mathbf{R}}$ has stationary points at values of $\mathbf{x}$ corresponding to pure spatiotemporal sine waves. It can be seen from (5) that the maximum of all the stationary points occurs at argmax($\hat{\mathbf{G}}$). The curvature at this stationary point, however, remains to be determined.

4 Second-order constraints

If the goal is to maximize the objective function, it must be ensured that the curvature of the function is negative at the valid stationary points. While the full bordered Hessian is given by

$$\mathbf{H}_{\mathcal{L}} = \begin{pmatrix} \frac{\partial^2\mathcal{L}}{\partial\lambda^2} & \frac{\partial^2\mathcal{L}}{\partial\lambda\mathbf{x}} \\ \frac{\partial^2\mathcal{L}}{\partial\mathbf{x}\lambda} & \frac{\partial^2\mathcal{L}}{\partial\mathbf{x}^2} \end{pmatrix} = \begin{pmatrix} 0 & \frac{\partial^2\mathcal{L}}{\partial\lambda\mathbf{x}} \\ \frac{\partial^2\mathcal{L}}{\partial\mathbf{x}\lambda} & \frac{\partial^2\mathcal{L}}{\partial\mathbf{x}^2} \end{pmatrix} \tag{13}$$

The relevant curvature is given more simply by

$$\mathbf{H}_{\mathcal{L}_R} = \begin{pmatrix} 2g_1^2 - 2\lambda & 0 & \cdots & 0 \\ 0 & 2g_2^2 - 2\lambda & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 2g_n^2 - 2\lambda \end{pmatrix}$$

The curvature of the constrained objective can be characterized by the eigenvalues of this Hessian, which are given by the elements of the diagonal. Replacing λ with g_*^2 (where $g_* = \max(\hat{\mathbf{G}})$), a set of eigenvlaues is obtained that are negative for all values corresponding to $g_i \leq g_*$ and zero for values corresponding to $g_i = g*$. Therefore $\mathcal{L}_R$ is maximized with respect to all stimulus dimensions except for $\mathbf{x}_*$, where there is no curvature.

> **Conclusion:** the stimulus which maximizes the difference between any two eDOG model instances is the spatiotemporal sine wave at which the difference in the value of the two impulse-response functions is greatest.

5 Temporal frequency preferences

Mobarhan et al. (2018) have shown that an eDOG network configuration where inhibitory cortical feedback is delayed compared to excitatory feedback shifts the temporal filtering properties of dLGN relay cell responses to drifting grating stimuli from low-pass (when feedback weights are 0) to band-pass (for increasing feedback weights). Let $\hat{R}_r^{\alpha}$ represent the response of a model instance with feedback present, and $\hat{R}_r^{\beta}$ represent the response of a model instance without feedback. The corresponding impulse-response functions are given by

$$\hat{G}_r^{\alpha} = \frac{\hat{K}_{rg} + \hat{K}_{rig}}{1 - \hat{K}_{rcr}^{ex} - \hat{K}_{rcr}^{in}} \hat{G}_g$$
$$\hat{G}_r^{\beta} = \left(\hat{K}_{rg} + \hat{K}_{rig}\right) \hat{G}_g$$

And the stimulus which maximally separates the two responses is the sine wave pointing in the direction of

$$\text{argmax}\left(\hat{\mathbf{G}}_r^{\alpha} - \hat{\mathbf{G}}_r^{\beta}\right)$$

[depictions of the impulse-response matrices, their difference, and the optimal stimulus will go here]

6 Response nonlinearities

[...]

In cases where the system of equations given by $\nabla\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = \mathbf{0}$ cannot be solved analytically, a numerical approach must be adopted for optimization.

7 Newton optimization

In order to find the roots of a function $f(x)$, it is observed that, for x sufficiently close, the root a line tangent to $f(x)$, $x^{(0)}$, is closer to the true root of f than x itself. This tangent line is defined as

$$g(x) = f'(x)(x^{(0)} - x) + f(x) \tag{14}$$

Setting

$$g(x) = 0 \quad \longrightarrow \quad x^{(0)} = x - \frac{f(x)}{f'(x)}$$

This process can be performed iteratively

$$x_{n+1} := x_n - \frac{f(x_n)}{f'(x_n)} \tag{15}$$

If $f(x)$ is itself the gradient of another function R, the resultant x^* will, if the algorithm has converged to a satifactory degree, represent a stationary point of R. In this manner, Newton's method for finding the roots of a function can be used in optimization problems.

For a multi-dimensional vector-valued function $\mathbf{f}(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}^n$, the tangent is defined by

$$\mathbf{g}(\mathbf{x}) = \mathbf{J}(\mathbf{x})(\mathbf{x}^{(0)} - \mathbf{x}) + \mathbf{f}(\mathbf{x}) \tag{16}$$

Where $\mathbf{J}$ is the Jacobian of $\mathbf{f}$

$$\mathbf{J} = \begin{pmatrix} \frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_1} & \frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial \mathbf{f}(\mathbf{x})}{\partial x_n} \end{pmatrix} = \begin{pmatrix} \frac{\partial f_1(\mathbf{x})}{\partial x_1} & \frac{\partial f_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial f_2(\mathbf{x})}{\partial x_1} & \frac{\partial f_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_2(\mathbf{x})}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n(\mathbf{x})}{\partial x_1} & \frac{\partial f_n(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f_n(\mathbf{x})}{\partial x_n} \end{pmatrix}$$

For the roots given by $\mathbf{f} = \mathbf{0}$, the update rule becomes

$$\mathbf{x}_{n+1} := \mathbf{x}_n - \mathbf{J}^{-1}(\mathbf{x}_n)\mathbf{f}(\mathbf{x}_n) \tag{17}$$

If the function $\mathbf{f}$ is itself the gradient of a function $R(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$, then the update rule becomes

$$\mathbf{x}_{n+1} := \mathbf{x}_n - \mathbf{H}_R^{-1}(\mathbf{x}_n)\nabla R(\mathbf{x}_n) \tag{18}$$

where ∇R is the gradient of R and $\mathbf{H}_R$ is the Hessian. It is described in the appendix how this process is equivalent to iteratively optimizing a direct second order approximation of the response function R.

8 Appendix

8.1 A note on the partial derivatives of scalar-valued functions

The following convention for the gradient (∇), Jacobian ($\mathbf{J}$), and Hessian($\mathbf{H}$) of a function $f : \mathbb{R}^n \to \mathbb{R}$ is adopted

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} := \quad \nabla f = \begin{pmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} & \frac{\partial f(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f(\mathbf{x})}{\partial x_n} \end{pmatrix}^T \tag{19}$$

$$\mathbf{J}_f = \begin{pmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} & \frac{\partial f(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial f(\mathbf{x})}{\partial x_n} \end{pmatrix} \tag{20}$$

$$\mathbf{H}_f = \frac{\partial}{\partial \mathbf{x}} \left(\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \right) = \mathbf{J}_f (\nabla f)^T \tag{21}$$

8.2 Identities for gradients of vectorized scalar forms

Given the scalar $f(\mathbf{x}) = \mathbf{a}^T\mathbf{x}$, where

$$\mathbf{x} = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix}^T \qquad \mathbf{a} = \begin{pmatrix} a_1 & a_2 & \cdots & a_n \end{pmatrix}^T$$

$f(\mathbf{x})$ expands to

$$f(\mathbf{x}) = a_1x_1 + a_2x_2 + ... + a_nx_n$$

And by (19)

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \begin{pmatrix} a_1 & a_2 & \cdots & a_n \end{pmatrix}^T = \mathbf{a}$$

Because $\mathbf{a}^T\mathbf{x} = \mathbf{x}^T\mathbf{a}$,

$$\frac{\partial \mathbf{a}^T\mathbf{x}}{\partial \mathbf{x}} = \frac{\partial \mathbf{x}^T\mathbf{a}}{\partial \mathbf{x}} = \mathbf{a} \tag{22}$$

Given the scalar $f(\mathbf{x}) = \mathbf{x}^T\mathbf{A}\mathbf{x}$, where

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \qquad \mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}$$

$f(\mathbf{x})$ expands to

$$\begin{aligned} f(\mathbf{x}) = & a_{11}x_1^2 + a_{12}x_2x_1 + ... + a_{1n}x_nx_1 + \\ & a_{21}x_1x_2 + a_{22}x_2^2 + ... + a_{2n}x_nx_2 + ... + \\ & a_{11}x_1x_n + a_{12}x_2x_n + ... + a_{nn}x_n^2 \end{aligned}$$

The partial derivative with respect to $\mathbf{x}$ is given by (19), where

$$\begin{aligned}
\tfrac{\partial f(\mathbf{x})}{\partial x_1} &= (a_{11}+a_{11})x_1 + (a_{12}+a_{21})x_2 + ... + (a_{1n}+a_{n1})x_n \\
\tfrac{\partial f(\mathbf{x})}{\partial x_2} &= (a_{21}+a_{12})x_1 + (a_{22}+a_{22})x_2 + ... + (a_{2n}+a_{n2})x_n \\
&\vdots \\
\tfrac{\partial f(\mathbf{x})}{\partial x_n} &= (a_{n1}+a_{1n})x_1 + (a_{n2}+a_{2n})x_2 + ... + (a_{nn}+a_{nn})x_n
\end{aligned}$$

In this form, it is clear that

$$\begin{aligned}
\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} &= \begin{pmatrix} (a_{11}+a_{11}) & (a_{12}+a_{21}) & \cdots & (a_{1n}+a_{n1}) \\ (a_{21}+a_{12}) & (a_{22}+a_{22}) & \cdots & (a_{2n}+a_{n2}) \\ \vdots & \vdots & \ddots & \vdots \\ (a_{n1}+a_{1n}) & (a_{n2}+a_{2n}) & \cdots & (a_{nn}+a_{nn}) \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \\
&= (\mathbf{A}+\mathbf{A}^T)\mathbf{x}
\end{aligned}$$

Thus

$$\frac{\partial \mathbf{x}^T\mathbf{A}\mathbf{x}}{\partial \mathbf{x}} = (\mathbf{A}+\mathbf{A}^T)\mathbf{x} \tag{23}$$

8.3 Equivalence of second-order approximation and Newton optimization methods

The second-order Taylor expansion of a twice-differentiable function f at a point $x \to a$ is given by

$$f(x) \approx f(a) + f'(a)(x-a) + \frac{1}{2}f''(a)(x-a)^2 \tag{24}$$

For $f : \mathbb{R}^2 \to \mathbb{R}$, this becomes

$$\begin{aligned}
f(x,y) \approx & f(a,b) + f_x(a,b)(x-a) + f_y(a,b)(y-b) + \\
& \frac{1}{2}f_{xx}(a,b)(x-a)^2 + \frac{1}{2}f_{xy}(a,b)(x-a)(y-b) + \frac{1}{2}f_{yy}(a,b)(y-a)^2
\end{aligned} \tag{25}$$

Vectorizing (25) and generalizing to n dimensions, $f : \mathbb{R}^n \to \mathbb{R}$ is approximated by

$$f(\mathbf{x}) \approx f(\mathbf{a}) + \mathbf{J}(\mathbf{a})(\mathbf{x}-\mathbf{a}) + \frac{1}{2}(\mathbf{x}-\mathbf{a})^T\mathbf{H}(\mathbf{a})(\mathbf{x}-\mathbf{a}) \tag{26}$$

Where $\mathbf{J}$ is the Jacobian of f, and $\mathbf{H}$ is the Hessian (which is assumed to be symmetrical by the Schwarz-Clairaut theorem). Expanding, and collecting all terms dependent on $\mathbf{x}$

$$f(\mathbf{x}) \approx \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} - \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{a} - \frac{1}{2}\mathbf{a}^T\mathbf{H}\mathbf{x} + \mathbf{J}\mathbf{x} + ...$$

In order to find stationary points, the derivative is taken with respect to $\mathbf{x}$. By (23) and (22) respectively, the partial derivatives of the first and fourth terms with respect to $\mathbf{x}$ become

$$\begin{aligned}
\frac{\partial \mathbf{x}^T\mathbf{H}\mathbf{x}}{\partial \mathbf{x}} &= (\mathbf{H}+\mathbf{H}^T)\mathbf{x} \\
\frac{\partial \mathbf{J}\mathbf{x}}{\partial \mathbf{x}} = \frac{\partial (\mathbf{J}^T)^T\mathbf{x}}{\partial \mathbf{x})} &= \mathbf{J}^T
\end{aligned}$$

Next, it is noted that

$$\mathbf{A}^T\mathbf{B}^T = (\mathbf{BA})^T$$
$$\mathbf{H} = \mathbf{H}^T$$

Thus, if $\mathbf{z} = \mathbf{Ha} \;\; \rightarrow \;\; \mathbf{a}^T\mathbf{H} = \mathbf{z}^T$, and

$$\frac{\partial \mathbf{z}^T\mathbf{x}}{\partial \mathbf{x}} = \frac{\partial \mathbf{x}^T\mathbf{z}}{\partial \mathbf{x}} = \mathbf{z} = \mathbf{Ha}$$

Finally, assembling the partial derivatives of all terms

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = \mathbf{Hx} - \mathbf{Ha} + \mathbf{J}^T \tag{27}$$

Setting

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} = 0 \quad \longrightarrow \quad \mathbf{x} = \mathbf{a} - \mathbf{H}^{-1}\mathbf{J}^T$$

Algorithmically,

$$\mathbf{x}_{n+1} := \mathbf{x}_n - \mathbf{H}^{-1}(\mathbf{x}_n)\mathbf{J}^T(\mathbf{x}_n) \tag{28}$$

References

Gaute T Einevoll and Hans E Plesser. Extended difference-of-gaussians model incorporating cortical feedback for relay cells in the lateral geniculate nucleus of cat. *Cognitive neurodynamics*, 6(4):307–324, 2012.

Milad Hobbi Mobarhan, Geir Halnes, Torkel Hafting, Marianne Fyhn, Gaute T Einevoll, et al. Firing-rate based network modeling of the dlgn circuit: Effects of cortical feedback on spatiotemporal response properties of relay cells. *PLoS computational biology*, 14(5):e1006156, 2018.

Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.