Title: kCSD-python, reliable current source density estimation with quality control

Short title: kCSD-python

Chaitanya Chintaluri[1,2,†], Marta Bejtka[1,†], Władysław Średniawa[1,3], Michał Czerwiński[1], Jakub M. Dzik[1], Joanna Jędrzejewska-Szmek[1], Daniel K. Wójcik[1,*]

1 Laboratory of Neuroinformatics, Nencki Institute of Experimental Biology of Polish Academy of Sciences, 3 Pasteur Str., 02-093 Warsaw, Poland
2 Institute of Science and Technology Austria, 3400 Klosterneuburg, Austria
3 University of Warsaw, Faculty of Biology, Department of Zoology, 1 Miecznikowa Str., 02-096 Warsaw, Poland

***Corresponding author, d.wojcik@nencki.edu.pl**
† joint first authors

August 6, 2023

Abstract

Interpretation of the extracellular recordings can be difficult due to the long range of electric field but can be facilitated by estimating the density of current sources (CSD). Here we introduce `kCSD-python`, an open Python package implementing Kernel Current Source Density (kCSD) method, and introduce several new techniques to facilitate CSD analysis of experimental data and interpretation of the results. We investigate the limitations imposed by noise and assumptions in the method itself. `kCSD-python` allows CSD estimation for arbitrary distribution of electrodes in 1D, 2D, and 3D, assuming distributions of sources in tissue, a slice, or in a single cell, and includes a range of diagnostic aids. We demonstrate its features in a Jupyter notebook tutorial to facilitate uptake by the community.

1 Introduction

Extracellular potential recordings are a mainstay of neurophysiology. However, the long range of electric field still makes their interpretation challenging despite decades of our experience. Extracellular potential in tissue is produced by transmembrane currents. Its low-frequency part, called the Local Field Potential (LFP), is believed to mainly reflect dendritic processing of synaptic inputs (Nunez and Srinivasan, 2006; Buzsáki, Anastassiou, and Koch, 2012). To facilitate understanding of the processes underlying the recorded signal it is useful to estimate the density of transmembrane current sources (Current Source Density, CSD) (Pitts, 1952; Nicholson and Freeman, 1975; Mitzdorf, 1985; Einevoll et al., 2013; Gratiy et al., 2017). CSD gives direct access to the physiologically relevant information, which is often concealed in original data (Mitzdorf, 1985). The relation between the CSD and the extracellular potential can be described by the Poisson equation

$$C = -\nabla(\sigma \nabla V), \tag{1}$$

where C is the CSD, V is the extracellular potential, and σ — the conductivity tensor. For isotropic and homogeneous tissue Eq. (1) reduces to

$$C = -\sigma \Delta V \tag{2}$$

which can be solved:

$$V(\mathbf{x}) = \frac{1}{4\pi\sigma} \int d\mathbf{x}' \frac{C(\mathbf{x}')}{|\mathbf{x} - \mathbf{x}'|}. \tag{3}$$

Eq. (2) and Eq. (3) show that knowing the potential in the whole space, we can compute the CSD, and knowing the CSD in the whole space, we can compute the potential. Experimentally, we can only access the potential at discrete electrode locations, so direct determination of the CSD in the whole space from Eq. (2) is impossible. To deal with this problem different methods for estimation of current sources have been proposed since the middle of the past century (Pitts, 1952; Nicholson and Freeman, 1975; Pettersen et al., 2006; Łęski, Wójcik, et al., 2007; Łęski, Pettersen, et al., 2011; Potworowski et al., 2012; Chintaluri et al., 2021; Klein et al., 2021). Here, we present `kCSD-python`, an open Python toolbox implementing the kernel Current Source Density (kCSD) method (Potworowski et al., 2012; Chintaluri et al., 2021). It allows kCSD reconstruction of current sources for data from 1D setups (laminar probes and equivalent electrode distributions), 2D (planar MEA, multishaft silicon probes, Neuropixel or SiNAPS probes, etc), and 3D electrode setups (Utah arrays, multiple electrodes placed independently in space with controlled positions), where the sources are assumed to come from tissue (kCSD) or from single cells with known morphology (skCSD, Cserpan et al., 2017). Fig. 1 shows the different experimental scenarios for which this software is applicable.

We also introduce some new diagnostic features which we found useful in CSD analysis and illustrate the application of the package and the new diagnostic tools in typical analysis workflow implemented as a Jupyter notebook (Kluyver et al., 2016) tutorial.

For reader convenience we first briefly restate the kCSD method (Potworowski et al., 2012; Chintaluri et al., 2021). Then in the Results section we introduce several new diagnostic tools included in the presented package. First, we added L-curve method for parameter selection. Then, to get informed on reconstruction accuracy we introduce *measurement uncertainty maps*, which show how measurement noise affects the estimated CSD, and *reliability maps*, which help build intuition on reliability of estimates for classes of possible sources for a given setup. We then briefly introduce the new package which is extensively illustrated in the provided tutorial in the Supplementary Materials. This jupyter notebook (Kluyver et al., 2016) tutorial, which is part of the toolbox, is provided to facilitate understanding and usage of the kCSD method. The tutorial enables the user to conduct analysis of the CSD using simulated surrogate data (electrodes positions and recordings) or actual recordings.

2 Methods

Kernel Current Source Density method uses kernel interpolation of measured potential to obtain V(x) in the whole space:

$$V*(\mathbf{x}) = \sum_{j=1}^{N} \beta_j K(\mathbf{x}, \mathbf{x}_j), \tag{4}$$

where $\mathbf{x}_j$, $j = 1, \ldots, N$, are electrode positions, $K(\mathbf{x}, \mathbf{x}')$ is a symmetric kernel function. To avoid overfitting, correction for noise is made by minimizing prediction error

$$\mathrm{err}[V] = \sum_{i=1}^{N} (V(\mathbf{x}_i) - V_i)^2 + \lambda|\beta|^2, \tag{5}$$

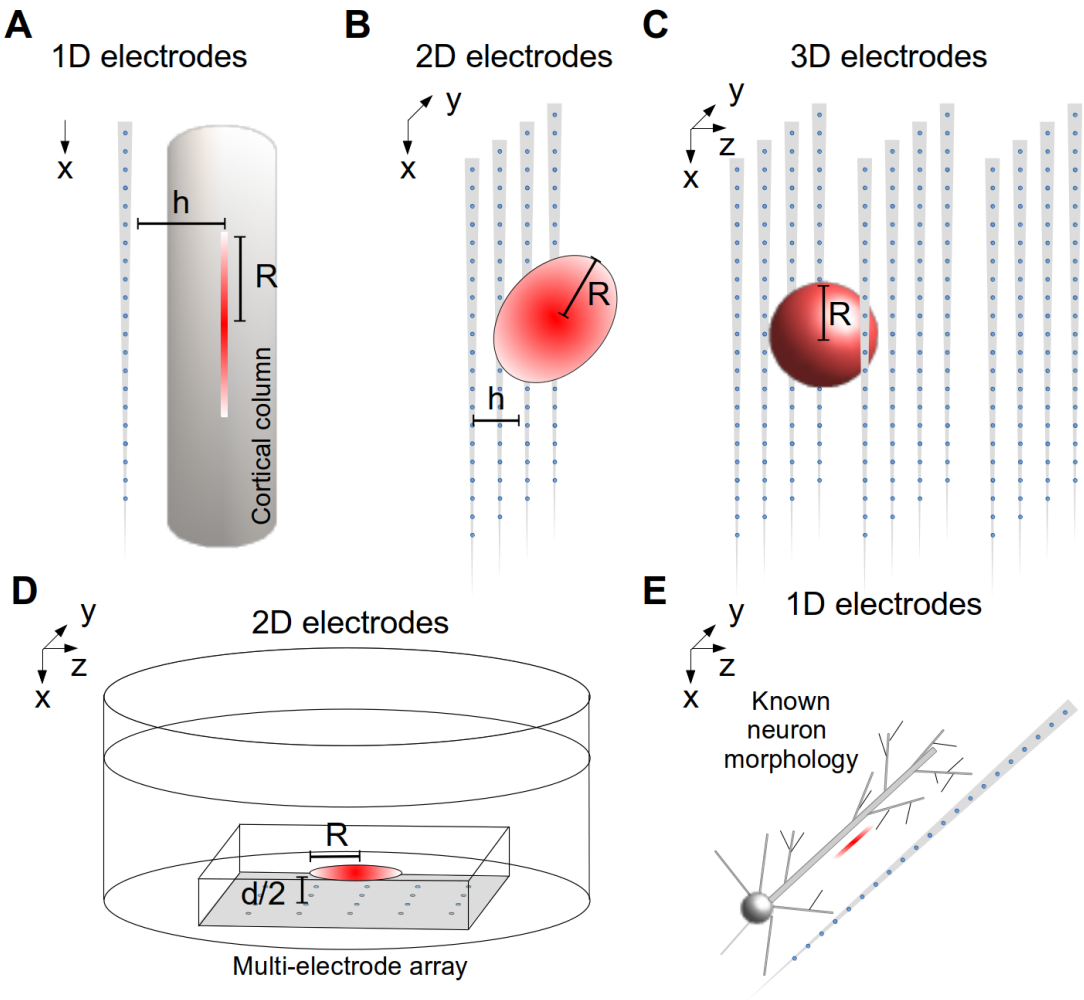

Figure 1: Overview of experimental contexts where `kCSD-python` is applicable. 1D setups such as A) laminar probes and equivalent; 2D setups, such as B) multishaft silicon probes, Neuropixel or SiNAPS probes, or D) planar MEA; 3D electrode setups, such as multiple multishaft silicon probes, Utah arrays, multiple electrodes placed independently in space with controlled positions, where the sources are assumed to come C) from tissue (kCSD) or E) from single cells with known morphology (skCSD). For description of parameters see Methods.

which gives

$$\beta = (\mathbf{K} + \lambda \mathbf{I})^{-1}\mathbf{V}, \tag{6}$$

where $\mathbf{V}$ is the vector of measured potentials, λ is regularization parameter, and

$$\mathbf{K}_{i,j} \equiv K(\mathbf{x}_i, \mathbf{x}_j). \tag{7}$$

In the second step, we use function $\widetilde{\mathbf{K}}(\mathbf{x}, \mathbf{x}')$, which we call cross-kernel, to estimate the CSD:

$$\mathbf{C}^* = \widetilde{\mathbf{K}}(\mathbf{K} + \lambda \mathbf{I})^{-1}\mathbf{V}. \tag{8}$$

This procedure would work for arbitrary smoothing kernels $\mathbf{K}$ but in general it is difficult to identify the relevant corresponding $\widetilde{\mathbf{K}}$, except in the simple cases. To circumvent this in kCSD we introduce a large basis of CSD sources spanning the region of interest, $\tilde{b}_j(\mathbf{x})$, think of many little Gaussians, and corresponding basis sources in the potential

space, $b_j(\mathbf{x})$, and construct both kernels from these basis sources (Potworowski et al., 2012; Chintaluri et al., 2021)

$$K(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{M} b_i(\mathbf{x}) b_i(\mathbf{x}'),\ \widetilde{K}(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{M} \tilde{b}_i(\mathbf{x}) b_i(\mathbf{y}). \tag{9}$$

In these basis the CSD and the potential are given by

$$C^*(\mathbf{x}) = \alpha_1 \tilde{b}_1(\mathbf{x}) + \cdots + \alpha_M \tilde{b}_M(\mathbf{x}), \tag{10}$$

$$V^*(\mathbf{x}) = \alpha_1 b_1(\mathbf{x}) + \cdots + \alpha_M b_M(\mathbf{x}). \tag{11}$$

The challenges of the method are how to construct $\mathbf{K}$ and $\widetilde{\mathbf{K}}$, how to select the basis, the relevant parameters, and reliability of the estimation, some of which were introduced before (Potworowski et al., 2012; Chintaluri et al., 2021) and some we introduce here and illustrate with the provided package. For more details of kCSD method and the theory behind, see (Potworowski et al., 2012; Chintaluri et al., 2021).

3 Results

This paper introduces the `kCSD-python` package, an implementation of the kernel Current Source Density method (Potworowski et al., 2012) and its two variants ((Ness et al., 2015) and (Cserpan et al., 2017)). It is open source and available under the modified BSD License (3-Clause BSD) on GitHub (`https://github.com/Neuroinflab/kCSD-python`). It utilizes the continuous integration provided by Travis CI. It supports Python 3.9 version[1] and has a bare minimal library requirements (`numpy`, `scipy` and `matplotlib`). It can be installed using the Python package installer (`pip`) or using Anaconda python package environment (`conda`). Details of the installation can be found in the package documentation at `https://kcsd-python.readthedocs.io/en/latest/INSTALL.html`.

The package contains a set of tools for kCSD analysis and to validate the results obtained from this analysis. To facilitate uptake of this resource, the package comes with extensive tutorials implemented in jupyter notebooks. These tutorials allow users to test different configurations of current sources and electrodes to see the method in action. The users can analyze their own data or explore the method with data generated *in silico*. These provisions illustrate the advantages and limitations of kCSD method to its users. The tutorials can also be accessed without any installation on a web browser via Binder (Project Jupyter et al., 2018). The package is extensively documented (`https://kcsd-python.readthedocs.io`) and includes all the necessary scripts to generate the figures in this manuscript.

An extensive tutorial overview of the `kCSD-python` package, is provided as an Appendix. Its goal is to show how to use `kCSD-python` to perform CSD analysis, how to apply the provided analysis tools, and to validate the results. We first consider a regular grid of ideal (noise-free) electrodes, where we compute the potentials from a known test source (the ground truth). We then use these potentials to reconstruct the sources which we compare with ground truth (Basic features). Then, we explore the effects of noise on the reconstruction and test the robustness of the method (Noisy electrodes). In the final part of the tutorial we look at how the errors in the estimation depend on the sources and

[1] supports also Python 2.7 version

the electrode configuration by testing the effects of broken electrodes on reconstruction (Broken electrodes).

Supplementary Fig. 11 shows error propagation maps, which we introduce below, for 1D regular grid of 12 electrodes. Supplementary Fig. 12 shows an example of 3D kCSD source reconstruction. Supplementary Fig. 13 shows an example of skCSD reconstruction (single cell kernel CSD) which corresponds to Fig. 8 from Cserpan et al., 2017. The simulation, reconstruction and visualization have all been re-implemented in Python.

We also provide some pre-computed examples of CSD estimations using our library for users to explore. We provide these as `.pdf` files available at `http://bit.ly/kCSD-supplementary`. Here, the files `small_srcs_3D_all.pdf` and `large_srcs_3D_all.pdf` show 100 example setups of 3D kCSD reconstructions from small and large sources. Similarly, files `small_srcs_all.pdf` and `large_srcs_all.pdf` show 100 example setups of 2D kCSD reconstructions from small and large sources. "Smallness" and "largeness" of sources is defined by the ratios of typical spatial scales of the source with respect to interelectrode distances.

3.1 Parameter selection

An important part of kCSD estimation is selection of parameters, in particular the regularization parameter, λ, but also the width of the basis source, R. Previously we proposed to use cross-validation (Potworowski et al., 2012). Here we introduce L-curve approach (P. Hansen, 2010; Kropf and Shmuel, 2016) for regularization. Both these methods are implemented in `kCSD-Python`.

Cross-validation To select parameters using cross-validation (Potworowski et al., 2012) we consider a range of parameter values, $\lambda \in [\lambda_0, \lambda_1]$. For any test value λ we select an electrode $i = 1, \ldots, N$ and ignore it. With eq. (4) we build a model from remaining measurements, $V_\lambda^{\prime i}(\mathbf{x})$, and use it to predict the value at the ignored electrode, $V_\lambda^{\prime i}(\mathbf{x}_i)$. Here

$$V_\lambda^{\prime i}(\mathbf{x}) = \sum_{j \neq i} \beta_{\lambda,j}^{\prime i} K(\mathbf{x}, \mathbf{x}_j), \tag{12}$$

where the minimizing vector

$$\beta_\lambda^{\prime i} = (\mathbf{K}^{\prime i} + \lambda \mathbf{I}^{\prime i})^{-1} \mathbf{V}_\lambda^{\prime i}, \tag{13}$$

and where $^{\prime i}$ means i-th column and row are removed from the given matrix (or vector). We repeat this for all the electrodes $i = 1, \ldots, N$ and compare predictions from the remaining electrodes against actual measurements:

$$\text{prediction error}(\lambda) = \sqrt{\sum_{i=1}^{N} (V_\lambda^{\prime i}(\mathbf{x}_i) - V_i)^2}. \tag{14}$$

For final analysis, λ giving minimum prediction error is selected. It is worth checking if the global minimum is also a local minimum. If the λ selected is one of the limiting values this may indicate that extending the range of λ might result is more optimal result or that the problem is ill-conditioned, for example too noisy, and we are either underfitting or overfitting, as we discuss below for the L-curve. As a rule of thumb, the range of tested λ parameters should cover eigenvalues of the (7).

L-curve Consider the error function, Eq. (5), which we minimize to get the regularized solution, $V_\lambda = \mathbf{K}\beta_\lambda$ It is a sum of two terms we are simultaneously minimizing, prediction error

$$\varrho_\lambda = \sum_{i=1}^{N} (V_\lambda(\mathbf{x}_i) - V_i)^2, \tag{15}$$

and the norm of the model

$$\eta_\lambda = \|V_\lambda(\mathbf{x})\|^2_{\mathcal{F}} = |\beta_\lambda^T \mathbf{K} \beta_\lambda|, \tag{16}$$

weighted with λ. Taking $\lambda = 0$ is equivalent to assuming noise-free data. In this case we are fitting the model to the data, in practice, overfitting. On the other hand, taking large λ means assuming very noisy data, in practice ignoring measurements, which results in a flat underfitted solution. Between these extremes there is usually a solution such that if we decrease λ, the prediction error, ϱ, slightly decreases, while the norm of the model, η, increases fast, and if we increase λ, the prediction error, ϱ, increases fast, while the norm of the model, η, slightly decreases; see Fig. 2D.

This is apparent when the prediction error and the norm of the model are plotted in the log-log scale. This curve follows the shape of the letter L, hence the name L-curve (P. C. Hansen, 1992). Several methods have been proposed to measure the curvature of the L-curve and to identify optimal parameters (P. C. Hansen, Jensen, and Rodriguez, 2007). In `kCSD-python`, we have implemented the triangle area method proposed by Castellanos, Gómez, and Guerra, 2002. To distinguish between convex and concave plot, clockwise directed triangle area is measured as negative Fig. 2.D shows this estimated curvature for our example as a function of λ.

To illustrate this method in the context of CSD reconstructions, we study an example of 1D dipolar current source with a split negative pole (sink; see Fig. 2.C, True CSD, red dashed line). We compute the potential at 32 electrodes (Fig. 2.A, C, black dots) with additive noise at every electrode. Notice that if we want to interpret the recorded potential directly (Fig. 2.A, red dots) it is difficult to discern the split sink. Fig. 2.D shows the estimated curvature for our example as a function of λ. The optimal value of λ is found by maximizing the curvature of the log-log plot of η versus ϱ, Fig. 2.B. The red dot in Fig. 2.B, D, indicates the ideal λ parameter for this setup obtained through the L-curve method.

3.2 Reconstruction accuracy

With kCSD procedure one can easily estimate optimal CSD consistent with the obtained data. However, so far we have not discussed estimation of errors on the reconstruction. Since the errors may be due to a number of factors — the procedure itself, measurement noise, incorrect assumptions — one may approach this challenge in different ways.

First, to understand the effects of the selected basis sources and setup, one may consider the estimation operator $\widetilde{\mathbf{K}}(\mathbf{K}+\lambda)^{-1}$ and the space of solutions it spans. This space is given by the eigensources, introduced and described thoroughly in (Chintaluri et al., 2021). The orthogonal complement of this space in the original estimation space, spanned by $\tilde{b}_j(\mathbf{x})$ basis functions, is not accessible to the kCSD method. The study of eigensources facilitates understanding which CSD features can be reconstructed and which are inaccessible.

Second, to consider the impact of the measurement noise on the reconstruction, for any specific recording consider the following model-based procedure. Reconstruct CSD from

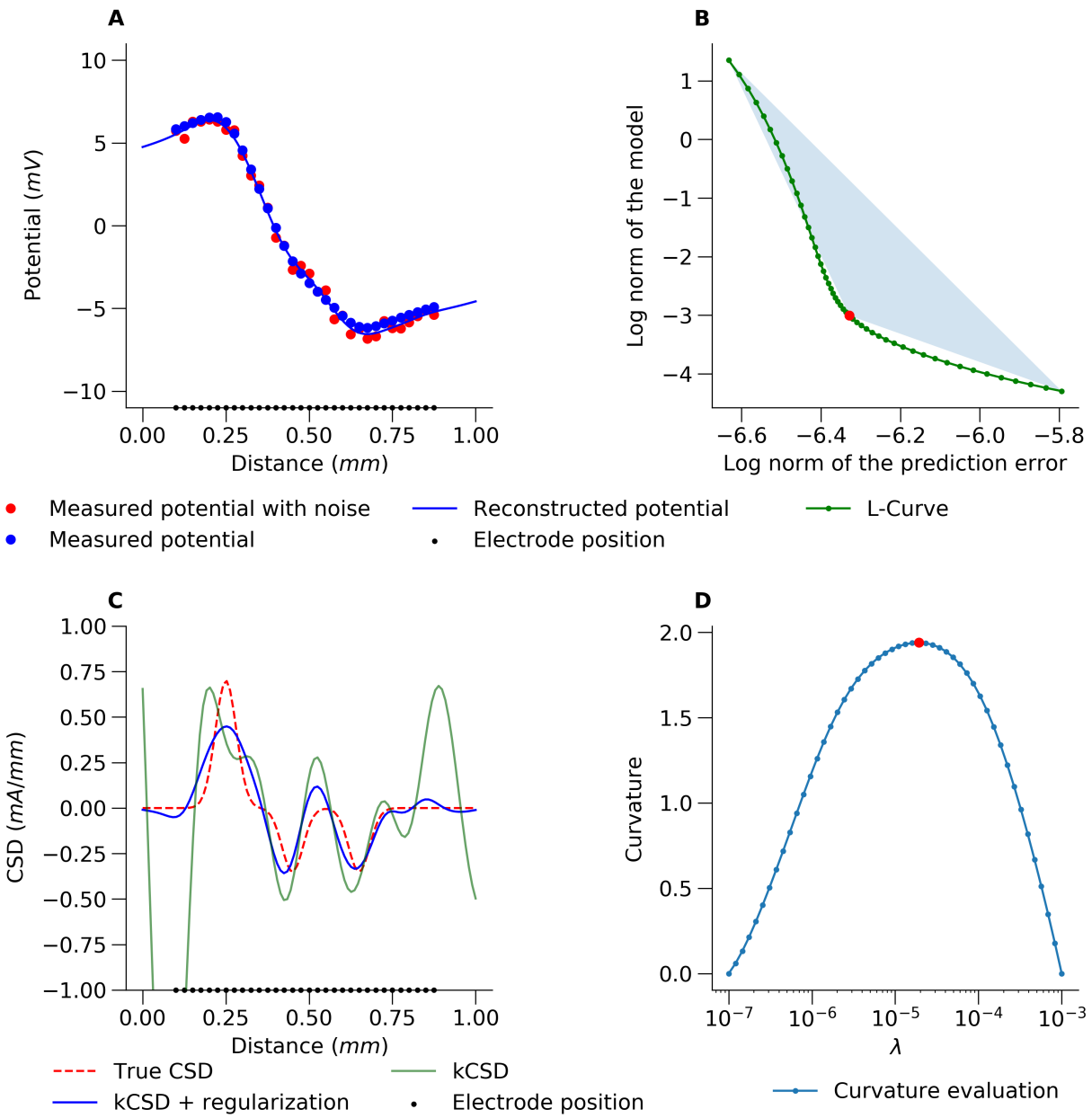

Figure 2: An example of the L-curve method for estimating kCSD parameters. A) The red points represent the potential used for CSD reconstruction. The black points show the electrode positions. The ground truth is shown in panel C with the red dashed curve. The measurement was simulated by adding small random noise to all the electrodes (32 values taken from a uniform distribution). The blue line shows a kernel interpolation of the potential which is the first step of kCSD method. B) L-curve plot for a single R parameter. The apex of the L-curve is numerically computed from the oriented area of directed triangles connecting the point on the L-curve with its two ends. C) Comparison of the true CSD and kCSD reconstruction for parameters obtained with L-curve regularization. D) Estimation of L-curve curvature with triangle method (see the Methods).

data with optimal parameters. Compute potential from estimated CSD. Add random noise to each computed potential. The noise could be estimated from data, either as a measure of fluctuations on a given electrode for a running signal, or from variability of

evoked potentials. Then, for any realization of noise, compute estimation of CSD. The pool of estimated CSD gives estimation of the error at any given point where estimation is made.

This computation can be much simplified by taking advantage of the linearity of the resolvent, $\mathbf{E} = \widetilde{\mathbf{K}}(\mathbf{K}+\lambda\mathbf{I})^{-1}$. Then, the i-th column ($\mathbf{E}_i$) represents contribution of unitary change of i-th measured potential (the i-th element of the vector $\mathbf{V}$) to the estimated CSD ($\underline{\mathbf{C}}^*$). As the contribution is proportional to the change, the column can be considered an *Error Propagation Map* for i-th measurement (Fig. 3.A). Note that these vectors (the columns of resolvent, $\mathbf{E}_i$) also happen to form another basis of the solution space, an alternative to the basis of eigensources.

If ε_i is an error of i-th measurement, then its contribution to $\underline{\mathbf{C}}^*$ is $\varepsilon_i\mathbf{E}_i$. Moreover, if the measurement errors follow multivariate normal

$$\varepsilon \sim \mathcal{N}(0, \Sigma_V), \tag{17}$$

then

$$\mathbf{V} \sim \mathcal{N}(\mathbf{V}_{exact}, \Sigma_V), \tag{18}$$

and the estimated CSD also follows multivariate normal

$$\underline{\mathbf{C}}^* \sim \mathcal{N}(\mathbf{E}\mathbf{V}_{exact}, \mathbf{E}\Sigma_V\mathbf{E}^T). \tag{19}$$

The diagonal of $\mathbf{E}\Sigma_V\mathbf{E}^T$ represents a *map of CSD measurement uncertainty* (uncertainty attributed to the noise in the measurement, Fig. 3.B)[2].

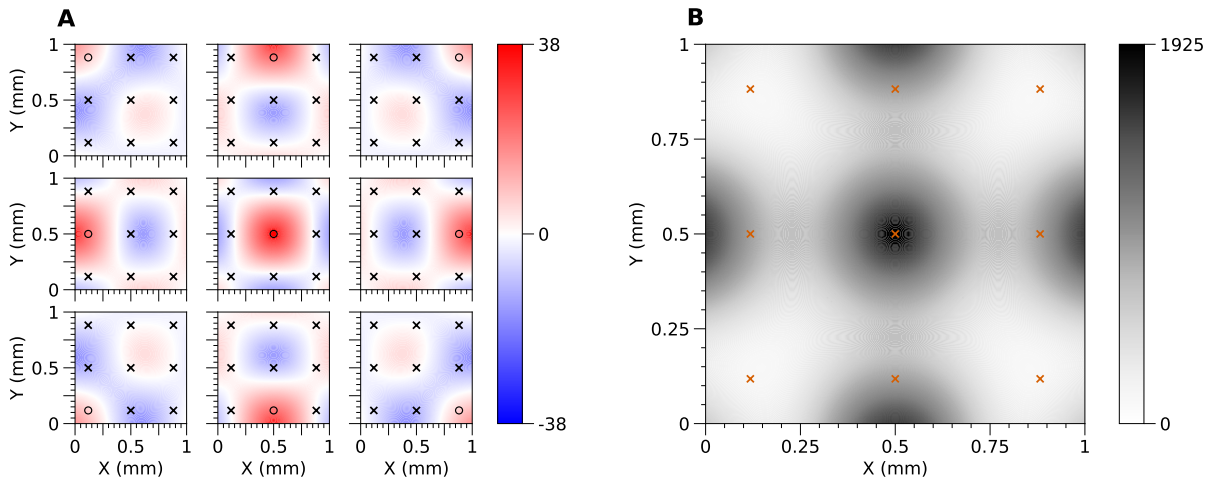

Figure 3: A) Error propagation maps for 3×3 regular grid of electrodes. Every panel represents the contribution of the potential measured at the corresponding electrode marked with a black circle ($\circ$) to the reconstructed CSD. Every other electrode is marked with a black cross ($\times$). B) Map of CSD measurement uncertainty for 3×3 regular grid of electrodes. The CSD measurement uncertainty is represented by variance of the CSD reconstruction caused by the uncertainty in measurement of the potentials. It is assumed that measurement errors for electrodes are mutually independent and follow standard normal distribution ($\varepsilon_i \sim \mathcal{N}(0, 1)$). Location of electrodes is marked with red crosses ($\times$).

[2]In the special case when ε_i are mutually independent and of equal variance σ^2, the map of CSD measurement uncertainty can be calculated as a diagonal of $\mathrm{Cov}[\underline{\mathbf{C}}^*] = \mathbf{E}\mathbf{E}^T\sigma^2$.

Third, one can study reconstruction accuracy for a meaningful family of test functions. This could be the Fourier modes for rectangular regions or a collection of Gaussian test functions, centered in different places, of single or multiple radii. For each of these test functions one would compute the potential, perform reconstruction, and compare the results with the original at every point. Finally, one could average this information over multiple different test sources computing a single *Reliability Map*, which we now introduce.

Reliability maps Assume the standard kCSD setup, that is a region $\mathcal{R} \subset \mathbb{R}^n$ where we want to estimate the sources, set of electrode positions, $\mathbf{x}_i$, and perhaps additional information, such as morphology for skCSD Cserpan et al., 2017. We now want to characterize predictive power of the combination of our setup and our selected basis, $\tilde{b}_i$. To do this we select a family of test functions, $C^i(\mathbf{x})$, for example Gaussian test functions, centered in different places, of multiple radii, or products of Fourier modes, etc. Then, for each C^i we compute $V^i = \mathcal{A}C^i$ by forward modeling, generating a surrogate dataset. Next, we apply the standard kCSD reconstruction procedure obtaining estimation of the tested ground truth, $\tilde{C}^i$. We can then compute reconstruction error using point-wise modification of Relative Difference Measure (RDM) proposed by (Meijs et al., 1988):

$$\mathrm{err}_i(\mathbf{x}) = \left| \frac{\tilde{C}^i(\mathbf{x})}{\|\tilde{C}^i\|} - \frac{C^i(\mathbf{x})}{\|C^i\|} \right| * \frac{\|C^i\|}{\max_{\mathbf{x}\in\mathcal{R}} |C^i|}, \tag{20}$$

where $i = 1, 2, \ldots$ enumerates different ground truth profiles. A simple measure of reconstruction accuracy is then given by the average over these profiles:

$$\mathrm{Reliability}(\mathbf{x}) := \langle \mathrm{err}_i(\mathbf{x}) \rangle_{i\in[1,M]} = \frac{\sum_{i=1}^{M} \mathrm{err}_i(\mathbf{x})}{M}. \tag{21}$$

Fig. 4 shows example reliability map for the case of 10x10 electrode distribution.

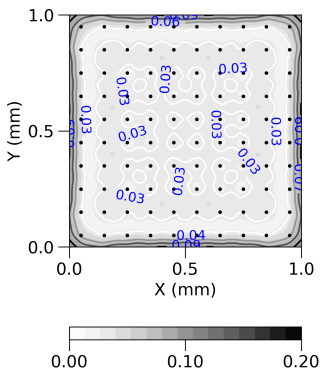

Figure 4: Reliability map created according to formula (20) and (21) for 10x10 regular grid of electrodes with noise-free symmetrized data. Black dots represent locations of contacts used in the study. Values on the map can be interpreted as follows: the closer to 0, the higher reconstruction accuracy might be achieved for a given measurement condition.

The class of functions used were the families of small and large sources mentioned above. We used eight mirror symmetries of the grid in computation.

We can use reliability map as another source of information about the precision of reconstruction, which is shown in Fig. 5. In A) we show some dipolar source which

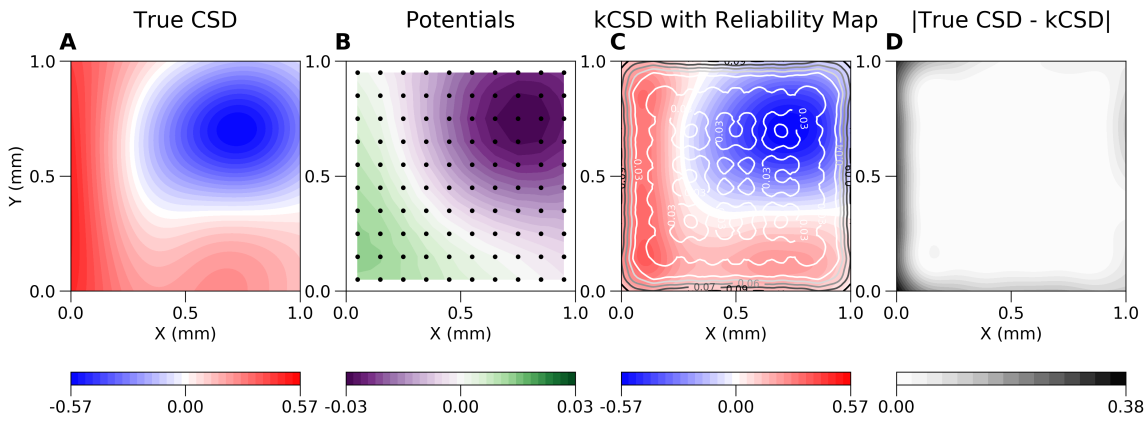

Figure 5: Example use of reliability maps. A) Example dipolar source (ground truth) which is used to compute the potential on a grid of electrodes shown in B). C) shows reconstructed sources superimposed on reliability map. D) shows the difference between the ground truth and the reconstruction.

is used to compute the potential on a grid of electrodes shown in B). Fig. 5.C) shows reconstructed sources superimposed on reliability map. Panel D) shows the difference between the ground truth and reconstruction. Note that plots such as shown in panel A) and D) are feasible only for simulated or model data, where we know actual sources and use them to validate the method. On the other hand, plots shown in panel B and C represent what can be routinely computed for experimental data. This functionality is implemented in `kCSD-python` and illustrated in the provided tutorial.

Another interesting question is the effect of broken or missing electrodes on the reconstruction. Formally one can attempt kCSD reconstruction from a single signal but it is naive to expect much insight this way. It is thus natural to ask what information can be obtained from a given setup and what we lose when part of it becomes inaccessible.

Fig. 6 shows the effect of removing electrodes on the reconstruction. Fig. 6.A shows average error of kCSD method across many random ground truth sources for a regular grid of 10x10 electrodes. Fig. 6.B to D show the increase of average reconstruction error as we remove 5 (B), 10 (C) and 20 (D) contacts. To emphasize the errors we show the difference between the reliability map for the broken grid minus the original one. Note the different scales in plots B–D versus A The consecutive rows show similar results when only small sources were used (E–H), or only large sources were used (I–L). Random sources in Fig. 6.A are both small and large sources (mentioned in the Results). This shows, among others, as we explained, that the reliability maps depend on the test function space, however, we feel they are more intuitive to understand than the individual eigensources spanning the solution space (Chintaluri et al., 2021).

4 Discussion

In the present work we introduced a new Python package for kCSD estimation, `kCSD-python`. This toolbox implements the basic functionality of source estimation with regularization. We introduced here L-curve approach as an approach to parameter selection alternative to cross-validation. This might be particularly important if regularization is attempted for high-density setups, such as multi-electrode arrays with thousands of channels. We also proposed several ways of diagnosing reconstruction errors, specifically Error Propagation

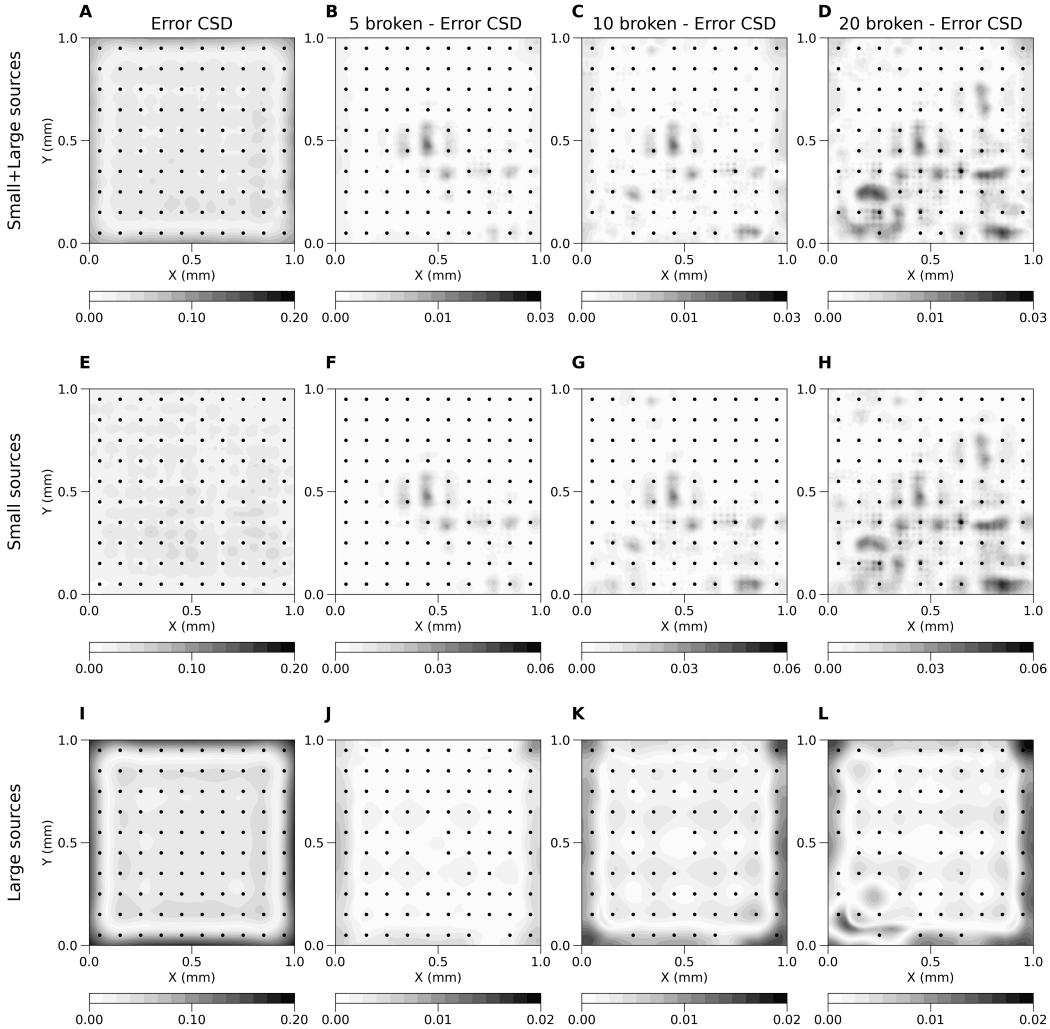

Figure 6: Average error (eq. 20) of kCSD method across random small and large (A), only small (E) and only large (I) sources for regular 10x10 electrodes grid and the same grid with broken 5 (B, F, J), 10 (C, G, K) and 20 (D, H, L) contacts. Plots (B, C, D, F, G, H, J, K, L) show difference between average error for regular grid and grid with broken contacts. Estimation was made in noise free scenario, R parameter selected in cross-validation. Black dots represent locations of contacts used in the study.

Maps and Reliability Maps, which can build intuition regarding veracity of reconstructions obtained in given experimental contexts. Finally, we provided a brief tutorial to the new package (see Supplementary Material) illustrating its main functions. All the figures in this paper showing CSD and LFP were computed with this package and source files are provided. In this section we discuss several issues related to CSD analysis in general and kCSD in particular.

The benefits of CSD estimation over direct LFP analysis. Extracellular potentials provide valuable insight about the functioning of the brain. Thanks to recent advances in multielectrode design and growing availability of sophisticated recording probes we can monitor the electric fields in the nervous system at thousands of sites, often simultane-

ously. One may wonder if this increased resolution makes CSD analysis unnecessary. In our view, as we have discussed many times, it is not so. The long range nature of electric potential means that even a single source contributes to every recording. Thus in principle we should always expect strong correlation between neighboring sites. However, if the separation between the electrodes becomes substantial, on the order of millimeters, the level of correlation between the recordings on different electrodes will decrease. This is because each electrode effectively picks up signals from a different composition of sources. Even if some are shared they are overshadowed by others which may lead to small interelectrode correlations. Still, our experience shows that significant correlations can be observed in the range of several millimeters (Łęski, Wójcik, et al., 2007; Katzner et al., 2009; Hunt et al., 2011; Kajikawa and Schroeder, 2011) which is consistent with computational models (Lindén, Tetzlaff, et al., 2011; Łęski, Lindén, et al., 2013). Fundamentally, the LFP profile is different from CSD profile, and may significantly distort or hide features of importance. For example, Fig. 2.C shows the source composed of three gaussians, while direct inspection of LFP suggests a simple dipole.

In view of these facts we argue that it is always beneficial to attempt kCSD analysis. The caveat is not to believe the reconstructed CSD blindly but always interpret it against known anatomical and physiological knowledge supported by the tools such as provided in the present work (eigensources, reliability maps, etc). This could be combined with other decomposition methods if desired which may give more physiologically interpretable results, in particular we have had very good results using independent component analysis, which we recommend (Łęski, Kublik, et al., 2010; Głąbska et al., 2014).

Approaches to CSD estimation. Several procedures for CSD estimation were introduced over the years and are still in use. The first approach, which probably still dominates today, was introduced by Walter Pitts in 1952 (Pitts, 1952) and gained popularity after Nicholson and Freeman adopted it for laminar recordings (Nicholson and Freeman, 1975). This was a direct numerical approximation to computation of the second derivative in the Poisson equation (1). Only minor improvements were introduced over the years to stabilize estimation (Rappelsberger, Pockberger, and Petsche, 1981) or handle boundaries (Vaknin, DiScenna, and Teyler, 1988). The first major conceptual change was introduced by Pettersen et al., 2006 who introduced model-based estimation of the sources. Their idea was to assume a parametric model of sources, for example, spline interpolated CSD between electrode positions, and using forward modeling to connect measured potentials to model parameters. This model-based approach was generalized by Potworowski et al., 2012 who proposed a non-parametric kernel Current Source Density method which is implemented in the presented toolbox.

Apart from these main approaches several CSD variant methods were proposed. For example, one may interpolate the potential first before applying traditional CSD approach, or the opposite, interpolate traditionally estimated CSD. Although in some cases the obtained results may look close to those obtained with kCSD, we do prefer kernel CSD approach due to the underlying theory which facilitates computation of estimation errors but also yields a unified framework for handling underlying assumptions, noisy data and irregular electrode distributions. In our view approaches combining ad hoc interpolation with numerical derivatives conceptually and computationally are less convincing to iCSD and kCSD and we would not recommend them.

Models of tissue. Throughout this work and in the toolbox we assumed purely ohmic character of the tissue. This has been debated in recent years (Bédard and Destexhe, 2011; Riera et al., 2012; Gratiy et al., 2017) and it is true that more complex biophysical models of the tissue, taking into account frequency dependent conductivity or diffusion currents, would influence the practice of source reconstruction or its interpretation. However, the available data indicate that in the range of frequencies of physiological interest these effects are small. While one should keep eyes open on the new data as they become available and keep in mind the different possible sources which may affect the reconstruction or interpretation, we believe that the traditional view of ohmic tissue is an adequate basis for typical experimental situations and going beyond that would probably require additional dedicated measurement for the experiment at hand which may not always be feasible. For example, as we discussed in (Ness et al., 2015), the specimen variability of the cortical conductivity in the rat is much bigger than the variability between different directions within a given rat (Goto et al., 2010). This means that unless we have conductivity measurements for our specific rat we are probably making smaller error assuming isotropic conductivity than taking different values from literature. We feel there is not enough data to justify inclusion of more complex terms in the standard CSD analysis to be applied throughout the brains and species.

In this manuscript and in the `kCSD-python` toolbox we also assumed constant conductivity (with exception of MoI case below). We are convinced this is a reasonable approximation for typical depth recordings. In general, however, this approximation needs to be justified or alternative models of tissue need to be considered. In principle, the kCSD method can be applied for a variety of tissue models as long as the basis potentials can be computed from the basis sources while incorporating the geometric and conductivity changes.

For example, Ness et al., 2015 considered a cortical slice placed on a microelectrode array (MEA) in which they included the geometry of the slice and modeled saline-slice interface with changing conductivity in the forward model. They found that Method of Images (MoI) gives a good approximation to the full solution obtained using finite-element model (FEM). This approximation was incorporated within the kCSD method as MoIkCSD variant and is available in the `kCSD-python` package.

It is possible to generalize kCSD to reconstruct sources from recordings of multiple electrical modalities — LFP, ECoG, EEG. In this case one needs to include the head geometry and the changing tissue properties within the forward model and in the kCSD method. The anisotropic (white matter tracts) and inhomogeneous (varying between skull, cerebro-spinal fluid, gray matter and white matter) electrical conductivity changes can be approximated using data obtained with imaging techniques such as MRI, CT or DTI. Such sophisticated head models require numerical solutions such as finite element modeling (FEM) to compute the basis potentials from the basis sources. We are currently working on this approach to make it generic for any animal head and to eventually utilize it as a source localization method for human data, for example, to localize foci of pharmacologically intractable epilepsy seizures in humans. We call this extension kernel Electrical Source Imaging (kESI).

High density microelectrode recordings. One of the trends clearly observed in modern neurotechnology is the drive towards increasing the number of sensors and their density (Buzsáki, 2004; Berdondini et al., 2005; Frey et al., 2009; Hottowy et al., 2012; Jun et al., 2017; Angotzi et al., 2019), for *in vitro* and *in vivo* studies. While it seems that a

better resolution for recording spiking activity of multiple cells is the main goal, also more precise stimulation and field potentials monitoring are targeted (Hottowy et al., 2012; Ferrea et al., 2012; Bakkum et al., 2013). Such massive high density data from thousands of electrodes should greatly increase insight into the studied systems and significantly improve results of CSD reconstructions. There are two obstacles to fully benefit from kCSD analysis of data from these new systems. First, kCSD involves inversion of the kernel matrix which is quadratic in the number of electrodes. Combined with cross-validation the necessary matrix operations quickly become overwhelming. This can be mitigated in a number of ways, by subsampling the data, approximate inversions, and by switching from cross-validation to L-curve method, but the challenge remains. This is the easy problem. The difficult problem is physical. As we move away from a source its contribution to the recorded potential goes down. In consequence, since the present version of kCSD uses all recordings to estimate every source, when using remote signals to estimate local source, we obtain mainly contributions from noise. In effect we get a very reliable estimation of sources varying slowly in space but the sources changing fast in space are treated as noise and silenced by the regularization. To take full advantage of these data a different approach may give better results. One possibility could be a multiscale approach where one would perform reconstructions in small windows in multiple scales to optimally reconstruct multiscale features of the source distribution. The challenge would be to efficiently and correctly stitch them together.

Parameter selection. In the Results section we discussed our strategy for data-based parameter selection using cross-validation or L-curve. Often, we need to tune not just λ but also other parameters. For example, for Gaussian basis sources we may want to decide on the width of the Gaussian used, R. To obtain the optimal set of parameters in that case we compute the curvature of the L-curve or the cross-validation error for some ranges of parameters considered and select parameters corresponding to the maximum curvature / minimum error in the parameter space. This is a simplification of the proposition by Belge, Kilmer, and Miller, 2002 which in practice we found very effective.

As an example, in Fig. 7 we show results of such a scan for the problem shown in

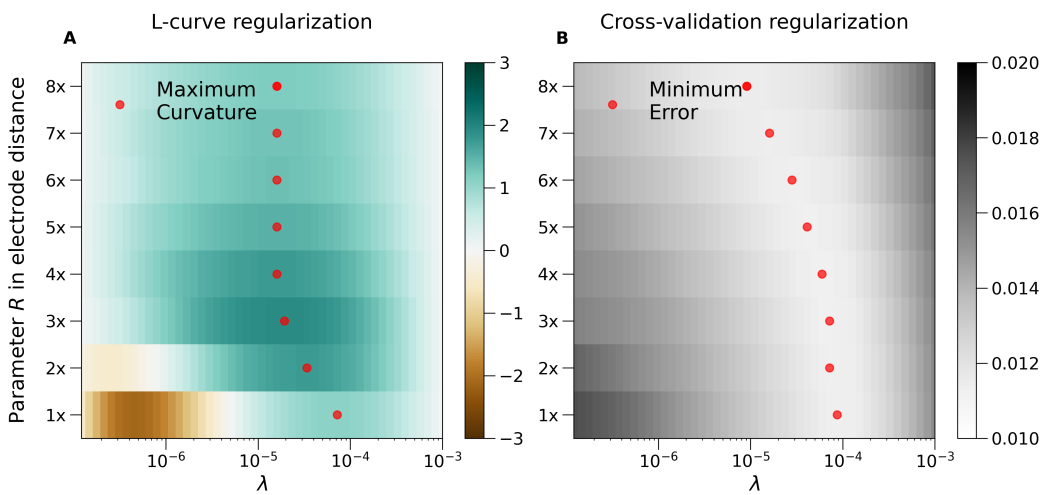

Figure 7: L-curve curvature (left) and CV-error (right) for the problem studied in Fig. 2. Observe that in both cases there are ranges of promising candidate parameter pairs, R, λ, which can give good reconstruction given the measured data. Red dots shows local extrema for each value of R fixed. See text for discussion of this effect.

Fig. 2. The range of λ to be considered can be set by hand but by default we base it on the eigenvalues of K. The smallest λ is set as the minimum eigenvalue of K which here was around 1e-10. We set maximum λ at standard deviation of the eigenvalues, which here was around 1e-3. The range of R values studied was from the minimum interelectrode distance to half the maximum interelectrode distance. Note that for very inhomogeneous distributions of electrodes this approach may be inadequate.

What we find is that apart from a global minimum in R, λ space there is a range of R values fixing which we can find optimal $\lambda(R)$ which leads to very close curvatures / CV-errors / estimation results. What happens is that within some limits we may achieve similar smoothing effects changing either λ or R. Bigger λ means more smoothing, but bigger R means broader basis functions and effectively also smoother reconstruction space. This is why the CV-error and curvature landscapes are relatively flat, or have these marked valleys observed in Fig. 7. This effect supports robustness of the kCSD approach.

The sources of error in kCSD estimation and how to deal with them. Kernel CSD method assumes a set of electrode positions and a corresponding set of recordings. Additionally, single cell kCSD requires morphology of the cell which contributed to the recordings and its position relative to the electrodes. Each of these may be subject to errors.

In analysis, we assume that the electrode positions are known precisely. This is a justified assumption in case of multishaft silicon probes or integrated CMOS-MEA but not necessarily when multiple laminar probes are placed independently within the brain or for many other scenarios. For example, for SEEG electrodes in human patients used in presurgical evaluation we expect the localization errors due to the workflows used clinically to be significant. We do not provide dedicated tools to study the effects of misplaced electrodes on the reconstructed CSD, however, this can be achieved easily with the provided package if needed. The location of the cell with respect to the electrodes is much more questionable, especially in 3D. Nevertheless, the necessary data to perform skCSD today are too scarce to start addressing these issues.

On the other hand we do assume that the recordings are noisy and we use regularization to counteract the effects of noise. We have no mechanism to differentiate between electrodes with varying degrees of noise to compensate this differently. However, we observed that for cases with very bad electrodes, similar results are obtained for analysis of complete data and for analysis of partial data with bad electrodes removed from analysis. The difference was in λ selected which was larger when broken electrodes were included in the analysis. Depending on situation, if there is a big difference in the noise visible in different channels, an optimal strategy may be to discard the noisy data and perform reconstruction from the good channels only, which kCSD permits. In the end, data analysis remains an art and a healthy dose of common sense is always recommended.

The main limitation of the method itself lies in the character of any inverse problem. Here it means that there is an infinite number of possible CSD distributions each consistent with the recorded potential. It is thus necessary to impose conditions which allow unique reconstruction and this is what every variant of CSD method is about. In kCSD this condition is minimization of regularized prediction error. In practical terms one may think of the function space in which we are making the reconstruction. This space is span by the eigensources we discussed before (Chintaluri et al., 2021). We feel it is useful to consider both this space as well as its complement, that is the set of CSD functions whose contribution to every potential is zero. This can facilitate understanding of which features

of the underlying sources can be recovered and which are inaccessible to the given setup. While for the most common regular setups, such as rectangular or hexagonal MEA grids or multishaft probes, intuitions from Fourier analysis largely carry over, in less regular cases this quickly becomes non-obvious.

To facilitate intuition building in the provided toolbox we include tools to compute the eigensources for a given setup. We also proposed here reliability maps, heuristic tools to build intuition regarding which parts of the reconstructed CSD can be trusted and which seem doubtful. These reliability maps are built around specific test ground truth distributions and some default parameters facilitating validation for any given setup are provided. Due to the open source nature of the provided toolbox more complex analysis is possible if the setup or experimental context require that.

Acknowledgements

The Python implementation of kCSD was started by Grzegorz Parka during Google Summer of Code project through the International Neuroinformatics Coordinating Facility. Jan Mąka implemented the first Python version of skCSD class. The study received funding from the Polish National Science Centre's grants (2013/08/W/NZ4/00691) and (2015/17/B/ST7/04123). The authors declare no conflict of interest.

References

Angotzi, Gian Nicola et al. (2019). "SiNAPS: An implantable active pixel sensor CMOS-probe for simultaneous large-scale neural recordings." In: *Biosensors & bioelectronics* 126 (Feb. 2019), pp. 355–364. ISSN: 1873-4235. DOI: 10.1016/j.bios.2018.10.032.

Ascoli, Giorgio A. (2006). "Mobilizing the base of neuroscience data: the case of neuronal morphologies." eng. In: *Nat Rev Neurosci* 7.4 (Apr. 2006), pp. 318–324. DOI: 10.1038/nrn1885. URL: http://dx.doi.org/10.1038/nrn1885.

Bakkum, Douglas J. et al. (2013). "Tracking axonal action potential propagation on a high-density microelectrode array across hundreds of sites." eng. In: *Nat Commun* 4, p. 2181. DOI: 10.1038/ncomms3181. URL: http://dx.doi.org/10.1038/ncomms3181.

Bédard, Claude and Alain Destexhe (2011). "Generalized theory for current-source-density analysis in brain tissue." eng. In: *Phys Rev E Stat Nonlin Soft Matter Phys* 84.4 Pt 1 (Oct. 2011), p. 041909.

Belge, Murat, Misha E. Kilmer, and Eric L. Miller (2002). "Efficient determination of multiple regularization parameters in a generalized L-curve framework". In: *Inverse Problems* 18.4, pp. 1161–1183. ISSN: 02665611. DOI: 10.1088/0266-5611/18/4/314.

Berdondini, L. et al. (2005). "High-density electrode array for imaging in vitro electrophysiological activity." eng. In: *Biosens Bioelectron* 21.1 (July 2005), pp. 167–174. DOI: 10.1016/j.bios.2004.08.011. URL: http://dx.doi.org/10.1016/j.bios.2004.08.011.

Buzsáki, György (2004). "Large-scale recording of neuronal ensembles". In: *Nat. Neurosci.* 7.5 (May 2004), pp. 446–451. DOI: 10.1038/nn1233. URL: http://dx.doi.org/10.1038/nn1233.

Buzsáki, György, Costas A Anastassiou, and Christof Koch (2012). "The origin of extracellular fields and currents — EEG, ECoG, LFP and spikes." eng. In: *Nat Rev Neurosci* 13.6 (June 2012), pp. 407–420. DOI: 10.1038/nrn3241. URL: http://dx.doi.org/10.1038/nrn3241.

Castellanos, J. Longina, Susana Gómez, and Valia Guerra (2002). "The triangle method for finding the corner of the L-curve". In: *Applied Numerical Mathematics* 43.4, pp. 359–373. ISSN: 01689274. DOI: 10.1016/S0168-9274(01)00179-9.

Chintaluri, Chaitanya et al. (2021). "What we can and what we cannot see with extracellular multielectrodes". In: *PLoS Computational Biology* 17.5, pp. 1–20. ISSN: 15537358. DOI: 10.1371/journal.pcbi.1008615.

Cserpan, Dorottya et al. (2017). "Revealing the distribution of transmembrane currents along the dendritic tree of a neuron from extracellular recordings". In: *eLife* 6, e29384. DOI: 10.7554/eLife.29384. URL: https://elifesciences.org/articles/29384.

Einevoll, Gaute T. et al. (2013). "Modelling and analysis of local field potentials for studying the function of cortical circuits." eng. In: *Nat Rev Neurosci* 14.11 (Nov. 2013), pp. 770–785. DOI: 10.1038/nrn3599. URL: http://dx.doi.org/10.1038/nrn3599.

Ferrea, E. et al. (2012). "Large-scale, high-resolution electrophysiological imaging of field potentials in brain slices with microelectronic multielectrode arrays." eng. In: *Frontiers in Neural Circuits* 6, p. 80. DOI: 10.3389/fncir.2012.00080. URL: http://dx.doi.org/10.3389/fncir.2012.00080.

Frey, U. et al. (2009). "Microelectronic system for high-resolution mapping of extracellular electric fields applied to brain slices." eng. In: *Biosens Bioelectron* 24.7 (Mar. 2009), pp. 2191–2198. DOI: 10.1016/j.bios.2008.11.028. URL: http://dx.doi.org/10.1016/j.bios.2008.11.028.

Głąbska, Helena et al. (2014). "Independent components of neural activity carry information on individual populations." In: *PloS one* 9 (8 2014), e105071. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0105071.

Goto, Takakuni et al. (2010). "An evaluation of the conductivity profile in the somatosensory barrel cortex of Wistar rats." In: *Journal of neurophysiology* 104 (6 Dec. 2010), pp. 3388–3412. ISSN: 1522-1598. DOI: 10.1152/jn.00122.2010.

Gratiy, Sergey L et al. (2017). "From Maxwell's equations to the theory of current-source density analysis." In: *The European journal of neuroscience* 45 (8 Apr. 2017), pp. 1013–1023. ISSN: 1460-9568. DOI: 10.1111/ejn.13534.

Hansen, P. (2010). *Discrete Inverse Problems*. Society for Industrial and Applied Mathematics. DOI: 10.1137/1.9780898718836. eprint: http://epubs.siam.org/doi/pdf/10.1137/1.9780898718836. URL: http://epubs.siam.org/doi/abs/10.1137/1.9780898718836.

Hansen, Per Christian (1992). "Analysis of Discrete Ill-Posed Problems by Means of the L-Curve". In: *SIAM Review* 34.4, pp. 561–580. ISSN: 0036-1445. DOI: 10.1137/1034115. arXiv: arXiv:1011.1669v3. URL: http://epubs.siam.org/doi/10.1137/1034115.

Hansen, Per Christian, Toke Jensen, and Giuseppe Rodriguez (2007). "An adaptive pruning algorithm for the discrete L-curve criterion". In: *Journal of Computational and Applied Mathematics* 198 (Jan. 2007), pp. 483–492. DOI: 10.1016/j.cam.2005.09.026.

Hottowy, Paweł et al. (2012). "Properties and application of a multichannel integrated circuit for low-artifact, patterned electrical stimulation of neural tissue". In: *Journal of neural engineering* 9.6, p. 066005.

Hunt, Mark J. et al. (2011). "Differential effects produced by ketamine on oscillatory activity recorded in the rat hippocampus, dorsal striatum and nucleus accumbens." eng. In:

J Psychopharmacol 25.6 (June 2011), pp. 808–821. DOI: 10.1177/0269881110362126. URL: http://dx.doi.org/10.1177/0269881110362126.

Jun, James J et al. (2017). "Fully integrated silicon probes for high-density recording of neural activity." In: *Nature* 551 (7679 Nov. 2017), pp. 232–236. ISSN: 1476-4687. DOI: 10.1038/nature24636.

Kajikawa, Yoshinao and Charles E Schroeder (2011). "How local is the local field potential?" In: *Neuron* 72.5, pp. 847–858. DOI: 10.1016/j.neuron.2011.09.029. URL: http://dx.doi.org/10.1016/j.neuron.2011.09.029.

Katzner, Steffen et al. (2009). "Local origin of field potentials in visual cortex." In: *Neuron* 61.1, pp. 35–41. DOI: 10.1016/j.neuron.2008.11.016. URL: http://dx.doi.org/10.1016/j.neuron.2008.11.016.

Klein, Natalie et al. (2021). "Cross-population coupling of neural activity based on Gaussian process current source densities." In: *PLoS computational biology* 17 (11 Nov. 2021), e1009601. ISSN: 1553-7358. DOI: 10.1371/journal.pcbi.1009601. epublish.

Kluyver, Thomas et al. (2016). "Jupyter Notebooks – a publishing format for reproducible computational workflows". In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. Ed. by F. Loizides and B. Schmidt. IOS Press, pp. 87–90.

Kong, Jee-Hyun et al. (2005). "Diversity of ganglion cells in the mouse retina: unsupervised morphological classification and its limits." eng. In: *J Comp Neurol* 489.3 (Aug. 2005), pp. 293–310. DOI: 10.1002/cne.20631. URL: http://dx.doi.org/10.1002/cne.20631.

Kropf, Pascal and Amir Shmuel (2016). "1D Current Source Density (CSD) Estimation in Inverse Theory: A Unified Framework for Higher-Order Spectral Regularization of Quadrature and Expansion-Type CSD Methods." In: *Neural computation* 28 (7 July 2016), pp. 1305–1355. ISSN: 1530-888X. DOI: 10.1162/NECO_a_00846.

Łęski, Szymon, Ewa Kublik, et al. (2010). "Extracting functional components of neural dynamics with Independent Component Analysis and inverse Current Source Density." In: *Journal of computational neuroscience* 29 (3 Dec. 2010), pp. 459–473. ISSN: 1573-6873. DOI: 10.1007/s10827-009-0203-1.

Łęski, Szymon, Henrik Lindén, et al. (2013). "Frequency dependence of signal power and spatial reach of the local field potential." eng. In: *PLoS Comput Biol* 9.7 (July 2013), e1003137. DOI: 10.1371/journal.pcbi.1003137. URL: http://dx.doi.org/10.1371/journal.pcbi.1003137.

Łęski, Szymon, Klas H Pettersen, et al. (2011). "Inverse Current Source Density method in two dimensions: Inferring neural activation from multielectrode recordings." eng. In: *Neuroinformatics* 9.4 (Dec. 2011), pp. 401–425. DOI: 10.1007/s12021-011-9111-4. URL: http://dx.doi.org/10.1007/s12021-011-9111-4.

Łęski, Szymon, Daniel K Wójcik, et al. (2007). "Inverse Current-Source Density method in 3D: reconstruction fidelity, boundary effects, and influence of distant sources." eng. In: *Neuroinformatics* 5.4, pp. 207–222. DOI: 10.1007/s12021-007-9000-z. URL: http://dx.doi.org/10.1007/s12021-007-9000-z.

Lindén, Henrik, Espen Hagen, et al. (2013). "LFPy: a tool for biophysical simulation of extracellular potentials generated by detailed model neurons." eng. In: *Front Neuroinform* 7, p. 41. DOI: 10.3389/fninf.2013.00041. URL: http://dx.doi.org/10.3389/fninf.2013.00041.

Lindén, Henrik, Tom Tetzlaff, et al. (2011). "Modeling the spatial reach of the LFP." eng. In: *Neuron* 72.5 (Dec. 2011), pp. 859–872. DOI: 10.1016/j.neuron.2011.11.006. URL: http://dx.doi.org/10.1016/j.neuron.2011.11.006.

Meijs, J. W. H. et al. (1988). "Relative influence of model assumptions and measurement procedures in the analysis of the MEG". In: *Medical and Biological Engineering and Computing* 26.2 (Mar. 1988), pp. 136–142. ISSN: 1741-0444. DOI: 10.1007/BF02442255. URL: https://doi.org/10.1007/BF02442255.

Mitzdorf, U. (1985). "Current source-density method and application in cat cerebral cortex: investigation of evoked potentials and EEG phenomena." In: *Physiological Reviews* 65, pp. 37–100.

Ness, Torbjørn V et al. (2015). "Modelling and analysis of electrical potentials recorded in microelectrode arrays (MEAs)". In: *Neuroinformatics* 13.4, pp. 403–426.

Nicholson, C. and J. A. Freeman (1975). "Theory of current source-density analysis and determination of conductivity tensor for anuran cerebellum." In: *J Neurophysiol* 38.2 (Mar. 1975), pp. 356–368.

Nunez, Paul L. and Ramesh Srinivasan (2006). *Electric Fields of the Brain.* 2nd. New York: Oxford University Press.

Pettersen, Klas H et al. (2006). "Current-source density estimation based on inversion of electrostatic forward solution: effects of finite extent of neuronal activity and conductivity discontinuities." eng. In: *J Neurosci Methods* 154.1-2 (June 2006), pp. 116–133. DOI: 10.1016/j.jneumeth.2005.12.005. URL: http://dx.doi.org/10.1016/j.jneumeth.2005.12.005.

Pitts, Walter (1952). "Investigations on synaptic transmission". In: *Cybernetics, Trans. 9th Conf. Josiah Macy, New York*, pp. 159–162.

Potworowski, Jan et al. (2012). "Kernel current source density method." eng. In: *Neural Comput* 24.2 (Feb. 2012), pp. 541–575. DOI: 10.1162/NECO_a_00236. URL: http://dx.doi.org/10.1162/NECO_a_00236.

Project Jupyter et al. (2018). "Binder 2.0 - Reproducible, Interactive, Sharable Environments for Science at Scale." In: *Proceedings of the 17th Python in Science Conference.* DOI: 10.25080/Majora-4af1f417-011.

Rappelsberger, P, H Pockberger, and H Petsche (1981). "Current source density analysis: methods and application to simultaneously recorded field potentials of the rabbit's visual cortex." In: *Pflugers Archiv : European journal of physiology* 389 (2 Jan. 1981), pp. 159–170. ISSN: 0031-6768.

Riera, Jorge J. et al. (2012). "Pitfalls in the dipolar model for the neocortical EEG sources." In: *Journal of neurophysiology* 108 (4 Aug. 2012), pp. 956–975. ISSN: 1522-1598. DOI: 10.1152/jn.00098.2011.

Vaknin, G, P G DiScenna, and T J Teyler (1988). "A method for calculating current source density (CSD) analysis without resorting to recording sites outside the sampling volume." In: *Journal of neuroscience methods* 24 (2 June 1988), pp. 131–135. ISSN: 0165-0270.

Appendix: kCSD-python package tutorial

In this section we first illustrate the use of kCSD package for CSD reconstruction in the simplest case of a regular 2D square grid. This is a simplified version of a slice on a microelectrode array (Ness et al., 2015), or a planar silicone probe within the brain, where we assume constant conductivity in the whole space. In the following sections we show how we validate our methods and what kind of diagnostics we find useful in the analysis of experimental data. This tutorial is available as a jupyter notebook and can also be accessed through a web-browser without installation. For more details, see 'Overview of kCSD-python package' in the Discussion.

Basic features

We start with the basic CSD estimation on a regular grid. First, we define a region of interest. Then, using predefined test functions for the current sources, we place a ground truth current source in this region. We define the distribution of electrodes. Assuming ideal electrodes, we compute the potential generated by the selected current sources as measured at the electrodes. Given these potentials and the electrode locations we estimate the current source density using kCSD. As a final step, we perform cross-validation to avoid overfitting. Since we know the ground truth used to generate the potentials that were used in the kCSD estimation, we can compare the ground truth to the estimate and see the reconstruction accuracy.

Defining region of interest

```
In [1]: %matplotlib inline
        import numpy as np
        csd_at = np.mgrid[0.:1.:101j,
                          0.:1.:101j]
        csd_x, csd_y = csd_at
```

We define the region of interest between `0` and `1` in the `xy` plane with a resolution of `101` points in each dimension. We will assume the distance is given in mm, so we want to perform a reconstruction on a square patch of $1mm^2$ size.

Setting up the ground truth The `kCSD-python` library provides functions to generate test sources which can be imported from the `csd_profile` module. Here we use the `gauss_2d_small` function to generate two-dimensional Gaussian sources which are small in the scale set by the interelectrode distance. The other implemented option for two-dimensional test sources is the `gauss_2d_large` function. To generate the exact same sources in each run we must invoke this function using the same random seed which is stored in the `seed` variable. For simplicity, these current sources are static and do not change with time. We visualize the current sources as a heatmap.

```
In [2]: from kcsd import csd_profile as CSD
        CSD_PROFILE = CSD.gauss_2d_small
        true_csd = CSD_PROFILE(csd_at, seed=15)
```

The code below displays this test source as the True CSD. For convenience we define this as a function `make_plot`. The output for this code is shown in Fig. 8A.

```
In [3]: import matplotlib.pyplot as plt
        import matplotlib.cm as cm

        def make_plot(xx, yy, zz, title='True CSD', cmap=cm.bwr):
            fig = plt.figure(figsize=(7, 7))
            ax = plt.subplot(111)
            ax.set_aspect('equal')
            t_max = np.max(np.abs(zz))
            levels = np.linspace(-1 * t_max, t_max, 32)
            im = ax.contourf(xx, yy, zz, levels=levels, cmap=cmap)
            ax.set_xlabel('X (mm)')
            ax.set_ylabel('Y (mm)')
            ax.set_title(title)
            ticks = np.linspace(-1 * t_max, t_max, 3, endpoint=True)
            plt.colorbar(im, orientation='horizontal', format='%.2f', ticks=ticks)
            return ax
        make_plot(csd_x, csd_y, true_csd, title='True CSD', cmap=cm.bwr)
```

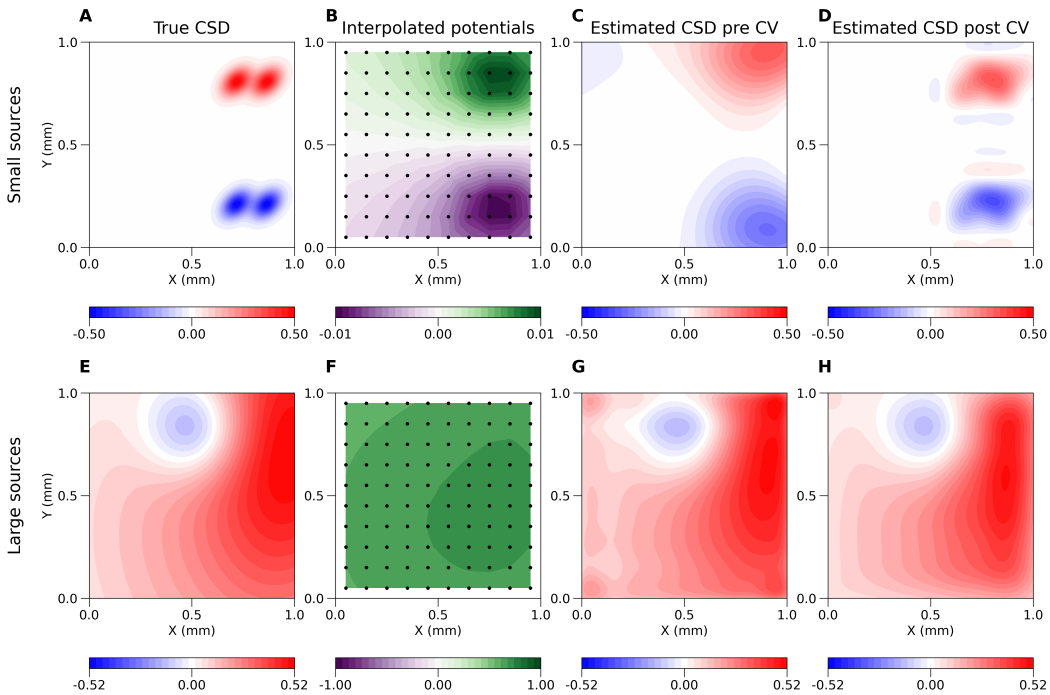

Figure 8: Basic features tutorial A) Shows the ground truth (True CSD), here, two-dimensional small Gaussian current sources, for the CSD seed of 15. B) The interpolated potentials generated by this current source are shown, the electrodes are displayed as black dots. C) CSD estimated with kCSD using the potentials from the electrode positions, without cross-validation. D) Same as C but cross-validation was used. E-H) Analogous to A–D, except large Gaussian current sources for seed 6 were used.

Place electrodes We now define the virtual electrodes within the region of interest. We place them between `0.05` *mm* and `0.95` *mm* of the region of interest, with a resolution of

10 (as indicated by 10j in mgrid) in each dimensions, totalling to 100 electrodes. Notice that the electrodes do not span the entire region of interest. Although in this example the electrodes are distributed on a regular grid, this is not required by the kCSD method as it can handle arbitrary distributions of electrodes.

```
In [4]: ele_x, ele_y = np.mgrid[0.05: 0.95: 10j,
                                0.05: 0.95: 10j]
        ele_pos = np.vstack((ele_x.flatten(), ele_y.flatten())).T
```

Compute potential To obtain the potential, pots, at the given electrode positions due to the current sources that were placed in the previous steps we use the function forward_method. We assume the sources are localized within a slab of tissue of thickness 2h on top the MEA (See Łęski, Pettersen, et al., 2011; Ness et al., 2015 and Methods). We also assume infinite homogeneous medium of conductivity sigma equal to 1 S/m. Finally, we assume that the electrodes are ideal, point-size and noise-free.

```
In [5]: from scipy.integrate import simps

        def integrate_2d(csd_at, true_csd, ele_pos, h, csd_lims):
            csd_x, csd_y = csd_at
            xlin = csd_lims[0]
            ylin = csd_lims[1]
            Ny = ylin.shape[0]
            m = np.sqrt((ele_pos[0] - csd_x)**2 + (ele_pos[1] - csd_y)**2)
            m[m < 0.0000001] = 0.0000001
            y = np.arcsinh(2 * h / m) * true_csd
            integral_1D = np.zeros(Ny)
            for i in range(Ny):
                integral_1D[i] = simps(y[:, i], ylin)
            return simps(integral_1D, xlin)

        def forward_method(ele_pos, csd_at, true_csd):
            pots = np.zeros(ele_pos.shape[0])
            xlin = csd_at[0, :, 0]
            ylin = csd_at[1, 0, :]
            h = 50.  # distance between the electrode plane and midslice
            sigma = 1.0 # S/m
            for ii in range(ele_pos.shape[0]):
                pots[ii] = integrate_2d(csd_at, true_csd,
                                        [ele_pos[ii][0], ele_pos[ii][1]], h,
                                        [xlin, ylin])
            return pots / 2 * np.pi * sigma

        pots = forward_method(ele_pos, csd_at, true_csd)
```

To visualize the potential, we interpolate the hundred values computed at the electrodes positions with interpolate.griddata function. Note that the kCSD estimation uses only the potential recorded at the electrode positions. To distinguish between the potentials and CSD plots we use different colormaps. The electrodes are marked with dots in this plot. The output from this step is shown in Fig. 8B.

```
In [6]: from scipy.interpolate import griddata

        def grid(x, y, z):
            x = x.flatten()
            y = y.flatten()
            z = z.flatten()
            xi, yi = np.mgrid[min(x):max(x):100j,
                              min(y):max(y):100j]
            zi = griddata((x, y), z, (xi, yi), method='linear')
            return xi, yi, zi

        pot_X, pot_Y, pot_Z = grid(ele_pos[:, 0], ele_pos[:, 1], pots)
        ax = make_plot(pot_X, pot_Y, pot_Z, title='Interpolated potentials',
                       cmap=cm.PRGn)
        ax.scatter(ele_pos[:, 0], ele_pos[:, 1], 10, c='k')
```

kCSD method Here we illustrate the most basic estimation of CSD with the `kcsd` library. Since our example is two dimensional the relevant method is `KCSD2D`. For convenience we encapsulate the actual method call with parameters being set inside a function `do_kcsd`. We first set `h` and `sigma` parameters of the forward model. Then we restrict the potentials to the first time point of the recording. For typical experimental data the shape of this matrix would be $N_{ele} \times N_{time}$, where N_{ele} is the number of electrodes and N_{time} is the total number of recorded time points. Next, we call the `KCSD2D` class with the relevant parameters. The only required parameters are the electrode positions, `ele_pos`, and the potentials they see, `pots`. We can also provide the parameters for the forward model, `h` and `sigma` here. We define a rectangular region of estimation by setting the values `xmin`, `xmax` and `ymin`, `ymax`. The number of basis functions, `n_src_init` is set to `1000`, basis functions are of the type `gauss`, and the width of the Gaussian basis source `R_init` is set to be `1`. Finally, estimated CSD is stored as `est_csd`.

```
In [7]: from kcsd import KCSD2D

        def do_kcsd(ele_pos, pots):
            h = 50.  # slice thickness
            sigma = 1.0 # S/m
            pots = pots.reshape((-1, 1)) # first time point
            return KCSD2D(ele_pos, pots, h=h, sigma=sigma,
                          xmin=0.0, xmax=1.0,
                          ymin=0.0, ymax=1.0,
                          n_src_init=1000, src_type='gauss', R_init=1.)

        k = do_kcsd(ele_pos, pots)
        est_csd = k.values('CSD')
```

Estimated current sources are shown in Fig. 8C. Compare this to the True CSD obtained before, Fig. 8A. Observe that the estimation is not very faithful. This is caused by the ground truth varying significantly in the scale of a single inter-electrode distance. In the next step we will use cross-validation to select better reconstruction parameters.

```
In [8]: make_plot(k.estm_x, k.estm_y, est_csd[:, :, 0],
                  title='Estimated CSD without CV', cmap=cm.bwr) # First time point
```

Cross validation Leave-one-out cross-validation is performed with a single line command. In this procedure we scan a range of `R` values which set the size of the Gaussian basis functions and the regularization parameter λ values. At the end of this step we obtain the optimal parameters that would correct for overfitting. The function outputs the progress of the cross-validation step and displays the optimal candidates in the last line. Alternatively, one could use the L-curve method to find these optimal parameters. Fig. 8D shows the kCSD reconstruction obtained after cross-validation. We find that this estimation of the current sources resembles the True CSD better.

```
In [9]: k.cross_validate(Rs=np.linspace(0.01, 0.15, 15))
        est_csd = k.values('CSD')
```

```
No lambda given, using defaults
Cross validating R (all lambda) : 0.01
Cross validating R (all lambda) : 0.02
...
Cross validating R (all lambda) : 0.15
R, lambda : 0.11 1.46779926762e-06
```

```
In [10]: make_plot(k.estm_x, k.estm_y, est_csd[:, :, 0],
                   title='Estimated CSD with CV', cmap=cm.bwr) # First time point
```

Noisy electrodes

Until now we assumed noise-free data, however, experimental data are always noisy. In this section we investigate how noise affects the kCSD estimation. We first show how to compute the reliability map which we introduced before, Eq. (21). Then we discuss reproducible generation of noisy data with varying noise amplitude. Finally, we study the error in the reconstruction as a function of changing noise level.

Reconstruction quality measure To assess the estimation quality we measure the point-wise difference between the true sources and the sources reconstructed with the kcsd. We define a function `point_errors` which takes the `true_csd` and the `estimated_csd` as the inputs, normalizes them individually, and computes the Frobenius norm of their difference.

```
In [11]: def point_errors(true_csd, est_csd):
             true_csd_r = true_csd.reshape(true_csd.size, 1)
             est_csd_r = est_csd.reshape(est_csd.size, 1)
             epsilon = np.linalg.norm(true_csd_r)/np.max(abs(true_csd_r))
             err_r = abs(est_csd_r/(np.linalg.norm(est_csd_r)) -
                       true_csd_r/(np.linalg.norm(true_csd_r)))
             err_r *= epsilon
             return err_r.reshape(true_csd.shape)

         error_ideal = point_errors(true_csd, est_csd)
```

We visualize this difference as before, except we use greyscale colormap to display the intensity of the reconstruction error. For convenience we define the plotting in a function called `make_error_plot`. The output from this step is shown in Fig. 9A.

```
In [12]: def make_error_plot(xx, yy, error, title='Error CSD'):
             fig = plt.figure(figsize=(7, 7))
             ax = plt.subplot(111)
             ax.set_aspect('equal')
             t_max = np.max(np.abs(error))
             levels = np.linspace(0, t_max, 32)
             im = ax.contourf(xx, yy, error, levels=levels, cmap=cm.Greys)
             ax.set_xlabel('X (mm)')
             ax.set_ylabel('Y (mm)')
             ax.set_title(title)
             ticks = np.linspace(0, t_max, 3, endpoint=True)
             plt.colorbar(im, orientation='horizontal', format='%.2f', ticks=ticks)
             return ax

         make_error_plot(k.estm_x, k.estm_y, error_ideal,
                         title='Error CSD, no noise')
```

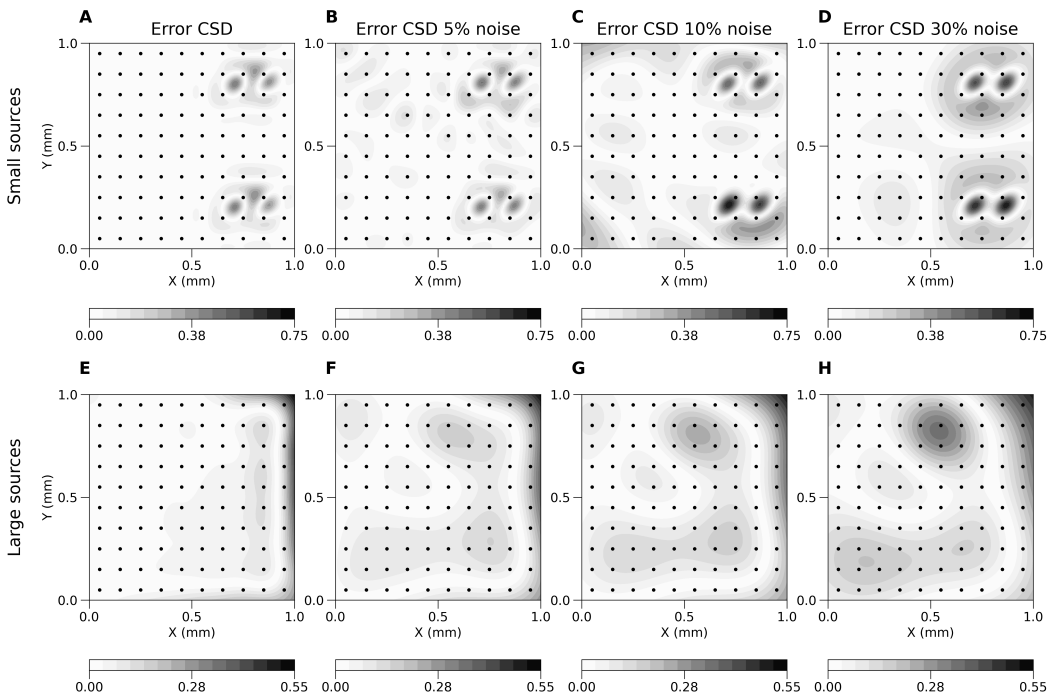

Figure 9: Noisy electrodes. A) The error between the True CSD and the estimation obtained with kCSD for a 2 dimensional small Gaussian current source, using the csd seed of 15. The electrodes in this case are assumed to be noise-free. B, C, D) Same as A, however, noise is added to the recorded potentials, whose magnitude is 5%, 10%, or 30%, respectively. E-H) Analogous to A–D, except in this case large Gaussian sources with seed 6 were used.

Noise definition To study resilience of the reconstruction against the noise in a controlled way we seed the random number generator in the function `add_noise`. We consider normally distributed noise with the mean and standard deviation set by reference to the recorded potentials.

```
In [13]: def add_noise(pots, noise_level=0, noise_seed=23):
             rstate = np.random.RandomState(noise_seed)
             noise = noise_level*0.01*rstate.normal(np.mean(pots),
                                                    np.std(pots),
                                                    size=(len(pots), 1))
             return pots + noise.reshape(pots.shape)

         pots_noise = add_noise(pots, noise_level=15, noise_seed=23)
```

Source reconstruction from noisy data With these tools we can study the effects of noise on the reconstruction. We now generate noise for a given noise level between 0 and 100, add it to the simulated potential, and estimate CSD from these noisy potentials. We can then use the error plots to compare the reconstruction with the True CSD. Notice that the parameters giving best reconstruction obtained for noisy data in general will be different from those obtained for clean potentials to compensate for noise.

```
In [14]: k_noise = do_kcsd(ele_pos, pots_noise)
         k_noise.cross_validate(Rs=np.linspace(0.01, 0.15, 15))
         estm_csd_noise = k_noise.values('CSD')
         error_noise = point_errors(true_csd, estm_csd_noise)
```

```
No lambda given, using defaults
Cross validating R (all lambda) : 0.01
Cross validating R (all lambda) : 0.02
...
Cross validating R (all lambda) : 0.15
R, lambda : 0.01 0.00110069417125
```

We can display this error with the `make_error_plot` plotting function which we defined earlier. Changing the `noise_level` and the `noise_seed` affects the reconstruction, but the error depends also on the sources, so changing the True CSD type to a `gauss_2d_large` or changing `csd_seed` will lead to different results. This is illustrated in Fig. 9A–D for small Gaussian sources, and Fig. 9E–H for large Gaussian sources, with varying noise levels. The actual ground truth and reconstructions are shown in Fig. 8.

```
In [15]: make_error_plot(k_noise.estm_x, k_noise.estm_y, error_noise,
                         title='Error CSD, with noise')
```

Broken electrodes

It is often the case that due to experimental constraints, some subset of recordings are discarded in the final analysis. This can happen when some electrodes are used for stimulation and cannot be used for recording, or due to bandwidth limitations requiring a compromise between sampling rates and the number of simultaneously recording electrodes, or in the

event of an electrode break down. In this section of the tutorial we discuss how to handle such cases and to estimate the errors in reconstruction despite the loss of recordings. We first show how we remove recordings from selected (broken) electrodes from considered data. Then we calculate the estimation error for a given source for data from such a damaged setup. Finally, we compute the average error across many sources from such an incomplete setup. Note that kCSD reconstruction is designed to work with arbitrary electrode setups and removing specific electrodes does not change the situation significantly. We focus on broken electrodes as it is a common situation in practice and deserves consideration. This may be used to gain intuition regarding ways in which `CSD` reconstruction may go wrong, due to slight disturbances in a familiar setup.

Remove broken electrodes To test the effects of removed electrodes on reconstruction from a given setup we simulate this with a function `remove_electrodes` that takes all the electrode positions for this setup and the number of electrodes that are to be removed. In this example we remove the electrodes randomly. Like we did previously, to facilitate repeatability we also pass a `broken_seed` variable, so that at each subsequent run the same electrodes are discarded. By changing this seed we select a different set of electrodes for removal.

```
In [16]: def remove_electrodes(ele_pos, num_broken, broken_seed=42):
             rstate = np.random.RandomState(broken_seed)
             rmv = rstate.choice(ele_pos.shape[0], num_broken, replace=False)
             ele_pos = np.delete(ele_pos, rmv, 0)
             return ele_pos

         ele_pos_new = remove_electrodes(ele_pos, 5)  # Discard 5 electrodes of 100
```

Error in estimation with broken electrodes After removing the broken electrodes we compute the estimation error to gauge the effect of electrode removal on reconstruction. Here, a fuction `calculate_error` takes a `csd_seed` as an input, which selects a specific ground truth source, and all the remaining electrode positions, `ele_pos`. The function computes the True CSD for a `gauss_2d_small` type source, computes the potential at these electrode locations, performes kcsd estimation from these data, and computes the error in the estimation of the true csd.

```
In [17]: def calculate_error(csd_seed, ele_pos):
             true_csd = CSD_PROFILE(csd_at, seed=csd_seed)
             pots = forward_method(ele_pos, csd_at, true_csd)
             k = do_kcsd(ele_pos, pots)
             k.cross_validate(Rs=np.linspace(0.01, 0.15, 15))
             err = point_errors(true_csd, k.values('CSD'))
             return k, err

         k, err = calculate_error(csd_seed=15, ele_pos=ele_pos_new)
```

```
No lambda given, using defaults
Cross validating R (all lambda) : 0.01
Cross validating R (all lambda) : 0.02
...
```

```
Cross validating R (all lambda) : 0.15
R, lambda : 0.14 1.33352143216e-05
```

Below (Fig. 10) we plot these errors. We also display the electrodes which were used in the kcsd estimation.

```
In [18]: ax = make_error_plot(k.estm_x, k.estm_y, err,
                              title='Error CSD, 5 broken electrodes')
         ax.scatter(ele_pos_new[:, 0], ele_pos_new[:, 1], 10, c='k')
```

Average error for multiple sources As we can see, the estimation error depends on the test current sources used. To better understand the effects of the setup we compute the average error across multiple sources. As an example here we show this for two seeds. In principle, any type and number of sources may be tested, as we showed before in analysis of reliability maps. This step is computationally expensive, however, it would normally be carried out only once for a given electrode design configuration. We believe this approach offers useful diagnostics and builds intuition regarding the estimation power for the given setup.

```
In [19]: seed_list = range(15, 17)
         error_list = []
         for csd_seed in seed_list:
             k, err = calculate_error(csd_seed=csd_seed, ele_pos=ele_pos_new)
             error_list.append(err)
         avg_error = sum(error_list) / len(error_list)
```

```
No lambda given, using defaults
Cross validating R (all lambda) : 0.01
...
Cross validating R (all lambda) : 0.15
R, lambda : 0.14 1.33352143216e-05
No lambda given, using defaults
Cross validating R (all lambda) : 0.01
...
Cross validating R (all lambda) : 0.15
R, lambda : 0.04 3.48070058843e-15
```

In Fig. 10A–D we show this for the case of 0, 5, 10 and 20 broken electrodes, when the average error for 100 small Gaussian sources was considered. In Fig. 10E–H we show the same for large Gaussian sources.

```
In [23]: ax = make_error_plot(k.estm_x, k.estm_y, avg_error,
                              title='Average Error, 5 broken electrodes')
         ax.scatter(ele_pos_new[:, 0], ele_pos_new[:, 1], 10, c='k')
```

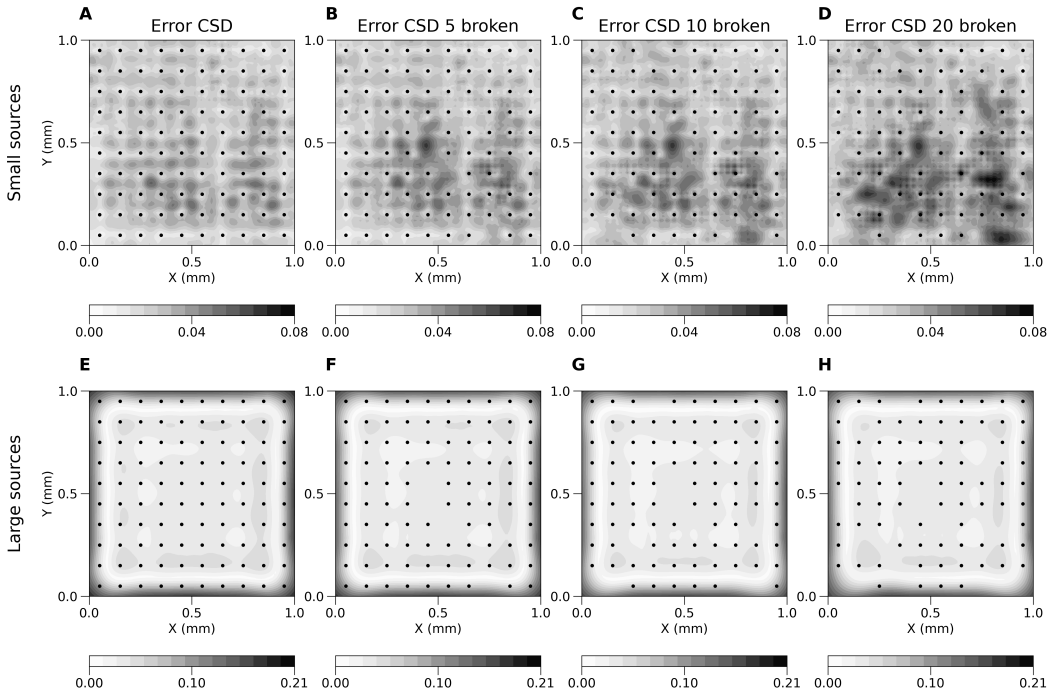

Figure 10: Broken electrodes. A) Shows the average error between the True CSD and the CSD estimated with kcsd for 100 random small Gaussian current sources. B) The same average error as in A, except in this case 5 electrodes were discarded in the estimation. Likewise for C and D, where 10 and 20 electrodes out of the 100 were considered broken. E-H) analogous to A–D, except in this case we show the averages for 100 large Gaussian current sources.

Supplementary Materials

S1 Fig

Error propagation maps in a 1D example.

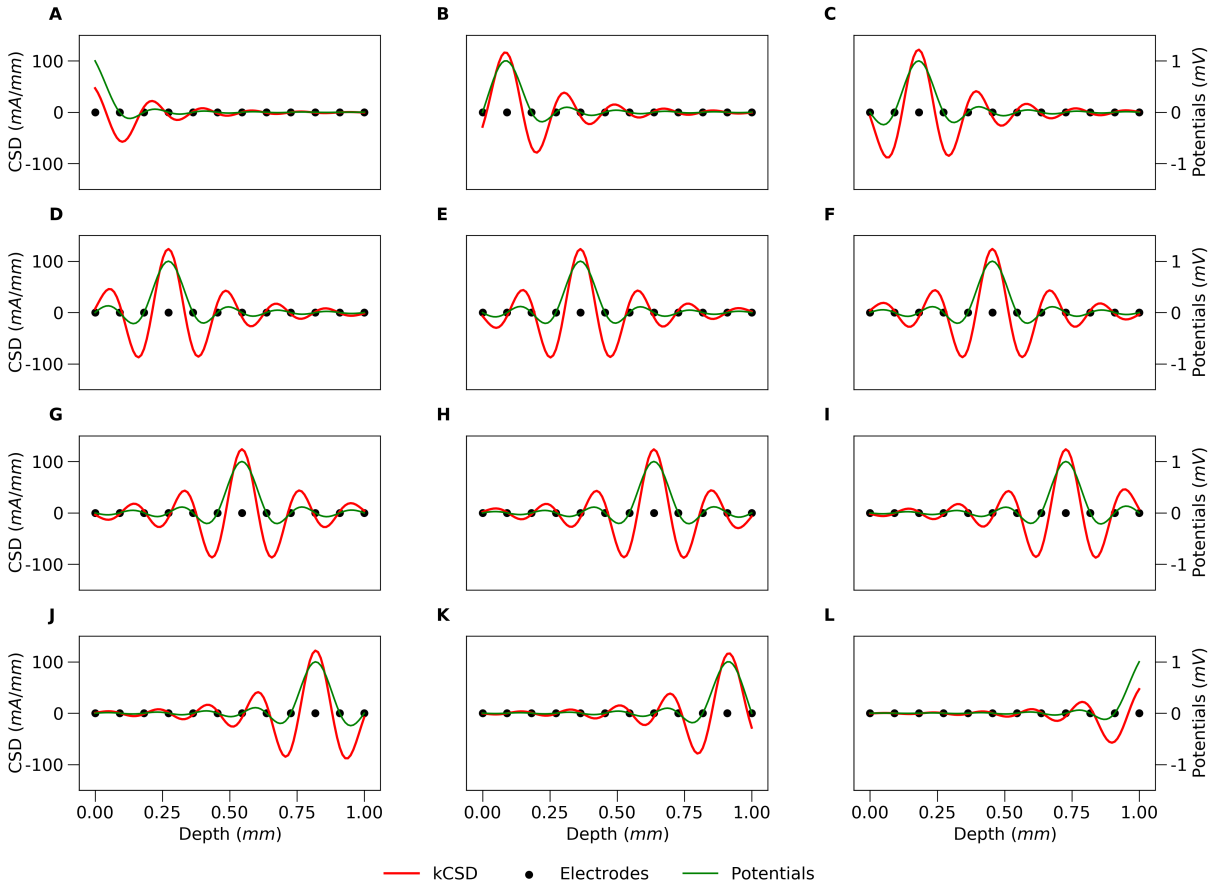

Figure 11: **Error propagation maps** for 1D regular grid of 12 electrodes Every panel represents the CSD contribution (red line) of the potential measured at the corresponding electrode, for which the potential is 1 (green line).

S2 Fig

An example of 3D source reconstruction.

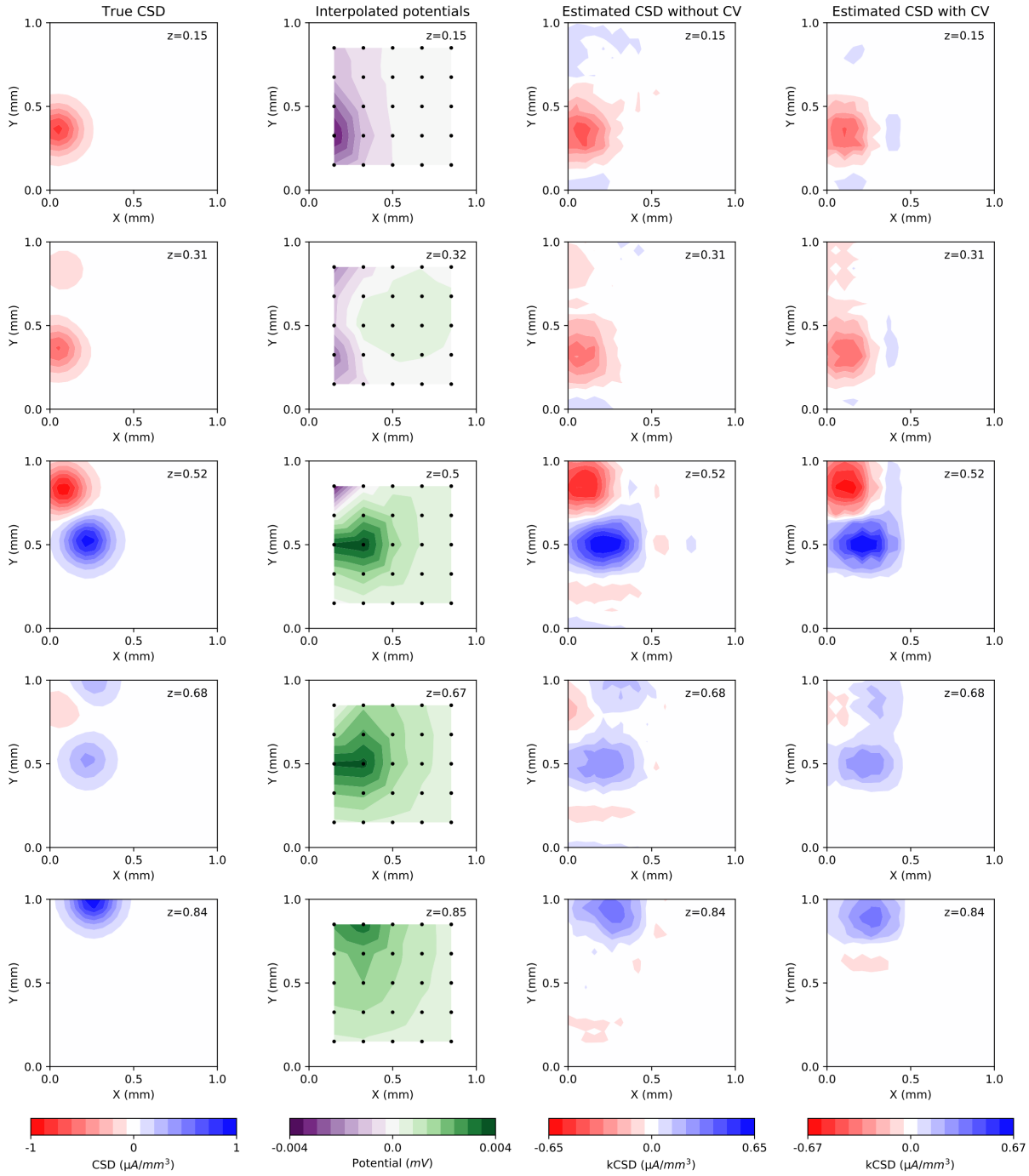

Figure 12: **An example of 3D kCSD source reconstruction**. Each column shows five consecutive parallel cuts through a box of size 1. A) Ground truth for the CSD seed of 16. B) Estimated potential; black dots indicate electrodes where potential is collected for further reconstruction. C) 3D kCSD reconstruction from the measured potentials, $\lambda = 0$. D) 3D kCSD reconstruction with cross-validation.

S3 Fig

An example of skCSD source reconstruction.

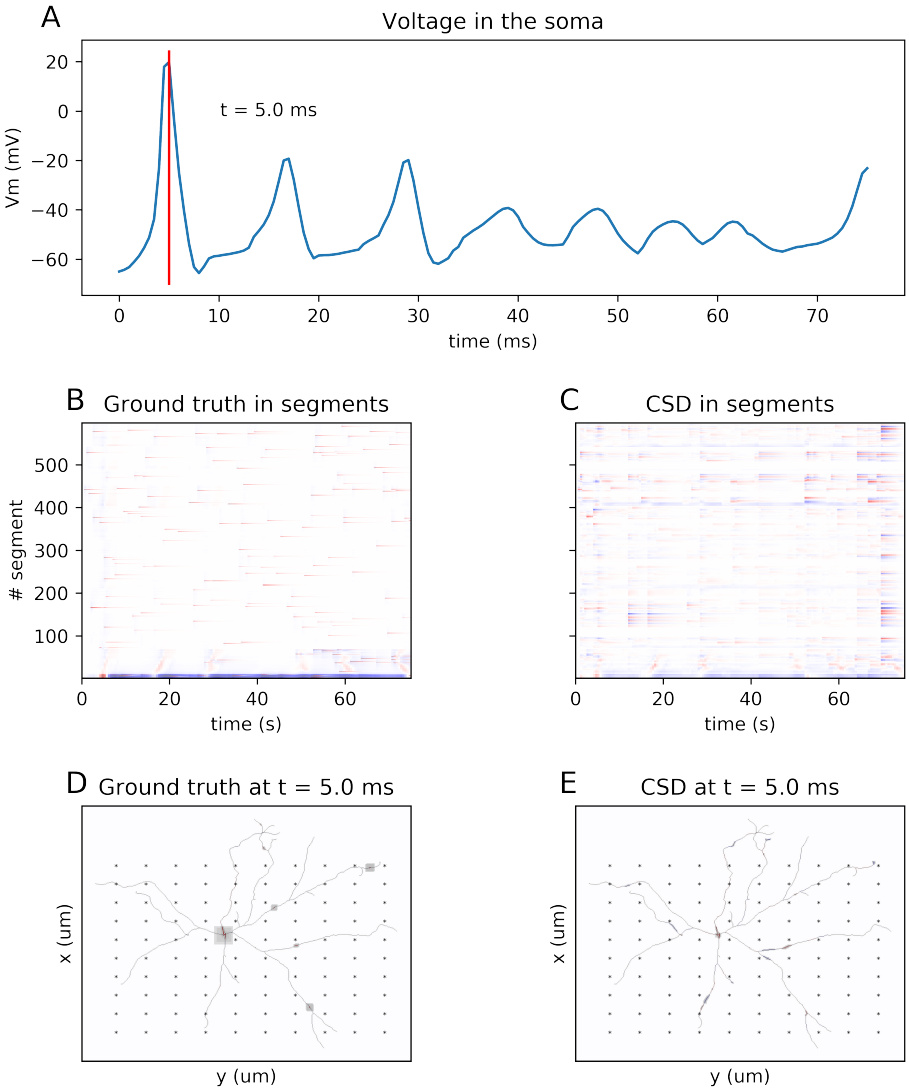

Figure 13: **An example of skCSD reconstruction** Cserpan et al., 2017 of somatic current injection together with random synaptic input patterns for a retinal ganglion cell model. A) Somatic membrane potential. B) Current density and its C) skCSD reconstruction in the segment space. Projection of D) ground truth and E) skCSD reconstruction on the neuron's morphology at 5 s of the simulation. We simulated a multicompartmental model of a mouse retinal ganglion cell (morphology Kong et al., 2005 obtained from NeuroMorpho.Org Ascoli, 2006) with Hodgkin-Huxley sodium, potassium, and leakage channels in the soma (hh mechanism) in NEURON simulation environment. For calculation of the measured extracellular potentials we used LFPy package Lindén, Hagen, et al., 2013. The model neuron was stimulated by an injection of oscillatory current to the soma (with frequency of 24.5 1/ms and amplitude of 3.6 nA) together with random synaptic inputs (weight of 0.04 μS) to the dendritic tree. The activity of the model neuron was measured by a rectangular grid of 100 electrodes (10×10, -400 $\mu m \times 400 \mu m$). The figure corresponds to Fig. 8 from (Cserpan et al., 2017).