

# Greedy low-rank algorithm for spatial connectome regression

Patrick Kürschner\*    Sergey Dolgov †    Kameron Decker Harris ‡  
 Peter Benner§

August 15, 2018

## Abstract

We study a smooth matrix regression problem formulated to recover brain connectivity from data. The problem is set up as a general algebraic linear matrix equation of high dimension, and the desired solution is approximated by a low-rank factorization. This factorization is computed iteratively by an efficient greedy low-rank algorithm that exploits important properties of the original matrix equation, such as sparsity and positive definiteness. The performance of the method is evaluated on three test cases: a toy artificial dataset and two cortical problems employing data from the Allen Mouse Brain Connectivity Atlas. We find numerically that the method is significantly faster than previous gradient-based methods and converges to the full rank solution as the rank increases. This speedup allows for the estimation of increasingly large-scale connectomes as these data become available from tracing experiments.

**Keywords:** *Matrix equations, computational neuroscience, low-rank approximation, networks*

## 1 Introduction

Neuroscience and machine learning are now enjoying a shared moment of intense interest and exciting progress. Many computational neuroscientists find themselves inspired by unprecedented datasets to develop innovative methods of analysis. Exciting examples of such next-generation experimental methodology and datasets are large-scale recordings and precise manipulations of brain activity, genetic atlases, and neuronal network tracing efforts. Building on previous work [15, 18], we present a scalable method to infer spatially-resolved brain network structure using data from the Allen Mouse Brain Connectivity Atlas [25]. This resource is one of the most comprehensive publicly available datasets, but similar data are being collected for fly [17], rat [5], and marmoset [23], among others. Thus, techniques which summarize many experiments into a consensus network are increasingly important.

---

\*Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg (kuerschner@mpi-magdeburg.mpg.de).

†University of Bath, Mathematical Sciences (s.dolgov@bath.ac.uk)

‡University of Washington, Computer Science & Engineering (kamdh@uw.edu).

§Max Planck Institute for Dynamics of Complex Technical Systems, Magdeburg (benner@mpi-magdeburg.mpg.de).

Many believe that uncovering such network structures will help us unlock the principles underlying neural computation and brain disorders [13].

Structural connectivity refers to the synaptic connections formed between axons (output processes) and dendrites (inputs) of neurons, which allow them to communicate chemically and electrically. We represent such networks as a weighted, directed graph encoded by a nonnegative adjacency matrix  $W$ . The network of whole-brain connections or *connectome* is currently studied at a number of scales [31]: Microscopic connectivity catalogues individual neuron connections but currently is restricted to small volumes due to difficult tracing of convoluted geometries. Macroscopic connectivity refers to connections between larger brain regions and is currently known for a number of model organisms. Mesoscopic connectivity [24] lies between these two extremes, and captures projection patterns of groups of hundreds to thousands of neurons (of the  $10^6$ – $10^{10}$  neurons in a typical mammalian brain).

We focus on the mesoscale because it is naturally captured by viral tracing experiments. In these experiments, a virus is injected into a specific location in the brain, where it loads the cells with proteins that can then be imaged, tracing out the projections of those neurons with cell bodies located in the injection site. Performing many such experiments and varying the injection sites to cover the brain, one can then “stitch” together a mesoscopic connectome. We refer the interested reader to [25] for more details of the experimental procedures.

## 1.1 Previous methods of mesoscale connectome regression

Much of the work on mesoscale connectomes that we are familiar with leverages the data and processing pipelines of the Allen Mouse Brain Connectivity Atlas available at <http://connectivity.brain-map.org> [21, 25]. In the early examples of such work, Oh et al. used viral tracing data to construct regional connectivity matrices [25]. The mathematical techniques employed for regional connectivity was a multivariate (i.e. matrix) nonnegative regression. First, the injection data were processed into a pair of matrices  $X^{\text{Reg}}$  and  $Y^{\text{Reg}}$  containing the regionalized injection volumes and projection volumes, respectively. Oh et al. then used nonnegative least squares to fit a matrix  $W^{\text{Reg}}$  such that  $Y^{\text{Reg}} \approx W^{\text{Reg}} X^{\text{Reg}}$ . Due to numerical ill-conditioning and a lack of data, some regions were excluded from the analysis. Ypma and Bullmore [34] used a different, likelihood-based Markov chain Monte Carlo method to infer regional connectivity and weight uncertainty with the same data.

Harris et al. [15] made a conceptual and methodological leap when they presented a method to use such data for spatially-explicit mesoscopic connectivity. The Allen Mouse Brain Atlas is essentially a coordinate mapping which discretizes the “average” mouse brain into 10–100  $\mu\text{m}$  cubic voxels, where each voxel is assigned to a given region in a hierarchy of brain regions. They developed a method which used an assumption of spatial smoothness to fit the very large matrix of nonnegative voxel-voxel connection weights. See Figure 1.1 for a depiction of the data used and the interpretation of the matrix. The problem was cast as nonnegative matrix regression with a Tikhonov smoothing penalty and applied to the visual areas of the mouse brain. The smoothing term is essentially a high-dimensional thin plate spline [33]. Using a simple low-rank version, Harris et al. argued that such a method could scale to larger brain areas. However, the initial low-rank implementation, employing projected gradient descent, turned out to be too slow to converge for large-scale applications. Times to convergence were not reported in [15], but the full rank version typically took around a day, while the low-rank version needed multiple days to reach a minimum<sup>1</sup>.

---

<sup>1</sup>KD Harris, personal communication, 2017.

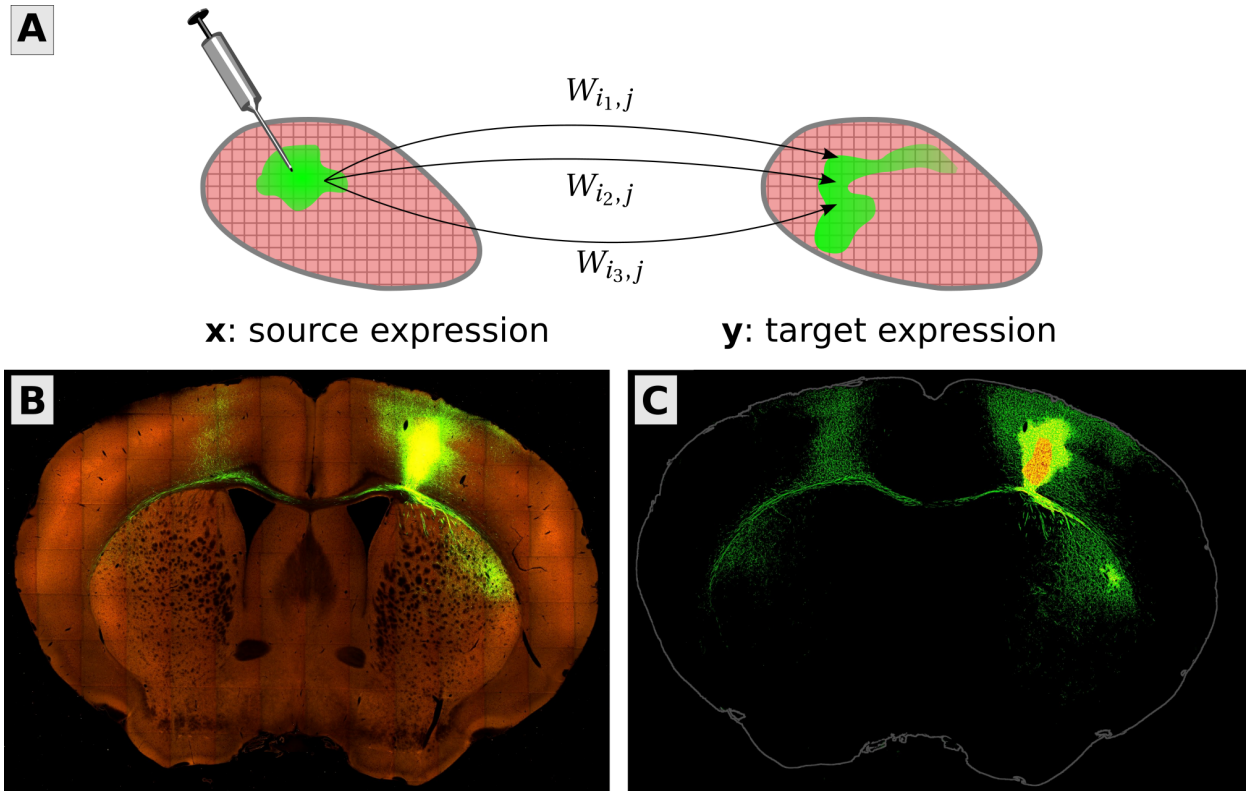


Figure 1.1: Schematic of the mesoscale connectome inference problem. **A)** The goal is to find a matrix  $W$  so that the pattern of neural projections  $y$  arising from an injection  $x$  is recovered via matrix-vector multiplication:  $y \approx Wx$ . These vectors  $x, y$  come from viral tracing experiments. **B)** An example of the data, in this case a coronal slice from a tracing experiment delivered to primary motor cortex (MOp). Bright green areas are neural cells expressing the green fluorescent protein. **C)** The raw data are preprocessed to separate the injection site (red/orange) from its projections (green). Fluorescence values in the injection site enter into the source vector  $x$ , whereas fluorescence everywhere else is stored in the entries of the target vector  $y$ . Each of these vectors is a discretized volume image of the brain. An entry  $W_{ij}$  in the connectome matrix is the amount of fluorescence at location  $i$  given one unit of source fluorescence at location  $j$ . The matrix  $W$  is a linear operator mapping source images to target images. Image credit (B and C): Allen Institute for Brain Science.

In their next attempt, Knox et al. [18] simplified the mathematical problem by assuming that the injections were delivered to just the center of mass of the injection volume. Furthermore, smoothing was performed in just the injection space, i.e. over just the columns rather than both columns and rows of  $W$ . The multivariate regression problem ends up simplifying into a kernel regression problem which is explicitly low-rank, due to the “kernel trick” [33]. Such a method was then applied to the whole mouse brain, yielding the first estimate of voxel-voxel whole-brain connectivity for this animal. However, because the smoothing was performed only in the source space, it would be better to improve the low-rank method for the original spline problem [15] and apply it to the whole brain.

## 1.2 Mathematical outline of the spatial connectome regression problem

Here, we present a new low-rank approach to solving the smoothness-regularized optimization problem posed by Harris et al. [15]. Specifically, they considered solving

$$W^* = \arg \min_{W \geq 0} \frac{1}{2} \|P_\Omega(WX - Y)\|_F^2 + \frac{\lambda}{2} \|L_y W + W L_x^\top\|_F^2, \quad (1.1)$$

where the minimum is taken over all nonnegative matrices,  $P_\Omega$  defines an entry-wise product (Hadamard product)  $P_\Omega(M) = M \circ \Omega$ , for any matrix  $M \in \mathbb{R}^{n_Y \times n_{\text{inj}}}$ , and  $\Omega$  is a binary matrix, masking out the injection sites where the entries of  $Y$  are unknown. Moreover, the data matrices are  $X \in \mathbb{R}^{n_X \times n_{\text{inj}}}$  and  $Y \in \mathbb{R}^{n_Y \times n_{\text{inj}}}$ , and the smoothing matrices  $L_y \in \mathbb{R}^{n_Y \times n_Y}$  and  $L_x \in \mathbb{R}^{n_X \times n_X}$ . The parameters  $n_X$  and  $n_Y$  are the number of locations in the discretized source and target regions of the brain. In general, these could be unequal, e.g. if injections were only delivered to the right hemisphere of the brain. The other parameter  $n_{\text{inj}}$  is the number of injections performed, i.e. the number of pairs  $(x_i, y_i)$  we have available. Note that we choose a regularization parameter  $\hat{\lambda}$  and set  $\lambda = \hat{\lambda} \frac{n_{\text{inj}}}{n_X}$  to avoid dependence on the problem size. Also note that, in this paper, we take a different convention for  $\Omega$  (the complement) as in the previous paper [15].

The spatial problem (1.1), with orders of magnitude more entries in  $W$  than the regional case, is highly underdetermined. The gridded brain at 100  $\mu\text{m}$  resolution contains approximately  $n_X, n_Y \in \mathcal{O}(10^5)$  voxels in 3-D. On the other hand, the number of experiments  $n_{\text{inj}}$  is less than  $\mathcal{O}(10^3)$ . By projecting the 3-D cortical data into 2-D, as we do in this paper, we can reduce the size by an order of magnitude  $n_X, n_Y \in \mathcal{O}(10^4)$ . However, focusing on the cortex also leads to a reduced  $n_{\text{inj}} \in \mathcal{O}(10^2)$ . In any case, inferring a full spatial connectivity will always be an underdetermined problem until orders of magnitude more tracing experiments are performed. The smooth regularizer is thus essential for filling the gaps in injection coverage. Furthermore, the vast size of the  $n_Y \times n_X$  matrix  $W$  for whole-brain connectivities has motivated our search for scalable, fast low-rank methods. As in many regularized underdetermined problems, low-rank solutions will often fit the data nearly as well as a full-rank ones.

### 1.2.1 Continuum version explains the numerical difficulties of naive iterative methods

We will now describe some of the mathematical properties of this problem, in order to illuminate some of the reasons why it can be challenging to solve. Equation (1.1) can be seen as a discrete version of a continuous problem, where the cost which is minimized is:

$$\frac{1}{2} \sum_{i=1}^{n_{\text{inj}}} \int_{T \cap \Omega_i} \left( \int_S \mathcal{W}(x, y) X_i(x) dx - Y_i(y) \right)^2 dy + \frac{\lambda}{2} \int_T \int_S (\Delta \mathcal{W}(x, y))^2 dx dy. \quad (1.2)$$

Here,  $\mathcal{W} : T \times S \rightarrow \mathbb{R}$  is the continuous connectome to be fit, in this case represented as the kernel of an integral operator from source space  $S$  to target space  $T$ . These regions  $S$  and  $T$  are both compact subsets of  $\mathbb{R}^3$ . For simplicity, one can consider  $S = T =$  the whole brain. The mask region  $\Omega_i \subset T$  is the subset of the brain excluding the injection site. Finally, the discrete Laplacian terms  $L$  have been replaced by the continuum Laplacian operator  $\Delta$ .

For simplicity, consider  $\Omega_i = T$  for all  $i = 1, \dots, n_{\text{inj}}$  and relax the constraint of nonnegativity on  $\mathcal{W}$ . Taking the first variational derivative of (1.2) and setting it to zero yields

the Euler-Lagrange equations for this problem without the nonnegativity constraint on  $\mathcal{W}$ . After simplification, these are

$$\begin{aligned} 0 &= \lambda \Delta^2 \mathcal{W}(x, y) - \sum_{i=1}^{n_{\text{inj}}} X_i(x) Y_i(y) + \int_S \mathcal{W}(x', y) \left( \sum_{i=1}^{n_{\text{inj}}} X_i(x') X_i(x) \right) dx' \\ &= \lambda \Delta^2 \mathcal{W}(x, y) - g(x, y) + \int_S \mathcal{W}(x', y) f(x', x) dx', \end{aligned} \quad (1.3)$$

where for convenience we have defined the functions

$$f(x', x) = \sum_{i=1}^{n_{\text{inj}}} X_i(x') X_i(x) \quad \text{and} \quad g(x, y) = \sum_{i=1}^{n_{\text{inj}}} X_i(x) Y_i(y).$$

Note that  $f$  and  $g$  are analogous to  $XX^T$  and  $YX^T$ , respectively, which represent data covariance matrices that occur in the discrete version of the problem. The smoothing operator  $\Delta^2$  is the biharmonic operator (also known as the bi-Laplacian).

Equation (1.3) is a fourth-order partial integro-differential equation. These equations are typically quite difficult to solve because they are *nonlocal*. Iterative solutions such as gradient-based methods to biharmonic and similar equations can be pathologically slow to converge. It takes many iterations to propagate the highly local action of the biharmonic differential operator across global spatial scales; this can be likened to the slow  $O(\sqrt{t})$  convergence of the heat equation to a stationary distribution. In these problems, appropriate preconditioners or multigrid methods are very important. However, they might be difficult to derive for the mixed integro-differential equation.

Very slow convergence is what we have found when applying gradient-based methods to problem (1.1) and low-rank versions of it. When we attempted to solve the whole-cortex top view and flatmap problems in Sections 3.2 and 3.3, the method had not converged (from a naive initialization) after a week of computation. However, initializing with the output of the method we present here speeds things up significantly.

### 1.2.2 Outline of the paper

As an alternative, we present a greedy low-rank algorithm tailored to the previously mentioned problem. Specifically, we relax the nonnegativity constraint and consider solutions to

$$W^* = \arg \min_W \frac{1}{2} \|P_\Omega(WX - Y)\|_F^2 + \frac{\lambda}{2} \|L_y W + W L_x^\top\|_F^2, \quad (1.4)$$

where all of the matrices and parameters are as in (1.1). In practice, we find that the solutions to the linear problem (1.4) are very close to those of the nonnegative problem (1.1), since the data matrices  $X$  and  $Y$  themselves are nonnegative. Setting any negative entries in the computed solution  $W^*$  to zero is adequate, or can serve as an excellent initial guess to an iterative solver for the slower nonnegative problem.

Equation (1.4) is a regularized least-squares problem. In Section 2.1, we show that taking the gradient and setting it equal to zero leads to a linear matrix equation in the unknown  $W$ , the normal equations for this problem. This takes the form of a generalized Sylvester equation with coefficient matrices formed from the data and Laplacian terms. The data matrices are, in fact, of low rank since  $n_{\text{inj}} \ll n_X, n_Y$ , and thus we can expect a low-rank approximation  $W \approx UV^\top$  to the full solution to perform well (see [15], although we are not sure how to justify this rigorously in our specific case).

We provide a brief survey of some low-rank methods for linear matrix equations in Section 2.2. We employ a greedy solver that finds rank-one components  $u_i v_i^\top$  one at a time, explained in Section 2.3. After a new component is found, it is orthogonalized and a Galerkin refinement step is applied. This leads to our complete method, Algorithm 1.

We then test the method on a few connectome fitting problems. First, in Section 3.1, we test on a fake “toy” connectome, where we know the truth. This is the same test problem considered in [15], consisting of a 1-D brain with smooth connectivity. We find that the output of our algorithm converges to the true solution as the rank increases and as the stopping tolerance decreases.

Next, we present two benchmarks using real viral tracing data from the isocortices of mice, provided by the Allen Institute for Brain Science (see Section 5 for details and links to data and code to reproduce our results). In each case, we work with 2-D data in order to limit the problem size and because the cortex is a relatively flat, 2-D shape. It has also been argued that such a projection also denoises such data [32, 14]. In Section 3.2, we work with data that are averaged directly over the superior-inferior axis to obtain a flattened cortex. We refer to this as the *top view* projection. In contrast, for Section 3.3, the data are flattened by averaging along curved streamlines of cortical depth. We call this the *flatmap* projection.

## 2 Greedy low-rank method

In this section, we detail our method for problem (1.4).

### 2.1 Linear matrix equation for the unknown connectivity

Denote the objective function in (1.4) as  $J(W)$ ; then it can be decomposed as

$$J(W) = J_P(W) + J_L(W) = \frac{1}{2} \|P_\Omega(WX - Y)\|_F^2 + \frac{\lambda}{2} \|L_y W + W L_x^\top\|_F^2.$$

Writing  $J_P$  indexwise, we obtain (note that  $\Omega \circ \Omega = \Omega$ )

$$J_P = \frac{1}{2} \sum_{i,\alpha=1}^{n,n_{\text{inj}}} \Omega_{i,\alpha} \left( \sum_{k=1}^m W_{i,k} X_{k,\alpha} - Y_{i,\alpha} \right)^2.$$

The derivative reads

$$\begin{aligned} \frac{\partial J_P}{\partial W_{i,\hat{k}}} &= \sum_{i,\alpha=1}^{n,n_{\text{inj}}} \Omega_{i,\alpha} \left( \sum_{k=1}^m W_{i,k} X_{k,\alpha} - Y_{i,\alpha} \right) X_{\hat{k},\alpha} \delta_{i,\hat{i}} \\ &= \sum_{\alpha=1}^{n_{\text{inj}}} \Omega_{i,\alpha} X_{\hat{k},\alpha} \sum_{k=1}^m \left( X_{k,\alpha} W_{i,k} - X_{\hat{k},\alpha} \Omega_{i,\alpha} Y_{i,\alpha} \right), \end{aligned}$$

or in the vector form

$$\frac{\partial J_P}{\partial \text{vec}(W)} = \sum_{\alpha=1}^{n_{\text{inj}}} [(X_\alpha X_\alpha^\top) \otimes \text{diag}(\Omega_\alpha)] \text{vec}(W) - \text{vec}((\Omega \circ Y) X^\top),$$

where  $X_\alpha$  is the  $\alpha$ -th column of  $X$ . Setting the derivative equal to zeros leads to the system of normal equations

$$A \text{vec}(W) = \text{vec}((\Omega \circ Y) X^\top), \quad (2.1)$$

where  $\text{vec}(W)$  is the vector of all columns of  $W$  stacked on top of each other. This linear system features the following matrix, consisting of  $n_{\text{inj}} + 3$  Kronecker products,

$$A = \sum_{\alpha=1}^{n_{\text{inj}}} (X_{\alpha} X_{\alpha}^{\top}) \otimes \text{diag}(\Omega_{\alpha}) + \lambda(L_x^2 \otimes I + 2L_x \otimes L_y + I \otimes L_y^2). \quad (2.2)$$

Note that without the observation mask,  $\Omega$  is a matrix of all ones, and the first term compresses to  $XX^{\top} \otimes I$ .

Linear system (2.1) can be recast as linear matrix equation

$$\mathcal{A}(W) = D, \quad (2.3)$$

with the operator  $\mathcal{A}(W) := \lambda\mathcal{B}(W) + \mathcal{C}(W)$ , where

$$\mathcal{B}(W) := WL_x^2 + 2L_yWL_x + L_y^2W, \quad \mathcal{C}(W) := \sum_{\alpha=1}^{n_{\text{inj}}} \text{diag}(\Omega_{\alpha})WX_{\alpha}X_{\alpha}^{\top},$$

and the right hand side is

$$D := (\Omega \circ Y)X^{\top}.$$

Note that the smoothing term  $\mathcal{B}$  can be expressed as a squared standard Sylvester operator

$$\mathcal{B}(W) = \mathcal{S}(\mathcal{S}(W)), \quad \text{where } \mathcal{S}(W) := L_yW + WL_x.$$

Operator  $\mathcal{S}$  is the discrete Laplacian operator on the tensor product space  $T \times S$ . Furthermore, the right hand side  $D$  is a matrix of rank  $n_{\text{inj}}$ , since it is an outer product of two rank  $n_{\text{inj}}$  matrices.

## 2.2 Numerical low-rank methods for linear matrix equations

Because of the potentially high dimensions  $n_X, n_Y$ , directly solving the algebraic matrix equation (2.3) is numerically inefficient since the solution will be a dense  $n_Y \times n_X$  matrix, making even storing it infeasible. However, since  $(\Omega \circ Y) \in \mathbb{R}^{n_Y \times n_{\text{inj}}}$ ,  $X \in \mathbb{R}^{n_X \times n_{\text{inj}}}$ , the rank of the right hand side of (2.3) is at most  $n_{\text{inj}}$ . By assumption, this is much smaller than the sizes of overall problem dimensions  $n_X, n_Y$ . It is often observed and theoretically shown [12, 2, 16] that the solutions of large matrix equations with low-rank right hand sides exhibit rapidly decaying singular values. Hence, the solution  $W$  is expected to have a very small numerical rank in the sense that only very few of its singular values are larger than the machine precision. This motivates us to approximate the solution of (2.3) by a low-rank approximation  $W \approx UV^{\top}$  with  $U \in \mathbb{R}^{n_Y \times r}$ ,  $V \in \mathbb{R}^{n_X \times r}$  and  $r \ll \min(n_X, n_Y)$ . The low-rank factors are then typically computed by iterative methods which never form the approximation  $UV^{\top}$  explicitly because they operate on the factors instead.

Consider a standard Sylvester equation  $AX + XB = D$ , with low-rank right hand side  $D$ . This case has been intensively researched and several efficient numerical algorithms for computing low-rank solution factors were developed [1, 3, 4, 30]. For general linear matrix equations, like the one we are faced with in (2.3), computing low-rank solution factors is significantly more demanding. Often, specialized low-rank methods are considered for problems having particular structures or properties [8, 2, 29, 10, 16, 28] (e.g.,  $\mathcal{B}, \mathcal{C}$  have to form a convergent splitting of  $\mathcal{A}$ ), which are not present in the problem (2.3) at hand. However, our problem *is* positive definite.

One possibility for dealing with general linear matrix equations are iterative matrix-valued Krylov subspace methods [8, 20, 2] that work with low-rank matrices instead of vectors. For rapid convergence of these methods, we typically need a preconditioner, whose selection is not always easily possible. In our problem, (2.3), these Krylov methods performed rather poorly, since we need to perform rank truncations after major subcalculations (updates of iterates, preconditioner and operator applications), which occur at every iteration step. Rank truncation can, e.g., be carried out via thin QR or SVD decompositions. However, computing these decompositions was very quickly getting too expensive because of the sheer amount of necessary rank truncations in the Krylov method.

An approach that does not have these drawbacks is a greedy method, as proposed by Kressner and Sirković [19], which is based on successive rank-1 approximations of the error. As first explained by Kressner and Sirković, this method has a number of advantages which we recap here. Each rank-1 approximation requires solving the original problem reduced to a column or row space only. This reduction inherits the sparsity of the original matrices (in our case,  $L_x$  and  $L_y$  are sparse), while reducing the unknown variable from a matrix to a vector. Hence, it can be solved efficiently using sparse direct linear solvers.

Approximating directly the error in the solution instead of the residual or Krylov vectors is beneficial for two reasons. First, the convergence of the low-rank approximation is governed by the smoothness of the solution, and not the condition number of the matrix  $A$  (2.2). The error-greedy method should converge dependent on only the condition numbers of the reduced systems. However, we can employ efficient and accurate solvers to these reduced systems, leading to good performance on the overall problem. Second, by improving the approximations only by rank-1 updates in every step, and by getting rid of the residual and Krylov vectors, we avoid the exponentially fast increasing column ranks in later iterations, and no expensive rank truncation operations are required. After computing a rank-1 increment, we refine the solution by solving the Galerkin projection onto the bases of all previous rank-1 greedy approximations in *both* rows and columns. This makes this extra Galerkin system small and well-conditioned, and thus amenable to unpreconditioned iterative methods. In turn, the Galerkin refinement reweights the sub-optimal components produced by the greedy steps in a global way. This improves the convergence rate and prevents the ranks from becoming unnecessarily high. Since we never assume any particular properties of the matrix expansion (2.2) besides sparsity of certain matrices, the greedy method is adequate for the general matrix equation (2.1).

### 2.3 Description and application of the greedy low-rank solver

Here we briefly review the algorithm from [19] and give information on how it is employed for handling our particular problem (2.3).

Starting from an approximate solution  $W_j \approx W$  of the linear matrix equation  $\mathcal{A}(W) = D$ , equation (2.3), we attempt to improve it by a update of rank one:  $W_{j+1} = W_j + u_{j+1}v_{j+1}^\top$ , where  $u_{j+1} \in \mathbb{R}^{n_y}$  and  $v_{j+1} \in \mathbb{R}^{n_x}$ . The update vectors  $u_{j+1}$ ,  $v_{j+1}$  are computed by minimizing a certain error functional. In our case, the matrix  $A$  associated to  $\mathcal{A}$  is positive definite and, thus, induces the  $\mathcal{A}$ -inner product  $\langle X, Y \rangle_{\mathcal{A}} = \text{Tr}(Y^\top \mathcal{A}(X))$  and the  $\mathcal{A}$ -norm  $\|Y\|_{\mathcal{A}} := \sqrt{\langle Y, Y \rangle_{\mathcal{A}}}$ . Hence,  $u_{j+1}, v_{j+1}$  can be determined by minimizing the squared



error in the  $\mathcal{A}$ -norm:

$$\begin{aligned}
(u_{j+1}, v_{j+1}) &= \arg \min_{u,v} \|W - W_j - uv^\top\|_{\mathcal{A}}^2 & (2.4) \\
&= \arg \min_{u,v} \text{Tr}((W - W_j - uv^\top)^\top \mathcal{A} (W - W_j - uv^\top)) \\
&= \arg \min_{u,v} \text{Tr}((W - W_j - uv^\top)^\top (D - \mathcal{A}(W_j) - \mathcal{A}(uv^\top))).
\end{aligned}$$

Discarding constant terms, noting that  $\langle X, Y \rangle_{\mathcal{A}} = \langle Y, X \rangle_{\mathcal{A}}$ , since we are dealing with real matrices, and setting  $R_j = D - \mathcal{A}(W_j)$  leads to

$$(u_{j+1}, v_{j+1}) = \arg \min_{u,v} \langle uv^\top, uv^\top \rangle_{\mathcal{A}} - 2 \text{Tr}(uv^\top R_j). \quad (2.5)$$

Notice that the rank-1 decomposition  $uv^\top$  is not unique: we can rescale the factors by any nonzero scalar  $c$  such that  $(uc)(v/c)^\top$  represents the same matrix. This reflects the fact that the optimization problem (2.5) is not convex. However, it is convex in each of the factors  $u$  and  $v$  separately.

We obtain the updates  $u_{j+1}, v_{j+1}$  via an alternating least squares (ALS) scheme [26]. First, a fixed  $v$  is used in (2.5) and a minimizing  $u$  is computed which is in the next stage kept fixed and (2.5) is solved for a minimizing  $v$ . Of course, the order of this calculation can be reversed. For a fixed vector  $v$  with  $\|v\| = 1$  the minimizing problem is

$$\begin{aligned}
\hat{u} &= \arg \min_u \langle uv^\top, uv^\top \rangle_{\mathcal{A}} - 2 \text{Tr}(uv^\top R_j) \\
&= \arg \min_u \left( \lambda \left( (u^\top u) v^\top L_x^2 v + 2(u^\top L_y v)(v^\top L_x v) + u^\top L_y^2 u \right) \right. \\
&\quad \left. + \sum_{\alpha=1}^{n_{\text{inj}}} (u^\top \text{diag}(\Omega_\alpha) u) (v^\top X_\alpha X_\alpha^\top v) \right) - 2u^\top R_j v
\end{aligned}$$

and, hence,  $\hat{u}$  is obtained by solving the linear system of equations

$$\hat{A} \hat{u} = R_j v, \quad \hat{A} := \lambda \left( (v^\top L_x^2 v) I + 2L_y (v^\top L_x v) + L_y^2 \right) + \sum_{\alpha=1}^{n_{\text{inj}}} \text{diag}(\Omega_\alpha) (v^\top X_\alpha X_\alpha^\top v). \quad (2.6a)$$

The second half iteration starts from the fixed  $u = \hat{u}/\|\hat{u}\|$  and tries to find a minimizing  $\hat{v}$  by solving

$$\hat{B} \hat{v} = R_j^\top u, \quad \hat{B} := \lambda \left( L_x^2 + 2L_x (u^\top L_y u) + (u^\top L_y^2 u) I \right) + \sum_{\alpha=1}^{n_{\text{inj}}} (u^\top \text{diag}(\Omega_\alpha) u) (X_\alpha X_\alpha^\top) \quad (2.6b)$$

which can be derived by similar steps. The linear systems (2.6a) and (2.6b) inherit the sparsity from  $L_x, L_y$  and  $\Omega$ . Therefore they can be solved by sparse direct or iterative methods. We chose to employ a sparse direct solver for (2.6a) as this was faster than other alternatives. The coefficient matrix  $\hat{B}$  in (2.6b) is the sum of a sparse (Laplacian terms) and a low-rank (rank  $n_{\text{inj}}$  data terms) matrix. In this case, we solve (2.6b) using the Sherman-Morrison-Woodbury formula [11] and a direct solver for the sparse inversion.

Both half steps form the ALS iteration which should be stopped when the iterates are close enough to a critical point. This, however, might be difficult to check. Here we propose a simpler approach compared to the one in [19]. Since we rescale  $u$  and  $v$  such that  $\|u\|_2 = \|v\|_2 = 1$ , the norm of the other factor is equal to the norm of the full matrix. In

other words,  $\|\hat{u}\|_2 = \|\hat{u}v^\top\|_2$  after solving for  $\hat{u}$ , and hence  $\|\hat{u}\|_2$  should converge to the norm of the exact solution. This motivates a simple criterion: we stop the ALS when

$$(1 - \delta)\|\hat{u}\|_2 \leq \|\hat{v}\|_2 \leq (1 + \delta)\|\hat{u}\|_2,$$

where  $\hat{u}$  and  $\hat{v}$  are taken from two consecutive ALS steps, and  $\delta < 1$  is a small threshold. It turns out that a relatively crude tolerance of  $\delta = 0.1$ , corresponding to 2–4 ALS iterations, is sufficient in practice for the overall convergence of the algorithm.

The second stage of the method is a non-greedy Galerkin refinement of the low-rank factors. Suppose a rank  $j$  approximation  $W_j = \sum_{i=1}^j u_i v_i^\top$  of  $W$  has been already computed. Let  $U \in \mathbb{R}^{n_Y \times j}$  and  $V \in \mathbb{R}^{n_X \times j}$  have orthonormal columns, spanning the spaces  $\text{span}\{u_1, \dots, u_j\}$  and  $\text{span}\{v_1, \dots, v_j\}$ , respectively. We compute a refined approximation  $UZV^\top$  for  $Z \in \mathbb{R}^{j \times j}$  by imposing a Galerkin condition onto the residual:

$$\text{Find } Z \text{ so that } \quad \mathcal{A}(UZV^\top) - D \quad \perp \quad \{UZV^\top \in \mathbb{R}^{n_Y \times n_X}, \quad Z \in \mathbb{R}^{j \times j}\}.$$

This leads to the dense, square matrix equation in  $Z$

$$\begin{aligned} \lambda (Z(V^\top L_x^2 V) + 2(U^\top L_y U)Z(V^\top L_x V) + (U^\top L_y^2 U)Z) \\ + \sum_{\alpha=1}^{n_{\text{inj}}} (U^\top \text{diag}(\Omega_\alpha)U)Z(V^\top X_\alpha X_\alpha^\top V) = U^\top D V \end{aligned} \quad (2.7)$$

of dimension  $j \leq r \ll n_X, n_Y$ .

Equation (2.7) is a projected version of (2.3) and inherits its structure, especially the positive definiteness of the defining operator. Instead of using a direct method to solve (2.7) as in [19], we employ an iterative method similarly to [28]. Due to the positive definiteness, the obvious method of choice is a dense, matrix-valued conjugate gradient method (CG). Moreover, we reduce the number of iterations significantly by taking the solution  $Z$  from the previous greedy step as an initial guess. The improved solution  $W_{j+1} = UZV^\top$  yields a new residual  $R_{j+1} = D - \mathcal{A}(W_{j+1})$  onto which the ALS scheme is applied to obtain the next rank-1 updates. As shown in [19], this Galerkin refinement will typically substantially improve the greedy approximation, leading to a faster convergence rate. We can say that the ALS scheme is primarily used to sketch the projection bases for the Galerkin solution, which justifies the limited number of ALS steps. The complete procedure is illustrated in Algorithm 1.

### 3 Performance of the greedy low-rank solver on three problems

There are three test problems to which we apply Algorithm 1: a toy problem with synthetic data (Section 3.1), the top view projected mouse connectivity data (Section 3.2), and the flatmap projected data (Section 3.3).

The goal of this study is to investigate the computational complexity and convergence of the greedy algorithm. Since the matrices in (2.6a) are sparse, the ALS steps need  $\mathcal{O}(nr^2n_{\text{inj}})$  operations in total for the final solution rank  $r$ , where  $n = \max(n_X, n_Y)$ . In turn, if the solution of (2.7) takes  $\gamma$  CG iterations, this step will have a cost of  $\mathcal{O}(\gamma r^3 n_{\text{inj}})$ . Although  $\gamma$  can be kept at the same level for all  $j$ , it depends on the stopping tolerance  $\tau$ , as does the

---

**Algorithm 1:** Greedy rank-1 method with Galerkin projection for (2.3)

---

**Input** : Matrices  $L_x, L_y, X, \Omega, Y$  defining (2.3), maximal allowed rank  $r$ , tolerance  $0 < \tau \ll 1$

**Output:** Low-rank approximation of  $W = UZV^\top$  in factored form

1 Initialize  $W_0 = 0, R_0 = D, U_0 = V_0 = [\ ]$ ,  $j = 0$

2 **repeat**

3 | Pick initial vector  $v$  for ALS with  $\|v\| = 1$ , then get rank-1 update:

4 | **while**  $\delta > 0.1$  **do**

5 |     Solve  $\hat{A}\hat{u} = R_j v$  for  $\hat{u}$  and set  $u = \hat{u}/\|\hat{u}\|$  *Eqn. (2.6a)*

6 |     Solve  $\hat{B}\hat{v} = R_j^\top u$  for  $\hat{v}$  and set  $v = \hat{v}/\|\hat{v}\|$  *Eqn. (2.6b)*

7 |      $\delta = \left| \frac{\|\hat{u}\|}{\|\hat{v}\|} - 1 \right|$

8 |      $U_{j+1} = \text{orth}([U_j, u]), V_{j+1} = \text{orth}([V_j, v])$  *Orthogonalize new factors*

9 |     Increment rank  $j \leftarrow j + 1$

10 |     Solve Eqn. (2.7) for  $Z_j$  using conjugate gradients *Galerkin update*

11 |      $R_j = D - \mathcal{A}(U_j Z_j V_j^\top)$  *Update residual*

12 |      $\delta W = \|U_j Z_j V_j^\top - U_{j-1} Z_{j-1} V_{j-1}^\top\|_F / \|U_j Z_j V_j^\top\|_F$

13 **until**  $j = r$  or  $\delta W \leq \tau$

---

rank  $r$ . We will therefore investigate the cost in terms of the total computation time and the corresponding solution accuracy for a range of solution rank values.

The numerical experiments were performed on an Intel<sup>®</sup> E5-2650 v2 CPU with 8 threads and 64Gb RAM, as well as a NVIDIA<sup>®</sup> P100 GPU card for certain subtasks. In particular, the Galerkin update relies primarily on dense linear algebra for solving (2.7) by means of the CG method, such that this stage admits an efficient GPU implementation. We use MATLAB<sup>®</sup> R2017b for all computations in Algorithm 1, running on the Balena High Performance Computing (HPC) Service at the University of Bath. See Section 5 for additional data and code resources.

Throughout the experiments, we measure errors in the solution using the root mean squared error. Given any reference solution  $W_\star$  of size  $n_Y \times n_X$ , e.g. the truth or a full-rank solution when the truth is unknown, and a low-rank solution  $W_r$ , the RMS error is computed as follows,

$$\mathcal{E}(W_r, W_\star) = \frac{\|W_r - W_\star\|_F}{\sqrt{n_Y n_X}}. \quad (3.1)$$

Also, we report the relative error in the Frobenius norm,

$$\mathcal{E}_{\text{rel}}(W_r, W_\star) = \frac{\|W_r - W_\star\|_F}{\|W_\star\|_F}. \quad (3.2)$$

### 3.1 Test problem: a toy brain

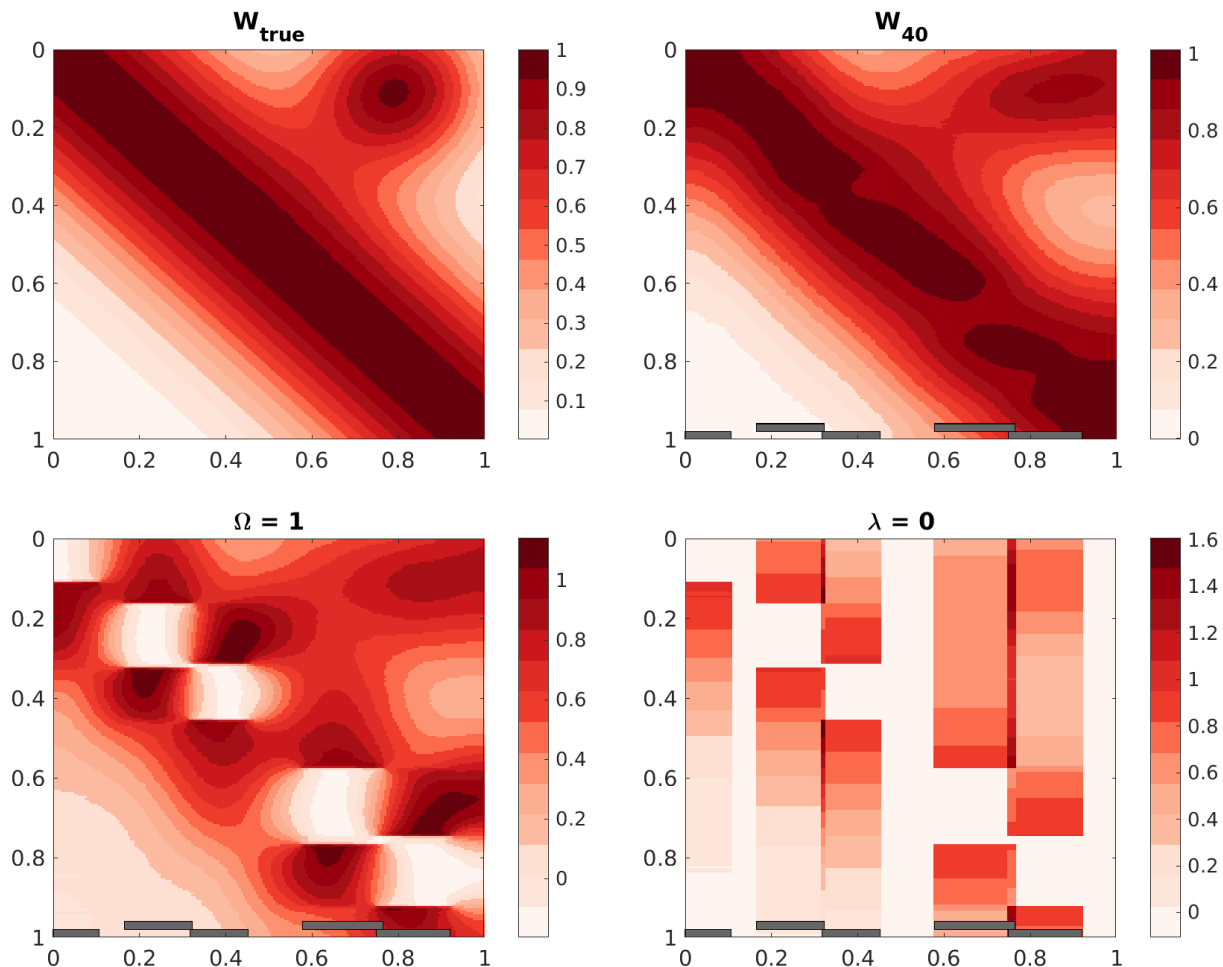
We use the same test problem as in [15], a one-dimensional ‘‘toy brain.’’ The source and target space are  $S = T = [0, 1]$ . The true connectivity kernel corresponds to a Gaussian profile about the diagonal plus an off-diagonal bump:

$$W_{\text{true}}(x, y) = \exp \left\{ - \left( \frac{x - y}{0.4} \right)^2 \right\} + 0.9 \exp \left\{ - \frac{(x - 0.8)^2 + (y - 0.1)^2}{(0.2)^2} \right\}. \quad (3.3)$$

Table 3.1: Computing times and errors for the toy brain test problem. The output  $W$  is compared to truth and a rank 140 reference solution. In this case, the truth  $W_{\text{true}}$  is known from (3.3).

| rank( $W$ )                                    | 10         | 20         | 40         | 60         | 80         |
|--|------------|------------|------------|------------|------------|
| CPU time (sec.)                                | 0.0396     | 0.1554     | 0.9653     | 2.6398     | 3.1108     |
| $\mathcal{E}(W, W_{140})$                      | 3.2324e-01 | 5.5407e-02 | 1.4162e-02 | 1.2125e-03 | 3.1549e-04 |
| $\mathcal{E}(W, W_{\text{true}})$              | 2.9418e-01 | 7.9921e-02 | 7.1537e-02 | 6.9777e-02 | 6.9821e-02 |
| $\mathcal{E}_{\text{rel}}(W, W_{140})$         | 4.3320e-01 | 8.9700e-02 | 2.4900e-02 | 2.5000e-03 | 5.1300e-04 |
| $\mathcal{E}_{\text{rel}}(W, W_{\text{true}})$ | 4.0130e-01 | 1.1410e-01 | 1.0350e-01 | 1.0040e-01 | 1.0040e-01 |

Figure 3.1: Top: true connectivity map  $W_{\text{true}}$  (left) and the low-rank solution for the test problem with rank = 40 and  $\lambda = 100$  (right). Bottom: solutions with  $\Omega = 1$  (left) and  $\lambda = 0$  (right). The locations of simulated injections are shown by the grey bars.



See Fig. 3.1. The input and output spaces were discretized using  $n_X = n_Y = 200$  uniformly spaced lattice points. Injections are delivered at  $n_{\text{inj}} = 5$  approximately evenly-spaced locations in  $S$ , with a width of  $0.12 + 0.1\epsilon$ , where  $\epsilon \sim \text{Uniform}(0, 1)$ . The values of  $X$  are set to 1 within the injection region and 0 elsewhere,  $Y$  is set to 0 within the injection region, and we add Gaussian noise with standard deviation  $\sigma = 0.1$ . The matrices  $L_x = L_y$  are the 5-point finite difference Laplacians for the rectangular lattice.

In Fig. 3.1 we depict the true toy connectivity  $W_{\text{true}}$  as well as a number of low-rank solutions output by our method. As in [15], we show that both the mask and regularization are required for good performance. If we remove the mask, setting  $\Omega$  equal to the matrix of

Figure 3.2: Computing times and errors for the top view data.

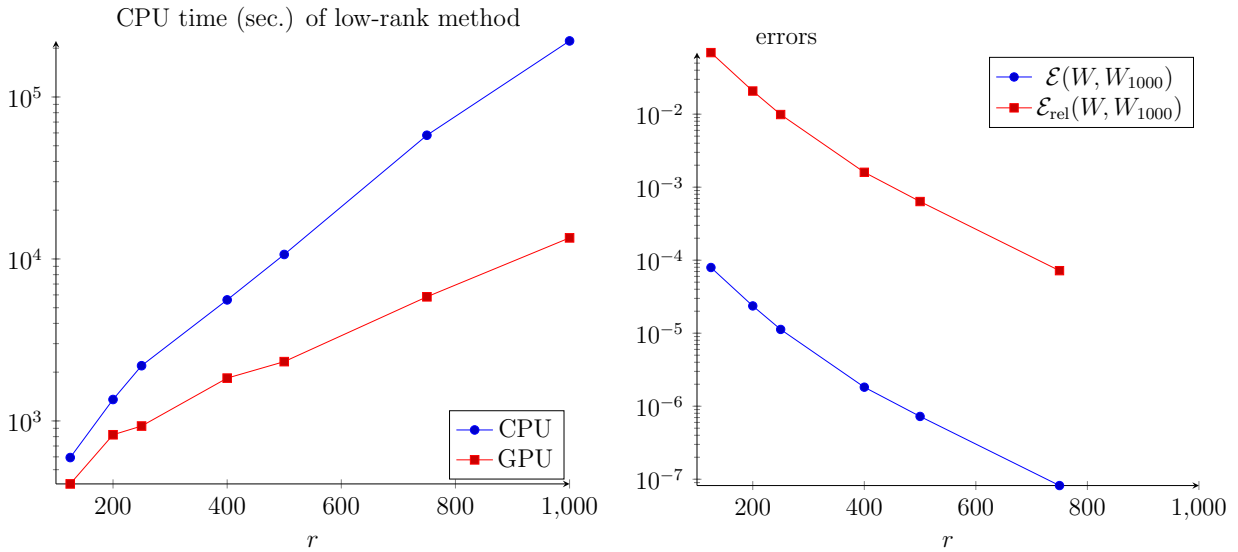
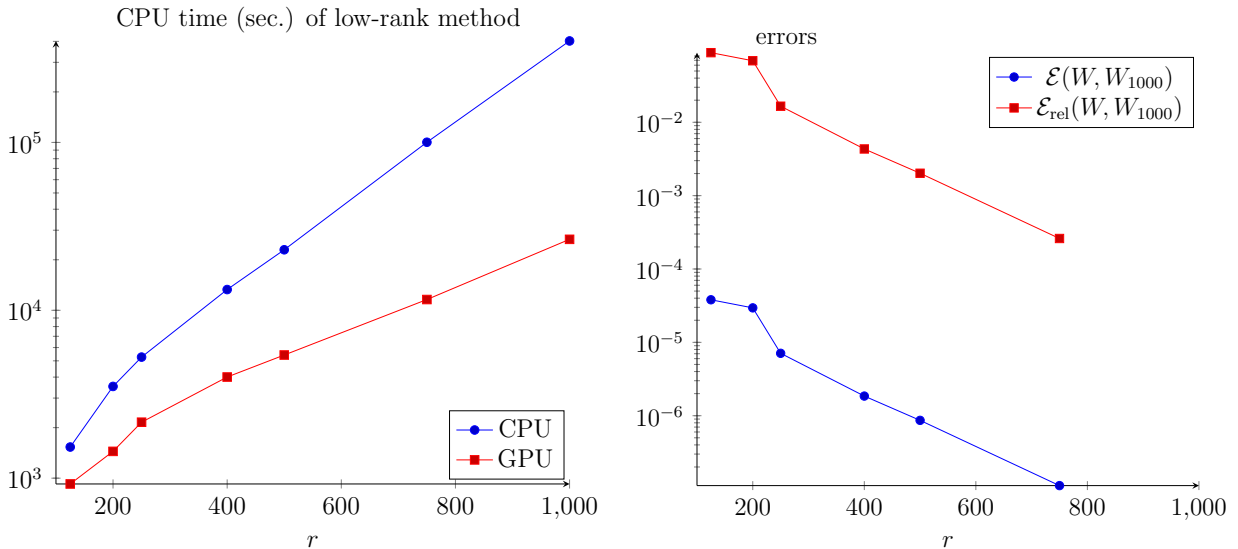


Figure 3.3: Computing times and errors for the flatmap data.



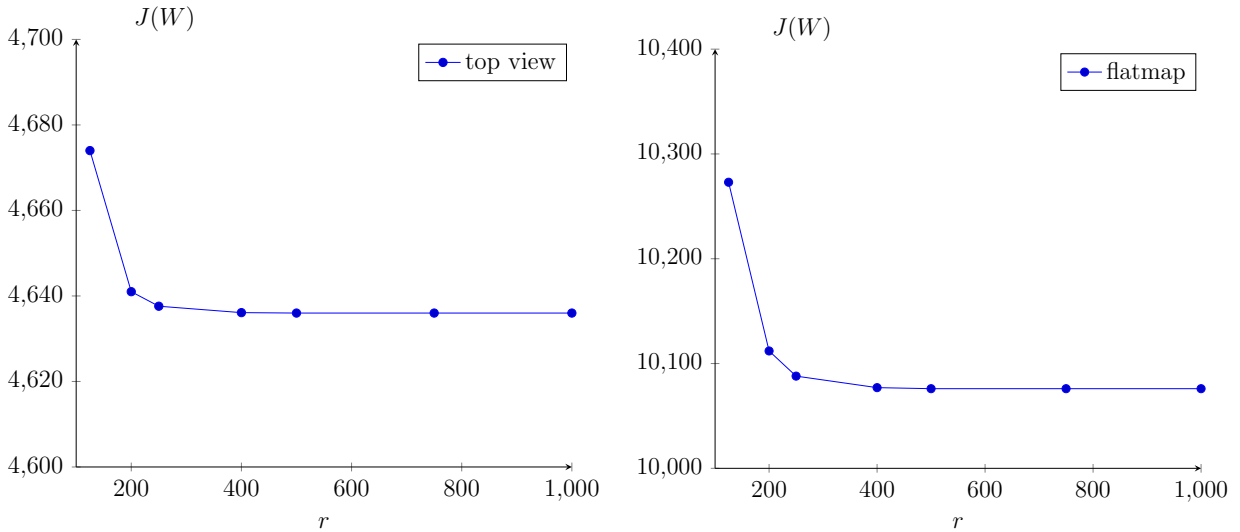
all ones, then there are holes in the data at the location of the injections. If we try fitting with  $\lambda = 0$ , i.e. no smoothing, then the method cannot fill in holes or extrapolate outside the injection sites. It is only with the combination of all ingredients (a mask that reflects the injection sites and in this case  $\lambda = 100$ ), that we recover the true connectivity.

In Tab. 3.1 we show the performance of the algorithm for ranks 10, 20, 40, 60, and 80. The output  $W$  is compared to  $W_{\text{true}}$  as well as the rank 140 output of the algorithm. We see that the RMS distance to the reference solution  $W_{140}$  decreases as we increase the rank, as does the relative distance. However, the RMS and relative distances from  $W_{\text{true}}$  asymptote to roughly 0.07 and 10%, respectively, by rank 40. This shows that rank 40 is a suitable maximum rank for this problem given the standard deviation of the problem.

### 3.2 Mouse cortex: top view connectivity

We next benchmark the low-rank greedy solver on mouse cortical data projected into a top-down view. See Section 5 for details about how we obtained these data. Here, the problem

Figure 3.4: Cost functions of the low-rank solutions



sizes are  $n_Y = 44\,478$  and  $n_X = 22\,377$  and the number of injections  $n_{\text{inj}} = 126$ . We use the smoothing parameter  $\tilde{\lambda} = 10^6$ .

We run the low-rank solver with the target solution rank varying from 125 to 1000. The stopping tolerances for the inner PCG iteration for (2.7) were decreased geometrically from  $10^{-3}$  for  $r = 125$  to  $10^{-6}$  for  $r = 1000$ . This delivers accurate but cheap solution to the Galerkin system (2.7).

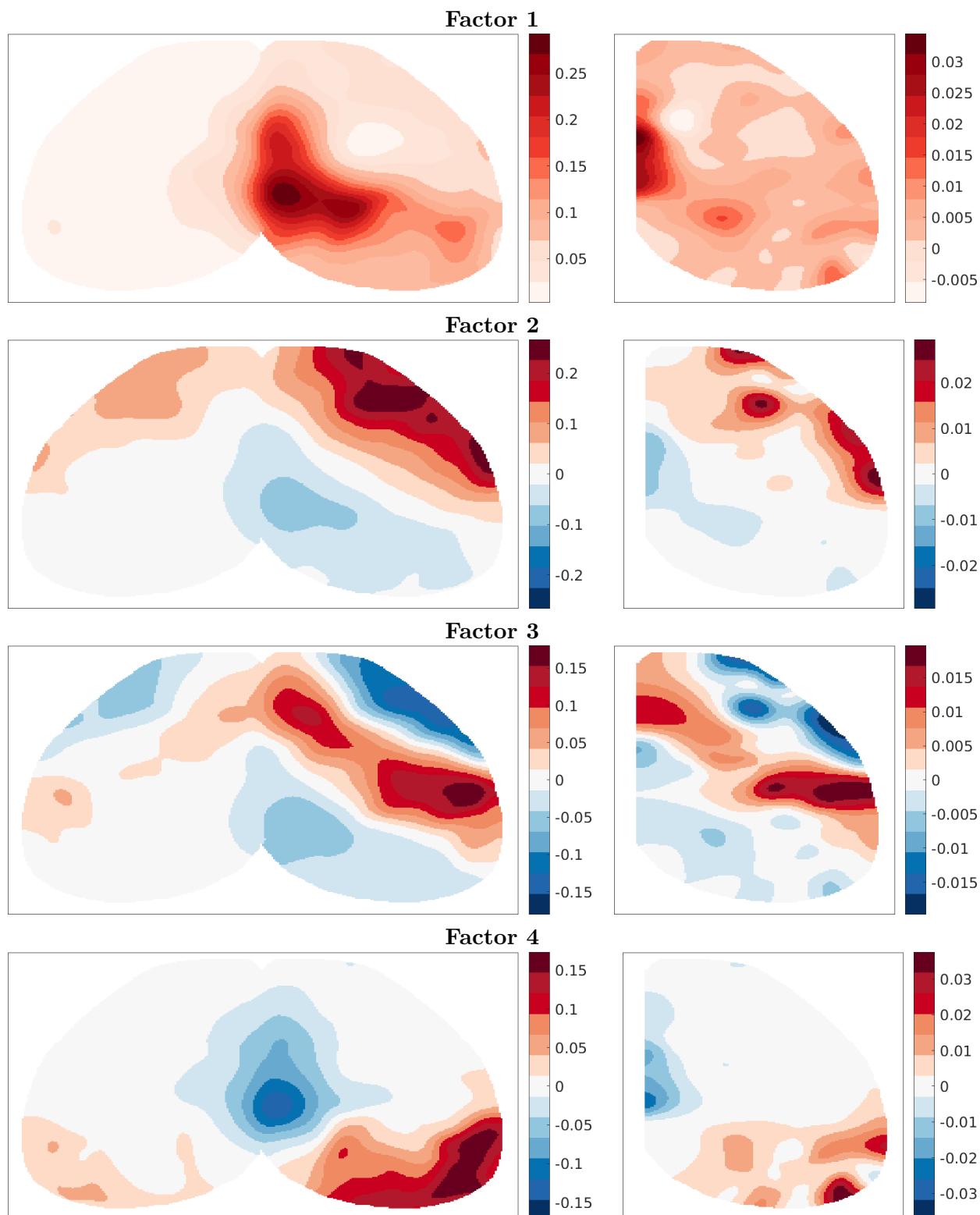
These low-rank solutions are compared to the full full-rank solution  $W_{\text{full}}$  (i.e. rank =  $n_X = 22\,377$ ) produced by an iterative, gradient-based method. Note that this full rank algorithm was initialized from the output of the low-rank algorithm. This led to a very significant speedup; the full rank method had not reached a considerable value of the cost function (1.4) after multiple *days* of computation.

The computing times and errors are presented in Fig. 3.2. We see that the RMS errors are relatively small for ranks above 500, below  $10^{-6}$ . Neither the RMS or relative error seem to have plateaued at rank 1000, but they are small. At rank 1000, the vector  $\ell_\infty$  error (maximum absolute deviation of the matrices as vectors, not the matrix  $\infty$ -norm)  $\|W_{1000} - W_{\text{full}}\|_\infty$  is less than  $10^{-6}$ , which is certainly within experimental uncertainty.

Note that the output of the algorithm is  $W_r = UZV^\top$ . We perform a final SVD of the Galerkin matrix,  $Z = \tilde{U}\Sigma\tilde{V}^\top$  and set  $\hat{U} = U\tilde{U}$  and  $\hat{V} = V\tilde{V}$ , so that  $W_r = \hat{U}\Sigma\hat{V}^\top$  is the SVD of the output.

The first four of these singular vectors of the solution are shown in Fig. 3.5. Note that the orientation here has the medial to lateral axis aligned from left to right and anterior to posterior aligned from top to bottom, as in a transverse slice. The midline of the cortex is in the center of the target plots, whereas it is on the left edge of the source plots. We observe that the leading rank-1 component is a strong projection from medial areas of the cortex near the midline to nearby locations. The second component provides a correction which adds local connectivity among posterior areas and anterior areas. Note that increased anterior connectivity arises from negative entries in both source and target vectors. The sign change along the roughly anterior-posterior axis manifests as a reduction in connectivity from anterior to posterior regions as well as from posterior to anterior regions. The third component is a strong local connectivity among somatomotor areas located medially along the anterior-posterior axis and stronger on the lateral side where the barrel fields, important sensory areas for whisking, are located. Finally, the fourth component is concentrated in

Figure 3.5: Top four singular vectors of the top view connectivity with  $r = 500$ . Left: scaled target vectors  $\hat{U}\Sigma$ . Right: source vectors  $\hat{V}$ .



posterior locations, mostly corresponding to the visual areas, as well as more anterior and medial locations in the retrosplenial cortex (thought to be a memory and association area). The visual and retrosplenial parts of the component show opposite signs, reflecting stronger local connectivity within these regions than distal connectivity between them.

The patterns we observe in Fig. 3.5 are reasonable, since connectivity in the brain is

dominantly local with some specific long-range projections. We also observe that the projection patterns (left components  $\hat{U}\Sigma$ ) are fairly symmetric across the midline. This is also expected due to the mirroring of major brain areas in both hemispheres, despite the evidence for some lateralization, especially in humans. The more specific projections between brain regions will show up in later, higher frequency components. However, it becomes increasingly difficult to interpret lower energy components as specific pathways, since these combine in complicated ways.

### 3.3 Mouse cortex: flatmap connectivity

Finally, we test the method on another problem which is a flatmap projection of the brain (see Section 5 for details). This projection more faithfully represents areas of the cortex which are missing from the top view since they curl underneath that vantage point. The flatmap is closer to the kind of transformation used by cartographers to flatten the globe, whereas the top view is like a satellite image taken far from the globe.

The problem size is now larger by roughly a factor of three relative to the top view. Here,  $n_Y = 126\,847$  and  $n_X = 63\,435$ . The number of experiments is the same,  $n_{\text{inj}} = 126$ , whereas the regularization parameter is set to  $\tilde{\lambda} = 3 \times 10^7$ . The smoothing parameter was set to give the same level of smoothness, measured “by eye,” in the components as  $\tilde{\lambda} = 10^6$  in the top view experiment.

In this case, the computing time of the full solver would be excessively large, so we do not estimate the error by comparison to the full solution. Instead, we take the low-rank solution with  $r = 1000$  as the reference solution  $W_\star = W_{1000}$ . The computing times and the errors are shown in Fig. 3.3.

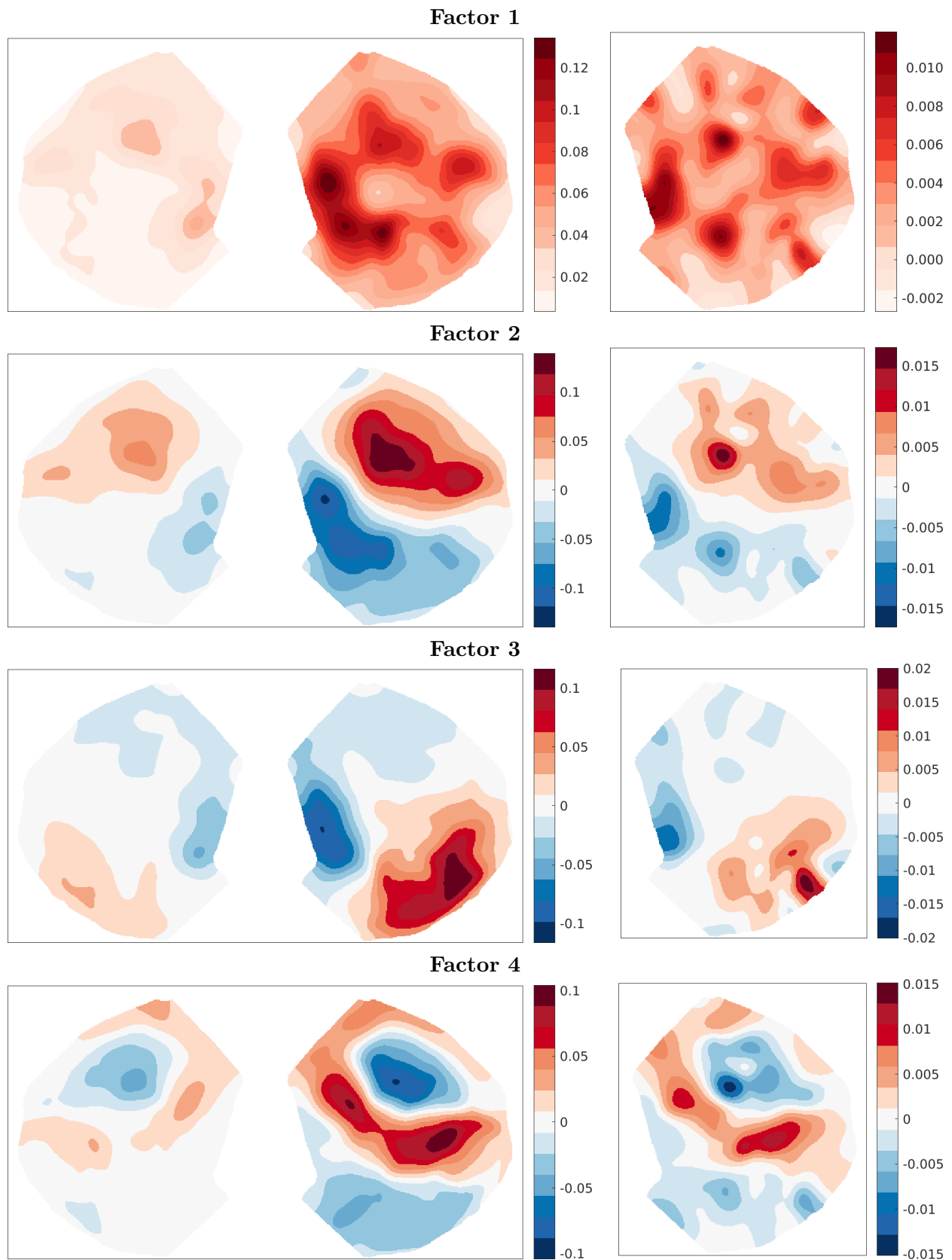
Again, we see that by  $r = 500$  the distance from  $W_\star$  is smaller than 10%. Furthermore, the RMS error between rank 500 and 1000 is again less than  $10^{-6}$ , so we believe rank 500 is probably a very good approximation to the full solution. Here, the benefits by using the GPU implementation for solving (2.7) were more significant than for the top view case. We obtained the rank 500 solution in approximately 1.5 hours, which is significantly less than pure CPU implementation, which took 6.4 hours. Comparing Figs. 3.2 and 3.3, the computation times for the flatmap problem with  $r = 500$  and 1000 are roughly twice as large as for the top view problem. On the other hand, for  $r = 125$  and 250, the compute times are about three times as long for flatmap versus top view. The observed scaling in compute time appears slightly slower than  $\mathcal{O}(n)$  in these tests.

The four dominant singular vectors of the flatmap solution are shown in Fig. 3.6. The orientation is the same as in Fig. 3.5, with the anterior-posterior axis from top to bottom and the medial-lateral axis from left to right. The first two rank-1 components are directly comparable between the two problem outputs, although we see more structure in the flatmap components. This could be due to employing a projection which more accurately represents these 3-D data in 2-D, or due to the choice of smoothing parameter  $\lambda$ . The third and fourth components, on the other hand, are comparable to the fourth and third components in the top view problem, respectively.

Again, all of these patterns are reasonable and expected. The raw 3-D data that were fed into the top view and flatmap projections were the same, but the greedy algorithm is run using different projected datasets. It is reassuring that we can interpret the first few factors and directly compare them against those in the top view.



Figure 3.6: Top four singular vectors of the flatmap connectivity with  $r = 500$ . Left: scaled target vectors  $\hat{U}\Sigma$ . Right: source vectors  $\hat{V}$ .



## 4 Discussion

We have studied a numerical method specifically tailored for the important neuroscience problem of connectome regression from mesoscopic tract tracing experiments. This connectome inference problem was formulated as a regularized, multivariate, structured regression problem, the convex program (1.4). The optimality conditions for this problem turn out to be a linear matrix equation in the unknown connectivity  $W$ . Our numerical results show that the low-rank greedy algorithm, as proposed by Kressner and Sirković [19], is a viable choice for acquiring low-rank factors of  $W$  with a computation cost that was significantly smaller compared to other approaches [15, 2, 20]. This allows us to infer the flatmap matrix, with approximately  $140\times$  more entries than previously obtained for the visual system, while taking less time<sup>2</sup>. The first few components of these 2-D cortical connectivity matrices are interpretable and reasonable, although a full anatomical study of this inferred connectivity is outside the scope of the current paper.

The major required ingredients for Algorithm 1 were: (1) solving large, sparse linear systems of equations at each ALS iteration, and (2) solving a much smaller, dense, projected version (2.7) of the original linear matrix equation (2.3) for the Galerkin step. The last subtask can be seen as a numerical bottleneck of the algorithm because of the cubic complexity required for solving (2.7) and also due to the absence of direct numerical methods to handle dense, moderately sized general linear matrix equations. Hence, any progress in this direction could improve the workload coming from the Galerkin acceleration. Here, we employed a matrix valued CG iteration for approximately solving (2.7) that was carried out on a GPU to decrease the computation time of this stage. One could, of course, also argue that equipping this CG iteration with a preconditioner might speed up its convergence, but so far we were not successful in finding a preconditioner that both reduced the number of CG steps and the computational time. A potential further research direction might therefore be to derive an adequate preconditioning strategy for the particular problem structure at hand in (2.3), that would increase the efficiency of the employed Krylov method. Of course, if such preconditioner would be found, it would also be possible to give low-rank matrix-valued Krylov methods [8, 20, 2] directly applied to the large-scale problem (2.3) another try.

The original regression problem proposed by Harris et al. [15], problem (1.1), demands that the solution  $W$  be nonnegative. So far, this constraint is not considered by the employed algorithm. However, for the test problem and data we have tried, the computed matrix turns out to be majority nonnegative<sup>3</sup>. We find typically small negative entries that could be safely neglected without sacrificing the accuracy. Although a mostly nonnegative solution is not generally expected when solving the unconstrained problem (1.4), it appears that such behavior is typical for nonnegative data matrices  $X$  and  $Y$ .

Working directly with nonnegative factors  $U \geq 0$  and  $V \geq 0$  was originally proposed by Harris et al. [15], where they applied a projected gradient method to find such an approximation for connectome of mouse visual areas. Such a formulation would be very useful, since it would lead to a nonnegative  $W$  and would allow interpreting each factor as a combination of neural pathways. This is analogous to how nonnegative matrix factorization (NNMF) is interpretable as a clustering [9]. However, as we started to apply the projected

---

<sup>2</sup> We found the flatmap solution in hours versus the days it took to find the low-rank visual matrix presented in [15].

<sup>3</sup> In the top view problem, comparing the nonnegative versus unconstrained solutions, (1.1) versus (1.4), we see that  $\mathcal{E}(W_{\text{full}}, W_{\text{nonneg}}) = 3.99\text{e-}04$ . Projecting  $W_{\text{full}}$  onto the nonnegative orthant leads to  $\mathcal{E}(W_{\text{proj}}, W_{\text{nonneg}}) = 3.67\text{e-}04$ . In either case the  $\infty$ -norm difference is 0.009.

gradient approach to much larger problems, slow convergence as a result of nonlocality and ill-conditioning made such an algorithm ineffective.

Modifying Algorithm 1 to compute nonnegative low-rank factors  $U$  and  $V$  or ensuring that the low-rank approximation  $UV^\top \approx W$  is nonnegative, which is a nonlinear constraint, is a much harder goal to achieve. The bottleneck is again the Galerkin step. For instance, even if one generated nonnegative factor matrices  $U$  and  $V$ , e.g. by nonnegative alternating least squares, the orthogonalization phase for the Galerkin update, and the update itself, would destroy the nonnegativity. However, new methods of NNMF, many of which incorporate regularizations similar to our Laplacian terms [7, 6], are an area of ongoing research. These include other techniques developed with neuroscience in mind, such as neuron segmentation and calcium deconvolution [27] as well as sequence identification [22]. We hope to study nonnegative, low-rank methods in the future.

## 5 Data and code

We tested out algorithm on two datasets (top view and flatmap) generated from Allen Institute for Brain Science Mouse Connectivity Atlas data <http://connectivity.brain-map.org>. These data were obtained with the Python SDK `allensdk` version 0.13.1 available from <http://alleninstitute.github.io/AllenSDK/>. Our data pulling and processing scripts are available from <https://github.com/kharris/allen-voxel-network>.

We used the `allensdk` to retrieve 10  $\mu\text{m}$  injection and projection density volumetric data for 126 *wildtype* experiments in cortex. These data were projected from 3-D to 2-D using either the top view or flatmap paths and saved as 2-D arrays. Next, the projected coordinates were split into left and right hemispheres. Since *wildtype* injections were always delivered into the right hemisphere, this becomes our source space  $S$  whereas the union of left and right are the target space  $T$ . We constructed 2-D finite difference 5-point Laplacian matrices on these grids. Finally, the 2-D projected data were downsampled  $4\times$  along each dimension to obtain 40  $\mu\text{m}$  resolution. The injection and projection data were then stacked into the matrices  $X$  and  $Y$ , respectively. The mask  $\Omega$  was set via  $\Omega_{ij} = 1_{\{X_{ij} \leq 0.4\}}$ .

MATLAB code which implements our greedy low-rank algorithm (1) is included in the repository: [https://gitlab.mpi-magdeburg.mpg.de/kuerschner/lowrank\\_connectome](https://gitlab.mpi-magdeburg.mpg.de/kuerschner/lowrank_connectome). We also include the problem inputs  $X, Y, L_x, L_y, \Omega$  for our three example problems (test, top view, and flatmap) as MATLAB files. Note that  $\Omega$  is stored as  $1 - \Omega$  in these files, as this matches the convention of [15].

## Acknowledgements

We would like to thank Lydia Ng, Nathan Gouwens, Stefan Mihalas, Nile Graddis and others at the Allen Institute for the top view and flatmap paths and general help accessing the data. Thank you to Braden Brinkman discussions of the continuous problem. Thank you to Stefan Mihalas and Eric Shea-Brown for discussions of connectivity regression and the suitability or not of low-rank formulations. KDH was supported by the Big Data for Genomics and Neuroscience NIH training grant. SD is thankful to the Engineering and Physical Sciences Research Council (UK) for their support through Fellowship EP/M019004/1, and the kind hospitality of the Erwin Schrödinger International Institute for Mathematics and Physics (ESI), where part of this research was developed under the frame of the Thematic Programme *Numerical Analysis of Complex PDE Models in the Sciences*.

## References

- [1] P. BENNER, *Solving Large-Scale Control Problems*, IEEE Control Syst. Mag., 14 (2004), pp. 44–59.
- [2] P. BENNER AND T. BREITEN, *Low rank methods for a class of generalized Lyapunov equations and related issues*, Numer. Math., 124 (2013), pp. 441–470, <https://doi.org/10.1007/s00211-013-0521-0>.
- [3] P. BENNER, R.-C. LI, AND N. TRUHAR, *On the ADI Method for Sylvester Equations*, J. Comput. Appl. Math., 233 (2009), pp. 1035–1045.
- [4] P. BENNER AND J. SAAK, *Numerical solution of large and sparse continuous time algebraic matrix Riccati and Lyapunov equations: a state of the art survey*, GAMM Mitteilungen, 36 (2013), pp. 32–52, <https://doi.org/10.1002/gamm.201310003>.
- [5] M. BOTA, H.-W. DONG, AND L. W. SWANSON, *From Gene Networks to Brain Networks*, Nature Neuroscience, 6 (2003), pp. 795–799, <https://doi.org/10.1038/nn1096>.
- [6] D. CAI, X. HE, J. HAN, AND T. S. HUANG, *Graph Regularized Nonnegative Matrix Factorization for Data Representation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 33 (2011), pp. 1548–1560, <https://doi.org/10.1109/TPAMI.2010.231>.
- [7] A. CICHOCKI, R. ZDUNEK, A. H. PHAN, AND S.-I. AMARI, *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-Way Data Analysis and Blind Source Separation*, John Wiley & Sons, July 2009.
- [8] T. DAMM, *Direct methods and ADI-preconditioned Krylov subspace methods for generalized Lyapunov equations*, Numer. Lin. Alg. Appl., 15 (2008), pp. 853–871.
- [9] C. DING, X. HE, AND H. SIMON, *On the Equivalence of Nonnegative Matrix Factorization and Spectral Clustering*, in Proceedings of the 2005 SIAM International Conference on Data Mining, Proceedings, Society for Industrial and Applied Mathematics, Apr. 2005, pp. 606–610.
- [10] E. RINGH, G. MELE, J. KARLSSON, E. JARLEBRING, *Sylvester-based preconditioning for the waveguide eigenvalue problem*, Linear Algebra Appl., 542 (2018), pp. 441–463.
- [11] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, Johns Hopkins University Press, Baltimore, fourth ed., 2013.
- [12] L. GRASEDYCK, *Existence of a low rank or H-matrix approximant to the solution of a Sylvester equation*, Numer. Lin. Alg. Appl., 11 (2004), pp. 371–389.
- [13] S. GRILLNER, N. IP, C. KOCH, W. KOROSHETZ, H. OKANO, M. POLACHEK, M.-M. POO, AND T. J. SEJNOWSKI, *Worldwide Initiatives to Advance Brain Research*, Nature Neuroscience, (2016), <https://doi.org/10.1038/nn.4371>.
- [14] R. GĂMĂNUȚ, H. KENNEDY, Z. TOROCZKAI, M. ERCSEY-RAVASZ, D. C. VAN ESSEN, K. KNOBLAUCH, AND A. BURKHALTER, *The Mouse Cortical Connectome, Characterized by an Ultra-Dense Cortical Graph, Maintains Specificity by Distinct Connectivity Profiles*, Neuron, 97 (2018), pp. 698–715.e10, <https://doi.org/10.1016/j.neuron.2017.12.037>.

- [15] K. D. HARRIS, S. MIHALAS, AND E. SHEA-BROWN, *High Resolution Neural Connectivity from Incomplete Tracing Data Using Nonnegative Spline Regression*, in *Neural Information Processing Systems*, 2016.
- [16] E. JARLEBRING, G. MELE, D. PALITTA, AND E. RINGH, *Krylov methods for low-rank commuting generalized Sylvester equations*, *Numer. Lin. Alg. Appl.*, (2018), <https://doi.org/10.1002/nla.2176>.
- [17] A. JENETT, G. M. RUBIN, T.-T. B. NGO, D. SHEPHERD, C. MURPHY, H. DIONNE, B. D. PFEIFFER, A. CAVALLARO, D. HALL, J. JETER, N. IYER, D. FETTER, J. H. HAUSENFLUCK, H. PENG, E. T. TRAUTMAN, R. R. SVIRSKAS, E. W. MYERS, Z. R. IWINSKI, Y. ASO, G. M. DEPASQUALE, A. ENOS, P. HULAMM, S. C. B. LAM, H.-H. LI, T. R. LAVERTY, F. LONG, L. QU, S. D. MURPHY, K. ROKICKI, T. SAFFORD, K. SHAW, J. H. SIMPSON, A. SOWELL, S. TAE, Y. YU, AND C. T. ZUGATES, *A GAL4-Driver Line Resource for Drosophila Neurobiology*, *Cell Reports*, 2 (2012), pp. 991–1001, <https://doi.org/10.1016/j.celrep.2012.09.011>.
- [18] J. E. KNOX, K. D. HARRIS, N. GRADDIS, J. D. WHITESELL, H. ZENG, J. A. HARRIS, E. SHEA-BROWN, AND S. MIHALAS, *High Resolution Data-Driven Model of the Mouse Connectome*, *bioRxiv*, (2018), p. 293019, <https://doi.org/10.1101/293019>.
- [19] D. KRESSNER AND P. SIRKOVIĆ, *Truncated low-rank methods for solving general linear matrix equations*, *Numer. Lin. Alg. Appl.*, 22 (2015), pp. 564–583, <https://doi.org/10.1002/nla.1973>.
- [20] D. KRESSNER AND C. TOBLER, *Krylov Subspace Methods for Linear Systems with Tensor Product Structure*, *SIAM J. Matrix Anal. Appl.*, 31 (2010), pp. 1688–1714.
- [21] E. S. LEIN, M. J. HAWRYLYCZ, N. AO, M. AYRES, A. BENSINGER, A. BERNARD, A. F. BOE, M. S. BOGUSKI, K. S. BROCKWAY, E. J. BYRNES, L. CHEN, L. CHEN, T.-M. CHEN, M. C. CHIN, J. CHONG, B. E. CROOK, A. CZAPLINSKA, C. N. DANG, S. DATTA, N. R. DEE, A. L. DESAKI, T. DESTA, E. DIEP, T. A. DOLBEARE, M. J. DONELAN, H.-W. DONG, J. G. DOUGHERTY, B. J. DUNCAN, A. J. EBBERT, G. EICHELE, L. K. ESTIN, C. FABER, B. A. FACER, R. FIELDS, S. R. FISCHER, T. P. FLISS, C. FRENSLEY, S. N. GATES, K. J. GLATTFELDER, K. R. HALVERSON, M. R. HART, J. G. HOHMANN, M. P. HOWELL, D. P. JEUNG, R. A. JOHNSON, P. T. KARR, R. KAWAL, J. M. KIDNEY, R. H. KNAPIK, C. L. KUAN, J. H. LAKE, A. R. LARAMEE, K. D. LARSEN, C. LAU, T. A. LEMON, A. J. LIANG, Y. LIU, L. T. LUONG, J. MICHAELS, J. J. MORGAN, R. J. MORGAN, M. T. MORTRUD, N. F. MOSQUEDA, L. L. NG, R. NG, G. J. ORTA, C. C. OVERLY, T. H. PAK, S. E. PARRY, S. D. PATHAK, O. C. PEARSON, R. B. PUCHALSKI, Z. L. RILEY, H. R. ROCKETT, S. A. ROWLAND, J. J. ROYALL, M. J. RUIZ, N. R. SARNO, K. SCHAFFNIT, N. V. SHAPOVALOVA, T. SIVISAY, C. R. SLAUGHTERBECK, S. C. SMITH, K. A. SMITH, B. I. SMITH, A. J. SODT, N. N. STEWART, K.-R. STUMPF, S. M. SUNKIN, M. SUTRAM, A. TAM, C. D. TEEMER, C. THALLER, C. L. THOMPSON, L. R. VARNAM, A. VISEL, R. M. WHITLOCK, P. E. WOHNOUTKA, C. K. WOLKEY, V. Y. WONG, M. WOOD, M. B. YAYLAOGLU, R. C. YOUNG, B. L. YOUNGSTROM, X. F. YUAN, B. ZHANG, T. A. ZWINGMAN, AND A. R. JONES, *Genome-Wide Atlas of Gene Expression in the Adult Mouse Brain*, *Nature*, 445 (2007), pp. 168–176, <https://doi.org/10.1038/nature05453>.

- [22] E. L. MACKEVICIUS, A. H. BAHLE, A. H. WILLIAMS, S. GU, N. I. DENISSENKO, M. S. GOLDMAN, AND M. S. FEE, *Unsupervised Discovery of Temporal Sequences in High-Dimensional Datasets, with Applications to Neuroscience*, bioRxiv, (2018), p. 273128, <https://doi.org/10.1101/273128>.
- [23] P. MAJKA, CHAPLIN, TRISTAN A., YU, HSIN-HAO, TOLPYGO, ALEXANDER, MITRA, PARTHA P., WÓJCIK, DANIEL K., AND ROSA, MARCELLO G.P., *Towards a Comprehensive Atlas of Cortical Connections in a Primate Brain: Mapping Tracer Injection Studies of the Common Marmoset into a Reference Digital Template*, *Journal of Comparative Neurology*, 524 (2016), pp. 2161–2181, <https://doi.org/10.1002/cne.24023>.
- [24] P. P. MITRA, *The Circuit Architecture of Whole Brains at the Mesoscopic Scale*, *Neuron*, 83 (2014), pp. 1273–1283, <https://doi.org/10.1016/j.neuron.2014.08.055>.
- [25] S. W. OH, J. A. HARRIS, L. NG, B. WINSLOW, N. CAIN, S. MIHALAS, Q. WANG, C. LAU, L. KUAN, A. M. HENRY, M. T. MORTRUD, B. OUELLETTE, T. N. NGUYEN, S. A. SORENSEN, C. R. SLAUGHTERBECK, W. WAKEMAN, Y. LI, D. FENG, A. HO, E. NICHOLAS, K. E. HIROKAWA, P. BOHN, K. M. JOINES, H. PENG, M. J. HAWRYLYCZ, J. W. PHILLIPS, J. G. HOHMANN, P. WOHNOUTKA, C. R. GERFEN, C. KOCH, A. BERNARD, C. DANG, A. R. JONES, AND H. ZENG, *A Mesoscale Connectome of the Mouse Brain*, *Nature*, 508 (2014), pp. 207–214, <https://doi.org/10.1038/nature13186>.
- [26] J. M. ORTEGA AND W. C. RHEINBOLDT, *Iterative solution of nonlinear equations in several variables*, SIAM, 2000.
- [27] E. A. PNEVMATIKAKIS, D. SOUDRY, Y. GAO, T. A. MACHADO, J. MEREL, D. PFAU, T. REARDON, Y. MU, C. LACEFIELD, W. YANG, M. AHRENS, R. BRUNO, T. M. JESSELL, D. S. PETERKA, R. YUSTE, AND L. PANINSKI, *Simultaneous Denoising, Deconvolution, and Demixing of Calcium Imaging Data*, *Neuron*, 89 (2016), pp. 285–299, <https://doi.org/10.1016/j.neuron.2015.11.037>.
- [28] C. E. POWELL, D. SILVESTER, AND V. SIMONCINI, *An Efficient Reduced Basis Solver for Stochastic Galerkin Matrix Equations*, *SIAM J. Sci. Comput.*, 39 (2017), pp. A141–A163, <https://doi.org/10.1137/15M1032399>.
- [29] S. D. SHANK, V. SIMONCINI, AND D. B. SZYLD, *Efficient low-rank solution of generalized Lyapunov equations*, *Numer. Math.*, 134 (2015), pp. 327–342.
- [30] V. SIMONCINI, *Computational methods for linear matrix equations*, *SIAM Rev.*, 38 (2016), pp. 377–441.
- [31] O. SPORNS, *Networks of the Brain*, The MIT Press, 1st ed., 2010.
- [32] D. C. VAN ESSEN, *Cartography and Connectomes*, *Neuron*, 80 (2013), pp. 775–790, <https://doi.org/10.1016/j.neuron.2013.10.027>.
- [33] G. WAHBA, *Spline Models for Observational Data*, SIAM, Sept. 1990.
- [34] R. J. F. YPMA AND E. T. BULLMORE, *Statistical Analysis of Tract-Tracing Experiments Demonstrates a Dense, Complex Cortical Network in the Mouse*, *PLOS Comput Biol*, 12 (2016), p. e1005104, <https://doi.org/10.1371/journal.pcbi.1005104>.