Regression on Latent Spaces for the Analysis of Multi-Condition Single-Cell RNA-Seq Data

Constantin Ahlmann-Eltze [1] **Wolfgang Huber** [1]

Abstract

Multi-condition single-cell data reveals expression differences between corresponding cell subpopulations in different conditions. Here, we propose to use regression on latent spaces to simultaneously account for variance from known and latent factors. Our approach is built around multivariate regression on Grassmann manifolds. We use the method to analyze a drug treatment experiment on brain tumor biopsies. The method is a versatile new approach for identifying differentially expressed genes from single-cell data of heterogeneous cell subpopulations under arbitrary experimental designs without clustering.

1. Introduction

Compared to bulk RNA-seq, the novelty of single-cell RNA-seq is that it can disentangle expression changes between corresponding cells (i.e., the same cell type and state) under different conditions from those between cell types and states. This combination of explicitly known and latent covariates poses a challenge to existing regression approaches. Variances observed in multi-condition single-cell data can be decomposed into four sources: (1) the conditions, which are explicitly known or even set by the experimenter, (2) cell type or state, which we consider a latent variable that is not explicitly given but can be inferred from the data with some degree of confidence, (3) interactions between the two, and (4) unexplained residual variability.

Currently, the most popular approach is to convert the latent variation into discrete categories by unsupervised clustering or supervised classification. In practice, each cell is assigned to a cluster or group, and expression differences are calculated across conditions using methods initially developed for bulk RNA-seq data (Crowell et al., 2020). This approach has potential drawbacks: most importantly, while discrete cell types might serve as a helpful first-line abstraction, they are insufficient to represent gradual variability, and second, the optimal group size is unclear.

Here, we present a new statistical model for differential expression analysis of multi-condition single-cell data that combines the ideas of linear models and principal component analysis (PCA). We call the method Latent Embedding MUltivariate Regression (LEMUR) and implemented it as an `R` package.

LEMUR is built around the idea of regression on Grassmann manifolds. The Grassmann manifold $\mathrm{Gr}(G, P)$ is the set of all subspaces of dimension P embedded in a $G > P$ dimensional space (Edelman et al., 1998; Bendokat et al., 2020). The elements of Grassmann manifolds can be represented using orthonormal matrices of size $G \times P$. This representation is not unique because multiple matrices can represent the same subspace, but convenient because of its small memory footprint.

Geodesic regression generalizes regular linear regression to optimization problems on manifolds (Fletcher, 2011; Rentmeesters, 2011)

$$\underset{b,v}{\arg\min}\, \frac{1}{2}\sum_{i=1}^{N} d(\mathrm{Exp}_b(x_i v), y_i)^2 \tag{1}$$

where $\boldsymbol{x}$ are the known covariates, $b \in \mathcal{M}$ is a point on the manifold, and $v \in \mathcal{T}_b\mathcal{M}$ is a tangent vector at b. Exp is the exponential map on the manifold $\mathcal{M}$, and the distance function d is defined as the norm of the logarithmic map ($d(b_1, b_2) = ||\,\mathrm{Log}_{b_1}(b_2)||$).

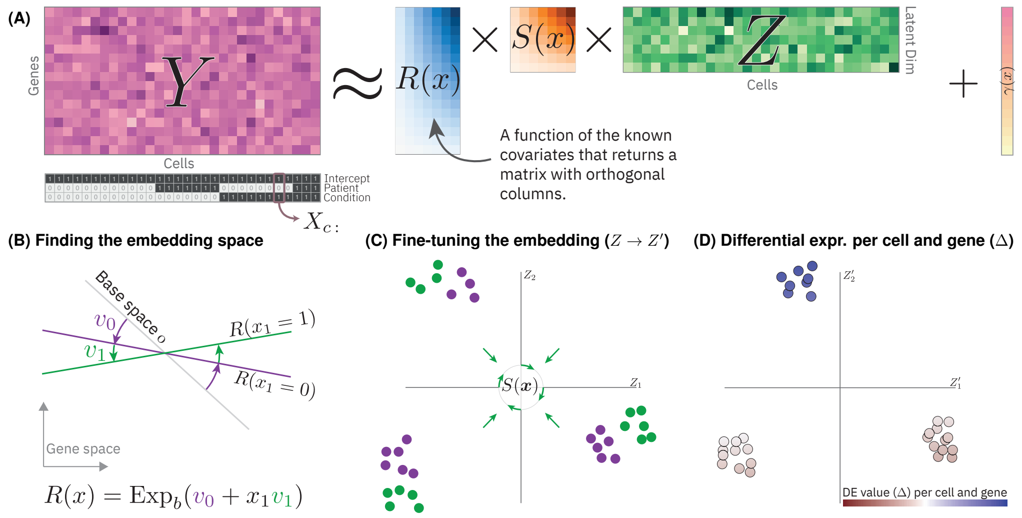
Kim et al. (2014) extended model (1) to a multivariate setting

$$\underset{b,v_1,\ldots,v_K}{\arg\min}\, \frac{1}{2}\sum_{i=1}^{N} d\left(\mathrm{Exp}_b\left(\sum_{k=1}^{K} X_{ik} v_k\right), y_i\right)^2 \tag{2}$$

where instead of just optimizing a single coefficient, we optimize over K coefficients matching the number of columns in a design matrix $X \in \mathbb{R}^{N\times K}$. The additional flexibility is helpful, for example, to model known confounders.

[1]Genome Biology Unit, EMBL Heidelberg, Heidelberg. Correspondence to: Constantin Ahlmann-Eltze <constantin.ahlmann@embl.de>.

Presented at the 2^{nd} Annual Workshop on Topology, Algebra, and Geometry in Machine Learning (TAG-ML) at the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. 2023. Copyright 2023 by the author(s).



(A)
Genes
Y
Cells
Intercept
Patient
Condition
$X_{c:}$
$\approx$ $R(x)$ $\times$ $S(x)$ $\times$ Z
Latent Dim
Cells
$+$ $\gamma(x)$
A function of the known covariates that returns a matrix with orthogonal columns.
(B) Finding the embedding space
Base space o
v_0
v_1
$R(x_1 = 1)$
$R(x_1 = 0)$
Gene space
$R(x) = \text{Exp}_b(v_0 + x_1 v_1)$
(C) Fine-tuning the embedding ($Z \to Z'$)
Z_2
Z_1
$S(x)$
(D) Differential expr. per cell and gene (Δ)
Z'_2
Z'_1
DE value (Δ) per cell and gene

Figure 1. A conceptual overview of latent embedding multivariate regression (LEMUR). (A) Graphical depiction of the matrix factorization at the core of LEMUR. (B) The function $R(x)$ returns latent spaces and is parameterized by parameters v_k acting as high-dimensional rotations. (C) The function $S(x)$ does not affect the approximation of Y but changes the latent positions of the conditions relative to each other and can bring corresponding subpopulations closer. (D) Contrasting the predicted expression level from two conditions for each cell produces a differential expression (DE) value (Δ) for each gene and cell.

Fletcher and Rentmeesters optimized equation (1) using Jacobi fields, but this approach does not work for eq. (2). Instead, Kim et al. suggested working in the tangent space of the base point b to approximate the solution. They optimized

$$\underset{v_1,\ldots,v_K}{\arg\min}\ \frac{1}{2}\sum_{i=1}^{N}||\sum_{k=1}^{K}X_{ik}v_k - y_i^{\wr}||^2 \tag{3}$$

where $y_i^{\wr} = \text{Log}_b(y_i)$ and which can be solved using regular linear regression.

Performing regression on Grassmann manifolds has been used before, for example, to analyze images or predict domain adaption, however the existing work is not directly applicable because it is either limited to a single covariate (Batzies et al., 2015; Hong et al., 2016) or built around kernel regression (Lui, 2012; Yang & Hospedales, 2016).

2. Results

Our method LEMUR disentangles the four sources of variance in multi-condition single-cell data by performing multivariate regression on Grassmann manifolds.

LEMUR takes as input a data matrix $Y \in \mathbb{R}^{G\times C}$, where G is the number of genes and C is the number of cells. The method assumes that appropriate preprocessing, including size factor normalization and variance stabilization, was performed (Ahlmann-Eltze & Huber, 2023). In addition, it expects specification of the design matrix $X \in \mathbb{R}^{C\times K}$. It produces several outputs:

- a low-dimensional representation of all cells,
- explicitly parameterized, bijective transformations that map the latent spaces into a joint space, and
- the predicted expression changes between any pair of conditions for each gene and cell.

2.1. Regression on Latent Spaces

LEMUR is both a regression and a matrix factorization algorithm (Fig. 1A). The simplest matrix factorization is PCA (and similarly SVD) which can be used to approximate a data matrix Y by a product of two simpler matrices

$$Y \approx RZ + \gamma_{\text{offset}}. \tag{4}$$

Here, $R \in \mathbb{R}^{G\times P}$ is called *principal vectors* (or sometimes rotation or loadings matrix). The columns of R are orthonormal ($R^T R = I$). The *embedding matrix* $Z \in \mathbb{R}^{P\times C}$ contains the P-dimensional coordinates of each cell in the latent space. If $P < \min(G, C)$, PCA reduces the dimension of the data. γ_{offset} is a vector with G rows and centers the observations[1].

LEMUR combines these ideas with regression analysis in the presence of covariates for the cells encoded in a design matrix X. Instead of R being fixed, we treat it as a function of the covariates,

$$R : \mathbb{R}^K \to \{A \in \mathbb{R}^{G\times P} \mid A^T A = I_P\} \tag{5}$$

where the function arguments are rows of the design matrix and the output is the set of orthonormal $G \times P$ matrices. We

[1]We overload the sum operator (+) for a matrix and a conformable column vector to produce another matrix: $C_{ij} = A_{ij} + b_i$

parameterize R using the exponential map of a Grassmann manifold

$$R(x) = \mathrm{Exp}_b\left(\sum_{k=1}^{K} x_k v_k\right). \tag{6}$$

Thus our model is

$$Y_{:c} \approx R(X_{c:})\, Z_{:c} + \gamma(X_{c:}), \tag{7}$$

where we use the notation : to indicate extracting row or column vectors from a matrix (e.g., $Z_{:c}$ is a vector of length P that contains the latent space representation of cell c). We allow the offset γ to depend on the covariates, too.

$R(x)$ is the latent space for all cells in condition x, i.e., all cells whose corresponding row in the design matrix equals x. Since R is defined on all of $\mathbb{R}^K$, the model can interpolate or extrapolate conditions that were not even measured. Informally, we think of the function R in analogy to link functions in generalized linear models, which map linear predictors to statistical distributions from which observations are drawn. In our model, R maps the linear predictor for a cell to a linear subspace of the full gene expression space, in which we believe this cell's gene expression should lie (Fig. 1B).

Model (7) addresses the variance decomposition challenge posed in the introduction: known sources of variation are encoded in the design matrix X and act through the function $R(X)$; the latent variation (cell types or states) takes place in the linear space spanned by $R(X)$ and is parameterized by each cell's coordinates in Z. Interactions between the two are represented by condition-dependent changes in $R(x)$ that can differ in different directions of the embedding space Z, and unexplained variability is absorbed in the residuals of the approximation (Fig. 1B).

To optimize the coefficients in model (7), we first fit $\gamma(x) = \Gamma x$ using linear regression. Then, we optimize the coefficients $v_1, \ldots, v_K$ for $Y - \gamma(X)$ using the approximation of Kim et al. (2014). For each condition (i.e., set of cells with the same row x in the design matrix), we find the corresponding point $y^l \in \mathrm{Gr}(G, P)$ on the Grassmann manifold by calculating the principal vectors via PCA for the cells in that condition. As the base point b, we use the principal vectors from PCA on all cells (details in Appendix A).

2.1.1. Fine-tuning the embedding

Model (7) assumes that corresponding cell subpopulations from different conditions can be matched just by aligning their respective latent spaces through a high-dimensional rotation. Sometimes, this is not flexible enough, e.g., if a treatment drastically affects some, but not all, cell subpopulations, and thus the relative distances between subpopulations change. To model such localized changes (Fig. 1C), we extend eq. (7) by a condition-dependent linear alignment matrix S:

$$Y_{:c} \approx R(X_{c:})\; S(X_{c:})\; Z'_{:c} + \gamma(X_{c:}). \tag{8}$$

The $P \times P$ matrix $S(x)$ is invertible, and we define $Z'_{:c} \coloneqq S^{-1}(X_{c:})Z_{:c}$. This ensures that S only influences which subpopulations are considered "corresponding" and does not affect the approximation of Y. The analyst can provide cell-type labels or use tools like Harmony (Korsunsky et al., 2019) to find those labels automatically; alternatively, we set $S \equiv I$ to skip the alignment.

We parameterize S as

$$S(x) = \left(I + \sum_{k=1}^{K} x_k w_k\right)^{-1}, \tag{9}$$

where $w_1, \ldots, w_K$ are the coefficients of S. For a given vector of labels $\boldsymbol{u} = \{1, \ldots, Q\}^C$ that assigns each cell to a group q, we find the coefficients $w_1, \ldots, w_K$ by optimizing

$$\underset{w_1,\ldots,w_K}{\arg\min} \sum_{c=1}^{C} ||M_{u_c} - S^{-1}(X_{c:})Z_{:c}||^2 + \lambda \sum_{k=1}^{K} ||w_k||^2 \tag{10}$$

where $M_q = \frac{1}{\#(\boldsymbol{u}=q)} \sum_{c \in \{\boldsymbol{u}=q\}} Z_{:c}$ is the mean of the latent positions of the cells in that group. The parameter λ controls the strength of the ridge penalty, and for a large enough λ the matrix $S(x)$ will be invertible.

2.1.2. Differential expression analysis

Model (8) predicts gene expression given a value of the covariates x and a position in the embedding space z. We calculate the differential expression for each gene and cell by comparing the predictions for any contrast (e.g., between two conditions $x^{(A)}$ and $x^{(B)}$) for all Z' (Fig 1E).

We use the matrix of differential expression values Δ ($G \times C$) to visualize the differential expression patterns for selected genes as a function of the latent space (in practice, we use for this a convenient 2D embedding of it, such as UMAP (McInnes et al., 2018)) to see how the differential expression possibly changes across that space.

2.2. Analysis of a Drug Perturbation in Glioblastoma

The glioblastoma study by Zhao et al. (2021) reported single-cell RNA-seq data of glioblastoma biopsies from five patients, each in two conditions: control and panobinostat, a non-selective histone deacetylase (HDAC) inhibitor. Fig. 2A shows the paired experimental design. In total, there are $C = 47\,900$ cells, and we considered the $G = 6\,000$ most variable genes.

A two-dimensional visualization of the cells by applying UMAP to the normalized matrix Y showed patterns most

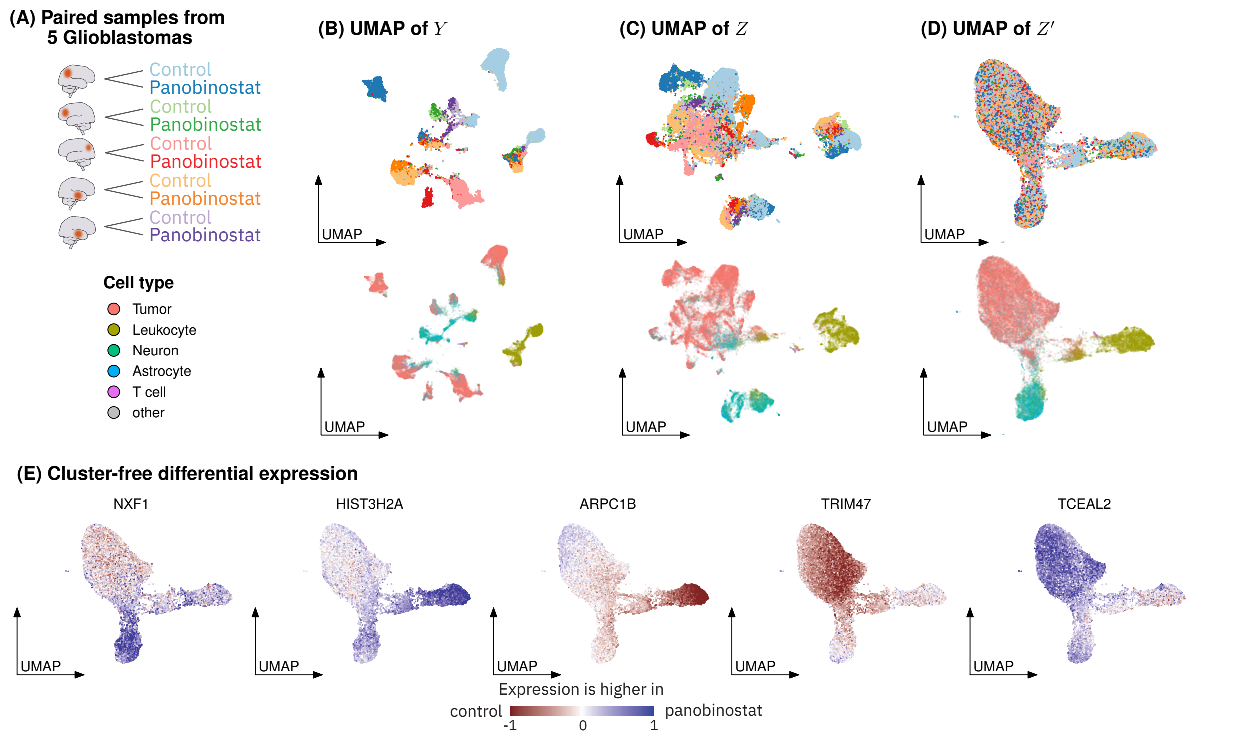


Figure 2. Results from applying LEMUR to a glioblastoma dataset. (A) A schematic of the experimental design. (B-D) UMAP plots colored by condition on the input data (top row) or the cell type (bottom row): UMAP of (B) Y, (C) the inferred latent position Z with $S(x) = I$, and (D) Z' after adjustment using maximum diversity clustering. (E) The differential expression inferred by LEMUR for five genes, with cells laid out by UMAP of Z'.

distinctively associated with the known covariates patient ID and treatment condition (Fig. 2B). We used LEMUR to absorb patient and treatment effects into R, using a $P = 15$ dimensional latent space and fixing $S(x) = I$. Fig. 2C shows a UMAP of the matrix Z of latent coordinates for each cell. As a result, cells from different samples were more intermixed, and the visualization reflected more within-sample cellular heterogeneity. This picture became even clearer after we used S to encode an alignment between cell subpopulations across samples using Harmony's maximum diversity clustering. Here, a large tumor subpopulation (classified by Zhao et al. (2021) based on chromosome 7 amplification and chromosome 10 deletion) and two non-tumor subpopulations became apparent (Fig. 2D).

We plotted the predicted expression change between panobinostat treatment and the control condition for five genes on the UMAP visualization of Z' (Fig. 2E). We found that the differential expression patterns correspond well with the manually assigned cell types.

3. Discussion

We have introduced a method for analyzing single-cell expression data of heterogeneous tissues under multiple conditions with arbitrary experimental designs. LEMUR uses regression on latent spaces to enable cluster-free differential expression analysis. We have shown how it can harmonize data using linear transformations. We demonstrated its utility for finding differentially expressed genes and subsets of affected cells. Applied to the glioblastoma dataset by Zhao et al. (2021), LEMUR identified biologically relevant expression patterns.

Some aspects of the current implementation leave room for improvement: we only approximately solve the optimization of model (7), which could be improved with gradient descent (but calculating the gradient is non-trivial). Second, the optimization of S is a regression-based version of the generalized Procrustes problem. This might allow us to use a variant of Bai and Bartolis's (2022) eigenvalue-based approach to find the targets M_q instead of simple averaging.

Overall, we believe that LEMUR is an exciting application of geometric machine learning to single-cell data analysis. Compared to approaches that require clustering before differential expression analysis, the representation of cells in a continuous latent space may be a better fit for the underlying biology. Compared to deep-learning-based latent space approaches, LEMUR's interpretable, simple, and easy-to-inspect model should facilitate follow-up investigation of its discoveries.

Software and Data

The single-cell glioblastoma data is publicly available on the GEO database with the identifier GSE148842. An implementation of the method is available as an R package on github.com/const-ae/lemur.

Acknowledgments

We thank Dr. Simon Anders, Dr. Oliver Stegle and Dr. Judith Zaugg for valuable feedback and discussions. We thank Prof. Dr. Ronny Bergmann for his advice on optimization on manifolds.

This work has been supported by the EMBL International Ph.D. Programme, by the German Federal Ministry of Education and Research (CompLS project SIMONA under grant agreement no. 031L0263A), and the European Research Council (Synergy Grant DECODE under grant agreement no. 810296).

References

Ahlmann-Eltze, C. and Huber, W. Comparison of transformations for single-cell RNA-seq data. *Nature Methods*, pp. 1–8, 2023.

Bai, F. and Bartoli, A. Procrustes analysis with deformations: A closed-form solution by eigenvalue decomposition. *International Journal of Computer Vision*, pp. 1–27, 2022.

Batzies, E., Hüper, K., Machado, L., and Leite, F. S. Geometric mean and geodesic regression on Grassmannians. *Linear Algebra and its Applications*, 466:83–101, 2015. ISSN 0024-3795. doi: https://doi.org/10.1016/j.laa.2014.10.003. URL `https://www.sciencedirect.com/science/article/pii/S0024379514006508`.

Bendokat, T., Zimmermann, R., and Absil, P.-A. A Grassmann manifold handbook: Basic geometry and computational aspects. *arXiv preprint arXiv:2011.13699*, 2020.

Crowell, H. L., Soneson, C., Germain, P.-L., Calini, D., Collin, L., Raposo, C., Malhotra, D., and Robinson, M. D. Muscat detects subpopulation-specific state transitions from multi-sample multi-condition single-cell transcriptomics data. *Nature Communications*, 11(1):6077, 2020.

Edelman, A., Arias, T. A., and Smith, S. T. The geometry of algorithms with orthogonality constraints. *SIAM journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

Fletcher, T. Geodesic regression on Riemannian manifolds. In *Proceedings of the Third International Workshop on Mathematical Foundations of Computational Anatomy-Geometrical and Statistical Methods for Modelling Biological Shape Variability*, pp. 75–86, 2011.

Hong, Y., Kwitt, R., Singh, N., Vasconcelos, N., and Niethammer, M. Parametric regression on the Grassmannian. *IEEE transactions on pattern analysis and machine intelligence*, 38(11):2284–2297, 2016.

Kim, H. J., Adluru, N., Collins, M. D., Chung, M. K., Bendlin, B. B., Johnson, S. C., Davidson, R. J., and Singh, V. Multivariate general linear models (MGLM) on Riemannian manifolds with applications to statistical analysis of diffusion weighted images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2705–2712, 2014.

Korsunsky, I., Millard, N., Fan, J., Slowikowski, K., Zhang, F., Wei, K., Baglaenko, Y., Brenner, M., Loh, P.-r., and Raychaudhuri, S. Fast, sensitive and accurate integration of single-cell data with Harmony. *Nature Methods*, 16 (12):1289–1296, 2019.

Lui, Y. M. A least squares regression framework on manifolds and its application to gesture recognition. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 13–18. IEEE, 2012.

McInnes, L., Healy, J., and Melville, J. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Rentmeesters, Q. A gradient method for geodesic data fitting on some symmetric Riemannian manifolds. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pp. 7141–7146. IEEE, 2011.

Yang, Y. and Hospedales, T. M. Multivariate regression on the Grassmannian for predicting novel domains. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5071–5080, 2016.

Zhao, W., Dovas, A., Spinazzi, E. F., Levitin, H. M., Banu, M. A., Upadhyayula, P., Sudhakar, T., Marie, T., Otten, M. L., Sisti, M. B., et al. Deconvolution of cell type-specific drug responses in human tumor tissue with single-cell RNA-seq. *Genome Medicine*, 13(1):82, 2021.

A. Appendix

To illustrate LEMUR's algorithm, we provide a simplified implementation of the core functions in `R` (minimum version 4.1). The `multicondition_pca` function optimizes model (7) with $S(x) = I$.

```
grassmann_log <- function(p, q){
  n <- nrow(p)
  k <- ncol(p)
  z <- t(q) %*% p
  At <- t(q) - z %*% t(p)
  Bt <- lm.fit(z, At)$coefficients
  svd <- svd(t(Bt), k, k)
  svd$u %*% diag(atan(svd$d), nrow = k) %*% t(svd$v)
}

grassmann_map <- function(x, base_point){
  svd <- svd(x)
  base_point %*% svd$v %*% diag(cos(svd$d), nrow = length(svd$d)) %*% t(svd$v) +
    svd$u %*% diag(sin(svd$d), nrow = length(svd$d)) %*% t(svd$v)
}

#' @param Y is a matrix with features in the rows and observations in the columns
#' @param design_matrix a matrix with one row per observation coding the covariates
#'
#' @return a list with the embedding, the base_point, the coefficients for the
#'   Grassmann exponential map, and the coefficients for the linear regression.
multicondition_pca <- function(Y, design_matrix, n_embedding = 15){
  # Center observations with linear regressiong
  fit <- lm.fit(design_matrix, t(as.matrix(Y)))
  Y <- t(residuals(fit))

  # Find base point with PCA over all data points
  base_point <- irlba::prcomp_irlba(t(Y), n = n_embedding, center = FALSE)$rotation

  # Find the subspace of each condition
  red_design <- unique(design_matrix)
  cond_ids <- vctrs::vec_group_id(design_matrix)
  cond_weights <- c(table(cond_ids))
  cond_subspaces <- lapply(seq_len(nrow(red_design)), \(cond){
    irlba::prcomp_irlba(t(Y[,cond_ids == cond,drop=FALSE]),
                        n = n_embedding, center = FALSE)$rotation
  })

  # Find coefficients of Grassmann exponential map
  log_points <- do.call(cbind, lapply(cond_subspaces, \(subspace){
    as.vector(grassmann_log(base_point, subspace))
  }))
  coefficients <- t(lm.wfit(red_design, t(log_points), w = cond_weights)$coefficients)
  coefficients <- array(coefficients, dim = c(nrow(Y), n_embedding, ncol(design_matrix)))

  # Project the points on the embedding
  embedding <- matrix(NA, nrow = n_embedding, ncol = ncol(Y))
  for(cond in seq_len(nrow(red_design))){
    tang_vec <- matrix(0, nrow = nrow(Y), ncol = n_embedding)
    for(k in seq_len(ncol(red_design))){
      tang_vec <- tang_vec + red_design[cond, k] * coefficients[,,k]
    }
    embedding[,cond_ids == cond] <- t(grassmann_map(tang_vec, base_point)) %*% Y[,cond_ids == cond]
  }

  # Order axes by variance
  svd_emb <- svd(embedding)
  base_point <- base_point %*% svd_emb$u
  for(k in seq_len(ncol(design_matrix))){
    coefficients[,,k] <- coefficients[,,k] %*% svd_emb$u
  }
  embedding <- t(svd_emb$v) * svd_emb$d

  # Return values
  list(embedding = embedding, base_point = base_point,
       coefficients = coefficients, linear_coefficients = t(fit$coefficients))
}
```

The `align` function optimizes the parameters of $S(x)$ and combined with `multicondition_pca` the functions solve model (8).

```
ridge_regression <- function(Y, X, ridge_penalty = 0, weights = rep(1, nrow(X))){
  ridge_penalty <- diag(ridge_penalty, nrow = ncol(X))
  weights_sqrt <- sqrt(weights)
  X_extended <- rbind(X * weights_sqrt, sqrt(sum(weights)) * (t(ridge_penalty) %*% ridge_penalty))
  Y_extended <- cbind(t(t(Y) * weights_sqrt), matrix(0, nrow = nrow(Y), ncol = ncol(X)))
  qr <- qr(X_extended)
  t(solve(qr, t(Y_extended)))
}

align <- function(embedding, design, groups, ridge_penalty){
  groups <- as.integer(as.factor(groups))
  # Calculate target shape using mean per group
  means <- lapply(seq_len(max(groups)), \(gr) matrixStats::rowMeans2(embedding, cols = groups == gr))
  M <- do.call(cbind, means[groups])
  # Solve optimization of eq. (10) with ridge regression
  interact_design_matrix <- (design %x% matrix(1, ncol = nrow(embedding))) *
    (matrix(1, ncol = ncol(design)) %x% t(embedding))
  alignment_coefs <- ridge_regression(M - embedding, interact_design_matrix, ridge_penalty)
  alignment_coefs <- array(alignment_coefs, dim = c(nrow(embedding), nrow(embedding), ncol(design)))
  # Apply alignment to embedding
  cond_ids <- vctrs::vec_group_id(design)
  for(id in unique(cond_ids)){
    tang_vec <- matrix(0, nrow = nrow(embedding), ncol = nrow(embedding))
    for(k in seq_len(ncol(design))){
      tang_vec <- tang_vec + design[which(cond_ids == id)[1], k] * alignment_coefs[,,k]
    }
    embedding[,cond_ids == id] <- (diag(nrow = nrow(embedding)) + tang_vec) %*% embedding[,cond_ids == id]
  }
  # Return result
  list(alignment_coefs = alignment_coefs, embedding = embedding)
}
```