Parametric Regression on the Grassmannian
Supplementary Material

This supplementary material contains technical details and additional results that were left-out in the original submission for brevity. References to sections or figures in the manuscript are marked as [Paper, XXX]. The source code for all approaches (including our implementations of Su et al. [1] and Rentmeesters [2]) is publicly available at:

https://bitbucket.org/yi_hong/ggr_all

1 Residuals to curves on $\mathcal{G}(p, n)$

In [Paper, Algorithms 1 and 2] we need the gradient of the residuals $d^2_g(Y(r_i), Y_i)$ (here, $Y(r_i)$ is equal to $X_1(r_i)$) with respect to the base point $Y(r_i)$ in order to compute the jump conditions for the adjoint variable $\lambda_1$. The residual measures the squared geodesic distance between the point $Y(r_i)$ on the fitted curve and the corresponding measurement $Y_i$. To derive this gradient, we consider the constrained minimization problem of two points with exact matching:

$$E(Y(r)) = \int_{r_0}^{r_1} \text{tr} \, \dot{Y}(r)^\top \dot{Y}(r) \, dr,$$

subject to $Y(r_0) = Y_0, Y(r_1) = Y_1$

and $Y(r_0)^\top Y(r_0) = I_p$,

with $\dot{Y}(r) = (I_n - Y(r)Y(r)^\top)C$. We know that the squared distance can be formulated as $d^2_g(Y_0, Y_1) = \min_{Y(r)} E(Y(r))$ for $r_0 = 0$ and $r_1 = 1$. After adding the constraint on the form of $\dot{Y}(r)$ via the time-dependent adjoint variable $\lambda$ and the constraint $Y(0)^\top Y(0) = I_p$ via $\lambda_c$, we obtain (by taking variations), among other terms, the optimality condition

$$(2C^\top - \lambda^\top)(I_n - Y(r)Y(r)^\top) = 0 ,$$

and another optimality condition for a free initial condition$^{1}$ $Y(0)$

$$\nabla_{Y(0)} E = -\lambda(0) + Y(0)(\lambda_c^\top + \lambda_c) = 0 .$$

$^{1}$Note that technically we started with $Y(r_0) = Y_0$, i.e., this condition would not be free and we would not need to consider variations of $Y(r_0)$. However, our goal is to compute the energy gradient with respect to the initial condition $Y(r_0)$. Consequentially, the variation of the energy with respect to it allows us to compute $\nabla_{Y(0)} E$. 

1
Left-multiplication by $Y(0)^T$ yields $\lambda_c + \lambda_c^T = Y(0)^T \lambda(0)$ which we can use to obtain, upon back-substitution into Eq. (3),

$$-(I_n - Y(0)Y(0)^T)\lambda(0) = 0 \quad (4)$$

Using Eq. (2) and the above expression for $\dot{Y}(r)$, we can obtain

$$\nabla_{Y(0)}E = -(I_n - Y(0)Y(0)^T)\lambda(0) = -2\dot{Y}(0) \quad (5)$$

with $\dot{Y}(0)$ being the tangent vector at $Y(0)$ such that $\text{Exp}_{Y(0)}(\dot{Y}(0)) = Y(1)$. This tangent vector can be computed via the log-map $\dot{Y}(0) = \text{Log}_{Y(0)}(Y(1))$ using the efficient algorithms in [3] for instance. Further details are also provided in Section 5 of this supplementary material.

2 Line search on the Grassmannian

Performing a line search on the Grassmann manifold in [Paper, Algorithm 1] is not as straightforward as in Euclidean space since we need to assure that the constraints for $Y(r_0)$ and $\dot{Y}(r_0)$ are fulfilled for any given step. In particular, changing $Y(r_0)$ will change the associated tangent vector $\dot{Y}(r_0)$. Once, we have updated $Y(r_0)$ to $Y^u(r_0)$ by moving along the geodesic defined by $Y(r_0)$ and the gradient of the energy with respect to this initial point, i.e., $\nabla_{Y(r_0)}E$, we can transport the tangent $\dot{Y}(r_0)$ to $Y^u(r_0)$ using the closed form solution for parallel transport of [4] (see also Section 5). In particular,

$$\dot{Y}^u(r_0) = [Y(r_0)V;U]\begin{bmatrix} -\sin t\Sigma \\ \cos t\Sigma \end{bmatrix} U^T + (I_n - UU^T)\dot{Y}(r_0) \quad (6)$$

where $H = U\Sigma V^T$ is the compact SVD of the tangent vector at $Y(r_0)$ along the geodesic connecting $Y(r_0)$ and $Y^u(r_0)$. Note that $t = 1$ in Eq. (6). Algorithm 1 lists the line search procedure in full technical detail.

---

**Algorithm 1:** Grassmannian equivalent of $x^{k+1} = x^k - \Delta t g$, where $\Delta t$ is the timestep and $g$ is the gradient.

Data: $Y(r_0), \dot{Y}(r_0), \nabla_{Y(r_0)}E, \nabla_{\dot{Y}(r_0)}E, \Delta t$

Result: Updated $Y^u(r_0)$ and $\dot{Y}^u(r_0)$

1. Compute $\dot{Y}^u(r_0) = \dot{Y}(r_0) - \Delta t\nabla_{X(0)}E$
2. Compute $Y^u(r_0)$ by flowing for $\Delta t$ along geodesic with initial condition $(Y(r_0), -\nabla_{X(0)}E)$
3. Transport $\dot{Y}^u(r_0)$ along the geodesic connecting $Y(r_0)$ to $Y^u(r_0)$, using (6), resulting in $\dot{Y}^u_T(r_0)$
4. Project updated initial velocity onto the tangent space (for consistency):
   $$\dot{Y}^u(r_0) \leftarrow (I_n - Y^u(r_0)Y^u(r_0)^T)\dot{Y}^u_T(r_0).$$

Note: when implementing [Paper, Algorithm 1] and [Paper, Algorithm 2] it is important to pay attention to the ordering of the matrix multiplications, as performing them in an appropriate order will reduce time and memory complexity.
3 Cubic splines on $\mathcal{G}(p, n)$

We derive the evolution equations for $X_3$ and $X_4$ of [Paper, Section 5.5]. First, we show that by enforcing $X_1^\top X_2 = 0$ at all time points, we only need to enforce $X_1^\top X_1 = I_p$ initially. This can be seen by taking the derivative of $X_1^\top X_1$ with respect to $r$, i.e.,

$$\frac{d}{dr} X_1^\top X_1 = \dot{X}_1^\top X_1 + X_1^\top \dot{X}_1 = X_2^\top X_1 + X_1^\top X_2 = 0 .$$  \hspace{1cm} (7)

In other words, if $X_1(0)^\top X_1(0) = I_p$ holds, then $X_1^\top X_1 = I_p$ holds for all $r$ by enforcing $X_1^\top X_2 = 0$ at all time points. With this result in mind, the optimization problem of [Paper, Eq. (22)] can be rewritten as

$$\min_\Theta E(\Theta) = \frac{1}{2} \int_0^1 \text{tr} X_3^\top X_3 dr$$

subject to

$$X_2 = \dot{X}_1, \quad X_3 = \dot{X}_2 + X_1 X_2^\top X_2,$$

$$X_1^\top X_2 = 0, \quad X_1(0)^\top X_1(0) = I_p .$$  \hspace{1cm} (8)

For the first three time-evolving constraints, we introduce three time-dependent adjoint variables, $\lambda_1$, $\lambda_2$, and $\lambda_3$, to convert the constrained problem to an unconstrained one. We then take the variations with respect to the state variables which results in the following system of adjoint equations:

$$-\dot{\lambda}_1^\top + (X_2^\top X_2)\lambda_2^\top + \lambda_3 X_2^\top = 0,$$

$$-\dot{\lambda}_2^\top - \lambda_1^\top + X_1^\top \lambda_2 X_2^\top + \lambda_2^\top X_1 X_2^\top + \lambda_3^\top X_1^\top = 0,$$

$$X_3^\top - \lambda_2^\top = 0 .$$  \hspace{1cm} (9)

Using $X_3 = \lambda_2$ and setting $X_4 = \lambda_1$, we can rewrite the adjoint equations (9) as

$$\dot{X}_4^\top = (X_2^\top X_2)X_3^\top + \lambda_3 X_2^\top ,$$

$$\dot{X}_3^\top = -X_4^\top + X_1^\top X_3 X_2^\top + X_3^\top X_1 X_2^\top + \lambda_3^\top X_1^\top .$$  \hspace{1cm} (10)

Next, we derive the form of $X_3$ using the dynamic constraints. By taking the time-derivative of $X_1^\top X_2$ we get

$$\frac{d}{dr} X_1^\top X_2 = X_1^\top \dot{X}_2 + X_1^\top X_2$$

$$= \dot{X}_1^\top X_2 + X_1^\top (X_3 - X_1 X_2^\top X_2)$$

$$= X_2^\top X_2 + X_1^\top X_3 - X_2^\top X_2$$

$$= X_1^\top X_3 .$$  \hspace{1cm} (12)
However, the constraint $X_1^T X_2 = 0$ implies $\frac{d}{dt} X_1^T X_2 = 0$ which yields
\[ X_1^T X_3 = 0. \tag{13} \]

Next $X_1^T X_3 = 0$ implies $\frac{d}{dt} X_1^T X_3 = 0$ and we get, as a side result,
\[ X_2^T X_3 + X_1^T \dot{X}_3 = 0. \tag{14} \]

We can then use (13) to simplify Eq. (11) to
\[ -\dot{X}_3^T - X_1^T + \lambda_3^T X_1 = 0 \iff -X_3 - X_4 + X_1 \lambda_3 = 0. \tag{15} \]

Upon left-multiplication of Eq. (15) by $X_1$ we obtain the expression for $\lambda_3$ as
\[ \lambda_3 = X_1^T X_3 + X_1^T X_4 \tag{14} = X_1^T X_4 - X_2^T X_3. \tag{16} \]

Substituting $\lambda_3$ into Eq. (15) yields the evolution equation for $\dot{X}_3$ as
\[ \dot{X}_3 = -X_4 + X_1 X_1^T X_4 - X_1 X_2^T X_3 \tag{17} \]

and substituting $\lambda_3$ in Eq. (10) yields the evolution equation for $\dot{X}_4$
\[ \dot{X}_4 = X_3 X_2^T X_2 + X_2 X_4^T X_1 - X_2 X_3^T X_2. \tag{18} \]

4 System identification

To support a non-uniform weighting of samples during system identification in [Paper, Section 7], we propose a temporally localized variant of [5]. This is beneficial in situations where we need a considerable number of frames for stable system identification, yet not all samples should contribute equally to the LDS parameter estimates. Specifically, given the measurement matrix $M = [y_1, \cdots, y_T]$ and a set of weights $w = [w_1, \cdots, w_T]$, such that $\sum_i w_i = \tau$, we perform a weighted SVD of $M$, i.e.,
\[ U \Sigma V^T = \text{Mdiag}(\sqrt{w}) . \tag{19} \]

Then, as in [5], $C = U$ and $X = \Sigma V^T$. Once the state matrix $X$ has been determined, $A$ can be computed as $A = X_2^T W^{1/2} (X_1^{-1} W^{1/2})^\dagger$, where $\dagger$ denotes the pseudoinverse, $X_2^T = [x_2, \cdots, x_T]$, $X_1^{-1} = [x_1, \cdots, x_{T-1}]$ and $W^{1/2}$ is a diagonal matrix with $W_{ii}^{1/2} = [\frac{1}{2}(w_i + w_{i+1})]^{1/2}$.

5 Differential geometric tools on $G(p, n)$

In this part of the supplementary material, we present the differential geometric “tools” required for our forward / backward shooting approach. In particular, we provide formulas / derivations for the Riemannian exponential map (exp-map), the inverse exponential map (aka log-map) and parallel transport on $G(p, n)$. Additionally, we list exemplary implementations in MATLAB (partly taken from the manopt [6] toolbox and our source code).
5.1 Exponential map

The exponential map (exp-map) maps a point $\mathbf{D}$ in the tangent space $T_Y \mathcal{G}(p,n)$ at $Y = \text{span}(\mathbf{Y})$ to a point $\mathbf{Z} = \text{span}(\mathbf{Z})$ on the manifold $\mathcal{G}(p,n)$, i.e.,

$$\text{Exp}_Y(t\mathbf{D}) = \mathbf{Z}, \quad t \in [0,1]$$

along the geodesic that connects $Y$ and $Z$. By letting $\mathbf{D} = \mathbf{U}\Sigma\mathbf{V}^\top$ denote the compact SVD decomposition of $\mathbf{D}$, the exp-map on $\mathcal{G}(p,n)$, in terms of representers $\mathbf{Y}$ and $\mathbf{Z}$, can be written as

$$\mathbf{Z} = \mathbf{Y}\mathbf{V} \cos(t\Sigma)\mathbf{V}^\top + \mathbf{U}\sin(t\Sigma)\mathbf{V}^\top.$$  \hspace{1cm} (20)

This computation is implemented in the manopt [6] toolbox. For convenience, MATLAB code (adjusted for readability) is listed below. Computation of $\mathbf{D}$ via the inverse exponential map, given $\mathbf{Y}$ and $\mathbf{Z}$ as input, is described in the next section.

```matlab
function \mathbf{Z} = \exp_map(\mathbf{Y},\mathbf{D},t)
% EXP_MAP computes the exponential map on the Grassmannian \mathcal{G}(p,n).
% Input:
% \mathbf{Y} - (n \times p) matrix (representer for subspace \mathcal{Y})
% \mathbf{D} - (n \times p) matrix (tangent vector at \mathcal{Y})
% t - Time in [0,1]
% Output:
% \mathbf{Z} - (n \times p) matrix; Result of walking along the geodesic, defined
% by (\mathbf{Y},\mathbf{D}), for time t;
% Reference - Eq. (2.65) from
% Edelman et al.
% The Geometry of algorithms with orthogonality constraints
t\mathbf{D} = t*\mathbf{D};
[\mathbf{U}, \mathbf{S}, \mathbf{V}] = \text{svd}(\ t\mathbf{D}, \ 0\ );
\mathbf{cosS} = \text{diag}(\ \cos(\ \text{diag}(\ \mathbf{S}\ ))\ ) ;
\mathbf{sinS} = \text{diag}(\ \sin(\ \text{diag}(\ \mathbf{S}\ ))\ ) ;
\mathbf{Z} = \mathbf{Y}\mathbf{V}\cos(t\Sigma)\mathbf{V}^\top + \mathbf{U}\sin(t\Sigma)\mathbf{V}^\top;
[\mathbf{Q}, \text{unused}] = \text{qr}(\ \mathbf{Z}, \ 0\ );
\mathbf{Z} = \mathbf{Q};
end
```

5.2 Inverse exponential map

The inverse exponential map (log-map) computes the mapping from a neighborhood of $U \subset \mathcal{G}(p,n)$ of $\mathcal{X}$ to $T_X \mathcal{G}(p,n)$. In terms of representers $\mathbf{X}, \mathbf{Y}$ for the subspaces $\mathcal{X} = \text{span}(\mathbf{X}), \mathcal{Y} = \text{span}(\mathbf{Y})$ we will write

$$\mathbf{H} = \text{Log}_\mathbf{X}(\mathbf{Y}) .$$

In other words, $\mathbf{H}$ is the direction matrix which allows you to start at $\mathcal{X}$ and walk along the geodesic in the direction of $\mathbf{H}$ to reach $\mathcal{Y}$ in unit time ($t = 1$), i.e.,

$$\mathbf{Y} = \text{Exp}_\mathbf{X}(\mathbf{H}) .$$
Let $H = U \Sigma V^\top$. By multiplying Eq. (20) with $X^\top$ on the left-hand side (and $t = 1$), we get
\[
X^\top Y = \begin{bmatrix} X^\top X \cos(\Sigma) V^\top + X^\top U \sin(\Sigma) V^\top \\
\end{bmatrix}_{1:p} = V \cos(\Sigma) V^\top.
\]
Consequently, it follows that
\[
U \sin(\Sigma) V^\top = Y - XX^\top Y.
\]
We can now write
\[
U \sin(\Sigma) V^\top (V \cos(\Sigma) V^\top)^{-1} = (Y - XX^\top Y)(X^\top Y^\top)^{-1}
\]
which – upon noting that (1) $(V \cos(\Sigma) V^\top)^{-1} = V \cos(\Sigma)^{-1} V^\top$ and (2) $V^\top V = I_p$ – reduces to
\[
U \tan(\Sigma) V^\top = (Y - XX^\top Y)(X^\top Y^\top)^{-1}.
\]
This yields $H$ via a SVD decomposition of $(Y - XX^\top Y)(X^\top Y^\top)^{-1}$ as
\[
H = \text{Log}_X(Y) = U \arctan(\Sigma) V^\top.
\]
This is also the way the Grassmannian log-map is implemented in the manopt [6] toolbox. For convenience, MATLAB code (adjusted for readability) is listed below.

```matlab
function H = log_map(X,Y)
% LOG_MAP computes the inverse exponential map on the Grassmannian.
% Input:
% X - (n x p) matrix (representer for subspace \mathcal{X})
% Y - (n x p) matrix (representer for subspace \mathcal{Y})
% Output:
% H - (n x p) direction matrix in the tangent space at subspace \mathcal{X}
[unused, p] = size(X);
ytx = Y.'*X;
At = Y.'-ytx*X.';
Bt = ytx\At;
[U, S, V] = svd(Bt.' , 'econ');
U = U(:, 1:p);
S = diag(S);
S = S(1:p);
V = V(:, 1:p);
H = U*diag(atan(S))*V.';
end
```

### 5.3 Parallel transport

Given two subspaces $\mathcal{X}, \mathcal{Y}$, represented via $X, Y$ and a direction matrix $H \in T_X G(p, n)$ such that $Y = \text{Exp}_X(H)$, the objective is to transport an arbitrary tangent vector $\Delta$ at $T_X G(p, n)$ to $T_Y G(p, n)$.
along the geodesic connecting $\mathcal{X}$ and $\mathcal{Y}$. According to Edelman et al. [4], letting $H = U \Sigma V^T$, parallel transport (denoted by $\tau$) can be computed via

$$
\tau \Delta(t) = -Y V \sin(t \Sigma) U^T \Delta + U \cos(t \Sigma) U^T + (I_n - U U^T) \Delta, \quad t \in [0, 1].
$$

This is not implemented in the manopt toolbox. For convenience, MATLAB code is listed below.

```matlab
function Delta = parallel_transport(O1,B,O2,t)

% PARALLEL_TRANSPORT parallel transports a tangent along geodesic.
%
% Input:
% O1 - (n x p) matrix (representer for subspace \mathcal{X})
% O2 - (n x p) matrix (representer for subspace \mathcal{Y})
% B - (n x p) matrix (tangent vector at \mathcal{X})
% t - time in [0,1]
%
% Output:
%
% Delta - (n x p) matrix (tangent vector B, transported to the
tangent space at \mathcal{Y})
%
% H = logmap(O1,O2); % compute direction matrix
% [U,S,V] = svd(H, 'econ');
% Sigmat = diag(S)*t;
% M = [diag(-sin(Sigmat)); diag(cos(Sigmat))];
% part0 =[O1*V U]*(M*(U'*B));
% part1 = B - U*(U'*B);
% Delta = part0 + part1;
end
```

5.4 Miscellaneous

The paper only gives a very concise description of the Riemannian structure of $\mathcal{G}(p,n)$. Here, we provide a more comprehensive description of the relationship between $\Delta \mathcal{Y}$ and $\mathcal{C}$ from [Paper, Eq. (16)]. Every tangent direction $\Delta \mathcal{Y} \in T_Y \mathcal{G}(p,n)$ at a point $Y \in \mathcal{G}(p,n)$ (represented by $Y$) can be written as

$$
\Delta Y = (I_n - YY^T)C
= (Y_\perp Y_\perp^T)C
$$

where $C$ is an arbitrary $(n \times p)$ matrix and $Y_\perp \in \mathbb{R}^{n \times (n-p)}$ denotes the orthogonal complement of $Y$. This can be interpreted as projecting an arbitrary $(n \times p)$ matrix into the subspace span$(Y_\perp)$. Now, if we take the canonical Euclidean metric, we get

$$
\text{tr} (\Delta_Y^T \Delta_Y) = \text{tr} [(I_n - YY^T)C]^T [(I_n - YY^T)C]
= \text{tr} [C^T (I_n - YY^T)C]
$$

since, for orthogonal projectors $P = (I_n - YY^T)$, it holds that $P = P^2$. This is the result listed in [Paper, Eq. (16)].
6 Additional curve-fitting comparisons

6.1 Synthetic data

In [Paper, Fig. 4], we present a comparison to the work of Su et al. [1] for only one specific choice of data-matching (controlled via $\lambda_1$) vs. smoothness (controlled via $\lambda_2$) balance (due to space restrictions), in particular, $(\lambda_1, \lambda_2) = (1, 0.1) \Rightarrow \lambda = \lambda_1 / \lambda_2 = 10$. Fig. 1 shows additional comparisons for various choices of $\lambda$ from $\lambda = 10$ in Fig. 1(a) to $\lambda = 0.1$ in Fig. 1(d).

6.2 Real (shape) data

In the paper, we compare our methods (Std-GGR, TW-GGR and CS-GGR) to Rentmeesters [2] and Su et al. [1] on the shape data of [Paper, Section 7]. For [2], the evaluation protocol is exactly the same as for our approaches, since we also obtain a parametric regression model.

For Su et al. [1], we note two subtle adjustments in the evaluation protocol. First, we cannot run full leave-one-subject-out cross-validation (CV), since we cannot leave-out the first or the last data point, simply because we obtain the sought-for curve as a collection of samples along that curve. Consequently, there would be no way to extrapolate to the left-out time points. For that reason, we always keep the data instances at the first and last time point and perform leave-one-subject-out CV on the remaining instances. Computation of the $R^2$ score and the MSE on the corpus callosum data is then done in the same way as for Std-GGR, TW-GGR and CS-GGR. Table 1 lists the detailed results of Su et al. [1] for various choices of $\lambda_1 / \lambda_2$.

<table>
<thead>
<tr>
<th>$\lambda_1$</th>
<th>$\lambda_2$</th>
<th>MSE (e-2)</th>
<th>$R^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.1</td>
<td>1.36</td>
<td>0.29</td>
</tr>
<tr>
<td>1</td>
<td>1.0</td>
<td>1.27</td>
<td>0.20</td>
</tr>
<tr>
<td>1</td>
<td>5.0</td>
<td>1.25</td>
<td>0.16</td>
</tr>
<tr>
<td>1</td>
<td>10.0</td>
<td>1.25</td>
<td>0.15</td>
</tr>
</tbody>
</table>

Table 1: Results of Su et al. [1] on the corpus callosum data of [Paper, Section 7] for various choices of $\lambda_1 / \lambda_2$. The best result of our methods, reported in [Paper, Table 2], is an MSE of 1.22e-2 (with an $R^2$ score of 0.15). We report the result of Su et al. at $\lambda_1 / \lambda_2 = 1/10$ (i.e., data matching vs. smoothness) in the paper (marked green here), since the corresponding MSE of 1.25e-2 is closest to the MSE of our best result. Any further increase in $\lambda_2$ did not lead to better MSE scores.

The second difference is in the evaluation protocol for the rat calvarium data. This dataset has multiple data instances at each time point. In particular, we have 18 data instances (i.e., the 18 individuals) at each of the 8 time points in the data. The method of Su et al. does not directly support such a setting. For that reason, we fit one model per individual, compute MSE and $R^2$ and then average the results. Table 2 reports the detailed results for various choices of $\lambda_1 / \lambda_2$. 
Table 2: Results of Su et al. [1] on the rat calvarium data of [Paper, Section 7] for various choices of $\lambda_1/\lambda_2$. The best result of our methods, reported in [Paper, Table 2], is an MSE of 1.2e-3 (with a $R^2$ score of 0.81). We report the result of Su et al. at $\lambda_1/\lambda_2 = 1/10$ (i.e., data matching vs. smoothness) in the paper (marked green here), since the corresponding MSE of 4.1e-3 is closest to the MSE of our best result. Any further increase in $\lambda_2$ did not lead to better MSE scores.

References


Figure 1: Additional comparisons to Su et al. [1] for various choices of $\lambda_1, \lambda_2$ and different levels of Gaussian noise added to the signal frequencies. The $x$-axis shows the index of 20 different signal frequencies (obtained by sampling a sine function), the $y$-axis shows the real-part of the largest eigenvalue of the state-transition matrix $A$, reconstructed from the observability matrices of the associated linear dynamical systems (Fig. 1(a) is shown in [Paper, Fig. 4]).