MATH-512 - Project 2
Estimating rotations from relative measurements

Luca Bulgarelli Tommaso Fioratti Greta Pizzichini

May 2025

The aim of this project is to recover m rotations in $\mathrm{SO}(d)$ for noisy measurements of some of the relative rotations. This goal is pursued through maximum likelihood estimation (MLE). We will use the log-likelihood function

$$f(X) = \sum_{(i,j)\in E} \log\left(p(X_i^\top H_{i,j} X_j)\right)$$

defined on the manifold $\mathcal{M} = \{(X_1, X_2, \ldots, X_m) \text{ such that } X_i = R_i \text{ for } i \in J\}$, where J is the set of indices of the know rotations, called anchors.

1 Application description and warmup problems

1) Let us assume that there is no noise in the measurements, that $\mathcal{G}$ is complete and that no anchor is provided. Our aim is to estimate the rotations $R_1, R_2, \ldots, R_m$. Consider the following matrix

$$H = \begin{bmatrix} I_d & H_{1,2} & H_{1,3} & \ldots & H_{1,m} \\ H_{2,1} & I_d & H_{2,3} & \ldots & H_{2,m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ H_{m,1} & H_{m,2} & H_{m,3} & \ldots & I_d \end{bmatrix}$$

where $H_{i,j} = R_i R_j^\top$ for all $i, j \in \{1, \ldots, m\}$. Since $H_{i,j} = R_i R_j^\top = (R_j R_i^\top)^\top = H_{j,i}^\top$, the matrix H is symmetric. In addition $H_{i,j} = R_i R_j^\top = R_i R_{i+1}^\top R_{i+1} \ldots R_{j-1}^\top R_{j-1} R_j^\top = H_{i,i+1} H_{i+1,i+2} \ldots H_{j-1,j}$. This shows the following equivalence:

$$\text{knowing } H_{i,j} \text{ for all } i, j \in \{1, \ldots, m\} \iff \text{knowing } H_{1,2}, H_{2,3}, \ldots, H_{m-1,m}$$

In the end, we have $m - 1$ conditions (the matrices on the upper-diagonal of H) to find m rotations. This is enough to conclude that, without more information, is not possible to estimate the rotations $R_1, \ldots, R_m$ uniquely.

2) From the conclusion of the previous point, we can state that the presence of one single anchor is sufficient to estimate $R_1, \ldots, R_m$ uniquely if there is no noise. In fact, let us suppose without loss of generality that the rotation R_1 is the anchor, then the following algorithm recovers all R_i where $i \in \{2, \ldots, m\}$:

for i from 2 to m
    $R_i = H_{i,i-1} R_{i-1}$
end

In other words, if we interpret the matrices $H_{i,j}$ as arches of the complete graph $\mathcal{G}$ and R_i as the vertices, then infinitely many paths exist from the anchor R_1 to R_i for all $i \in \{2, \dots, m\}$. In the case of no noise all these paths lead to the same matrix R_i, thus we can choose randomly one of this.

3) Using the algorithm above, we obtain:

$$\begin{aligned}
&\widehat{R}_1 = R_1, \\
&\widehat{R}_2 = H_{2,1}\widehat{R}_1 = (R_2 Z_{2,1} R_1^\top) R_1 = R_2 Z_{2,1}, \\
&\widehat{R}_3 = H_{3,2}\widehat{R}_2 = (R_3 Z_{3,2} R_2^\top)(R_2 Z_{2,1}) = R_3 Z_{3,2} Z_{2,1}, \\
&\quad \vdots \\
&\widehat{R}_k = H_{k,k-1}\widehat{R}_{k-1} = \cdots = R_k Z_{k,k-1} Z_{k-1,k-2} \dots Z_{2,1}
\end{aligned}$$

The issues with this algorithm are:

- We do not recover the true rotation R_k exactly: due to the presence of noise, the estimated rotations $\widehat{R}_k$ differ from the true ones.
- The estimation error accumulate along the path: the farther the rotation R_k is from the anchor (in terms of graph distance), the larger the error becomes, because the noise terms propagate and multiply along the path.
- The final estimate $\widehat{R}_k$ depends on the specific path chosen from the anchor to the vertex k: in the presence of noise, different paths yield different products of the noise matrices $Z_{j,j-1}$, resulting in different estimates.

2 The geometry of the manifold

4) In order to understand what the tangent spaces of $\mathcal{M}$ are, let us find a local defining function h for $\mathcal{M}$ around one of its points X. For the sake of simplicity, let us assume that $J = \{1, 2, \dots, k\}$, i.e. the anchor set is made up of the first $k = |J|$ indices. Thanks to this assumption if $X \in \mathcal{M}$ then $X = (R_1, R_2, \dots, R_k, X_{k+1}, \dots X_m)$ with $X_i \in \mathrm{SO}(d)$ for $i \in \{k+1, ..., m\}$. In the following we will refer to the first k components of X as R_i or X_i. Let $U = (U_1, \dots, U_k, U_{k+1}, \dots, U_m)$ be a neighborhood of X in $\mathcal{E} = (\mathbb{R}^{d\times d})^m$, in particular U_i is a neighborhood of X_i in $\mathbb{R}^{d\times d}$. Consider the function

$$\begin{aligned}
h : U &\longrightarrow (\mathbb{R}^{d\times d})^k \times (\mathbb{R}^{\frac{d(d+1)}{2}})^{(m-k)} \\
X &\longmapsto h(X) = (R_1 - X_1, \dots, R_k - X_k, X_{k+1}^\top X_{k+1} - \mathbb{I}_d, \dots, X_m^\top X_m - \mathbb{I}_d)
\end{aligned}$$

where we used that $h_i : U_i \to \mathbb{R}^{\frac{d(d+1)}{2}}$, $h_i(X) = X^\top X - I_d$ is a local defining function for the manifold $\mathrm{SO}(d)$ around X_i for $i \in \{k+1, \dots, m\}$[1]. Let us analyze the properties of the function h:

- h is smooth: each component is smooth (the functions h_i themselves are smooth since they are local defining functions of $\mathrm{SO}(d)$).

[1]This result is stated in the 7.4 paragraph of the book "An introduction to optimization on manifolds", Nicolas Boumal

- $h^{-1}(0) = (R_1, \dots, R_k, h_{k+1}^{-1}(0), \dots, h_m^{-1}(0)) = (R_1, \dots, R_k, U_{k+1} \cap \mathrm{SO}(d), \dots, U_m \cap \mathrm{SO}(d)) = U \cap \mathcal{M}$
- $Dh(X) = \mathrm{diag}(-\mathrm{Id}, \dots, -\mathrm{Id}, Dh_{k+1}(X_{k+1}), \dots, Dh_m(X_m))$, thus $Dh(X)$ has maximum rank.

This shows that h is a local defining function for $\mathcal{M}$ around X. Thus the dimension of $\mathcal{M}$ is $md^2 - kd^2 - (m-k)\frac{d(d+1)}{2} = (m-k)\frac{d(d-1)}{2}$ Thanks to what we have studied during lectures:

$$\begin{aligned} T_X\mathcal{M} &= \ker Dh(X) = \\ &= \{(0_d, \dots, 0_d, V_{k+1}, \dots V_m) : V_i \in \ker Dh_i(X_i) \text{ for all } i = k+1, \dots, m\} = \\ &= \{(0_d, \dots, 0_d, V_{k+1}, \dots V_m) : V_i \in T_{X_i}\mathrm{SO}(d) \text{ for all } i = k+1, \dots, m\} = \\ &= \{(0_d, \dots, 0_d, V_{k+1}, \dots V_m) : V_i \in X_i\mathrm{Skew}(d) \text{ for all } i = k+1, \dots, m\} \end{aligned}$$

where 0_d is the $d \times d$ matrix whose elements are all zeros, $\mathrm{Skew(d)} = \{X \in \mathbb{R}^{d\times d} : X^\top = -X\}$ and $X\mathrm{Skew(d)} = \{X\Omega \in \mathbb{R}^{d\times d} : \Omega \in \mathrm{Skew}(d)\}$.

5) Let $X = (R_1, \dots, R_k, X_{k+1}, \dots X_m) \in \mathcal{M}$. Using the notation $Y = (Y_1, \dots, Y_m)$ and $\mathrm{skew(X)} = \frac{X - X^\top}{2}$, the function

$$\begin{aligned} \mathrm{Proj}_X : &\left(\mathbb{R}^{d\times d}\right)^m \longrightarrow T_X\mathcal{M} \\ &Y \longmapsto (0_d, \dots, 0_d, X_{k+1}\mathrm{skew}(X_{k+1}^\top Y_{k+1}), \dots, X_m\mathrm{skew}(X_m^\top Y_m)) \end{aligned}$$

is the orthogonal projector [2]. In fact, from the previous point, $\mathrm{Proj}_X(Y) \in T_X\mathcal{M}$.

6) As a second-order retraction we propose the following map

$$\begin{aligned} &R : T\mathcal{M} \longrightarrow \mathcal{M} \\ &(X, V) \longmapsto (R_1 \exp(R_1^\top V_1,) \dots, R_k \exp(R_k^\top V_k), X_{k+1} \exp(X_{k+1}^\top V_{k+1}), \dots, X_m \exp(X_m^\top V_m)) \end{aligned}$$

where the notation $\exp(X)$ stands for the exponential of matrices, i.e.

$$\exp(X) = e^X = \sum_{i=0}^{\infty} \frac{1}{i!} X^i$$

The two following observations assure that the map is well posed and it is a retraction:

1. The matrices $R_i^\top V_i$ and $X_i^\top V_i$ are squared matrices, so their powers always exist. In addition, since the radius of convergence of the series which defines the exponential is infinity, $\exp(X)$ always exists.
2. Let $X \in \mathcal{M}$ and $V \in T_X\mathcal{M}$, then $V = (0_d, \dots, 0_d, X_{k+1}U_{k+1}, \dots, X_mU_m)$ where $U_i \in \mathrm{Skew}(d)$ for all $i \in \{k+1, \dots, m\}$. Through basic notions of linear algebra [3], we get:

   components $1, \dots, k$: $R_i \exp(R_i^\top V_i) = R_i \exp(R_i^\top 0_d) = R_i$

   components $k+1, \dots, m$: $X_i \exp(X_i^\top V_i) = X_i \exp(X_i^\top X_i U_i) = X_i \exp(U_i) \in \mathrm{SO}(d)$

[2] As before, we are using results stated in the paragraph 7.4 of the book "An introduction to optimization on manifolds", Nicolas Boumal

[3] We use the trivial equality: $\exp(0_d) = I_d$. In addition, let $U \in \mathrm{Skew}(d)$, then $\left(\exp(U)\right)^\top \exp(U) = \exp(U^\top)\exp(U) = \exp(-U)\exp(U) = \exp(-U+U) = \exp(0_d) = I_d$. Thus, the exponential of a skew-symmetric matrix is an orthogonal matrix.

Thus, $R_X(V) \in \mathcal{M}$.
Let $c(t) = R_X(tV)$ be a curve on the manifold $\mathcal{M}$. Then $c(0) = X$. Moreover, if we compute its derivative we obtain

$$\begin{aligned} c'(t) &= (0_d, \dots, 0_d, X_{k+1}X_{k+1}^\top V_{k+1} \exp(tX_{k+1}^\top V_{k+1}), \dots, X_m X_m^\top V_m \exp(tX_m^\top V_m)) = \\ &= (0_d, \dots, 0_d, V_{k+1} \exp(tX_{k+1}^\top V_{k+1}), \dots, V_m \exp(tX_m^\top V_m)) \end{aligned}$$

which yields $c'(0) = V$ in compliance with the definition of retraction.

To verify that it is a second-order retraction, we compute the second order derivative

$$\begin{aligned} c''(t) &= \mathrm{Proj}_X\left(\frac{d}{dt}c'(t)\right) = \\ &= \mathrm{Proj}_X\left(0_d, \dots, 0_d, \frac{d}{dt}\left(X_{k+1}X_{k+1}^\top V_{k+1} \exp(tX_{k+1}^\top V_{k+1})\right), \dots, \frac{d}{dt}\left(X_m X_m^\top V_m \exp(X_m^\top V_m)\right)\right) = \\ &= \mathrm{Proj}_X\left(0_d, \dots, 0_d, X_{k+1}(X_{k+1}^\top V_{k+1})^2 \exp(tX_{k+1}^\top V_{k+1})\right), \dots, X_m(X_m^\top V_m)^2 \exp(X_m^\top V_m)\Big) = \\ &= \Big(0_d, \dots, 0_d, X_{k+1}\mathrm{skew}\left((X_{k+1}^\top V_{k+1})^2 \exp(tX_{k+1}^\top V_{k+1})\right), \dots, X_m \mathrm{skew}\left((X_m^\top V_m)^2 \exp(tX_m^\top V_m)\right)\Big) \end{aligned}$$

For $t = 0$ we have

$$c''(0) = \left(0_d, \dots, 0_d, X_{k+1}\mathrm{skew}((X_{k+1}^\top V_{k+1})^2), \dots, X_m \mathrm{skew}((X_m^\top V_m)^2)\right) = 0_{d \times dm}$$

since the skew-symmetric part of the square of a skew-symmetric matrix is equal to the zero matrix.

7) The points $X \in \mathcal{M}$ will be stored as 3D vectors with sizes $d \times d \times m$, and the same structure will be used for the tangent vectors. The memory needed to store a single element of the manifold (or the tangent) is $\mathcal{O}(d^2 m)$.

8) The inner product is computed as follows

```
M.inner = @(X, U, V)
    sum(multitrace(multiprod(multitransp(U(:,:,I)), V(:,:,I))));
```

where I is the set of indices of the non-anchored rotations.

9) The factory for the $\mathrm{SO}(d)^m$ manifold is implemented as follows

```
function M = ManyRotationsFactory(d, m, J, R)

    k = length(J);
    I = setdiff(1:m, J);

    % Inner product and norm
    M.inner = @(X, U, V)
        sum(multitrace(multiprod(multitransp(U(:,:,I)),
        V(:,:,I))));
    M.norm  = @(X, U) sqrt(M.inner(X, U, U));

    % Projection
    M.proj = @projection;
```

```
    function P = projection(X, H)
        P = zeros(d,d,m);
        P(:,:,I) = multiprod(X(:,:,I),
           multiskew(multiprod(multitransp(X(:,:,I)), H(:,:,I))));
        P(:,:,J) = 0;
    end

    % Retraction
    M.retr = @retraction;
    function Q = retraction(X, H, t)
        if nargin < 3
            t = 1;
        end
        Q = X;
        for j = I
            Q(:,:,j) = X(:,:,j) * expm(t * X(:,:,j)' * H(:,:,j));
                % safer than using Q(:,:,j) on RHS
        end
    end
    M.retr2 = M.retr;

    % Random point
    M.rand = @randomRot;
    function X = randomRot()
        X = zeros(d, d, m);
        X(:,:,J) = R;
        X(:,:,I) = randrot(d, length(I));
    end

    % Random tangent vector
    M.randvec = @randomTang;
    function V = randomTang(X)
        V = zeros(d, d, m);
        V(:,:,I) = multiprod(X(:,:,I), randskew(d, numel(I)));
    end

    % Zero tangent vector
    M.zerovec = @(x) zeros(d, d, m);

    % Linear combination
    M.lincomb = @matrixlincomb;

    % Retangentialization and gradient conversions
    M.tangent = M.proj;
    M.egrad2rgrad = M.proj;

    % Euclidean to Riemannian Hessian
    M.ehess2rhess = @ehess2rhess;
    function Rhess = ehess2rhess(X, Egrad, Ehess, U)
        Xt = multitransp(X);
        XtEgrad = multiprod(Xt,Egrad);
        skewXtEgrad = multiskew(XtEgrad);
        UskewXtEgrad = multiprod(U,skewXtEgrad);
        Ut = multitransp(U);
        UtEgrad = multiprod(Ut,Egrad);
        skewUtEgrad = multiskew(UtEgrad);
        XskewUtEgrad = multiprod(X,skewUtEgrad);
```

```
        XtEhess = multiprod(Xt,Ehess);
        skewXtEhess = multiskew(XtEhess);
        XskewXtEhess = multiprod(X,skewXtEhess);
        Rhess = M.proj(X,UskewXtEgrad + XskewUtEgrad +
            XskewXtEhess);

    end

    M.tangent2ambient_is_identity = true;
    M.tangent2ambient = @(X, U) U;
    M.transp = @(x1, x2, eta) multiprod(x2,
        multiprod(multitransp(x1), eta));
    M.dim = (m - k) * d * (d - 1) / 2;
end
```

3 The cost function

10) The optimization problem admits a global maximizer by the Weierstrass theorem. As a first step, we demonstrate that $\mathcal{M}$ is a compact space.

- $\mathrm{SO}(d)$ is closed because it is described by two closed conditions. It is also bounded [4] because if $X \in \mathrm{SO}(d)$, then $\|X\|_F^2 = \mathrm{Tr}(X^\top X) = \mathrm{Tr}(I_d) = d$, so $\|X\|_F = \sqrt{d}$. Thus, $\mathrm{SO}(d) \subset B(0, \sqrt{d}+1)$ where $B(0,r)$ is the open ball of $\mathbb{R}^{d\times d}$ centered at 0 with radius r. These two conditions assure that $\mathrm{SO}(d)$ is compact in $\mathbb{R}^{d\times d}$.
- $\forall i \in \{1,\dots,k\}$ the single rotation R_i is a compact set of $\mathbb{R}^{d\times d}$ because it is closed and bounded.

Since $\mathcal{M}$ is a finite product of compact spaces, it is itself a compact space. The second and last step to apply the Weierstrass theorem is to demonstrate that the objective function f is continuous on $\mathcal{M}$. The cost function is a finite sum of logarithms whose arguments are strictly positive values. The distribution is continuous since the trace of a matrix is a continuous function. In the end, also the product of two matrices and the transposition are continuous. These arguments assure the continuity of f on $\mathcal{M}$. We conclude that, since $\mathcal{M}$ is compact and f is continuous, f attains its maximum on $\mathcal{M}$.
If there are no anchors, the maximizer is not unique. Indeed, consider the right action of $\mathrm{SO}(d)$ on $\mathcal{M}$:

$$\begin{aligned} \Gamma : \mathcal{M}\times\mathrm{SO}(d) &\longrightarrow \mathcal{M} \\ (X, R) &\longmapsto (R_1R, \dots, R_kR, X_{k+1}R, \dots, X_mR) \end{aligned}$$

where $X = (R_1, \dots, R_k, X_{k+1}, \dots, X_m)$ as above. The objective function f is invariant under Γ, i.e. $f(X) = f(\Gamma(X,R))$ for all $R \in \mathrm{SO}(d)$, as can be deduced by equation (1)

$$\mathrm{Tr}\left((X_iR)^\top H_{i,j}(X_jR)\right) = \mathrm{Tr}\left(R^\top X_i^\top H_{i,j}X_jR\right) = \mathrm{Tr}\left(X_i^\top H_{i,j}X_jRR^\top\right) = \mathrm{Tr}\left(X_i^\top H_{i,j}X_j\right) \quad (1)$$

In equation (1) the cyclic property of the trace has been used to obtain the second equality [5]. In particular,

$$(R_1, \dots, R_k, X_{k+1}, \dots, X_m) \text{ maximizer } \Rightarrow (R_1R, \dots, R_kR, X_{k+1}R, \dots, X_mR) \text{ maximizer}$$

[4] Here, we consider the space $\mathbb{R}^{d\times d}$ with the Frobenius norm, as indicated in the text of the project.
[5] The cyclic property is: $\mathrm{Tr}(ABCD) = \mathrm{Tr}(BCDA) = \mathrm{Tr}(CDAB) = \mathrm{Tr}(DABC)$.

This demonstrates that the maximizer is not unique.
Now suppose that there is at least one anchor and that there is no noise. Then

$$X_{\max} = (R_1, \dots, R_k, H_{k+1,1}R_1, \dots, H_{m,1}R_1) = \arg\max_{X \in \mathcal{M}} f(X) \tag{2}$$

is the only maximizer of f on the manifold $\mathcal{M}$. To prove this statement it is necessary to go through two points:

1. $X_{\max}$ is a point where f assumes its maximum. Indeed, using the notation $X_i = R_i$ if $i \in \{1, \dots, k\}$ and $X_i = H_{i,1}R_1$ if $i \in \{k+1, \dots, m\}$ the equality $X_i^\top H_{i,j} X_j = I_d$ holds for every $i, j \in \{1, \dots, m\}$. Then [6]

$$p(X_i^\top H_{i,j} X_j) = p(I_d) = \max_{Z \in \mathrm{SO}(d)} p(Z) \tag{3}$$

   Since the logarithm is an increasing function, this argument demonstrates (2).

2. $X_{\max}$ is the only maximizer. Indeed, let us suppose that there is another maximizer $\hat{X} = (\hat{X}_1, \dots, \hat{X}_k, \hat{X}_{k+1}, \dots, \hat{X}_m) \in \mathcal{M}$. Then it must be that $\hat{X}_i^\top H_{i,j} \hat{X}_j = I_d$ for all $i, j \in \{1, \dots, m\}$(otherwise $f(\hat{X})$ would not be the maximum of f on $\mathcal{M}$). Recall that $\hat{X}_i = X_i = R_i$ for all $i \in \{1, \dots, k\}$. In addition

$$X_i^\top H_{i,j} X_j = I_d = \hat{X}_i^\top H_{i,j} \hat{X}_j \quad \text{for all } i, j \in \{1, \dots, m\}$$

   In particular, $R_1^\top H_{1,j} \hat{X}_j = R_1^\top H_{1,j} X_j$ which implies $X_j = \hat{X}_j$ for all $j \in \{k+1, \dots, m\}$. In conclusion: $X_i = \hat{X}_i$ for all $i \in \{1, \dots, m\}$ and $X_{\max} = \hat{X}$.

11) Noting that

$$\begin{aligned}
\sum_{(i,j) \in E} \|H_{i,j} X_j - X_i\|_F^2 &= \sum_{(i,j) \in E} \mathrm{Tr}\left((H_{i,j} X_j - X_i)^\top (H_{i,j} X_j - X_i)\right) = \\
&= \sum_{(i,j) \in E} \left(\mathrm{Tr}(X_j^\top H_{i,j}^\top H_{i,j} X_j) + \mathrm{Tr}(X_i^\top X_i) - 2\,\mathrm{Tr}(X_i^\top H_{i,j} X_j)\right) = \\
&= \sum_{(i,j) \in E} \left(2d - 2\,\mathrm{Tr}(X_i^\top H_{i,j} X_j)\right) = \\
&= 2d|E| - 2 \sum_{(i,j) \in E} \mathrm{Tr}(X_i^\top H_{i,j} X_j)
\end{aligned}$$

where the facts $H_{i,j}^\top H_{i,j} = I_d$ and $X_i^\top X_i = I_d$ has been used. The identity above and the relation

$$\log p(Z) = -\log c_d(\kappa) + \kappa\,\mathrm{Tr}(Z)$$

[6] From classical results of linear algebra for every $Z \in \mathrm{SO}(d)$ exists $P \in \mathbb{R}^{d \times d}$ such that $P^\top P = I_d$ and $P^\top Z P = \mathrm{diag}(B_1, \dots, B_{s-1}, B_s)$ where $B_i = \begin{pmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{pmatrix}$ is a 2×2 rotation for all $i \in \{1, \dots, s-1\}$ and $B_s = 1$ if d is odd or a rotation of angle θ_s if d is even. Then $\mathrm{Tr}(Z) = \sum_{i=1}^{s-1} \mathrm{Tr}(B_i) + \mathrm{Tr}(B_s) = 2\sum_{i=1}^{s-1} \cos\theta_i + \mathrm{Tr}(B_s)$. In both cases, d odd and d even, the maximum is reached when $\cos\theta_i = 1 \implies \theta_i = 0$ for all i. Therefore, $P^\top Z P = I_d \Rightarrow Z = I_d$. This argument shows that the maximum of the trace on $\mathrm{SO}(d)$ is attained on the identity matrix I_d.

lead to the final argument:

$$\arg\max_{X\in\mathcal{M}} \sum_{(i,j)\in E} \log(p(X_i^\top H_{ij} X_j)) = \arg\max_{X\in\mathcal{M}} \sum_{(i,j)\in E} \mathrm{Tr}(X_i^\top H_{ij} X_j) = \arg\min_{X\in\mathcal{M}} \sum_{(i,j)\in E} \|H_{ij}X_j - X_i\|_F^2$$

12) Let us consider $\mathcal{M}$ as a Riemannian submanifold of $\mathbb{R}^{d\times d}$ as indicated in the text of the project. The strategy for computing the Riemannian gradient of f in $X \in \mathcal{M}$ is to find a smooth extension $\overline{f}$ of f on the whole $\mathbb{R}^{d\times d}$, to compute $\mathrm{grad}\overline{f}(X)$ and then to project it into $T_X\mathcal{M}$. As an extension of f let us take:

$$\begin{aligned} \overline{f} : (\mathbb{R}^{d\times d})^m &\longrightarrow \mathbb{R} \\ X &\longmapsto \sum_{(i,j)\in E} \log\left(p(X_i^\top H_{i,j} X_j)\right) \end{aligned}$$

where $X = (X_1, \dots X_m)$. Note that $\overline{f}$ has the same expression that f, therefore it is clearly an extension. In addition is smooth thanks to the considerations made above. As seen in exercises of week 4

$$\mathrm{grad}\overline{f}(X) = \left(\mathrm{grad}\overline{f}_1(X_1), \mathrm{grad}\overline{f}_2(X), \dots, \mathrm{grad}\overline{f}_m(X_m)\right) \tag{4}$$

where we use the notation $\overline{f}_h : \mathbb{R}^{d\times d} \to \mathbb{R}$ to denote the function obtained from $\overline{f}$ by fixing all the inputs except the hth input. With this in mind, the next step is to compute

$$\mathrm{grad}\overline{f}_h(X) = \begin{pmatrix} \frac{\partial \overline{f}_h}{\partial x_{1,1}}(X) & \frac{\partial \overline{f}_h}{\partial x_{1,2}}(X) & \dots & \frac{\partial \overline{f}_h}{\partial x_{1,d}}(X) \\ \frac{\partial \overline{f}_h}{\partial x_{2,1}}(X) & \frac{\partial \overline{f}_h}{\partial x_{2,2}}(X) & \dots & \frac{\partial \overline{f}_h}{\partial x_{2,d}}(X) \\ \vdots & \vdots & & \vdots \\ \frac{\partial \overline{f}_h}{\partial x_{d,1}}(X) & \frac{\partial \overline{f}_h}{\partial x_{d,2}}(X) & \dots & \frac{\partial \overline{f}_h}{\partial x_{d,d}}(X) \end{pmatrix} \tag{5}$$

Since $H_{i,j} = H_{j,i}^\top$, the graph $\mathcal{G}$ is not oriented, thus the two elements (i,j) and (j,i) are the same (note also that $p(X_i^\top H_{i,j} X_j) = p(X_j^\top H_{j,i} X_i)$ because the trace of a matrix is invariant under transposition). This argument allows us to suppose, without loss of generality, that if h is present in one element of E, it is always in the first position. Consider $s, t \in \{1, \dots, d\}$, then

$$\begin{aligned} \frac{\partial \overline{f}_h}{\partial x_{s,t}}(X_h) &= \frac{\partial}{\partial x_{s,t}} \left(\sum_{(h,j)\in E} \log\left(p(X_h^\top H_{h,j} X_j)\right) + \sum_{(i,j)\in E, i\neq h, j\neq h} \log\left(p(X_i^\top H_{i,j} X_j)\right) \right) = \\ &= \sum_{(h,j)\in E} \frac{\partial}{\partial x_{s,t}} \log\left(p(X_h^\top H_{h,j} X_j)\right) \end{aligned} \tag{6}$$

For the sake of simplicity let us compute the partial derivative in a point $X \in \mathbb{R}^{d\times d}$:

$$\begin{aligned}
&\frac{\partial}{\partial x_{s,t}} \log\big(p(X^\top H_{h,j} X_j)\big) = \frac{1}{p(X^\top H_{h,j} X_j)} \frac{\partial}{\partial x_{s,t}} p(X^\top H_{h,j} X_j) = \\
&= \frac{1}{p(X^\top H_{h,j} X_j)} \left(q \frac{\partial}{\partial x_{s,t}} p_{\kappa_1}(X^\top H_{h,j} X_j) + (1-q) \frac{\partial}{\partial x_{s,t}} p_{\kappa_2}(X^\top H_{h,j} X_j) \right) = \\
&= \frac{1}{p(X^\top H_{h,j} X_j)} \left(\frac{q\kappa_1}{c_d(\kappa_1)} e^{\kappa_1 \mathrm{Tr}(X^\top H_{h,j} X_j)} \frac{\partial}{\partial x_{s,t}} \mathrm{Tr}(X^\top H_{h,j} X_j) + \right. \\
&\left. + \frac{(1-q)\kappa_2}{c_d(\kappa_2)} e^{\kappa_2 \mathrm{Tr}(X^\top H_{h,j} X_j)} \frac{\partial}{\partial x_{s,t}} \mathrm{Tr}(X^\top H_{h,j} X_j) \right) = \\
&= \frac{1}{p(X^\top H_{h,j} X_j)} \left(\frac{q\kappa_1}{c_d(\kappa_1)} e^{\kappa_1 \mathrm{Tr}(X^\top H_{h,j} X_j)} + \right. \\
&\left. + \frac{(1-q)\kappa_2}{c_d(\kappa_2)} e^{\kappa_2 \mathrm{Tr}(X^\top H_{h,j} X_j)} \right) \frac{\partial}{\partial x_{s,t}} \mathrm{Tr}(X^\top H_{h,j} X_j) = \\
&= \frac{1}{p(X^\top H_{h,j} X_j)} \left(\frac{q\kappa_1}{c_d(\kappa_1)} e^{\kappa_1 \mathrm{Tr}(X^\top H_{h,j} X_j)} + \frac{(1-q)\kappa_2}{c_d(\kappa_2)} e^{\kappa_2 \mathrm{Tr}(X^\top H_{h,j} X_j)} \right) (H_{h,j} X_j)_{s,t} = \\
&= g(X^\top H_{h,j} X_j)(H_{h,j} X_j)_{s,t}
\end{aligned} \tag{7}$$

where $g(X^\top H_{h,j} X_j) = \frac{1}{p(X^\top H_{h,j} X_j)} \left(\frac{q\kappa_1}{c_d(\kappa_1)} e^{\kappa_1 \mathrm{Tr}(X^\top H_{h,j} X_j)} + \frac{(1-q)\kappa_2}{c_d(\kappa_2)} e^{\kappa_2 \mathrm{Tr}(X^\top H_{h,j} X_j)} \right)$. More details of linear algebra regarding $\frac{\partial}{\partial x_{s,t}} \mathrm{Tr}(X^\top H_{h,j} X_j)$ are in the footnote [7]. Thus, the equation (6) becomes:

$$\frac{\partial \overline{f}_h}{\partial x_{s,t}}(X_h) = \sum_{(h,j)\in E} g(X_h^\top H_{h,j} X_j)(H_{h,j} X_j)_{s,t}$$

Now we can rewrite expression (5) as:

$$\mathrm{grad} \overline{f}_h(X_h) = \sum_{(h,j)\in E} g(X_h^\top H_{h,j} X_j) H_{h,j} X_j$$

In conclusion:

$$\mathrm{grad} f(X) = \big(0_d, \dots, 0_d, A_{k+1}, \dots A_m\big)$$

where

$$\begin{aligned}
A_h &= X_h \mathrm{skew}\left(X_h^\top \sum_{(h,j)\in E} g(X_h^\top H_{h,j} X_j) H_{h,j} X_j \right) = \\
&= \sum_{(h,j)\in E} g(X_h^\top H_{h,j} X_j) X_h \mathrm{skew}\big(X_h^\top H_{h,j} X_j\big)
\end{aligned}$$

[7] To compute $\frac{\partial}{\partial x_{s,t}} \mathrm{Tr}(X^\top H_{h,j} X_j)$ is necessary to understand where the element $x_{s,t}$ of X is present in the product $X^\top H_{h,j} X_j$. Since the trace takes into consideration only the diagonal element, let us study the element $(X^\top H_{h,j} X_j)_{a,a} = \sum_{k=1}^{d} (X^\top)_{a,k} (H_{h,j} X_j)_{k,a} = \sum_{k=1}^{d} (X)_{k,a} (H_{h,j} X_j)_{k,a} = \sum_{k=1}^{d} x_{k,a} (H_{h,j} X_j)_{k,a}$. From this expression we easily deduce that $\frac{\partial}{\partial x_{s,t}} \mathrm{Tr}(X^\top H_{h,j} X_j) = (H_{h,j} X_j)_{s,t}$.

As a particular instance, we now give the expression of $\mathrm{grad} f(X)$ where $p = p_\kappa$. Defining $g^L(X_h^\top H_{h,j} X_j) = \frac{1}{p_\kappa(X_h^\top H_{h,j} X_j)} \frac{\kappa}{c_d(\kappa)} e^{\kappa \mathrm{Tr}(X_h^\top H_{h,j} X_j)}$, the following equality holds:

$$\mathrm{grad} f(X) = (0_d, \dots, 0_d, A_{k+1}^L, \dots, A_m^L)$$

where

$$A_h^L = \sum_{(h,j)\in E} g^L(X_h^\top H_{h,j} X_j) X_h \mathrm{skew}\big(X_h^\top H_{h,j} X_j\big)$$

13) Let $(X, U) \in T_X\mathcal{M}$, we now want to derive an expression for the Riemannian Hessian-vector product $\mathrm{Hess} f(X)[U]$. As seen in the lectures

$$\mathrm{Hess} f(X)[U] = \nabla_U \mathrm{grad} f(X) = \mathrm{Proj}_X(D\overline{\mathrm{grad} f}(X)[U])$$

where, for the second equality, we exploited that $\mathcal{M}$ is an embedded submanifold of the Euclidean space $(\mathbb{R}^{d\times d})^m$ equipped with the scalar product $\langle X, Y\rangle = \sum_{i=1}^m \mathrm{Tr}(X_i^\top Y_i)$. As first step, define the smooth extension of $\mathrm{grad} f$ with the same expression of $\mathrm{grad} f$, now $\overline{\mathrm{grad} f} : (\mathbb{R}^{d\times d})^m \to (\mathbb{R}^{d\times d})^m$. Our aim is now to compute $D\overline{\mathrm{grad} f}(X)[U]$ for $X \in \mathcal{M}$ and $U \in T_X\mathcal{M}$. In the following we will use the notation

$$\gamma(X_i^\top H_{i,j} X_j) = \frac{q\kappa_1}{c_d(\kappa_1)} e^{\kappa_1 \mathrm{Tr}(X_i^\top H_{i,j} X_j)} + \frac{(1-q)\kappa_2}{c_d(\kappa_2)} e^{\kappa_2 \mathrm{Tr}(X_i^\top H_{i,j} X_j)}$$

$$\Downarrow$$

$$g(X_h^\top H_{h,l} X_l) = \frac{\gamma(X_h^\top H_{h,l} X_l)}{p(X_h^\top H_{h,l} X_l)}$$

The manifold $\mathcal{M}$ can be seen as a product of k fixed rotations and $m-k$ manifolds $\mathrm{SO}(d)$. This structure allows us to work component-wise. In addition we know that $(\overline{\mathrm{grad} f}(X))_i = 0_d$ for all $i = 1, \dots k$, this implies that $(\mathrm{Hess} f(X)[U])_i = 0_d$ for all $X \in \mathcal{M}$ and for all $U \in T_X\mathcal{M}$. What we need now to achieve our objective is to compute $(\mathrm{Hess} f(X)[U])_i$ for all $i = k+1, \dots, m$, i.e. for the components which are not anchors. We start from i$^{\text{th}}$ component of the extended gradient of f at X:

$$\big(\overline{\mathrm{grad} f}(X)\big)_i = X_i \mathrm{skew}\left(X_i^\top \sum_{(i,j)\in E} g(X_i^\top H_{i,j} X_j) H_{i,j} X_j \right)$$

Now we can compute the i$^{\text{th}}$ component of the differential of $\overline{\mathrm{grad} f}(X)$ in U:

$$\begin{aligned}\big(D\overline{\mathrm{grad} f}(X)[U]\big)_i = {} & U_i \mathrm{skew}\left(X_i^\top \sum_{(i,j)\in E} g(X_i^\top H_{i,j} X_j) H_{i,j} X_j \right) + \\ & + X_i \mathrm{skew}\left(U_i^\top \sum_{(i,j)\in E} g(X_i^\top H_{i,j} X_j) H_{i,j} X_j \right) + \\ & + X_i \mathrm{skew}\left(X_i^\top D\left(\sum_{(i,j)\in E} g(X_i^\top H_{i,j} X_j) H_{i,j} X_j \right)(X)[U] \right)\end{aligned} \tag{8}$$

Let us compute the last term of expression (8), which is the Euclidean Hessian of f at X evaluated in U:

$$
\begin{aligned}
&D\left(\sum_{(i,j)\in E} g(X_i^\top H_{i,j}X_j)H_{i,j}X_j\right)(X)[U] = D\left(\sum_{(i,j)\in E}\frac{\gamma(X_h^\top H_{h,l}X_l)}{p(X_h^\top H_{h,l}X_l)}H_{i,j}X_j\right)(X)[U] = \\
&= \sum_{(i,j)\in E}\Bigg(-\frac{\gamma^2(X_h^\top H_{i,j}X_j)}{p^2(X_i^\top H_{i,j}X_j)}\big(\langle U_i, H_{i,j}X_j\rangle + \langle U_j, H_{j,i}X_i\rangle\big)H_{i,j}X_j + \\
&+ \frac{1}{p(X_i^\top H_{i,j}X_j)}\Big(\frac{q\kappa_1^2}{c_d(\kappa_1)}e^{\kappa_1\operatorname{Tr}(X_i^\top H_{i,j}X_j)} + \frac{q\kappa_2^2}{c_d(\kappa_2)}e^{\kappa_2\operatorname{Tr}(X_i^\top H_{i,j}X_j)}\Big)\big(\langle U_i, H_{i,j}X_j\rangle + \langle U_j, H_{j,i}X_i\rangle\big)H_{i,j}X_j + \\
&+ \frac{\gamma(X_h^\top H_{h,l}X_l)}{p(X_h^\top H_{h,l}X_l)}H_{i,j}U_j\Bigg) = \\
&= \sum_{(i,j)\in E}\Bigg(\Big(\frac{\gamma_p(X_i^\top H_{i,j}X_j)}{p(X_i^\top H_{i,j}X_j)} - g^2(X_i^\top H_{i,j}X_j)\Big)(\operatorname{Tr}(U_i^\top H_{i,j}X_j) + \operatorname{Tr}(U_j^\top H_{j,i}X_i))H_{i,j}X_j + \\
&+ g(X_i^\top H_{i,j}X_j)H_{i,j}U_j\Bigg)
\end{aligned}
\tag{9}
$$

where

$$\gamma_p(X_i^\top H_{i,j}X_j) = \frac{q\kappa_1^2}{c_d(\kappa_1)}e^{\kappa_1\operatorname{Tr}(X_i^\top H_{i,j}X_j)} + \frac{(1-q)\kappa_2^2}{c_d(\kappa_2)}e^{\kappa_2\operatorname{Tr}(X_i^\top H_{i,j}X_j)}$$

For a matter of simplicity we do not report the expression (9) in (8). The i$^{\text{th}}$ component of the Riemmanian Hessian is:

$$
\begin{aligned}
(\operatorname{Hess} f(X)[U])_i &= \operatorname{Proj}_{X_i}(D\overline{\operatorname{grad} f}(X)[U])_i = \\
&= X_i\operatorname{skew}\big(X_i^\top(D\overline{\operatorname{grad} f}(X)[U])_i\big)
\end{aligned}
$$

What we obtain is the following element:

$$\operatorname{Hess} f(X)[U] = \begin{pmatrix} (\operatorname{Hess} f(X)[U])_1 & (\operatorname{Hess} f(X)[U])_2 & \dots & (\operatorname{Hess} f(X)[U])_m \end{pmatrix}$$

As a particular instance, let us write the Riemmanian Hessian in the case of $p = p_\kappa$: to do so is sufficient to replace

- $g(X_i^\top H_{i,j}X_j)$ with $g^L(X_i^\top H_{i,j}X_j)$ in equation (8)
- $\gamma(X_i^\top H_{i,j}X_j)$ with $\gamma^L(X_i^\top H_{i,j}X_j) = \frac{\kappa}{c_d(\kappa)}e^{\kappa\operatorname{Tr}(X_i^\top H_{i,j}X_j)}$ in equation (9)
- $\gamma_p(X_i^\top H_{i,j}X_j)$ with $\gamma_p^L(X_i^\top H_{i,j}X_j) = \frac{\kappa^2}{c_d(\kappa)}e^{\kappa\operatorname{Tr}(X_i^\top H_{i,j}X_j)}$ in equation (9).

14) The gradient of $f(X)$ involves, for each $(h,j) \in E$, computing the matrix product $X_h^\top H_{h,j}X_j$, applying the skew-symmetrisation, and finally multiplying by X_h. Each term requires approximately $3d^3 + 3d^2$ flops, resulting in a total cost of $\mathcal{O}(|E|d^3)$ flops to compute the full gradient.

15) The following function to evaluate the likelihood will be used in the evaluations of the cost, gradient and hessian.

```
function prob = p(Z,k1,k2,q,constant1,constant2)
    d = size(Z,1);
    prob = q./constant1 .* exp(k1 .* (multitrace(Z)-d)) + ...
        (1-q)./constant2 .* exp(k2 .* (multitrace(Z)-d));
end
```

The cost function was evaluated as follows

```
function y = f(Xvar,problem)

    E = [problem.I, problem.J];
    H = problem.H;
    k1 = problem.kappa1;
    k2 = problem.kappa2;
    q = problem.p;
    constant1 = problem.c1;
    constant2 = problem.c2;


    H_proj = multiprod(H, Xvar(:,:,E(:,2)));
    Y = multiprod(multitransp(Xvar(:,:,E(:,1))), H_proj);
    prob_values = p(Y, k1, k2, q, constant1, constant2);
    y = -sum(sum(log(prob_values)));

end
```

The Riemannian gradient was obtained by projecting, through the `M.proj` method implemented at point 9, the Euclidean gradient computed as

```
function G = egradf(X,problem)

    I = problem.A;
    E = [problem.I, problem.J];
    H = problem.H;
    q = problem.p;
    k1 = problem.kappa1;
    k2 = problem.kappa2;
    c1 = problem.c1;
    c2 = problem.c2;

    G = zeros(size(X));
    d = size(X,1);
    for ii = 1:size(E,1)
        if(all(I~=E(ii,1)))
            h = E(ii,1);
            j = E(ii,2);
            M = X(:,:,h)'*H(:,:,ii)*X(:,:,j);
            ghj = 1./p(M,k1(ii),k2(ii),q(ii),c1(ii),c2(ii)) .* ...
                (q(ii).*k1(ii)./c1(ii) .*
                   exp(k1(ii).*(trace(M)-d))+ ...
                (1-q(ii)).*k2(ii)./c2(ii) .*
                   exp(k2(ii).*(trace(M)-d)));

            G(:,:,h) = G(:,:,h) - double(ghj) *
               eye(size(G(:,:,h))) * H(:,:,ii)*X(:,:,j);
        end
        if(all(I~=E(ii,2)))
```

```
            h = E(ii,2);
            j = E(ii,1);
            M = X(:,:,h)'*H(:,:,ii)'*X(:,:,j);
            ghj = 1./p(M,k1(ii),k2(ii),q(ii),c1(ii),c2(ii)) .* ...
                (q(ii).*k1(ii)./c1(ii) .*
                    exp(k1(ii).*(trace(M)-d))+ ...
                (1-q(ii)).*k2(ii)./c2(ii) .*
                    exp(k2(ii).*(trace(M)-d)));

            G(:,:,h) = G(:,:,h) - double(ghj) *
                eye(size(G(:,:,h))) * H(:,:,ii)'*X(:,:,j);
        end
    end
end
```

The gradient check tool reported the plot in figure 1 and the following message:

```
# Gradient check
The slope should be 2. It appears to be: 2.00005.
If it is far from 2, then the gradient might be erroneous.
The gradient at x must be a tangent vector at x.
If so, the following number is zero up to machine precision:
   2.62269e-14.
If it is far from 0, the gradient is not tangent.
```

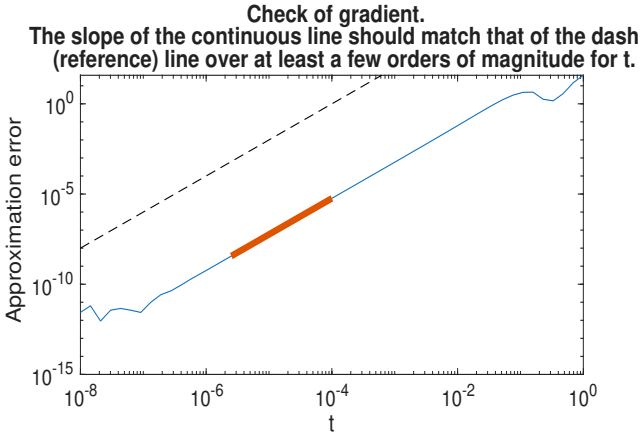

Figure 1: Plot of the gradient check.

As for the Riemannian Hessian, this was again computed through a transformation of the Euclidean Hessian by means of the `M.ehess2rhess` method implemented at point 9. The Euclidean Hessian was computed as

```
function Ehess = ehessf(problem,X,U)

    [d,~,m] = size(X);
    H = problem.H;
    A = problem.A;
    I = problem.I;
    J = problem.J;
    E = [problem.I problem.J];
    q = problem.p;
    k1 = problem.kappa1;
```

```
    k2 = problem.kappa2;
    c1 = problem.c1;
    c2 = problem.c2;
    Ehess = zeros(d,d,m);
    Xt = multitransp(X);
    gamma = q.*k1./c1 .* exp(k1.*
       (multitrace(multiprod(Xt(:,:,I),
       multiprod(H,X(:,:,J))))-d)) + ...
        (1-q).*k2./c2 .* exp(k2.*
           (multitrace(multiprod(Xt(:,:,I),
           multiprod(H,X(:,:,J))))-d));
    gammaprime = q.*k1.^2 ./c1 .* exp(k1.*
       (multitrace(multiprod(Xt(:,:,I),
       multiprod(H,X(:,:,J))))-d)) + ...
        (1-q).*k2.^2 ./c2 .* exp(k2.*
           (multitrace(multiprod(Xt(:,:,I),
           multiprod(H,X(:,:,J))))-d));

    for ii = 1:size(E,1)
        i = E(ii,1);
        j = E(ii,2);
        pval = p(X(:,:,i)'*
           H(:,:,ii)*X(:,:,j),k1(ii),k2(ii),q(ii),c1(ii),c2(ii));
        Ehess(:,:,i) = Ehess(:,:,i) + (- gamma(ii)^2 / pval^2
           +...
            gammaprime(ii) / pval) * ...
            (trace(U(:,:,i)'*H(:,:,ii)*X(:,:,j)) +
               trace(X(:,:,i)'*H(:,:,ii)*U(:,:,j))) * H(:,:,ii) *
               X(:,:,j) + ...
            gamma(ii) / pval * H(:,:,ii) * U(:,:,j);

        Ehess(:,:,j) = Ehess(:,:,j) + (- gamma(ii)^2 / pval^2
           +...
            gammaprime(ii) / pval) * ...
            (trace(U(:,:,i)'*H(:,:,ii)*X(:,:,j)) +
               trace(X(:,:,i)'*H(:,:,ii)*U(:,:,j))) * H(:,:,ii)'
               * X(:,:,i) + ...
            gamma(ii) / pval * H(:,:,ii)' * U(:,:,i);
    end
    Ehess = - Ehess;
end
```

The Hessian check tool reported the plot in figure 2 and the following message:

```
# Hessian check
The slope should be 3. It appears to be: 2.99906.
If it is far from 3, then the gradient or the Hessian might be
   erroneous.
Hess f(x)[d] must be a tangent vector at x.
If so, the following number is zero up to machine precision:
   1.5266e-13.
If it is far from 0, the Hessian returns non-tangent vectors.
The Hessian at x must be linear on the tangent space at x.
If so, ||a*H[d1] + b*H[d2] - H[a*d1+b*d2]|| is zero up to machine
   precision.
        Value: 6.16195e-13 (norm of H[a*d1+b*d2]: 958.555)
If it is far from 0, then the Hessian is not linear.
```

```
The Hessian at x must be symmetric on the tangent space at x.
If so, <d1, H[d2]> - <H[d1], d2> is zero up to machine precision.
        Value: -509.235 - -509.235 = -1.7053e-12.
If it is far from 0, then the Hessian is not symmetric.
```

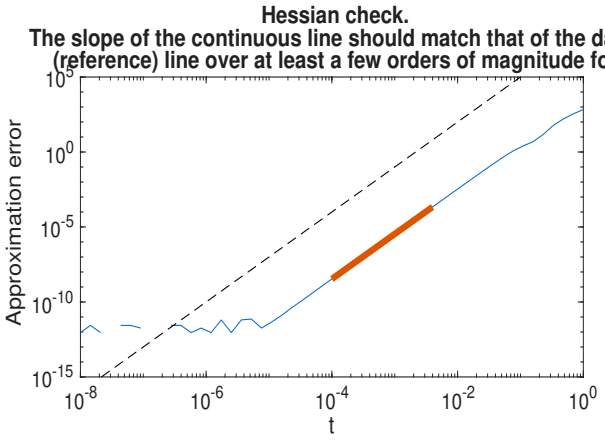

Figure 2: Plot of the hessian check.

4 Generating data and a warm-up experiment

5 A warm-up experiment and an initialization heuristic

16) To show the non-convexity of the problem we ran an experiment with parameters $d = 3, m = 10, \kappa_1 = 5, \kappa_2 = 0, q = 0.7$ and it turned out, as can be seen from the plots in Figure 3, that the method with random starting point gets stuck in a local minimum with higher value of the cost function with respect with the value of the actual minimum. The existence of this bad local minimum is an indicator of the non convexity of the problem.

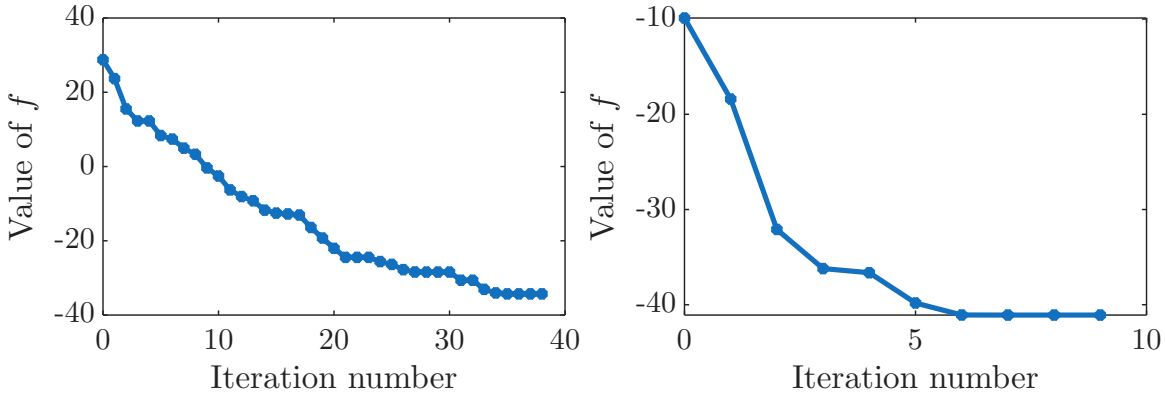

Figure 3: Value of the cost function with random initialisation (left) and spectral initialisation (right).

17) As a first step, let us rewrite the maximum likelihood estimator when $p = p_\kappa$, $\kappa > 0$ and

with no anchors:

$$\begin{aligned}\arg\max_{X\in\mathcal{M}} \sum_{(i,j)\in E} \log\left(p(X_i^\top H_{i,j} X_j)\right) &= \arg\max_{X\in\mathcal{M}} \sum_{(i,j)\in E} \log\left(\frac{1}{c_d(\kappa)} e^{\kappa \operatorname{Tr}(X_i^\top H_{i,j} X_j)}\right) = \\ &= \arg\max_{X\in\mathcal{M}} \sum_{(i,j)\in E} \log\left(\frac{1}{c_d(\kappa)}\right) + \kappa \operatorname{Tr}(X_i^\top H_{i,j} X_j) = \\ &= \arg\max_{X\in\mathcal{M}} \sum_{(i,j)\in E} \operatorname{Tr}(X_i^\top H_{i,j} X_j)\end{aligned}$$

The structures of Y and W_1 allow us to write:

$$\begin{aligned}\operatorname{Tr}(Y^\top W_1 Y) &= \sum_{i,j=1}^{d} \operatorname{Tr}(Y_i^\top (W_1)_{i,j} Y_j) = \sum_{(i,j)\in E} \operatorname{Tr}(Y_i^\top H_{i,j} Y_j) + \sum_{i=1}^{m} \operatorname{Tr}(Y_i^\top I_d Y_i) = \\ &= \sum_{(i,j)\in E} \operatorname{Tr}(Y_i^\top H_{i,j} Y_j) + dm\end{aligned}$$

where we exploited the following information

$$(W_1)_{i,j} = \begin{cases} H_{i,j} & \text{if } (i,j)\in E, \\ 0 & \text{otherwise,} \\ I_d & \text{if } i = j \end{cases}$$

Thus we deduce:

$$\arg\max_{Y\in\mathbb{R}^{dm\times d}, Y_j\in \mathrm{SO}(d)} \operatorname{Tr}(Y^\top W_1 Y) = \arg\max_{Y\in\mathbb{R}^{dm\times d}, Y_j\in \mathrm{SO}(d)} \sum_{(i,j)\in E} \operatorname{Tr}(Y_i^\top H_{i,j} Y_j)$$

To conclude, notice that, since there are no anchors, the manifold $\mathcal{M}$ coincides with the set where we search the maximum of $\operatorname{Tr}(Y^\top W_1 Y)$. This remark finishes the proof of the equivalence.

18) The request of this point is a straightforward computation:

$$Y^\top D_1 Y = \sum_{j=1}^{m} \deg(j) Y_j^\top Y_j = \sum_{j=1}^{m} \deg(j) I_d = \left(\sum_{j=1}^{m} \deg(j)\right) I_d = \operatorname{Tr}(D) I_d$$

19) In the following, the notation $D_1^{1/2}$ is used to address the diagonal matrix whose elements are the square roots of the elements of D_1. The notation is reasonable since $D_1^{1/2} D_1^{1/2} = D_1$. The assumption that the graph $\mathcal{G}$ is connected ensures that the matrix $D^{1/2}$ is invertible. Let us suppose that the d columns of Y form a dominant set of generalized eigenvectors of the GEP with pencil (W_1, D_1) and that Y satisfies $Y^\top D_1 Y = \operatorname{Tr}(D) I_d$. In other words, if $Y = \begin{pmatrix} y_1 & y_2 & \dots & y_d \end{pmatrix}$ the two following conditions hold

a) $W_1 y_i = \lambda_i D_1 y_i$ for all $i \in \{1, \dots, d\}$, where $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_{dm}|$

b) Y satisfies $Y^\top D_1 Y = \operatorname{Tr}(D) I_d$

Let $Z = D_1^{1/2} Y \in \mathbb{R}^{dm \times d}$, i.e. $Z = \begin{pmatrix} z_1 & z_2 & \dots & z_d \end{pmatrix}$ where $z_i = D_1^{1/2} y_i$. Then the condition (a) is rewritten in term of Z as follows

$$W_1 D_1^{-1/2} z_i = \lambda_i D_1 D_1^{-1/2} z_i \iff D_1^{-1/2} W_1 D_1^{-1/2} z_i = \lambda_i z_i$$

Therefore the columns of Z are the top d eigenvectors of the matrix $M = D_1^{-1/2} W_1 D_1^{-1/2} \in \mathbb{R}^{dm \times dm}$. The condition (b) is rewritten as

$$Z^\top D_1^{-1/2} D_1 D_1^{-1/2} Z = \mathrm{Tr}(D) I_d \iff Z^\top Z = \mathrm{Tr}(D) I_d$$

This shows that Z belongs to a scaled Stiefel manifold $\mathrm{St}(dm, d)$. By exercises of week 1, we know that Z is the global maximizer of the problem:

$$\max_{Z \in \mathbb{R}^{dm \times d}} \mathrm{Tr}(Z^\top M Z) = \sum_{i=0}^{d} \lambda_i(M) \quad \text{subject to} \quad Z^\top Z = \mathrm{Tr}(D) I_d$$

To conclude, is enough to note that the following equivalence holds:

$$Z \quad \text{is a global maximizer of} \quad \max_{Z \in \mathbb{R}^{dm \times d}} \mathrm{Tr}(Z^\top M Z) \quad \text{subject to} \quad Z^\top Z = \mathrm{Tr}(D) I_d \tag{10}$$

$$\Updownarrow$$

$$Y \quad \text{is a global maximizer of} \quad \max_{Y \in \mathbb{R}^{dm \times d}} \mathrm{Tr}(Y^\top W_1 Y) \quad \text{subject to} \quad Y^\top D_1 Y = \mathrm{Tr}(D) I_d \tag{11}$$

20) We can compute the top d eigenvectors of $M = D_1^{-1/2} W_1 D_1^{-1/2}$ using Matlab's `eigs` or Python's `scipy.sparse.linalg.eigsh`, and then recover Y as $Y = D_1^{-1/2} Z$, where Z is the matrix of eigenvectors.

21) Suppose that we have a complete measurement graph and there is no noise.

(a) Let Y be the $dm \times d$ matrix whose columns form a set of generalized eigenvectors of the GEP with pencil (W_1, D_1) and that $Y^\top D_1 Y = \mathrm{Tr}(D) I_d$. Alternatively, one can ask that Y satisfies both the equations

$$W_1 Y = D_1 Y \Lambda \quad \text{and} \quad Y^\top D_1 Y = \mathrm{Tr}(D) I_d \tag{12}$$

where $\Lambda \in \mathbb{R}^{d \times d}$ is the diagonal matrix whose elements are the top d generalized eigenvalues of the GEP problem with pencil (W_1, D_1).
We first verify that the matrix $\mathcal{R}$ satisfies the equation $W_1 \mathcal{R} = D_1 \mathcal{R}$, which has the same form as the first equation of (12) with $\Lambda = I_d$. Indeed, for each block-row i, we have

$$(W_1 \mathcal{R})_i = \sum_{k=1}^{m} (W_1)_{ik} R_k = \sum_{k=1}^{m} R_i R_k^\top R_k = \sum_{k=1}^{m} R_i = m R_i$$

Moreover, from $R_i \in \mathrm{SO}(d)$ for $i \in \{1, \dots, d\}$ follows that

$$\mathcal{R}^\top D_1 \mathcal{R} = \sum_{i=1}^{m} m R_i^\top R_i = m \sum_{i=1}^{m} I_d = m^2 I_d = \mathrm{Tr}(D) I_d$$

Thus, $\mathcal{R}$ satisfies both equations in (12) with $\Lambda = I_d$ and its d columns are generalized eigenvectors of the GEP with pencil (W_1, D_1) whose generalized eigenvalue is 1.
As a next step we want to study the generalized eigenvalues of the GEP with pencil (W_1, D_1): as seen in the previous point, they are the eigenvalues of the matrix $M = D_1^{-1/2} W_1 D_1^{-1/2}$. The matrix M is symmetric, therefore its eigenvalues are real. In addition $\mathrm{rk}(M) = \mathrm{rk}(W_1)$ since $D_1^{-1/2}$ is invertible. Using $\mathrm{rk}(W_1) = \mathrm{rk}(\mathcal{R}\mathcal{R}^\top) = d$ we know that the matrix M has only d non-zero eigenvalues. This argument implies that the columns of Y and $\mathcal{R}$ must be the same up to a permutation, therefore exists a permutation matrix $Q \in \mathbb{R}^{d\times d}$ such that $Y = \mathcal{R}Q$. The last step is to prove that $Q \in \mathrm{O}(d)$:

$$\begin{aligned} Y^\top D_1 Y &= Q^\top \mathcal{R}^\top D_1 \mathcal{R} Q = \mathrm{Tr}(D) Q^\top Q \\ &= \mathrm{Tr}(D) I_d \end{aligned}$$

which implies $Q^\top Q = I_d$, i.e., $Q \in \mathrm{O}(d)$.

Conversely, suppose $Y = \mathcal{R}Q$ with $Q \in \mathrm{O}(d)$. In point (a) we have already proved that $\mathcal{R}$ satisfies the GEP with $\Lambda = I_d$ and that the generalized eigenvalues of the pencil (W_1, D_1) are all zeros except for d which are ones. These observations imply the following equalities:

$$W_1 Y = W_1 \mathcal{R} Q = D_1 \mathcal{R} Q = D_1 Y$$

This demonstrates that Y satisfies the eigenvalue equation with $\Lambda = I_d$. Moreover,

$$\begin{aligned} Y^\top D_1 Y &= Q^\top \mathcal{R}^\top D_1 \mathcal{R} Q = \mathrm{Tr}(D) Q^\top Q \\ &= \mathrm{Tr}(D) I_d \end{aligned}$$

Thus the normalization condition is also satisfied. In conclusion, Y is a solution of the GEP corresponding to the dominant eigenvalues.

(b) From point (a) we know that the blocks Y_i of Y are orthogonal, i.e. $Y_i^\top Y_i = I_d$ for all $i \in \{1, \dots, m\}$. Using the previous result, the dominant solution of GEP can be written as $Y = \mathcal{R}Q$ with $Q \in \mathrm{O}(d)$. Every block Y_i satisfies

$$\det(Y_i) = \det(R_i)\det(Q) = \det(Q) \quad \text{for all} \quad i \in \{1, \dots, m\}$$

If $\det(Q) = 1$ then $\det(Y_i) = 1$ for all $i \in \{1, \dots, m\}$. Therefore $Y_i = Y_i^{(a)} \in \mathrm{SO}(d)$. Otherwise, if $\det(Q) = -1$, then $\det(Y_i) = \det(Y_i)^{(a)} = -1$ and $\det(Y_i^{(b)}) = \det(Y_i^{(a)})\det(\mathcal{J}) = -\det(Y_i^{(a)}) = 1$. Thus, one between $Y^{(a)}$ and $Y^{(b)}$ is such that each of the m blocks is in $\mathrm{SO}(d)$ and still solves the dominant GEP. Therefore it is a valid solution of the problem addressed in point 17.

22) Let R_j, $\tilde{X}_j \in \mathrm{SO}(d)$ for $j \in J$ and $Q \in \mathrm{SO}(d)$ as in the text of the project. In the next computations, we use this simple remark: $\|A\|_F^2 = \mathrm{Tr}(A^\top A) = \mathrm{Tr}(I_d) = d$ for all $A \in \mathrm{SO}(d)$.

$$\begin{aligned} \sum_{j\in J} \|R_j - \tilde{X}_j Q\|_F^2 &= \sum_{j\in J} \left(\|R_j\|_F^2 + \|\tilde{X}_j Q\|_F^2 - 2\ \mathrm{Tr}(R_j^\top \tilde{X}_j Q) \right) \\ &= 2d|J| - 2\sum_{j\in J} \mathrm{Tr}(Q^\top \tilde{X}_j^\top R_j) = 2d|J| - 2\ \mathrm{Tr}\left(Q^\top \sum_{j\in J} \tilde{X}_j^\top R_j \right) \end{aligned}$$

In addition,

$$\left\| Q - \sum_{j\in J} \tilde{X}_j^\top R_j \right\|_F^2 = \operatorname{Tr}\left((Q - \sum_{j\in J} \tilde{X}_j^\top R_j)^\top (Q - \sum_{i\in J} \tilde{X}_i^\top R_i) \right) =$$
$$= \operatorname{Tr}(Q^\top Q) - 2\operatorname{Tr}\left(Q^\top \sum_{j\in J} \tilde{X}_j^\top R_j \right) + \operatorname{Tr}\left(\sum_{j\in J}\sum_{i\in J} R_j^\top \tilde{X}_j \tilde{X}_i^\top R_i \right)$$

Noticing that the first and last terms of the previous equality are constants with respect to Q, we obtain

$$\arg\min_{Q\in \mathrm{SO}(d)} \sum_{j\in J} \|R_j - \tilde{X}_j Q\|_F^2 = \arg\max_{Q\in \mathrm{SO}(d)} \operatorname{Tr}\left(Q^\top \sum_{j\in J} \tilde{X}_j^\top R_j \right) = \arg\min_{Q\in \mathrm{SO}(d)} \left\| Q - \sum_{j\in J} \tilde{X}_j^\top R_j \right\|_F^2$$

and the last term is, by definition, the projection of $\sum_{j\in J} \tilde{X}_j^\top R_j$ onto $\mathrm{SO}(d)$.

23) The spectral initial guess was implemented as follows:

```
function X0 = initialGuess(problem)

    E = [problem.I problem.J];
    d = problem.n;
    m = problem.N;
    H = problem.H;
    A = problem.A;
    notA = setdiff(1:m,A);
    Ra = problem.Ra;

    num_nodes = max(E(:));

    degrees = accumarray(E(:), 1, [num_nodes, 1]);

    D = spdiags(degrees,0,numel(degrees),numel(degrees));

    D1 = kron(D,speye(d));


    W1 = speye(m*d);

    for ii = 1:size(E,1)

        i = E(ii,1);
        j = E(ii,2);

        W1((i-1)*d+1:i*d,(j-1)*d+1:j*d) = H(:,:,ii);
        W1((j-1)*d+1:j*d,(i-1)*d+1:i*d) = H(:,:,ii)';

    end
    invsqD1 = sqrt(D1)\speye(m*d);
    M = invsqD1 * W1 * invsqD1;
    [Z,~] = eigs(M,d);
    Y = invsqD1 * Z;

```

```
    Xa = zeros(d,d,m);
    Xb = zeros(d,d,m);
    for i = 0:m-1
        [U,~,V] = svd(Y(i*d+1:(i+1)*d,:));
        s = sign(det(U*V'));
        Xa(:,:,i+1) = U * diag([ones(d-1,1);s]) * V';

        Yj = Y*diag([ones(d-1,1);-1]);
        [U,~,V] = svd(Yj(i*d+1:(i+1)*d,:));
        s = sign(det(U*V'));
        Xb(:,:,i+1) = U * diag([ones(d-1,1);s]) * V';
    end

    if f(Xa, problem) >= f(Xb, problem)
        Xtilde = Xa;
    else
        Xtilde = Xb;
    end

    XtildeA = sum(multiprod(multitransp(Xtilde(:,:,A)),Ra),3);
    [U,~,V] = svd(XtildeA);
    s = sign(det(U*V'));
    Q = U * diag([ones(d-1,1);s]) * V';

    X0 = zeros(d,d,m);
    X0(:,:,A) = Ra;
    X0(:,:,notA) = multiprod(Xtilde(:,:,notA),Q);
end
```

6 Algorithms and experiments

24) We decided to set the maximum trust region radius at $\overline{\Delta} = \sqrt{\dim\mathcal{M}} = \sqrt{(m-k)\frac{d(d-1)}{2}}$, since it is a conventional choice. Therefore, the initial trust region radius is set at $\Delta_0 = \overline{\Delta}/8 = \sqrt{(m-k)\frac{d(d-1)}{2}}/8$.

25) The implementation of the MSE function is reported below

```
function mse = MSE(X,Xhat,problem)

    m = problem.N;
    A = problem.A;
    notA = setdiff(1:m,A);
    j = length(A);

    L = 0;
    for i = notA
        P = logm(X(:,:,i)'*Xhat(:,:,i));
        L = L + trace(P'*P);
    end
    mse = L / (m-j);

end
```

26) We ran the first experiment with a random starting point, both with RGD and RTR. Figures 4 and 5 show the behaviour of the two optimisation methods.

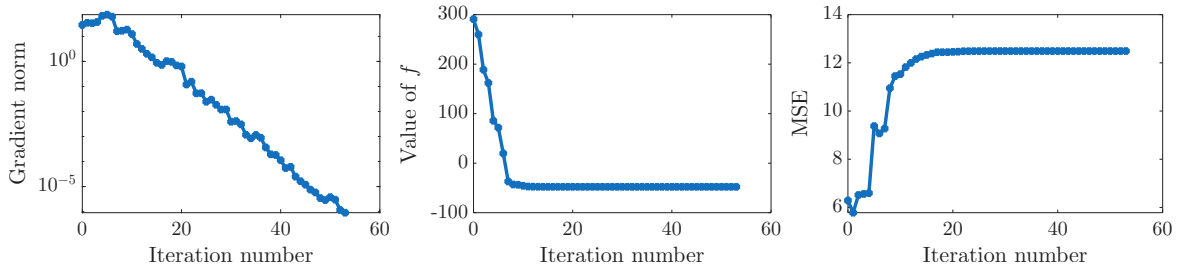

Figure 4: Gradient norm (left), cost value (center) and MSE (right) as a function of iterates for RGD with random starting point for the first experiment.

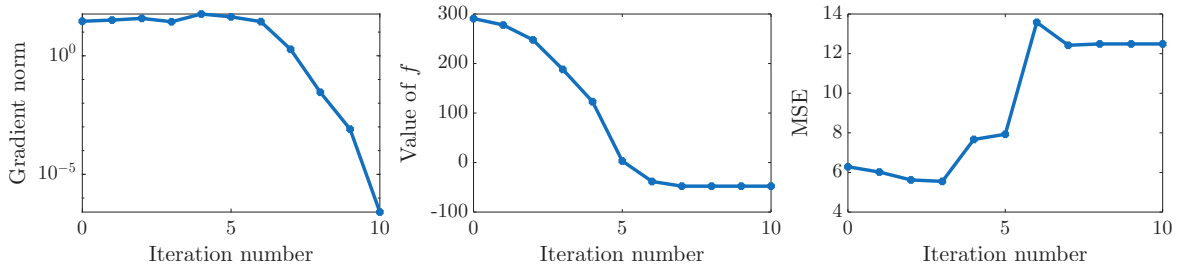

Figure 5: Gradient norm (left), cost value (center) and MSE (right) as a function of iterates for RTR with random starting point for the first experiment.

It is clear that the trust regions method is quite faster in terms of iterations needed to reach convergence. The behaviour of the two methods is different. Riemmanian gradient descent has a steep drop at the beginning, but approaching the minimum it struggles to actually find it. On the other hand, the trust regions method stalls in the beginning, but when it gets closer to the minimum the convergence is very fast. We notice though how the MSE starts from a lower point and ends up being higher than at the beginning. This may be due to the fact that the starting point was too far from the ground truth and the methods get stuck in some local minimum that best explains the noise but which is more dissimilar to the ground truth than the starting point.

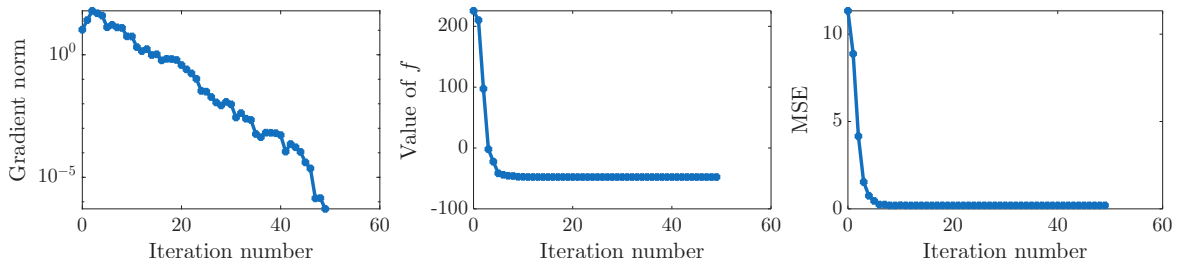

Figure 6: Gradient norm (left), cost value (center) and MSE (right) as a function of iterates for RGD with spectral starting point for the first experiment.

With spectral initialisation (see Figures 6 and 7), both methods converge slightly faster with a trend similar to the random initialisation case. In this case we can notice how MSE and cost share the same behaviour: this is likely due to the fact that the initial guess is close enough

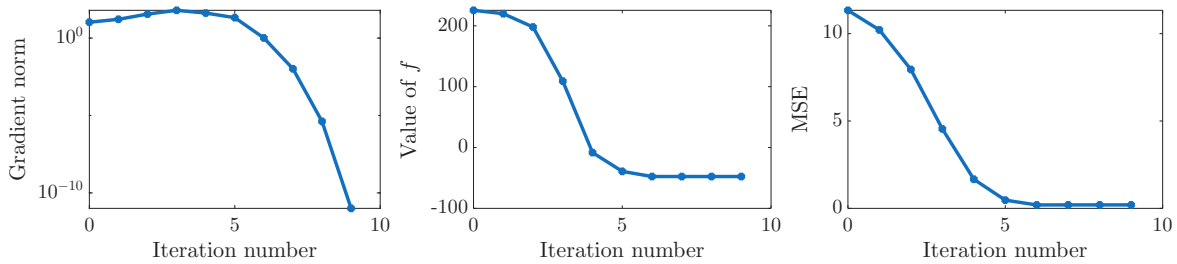

Figure 7: Gradient norm (left), cost value (center) and MSE (right) as a function of iterates for RTR with spectral starting point for the first experiment.

to the ground truth so that the closest minimizer of the noisy likelihood is the ground truth itself.

27) We repeated the same experiments for a different choice of parameters, on a larger manifold and with some outliers in the distribution.

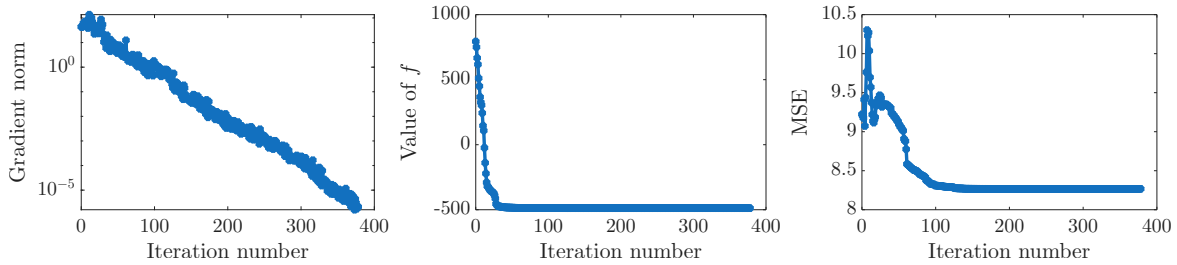

Figure 8: Gradient norm (left), cost value (center) and MSE (right) as a function of iterates for RGD with random starting point for the second experiment.

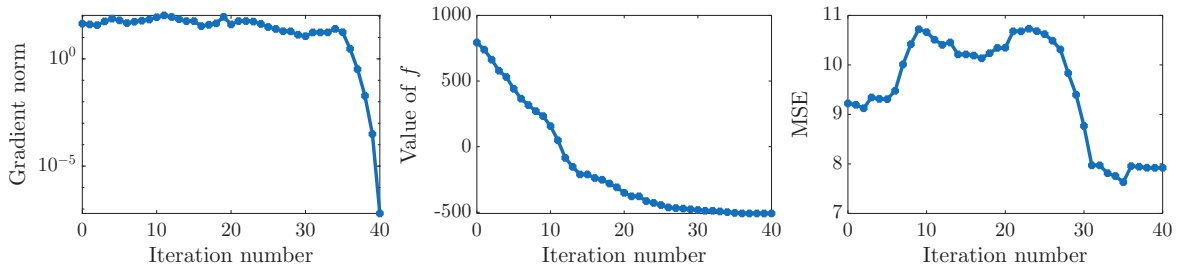

Figure 9: Gradient norm (left), cost value (center) and MSE (right) as a function of iterates for RTR with random starting point for the second experiment.

Figures 8 and 10 highlight a significant difference between the two methods: the RGD method took more than 300 iterations to reach the minimum, while RTR needed only 40 iterations with random starting point.
The difference is even bigger in the case of spectral initialisation, shown in Figures 10 and 11. Here the RGD had even worse performances in terms of time, though reaching a better MSE. The trust regions method converged faster, with just 25 iterations. Both methods reach a worse final MSE, denoting that in the framework of high dimension with a complete graph (which implies many noisy measures, likely not coherent among themselves) there are configurations different from the ground truth that better fit the noisy data.The methods are led towards these points by the spectral initialisation.

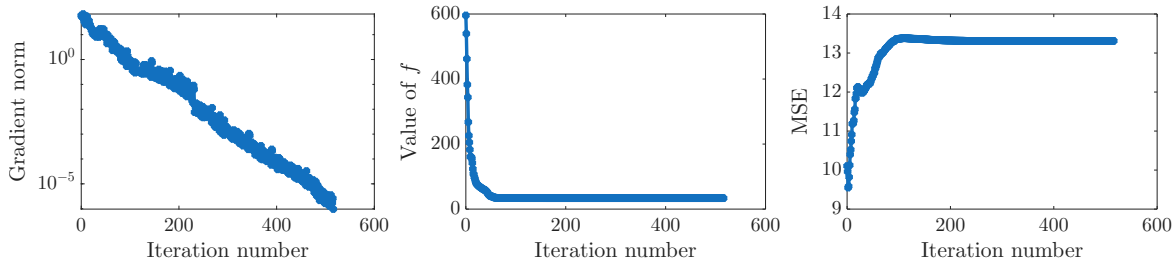

Figure 10: Gradient norm (left), cost value (center) and MSE (right) as a function of iterates for RGD with spectral starting point for the second experiment.

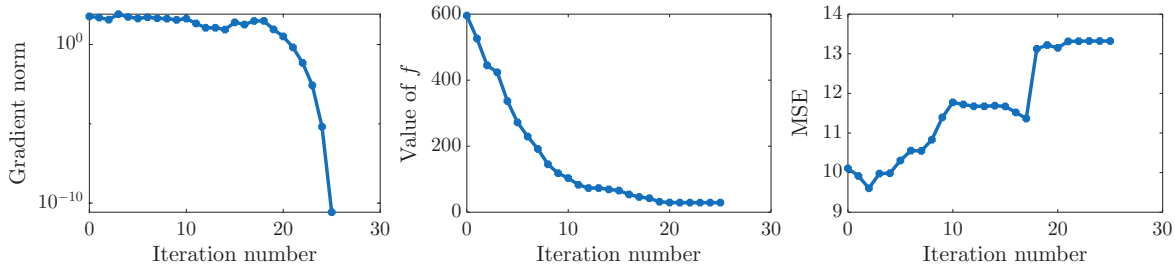

Figure 11: Gradient norm (left), cost value (center) and MSE (right) as a function of iterates for RTR with spectral starting point for the second experiment.

28) In this third experiment, the parameters remained the same, but we reduced the density of connections in the graph from 1 to 0.75.

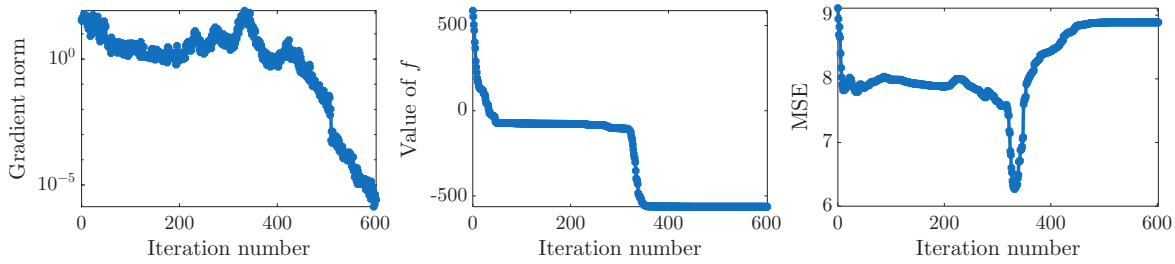

Figure 12: Gradient norm (left), cost value (center) and MSE (right) as a function of iterates for RGD with random starting point for the third experiment.

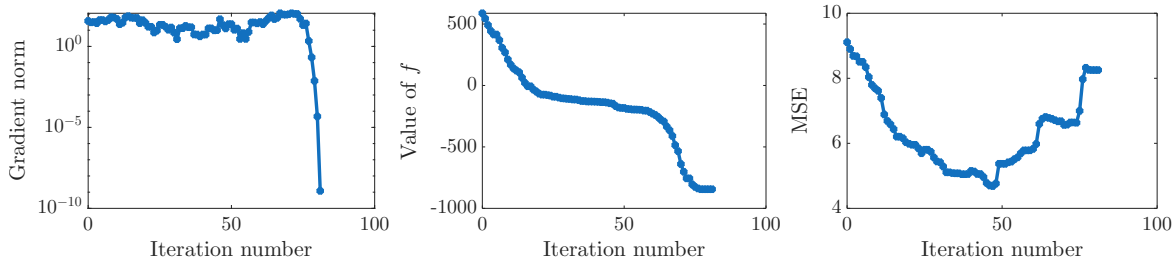

Figure 13: Gradient norm (left), cost value (center) and MSE (right) as a function of iterates for RTR with random starting point for the third experiment.

As could be expected, again the RTR method outperformed the RGD by a great amount of iterations. In this case, spectral initialisation made a great difference, leading the two methods (the RTR in particular) to converge to a much better estimator than in the random case.

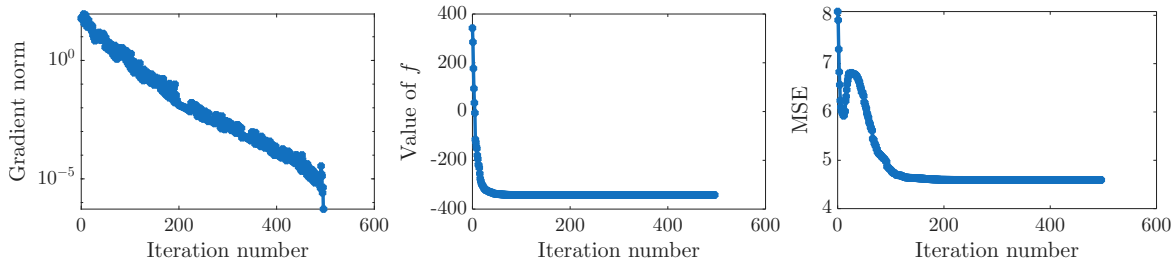

Figure 14: Gradient norm (left), cost value (center) and MSE (right) as a function of iterates for RGD with spectral starting point for the third experiment.

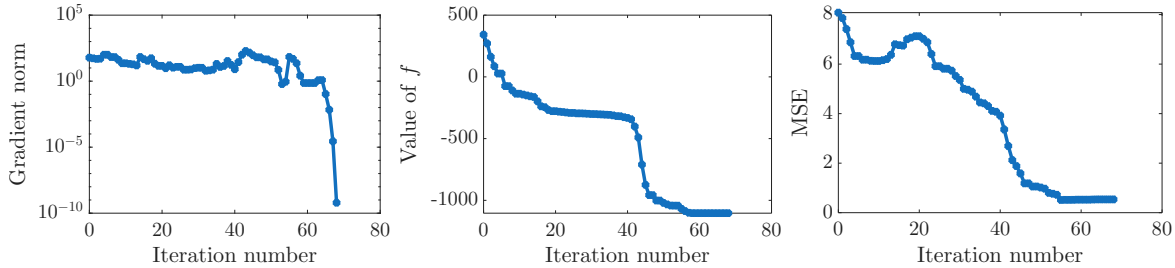

Figure 15: Gradient norm (left), cost value (center) and MSE (right) as a function of iterates for RTR with spectral starting point for the third experiment.

29) We tested the method on some configurations and it turns out that for $\kappa_1 = \kappa_2 = 5$ the trust region method works much better than a random estimator, with a MSE of around 0, as can be seen from Figure 16, and similar results are obtained provided that both κ_1 and κ_2 are not 0. On the other hand, when one of the two κ_i is 0 the method has a comparable performance to that of a random estimator when q favors the zero part of the mixture. Again in Figure 16 we notice how, for $\kappa_1 = 5$ and $\kappa_2 = 0$, the values of the MSE are very high for q approaching 0, and decrease for values closer to 1.

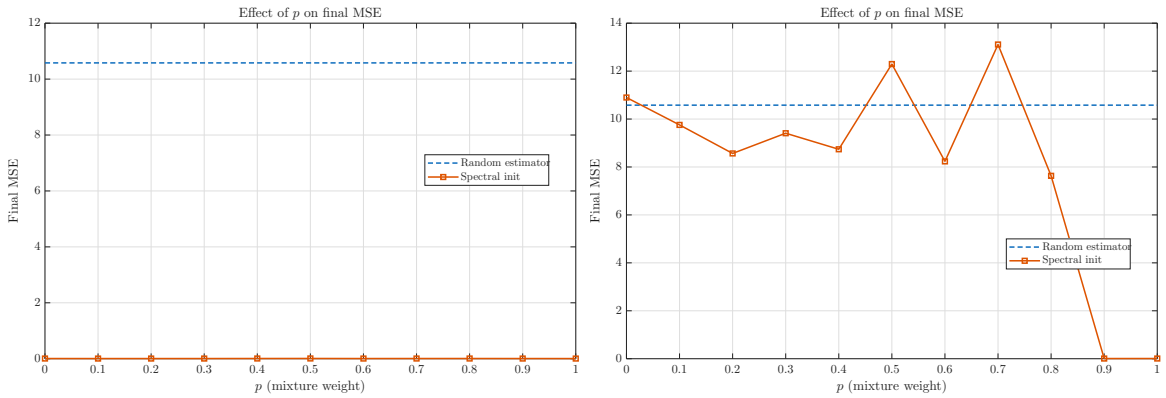

Figure 16: MSE of RTR with spectral initialisation compared to the random estimator for $\kappa_1 = \kappa_2 = 5$ (left) and $\kappa_1 = 5, \kappa_2 = 0$ (right).

As far as the initialisation is concerned, running several experiments we found out that the situations where the spectral initialisation has strong impact are those where the problem is not too asymmetrical between the components of the mixtures and where the manifold has bigger dimension. On the other hand, as seen also in the previous plot, the spectral initialisation has no relevant impact on MSE when one of the components has $\kappa = 0$ and such component is favoured by the mixing probability.

κ_1	κ_2	q	N	Iter. (rand)	MSE (rand)	Iter. (spec)	MSE (spec)
1	0	0.2	20	34	9.1588	26	9.5583
1	0	0.2	40	58	9.8962	37	8.7611
1	0	0.5	20	34	6.3428	27	1.5061
1	0	0.5	40	42	4.3199	30	1.1487
1	0	0.7	20	28	14.075	15	0.8008
1	0	0.7	40	19	15.839	17	0.4786
1	5	0.2	20	19	18.226	15	0.0466
1	5	0.2	40	24	6.0752	17	0.0216
1	5	0.5	20	19	1.3731	18	0.0627
1	5	0.7	40	16	7.4823	20	0.0693
5	0	0.2	20	50	8.9273	73	5.6821
5	0	0.2	40	99	11.247	41	11.738
5	0	0.5	20	42	11.936	29	14.440
5	0	0.5	40	64	8.4474	57	10.948
5	0	0.7	20	39	9.9317	27	2.9596
5	0	0.7	40	36	11.756	26	12.170
5	5	0.2	20	13	13.927	13	0.0256
5	5	0.2	40	18	15.512	14	0.0152
5	5	0.5	20	13	14.887	10	0.0385
5	5	0.7	40	16	16.024	12	0.0144

Table 1: Comparison between random and spectral initialisation across several parameters.

Appendix A

How to run the code

Opening the folder, you will find the following functions:

- `egradf.m`: computes the euclidean gradient of f
- `ehessf.m`: computes the euclidean Hessian - vector product of f
- `f.m`: computes the value of the cost function f
- `initialGuess.m`: computes the initial guess based on GEP
- `MSE.m`: computes the MSE w.r.t. the ground truth
- `p.m`: computes the likelihood
- `saveMSEstats.m`: utility function to save MSE values for each iteration

The runnable codes are:

- `nonConvex.m`: code to assess the non convexity of the problem (point 16)
- `Experiment1.m`: renders the plot for the first configuration (point 26)

- `Experiment2.m`: renders the plot for the second configuration (point 27)
- `Experiment3.m`: renders the plot for the third configuration (point 28)
- `finalExperiment1.m`: renders the plot to assess the impact of q (point 29)
- `finalExperiment2.m`: computes the value of the configurations in Table 1 (point 29)