MATH-512 - Project 1

Luca Bulgarelli Tommaso Fioratti Greta Pizzichini

March 2025

1 Repelling particles on the sphere

The aim of this project is to arrange n particles on the sphere $S^{d-1} \subset \mathbb{R}^d$ in such a way that they are as far as possible from each other. In order to do so, three main objects are introduced:

- the oblique manifold $\mathrm{OB}(d,n) = S^{d-1} \times \cdots \times S^{d-1}$
- the diagonal set $\Delta = \{X \in \mathrm{OB}(d,n) \mid x_i = x_j \text{ for some } i \neq j\}$
- the manifold $\mathcal{M} := \mathrm{OB}(d,n) \setminus \Delta$

The set $\mathcal{M}$ is considered as the domain of the real function

$$f(X) = -\frac{2}{n(n-1)} \sum_{i<j} \log(1 - x_i^\top x_j)$$

where $X = [x_1\ x_2 \dots x_{n-1}\ x_n]$ is a real matrix $d \times n$ whose columns are unit norm vectors.
In this project, the Riemannian Gradient Descent with backtracking line-search on f is implemented from scratch and and by using the Manopt toolboxes.

2 The Geometry of $\mathrm{OB}(d,n)$

1) Let $X = [x_1 \cdots x_n]$ be a point of the set $\mathcal{M} = \mathrm{OB}(d,n) \setminus \Delta$. In particular X is a point of $\mathrm{OB}(d,n)$, which is an embedded submanifold of $\mathbb{R}^{d\times n}$ because it is the product of n spheres embedded in $\mathbb{R}^d$ (result seen in lectures). Then $\exists\ U = [U_1 \cdots U_n]$ neighborhood of X in $\mathbb{R}^{d\times n}$ and $\exists\ h = [h_1 \cdots h_n] : U \to \mathbb{R}^n$ such that:

- U_i is a neighborhood of x_i in $\mathbb{R}^d$ and $h_i : U_i \to \mathbb{R}$ is a local defining function of S^{d-1} around x_i
- h is smooth
- $h^{-1}(0) = U \cap \mathrm{OB}(d,n)$
- $Dh(X)$ has rank n.

Since $X \notin \Delta$, the relation $x_i \neq x_j$ is true $\forall i,j \in \{1,...n\}$ with $i \neq j$, this allows us to define $\overline{U} = [\overline{U_1} \cdots \overline{U_n}]$ such that:

- $x_i \in \overline{U_i} \subseteq U_i\ \forall i \in \{1,...n\}$
- $\overline{U_i}$ is open in $\mathbb{R}^d$
- $\bigcap_{i=1}^n \overline{U_i} = \emptyset$

With these requirements $\overline{U}$ is a neighborhood of X in $\mathbb{R}^{d\times n}$ and $\mathrm{OB}(d,n) \cap \overline{U} \subset \mathcal{M}$. Let us consider the function $\overline{h} = h_{|\overline{U}} = [h_{1_{|\overline{U_1}}} \dots\ h_{n_{|\overline{U_n}}}]$. We note that

- $\overline{h} : \overline{U} \to \mathbb{R}^n$ is smooth: this is true because $\overline{h}$ is the restriction of a smooth function;
- $\overline{h}^{-1}(0) = \overline{U} \cap \mathcal{M}$: this is true by construction, in fact

$$\overline{h}^{-1}(0) = h^{-1}(0) \cap \overline{U} = U \cap \mathrm{OB}(d,n) \cap \overline{U} = \mathcal{M} \cap \overline{U}$$

- $D\overline{h}(X)$ has rank n: the map $D\overline{h}(X)$ is a diagonal matrix whose diagonal elements are $D\overline{h_i}(x_i) = Dh_{i_{|\overline{U_i}}}(x_i)$. The rank of $D\overline{h_i}(x_i)$ is 1 because h_i is a local defining function of S^{d-1} around x_i and $\overline{h_i}$ coincides with h_i in the restricted neighborhood so do their differentials.

Thus the map $\overline{h}$ is a local defining function for $\mathcal{M}$ around X. Thanks to arbitrariness of X, we conclude that $\mathcal{M}$ is a manifold.

2) In order to describe $T_X\mathrm{OB}(d,n)$ let us look closer to the local defining function h from the previous point. As seen during lectures, a local defining function for the embedded sphere S^{d-1} in $\mathbb{R}^d$ is

$$\begin{aligned} k :&\mathbb{R}^d \longrightarrow \mathbb{R} \\ &x \longmapsto x^\top x - 1 \end{aligned}$$

With this in mind, it is clear that we can express the local defining function (actually this function describes the whole manifold $\mathrm{OB}(d,n)$) h of $\mathrm{OB}(d,n)$ around X in the following way: let $Y = [y_1 \cdots y_n] \in U$

$$h(Y) = \begin{pmatrix} y_1^\top y_1 - 1 \\ \vdots \\ y_n^\top y_n - 1 \end{pmatrix} = \mathrm{diag}(Y^\top Y) - \mathbb{I}$$

where $\mathbb{I} = (1, \dots 1)^\top = \mathrm{ones(n,1)} \in \mathbb{R}^n$ and $\mathrm{diag}(X)$ returns a column vector containing the diagonal elements of X. To describe $T_X\mathrm{OB}(d,n)$ we exploit the local defining function mentioned above and the equality

$$T_X\mathrm{OB}(d,n) = \ker Dh(X)$$

In order to do so, let us compute $Dh(X)$ in $V = [v_1 \cdots v_n] \in \mathbb{R}^{d\times n}$:

$$\begin{aligned} Dh(X)[V] &= \lim_{t\to 0} \frac{\mathrm{diag}((X+tV)^\top(X+tV)) - \mathbb{I} - \mathrm{diag}(X^\top X) + \mathbb{I}}{t} = \\ &= \lim_{t\to 0} \frac{\mathrm{diag}(X^\top X + t(X^\top V + V^\top X) + t^2 V^\top V) - \mathrm{diag}(X^\top X)}{t} = \\ &= \lim_{t\to 0} \frac{t\cdot \mathrm{diag}(X^\top V + V^\top X) + t^2 \cdot \mathrm{diag}(V^\top V)}{t} = \\ &= \mathrm{diag}(X^\top V + V^\top X) = 2\mathrm{diag}(X^\top V) \end{aligned}$$

where the last equality comes from the fact that $X^\top V = (V^\top X)^\top$, i.e. they have the same diagonal elements. In conclusion, we obtain

$$T_X\mathrm{OB}(d,n) = \ker Dh(X) = \{V \in \mathbb{R}^{d\times n} \mid \mathrm{diag}(X^\top V) = \mathbb{O}\}$$

where $\mathbb{O} = (0, \dots 0)^\top = \mathrm{zeros(n,1)}$.
Now, let $Z \in \mathbb{R}^{d\times n}$. To compute the orthogonal projection onto the tangent space $T_X\mathrm{OB}(d,n)$ the idea is to subtract to each column z_i of Z the component of z_i along x_i:

$$z_i \longmapsto z_i - x_i(x_i^\top z_i)$$

Thus, for all columns, this becomes:

$$P_{T_X\mathrm{OB}(d,n)}(Z) = Z - X.^*\mathrm{diag}(X^\top Z)^\top$$

Indeed, we can verify that $P_{T_X\mathrm{OB}(d,n)}(Z) \in T_X\mathrm{OB}(d,n)$ since:

$$\begin{aligned} \mathrm{diag}(X^\top P_{T_X\mathrm{OB}(d,n)}(Z)) &= \mathrm{diag}(X^\top Z - X^\top X.^*\mathrm{diag}(X^\top Z)^\top) = \\ &= \mathrm{diag}(X^\top Z) - \mathrm{diag}(X^\top X.^*\mathrm{diag}(X^\top Z)^\top) = \\ &= \mathrm{diag}(X^\top Z) - \mathrm{diag}(X^\top X)\mathrm{diag}(X^\top Z) = \\ &= \mathrm{diag}(X^\top Z) - \mathrm{diag}(X^\top Z) = \mathbb{O} \end{aligned}$$

Therefore, $P_{T_X\mathrm{OB}(d,n)}(Z) \in T_X\mathrm{OB}(d,n)$.

3) Let $X = [x_1 \cdots x_n] \in \mathrm{OB}(d,n)$ and $U = [u_1 \cdots u_n]$, $V = [v_1 \cdots v_n]$ two points of $T_X\mathrm{OB}(d,n)$. We can interpret the tangent space of a product of manifolds as the product of the tangent spaces [1]:

$$T_X\mathrm{OB}(d,n) = T_{x_1}S^{d-1} \times \cdots \times T_{x_n}S^{d-1}$$

In addition, we can turn $\mathrm{OB}(d,n)$ into a Riemannian manifold by giving it the Riemannian product metric:

$$\langle U, V\rangle_X := \sum_{i=1}^{n}\langle u_i, v_i\rangle_{x_i} = \sum_{i=1}^{n}\langle u_i, v_i\rangle_{\mathbb{R}^d} = \sum_{i=1}^{n} u_i^\top v_i = \mathrm{trace}(U^\top V)$$

Here the inclusion $S^{d-1} \subset \mathbb{R}^d$ has been exploited to obtain a Riemannian metric for the manifold S^{d-1} as an embedded manifold [2] of the Riemannian manifold $\mathbb{R}^d$ with the metric $\langle u, v\rangle_x = \langle u, v\rangle_{\mathbb{R}^n} = u^\top v$.

4) Our propose for the retraction of the manifold $\mathrm{OB}(d,n)$ is the following:

$$\begin{aligned} R : T\mathrm{OB}(d,n) &\longrightarrow \mathrm{OB}(d,n) \\ (X,V) &\longmapsto R_X(V) = (X+V)./\mathrm{vecnorm}(X+V) \end{aligned}$$

where $T\mathrm{OB}(d,n)$ is the tangent bundle of $\mathrm{OB}(d,n)$ and $\mathrm{vecnorm}(X)$ returns the row vector containing the euclidean norm of each column of X. To prove that the map R is a retraction follow the next steps:

- R is smooth: to see clearly this point let us rewrite R in the following way:

$$R_X(V) = \begin{bmatrix} \frac{x_1+v_1}{\|x_1+v_1\|} & \cdots & \frac{x_n+v_n}{\|x_n+v_n\|} \end{bmatrix} \tag{1}$$

  From this expression, it is evident that R acts on each column of an element of $\mathbb{R}^{d\times n}$ as the retraction that we have encountered during lectures for the manifold S^{d-1} in $\mathbb{R}^d$. Thanks to the smoothness of the latter we have the smoothness of R.

- Let $c(t) = R_X(tV)$ be a curve onto $\mathrm{OB}(d,n)$. Than we have:

$$c(0) = R_X(0) = X./\mathrm{vecnorm}(X)) = X$$

  because $\mathrm{vecnorm}(X) = \mathbb{I}$. In addition, the equality $c'(0) = V$ holds thanks to what we have studied during lectures and expression (1).

5) Our implementation of the oblique manifold follows.

a) Sample a point X from $\mathrm{OB}(d,n)$ uniformly at random.

```
function X = sampleOB(d,n)
    X = randn(d,n);
    X = X./vecnorm(X,2,1);
end
```

b) Compute the orthogonal projection of Z into $T_X\mathrm{OB}(d,n)$.

```
function P = project(X,Z)
    P = Z - X * diag(diag(X' * Z));
end
```

c) Compute the value of the metric $\langle U, V\rangle_X$ for $U, V \in T_X\mathrm{OB}(d,n)$.

```
function r = riemannMetric(U,V)
    r = trace(U' * V);
end
```

[1] As seen in exercises of week 4
[2] As seen during the lecture of week 4

d) Sample a random tangent vector of unit norm.

```
function V = sampleTangent(X)
    [d,n] = size(X);
    V = randn(d,n);
    V = project(X,V);
    V = V/sqrt(riemannMetric(V,V));
end
```

e) Compute the retraction $R_X(V)$.

```
function R = retract(X,V)
    R = (X+V)./vecnorm(X+V,2,1);
end
```

6) To test our code of point b) we checked that the Riemannian metric between X and V is zero:

```
X = sampleOB(3,3);
V = randn(3,3);
Vproj = project(X,V);
```

```
<X,V> = tr(X'*V) = 0.000000
```

As far as point c) is concerned, we did not implement any specific function, since if the Riemannian metric is indeed correct the implementation does not leave much room for mistakes.

To test our code of point e) we checked that $R_X(V)$ belongs to $\mathrm{OB}(d,n)$, i.e. that $\mathrm{vecnorm}(R_X(V)) = [1 \dots 1]$:

```
X = sampleOB(3,3);
V = sampleTangent(X);
R = retract(X,V);
disp (vecnorm(R,2,1))
```

```
1.0000    1.0000    1.0000
```

3 Implementing RGD with a line-search

7) To evaluate f in $X = [x_1 \cdots x_n]$, we need to compute the dot product $x_i^\top x_j$ for each pair of vectors $x_i, x_j \in \mathbb{R}^d$ with $i, j \in \{1, \dots n\}$. Computing a dot product costs $\mathcal{O}(2d)$ operations, and computing the logarithm costs $\mathcal{O}(1)$. Since there are $\binom{n}{2} = \frac{n(n-1)}{2} = \mathcal{O}(\frac{n^2}{2})$ pairs, the total cost becomes $\mathcal{O}(n^2 d)$. Therefore, the overall complexity is $\mathcal{O}(n^2 d)$ (in particular the constant is exactly 1).

8) The aim of this subsection is to find an expression for the Riemannian gradient of f. Thanks to the choice of the metric made above, $\mathcal{M}$ is a Riemannian submanifold of $\mathbb{R}^{d\times n}$. This fact can be exploited to compute the gradient of f through a smooth extension Γ:

$$\mathrm{grad}_X f = \mathrm{Proj}_X(\mathrm{grad}_X \Gamma) \tag{2}$$

where $\Gamma : \mathbb{R}^{d\times n} \to \mathbb{R}$ is smooth and has the same expression of f applied to all the elements of $\mathbb{R}^{d\times n}$. From exercises of week 4, we know that

$$\mathrm{grad}_X \Gamma = (\mathrm{grad}_{x_1} \Gamma_1, \dots \mathrm{grad}_{x_n} \Gamma_n) \tag{3}$$

where $\Gamma_k : \mathbb{R}^d \to \mathbb{R}$ is the function obtained from Γ by fixing all the components except for the k^{th} one, i.e. $\Gamma_k(x) = \Gamma([x_1 \cdots x \cdots x_n])\ \forall k \in \{1, \dots n\}$. Thanks to (3), it is sufficient to compute $\mathrm{grad}_x \Gamma_k$ for $x \in \mathbb{R}^d$. Let us express Γ_k in the following way

$$\Gamma_k(x) = -\frac{2}{n(n-1)} \left[\sum_{i<j, i\neq k, j\neq k} \log(1 - x_i^\top x_j) + \sum_{j=1, j\neq k}^{n} \log(1 - x^\top x_j) \right]$$

Now we can compute $\text{grad}_x \Gamma_k : \mathbb{R}^d \to \mathbb{R}$. Considering that [3]

$$\text{grad}_x \Gamma_k = \left(\frac{\partial \Gamma_k}{\partial x^1}(x), \dots \frac{\partial \Gamma_k}{\partial x^n}(x) \right)$$

we are interested in computing $\frac{\partial \Gamma_k}{\partial x^s}(x)$ for $s \in \{1, ...n\}$.

$$\begin{aligned}
\frac{\partial \Gamma_k}{\partial x^s}(x) &= -\frac{2}{n(n-1)} \left[\frac{\partial}{\partial x^s} \left(\sum_{j=1, j\neq k}^{n} \log(1 - x^\top x_j) \right) \right] = \\
&= -\frac{2}{n(n-1)} \left[\sum_{j=1, j\neq k}^{n} \frac{\partial}{\partial x^s} \left(\log(1 - x^\top x_j) \right) \right] = \\
&= -\frac{2}{n(n-1)} \left[\sum_{j=1, j\neq k}^{n} \frac{-x_j^s}{1 - x^\top x_j} \right] = \\
&= \frac{2}{n(n-1)} \sum_{j=1, j\neq k}^{n} \frac{x_j^s}{1 - x^\top x_j}
\end{aligned}$$

Since we are interested in computing the Riemannian gradient of f in X, from now on $x = x_k$. We can conclude that

$$\text{grad}_{x_k} \Gamma_k = \frac{2}{n(n-1)} \Big(\sum_{j=1, j\neq k}^{n} \frac{x_j^1}{1 - x_k^\top x_j} \quad \dots \quad \sum_{j=1, j\neq k}^{n} \frac{x_j^n}{1 - x_k^\top x_j} \Big) \tag{4}$$

For the sake of simplicity we will consider the gradient as a column vector, so that $\text{grad}_{x_k} \Gamma_k$ is the transposed of the vector in (4). To obtain the Riemannian gradient of the extension of f in X we have to put together the gradients of Γ_k in x_k:

$$\begin{aligned}
\text{grad}_X \Gamma &= \frac{2}{n(n-1)} XW = \\
&= \frac{2}{n(n-1)} [x_1 \ x_2 \ \dots \ x_n] \begin{bmatrix} 0 & \frac{1}{1-x_1^\top x_2} & \dots & \frac{x_n^1}{1-x_1^\top x_n} \\ \frac{1}{1-x_2^\top x_1} & 0 & \dots & \frac{1}{1-x_2^\top x_n} \\ \vdots & \ddots & \vdots & \\ \frac{1}{1-x_n^\top x_1} & \frac{1}{1-x_n^\top x_2} & \dots & 0 \end{bmatrix}
\end{aligned}$$

where we used the notation $W_{ij} = \frac{1}{1-G_{ij}}$ if $i \neq j$ and $W_{ii} = 0$ and $G = X^\top X$. In order to project the gradient of Γ, we now substract at column k^{th} of $\text{grad}_x \Gamma$ the component of such column along the vector x_k. In conclusion

$$\text{grad}_X f = \frac{2}{n(n-1)} \Big(XW - X^* \text{diag}(\text{diag}(GW)) \Big)$$

9) The cost of computing the Riemannian gradient of f at X is $2n^3 = \mathcal{O}(n^3)$. In the hypothesis $n \gg d$, the cost of computing the Riemannian gradient of f is greater than the cost of computing f, which is $\mathcal{O}(n^2 d)$. The cost of computing f in X is $n^2 d$, which is the cost of computing the upper triangular half of the matrix product $G = X^\top X$. Once computed G, the cost of computing the gradient in X is dominated by the term $2n^3$, which is the cost of the computation GW.

10) Here is the implementation of the function to compute $f(X)$:

```
function y = f(X)
    n = size(X,2);
    y = -2/(n*(n-1)) * sum(sum(log(1 - triu(X' * X,1))));
end
```

and the code to compute the Riemannian gradient of f at the point X.

[3] The notation x^s is used to identify the s^{th} components of the vector x.

```
function V = gradf(X)
    n = size(X,2);
    G = X' * X;
    G2 = G - diag(diag(G));
    W = 1./(1-G2);
    W = W - diag(diag(W));
    V = 2/(n*(n-1)) * (X*W - X * diag(diag(G * W)));
end
```

To check the gradient, the following function was implemented:

```
function [result] = test_gradient(d,n)
    X = sampleOB(d,n);
    V = sampleTangent(X);
    fval = f(X);
    gradval = gradf(X);
    projGrad = project(X,gradval);
    if (norm(gradval - projGrad) > 1e-15)
        result = "Gradient not in the tangent space";
        return
    end
    scalarProd = riemannMetric(gradval,V);
    fprintf('Value of x is: %f\n', scalarProd);
    tt = logspace(1e-8,1,10);
    E = @(t) abs(f(retract(X, t * V)) - fval - t * scalarProd);
    loglog(tt,arrayfun(E,tt),'-o',tt,2*tt,'--')
end
```

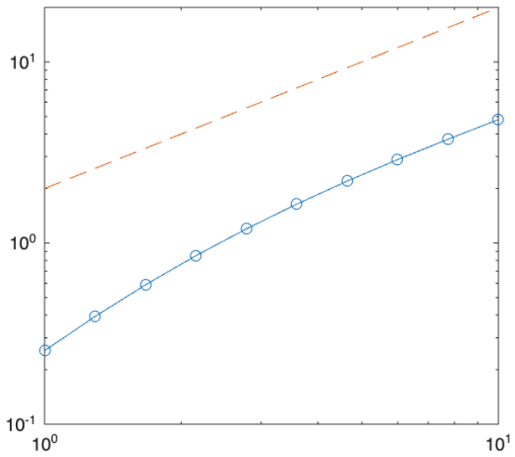


Figure 1: Gradient check plot.

11) The RGD algorithm with backtracking line-search has been implemented as follows

```
function [xmin, fvals, gradnorms] = RGD(f,gradf, x0)

    tic

    % backtracking line-search parameters
    tau = 0.5;
    r = 1e-4;
    alphabar = 10;

    % tolerance
    eps_grad = 1e-6;
    eps_fun = 1e-16;
    fundiff = eps_fun + 1;

    % initial values
    xk = x0;
    xold = x0;

```

```
    fvals = f(xold);
    gradval = gradf(xold);
    normgrad = sqrt(riemannMetric(gradval,gradval));
    gradnorms = normgrad;

    % max number of iterations
    maxit = 10000;

    k = 0;
    while(normgrad > eps_grad && k<maxit && fundiff > eps_fun)

        k = k + 1;

        %backtracking line-search
        fval = fvals(end);
        alpha = alphabar;
        while (fval - f(retract(xold, -alpha * gradval)) < r * alpha * normgrad^2
            && alpha>1e-16)
            alpha = alpha * tau;
            if(alpha == 0)
                disp("alpha")
            end
        end

        % configuration update
        xnew = retract(xold,-alpha*gradval);
        xk = [xk xnew];
        xold = xnew;

        %new values
        fval = f(xnew);
        fvals = [fvals fval];
        gradval = gradf(xnew);
        normgrad = sqrt(riemannMetric(gradval,gradval));
        gradnorms = [gradnorms normgrad];

        % function increment
        fundiff = abs(fvals(end-1) - fval);
    end

    % termination condition messages
    if(normgrad <= eps_grad)
        disp('Algorithm converged: gradient norm condition verified')
    elseif(fundiff <= eps_fun)
        disp('Algorithm converged: function increment condition verified')
    else
        disp('Algorithm did not converge: max number of iteration reached')
    end

    fprintf('Time: %f\n',toc)

    xmin = xnew;
    fmin = f(xmin);
    fvals = [fvals fmin];
end
```

4 Experiments

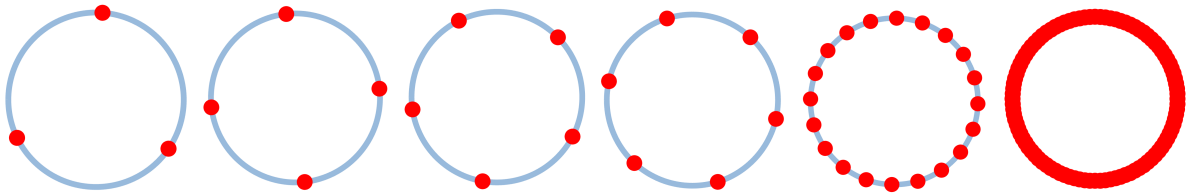
12) We ran 30 experiments for each configuration and for each we collected the smallest (Table 1) and biggest (Table 2) values obtained at convergence. Moreover, Figures 2 and 3, report the configurations reached for each pair (d, n).

Dimensions	**2**	**3**	**4**	**5**	**6**	**12**	**16**	**20**	**50**	**100**	**128**
2	-	-0.4055	-0.2310	-0.1116	-0.0236	-	-	0.3778	-	0.6001	-
3	-0.6931	-0.4055	-0.2877	-0.1910	-0.1386	0.0384	0.0914	-	0.2172	-	0.2648

Table 1: Results of the experiments: minimum value of the function obtained at convergence.

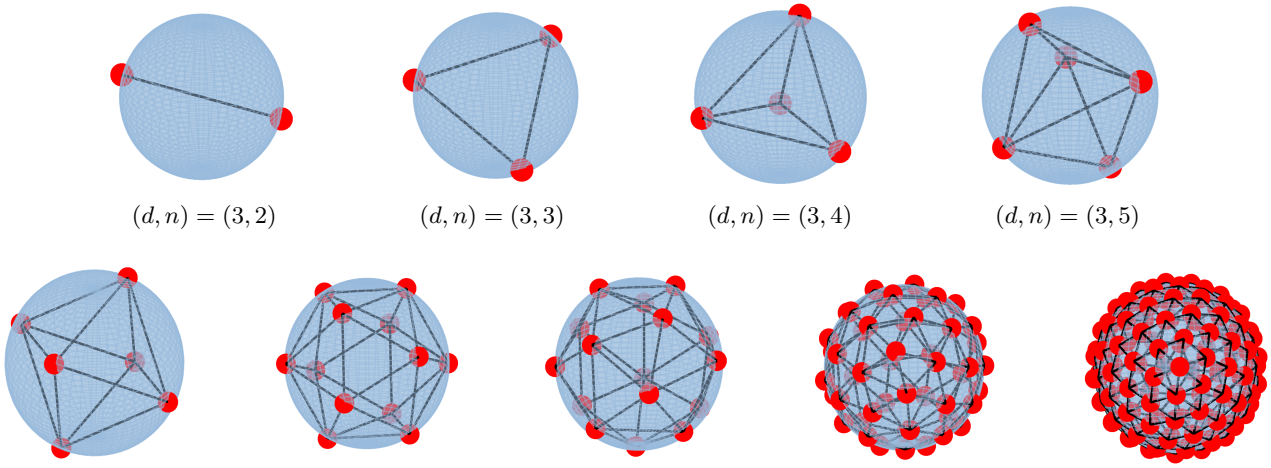
Dimensions	**2**	**3**	**4**	**5**	**6**	**12**	**16**	**20**	**50**	**100**	**128**
2	-	-0.4055	-0.2310	-0.1116	-0.0236	-	-	0.3778	-	0.6001	-
3	-0.6931	-0.4055	-0.2877	-0.1910	-0.1386	0.0384	0.0914	-	0.2172	-	0.2648

Table 2: Results of the experiments: maximum value of the function obtained at convergence.



$(d, n) = (2, 2)$ $(d, n) = (2, 4)$ $(d, n) = (2, 5)$ $(d, n) = (2, 6)$ $(d, n) = (2, 20)$ $(d, n) = (2, 100)$

Figure 2: Minimum configurations with $d = 2$.



$(d, n) = (3, 2)$ $(d, n) = (3, 3)$ $(d, n) = (3, 4)$ $(d, n) = (3, 5)$

$(d, n) = (3, 6)$ $(d, n) = (3, 12)$ $(d, n) = (3, 16)$ $(d, n) = (3, 50)$ $(d, n) = (3, 128)$

Figure 3: Minimum configurations with $d = 3$.

13) There is no change in the final value of the function across the various experiments in the same setting: the algorithm seemingly converges always to an equally optimal solution each time. The optimal configurations vary from one another just through rotations and this is reflected also in the rotational symmetry of the function: noting that the argument of the function only appears through a scalar product, applying a rotation matrix does not affect the result.
14) The same considerations above also hold for $d = 3$. For $n < 20$ we can notice from Figure 3 that the obtained configurations coincide with the theoretical ones and the values of the function calculated in the known optimal configurations are the same as those reported in Table 1 (or 2).

5 Proof of optimality for $d = 3$ and $n \in \{2, 3\}$

15) The function $f : \mathcal{M} \to \mathbb{R}$ is defined as

$$f(X) = f([x_1 \ \dots \ x_n]) = -\frac{2}{n(n-1)} \sum_{i<j} \log(1 - x_i^\top x_j)$$

Since $x_i, x_j \in S^{d-1}$ $\forall i, j \in \{1, \dots n\}$ the following relation holds:

$$\langle x_i, x_j \rangle = x_i^\top x_j = \cos\left(\widehat{x_i x_j}\right) \in [0, 1)$$

Indeed, thanks to the fact that $x_i \neq x_j$, the value 1 is not reached by the cosine function. Hence $1-x_i^\top x_j \in (0,1]$ and $\log(1-x_i^\top x_j)$ is a continuous function on this domain. This argument leads to conclude that f is a continuous function on $\mathcal{M}$ and that $\mathrm{Im}(f) \subseteq [0,+\infty)$.
Let us define

$$A := \{X \in \mathcal{M} | f(X) \leq 1\} \subseteq \mathcal{M}$$

The subset A is not empty: it is sufficient to take $x_1, \dots x_n \in S^{d-1}$ such that $\cos(\widehat{x_i x_j}) \leq 1-\frac{1}{\sqrt{e}} \backsimeq 0,393\ \forall i<j$. Indeed:

$$\begin{aligned}
&\cos(\widehat{x_i x_j}) \leq 1-\frac{1}{\sqrt{e}} \Longrightarrow x_i^\top x_j \leq 1-\frac{1}{\sqrt{e}} \Longrightarrow 1-x_i^\top x_j \geq \frac{1}{\sqrt{e}} \Longrightarrow \\
&\Longrightarrow \log(1-x_i^\top x_j) \geq -\frac{1}{2} \Longrightarrow \sum_{i<j} \log(1-x_i^\top x_j) \geq -\frac{n(n-1)}{4} \Longrightarrow f(x) \leq \frac{1}{2} \\
&\Longrightarrow f(x) \leq 1
\end{aligned}$$

Now we can state that

$$X \text{ is a minimizer for } f \text{ in } \mathcal{M} \Longleftrightarrow X \text{ is a minimizer for } f \text{ in } A$$

Let us note that:

A **closed:** A is the preimage, through a continuous function, of the subset $[0,1]$, which is closed in $\mathbb{R}$.

A **bounded:** From example 8.28 in the textbook, we know that S^{d-1} is a bounded subset of $\mathbb{R}^d$, and so is $\mathrm{OB}(d,n)$ in $\mathbb{R}^{d\times n}$ (it is sufficient to take the n-fold product of the set containing S^{d-1} in $\mathbb{R}$). Thus, every subset $\mathcal{M} \subset \mathrm{OB}(d,n)$ is bounded.

Due to the fact that we are working in an euclidean space, this is sufficient to demonstrate that A is compact. Since a continuous function on a compact set has a minimizer, we can conclude that f has a minimizer in A, therefore in $\mathcal{M}$.

16) Case $d=3$ and $n=2$
Consider the optimal configuration $X = [x_1\ x_2]$, where $x_i \in S^1 \subset \mathbb{R}^2$ for $i \in \{1,2\}$. Since X is a minimizer of f, it is a critical point of f and we can apply the result proved in the following subsections. Hence, we get:

$$x_1 + x_2 = 0 \Longrightarrow x_1 = -x_2$$

This means that x_1 and x_2 are two antipodal points in the one-dimensional sphere S^1.

Case $d=3$ and $n=3$
Consider $X = [x_1\ x_2\ x_3]$, where $x_i \in S^2 \subset \mathbb{R}^3$ for $i \in \{1,2,3\}$. Since X is a minimizer of f, it is a critical point of f and we can apply the results proved in the following points. From the relation

$$x_1 + x_2 + x_3 = 0 \tag{5}$$

follows that the barycentre of the three points x_1, x_2 and x_3 is the origin of $\mathbb{R}^3$. This means that the origin belongs to the plane where the three points lie. We can conclude that this plane is an equator of the sphere S^2. From now on we study the three points as elements of this plane and we want to show that they are the vertices of an equilateral triangle lying on the plane. Let us look at the relation

$$\sum_{i=1}^{3} \|x - x_i\|^2 = 2n = 6 \tag{6}$$

Since (6) holds for all $x \in S^2$, let us choose $x = x_1$, thus we get:

$$\begin{aligned}
\|x_1 - x_2\|^2 + \|x_1 - x_3\|^2 = 2 - 2x_1^\top x_2 + 2 - 2x_1^\top x_3 = 4 - 2x_1^\top x_2 - 2x_1^\top x_3 = 6 \\
\Longleftrightarrow x_1^\top x_2 + x_1^\top x_3 = -1
\end{aligned}$$

Using again (5), we know that $x_1 = -x_2 - x_3$ and we obtain

$$\begin{aligned}
(-x_2 - x_3)^\top x_2 + (-x_2 - x_3)^\top x_3 = -x_2^\top x_2 - x_3^\top x_2 - x_2^\top x_3 - x_3^\top x_3 = -2x_2^\top x_3 - 2 = -1 \\
\Longleftrightarrow -2x_2^\top x_3 = 1 \Longleftrightarrow x_2^\top x_3 = -\frac{1}{2}
\end{aligned}$$

Likewise, it is possible to obtain two more relations: $x_3^\top x_1 = -\frac{1}{2}$ and $x_1^\top x_2 = -\frac{1}{2}$. Exploiting that $x_1, x_2, x_3 \in S^2$, the three relations obtained can be rewritten as:

$$\cos(\widehat{x_1x_2}) = \cos(\widehat{x_2x_3}) = \cos(\widehat{x_3x_1}) = -\frac{1}{2} \quad \Longrightarrow \quad \widehat{x_1x_2} = \widehat{x_2x_3} = \widehat{x_3x_1} = \frac{2}{3}\pi$$

The implication above holds up to renaming the points. The last sequence of equalities demonstrates that the three points are the vertices of an equilateral triangle.

17) From the expression of the Riemannian gradient (cfr. Point 8) a critical point is a point $X \in \mathrm{OB}(d,n)$ such that:

$$(\mathbb{I} - x_k x_k^T) \sum_{i \neq k} \frac{x_i}{1 - x_i^T x_k} = 0 \quad \forall k \in \{1, \dots n\}$$

Noting that $\mathbb{I} - x_k x_k^T$ is the projector in the complement of the space generated by x_k the previous expression is equal to:

$$(\mathbb{I} - x_k x_k^T) \sum_{i \neq k} \frac{x_i - x_k}{1 - x_i^T x_k} = 2(\mathbb{I} - x_k x_k^T) \sum_{i \neq k} \frac{x_i - x_k}{1 - 2x_i^T x_k + 1} = 2(\mathbb{I} - x_k x_k^T) \sum_{i \neq k} \frac{x_i - x_k}{\|x_i - x_k\|^2} = 0 \tag{7}$$

The last equality in (7) could be written as

$$\sum_{i \neq k} \frac{x_i - x_k}{\|x_i - x_k\|^2} = x_k x_k^T \sum_{i \neq k} \frac{x_i - x_k}{\|x_i - x_k\|^2}$$

In conclusion:

$$\begin{aligned} \sum_{i \neq k} \frac{x_k - x_i}{\|x_i - x_k\|^2} &= x_k x_k^T \sum_{i \neq k} \frac{x_k - x_i}{\|x_i - x_k\|^2} = x_k \sum_{i \neq k} \frac{1 - x_k^T x_i}{\|x_i - x_k\|^2} = \\ &= \frac{1}{2} x_k \sum_{i \neq k} \frac{2 - 2x_k^T x_i}{\|x_i - x_k\|^2} = \frac{1}{2} x_k \sum_{i \neq k} \frac{\|x_i - x_k\|^2}{\|x_i - x_k\|^2} = x_k \frac{N-1}{2} \end{aligned} \tag{8}$$

18) Since the relation before holds for all k we can sum all those relations and get:

$$\sum_k \sum_{i \neq k} \frac{x_k - x_i}{\|x_i - x_k\|^2} = \frac{N-1}{2} \sum_k x_k$$

The sum on the left hand side is the sum of all the elements of the following matrix

$$\begin{bmatrix} 0 & \frac{x_1 - x_2}{\|x_2 - x_1\|^2} & \frac{x_1 - x_3}{\|x_3 - x_1\|^2} & \dots & \frac{x_1 - x_n}{\|x_n - x_1\|^2} \\ \frac{x_2 - x_1}{\|x_1 - x_2\|^2} & 0 & \frac{x_2 - x_3}{\|x_3 - x_2\|^2} & \dots & \frac{x_2 - x_n}{\|x_n - x_2\|^2} \\ \vdots & & & & \vdots \\ \frac{x_n - x_1}{\|x_1 - x_n\|^2} & \frac{x_n - x_2}{\|x_2 - x_n\|^2} & \frac{x_n - x_3}{\|x_3 - x_n\|^2} & \dots & 0 \end{bmatrix}$$

Since this matrix is antisymmetric, the sum of all its elements is zero. Exploiting that the factor $\frac{N-1}{2}$ is not zero, it is demonstrated that

$$\sum_k x_k = 0$$

19) Given that $x \in S^{d-1}$ and $x_i \in S^{d-1}$ for all i, we know that $\|x\|^2 = \|x_i\|^2 = 1$. Using this, we can compute the sum of squared distances as follows:

$$\sum_i \|x - x_i\|^2 = \sum_i \left(\|x\|^2 + \|x_i\|^2 - 2x^T x_i \right) = \sum_i (1 + 1 - 2x^T x_i) = \sum_i 2 - 2x^T x_i$$

Factoring out the constants:

$$\sum_i \|x - x_i\|^2 = 2n - 2x^T \sum_i x_i$$

From the previous result, we have $\sum_i x_i = 0$, so:

$$\sum_i \|x - x_i\|^2 = 2n$$

20) In point (16) we found that for $n = 2$ and $n = 3$ being a critical point implies having an optimal configuration, i.e. being a global minimizer, and thus no non-optimal critical points exist, making convergence of Riemannian gradient descent to the global minimum highly plausible. For $n \geq 4$, however, we conjecture the existence of non-optimal critical points, e.g. for $n = 4$ a configuration forming a square on an equatorial plane. Consider the four unit vectors at the corners of a square in $\mathbb{R}^3$:

$$x_1 = \begin{bmatrix}1\\0\\0\end{bmatrix}, \quad x_2 = \begin{bmatrix}0\\1\\0\end{bmatrix}, \quad x_3 = \begin{bmatrix}-1\\0\\0\end{bmatrix}, \quad x_4 = \begin{bmatrix}0\\-1\\0\end{bmatrix}$$

We check the condition for $k = 1$:

$$\left(\mathbb{I} - x_1 x_1^T\right) \sum_{i \neq 1} \frac{x_i}{1 - x_i^T x_1} = \left(\mathbb{I} - x_1 x_1^T\right)\left(x_2 + \frac{1}{2}x_3 + x_4\right) = x_2 + \frac{1}{2}x_3 + x_4 + \frac{1}{2}x_1 = 0$$

noticing that $x_2 = -x_4$ and $x_3 = -x_1$. With completely analogous computations, the same result can be obtained for each component of the gradient, showing that this configuration is indeed a critical point. The value of the function, though, is -0.2310 (that reasonably coincides with the minimum of the function with 4 points in $\mathbb{R}^2$) which is higher than the minimum obtained with the tetrahedron.
These stationary points for f which are not global minima, start arising in higher dimensions indicating that the landscape becomes more intricate and the method may converge to suboptimal configurations.

6 Using (Py)Manopt

21) The optimization setup with Manopt is reported below:

```
d = 3;
n = 128;
manifold = obliquefactory(d,n) ;
problem.M = manifold ;
problem.cost = @(x) f(x);
problem.egrad = @(x) egradf(x);
checkgradient(problem);
```

where $f(x)$ id the function implemented in point 10) and the euclidean gradient was implemented as follows

```
function V = egradf(X)
    n = size(X,2);
    G = X' * X;
    G2 = G - diag(diag(G));
    W = 1./(1-G2);
    W = W - diag(diag(W));
    V = 2/(n*(n-1)) * (X*W);
end
```

The gradient check tool reported the plot in figure 4 and the following message:

```
# Gradient check
The slope should be 2. It appears to be: 2.00003.
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: 2.62812e-16.
If it is far from 0, the gradient is not tangent.
```

22) The optimization yields the same optimal value of 0.2648 as our implementation in a comparable time frame.

23) The conjugate gradient descent algorithm reaches again the same result, though in smaller amount of steps.

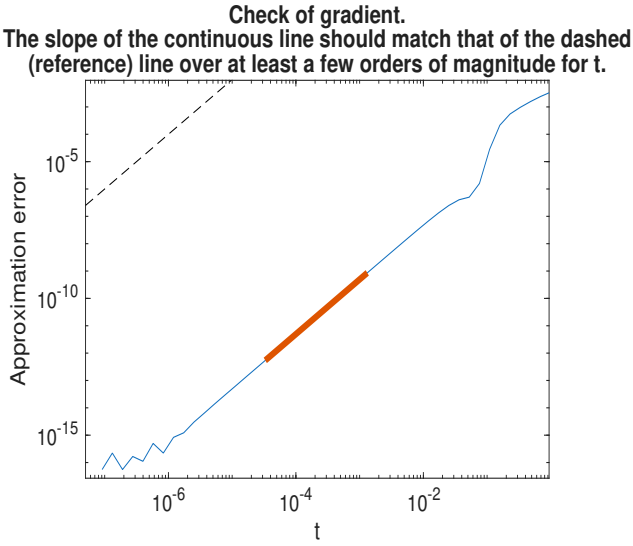


Figure 4: Plot of the gradient check.

7 Dealing with Δ

24) Let's consider for this point $X = [x\ y] \in \mathrm{OB}(2,2)\backslash\Delta$ and the exponential map as a retraction, i.e. the updates are of the form:

$$x_{k+1} = \cos(\alpha_k\|\nabla f_x(X_k)\|)x_k + \frac{\sin(\alpha_k\|\nabla f_x(X_k)\|)}{\alpha_k\|\nabla f_x(X_k)\|}(-\alpha_k\nabla f_x(X_k))$$

$$y_{k+1} = \cos(\alpha_k\|\nabla f_y(X_k)\|)y_k + \frac{\sin(\alpha_k\|\nabla f_y(X_k)\|)}{\alpha_k\|\nabla f_y(X_k)\|}(-\alpha_k\nabla f_y(X_k))$$

We want to determine α_k such that we have

$$x_{k+1} = y_{k+1} \tag{9}$$

We notice that the two components of the Riemannian gradient $\nabla f_x(X_k)$ and $\nabla f_y(X_k)$ have the same norm, in fact

$$\|\nabla f_x(X_k)\|^2 = \left\|(\mathbb{I} - x_k x_k^\top)\frac{y_k}{1 - x_k^\top y_k}\right\|^2 = \frac{1}{(1 - x_k^\top y_k)^2}\left(\|y_k\|^2 + \|x_k\|^2(x_k^\top y_k)^2 - 2(x_k^\top y_k)^2\right) = \frac{1 - (x_k^\top y_k)^2}{(1 - x_k^\top y_k)^2}$$

$$\|\nabla f_y(X_k)\|^2 = \left\|(\mathbb{I} - y_k y_k^\top)\frac{x_k}{1 - y_k^\top x_k}\right\|^2 = \frac{1}{(1 - y_k^\top x_k)^2}\left(\|x_k\|^2 + \|y_k\|^2(y_k^\top x_k)^2 - 2(y_k^\top x_k)^2\right) = \frac{1 - (y_k^\top x_k)^2}{(1 - y_k^\top x_k)^2}$$

The equality follows by symmetry of the euclidean inner product. We pose $\|\nabla f_x(X_k)\| = \|\nabla f_y(X_k)\| = c$ and $\alpha_k c = \theta$, so that 9 reads

$$\cos(\theta)x_k - \frac{\sin(\theta)}{c}\nabla f_x(X_k) = \cos(\theta)y_k - \frac{\sin(\theta)}{c}\nabla f_y(X_k)$$

By rearranging the terms we obtain

$$\cos(\theta)(x_k - y_k) = \frac{\sin(\theta)}{c}(\nabla f_x(X_k) - \nabla f_y(X_k))$$

To obtain a scalar equation we multiply both terms by the transpose of $x_k - y_k$, which yields to

$$\cos\theta\|x_k - y_k\|^2 = \sin\theta\frac{\langle\nabla f_x(X_k) - \nabla f_y(X_k), x_k - y_k\rangle}{c}$$

Dividing by $\cos\theta$ we obtain

$$\tan\theta = c\frac{\|x_k - y_k\|^2}{\langle \nabla f_x(X_k) - \nabla f_y(X_k), x_k - y_k\rangle}$$

This equation has infinitely many solutions in the form:

$$\theta = \arctan\left[c\frac{\|x_k - y_k\|^2}{\langle \nabla f_x(X_k) - \nabla f_y(X_k), x_k - y_k\rangle}\right] + m\pi$$

with $m \in \mathbb{Z}$. To obtain a positive value of α_k, coherently with the gradient descent, we choose $m = 1$.

$$\alpha_k = \frac{1}{c}\arctan\left[c\frac{\|x_k - y_k\|^2}{\langle \nabla f_x(X_k) - \nabla f_y(X_k), x_k - y_k\rangle}\right] + \frac{\pi}{c}$$

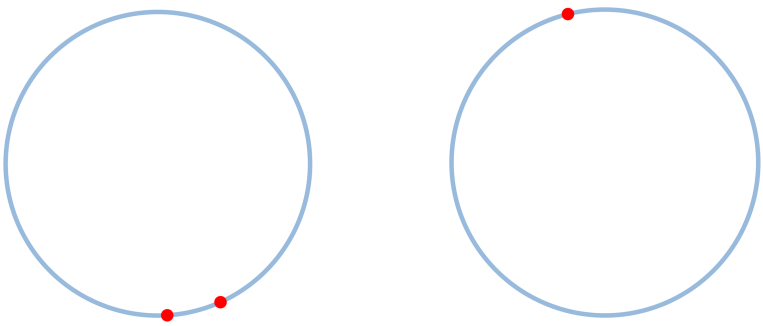


Figure 5: The points coincide after one step.

Appendix A

How to run the code

Opening the folder, you will find the following functions:

- `egrad.m`: computes the euclidean gradient of f
- `experiment.m`: runs a single experiment
- `f.m`: computes the value of f
- `gradf.m`: computes the Riemannian gradient of f
- `project.m`: computes the projection on the tangent space
- `retract.m`: computes the retraction
- `RGD.m`: Riemannian gradient descent algorithm
- `riemannMetric.m`: computes the Riemannian metric
- `sampleOB.m`: samples a random point on the manifold
- `sampleTangent.m`: samples a random point on the tangent space
- `test_gradient.m`: tests the correctness of the gradient

The runnable codes are:

- `main.m`: runs the experiments in the requested configurations (first section sets up the table to save the values, second sections runs experiments for $d = 2$, third section runs the experiments for $d = 3$)
- `tests.m`: runs the tests on the projection and on the retraction
- `manopt_solve.m`: runs the optimisation algorithms with Manopt (first section with Steepest Descent, second section with Conjugate Gradient)