A. Proofs

Sketch of the proof for Theorem 1. We need to the show that for every $\tilde{Q} \in \mathbf{O}(n)$, there exits a tuple of vectors $(\mathbf{u}_1, \dots, \mathbf{u}_n) \in \mathbb{R} \times \cdots \times \mathbb{R}^n$ such that $\tilde{Q} = \mathcal{M}_1(\mathbf{u}_1, \dots, \mathbf{u}_n)$. Algorithm 1 shows how a QR decomposition can be performed using the matrices $\{\mathcal{H}_k(\mathbf{u}_k)\}_{k=1}^n$ while ensuring that the upper triangular matrix R has positive diagonal elements. If we apply this algorithm to an orthogonal matrix $\tilde{Q}$, we get a tuple $(\mathbf{u}_1, \dots, \mathbf{u}_n)$ which satisfies

$$QR = \mathcal{H}_n(\mathbf{u}_n) \dots \mathcal{H}_1(\mathbf{u}_1) R = \tilde{Q}.$$

Note that the matrix R must be orthogonal since $R = Q'\tilde{Q}$. Therefore, $R = I$, since the only upper triangular matrix with positive diagonal elements is the identity matrix. Hence, we have

$$\mathcal{M}_1(\mathbf{u}_1, \dots, \mathbf{u}_n) = \mathcal{H}_n(\mathbf{u}_n) \dots \mathcal{H}_1(\mathbf{u}_1) = \tilde{Q}.$$

□

Algorithm 1 QR decomposition using the mappings $\{\mathcal{H}_k\}$. For a matrix $B \in \mathbb{R}^{n \times n}$, $\{B_{k,k}\}_{1 \le k \le n}$ denote its diagonal elements, and $B_{k..n,k} = (B_{k,k}, \dots, B_{n,k})' \in \mathbb{R}^{n-k+1}$.

Require: $A \in \mathbb{R}^{n \times n}$ is a full-rank matrix.
Ensure: Q and R where $Q = \mathcal{H}_n(\mathbf{u}_n) \dots \mathcal{H}_1(\mathbf{u}_1)$ and R is upper triangular with positive diagonal elements such that $A = QR$

$R \leftarrow A$
$Q \leftarrow I$ {Initialise Q to the identity matrix}
for $k = 1$ to $n - 1$ **do**
  if $R_{k,k} == \|R_{k..n,k}\|$ **then**
    $\mathbf{u}_{n-k+1} = (0, \dots, 0, 1)' \in \mathbb{R}^{n-k+1}$
  else
    $\mathbf{u}_{n-k+1} \leftarrow R_{k..n,k} - \|R_{k..n,k}\| (1, 0, \dots, 0)'$
    $\mathbf{u}_{n-k+1} \leftarrow \mathbf{u}_{n-k+1} / \|\mathbf{u}_{n-k+1}\|$
  end if
  $R \leftarrow \mathcal{H}_{n-k+1}(\mathbf{u}_{n-k+1}) R$
  $Q \leftarrow Q \mathcal{H}_{n-k+1}(\mathbf{u}_{n-k+1})$
end for
$\mathbf{u}_1 = sgn(R_{n,n}) \in \mathbb{R}$
$R \leftarrow \mathcal{H}_1(\mathbf{u}_1) R$
$Q \leftarrow Q \mathcal{H}_1(\mathbf{u}_1)$

Lemma 1. *(Giles, 2008) Let A, B, and C be real or complex matrices, such that $C = f(A, B)$ where f is some differentiable mapping. Let $\mathcal{L}$ be some scalar quantity which depends on C. Then we have the following identity*

$$Tr(\overline{C}' dC) = Tr(\overline{A}' dA) + Tr(\overline{B}' dB),$$

where dA, dB, and dC represent infinitesimal perturbations and

$$\overline{C} := \frac{\partial \mathcal{L}}{\partial C}, \ \overline{A} := \left[\frac{\partial C}{\partial A}\right]' \frac{\partial \mathcal{L}}{\partial C}, \ \overline{B} := \left[\frac{\partial C}{\partial B}\right]' \frac{\partial \mathcal{L}}{\partial C}.$$

Proof of Theorem 2. Let $C = h - UT^{-1}U'h$ where $(U, h) \in \mathbb{R}^{n \times m} \times \mathbb{R}^n$ and $T = \text{striu}(U'U) + \frac{1}{2}\text{diag}(U'U)$. Notice that the matrix T can be written using the Hadamard product as follows

$$T = B \circ (U'U), \tag{1}$$

where $B = \text{striu}(J_m) + \frac{1}{2} I_m$ and J_m is the $m \times m$ matrix of all ones.

Calculating the infinitesimal perturbations of C gives

$$\begin{aligned} dC =&(I - UT^{-1}U')dh \\ &- dUT^{-1}U'h - UT^{-1}dU'h \\ &+ UT^{-1}dTT^{-1}U'h. \end{aligned}$$

Using Equation (1) we can write

$$dT = B \circ (dU'U + U'dU).$$

By substituting this back into the expression of dC, multiplying the left and right-hand sides by $\overline{C}'$, and applying the trace we get

$$\begin{aligned} \text{Tr}(\overline{C}' dC) &= \text{Tr}(\overline{C}'(I - UT^{-1}U')dh) \\ &- \text{Tr}(\overline{C}' dUT^{-1}U'h) - \text{Tr}(\overline{C}' UT^{-1}dU'h) \\ &+ \text{Tr}(\overline{C}' UT^{-1}(B \circ (dU'U + U'dU))T^{-1}U'h). \end{aligned}$$

Now using the identity $\text{Tr}(AB) = \text{Tr}(BA)$, where the second dimension of A agrees with the first dimension of B, we can rearrange the expression of $\text{Tr}(\overline{C}' dC)$ as follows

$$\begin{aligned} \text{Tr}(\overline{C}' dC) &= \text{Tr}(\overline{C}'(I - UT^{-1}U')dh) \\ &- \text{Tr}(T^{-1}U'h\overline{C}' dU) - \text{Tr}(h\overline{C}' UT^{-1}dU') \\ &+ \text{Tr}(T^{-1}U'h\overline{C}' UT^{-1}(B \circ (dU'U + U'dU))). \end{aligned}$$

To simplify the expression, we will use the short notations

$$\begin{aligned} \tilde{C} &= (T')^{-1}U'\overline{C}, \\ \tilde{h} &= T^{-1}U'h, \end{aligned}$$

$\text{Tr}(\overline{C}' dC)$ becomes

$$\begin{aligned} \text{Tr}(\overline{C}' dC) &= \text{Tr}((\overline{C}' - \tilde{C}'U')dh) \\ &- \text{Tr}(\tilde{h}\overline{C}' dU) - \text{Tr}(h\tilde{C}' dU') \\ &+ \text{Tr}(\tilde{h}\tilde{C}'(B \circ (dU'U + U'dU))). \end{aligned}$$

Now using the two following identities of the trace

$$\mathrm{Tr}(A') = \mathrm{Tr}(A),$$
$$\mathrm{Tr}(A(B \circ C)) = \mathrm{Tr}((A \circ B')C)),$$

we can rewrite $\mathrm{Tr}(\overline{C}' dC)$ as follows

$$\begin{aligned}\mathrm{Tr}(\overline{C}' dC) =& \mathrm{Tr}((\overline{C}' - \tilde{C}'U')dh) \\ &- \mathrm{Tr}(\tilde{h}\overline{C}' dU) - \mathrm{Tr}(h\tilde{C}' dU') \\ &+ \mathrm{Tr}((\tilde{h}\tilde{C}' \circ B')dU'U) \\ &+ \mathrm{Tr}((\tilde{h}\tilde{C}' \circ B')U'dU).\end{aligned}$$

By rearranging and taking the transpose of the third and fourth term of the right-hand side we obtain

$$\begin{aligned}\mathrm{Tr}(\overline{C}' dC) =& \mathrm{Tr}((\overline{C}' - \tilde{C}'U')dh) \\ &- \mathrm{Tr}(\tilde{h}\overline{C}' dU) - \mathrm{Tr}(\tilde{C}h' dU) \\ &+ \mathrm{Tr}(((\tilde{C}\tilde{h}') \circ B)U'dU) \\ &+ \mathrm{Tr}(((\tilde{h}\tilde{C}') \circ B')U'dU).\end{aligned}$$

Factorising by dU inside the Tr we get

$$\begin{aligned}&\mathrm{Tr}(\overline{C}' dC) = \mathrm{Tr}((\overline{C}' - \tilde{C}'U')dh) - \\ &\mathrm{Tr}((\tilde{h}\overline{C}' + \tilde{C}h' - \left[(\tilde{C}\tilde{h}') \circ B + (\tilde{h}\tilde{C}') \circ B'\right] U')dU).\end{aligned}$$

Using lemma 1 we conclude that

$$\begin{aligned}\overline{U} =& U\left[(\tilde{h}\tilde{C}') \circ B' + (\tilde{C}\tilde{h}') \circ B\right] - \overline{C}\tilde{h}' - h\tilde{C}', \\ \overline{h} =& \overline{C} - U\tilde{C}.\end{aligned}$$

□

Sketch of the proof for Corollary 1. For any nonzero complex valued vector $x \in \mathbb{C}^n$, if we chose $u = x + e^{i\theta}\left\|x\right\| e_1$ and $H = -e^{-i\theta}(I - 2\frac{uu^*}{\|u\|^2})$, where $\theta \in \mathbb{R}$ is such that $x_1 = e^{i\theta}|x_1|$, we have (Mezzadri, 2006)

$$Hx = ||x||e_1 \tag{2}$$

Taking this fact into account, a similar argument to that used in the proof of Theorem 1 can be used here. □

B. Algorithm Explanation

Let $U := (v_{i,j})^{1\leq j\leq m}_{1\leq i\leq n}$. Then the element of the matrix $T := \mathrm{striu}(U'U) + \frac{1}{2}\mathrm{diag}(U'U)$ can be expressed as

$$t_{i,j} = [\![i \leq j]\!] \frac{\sum_{k=j}^{n} v_{k,i}v_{k,j}}{1 + \delta_{i,j}},$$

where $\delta_{i,j}$ is the Kronecker delta and $[\![\cdot]\!]$ is the Iversion bracket (i.e. $[\![p]\!] = 1$ if p is true and $[\![p]\!] = 0$ otherwise).

In order to compute the gradients in Equations (14) and (15). we first need to compute $\tilde{h} = T^{-1}U'h$ and $\tilde{C} = (T')^{-1}U'\frac{\partial\mathcal{L}}{\partial C}$. This is equivalent to solving the triangular systems of equations $T\tilde{h} = U'h$ and $T'\tilde{C} = U'\frac{\partial\mathcal{L}}{\partial C}$.

Solving the triangular system $T\tilde{h} = U'h$. For $1 \leq k \leq m$, we can express the k-th row of this system as

$$\begin{aligned}t_{k,k}\tilde{h}_k + \sum_{j=k+1}^{m} t_{k,j}\tilde{h}_j &= \sum_{j=k}^{n} v_{j,k}h_j, \\ &= \sum_{j=k}^{n} v_{j,k}h_j - \sum_{j=k+1}^{m}\sum_{r=j}^{n} v_{r,k}v_{r,j}\tilde{h}_j, \\ &= \sum_{r=k}^{n} v_{r,k}h_r - \sum_{r=k+1}^{n} v_{r,k}\sum_{j=k+1}^{r} v_{r,j}\tilde{h}_j, \qquad (3) \\ &= U'_{*,k}(h - \sum_{j=k+1}^{m} U_{*,j}\tilde{h}_j), \qquad (4)\end{aligned}$$

where the passage from Equation (3) to (4) is justified because $v_{r,j} = 0$ for $j > r$. Therefore, $\sum_{j=k+1}^{r} v_{r,j}\tilde{h}_j = \sum_{j=k+1}^{m} v_{r,j}\tilde{h}_j$.

By setting $H_{*,k+1} := h - \sum_{j=k+1}^{m} U_{*,j}\tilde{h}_j$, and noting that $t_{k,k} = \frac{U'_{*,k}U_{*,k}}{2}$, we get

$$\tilde{h}_k = \frac{2}{U'_{*,k}U_{*,k}}U'_{*,k}H_{*,k+1}, \tag{5}$$

$$H_{*,k} = H_{*,k+1} - \tilde{h}_k U_{*,k}. \tag{6}$$

Equations (5) and (6) explain the lines 8 and 9 in Algorithm 1. Note that $H_{*,1} = h - \sum_{j=1}^{m} U_{*,j}\tilde{h}_j = h - \sum_{j=1}^{m} U_{*,j}[T^{-1}U'h]_j = h - UT^{-1}U'h = Wh$. Hence, when $h = h^{(t-1)}$, we have $H_{*,1} = C^{(t)}$, which explains line 16 in Algorithm 1.

Solving the triangular system $T'\tilde{C} = U'\frac{\partial\mathcal{L}}{\partial C}$. Similarly to the previous case, we have for $1 \leq k \leq m$

$$\begin{aligned}t_{k,k}\tilde{C}_k + \sum_{j=1}^{k-1} t_{j,k}\tilde{C}_j &= \sum_{j=k}^{n} v_{j,k}\left[\frac{\partial\mathcal{L}}{\partial C}\right]_j, \\ &= \sum_{j=1}^{n} v_{j,k}\left[\frac{\partial\mathcal{L}}{\partial C}\right]_j - \sum_{j=1}^{k-1}\sum_{r=k}^{n} v_{r,j}v_{r,k}\tilde{C}_j, \qquad (7) \\ &= \sum_{r=1}^{n} v_{r,k}\left[\frac{\partial\mathcal{L}}{\partial C}\right]_r - \sum_{r=1}^{n} v_{r,k}\sum_{j=1}^{k-1} v_{r,j}\tilde{C}_j, \qquad (8) \\ &= U'_{*,k}\left(\frac{\partial\mathcal{L}}{\partial C} - \sum_{j=1}^{k-1} U_{*,j}\tilde{C}_j\right),\end{aligned}$$

	Operation	Flop count for iteration k	Total Flop count for m iteration
FP	$\tilde{h}_k \leftarrow \frac{2}{N_k} U'_{*,k} H_{*,k+1}$	$2(n-k)+3$	$(4n-m+2)m$
	$H_{*,k} \leftarrow H_{*,k+1} - \tilde{h}_k U_{*,k}$	$2n$	
BP	$\tilde{h}_k \leftarrow \frac{2}{N_k} U'_{*,k} H_{*,k+1}$	$2(n-k)+3$	$(7n-2m+3)m$
	$g \leftarrow g - \tilde{C}_k U_{*,k}$	$2(n-k+1)$	
	$G_{*,k} \leftarrow -\tilde{h}_k g - \tilde{C}_k H_{*,k+1}$	$3n$	

Table 1. Time complexities of different operations in algorithm 1. It is assumed that the matrix $U \in \mathbb{R}^{n \times m}$ is defined as in Equation (9).

where the passage from Equation (7) to (8) is justified by the fact that $\sum_{r=k}^{n} v_{r,j} v_{r,k} \tilde{C}_j = \sum_{r=1}^{n} v_{r,j} v_{r,k} \tilde{C}_j$ (since $v_{r,k} = 0$ for $r < k$).

By setting $g := \frac{\partial \mathcal{L}}{\partial C^{(t)}} - \sum_{j=1}^{k-1} U_{*,j} \tilde{C}_j$, we can write $\tilde{C}_k = \frac{2}{U'_{*,k} U_{*,k}} U'_{*,k} g$ which explains the lines 12 and 13 in Algorithm 1. Note also that after m-iterations in the backward propagation loop in Algorithm 1, we have $g = \frac{\partial \mathcal{L}}{\partial C^{(t)}} - \sum_{j=1}^{m} U_{*,j} \tilde{C}_j = \frac{\partial \mathcal{L}}{\partial C^{(t)}} - U\tilde{C} = \frac{\partial \mathcal{L}}{\partial h^{(t-1)}}$. This explains line 17 of Algorithm 1.

Finally, note that from Equation (14), we have for $1 \leq i \leq n$ and $1 \leq k \leq m$

$$
\begin{aligned}
\left[\frac{\partial \mathcal{L}}{\partial U}\right]_{i,k} =& -\left[\frac{\partial \mathcal{L}}{\partial C}\right]_i \tilde{h}_k - h_i \tilde{C}_k + \\
& \sum_{j=1}^{m} v_{i,j} \left(((\tilde{h}\tilde{C}') \circ B')_{j,k} + ((\tilde{C}\tilde{h}') \circ B)_{j,k} \right), \\
=& -\left[\frac{\partial \mathcal{L}}{\partial C}\right]_i \tilde{h}_k - h_i \tilde{C}_k + \\
& \sum_{j=1}^{m} v_{i,j} \left(\tilde{h}_j \tilde{C}_k \frac{[\![k \leq j]\!]}{1+\delta_{j,k}} + \tilde{C}_j \tilde{h}_k \frac{[\![j \leq k]\!]}{1+\delta_{j,k}} \right), \\
=& -\left[\frac{\partial \mathcal{L}}{\partial C}\right]_i \tilde{h}_k - h_i \tilde{C}_k + \\
& \sum_{j=1}^{m} v_{i,j} \left(\tilde{h}_j \tilde{C}_k [\![k < j]\!] + \tilde{C}_j \tilde{h}_k [\![j \leq k]\!] \right), \\
=& \tilde{C}_k \left(\sum_{j=k+1}^{m} v_{i,j} \tilde{h}_j - h_i \right) \\
& + \tilde{h}_k \left(\sum_{j=1}^{k} v_{i,j} \tilde{C}_j - \left[\frac{\partial \mathcal{L}}{\partial C}\right]_i \right).
\end{aligned}
$$

Therefore, when $C = C^{(t)}$ and $h = h^{(t-1)}$ we have

$$
\left[\frac{\partial \mathcal{L}}{\partial U^{(t)}}\right]_{*,k} = -\tilde{C}_k H_{*,k+1} - \tilde{h}_k g,
$$

where $g = \frac{\partial \mathcal{L}}{\partial C^{(t)}} - \sum_{j=1}^{k-1} \tilde{C}_j U_{*,j}$. This explains lines 14 and 18 of Algorithm 1.

C. Time complexity

Table 1 shows the flop count for different operations in the local backward and forward propagation steps in Algorithm 1.

D. Matlab implementation of Algorithm 1

```
% Inputs: U - matrix of reflection vectors
%         h - hidden state at time-step t-1
%         BPg - Grad of loss w.r.t C=Wh
% Outputs: g, G, C=Wh
[n, m] = size(U);
G=zeros(n, m); H = zeros(n, m+1);
N = zeros(m); h_tilde = zeros(m);
%  Zero-initialisation not required above!
H(:,m+1)=h; g=BPg;
%%--Forward propagation--%%
for k =0:m-1
    N(m-k) = U(:, m-k)' * U(:, m-k);
    h_tilde(m-k)=2 / N(m-k) * ...
            U(:, m-k)' * H(:,m-k+1);
    H(:,m-k)=H(:,m-k+1) - ...
            h_tilde(m-k) * U(:,m-k);
end
C = H(:,1)
%%--Backward propagation--%%
for k=1:m
    c_tilde_k = 2*U(:,k)' * g / N(k);
    g = g - c_tilde_k * U(:,k);
    G(:, k)=-h_tilde(k) * g - ...
            c_tilde_k*H(:,k+1);
end
```

Figure 1. MATLAB code performing one-step FP and BP required to compute $C^{(t)}$, $\frac{\partial \mathcal{L}}{\partial h^{(t-1)}}$ (variable g is the code), and $\frac{\partial \mathcal{L}}{\partial U^{(t)}}$ (variable G is the code). The required inputs for the FP and BP are, respectively, the tuples $(U, h^{(t-1)})$ and $(U, C^{(t)}, \frac{\partial \mathcal{L}}{\partial C^{(t)}})$. Note that $\frac{\partial \mathcal{L}}{\partial C^{(t)}}$ is variable BPg in the Matlab code.

References

Giles, Mike B. An extended collection of matrix derivative results for forward and reverse mode automatic differentiation. 2008.

Mezzadri, Francesco. How to generate random matrices from the classical compact groups. *arXiv preprint math-ph/0609050*, 2006.