A. Proofs

Sketch of the proof for Theorem 1. We need to show that for every $Q \in O(n)$, there exists a tuple of vectors $(u_1, \ldots, u_n) \in \mathbb{R} \times \cdots \times \mathbb{R}^n$ such that $Q = Q_1(u_1, \ldots, u_n)$. Algorithm 1 shows how a QR decomposition can be performed using the matrices \{H_k(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 $Q$, we get a tuple $(u_1, \ldots, u_n)$ which satisfies

$$QR = H_n(u_n) \ldots H_1(u_1) R = \tilde{Q}.$$ 

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

$$M_1(u_1, \ldots, u_n) = H_n(u_n) \ldots H_1(u_1) = \tilde{Q}.$$ 

\hfill $\square$

Algorithm 1 QR decomposition using the mappings \{H_k\}.

For a matrix $B \in \mathbb{R}^{n \times n}$, \{B_{k,k}\}_{1 \leq k \leq n} denote its diagonal elements, and $B_{k,n,k} = (B_{k,k}, \ldots, B_{n,k})' \in \mathbb{R}^{n-k+1}$.

Require: $A \in \mathbb{R}^{n \times n}$ is a full-rank matrix.

Ensure: $Q$ and $R$ where $Q = H_n(u_n) \ldots H_1(u_1)$ and $R$ is upper triangular with positive diagonal elements such that $A = QR$

$Q \leftarrow I$ \{Initialise $Q$ to the identity matrix\}

for $k$ from $1$ to $n-1$ do

  if $R_{k,k} = \|R_{k,n,k}\|$ then

    $u_{n-k+1} = (0, \ldots, 0, 1)' \in \mathbb{R}^{n-k+1}$

  else

    $u_{n-k+1} = R_{k,n,k} - \|R_{k,n,k}\| (1, 0, \ldots, 0)'$

    $u_{n-k+1} = u_{n-k+1} / \|u_{n-k+1}\|$ end if

  $R \leftarrow H_{n-k+1}(u_{n-k+1}) R$

  $Q \leftarrow QH_{n-k+1}(u_{n-k+1})$

end for

$u_1 = \text{sgn}(R_{1,n}) \in \mathbb{R}$

$R \leftarrow H_1(u_1) R$

$Q \leftarrow QH_1(u_1)$

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

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

where $dA$, $dB$, and $dC$ represent infinitesimal perturbations and

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

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 $T = \text{streu}(U'U) + \frac{1}{2} \text{diag}(U'U)$. Notice that the matrix $T$ can be written using the Hadamard product as follows

$$T = B \circ (U'U), \quad (1)$$

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

Calculating the infinitesimal perturbations of $C$ gives

$$dC = (I - UT^{-1}U')dh$$

$$- dUT^{-1}U'h - UT^{-1}dU'h$$

$$+ UT^{-1}dT T^{-1}U'h.$$ 

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 $\tilde{C}'$, and applying the trace we get

$$\text{Tr}(\tilde{C}'dC) = \text{Tr}(\tilde{C}'(I - UT^{-1}U')dh)$$

$$- \text{Tr}(\tilde{C}'dUT^{-1}U'h) - \text{Tr}(\tilde{C}'UT^{-1}dU'h)$$

$$+ \text{Tr}(\tilde{C}'UT^{-1}(B \circ (dU'U + U'dU))T^{-1}U'h).$$

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 $\text{Tr}(\tilde{C}'dC)$ as follows

$$\text{Tr}(\tilde{C}'dC) = \text{Tr}(\tilde{C}'(I - UT^{-1}U')dh)$$

$$- \text{Tr}(T^{-1}U'h\tilde{C}'dU) - \text{Tr}(h\tilde{C}'UT^{-1}dU')$$

$$+ \text{Tr}(T^{-1}U'h\tilde{C}'UT^{-1}(B \circ (dU'U + U'dU))).$$

To simplify the expression, we will use the short notations

$$\hat{C} = (T')^{-1}U'\tilde{C};$$

$$\hat{h} = T^{-1}U'h,$$

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

$$\text{Tr}(\tilde{C}'dC) = \text{Tr}((\tilde{C}' - \hat{C}'U')dh)$$

$$- \text{Tr}(\hat{h}\tilde{C}'dU) - \text{Tr}(h\hat{C}'dU')$$

$$+ \text{Tr}(\hat{h}\hat{C}'(B \circ (dU'U + U'dU))).$$

\hfill $\square$
Now using the two following identities of the trace

\[ \text{Tr}(A') = \text{Tr}(A), \]
\[ \text{Tr}(A(B \circ C)) = \text{Tr}((A \circ B')C)), \]

we can rewrite \( \text{Tr}(\tilde{C}'dC) \) as follows

\[ \text{Tr}(\tilde{C}'dC) = \text{Tr}((\tilde{C} - \tilde{C}'U')dh) \]
\[ - \text{Tr}(\tilde{h}\tilde{C}'dU) - \text{Tr}(\tilde{h}\tilde{C}'dU') \]
\[ + \text{Tr}(\tilde{h}\tilde{C}' \circ B')dU'U) \]
\[ + \text{Tr}(\tilde{h}\tilde{C}' \circ B')dU'dU). \]

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

\[ \text{Tr}(\tilde{C}'dC) = \text{Tr}((\tilde{C} - \tilde{C}'U')dh) \]
\[ - \text{Tr}(\tilde{h}\tilde{C}'dU) - \text{Tr}(\tilde{C}'h'dU) \]
\[ + \text{Tr}((\tilde{C}\tilde{h}' \circ B)U'dU) \]
\[ + \text{Tr}((\tilde{h}\tilde{C}' \circ B')dU'dU). \]

Using lemma 1 we conclude that

\[ \tilde{U} = U \left[ \tilde{h}\tilde{C}' \circ B' + (\tilde{C}\tilde{h}') \circ B \right] - \tilde{C}\tilde{h}' - h\tilde{C}', \]
\[ \tilde{h} = \tilde{C} = U\tilde{C}. \]

Sketch of the proof for Corollary 1. For any nonzero complex valued vector \( x \in \mathbb{C}^n \), if we chose \( u = x + e^{i\theta}||x||e_1 \) 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)

\[ Hx = ||x||e_1 \]  
\[ (2) \]

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 \( U := (v_{i,j})_{1 \leq i \leq n} \). Then the element of the matrix \( T := \text{striu}(U') + \frac{1}{2} \text{diag}(U') \) 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 \( ||x|| \) 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 \( \tilde{C} = (T')^{-1}U' \frac{\partial C}{\partial C} \). This is equivalent to solving the triangular systems of equations \( T\tilde{h} = U'h \) and \( T'\tilde{C} = U' \frac{\partial C}{\partial C} \).

**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

\[ t_{k,k} \tilde{h}_k + \sum_{j=k+1}^{m} t_{k,j} \tilde{h}_j = \sum_{j=k}^{n} v_{k,j} h_j, \]
\[ = \sum_{j=k}^{n} v_{j,k} h_j, \]
\[ = \sum_{r=k}^{m} v_{r,k} h_r, \]
\[ = \sum_{j=k}^{m} v_{r,j} h_j - \sum_{r=k}^{m} v_{r,k} \sum_{j=k}^{r} v_{r,j} h_j, \]
\[ = U'_{*,k}(h - \sum_{j=k+1}^{m} U_{*,j} \tilde{h}_j), \]
\[ (3) \]

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

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

\[ \tilde{h}_k = \frac{2}{U'_{*,k} U_{*,k}} U'_{*,k}H_{*,k+1}, \]
\[ (5) \]
\[ H_{*,k} = H_{*,k+1} - \tilde{h}_k U_{*,k}. \]
\[ (6) \]

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** \( T'\tilde{C} = U' \frac{\partial C}{\partial C} \). Similarly to the previous case, we have for \( 1 \leq k \leq m \)

\[ t_{k,k} \tilde{C}_k + \sum_{j=1}^{k-1} t_{j,k} \tilde{C}_j = \sum_{j=k}^{n} v_{k,j} \frac{\partial C}{\partial C}_j, \]
\[ = \sum_{j=1}^{n} v_{j,k} \frac{\partial C}{\partial C}_j - \sum_{r=1}^{k-1} \sum_{j=r}^{n} v_{r,j} v_{r,k} \tilde{C}_j, \]
\[ = \sum_{r=1}^{n} v_{r,k} \frac{\partial C}{\partial C}_r - \sum_{r=1}^{n} v_{r,k} \sum_{j=1}^{k-1} v_{r,j} \tilde{C}_j, \]
\[ (8) \]
\[ = U'_{*,k} \frac{\partial C}{\partial C} - \sum_{j=1}^{k-1} U_{*,j} \tilde{C}_j, \]
where the passage from Equation (7) to (8) is justified by the fact that \( \sum_{r=k}^{n} v_{r,k} v_{r,k} C_{j} = \sum_{r=1}^{n} v_{r,k} v_{r,k} C_{j} \) (since \( v_{r,k} = 0 \) for \( r < k \)).

By setting \( g = \frac{\partial C}{\partial U} - \sum_{j=1}^{b-1} U_{j} \tilde{C}_{j} \), we can write \( \tilde{C}_{k} = \frac{2}{U_{k} U_{k}^{T}} U_{k} g \) 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 \( g = \frac{\partial C}{\partial U} - \sum_{j=1}^{m} U_{j} \tilde{C}_{j} = \frac{\partial C}{\partial h} - U \tilde{C} = \frac{\partial C}{\partial h} \). This explains line 17 of Algorithm 1.

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

\[
\left[ \frac{\partial L}{\partial U} \right]_{i,k} = -\left[ \frac{\partial L}{\partial C} \right]_{i} \tilde{h}_{k} - h_{i} \tilde{C}_{k} + \sum_{j=1}^{m} v_{i,j} \left( \left( \left( \tilde{h} C' \right) o B' \right)_{j,k} + \left( \left( \tilde{C} h' \right) o B \right)_{j,k} \right),
\]

\[
= -\left[ \frac{\partial L}{\partial C} \right]_{i} \tilde{h}_{k} - h_{i} \tilde{C}_{k} + \sum_{j=1}^{m} v_{i,j} \left( \tilde{h}_{j} \tilde{C}_{k} \frac{[k \leq j]}{1 + \delta_{j,k}} + \tilde{C}_{j} \tilde{h}_{k} \frac{[j \leq k]}{1 + \delta_{j,k}} \right),
\]

\[
= -\left[ \frac{\partial L}{\partial C} \right]_{i} \tilde{h}_{k} - h_{i} \tilde{C}_{k} + \sum_{j=1}^{m} v_{i,j} \left( \tilde{h}_{j} \tilde{C}_{k} \frac{[k \leq j]}{1 + \delta_{j,k}} + \tilde{C}_{j} \tilde{h}_{k} \frac{[j \leq k]}{1 + \delta_{j,k}} \right),
\]

\[
= \tilde{C}_{k} \left( \sum_{j=1}^{m} v_{i,j} \tilde{h}_{j} - h_{i} \right)
+ \tilde{h}_{k} \left( \sum_{j=1}^{m} v_{i,j} \tilde{C}_{j} - \left[ \frac{\partial L}{\partial C} \right]_{i} \right).
\]

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

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

where \( g = \frac{\partial C}{\partial h} - \sum_{j=1}^{b-1} \tilde{C}_{j} U_{*,j} \). 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

```matlab
% 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(:,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+1) = 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(:,m-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 \( C^{(t)} \), \( \frac{\partial C}{\partial h^{(t-1)}} \) (variable \( g \) is the code), and \( \frac{\partial C}{\partial U^{(t)}} \) (variable \( g \) is the code). The required inputs for the FP and BP are, respectively, the tuples \( (U, h^{(t-1)}) \) and \( (U, C^{(t)}, \frac{\partial C}{\partial h^{(t-1)}}) \).

Note that \( \frac{\partial C}{\partial U^{(t)}} \) is variable BP\( g \) in the Matlab code.

### References

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