This document pertians to the function `'trainBust'`

# 1 definitions

For the purposes to follow, `'trainBust'`has the following properties:

1. Performs 1 step of an implicit-qr algorithm (using householder projections) on a Hessenberg matrix $H$
2. extracts a series of $m$ numbers upon completion, to be referred to as outputs.
3. takes an input of $s$ shifts to use in the implicit-qr algorithm. to be referred to as inputs.
4. capable of taking an input of size $s \times d$ to represent $d$ seeds for automatic differentiation
5. capable of returning the resulting $d \times 2m$ outputs of the automatic differentiation

For those interested, the extraction of the $2 \cdot m$ numbers is performed by performing a schur decomposition of a submatrix embedded within the larger matrix. This schur decomposition results in a spike (or 2 spikes if perfomed in the middle), the values of which are the $m$ numbers.

In the framework considering the shifts as inputs and the spikes as outputs, we can view this algorith as performing the following transformation

$$T : \mathbb{C}^s \to \mathbb{C}^m$$

Let the shifts be $\vec{\sigma} \in \mathbb{C}^s$ and the spikes be $\vec{m} \in \mathbb{C}^m$. Then we can define the Jacobian as

$$J \in \mathbb{C}^{s \times m} J_{i,j} = \frac{\partial T_i}{\partial \vec{\sigma}_j}$$

Where $T_i$ is the $i^{th}$ output of the function $T$.

If we provide a seed $\vec{\delta} \in \mathbb{C}^s$ to the automatic differentiation provided in `'trainBust'`, the end result is equivalent to

$$\vec{\delta} \cdot J = [\vec{\delta} \cdot \bigtriangledown T_i]$$

Where the surrounding square brackets indicate that this is a vector over the free indices (in this case $i$).

# 2 Automatic Differentiation Performance

Note that

$$\vec{\delta} \cdot J \Big|_{\vec{\sigma}} \approx \left[ \frac{T_i(\vec{\sigma} + \varepsilon\vec{\delta}) - T_i(\vec{\sigma} - \varepsilon\vec{\delta})}{2\varepsilon\|\vec{\delta}\|} \right]$$

for $\varepsilon > 0$.

The following results are based upon using $H \in \mathbb{R}^{100 \times 100}$, $H$ hessenberg, chosen by running a hessenberg decomposition on a matrix chosen from randomly from a normal distribution, $\vec{\sigma}$ chosen from a random normal distribution (multiplied by 100), $\vec{\delta}$ was chosen from a uniform distribution and then normalized, and $s = 8$ ($\varepsilon$ is specified below).

Before considering the function $T$, consider the following two processes:

1. `'trainBust'`after it has pushed the bulges through the matrix, but before performing a schur decomposition to create spikes

2. the schur decomposition and its derivative.

Let $B$ be the output of item (1), so $B = B(\vec{\sigma})$. Furthermore, let $\dot{B}$ represent the derivative of $B$ with respect to the normalized direction $\vec{\delta}$. Then

$$\dot{B} \approx \frac{B(\vec{\sigma} + \varepsilon\vec{\delta}) - B(\vec{\sigma})}{\varepsilon}$$

The Schur decomposition is $A = Q'TQ$, where the outputs are $Q$ and $T$. For the direction $\Delta$, we have that the schur decomposition is $A + \varepsilon\Delta = Q_\varepsilon T_\varepsilon Q_\varepsilon$. The `schurAD` fuction calculates $\dot{Q}$ and $\dot{T}$ given $\Delta$. Thus we have that

$$A + \varepsilon\Delta \approx (Q + \dot{Q})(T + \dot{T})(Q + \dot{Q})'$$

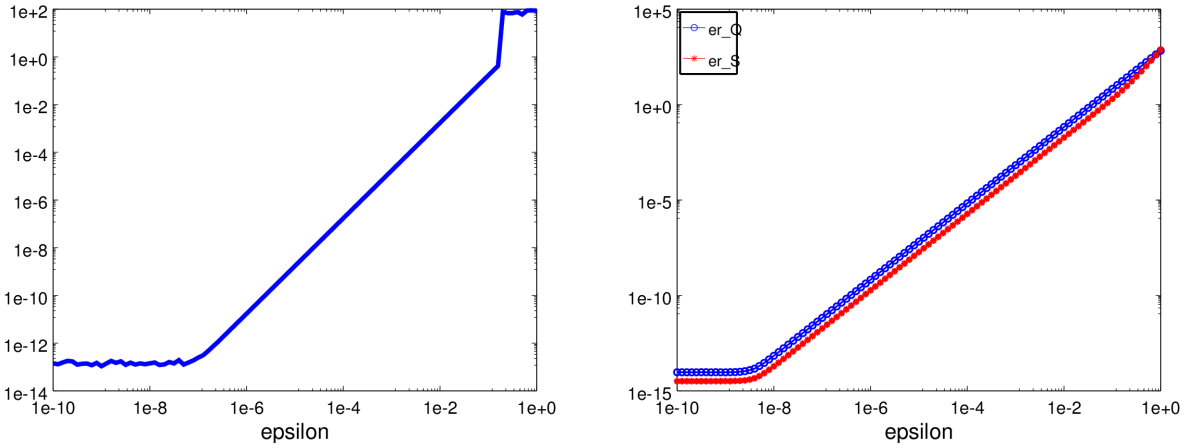
Thus treating the automatic differentiation result as the exact derivative, the above finite difference approximations should yield accuracy of $O(\varepsilon)$. The following equations are used to caculate the error:

$$er_F(\varepsilon) = \frac{\|B(\vec{\sigma} + \varepsilon\vec{\delta}) - (B(\vec{\sigma}) + \varepsilon \cdot \dot{B})\|_F}{\|B(\vec{\sigma})\|_F}$$

$$er_S(\varepsilon) = \frac{\|(Q + \varepsilon\dot{Q})(T + \varepsilon\dot{T})(Q + \varepsilon\dot{Q})' - (A + \varepsilon\Delta)\|_F}{\|A\|_F}$$

$$er_Q(\varepsilon) = \|(Q + \varepsilon\dot{Q})(Q + \varepsilon\dot{Q})' - I\|_F$$

Figure 1 shows that the errors are indeed of $O(\varepsilon^2)$ (thus accurate to first order), indicating that the AD approximation does indeed yield the derivative exact derivative which the finite difference is approximating.




(a) $er_F(\varepsilon)$ (b) $er_S(\varepsilon)$ and $er_Q(\varepsilon)$

Figure 1: Finite difference errors

# 3 Completing the process

Figure 2 shows $er_F$ applied after the schur decomposition has been used to eliminate the blocks, for 2 different matrices. The matrices used 10 shifts and were size 100.

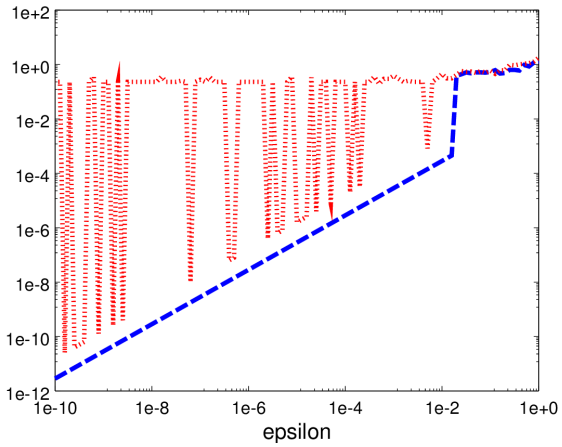



Figure 2: $er_F(\varepsilon)$ applied to completed process

Clearly some matrices break the automatic differentiation as applied/interpreted. This is due to the pivoting (which can be highly sensitive to perturbations) during the Schur decomposition, and can be alleviated as follows.

Given a Schur Decomposition $AQ = QT$, let $P_i$ be signed permutation matrices. Then

$$(P_r' A P_r)(P_r' Q P_c) = (P_r' Q P_c)(P_c T P_c)$$

, [1] which amounts to pivoting on $Q$. Since the spikes are generated only by the first and last rows of $Q$, this pivoting will effect which values appear in in the spiked matrix. Thus the poor performance observed in figure 2 can be attributed towards different pivoting positions.

To remedy this, we can apply pivoting after the schur decomposition to recover a uniform pivoting scheme across different choices of shifts. Since we are interested in stabilizing $Q$, we can view $P_c$ as permuting the columns of $Q$, and $P_r$ as permuting the rows of $Q$.

Our desire is that most of the 'weight' along the spikes are in less than half the entries, which provides for deflation opportunities. Note that the spikes correspond to scaled versions of the top and bottom rows of $Q$. To find the rows of $Q$ which best achieve this, we can find the 2 rows with the smallest $L_1$ norm, [2] and rotate them to the top and bottom rows of $Q$. The rest of $P_r$ can be left as the identity since the non-spike rows are of no interest.

For convenience, it would be nice if all the zeros are on the tips of the spikes rather than near the diagonal. Thus ideally all the zeros in the first row are in the last columns, and all the zeros in the last row are in the first columns. This can be achieved by having $P_c$ sort the columns in descending order by the metric $(|Q(n,i)| - |Q(1,i)|)$. Furthermore, to avoid sign changes, $P_c$ should also flip all the signs of the first row to positive.

[1] Note that this also applies to arbitrary rotations. We will only concern ourselves with signed permutations, as algorithms to compute the Schur decomposition often rely upon pivoting, but non-pivot rotations are boiled into $Q$

[2] This can be seen by partitioning the row $Q_{i,:}$ into two parts, and considering it as a vector in $\mathbb{C}^2$. Then the $L_1$ norm is minimized across unit vectors with only 1 nonzero entry.