Student Learning Targets, and Objectives

Target Numerical Indications of (Backward) Stable Algorithms
Objective Explain how forward errors are impacted by the condition number
Objective Explain how the size of backward errors may indicated backward stability

Target Backward Stable Solution Strategies
Objective Be able to show backward stability of a solution strategy using backward stable algorithmic building blocks

Reference: Key Floating Point Axioms

Axiom (Floating Point Representation)
\[ \forall x \in \mathbb{R}, \text{there exists } \varepsilon \text{ with } |\varepsilon| \leq \varepsilon_{\text{mach}}, \text{such that } \text{fl}(x) = x(1 + \varepsilon). \]

Axiom (The Fundamental Axiom of Floating Point Arithmetic)
For all \( x, y \in \mathbb{F}_n \) (where \( \mathbb{F}_n \) is the set of \( n \)-bit floating point numbers), there exists \( \varepsilon \) with \( |\varepsilon| \leq \varepsilon_{\text{mach}}(\mathbb{F}_n) \), such that
\[
\begin{align*}
x \oplus y &= (x + y)(1 + \varepsilon), & x \ominus y &= (x - y)(1 + \varepsilon), \\
x \odot y &= (x \cdot y)(1 + \varepsilon), & x \oslash y &= (x/y)(1 + \varepsilon)
\end{align*}
\]
Reference: Key Stability Definitions

**Definition (Stable Algorithm)**

We say that \( \tilde{f} \) is a **stable algorithm** if \( \forall \mathbf{x} \in X \)

\[
\frac{\| \tilde{f}(\mathbf{x}) - f(\mathbf{x}) \|}{\| f(\mathbf{x}) \|} = O(\varepsilon_{\text{mach}}),
\]

for some \( \tilde{\mathbf{x}} \) with

\[
\frac{\| \tilde{\mathbf{x}} - \mathbf{x} \|}{\| \mathbf{x} \|} = O(\varepsilon_{\text{mach}}).
\]

“A stable algorithm gives approximately the right answer, to approximately the right question.”

Reference: Accuracy — The Goal!

**Definition (Accuracy)**

We say that the algorithm \( \tilde{f} \) is **accurate** if \( \forall \mathbf{x} \in X \)

\[
\frac{\| \tilde{f}(\mathbf{x}) - f(\mathbf{x}) \|}{\| f(\mathbf{x}) \|} = O(\varepsilon_{\text{mach}}).
\]

This is what we want to do — write algorithms that **accurately** solve problems!

Last time, we finally tied the inherent difficulty of the problem, the **conditioning**, and the quality of the algorithm, the **stability** together in a theorem —

Reference: Key Stability Definitions

**Definition (Backward Stable Algorithm)**

An algorithm \( \tilde{f} \) is **backward stable** if \( \forall \mathbf{x} \in X \)

\[
\tilde{f}(\mathbf{x}) = f(\mathbf{x}),
\]

for some \( \tilde{\mathbf{x}} \) with

\[
\frac{\| \tilde{\mathbf{x}} - \mathbf{x} \|}{\| \mathbf{x} \|} = O(\varepsilon_{\text{mach}}).
\]

“A backward stable algorithm gives exactly the right answer, to approximately the right question.”

Reference: Accuracy (stability, conditioning)

**Theorem (Computational Accuracy)**

Suppose a backward stable algorithm is applied to solve a problem \( f : X \to Y \) with condition number \( \kappa \) in a floating point environment satisfying the floating point representation axiom, and the fundamental axiom of floating point arithmetic.

Then the relative errors satisfy

\[
\frac{\| \tilde{f}(\mathbf{x}) - f(\mathbf{x}) \|}{\| f(\mathbf{x}) \|} = O(\kappa(\mathbf{x})\varepsilon_{\text{mach}}).
\]

**Recall:** The definition of the relative condition number

\[
\kappa(\mathbf{x}) = \sup_{\delta x} \left[ \frac{\| \delta f \|}{\| \delta x \|} \right] \frac{\| f(\mathbf{x}) \|}{\| \mathbf{x} \|}
\]

as the ratio of the relative (infinitesimal) change in \( f \) induced by an infinitesimal change in \( \mathbf{x} \).


Reference: Algorithm — Householder QR

Algorithm (Householder QR-Factorization, $Q^*b$ - Version)

1: for $k \in \{1, \ldots, n\}$ do
2: \[ \vec{x} \leftarrow A(k:m,k) \]
3: \[ \vec{v}_k \leftarrow \text{sign}(x_k) \| \vec{x} \|_2 \vec{e}_1 + \vec{x} \]
4: \[ \vec{v}_k \leftarrow \vec{v}_k / \| \vec{v}_k \|_2 \]
5: \[ A(k:m,k:n) \leftarrow A(k:m,k:n) - 2 \vec{v}_k (\vec{v}_k^* A(k:m,k:n)) \]
6: \[ \vec{b}(k:m) \leftarrow \vec{b}(k:m) - 2 \vec{v}_k (\vec{v}_k^* \vec{b}(k:m)) \] /* Compute $Q^*b$ */
7: end for

$A(k:m,k)$ Denotes the $k$th thru $m$th rows, in the $k$th column of $A$ — a vector quantity.

$A(k:m,k:n)$ Denotes the $k$th thru $m$th rows, in the $k$th thru $n$th columns of $A$ — a matrix quantity.

The Road Ahead: Stability of Algorithms

With our new toolbox in hand, we re-visit some of the algorithms previously discussed. This second look will reveal, in a more rigorous way, why the algorithms perform the way they do...

The **Householder Triangularization** method of computing the QR-factorization is a backward stable (HT-QR for short).

First, we look at some numerical experiments showcasing this; and then we combine HT-QR with other backward stable algorithmic fragments to build a stable solver for our fundamental problem

$$ A \vec{x} = \vec{b}. $$

![The non-zero pattern of the matrix $R$.](image)

We generate a matrix $A$ with known QR-factorization, and compute the Householder QR-factorization using

\[
\begin{align*}
R & \quad = \text{triu}(\text{randn}(64)); \\
Q, \sim & \quad = \text{qr}(\text{randn}(64)); \\
A & \quad = Q \cdot R; \\
Q2, R2 & \quad = \text{qr}(A);
\end{align*}
\]

It turns out that $Q_2$ and $R_2$ are quite far from $Q$ and $R$:

\[
\begin{align*}
\text{norm}(Q_2-Q) / \text{norm}(Q) & \quad = 0.003427 \\
\text{norm}(R_2-R) / \text{norm}(R) & \quad = 0.000440
\end{align*}
\]

It seems like a disaster has occurred, but

\[
\begin{align*}
\text{norm}(A-Q_2\cdot R_2) / \text{norm}(A) & \quad = 1.032309e-15
\end{align*}
\]

Now consider $Q_3$ and $R_3$

\[
\begin{align*}
Q_3 & \quad = Q + 1e-4 \cdot \text{randn}(64) \\
R_3 & \quad = R + 1e-4 \cdot \text{randn}(64) \\
\text{norm}(Q_3-Q) / \text{norm}(Q) & \quad = 0.001585 \\
\text{norm}(R_3-R) / \text{norm}(R) & \quad = 0.000129 \\
\text{norm}(A-Q_3\cdot R_3) / \text{norm}(A) & \quad = 1.086451e-03
\end{align*}
\]
The Moral of the Story

The errors in $Q_2$ and $R_2$ are known as forward errors. Large forward errors are the result of an ill-conditioned problem and/or an unstable algorithm. — In our example it is the former

$$\kappa(A) = \text{cond}(A) = 2.0223\times10^{16}.$$  

The error in the result of the matrix product $Q_2R_2$ is known as the backward error, or residual.

The fact that the backward error is small suggests that Householder Triangularization is backward stable.

Note: Due to the specific way the Householder reflections are performed, the algorithm above may have to be run a couple of times in order to produce (similar) results. A relative error in $Q_2$ of size $\sim 2$ indicates that the initial random $Q$ and $R$ could not possibly have come from a HT-QR algorithm (due to "sign-flips.")

Now, we define $\tilde{Q}$ to be the exactly unitary matrix

$$\tilde{Q} = \tilde{Q}_1\tilde{Q}_2\cdots\tilde{Q}_n,$$

this matrix will take the place of the computed $Q$ in our discussion.

This approach is natural since in general the matrix $Q$ is not formed explicitly, but rather used implicitly to get the action $Q^*\vec{b}$.

With these definitions, we are ready to state the theorem...

It turns out that HT-QR is backward stable for all matrices $A$ in any floating-point environment satisfying the floating point axioms.

The formal result takes the form

$$\tilde{Q}\tilde{R} = A + \delta A, \quad \delta A \text{ "small,}$$

where $\tilde{R}$ is the upper triangular matrix constructed by the HT-QR algorithm.

Since the HT-QR algorithm does not explicitly compute $\tilde{Q}$ (in the "fast mode," ) we must define what we mean by $\tilde{Q}$.

Let $\tilde{Q}_k$ denote the exactly unitary reflector defined by the floating point vector $\vec{v}_k$

$$\tilde{Q}_k = I - 2\tilde{v}_k\tilde{v}_k^*.$$

Solving $Ax = b$, Using Householder QR-Factorization

Computing the QR-factorization is not an end in itself. Usually it is one of the first steps in trying to solve a system of linear equations, a least squares problem, or an eigenvalue problem.

At this point we know that $HT-QR$ is backward stable, but is that enough?!? As we have seen, the individual factors $Q$ and $R$ may carry large forward errors.

The good news is that accuracy of the product $QR$ is sufficient for most purposes.

We consider the following algorithm for solving $Ax = b$

Algorithm (Solution of a Linear System, $Ax = b$)

1: $QR \leftarrow A$ — Compute the QR-factorization by $HT-QR$
2: $\tilde{y} \leftarrow Q^*b$ — Construct $Q^*b$ by $HT-QR$
3: $\tilde{x} \leftarrow R^{-1}\tilde{y}$ — Solve by back substitution

Proof: From step #2 and step #3 we have

$$(\tilde{Q} + \delta Q)\tilde{y} = b, \quad (\tilde{R} + \delta R)\tilde{x} = \tilde{y},$$

combining the two gives

$$b = (\tilde{Q} + \delta Q)(\tilde{R} + \delta R)\tilde{x} = \tilde{Q}\tilde{R} + (\delta Q)\tilde{R} + \tilde{Q}(\delta R) + (\delta Q)(\delta R)\tilde{x}.$$ 

Now, using the result for step #1

$$\tilde{Q}\tilde{R} = A + \delta A$$

we get

$$\tilde{b} = A + \delta A + (\delta Q)\tilde{R} + \tilde{Q}(\delta R) + (\delta Q)(\delta R)\tilde{x}.$$
Solving $A\bar{x} = \bar{b}$, Using Householder QR-Factorization

Next, we must show that the perturbation

$$\Delta A = \delta A + (\delta Q)\bar{R} + \bar{Q}(\delta R) + (\delta Q)(\delta R)$$

is small relative to $A$.

Since $\bar{Q}\bar{R} = A + \delta A$, and $\bar{Q}$ is unitary we have $\bar{R} = \bar{Q}^*(A + \delta A)$

$$\|\bar{R}\| \leq \|\bar{Q}^*\|\|A + \delta A\| = O(1), \quad \varepsilon_{mach} \to 0.$$ 

Hence, the relative size of the second term is bounded

$$\frac{\|(\delta Q)\bar{R}\|}{\|A\|} \leq \|(\delta Q)\| \frac{\|\bar{R}\|}{\|A\|} = O(\varepsilon_{mach}).$$

Finally, the fourth term

$$\Delta A = \delta A + (\delta Q)\bar{R} + \bar{Q}(\delta R) + (\delta Q)(\delta R)$$

$$\frac{\|(\delta Q)(\delta R)\|}{\|A\|} \leq \|(\delta Q)\| \frac{\|(\delta R)\|}{\|A\|}$$

We know

$$\|(\delta Q)\| = O(\varepsilon_{mach}), \quad \text{and} \quad \frac{\|(\delta R)\|}{\|A\|} = O(\varepsilon_{mach})$$

So,

$$\frac{\|(\delta Q)(\delta R)\|}{\|A\|} = O(\varepsilon_{mach}^2)$$

We collect our findings, and note that as required

$$\frac{\|\Delta A\|}{\|A\|} \leq \frac{\|\delta A\|}{\|A\|} + \frac{\|(\delta Q)\bar{R}\|}{\|A\|} + \frac{\|\bar{Q}(\delta R)\|}{\|A\|} + \frac{\|(\delta Q)(\delta R)\|}{\|A\|} = O(\varepsilon_{mach}).$$

This completes the proof. $\square$

If we combine this result with the accuracy theorem we showed last time, we get the following result about the accuracy of solutions of $A\bar{x} = \bar{b}$ using the Householder-Triangularization + Back-substitution algorithm:
Accuracy of the Solution to $A\tilde{x} = \tilde{b}$ using Householder QR and Back-substitution

The solution $\tilde{x}$ computed by the Householder-Triangularization + Back-substitution algorithm satisfies

$$\frac{\|\tilde{x} - \bar{x}\|}{\|\tilde{x}\|} = O(\kappa(A)\varepsilon_{mach})$$

Patching Some Holes...

We have left three major holes in the argument — the statement, without proof, that the individual steps are backward stable.

It is instructive to see at least one such proof from “scratch.” — Next, we turn our attention to the back-substitution algorithm.

Even though back substitution is one of the easiest problems of numerical linear algebra, the stability proof is quite lengthy... and provides the general structure / workflow for all such proofs.