Numerical Matrix Analysis

Notes #12 — Conditioning and Stability
Stability of Householder QR for $A\vec{x} = \vec{b}$

Peter Blomgren
⟨blomgren@sdsu.edu⟩

Department of Mathematics and Statistics
Dynamical Systems Group
Computational Sciences Research Center
San Diego State University
San Diego, CA 92182-7720

http://terminus.sdsu.edu/

Spring 2024
(Revised: March 5, 2024)

Outline

1. Student Learning Targets, and Objectives
   - SLOs: Stability of Householder QR for $A\vec{x} = \vec{b}$
2. Reference
   - Floating Point Axioms
   - Stability Definitions
   - Accuracy
   - Householder QR
3. Stability of Algorithms
   - Householder Triangularization: Numerical Experiment
   - Householder Triangularization: Backward Stability
4. Solving $A\vec{x} = \vec{b}$
   - Theorem — Householder-Triangularization + Back-Substitution
   - Three Major Holes to Patch...

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)

$\forall x \in \mathbb{R}$, there exists ε with $|\varepsilon| \leq \varepsilon_{\text{mach}}$, such that $\texttt{fl}(x) = x(1+\varepsilon)$.

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 ε with $|\varepsilon| \leq \varepsilon_{\text{mach}}(\mathbb{F}_n)$, such that

$$\begin{aligned} x \oplus y &= (x+y)(1+\varepsilon), & x \ominus y &= (x-y)(1+\varepsilon), \\ x \otimes y &= (x*y)(1+\varepsilon), & x \oslash y &= (x/y)(1+\varepsilon) \end{aligned}$$

Jump to: accuracy theorem, HT-QR stability

Reference: Key Stability Definitions

Definition (Stable Algorithm)

We say that $\tilde{f}$ is a **stable algorithm** if $\forall \vec{x} \in X$

$$\frac{\|\tilde{f}(\vec{x}) - f(\tilde{\vec{x}})\|}{\|f(\tilde{\vec{x}})\|} = \mathcal{O}(\varepsilon_{\text{mach}}),$$

for some $\tilde{\vec{x}}$ with

$$\frac{\|\tilde{\vec{x}} - \vec{x}\|}{\|\vec{x}\|} = \mathcal{O}(\varepsilon_{\text{mach}}).$$

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

Reference: Key Stability Definitions

Definition (Backward Stable Algorithm)

An algorithm $\tilde{f}$ is **backward stable** if $\forall \vec{x} \in X$

$$\tilde{f}(\vec{x}) = f(\tilde{\vec{x}}),$$

for some $\tilde{\vec{x}}$ with

$$\frac{\|\tilde{\vec{x}} - \vec{x}\|}{\|\vec{x}\|} = \mathcal{O}(\varepsilon_{\text{mach}}).$$

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

Jump to: accuracy theorem.

Reference: Accuracy — The Goal!

Definition (Accuracy)

We say that the algorithm $\tilde{f}$ is **accurate** if $\forall \vec{x} \in X$

$$\frac{\|\tilde{f}(\vec{x}) - f(\vec{x})\|}{\|f(\vec{x})\|} = \mathcal{O}(\varepsilon_{\text{mach}}).$$

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 —

Last Time: Accuracy(stability,conditioning)

Theorem (Computational Accuracy)

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

Then the relative errors satisfy

$$\frac{\|\tilde{f}(x) - f(x)\|}{\|f(x)\|} = \mathcal{O}(\kappa(x)\varepsilon_{mach}).$$

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

$$\kappa(\vec{x}) = \sup_{\delta\vec{x}} \left[\frac{\|\delta f\|}{\|f(\vec{x})\|} \Big/ \frac{\|\delta\vec{x}\|}{\|\vec{x}\|} \right]$$

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

Reference: Algorithm — Householder QR

From [Lecture#7]

Algorithm (Householder QR-Factorization, $Q^*\vec{b}$- Version)

for $k \in \{1, \ldots, n\}$ **do**
  $\vec{x} \leftarrow A(\texttt{k:m,k})$
  $\vec{v}_k \leftarrow \text{sign}(x_1)\|\vec{x}\|_2\vec{e}_1 + \vec{x}$
  $\vec{v}_k \leftarrow \vec{v}_k / \|\vec{v}_k\|_2$
  $A(\texttt{k:m,k:n}) \leftarrow A(\texttt{k:m,k:n}) - 2\vec{v}_k(\vec{v}_k^* A(\texttt{k:m,k:n}))$
  $\vec{b}(\texttt{k:m}) \leftarrow \vec{b}(\texttt{k:m}) - 2\vec{v}_k(\vec{v}_k^*\vec{b}(\texttt{k:m}))$ /* Compute $Q^*\vec{b}$ */
end for

$A(\texttt{k:m,k})$ — Denotes the kth thru mth rows, in the kth column of A — a vector quantity.

$A(\texttt{k:m,k:n})$ — Denotes the kth thru mth rows, in the kth thru nth 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

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

Householder Triangularization: Numerics

1 of 3

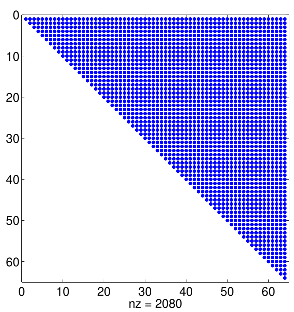

Figure: The non-zero pattern of the matrix R.

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

```
R        =  triu(randn(64));      || R       =  np.triu(np.random.rand(64,64))
[Q,~]    =  qr(randn(64));        || Q,_     =  np.linalg.qr(np.random.rand(64,64))
A        =  Q*R;                  || A       =  np.matmul(Q,R)
[Q2,R2]  =  qr(A);                || Q2, R2  =  np.linalg.qr(A)
```


Householder Triangularization: Numerics

2 of 3

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

```
norm(Q2-Q) / norm(Q)   =   0.003427
norm(R2-R) / norm(R)   =   0.000440
```

It seems like a disaster has occurred, but

```
norm(A-Q2*R2) / norm(A)   =   1.032309e-15
```

Now consider Q_3 and R_3

```
Q3                        =   Q + 1e-4*randn(64)
R3                        =   R + 1e-4*randn(64)
norm(Q3-Q) / norm(Q)      =   0.001595
norm(R3-R) / norm(R)      =   0.000129
norm(A-Q3*R3) / norm(A)   =   1.065451e-03
```

Householder Triangularization: Numerics — 3 of 3

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

$\kappa(A) = \texttt{cond(A)} = 2.0223\text{e+}16.$ `np.linalg.cond`

The error in the result of the matrix product $Q_2 R_2$ 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 ~ 2 indicates that the initial random Q and R could not possibly have come from a HT-QR algorithm (due to "sign-flips.")

Householder Triangularization: Backward Stability — 1 of 3

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

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

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 $\tilde{\mathbf{v}}_k$

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

Householder Triangularization: Backward Stability — 2 of 3

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...

Householder Triangularization: Backward Stability — 3 of 3

Theorem (Backward Stability of Householder QR)

Let the QR-factorization $A = QR$ of a matrix $A \in \mathbb{C}^{m\times n}$ be computed by Householder triangularization in a floating-point environment satisfying the floating-point axioms, and let the computed factors $\tilde{Q}$ and $\tilde{R}$ be as discussed on the previous two slides. Then we have

$$\tilde{Q}\tilde{R} = A + \delta A, \quad \frac{\|\delta A\|}{\|A\|} = \mathcal{O}(\varepsilon_{mach})$$

for some $\delta A \in \mathbb{C}^{m\times n}$.

The full proof can be found in: —

Nicholas J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed., ISBN 0-89871-521-0, SIAM, Philadelphia, 2002. (pp. 357–361)

Solving $A\vec{x} = \vec{b}$, 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 $\tilde{Q}$ and $\tilde{R}$ may carry large forward errors.

The **good news** is that accuracy of the product $\tilde{Q}\tilde{R}$ is sufficient for most purposes.

We consider the following algorithm for solving $A\vec{x} = \vec{b}$

Algorithm (Solution of a Linear System, $A\vec{x} = \vec{b}$)

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

Solving $A\vec{x} = \vec{b}$, Using Householder QR-Factorization

It turns out that this algorithm is backward stable. The three steps are backward stable. For now we state these results without proof, and then combine them to form the larger result.

We have already expressed the backward stability of HT-QR in a previous theorem.

The second step computes $\tilde{Q}^*\vec{b}$, due to floating-point errors, the result $\tilde{y}$ is not equal to $\vec{y} = \tilde{Q}^*\vec{b}$, but the operation is backward stable

$$(\tilde{Q} + \delta Q)\tilde{y} = \vec{b}, \quad \|\delta Q\| = \mathcal{O}(\varepsilon_{\text{mach}}).$$

The solution $\tilde{x}$ of the back substitution in the third step satisfies

$$(\tilde{R} + \delta R)\tilde{x} = \tilde{y}, \quad \frac{\|\delta R\|}{\|\tilde{R}\|} = \mathcal{O}(\varepsilon_{\text{mach}}).$$

Solving $A\vec{x} = \vec{b}$, Using Householder QR-Factorization

With these unproven (for now) building blocks, we are ready to state and prove the following theorem

Theorem

The three step algorithm described above for solving $A\vec{x} = \vec{b}$ is backward stable, satisfying

$$(A + \Delta A)\tilde{x} = \vec{b}, \quad \frac{\|\Delta A\|}{\|A\|} = \mathcal{O}(\varepsilon_{mach}),$$

for some $\Delta A \in \mathbb{C}^{m\times m}$.

Solving $A\vec{x} = \vec{b}$, Using Householder QR-Factorization

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

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

combining the two gives

$$\vec{b} = (\tilde{Q} + \delta Q)(\tilde{R} + \delta R)\tilde{x} = \left[\tilde{Q}\tilde{R} + (\delta Q)\tilde{R} + \tilde{Q}(\delta R) + (\delta Q)(\delta R)\right]\tilde{x}.$$

Now, using the result for step #1

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

we get

$$\vec{b} = \left[A + \underbrace{\delta A + (\delta Q)\tilde{R} + \tilde{Q}(\delta R) + (\delta Q)(\delta R)}_{\Delta A}\right]\tilde{x}.$$

Solving $A\vec{x} = \vec{b}$, Using Householder QR-Factorization

Next, we must show that the perturbation

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

is small relative to A.

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

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

Hence, the relative size of the second term is bounded

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

Solving $A\vec{x} = \vec{b}$, Using Householder QR-Factorization

Now, consider the third term

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

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

Since

$$\|\tilde{Q}\| = \mathcal{O}(1), \quad \frac{\|(\delta R)\|}{\|\tilde{R}\|} = \mathcal{O}(\varepsilon_{\text{mach}}), \quad \text{and} \quad \frac{\|\tilde{R}\|}{\|A\|} = \mathcal{O}(1),$$

we have

$$\frac{\|\tilde{Q}(\delta R)\|}{\|A\|} = \mathcal{O}(\varepsilon_{\text{mach}}).$$

Solving $A\vec{x} = \vec{b}$, Using Householder QR-Factorization

Finally, the fourth term

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

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

We know

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

So,

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

Solving $A\vec{x} = \vec{b}$, Using Householder QR-Factorization

We collect our findings, and note that as required

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

This completes the proof. □

If we combine this result with the accuracy theorem we showed last time, we get the following result about the **accuracy** of solutions of $A\vec{x} = \vec{b}$ using the *Householder-Triangularization + Back-substitution algorithm:*

Accuracy of the Solution to $A\vec{x} = \vec{b}$ using Householder QR and Back-substitution

Theorem (Accuracy of the Solution to $A\vec{x} = \vec{b}$ using Householder QR-factorization and Back-substitution)

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

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

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 `step-3`, *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. $\rightsquigarrow$ That will be our next order of business in [Lecture#13].

* For `step-1` (the QR-factorization), we have "proof by reference."