Lecture 16 - *Stability of Householder Triangularization*

OBJECTIVE:
To see backward error analysis in action!
In particular, we observe backward stability of Householder triangularization via a Matlab experiment.
This step can then be combined with other backward stable pieces to obtain a backward stable algorithm for solving $\mathbf{Ax} = \mathbf{b}$.

$\diamond$ EXPERIMENT

Householder factorization is a backward stable algorithm for computing QR factorizations.

We can illustrate this with the following Matlab experiment (in double precision; $\epsilon_{\text{machine}} = 1.11 \times 10^{-16}$).

```
R=triu(randn(50));     % Set R to a 50x50 upper-triangular matrix
                       % with normal random entries
[Q,X]=qr(randn(50));   % Set Q to a 50x50 random orthogonal matrix
                       % by orthogonalizing a random matrix
A=Q*R;                 % Set A=QR, up to rounding errors
[Q2,R2]=qr(A);         % Compute QR factorization by Householder
```

$\rightarrow$ We can now compare $\mathbf{A} = \mathbf{QR}$ to $\tilde{\mathbf{A}} = \mathbf{Q}_2\mathbf{R}_2$.

$\mathbf{Q}_2, \mathbf{R}_2$ are far from exact!

```
norm(Q2-Q)             % How accurate is Q2?
     ans = 0.00889
norm(R2-R)
     ans = 0.00071     % How accurate is R2?
```

These errors are huge! (Of course, do not take them literally - only order of magnitude is sensible.)

$\rightarrow$ Our calculations have been done with 16 digits of accuracy, yet the final result are only accurate to 2 or 3 digits.

i.e., rounding errors have been amplified by $\sim 10^{13}$!

Note 1. *You may need to renormalize the columns of* $\mathbf{Q}$ *and rows of* $\mathbf{R}$ *by* ± 1 *if* $\mathbf{Q}_2$ *differs a lot from* $\mathbf{Q}$ *because of incompatible signs.*

Although we seem to have lost about 12 digits of accuracy in the individual components, an astonishing thing happens when we multiply these inaccurate matrices.

```
norm(A-Q2*R2)/norm(A)  % How accurate is Q2R2?
     ans = 1.432e-15
```

So the product $\mathbf{Q}_2\mathbf{R}_2$ is accurate to a full 15 digits!
$\rightarrow$ the errors in $\mathbf{Q}_2$ and $\mathbf{R}_2$ must be *diabolically correlated* (as Wilkinson used to say).

To emphasize this, consider two other matrices $\mathbf{Q}_3, \mathbf{R}_3$ that are about as accurate as $\mathbf{Q}_2, \mathbf{R}_2$ (or even a little more!)

```
Q3=Q+1e-4*randn(50);   % Set Q3 to a perturbation of Q
                       % Note that Q3 is closer to Q than Q2 is!
R3=R+1e-4*randn(50);   % Set R3 to a perturbation of R
                       % Again, R3 is closer to R than R2 is!
norm(A-Q3*R3)/norm(A)  % How accurate is Q3R3?
     ans = 0.00088
```

Now the error in $\mathbf{Q}_3\mathbf{R}_3$ is huge!

Note 2. *We did not make* $\mathbf{R}_3$ *upper triangular or* $\mathbf{Q}_3$ *orthogonal; but this would not have materially affected the result.*

The errors in $\mathbf{Q}_2$ and $\mathbf{R}_2$ are *forward errors*.

Large forward errors usually happen because a problem is ill-conditioned, or the solution algorithm is unstable (here we have a ill-conditioned problem; recall Gram-Schmidt).

The error in $\mathbf{Q}_2\mathbf{R}_2$ is the *backward error* or *residual*. Because this is small, we expect Householder triangularization to be backward stable.

$\diamond$ THEORETICAL RESULT

Householder triangularization is backward stable for all matrices $\mathbf{A}$:

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

where $\delta\mathbf{A}$ is "small".

i.e., $\tilde{\mathbf{Q}}\tilde{\mathbf{R}}$ is the <u>exact</u> $\mathbf{QR}$ factorization of a matrix that is "close" to the one whose $\mathbf{QR}$ factorization we want.

$\tilde{\mathbf{R}}$ is an upper-triangular matrix constructed by Householder triangularization but $\tilde{\mathbf{Q}}$ is an *exactly orthogonal* matrix.

Recall

$$\mathbf{Q} = \mathbf{Q}_1\mathbf{Q}_2 \ldots \mathbf{Q}_n,$$

where $\mathbf{Q}_k$ is defined by the vector $\mathbf{v}_k$ in the k^{th} step of Algorithm 10.1.

In the actual computation, we obtain a sequence of floating-point vectors $\tilde{\mathbf{v}}_k$.

Let $\tilde{\mathbf{Q}}_k$ denote the *exactly orthogonal* matrix defined by $\tilde{\mathbf{v}}_k$ and define

$$\tilde{\mathbf{Q}} = \tilde{\mathbf{Q}}_1\tilde{\mathbf{Q}}_2 \ldots \tilde{\mathbf{Q}}_n$$

This exactly orthogonal matrix will be our "computed $\mathbf{Q}$".

It makes sense to do this because in practice $\mathbf{Q}$ is never computed anyway
$\rightarrow$ only the $\tilde{\mathbf{v}}_k$ are.

Here is the formal statement of the theorem that explains what we saw in our Matlab experiment.

Theorem 1. *Let* $\mathbf{A} \in \mathbb{R}^{m \times n}$ *have the factorization* $\mathbf{A} = \mathbf{QR}$ *computed by Householder triangularization. Let* $\tilde{\mathbf{Q}}, \tilde{\mathbf{R}}$ *be the computed factors as described above. Then*

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

where

$$\frac{\|\delta\mathbf{A}\|}{\|\mathbf{A}\|} = \mathcal{O}(\epsilon_{\mathrm{machine}})$$

for some $\delta\mathbf{A} \in \mathbb{R}^{m \times n}$*.*

Recall: This holds uniformly for all matrices $\mathbf{A}$ as $\epsilon_{\mathrm{machine}} \to 0$ for any *fixed* dimensions m and n.
It cannot be viewed as being true as a function of m and n.

◇ ANALYSIS OF AN ALGORITHM TO SOLVE $\mathbf{Ax} = \mathbf{b}$

Householder triangularization is backward stable but not always forward accurate (it turns out this is true of most matrix factorizations in numerical linear algebra).

Now, $\mathbf{QR}$ is usually not an end in itself.

Often we use $\mathbf{QR}$ along the way to solve other problems such as linear systems, least-squares, or eigenvalue problems.

Happily, the accuracy of the product $\mathbf{QR}$ (i.e., the backward stability property) is usually enough for practical purposes
$\longrightarrow$ the lack of accuracy of the individual $\mathbf{Q}$ and $\mathbf{R}$ matrices is not problematic.

Consider solving a linear system $\mathbf{Ax} = \mathbf{b}$ by means of a $\mathbf{QR}$ factorization via Householder triangularization (this idea was first introduced in Lecture 7).

ALGORITHM 16.1: HOUSEHOLDER QR FACTORIZATION

$\mathbf{QR} = \mathbf{A}$	Factor $\mathbf{A}$ into $\mathbf{QR}$ by Algorithm 10.1, with $\mathbf{Q}$ represented as the product of reflectors
$\mathbf{y} = \mathbf{Q}^T\mathbf{b}$	Construct $\mathbf{Q}^T\mathbf{b}$ by Algorithm 10.2
$\mathbf{x} = \mathbf{R}^{-1}\mathbf{y}$	Solve the upper triangular system $\mathbf{Rx} = \mathbf{y}$ by back substitution (Algorithm 17.1)

This algorithm is backward stable, and this is easy to prove if we assume the three steps are themselves backward stable.

1. We have already argued that the computation of the $\mathbf{QR}$ factorization of a matrix is backward stable (see Theorem 16.1).

2. It can be shown that this step is also backward stable; i.e., the computed $\tilde{\mathbf{y}}$ satisfies

$$(\tilde{\mathbf{Q}} + \delta\mathbf{Q})\tilde{\mathbf{y}} = \mathbf{b}$$

   for some $\delta\mathbf{Q}$ satisfying

$$\|\delta\mathbf{Q}\| = \mathcal{O}(\epsilon_{\text{machine}}).$$

3. We will show in the next lecture (Theorem 17.1) that this step is backward stable; i.e., the computed $\tilde{x}$ satisfies

$$(\tilde{\mathbf{R}} + \delta\mathbf{R})\tilde{\mathbf{x}} = \tilde{\mathbf{y}}$$

   for some $\delta\mathbf{R}$ satisfying

$$\frac{\|\delta\mathbf{R}\|}{\|\mathbf{R}\|} = \mathcal{O}(\epsilon_{\text{machine}})$$

Theorem 2. *Algorithm 16.1 is backward stable*

i.e.,

$$(\mathbf{A} + \Delta\mathbf{A})\tilde{\mathbf{x}} = \mathbf{b}$$

for some $\Delta\mathbf{A}$ satisfying

$$\frac{\|\Delta\mathbf{A}\|}{\|\mathbf{A}\|} = \mathcal{O}(\epsilon_{\mathrm{machine}})$$

<u>Proof</u>:

$$\begin{aligned}
\mathbf{b} &= (\tilde{\mathbf{Q}} + \delta\mathbf{Q})(\tilde{\mathbf{R}} + \delta\mathbf{R})\tilde{\mathbf{x}} \\
&= [\tilde{\mathbf{Q}}\tilde{\mathbf{R}} + (\delta\mathbf{Q})\tilde{\mathbf{R}} + \tilde{\mathbf{Q}}(\delta\mathbf{R}) + (\delta\mathbf{Q})(\delta\mathbf{R})]\tilde{\mathbf{x}} \\
&= [(\mathbf{A} + \delta\mathbf{A}) + (\delta\mathbf{Q})\tilde{\mathbf{R}} + \tilde{\mathbf{Q}}(\delta\mathbf{R}) + (\delta\mathbf{Q})(\delta\mathbf{R})]\tilde{\mathbf{x}} \\
&= [\mathbf{A} + \Delta\mathbf{A}]\tilde{\mathbf{x}}
\end{aligned}$$

where

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

Now, we simply show that $\|\Delta\mathbf{A}\|$ is small.

First,

$$\begin{aligned} \frac{\|\tilde{\mathbf{R}}\|}{\|\mathbf{A}\|} &= \frac{\|\tilde{\mathbf{Q}}^T(\mathbf{A}+\delta\mathbf{A})\|}{\|\mathbf{A}\|} \\ &\leq \|\tilde{\mathbf{Q}}^T\|\frac{\|\mathbf{A}+\delta\mathbf{A}\|}{\|\mathbf{A}\|} \\ &= \mathcal{O}(1) \end{aligned}$$

Thus,

$$\begin{aligned} \frac{\|(\delta\mathbf{Q})\tilde{\mathbf{R}}\|}{\|\mathbf{A}\|} &\leq \|\delta\mathbf{Q}\|\frac{\|\tilde{\mathbf{R}}\|}{\|\mathbf{A}\|} \\ &= \mathcal{O}(\epsilon_{\text{machine}}) \end{aligned}$$

Similarly,

$$\begin{aligned} \frac{\|\tilde{\mathbf{Q}}(\delta\mathbf{R})\|}{\|\mathbf{A}\|} &\leq \|\tilde{\mathbf{Q}}\|\frac{\|\delta\mathbf{R}\|}{\|\tilde{\mathbf{R}}\|}\frac{\|\tilde{\mathbf{R}}\|}{\|\mathbf{A}\|} \\ &= \mathcal{O}(\epsilon_{\text{machine}}) \end{aligned}$$

Finally,

$$\begin{aligned}\frac{\|(\delta \mathbf{Q})(\delta \mathbf{R})\|}{\|\mathbf{A}\|} &\leq \|\delta \mathbf{Q}\| \frac{\|\delta \mathbf{R}\|}{\|\mathbf{A}\|} \\ &= \mathcal{O}(\epsilon_{\mathrm{machine}}^2)\end{aligned}$$

$$\begin{aligned}\therefore \frac{\|\Delta \mathbf{A}\|}{\|\mathbf{A}\|} &\leq \frac{\|\delta \mathbf{A}\|}{\|\mathbf{A}\|} + \frac{\|(\delta \mathbf{Q})\tilde{\mathbf{R}}\|}{\|\mathbf{A}\|} + \frac{\|\tilde{\mathbf{Q}}(\delta \mathbf{R})\|}{\|\mathbf{A}\|} \\ &\qquad + \frac{\|(\delta \mathbf{Q})(\delta \mathbf{R})\|}{\|\mathbf{A}\|} \\ &= \mathcal{O}(\epsilon_{\mathrm{machine}}), \qquad \text{as required.}\end{aligned}$$

Combining Theorems 12.2, 15.1, and 16.2 finally yields

Theorem 3. *The solution* $\tilde{\mathbf{x}}$ *computed by Algorithm 16.1 to solve* $\mathbf{A}\mathbf{x} = \mathbf{b}$ *satisfies*

$$\frac{\|\tilde{\mathbf{x}} - \mathbf{x}\|}{\|\mathbf{x}\|} = \mathcal{O}(\kappa(\mathbf{A})\epsilon_{\mathrm{machine}}).$$