# Stability of QR factorization

Recall: QR factorization is a series of steps of the form

$$\begin{bmatrix} I & \\ & H(u_k) \end{bmatrix} \begin{bmatrix} * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & * & * & * \end{bmatrix} = \begin{bmatrix} * & * & * & * & * \\ 0 & * & * & * & * \\ 0 & 0 & * & * & * \\ 0 & 0 & 0 & * & * \\ 0 & 0 & 0 & * & * \end{bmatrix}$$

Each of these steps, alone, is backward stable: the computed $R_k, u_k$ is the exact result that we'd get if we started from $R_{k-1} + \Delta R_{k-1}$, with

$$\|\Delta R_{k-1}\| = O(\mathbf{u})\|R_{k-1}\|.$$

(The $O(\mathbf{u})$ notation here may 'hide' polynomial factors in $n$).

# Backward stability combined

At each step, we can pretend that all operations we have made up to there are the exact result obtained if we'd start from

$$A + \Delta_1 A + \Delta_2 A + \cdots + \Delta_{k-1} A.$$

and for each $i$ we have $\|\Delta_i A\| \leq O(\mathbf{u})\|R_{k-1}\| \approx O(\mathbf{u})\|A\|$.

Important detail: $\|R_{k-1}\| \approx \|A\|$ because we get one from the other through orthogonal transformations.

QR factorization is backward stable: the computed $Q, R$ are the exact result of $\mathtt{qr}(A + \Delta A)$, $\|\Delta A\| \leq O(\mathbf{u})\|A\|$.

(Some tedious algebra required to formalize all this — not our goal in this course.)

# Backward stability of least squares algorithms

```
function x = solve_ls_QR(A, b)
[Q1, R1] = qr(A, 0); %backward stable
c = Q1'*b; %backward stable
x = R1 \ c; %backward stable
```

Backward stable: all errors here of size $O(\mathbf{u})\|A\|$.

```
function x = solve_ls_SVD(A, b)
[U, S, V] = svd(A, 0); %backward stable
c = U'*b; %backward stable
d = c ./ diag(S); %backward stable
x = V*d; %backward stable
```

Backward stable: all errors here of size $O(\mathbf{u})\|A\|$.

(SVD is backward stable too — we still haven't seen how to compute it, but it's all orthogonal transformations, too.)

# The problem with normal equations

```
function x = solve_ls_NE(A, b)
C = A' * A;
d = A' * b;
x = C \ d;
```

How is this not backward stable? Same kind of operations...

However,

- We use inputs multiple times: so for instance $C = (A + \Delta_1 A)^T (A + \Delta_2 A)$: not necessarily the same perturbation.
- In the last line, errors have size $O(\mathbf{u})(\|C\|) = O(\mathbf{u})(\|A\|^2)$.

So, long story short (we didn't *prove* everything)

The QR and SVD methods are backward stable, but normal equations always give errors of size $\kappa(A)^2$, never $\kappa(A)$.

# Comparison of least squares algorithms

| | Normal eqns | QR | SVD |
|---|---|---|---|
| $m \approx n$ | $\frac{4}{3}n^3$ | $\frac{4}{3}n^3$ | $\approx 13n^3$ |
| $m \gg n$ | $mn^2$ | $2mn^2$ | $2mn^2$ |
| | Unstable when $cond \approx \kappa(A)$ | Backward stable | Backward stable; reveals info on sensitivity, allows regularization |

Know when to use each one. QR is a good 'generic' choice.

# Exercises

1. Show that $\|A^T A\| = \|A\|^2$. Hint: recall how $\|A\| = \|U\Sigma V^T\| = \|\Sigma\| = \sigma_1$: can we do something similar for $A^T A$?
2. Show that $\kappa(A^T A) = \kappa(A)^2$.