- Reducing $Ax-b$ to a triangular form while preserving $\|Ax-b\|_2$
- Illustration of numerical issues
- More general problems: adding LS constraints, eigenvalues; the details in 6.337 / 18.335
- If A does not have this structure, we will transform it and \( r \) so \( A \) is \( [\overline{r}] \) while preserving \( \Sigma r_i^2 \), so we end up with the optimal.

-Trick 1 (of 2): mix pairs of rows to introduce one zero at a time to matrix while preserving \( \Sigma r_i^2 \).

\[
\begin{bmatrix}
A_i
\end{bmatrix}
\begin{bmatrix}
x_i
\end{bmatrix} -
\begin{bmatrix}
b_i
\end{bmatrix}
\equiv
\begin{bmatrix}
-A_{i-1}
\vdots
-A_{m-1}
\end{bmatrix}
\begin{bmatrix}
x_{i-1}
\vdots
x_n
\end{bmatrix}
-
\begin{bmatrix}
b_{i-1}
\vdots
b_m
\end{bmatrix}
\]

\[
\begin{bmatrix}
-A_{i-1}
\vdots
-A_{m-1}
\end{bmatrix}
\begin{bmatrix}
b_{i-1}
\vdots
b_m
\end{bmatrix}
\overset{s}{\rightarrow}
\begin{bmatrix}
A_{i-1}
\vdots
A_{m-1}
\end{bmatrix}
\begin{bmatrix}
b_{i-1}
\vdots
b_m
\end{bmatrix}
\begin{bmatrix}
ca_{m-1}+sca_{m}
\vdots
ca_{m-1}+sca_{m}
\end{bmatrix}
\begin{bmatrix}
cb_{m-1}+scb_{m}
\vdots
-cb_{m-1}+scb_{m}
\end{bmatrix}
\]

Let's find \( c, s \) such that position \((m,1)\) becomes zero.

We have two constraints:

\[-Sa_{m-1, 1} + ca_{m, 1} = 0\]
\[(cr_{m-1} + sr_{m})^2 + (-sr_{m-1} + cr_{m})^2 = r_{m-1}^2 + r_m^2\]
\[(c^2 + s^2)r_{m-1}^2 + (s^2 + c^2)r_m^2 = r_{m-1}^2 + r_m^2\]
\[c^2 + s^2 = 1\]

and \( c = \frac{Sa_{m-1, 1}}{a_{m, 1}} \) or \( s = \frac{ca_{m, 1}}{a_{m-1, 1}} \).

Solve the quadratic in \( c \) (or \( s \)) and we are done.
- Trick 2: Use trick 1 to create more and more zeros in $A$ without destroying earlier ones (or while destroying just a few in more advanced versions).

\[
\begin{bmatrix}
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots \\
\end{bmatrix}
\begin{bmatrix}
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots \\
\end{bmatrix}
\rightarrow
\begin{bmatrix}
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots \\
\end{bmatrix}
\begin{bmatrix}
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots \\
\end{bmatrix}
\rightarrow
\begin{bmatrix}
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots \\
\end{bmatrix}
\begin{bmatrix}
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots \\
\end{bmatrix}
\]

- This algorithm is called the Givens QR factorization (we factored $A = Q_1Q_2\cdots Q_K R = QR$, $Q$ orthonormal and upper triangular).

- Complexity: $O(n)$ operations per rotation

  $O(nm)$ rotations

  $O(mn^2)$ operations total.

- Numerical issues:

  - Avoid divisions by 0 (or by small numbers) when computing $c$, $s$. This is easy.

  - Zero in $a_{ii}$: set $x_i$ arbitrarily (e.g. $x_i=0$) because $r_i = b_i$ anyway. Then go on.

  Small nonzeros $a_{ii}$ are hard to deal with; advanced
Solving Sparse Least Squares Problems

In the smooth-function reconstruction problem, A has plenty of zeros to begin with; we can exploit that, 

Evolution of the non-zero structure of the matrix
Order of elimination
Order of filling in

For this problem, if we only constrain the values of f(x) at a constant # points, total cost is O(N).

There are clever elimination-ordering algs for other sparsity patterns, for general sparse matrices, and for other factorizations.
Additional issues in Givens Rotations

- Solving a quadratic \( ax^2 + bx + c = 0 \)
  \[ x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \]

- Make \(-b\) and \(\pm \sqrt{b^2 - 4ac}\) have same sign;
  avoid subtracting large #’s to recover a small difference

- The choice boils down to which of 2 rotations to use:

- In Python:
  \[ x = 1.0 \]
  \[ y = 1e-17 \]
  \[ x + y - x \implies 0.0 \]
  \[ x - y + y \implies 1e-17 \] (floating point arithmetic is not commutative; other axioms that hold)