Then $w_2^T w_2 = 1$, so

Then $P^{(2)} = I - 2w_2w_2^T$ is a Householder matrix

and

$$P^{(2)}\tilde{A} = \begin{pmatrix} s_1 & x & x & \cdots & x \\ 0 & s_2 & x & & \vdots \\ \vdots & 0 & x & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & x & \cdots & x \end{pmatrix}$$

↳ the action of $P^{(2)}$ is to leave the first line unchanged, to set $\tilde{a}_{22}$ to s_2 and zero out all coefficients in the second column below $\tilde{a}_{22}$.

(c) → If we do this repetitively then

$$P^{(n-1)} \cdots P^{(2)} P^{(1)} A = R$$ ← the successive operations turn A into an upper triangular matrix.

But since $P^{(i)}$ is an orthogonal matrix,

$\prod_i P^{(i)}$ is also orthogonal

→ let $Q^T = Q^{-1} = \prod_i P^{(i)}$ then

$A = QR$ as desired.

(4) Use of QR for least-square methods

In standard least-square solution, the over-determined $n \times m$ problem

$$AX = B \qquad (n > m)$$

is reduced to

$$A^T A X = A^T B$$ → an $(m \times m)$ problem.

A common problem arises if the entries of A span several orders of magnitude ⇒ truncation errors accumulate in the calculation of $A^T A$

which lead to even larger errors in the calculation of the solution x.

$\rightarrow$ The idea is to avoid constructing A^TA but instead to work with A only

QR method for Least-Square problems

if $\quad Ax = b \quad$ then

$$QRx = b \quad \Rightarrow \quad \underbrace{R}_{n\times m}\,\underbrace{x}_{m} = \underbrace{Q^T}_{n\times n}\,\underbrace{b}_{n}.$$

Now R is upper triangular, and can be re-written as

$$R = \begin{pmatrix} \nabla \\ 0 \end{pmatrix} = \begin{pmatrix} \tilde{R} \\ 0 \end{pmatrix} \begin{array}{l} \} \text{ an } m\times m \text{ matrix} \\ \} \text{ an } (n-m)\times m \quad 0 \text{ matrix}. \end{array}$$

$$\Rightarrow \quad Rx = \begin{pmatrix} \tilde{R}x \\ 0 \end{pmatrix} \begin{array}{l} \} \ m \text{ elements} \\ \} \ n-m \text{ elements, all zero}. \end{array}$$

So if we write $\quad Q^T = \begin{pmatrix} \tilde{Q}_1^T \\ \hdashline \tilde{Q}_2^T \end{pmatrix} \begin{array}{l} \leftarrow (m\times n) \\ \leftarrow (n-m \times n) \end{array}$

then the system becomes (exactly)

$$\begin{pmatrix} \tilde{R}x \\ 0 \end{pmatrix} = \begin{pmatrix} \tilde{Q}_1^T b \\ \tilde{Q}_2^T b \end{pmatrix} \quad \Rightarrow \quad \begin{cases} \tilde{R}x = \tilde{Q}_1^T b \\ 0 = \tilde{Q}_2^T b. \end{cases}$$

<u>Claim</u> : The solution of this system which minimizes the (approximate) error is the (exact) solution of the reduced system $\hat{R}x = \tilde{Q}_1^T b$

Proof :

The error is measured with the Euclidian norm as

$$\| Ax-b\| = \| QRx-b\|$$

$$= \| Q^TQRx - Q^Tb \|$$ since $\|AB\| = \|A\|\|B\|$ and $\|Q\| = \|Q^T\| = 1$

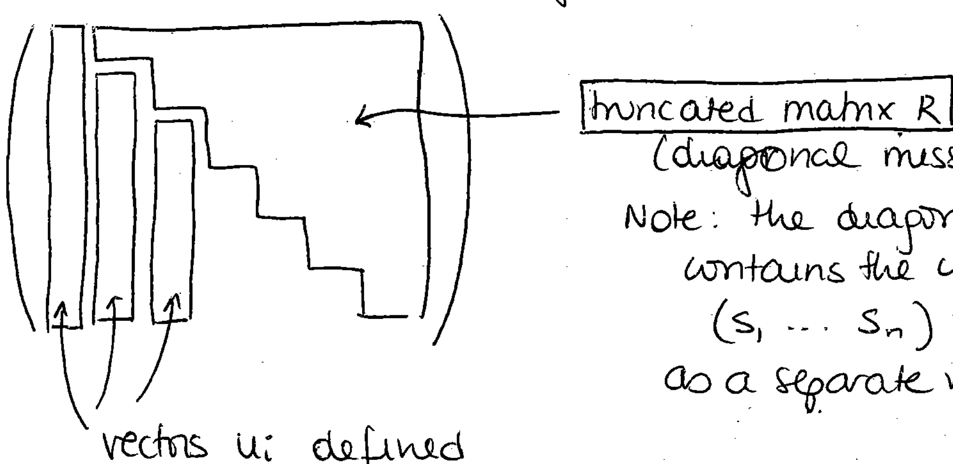
$$= \| Rx - Q^Tb \|$$ since $Q^TQ = I$

$$= \sqrt{\|\tilde{R}x - \tilde{Q}_1^T b\|^2 + \|\tilde{Q}_2^T b\|^2}$$

$\rightarrow$ to minimize this number, find x such that $\tilde{R}x = \tilde{Q}_1^T b$.

5 Practical implementation of QR factorization

- A standard QR algorithm will return the QR factorization in the original matrix A as



Note: the diagonal, which contains the coefficients $(s_1 \dots s_n)$ is returned as a separate vector.

as $\underline{u}_i = \sqrt{2s_i(s_i - a_{ii})}\,\underline{w}_i$

- Note how the algorithm does not return Q nor the matrices P_i $\Rightarrow$ only the vectors "$\underline{w}_i$" are returned, which can be used to reconstruct $P_i^{(i)}$ and Q easily
- It turns out that the vectors $\underline{u}_i$ are easier to store than the $\underline{w}_i$, as they naturally arise as the algorithm unfolds.

(a) <u>Algorithm for QR reduction</u> (for a square matrix A)

do $k = 1, n-1$ ← $n-1$ steps of QR

- calculate $s_k = \pm\sqrt{\sum_{i=k}^{n} a_{ik}^2}$
- store $c_k = s_k(s_k - a_{kk})$ in vector $\underline{c}$
- "Apply $P^{(k)}$ to A" which involves

do nothing ① $\left\{ \begin{array}{l} \text{lines } 1 \to k-1 \text{ of matrix are unchanged} \\ \text{columns } 1 \to k-1 \text{ of matrix are unchanged} \end{array} \right.$

this stores vector u_k in column k ② $\left\{ \begin{array}{l} a_{kk} = a_{kk} - s_k \quad \leftarrow \text{diagonal element is now } a_{kk} - s_k \\ \text{rest of column } k \text{ is unchanged} \end{array} \right.$

③ Calculate the effect of P^k on A on the submatrix $i = k \to n$, $j = k+1 \to n$

```
do j = k+1, n
    do i = k, n
        a_ij := (P^(k) A)_ij        (*)
    enddo
enddo
```

enddo

The step (*) involves calculating ~~for~~

$$\sum_{m=1}^{n} \left(I_{im} - 2w_i^{(k)} w_m^{(k)}\right) a_{mj} = \sum_{m=1}^{n} \delta_{im} a_{mj} - \frac{u_i^{(k)} u_m^{(k)}}{c_k} a_{mj}$$

$$= a_{ij} - \sum_{m=k}^{n} \frac{1}{c_k} u_i^{(k)} u_m^{(k)} a_{mj}$$

but the $u_i^{(k)}$ and $u_m^{(k)}$ components are stored already in a_{ik} and a_{mk} so

$$a_{ij} := a_{ij} - \sum_{m=k}^{n} \frac{a_{ik} a_{mk} a_{mj}}{c_k}$$

$$= a_{ij} - \frac{a_{ik}}{c_k} \sum_{m=k}^{n} a_{mk} a_{mj}$$

(b) Solution of $QRX = b$ (for a square system)

$\Rightarrow$ This algorithm returns the matrix A as shown, as well as the vectors $\underline{s}$ (containing the s_k coefficients) and $\underline{c}$ (containing $s_k(s_k - a_{kk})$).

Now we simply have to use the given algorithm output to find solutions to $QRX = b \iff Rx = Q^T b$.

To do this, we must evaluate $Q^T b$, then simply perform a back-substitution. To evaluate $Q^T b$, note that

$$Q^T b = \left(P^{(n-1)} P^{(n-2)} \cdots P^{(2)} P^{(1)} \right) b$$

$\rightarrow$ we must first calculate $P^{(1)} b$, then $P^{(2)} P^{(1)} b$, etc... . Remember that

$$P^{(i)} b = (I - 2\underline{w}_i \underline{w}_i^T) \underline{b}$$

$$= \left(I - \frac{1}{c_i} \underline{u}_i \underline{u}_i^T \right) \underline{b}$$

but the vector $\underline{u}_i$ is now stored in the i-th lower-column of A so the following algorithm will return $Q^T b$:

```
do k = 1, n-1
|   do i = 1, n
|   |       b_i = b_i - (a_ik / c_k) Σ_{m=k}^{n} a_mk b_m
|   enddo
enddo
```

Finally, backsubstitute with the stored R, remembering that the diagonal of R is stored in $\underline{s}$.