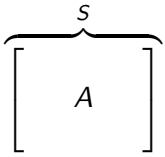
Quiz

- Let $\mathcal{W}$ be a vector space, and let $\mathcal{U}$ be a subspace of $\mathcal{W}$.
  Define the *orthogonal complement* of $\mathcal{U}$ in $\mathcal{W}$.
- Let $\mathcal{W} = \mathbb{R}^3$. Let $\mathcal{U} = \text{Span}\ \{[1, 2, 0]\}$. Give a basis for the orthogonal complement of $\mathcal{U}$ in $\mathcal{W}$.
- Give a matrix A such that the basis in the previous question is a basis for Null A.
- (Not required; no credit) Give a matrix B such that Null $B = \mathcal{U}$.

Orthogonal complement in $\mathbb{R}^n$ and null space of a matrix

Let A be an $R \times S$ matrix. Let $\mathcal{W} = \mathbb{R}^S$ and let $\mathcal{U} = \text{Row } A$.



For any vector $\mathbf{v}$ in the orthogonal complement of $\mathcal{U}$ with respect to $\mathcal{W}$,
inner product of each row with $\mathbf{v}$ equals zero,
so by dot-product interpretation of matrix-vector mult, $A\mathbf{v} = \mathbf{0}$
showing that orthogonal complement of $\mathcal{U}$ in $\mathbb{R}^S$ is a subspace of Null A

Conversely, if $A\mathbf{v} = \mathbf{0}$ then inner product of each row of A with $\mathbf{u}$ equals zero,
so inner product with $\mathbf{v}$ of any *linear combination of rows of A* is zero,
so $\mathbf{v}$ is orthogonal to every vector in Row A,
so $\mathbf{v}$ is in orthogonal complement of $\mathcal{U}$
showing Null A is a subspace of orthogonal complement of $\mathcal{U}$ in $\mathbb{R}^S$

Computing the orthogonal complement

Suppose we have a basis $\mathbf{u}_1, \ldots, \mathbf{u}_k$ for $\mathcal{U}$ and a basis $\mathbf{w}_1, \ldots, \mathbf{w}_n$ for $\mathcal{W}$.
How can we compute a basis for the orthogonal complement of $\mathcal{U}$ in $\mathcal{W}$?
One way: use `orthogonalize(vlist)` with vlist $= [\mathbf{u}_1, \ldots, \mathbf{u}_k, \mathbf{w}_1, \ldots, \mathbf{w}_n]$

Write list returned as $[\mathbf{u}_1^*, \ldots, \mathbf{u}_k^*, \mathbf{w}_1^*, \ldots, \mathbf{w}_n^*]$

These span the same space as input vectors $\mathbf{u}_1, \ldots, \mathbf{u}_k, \mathbf{w}_1, \ldots, \mathbf{w}_n$, namely $\mathcal{W}$, which has dimension n.

Therefore exactly n of the output vectors $\mathbf{u}_1^*, \ldots, \mathbf{u}_k^*, \mathbf{w}_1^*, \ldots, \mathbf{w}_n^*$ are nonzero.

The vectors $\mathbf{u}_1^*, \ldots, \mathbf{u}_k^*$ have same span as $\mathbf{u}_1, \ldots, \mathbf{u}_k$ and are all nonzero since $\mathbf{u}_1, \ldots, \mathbf{u}_k$ are linearly independent.

Therefore exactly $n - k$ of the remaining vectors $\mathbf{w}_1^*, \ldots, \mathbf{w}_n^*$ are nonzero.

Every one of them is orthogonal to $\mathbf{u}_1, \ldots, \mathbf{u}_n$...
so they are orthogonal to every vector in $\mathcal{U}$...
so they lie in the orthogonal complement of $\mathcal{U}$.
By Direct-Sum Dimension Lemma, orthogonal complement has dimension $n - k$, so the remaining nonzero vectors are a basis for the orthogonal complement.

Finding basis for null space using orthogonal complement

To find basis for null space of an $m \times n$ matrix $A = \left[\begin{array}{c} \mathbf{a}_1 \\ \hline \vdots \\ \hline \mathbf{a}_m \end{array}\right]$,

find orthogonal complement of Span $\{\mathbf{a}_1, \ldots, \mathbf{a}_m\}$ in $\mathbb{R}^n$:

- Let $\mathbf{e}_1, \ldots, \mathbf{e}_n$ be the standard basis vectors $\mathbb{R}^n$.
- Let $[\mathbf{a}_1^*, \ldots, \mathbf{a}_m^*, \mathbf{e}_1^*, \ldots, \mathbf{e}_n^*] = \texttt{orthogonalize}([\mathbf{a}_1, \ldots, \mathbf{a}_m, \mathbf{e}_1, \ldots, \mathbf{e}_n])$
- Find the nonzero vectors among $\mathbf{e}_1^*, \ldots, \mathbf{e}_n^*$

Augmenting `project_along`

```
def project_along(b, v):
    sigma = 0 if v.is_almost_zero() else (b*v)/(v*v)
    return sigma * v
```

Want to also output the scalar that when multiplied by **v** gives the projection

```
def project_along(b, v):
    sigma = 0 if v.is_almost_zero() else (b*v)/(v*v)
    return sigma * v, sigma
```

Augmenting project_onto

```
def project_onto(b, vlist):
    return sum([project_along(b, v) for v in vlist], zero_vec(b.D))
```

Want to also output the dictionary (not Vec) that maps the index of each vector in vlist to the coefficient giving the projection of **b** onto that vector

```
def aug_project_onto(b, vlist):
    L = [aug_project_along(b,v) for v in vlist]
    b_par = sum([v_proj for (v_proj, sigma) in L], zero_vec(b.D))
    sigma_dict = {i:L[i][1] for i in range(len(L))}
    return b_par, sigma_dict
```

Augmenting `project_orthogonal`

Suppose `vlist`=$[\mathbf{v}_0^*, \ldots, \mathbf{v}_{n-1}^*]$.
Let $\mathbf{b}^{||}$ = projection of $\mathbf{b}$ onto Span $\{\mathbf{v}_0^*, \ldots, \mathbf{v}_{n-1}^*\}$.
Let $\mathbf{b}^{\perp}$ = projection of $\mathbf{b}$ orthogonal to Span $\{\mathbf{v}_0^*, \ldots, \mathbf{v}_{n-1}^*\}$.

$$\begin{aligned} \mathbf{b} &= \mathbf{b}^{||} + \mathbf{b}^{\perp} \\ &= \sigma_0 \mathbf{v}_0^* + \cdots + \sigma_{n-1}\mathbf{v}_{n-1}^* + \mathbf{b}^{\perp} \end{aligned}$$

$$\left[\begin{array}{c} \\ \mathbf{b} \\ \\ \end{array}\right] = \left[\begin{array}{c|c|c|c} & & & \\ \mathbf{v}_0^* & \cdots & \mathbf{v}_{n-1}^* & \mathbf{b}^{\perp} \\ & & & \end{array}\right] \left[\begin{array}{c} \sigma_0 \\ \vdots \\ \sigma_{n-1} \\ 1 \end{array}\right]$$

The procedure `project_orthogonal(b, vlist)` can be augmented to output the vector of coefficients.

For technical reasons, we will represent the vector of coefficents as a dictionary, not a `Vec`.

Augmenting `project_orthogonal`

$$\begin{bmatrix} \\ \mathbf{b} \\ \\ \end{bmatrix} = \left[\begin{array}{c|c|c|c} & & & \\ \mathbf{v}_0 & \cdots & \mathbf{v}_n & \mathbf{b}^\perp \\ & & & \end{array}\right] \begin{bmatrix} \alpha_0 \\ \vdots \\ \alpha_n \\ 1 \end{bmatrix}$$

Must create and populate a dictionary.

- One entry for each vector in `vlist`
- One additional entry, 1, for $\mathbf{b}^\perp$

Initialize dictionary with the additional entry.

```
def aug_project_onto(b, vlist):
 L = ...
 b_par = ...
 sigma_dict = {i:L[i][1] for i
                in range(len(L))}
 return b_par, sigma_dict

def project_orthogonal(b, vlist):
    return b - project_onto(b, vlist)
```

```
def aug_project_orthogonal(b, vlist):
    b_par, sigma_dict = aug_project_ont
    sigma_dict[len(vlist)] = 1
    return b - b_par, sigma_dict
```

Augmenting orthogonalize(vlist)

We will write a procedure aug_orthogonalize(vlist) with the following spec:

- *input:* a list $[\mathbf{v}_1, \ldots, \mathbf{v}_n]$ of vectors
- *output:* the pair $([\mathbf{v}_1^*, \ldots, \mathbf{v}_n^*], [\mathbf{r}_1, \ldots, \mathbf{r}_n])$ of lists of vectors such that $\mathbf{v}_1^*, \ldots, \mathbf{v}_n^*$ are mutually orthogonal vectors whose span equals Span $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$, and

$$\left[\begin{array}{c|c|c} & & \\ \mathbf{v}_1 & \cdots & \mathbf{v}_n \\ & & \end{array}\right] = \left[\begin{array}{c|c|c} & & \\ \mathbf{v}_1^* & \cdots & \mathbf{v}_n^* \\ & & \end{array}\right] \left[\begin{array}{c|c|c} & & \\ \mathbf{r}_1 & \cdots & \mathbf{r}_n \\ & & \end{array}\right]$$

```
def orthogonalize(vlist):
  vstarlist = []
  for v in vlist:
    vstarlist.append(
      project_orthogonal(v, vstarlist))
  return vstarlist
```

```
def aug_orthogonalize(vlist):
  vstarlist = []
  sigma_vecs = []
  D = set(range(len(vlist)))
  for v in vlist:
    vstar, sigma_dict =
      aug_project_orthogonal(v, vstarlist
    vstarlist.append(vstar)
    sigma_vecs.append(Vec(D, sigma_dict))
  return vstarlist, sigma_vecs
```

Towards QR factorization

We will now develop the *QR factorization*. We will show that certain matrices can be written as the product of matrices in special form.
Matrix factorizations are useful mathematically and computationally:

- *Mathematical:* They provide insight into the nature of matrices—each factorization gives us a new way to think about a matrix.
- *Computational:* They give us ways to compute solutions to fundamental computational problems involving matrices.

Matrices with mutually orthogonal columns

$$\begin{bmatrix} \mathbf{v}_1^{*T} \\ \hline \vdots \\ \hline \mathbf{v}_n^{*T} \end{bmatrix} \left[\begin{array}{c|c|c} & & \\ \mathbf{v}_1^* & \cdots & \mathbf{v}_n^* \\ & & \end{array}\right] = \begin{bmatrix} \|\mathbf{v}_1^*\|^2 & & \\ & \ddots & \\ & & \|\mathbf{v}_n^*\|^2 \end{bmatrix}$$

Cross-terms are zero because of mutual orthogonality.

To make the product into the identity matrix, can *normalize* the columns.

Normalizing a vector means scaling it to make its norm 1.

Just divide it by its norm.

```
>>> def normalize(v): return v/sqrt(v*v)
>>> q = normalize(list2vec[1,1,1])
>>> q * q
1.0000000000000002
```

Matrices with mutually orthogonal columns

$$\begin{bmatrix} \mathbf{v}_1^{*T} \\ \hline \vdots \\ \hline \mathbf{v}_n^{*T} \end{bmatrix} \left[\begin{array}{c|c|c} & & \\ \mathbf{v}_1^* & \cdots & \mathbf{v}_n^* \\ & & \end{array}\right] = \begin{bmatrix} \|\mathbf{v}_1^*\|^2 & & \\ & \ddots & \\ & & \|\mathbf{v}_n^*\|^2 \end{bmatrix}$$

Cross-terms are zero because of mutual orthogonality.

To make the product into the identity matrix, can *normalize* the columns.

Normalize columns $\left[\begin{array}{c|c|c} & & \\ \mathbf{v}_1^* & \cdots & \mathbf{v}_n^* \\ & & \end{array}\right] \Rightarrow \left[\begin{array}{c|c|c} & & \\ \mathbf{q}_1 & \cdots & \mathbf{q}_n \\ & & \end{array}\right]$

Matrices with mutually orthogonal columns

$$\begin{bmatrix} \mathbf{q}_1^T \\ \hline \vdots \\ \hline \mathbf{q}_n^T \end{bmatrix} \left[\begin{array}{c|c|c} & & \\ \mathbf{q}_1 & \cdots & \mathbf{q}_n \\ & & \end{array}\right] = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix}$$

Normalize columns $\left[\begin{array}{c|c|c} & & \\ \mathbf{v}_1^* & \cdots & \mathbf{v}_n^* \\ & & \end{array}\right] \Rightarrow \left[\begin{array}{c|c|c} & & \\ \mathbf{q}_1 & \cdots & \mathbf{q}_n \\ & & \end{array}\right]$

Matrices with mutually orthogonal columns

$$\left[\begin{array}{c} \mathbf{q}_1^T \\ \hline \vdots \\ \hline \mathbf{q}_n^T \end{array}\right] \left[\begin{array}{c|c|c} & & \\ \mathbf{q}_1 & \cdots & \mathbf{q}_n \\ & & \end{array}\right] = \begin{bmatrix} 1 & & \\ & \ddots & \\ & & 1 \end{bmatrix}$$

Proposition: If columns of Q are mutually orthogonal with norm 1 then $Q^T Q$ is identity matrix.

Definition: Vectors that are mutually orthogonal and have norm 1 are *orthonormal*.

Definition: If columns of Q are orthonormal then we call Q a *column-orthogonal* matrix. Should be called *orthonormal* but oh well

Definition: If Q is square and column-orthogonal, we call Q an *orthogonal* matrix.

Proposition: If Q is an orthogonal matrix then its inverse is Q^T.

Projection onto columns of a column-orthogonal matrix

Suppose $\mathbf{q}_1, \ldots, \mathbf{q}_n$ are orthonormal vectors.

Projection of $\mathbf{b}$ onto $\mathbf{q}_j$ is $\mathbf{b}^{\|\mathbf{q}_j} = \sigma_j \mathbf{q}_j$ where $\sigma_j = \dfrac{\langle \mathbf{q}_j, \mathbf{b} \rangle}{\langle \mathbf{q}_j, \mathbf{q}_j \rangle} = \langle \mathbf{q}_j, \mathbf{b} \rangle$

Vector $[\sigma_1, \ldots, \sigma_n]$ can be written using dot-product definition of matrix-vector multiplication:

$$\begin{bmatrix} \sigma_1 \\ \vdots \\ \sigma_n \end{bmatrix} = \begin{bmatrix} \mathbf{q}_1 \cdot \mathbf{b} \\ \vdots \\ \mathbf{q}_n \cdot \mathbf{b} \end{bmatrix} = \begin{bmatrix} \text{---} \; \mathbf{q}_1^T \; \text{---} \\ \vdots \\ \text{---} \; \mathbf{q}_n^T \; \text{---} \end{bmatrix} \begin{bmatrix} \\ \mathbf{b} \\ \\ \end{bmatrix}$$

and linear combination $\sigma_1 \mathbf{q}_1 + \cdots + \sigma_n \mathbf{q}_n = \left[\begin{array}{c|c|c} & & \\ \mathbf{q}_1 & \cdots & \mathbf{q}_n \\ & & \end{array}\right] \begin{bmatrix} \sigma_1 \\ \vdots \\ \sigma_n \end{bmatrix}$

Towards QR factorization

Orthogonalization of columns of matrix A gives us a representation of A as product of

- matrix with mutually orthogonal columns
- invertible triangular matrix

$$\begin{bmatrix} | & | & | & & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 & \cdots & \mathbf{v}_n \\ | & | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & | & & | \\ \mathbf{v}_1^* & \mathbf{v}_2^* & \mathbf{v}_3^* & \cdots & \mathbf{v}_n^* \\ | & | & | & & | \end{bmatrix} \begin{bmatrix} 1 & \alpha_{12} & \alpha_{13} & & \alpha_{1n} \\ & 1 & \alpha_{23} & & \alpha_{2n} \\ & & 1 & & \alpha_{3n} \\ & & & \ddots & \\ & & & & \alpha_{n-1,n} \\ & & & & 1 \end{bmatrix}$$

Suppose columns $\mathbf{v}_1, \ldots, \mathbf{v}_n$ are linearly independent. Then $\mathbf{v}_1^*, \ldots, \mathbf{v}_n^*$ are nonzero.

- Normalize $\mathbf{v}_1^*, \ldots, \mathbf{v}_n^*$ (Matrix is called Q)
- To compensate, scale the rows of the triangular matrix. (Matrix is R)

The result is the QR factorization.

Q is a column-orthogonal matrix and R is an upper-triangular matrix.

Towards QR factorization

Orthogonalization of columns of matrix A gives us a representation of A as product of

- matrix with mutually orthogonal columns
- invertible triangular matrix

$$\begin{bmatrix} | & | & | & | \\ \mathbf{v}_2 & \mathbf{v}_3 & \cdots & \mathbf{v}_n \\ | & | & | & | \end{bmatrix} = \begin{bmatrix} | & | & | & | & | \\ \mathbf{q}_1 & \mathbf{q}_2 & \mathbf{q}_3 & \cdots & \mathbf{q}_n \\ | & | & | & | & | \end{bmatrix} \begin{bmatrix} \|\mathbf{v}_1^*\| & \beta_{12} & \beta_{13} & & \beta_{1n} \\ & \|\mathbf{v}_2^*\| & \beta_{23} & & \beta_{2n} \\ & & \|\mathbf{v}_3^*\| & & \beta_{3n} \\ & & & \ddots & \\ & & & & \beta_{n-1,n} \\ & & & & \|\mathbf{v}_n^*\| \end{bmatrix}$$

Suppose columns $\mathbf{v}_1, \ldots, \mathbf{v}_n$ are linearly independent. Then $\mathbf{v}_1^*, \ldots, \mathbf{v}_n^*$ are nonzero.

- Normalize $\mathbf{v}_1^*, \ldots, \mathbf{v}_n^*$ (Matrix is called Q)
- To compensate, scale the rows of the triangular matrix. (Matrix is R)

The result is the QR factorization.

Q is a column-orthogonal matrix and R is an upper-triangular matrix.