1. Symmetric Matrices

We want to show that every real symmetric matrix is diagonalized by a matrix of its orthonormal eigenvectors. In other words, a symmetric matrix has full complement of eigenvectors that are all orthogonal to each other.

In discussion section, you have seen a recursive derivation of a related fact. Formally however, such recursive derivations are usually turned into proofs by using induction. This problem serves to both freshen your mind regarding induction as well as to give you a chance to prove for yourself this very important theorem. (This is the same essential proof as that of Schur upper-triangularization. So understanding this problem will help solidify your understanding of that proof as well.)

(a) You will start by proving a basic lemma about real symmetric matrices under an orthonormal change of basis. Prove that if $S$ is a symmetric matrix ($S = S^T$) and $U$ is a matrix whose columns are orthonormal, then $U^T S U$ (that is, $S$ represented in the basis $U$) is also symmetric.

Solution: $U$ is a matrix with orthonormal column vectors. The transpose of $U^T S U$, $(U^T S U)^T$ is identical to $U^T S U$ as follows:

$$(U^T S U)^T = (SU)^T U = U^T S^T U = U^T S U$$

since $S^T = S$. Notice that it is not even necessary that $U$ is a square matrix or that it has orthonormal columns for this part to be true. If $S$ is an $m \times m$ matrix, $U$ could be $m \times n$ with $n \neq m$.

(b) Another useful lemma is that real symmetric matrices have real eigenvalues. Prove that the eigenvalues $\lambda$ of real, symmetric matrix $S$ are real. Hint: Suppose that $S$ had a complex eigenvalue $\lambda$ with eigenvector $\vec{v}$. Because $S$ is a real matrix, what do you know about $S \vec{v}$ — applying $S$ to the complex conjugate of $\vec{v}$? What happens when you take a potentially complex number and multiply it by its own complex conjugate? Consequently, what do you know happens if you multiply a complex vector $\vec{v}$ by the conjugate of its transpose: i.e. consider $\overline{\vec{v}^T} \vec{v}$? Since $S$ is symmetric, what do you know about $\vec{v}^T S$? Put these ingredients together as shown in lecture.

Solution: Let $\lambda$ be a possibly complex eigenvalue of a real symmetric matrix $A$. Thus, there is a nonzero vector $v$ such that $Av = \lambda v$.

Note, here we are omitting the arrow over the $v$ to avoid visual confusion with taking the complex conjugate with a bar over $v$.

By taking the complex conjugates of both sides, and noting that $\overline{A} = A$ since $A$ has real entries, we get $\overline{Av} = \overline{\lambda v} \implies A\overline{v} = \overline{\lambda v}$

Now, using $A^T = A$

$$\overline{v}^T Av = \overline{v}^T (Av) = \overline{v}^T (\lambda v) = \lambda (\overline{v}^T v)$$

$$\overline{v}^T Av = (A\overline{v})^T v = (\overline{\lambda v})^T v = \overline{\lambda} (\overline{v}^T v)$$

Since $v \neq 0$, we have that $\overline{v}^T v > 0$. Thus, $\lambda = \overline{\lambda}$ and consequently $\lambda \in \mathbb{R}$. 

© UCB EECS 16B, Fall 2019. All Rights Reserved. This may not be publicly shared without explicit permission.
(c) A third useful lemma is one about finding orthonormal bases. Show that given a single nonzero vector \( \vec{u}_0 \) of dimension \( n \), that it is possible to find an orthonormal set of \( n \) vectors, \( \vec{v}_0, \ldots, \vec{v}_{n-1} \) such that \( \vec{v}_0 = \alpha \vec{u}_0 \) for some scalar \( \alpha \).

(Hint: Use the Gram-Schmidt process on the list of \( n+1 \) vectors obtained by starting with the given vector and appending the standard basis — i.e. the columns of the identity matrix.)

**Solution:** Finding \( \vec{v}_0, \ldots, \vec{v}_{n-1} \) can be done with two steps: (1) based on \( \vec{u}_0 \), find \( \vec{u}_1, \ldots, \vec{u}_N \), such that \( \text{span}(\vec{u}_0, \vec{u}_1, \ldots, \vec{u}_N) = \mathbb{R}^n \) with \( N \geq n \) and (2) apply the Gram-Schmidt process to orthonormalize that set of vectors, dropping any zero vectors that we encounter along the way.

For (1), we can simply append the standard basis to \( \vec{v}_0 = \frac{\vec{u}_0}{\|\vec{u}_0\|} \). This gives us \( N = n + 1 \geq n \) vectors that certainly span all of \( \mathbb{R}^n \) since the standard basis definitely spans all of \( \mathbb{R}^n \) even by itself.

For (2), we can simply apply the Gram-Schmidt process to find an orthonormal set of \( n \) vectors. The Gram-Schmidt process takes a finite, linearly independent set \( U = \{\vec{u}_0, \vec{u}_1, \ldots, \vec{u}_N\} \) and generates an orthonormal set \( W = \{\vec{w}_0, \vec{w}_1, \ldots, \vec{w}_{n-1}\} \) that spans the same space as \( U \). If along the way, it finds a zero vector, it just drops that vector.

For this problem, we first compute the set \( W \) based on \( U \): let \( \vec{w}_0 = \vec{u}_0 \), \( \vec{w}_1 = \vec{u}_1 - \frac{(\vec{w}_0, \vec{u}_1)}{\|\vec{w}_0\|^2} \vec{w}_0 \), and keep computing other \( \vec{w}_k \) in \( W \) by subtracting \( \vec{u}_k \) with its projections upon the prior existing \( \vec{w}_0, \vec{w}_1, \ldots, \vec{w}_{k-1} \).

If the result of the subtraction is zero, we can skip it. After that, we normalize each vector in \( W \) to make it an orthonormal basis \( V \), where \( \vec{v}_0 = \frac{\vec{w}_0}{\|\vec{w}_0\|} = \frac{\vec{u}_0}{\|\vec{u}_0\|} = \alpha \vec{u}_0 \).

(d) For the main proof that every real symmetric matrix is diagonalized by a matrix of its orthonormal eigenvectors, we will proceed by formal induction. Recall that for a proof by induction, we have to start with a base case - this is also the base case in a recursive derivation. Consider the trivial case of \( S \) having dimensions \([1 \times 1]\) (\( n = 1 \)). Does \( S \) have an eigenvector? Can this eigenbasis be made orthonormal? Is the matrix diagonal in this basis? Are the entries real?

**Solution:** Yes, it has an eigenvector, because \( S \) would be a scalar. Let \( S = [s] \) and \( \vec{u} = 1 \). Then note that \( S\vec{u} = s\vec{u} \). This implies that \( s \) is an eigenvalue and \( \vec{u} = 1 \) is an eigenvector. This eigenbasis is trivially orthonormal, because there is no other vector which is not orthogonal to \( \vec{u} \). We can think of \( S \) as a \( 1 \times 1 \) matrix, such that the only entry is real. Also, \( S \) is diagonal in that basis because there is only one element.

(e) After the base case, we do an inductive stage of the main proof. The first step in the inductive stage is to write down the induction hypothesis. Assume that the property that every real symmetric matrix is diagonalized by a matrix of its orthonormal eigenvectors holds for all symmetric matrices with size \([ (n-1) \times (n-1) ] \).

Write down the statement of the fact you want to assume in your own words using mathematical notation. (Hint: In general for proofs by induction, you want to start with the strongest version of what you want to prove. This gives you the most powerful inductive hypothesis.)

**Solution:** Any \( (n-1) \times (n-1) \) real symmetric matrix can be diagonalized by a matrix of its orthonormal real eigenvectors.

Mathematical notation: an \( (n-1) \times (n-1) \) real symmetric matrix \( Q \) can be diagonalized by its orthonormal real eigenvectors, \( \vec{u}_0, \ldots, \vec{u}_{n-1} \). Let \( U \) be an \( n-1 \times n-1 \) matrix, where the column vectors are those orthonormal real eigenvectors of \( Q \). Then we can express \( Q \) as \( U\Lambda_Q U^T \), where \( \Lambda_Q \) is a diagonal matrix with real eigenvalues of \( Q \) along its diagonal.

(f) Now think about a symmetric matrix \( S \) with size \([ n \times n ] \). Consider a real eigenvalue \( \lambda_0 \) of \( S \) and the corresponding eigenvector \( \vec{u}_0 \) (a column vector with size \( n \)). Use an appropriate orthonormal
change of basis $V$ to show that $S = VXV^T$, where $X$ is of the form

$$X = \begin{bmatrix}
\lambda_0 & x_{1,2} & \cdots & x_{1,n} \\
0 & x_{2,2} & \cdots & x_{2,n} \\
\vdots & \vdots & \ddots & \vdots \\
0 & x_{n,2} & \cdots & x_{n,n}
\end{bmatrix}$$

That is, the first column of $X$ is

$$\begin{bmatrix}
\lambda_0 \\
0 \\
\vdots \\
0
\end{bmatrix}.$$ 

**Solution:** According to (c), we could derive an orthonormal set of $n$ vectors, $\vec{v}_0, \ldots, \vec{v}_{n-1}$, based on $\vec{u}_0$. Let $\vec{v}_0, \ldots, \vec{v}_{n-1}$ be the columns of matrix $V$. Suppose $X = V^T S V$, such that $S = VXV^T$ (because $V$ is an orthonormal basis, $V^T V = V V^T = I$).

Hence we have

$$X = \begin{bmatrix}
\vec{v}_0^T \\
\vec{v}_1^T \\
\vdots \\
\vec{v}_{n-1}^T
\end{bmatrix} S \begin{bmatrix}
\vec{v}_0 \\
\vec{v}_1 \\
\vdots \\
\vec{v}_{n-1}
\end{bmatrix},$$

where $\vec{v}_0$ is the normalized eigenvector corresponding to the eigenvalue $\lambda_0$ of $S$, which means $S \vec{v}_0 = \lambda_0 \vec{v}_0$. The first column of $X$ will be

$$\begin{bmatrix}
\vec{v}_0^T \\
\vec{v}_1^T \\
\vdots \\
\vec{v}_{n-1}^T
\end{bmatrix} S \vec{v}_0 = \begin{bmatrix}
\vec{v}_0^T \\
\vec{v}_1^T \\
\vdots \\
\vec{v}_{n-1}^T
\end{bmatrix} \lambda_0 \vec{v}_0.$$ 

Since $V$ is an orthonormal matrix, $\vec{v}_i^T \vec{v}_j = 1$ when $i = j$; otherwise it is zero. Therefore, the first column of $X$ is

$$\begin{bmatrix}
\lambda_0 \\
0 \\
\vdots \\
0
\end{bmatrix},$$

which is what we wanted to show.

(g) Continue the previous part to show that in fact the matrix relation can be written as

$$S = V \begin{bmatrix}
\lambda_0 & 0 & \cdots & 0 \\
0 & 0 & & \\
\vdots & \ddots & \ddots & \vdots \\
0 & \cdots & 0 & 0
\end{bmatrix} V^T$$

where $Q$ is an $[n-1 \times n-1]$ symmetric matrix. *Hint: Recall that $S$ is a symmetric matrix.*

**Solution:** Recall the lemma we proved in (a), $X$ must be a symmetric matrix, because $X = V^T S V$ where $S$ is a symmetric matrix and $V$ is orthonormal. Since we have proved the first column of this...
matrix is
\[
\begin{bmatrix}
\lambda_0 \\
0 \\
\vdots \\
0
\end{bmatrix},
\]
the transpose of this is the first row. Hence \( X \) can be written as
\[
\begin{bmatrix}
\lambda_0 \\
0 \\
\vdots \\
0
\end{bmatrix}
\begin{bmatrix}
\vec{v}_0^T \\
\vec{v}_0 Q
\end{bmatrix},
\]
where \( Q \) is an \( n - 1 \) by \( n - 1 \) symmetric matrix. This is a short direct argument that gets us what we want.

An alternative approach was shown in discussion where we computed \( X \) directly. Let \( R \) be a matrix with columns \( \vec{v}_1, \ldots, \vec{v}_{n-1} \), which are orthogonal unit vectors. Recall that \( \lambda_0 \) is an eigenvalue of \( S \), with the corresponding eigenvector \( \vec{v}_0 \). From here, we can compute \( X \) step by step:

\[
X = V^T S V = \begin{bmatrix} \vec{v}_0 & R \end{bmatrix}^T S \begin{bmatrix} \vec{v}_0 & R \end{bmatrix} = \begin{bmatrix} \vec{v}_0^T & R^T \end{bmatrix} S \begin{bmatrix} \vec{v}_0 & R \end{bmatrix} = \begin{bmatrix} \vec{v}_0^T & R^T \end{bmatrix} \begin{bmatrix} \vec{v}_0 & SR \end{bmatrix} = \begin{bmatrix} \vec{v}_0^T & R^T \end{bmatrix} \lambda_0 \vec{v}_0 SR
\]

Now we could write \( X \) as
\[
\begin{bmatrix}
\lambda_0 \vec{v}_0^T \\
\lambda_0 R^T \vec{v}_0 \\
\vec{v}_0 SR
\end{bmatrix},
\]
Because \( \vec{v}_0 \) is a unit vector, \( \vec{v}_0^T \vec{v}_0 = 1 \). Since \( V \) is an orthonormal matrix, the inner product of \( \vec{v}_0 \) with column vectors in \( R \) must be 0, so \( R^T \vec{v}_0 = 0 \). Also, \( \vec{v}_0^T SR = (S\vec{v}_0)^T R = \lambda_0 \vec{v}_0^T R = \vec{0}^T \).

Then we get the desired form of \( X \), where \( Q = R^T SR \). Recall the lemma we proved in (a), because \( S \) is a symmetric matrix, while the columns of \( R \) are orthonormal, \( Q \) must be symmetric. Either one of these approaches is fine.

(h) According to our induction hypothesis, we can write \( Q \) as \( U \Lambda U^T \) where \( U \) is an orthonormal \( [n - 1 \times n - 1] \) square matrix and \( \Lambda \) is a diagonal matrix with real entries along the diagonal and 0s everywhere else. Use this fact to show that indeed there must exist an orthonormal \( [n \times n] \) square matrix \( W \) such that
\[
S = W \begin{bmatrix}
\lambda_0 \\
0 \\
\vdots \\
0
\end{bmatrix} W^T
\]

(Hint: What is the product of orthonormal matrices?)

Solution:
From the previous part, we know that, \( S \) can be written as follows.
\[
S = \begin{bmatrix} \vec{v}_0 & R \end{bmatrix} \begin{bmatrix}
\lambda_0 \\
0 \\
\vdots \\
0
\end{bmatrix} \begin{bmatrix}
\vec{v}_0^T \\
\vec{v}_0 Q
\end{bmatrix}
\]

Thanks to our induction hypothesis, we can compute \( S \) as follows:
\[
S = \begin{bmatrix} \vec{v}_0 & R \end{bmatrix} \begin{bmatrix}
\lambda_0 \\
0 \\
\vdots \\
0
\end{bmatrix} \begin{bmatrix}
\vec{v}_0^T \\
\vec{v}_0 U \Lambda U^T
\end{bmatrix}
\]

© UCB EECS 16B, Fall 2019. All Rights Reserved. This may not be publicly shared without explicit permission.
At this point, what we would like to do is to pull the $U$s out of the inner matrix and stick them with the $R$s in the outer matrices. Then we would be done. This could be done immediately because of the properties of block matrix multiplication, or we could further justify it.

If you wanted to further justify it (this was not required for full credit), we could calculate as follows:

$$
\begin{bmatrix}
\vec{v}_0 & R
\end{bmatrix}
\begin{bmatrix}
\lambda_0 & 0^T \\
0 & U \Lambda U^T
\end{bmatrix}
\begin{bmatrix}
\vec{v}_0^T \\
R^T
\end{bmatrix}
= (\lambda_0 \vec{v}_0 \ O_{(n\times n-1)}) + (0 \ RU \Lambda^T)
\begin{bmatrix}
\vec{v}_0^T \\
R^T
\end{bmatrix},
$$

where $O_{(n\times n-1)}$ is a matrix filled with zeros.

Then

$$S = \begin{bmatrix}
\lambda_0 \vec{v}_0^T \\
0
\end{bmatrix} + [RU \Lambda^T R^T].$$

Then we can place $S$ into the orthonormal basis $W = [\vec{v}_0 \ RU]$ to verify:

$$S = \begin{bmatrix}
\vec{v}_0 & RU
\end{bmatrix}
\begin{bmatrix}
\lambda_0 & 0^T \\
0 & U^T R^T
\end{bmatrix}
= \begin{bmatrix}
\lambda_0 \vec{v}_0 & RU \Lambda
\end{bmatrix}
\begin{bmatrix}
\vec{v}_0^T \\
U^T R^T
\end{bmatrix}
= \begin{bmatrix}
\lambda_0 \vec{v}_0 & O_{(n\times n-1)} + 0 \ RU \Lambda
\end{bmatrix}
\begin{bmatrix}
\vec{v}_0^T \\
U^T R^T
\end{bmatrix},$$

which also becomes

$$\begin{bmatrix}
\lambda_0 \vec{v}_0 \vec{v}_0^T \\
0
\end{bmatrix} + [RU \Lambda^T R^T].$$

Regardless of whether we felt required to justify pulling the $U$s out, we know that the multiplication of two orthonormal matrices (in this case, $R$ and $U$) is an orthonormal matrix as well. Why? Because $(RU)^T (RU) = U^T R^T RU = U^T IU = U^T U = I$.

It is easy to further verify that $\vec{v}_0$ is orthonormal to all columns of $RU$ by seeing what happens to $\vec{v}_0^T RU$: $\vec{v}_0^T RU = (\vec{v}_0^T R)U = \vec{0}^T U = \vec{0}^T$, because $\vec{0}$ is orthogonal to column vectors of $R$. Therefore, we can now define the orthonormal basis $W$ as,

$$W = \begin{bmatrix}
\vec{v}_0 & RU
\end{bmatrix}$$

This shows what we wanted to prove:

$$S = W \begin{bmatrix}
\lambda_0 & 0^T \\
0 & \Lambda
\end{bmatrix} W^T$$

and we are done.

By induction, we are now done since we have proved that having the desired property for $n - 1$ implies that we have the property for $n$ and we also have a valid base case at $n = 1$.

According to the base case and inductive steps we just proved, the statement, “every real symmetric matrix is diagonalized by a matrix of its real orthonormal eigenvectors” is proved by induction.

**Solution:** This is called the spectral theorem for real symmetric matrices. The proof here also has this nice recursive character to it that goes along well with what you have learned in our sister courses 61ab.

You will get more practice doing inductive proofs in our successor course 70.
2. The Moore-Penrose pseudoinverse for “wide” matrices

Say we have a set of linear equations given by \( A\vec{x} = \vec{y} \). If \( A \) is invertible, we know that the solution for \( \vec{x} \) is \( \vec{x} = A^{-1}\vec{y} \). However, what if \( A \) is not a square matrix? In 16A, you saw how this problem could be approached for tall “standing up” matrices \( A \) where it really wasn’t possible to find a solution that exactly matches all the measurements, using linear least-squares. The linear least-squares solution gives us a reasonable answer that asks for the “best” match in terms of reducing the norm of the error vector.

This problem deals with the other case — when the matrix \( A \) is wide and “lying down” — with more columns than rows. In this case, there are generally going to be lots of possible solutions — so which should we choose? Why? We will walk you through the **Moore-Penrose pseudoinverse** that generalizes the idea of the matrix inverse and is derived from the singular value decomposition.

This approach to finding solutions complements the OMP approach that you learned in 16A and that we used earlier in 16B in the context of outlier removal during system identification.

(a) Say you have the matrix

\[
A = \begin{bmatrix}
1 & -1 & 1 \\
1 & 1 & -1
\end{bmatrix}.
\]

To find the Moore-Penrose pseudoinverse we start by calculating the SVD of \( A \). That is to say, calculate \( U, \Sigma, V \) such that

\[ A = U\Sigma V^T \]

where \( U \) and \( V \) are orthonormal matrices.

Here we will give you that the decomposition of \( A \) is:

\[
A = \begin{bmatrix}
\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\
\frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}}
\end{bmatrix} \begin{bmatrix}
2 & 0 & 0 \\
0 & \sqrt{2} & 0
\end{bmatrix} \begin{bmatrix}
0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\
-1 & 0 & 0 \\
0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}
\end{bmatrix}
\]

where:

\[
U = \begin{bmatrix}
\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\
\frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}}
\end{bmatrix},
\]

\[
\Sigma = \begin{bmatrix}
2 & 0 & 0 \\
0 & \sqrt{2} & 0
\end{bmatrix},
\]

\[
V^T = \begin{bmatrix}
0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\
1 & 0 & 0 \\
0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}}
\end{bmatrix}
\]

It is a good idea to be able to calculate the SVD yourself as you may be asked to solve similar questions on your own in the exam.

**Solution:** Though you did not have to do any work for this sub-part the following solutions will walk you through how to solve for the SVD:

\[ A = U\Sigma V^T \]

\[ AA^\top = \begin{bmatrix}
3 & -1 \\
-1 & 3
\end{bmatrix}. \]
Which has characteristic polynomial $\lambda^2 - 6\lambda + 8 = 0$, producing eigenvalues 4 and 2. Solving $Av = \lambda_i v$ produces eigenvectors $\left[ \frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}} \right]^T$ and $\left[ \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \right]^T$ associated with eigenvalues 4 and 2 respectively. The singular values are the square roots of the eigenvalues of $AA^\top$, so

$$\Sigma = \begin{bmatrix} 2 & 0 & 0 \\ 0 & \sqrt{2} & 0 \end{bmatrix}$$

and

$$U = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix}$$

We can then solve for the $v$ vectors using $A^\top \tilde{u}_i = \sigma_i \tilde{v}_i$, producing $\tilde{v}_1 = [0, -\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]^T$ and $\tilde{v}_2 = [1, 0, 0]^T$. The last $\tilde{v}$ must be orthonormal to the other two, so we can pick $[0, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]^T$.

The SVD is:

$$A = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 2 & 0 & 0 \\ -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 & \sqrt{2} & 0 \end{bmatrix} \begin{bmatrix} 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 1 & 0 & 0 \\ \frac{1}{\sqrt{2}} & 0 & 1 & \frac{1}{\sqrt{2}} \end{bmatrix}$$

Let us now think about what the SVD does. Let us look at matrix $A$ acting on some vector $\tilde{x}$ to give the result $\tilde{y}$. We have

$$A\tilde{x} = U\Sigma V^T \tilde{x} = \tilde{y}.$$ 

Observe that $V^T\tilde{x}$ rotates the vector, $\Sigma$ scales the result, and $U$ rotates it again. We will try to "reverse" these operations one at a time and then put them together to construct the Moore-Penrose pseudoinverse.

**If $U$ "rotates" the vector $(\Sigma V^T) \tilde{x}$, what operator can we derive that will undo the rotation?**

**Solution:** By orthonormality, we know that $U^T U = UU^T = I$. Therefore, $U^T$ undoes the rotation.

**(b) Derive a matrix that will "unscale", or undo the effect of $\Sigma$ where it is possible to undo.** Recall that $\Sigma$ has the same dimensions as $A$. Ignore any division by zeros (that is to say, let it stay zero).

**Solution:** If you observe the equation:

$$\Sigma \tilde{x} = U^T \tilde{y} = \tilde{y},$$

you can see that $\sigma_i x_i = \tilde{y}_i$ for $i = 0, ..., m - 1$, which means that to obtain $x_i$ from $y_i$, we need to multiply $y_i$ by $\frac{1}{\sigma_i}$. For any $i > m - 1$, the information in $x_i$ is lost by multiplying with 0. If the corresponding $\tilde{y}_i \neq 0$, there is no way of solving this equation. No solution exists, and we have to accept an approximate solution. If the corresponding $\tilde{y}_i = 0$, then any $x_i$ would still work. Either way, it is reasonable to just say $x_i = 0$ in the case that $\sigma_i = 0$. That’s why we can legitimately pad 0s in the bottom of $\Sigma$ given below:

If $\Sigma = \begin{bmatrix} \sigma_0 & 0 & 0 & 0 & 0 & \ldots & 0 \\ 0 & \sigma_1 & 0 & 0 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & 0 & \sigma_{m-1} & 0 & \ldots & 0 \end{bmatrix}$, then $\bar{\Sigma} = \begin{bmatrix} \frac{1}{\sigma_0} & 0 & \ldots & 0 \\ 0 & \frac{1}{\sigma_1} & \ldots & 0 \\ \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \ldots & \frac{1}{\sigma_{m-1}} \\ 0 & 0 & \ldots & 0 \end{bmatrix}$

This $\bar{\Sigma}$ matrix undoes the effect of $\Sigma$ where that effect can be undone.
(c) Derive an operator that would "unrotate" by $V^T$.

**Solution:** By orthonormality, we know that $V^TV = VV^T = I$. Therefore, $V$ undoes the rotation.

(d) Try to use this idea of "unrotating" and "unscaling" to derive an "inverse", denoted as $A^\dagger$. That is to say,

$$\bar{x} = A^\dagger \bar{y}$$

The reason why the word inverse is in quotes (or why this is called a pseudo-inverse) is because we’re ignoring the "divisions" by zero.

**Solution:** We can use the unrotation and unscaling matrices we derived above to "undo" the effect of $A$ and get the required solution. Of course, nothing can possibly be done for the information that was destroyed by the nullspace of $A$ — there is no way to recover any component of the true $\bar{x}$ that was in the nullspace of $A$. However, we can get back everything else.

$$\begin{align*}
\bar{y} & = A\bar{x} = USV^T\bar{x} \\
U^T\bar{y} & = \Sigma V^T \bar{x} \quad \text{Unrotating by $U$} \\
\Sigma U^T\bar{y} & = V\bar{x} \quad \text{Unscaling by $\Sigma$} \\
V\Sigma U^T\bar{y} & = \bar{x} \quad \text{Unrotating by $V$}
\end{align*}$$

Therefore, we have $A^\dagger = V\Sigma U^T$, where $\Sigma$ is given in (c).

(e) Use $A^\dagger$ to solve for a vector $\bar{x}$ in the following system of equations.

$$\begin{bmatrix}
1 & -1 & 1 \\
1 & 1 & -1
\end{bmatrix} \begin{bmatrix}
\bar{x}
\end{bmatrix} = \begin{bmatrix}
2 \\
4
\end{bmatrix}$$

**Solution:** From the above, we have the solution given by:

$$\begin{align*}
\bar{x} & = A^\dagger \bar{y} = V\Sigma U^T\bar{y} \\
& = \begin{bmatrix}
0 & 1 & 0 \\
-\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}}
\end{bmatrix}
\begin{bmatrix}
\frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}}
\end{bmatrix}
\begin{bmatrix}
2 \\
4
\end{bmatrix} \\
& = \begin{bmatrix}
\frac{3}{2} \\
\frac{1}{2} \\
-\frac{1}{2}
\end{bmatrix}
\end{align*}$$

Therefore, a reasonable solution to the system of equations is:

$$\bar{x} = \begin{bmatrix}
\frac{3}{2} \\
\frac{1}{2} \\
-\frac{1}{2}
\end{bmatrix}$$

(f) Now we will see why this matrix is a useful proxy for the matrix inverse in such circumstances. Show that the solution given by the Moore-Penrose pseudoinverse satisfies the minimality property that if $\tilde{x}$ is the pseudo-inverse solution to $A\tilde{x} = \bar{y}$, then $\|\tilde{x}\| \leq \|\bar{z}\|$ for all other vectors $\bar{z}$ satisfying $A\bar{z} = \bar{y}$. (Hint: look at the vectors involved in the $V$ basis. Think about the relevant nullspace and how it is connected to all this.)

© UCB EECS 16B, Fall 2019. All Rights Reserved. This may not be publicly shared without explicit permission.
This minimality property is useful in many applications. You saw a control application in lecture. You’ll see a communications application in another problem. This is also used all the time in machine learning, where it is connected to the concept behind what is called ridge regression or weight shrinkage.

Solution: Since \( \hat{x} \) is the pseudoinverse solution, we know that,

\[
\hat{x} = V \Sigma U^T \hat{y}
\]

Let us write down what \( \hat{x} \) is with respect to the orthonormal basis formed by the columns of \( V \). Let there be \( k \) non-zero singular values. The following expression comes from expanding the matrix multiplication.

\[
\hat{x} \left|_V \right. = V^T \hat{x}
\]

\[
= V^T A^+ \hat{y} = V^T \Sigma U^T \hat{y} = \Sigma U^T \hat{y}
\]

\[
= \left[ \frac{\hat{u}_0^T \hat{y}}{\sigma_0}, \frac{\hat{u}_1^T \hat{y}}{\sigma_1}, \ldots, \frac{\hat{u}_{k-1}^T \hat{y}}{\sigma_{k-1}}, 0, \ldots, 0 \right]^T
\]

The \( n - k \) zeros at the end come from the fact that there are only \( k \) non-zero singular values. Therefore, by construction, \( \hat{x} \) is a linear combination of the first \( k \) columns of \( V \).

Since any other \( \tilde{z} \) is also a solution to the original problem, we have

\[
A \tilde{z} = U \Sigma V^T \tilde{z} = U \Sigma \left|\tilde{z}\right|_V = \hat{y},
\]

where \( \left|\tilde{z}\right|_V \) is the projection of \( \tilde{z} \) in the \( V \) basis. Using the idea of “unscaling” for the first \( k \) elements (where the unscaling is clearly invertible) and “unrotation” after that, we see that the first \( k \) elements of \( \tilde{z}|_V \) must be identical to those first \( k \) elements of \( \hat{x}|_V \).

However, since the information for the last \( n - k \) elements of \( \tilde{z}|_V \) is lost by multiplying 0s, any values \( \alpha_i \) there are unconstrained as weights on the last part of the \( V \) basis — namely the weights on the basis for the nullspace of \( A \). Therefore,

\[
\tilde{z}|_V = \left[ \frac{\hat{u}_0^T \hat{y}}{\sigma_0}, \frac{\hat{u}_1^T \hat{y}}{\sigma_1}, \ldots, \frac{\hat{u}_{k-1}^T \hat{y}}{\sigma_{k-1}}, \alpha_k, \alpha_{k+1}, \ldots, \alpha_{n-1} \right]^T
\]

Now, since the columns of \( V \) are orthonormal, observe that,

\[
||\hat{x}||^2 = \sum_{i=0}^{k-1} \left| \frac{\hat{u}_i^T \hat{y}}{\sigma_i} \right|^2
\]

and that,

\[
||\tilde{z}||^2 = \sum_{i=0}^{k-1} \left| \frac{\hat{u}_i^T \hat{y}}{\sigma_i} \right|^2 + \sum_{i=k}^{n-1} |\alpha_i|^2
\]

Therefore,

\[
||\tilde{z}||^2 = ||\hat{x}||^2 + \sum_{i=k}^{n-1} |\alpha_i|^2
\]

This tells us that,

\[
||\tilde{z}|| \geq ||\hat{x}||
\]
(g) Consider a generic wide matrix $A$. We know that $A$ can be written using $A = U\Sigma V^T$ where $U$ and $V$ each are the appropriate size and have orthonormal columns, while $\Sigma$ is the appropriate size and is a diagonal matrix — all off-diagonal entries are zero. Further assume that the rows of $A$ are linearly independent. **Prove that** $A^\dagger = A^T (AA^T)^{-1}$.

*(HINT: Just substitute in $U\Sigma V^T$ for $A$ in the expression above and simplify using the properties you know about $U, \Sigma, V$. Remember the transpose of a product of matrices is the product of their transposes in reverse order: $(CD)^T = D^TC^T$.)

**Solution:**

We just substitute in to see what happens:

\[
A^T (AA^T)^{-1} = (U\Sigma V^T)^T (U\Sigma V^T (U\Sigma V^T)^T)^{-1} \tag{3}
\]

\[
= V\Sigma^T U^T (U\Sigma V^T V\Sigma^T U^T)^{-1} \tag{4}
\]

\[
= V\Sigma^T U^T (U(\Sigma\Sigma^T)U^T)^{-1} \tag{5}
\]

\[
= V\Sigma^T U^T U(\Sigma\Sigma^T)^{-1}U^T \tag{6}
\]

\[
= V\Sigma^T (\Sigma\Sigma)^{-1}U^T. \tag{7}
\]

At this point, we are almost done in reaching $A^\dagger = V\tilde{\Sigma}U^T$. We have the leading $V$ and the ending $U^T$. All that we need to do is multiply out the diagonal matrices in the middle.
This concludes the proof.

In case you were wondering, the alternative form $A^T (AA^T)^{-1}$ comes from first noticing that the columns of $A^T$ are all orthogonal to the nullspace of $A$ by definition, and so using them as the basis for the subspace in which we want to find the solution. The $(AA^T)$ are the columns of where each of these basis elements ends up through $A$. Inverting this tells us how to get to where we want.

Anyway, it is interesting to step back and see that at this point, between 16A and 16B, you now know two different ways to solve problems in which there are fewer linear equations than you have unknowns. You learned OMP in 16A which proceeded in a greedy fashion and basically tried to minimize the number of variables that it set to anything other than zero. And now you have learned the Moore-Penrose Pseudoinverse that finds the solution that minimizes the Euclidean norm.

Both of these, as well as least-squares, are ways to manifest the philosophical principle of Occam’s Razor algorithmically for learning. Occam’s Razor says “ Nunquam ponenda est pluralitas sine necessitate” (translation: don’t posit more than you need.) But there are two different ways to measure “more” — counting and weighing. Least squares (when we have more equations than variables) is on
the path of weighing — the norm of the error is minimized. OMP iterates that to also follow the path of counting, where the number of nonzero variables corresponds to the things that are counted. The Moore-Penrose Pseudoinverse is fully in the path of weighing.

Both of these paths grow into major themes in machine learning generally, and both play a very important role in modern machine learning in particular. This is because in many contemporary approaches to machine learning, we try to learn models that have more parameters than we have data points.

3. Using upper-triangularization to solve differential equations

You know that for any square matrix $A$ with real eigenvalues, there exists a real matrix $V$ with orthonormal columns and a real upper triangular matrix $R$ so that $A = VRV^T$. In particular, to set notation explicitly:

$$V = \begin{bmatrix} \vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n \end{bmatrix}$$
$$R^T = \begin{bmatrix} \vec{r}_1, \vec{r}_2, \ldots, \vec{r}_n \end{bmatrix}$$

where the rows of the upper-triangular $R$ look like

$$\vec{r}_1^T = [\lambda_1, r_{1,2}, r_{1,3}, \ldots, r_{1,n}]$$
$$\vec{r}_2^T = [0, \lambda_2, r_{2,3}, r_{2,4}, \ldots, r_{2,n}]$$
$$\vec{r}_i^T = [0, \ldots, 0, \lambda_i, r_{i,i+1}, r_{i,i+2}, \ldots, r_{i,n}]$$
$$i-1 \text{ times}$$
$$\vec{r}_n^T = [0, \ldots, 0, \lambda_n]$$
$$n-1 \text{ times}$$

where the $\lambda_i$ are the eigenvalues of $A$.

Here, we also use bracket notation to index into vectors so

$$r_i[k] = \begin{cases} 0 & \text{if } k < i \\ \lambda_i & \text{if } k = i \\ r_{i,k} & \text{if } k > i \end{cases}$$

Note: we use 1-indexing so the first entry has index 1.

Suppose our goal is to solve the $n$-dimensional system of differential equations written out in vector/matrix form as:

$$\frac{d}{dt} \vec{x}(t) = A\vec{x}(t) + \vec{u}(t),$$
$$\vec{x}(0) = \vec{x}_0,$$

where $\vec{x}_0$ is a specified initial condition and $\vec{u}(t)$ is a given vector of functions of time.

Assume that the $V$ and $R$ have already been computed and are accessible to you using the notation above.

Assume that you have access to a function $\text{ScalarSolve}(\lambda, y_0, \vec{u})$ that takes a real number $\lambda$, a real number $y_0$, and a real-valued function of time $\vec{u}$ as inputs and returns a real-valued function of time that is the solution to the scalar differential equation $\frac{d}{dt}y(t) = \lambda y(t) + \vec{u}(t)$ with initial condition $y(0) = y_0$.

Also assume that you can do regular arithmetic using real-valued functions and it will do the right thing. So if $u$ is a real-valued function of time, and $g$ is also a real-valued function of time, then $5u + 6g$ will be a real valued function of time that evaluates to $5u(t) + 6g(t)$ at time $t$. 

© UCB EECS 16B, Fall 2019. All Rights Reserved. This may not be publicly shared without explicit permission.
Use $V,R$ to construct a procedure for solving this differential equation

$$\frac{d}{dt}\vec{x}(t) = A\vec{x}(t) + \vec{u}(t),$$

$$\vec{x}(0) = \vec{x}_0,$$

for $\vec{x}(t)$ by filling in the following template in the spots marked ♣, ◇, ▽, ♢.

(HINT: Think back to the RLC circuit on homework 3. You also might want to write out for yourself what the differential equation looks like in $V$-coordinates.)

1: $\vec{x}_0 = V^T\vec{x}_0$ $\triangleright$ Change the initial condition to be in $V$-coordinates
2: $\vec{u} = V^T\vec{u}$ $\triangleright$ Change the external input functions to be in $V$-coordinates
3: for $i = n$ down to 1 do $\triangleright$ Iterate up from the bottom
4: $\vec{u}_i = ♣ + \sum_{j=i+1}^n ♢$ $\triangleright$ Make the effective input for this level
5: $\vec{x}_i = ScalarSolve(◇, \vec{x}_0[i], \vec{u}_i)$ $\triangleright$ Solve this level’s scalar differential equation
6: end for
7: $\vec{x}(t) = \heartsuit \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}(t)$ $\triangleright$ Change back into original coordinates

(a) Give the expression for ◇ on line 7 of the algorithm above. (i.e. How do you get from $\vec{x}(t)$ to $\vec{x}(t)$?)

Solution: Since $\vec{x}_0 = V^T\vec{x}_0$ we know we are changing to $V$-basis. So, the implicit change of variable that we are doing is $\vec{x} = V^T\vec{x}$, this means that to come back, $\vec{x} = V\vec{x}$.
Thus, ◇ = $V$.

(b) Give the expression for ◇ on line 5 of the algorithm above. (i.e. What are the $\lambda$ arguments to $ScalarSolve$ for the $i$-th iteration of the for-loop?)

Solution: We begin with $\frac{d}{dt}\vec{x}(t) = A\vec{x}(t) + \vec{u}(t) = VRV^T\vec{x}(t) + \vec{u}(t)$. After pre-multiplying both sides in $\frac{d}{dt}\vec{x}(t) = VRV^T\vec{x}(t) + \vec{u}(t)$ by $V^T$, we get $V^T\frac{d}{dt}\vec{x}(t) = RV^T\vec{x}(t) + V^T\vec{u}(t)$ (since $V^TV = I$ because $V$ is orthonormal).

Now, we perform change of variables, $\vec{x} = V^T\vec{x}$ and $\vec{u} = V^T\vec{u}$ so we get,

$$\frac{d}{dt} \vec{x}_i(t) = R\vec{x}_i(t) + \vec{u}_i(t)$$

Thus, the $i^{th}$ equation in this system is,

$$\frac{d}{dt} \vec{x}_i(t) = \sum_{j=1}^{i-1} r_{i,j} \vec{x}_i(t) + r_{i,i+1} \vec{u}_i(t)$$

Using, $\vec{r}_i = [0, \ldots, 0, \lambda_i, r_{i,i+1}, r_{i,i+2}, \ldots, r_{i,n}]$ we get,

$$\frac{d}{dt} \vec{x}_i(t) = \lambda_i \vec{x}_i(t) + r_{i,i+1} \vec{x}_{i+1}(t) + r_{i,i+2} \vec{x}_{i+2}(t) + \cdots + r_{i,n} \vec{x}_n(t) + \vec{u}_i(t)$$

Thus, $\frac{d}{dt} \vec{x}_i(t) = \lambda_i \vec{x}_i(t) + \vec{u}_i(t) + \sum_{j=i+1}^n r_{i,j} \vec{x}_j(t)$

We remember that the $\lambda_i$ is in the $i$-th position of $\vec{r}_i$ or $r_i[i]$ using our bracket-indexing notation.

Hence, ◇ = $r_i[i]$ since the code has to run in terms of the given $V$ and $R$.

(c) Give the expression for ♣ on line 4 of the algorithm above.

Solution:

Since, from above, $\frac{d}{dt} \vec{x}_i(t) = \lambda_i \vec{x}_i(t) + \vec{u}_i(t) + \sum_{j=i+1}^n r_{i,j} \vec{x}_j(t)$ we can see that the $\vec{u}_i$ is the input term that does not depend on the inner sum. From this we conclude that

♣ = $\vec{u}_i$.

Remember, we want a function of time. We can also index using the bracket notation $\vec{u}[i]$. 
(d) Give the expression for ♠ on line 4 of the algorithm above.

Solution:
Since, from above, \( \frac{d}{dt}\vec{x}(t) = \lambda_i\vec{x}_i(t) + \vec{u}_i(t) + \sum_{j=i+1}^n r_{i,j}\vec{x}_j(t) \) and so we know what is inside the inner sum:
\[ ♠ = r_{i,j}\vec{x}_j. \]
Here we can also use the bracket indexing notation \( r_i[j]\vec{x}[j] \).
Congratulations! You now know how to systematically solve any system of differential equations with constant coefficients, as long as you know how to solve the scalar case with inputs. The same argument style applies for recurrence relations. The only gap that remains is the assumption that all the eigenvalues are real, but now that you understand orthogonality for complex vectors, you can also update your understanding of upper-triangularization to allow for complex matrices as well.

(e) Let us complete the algorithm by investigating how \( \text{ScalarSolve}(\lambda, y_0, \vec{u}) \) works.

Consider an input that is a weighted sum of polynomials times exponentials.
\[
\vec{u}(t) = \sum_{k=1}^N \alpha[k]t^\beta[k]e^{\gamma[k]t}
\]
Here, the \( \alpha[k] \) are real constants, the \( \beta[k] \) are non-negative integer powers, and the \( \gamma[k] \) are real exponents. Assume that \( \alpha, \beta, \gamma \) are all lists of the same size \( N \).

What function should \( \text{ScalarSolve}(\lambda, y_0, \vec{u}) \) return for the above \( \vec{u} \)? Express the answer in terms of new lists \( \alpha'[k], \beta'[k], \gamma'[k] \) and a procedure to construct them.

(Hint: Recall the integral solution from HW 2 and consider integration by parts. Now, walk down the original lists and build your new lists as you go. You are going to have to deal with the case that \( \lambda \) equals the relevant \( \gamma \) entry differently from how you deal with the case where \( \lambda \) doesn’t equal that \( \gamma \). Finally, don’t forget about the initial condition.)

Solution: \( \text{ScalarSolve}(\lambda, y_0, \vec{u}) \) should return the solution to the differential equation

\[ \frac{d}{dt}y(t) = \lambda y(t) + \vec{u}(t) \]  \hspace{1cm} (12)

with initial condition \( y(0) = y_0 \).

Recall from HW 2 the following general integral solution to such a differential equation

\[ y(t) = y_0e^{\lambda t} + \int_0^t \vec{u}(\tau)e^{\lambda(t-\tau)}d\tau \]  \hspace{1cm} (13)

Let us consider a particular case for \( \vec{u}(t) \): \( \vec{u}(t) = \alpha t^\beta e^{\gamma t} \) where \( \alpha \) and \( \gamma \) are real, and \( \beta \) is a non-negative integer.

Plugging this \( \vec{u}(t) \) into \( \{13\} \) yields

\[ y(t) = y_0e^{\lambda t} + \int_0^t \alpha \tau^\beta e^{\gamma t}e^{\lambda(t-\tau)}d\tau \]
\[ = y_0e^{\lambda t} + \alpha e^{\lambda t} \int_0^t \tau^\beta e^{\gamma t - \lambda \tau}d\tau \]  \hspace{1cm} (14)

There are now two cases: if \( \gamma = \lambda \) and if \( \gamma \neq \lambda \).

Case 1: \( \gamma = \lambda \)
In this case, we can simplify (15) to

\[
y(t) = y_0 e^{\lambda t} + \alpha e^{\lambda t} \int_0^t \tau^\beta d\tau
\]

(16)

\[
y(t) = y_0 e^{\lambda t} + \alpha e^{\lambda t} \left[ \frac{1}{\beta + 1} \tau^{\beta+1} \right] \bigg|_{\tau=0}^{\tau=t}
\]

(17)

\[
y(t) = y_0 e^{\lambda t} + \alpha e^{\lambda t} \left( \frac{\tau^{\beta+1}}{\beta + 1} \right) \bigg|_{\tau=0}^{\tau=t}
\]

(18)

Notice that in this case, we increment the power of \( t \) — that is what happens when we encounter this exact \( \lambda \) that matches the input exponential.

**Case 2: \( \gamma \neq \lambda \)**

In this case, we must use integration by parts to solve for the integral in (15). It would have been fine if you had just looked up the formula. But for completeness, we show the procedure from integral calculus.

Using the tabular integration method, one can solve for \( \int F(t)G(t)\,dt \) by creating a table where the function \( F(t) \) is successively differentiated on the left column, and the function \( G(t) \) is successively integrated in the right column. Every other entry is negated in the first column, and finally, take the sum of the entries from the first column times the entries in the second column that are one row below.

Considering the integral from (15):

\[
\mathcal{I} = \int_0^t \tau^\beta e^{(\gamma - \lambda)\tau} d\tau
\]

Writing the summation yields

\[
\mathcal{I} = \sum_{m=0}^\beta (-1)^m \frac{\beta!}{(\beta-m)!} \frac{1}{(\gamma - \lambda)(1+m)} \tau^{\beta-m} e^{(\gamma - \lambda)\tau}
\]

(19)

We must evaluate \( \mathcal{I} \) at the integration bounds \( \tau = 0 \) and \( \tau = t \), which yields

\[
\mathcal{I}_{\text{def}} = \sum_{m=0}^\beta (-1)^m \frac{\beta!}{(\beta-m)!} \frac{1}{(\gamma - \lambda)(1+m)} t^{\beta-m} e^{(\gamma - \lambda)t}
\]

(20)

Notice that this case doesn’t spawn any power of \( t \) that is higher than the original \( \beta \) that we started with, although lower powers of \( t \) can be spawned in this process. The only way to get higher powers is to encounter the exact same \( \lambda = \gamma \).
We use the integral we found $\mathcal{I}_{def}$ to find the solution for (15) as:

$$y(t) = y_0 e^{\lambda t} + \alpha e^{\lambda t} \left( \sum_{m=0}^{\beta} (-1)^m \frac{\beta!}{(\beta-m)! (\gamma-\lambda)^{(1+m)}} t^{\beta-m} e^{(\gamma-\lambda)} \right)$$

(21)

$$= y_0 e^{\lambda t} + \alpha \left( \sum_{m=0}^{\beta} (-1)^m \frac{\beta!}{(\beta-m)! (\gamma-\lambda)^{(1+m)}} t^{\beta-m} e^{\gamma t} \right)$$

(22)

So far, we have found the output of $\text{ScalarSolve}$ for a particular $\bar{u} = \alpha \beta e^{\gamma t}$. We now need to consider when $\bar{u}$ is a sum of polynomials and exponentials. When $\bar{u}$ is a sum, we will get the same form of solution as in cases 1 or 2 (depending on whether $\gamma = \lambda$). Thus, for each $\alpha[k]$, $\beta[k]$, $\gamma[k]$ triplet, write out the solutions to that particular differential equation as either (18) or (22), depending on if $\gamma[k] = \lambda$ or $\gamma[k] \neq \lambda$ and add up all the terms.

Finally, do not forget the initial condition term, which is shared across each of the particular solutions.

The final equation that $\text{ScalarSolve}$ should return is:

$$y(t) = y_0 e^{\lambda t} + \sum_{k=1}^{N} \left\{ \frac{\alpha[k]}{\beta[k]+1} t^{\beta[k]+1} e^{\lambda t} \right\}$$

$$\gamma[k] = \lambda$$

$$\gamma[k] \neq \lambda$$

(23)

Notice here that the total number of terms in the sum can grow. But it cannot grow that drastically. This is because we can always fold together terms that have the same power of $t$ and the same exponent $\gamma$. Notice also that no new $\gamma$’s can be spawned beyond those in the original input and the $\lambda$ that come from the eigenvalues.

The approach here is completely algorithmic and leans on the linear-algebra of upper-triangulation.

In later courses (like 120), you will learn other techniques to get the same solutions that rely on complex analysis based approaches called Laplace Transforms. The overall work is the same in both cases. The advantage to the given approach is just that the proof/derivation is entirely elementary. Meanwhile, the Laplace Transform approach needs to rely on the uniqueness of Laplace Transforms which requires the techniques of Math 185 and beyond to establish.

4. Weighted minimum norm

You saw in lecture in the context of open-loop control, how we consider problems in which we have a wide matrix $A$ and solve $Ax = \bar{y}$ such that $\bar{x}$ is a minimum norm solution:

$$\|\bar{x}\| \leq \|\bar{z}\|$$

for all $\bar{z}$ such that $A\bar{z} = \bar{y}$. You then saw this idea again earlier in this HW where you saw how to compute the appropriate “pseudo-inverse” for such wide matrices.

But what if you weren’t interested in just the norm of $\bar{x}$? What if you instead cared about minimizing the norm of a linear transformation $C\bar{x}$? For example, suppose that controls were more or less costly at different times.

The problem can be written out mathematically as:

Given a wide matrix $A$ and a matrix $C$ find $\bar{x}$ such that $A\bar{x} = \bar{y}$ and $\|C\bar{x}\| \leq \|C\bar{z}\|$ for all $\bar{z}$ such that $A\bar{z} = \bar{y}$.
(a) Let’s start with the case of $C$ being invertible. **Solve this problem** (i.e. find the optimal $\bar{x}$ with the minimum $\|C\bar{x}\|$) for the specific matrices and $\bar{y}$ given below. Show your work.

It is fine to leave your answer as an explicit product of matrices and vectors.

*(HINT: You might want to change variables to solve this problem. Don’t forget to change back!)*

We have

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}, \quad C = \begin{bmatrix} 0 & 0 & 2 \\ 0 & 1 & 0 \\ 2 & 0 & 0 \end{bmatrix}, \quad \bar{y} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

For convenience, $C^{-1} = \begin{bmatrix} 0 & 0 & 0.5 \\ 0 & 1 & 0 \\ 0.5 & 0 & 0 \end{bmatrix}$ and you are also given some SVDs on the following page.

Using these SVDs, we have:

$$A = (U_A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix})(\Sigma_A = \begin{bmatrix} \sqrt{2} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix})(V_A^T = \begin{bmatrix} 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 1 & 0 & 0 \\ 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}) \quad (24)$$

$$C = (U_C = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix})(\Sigma_C = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix})(V_C^T = \begin{bmatrix} 0 & 0 & -1 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}) \quad (25)$$

$$AC = (U_{AC} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix})(\Sigma_{AC} = \begin{bmatrix} \sqrt{5} & 0 & 0 \\ 0 & 2 & 0 \end{bmatrix})(V_{AC}^T = \begin{bmatrix} \frac{2}{\sqrt{5}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{5}} & \frac{1}{\sqrt{5}} \\ 0 & \frac{2}{\sqrt{5}} & 0 \end{bmatrix}) \quad (26)$$

$$AC^{-1} = (U_{AC^{-1}} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix})(\Sigma_{AC^{-1}} = \begin{bmatrix} \frac{\sqrt{3}}{2} & 0 & 0 \\ 0 & 0.5 & 0 \end{bmatrix})(V_{AC^{-1}}^T = \begin{bmatrix} \frac{\sqrt{3}}{2} & 0 & 0 \\ 0 & 0 & 1 \\ -\frac{1}{\sqrt{3}} & \frac{\sqrt{3}}{2} & 0 \end{bmatrix}) \quad (27)$$

**Solution:** In lecture and earlier problems on this HW, you solved a similar problem $A\bar{x} = \bar{y}$ such that $\bar{x}$ is a minimum norm solution: $\|\bar{x}\| \leq \|\bar{z}\|$ for any $\bar{z}$ that satisfies $A\bar{z} = \bar{y}$.

When you solved this problem, you computed the appropriate psuedoinverse to solve for $\bar{x}$. This was the Moore Penrose psuedo inverse — sometimes depicted as $A^\dagger$.

Seeing that we already know how to solve such problems, we can first try to reformulate the current problem: $A\bar{x} = \bar{y}$ such that $\|C\bar{x}\| \leq \|C\bar{z}\|$, into the problem that we already know how to solve. To do this we can do a change of variables (as the hint told us to do). Using the change of variables $\bar{x} = C\bar{z}$ and $\bar{p} = C\bar{z}$ we get the new constraint: $\|\bar{x}\| \leq \|\bar{p}\|$ for any vector $\bar{p}$ that satisfies something. What is this something?

Originally, we had $A\bar{x} = \bar{y}$ and so in the changed variables, we have $\bar{x} = C^{-1}\bar{z}$ and so the constraint that needs to be satisfied is $AC^{-1}\bar{x} = \bar{y}$.

So our new problem is to solve $AC^{-1}\bar{x} = \bar{y}$ such that $\bar{x}$ is a minimum norm solution: $\|\bar{x}\| \leq \|\bar{p}\|$ for all $\bar{p}$ that satisfy $AC^{-1}\bar{p} = \bar{y}$.

This is exactly like the minimum norm question earlier in this homework except now the matrix multiplying the vector is $AC^{-1}$. 
To solve this we proceed exactly like we did earlier and find the Moore-Penrose psuedo inverse of $AC^{-1}$:

$$\bar{x} = V_{\text{compact},AC^{-1}}\Sigma_{\text{compact},AC^{-1}}^{-1}U_{AC^{-1}}^T\bar{y}$$  

(28)

where here, we need to be using the compact form of the SVD vis-a-vis $AC^{-1}$. Why compact? We need the $\Sigma$ matrix to be square so we can invert it. This just means that we drop the parts of $V^T$ that are just a basis for the nullspace of $AC^{-1}$ — the last row. To be explicit, the compact SVD is:

$$AC^{-1} = (U_{AC^{-1}} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix})(\Sigma_{\text{compact},AC^{-1}} = \begin{bmatrix} \sqrt{5} & 0 \\ 0 & 0.5 \end{bmatrix})(V_{\text{compact},AC^{-1}} = \begin{bmatrix} 1/\sqrt{2} & 2/\sqrt{5} & 0 \\ 0 & 0 & 1 \end{bmatrix}).$$  

(29)

Calculating this out, we get that $\bar{x} = \begin{bmatrix} 2 \\ 0.8 \\ 0.2 \end{bmatrix}$.

However since the original question was to find $\bar{x}$ we have one more substitution to arrive at our final answer:

$$\bar{x} = C^{-1}\bar{x} = C^{-1}V_{\text{compact},AC^{-1}}\Sigma_{\text{compact},AC^{-1}}^{-1}U_{AC^{-1}}^T\bar{y}$$  

(30)

$$= \begin{bmatrix} 2 \\ 0.8 \\ 0.2 \end{bmatrix}. \quad (31)$$

This could also have been solved (for full credit) by brute-force using calculus in this particular case since the first coordinate is forced to be 2 in order to get the desired first coordinate of $\bar{y}$. So we have only two variables left, with one linear constraint, which means that we could reduce the problem to one of minimization of a quadratic function in one variable. However, this brute force calculus-based approach doesn’t help us deal with the next part of this problem.

(b) What if $C$ were a tall matrix with linearly independent columns? **Explicitly describe how you would solve this problem in that case, step by step.**

For convenience, we have copied the problem statement again here: Given a wide matrix $A$ and a matrix $C$ find $\bar{x}$ such that $A\bar{x} = \bar{y}$ and $\|\bar{x}\| \leq \|C\bar{x}\|$ for all $\bar{z}$ such that $A\bar{z} = \bar{y}$.

Here, you can assume that the wide matrix $A$ has linearly-independent rows but is otherwise generic. Similarly, $\bar{y}$ is a generic vector.

*(HINT: Does $C$ have a nullspace? Does $C^TC$ have a nullspace? Does the SVD of $C$ suggest any (invertible) change of coordinates from $\bar{x}$ to $\bar{z}$ such that $\|\bar{x}\| = \|C\bar{x}\|$?)*

**Solution:**

Now we have the condition where $C$ is a tall matrix with linearly independent columns. This means that $C$ itself is no longer invertible and we cannot just repeat the procedure done in the previous part of the problem. We don’t have access to a $C^{-1}$ and so need to stop and think. What we want is a square matrix $\bar{C}$ that is invertible, and gives us the same norm to minimize. That is, we need $\|\bar{C}\bar{x}\| = \|C\bar{x}\|$.

Writing this out, we see that since $\|C\bar{x}\|^2 = \bar{x}^TC^TC\bar{x}$, what we want is that $C^TC = \bar{C}^T\bar{C}$. Following the hint and using the compact-form SVD of $C = U_{\text{compact},C}\Sigma_{\text{compact},C}V_{C}^T$ in which $\Sigma_{\text{compact},C}$ is square. So, $C^TC = V_{C}\Sigma_{\text{compact},C}^2V_{C}^T$ since $U_{\text{compact},C}$ has orthonormal columns. This immediately suggests using $\bar{C} = \Sigma_{\text{compact},C}V_{C}^T$. Clearly $C^TC = \bar{C}^T\bar{C}$ by construction.
The only question now is whether $\tilde{C}$ is invertible. Because $C$ has linearly independent columns, it cannot have a nullspace. But we know from lecture that if $C^T C\tilde{v} = \tilde{0}$, that indeed $C\tilde{v} = \tilde{0}$ and so $C^T C$ also does not have a nullspace. So $C^T C$ is invertible, and since $V_C \Sigma^2_C V_C^T$ is the diagonalization of $C^T C$ by the basis $V_C$ of eigenvectors, $\Sigma_{\text{compact},C}$ is also invertible. The product of invertible matrices is invertible, and so indeed $C^T C$ is invertible, and since $V C \Sigma_2$ compact, $C V T C$ is the diagonalization of $C^T C$ by the basis $V_C$ of eigenvectors, $\Sigma_{\text{compact},C}$ compact, $C V T C$ is also invertible. The product of invertible matrices is invertible, and so indeed $\tilde{C}$ is invertible.

At this point, we have reduced this problem to what we did in the previous part. We just want to minimize $\|\tilde{C}\tilde{x}\|$ over all $\tilde{x}$ that satisfy $A\tilde{x} = \tilde{y}$. This is equivalent to minimizing $\|\tilde{x}\|$ over all $\tilde{x}$ that satisfy $A\tilde{C}^{-1}\tilde{x} = \tilde{y}$.

So in terms of an explicit procedure:

i. Compute the compact SVD of $C = U_{\text{compact},C} \Sigma_{\text{compact},C} V_C^T$.

ii. Compute the matrix $\tilde{C} = \Sigma_{\text{compact},C} V_C^T$.

iii. Compute the compact form SVD of the matrix $A\tilde{C}^{-1} = U\Sigma V^T$.

iv. Compute the solution $\tilde{x} = \tilde{C}^{-1} V \Sigma^{-1} U^T \tilde{y}$.

This comes from changing variables to $\tilde{x} = \tilde{C}^{-1} \tilde{x}$ and finding the minimum norm $\tilde{x}$ that works.

An alternative solution (that amounts to the same thing, effectively) exists where we use the pseudo-inverse of $C$ (i.e. the least-squares solution) and build our solution around that instead. Arguing why that works is a bit more involved.

5. Classification of Sinusoids

This HW problem can be viewed as a warm-up for the next topic in the course: which is going to be motivated by figuring out how to process signals recorded from the brain to decipher what a person wants to do in terms of a specific command to their robot arm. These kinds of problems are called “classification” problems. In this exercise, you will be using jupyter to classify sinusoids.

The iPython notebook Sinusoidal_Projection_fa19_prob.ipynb will guide you through the process of performing sinusoidal projections.

Suppose you already know the true potential frequencies $f_i$ and potential phases $\phi_i$ of a set of sinusoidal signals

$$S := \{\sin(2\pi f_i + \phi_i), i = 1, 2, \ldots, n\},$$

and you have some noisy samples of these true sinusoidal signals. You want to determine the true sinusoidal signal for each of these noisy samples—How would you approach the problem?

We will show in this problem that we can project noisy sinusoidal signals onto noiseless sinusoids to achieve good classification.

In the realistic world, one often doesn’t have the complete waveform of a continuous function, instead oftentimes one works with samples of the continuous function.

In our case, we generate noisy samples of the true sinusoidal signals in the following way. For each of the num_sinusoids true frequencies, each noisy sample $y_i$ consists of $S$ sample points sampled with a sampling rate of $F_s$ sample rate, and corrupted by noise scaled by $\sigma$.

$$y_i(k) = \sin(2\pi f \cdot k / F_s) + \sigma \cdot \text{Noise}. \quad k = 1, 2, \ldots, S.$$
(a) Run the first part of the jupyter notebook to generate our noisy data points. Use $\sigma = 0.1, 1.0, 10.0, 100.0$ and comment on what you observe in the plots.

**Solution:** The solutions notebook includes the plots. As you increase the scaling of the noise, you should observe that the averages and observations become noisier. When the noise is very small, we don’t need to do much to classify the noisy data.

(b) Complete part (b) of the notebook to project noisy sinusoids onto potential true sinusoids.

Sketch by hand the resulting 3D plot of projections qualitatively. Comment on what happens when you try the noise scalings $\sigma = 0.1, 1.0, 10.0, 100.0$.

**Solution:** The solutions notebook includes the plots. As you increase the scaling of the noise, you should observe that the clustering becomes more indistinguishable.

(c) Complete part (c) of the notebook to classify the data points and calculate the number of misclassified points.

Report the number of misclassifications for $\sigma = 0.1, 1.0, 10.0, 100.0$. Explain what happens when there is a high level of noise. Recall that our noisy process is random so that there can be cases where there are misclassifications even in low noise.

**Solution:** At values less than 2.0, the number of misclassified points should be close to 0. At 5.0, the number of misclassified points should be very high. This is natural because our sinusoids only have magnitude 1. With very high noise, it becomes harder to distinguish true signal from noise.

(d) For what qualitative regions of the noise level is it very beneficial for us to use projections? For very low values of noise, do you have to do projections to successfully classify? What else could you have done? This question is asking you to reflect on what you have observed.

**Solution:** When noise makes the signals indistinguishable by eye, it is helpful for us to use projections.

6. Write Your Own Question And Provide a Thorough Solution.

Writing your own problems is a very important way to really learn material. The famous “Bloom’s Taxonomy” that lists the levels of learning is: Remember, Understand, Apply, Analyze, Evaluate, and Create. Using what you know to create is the top level. We rarely ask you any homework questions about the lowest level of straight-up remembering, expecting you to be able to do that yourself (e.g. making flashcards). But we don’t want the same to be true about the highest level. As a practical matter, having some practice at trying to create problems helps you study for exams much better than simply counting on solving existing practice problems. This is because thinking about how to create an interesting problem forces you to really look at the material from the perspective of those who are going to create the exams. Besides, this is fun. If you want to make a boring problem, go ahead. That is your prerogative. But it is more fun to really engage with the material, discover something interesting, and then come up with a problem that walks others down a journey that lets them share your discovery. You don’t have to achieve this every week. But unless you try every week, it probably won’t ever happen.

7. Homework Process and Study Group

Citing sources and collaborators are an important part of life, including being a student! We also want to understand what resources you find helpful and how much time homework is taking, so we can change things in the future if possible.

(a) What sources (if any) did you use as you worked through the homework?
(b) **Who did you work on this homework with?** List names and student ID’s. (In case of homework party, you can also just describe the group.)

(c) **How did you work on this homework?** (For example, *I first worked by myself for 2 hours, but got stuck on problem 3, so I went to office hours. Then I went to homework party for a few hours, where I finished the homework.*)

(d) **Roughly how many total hours did you work on this homework?**

---

**Contributors:**

- Siddharth Iyer.
- Yu-Yun Dai.
- Sanjit Batra.
- Anant Sahai.
- Sidney Buchbinder.
- Justin Yim.
- Nikhil Shinde.
- Kuan-Yun Lee.