EECS 16B Designing Information Devices and Systems II
Spring 2019 UC Berkeley

Homework 11

This homework is due on Wednesday, April 24, 2019, at 11:59PM.
Self-grades are due on Saturday, April 27, 2019, at 11:59PM.

1. Inverse Kinematics

Inverse Kinematics is critical in robotics, control, and computer graphics applications. We need to be able to go backward from what we want to have happen in the real (or virtual) world to how to set parameters.

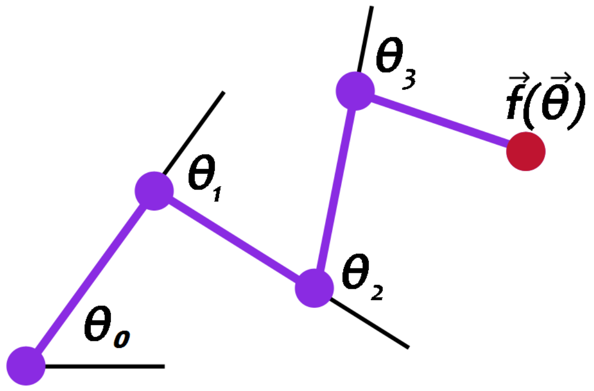


Figure 1: An example of an arm parameterized by θ_0, θ_1, θ_2, and θ_3 with the end effector at point $\vec{f}(\vec{\theta})$.

Suppose you have a robotic arm composed of several rotating joints.

The lengths r_i of the arm are fixed, but you can control the arm by specifying the amount of rotation θ_i for each joint. If we have an arm with four joints, it can be parameterized by:

$$\vec{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}. \tag{1}$$

Suppose further that we have some target $\vec{t} \in \mathbb{R}^2$, wich represents a point in the 2D space, and we would like for the end of the arm, called the end effector, to reach for the target. We have some function $\vec{f}(\vec{\theta})$ that rotates each joint of the arm according to the input and returns the position of the end effector. Figure 1 shows a visualization of an arm rotated by $\vec{\theta}$. To make the arm reach for the target $\vec{t}$, we want to find where the function $\vec{g}$ defined as

$$\vec{\theta} \in \mathbb{R}^4 \mapsto \vec{g}(\vec{\theta}) = \vec{f}(\vec{\theta}) - \vec{t} \tag{2}$$

is equal to $\vec{0}$. To accomplish this, we use the spirit of Newton's method for solving potentially nonlinear equations. (This is related to the spirit of the earlier problem on using iterative ways of solving least-squares problems.)

You might have seen Newton's method in your calculus course in the 1-d case. In this case, you have a real function g of a single parameter θ and we want to find a $\hat{\theta}$ so that $g(\hat{\theta}) = 0$. The method is the following. Step i of Newton's method does the following:

(a) Linearize g around θ^i, the current estimate of $\hat{\theta}$: $\hat{g}^i = g(\theta^{(i)}) + g'(\theta^{(i)})(\theta - \theta^{(i)})$

(b) θ^{i+1} solves $\hat{g}^i(\theta) = 0$. Note that this is easy to solve, since $\hat{g}^i$ is linear and 1-d. Note also that doing this is solving: $g'(\theta^{(i)})(\theta - \theta^{(i)}) = g(\theta^{(i)})$, which reduces to "inverting" the linear operator $z \mapsto g'(\theta^{(i)})z$.

and iterate this until $g(\theta)$ is close enough to 0 for our application. In practice, instead of solving exactly for $\hat{g}^i = 0$ in the second step of iteration i, we may chose to move θ^i by a fixed step-size η in the direction that the first-order approximation to the function suggests, but not all the way. This is done because the derivative $g'(\theta)$ where $\hat{g}(\theta) = 0$ might be very different from $g'(\theta^{i+1})$ where θ^{i+1} . After all, linearization is only valid in a local neighborhood. (This is the spirit of gradient descent as well.)

While you might have seen Newton's method as described above in your calculus courses, you might not have seen the vector-generalization of it. It follows exactly the same spirit. The first-order approximation to the vector valued function $\vec{g}(\vec{\theta})$ at $\vec{\theta}^{(i)}$ is now $\vec{g}(\vec{\theta}^{(i)}) + J_{\vec{g}}(\vec{\theta}^{(i)})(\vec{\theta} - \vec{\theta}^{(i)})$ where $J_{\vec{g}}(\vec{\theta})$ is the Jacobian matrix of the function $\vec{g}(\vec{\theta})$. For this problem, we will be using a robotic arm with 4 joints in a 2-dimensional space. Therefore, the Jacobian of $\vec{g}(\vec{\theta})$ will be a 2x4 matrix, and it is computed by calculating the partial derivatives of $\vec{g}(\vec{\theta})$:

$$J_{\vec{g}} = \begin{bmatrix} \frac{\partial g_x(\vec{\theta})}{\partial \theta_1} & \frac{\partial g_x(\vec{\theta})}{\partial \theta_2} & \frac{\partial g_x(\vec{\theta})}{\partial \theta_3} & \frac{\partial g_x(\vec{\theta})}{\partial \theta_4} \\ \frac{\partial g_y(\vec{\theta})}{\partial \theta_1} & \frac{\partial g_y(\vec{\theta})}{\partial \theta_2} & \frac{\partial g_y(\vec{\theta})}{\partial \theta_3} & \frac{\partial g_y(\vec{\theta})}{\partial \theta_4} \end{bmatrix}. \tag{3}$$

In this notation, we use $\vec{g}(\vec{\theta}) = [g_x(\vec{\theta}) \quad g_y(\vec{\theta})]^T$ where $g_x(\vec{\theta})$ is the x coordinate of the end effector and $g_y(\vec{\theta})$ is the y coordinate in our 2D space. There is nothing mysterious about this, if you think about it, this matrix of partial derivatives (a partial derivative is just a regular derivative with respect to a particular variable, treating all the other variables as constants) is the natural n-d candidate to replace g'.

The Newton algorithm in this case is an iterative method that gives us successively better estimates for our vector $\hat{\vec{\theta}}$. If we start with some guess $\vec{\theta}^{(i)}$, then the next guess is given by

$$\vec{\theta}^{(i+1)} = \vec{\theta}^{(i)} - \eta J_{\vec{g}}^{-1}(\vec{\theta}^{(i)})\vec{g}(\vec{\theta}^{(i)}) \tag{4}$$

where η is adjusted to determine how large of a step we make between $\vec{\theta}^{(i)}$ and $\vec{\theta}^{(i+1)}$. Notice that we need to invert the Jacobian matrix of first-partial-derivatives, and this matrix is not square. It is in fact a wide matrix. Fortunately, we know how to "invert" wide matrices, using the Moore Penrose pseudo-inverse that you saw in a previous homework. The minimality property of the Moore-Penrose pseudoinverse that we studied then is useful here because we would rather take small steps than big ones. And when tracking a moving reference, we'd like to have the joint angles change in a minimal way rather than in some very convoluted fashion.

The following problem will guide you step-by-step through the implementation of the pseudoinverse. The three steps of the psuedoinverse algorithm are:

- First, compute the compact-form SVD of the input matrix.
- Next, we compute Σ^{-1} by inverting each singular value σ_i.
- Finally, we compute the pseudoinverse by multiplying the matrices together in the right order.

There are three test cases that you can use to determine if your pseudoinverse function works correctly. In the first case, the arm is able to reach the target, and the end of the arm will be touching the target. In the second case, the arm should be pointing in a straight line towards the blue circle. The last case is the same as the second with the addition that a singular value will be very close to zero to test your pseudoinverse function's ability to handle small singular values. There is also an animated test case that will move the target in and out of the reach of the arm. Verify that the arm follows the target correctly and points towards the target when it is out of reach.

(a) In the "pseudoinverse" function, **compute** the SVD of the input matrix A by using the appropriate NumPy function.

(b) To save memory space, the NumPy algorithm returns the matrix Σ as a one-dimensional array of the singular values. Use this vector to **compute the diagonal entries of Σ^{-1}**. Be careful of numerical issues: first threshold the singular values, and only invert the singular values above a certain value ε, considering smaller ones are 0.

(c) We now have all of the parts to compute the pseudoinverse of A. **Add this computation to the function.**

2. Low Rank Approximation of a Matrix (optional)

(The more challenging parts of this question are marked optional and deemed out of scope. Feel free to skip. The rest of this question is in-scope in that it is just using SVD and projection properties that you know, but is also optional because it is more important for you to redo the midterm.)

In this question we will study the so called "low rank approximation" problem. As the name implies, consider an arbitrary matrix $X \in \mathbb{R}^{m \times n}$, with $m \geq n$. we are interested in finding another matrix $\widehat{X}$ having specified lower rank k, such that $\widehat{X}$ is "closest" to X, i.e.,

$$\min_{\widehat{X}} \|X - \widehat{X}\|_F \tag{5}$$

$$\text{subject to} \quad \text{rank}(\widehat{X}) \leq k \tag{6}$$

This problem goes to the heart of how we use the SVD for dimensionality reduction and to look at data. If we view a data matrix as a collection of columns where each of the columns is a different data point, then a rank-k approximation to that matrix is a collection of columns all of which represent points that are all on a k-dimensional subspace. *This discovery of hidden subspace structure is what finding low-rank approximations is truly about.* (The analogous story could be told about rows, but we'll keep our focus on columns since you are all more comfortable working with columns from 16A and 16B so far.)

To understand this problem, we will have to also think about what it might mean to approximate a matrix and we use the natural matrix norm to do this. In HW 9, you were introduced to the Frobenius norm — which involved treating a matrix as though it was just a big long vector filled with its entries.

(a) First, let's understand one of the simpler interpretations of why rank-r approximations to huge matrices are so useful. To specify an arbitrary $m \times n$ matrix, we have to choose $m \times n$ independent elements (entries). In other words, an arbitrary $m \times n$ matrix has $m \times n$ degree of freedom. **How much information (independent elements/degree of freedom) do we have to know to specify a rank r matrix of the same $m \times n$ size. How is low-rank approximation a kind of lossy compression for a matrix?** *(HINT: Think about outer-product representations for a rank r matrix, for example, that given by the SVD.)*

(b) Now, let us start into understanding the approximation itself. Before we get into the "hidden subspace" aspect, we need to think about what it means to approximate. Suppose we view a matrix A as a list of columns:

$$A = [\vec{a}_1, \vec{a}_2, \cdots, \vec{a}_n] \tag{7}$$

Show that the Frobenius norm $\|A\|_F == \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2}$ for a matrix A can be understood in terms of the regular Euclidean norms of its columns as:

$$\|A\|_F = \sqrt{\sum_{j=1}^{n} \|\vec{a}_j\|^2} \tag{8}$$

This is useful because it justifies using Frobenius norm as a way to measure the length of a matrix when we are really viewing the matrix as a collection of columns. (It turns out the same thing is true for viewing it as a collection of rows.)

(c) The Frobenius norm is also related to a natural inner-product for matrices, where again we treat the matrix as one big long vector. So $\langle C,D\rangle = \sum_{j=1}^{n}\langle \vec{c}_j, \vec{d}_j\rangle = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}d_{ij} = \sum_{j=1}^{n} \vec{c}_j^T\vec{d}_j$. Where $\vec{c}_j$ and $\vec{d}_j$ are the j^{th} column of C and D respectively.

Recall that in an earlier HW, we defined the **trace** of a square matrix as the sum of its diagonal terms, i.e., $\text{trace}(S) = \sum_{i=1}^{n} S_{ii}$. **Show that $\langle C,D\rangle = \textbf{trace}(C^TD)$.**

This in turn establishes that $\|A\|_F = \sqrt{\text{trace}(A^TA)} = \sqrt{\text{trace}(AA^T)}$.

(d) A useful lemma for you to show is: **Given rank one matrices $D_1, D_2 \ldots, D_k$, their sum $\sum_{i=1}^{k} D_k$ has rank at most k.** *(Hint: recall that the rank of a matrix is the dimension of the span of its columns. A rank one matrix therefore has a column span that is one dimensional. What does that mean for all the columns of that matrix?)*

(e) Now we want to get into the "hidden subspace" aspect of this problem. To do this, we need a way of talking about a potential subspace. To define a subspace, we need a basis. For convenience, we might as well think of an orthonormal basis. Let the matrix B consist of k orthonormal columns. The matrix B defines a subspace S of dimension k. Our underlying goal is to find an optimal such subspace S for approximating the data in X and equivalently, to find an optimal basis B for it. (Note here that $B \in \mathbb{R}^{m\times k}$.)

Before we worry about finding such a basis or subspace, it is good to understand how we would approximate X using it. But we know how to approximate a column in a subspace! We can project into it. So we can project all of X into that subspace by computing BB^TX. To understand the quality of this approximation, **show that $\|X - BB^TX\|_F^2 = \|X\|_F^2 - \|BB^TX\|_F^2$** .

(HINT: Give things names and invoke the Pythagorean theorem. Let $\vec{x}_i$ be the i-th column of X and let $Y_B = BB^TX$, and let $\vec{y}_{B,i}$ be the i-th column of Y_B. Let $R_B = X - Y_B$ be the residual that remains when estimating X using the basis B. Then show the desired result for each column using the properties of projection and put things together.)

(f) Now, our goal is to find the optimal such basis B. We can see from the previous part that since $\|X\|_F^2$ is outside our control, minimizing $\|X - BB^TX\|_F^2$ is the same as finding a matrix B with k orthonormal columns that maximizes $\|BB^TX\|_F^2$.

Use what you know about the Frobenius norm to argue that $\|BB^TX\|_F^2 = \|B^TX\|_F^2$.

Consequently, it suffices to find a B with orthonormal columns that maximizes $\|B^TX|_F^2$.

(g) Now, we are going to zoom in on a doubly special case of our main theorem first. Consider the special case of square $X = \Sigma$ matrices of size $n \times n$ that are already diagonal. Further suppose that the diagonal of Σ is non-negative and sorted so that it has $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$ down the diagonal.

To warm up, further restrict attention to matrices B that are made up of only standard basis vectors (i.e. each of the k columns of B has at most one 1 in them and the rest of that column is zero.) Furthermore, to be orthonormal, no two columns can have a 1 in the same row.

Under that assumption, **show that for such a B, the $\|B^TX\|_F^2$ must be a sum of k different σ_i^2.**

(HINT: Realize that each of the columns of B basically will pick out exactly one of the σ_i. Why should we pick the biggest ones?)

(h) Building on the previous part and its assumptions, **show that the best such $Y_B = BB^T\Sigma$ (for minimizing $\|\Sigma - Y_B\|_F^2$ has $\sigma_1, \sigma_2, \ldots, \sigma_k, 0, \ldots, 0$ down the diagonal.**

(i) (optional — out of scope) Now, we want to relax our artificial restriction that B has to be made of standard basis vectors. Let's look at the columns $\vec{c}_i$ of $C = B^T$. We can immediately see that $B^T\Sigma = C\Sigma = [\sigma_1\vec{c}_1, \sigma_2\vec{c}_2, \cdots, \sigma_n\vec{c}_n, \vec{0}, \vec{0}, \cdots, \vec{0}]$.

So we need to get a handle on these columns. Since $\|C\|_F^2 = \|B\|_F^2 = k$, we know that $\sum_{i=1}^n \|\vec{c}_i\|^2 = k$. We also know that since they are norms, that each of the $\|\vec{c}_i\|^2 \geq 0$.

Show that $\|\vec{c}_i\|^2 \leq 1$ for every $i = 1, \ldots, n$.

(HINT: Invoke Gram-Schmidt to assert that you can extend B with $n-k$ more orthonormal vectors to get a square orthonormal matrix $\widetilde{B}$. Then, since $\widetilde{B}^T\widetilde{B} = \widetilde{B}\widetilde{B}^T = I$, what do you know about the norms of the columns of $\widetilde{B}^T$? What is the relationship of those norms to the norms of $\vec{c}_i$?)

(j) (optional — out of scope) You know from above that $\|B^T\Sigma\|_F = \sum_{i=1}^n \sigma_i^2\|\vec{c}_i\|^2$ and that further $0 \leq \|\vec{c}_i\|^2 \leq 1$ for each i and their sum $\sum_{i=1}^n \|\vec{c}_i\|^2 = k$.

Show that under those constraints, $\sum_{i=1}^n \sigma_i^2\|\vec{c}_i\|^2 \leq \sum_{i=1}^k \sigma_i^2$.

(Hint: For convenience in writing, give new names $f_i = \|\vec{c}_i\|^2$ and remember that $\sum f_i = k$ while $0 \leq f_i \leq 1$. Getting an upper bound is about maximizing. You know how to reach that bound. Assume that you have some different allocation of the k total weight across the f_i that is claimed to be optimal and bigger than your claimed bound. Show that you can make it even bigger by redistributing the weight while keeping your constraints all satisfied. This contradicts the supposed optimality of the claimed optimal allocation. And so no such different allocation can exist.)

Because you know this bound can be hit, you have actually proved that the best k-dimensional subspace for approximating Σ in Frobenius norm is just that spanned by the first k standard basis vectors. In other words, the best rank k approximation to a diagonal matrix Σ with non-negative elements $\sigma_i \geq 0$ on the diagonal that are non-decreasing (i.e. $\sigma_i \geq \sigma_j$ if $i < j$) is the diagonal matrix with $\sigma_1, \sigma_2, \ldots, \sigma_k$ on the diagonal at the beginning and zero everywhere else.

(k) We have already proved in another homework that using the SVD, the Frobenius norm can be understood in terms of the singular values $\|A\|_F^2 = \sum_{i=0}^{\min(m,n)} \sigma_i^2$ (Recall that a $m \times n$ matrix can have at most $\min(m,n)$ nonzero singular values), where σ_i's are the singular values of A.

This was proved as a consequence of the fact that if U has orthonormal columns, then $\|UA\|_F^2 = \text{trace}(A^TU^TUA) = \text{trace}(A^TIA) = \text{trace}(A^TA) = \|A\|_F^2$ even if U is not square, and similarly if V^T has orthonormal rows, then $\|AV^T\|_F^2 = \text{trace}(AV^TVA^T) = \text{trace}(AIA^T) = \text{trace}(AA^T) = \|A\|_F^2$. That means that the SVD $A = U\Sigma V^T$ implies that $\|A\|_F = \|\Sigma\|_F$.

Now **please solve for $\hat{X}$** in equation (5) with Frobenius norm, you may want to use the results you developed so far in earlier parts of this problem. Here, you should use the conclusion of the optional sequence in the middle of this problem about the case of approximating Σ that is diagonal.

Congratulations! You have now been walked (hopefully not dragged!) through a proof of why the SVD gives you the best low-rank approximation to a matrix of data. In the linear-algebraic (and machine learning) literature, this is called the Eckhart-Young-Mirsky Theorem but all of the proofs that are easily accessible online or in standard texts take a more challenging (but shorter) route to the result. The argument given here is in more elementary 16AB style. This result justifies many uses of the SVD in machine learning and statistics.

3. System Identification

You are given a discrete-time system as a black-box. You don't know the specifics of the system but you know that it takes one scalar input and has two states that you can observe. You assume that the system is linear and of the form

$$\vec{x}(t+1) = A\vec{x}(t) + Bu(t) + \vec{w}(t), \tag{9}$$

where $\vec{w}(t)$ is an external unseen disturbance that you hope is small, $u(t)$ is a scalar input, and

$$A = \begin{bmatrix} a_0 & a_1 \\ a_2 & a_3 \end{bmatrix}, \quad B = \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}, \quad x(t) = \begin{bmatrix} x_0(t) \\ x_1(t) \end{bmatrix}. \tag{10}$$

You want to identify the system parameters from measured data. You need to find the unknowns: a_0, a_1, a_2, a_3, b_0 and b_1, however, you can only interact with the system via a blackbox model.

(a) You observe that the system has state $\vec{x}(t) = [x_0(t), x_1(t)]^T$ at time t. You pass input $u(t)$ into the blackbox and observe the next state of the system: $\vec{x}(t+1) = [x_0(t+1), x_1(t+1)]^T$.

Write scalar equations for the new states, $x_0(t+1)$ **and** $x_1(t+1)$. Write these equations in terms of the a_i, b_i, the states $x_0(t)$, $x_1(t)$ and the input $u(t)$. Here, assume that $\vec{w}(t) = \vec{0}$ (i.e. the model is perfect).

(b) Now we want to identify the system parameters. We observe the system at the start state $\vec{x}(0) = \begin{bmatrix} x_0(0) \\ x_1(0) \end{bmatrix}$. We can then input $u(0)$ and observe the next state $\vec{x}(1) = \begin{bmatrix} x_0(1) \\ x_1(1) \end{bmatrix}$. We can continue this for an m long sequence of inputs.

Let us define an m long trace to be $[x_0(0), x_1(0), u(0), x_0(1), x_1(1), u(1), x_0(2), x_1(2), u(2), \ldots, x_0(m-1), x_1(m-1), u(m-1), x_0(m), x_1(m)]$. **What is the minimum value of m you need to identify the system parameters?**

(c) Say we feed in a total of 4 inputs $[u(0), u(1), u(2), u(3)]$ into our blackbox. This allows us to observe the following states $[x_0(0), x_0(1), x_0(2), x_0(3), x_0(4)]$ and $[x_1(0), x_1(1), x_1(2), x_1(3), x_1(4)]$, which we can use to identify the system.

To identify the system we need to set up an approximate (because of potential disturbances) matrix equation

$$D\vec{s} \approx \vec{y}$$

using the observed values above and the unknown parameters we want to find. Suppose you are given the form of D in terms of some of the observed data:

$$D = \begin{bmatrix} x_0(0) & x_1(0) & u(0) & 0 & 0 & 0 \\ x_0(1) & x_1(1) & u(1) & 0 & 0 & 0 \\ x_0(2) & x_1(2) & u(2) & 0 & 0 & 0 \\ x_0(3) & x_1(3) & u(3) & 0 & 0 & 0 \\ 0 & 0 & 0 & x_0(0) & x_1(0) & u(0) \\ 0 & 0 & 0 & x_0(1) & x_1(1) & u(1) \\ 0 & 0 & 0 & x_0(2) & x_1(2) & u(2) \\ 0 & 0 & 0 & x_0(3) & x_1(3) & u(3) \end{bmatrix}. \tag{11}$$

For this D, **what are $\vec{y}$ and the unknowns $\vec{s}$ so that $D\vec{s} \approx \vec{y}$ makes sense?** Tell us what the components of these vectors are, written in vector form.

(Feel free to assume the columns of D are linearly independent if that is causing concern.)

(Hint: Recall what you did in lab.)

(d) Now that we have set up $D\vec{s} \approx \vec{y}$, **explain how you would use this approximate equation to estimate the unknown values** a_0**,** a_1**,** a_2**,** a_3**,** b_0 **and** b_1**.** In particular, give an expression for your estimate $\widehat{\vec{s}}$ for the unknowns in terms of the D and $\vec{y}$.

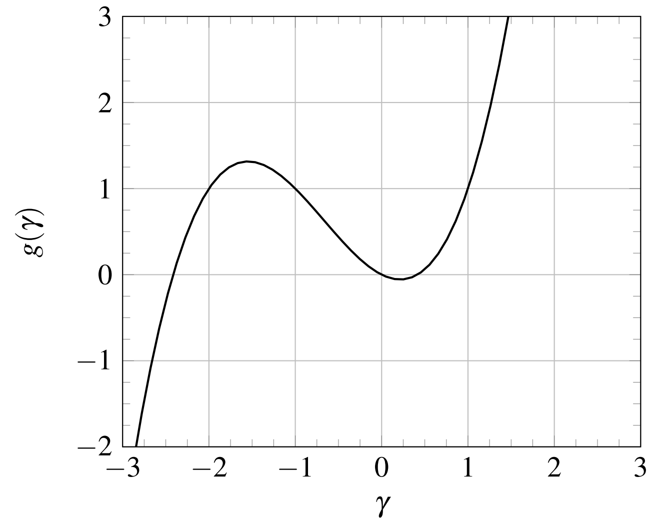
(HINT: Don't forget that D is not a square matrix. It is taller than it is wide.)

4. Linearizing a Two-state System

We have a two-state nonlinear system defined by the following differential equation:

$$\frac{d}{dt}\begin{bmatrix} \beta(t) \\ \gamma(t) \end{bmatrix} = \frac{d}{dt}\vec{x}(t) = \begin{bmatrix} -2\beta(t) + \gamma(t) \\ g(\gamma(t)) + u(t) \end{bmatrix} = \vec{f}(\vec{x}(t), u(t)) \tag{12}$$

where $\vec{x}(t) = \begin{bmatrix} \beta(t) \\ \gamma(t) \end{bmatrix}$ and $g(\cdot)$ is a nonlinear function with the following graph:



The $g(\cdot)$ is the only nonlinearity in this system. We want to linearize this entire system around a DC operating point.

(a) Take the DC operating input to be $u^\star = -1$. For this choice of $u^\star$, this system has three DC operating points. In other words, there are three states, $\vec{x}_1^\star$, $\vec{x}_2^\star$, and $\vec{x}_3^\star$, such that $\vec{f}(\vec{x}_1^\star, u^\star) = \vec{f}(\vec{x}_2^\star, u^\star) = \vec{f}(\vec{x}_3^\star, u^\star) = \vec{0}$. To be unambiguous, order them by increasing γ coordinates.

For this part, **find $\vec{x}_1^\star$, $\vec{x}_2^\star$, and $\vec{x}_3^\star$.** Your argument needn't be algebraic, especially since you don't have an expression for $g(\gamma)$. A graphical argument, using the above plot of $g(\gamma)$, will suffice for the γ parts. *(HINT: understand what is happening with γ first, then figure out β.)*

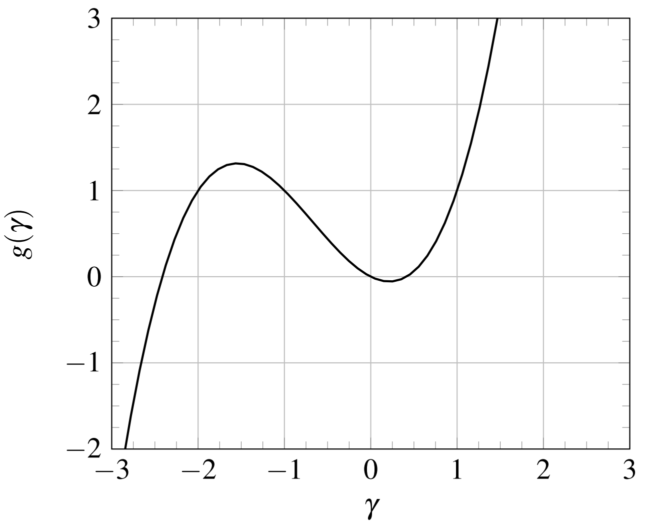
(b) Now that you have the three DC operating points, **linearize the system about the DC operating point $(\vec{x}_3^\star, u^\star)$ that has the largest value for γ.** Specifically, what we want is as follows. Let $\vec{\delta x}_i(t) = \vec{x}(t) - \vec{x}_i^\star$ for $i = 1,2,3$, and $\delta u(t) = u(t) - u^\star$. We can in principle write the *linearized system* for each DC operating point in the following form:

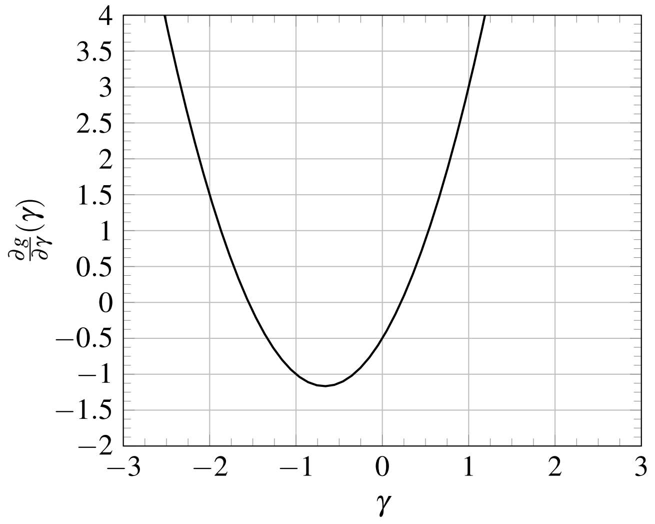
$$\text{(linearization about } (\vec{x}_i^\star, u^\star)) \qquad \frac{d}{dt}\vec{\delta x}_i(t) = A_i \vec{\delta x}_i(t) + B_i \delta u(t) + \vec{w}_i(t) \tag{13}$$

where $\vec{w}_i(t)$ is a disturbance that also includes the approximation error due to linearization.

For this part, **find A_3 and B_3.**

We have provided below the function $g(\gamma)$ and its derivative $\frac{\partial g}{\partial \gamma}$.





5. Smog Control

Smog is created with the emission of nitric oxide NO interacting with ozone O_3.

Let's examine the regulation of nitric oxide and ozone to see how we might control smog in our cities. We set our state vector to be

$$\vec{x} = \begin{bmatrix} N_{\text{NO}} \\ N_{\text{O}_3} \end{bmatrix},$$

where N_{NO} is the amount of NO molecules and N_{O_3} is the amount of O_3 molecules in the air. Say we have a simplified discrete-time model where the evolution of the molecules every t weeks is given by

$$\vec{x}(t+1) = \begin{bmatrix} \frac{4}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{5}{6} \end{bmatrix} \vec{x}(t).$$

(a) **Is the system stable? Why or why not?**

Here, we give you that

$$A = \begin{bmatrix} \frac{4}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{5}{6} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{5}} & -\frac{2}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \end{bmatrix} \begin{bmatrix} \frac{2}{3} & 0 \\ 0 & \frac{3}{2} \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \\ -\frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \end{bmatrix}$$

and that the characteristic polynomial $\det(\lambda I - A) = \lambda^2 - \frac{13}{6}\lambda + 1$.

(b) Suppose that we can use chemical methods to capture NO from the air. We can model the effects of these controls as an input $u(t)$ to our system:

$$\vec{x}(t+1) = \begin{bmatrix} \frac{4}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{5}{6} \end{bmatrix} \vec{x}(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t).$$

Is the system controllable?

(c) We want to convert the system

$$\vec{x}(t+1) = \begin{bmatrix} \frac{4}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{5}{6} \end{bmatrix} \vec{x}(t) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(t)$$

to controllable canonical form, which in this case is given by

$$\vec{z}(t+1) = \begin{bmatrix} 0 & 1 \\ -1 & \frac{13}{6} \end{bmatrix} \vec{z}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$

where $\vec{z}(t)$ is an appropriately transformed version of $\vec{x}(t)$; that is, $\vec{z}(t) = T\vec{x}(t)$. **Find the transformation T.**

(d) In order to keep smog under control, we want to set the closed-loop eigenvalues of the system

$$\vec{z}(t+1) = \begin{bmatrix} 0 & 1 \\ -1 & \frac{13}{6} \end{bmatrix} \vec{z}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$

to be $\lambda_1 = \frac{2}{3}, \lambda_2 = \frac{1}{6}$ using state feedback

$$u(t) = -\begin{bmatrix} \widetilde{k}_1 & \widetilde{k}_2 \end{bmatrix} \vec{z}(t).$$

What specific numeric values of $\widetilde{k}_1$ and $\widetilde{k}_2$ should we use?
If $\vec{z}(t) = T\vec{x}(t)$, **how can you use T to get what the control gains $\vec{k}$ should be so that $u(t) = -\vec{k}^T\vec{x}(t)$?** You don't need to compute explicit numeric values for the entries of $\vec{k}$.

6. SVD

(a) Given the matrix

$$A = \frac{1}{\sqrt{50}} \begin{bmatrix} 3 \\ 4 \end{bmatrix} \begin{bmatrix} 1 & -1 \end{bmatrix} + \frac{3}{\sqrt{50}} \begin{bmatrix} -4 \\ 3 \end{bmatrix} \begin{bmatrix} 1 & 1 \end{bmatrix},$$

write out a singular value decomposition of matrix A in the form $U\Sigma V^T$. Note the ordering of the singular values in Σ should be from the largest to smallest.

HINT: You don't have to compute any eigenvalues for this. Some useful observations are that

$$\begin{bmatrix} 3, 4 \end{bmatrix} \begin{bmatrix} -4 \\ 3 \end{bmatrix} = 0, \quad \begin{bmatrix} 1, -1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 0, \quad \| \begin{bmatrix} 3 \\ 4 \end{bmatrix} \| = \| \begin{bmatrix} -4 \\ 3 \end{bmatrix} \| = 5, \quad \| \begin{bmatrix} 1 \\ -1 \end{bmatrix} \| = \| \begin{bmatrix} 1 \\ 1 \end{bmatrix} \| = \sqrt{2}.$$

7. Norm

Given a matrix $A \in \mathbb{R}^{m,n}$ of rank r with singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_n = 0$. We want to show that

$$\max_{\vec{x} \neq \vec{0}} \frac{||A\vec{x}||}{||\vec{x}||} = \sigma_1$$

where $\vec{x} \in \mathbb{R}^n$ (i.e. that the maximum factor by which that A can grow the Euclidean norm is σ_1).

This problem will have you write out two steps in that proof.

We start by noting that since the matrix A^TA is symmetric, we know A^TA has eigenvalue decomposition $V\Lambda V^T$. Let $\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_n$ be a set of orthonormal eigenvectors of A^TA with associated eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$, where $\lambda_i = \sigma_i^2$.

(a) For $\vec{x} \in \mathbb{R}^n$, **express** $||A\vec{x}||^2$ **in terms of** $\vec{v}_i$**,** $\vec{x}$**, and** σ_i, for $i \in \{1, 2, ..., n\}$.
(HINT: Remember that $||\vec{y}||^2 = \vec{y}^T\vec{y}$. Also, you might want to decompose $\vec{x}$ as a linear combination of the set of orthonormal eigenvectors of A^TA.)

(b) **What is** $||A\vec{v}_1||$ **in terms of other information given in this problem? Show work and simplify as much as possible.**

8. Using upper-triangularization to solve differential equations

This problem really starts on the next page. This page is background that defines the notation.

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:

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

where the rows of the upper-triangular R look like

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

where the λ_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 $ScalarSolve(\lambda, y_0, \check{u})$ that takes a real number λ, a real number y_0, and a real-valued function of time $\check{u}$ as inputs and returns a real-valued function of time that is the solution to the scalar differential equation $\frac{d}{dt}y(t) = \lambda y(t) + \check{u}(t)$ 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 funcion of time, then $5u + 6g$ will be a real valued function of time that evaluates to $5u(t) + 6g(t)$ at time t.

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 $\clubsuit, \diamondsuit, \heartsuit, \spadesuit$.

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

```
1: $\vec{\tilde{x}}_0 = V^T\vec{x}_0$                          ▷ Change the initial condition to be in $V$-coordinates
2: $\vec{\tilde{u}} = V^T\vec{u}$                              ▷ Change the external input functions to be in $V$-coordinates
3: for $i = n$ down to 1 do                                     ▷ Iterate up from the bottom
4:     $\check{u}_i = \clubsuit + \sum_{j=i+1}^{n} \spadesuit$  ▷ Make the effective input for this level
5:     $\tilde{x}_i = ScalarSolve(\diamondsuit, \tilde{x}_0[i], \check{u}_i)$  ▷ Solve this level's scalar differential equation
6: end for
7: $\vec{x}(t) = \heartsuit \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \\ \vdots \\ \tilde{x}_n \end{bmatrix}(t)$  ▷ Change back into original coordinates
```

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

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

(c) Give the expression for $\clubsuit$ on line 4 of the algorithm above.

(d) Give the expression for $\spadesuit$ on line 4 of the algorithm above.

9. 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.

10. 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:

- Stephen Bailey.
- Yuxun Zhou.
- Anant Sahai.
- RAMW.
- Nikhil Shinde.
- alex devonport.
- Regina Eckert.
- Elena Jia.