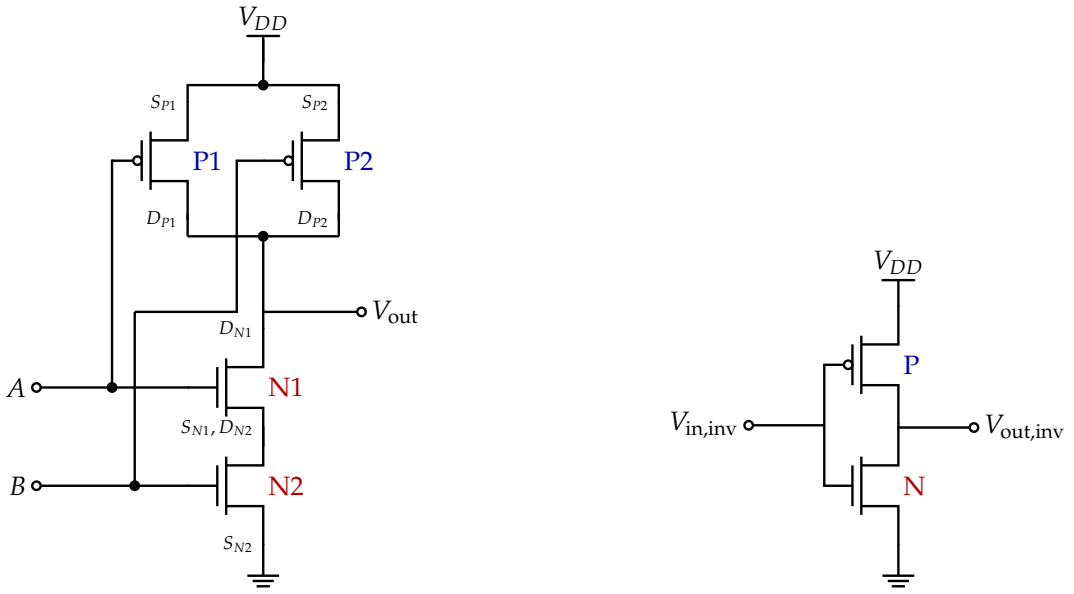
Homework 8

This optional homework is "due" on Thursday, August 11th at 5:59 pm PT. There will be no submission for this homework as it will be strictly practice for your final.

1. Transistor Switch Model (Spring 2021 Midterm)

In this problem, we will analyze the behavior of a NAND gate driving an inverter. Figure 1a shows the transistor model of a NAND gate and Figure 1b shows the transistor model of an inverter.

In this question assume that V_{DD} is greater than both the NMOS threshold $V_{\text{th,n}}$ and PMOS threshold $|V_{\text{th,p}}|$.



(a) NAND schematic with transistors.

(b) Inverter schematic with transistors.

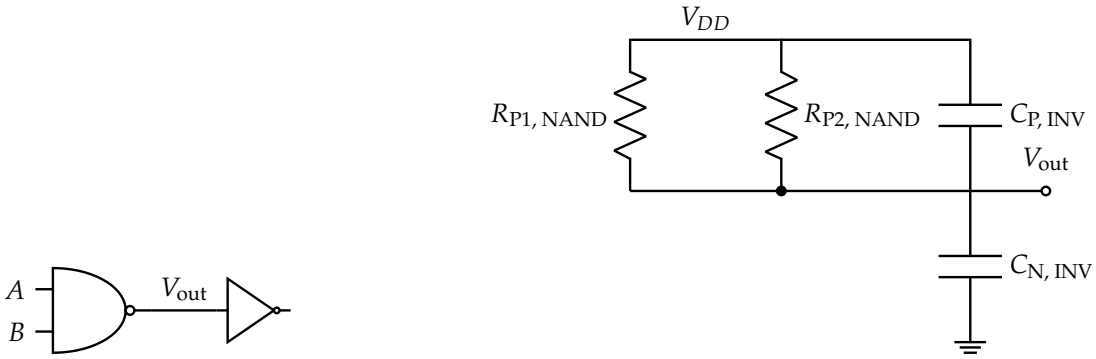
Figure 1: Transistor schematics

(a) A diagram of a NAND gate driving an inverter is shown in Figure 2a. Consider the case where $A = V_{DD}$ and $B = V_{DD}$ for a long time before $t = 0$. Then at $t = 0$, we switch A and B to 0 V. The equivalent simplified circuit after this transition is shown in Figure 2b. **Find V_{out} at time $t = 0$.**

(b) **Write the differential equation for solving $V_{\text{out}}(t)$ for $t \geq 0$ in the circuit shown in Figure 2b.** Specifically, **find coefficients λ and b in the following symbolic differential equation:**

$$\frac{dV_{\text{out}}(t)}{dt} = \lambda V_{\text{out}}(t) + bV_{DD}, \tag{1}$$

as a function of R_P, $C_{\text{P, INV}}$, $C_{\text{N, INV}}$, and V_{DD}. Assume that $R_{\text{P1,NAND}} = R_{\text{P2, NAND}} = R_P$.



(a) NAND driving an Inverter

(b) A Simplified Equivalent Circuit for when $A = B = 0\text{V}$

Figure 2: Schematic and model of a NAND gate driving an inverter

(c) **Solve $V_{\text{out}}(t)$ in the differential equation eq. (1) and the initial condition $V_{\text{out}}(0)$.** You should leave your answer in terms of λ, b, V_{DD}, and $V_{\text{out}}(0)$.

(d) Now consider the case where $A = 0\text{V}$ and $B = 0\text{V}$ for a long time before $t = 0$ in Figure 2a. At $t = 0$ we switch A and B to V_{DD}. **Write down the state (ON/OFF) of transistors P1, P2, N1, and N2 in the NAND gate. Draw the equivalent simplified circuit for this transition that will help us with writing the differential equation of $V_{\text{out}}(t)$.**

Hint: You may find the NAND resistor-switch model in Figure 3 helpful. Don't forget to include the inverter's capacitors, $C_{N,\,INV}$ and $C_{P,\,INV}$, which are loading the NAND gate.

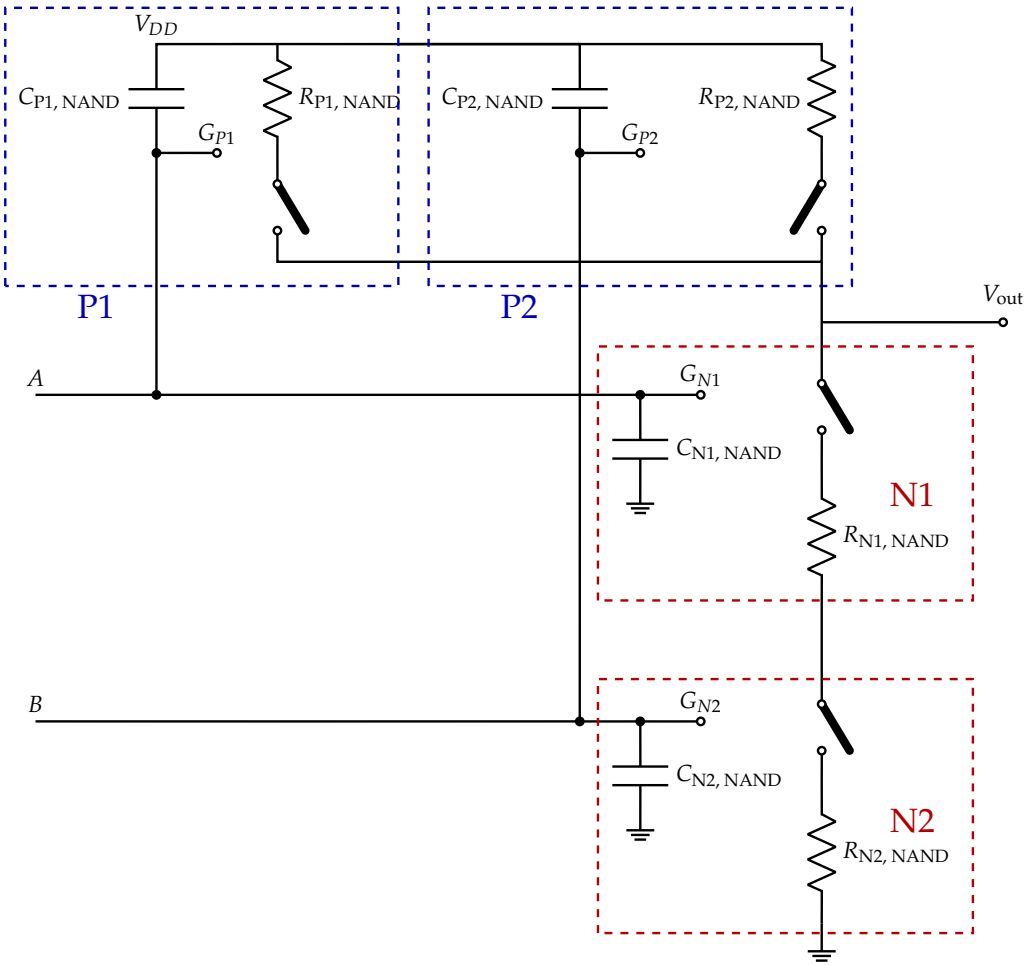
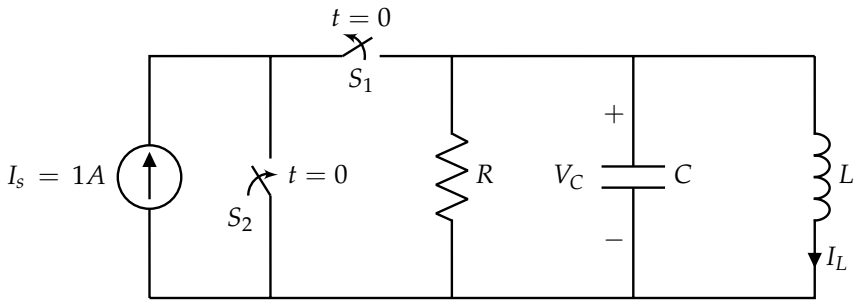


Figure 3: NAND Model: Capacitances

2. Parallel RLC with Current Source (Fall 2021 Midterm)

Consider the following circuit:



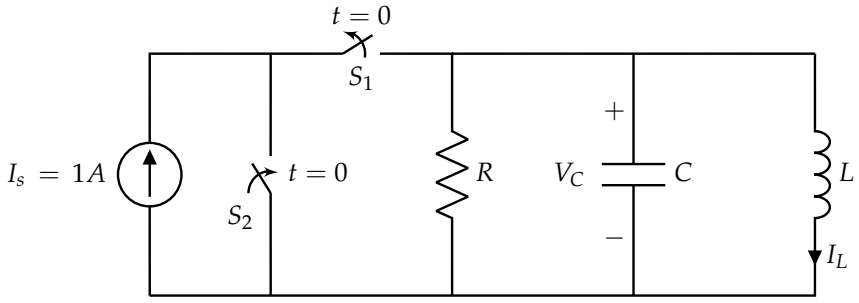
(a) At $t = 0$, switch S_1 became open and switch S_2 became closed. We first need to construct our state space system for $t \geq 0$. Our natural state variables are the current through the inductor $x_1(t) = I_L(t)$ and the voltage across the capacitor $x_2(t) = V_C(t)$ since these are the quantities whose derivatives show up in the system of equations governing our circuit.

Find the system of differential equations in terms of our state variables that describes this circuit for $t \geq 0$. Leave the system symbolic in terms of $I_s, R, L,$ and C. Write the system of differential equations in vector/matrix form with the vector state variable:

$$\vec{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} I_L(t) \\ V_C(t) \end{bmatrix} \tag{2}$$

This should be in the form $\frac{\mathrm{d}}{\mathrm{d}t}\vec{x}(t) = A\vec{x}(t)$ with a 2×2 matrix A.
Show your work.

(b)



For a long time in the past $t < 0$, assume that the switch S_1 had remained closed and S_2 had remained open. **What is $I_L(0)$ and $V_C(0)$? Give a brief justification as well.**

(c) For the rest of this problem, assume $R = 1\,\mathrm{M\Omega}, L = 25\,\mu\mathrm{H}, C = 10\,\mathrm{nF}$. With these values, we get the following eigenvalues and eigenvectors for A in the differential equation $\frac{\mathrm{d}}{\mathrm{d}t}\vec{x}(t) = A\vec{x}(t)$.

$$\lambda_1 = -Z_0 + \mathrm{j}\omega_0 \qquad \lambda_2 = -Z_0 - \mathrm{j}\omega_0, \tag{3}$$

$$\text{where } Z_0 = \sqrt{\frac{L}{C}} = 50 \text{ and } \omega_0 = \frac{1}{\sqrt{LC}} = 2 \times 10^6$$

$$V = \begin{bmatrix} \vec{v}_{\lambda_1} & \vec{v}_{\lambda_2} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ a & \overline{a} \end{bmatrix} \tag{4}$$

$$V^{-1} = \begin{bmatrix} b & -\text{j}0.01 \\ \bar{b} & \text{j}0.01 \end{bmatrix} \tag{5}$$

Consider a nice coordinate system for which we can write $\vec{x}(t) = V\vec{\tilde{x}}(t)$.
What is the $\widetilde{A}$ so that $\frac{\mathrm{d}}{\mathrm{d}t}\vec{\tilde{x}}(t) = \widetilde{A}\vec{\tilde{x}}(t)$? You can leave the answer symbolic in terms of λ_1, λ_2. Note that you **do not** need to find/evaluate a, b. You can still answer all the questions below without knowing these values.

1in

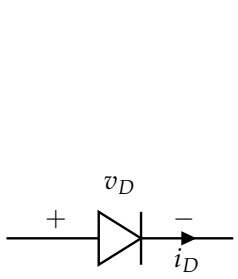
(d) Now, suppose that our initial conditions for $I_L(0)$ and $V_C(0)$ have changed to the following:

$$\begin{bmatrix} I_L(0) \\ V_C(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{6}$$

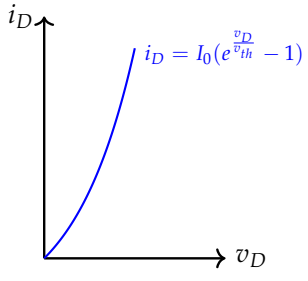
Using the information from (c), **find $I_L(t)$ for $t \geq 0$ and write the answer in a form involving real exponentials and sinusoids. Does $I_L(t)$ converge as $t \to \infty$? If so, what does it converge to?**
Show your work.

3. Nonlinear Circuit Analysis and Control (Spring 2021 Final)

So far, we have mainly focused on analyzing circuits with linear circuit elements, including resistors, capacitors, and inductors. However, we now have the tools to analyze circuits with nonlinear components. One such component is the diode. Diodes show up in many circuit applications, such as a buck-boost converter, which is a DC-to-DC converter commonly used to raise or lower some supply voltage and feed it to some other part of your circuit. We give a circuit diagram of a diode as well as its defining IV relationship below.



(a) Diode circuit diagram



(b) $i_D = I_0(e^{\frac{v_D}{v_{th}}} - 1)$

Figure 4: Diode circuit element description

For simplicity, we will be assuming parameters (perhaps unrealistically) such that the I-V relationship for our diode is:

$$i_D = e^{v_D} - 1. \tag{7}$$

(a)

(b) We want to analyze the circuit below.

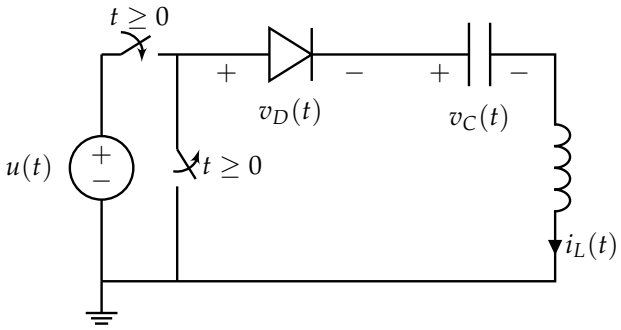


Figure 5: Diode LC Circuit Diagram

First, we'll define a model where $\vec{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} = \begin{bmatrix} v_C(t) \\ i_L(t) \end{bmatrix}$.

Use KCL, KVL, and the element I-V relationships to get a system of differential equations that describe $\vec{x}(t)$ for $t \geq 0$ as a vector-valued function in terms of $v_C(t), i_L(t), u(t)$:

$$\frac{d}{dt}\vec{x}(t) = \vec{f}(v_C, i_L, u) = \begin{bmatrix} f_1(v_C, i_L, u) \\ f_2(v_C, i_L, u) \end{bmatrix}.$$

What are f_1 and f_2? Note that these may be non-linear functions, but they cannot contain derivatives. Show your work.

(c) Say that one of the equations you got above was in the form:

$$\frac{d}{dt}y(t) = \frac{1}{L}\ln(y(t)+a) + \frac{1}{L}u(t), \tag{8}$$

where $a \in \mathbb{R}$ is a constant and $u(t)$ can be thought of as a control input. (This is not necessarily the correct answer for the earlier part). You choose $y^* = 0$ and $u^* = 1\,\text{V}$ as the operating point. **Linearize the above equation (8) about this operating point.** Recall that $\frac{d}{dz}\ln(z) = \frac{1}{z}$. Show your work.

(d) Now suppose you chose a capacitance and inductance such that the linearized model for the system in Fig. 5 around a particular equilibrium point looked like:

$$\frac{d}{dt}\vec{x}(t) = \underbrace{\begin{bmatrix} 0 & 1 \\ -4 & -4 \end{bmatrix}}_{A}\vec{x}(t) + \begin{bmatrix} 0 \\ 4 \end{bmatrix} u(t) \tag{9}$$

In order to solve this system, you need to convert A into a more convenient form.

Find an orthonormal matrix V and an upper-triangular matrix T such that $A = VTV^\top$. Show your work.

Hint: You may use the fact that the eigenvalues of A are $-2, -2$, with eigenspace $\text{span}(\vec{v}_1)$, where $\vec{v}_1 = \begin{bmatrix} -\frac{1}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} \end{bmatrix}$.

(e) We now want to move the eigenvalues of our linearized system more left in the complex plane to have our state approach the equilibrium point faster. The system is given below again for convenience:

$$\frac{d}{dt}\vec{x}(t) = \underbrace{\begin{bmatrix} 0 & 1 \\ -4 & -4 \end{bmatrix}}_{A}\vec{x}(t) + \underbrace{\begin{bmatrix} 0 \\ 4 \end{bmatrix}}_{\vec{b}} u(t).$$

Design a state-feedback controller $u = \vec{k}^\top\vec{x} = \begin{bmatrix} k_1 & k_2 \end{bmatrix}\vec{x}$ to move the eigenvalues of the system to $\lambda = -4, -5$. That is, find k_1, k_2 to give the desired eigenvalues.

4. Minimum Norm Solutions for Circuits involving Resistors (Fall 2021 Final)

Consider a current i_s flowing into a network of two parallel resistors R_1 and R_2, as shown in fig. 6 below.

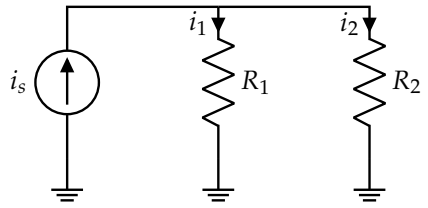


Figure 6: Current i_s dividing into i_1 and i_2.

From EECS 16A, we know that we can equate the voltage drops across the parallel resistors to derive $i_1 = \frac{R_2}{R_1+R_2} i_s$ and $i_2 = \frac{R_1}{R_1+R_2} i_s$. In this problem, we will try to derive the same current division result using the concept of minimum norm instead of voltage analysis.

It turns out that the current i_s will divide into two parts i_1 and i_2 in such a way that minimizes the total power dissipation $P = i_1^2 R_1 + i_2^2 R_2$ in the resistors.

(a) Argue that the current division result given by $i_1 = \frac{R_2}{R_1+R_2} i_s$ and $i_2 = \frac{R_1}{R_1+R_2} i_s$ minimizes the total power dissipation $P = i_1^2 R_1 + i_2^2 R_2$ using calculus.

Use the fact that KCL gives $i_2 = i_s - i_1$ to express P as a function of i_1 only.

(HINT: Once you solve for the optimal i_1, you don't have to do calculus again for i_2. Just use KCL.)

(b) Instead of using calculus to minimize the total power dissipation P, we can represent the current division problem as a minimum norm problem. Consider the vector $\vec{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ where $x_1 = i_1\sqrt{R_1}$ and $x_2 = i_2\sqrt{R_2}$. Notice that $P = i_1^2 R_1 + i_2^2 R_2 = x_1^2 + x_2^2 = \|\vec{x}\|^2$. Find the row vector A so that the KCL constraint $i_1 + i_2 = i_s$ can be written as $A\vec{x} = i_s$.

(c) Using the A matrix you found above, **what is the minimum norm solution to $A\vec{x} = i_s$?** Show your work.

To help you save computation, the compact SVD of a general 1×2 row vector is given by

$$\begin{bmatrix} a & b \end{bmatrix} = \underbrace{\begin{bmatrix} 1 \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} \sqrt{a^2+b^2} \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} \frac{a}{\sqrt{a^2+b^2}} & \frac{b}{\sqrt{a^2+b^2}} \end{bmatrix}}_{V^\top} \tag{10}$$

(d) Transform the minimum norm solution of $A\vec{x} = i_s$ to the original variables i_1 and i_2, and confirm that the result is $i_1 = \frac{R_2}{R_1+R_2} i_s$ and $i_2 = \frac{R_1}{R_1+R_2} i_s$ as the current-divider formula predicts. Show your work.

(e) We can solve any arbitrarily complicated circuit network using KCL and norm minimization, following the same technique that we used for the simple network in fig. 6. Consider a resistor network which has n resistor branches, with currents i_1, i_2, ..., i_n across the branch resistances R_1, R_2, ..., R_n respectively, and m total nodes each with current sources i_{s_1}, i_{s_2}, ..., i_{s_m}, which may be positive, negative or zero, as shown in fig. 7. Then the m KCL equations at the m nodes

can be written as $K\vec{i} = \vec{i}_s$, where $K \in \mathbb{R}^{m \times n}$, $\vec{i} = \begin{bmatrix} i_1 \\ i_2 \\ \vdots \\ i_n \end{bmatrix} \in \mathbb{R}^n$, and $\vec{i}_s = \begin{bmatrix} i_{s_1} \\ i_{s_2} \\ \vdots \\ i_{s_m} \end{bmatrix} \in \mathbb{R}^m$. This KCL constraint $K\vec{i} = \vec{i}_s$ completely captures what is visualized in fig. 7, so you don't have to write any additional KCL. Note that fig. 6 is a simple example of fig. 7 with $n = 2$ and $m = 1$.

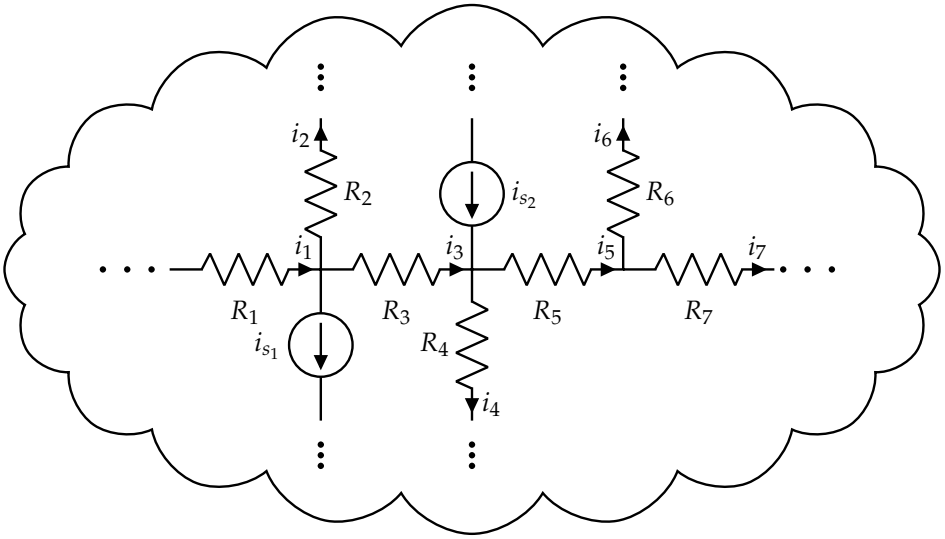


Figure 7: A section of an arbitrarily complicated network with n branches and m nodes.

We can change variables to $\vec{x} = D\vec{i}$ to represent the KCL constraint $K\vec{i} = \vec{i}_s$ as $A\vec{x} = \vec{i}_s$, and so the minimization of dissipated power $P = \sum_{j=1}^{n} i_j^2 R_j$ is just the minimization of $\sum_{j=1}^{n} x_j^2 = \|\vec{x}\|^2$.
Find the diagonal matrix D, and then find the matrix A in terms of D and K.
(HINT: Look at how $\vec{x}$ was defined in the previous part.)

(f) Assume the compact SVD of A is given by $A = U\Sigma V^\top$.
Use the minimum norm solution to $A\vec{x} = \vec{i}_s$ to solve for $\vec{i}$. Recall from the previous part that $\vec{x} = D\vec{i}$. Your final answer for $\vec{i}$ can only use U, Σ, V, D, $\vec{i}_s$ as well as standard matrix operations like inverses, etc.

5. Affine Control (Spring 2022 Midterm)

In this problem, we will analyze a *affine* model of the form

$$x[i+1] = \alpha x[i] + \beta u[i] + \gamma \tag{11}$$

where $\alpha, \beta, \gamma \in \mathbb{R}$, $x\colon \mathbb{N} \to \mathbb{R}$ is the state, and $u\colon \mathbb{N} \to \mathbb{R}$ is the input. Affine models are ubiquitous in control theory – in fact, our robot car from lab obeys a two-state-variable affine model.

(a) Suppose (for this part only) that:

- $\alpha = 1$,
- $\beta = 0$,
- $\gamma \neq 0$,
- $x[0]$ is anything.

so the model is of the form

$$x[i+1] = x[i] + \gamma. \tag{12}$$

Is the state x bounded? *Justify your answer.*

(b) Suppose (for this part only) that the state evolves according to eq. (11), i.e.,

$$x[i+1] = \alpha x[i] + \beta u[i] + \gamma \tag{13}$$

and

- $\alpha \neq 0$,
- $\beta > 0$,
- $\gamma \neq 0$,
- $x[0] = 0$.

Suppose that we supply feedback control of the form

$$u[i] = f \cdot x[i] \tag{14}$$

for $f \in \mathbb{R}$.

i. **For the specific case of $f = \frac{-1-\alpha}{\beta}$, show that the state x is bounded.**

ii. **In terms of α and β, give a range of f that keeps the state x bounded.**

(c) Suppose (for this part only) that the state evolves according to eq. (11), i.e.,

$$x[i+1] = \alpha x[i] + \beta u[i] + \gamma \tag{15}$$

and

- α is anything,
- β is anything,
- γ is anything,
- $x[0]$ is anything.

Suppose that we are setting up a least-squares system identification procedure to learn α, β, and γ, and that we have data of the form $(x[i], u[i], x[i+1])$, for $i \in \{0, 1, \ldots, \ell - 1\}$. **Set up a least-squares problem $D\vec{p} \approx \vec{s}$ to learn estimates for α, β, γ. What are D, $\vec{p}$, and $\vec{s}$?**

NOTE: Your answer for D should be as compact as possible.

NOTE: You do not need to solve the least squares problem; just set it up.

(d) Suppose (for this part only) that the state evolves according to Equation (11), i.e.,

$$x[i+1] = \alpha x[i] + \beta u[i] + \gamma \tag{16}$$

and

- $\alpha > 1$,
- $\beta > 0$,
- $\gamma > 0$,
- $x[0]$ is anything.

Suppose that we actually got our discrete-time model

$$x[i+1] = \alpha x[i] + \beta u[i] + \gamma \tag{17}$$

by discretizing a continuous-time model

$$\frac{\mathrm{d}}{\mathrm{d}t}x(t) = ax(t) + bu(t) + c \tag{18}$$

where the sampling interval length is $\Delta = 1$, i.e., $x[i] = x(i\Delta)$, and $u(t)$ is piecewise constant over intervals of length Δ, i.e., $u(t) = u(i\Delta) = u[i]$ for $t \in [i\Delta, (i+1)\Delta)$. **In terms of α, β, γ, what are a, b, and c?**

(HINT: You can use any discretization formulas we derived in class, as long as they apply. Alternatively, you may use the following formula in your derivation.

For a constant input v, and a time t_0 for which $x(t_0)$ is known, the solution to the differential equation

$$\frac{\mathrm{d}}{\mathrm{d}t}x(t) = ax(t) + v \qquad t \geq t_0 \tag{19}$$

is given by

$$x(t) = \mathrm{e}^{a(t-t_0)}x(t_0) + \frac{\mathrm{e}^{a(t-t_0)} - 1}{a} \cdot v, \qquad t \geq t_0. \tag{20}$$

when $a \neq 0$. Also, recall from the problem statement above that the sampling interval length $\Delta = 1$.)

6. SVD Calculation (Fall 2021 Final)

(a) Let $A = \begin{bmatrix} 0 & 2 \\ \sqrt{2} & 0 \\ 0 & 1 \end{bmatrix}$. The eigenvalues of $AA^\top = \begin{bmatrix} 4 & 0 & 2 \\ 0 & 2 & 0 \\ 2 & 0 & 1 \end{bmatrix}$ are $5, 2, 0$ with corresponding unnormalized eigenvectors $\begin{bmatrix} 2 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \\ 2 \end{bmatrix}$.

In addition, we know that:

$$A^\top \begin{bmatrix} 2 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 2 \end{bmatrix} = \begin{bmatrix} 0 & \sqrt{2} & 0 \\ 5 & 0 & 0 \end{bmatrix} \tag{21}$$

Write out the singular value decomposition (SVD) of A in any form you choose (outer product form, compact, or full). (No need to show work.)

(b) **What is the best rank 1 approximation of A (i.e., what is the rank 1 matrix B that minimizes $\|A - B\|_F$)? Write your answer as a 3×2 dimensional matrix.** (No need to show work)

7. Dynamical system approach to solving Ridge Regression (Spring 2022 Final)

In this problem, we will derive a dynamical system based approach to solving a modified version of the least-squares problem, commonly known as "ridge regression". This problem attempts to find the $\vec{x}$ that minimizes $\|A\vec{x} - \vec{y}\|^2 + \lambda \|\vec{x}\|^2$. Here we assume $A \in \mathbb{R}^{m \times n}$ is full column rank and scalar $\lambda \geq 0$.

The solution to the ridge regression problem is

$$\vec{\hat{x}} = (A^\top A + \lambda I)^{-1} A^\top \vec{y}. \tag{22}$$

Note that this solution is quite similar to the solution of least-squares. In many cases, direct computation of the solution to ridge regression is too slow, because it requires computing the matrix inverse $(A^\top A + \lambda I)^{-1}$, which is generally very costly for A with very large dimensions. We will instead solve the problem iteratively by using an update rule which turns this particular problem into an analysis of a particular discrete-time state-space dynamical system.

(a) We first connect the ridge regression problem to the familiar ordinary least-squares problem. **State the condition on λ in (22) needed to recover the least squares solution.**

(b) Using (22), **show that** $(A^\top A + \lambda I)\vec{\hat{x}} - A^\top \vec{y} = 0$**.**

(c) In iterative optimization schemes, we will get a sequence of estimates for $\vec{\hat{x}}$ at each timestep. Let $\vec{x}[i]$ denote our estimate for $\vec{\hat{x}}$ at timestep i.

In this problem we will consider the following update rule for solving the ridge regression problem:

$$\vec{x}[i+1] = \vec{x}[i] - \alpha \left((A^\top A + \lambda I)\vec{x}[i] - A^\top \vec{y} \right) \tag{23}$$

that gives us an updated estimate $\vec{x}[i+1]$ using the previous one $\vec{x}[i]$. Here α is the "step size" in our update rule which controls how much we update our solution estimate at each time step. For the purposes of this problem, it doesn't matter where we got the update rule, but the important thing to note is that if $\vec{x}[i] = \vec{\hat{x}}$, then by the previous part, $\vec{x}[i+1] = \vec{\hat{x}}$ and the system remains in equilibrium at $\vec{\hat{x}}$ for all time.

To show that $\vec{x}[i] \to \vec{\hat{x}}$, we define a new state variable $\Delta\vec{x}[i] = \vec{x}[i] - \vec{\hat{x}}$. It represents the deviation from where we want to be.

Derive the discrete-time state evolution equation for $\Delta\vec{x}[i]$, and show that it takes the form:

$$\Delta\vec{x}[i+1] = (I - \alpha G)\Delta\vec{x}[i]. \tag{24}$$

What is G?

(d) We would like to select α such that $\Delta\vec{x}[i]$ converges to 0. In particular, we want to make sure that we have a stable system. To do this, we need to understand the eigenvalues of $I - \alpha G$. **Given that $\lambda_k\{G\}$ are the eigenvalues of G, for $k \in \{1, 2, \ldots, n\}$ what are the eigenvalues of the matrix $I - \alpha G$?** *(Please fill in one of the circles for the options below. You will only be graded on your final answer.)*

i. $1 - \alpha\lambda_k\{G\}$ for $k \in \{1, 2, \ldots, n\}$

ii. $\alpha\lambda_k\{G\}$ for $k \in \{1, 2, \ldots, n\}$

iii. $1 - \lambda_k\{G\}$ for $k \in \{1, 2, \ldots, n\}$

iv. $1 + \alpha\lambda_k\{G\}$ for $k \in \{1, 2, \ldots, n\}$

Option	i	ii	iii	iv
Answer	◯	◯	◯	◯

(e) For system (24) to be stable, we need all the eigenvalues of $I - \alpha G$ to have magnitudes that are smaller than 1 (since this is a discrete-time system). **State the condition on α that would ensure that system (24) is stable.** You may assume that $\lambda_k\{G\}$ are real and $\lambda_k\{G\} > 0$ for $k \in \{1, 2, \ldots, n\}$.

8. Brain-Machine Interface and Neuron Sorting (Homework Practice)

The iPython notebook `pca_BMI.ipynb` will guide you through the process of analyzing brain machine interface data leveraging the SVD. This will help you to prepare for the project.

For SIXT33N (your robot car), you used the SVD and PCA to classify your sound inputs so that your car does what you tell it to do.

Brain-Machine interfaces (BMIs) are a way for a brain to directly communicate to an external device. They're often used for research, mapping, assisting or repairing human cognitive or sensory-motor function.

Data was collected that shows waveform traces of (simulated) brain waves of a subject whose arm is pointing in certain directions over time. These waveform traces are gathered from electrodes inserted into the brain, but the electrodes are generally recording more than 1 neuron at the same time — the brain is crowded after all. To make predictions based on neuron firing rates, we need to be able to distinguish waveforms that come from different neurons.

Each neuron has its own waveform "signature" shape unique to that neuron. This is due to the physical characteristics of neurons, such as their physical shape and structure. It is impossible to know beforehand how many neurons an electrode measures or what each neuron's waveform looks like, so we must first separate the waveforms from the different neurons near the electrode.

The goal of this problem is to see how we can use the SVD (in its PCA manifestation) and clustering to decide which neuron fired. We will first look at a case where there are only two neurons, then a case where there are three neurons. The neurons have also been presorted using a professional software package, so we can check our model against presorted data.

Please complete the notebook by following the instructions given.

Task 1: Two Neuron Waveform Sorting

(a) The data you have are traces of length 32 timesteps. The data is arranged along the rows of the matrix, each row is one trace. Import the data sets and see the average waveform for each presorted neuron by running the corresponding cells in the .ipynb. **What is the shape of the training data matrix** `three_neurons_training`**?**

(b) **What do the columns and rows of the training data matrix:** `three_neurons_training` **represent?**

We can approximate each waveform in a lower dimensional space using PCA. In your implementation of PCA you will decide how to choose these components.

(c) You will be using the SVD in your PCA function. **What matrices does the SVD return? What are the dimensions of these matrices for the SVD of** `three_neurons_training`**?**

(d) To represent each waveform in a lower dimensional space we want a basis for the time signals of the waveforms of the neurons. To construct this basis we start by taking the SVD (of `three_neurons_training`). **Which of the U and V matrices can we use to construct our desired basis?**

(e) **Read through** `PCA_train` **and implement** `PCA_project`. We will use these functions throughout the rest of the notebook, so make sure you understand them.

(f) **Call** `PCA_train` **on** `two_neurons_training` **and plot the 2 principal components.** Note that since the dataset is randomized, you might get different plots every time you run the second code cell of this notebook (the `_make_training_set` function). Note that these components have no real "physical" relationship to the actual shape of the neuron plots. They are a part of a basis, not the representation of exemplar traces.

(g) Before we project onto our principal components, let us project our data onto 2 random 32 length vectors. **Run the corresponding part in the .ipynb file and comment on the separation of the 2 neuron's distinctive trace shapes in this projected basis.** Do you think that it would be easy to tell them apart just by looking at their projection into these two random dimensions?

(h) **Project** `two_neurons_test` **onto the two principal components found using the SVD and produce a 2D scatter plot in the new basis.** We will also try projecting the presorted data containing 2 neurons so we can see how the model behaves on the 2 neurons. **Comment on the difference between the projections here and part (g) where you projected onto random directions.**

(i) Now we will repeat this process for three principal components. **Do you see a significant improvement with 3 principal components?**

Task 2: Three Neuron Sorting

(j) In the previous part we sorted 2 neurons using two or three principal components. In this part we are now dealing with 3 neurons. **Replicate what we did in the previous parts by finding and projecting our data on the first two principal components of this three neuron dataset in the .ipynb.**

(k) Now **call** `PCA_train` **on** `three_neurons_training` **and plot the 3 principal components. Do the same classification process as the 2 neuron data, but now with the 3 neuron data. Compare your model's behavior with that of the presorted data.** The `plot_3D` function will be useful to view the results.

(l) **How many principal components do you actually need to cluster the 3 neurons?** *(HINT: Think about whether there was an increase in separability between the last two parts.)*

(m) Now that we know how to project our data onto a basis where it is clearly separable we can classify our points and determine which neuron fired. (This is what we need to know if we want to use the firings of different neurons to build a BMI.)

To classify our points we use the following algorithm:

i. Find centroids: points that represent the average waveform of a particular neuron firing, in the basis of principal components.

ii. Classify each incoming point by assigning it the value of the centroid closest to it (i.e., declare that the neuron waveform that a point represents is the same as the neuron waveform of the centroid the point is closest to).

Since we already have some presorted data, we can use this information to aid in classification. We can calculate our centroids by taking the empirical mean of the presorted data corresponding to a particular neuron firing. **Use this method to find centroids in the corresponding section of the .ipynb file. Then use** `which_neuron` **on the** `two_classified` **data set and count the number of times each neuron fired.**

Contributors:

- Anirudh Rengarajan.
- Kunmo Kim.
- Ayan Biswas.
- Vladimir Stojanovic.
- Anant Sahai.
- Michael Danielczuk.
- Nikhil Shinde.
- Ashwin Vangipuram.
- Daniel Abraham.