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

# HW 13

## 1. Bode Plot: Piecewise Linear Approximation

So far, you have evaluated the Bode plots for transfer functions by using a computer. However, understanding how to draw a Bode plot by hand helps to quickly evaluate the behavior of a transfer function. To do this, we just need to use a bit of approximation thinking — the same kind of thinking that we used for understanding linearization and PCA. In order to evaluate the magnitude response of a transfer function without the aid of a computer, we will need to develop approximations that are simple enough to compute by hand, but accurate enough to be useful for understanding system behavior.

Consider the function

$$f(x) = \frac{1}{(1+x)^k} \tag{1}$$

for $x > 0$ and some positive whole number $k$.

We will be using the following approximations: If $|x| \ll 1$, then $1+x \approx 1$. If $|x| \gg 1$, then $1+x \approx x$.

(a) **What is the approximate behavior of $f(x)$ when $|x| \ll 1$?**

(b) **What is the approximate behavior of $f(x)$ when $|x| \gg 1$?** (Your answer should still have $x$ in it.)

On a log-log plot, $f(x)$ can be plotted by drawing the function $\log\big(f(x)\big)$ on the y-axis against $\log(x)$ on the x-axis. Recall that we plot Bode plots for the magnitude of a transfer function on a log-log plot to allow us to easily compose transfer functions graphically (since multiplication turns into addition) while also allowing the very large dynamic range of frequencies of potential interest to fit on a single plot.

(c) For each of the two approximations above ($|x| \ll 1$ and $|x| \gg 1$), **express the approximate solutions in log-log form ($\log\big(f(x)\big)$ vs $\log(x)$).**
Notice that in log-log form, each approximation is a straight line (ie $\log\big(f(x)\big) = c_1 + c_2 \log(x)$, similar to a linear function $y = b + mx$).

Now, let us consider the transfer function

$$H(j\omega) = \frac{1}{1+\frac{j\omega}{\omega_0}}$$

The magnitude of the transfer function can be found as

$$|H(j\omega)| = \frac{1}{\sqrt{1+\left(\frac{\omega}{\omega_0}\right)^2}}$$

Let us apply the same type of approximation to the magnitude of the transfer function.

(d) **What is the behavior of the magnitude transfer function when $\omega \gg \omega_0$? What is the behavior in log-log form?** i.e. Write an approximation for the log of the magnitude as a function of the log of $\omega$. It is traditional in circuits (for ease of plotting by hand) to use a base-10 logarithm. For this part, this will cause no issues.

(e) **What is the behavior of the magnitude transfer function when $\omega \ll \omega_0$? What is the behavior in log-log form?** i.e. Write an approximation for the log of the magnitude as a function of the log of $\omega$.

Here, you might be tempted to resolve this more finely by thinking about what the logarithm looks like for something that is close to 1. There is no harm in wanting to do so, but realize that we are interested in the behavior with respect to the log of $\omega$. Does this level of detail matter?

Again, notice that in the log-log domain, under both limiting cases, the approximations are basically linear functions.

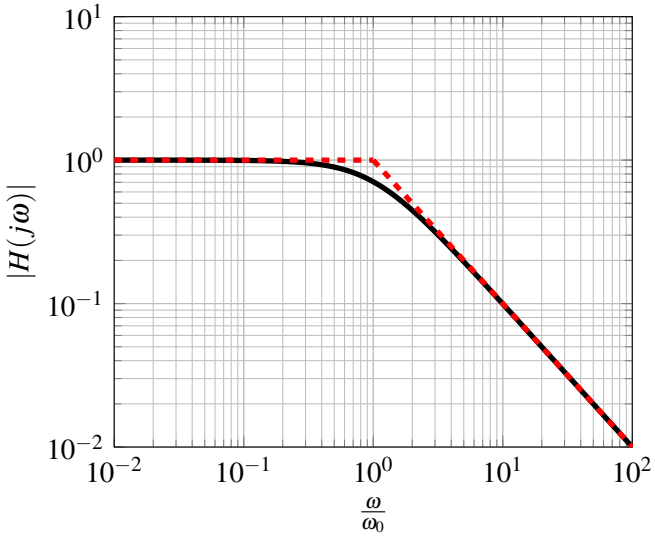
(f) **In both limiting cases, what is the ratio of $ER(j\omega) = \frac{|H(j\omega)|_{\text{approx}}}{|H(j\omega)|}$?** This ratio represents the error of the approximation. The further this ratio is from 1, the worse the approximation is. **What is the limit of the error ratio as $\omega \to 0$ (for the $\omega < \omega_0$ case) and as $\omega \to \infty$ (for the $\omega > \omega_0$ case)?**

Notice that as $\omega$ moves further from $\omega_0$, the error ratio tends towards 1, indicating that the approximation gets better the further we are from $\omega_0$.

(g) **What is the error ratio of each of the two approximations at $\omega = \omega_0$?**

Putting these results together, we can see how to approximate the magnitude bode plot for a transfer function of the form $H(j\omega) = \frac{1}{1+\frac{j\omega}{\omega_0}}$. We can simply draw straight lines on the log-log plot.

Bode Plot with Exact and Straight-Line Approximation




In future courses, you will learn how to apply this 'straight line' approximation for more complicated transfer functions — but in a sense, you already know how to do this since products become sums on a log-log plot, and reciprocals are sign changes. Between these two properties, you can essentially draw approximations to the magnitude of a transfer function on a log-log plot for anything that you can decompose easily into products or reciprocals of the basic form above. How can you turn a polynomial into a product? By factoring it! So the magnitude-response of anything that is a ratio of polynomials can now be handled by you.

## 2. Linearization to help classification: discovering logistic regression and how to solve it

We saw in lecture how the problem of linear classification between two categories boiled down to discovering a vector that told us what to project onto to get a score, together with a threshold that told us where to draw the boundary between the two categories. The goal in linear classification problems is to learn those vectors and thresholds from data — examples that have been labeled as belonging to the two categories in question.

You can, in spirit, reduce the problem of linear multi-class classification to that of binary classification by picking vectors that correspond to each of the categories "X" as compared with all the other examples categorized into a hybrid synthetic category of "not-X". This will give rise to vectors corresponding to each category with the winner selected by seeing which one wins. However, we will focus here on the binary problem since that is the conceptual heart of this approach.

As was discussed in lecture, the naive straightforward way of picking the decision boundary (by looking at the mean example of each category and drawing the perpendicular bisector) is not always the best. The included Jupyter Notebook includes synthetic examples that illustrate the different things that can happen so that you can better appreciate the pathway that almost inexorably leads us to discover logistic regression as a natural approach to solve this problem based on the conceptual building blocks that you have already seen. (See if you can catch why this problem follows the previous one on approximating Bode plots.)

It is no exageration to say that logistic regression is the default starting method for doing classification in machine learning contexts, the same way that straightforward linear regression is the default starting method for doing regression. A lot of other approaches can be viewed as being built on top of logistic regression. Consequently, getting to logistic regression is a nice ending-point for this part of the 16AB story as pertains to classification.

Let's start by giving things some names. Consider trying to classify a set of measurements $\vec{x}_i$ with given labels $\ell_i$. For the binary case of interest here, we will think of the labels as being "+" and "-" for reasons that should be clear from our discussion of multi-class classification above. For expository convenience, and because we don't want to have to carry it around separately, we will fold our threshold implicitly into the weights by augmenting our given measurements with the constant "1" in the first position of $\vec{x}_i$ for each $i$. Now, the classification rule becomes simple. We want to learn a vector of weights $\vec{w}$ so that we can deem any point with $\vec{x}_i^\top \vec{w} > 0$ as being a member of the "+" category and anything with $\vec{x}_i^\top \vec{w} < 0$ as being a member of the "-" category.

The way that we will do this, following the treatment in lecture, is to do a minimization in the spirit of least squares. Except, instead of necessarily using some sort of squared loss function, we will just consider a generic cost function that can depend on the label and the prediction for the point. For the $i$-th data point in our training data, we will incur a cost $c^{\ell_i}(\vec{x}_i^\top \vec{w})$ for a total cost that we want to minimize as:

$$\arg\min_{\vec{w}} \quad c_{\text{total}}(\vec{w}) = \sum_{i=1}^{m} c^{\ell_i}(\vec{x}_i^T \vec{w}) \tag{2}$$

Because this can be a nonlinear function, our goal is to solve this iteratively as a sequence of least-squares problems that we know how to solve.

Consider the following algorithm pattern

1: $\vec{w} = \vec{0}$ ▷ Initialize the weights to $\vec{0}$
2: **while** Not done **do** ▷ Iterate towards solution
3: Compute $p_i = \vec{w}^T \vec{x}_i$ ▷ Generate current estimated labels
4: Compute $(c^{\ell_i})'(p_i)$ ▷ Generate derivatives of the cost for update step
5: Compute $(c^{\ell_i})''(p_i)$ ▷ Generate second derivatives of the cost for update step

6: $\vec{\delta w} = LeastSquares(\cdot,\cdot)$ ▷ We will derive what to call least squares on
7: $\vec{w} = \vec{w} + \vec{\delta w}$ ▷ Update parameters
8: **end while**
9: Return $\vec{w}$

The key step above is figuring out with what arguments to call our good friend LeastSquares — who we know so well from 16A and from using it in 16B. We just have the labels $\ell_i$ and the points $\vec{x}_i$. We need to create new virtual inputs and outputs for least-squares to try to match.

Notice that this is the same algorithm pattern you saw in the last homework where you were trying to set the joint angles on a robot arm so that it reaches a desired target point. There, you solved a minimum-norm problem in each loop of the iteration, but decided to take only a cautious step partially towards that solution. Here, it is a least-squares problem, and we are going to accept taking bolder steps in that direction.

Recall from lecture and discussion that the derivative of a scalar function of a vector is row, and the second derivative can be represented by a matrix.

When the function $\vec{f}(\vec{x},\vec{y}) : \mathbb{R}^n \times \mathbb{R}^k \to \mathbb{R}^m$ takes in vectors and outputs a vector, the relevant derivatives for linearization are also represented by matrices:

$$D_{\vec{x}}\vec{f} = \begin{bmatrix} \frac{\partial f_1}{\partial x[1]} & \cdots & \frac{\partial f_1}{\partial x[n]} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x[1]} & \cdots & \frac{\partial f_m}{\partial x[n]} \end{bmatrix}.$$

$$D_{\vec{y}}\vec{f} = \begin{bmatrix} \frac{\partial f_1}{\partial y[1]} & \cdots & \frac{\partial f_1}{\partial y[k]} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial y[1]} & \cdots & \frac{\partial f_m}{\partial y[k]} \end{bmatrix}.$$

Then, the linearization (first-order expansion) becomes

$$\vec{f}(\vec{x},\vec{y}) \approx \vec{f}(\vec{x}_0,\vec{y}_0) + D_{\vec{x}}\vec{f}\cdot(\vec{x}-\vec{x}_0) + D_{\vec{y}}\vec{f}\cdot(\vec{y}-\vec{y}_0).$$

(a) Now, suppose we wanted to approximate the total cost function

$$c_{total}(\vec{w}) = \sum_{i=1}^{m} c^{\ell_i}(\vec{x}_i^T\vec{w}) \tag{3}$$

in the neighborhood of a weight vector $\vec{w}_0$. **Write out the first-order expression for approximating the cost function** $c_{total}(\vec{w}_0 + \vec{\delta w})$.

This should be something in vector/matrix form like you have seen for the approximation of nonlinear systems by linear systems. We don't want to take any second derivatives just yet — only first derivatives.

*(HINT: Use the linearity of derivatives and sums to compute the partial derivatives with respect to each of the $w[g]$ terms. Don't forget the chain rule and the fact that $\vec{x}_i^\top\vec{w} = \sum_{j=1}^n x_i[j]w[j] = x_i[g]w[g] + \sum_{j\neq g} x_i[j]w[j]$. Then put them together into an appropriate row. Your final answer should involve an appropriately weighed sum of rows.)*

(b) Now, we want a better approximation that includes second derivatives. For a general function, we would look for

$$f(\vec{x}_0+\vec{\delta}x) \approx f(\vec{x}_0) + f'(\vec{x}_0)\vec{\delta}x + \frac{1}{2}\vec{\delta}x^\top f(\vec{x}_0)''\vec{\delta}x \tag{4}$$

where $f'(\vec{x}_0)$ is an appropriate row and, as you've seen in class, $f''(\vec{x}_0)$ is a matrix that represents the second derivatives.

**Take the second derivatives of the total cost and write an expression for the second-order approximation for the total cost function.**

*(HINT: Once again, use the linearity of derivatives and sums to compute the partial derivatives with respect to each of the $w[h]$ terms. This will give you $\frac{\partial^2}{\partial w[g]\partial w[h]}$. Don't forget the chain rule and again use the fact that $\vec{x}_i^\top \vec{w} = \sum_{j=1}^n x_i[j]w[j] = x_i[h]w[h] + \sum_{j\neq h} x_i[j]w[j]$. Then put everything together into an appropriate matrix. Your final answer should involve an appropriately weighed sum of matrices. Each matrix will itself be representable in simple outer-product form.)*

(c) **Complete the squares to reformat the cost approximation in the form of a sum of a constant plus the sum of $(q_i^\top \vec{\delta}w - b_i)^2$.**

(d) **Interpret the result of the previous step as desiring for $\vec{\delta}w$ to be the solution to a standard least-squares problem. What is the $A$ matrix? What is the $\vec{y}$?**

(e) Consider the following cost functions:
squared error: $c_{sq}^+(p) = (p-1)^2,\ c_{sq}^-(p) = (p+1)^2$;
exponential: $c_{exp}^+(p) = e^{-p},\ c_{exp}^-(p) = e^p$;
and logistic: $c_{logistic}^+(p) = \ln\left(1+e^{-p}\right),\ c_{logistic}^-(p) = \ln(1+e^p)$.
**Compute the first and second derivatives of the above expressions with respect to $p$.**

(f) **Fill in the missing code and examine how these cost functions behave in the Jupyter notebook for classification. What did you see?**

Congratulations! You now know the basic optimization-theoretic perspective on logistic regression. After you understand the probability material in 70 and 126, you can understand the probabilistic perspective on it as well. After you understand the optimization material in 127, you will understand even more about the optimization-theoretic perspective on the problem including why this approach actually converges.

## 3. 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.

## 4. 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:**

- Sidney Buchbinder.
- Anant Sahai.
- Kuan-Yun Lee.
- Nathan Lambert.