Review notes: Important points to keep in mind

A collection of key points based on questions and missed points in exams.

Chapters 2 and 3. Mathematical preliminaries

- If the pole is $-a$, the corresponding time constant is $1/a$, and time domain term is e^{-at}. (Repeat this thinking exercise with $a = 1/\tau$.)
- If the poles are complex conjugates, $p_i = -\alpha \pm j\beta$, the time domain function is $e^{-\alpha t} \sin(\beta t + \phi)$, where ϕ is the phase lag.
- When we have a repeated pole at $-a$, we still have the exponential function e^{-at}, but it now has other terms with time "tagged" onto it to make the entire time function decays slower in time.
- Relatively speaking, poles that are closer to the origin are associated with larger time constants (or are slower) and are considered dominant.
- A transfer function is always derived from a differential equation with zero initial condition(s). That is why when we begin with a linear equation, we still have to put it in deviation variables first.
- The steady state gain of a transfer function is the final change of the output (deviation) variable relative to a unit change in the input. It may be easier to put the statement as: "the final value of the output when the input is a unit step." The steady state gain is easy to spot if we use the time constant form of a transfer function. That's because when a transfer function is expressed as a ratio of two polynomials, the steady state gain is the ratio of the two constant coefficients.
- Steady state gain and time constants (or you can say poles and zeros) are properties of the transfer function and are *independent* of the input.
- The most important part of a transfer function is the polynomial in the denominator; it is the characteristic polynomial of a differential equation model.
- Along the same line, if there are multiple inputs to the differential equation model, the transfer functions associated with each of these inputs will have the identical characteristic polynomial.
- For a second order transfer function, the natural time period τ is *not* the time constant. (Forgot why? Review comments in Section 3.2.)

Chapters 4 and 9. State-space models

- There is no unique state space representation of a model. (We see this quite well with MATLAB, but this also makes using its result confusing at the beginning. As always, check the eigenvalues. When we move to design controls in Chapter 9, make sure the MATLAB matrices are scaled properly. This point is explained in MATLAB Session 4.)
- Do not be intimidated by the formulation of the state-transition matrix. Its explanation in Section 4.2 is largely used to relate state-space models to transfer functions. For more complex problems beyond, for example, Example 4.7, there are numerical techniques available at more advanced levels.
- Speaking of which, learning the canonical form transformations and controller designs in Chapter 9 relies to a great extent the use of MATLAB at this introductory level. (General theoretical analysis requires linear algebra that can only be taken up in a second or more advanced course. Despite this note, you still need to read Section 9.3 slowly and carefully.)
- When a problem is simple enough, the state feedback gain is the same as the proportional gain in classical control. We did a couple of examples, e.g., Example 4.7B, to illustrate the point. But as soon as we move to slightly more complex problems, including the use of

integrating action, controller settings using state-space control are very different from classical control. We no longer have this "physical" PID feel, but that is fine because state-space control is handled invariably by computers.

Chapters 5 and 6. Closed-loop system analysis

- There are two ways for us to organize the material in Chapter 5. If you find the beginning too mathematical, start with Section 5.2.3 to get a better physical feel of the tasks at hand.
- The closed-loop characteristic polynomial is identical whether we solve the servo or the regulatory problem.
- For a given process model, we can only choose one manipulated variable. All other variables become disturbance (or load) variables.
- When we derive, for example, the closed-loop set point tracking function, we can set all other variables to zero. That's because we work with linear systems. We can consider each effect (input) individually and then, if needed to later, add them back together.
- When you solve a problem, it is very important to draw the block diagram (no matter how simple that is), and label it with the proper variables and their units.
- Then you go through each block and determine the proper signs of all the steady state gains. The decisions are based on your understanding of the process model and safety concerns (failed open versus failed closed).
- Integrating action eliminates offset—it is usually introduced by a PI or PID controller, but it also can be introduced by the process function (as in manipulating liquid level with a pump).
- If the process function has no integrating action itself, neither P nor PD control can eliminate offset. We need, once again, integrating action.
- When we do an open-loop step test, it reveals information about $G_a(s)G_p(s)G_m(s)$, and they can be anything (meaning in any form). We can only approximate the result as a first order with dead time function if the data reflect a self-regulating multicapacity process. You *do not blindly* force fit the data.
- In theory, internal model control allows us to design a controller based on our specification of the closed-loop response and the process model. The problem is that the theory is also based on perfect pole-zero cancellation, which we cannot expect in reality. Nevertheless, the analysis is instructive, and shows us, for example, how K_c is (more or less) inversely proportional to the process gain.
- There is no single best method and no one correct answer. We need to approach each problem individually and learn to make our judgment. Generally, all the methods require field tuning. Refer to summary Table 6.3 of tuning methods for a review.
- To detune a controller, we decrease the proportional gain or increase the integral time constant.
- Generally, when we add integral to proportional control, we need to lower the proportional gain. We can increase the proportional gain if we add derivative action to make a PID controller. (Read and scrutinize the results from `recipe.m` carefully to pick up the trends.)
- With a PID controller, we generally want the integral time constant to be larger than the derivative time constant: $\tau_D/\tau_I \approx 0.25$ (typically between 0.1 and 0.3). A simple thinking is that a small integral time constant can lead to a very underdamped behavior or even instability. (We get a better idea when we do root locus plots or Bode plots.)
- Tuning relations like Cohen-Coon and Ziegler-Nichols try to have an one-quarter system decay ratio. It may be too oscillatory (underdamped) for some systems.

- ITAE index provides the "best" conservative design among integral criteria and other empirical tuning relations. Ciancone-Marlin relation is too conservative.
- With significant dead time (relative to the dominant process time constant), the system can become unstable and hard to control, so be careful and conservative. Use smaller proportional gain and larger integral time constant.

Chapters 7. Stability

- A simple first or second order system is always stable. By simple, we mean a system with no open-loop zeros in the right-hand plane.
- The curves on a root locus plot are the closed-loop poles. We draw the open-loop poles and zeros only because they define the mathematical limits and are a handy guide to what the root locus plot may look like. (When you read other texts, the term open-loop pole is rarely used explicitly. Most people just talk about poles and you should know that what they mean are the open-loop poles.)
- For a polynomial to have solutions with only negative real parts (*i.e.*, stable), the necessary condition is that all the coefficients are positive definite.
- Being necessary does not mean sufficient. This is where the Routh array comes in; it provides the additional conditions required for stability. We apply that to a third and higher order polynomials. For a second order polynomial, the coefficient test provides the necessary and sufficient conditions.
- Substitution of $s = j\omega$ allows use to find the ultimate gain K_{cu} and ultimate frequency ω_{cu}. The ultimate gain should be identical to the value if we apply the Routh array (based on the limit of the inequality), or root locus (based on the intersection with the imaginary axis). The result is also identical to the ultimate gain and the gain cross over frequency ω_{cg} using a Bode plot.
- Using root locus as a computational tool, we can rationalize the choice of integral and derivative time constants with respect to time domain response that we may specify. The only drawback is that root locus plots cannot handle dead time easily.
- Back in Chapter 2, we introduced the idea of identifying dominant poles of an open-loop transfer function. When we do a root locus plot, we are now analyzing a closed-loop system and the dominant poles are the *root locus curves* that are closer to the imaginary axis, *not* the open-loop poles. We may still casually point to an open-loop zero and say that's where the dominant pole may end up, and that's a direct implication that a root locus starts from an open-loop pole and ends up at a zero.

Chapters 8. Frequency Response

- One main reason why we use frequency response analysis is that it can handle dead time easily. And if you are good, you can use the technique to explain *why* a certain system is stable—mostly based on reading the phase lag.
- The basis of the analysis is that if we have a *stable* transfer function and if we apply a sine input $A \sin \omega t$, the response *at large enough times* $y_\infty(t)$ is also a sine function with the same frequency. Here, there are two more important points.
  1. The amplitude and phase lag of the sinusoidal response at large enough times are functions of frequency. Generally, the amplitude becomes smaller and the phase lag larger with higher frequencies.
  2. If the transfer function is $G(s)$, then the amplitude and phase lag of the sinusoidal response in the time-domain, $y_\infty(t)/A$, are $|G(j\omega)|$ and $\angle G(j\omega)$. (This is one reason why frequency

response analysis is so confusing at the learning stage. The notations of time and frequency domains are blended together.)

- Controller design using frequency response analysis is based on the Nyquist stability criterion. The computation is based on the gain and phase margins, which are much easier to find on the Bode plot than the Nyquist plot.
- The computation of the gain and phase margins are based on a frequency plot of the open-loop function $G_c(j\omega)G_a(j\omega)G_p(j\omega)G_m(j\omega)$, but that *does not* mean that we are solving the open-loop problem. In very simple terms, the Nyquist stability criterion is applied on the basis of a shifted coordinate or the closed-loop equation written as $G_c(j\omega)G_a(j\omega)G_p(j\omega)G_m(j\omega) = -1$.
- Generally, we use this method to find the proportional gain K_c. So with a PID controller, we need to select some probable values of integral and derivative time constants beforehand. Since we can compute the ultimate gain effortlessly with MATLAB, the easiest route is to use the Ziegler-Nichols ultimate gain tuning relations.
- When we solve a problem, we do not need trial-and-error to find K_c. We take advantage of the fact that the phase angle of K_c is zero, and thus the phase angle plots of the open-loop function with or without K_c are identical.
- Remind yourself repeatedly that simple first and second order systems are always stable. It makes no sense to apply the Nyquist stability criterion.
- With a third order system without dead time, there are other methods to design a controller. We can use root locus plots. Or we approximate the process reaction curve with a first order with dead time function and then use empirical tuning relations. We could use IMC. And of course, we can use frequency response. With MATLAB, we can quickly find the ultimate gain and ultimate frequency, with which we apply to the Ziegler-Nichols ultimate gain tuning relations.
- Frequency response analysis does not reveal the probable time-domain response. To overcome that, older texts used the Nichols chart, which is actually a nice technique. Now, newer texts do not put much emphasis on it. We can only guess that it is because we can use MATLAB to plot easily the closed-loop time response, and if also need to, closed-loop modulus (definition in the Web Support supplementary notes).