Model Matching Control

The desired closed loop transfer function is determined and the controller required to make this happen is then determined. Steady state errors are determined directly from the closed loop transfer function $G_0(s)$.

Good: Easy to implement and understand.

Bad: Works by canceling the plant. Hence, it is very sensitive to having an accurate model of the plant. The controllers obtained may be difficult to implement in hardware, since an arbitrary transfer function must be implemented.

ITAE and Deadbeat: Only one free parameter (ω_0) which controls the speed of response and indirectly the required control effort. These methods won't work for systems with right half plane zeros in the plant (they cannot be cancelled by a stable controller). The resulting system is a type one system.

Quadratic Optimal: Only one free parameter (*q*) which can be used more directly to balance the speed of response with the control effort. This method will work for a plant with right half plane zeros. The resulting system is not necessarily a type one system, but sometimes the controller can be modified so this is true.

PID Control

This class of controllers is usually used in conjunction with the root locus method, and tools such as Matlab's *sisotool.* Steady state errors are usually determined from the plant and controller transfer functions using the position error constant and the velocity error constant.

Good: These controllers are very easy to implement with op-amps and are the most widely used in industry. They are generally more robust than model matching methods, since they do not work by trying to cancel the plant.

Bad: Closed loop pole locations must be on the root locus, which means you're not free to place them wherever you want. The controllers may introduce unwanted zeros in the closed loop transfer function. This can usually be overcome with a dynamic prefilter.

Diophantine Control

This class of controllers allows the user to specify the desired closed loop pole locations directly. Steady state errors are determined from the plant and controller transfer functions using the position error constant and the velocity error constant.

Good: The closed loop poles can be specified arbitrarily, which can lead to very fast response. The order of the controller can be increased to make the systems type 1 system. These controllers are fairly robust, especially if the closed loop pole can be specified to be far away from the $j\omega$ axis.

Bad: Unwanted zeros may be introduced into the closed loop transfer function, which can sometimes be cancelled by dynamic prefilters. The order of the system get large quite quickly, resulting in a large number of closed loop poles that must be assigned and fairly complicated controllers. The controllers may be difficult to implement in hardware.

State Variable Control

This class of controllers allows the user to specify the desired closed loop pole locations directly. Steady state errors are generally more difficult to determine. If the system is very complicated at all, most of the planet uses state variable methods.

Good: The closed loop poles can be specified arbitrarily, which can lead to very fast response. The controller does not introduce additional zeros or change the order of the system. The order of the controller can be increased to make the systems type 1 system, though that is beyond this course. These controllers are fairly robust, especially if the closed loop pole can be specified to be far away from the $j\omega$ axis. These are fairly simple to implement in hardware.

Bad: You need to be able to measure all of the states, which can be difficult or even impossible. This can be overcome by constructing *observers* (or state estimators), but this then makes implementation much more difficult.