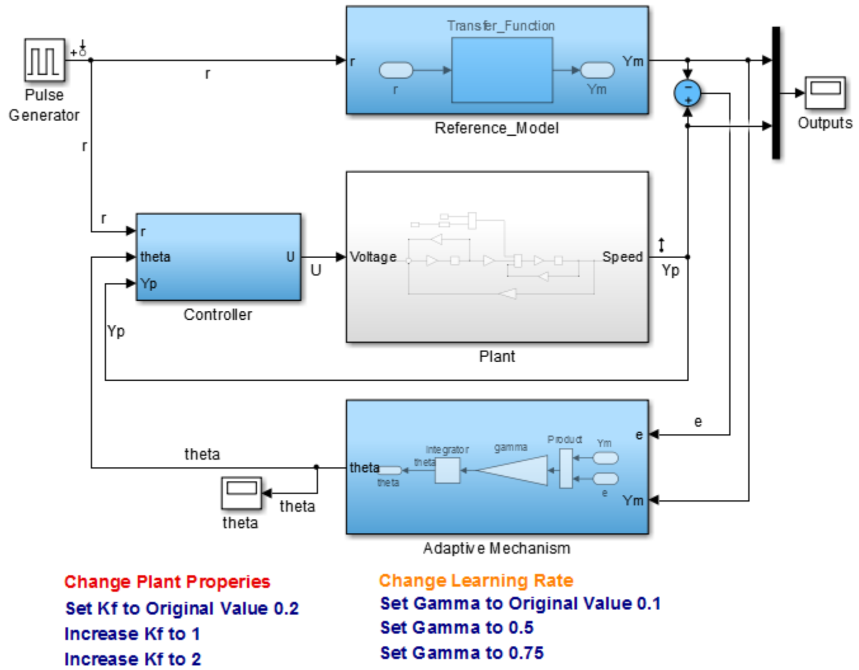
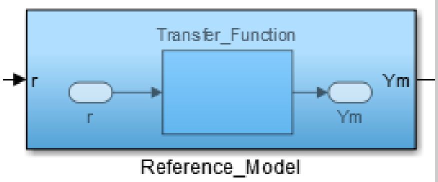
Adaptive Controller Example

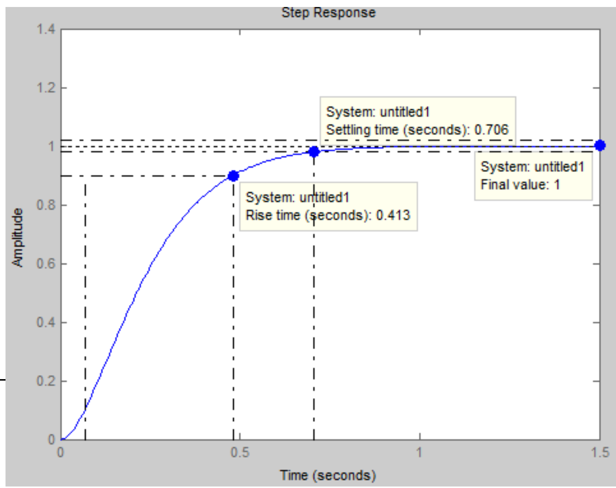
Introduction:

Objective of this example is to demonstrate how to design and model adaptive controller, tune and analyse its performance using Simulink®. For this example we have used direct adaptive method called Model Reference Adaptive Controller (MRAC). There are three main elements of this model: Reference Model, Plant Model and Adaptive Controller. Over all model looks like this:

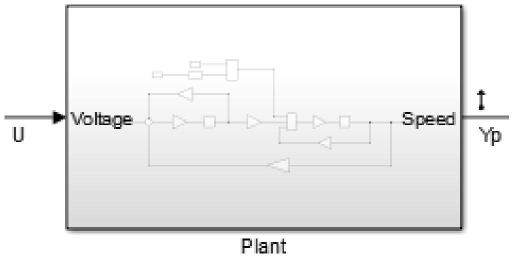


1.) Reference Model: This part of the controller captures\models the desired behaviour of closed-loop system. In other words, how you like your overall system to behave for a given input is modelled in this subsystem. In this example, reference behaviour is modelled as a transfer function. This can also come from closed-loop system specifications described in below figure as desired Rise Time (0.413sec), Settling Time (0.706sec) and Steady State Error (0). Reference Model output, *Ym* is desired reference trajectory which Plant output (*Yp*) has to follow.



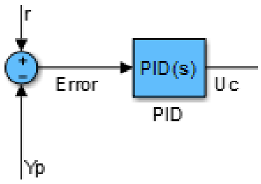


2.) <u>Plant Model:</u> In this example, plant is a DC Motor. One of the many motor parameters, *Kf* – Mechanical Damping, is considered to be varying. Initial value is assumed to be 0.2. And PID controller is tuned to achieve desired response with this initial value of *Kf*. Now as motor goes through ageing and impact of other environmental conditions, *Kf* changes, this will change motor behaviour. Plant output is *Yp*. Hence controller has to adapt\change its parameter values to achieve desired response (*Yp – Ym = error (e) = 0*).

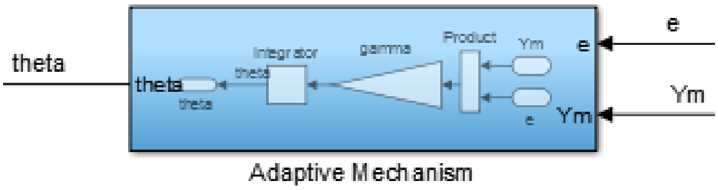


3.) Adaptive Controller: There are two sub components of this controller.

a. PID Controller: This part of controller is fixed and gains have been tuned for keeping initial plant condition in mind and to achieve overall stability. Output of PID controller is *Uc*.

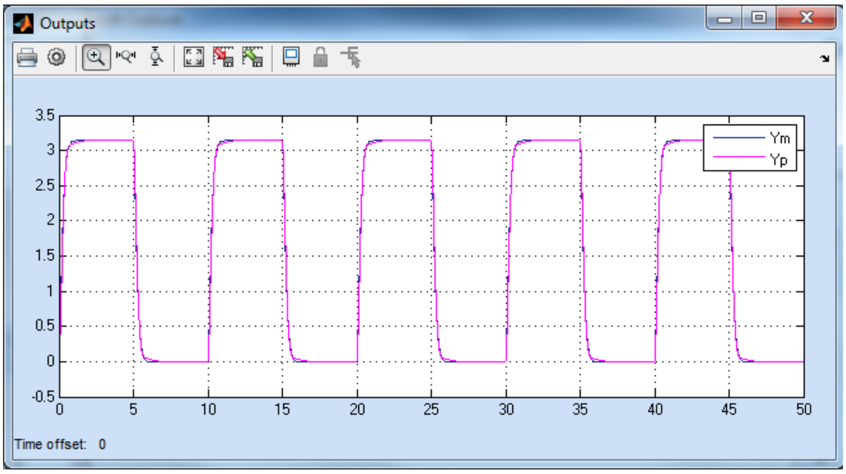


b. Adaptive Mechanism: This goal of this part of controller is to change its output (*theta*) based on error (*e*) between plant output (*Yp*) and reference model output (*Ym*). How fast it can adapt (or change its output) depends on parameter called learning rate, *gamma*. Higher the value of *gamma*, faster it can adapt to any changes in plant. But there are some side effects also. Controller output (*U*) is calculated by: *U = Uc * theta*.

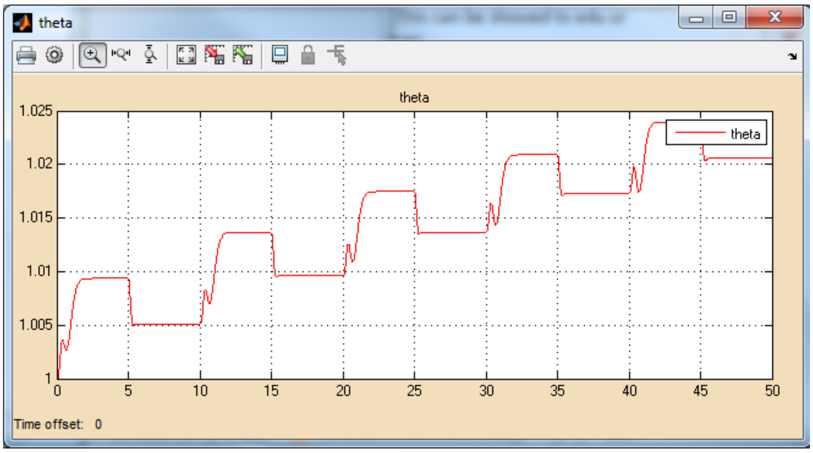


Using This Example:

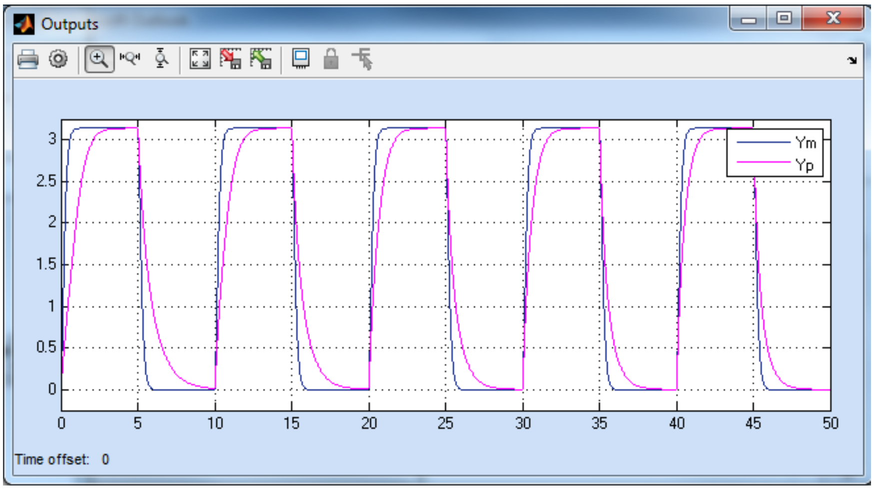
Step 1: Run this model with default values of plant, PID controller and learning rate. You can observe that overall all closed-loop system is behaving as per reference model. Look at the small difference between *Ym* and *Yp*. This small error is due to well-tuned PID controller for known plant values (*Kf* and others).



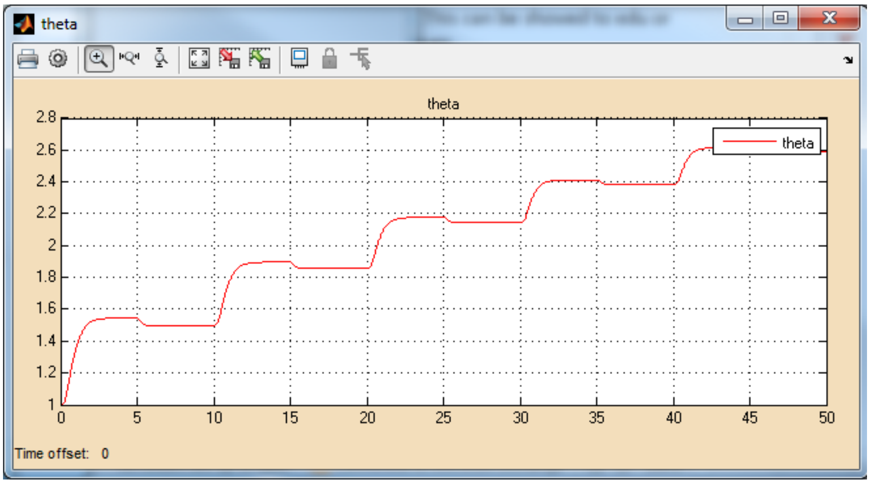
Also observe how theta, output of adaptive mechanism, changes as simulation progresses. It doesn't change much due to small error (*e = Yp-Ym*).



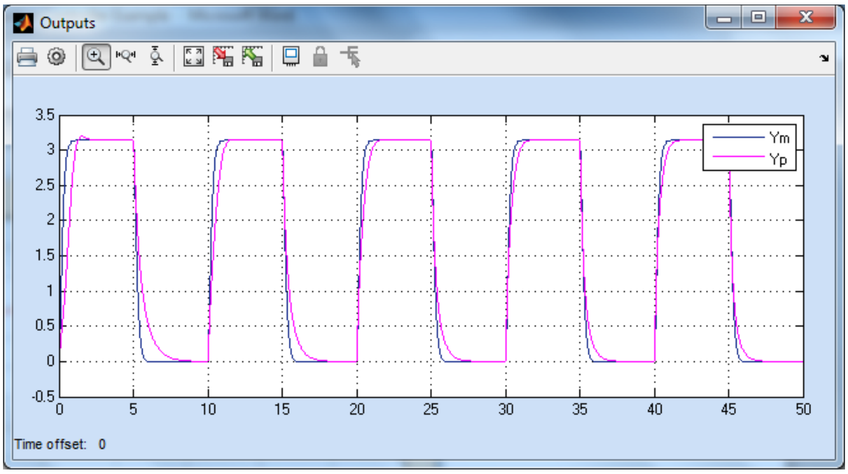
Step 2: Now change the plant behavior by increasing value of *Kf* from 0.2 to 1. This will result into different plant behavior initially and large error (*e*). But if our adaptive mechanism is working fast, it should reduce error as simulation progresses. We can see from plot below, *Yp* is approaching *Ym* slowly. You can run simulation longer to see effect properly.

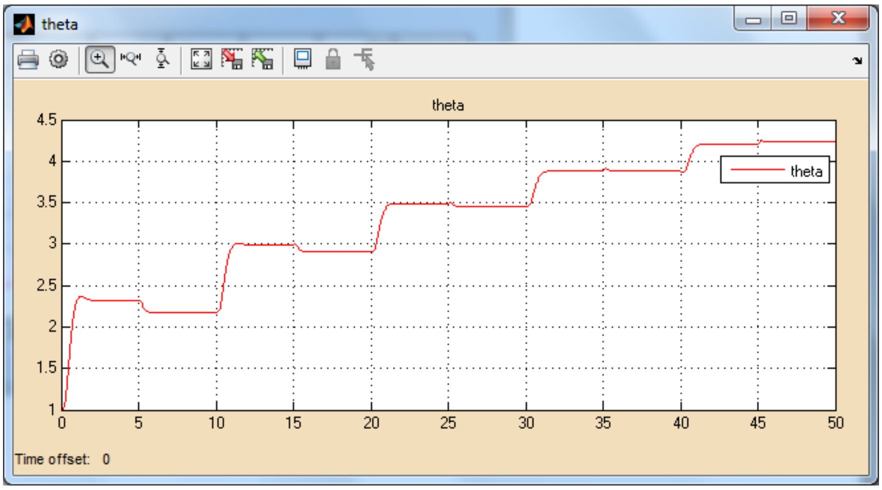


To understand how controller is adapting, look at *theta* plot below. As you can see, *theta* is going from its initial value 1 to 2.6 in 50 second and is not yet settled. If we run simulation log enough, theta will settle down at some higher values.



Step 3: Since results from Step 2 are not satisfactory, in terms of slow adoption rate, we can increase value of *gamma*, learning rate, to see if we can reduce the error (*Yp – Ym*) to acceptable limit in 50 second. Go ahead and increase value of gamma from 0.1 to 0.5 and run model. You can see simulation results below, where plant response is improving during simulation due to fast changing *theta*. This is a result of faster learning rate, *gamma*.





Step 4:

To understand the impact of *gamma*, plant parameters (*Kf* and others) and initial PID values, keep changing their values and understand how overall system behaves.