Week 2 Slides

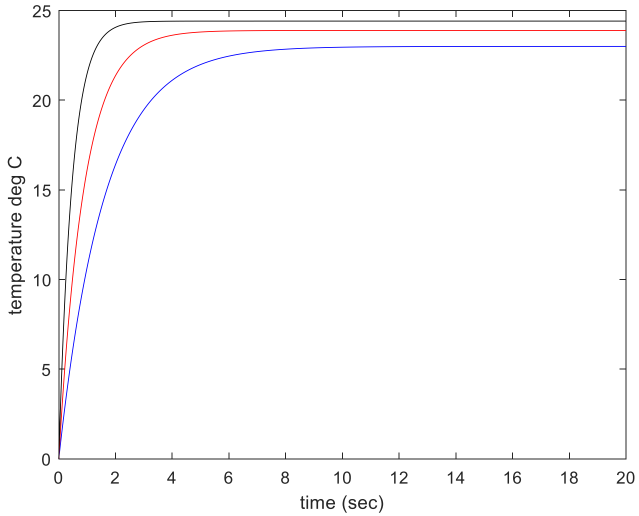
Recall from week 1 we did an overview of what control is about and how block diagrams and Simulink can help:

- What is control? Open-loop vs. closed-loop
- **System modelling using differential equations**
- **Converting model equations to block diagram form**
- Working with Simulink
- The integrator block and the **s-operator**

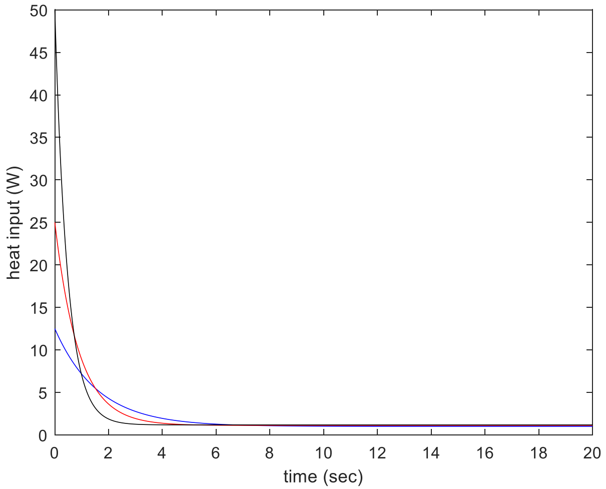
The highlighted topics are really important so we focus this week on going more deeply into them, with more examples – and a more creative approach to the s-operator.

First let's go back to tutorial questions 3 -5, those with closed-loop control. Running the closed-loop model (Temperature_02) with different values of

Kp = 0.5 (blue), 1 (red), 2 (black) gives the following temperature responses

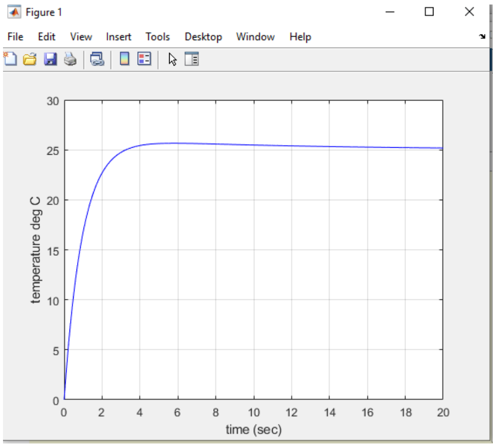


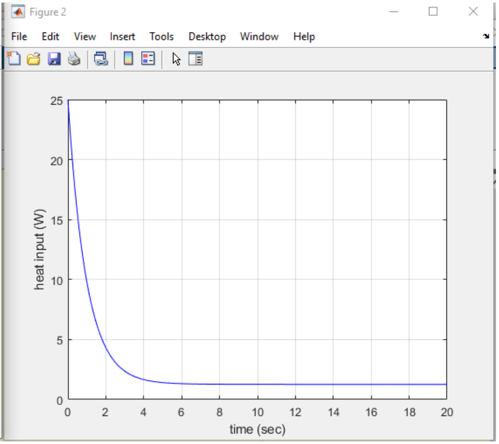
Increasing Kp gives a faster response and smaller steady-state error, but it does need some high levels of heat – here are the plots for Q1 (heat input):



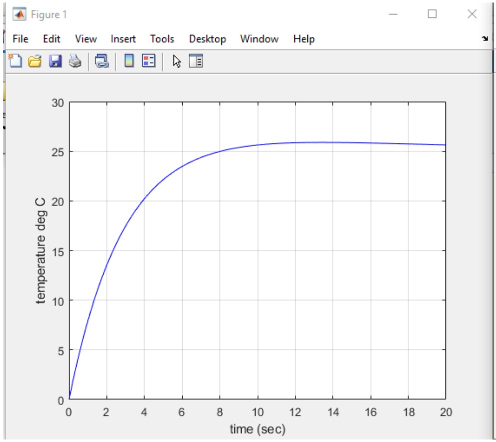
The higher gain requires a lot of heat to start with (and potentially more than the heater is capable of). In every case however the final heat level is low and not enough to reach the 25 degrees needed.

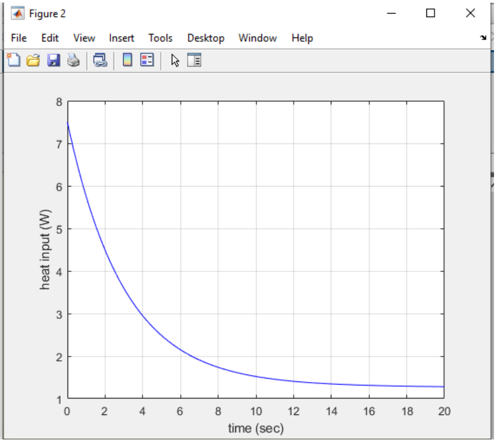
In tutorial question 5 the suggestion was to add an integral term to the controller. Here are the results with Kp=1, Ki=0.1. The temperature overshoots a little but it eventually settles down and without a very high intial heat demand





To be 'kind' to the heater, here are the result for Kp=0.3, Ki=0.03. It still overshoots for a while, but it settles close to 25 degrees and will eventually get there. This seems like a reasonable solution to the temperature control problem

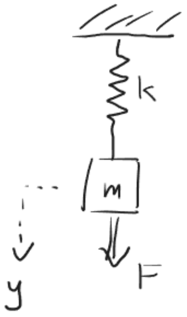




Can we explain the overshoot? Can we explain why it MUST reach the desired temperature eventually? Why is the final heat level the same, independent of the gains chosen?

Modelling Example

Now let’s look at modelling a simple “second order” plant, a mechanical system with a **mass and a spring**. The equations of motion are found from “force = mass x acceleration”. If y is the deflection from equilibrium we have



$$F - ky = m\ddot{y}$$

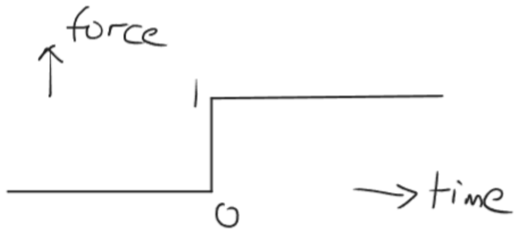
To keep in simple let’s assume $m = 1$ and $k = 1$. Our differential equation is then

$$\frac{d^2y}{dt^2} + y = F$$

and this is a “system model” for the plant, as a differential equation. F is input, y is output.

Later we shall try to control the height of the mass by using feedback.

Let’s concentrate on the plant model for now. Assume the mass is at rest in equilibrium, then we ‘switch on’ a unit (1 N) force (unit step input starting at time t=0)



We can solve the differential equation by the standard method of “particular integral + complementary function”. For t>0 the input is constant so…

particular integral: expect y=constant is a feasible solution, so

$$0 + y_{PI} = 1 \quad \text{so} \quad y_{PI} = 1$$

complementary function: we solve the equation for free motion (F=0):

$$\frac{d^2y}{dt^2} + y = 0$$

including arbitrary constants. We might already know the answer (or just check this is correct:

$$y_{CF} = A\cos t + B\sin t$$

(for any values of A and B). Hence

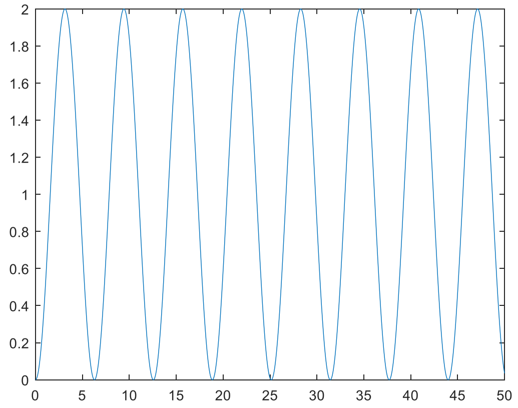
$$y(t) = y_{PI} + y_{CF} = 1 + A\cos t + B\sin t$$

Initial conditions are for equilibrium: (1) $y(0) = 0$ (2) $\dot{y}(0) = 0$ and these are used to set A and B. (1) gives $1 + A = 0$, and (2) gives $B = 0$.

Hence

$$y(t) = 1 - \cos t$$

and this is the final result. In Matlab we can plot the result using a one-line script: `t=0:0.01:50;y=1-cos(t);figure,plot(t,y)`

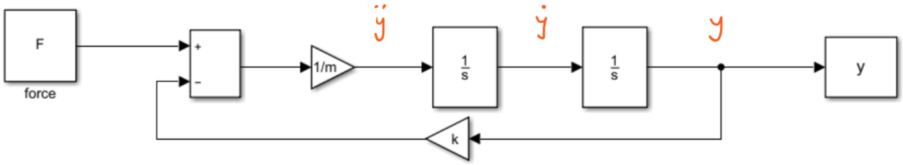


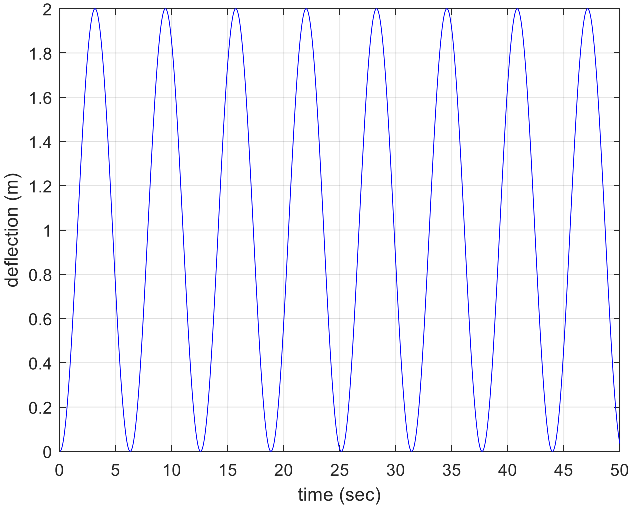
The step force causes the mass to move down then oscillate between zero and 2 meters (!) … and the oscillation persists forever (!!). In reality the spring stiffness may be larger and there may be some damping (see below).

Block diagram approach
We can also 'solve' the problem using a block diagram, by re-writing the model equation as

$$\ddot{y} = (1/m)F - (k/m)y$$

As before, we convert into block diagram format using integrator blocks, and then simulate in Simulink





Exactly as before. The model (**mass_spring_01**) and the calling script are posted on BB.

Tutorial Questions

1. Download and run the mass-spring model – confirm the results.
2. Increase the stiffness by a factor of 4. How does that affect the amplitude and frequency of vibration?
3. Reset the value of k and introduce damping, with force proportional to velocity, $F_{damping} = c\dot{y}$. Choose $c = 0.1$ and replot the results. Plot $y(t)$ with and without damping on the same Matlab plot. Use different colours and include a legend stating which has damping, which does not.

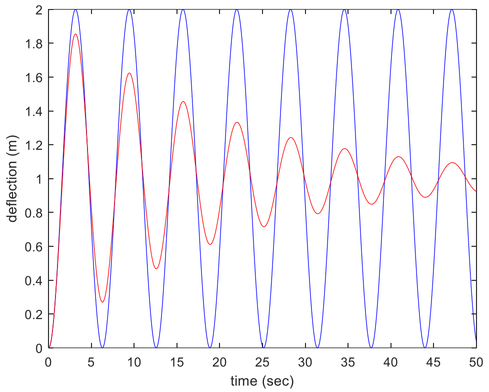
4. Set m=2, k=100, and choose c so that the **damping ratio** ζ equals 0.55.

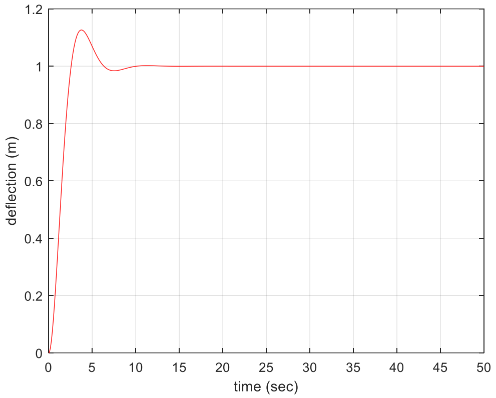
Method: the **standardized** second-order differential equation is written

$$\ddot{y} + 2\zeta\omega_n\dot{y} + \omega_n^2 y = \omega_n^2 u.$$

So, divide the model equation by m and match up the coefficients of y and $\dot{y}$. That gives enough information to find c.

Expected plots for questions 3 and 4 are shown below. The left plot uses c=0.1, the right plot uses the calculated value c=1.1.

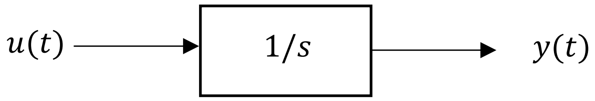




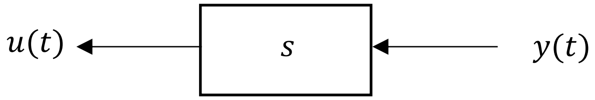
Damping ratio determines the general shape of the response and it is important! The second plot settles quickly and has an overshoot a little over 10% which is often acceptable in a control system. (Note, here is just the plant but the idea is the same, open-loop or closed-loop)

Introduction to Transfer Functions

Transfer functions are useful additions to the block diagram (new lego blocks). We are familiar already with the integrator



and the reasoning is the “s-operator” performs differentiation on a signal. We can also reverse the process



In both cases the block operates on the input signal to create the output signal. Both of these blocks are special cases of **transfer functions** – each a kind of a signal processor we can use in Simulink.

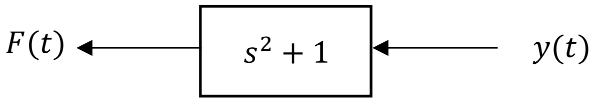
Going back to our mass-spring model

$$\ddot{y} + y = F$$

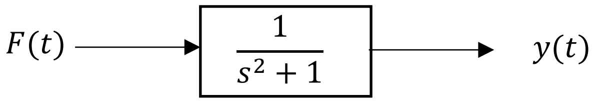
we can write this in operator format

$$(s^2 + 1)y = F$$

or

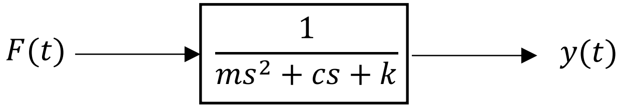


or



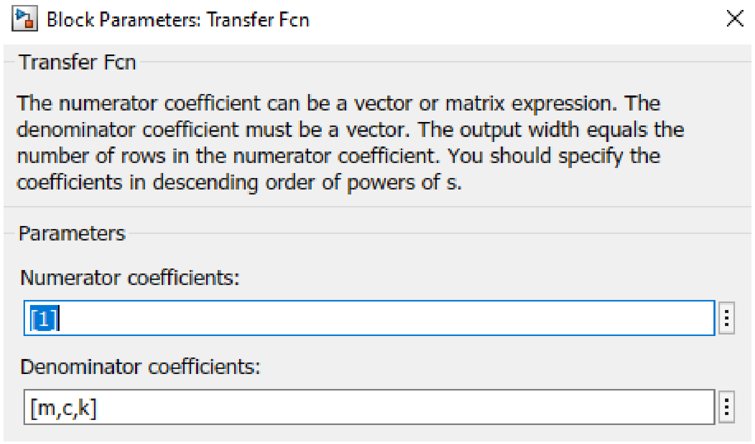
All these blocks are transfer functions and are available in Simlink

For the general mass-spring-damper 'plant' we find, in a similar way



$$F(t) \longrightarrow \boxed{\frac{1}{ms^2 + cs + k}} \longrightarrow y(t)$$

In Simulink the block is called "Transfer Fcn"



Block Parameters: Transfer Fcn

Transfer Fcn

The numerator coefficient can be a vector or matrix expression. The denominator coefficient must be a vector. The output width equals the number of rows in the numerator coefficient. You should specify the coefficients in descending order of powers of s.

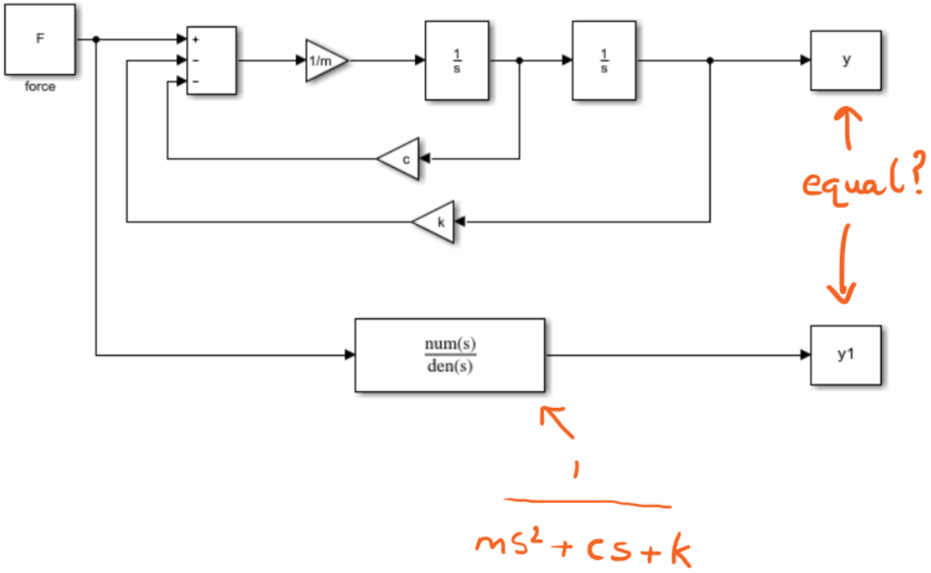
Parameters

Numerator coefficients:

[1]

Denominator coefficients:

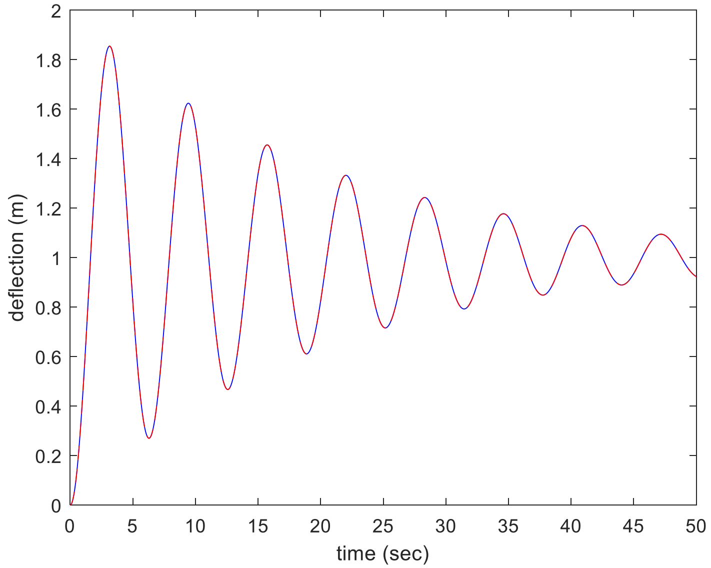
[m,c,k]


F
force
+
-
-
1/m
1/s
1/s
y
c
k
num(s)/den(s)
y1
equal?
$\frac{1}{ms^2+cs+k}$

Matlab script for running the model in Simulink:

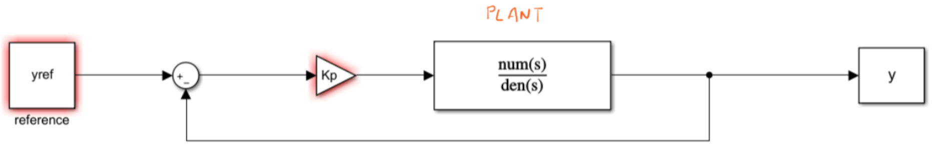
```
%script to run simulation mass_spring_03
clear
close all
%set parameter values
model='mass_spring_03';
m=1; %kg
k=1; %spring constant
F=1; %constant force
%add damping
c=0.1; % tutorial question 3

sim(model,50)
figure(1)
plot(tout,y,'b'),grid, hold on
xlabel('time (sec)')
ylabel('deflection (m)')
plot(tout,y1,'r--'),grid, hold on
```

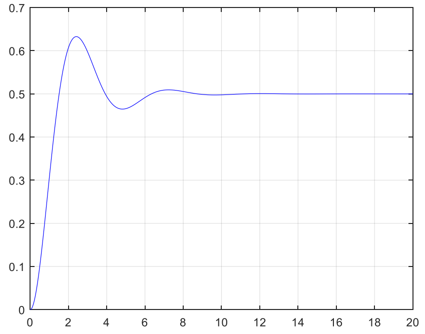


The blue and red-dash curves are exactly the same, as expected.

Finally let's close the loop to make a control system, using proportional control

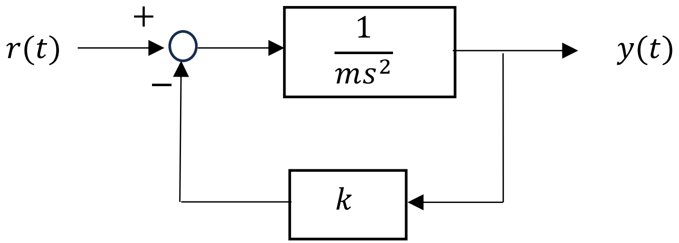


here we set c=1.1 in the plant and use Kp=1.

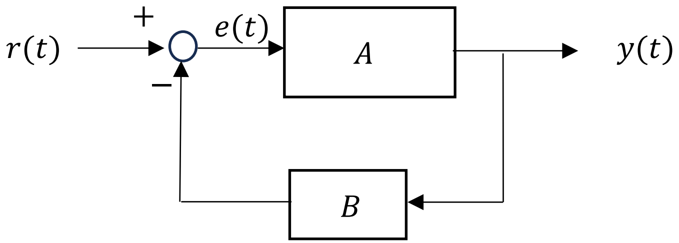


The result isn't great – we see there is a steady-state error, but that's OK, we already know about integral control so maybe that will help (leave this for later).

The more important point is that we can **reduce block diagrams** to simplify them. Let's try to do this directly in the block diagram using the undamped system of Slide 10. Clearly the block diagram there is equivalent to following



This has the following structure (with negative feedback)



[room for working out]

The **closed-loop transfer function** takes the form

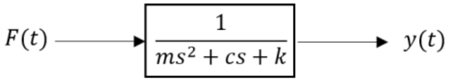
$$T(s) = \frac{A}{1 + AB}$$

= forward path/(1+loop)

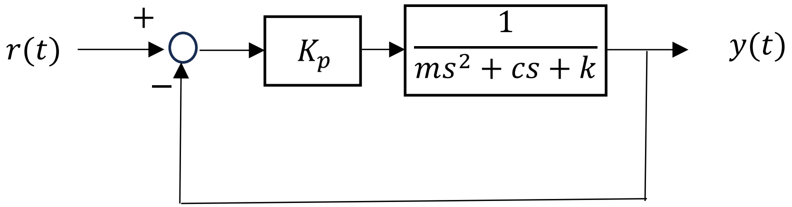
Use this for the undamped system:

$$T = \frac{\frac{1}{ms^2}}{1 + \frac{k}{ms^2}} = \frac{1}{ms^2 + k}$$

in agreement with slide 16 (when c=0)



Finally, let's use the feedback rule when we apply proportional feedback control for the mass-spring-damper plant (general case)

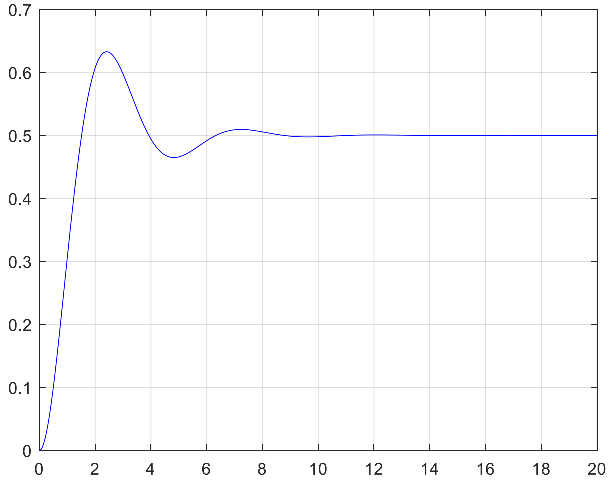


Now the feedback loop is to form the control loop

$$T = \frac{\dfrac{K_p}{ms^2+cs+k}}{1+\dfrac{K_p}{ms^2+cs+k}} = \frac{K_p}{ms^2+cs+(k+K_p)}$$

The effect of feedback is to increase the stiffness in the transfer function, which may not be good enough in practice.

For the case $m = 1, c = 1.1, k = 1, K_p = 1, r = 1$ we find



which is not particularly good (lots of steady-state error!). But at the moment we are not designing a controller, and hopefully we can fix this later!

Summary

So far we have focused on modelling based on simple physics – and therefore creating either a differential equation or a block diagram to represent the physics. We have then seen how to

- convert a differential equation to a block diagram using the 1/s operator only
- convert a differential equation to a transfer function using the s operator
- simplify a block diagram using products and feedback loops - using the rule: **forward path/(1+loop)**
- use the transfer function block in Simulink

Note that for transfer functions we always assume zero initial conditions, unlike the integrator block where we can set this. So in some cases we may prefer to use the integrator as part of the block diagram.

Tutorial Questions (continued)

5. Download and run mass_spring_03 which has both the basic block diagram and the transfer function version – confirm the results shown above.
6. Change some plant parameters and again check that the two signals y(t) and y1(t) give the same results.
7. Adapt the Simulink model **mass_spring_03** to include proportional control. You can throw away the original block diagram and close the loop using the transfer function model. Using the parameters given about (including Kp=1) check the closed-loop response.
8. Try to improve the control in some way, e.g. by changing the value of Kp. Maybe you can do better by using a PID controller?