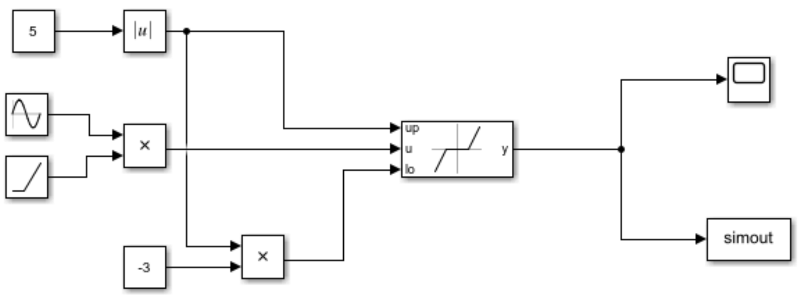
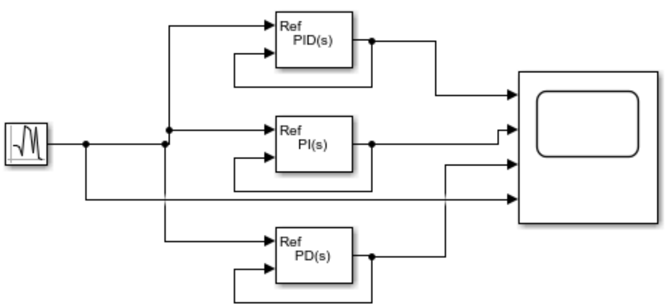
Table of Contents

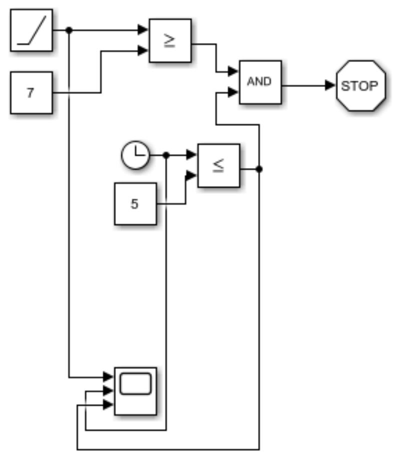

```
clear
clc

% Exercise 3
% Group 13
% Jakob Fichtl - 29450
% Michael Zappe - 29901
```

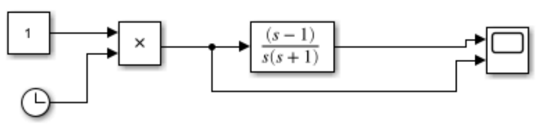
a)

```
playtime

% 1 (Deadzone)
%  New blocks: Dynamic Deadzone, Absolute, Sinus, Ramp
%  Comment: Dynamic Deadzone maps the input value 'u' to 0 if its
 between 'lo' and 'up'. If 'u' exceedes the lower or upper bound it
 gets damped by the corresponding bound.
%    Absolute: Maps any negative value to its positive value.
%    Sinus: Produces a discrete sinus signal according to the
 simulation settings.
%    Ramp: Produces a discrete signal, that has a delay and after that
 delay can be described by the form y = a * (time - delay).
%
% 2 (PID)
%  New blocks: PID, Random Number
%  Comment: PID: Has three modes PID, PI and PD. Controls the output
 signal to match the input signal, but remove sudden jumps in the
 signal.
%    Random Number: Generates random numbers.
%
% 3 (Zero-Pole)
%  New blocks: Zero-Pole
%    Zero-Pole: Is a rational transfer function.
%
% 4 (Logic)
%   New blocks: AND, Lower/Bigger or equal, STOP
%    AND: Combines two logic signals (0 = false or non 0 = true).
%    Lower/Bigger or equal: Checks a condition and either outputs 1 or
 0.
%    STOP: Stops the complete simulation if a 1 is passed to its
 input.
```



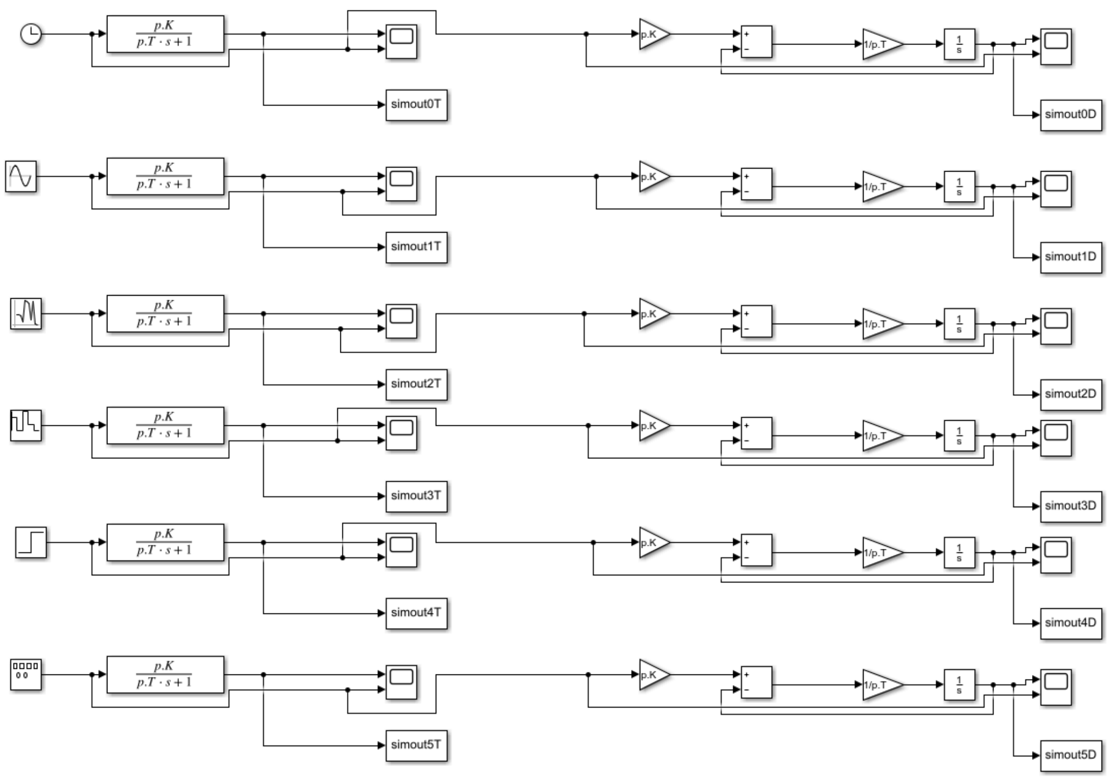




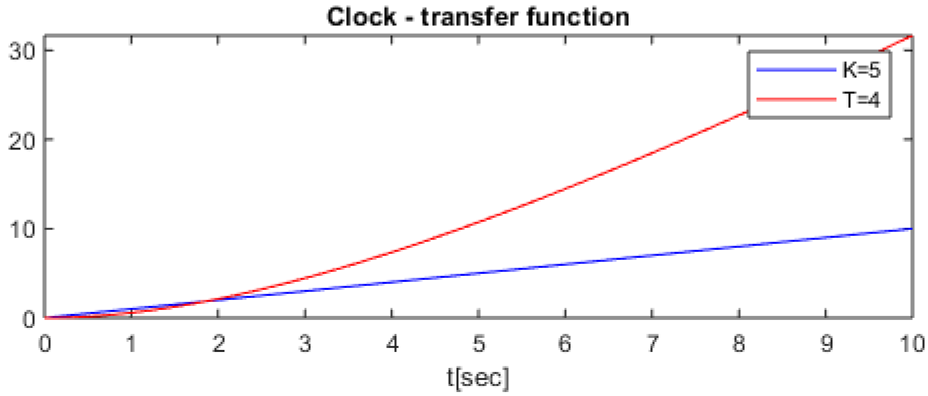


b)

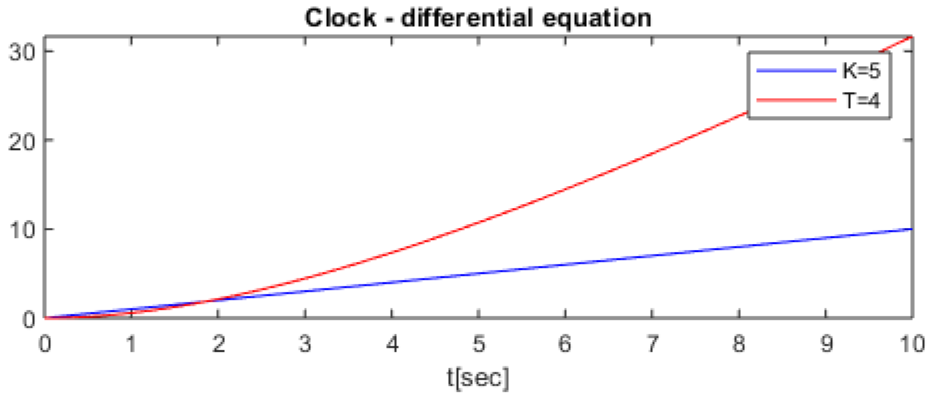
```
exercise_3b
clear
clc
p.TSim = 10;
p.K = 5;
p.T = 4;

sim('exercise_3b', [0 p.TSim]);

showPlot(scope0T, scope0D, "Clock", 1);
showPlot(scope1T, scope1D, "Sinus", 2);
showPlot(scope2T, scope2D, "Random Number", 3);
showPlot(scope3T, scope3D, "Repeating Sequence Stair", 4);
showPlot(scope4T, scope4D, "Step", 5);
showPlot(scope5T, scope5D, "Signal Generator", 6);
```



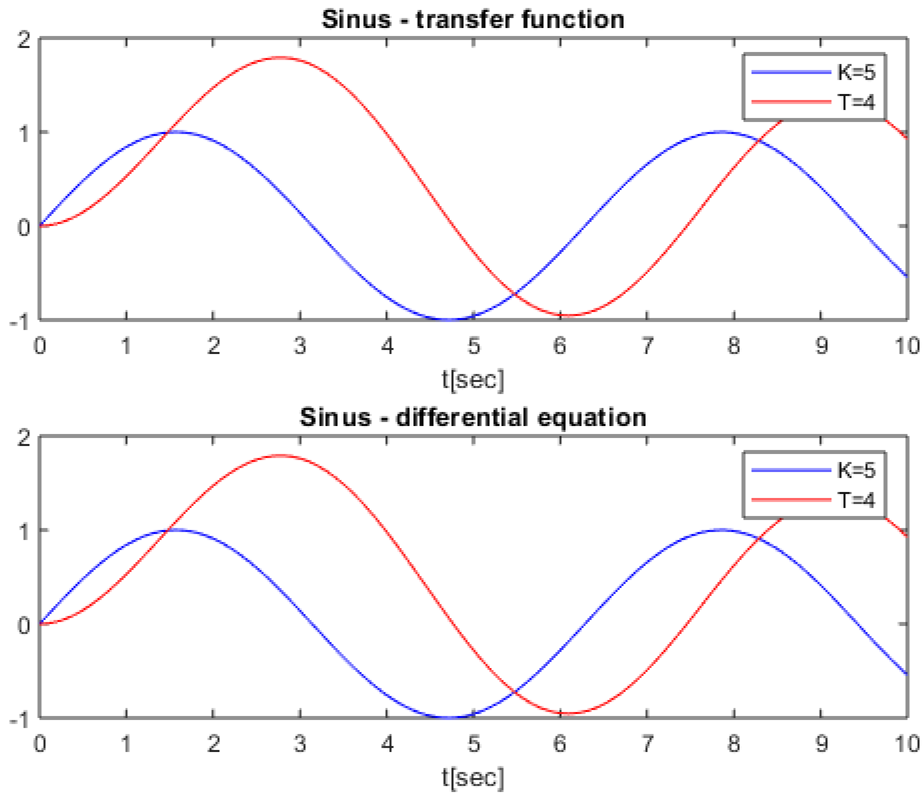
simout0T
simout0D
simout1T
simout1D
simout2T
simout2D
simout3T
simout3D
simout4T
simout4D
simout5T
simout5D

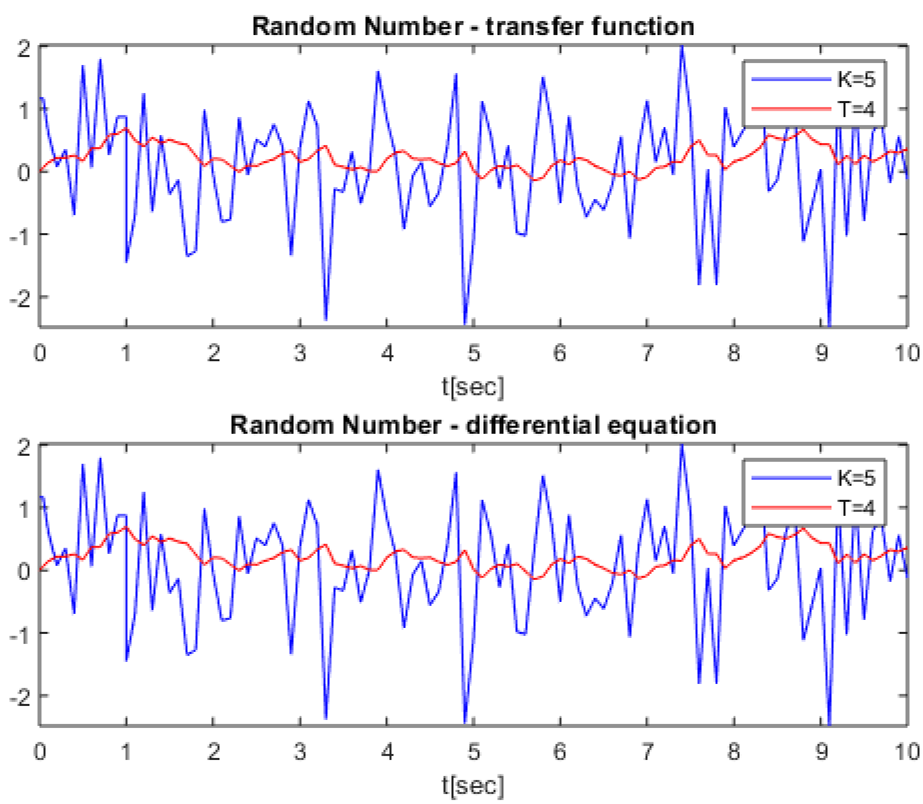


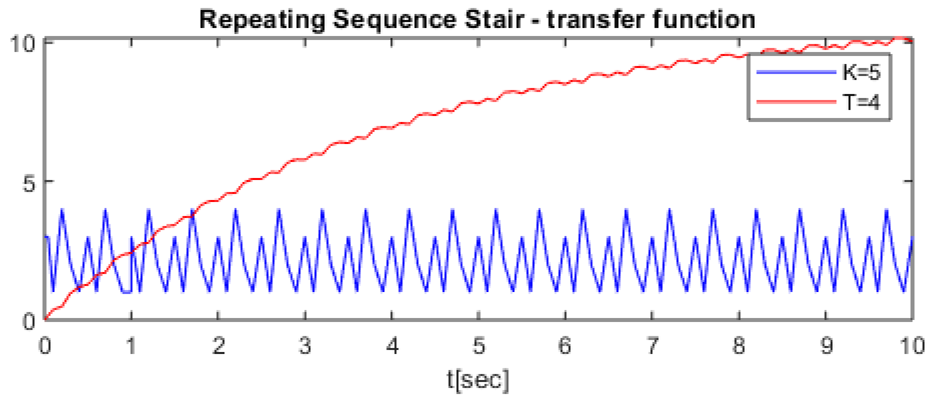
Clock - transfer function
K=5
T=4
t[sec]

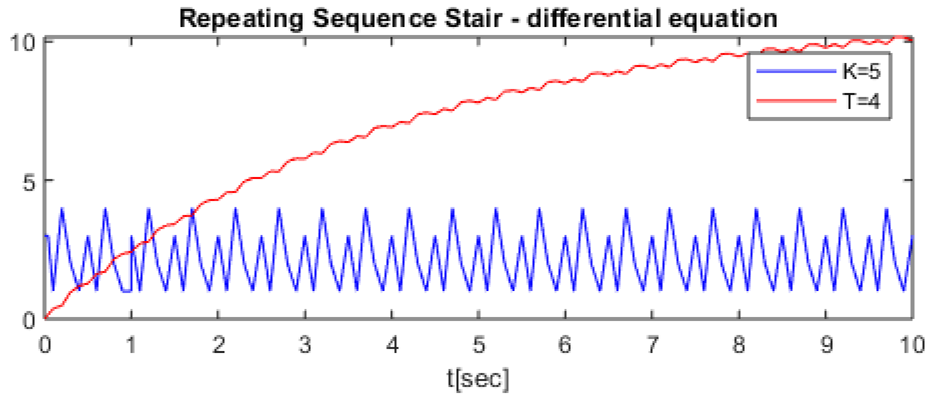


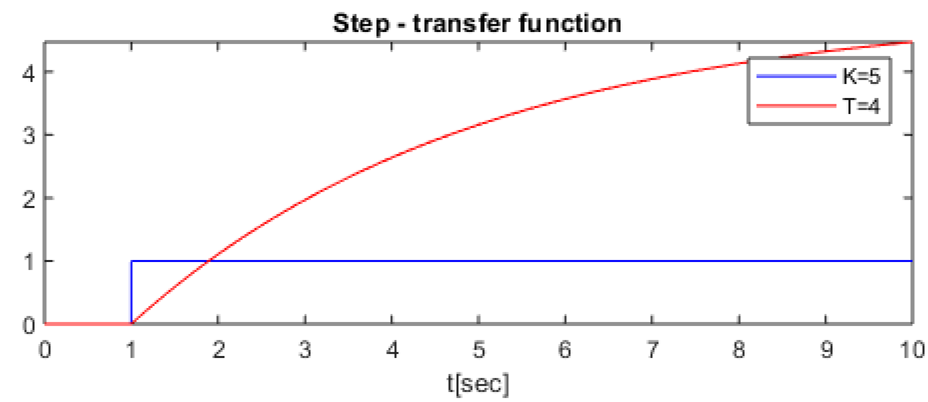
Clock - differential equation
K=5
T=4
t[sec]

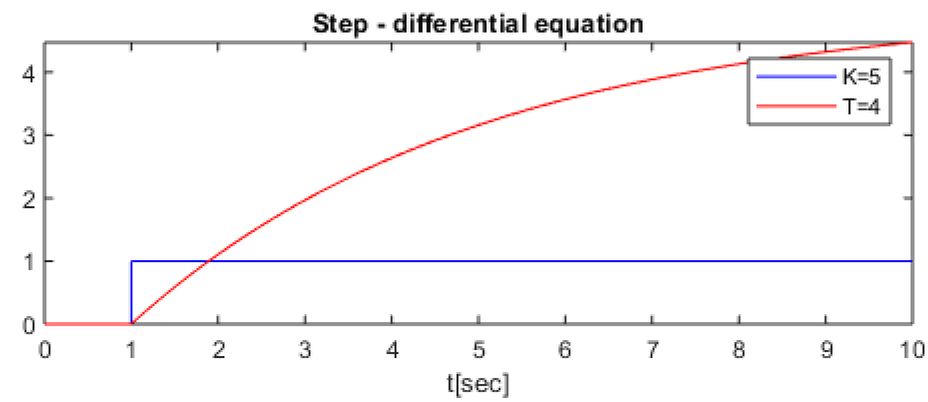

Sinus - transfer function
K=5
T=4
t[sec]
Sinus - differential equation
K=5
T=4
t[sec]

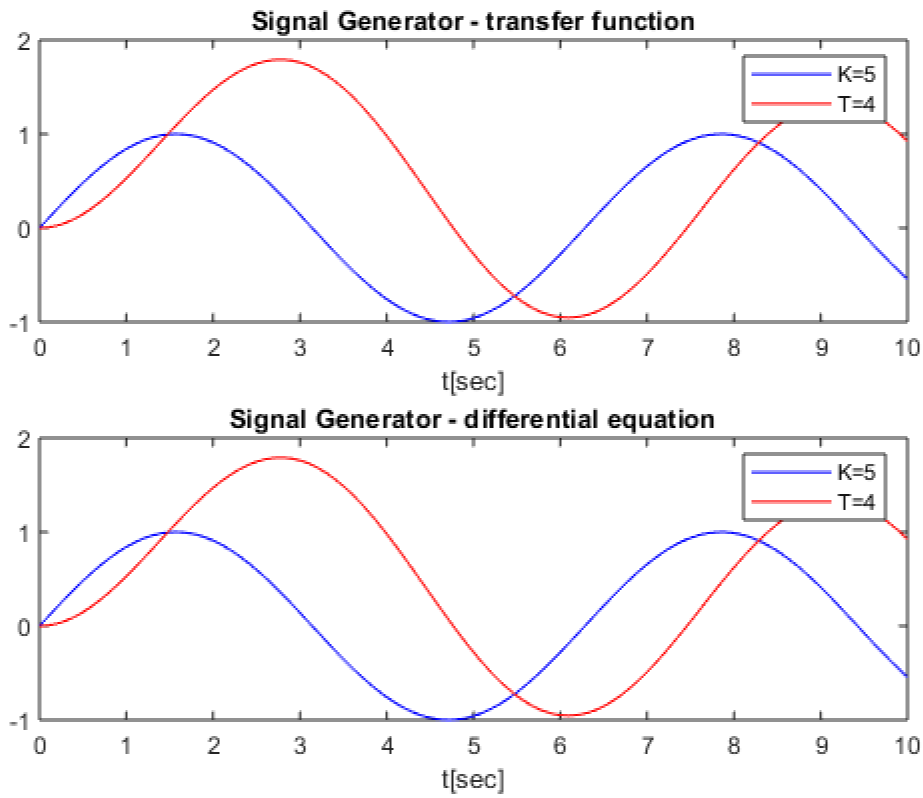

Random Number - transfer function
K=5
T=4
t[sec]
Random Number - differential equation
K=5
T=4
t[sec]


Repeating Sequence Stair - transfer function
K=5
T=4
t[sec]


Repeating Sequence Stair - differential equation
K=5
T=4
t[sec]


Step - transfer function
K=5
T=4
t[sec]


Step - differential equation
K=5
T=4
t[sec]

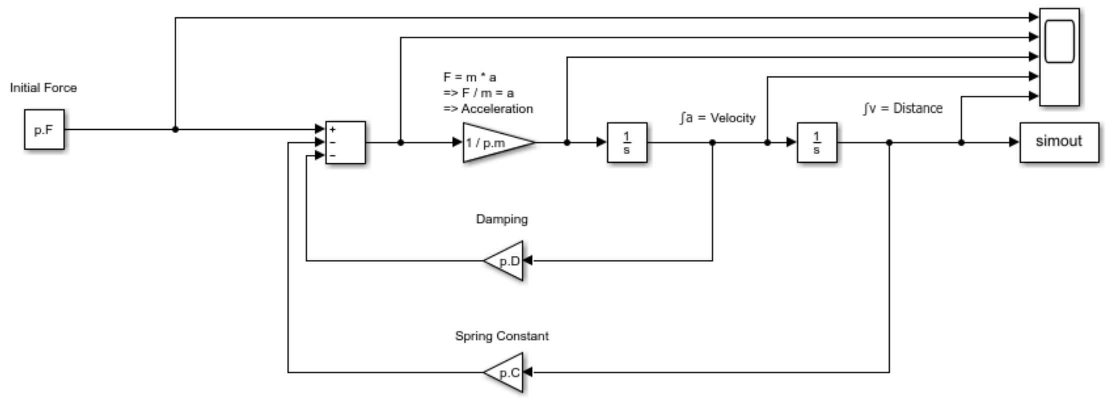


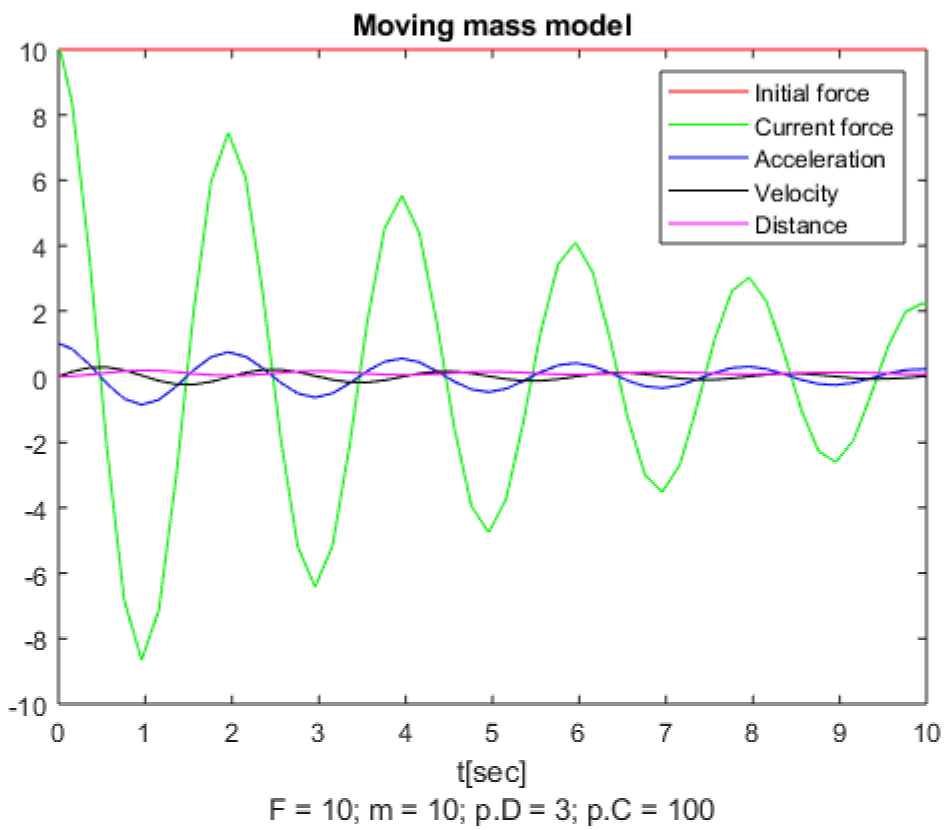
c)

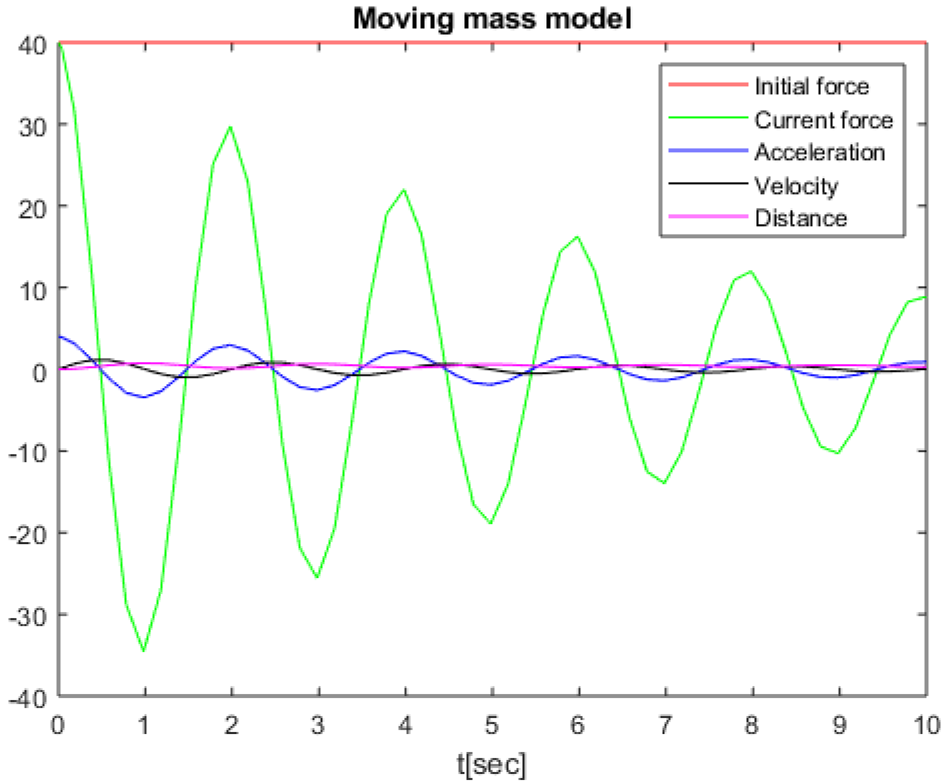
```
exercise_3c
clear
clc

p.TSim = 10;
p.F = 10;           % [N] inital force
p.m = 10;           % [kg] mass
p.D = 3;            % [N sec/m]damping
p.C = 100;          % [N/m] spring constant
runThis(p, 7)

p.F = 40;           % [N] inital force
runThis(p, 8)

p.F = 10;           % [N] inital force
p.m = 20;           % [kg] mass
runThis(p, 9)

p.m = 10;           % [kg] mass
p.D = 0.1;          % [N sec/m]damping
runThis(p, 10)
```

```
p.D = 3;          % [N sec/m]damping
p.C = 1337;       % [N/m] spring constant
runThis(p, 11)


p.F = 1;          % [N] inital force
p.m = 10;         % [kg] mass
p.D = -3;         % [N sec/m]damping
p.C = 100;        % [N/m] spring constant
runThis(p, 12)

p.F = 10;         % [N] inital force
p.m = 5;          % [kg] mass
p.D = 5;          % [N sec/m]damping
p.C = -2;         % [N/m] spring constant
runThis(p, 13)

function runThis(p, plotNr)
    sim('exercise_3c', [0 p.TSim]);


    figure(plotNr)
    y1 = scope;
    plot(y1.time, y1.signals(1).values,'r', ...
         y1.time, y1.signals(2).values,'g', ...
         y1.time, y1.signals(3).values,'b', ...
         y1.time, y1.signals(4).values,'black', ...
         y1.time, y1.signals(5).values,'m');

    title("Moving mass model");
    legend("Initial force", "Current
 force", "Acceleration", "Velocity", "Distance");
    str = "F = " + p.F + "; m = " + p.m + "; p.D = " + p.D + "; p.C =
 " + p.C;
    xlabel({"t[sec]";str});
end

function showPlot(scopeT, scopeD, name, plotNr)
    figure(plotNr);

    y1 = scopeT;
    subplot(2,1,1)
    plot(y1.time, y1.signals(2).values,'b', ...
         y1.time, y1.signals(1).values,'r');

    title(name + " - transfer function");
    legend('K=5','T=4');
    xlabel("t[sec]");

    y2 = scopeD;
    subplot(2,1,2)
    plot(y2.time, y2.signals(2).values,'b', ...
         y2.time, y2.signals(1).values,'r');
```

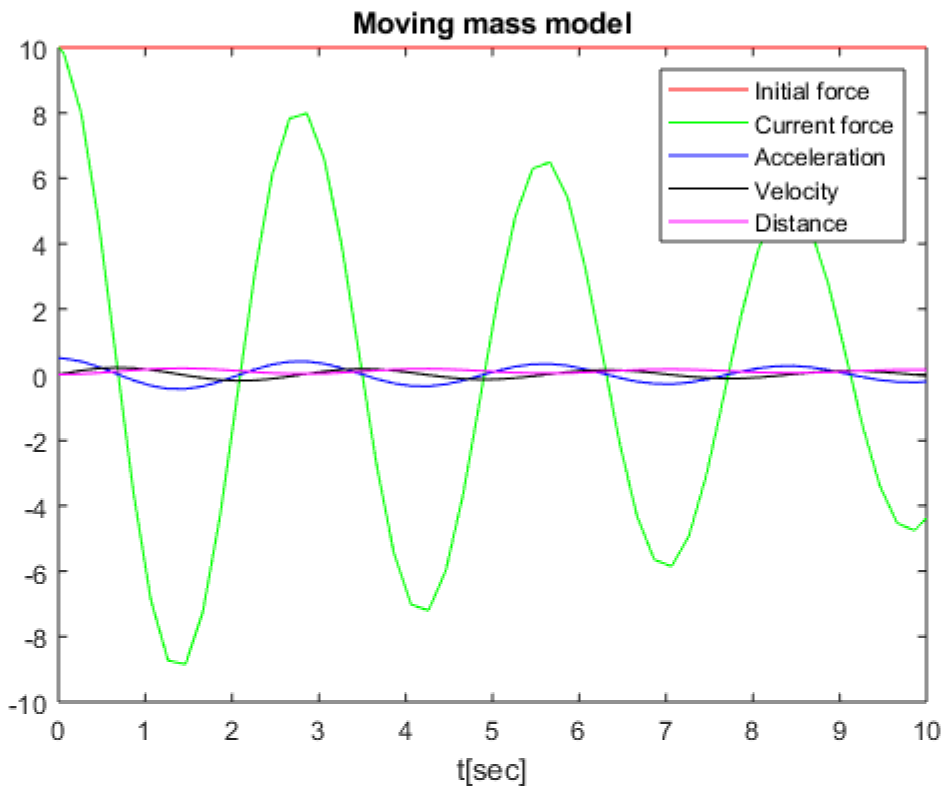
```
    title(name + " - differential equation");
    legend('K=5','T=4');
    xlabel("t[sec]");
end
```



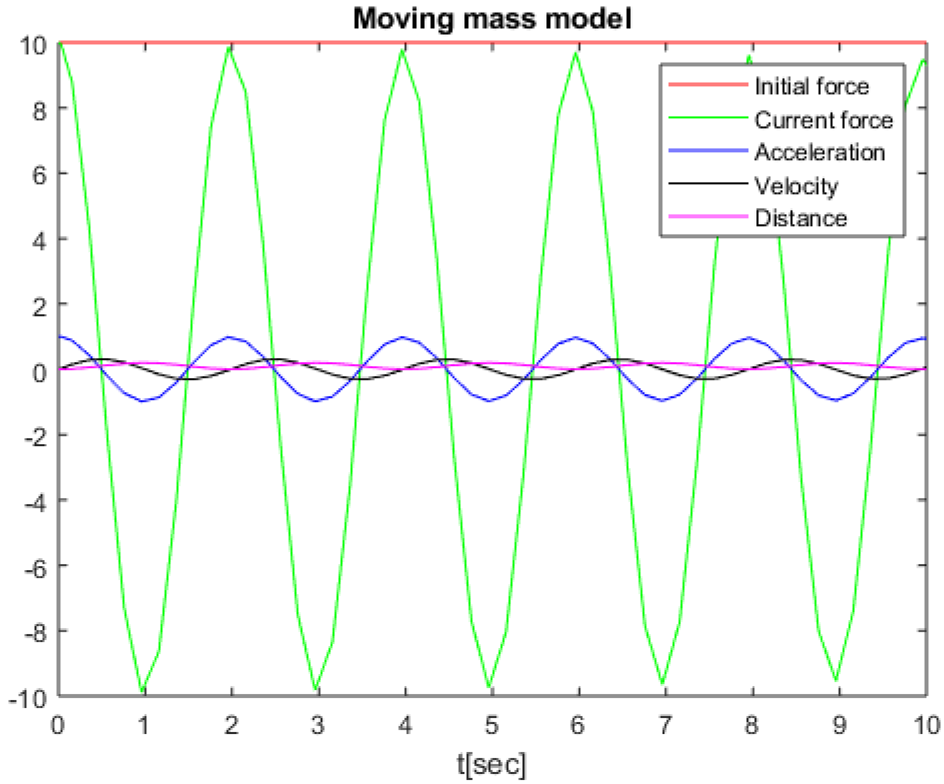




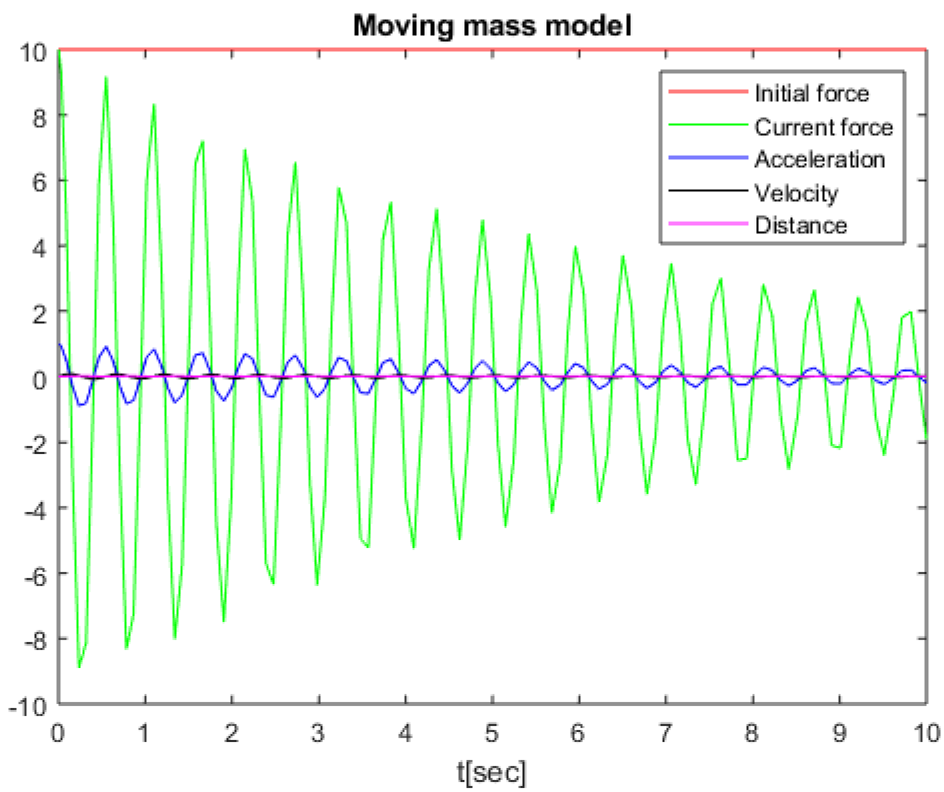
F = 40; m = 10; p.D = 3; p.C = 100



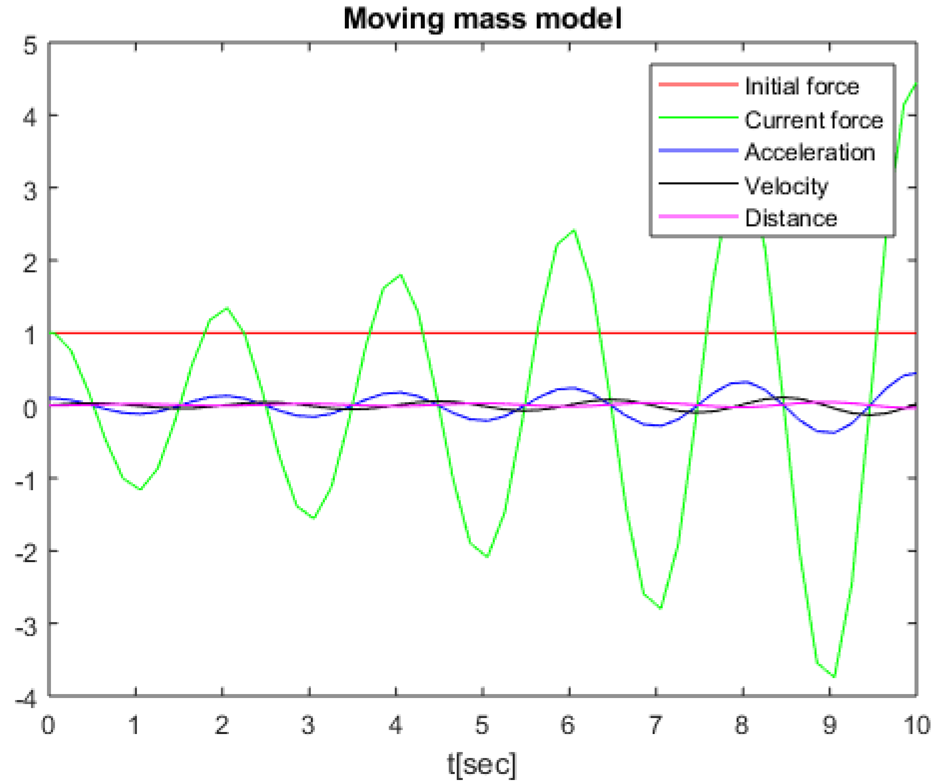
F = 10; m = 20; p.D = 3; p.C = 100



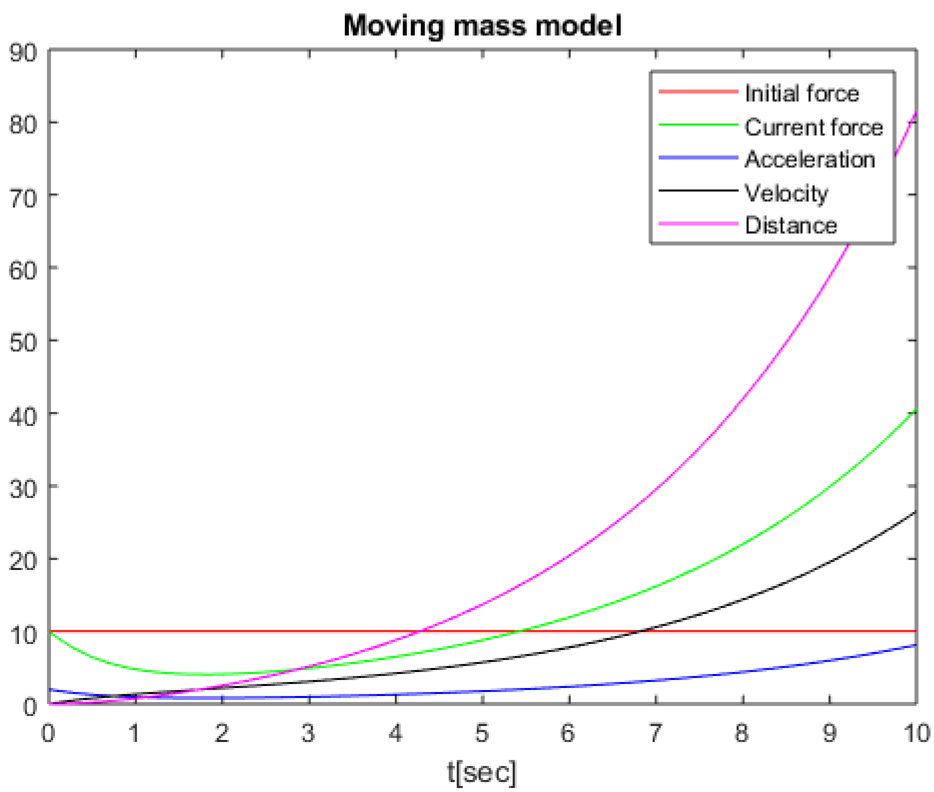
F = 10; m = 10; p.D = 0.1; p.C = 100



F = 10; m = 10; p.D = 3; p.C = 1337



F = 1; m = 10; p.D = -3; p.C = 100



F = 10; m = 5; p.D = 5; p.C = -2

Published with MATLAB® R2018a