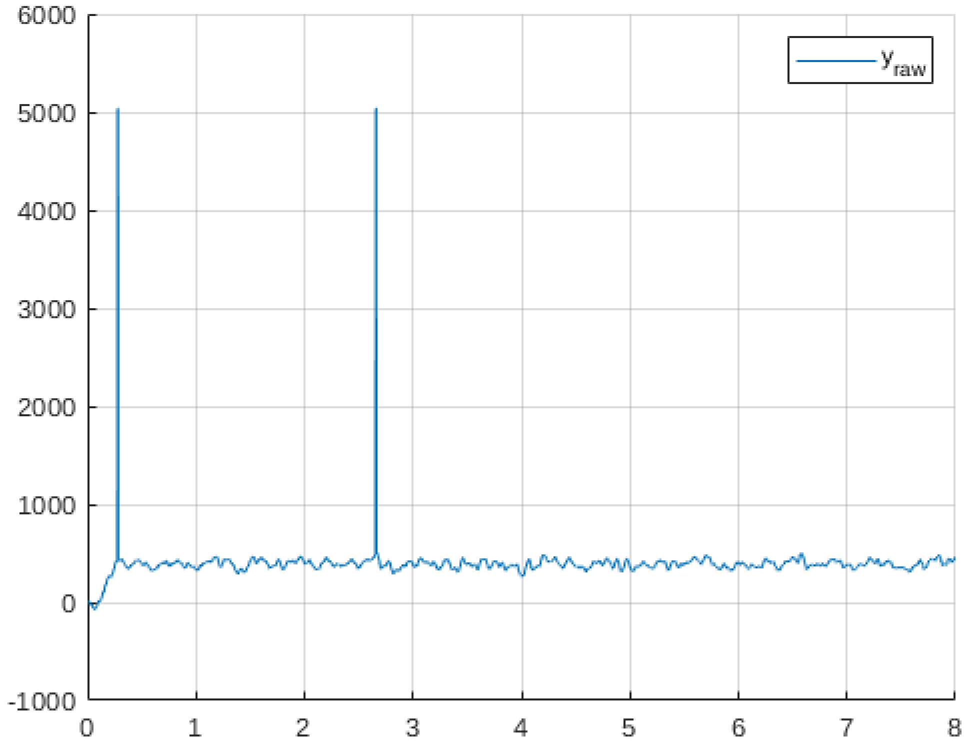
Table of Contents

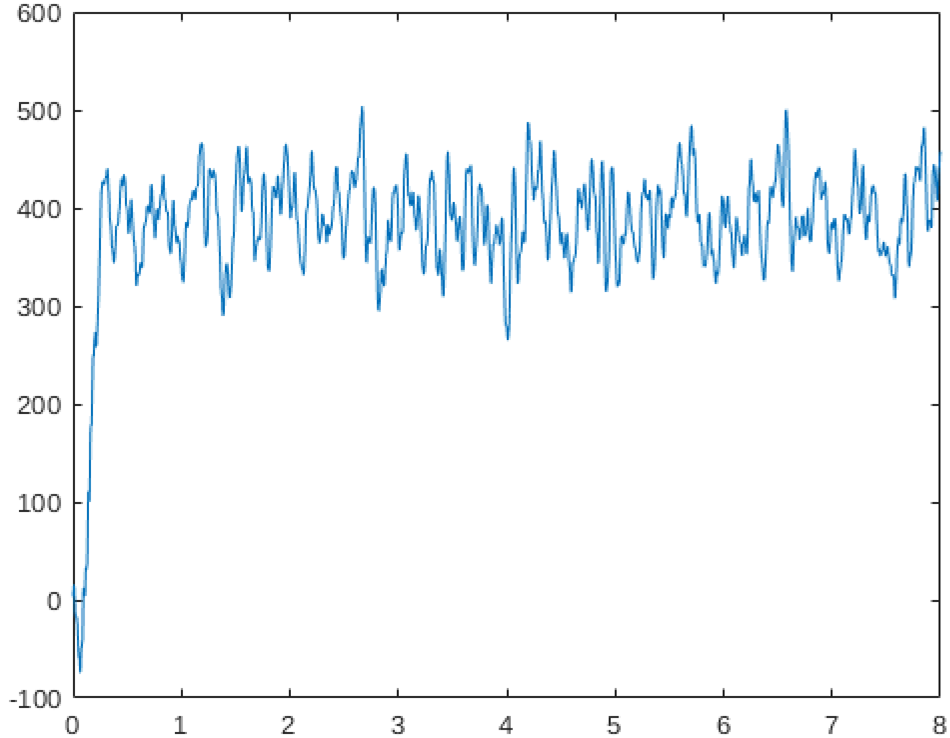
Assignment 1

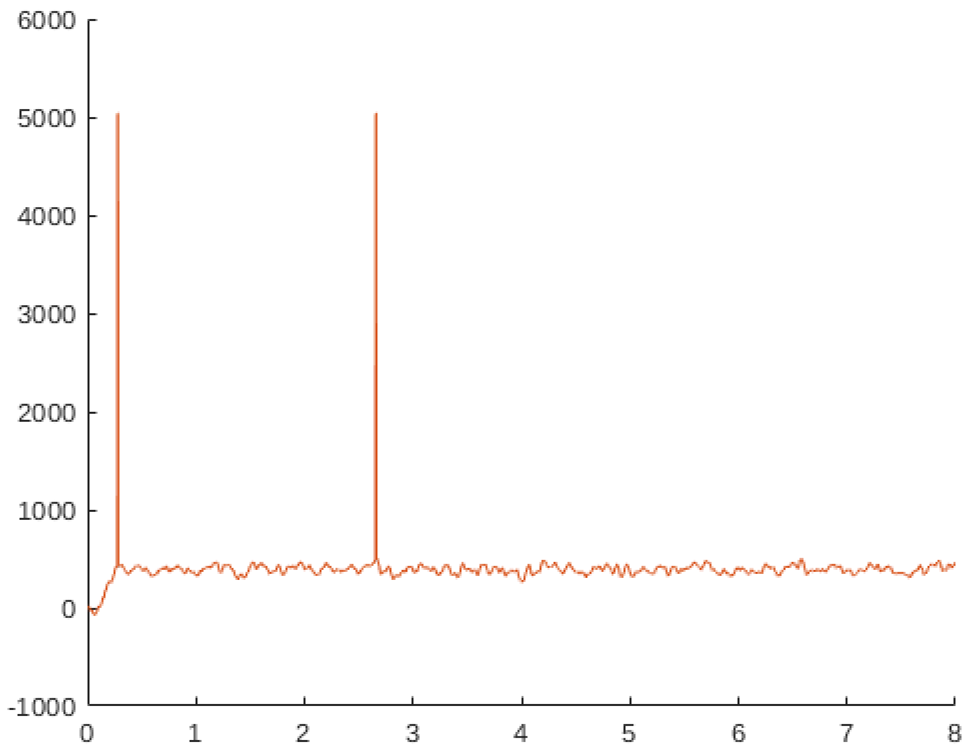
```
clear; clc;
fs = 44; % Hz
dt = 1/fs;
t_gen = 0:dt:8-dt;
u =7*uGen(t_gen, "step",1,9);

% Persistently exciting input signal (generated from unit step ut)
U = genU(u); % works only for unit step input

yraw = exciteSystem(5360188, U, fs);
ytest = exciteSystem(5360188,u,fs);
t = 0:dt/(countZeros(u)+1):(8)-dt/(countZeros(u)+1);

% The peak time is at .87 seconds and the signal starts rising after a
% delay of .45 seconds. This means that the rise time is about .42
 seconds.
% An appropriate sample rate would be to have 8 or 9 samples in this
 time period.
% So the sample time interval shoud be .42/9=.0467 seconds. i.e. a
 sampling
% frequency of 22Hz when rounded up. In hindsight, an announcement was
 made
% that the signal generation process is correllated to the sampling
% frequency and the we were allowed to eyeball a good frequency.
 Double the
% found frequency (44Hz) gives a nice workable result.

% the peak of the rise after a time delay was determined to be ca. .8
% seconds. This time was multiplied by 10 and taken as an appropriate
% duation for the simulation.

figure(1);
clf; hold on; grid on;
plot(t,yraw)
legend("y_{raw}")

clear ytest
```



spikes

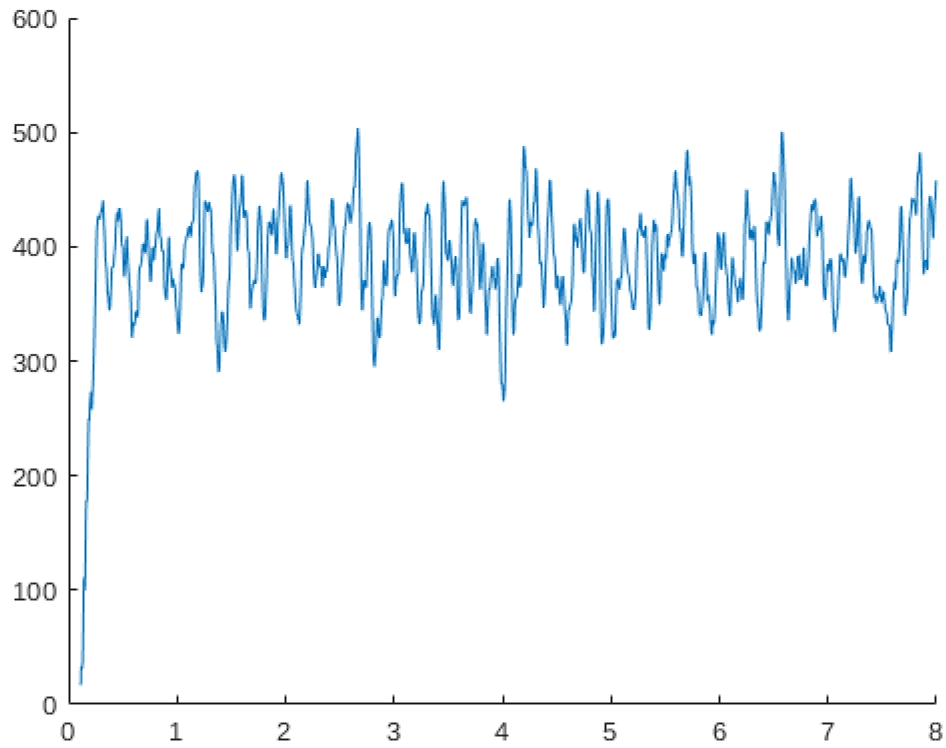
```
y = despike(yraw,10000,fs);

figure(2)
clf; grid on;
plot(t,y)

figure(1)
clf; hold on;
plot(t,y)
plot(t,yraw)
```


600
500
400
300
200
100
0
-100
0
1
2
3
4
5
6
7
8


6000
5000
4000
3000
2000
1000
0
-1000
0
1
2
3
4
5
6
7
8

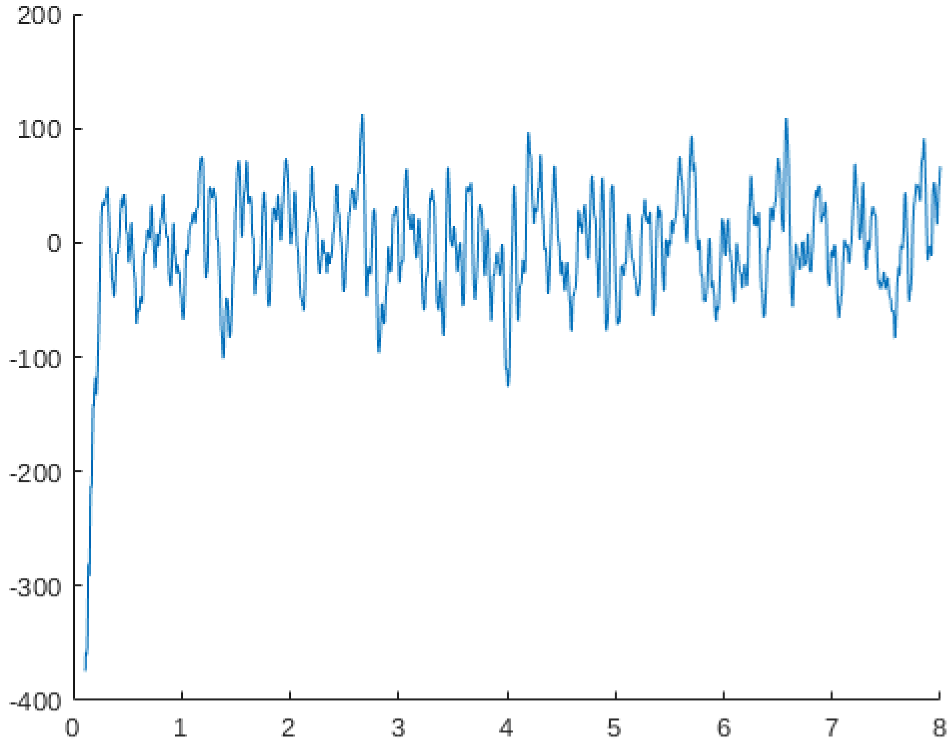
Timeshift

```
y = timeshift(y,500,fs);

cutoff = length(yraw)-length(y);
t_shifted = t(cutoff+1:end);

figure(3)
clf; hold on;
plot(t_shifted,y)

clear cutoff;
```



DC Offset

```
y = DCoffset(y);

figure(4)
clf; hold on;
plot(t_shifted,y)
```



Linearity Check

```
table = [];
fprintf("%0s | %10s \n","Input gain","IO gain")
for i = 1:20
 ut =i*uGen(t_gen, "step",1,9);

% Persistently exciting input signal
 Ut = genU(ut); % works only for unit step input
 yt = exciteSystem(5360188, Ut, fs);
 iogain = IOgain(ut,yt,fs);
%  fprintf("%10.0i | %f \n",ut(end), iogain)
 table = [table ; [ut(end) iogain]];
 fprintf("%-10.0i | %2.5f \n", ut(end), iogain)
end

clear ut Ut;

% The table below shows IO gains for a varyety of input gains. The IO
 gains
% seems constant by aproximation, which indicates linearit of the
 system.
% The system roughly behaves as: y*Igain=IOgain*u*Igain -->
 y=IOgain*u. I.E
% scaling the input gain scales the output gain with the same factor,
```

```
% resulting in a roughly constant IO gain of ca. 58.
```

Input gain | IO gain

Functions Assignment 1

```
function u = uGen(time,type, amp, periods)
 dt = time(2)-time(1);
 if type=="step"
  u = [zeros(periods,1) ; ones(length(time)-periods,1)]*amp;
 elseif type == "pulse"
  u = [1 ; zeros(length(time)-1,1)];
 elseif type == "sine"
  u = amp*sin(periods*time*2*pi/(dt*length(time)));
 else
  u = "Unknown input type";
 end
end

function z = countZeros(u) % counts zeros before unit step
    z = 0;
    while u(z+1)==0
        z=z+1;
    end
end

function U = genU(u)
    % Generates Persistently exciting unit step
    z = countZeros(u);
    sysOrder = z+1;
    gain = max(u);
    U = gain*ones(sysOrder,length(u));
    U(1,:) = u';
    for i=2:sysOrder
       U(i,:) = [u(i:end)' gain*ones(1,i-1)];
    end
end

function y = interp(sig)
 done = false;
 i = 2;
 while ~done
  if ~(sig(i) == sig(1))
   dy = sig(i)-sig(1);
   dx = i;
   for j=2:i
    sig(j) = sig(j-1)+dy/dx;
   end
   done = true;
  end
  i = i+1;
 end
 y = sig;
```

```
end

function y = despike(sig,slope,fs)
 spikes = [];
    for i = 2:length(sig)
        if fs*(sig(i)-sig(i-1))>slope
            sig(i) = sig(i-1);
   if  ~(ismember(i-3,spikes)) && ~(ismember(i-2,spikes))
    spikes = [spikes i-1];
   end
        end
 end
 for i = 1:length(spikes)
  sig = [sig(1:spikes(i)-1) ; interp(sig(spikes(i):end))];
 end
    y =sig;
end

function y = timeshift(sig,slope,fs)
    i = 2;
 done = false;
 while ~done
        if fs*(sig(i)-sig(i-1))>slope && sig(i-1)>=0
            sig = sig(i-1:end);
   done = true;
  end
  i = i+1;
 end
    y =sig;
end

function y = DCoffset(y)
 done = false;
 i=2;
 while ~done
  if y(i) > y(i-1) && y(i) > y(i+1) && y(i) >400
   sig = y(i:end);
   done = true;
  end
  i = i+1;
 end
 DC = mean(sig);
 y = y-DC;
end

function g = IOgain(u,y,fs)
 y = despike(y,10000,fs);
 y = timeshift(y,500,fs);
 g = mean(y)/mean(u);
end
```

Published with MATLAB® R2020a