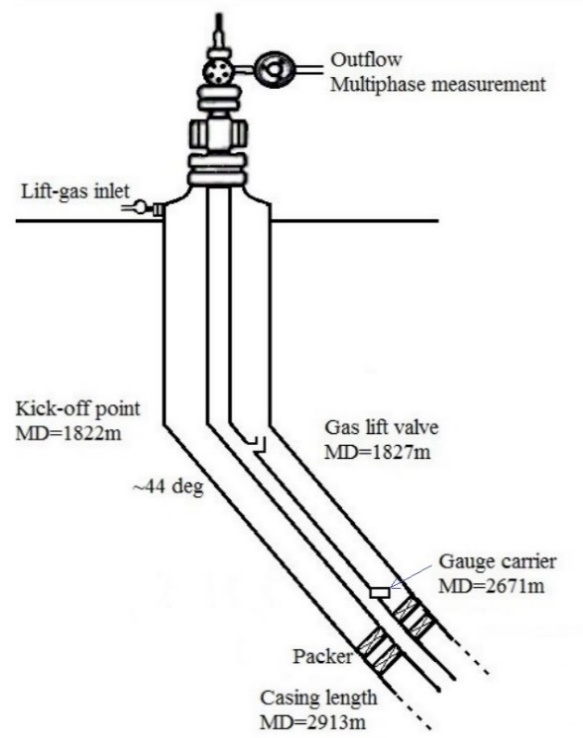
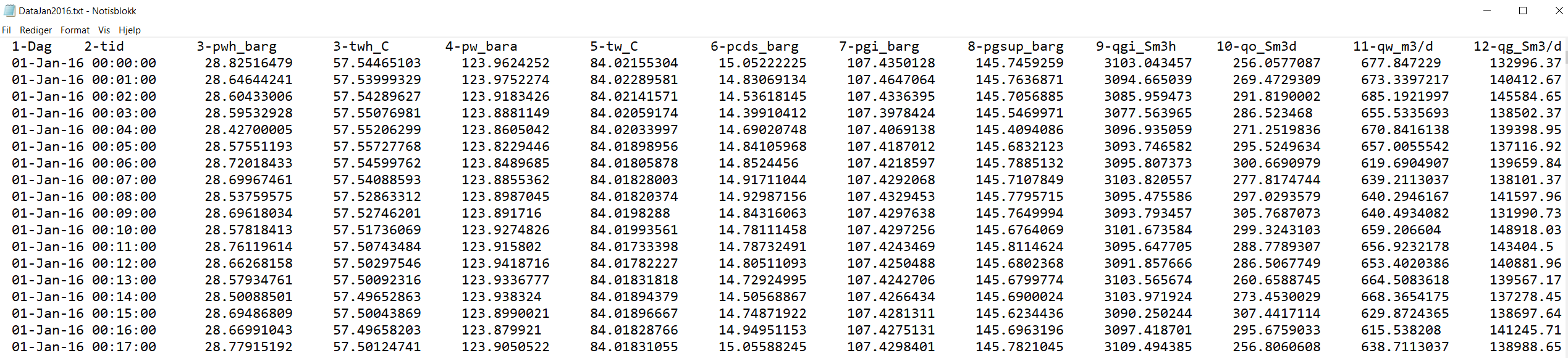
10 Decoding measurements

**10.1 Case**

A well illustrated below. Lift gas is injected into the tubing through a valve at MD 1827m. The gas lightens the flowing mixture, thereby reducing downhole pressure increasing inflow from the reservoir. Pressures and temperatures are logged at the tubing head and down hole, at 2671m measured depth. Injection pressure and rate are measured at the well head. Oil-, gas- and water production is measured at the outlet, by multiphase meters; those are generally considered less accurate than single phase meters.

**  
Figure 1: Gas lifted well , Sara Haugen 2017 (master thesis NTNU)**

The table below shows measurements (Courtesy Equinor), first 17 minutes of January 2016.



Data for the first day is given by: [www.ipt.ntnu.no/~asheim/TPG4135/DataJan2016.txt](http://www.ipt.ntnu.no/~asheim/TPG4135/DataJan2016.txt) . The ML- command: load DataJan2016.txt will enter the data file. Tubing head pressures are found in column 3; The ML function: pth=data(n1:n2,3)extracts measurements no.: n1 to n2 .

**Script: Data transfer and plotting**

disp('Plot: tubing head pressure')

clear

clf

delt=1; % time between measurements (minutes)

load DataJan2016.txt

data=DataJan2016;

ant=size(data); ntot=ant(1); % antall målepunkter

n1=1; n2=100;

pth=data(n1:n2,3);

% Tids-intervall

t1=delt\*n1/60;

t2=delt\*n2/60;

tp=delt\*n1:delt:delt\*n2; % time logged

ntp=length(tp);

stdtp(1:ntp)=mean(pth)+std(pth);

stdtm(1:ntp)=mean(pth)-std(pth);

meant(1:ntp)=mean(pth);

disp([' Midlet trykk: ',num2str(mean(pth)),'(Sm^3/d) Standardavvik: ',num2str(std(pth)) ])

subplot(2,1,1)

hold on

plot(tp/60/24,pth,'r.')

plot(tp/60/24,meant,'k-.')

plot(tp/60/24,stdtp,'b-.')

plot(tp/60/24,stdtm,'b-.')

hold off

legend('Measurements','Mean value','Standard deviation')

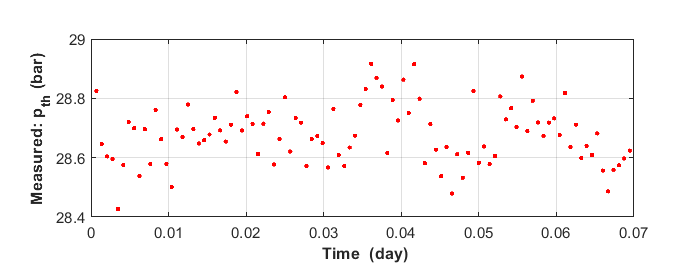
grid

xlabel('\bf Time (day)')

ylabel('\bf p\_{th} (bar)')

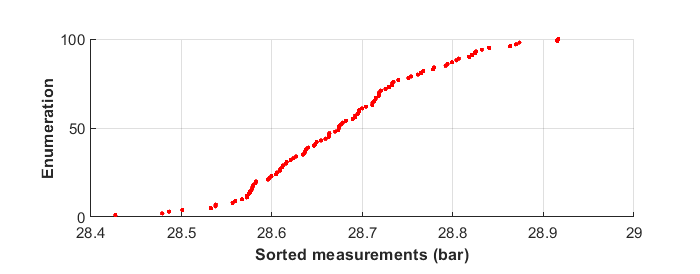
**10.2 Non-modelled variation**

Figure 3 shows the first 100 measurements of tubing head pressure, logged minute. If the well produces under controlled conditions, constant tubing head pressure will be predicted. However, the measurements show variations: 28.4 – 28.95bar . This indicates non-modelled dynamics, possibly caused by slugging. The well produces a mixture of oil, gas and water.

****

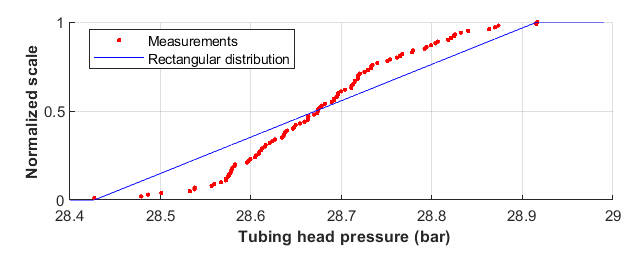
**Figure 3: Measured tubing head pressure variations**

Figure 4 shows sorted measurements (ML: sort(pth)), picturing a distinct distribution. If the vertical axis is divided by the number of measurements (100), it will scale from 0 to 1 and picture the relative distribution, implicitly assuming that new measurements will follow the same trend

****

**Figure 4: Sorted measurement**

The lowest pressure measured is bar and highest . Assuming rectangular probability density: , leads to linear cumulative distribution  illustrated below. This being a rough approximation.



**Figure 5: Measurements and distribution assuming even density of observations**

Regardless of distribution, the arithmetic mean of the measurements: provides an average, whereas standard deviation: measures the spread.

**** (1)

**** (2)

For the measurements above, the mean value is:**** and the standard deviation: **.**

The normal distribution.

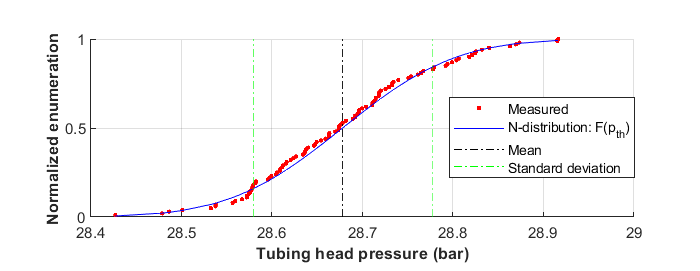
The normal distribution is a function of mean and standard deviation. Density and cumulative distribution are expressed

****  (3)

**** (4)

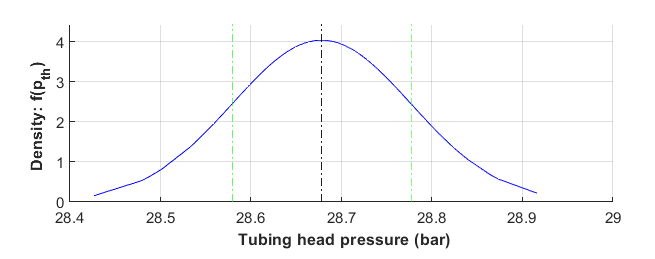
Figure 6 shows compares measurements and cumulative normal distribution calculated by ML: normcdf, using our estimates:****and: ****.The corrrepondence seems acceptable.

If boundary conditions such as reservoir pressure and outlet pressure remain constant, we may expect variation to be normally distributed with mean and standard deviation as before. Thus, changes signal changes in non-modelled dynamics.



**Figure 6 Measurements versus normal distribution**

Figure 7 illustrates the density distribution calculated by: normpdf(pth,28.7,0.10)

****

**Figure 7: Corresponding density distribution**

An implicit assumption of the analyses above is that each measured pressure is independent of the previous ones. This requires the time characteristic of the non-modelled processes causes the variation is much shorter than the logging interval: 1 minute.

**Script**

disp(' Normal distribution')

clear

clf

delt=1.;

load DataJan2016.txt

data=DataJan2016;

ant=size(data);

disp(['No. of data points in file: ',num2str(ant(1))])

pth=data(n1:n2,3);

% Tidsintervall

n1=1;

n2=100;

tp=delt\*n1:delt:delt\*n2;

ntp=length(tp);

disp(['No. of data points ',num2str(ntp)])

%

meanpth=mean(pth);

stdpth=std(pth);

% ----------------------- Cummulative density -----------------------

% empirisk

pths=sort(pth);

Femp=1:1:ntp;

Femp=Femp/ntp;

% normalfordeling

Fn = normcdf(pths,meanpth,stdpth);

subplot(2,1,1)

hold on

plot(pths,Femp,'r.')

plot(pths,Fn,'b')

plot([meanpth,meanpth],[0,1],'k-.')

plot([meanpth+stdpth,meanpth+stdpth],[0,1],'g-.')

plot([meanpth-stdpth,meanpth-stdpth],[0,1],'g-.')

hold off

legend('Measurements','N-distribution: F(p\_{th})','Mean','Standard deviation')

xlabel('\bfTubing head pressure (bar)')

ylabel('\bfCumulative')

grid

%

% density distribuion

fn=normpdf(pths,meanpth,stdpth);

subplot(2,1,2)

maxf=max(fn)\*1.1;

hold on

plot(pths,fn,'b')

plot([meanpth,meanpth],[0,maxf],'k-.')

plot([meanpth+stdpth,meanpth+stdpth],[0,maxf],'g-.')

plot([meanpth-stdpth,meanpth-stdpth],[0,maxf],'g-.')

hold off

grid

xlabel('\bf Tubing head pressure (bar)')

ylabel('\bfDensity: f(p\_{th}) ')

**10.3 Numerical filtering**

Average rates and pressure often requested. If the process is stable, the mean serves this purpose**.** However, simple averaging may misleading and numerical filter preferable if the process drifts.

**10.3.1 Moving average**

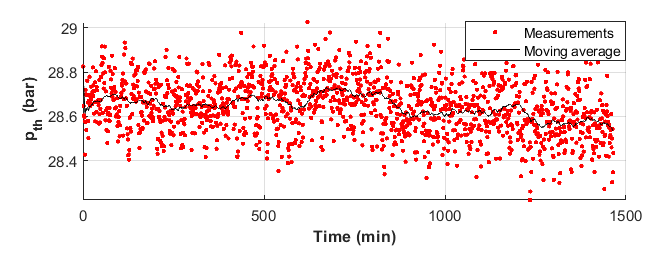
Obviously, averaging over a moving interval smooth the curve. Averaging over: *n* data points estimates the smoothed value

 (5a)

Next measurement:  provides a new smoothed value:. This can be calculated from (6a), but more efficiently by recursive updating of the previous estimate

 (5b)

Longer interval length causes stronger smoothing, and loss of information that may lie in shorter term variation. Figure 8 shows smoothing over interval length: 60 minutes (, with  being the time between logged measurements).



**Figure 8 Smoothing by moving average**

**10.3.2 Exponential filter**

This filter weights each new measurements: against the established estimate: , providing new estimate in recursive form

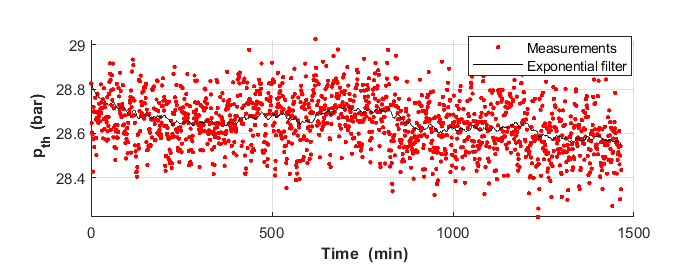
 (6a)

The established estimate depends on previous values: . Using this, expresses the filter

 (6b)

The weighing factor:  decides how each measurement influences the established estimate. The filter relation (5) implies that the influence of previous measurements decreases exponentially with time since they were made.

Figure 8 shows results weighing each new measurement 2%, 🡪 =0.02, chosen to achieve visual similarity to the moving average showed in figure 8 above



**Figure 9 Exponential filtering**

**Script**

% Numerical filters

clf

clear

delt=1.;

load DataJan2016.txt

data=DataJan2016;

ant=size(data);

disp(['No. of data points in file: ',num2str(ant(1))])

% Tidsinterval

n1=1;

n2=ant(1);

pth=data(n1:n2,3);

tp=delt\*n1:delt:delt\*n2;

n=length(tp);

disp(['No. of data points used: ',num2str(n)])

%

disp('---------------Exponential filter-------------------------------')

alfa=0.02;

disp([' Exponential weighing cofficient: ',num2str(alfa)])

pthe(1)=pth(1);

for i=2:n

pthe(i)=alfa\*pth(i)+(1-alfa)\*pthe(i-1);

end

subplot(2,1,1)

hold on

plot(tp,pth,'r.')

plot(tp,pthe,'k-')

hold off

grid

legend('Measurements','Exponential filter')

xlabel('\bf Time (min)')

ylabel('\bf p\_{th} (bar)')

disp('----------------Moving average-----------------')

k=60;

disp(['Averaged over: ',num2str(k),' points --> ',num2str(k\*delt),' minutes' ])

%

sum=0;

for i=1:k

sum=sum+pth(i);

pthma(i)=sum/i;

end

for i=k+1:n

sum=sum+pth(i)-pth(i-k);

pthma(i)=sum/k;

end

subplot(2,1,2)

hold on

plot(tp,pth,'r.')

plot(tp,pthma,'k-')

hold off

grid

legend('Measurements','Moving average')

xlabel('\bfTime (min)')

ylabel('\bf p\_{th} (bar)')

**10.4 Process dynamics**

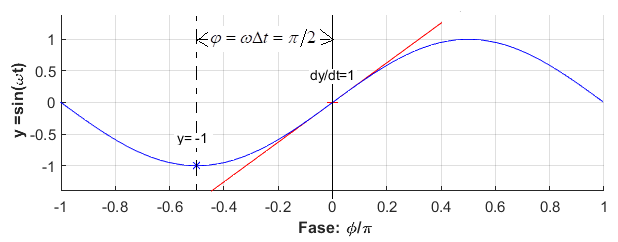
**10.4.1 Delayed response**

For many processes the change of “level” is proportional to current level: . This provides the solution: . Negative proportionality factor: <0, makes level decrease with time. Such processes are considered dynamically stable. Conversely, >0 makes level increase with time. Such processes are dynamically unstable and may blow up.

But, the response is often delayed. For example: High birth-rate will lead to increased population (huntable game, workforce, IS-warriors, or whatever). This may be quantified as

 (7)

According to eq. (7), the current gradient: dy/dt corresponds to the level at earlier t. Figure 10 shows that for a sine-function, the gradient is:  at: t=0, and:  at  earlier. With proper, negative, choice of the proportionality factor: **, eq. (7) may be fulfilled. Thus, delayed response may cause sinusoidal waves.



**Figure 10: Sinusoidal variation**

From:  follows oscillation period 4 times the delay :

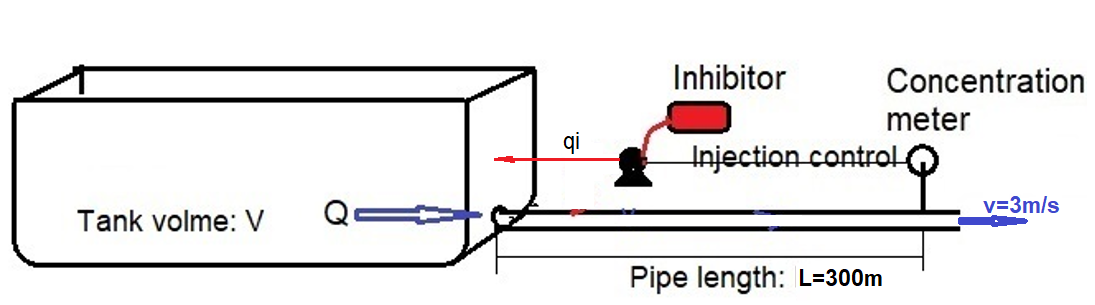
It can be shown the “proper choice of proportionality factor” is: . Lower, or higher values will lead to increasing, or decreasing oscillations.

**10.4.2 Biological clock**

The Nobel Prize in Physiology or Medicine for 2017 was awarded: Jeffrey C. Hall, Michael Rosbash and Michael W. Young **for their discoveries of “Molecular mechanisms controlling the circadian rhythm”. In more mundane language: to discover that reaction and diffusion through cell membranes makes a biological oscillator with period:** . This provides **the sense of time, common to most living organisms.**

**10.4.3 Process regulation**

The figure illustrates concentration control by supplemental injection



**Figure 11 Concentration control**

Assuming perfect mixing in the tank, concentration changes proportional to the injection rate

 (8)

Considering Injection rate regulation proportional to deviation from the goal: 

 (9)

Combining (8) and (9), concentration change:, is expressed equivalently to (8) above

 (10)

Prediction of dynamic respons

Regulation coefficient: =-0.01

Pipe length L=300 m

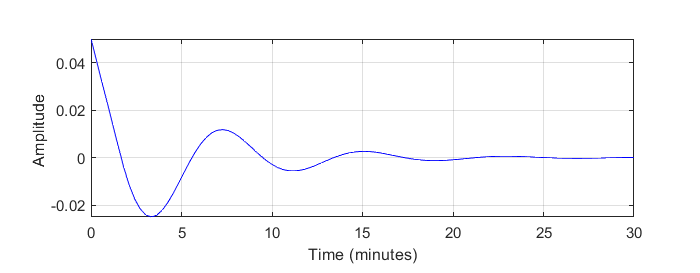
The design rate makes pipe flow velocity v=3 m/s.

1. Predict dynamic response to 5% concentration change.
2. Predict dynamic response to 5% concentration change, if the flow is reduced 50%

Simulation solution

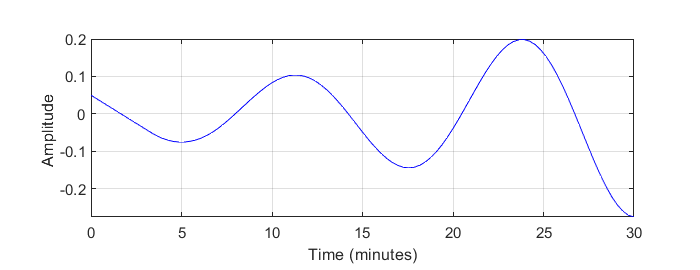
The program: ML handles delayed differential by: dde23(DDEFUN,LAGS,HISTORY,TSPAN).

1. At design flow condition



**Figure 12 Dynamic development for pipe flow velocity: v=3m/s**

1. Prediction for reduced flow rate



**Figure 13 Dynamic development for pipe flow velocity: v=1.5 m/s**

**Script**

disp(' Delayed response: dy/dt=b\*y(t-delT) ')

clear

clf

v=1.5;

L=300;

Vol=1;

beta=-0.01;

betalim=-v\*Vol/L\*pi/2;

delT=L/v ; %tidsforsinkelse

b=beta/Vol;

T=30\*60;

disp([ ' b= ',num2str(b),' \Delta t= ',num2str(delT) ])

disp([ ' bDt= ',num2str(b\*delT),' T= ',num2str(T),' s' ])

%

ts=linspace(0,T);

tmax=ts(length(ts));

yi=0.05; % disturbance

% ------------------ matLab dde23 -----------------

ddefun = @(t,y,ydelay) b\*ydelay;

History = @(t) [yi]; %

sol=dde23(ddefun,delT,History,[0,tmax]);

yint = deval(sol,ts);

ysol=yint(1,:);

% ---------------------------------------------------

subplot (2,1,1)

plot(ts/60,ysol,'b') % matlab

grid

xlabel('Time (minutes)')

ylabel('Amplitude ')

**10.5 Sinusoidal waves**

**10.5.1 Sinusoidal oscillation**

Sinusoidal waves progresses in time and space, described as

 (11)

Where:

angular frequency:

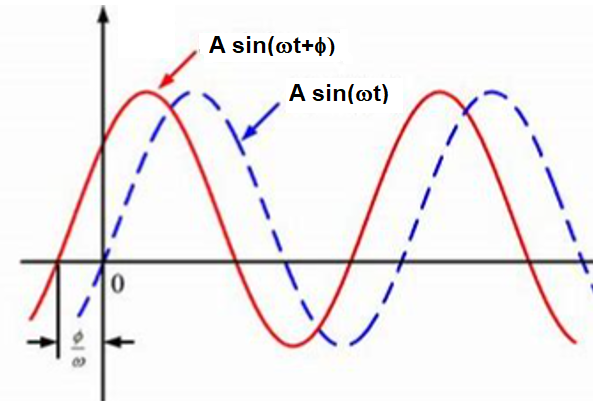
wave number: 

phase shift: 

At fixed location: x=0, the wave will produce oscillations: 

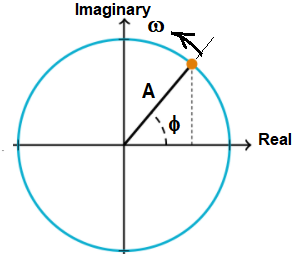
angular frequency:

phase shift: 

****

**Figure 14 Sinusoidal waves**

By complex numbers



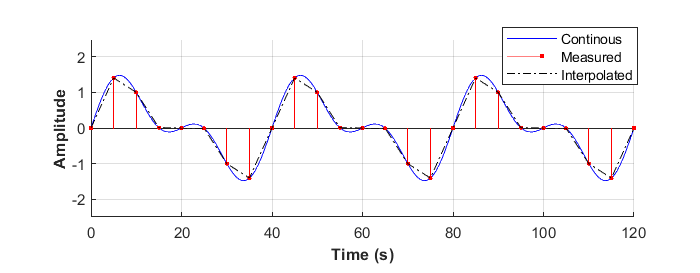
****

**10.5.2 Sampled data**

For the case we consider 2 sinusoidal waves, with amplitudes: A1=0.7, A2=1.0 and periods: T1=20s , T2=40s. Total amplitude becomes the sum of individual amplitudes

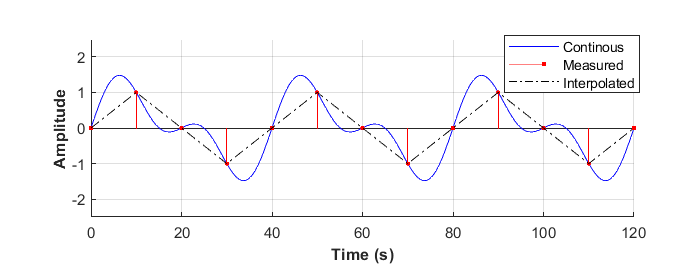
 (12)

Figure 15 illustrates actual amplitude and measurements. Linear interpolation between measurements approximates the actual oscillation

****

**Figure 15 : Samling interval: ts=5s**

Figure 16 shows measurements made each 10 second. Linear interpolation between measurements roughly approximates the slower oscillation with period: 40, but completely overlooks the faster with period 20

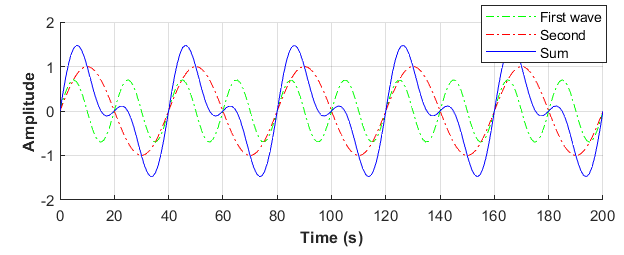
****

**Figure 16 : Samling interval: ts=10s**

Nyquist’s sampling theorem says that oscillation of shorter period than twice the sampling will not be registered. Thus, sampling interval 10 s will not register oscillations periods less than 20 seconds. This strictly relates to finite Fourier analyses, but also makes general sense.

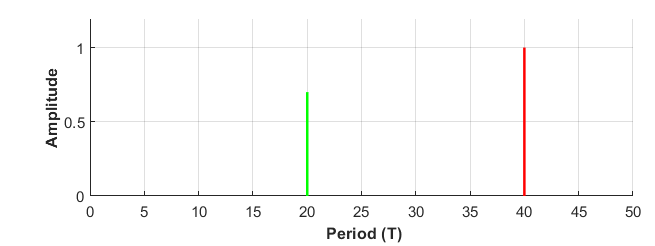
**10.5.3 Frequency spectrum**

Oscillations may be clearer expressed in the frequency domain, or period domain: T=1/f. Figure 17 shows the oscillations considered above.

****

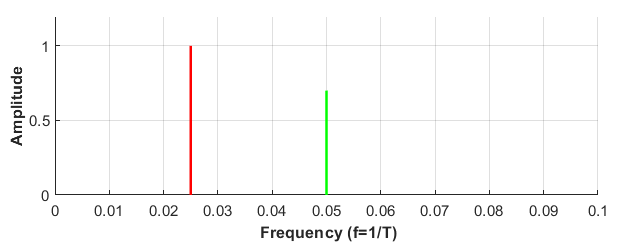
**Figure 17: Waveform**

This is made up of two waves with different period (frequency) and wavelengths and is equivalently expressed by the spectrum below



**Figure 18 Spectrum of the waveform considered**

Figure 19 shows the equivalent frequency spectrum

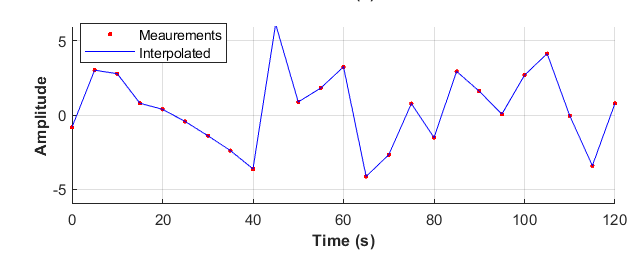
****

**Figure 19: Frequency spectrum**

Knowing frequencies and amplitudes, the spectrum is easily drawn. By the Fourier transform, the spectrum may be calculated for oscillation-, or pulse functions. The Fast Fourier Transform (FFT) calculates the spectrum of sampled oscillations.

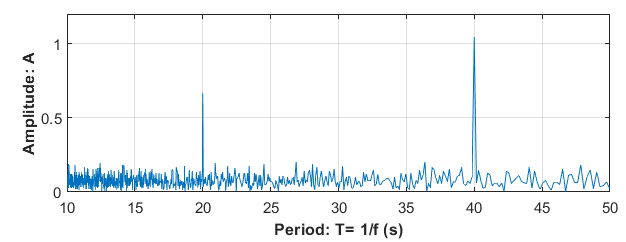
**10.5.4 Fast Fourier transform**

The Fast Fourier Transform: FFT ( or: “Finite Fourier Transform”) is an algorithm that operates on sampled data and calculates the frequency spectrum. Consider waveform of figure 15, sampled at t= 5s, corrupted by random noise. Figure 20 shows one realization: While interpolation between measurements in fig 15 approximated the actual waveform, such interpolation shown below will be grossly misleading

****

**Figure 20 : Signal corrupted by random noise (=2)**

Figure 21 shows the spectrum calculated by FFT. This compares quite well to the uncorrupted spectrum shown in figure 18 above. Thus, by FFT, the underlying waveform may be extracted from noisy measured data.

****

**Figure 21: Spectrum, estimated from 2000 data points**

**Script**

disp('Finite Fourier Transform (fft) of sinusoidal wave')

clear

clf

delT = 5; % logging interval(s)

Fs = 1/delT; % logging frequency

nt = 2000; % number of measurements

fny=0.5\*Fs;

t = (0:nt-1)\*delT; % Time span

% amplitudes

A1=0.7;A2=1;

%periods

T1=20;T2=40;

% waveform

y = A1\*sin(2\*pi/T1\*t) + A2\*sin(2\*pi/T2\*t)+2\*randn(size(t));

% plot logged data

subplot(2,1,1)

plot(t(1:120/5),y(1:120/5),'r.')

grid

xlabel('\bfTime (s)')

ylabel('\bfMeasurements')

%---------------- Finite Fourier transform--------------

Y = fft(y);

P2 = abs(Y/nt);

ampl = P2(1:nt/2+1);

ampl(2:end-1) = 2\*ampl(2:end-1);

f = Fs\*(0:(nt/2))/nt;

% Plot period spectrun

Tny=2\*delT; % shortest noticable period

subplot(2,1,2)

Tf=f.^-1;

plot(Tf,ampl)

axis([Tny,50,0,1.2])

xlabel('\bfPeriod: T= 1/f (s)')

ylabel('\bfAmplitude: A')

grid