TNG015 Signaler och System

VT 2024

Laboration 3: Bildfiltrering och adaptiva filter

**Namn: William Gabriel**

**LiU ID: wilga619**

**Epost: wilga619@student.liu.se**

**Program: Medieteknik**

**Namn: Samuel Hellqvist**

**LiU ID: samhe463**

**Epost: samhe463@student.liu.se**

**Program: Medieteknik**

## Inledande ord om laborationen

Den här laborationen tar dels upp lite om bildbehandlande åtgärder som du kan göra med kunskaper om signaler och system, men också några övningar med adaptiva filter eller det närbesläktade begreppet artificiella neurala nätverk (ANN).

Redovisningen av resultat görs på anvisade platser i laborationshandledningen. Platser där grafer ska redovisas är markerade, men man kan även inkludera grafer på andra ställen vid behov.

## Förberedelseuppgifter

Läs avsnitten i kurskompendiet som handlar om adaptiva filter. Det är en fördel om du hunnit titta på något övningsexempel i kompendiet som handlar om adaptiva filter.

## Uppgifter som ingår i laborationen

Obligatoriska uppgifter: Avsnitt I uppgifterna 1 – 4

Välj mellan avsnitt II uppgifterna 5 – 8 eller avsnitt III uppgifterna 9 – 12

## Labbrapport inlämningsinformation

Lämna in labrapport som **pdf** fil på Lisam Submission websidan. Inkludera dina teoretiska lösningar och simuleringsresultat under varje uppgift i detta dokument och spara som pdf fil. Ni får antigen att ta foton av teoretiska lösningar på paper och inkludera dem i Word dokumentet eller skriva ekvationerna direkt med Equation Editor i Word eller använda Latex.

## Filtrering av Bilder

Som du säkert förstår är det inte bara ljudsignaler eller andra ”endimensionella” signaler som kan filtreras. Även en bild, som är tvådimensionell (höjd och bredd), kan filtreras. När man pratar om bildfiltrering så är begreppet frekvens lite annorlunda än begreppet ljudfrekvens. En ljudsignal med hög frekvens motsvarar en hög ton som du kan lyssna på.

För bilder gäller att en bild kan ha olika *spatiala* frekvenser. Enheten för spatiala frekvenser är **perioder/meter** (perioder per meter) och är ett mått på hur ofta en sinusoidal (sinusformat mönster) upprepas i bilden. Det sinusformade mönstret tas fram ur Fouriertransformen för bilden. Man kan också speciellt i bildsammanhang använda enheten **perioder/mm**, men också **linjepar/mm**.

En bild med låg spatial frekvens har få linjepar per distansenhet, en bild med hög spatial frekvens har många linjepar per distansenhet.

A black and white gradient

Description automatically generatedTitta på följande två bilder (figur 1). Den övre bilden har låg spatial frekvens (horisontellt) medan den nedre bilden har relativt hög spatial frekvens (horisontellt). Vertikalt har båda bilderna samma spatiala frekvens, dvs så vitt vi kan se är den noll.

1. A black and white striped background

   Description automatically generated**Low spatial frequency**
2. **High spatial frequency**

**Figur 1: Spatiala (rumsliga) frekvensen i horisontell riktning**

När du filtrerar ljudsignaler har du lärt dig att det kan man se som en faltning mellan insignalens sampel x[n] och filterkoefficienterna (impulssvarets sampelvärden) h[n].

Formelmässigt ser det ut som:

När det gäller bilder så sker filtrering på liknande sätt. Skillnaden är att ”impulssvaret” för bildfiltersystemet nu är tvådimensionellt och att insignalen, dvs bilden, är tvådimensionell , där varje bildpunkt (pixel) har ett värde. Bilden byggs upp av rader (n-värden) och kolumner (m-värden). Om bilden har 256 rader och 512 kolumner så gäller att och att .

Filterkoefficienterna för är vanligtvis betydligt färre än antalet bildpunkter.

För svart/vita (gråskalebilder) är det vanligt att bildpunkterna har värden mellan 0-255 när de representeras i en dator (digitala bilder). T.ex. kan en bildpunkt på 23:e raden och 59:e kolumnen vara helt svart, vilket betyder att . Om bildpunkten på 45:e raden och 7:e kolumnen är helt vit, betyder det att . Representationen för svart och vitt kan också vara tvärtom (0=svart, 255=vitt). Bildpunktsvärdena mellan 0 och 255 representerar olika nyanser av grått. På så vis kan rimligt många nyanser mellan helt vitt och helt svart, representeras med 8 databitar (.

För färgbilder använder man en lite annan representation eftersom färgbilder (enligt en metod) byggs upp av tre grundfärger (rött, grönt och blått, RGB-färger). En specifik färg fås då genom att mixa ihop tre olika matriser, en röd-matris, en grön-matris och en blå-matris.

Faltning av gråskalebilder skulle formelmässigt se ut som:

(tvådimensionell faltning)

Tvådimensionell faltning utförs i MATLAB med funktionen *conv2( ).*

Nu är inte den här laborationen en laboration i bildbehandling i första hand utan tanken är bara att du ska få en liten inblick i att signalbehandlingsverktygen kan användas på områden som har med bilder att göra. Du kommer säkert att ha möjligheter att läsa mer om det här i en särskild kurs för bildbehandling.

Du ska nu försöka filtrera två enkla bilder (bilderna i figur 1) med hjälp av ett lågpassfilter för bilder. Om du kommer ihåg ett av de första LP-filtren för ljud som presenterades i kursen så var det ett enkelt medelvärdesbildande filter med differensekvationen

Medelvärdet här är summan av fem stycken x-sampel, dividerat med antalet. Impulssvaret för LP-filtret är alltså koefficienterna framför varje x-sampelterm, i det här fallet .

När det gäller bilder så kan vi skapa ett LP-filter för bilder på motsvarande sätt. Skillnaden ligger i att ”bildimpulssvaret” bör vara 2-dimensionellt (men det är inte nödvändigt).

1. Spatialfrekvens i bilder
   1. Du ska nu skapa och rita bilderna från figur 1 med följande kod:

colormap('gray');

for r=1:512

for c=1:512

B1(r,c)=uint8(128-127\*sin(2\*pi\*3\*c/512));

B2(r,c)=uint8(128-127\*sin(2\*pi\*45\*c/512));

end

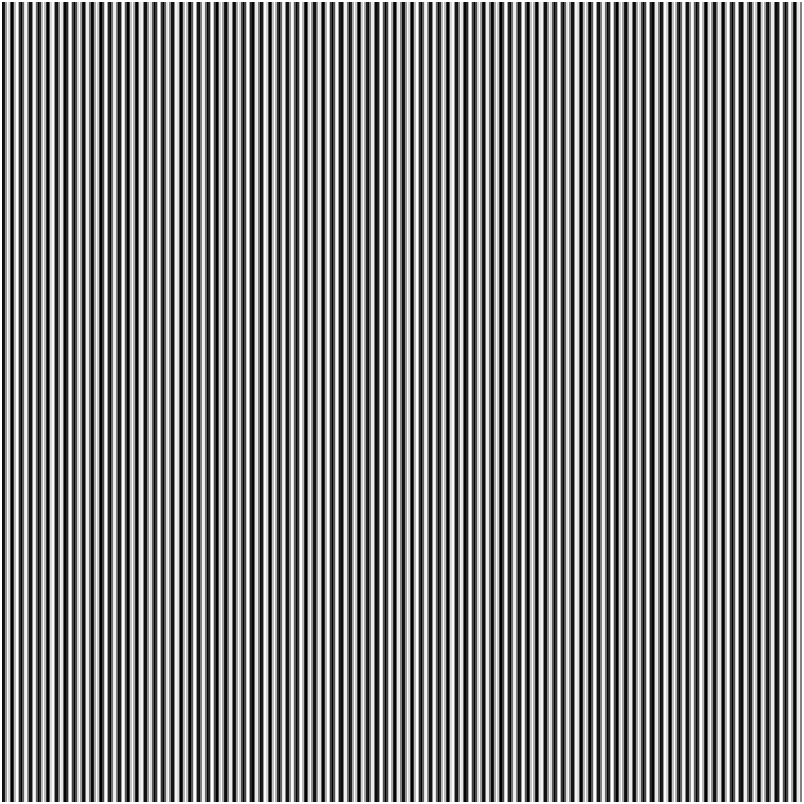
end

figure(1); imshow(B1); figure(2); imshow(B2);

* 1. Förklara hur formeln för skapandet av bildmatriserna B1 och B2 fungerar. Funktionen uint( ) i programkoden, skapar ett 8-bitars heltal.

Ur formeln för bildmatriserna får man ett heltal mellan 1-255 på rad r, kolonn c som representerar en nyans av grå där 255 är vitt och 1 är svart. Sinus-termens värde skiftar beroende på vilket index av kolonn som funktionen är på och därför skiftar färgen i vertikala rader.

* 1. Skapa ytterligare en bildmatris B3(r,c) med samma storlek men med dubbelt så hög spatiell frekvens som B2. Rita ut denna i figure(3).



1. Lågpass och högpass filtrering av bild.
   1. Bilderna B1, B2 och B3 ska nu filtreras med både ett LP-filter och ett HP-filter. Impulssvaret (filtret) för LP-filtrering skapas med matrisen:

h\_lp = (1/100)\*ones(10,10);

Impulssvaret för HP-filtrering skapas med matrisen:

h\_hp = [1 0 -1; 2 0 -2; 1 0 -1]; %observera hak-parenteser här!

Filtrera alla bilderna genom att utföra en 2-D faltning, med nedanstående MATLAB kod (funktionen *single( )* omvandlar ett flyttal till enkel precision, tar mindre plats i datorn). Ersätt matriselementet ”**h**” med respektive **h\_lp** och **h\_hp** samt ange de filtrerade bildmatriserna på motsvarande sätt.

B1\_filt = uint8(conv2(single(h),single(B1)));

B2\_filt = uint8(conv2(single(h),single(B2)));

B3\_filt = uint8(conv2(single(h),single(B3)));

%% 2

h\_lp = (1/100)\*ones(10,10); %impulsvaret för LP filtrering

h\_hp = [1 0 -1; 2 0 -2; 1 0 -1]; %impulssvaret för HP filtrering

B1\_filt\_lp = uint8(conv2(single(h\_lp),single(B1)));

B2\_filt\_lp = uint8(conv2(single(h\_lp),single(B2)));

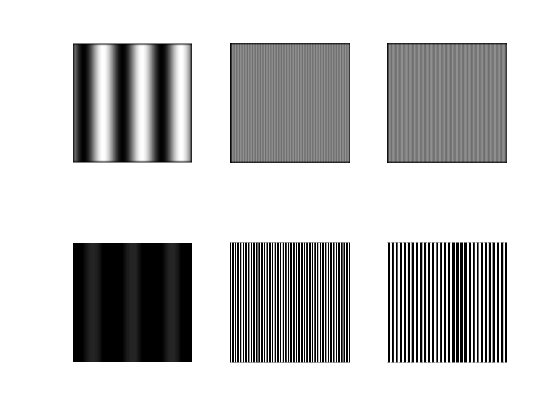
B3\_filt\_lp = uint8(conv2(single(h\_lp),single(B3)));

B1\_filt\_hp = uint8(conv2(single(h\_hp),single(B1)));

B2\_filt\_hp = uint8(conv2(single(h\_hp),single(B2)));

B3\_filt\_hp = uint8(conv2(single(h\_hp),single(B3)));

* 1. Rita alla filtrerade bilder (totalt 6 stycken, dvs LP/HP för tre olika bilder).



Första raden: 1–3 low pass, andra raden: 1–3 high pass

1. Frekvensanalys av bilder
   1. Frekvensanalys av bilder kan göras på motsvarande sätt som frekvensanalys av ljud. Eftersom bilder är tvådimensionella använder du kodraden:

B\_frek = fft2(B);

där **B** = aktuell bildmatris och **B\_frek** = frekvensmatris för aktuell bildmatris.

För att rita ut ett 2-d frekvensspektrum ska du använda kodraden:

mesh(abs(B\_frek(1:256, 1:256)));

där absolutbeloppet ger magnituden av de komplexa talen i frekvensmatrisen. Observera att det bara är relevant att titta på hälften av varje dimension i frekvensmatrisen, därav konstruktionen 1:256 för just denna bildmatris.

Skapa frekvensspektrum för bilden **B2** och rotera figuren i MATLAB:s graffönster (via ”Rotate 3D”-knappen) så att du ser så mycket som möjligt av frekvensspektrum.

* 1. Skapa även ett frekvensspektrum för den LP-filtrerade **B2** och jämför med ovanstående spektrum, som är för den ofiltrerade **B2**.

B\_frek = fft2(B2);

figure(1)

mesh(abs(B\_frek(1:256, 1:256)));

figure(2)

B2\_filt\_lp\_frek = fft2(B2\_filt\_lp);

mesh(abs(B2\_filt\_lp\_frek(1:256, 1:256)));

1. Det är möjligt att på ett enkelt sätt framhäva vissa strukturer i en bild och undertrycka andra. Bilden (figur 2) visar en fågel som sitter på en gren. Bildens mest framträdande struktur horisontellt kan sägas vara den gren som fågeln sitter på. Men det finns också tydliga vertikala strukturer i bilden som utgörs av kortare grenar, som pekar uppåt i bilden. Bilden finns på kurshemsidan ”Bird\_lab3”
   1. Läs in bilden och visa den på skärmen:

Bird = imread('Bird\_lab3.jpg');

imshow(Bird);



**Figur 2: Bild med tydliga horisontella och vertikala strukturer. (Svart-vit i kompendiet, färg på bildskärmen.)**

* 1. Du ska nu utföra två filtreringar av fågelbilden. Vid den ena filtreringen ska horisontella strukturer synas tydligt. Vid den andra filtreringen ska vertikala strukturer synas tydligt.

De filterkärnor (impulssvar) som behövs för filtreringen ska se ut som **h\_hp** matrisen i uppgift 2. Genom att falta med **h\_hp** gör du en högpassfiltrering i vertikal led. För att filtrera i horisontell led måste du skapa transponatet till **h\_hp**.

Innan filtreringen bör du förbereda fågelbilden genom att göra den till en svartvit bild. I MATLAB görs det med:

Bird\_black = rgb2gray(Bird);

imshow(Bird\_black);

Redovisa dina filtreringar genom att klistra in de två filtrerade fågelbilderna och kommentera hur du gjort filtreringen.



Vertikal strukturer



Horisontal strukturer

MATLAB koden:

%% 4

Bird = imread('Bird\_lab3.jpg');

%imshow(Bird);

h\_hp = [1 0 -1; 2 0 -2; 1 0 -1]; %impulssvaret för HP filtrering

h\_hp\_trans = transpose(h\_hp);

Bird\_black = rgb2gray(Bird);

figure(1)

imshow(Bird\_black)

Bird\_black\_filt\_hp\_vertical = uint8(conv2(single(h\_hp),single(Bird\_black)));

Bird\_black\_filt\_hp\_horizontal = uint8(conv2(single(h\_hp\_trans),single(Bird\_black)));

figure(2);

imshow(Bird\_black\_filt\_hp\_vertical);

figure(3);

imshow(Bird\_black\_filt\_hp\_horizontal);

## Adaptiva filter för att eliminera störningar (välj detta avsnitt eller III)

Studera det avsnitt i kurskompendiet som handlar om att eliminera störningar ur en signal. Du bör också förstå figur 3 som är en kopia från kurskompendiet.







**+**





**-**

ADAPTIVT SYSTEM



**Figur 3: Adaptivt system för att eliminera störningar.**

Det väsentliga här är att det adaptiva systemet ”tränas” med en störsignal som liknar den riktiga störningen. Den har samma egenskaper som den riktiga störningen. Men i allmänhet känner man inte till exakt hur den riktiga störningen ser ut. Visste man det skulle det vara enklare att eliminera den direkt genom att helt enkelt göra en subtraktion:

I det adaptiva systemet söks istället en så bra uppskattning av störningen , som möjligt, för att använda den i en subtraktion:

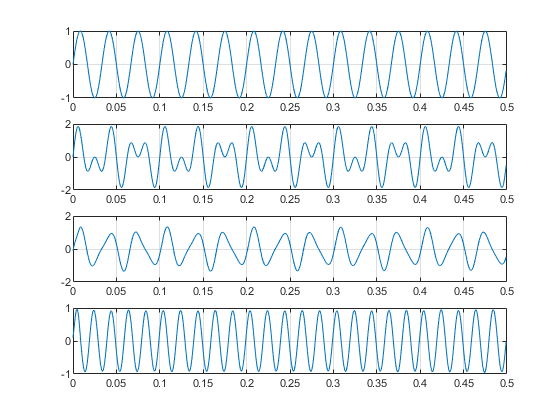
där

I följande laborationsuppgift ska du undersöka hur effektivt ett adaptivt system som bygger på en FIR-filterstruktur, kan bli på att eliminera störningar.

1. Adaptivt system för att eliminera störningar
   1. Studera programkoden i exempel 13.5 i kurskompendiet samt figur 3 (i denna handledning). Kopiera koden (finns på S: mappen) och klistra in den i en MATLAB script. Variera antal epoker när du kör koden. Du kan t.ex. börja med att bara köra en epok och titta på resultatet. Öka därefter antal epoker och jämför ifall resultatet blir bättre.

Observera att det kan ta relativt lång tid att exekvera koden, beroende på datorkapacitet.

* 1. Rita grafer på insignal, insignal+ störning samt utsignal och kommentera resultatet.

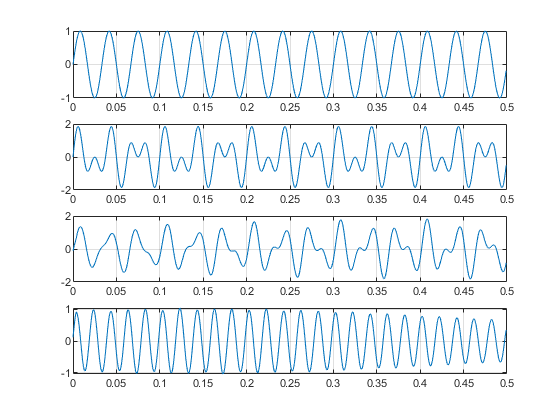


Error-grafen (3dje uppifrån) ska vara så lik den första grafen, som visar insignalen x utan störning, som möjligt. Desto mer likt felet är insignalen desto bättre filter har man.

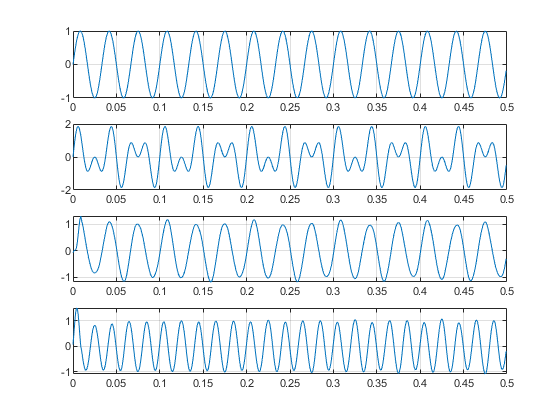
1. Förändra förutsättningarna i den adaptiva inlärningen genom att skapa ett mindre nätverk. Ett mindre nätverk kan skapas genom att förändra följande två kodrader till ett nätverk med 5 fördröjningselement:

**id = [0 1 2 3 4 5];** och raden **Pi = {0 0 0 0 0};**

Spara tidigare utdata på lämpligt sätt så att du kan jämföra den här inlärningen med tidigare inlärning. Provkör och rita resultat som tidigare dvs rita grafer på insignal, insignal+ störning samt utsignal.



1. Learining rate i adaptivt system
   1. En annan parameter som kan vara avgörande för slutresultatet är den s.k.”learning rate”-parametern (inlärningskoefficienten). Börja med att återställa antal inlärningsepoker till 30 stycken.
   2. Öka därefter ”learning rate” värdet stegvis från ursprungliga 0.01 till max. 0.30. Kör programmet med de nya förutsättningarna. Redovisa den bästa körningen.

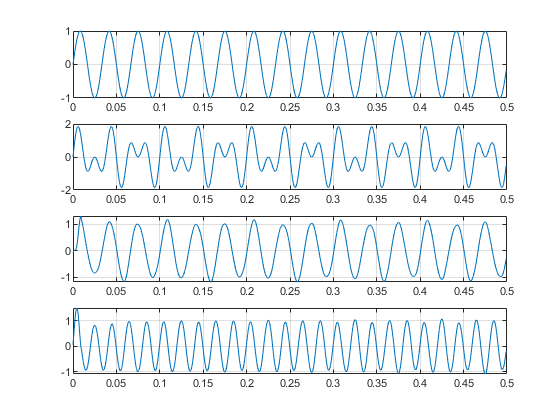


Genom att gå stegvis mellan 0.01 till 0.3 fann vi att en learning rate runt 0.04 gav bäst resultat. Bilden visar lr = 0.04.

1. För att kunna jämföra olika körningar (med olika parametrar) är det bra att ha ett kvalitetsmått, någon siffra som säger om körningen är bra eller dålig. Hitta därför på ett kvalitetsmått så att du kan jämföra de olika försöken och se vilken körning som ger bäst resultat. Tips: vad menas med bättre/sämre körning? Kan du jämföra insignal med utsignal på ett bra sätt?

När du har tänkt ut ett bra kvalitetsmått ska du testa programmet med egna värden på antal epoker, inlärningsfaktorn eller annat som du anser kan vara relevant för adaptionen.

Redovisa ditt bästa försök, dvs rita grafer (eller bifoga grafer från MATLAB) och ange vilka värden du använt vid den bästa körningen.



Passes = 30 och learning rate = 0.04 ger bästa körning.

Vår kod för en variabel ”quality” där mindre värde betyder bättre kvalité:

diff = zeros(1, 2001);

for i = 1:2001

diff(1:i) = abs(x\_signal(1:i) - [e{1:i}]);

end

quality = mean(diff);

## **Adaptiva filter för att utföra funktionsapproximering (välj detta avsnitt eller II)**

I de följande övningarna får du prova en nätstruktur som mer liknar det som brukar förknippas med A.N.N. Nätverket du ska använda är ett ”feedforward”-nätverk med olinjära processelement. Inlärningsalgoritmen blir ”backpropagation by error”, eller kortare ”backprop”-algoritmen.

1. Studera exempel 13.6 i kurskompendiet och kopiera in MATLAB-koden från mappen ”Data Files”. Kör programmet och rita ut (eller bifoga grafer för) både insignalen och utsignalen.
2. Undersök hur stabilt nätverket är genom att minska dess storlek. Gör detta genom att ta bort två eller fler processelement i det gömda lagret. Justera nätstorleken via kodraden:

**net = feedforwardnet(10, 'trainlm');**

Siffran 10 anger antal processelement i det gömda lagret.

Provkör programmet med olika storlek och rita ut både insignalen och de olika utsignalerna (eller bifoga grafer från MATLAB). Rita flera utsignaler i samma bild.

1. Återställ nätverket till 10 processelement. Variera därefter antal epoker som nätverket tränas via kodraden:

**net.trainParam.epochs = 20;** (här 20 epoker)

Redovisa några olika resultat på utdata baserat på ett nätverk som tränats olika mycket.

1. Du ska slutligen försöka att skapa ett nätverk som gör en närmast perfekt approximation. Experimentera med några av de parametrar som är möjliga att variera i MATLAB-koden nämligen nätstorlek, antal tränade epoker men också slutliga felet, det som anges av kodraden:

**net.trainParam.goal = 0.01;**

Redovisa Insignal och utsignal från det funktionsapproximerande nätverket (bästa körning)