Laboratorieøvelse i Fy1005- Termisk Fysikk Vår 2010

Institutt for Fysikk- NTNU

# *Maxwells fartsfordelingslov*

*Hensikten med oppgaven er å bli kjent med begrepet fordelingsfunksjoner og hvordan disse kan måles i praksis. En vil bestemme fartsfordelings funksjonen for partikler som utfører en to dimensjonal bevegelse, og sammenlikne denne med teoretisk forventet fordeling (Maxwell fordeling). En tar også sikte på å belyse begreper innen klassisk statistisk mekanikk som; temperatur, faserom og midlere fri veilengde.*

## *Oppgave*

*1: Finn fordelingen av partikler i posisjonsrommet.*

*2: Finn fordelingen av partikler i hastighetstsrommet.*

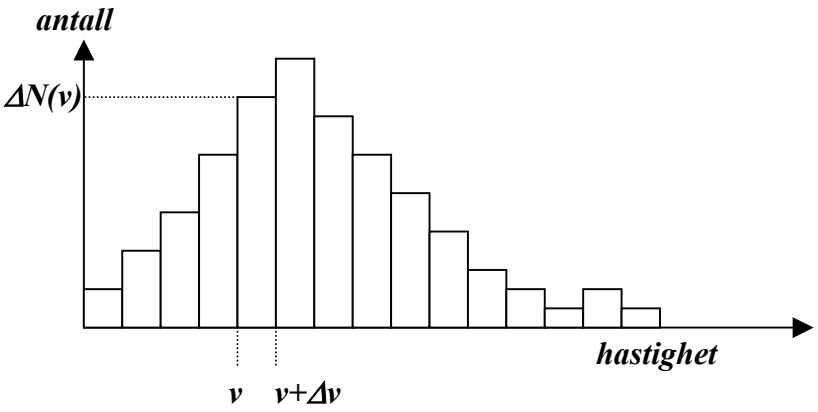
*3: Bestem fordelingen som funksjon av fart.*

*4: Finn midlere kinetisk energi (temperaturen) til partiklene og sammenlikne teoretisk og målt fordeling. Kommenter forskjellen.*

## *Teori*

***Fordelingsfunksjoner***

Vi forestiller oss en samling av ***N*** identiske partikler og at disse har forskjellig hastighet. Når vi skal finne fordelingsfunksjonen til hastighetene, teller vi antallet partikler med hastighet (***v***) i området mellom ***v og v +Δv***, som vi kaller ***ΔN***. I et koordinatsystem lar vi x-aksen være hastighet og vi deler x-aksen opp i like store intervaller; ***Δv***, som kalles boksbredden eller binbredden. Så teller vi opp antallet partikler med hastigheter i området mellom ***v og v +Δv*** og kaller dette ***ΔN(v)*** . I koordinatsystemet avsetter vi ***ΔN(v )*** langs y-aksen for hastigheten ***v*** (se figuren nedenfor).




*Eksperimentell bestemmelse av en hastighets fordelingsfunksjon.*

En målt fordelingsfunksjon skal ofte sammenliknes med en teoretisk fordelingsfunksjon. I teoretisk sammenheng erstattes ofte differenser ***(Δv)*** med differensialer (***dv***), som bare betyr at en lar boksbredden være infinitesimal. Fordelingsfunksjonen er altså:

$$F(v) = \frac{\Delta N(v)}{\Delta v} \text{ , (differensielt)} \qquad \text{eller:} \qquad F(v) = \frac{dN(v)}{dv} \text{ (infinitesimalt)}$$

Ved praktisk bestemmelse av en fordelingsfunksjon må boksbredden være endelig (differensiell).

Når vi har bestemt fordelingsfunksjonen har vi også bestemt sannsynlighetsfordelingen til systemet vi undersøker, i vårt tilfelle partiklene på tilnærmet friksjonsløst og horisontalt underlag. En sannsynlighet er definert som antallet tilfeller med et bestemt utfall dividert på totalt antall tilfeller. Det bestemte utfallet er i vårt tilfelle at partiklene har en bestemt hastighet, ***v***, og antallet av dette er ***ΔN(v)*** . Det totale antallet er i vårt tilfelle ***N***, som er antallet partikler, slik at sannsynlighets fordelingen blir:

$$p(v) = \frac{F(v)}{N} \qquad \text{sannsynlighets fordelingen}$$

Når en summerer over fordelingsfunksjonen får en totalt antall tilfeller N ;

$$\sum_{i=1}^{N} F(v_i) = N \ ,$$

og når en summerer (integrerer) over sannsynlighetsfordelingen får en ***1***;

$\sum_{i=1}^{N} p(v_i) = 1$ , eller: $\int_{v=0}^{\infty} p(v) \cdot dv = 1$

***Maxwell-Boltzmann (M-B) fordelingen for klassiske partikler.***

Vi vil utlede fartsfordelingsfunksjonen for punktpartikler for bevegelse i tre dimensjoner, som er i ***x, y*** og ***z*** retning, ut fra to forutsetninger om systemet.
Den første forutsetningen er at de tre retningene ***x, y*** og ***z*** i rommet er statistisk uavhengige. Dette medfører at fordelingsfunksjonen vil bli et produkt av tre uavhengige fordelingsfunksjoner, en for hver retning, som kalles; $f(v_x)$ , $f(v_y)$ og $f(v_z)$ .

$$dN = f(v_x) \cdot f(v_y) \cdot f(v_z) \cdot dv_x \cdot dv_y \cdot dv_z \quad - \qquad \textit{fordelingsfunksjonen}$$

*dN* er antallet partikler med fart i hastighets intervallet; $v_x + dv_x, v_y + dv_y, v_z + dv_z$ .
Siden aksene er likeverdige, som er den andre forutsetningen, vil fartsfordelings funksjonen bare avhenge av farten ***v***, som er:

$$v = \sqrt{v_x^2 + v_y^2 + v_z^2} \qquad - \qquad \textit{farten}$$

og denne fordelingsfunksjonen (*F)* vil bli en funksjon av ***v*** og er produktet av fordelingsfunksjonene for hver av aksene;

$$dN = f(v_x) \cdot f(v_y) \cdot f(v_z) \cdot dv_x \cdot dv_y \cdot dv_z = F \cdot dv \,, \qquad \text{eller:}$$

$$F(v) = f(v_x) \cdot f(v_y) \cdot f(v_z) \qquad - \qquad \textit{fordelingsfunksjonen med hensyn på fart}$$

Derivasjon av høyresiden til det siste uttrykket med hensyn på $v_x$ gir:

$$\frac{dF(v)}{dv_x} = \frac{dF}{dv} \cdot \frac{dv}{dv_x} = \frac{dF}{dv} \cdot \frac{v_x}{v} \,, \qquad \text{(høyresiden)}$$

og derivasjon av venstresiden gir:

$$\frac{df(v_x)}{dv_x} \cdot f(v_y) \cdot f(v_z) \,. \qquad \text{(venstresiden)}$$

Når disse utrykkene settes like og deles med *F(v),* fås:

$$\frac{1}{v} \cdot \frac{dF(v)}{F(v)} = \frac{1}{v_x} \cdot \frac{df(v_x)}{dv_x} \cdot \frac{1}{f(v_x)} \,.$$

I uttrykket over er dessuten $v_x$ flyttet over på høyresiden. Når to uttrykk, som er en funksjon av to forskjellige frie variable (*v* og $v_x$ kan variere uavhengig av hverandre), er like, betyr det at uttrykkene må være lik en konstant, som settes lik ***–k***. Dermed blir:

$$\frac{1}{v_x}\cdot\frac{df(v_x)}{dv_x}\cdot\frac{1}{f(v_x)}=-k\,,\quad \text{eller: } \frac{df(v_x)}{f(v_{x)}}=-kv_x\cdot dv_x\,.$$

Integrasjon av den siste likningen gir: $f(v_x)=C\cdot\exp(-k\cdot v_x^2/2)$

$C$ er en integrasjonskonstant, og siden $f(v_x)$ er en sannsynlighet, kan den bestemmes ved følgende likhet.

$$1=\int_{-\infty}^{\infty}f(v_x)dv_x=C\cdot\int_{-\infty}^{\infty}\exp(-k\cdot v_x^2/2)dv_x=\frac{C}{2}\cdot\sqrt{\frac{2\pi}{k}}\,,\qquad \text{som gir: } C=\sqrt{\frac{2k}{\pi}}$$

Til slutt lar ***k*** seg bestemme ved bruk av ekvipartisjonsprinsippet, som sier at energien til systemet er likt fordelt mellom de ulike frihetsgradene. Systemet vårt har tre frihetsgrader, og hver frihetsgrad har energien $\frac{1}{2}\cdot k_BT$ , der $k_B$ er Boltzsmanns konstant og $T$ er absolutt temperatur.

På den ene side er, i følge ekvipartisjonsprinsippet, kinetisk energi for x-bevegelsen:

$$E_x^{kin}=\tfrac{1}{2}\cdot k_BT\,.$$

Men den samlede kinetiske energien for x-retning kan også finnes ved integrasjon:

$$E_x^{kin}=\int_0^{\infty}\tfrac{1}{2}\cdot mv_x^2\cdot f(v_x)dv_x=\frac{m}{2}\cdot\sqrt{\frac{2k}{\pi}}\cdot\int_0^{\infty}v_x^2\cdot e^{-\frac{k}{2}\cdot v_x^2}\cdot dv_x=\frac{m}{2}\cdot\sqrt{\frac{2k}{\pi}}\cdot\frac{1}{4}\sqrt{\frac{\pi}{\left(\frac{k}{2}\right)^3}}=\frac{m}{2k}\,,$$

Når disse settes like, fås altså: $k=mk_BT$

(husk $\int_0^{\infty}x^2\cdot e^{-ax^2}dx=\frac{1}{4}\sqrt{\frac{\pi}{a^3}}$ )

og en får fordelingen for det en dimensjonale tilfellet .

$$f(v_x)=\sqrt{\frac{m}{2\pi k_BT}}\cdot\exp(-\frac{mv_x^{\,2}}{2k_BT})$$

Maxwell Boltzmann fordelingen lar seg altså utlede fra prinsippene om at de tre retningene i rommet er statistisk uavhengige, som tilsier at samlet sannsynlighet er et produkt av sannsynlighetene for hver retning, og at aksene er likeverdige, som tilsier at samlet sannsynlighet må være en funksjon av farten.

### *To-dimensjonal fartsfordeling*

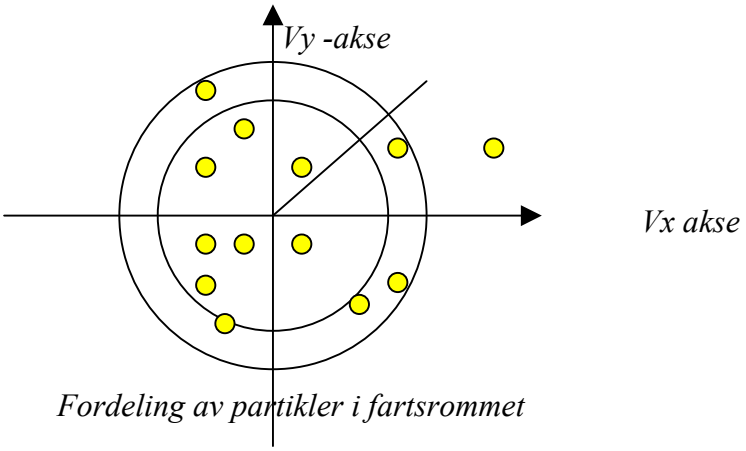
Den en-dimensjonale fartsloven var:

$$F(v_x)=\frac{dN_x}{dv_x}=\sqrt{\frac{m}{2\pi kT}}\cdot\exp(-\frac{mv_x^{\,2}}{2kT})\qquad \text{- } endimensjonalt$$

Dersom vi utvider betraktningen til å gjelde bevegelse i to dimensjoner ($\boldsymbol{v_x;v_y}$), får vi;

$$F(v_x, v_y) = \frac{dN}{dv_x \cdot dv_y} = \frac{dN_x}{dv_x} \cdot \frac{dN_x}{dv_x} = \frac{m}{2\pi kT} \cdot \exp(-\frac{mv^2}{2kT}) \quad \text{- todimensjonalt}$$

I figuren nedenfor er de to hastighetsretningene vinkelrett på hverandre og $dv_x \cdot dv_y$ blir et lite rektangel in $v_x$,$v_y$ planet, og fordelingsfunksjonen over forteller antallet partikler med hastighet i dette rektangelet




*Fordeling av partikler i fartsrommet*

I det to-dimensjonale tilfellet vil vi ofte ha fordelingen av partikler med hensyn på fart, altså antallet mellom $v$ og $v+dv$. Samlet fart er; $v = \sqrt{v_x^2 + v_y^2}$ , og dette er likningen for en sirkel. Mange kombinasjoner av $v_x$ og $v_y$ gir samme $v$ og arealet ($dA$) av sirkelringen med radius $v$ og tykkelse $dv$ er:

$$dA = 2\pi v \cdot dv$$

Fartsfordelingsloven i to dimensjoner blir derfor:

$$\frac{dN}{2\pi v \cdot dv} = \frac{m}{2\pi kT} \cdot \exp(-\frac{mv^2}{2kT}), \quad \text{eller:}$$

$$F(v) = \frac{dN}{dv} = 2\pi v \cdot \frac{m}{2\pi kT} \cdot \exp(-\frac{mv^2}{2kT}) = \frac{m}{kT} \cdot v \cdot \exp(-\frac{mv^2}{2kT}) \quad \text{- } \textit{todimensjonal fartsfordeling}$$

Den mest sannsynlige farten inntrer når *F(v)* er maksimal. Når uttrykket deriveres med hensyn på *v* og en setter dette lik null, vil en finne den mest sannsynlige farten .

Den mest sannsynlige farten vil være: $v_0 = \sqrt{\frac{kT}{m}}$ (vis dette)

slik at uttrykket kan skrives: $f(v) = \frac{\Delta N}{\Delta v} = \frac{N}{\pi \cdot v_0^2} \cdot v \cdot \exp(-\frac{v^2}{2 \cdot v_0^2})$.

***Systemet som skal undersøkes***

Partikler er makroskopiske skiver som beveger seg tilnærmet friksjonsløst på et luftputebord. På grunn av litt friksjon vil de tape energi. Men den vinnes tilbake ved hjelp av støt mot elastiske og bevegelige vegger. Etter hvert blir systemet stasjonært, det vil si at like mye

energi tapes som det tilføres i tidsenheten. Det er skivene med liten fart som tilføres forholdsvis mest energi når de slynges ut fra den vibrerende trådene som danner veggene på luftputebordet.
Vi tenker oss at dette er en modell for molekylbevegelse i gassfase, som ifølge teorien skal følge Maxwell-Boltzmann fordelingen. Vårt system vil avvike litt fra denne, friksjonene gjør at det blir forholdsvis flere med liten fart, og forholdsvis flere med den farten som veggene beveger seg med.

***Ensembel midling og tidsmidling***
Ordet *ensembel* betyr en samling. Om vi måler hastigheten til alle partiklene, hele samlingen, ved et tidspunkt, kan vi finne en hastighetsfordeling. Dette kalles *emsembel* fordeling og *ensembel* midling. Alternativt kunne vi følge en partikkel. Mellom hvert støt vil den ha en bestemt hastighet, som vi kan måle. Etter et støt vil den få en ny hastighet, som igjen lar seg måle ut; og så videre. Vi kan også lage en hastighetsfordeling ut fra målinger på en partikkel og det kalles *tidsmidlet* hastighetsfordeling.
*Ergode* teoremet i statistisk fysikk sier at disse fordelingene skal være like. Vi vil bestemme ensembelmiddelet.

***Midlere frie veilengde***
Når partiklene beveger seg på bordet, vil de før eller senere støte på hverandre. Den midlere frie veilengden ($\lambda$), er gjennomsnittlig avstand mellom to kollisjoner. I følge teori er denne størrelsen lik: $\lambda = \frac{1}{n \cdot d}$, der $n$ er antallet skiver (pucker) pr. kvadratmeter og $\boldsymbol{d}$ er diameteren til skiven. Dette kan ses slik: Når skiven har beveget seg en strekning; $v \cdot t$, der $\boldsymbol{v}$ er farten, vil den ha beskrevet et areal; $A = d \cdot vt$, der $\boldsymbol{d}$ er skivas diameter. Når $\boldsymbol{n}$ er antallet skiver pr. arealenhet, vil den midlere frie veilengde bli; $\lambda = \frac{vt}{A \cdot n} = \frac{1}{d \cdot n}$. Den midlere frie veilengden settes altså lik tilbakelagt strekning ($v \cdot t$) dividert med antallet kollisjoner på denne strekningen, og antallet kollisjoner er lik antallet skiver innen det utskrevne arealet $\boldsymbol{A}$, som blir $\boldsymbol{A}$ multiplisert med antallet partikler pr. arealenhet, $\boldsymbol{n}$. Er det eksempelvis $n = 20$ skiver på luftputebordet, som har ett areal på 1 $m^2$, og når skivene har en diameter på $d = 7cm$, vil den midlere frie veilengden bli 14 *cm*.

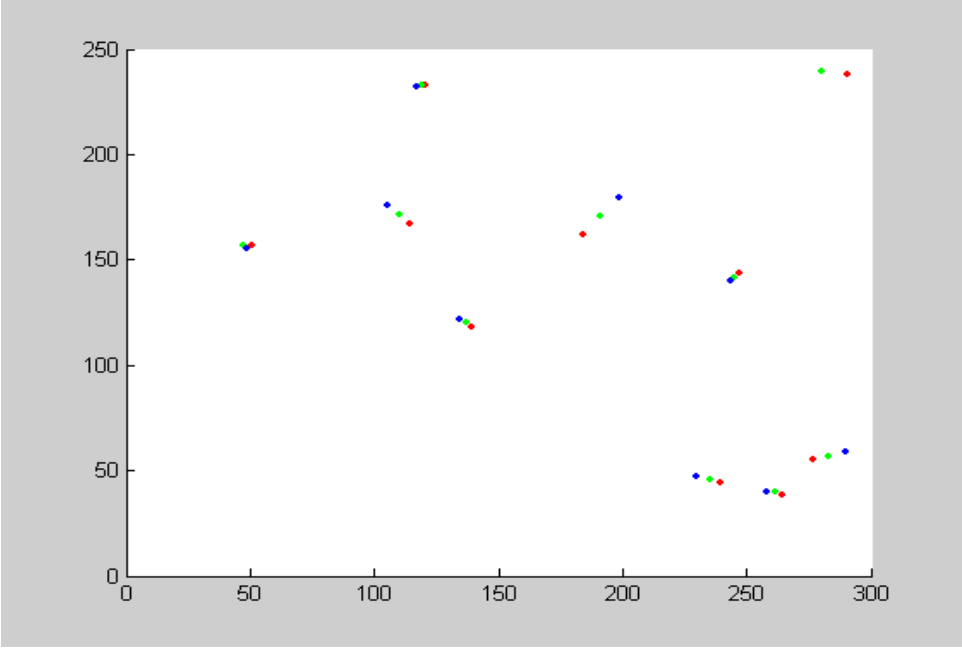
***Praktiske kommentarer***
Luftputebordet er tilnærmet friksjonsløst. De vibrerende veggene illuderer temperatur i vårt makroskopiske system. Med et videokamera blir det tatt bilder 30 ganger i sekundet i et tidsrom på omtrent 10-20 sekunder. Kameraet består av 320 x 240 lysfølsomme områder, *piksler*, som måler intensitetene i alle disse delene av bildet. Skivene på bordet er hvite og runde, i motsetning til bordet som er svart. I *Matlab* finnes det et programverktøy (*image processing toolbox*) som identifiserer runde objekter og finner koordinatene til senteret i sirklene. For sirkler er det et bestemt forhold mellom omkrets og diameter ($\pi$), og Matlab programmet bruker dette kriteriet for å gjenkjenne sirkelrunde objekter. De runde partiklene blir funnet for hvert bilde, og senterpunktet $(x_i, y_i)$ til partikkel $\boldsymbol{i}$ blir angitt med pikselverdier, som er et tall mellom 0 og 320 i x-retning og mellom 0 og 240 for y-retning.
Det tas 30 bilder i sekundet, og når en ser på posisjonsendringer fra bilde til bilde, kan farten til en partikkel bli funnet som;

$$v_i^x = \frac{\Delta x_i}{\Delta t} = \frac{x_{i+1} - x_i}{\Delta t}$$

der $\Delta x_i$ er forskjellen mellom pikselverdiene til to bilder tatt rett etter hverandre, med tidsdifferense $\Delta t = 1/30$ sek. På grunnlag av disse målingene kan en finne fordelingen av partiklene i faserommet, både med hensyn til posisjon og fart.

På bildet under er tre videobilder i rekkefølge blitt analysert og satt sammen til ett. Bildet viser at partiklene beveger seg rettlinjet med ulike hastigheter. Når antallet partikler er lite, vil

pikselverdier til to påfølgende bilder være nabopiksler. Denne egenskapen benyttes når




programmet beregner $\Delta x_i = x_{i+1} - x_i$. ***(i+1)*** er videobildet som følger det forrige ***(i)***. Blant alle pikselverdiene i neste bilde, $x_{j+1}$, velger Matlabprogrammet ut pucken med det *j* tallet som har minst avstand til partikkel *i*.

Det er såpass mange partikler på luftputebordet at de kolliderer ganske ofte. For hvert tiende bilde blir posisjoner og hastigheter plukket ut. En regner da med at alle partiklene har kollidert og en måler ut nye posisjoner og hastigheter. Dette blir da en *emsembelmidling*. Holdbarheten i dette kan kontrolleres ved å regne ut den midlere frie veilengde (λ), som er gjennomsnittlig avstand mellom to kollisjoner.

Når en skal finne fordelingsfunksjonene, må en velge seg et passelig stor intervall bredde (*binbredde*) både for posisjon (Δx og Δy) og hastighet ($\Delta v_x$ og $\Delta v_y$ ). Dette gjøres for eksempel slik at ca 10 % av tilfellene havner i intervallet med størst antall. Den grafiske framstillingen av antall tilfeller som funksjon av den variable kalles histogram framstilling. For en passelig valgt bredde for intervallene, *binbredden*, telles opp antallet tilfeller innen disse intervallene.

Farten til en partikkel blir,

$$v = (v_x^{\,2} + v_y^{\,2})^{1/2}, \qquad \textit{fart}$$

og denne kan regnes ut fra målingene av x-og y komponentene til farten.

Temperaturen (*T*) til partiklene blir etter definisjonen;

$$\tfrac{1}{2} \cdot kT = \tfrac{1}{2} \cdot m \cdot \langle v^2 \rangle = \sum_{i=1}^{N} \tfrac{1}{2} \cdot m \cdot v_i^2 / N, \quad \text{altså midlere kinetiske energi.}$$

***Ta bildene slik:***
-***Videokameraet*** ligger under ikonet Logitech; *Dobbelklikk* ***Logitech Qickcam***
*og aktiver* ***Quick Capture***
Om du vil sjekke/forandre innstillinger, bruk: ***Logitech control panel***
***Avanserte innstillinger; Exposure 1/30 sec og Zoom Ca.140***
(da får en med bare bordet). Du kan også bruke "face tracking" for å rette kamera inn sentralt mot bordet (eller flytte litt på bordet).
-***Video opptak***, bruk: ***Quick Capture – Record a video***. *Ikke la det gå lenger enn 15-20 sek* (det blir 15x 30 bilder, og hver bilde har 320x240 pixler = 0.076 Megapixler.

-***Bilder av puck***, for størrelseskalibrering. Bruk: ***Take a picture***.

De siste bildene som er tatt får høyest indeks i arkivet (my logitech pictures). Bilder og Videoer blir lagrert automatisk under ***My Logitech Pictures***.
-***Lagre*** dette videoopptaket (save as) og bilder under Matlab, på området:
***E/Matlab/work/luftpute/Video 22.avi*** (hvis f.eks videoen du tok hadde indeks 22).

***Beskrivelse av Matlab-programmer***

*-main (hovedprogrammet)*

Her leses inn Video og bilder. Bilder blir tatt for å kalibrere avstander på bordet mot pikselavstander i kameraet. Hvert tiende bilde blir analysert. En regner med at i mellomtiden har det skjedd kollisjoner og at en har en ny fordeling. Det skjer ved hjelp av programmpakke i Matlab som heter FinnRund. Runde objekter i bildet blir identifisert, og pikselverdien til senterposisjonen blir angitt. Denne blir omsatt til massemiddelpunket til pucken.

*-finnrund*

Ligger som Toolbox under Matlab, plukker ut runde objekter i bildet, og finner massemiddelpunktet til de runde objektene.

*-finnfart*

Hastigheter og fart finnes ved hjelp av programmet finnfart.

Dette gjøres ved at analyserer to påfølgende bilder. Finnrund programmet har funnet to sett med koordinater for skivene fra de to bildene. Finnfart programmet finner nærmeste nabo til skiven med koordinaten ($x_i$,$yi$) i neste bilde, for puck med indeks i. En forutsetter at nærmeste nabo i neste bilde har samme indeks, det vil si at skivene beveger seg lite i forholdet til avstanden mellom dem (dette kan gå galt noen ganger). Dermed kan hastighet finnes som avstand dividert med tidsrom mellom to påfølgende bilder.

*Histogrammer*
Disse lages ved bruk av Excel.

***APPENDIX***
***Programmbeskrivelse.***

***Main*** kalles rett fra matlab og trenger ingen variable. Main spør først brukeren etter filnavn og to bilder av enslige skiver i bestemt avstand for å finne sammenhengen mellom pikselverdier i kameraet og koordinater på luftputebordet.

***Subrutine Finnrund***
Dette programmet finner sirkulære objekter i et bilde (frame) i filmen.

Programmet henter inn bildet via *AVI read* og konverterer det til *index* bilde ved hjelp av *frame2im.*
Begge disse rutinene er innebygde i matlab.
Bildet blir så gråtoneskalert ved hjelp av ***im2bw*** rutinen og gråtone skalaen bestemmes av ***kontrast*** konstanten og ***graythres/rbg2gray*** rutinene.
***bwareaopen*** benyttes til å fjerne objekter som er mindre enn filter størrelsen. Et sirkel objekt blir definert ved hjelp av ***strel*** og bildet blir «lukket» og fylt slik at bare skivene blir igjen. (***imclose*** og ***imfill*** rutinene benyttes).

Grensene til disse objektene blir bestemt (Det benyttes ***bwboundaries*** og ***regionprops****)*

Grensene til objektene blir lest ut, arealet beregnet og sammenlignet med sirkelegenskapene. Hvis objektet godkjennes, lagres posisjonene i returvariablene.
Dette gjøres for alle identifiserte objekter i bildet.

***Kontrast*** (Skalerings parameter som skiller puckene fra bakgrunnen)
***Filter*** (Filtrerings parameter som filtrerer vekk objekter som er mindre enn kamera-oppløsningen)
***Sirkel*** (Parameter som angir største avvik fra perfekt sirkel som godtas som puck.)

Dette gjentas for hvert tiende bilde i filmen; hvert tiende bilde regnes å være en ny fordeling av skivene.
Alle posisjonene lagres i vektoren ***xx, yy***.

***Finnfart*** prosedyren finner hastigheter som skivene har for hvert tiende bilde. Posisjoner finnes for hvert tiende og ellevte bilde, og fra de to påfølgende bildene kan posisjonsendringer beregnes. Det er forholdsvis få skiver på bordet og de forflytter seg ganske lite i løpet av 1/30 sek, derfor blir nærmeste naboposisjon i neste bilde assosiert med samme skive. Dette gjøres for alle (n+10, n+11) bildene i videoen, altså 1,2; 10,11, 20,21, osv; til hele filmen er analysert. Programmet returnerer alle hastightene som en vektor ***vx, vy***.

## *Programmer*

Main

```
% Main rutine som kaller, koordinerer og kjører nødvendige rutiner,
% hånderer også data
disp('2 bilder og en video ligger lagret på: e\Matlab\work\luftpute; kalt 1,jpg,2.jpg og Video
1.avi,');
disp('hvis ikke; så sørg for dette');
pause

%Leser inn nødvendige data fra bruker.
film = input('Navnet på filmen: ','s');
bilde1=input('Navnet på første bilde: ','s');
bilde2=input('Navnet på andre bilde: ','s');

da = input('Valgt avstand mellom "pucken" i bilde 1 og 2 [cm] ');
```

```
%Henter ut antall frames og frames per sekund fra filmen
Vdat = aviinfo(film);
bilder = Vdat.NumFrames
fps = Vdat.FramesPerSecond


disp('Programmet vil nå beregne avstanden i pixler mellom de to puckene i');
disp('bilde en og to, for så å finne ut pixel til meter forholdet ved valgte instillinger');
pause
% Visualsiering av hvordan senteret i sirklene beregnes
demo = input('Ønsker du å se hvordan bildet blir dekomponert? [y/n] ','s');
[x1,y1] = pos(bilde1);
demo = 'n';
[x2,y2] = pos(bilde2);

%Antall pixler pr meter beregnes
ppm = sqrt((x1-x2)^2 +(y1-y2)^2)/(10^-2 *da);


disp('Programmet henter nå ut informasjon om posisjoner og hastigheter til puckene fra
filmen, dette tar litt tid');
pause
[xx,yy,vx,vy]=finnfart(bilder,fps,film,ppm); % Gir posisjonen til puckene og hastigheten til
disse

% Plotter XY posisjonene
figure
plot(xx,yy,'x');
xlabel('Posisjon i X-retning, oppgitt i meter')
ylabel('Posisjon i Y-rettning oppgitt i meter')
title('Posisjon i XY-planet')

% Plotter XY-hastighetene
figure
plot(vx,vy,'x');
xlabel('X-hastighet [m/s]')
ylabel('Y-hastighet [m/s]')
title('Hastighet i XY-planet')

% Beregner lengden av v i kvadrat
vv = sqrt((vx.^2+vy.^2));
fart=vv;
save F:\Fy1005\fart.dat fart -ascii;

disp('Hastighetene v[i] er lagret i datafil F:\Fy1005\fart.dat')
disp('The end.')
```

## *finnfart*

```
function[xx,yy,vx,vy]=finnfart(bilder,framesprsek,filnavn,ppm)

% k avstand mellom målinger for bestemmelse av posisjonsendring
k = 3; %
% Tidsintervall for målinger
t = k/framesprsek;
%Antall bilder mellom hver måling
m=10;

%Parametre benyttet i finnrund prosedyren
kontrast    = 2.5;
%kontrast    = 1.6;
filter      = 5;
%filter      = 10;
sirkel      = 0.7;

%For sløyfevariable
i=1;
j=0;

%Looper alle bilder fra 1 til #bilder med økning på m bilder pr loop
for n=1:m:(bilder-20)
%for n=1:m:500
   prosentferdig=n./bilder.*100
   %Kaller finnrund prosedyren (egen m-fil). Denne returnerer (x,y)
   %koordinatene til en "puck"
   [x0,y0]=finnrund(n,filnavn,kontrast,filter,sirkel);
   [x1,y1]=finnrund(n+k,filnavn,kontrast,filter,sirkel);
   %Kaller logikk prosedyren for å kombinere puckene
   [dx,dy]=logikk(x0,x1,y0,y1);
   a=sqrt(dx.^2+dy.^2);
   j=j+1;
   for n=1:length(a)
      m=n+i;
      vx(m)=(dx(n)/(ppm*t));
      vy(m)=(dy(n)/(ppm*t));
      xx(m)=x0(n)/ppm;
      yy(m)=y0(n)/ppm;
   end
   i=i+length(a);
end
```

## *finnrund*

```
function[xpos1,ypos1]=finnrund(frame,fil,kontrast,filter,sirkel)

% kontrast: Forteller hvor stor kontrast en sirkel må ha for å bli gjenkjent
% sirkel: forteller hvor sirkulært objektet må være for å karakteriseres som
% en sirkel. 1 = perfekt sirkel
```

```
% filter: filtrerer bort områder mindre en angitt verdi i pixler
r = 2; %Radius på morfologisk disk


%Bildeinnlesing gråtone skalering og konvertering
[RGB,Map] = frame2im(aviread(fil,frame));       %Leser inn bilde fra filmen og konverterer
filmbildet til Index bilde
%imshow(RGB);                                   %Viser bildet i farge
bw = im2bw(rgb2gray(RGB),graythresh(rgb2gray(RGB)).*kontrast);   %Konverterer
fargebildet til svart/hvit og setter gråtonene
%imshow(bw)                                     %Viser svart/hvitt bildet

%Filtrering
bw = bwareaopen(bw,filter);          % Fjerner objekter mindre enn filterstørrelsen
se = strel('disk',r);                % Lager strukturelement disk med radius r
bw = imclose(bw,se);
% Lukker åpne områder:
bw = imfill(bw,'holes');
%imshow(bw)                          %Viser bildet

[B,L] = bwboundaries(bw,'noholes');

%Opptegning av grensene:
%imshow(label2rgb(L,@jet,[.5 .5 .5]));
%hold on

for k = 1:length(B)
   boundary = B{k};
   %plot(boundary(:,2), boundary(:,1),'w','LineWidth,2);
end

stats = regionprops(L,'area','Centroid');
i = 1;
% looper alle områder
for k = 1:length(B)
   %Henter randinformasjon til objekt k
   boundary = B{k};
   %Estimerer randen
   delta_sq = diff(boundary).^2;
   perimeter = sum(sqrt(sum(delta_sq,2)));
   area = stats(k).Area;
   metric = 4*pi*area/perimeter^2;
   if metric > sirkel
      centroid = stats(k).Centroid;
      posx(i)=centroid(1);
      posy(i)=centroid(2);
      i = i + 1 ;
   end

end
```

```
xpos1=posx;
ypos1=posy;
%figure
%plot(posx,posy,'x');
%pause(0.1);
```

## *logikk*

```
% Logikk prosedyren får inn to vektorer med lengde m og n, og returnerer to
% vektorer dx, og dy med lengde m
% Denne prosedyren skal kombinere hvilke pucker som hører sammen,
% dette gjøres ved at posisjonen R0-R1 minimeres, hvor R er avstanden
% mellom pucker fra bilder med 2 frames avstand på filmen.
function [dy,dx] = logikk(xpos1,xpos2,ypos1,ypos2)

for m=1:length(xpos1)
   %Løper gjennom alle elementer i vektor X1
   for n=1:length(xpos2)
      %Delta er avstanden mellom fra puck m i (X0,Y0) til alle pucker i
      %(X1,Y1)
      delta(n)=sqrt((xpos2(n)-xpos1(m)).^2+(ypos2(n)-ypos1(m)).^2);
   end
   % I er nummeret på elementet i delta som er minst
   [avst,I]=min(delta);
   %Lagrer nummeret til indeksen i ny variabel og avstanden mellom de.
   i(m)=I;
   avstand(m)=avst;
   %Lagrer  avstanden i X og Y rettning i utgangsvariablene.
   dx(m)=xpos1(m)-xpos2(I);
   dy(m)=ypos1(m)-ypos2(I);
   %xpos2(I) = 10^100;
   %ypos2(I) = 10^100;
end
%Returnerer variablene dx og dy
```

## *sekvens*

```
clear
fil='gruppe7a.avi';
Vdat=aviinfo(fil);
bilder=Vdat.NumFrames;
fps=Vdat.FramesPerSecond;
kontrast=2;
filter=10;
sirkel=0.7;
frame=100;
fil='gruppe7a.avi';

[x1,y1]=finnrund(frame,fil,kontrast,filter,sirkel);
```

```
[x2,y2]=finnrund(frame+2,fil,kontrast,filter,sirkel);
[x3,y3]=finnrund(frame+4,fil,kontrast,filter,sirkel);
figure
hold on
plot(x1,y1,'r.')
hold on
plot(x2,y2,'g.')
hold on
plot(x3,y3,'b.')
hold off
```