Innlevering i nr. 4 i DAFE 1000
Løysingsforslag

Oppgåve 1

a) $\int_{-1}^{2} x \sin x^2 \, dx$

Dette integralet kan vi rekne ut med variabelbytet $u = x^2$.

$$\frac{du}{dx} = 2x, \quad dx = \frac{1}{2x} du$$
$$u(-1) = (-1)^2 = 1$$
$$u(2) = 2^2 = 4$$

Vi får

$$\int_{-1}^{2} x \sin x^2 \, dx = \int_{u(-1)}^{u(2)} x \sin u \, \frac{1}{2x} du = \frac{1}{2} \int_{1}^{4} \sin u \, du = \frac{1}{2} \left[-\cos u \right]_1^4 =$$
$$\frac{1}{2}(-\cos 4 - (-\cos 1))) = \underline{\frac{\cos 1 - \cos 4}{2}} \approx 0.5970 \quad .$$

b) $\int_{-1}^{1} \sin x^2 \, dx$

Dette integralet kan ikkje løysast ved antiderivasjon. I staden må vi bruke ein eller annan numerisk metode for å estimere det. Vi kan til dømes bruke trapesmetoden. Den kan implementerast slik:

```
% Skript som reknar ut eit integral ved trapesmetoden
% Input er funksjonen som skal integrerast, integranden,
% og grensene a og b. Desse er hard-koda.
% Oppdelinga, gitt ved talet på del-intervall, N, blir gitt
% frå kommandovindauga ved input-funksjonen

% Integranden
f=@(x) sin(x^2);

% Grensene
a=-1;
b=1;

% Oppdeling
N=input('Kor mange delintervall skal vi bruke? ');
dx=(b-a)/N;

% Initerar summen og tar med første og siste ledd
```

```
19  T=1/2*(f(a)+f(b))*dx;
20
21  % Summerar over alle "indre" punkt
22  x=a;
23  for n=1:(N-1)
24    x=x+dx;                    % Oppdaterar x
25    T=T+f(x)*dx;               % Oppdaterar summen
26  end
27
28  % Skriv estimatet til skjerm
29  T
```

For å kontrollere at estimatet har minst fire rette desimalar, køyrer vi skriptet, som heiter "Trapes.m", fleire gonger med stadig høgare `N`-verdi:

```
>> Trapes
Kor mange delintervall skal vi bruke? 10
T =
    0.6278
>> Trapes
Kor mange delintervall skal vi bruke? 20
T =
    0.6223
>> Trapes
Kor mange delintervall skal vi bruke? 50
T =
    0.6208
>> Trapes
Kor mange delintervall skal vi bruke? 100
T =
    0.6206
>> Trapes
Kor mange delintervall skal vi bruke? 200
T =
    0.6206
```

Altså: $\int_{-1}^{1} \sin x^2 \, dx \approx \underline{0.6206}$.

c) $\int_0^1 x^2 e^{2x} \, dx$

Dette integralet kan vi bestemme eksakt ved å bruke delvis integrasjon to gonger.
Delvis integrasjon:

$$\int_a^b u(x)\, v'(x)\, dx = [u(x)\, v(x)]_a^b - \int_a^b u'(x)\, v(x)\, dx \quad .$$

Med $u = x^2$ og $v' = e^{2x}$, får vi $u' = 2x$ og kan velge $v = 1/2\, e^{2x}$:

$$\int_0^1 x^2 e^{2x}\, dx = \left[x^2 \frac{1}{2} e^{2x}\right]_0^1 - \int_0^1 2x \frac{1}{2} e^{2x}\, dx = \frac{1}{2}\left(1^2 e^2 - 0\right) - \int_0^1 x e^{2x}\, dx = \frac{e^2}{2} - \int_0^1 x e^{2x}\, dx$$

For integralet som står att, har vi $u(x) = x$ slik at $u' = 1$ mens v' er som før:

$$\int_0^1 x e^{2x} = \left[x \frac{1}{2} e^{2x}\right]_0^1 - \int_0^1 1 \cdot \frac{1}{2} e^{2x}\, dx = \frac{1}{2}\left(1e^2 - 0\right) - \frac{1}{2}\int_0^1 e^{2x}\, dx = \frac{e^2}{2} - \frac{1}{4}\left[e^{2x}\right]_0^1 = \frac{e^2}{2} - \frac{1}{4}\left(e^2 - 1\right) = \frac{e^2 + 1}{4}$$

Alt i alt får vi altså at

$$\int_0^1 x^2 e^{2x}\, dx = \frac{e^2}{2} - \frac{e^2 + 1}{4} = \underline{\frac{e^2 - 1}{4}} \approx 1.5973 \quad .$$

d) $\int_{-2}^{2} x^2 e^{x^2}\, dx$

Igjen har vi eit integral som ikkje let seg løyse ved antiderivasjon. Vi endrar skriptet frå c) slik at det estimerar dette integralet. Det gjer vi ved å oppdatere linje 7 til 12:

```
% Integranden
f=@(x) x^2*exp(x^2);

% Grensene
a=-2;
b=2;
```

Igjen køyrer vi skriptet til vi har fått bestemt minst fire siffer. Denne gongen treng vi ein bruke ganske mange delintervall:

```
...
>> Trapes
Kor mange delintervall skal vi bruke? 1000
T =
   92.7466
>> Trapes
Kor mange delintervall skal vi bruke? 2000
T =
   92.7444
>> Trapes
Kor mange delintervall skal vi bruke? 5000
T =
```

```
   92.7438
>> Trapes
Kor mange delintervall skal vi bruke? 10000
T =
   92.7437
>> Trapes
Kor mange delintervall skal vi bruke? 20000
T =
   92.7437
```

Altså: $\int_{-2}^{2} x^2 e^{x^2}\,dx \approx \underline{92.7437}$.

Oppgåve 2

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\,dt$$

a) Som i deloppgave 1 c) og d) har vi med eit integral som ikkje let seg bestemme ved antiderivasjon å gjere. For å rekne ut erf(1), kan vi godt bruke trapes-skriptet frå oppgåve 1 – med passe justeringar. Men sidan vi i delppgåve c) skal implementere Newtons metode for eit problem der erf(x) er med, er det meir praktisk å lage ei *funksjonsfil* for feilfunksjonen. Dette kan gjerast slik:

```
1  function F=FeilFunk(x,N)
2
3  % Funksjonen estimerar feilfunksjonen ved hjelp av trapesmetoden.
4  % Input er argumentet og N, som er talet på delintervall vi brukar
5  % i trapes-estimatet.
6
7  % Integranden
8  f=@(t) 2/sqrt(pi)*exp(-t^2);
9
10 % Grensene
11 a=0;
12 b=x;
13
14 % Oppdeling
15 dx=(b-a)/N;
16
17 % Initerar summen og tar med første og siste ledd
18 F=1/2*(f(a)+f(b))*dx;
19
20 % Summerar over alle "indre" punkt
21 x=a;
```

```
22 for n=1:(N-1)
23   x=x+dx;                    % Oppdaterar x
24   F=F+f(x)*dx;               % Oppdaterar summen
25 end
```

Legg merke til at implementeringa er svært lik "Trapes.m". Legg også merke til at funksjonfila tar to input-parametrar – argumentet x og `N`, talet på delintervall i trapesmetoden. Sjølvsagt får ikkje svaret lov til å vere avhengig av `N`. Difor er det som vanleg viktig at vi kontrollerar at `N` er stor nok:

```
>> format long
>> FeilFunk(1,10)
ans =
   0.842008716616823
>> FeilFunk(1,50)
ans =
   0.842673118747576
>> FeilFunk(1,200)
ans =
   0.842699063333701
>> FeilFunk(1,500)
ans =
   0.842700516211347
>> FeilFunk(1,1000)
ans =
   0.842700723765129
```

Altså: $\mathrm{erf}(1) \approx \underline{0.842701}$. Lat oss sjekke at MATLAB er enig; implementeringa som finst i MATLAB frå før, heiter, ikkje overraskande, '`erf`':

```
>> erf(1)
ans =
   0.842700792949715
```

Med rett avrunding har vi seks like desimala; dette var vel innafor.

b) For å rekne ut den deriverte av feilfunksjonen, brukar vi fundamentalteoremet for kalkulus. Dette seier at dersom $F(x)$ er ein antiderivert til $f(x)$, altså dersom $F'(x) = f(x)$, er

$$\int_a^b f(x)\,dx = F(b) - F(a) \quad .$$

Om vi no tenkjer oss at $F(t)$ er ein antiderivert til e^{-t^2}, slik at $\mathrm{erf}(x) = 2/\sqrt{\pi}\,\int_0^x f(t)\,dt$, får vi at

$$(\mathrm{erf}(x))' = \frac{2}{\sqrt{\pi}}\,(F(x) - F(0))' = \frac{2}{\sqrt{\pi}}\,\left(F'(x) - (F(0))'\right) = \frac{2}{\sqrt{\pi}}\,(f(x) - 0) = \underline{\frac{2}{\sqrt{\pi}}\,e^{-x^2}} \quad .$$

c) I dette tilfellet skal vi iterere på denne formelen:

$$x_{n+1} = x_n - \frac{\text{erf(x)} - 0.8}{\frac{2}{\sqrt{\pi}} e^{-x^2}} \quad .$$

Her har vi brukt svaret frå b). Delppgåve a) gir hint om at $x_0 = 1$ kan vere eit bra val av utgongspunkt. Når vi no implementerar Newtons metode, brukar vi funksjonsfila frå a), sjå linje 29:

```
1  % Skript som løyser likninga
2  % erf(x)=0.8
3  % ved hjelp av Newtons metode.
4  % Den brukar funksjonsfila FeilFunk.m får å rekne ut
5  % erf(x). Denne funksjonsfila tar inn den numeriske
6  % parameteren N, som gir talet på delintervall i den
7  % numeriske berekninga av feil-funksjonen, erf(x). Denne
8  % vert lest inn frå skjerm ved hjelp av input-funksjonen.
9  % Det er viktig å kontrollere at sluttsvaret er uaghengig av N.
10
11 % Start-gjetting, x0:
12 x=1;
13
14 % Oppdelinga som blir brukt i integralet
15 N=input('Kor mange delintervall skal vi bruke? ');
16
17 % Den deriverte av erf(x):
18 erfDeriv=@(x) 2/sqrt(pi)*exp(-x^2);
19
20 % Presisjon
21 Pres=1e-6;
22
23 % Initerar xOld - den førre x-verdien
24 xOld=100;
25
26 % Itererar etter Newtons metode
27 while abs(x-xOld)>Pres
28   xOld=x;                                % Oppdaterar gamal x
29   x=x-(FeilFunk(x,N)-0.8)/erfDeriv(x);  % Newtons metode
30 end
31
32 % Skriv svaret til skjerm (med mange desimalar)
33 format long
34 x
35 format short
```

Her har vi ingen garanti for at løysinga som skriptet estimerar faktisk har den presisjonen som er sett i linje 21; dette skriptet, som vi har kalla "NewtonFeilFunk.m", tek ikkje omsyn til at svaret vi får frå funksjonsfila

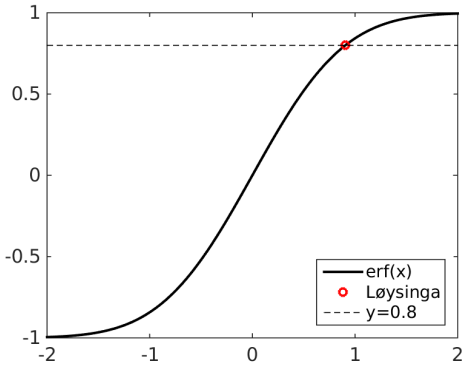
"FeilFunk.m" er `N`-avhengig. Igjen må vi køyre skriptet fleire gonger – med stadig aukande `N`:

```
>> NewtonFeilFunk
Kor mange delintervall skal vi bruke? 100
x =
   0.906206205431961
>> NewtonFeilFunk
Kor mange delintervall skal vi bruke? 200
x =
   0.906196903107621
>> NewtonFeilFunk
Kor mange delintervall skal vi bruke? 500
x =
   0.906194298540655
>> NewtonFeilFunk
Kor mange delintervall skal vi bruke? 1000
x =
   0.906193926462662
```

Det ser ut til å vere nok med 1000 delintervall; løysinga er $x \approx \underline{0.906194}$. For å sjekke at svaret ser ut til å henge på greip, kan vi plotte løysinga saman med feil-funksjonen:

```
>> xx=-2:1e-2:2;
>> plot(xx,erf(xx),'k-','linewidth',2)
>> hold on
>> plot(x,0.8,'ro','linewidth',2)
>> plot([-2 2],[0.8 0.8],'k--')
>> hold off
>> set(gca,'fontsize',15)
>> legend('erf(x)','Løysinga','y=0.8','location','southeast')
```

Resultatet er vist i figuren under.



Oppgåve 3

$$p(x) = x^{1/5} + \sin\left(\frac{\pi}{3}x\right), \quad D_p = [0, 6]$$

a) Vi kan rekne ut volumet ved å bruke denne formelen:

$$V = \pi \int_a^b [f(x)]^2 \, dx \quad ,$$

der $a = 0$, $b = 6$ og $f(x) = p(x) = x^{1/5} + \sin\left(\frac{\pi}{3}x\right)$ i vårt tilfelle. Det spelar ikkje noko rolle at x-aksen her peikar oppover og ikkje mot høgre, volumet er det same likevel.

Vi skal altså rekne ut integralet

$$V = \pi \int_0^6 \left(x^{1/5} + \sin\left(\frac{\pi}{3}x\right)\right)^2 dx =$$
$$\pi \int_0^6 \left(x^{2/5} + 2x^{1/5}\sin\left(\frac{\pi}{3}x\right) + \sin^2\left(\frac{\pi}{3}x\right)\right)^2 dx \quad .$$

Det første leddet i parantesen til høgre er lett å antiderivere. Det siste leddet let seg også gjere ved antiderivasjon. Men leddet i midten er det verre med. Vi gjer det numerisk i staden.

Igjen brukar vi ein modivisert versjon av implementeringa av trapesmetoden frå oppgåve 1:

```
% Skript som reknar ut volumet vi frå når vi roterar grafen til
% ein funksjon omkring x-aksen.
% Skriptet brukar trapesmetoden for å gjere dette.
% Input er funksjonen som skal roterast og grensene a og b.
% Desse er hard-koda.
% Oppdelinga, gitt ved talet på del-intervall, N, blir gitt
% frå kommandovindauga ved input-funksjonen

% Funksjonen som skal roterast
f=@(x) x^(1/5)+sin(pi/3*x);
% Integrand
Int=@(x) pi*(f(x))^2;

% Grensene
a=0;
b=6;

% Oppdeling
N=input('Kor mange delintervall skal vi bruke? ');
dx=(b-a)/N;

```

```
% Initerar summen og tar med første og siste ledd
T=1/2*(Int(a)+Int(b))*dx;

% Summerar over alle "indre" punkt
x=a;
for n=1:(N-1)
  x=x+dx;                    % Oppdaterar x
  T=T+Int(x)*dx;             % Oppdaterar summen
end

% Skriv estimatet til skjerm
T
```

Vi køyrer skriptet, som heiter "RotasjonsVolum.m", heilt til vi synest vi har mange nok rette desimalar:

```
>> RotasjonsVolum
Kor mange delintervall skal vi bruke? 100
T =
   33.5556
>> RotasjonsVolum
Kor mange delintervall skal vi bruke? 200
T =
   33.5634
>> RotasjonsVolum
Kor mange delintervall skal vi bruke? 500
T =
   33.5670
>> RotasjonsVolum
Kor mange delintervall skal vi bruke? 1000
T =
   33.5679
```

Volumet er 33.57 liter.

b) Dersom vatnet når høgda h, blir volumet av vatnet, på same måte som i deloppgåve a), gitt ved[1]

$$V = \pi \int_0^h [p(x)]^2 \, dx \quad .$$

Merk no at sidan vassmengda avtar, vil bådet volumet V og høgda h vere funksjonar av tida t. Likninga over koplar fartane som desse varierar med – altså $V'(t)$ og $h'(t)$. Om vi deriverar begge sider, brukar kjerneregelen og så fundamentalteoremet slik som i oppgåve 2b), får vi at

$$V'(t) = \frac{dV}{dh} \cdot \frac{dh}{dt} = \pi \, [p(h)]^2 \cdot h'(t) \quad .$$

[1] Her har vi tatt for gitt at vasen har nokså tynt skal.

I tillegg har vi fått gitt at V *avtar* med farten $k\sqrt{h}$. Altså har vi også at

$$V'(t) = -k\sqrt{h} \quad .$$

Dermed, sidan begge desse uttrykka skal vere lik $V'(t)$, må vi ha at

$$\begin{aligned} -k\sqrt{h} &= \pi \left[p(h)\right]^2 \cdot h'(t) \\ h'(t) &= -\frac{1}{\pi \left[p(h)\right]^2} k\sqrt{h} \\ h'(t) &= -\frac{k}{\pi} \frac{\sqrt{h}}{\left[p(h)\right]^2} \end{aligned}$$

Denne differensiallikninga vil, saman med eit startkrav, bestemme funksjonen $h(t)$ – altså korleis vasshøgda varierar med tida.

c) I utgongspunkt løyser vi startverdiproblem av typen

$$y' = F(x, y) \quad \text{der} \quad y(x_0) = y_0 \quad \text{er gitt}$$

med Eulers metode. Her er problemstillinga litt enklare sidan høgresida ikkje er eksplisitt avhengig av variabelen x – eller t, som han heiter her[2]. Dette utnyttar vi til å gjere implementeringa litt enklare.

Men der er ein ting som kompliserar implementeringa noko; om den numeriske løysinga vår for $h(t)$ skulle bli negativ for ein eller annann t, vil høgresida i differensiallikninga bli kompleks. Dette problemet unngår vi dersom vi implementerar Eulers metode med ei `while`-løkke som stoppar i det h blir negativ.

Dette kan gjerast slik:

```
% Dette skriptet som løser ei autonom differensiallikning,
% y'(t)=F(y).
% Input er funksjonen F(y), startverdien for x, x0,
% startverdien for y, y0, og steglengda h.
% Metoden er implementert slik at iterasjonane fortset
% så lenge y er positiv.
% Alle input-variablar utanom h er hardkoda.
% Den numeriske parameteren h blir gitt frå kommandovindauga
% ved input-funksjonen.

% Gir differensiallikninga
p=@(x) x^(1/5)+sin(pi/3*x);             % Profil-funksjonen
k=0.067;
F=@(y) -k/pi*sqrt(y)/(p(y))^2;

% Startkravet
x0=0;
```

[2] Vi seier at differensiallikninga er *autnom*.

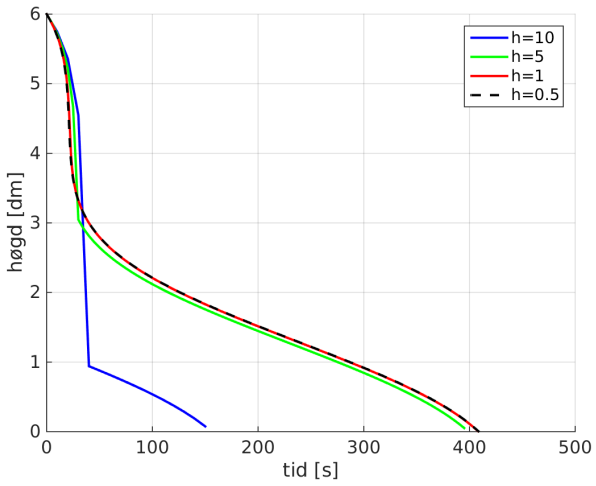
```
y0=6;

% Gir steglengda frå kommandovindauga
h=input('steglengda: ');

%
% Eulers metode - implementert ved ei while-løkke
%
% Start-verdiar
x=x0;
y=y0;
n=1;
while y>=0
  xVektor(n)=x;         % Tilordnar x- og y-verdiane til vektorar
  yVektor(n)=y;
  x=x+h;                % Oppadterar x
  y=y+F(y)*h;           % Oppdaterar y
  n=n+1;                % Oppdaterar vektor-indeksen
end

% Skriv siste x-verdi til skjerm
xFinal=x

% Plottar resultatet
hold on
plot(xVektor,yVektor,'linewidth',2)
set(gca,'fontsize',15)  % Skriftstorleik
xlabel('tid [s]')       % Tekst på aksane
ylabel('høgd [dm]')
grid on                 % Rutenett
hold off
```

Steglengda `h` blir gitt frå kommandovindauga når vi køyrer skriptet. Merk at skriptet, i tillegg til å plotte det numeriske estimatet, også skriv den siste t-verdien, eller `x`-verdien, til skjerm. Når vi køyrer skriptet finn vi at `h`=0.5 er lite nok:

```
>> EulersMetWhile
steglengda: 10
xFinal =
   160
>> EulersMetWhile
steglengda: 5
xFinal =
   400
>> EulersMetWhile
steglengda: 1
```

```
xFinal =
   409
>> EulersMetWhile
steglengda: .5
xFinal =
   409
```

Resultatet, plottet, er vist i figuren under. Legg merke til kor fort høgda



avtar akkurat i det vassflata passerar halsen på vasen – etter omtrent 20 sekund.

d) Av plottet ser vi at det tar drygt 400 sekund å tømme vasen. Meir nøyaktig er skjermutskrifta av `xFinal`, som gir 409 sekund for `h=0.5`. Om vi køyrer skriptet eit par gonger til med lågare `h`-verdiar, finn vi at 409.2 sekund er meir nøyaktig. Om vi nøyer oss med å finne tida avrunda til næraste heile sekund, får vi altså at det tar 409 sekund – eller 6 minutt og 49 sekund.

e) Farten som volumet endrar seg med, $V'(t)$, får no eit ledd til – eit positivt eitt. I tillegg renn det framleis vatn ut med farten $k\sqrt{h}$ – målt i liter per sekund. Dermed har vi at

$$V'(t) = -k\sqrt{h} + 0.15 \quad .$$

Etterkvart som vasshøgda aukar, vil det renne ut vatn med same fart som det renn inn. Når vi nærmar oss denne situasjonen, vil altså volumet V ikkje lenger endre seg; $V'(t) = 0$. Det gir at

$$\begin{aligned} 0 &= -k\sqrt{h} + 0.15 \\ k\sqrt{h} &= 0.15 \\ h &= \left(\frac{0.15}{k}\right)^2 = \left(\frac{0.15}{0.067}\right)^2 = 5.0123 \end{aligned}$$

Altså vil vasshøgda nærme seg 5.0123 dm.

f) Resonnementet fra deloppgåve b) – det som leidde fram til at $V' = \pi[p(h)]^2\, h'$ – er framleis rett:

$$-k\sqrt{h} + 0.15 = \pi\left[p(h)\right]^2 h'$$
$$h'(t) = \frac{0.15 - k\sqrt{h}}{\pi\left[p(h)\right]^2} \quad .$$

Dette er altså den oppdaterte differensiallikninga vår. Sidan vi startar med tom vase denne gongen, er startkravet

$$h(0) = 0 \quad .$$

Vi oppdaterar linje 11 til 18 i skriptet frå deloppgåve c)

```
% Gir differensiallikninga
p=@(x) x^(1/5)+sin(pi/3*x);              % Profil-funksjonen
k=0.067;
F=@(y) (.15-k*sqrt(y))/(pi*(p(y))^2);

% Startkravet
x0=0;
y0=0;
```

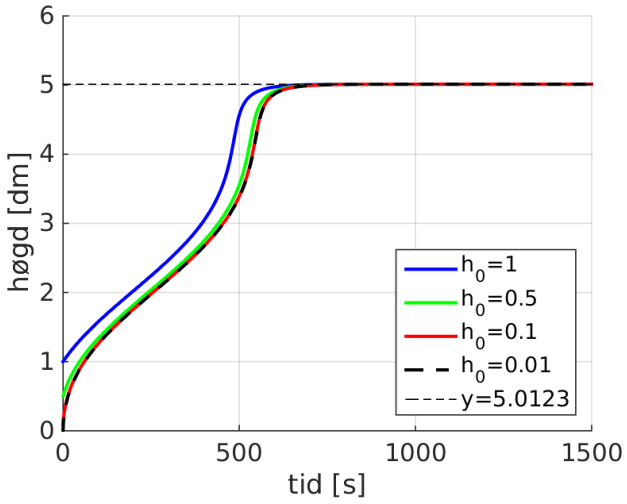
Vidare er det ikkje lenger ein god idé å bruke ei `while`-løkke med krav om at h (eller `y`) skal vere positiv for å avgjere kor lenge skriptet skal køyre; ei slik utrekning vil køyre veldig lenge ... I staden fikserar vi ein maksimal t-verdi. Om vi vél denne til å vere 1500 sekund, kan vi erstatte linje 30 med

```
while x<=1500
```

Men når vi skal køyre skriptet, får vi problem; `y`-verdiane blir berre `NaN` (not a number). I etterpåklokskapens lys er det kanskje lett å sjå grunnen: Om vi set startkravet $h = 0$ inn i differensiallikninga over, får vi null i nemnaren på høgresida; høgresida går mot uendeleg når $h \to 0^+$. I vår modell er jo for så vidt dette rett; sidan arealet vatnet skal dekke, forsvinn når høgda er null, $p(0) = 0$, vil høgda stige "uendeleg fort" akkurat i det vi begynnar å fylle vasen. Men numerisk er dette problematisk.

Vi kan forsøke å omgå problemet ved å erstatte $h_0 = 0$ med ein positiv men liten h_0 og sjå om oppførselen blir den same om vi lar h_0 bli mindre og mindre – utan å heilt bli null. I figuren under illustrerar vi at dette ser ut til å fungere. Der har vi plotta $h(t)$ for ulike verdiar av h_0. I alle tilfella ser vi at h ser ut til å nærme seg verdien vi fann i deloppgåve e)[3]. Dette gir oss grunn til å tru at det same vil skje i grensa $h_0 \to 0^+$. Ein annan sterk indikasjon på at dette er rett, er at det stemmer med sunn fornuft.

[3] Merk at vi ikkje eigentleg brukte noko startkrav i resonnementet vårt i deloppgåve e); svaret er uavhengig av $h(0)$.



g) I vårt tilfelle er integralet vi skal rekne ut for å finne arealet dette:

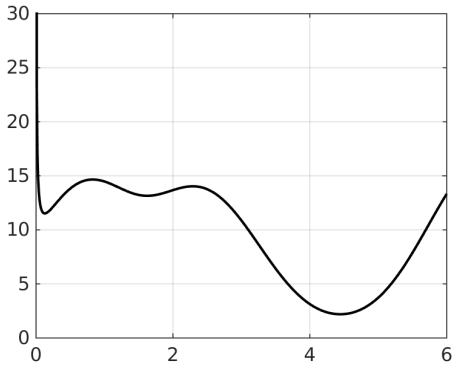
$$A = 2\pi \int_0^6 p(x)\sqrt{1 + [p'(x)]^2}\, dx \quad .$$

Vi treng den deriverte av profil-funksjonen:

$$p'(x) = \frac{1}{5}x^{1/5-1} + \cos\left(\frac{\pi}{3}x\right) \cdot \frac{\pi}{3} = \frac{1}{5x^{4/5}} + \frac{\pi}{3}\cos\left(\frac{\pi}{3}x\right) \quad .$$

Vi ser her at vi igjen får problem for $h = 0$: Vi kan ikkje sette inn $x = 0$ i integranden; $p'(x)$ går mot uendeleg når $x \to 0^+$.

Dette er illustrert i figuren under; der har vi plotta sjølve funksjonen som skal integrerast. Sidan denne går mot uendeleg når x går mot null, kan vi ikkje utan vidare sette inn 0 som nedre grense. Og om vi vel ein verdi like over null, er det grunn til å tru at svaret vil vere veldig kjensleg for kva verdi dette er.



Men arealet av overflata av vasen må jo vere veldefinert; vi kan jo ikkje la

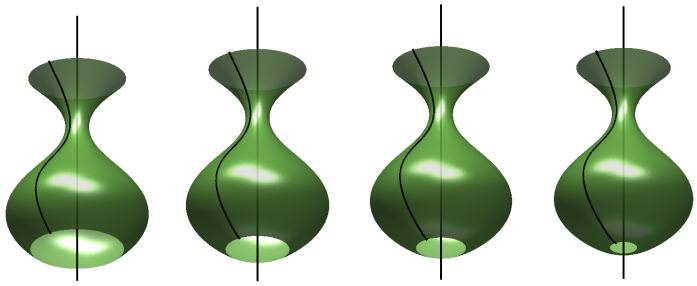
noko som ser ut til å vere ein "matematisk spissfindighet" vere til hinder for at vasen skal ha eit overflateareal.

Løysinga på problemet er å tolke integralet over som ein grenseverdi:

$$A = \lim_{a \to 0^+} 2\pi \int_a^6 p(x)\sqrt{1 + [p'(x)]^2}\, dx \quad .$$

Dette er eit eksempel på det ein kallar eit *ueigentleg integral*.

Arealet vi finn for ein endeleg verdi av a er illustrert i figuren under. Vi må tette holet ved å la a gå mot null. I figuren har vi brukt $a = 0.5, 0.2, 0.1$ og 0.01.



Numerisk kan vi forsøke å finne arealet på tilsvarande måte som i den førre deloppgåva: Vi estimerar A for stadig lågare verdi av den nedre integrasjonsgrensa, a, og ser kva arealet A nærmar seg. Dette er implementert i dette skriptet:

```
% Skript som estimerar volumet av ein vase
% ved å forsøke å ekstrapolere nedre integrasjons-
% grense til null.
% Skriptet brukar funksjonsfila RotasjonsArealFunk.m,
% som estimerar sjølve arealet ved trapesmetoden.
% I skriptet heiter nedre integrasjonsgrense a.

% Fikserar presisjonen vi ønsker i trapesintegralet
Pres=1e-3;

% Intierar variablane
A=1;                    % Areal-estimat
Aold=0;                 % Gamalt areal-estimat
N=10;                   % Talet på delintervall i estimat
a=1;                    % Nedre integrasjonsgrense
n=1;                    % Indeks for vektorane

while a>1e-5            % Set nedre grense for a
  % Itererar til integralet er nøyaktig nok
  while abs(Aold-A)>Pres
```

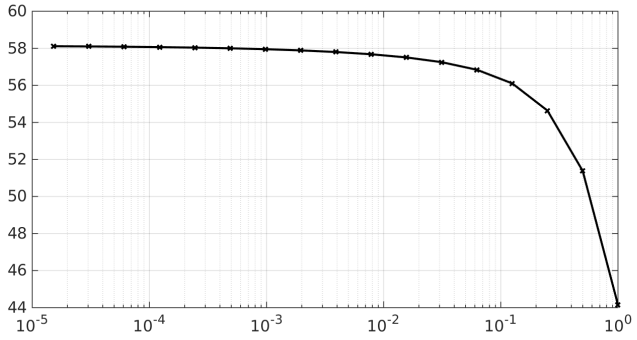
```
    Aold=A;              % Oppdaterar Aold
    N=2*N;               % Doblar oppdelinga
    A=RotasjonsArealFunk(a,N);  % Estimerar areal
  end
  % Tilordnar svara til vektorar
  aVektor(n)=a;
  Avektor(n)=A;
  a=a/2;                 % Halverar a
  n=n+1;                 % Oppdaterar indeks
  Aold=0;                % "Reset" Aold
  N=10;                  % "Reset" N
end

% Plottar resultatet med semi-logaritmisk akse
semilogx(aVektor,Avektor,'kx-','linewidth',2)
grid on
```

Skriptet kallar funksjonsfila "RotasjonsArealFunk.m", som estimerar integralet. Denne tek nedre integrasjonsgrense a og oppdelinga `N` som input. I denne funksjonsfila er integranden tilordna slik:

```
% Funksjonen som skal roterast
f=@(x) x^(1/5)+sin(pi/3*x);
% Den deriverte av funksjonen
fDeriv=@(x) 1/5*x^(-4/5)+pi/3*cos(pi/3*x);
% Integrand
Int=@(x) 2*pi*f(x)*sqrt(1+(fDeriv(x))^2);
```

Ellers er resten svært likt trapesmetodeimplementeringa vi har sett ein del gonger no.

Når vi køyrer skriptet, får vi resultatet vist i figuren under. Arealet ser ut



til å vere drygt 58 dm^2.

Men vi kan klare betre. Om vi tar ein kikk på figuren med hola over, ser vi at det som manglar, liknar veldig på ei sirkelskive. Om vi erstattar holet

med ei sirkelskive med radius $p(a)$, kan vi tilnærme arealet slik:

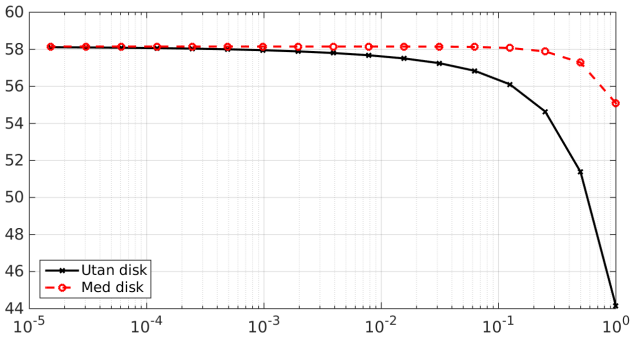
$$A \approx \pi \left[p(a)\right]^2 + 2\pi \int_a^6 p(x)\sqrt{1+\left[p'(x)\right]^2}\,dx \quad .$$

Dette er sjølvsagt frameleis feil. Men feilen blir mindre og mindre når a blir det. Og viktigare: Feilen er langt mindre enn om vi ikkje tok med "tallerkenen" i det heile.

Dette har vi implementert ved å legge til desse linjene til slutt i funksjonsfila RotasjonsArealFunk.m:

```
% Reknar ut arealet av disken vi har utelatt
disk=pi*(f(a))^2;
T=T+disk;
```

Med denne implementeringa frå vi den raude kurva i plottet under. Vi ser



at denne konvergerar mykje raskare enn den versjonen der vi ikkje gjorde noko anna forsøk på tette hole enn å la a bli liten. Med to desimalar får vi arealet $\underline{58.15\ \mathrm{dm}^2}$.