# TDAT3024 – Matteøving 3

Tore Bergebakken, Jon Åby Bergquist, Kristoffer Vanebo

**NB**: CP-oppgavene står bakenfor de tekstlige oppgavene siden de er eksportert fra Live Editor i Matlab.

## 8.1.4

Vi skal undersøke om bakoverdifferansemetoden er stabil for varmelikninger med $c < 0$. Antar at $c$ er konstanten som ellers kalles $D$, i en generell varmelikning

$$u_{tt} = cu_{xx}$$

Da er $\sigma = \frac{ck}{h^2} < 0$ siden både $k$ og $h$ er positive.

Fra Von Neumann-stabilitetsanalysen i læreboka vet vi at metodens kjerne, iterasjonen $w_j = A^{-1}w_{j-1} + b$, er stabil for $|1 + 2\sigma(1 - \cos x)| > 1 \Rightarrow \sigma > 0$ (8.17). Siden vi nå fant at for $c < 0$ er $\sigma < 0$, kan ikke bakoverdifferansemetoden være betingelsesløst stabil for varmelikninger med $c < 0$, Q.E.D.

## 8.2.1b

Skal vise at $u(x, t) = e^{-x-2t}$ er en løsning på

$$u_{tt} = 4u_{xx}$$

med initialbetingelsene

$$\begin{cases} u(x,0) = e^{-x} \text{ for } 0 \leq x \leq 1 \\ ut(x,0) = -2e^{-x} \text{ for } 0 \leq x \leq 1 \\ u(0,t) = e^{-2t} \text{ for } 0 \leq t \leq 1 \\ u(1,t) = e^{-1-2t} \text{ for } 0 \leq t \leq 1 \end{cases}$$

Viser først at løsningen passer i likninga:

$$
\begin{aligned}
u &= e^{-x-2t} \\
u_x &= -e^{-x-2t} \\
u_{xx} &= e^{-x-2t} \\
u_t &= -2e^{-x-2t} \\
u_{tt} &= 4e^{-x-2t} \\
\Rightarrow u_{tt} &= 4u_{xx}
\end{aligned}
$$

Så tar vi for oss betingelsene:

$$
\begin{aligned}
u(x,0) &= e^{-x-2\cdot 0} = e^{-x} \\
u_t(x,0) &= -2e^{-x-2\cdot 0} = -2e^{-x} \\
u(0,t) &= e^{-0-2t} = e^{-2t} \\
u(1,t) &= e^{-1-2t}
\end{aligned}
$$

Betingelsene er oppfylt, så $u$ er en gyldig løsning av likningen.

## 8.2.3

Skal vise at $u_1(x,t) = \sin \alpha x \cos c\alpha t$ og $u_2(x,t) = e^{x+ct}$ er løsninger av bølgelikningen $u_{tt} = c^2 u_{xx}$.

### 8.2.3 – Løsning 1

Deriverer $u_1$:

$$
\begin{aligned}
u_1 &= \sin \alpha x \cos c\alpha t \\
u_{1t} &= -c\alpha \sin \alpha x \sin c\alpha t \\
u_{1tt} &= -c^2\alpha^2 \sin \alpha x \cos c\alpha t \\
u_{1x} &= \alpha \cos \alpha x \cos c\alpha t \\
u_{1xx} &= -\alpha^2 \sin \alpha x \cos c\alpha t
\end{aligned}
$$

Setter inn i likningens to sider:

$$\begin{aligned}
u_{1tt} &= -c^2\alpha^2 \sin \alpha x \cos c\alpha t \\
c^2 u_{1xx} &= -\alpha^2 \sin \alpha x \cos c\alpha t \\
u_{1tt} &= c^2 u_{1xx} \\
-c^2\alpha^2 \sin \alpha x \cos c\alpha t &= -c^2\alpha^2 \sin \alpha x \cos c\alpha t
\end{aligned}$$

Har da vist at $u_1$ er en løsning på bølgelikningen.

### 8.2.3 – Løsning 2

Deriverer $u_2$:

$$\begin{aligned}
u_2 &= e^{x+ct} \\
u_{2t} &= ce^{x+ct} \\
u_{2tt} &= c^2 e^{x+ct} \\
u_{2x} &= e^{x+ct} \\
u_{2xx} &= e^{x+ct}
\end{aligned}$$

Setter inn i likningens to sider:

$$\begin{aligned}
u_{2tt} &= c^2 e^{x+ct} \\
c^2 u_{2xx} &= c^2 e^{x+ct} \\
u_{2tt} &= c^2 u_{2xx} \\
c^2 e^{x+ct} &= c^2 e^{x+ct}
\end{aligned}$$

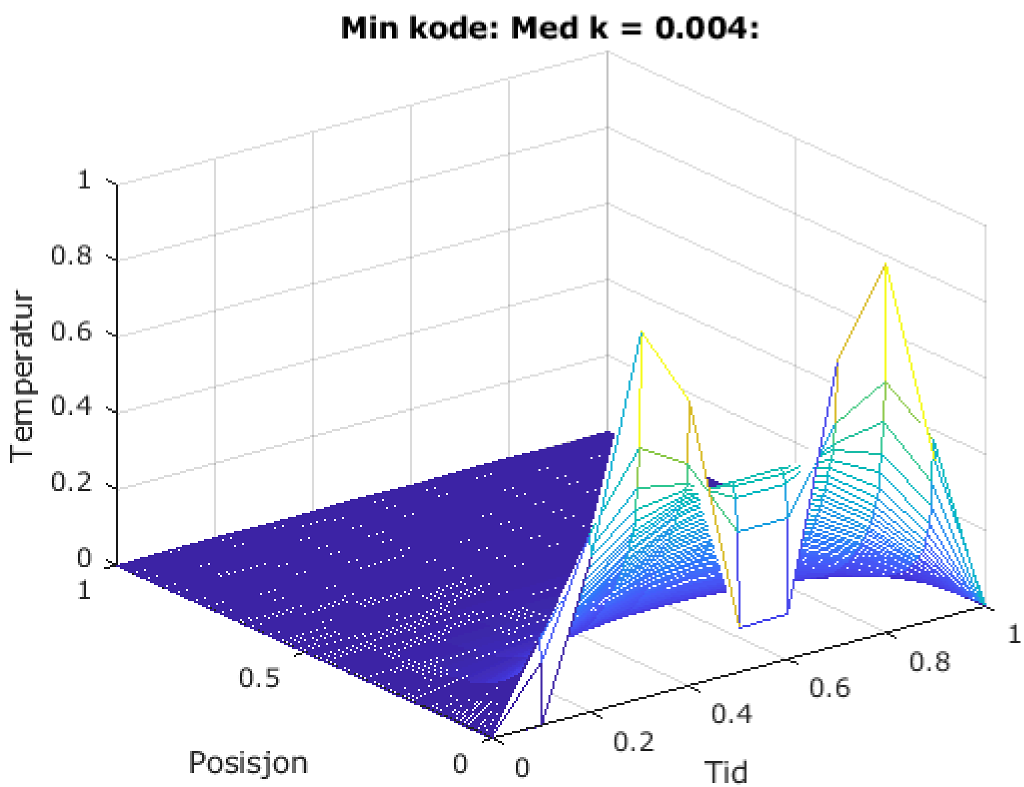
Har da vist at $u_2$ er en løsning på bølgelikningen.

# Forward Difference Method

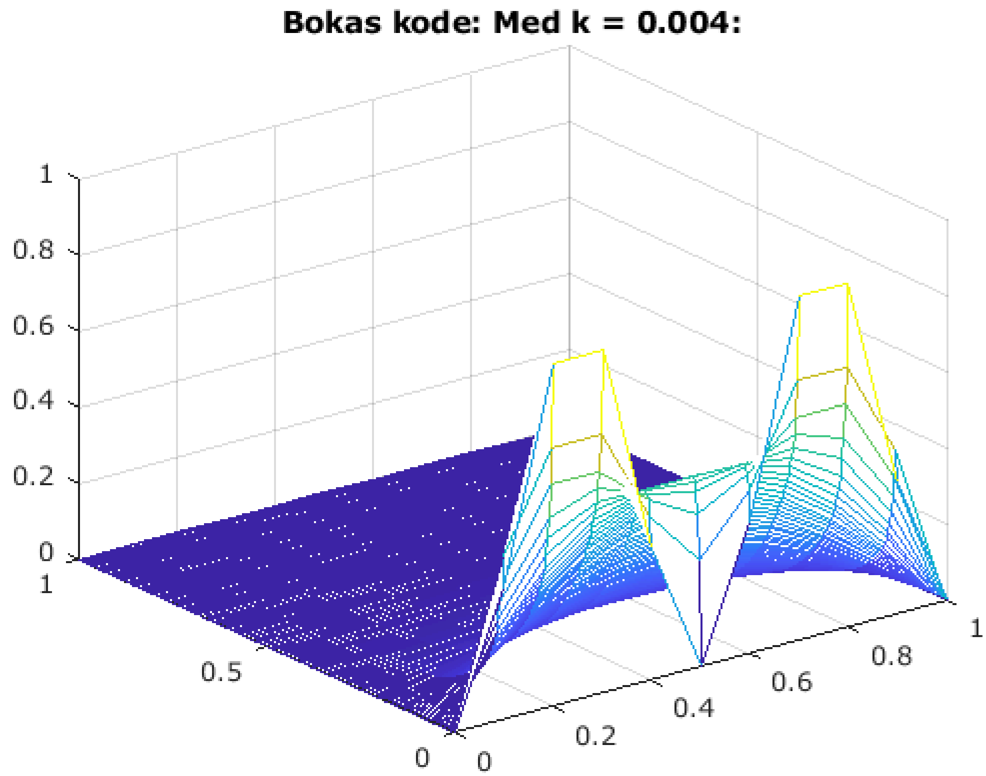
## En liten test

Tester først mot eksempelet i boka:

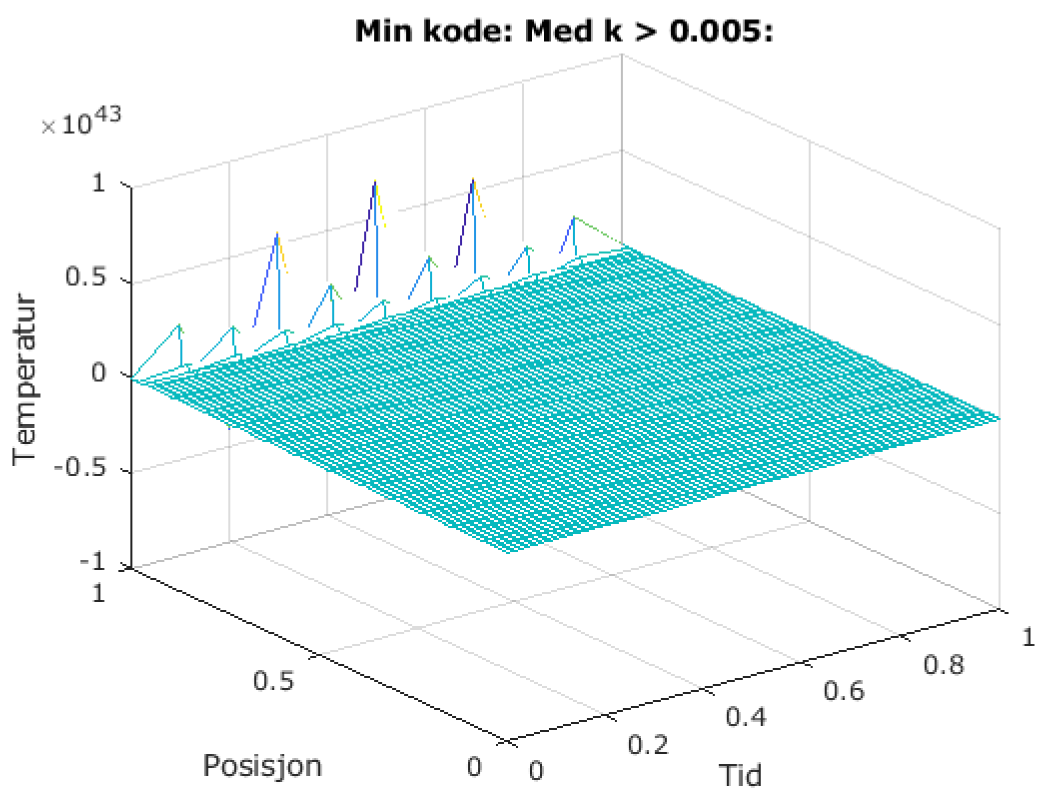
```
D = 1; M = 10; N = 250;
f = @(x) sin(2*pi*x).^2;
l = @(t) 0*t;
r = @(t) 0*t;
heatfdm(0, 1, 1, M, N, D, f, l, r);
title("Min kode: Med k = 0.004:");
```



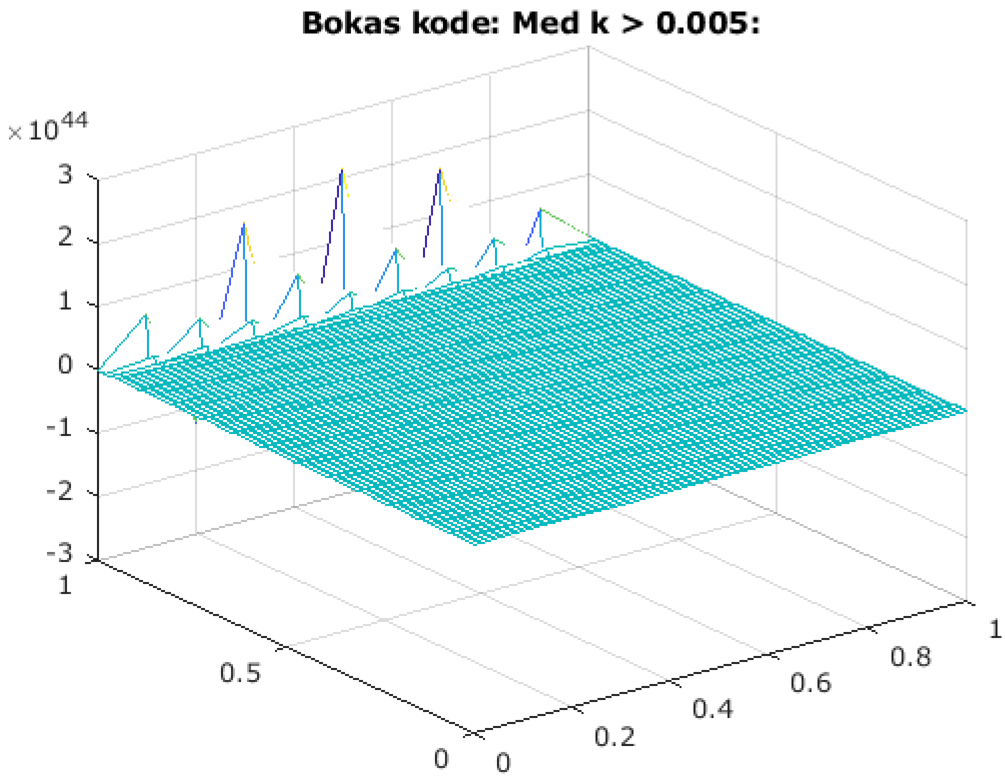

```
boka(0, 1, 0, 1, M, N, 1, f, l, r);
title("Bokas kode: Med k = 0.004:");
```




```
heatfdm(0, 1, 1, M, (1/0.01)-1, D, f, l, r);
title("Min kode: Med k > 0.005:");
```



```
boka(0, 1, 0, 1, M, (1/0.01)-1, 1, f, 1, r);
title("Bokas kode: Med k > 0.005:");
```




## CP1 – fellestrekk

Skal løse $u_t = 2u_{xx}$ i begge deloppgavene – altså er $D = 2$.

```
D = 2;
```

Beregningene skjer med h = 0.1 og k = 0.002, altså M = 10 og N = 500.

```
M = 1/0.1;
N = 1/0.002;
```

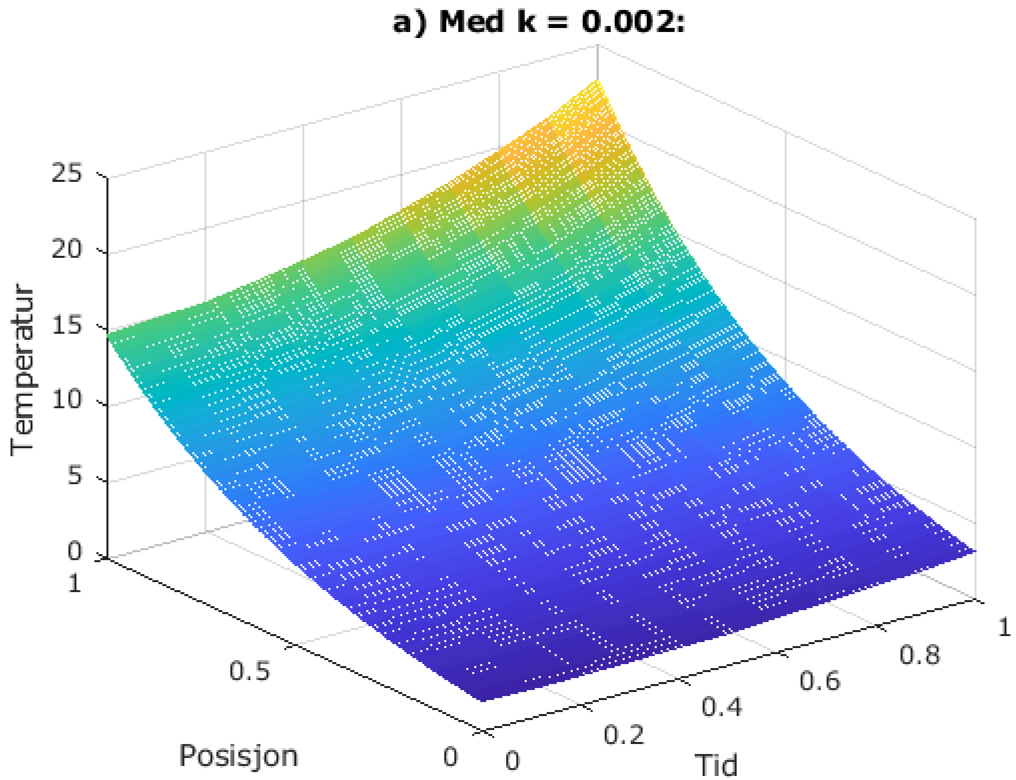
Vi skal sammenligne med k > 0.003:

```
N2 = round(1/0.003) - 1;
```

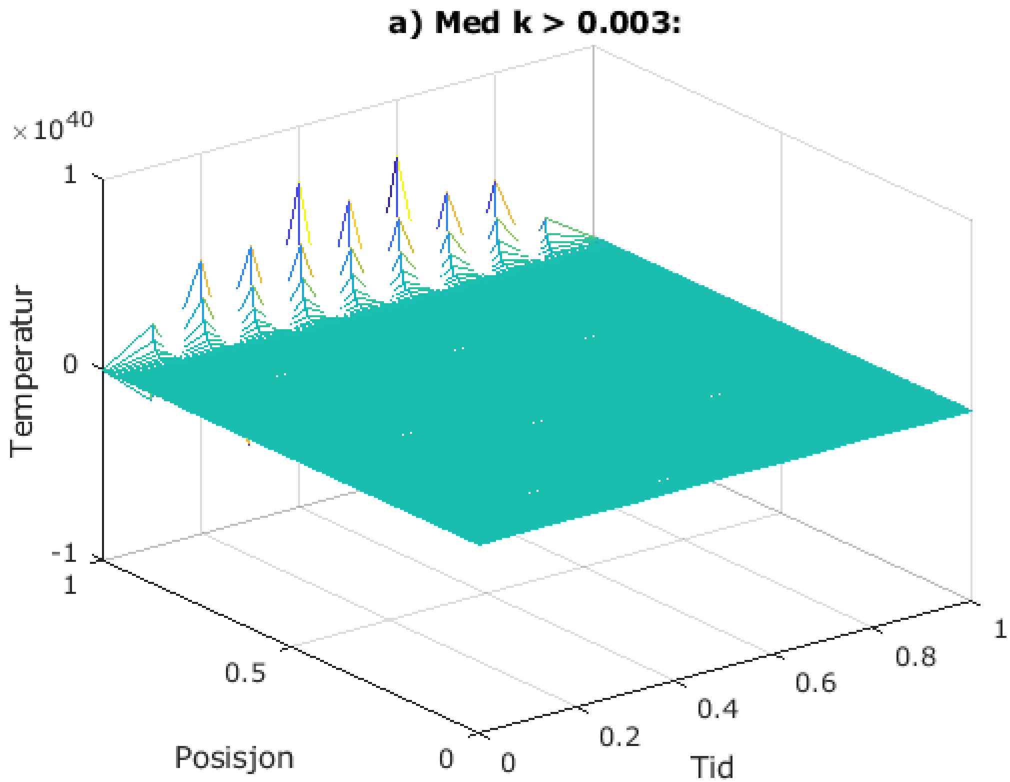
## CP1a

$u(x,0) = 2\cosh x$ for $0 \leq x \leq 1$
$u(0,t) = 2e^{2t}$ for $0 \leq t \leq 1$
$u(1,t) = (e^2+1)e^{2t-1}$ for $0 \leq t \leq 1$

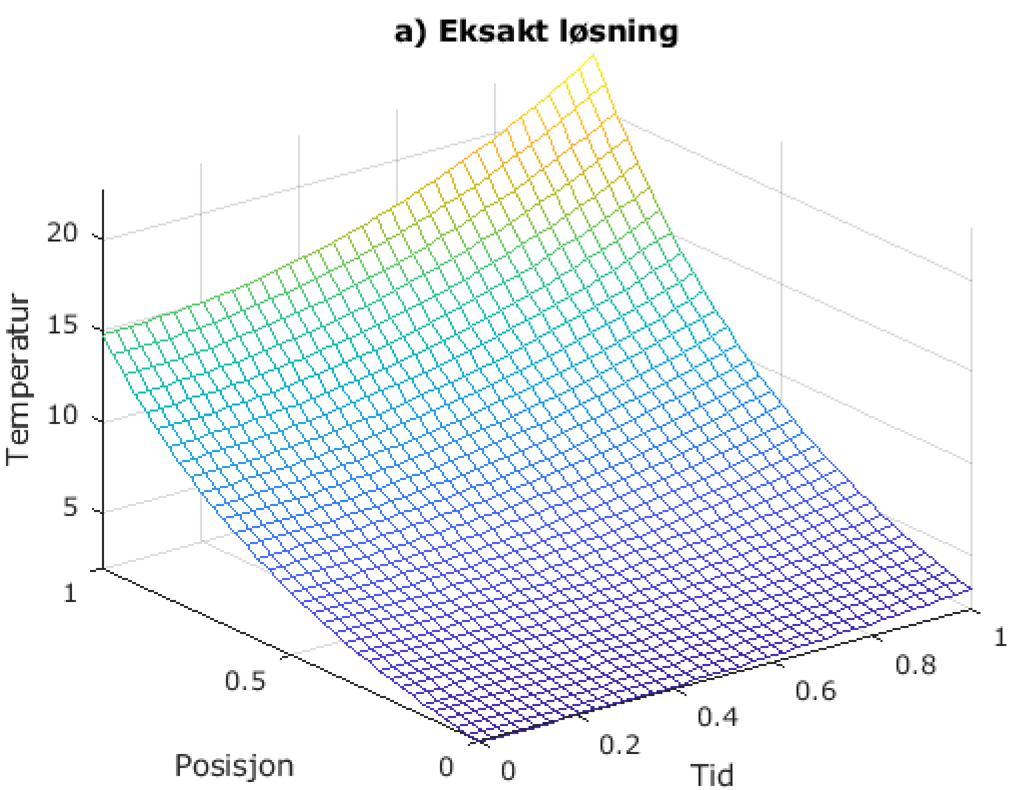
```
f = @(x) 2*cosh(x);
l = @(t) 2*exp(2*t);
r = @(t) (exp(2)+1)*exp(2*t-1);
heatfdm(0, 1, 1, M, N, D, f, l, r);
title("a) Med k = 0.002:");
```




```
heatfdm(0, 1, 1, M, N2, D, f, l, r);
title("a) Med k > 0.003:");
```



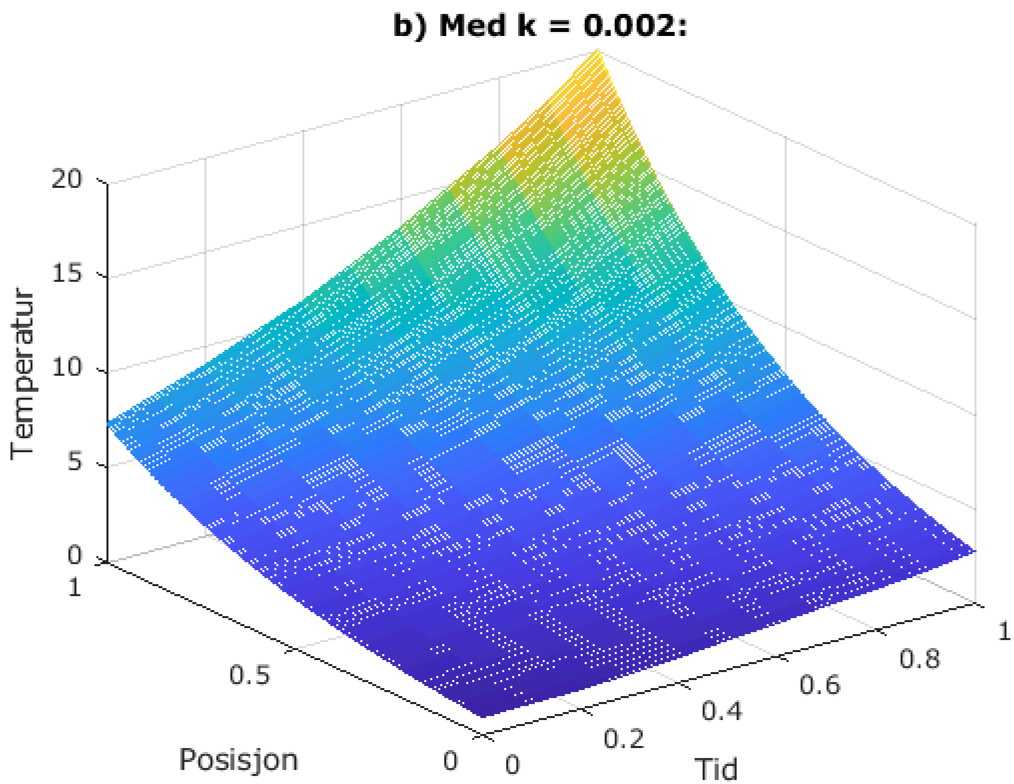

```
plotExact(@(x,t) exp(2*t+x) + exp(2*t-x));
title("a) Eksakt løsning");
```



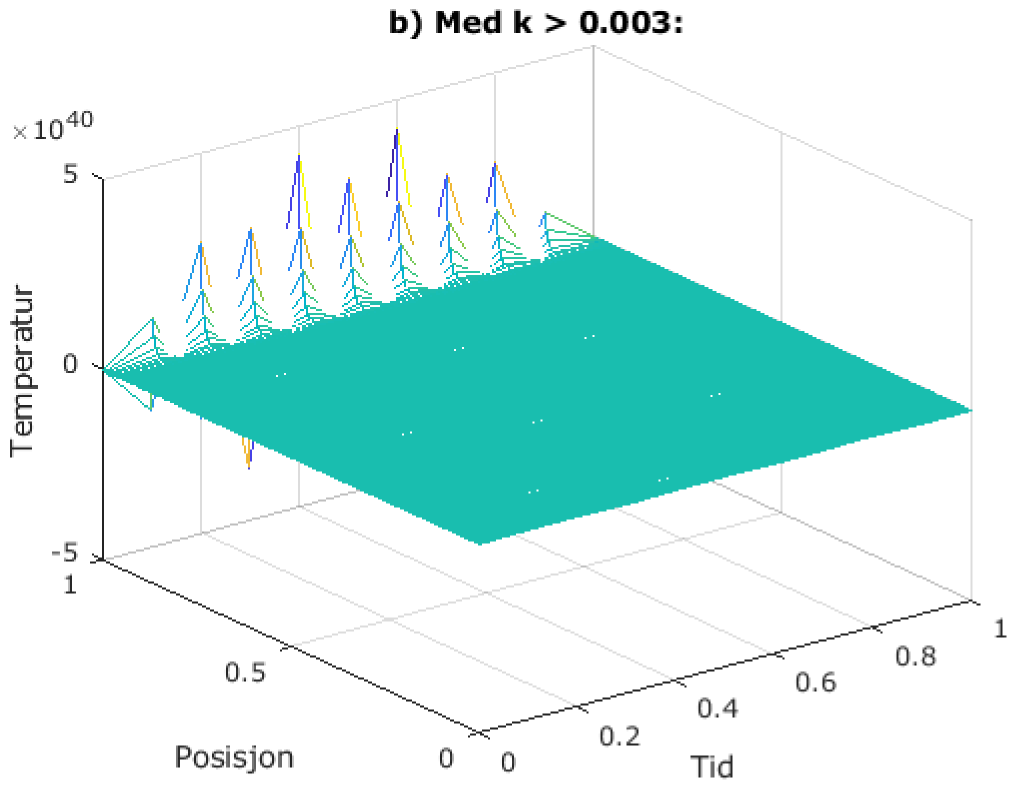
## CP1b

$u(x, 0) = e^x$ for $0 \leq x \leq 1$
$u(0, t) = e^{2t}$ for $0 \leq t \leq 1$
$u(1, t) = e^{2t+1}$ for $0 \leq t \leq 1$

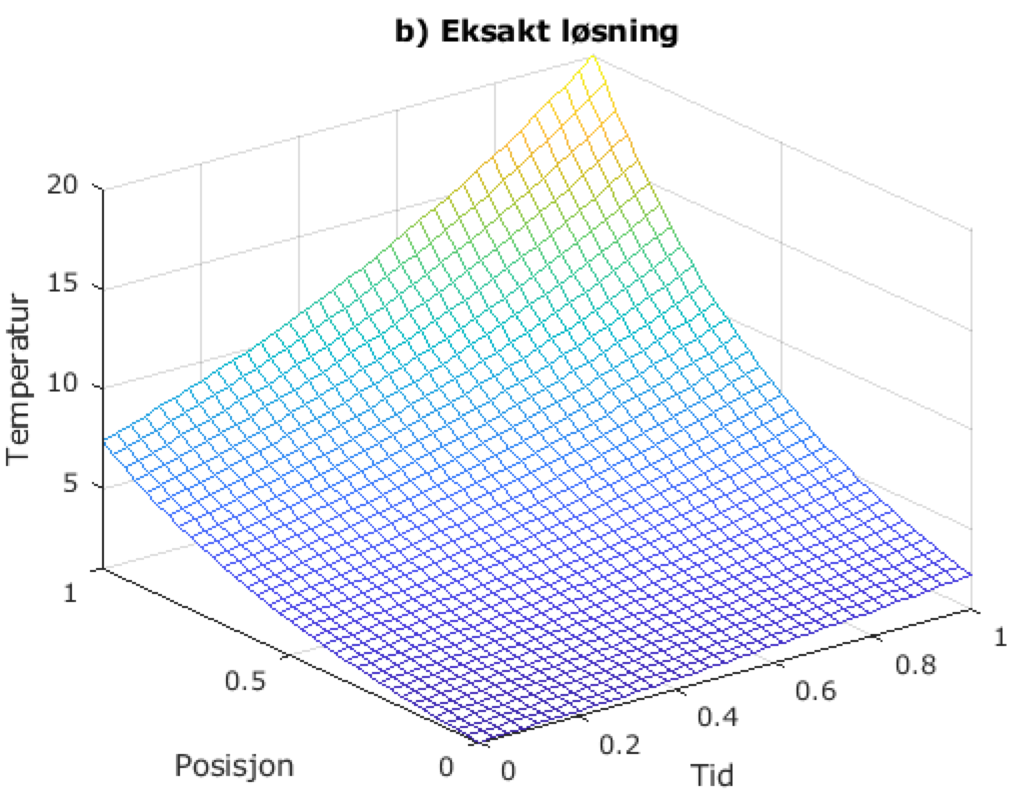
```
f = @(x) exp(x);
l = @(t) exp(2*t);
r = @(t) exp(2*t+1);
heatfdm(0, 1, 1, M, N, D, f, l, r);
title("b) Med k = 0.002:");
```




```
heatfdm(0, 1, 1, M, N2, D, f, l, r);
title("b) Med k > 0.003:");
```




```
plotExact(@(x,t) exp(2*t+x));
title("b) Eksakt løsning");
```



Funksjonene:

```
function w=heatfdm(a, b, T, M, N, D, f, l, r)
%HEATBDM Forward Difference Method for the heat equation
%   Løser  u_t = D u_xx  numerisk
%   a og b er grensebetingelsene u(a,t)=l(t) og u(b,t)=r(t)
%   T utgjør høyre grense i tidsintervallet [0,T]
%   f(x) er initialbetingelsen for u(x,0)
%   D er diffusjonskoeffisienten (i den originale likningen)
%   M og N er antall samplinger langs x- og t-aksen
%   NB: f, l og r er funksjoner

    % Beregner steglengder og sigma
    h = (b-a)/M;
    k = T/N;
    sigma = D*k / (h*h);
    m = M-1;

    % Setter opp aksene
    x = linspace(a,b,M)';
    t = linspace(0,T,N+1);

    % Lager diagonalmatrisen A
    e = ones(m, 1);
    A = spdiags([sigma*e, (1-2*sigma)*e, sigma*e], [-1,0,1], m, m);

    w = zeros(m,N);
    %w(1,1:N) = l(t); % uh oh - l er ikke en funksjon i x
    %w(N,1:N) = r(t);
    w(:,1) = f(x(1:m));
    for i = 1:N
        s = sigma*[l(i*k) zeros(1,m-2) r(i*k)]';
        w(:,i+1) = (A*w(:,i) + s);
    end

    w = [l(t); w; r(t)];

    [X,T] = meshgrid(linspace(a,b,M+1)',t);
    mesh(X,T,w')
    xlabel('Tid')
    ylabel('Posisjon')
    zlabel('Temperatur')
end

function plotExact(f)
    fmesh(f, [0 1])
    xlabel('Tid')
    ylabel('Posisjon')
    zlabel('Temperatur')
end

function w=boka(xl,xr,yb,yt,M,N,D,f,l,r)
```

```
% Dette er kode fra boka, limt inn for å ha noe å sammenligne med
% etter å ha gjort et forsøk.
%
% Program 8.1 Forward difference method for heat equation
% input: space interval [xl,xr], time interval [yb,yt],
%number of space steps M, number of time steps N
% output: solution w
% Example usage: w=heatfd(0,1,0,1,10,250)
% diffusion coefficient
h=(xr-xl)/M; k=(yt-yb)/N; m=M-1; n=N;
sigma=D*k/(h*h);
a=diag(1-2*sigma*ones(m,1))+diag(sigma*ones(m-1,1),1);
a=a+diag(sigma*ones(m-1,1),-1);
% define matrix a
lside=l(yb+(0:n)*k); rside=r(yb+(0:n)*k);
w(:,1)=f(xl+(1:m)*h)';
% initial conditions
for j=1:n
w(:,j+1)=a*w(:,j)+sigma*[lside(j);zeros(m-2,1);rside(j)];
end
w=[lside;w;rside];
% attach boundary conds
x=(0:m+1)*h;t=(0:n)*k;
mesh(x,t,w')
% 3-D plot of solution w
%view(60,30);axis([xl xr yb yt -1 1])
end
```

# Finite Difference Method

## CP2a

Betingelsene fra 8.2.2a)

$u_{tt} = 4u_{xx}$
$u(x,0) = 0 \text{ for } 0 \le x \le 1$
$u_t(x,0) = 2\pi \sin \pi x \text{ for } 0 \le x \le 1$
$u(0,t) = 0 \text{ for } 0 \le t \le 1$
$u(1,t) = 0 \text{ for } 0 \le t \le 1$

Dette gir at $c = \sqrt{4} = 2$, $g(x) = 2\pi \sin \pi x$, $f(x) = l(x) = r(x) = 0$

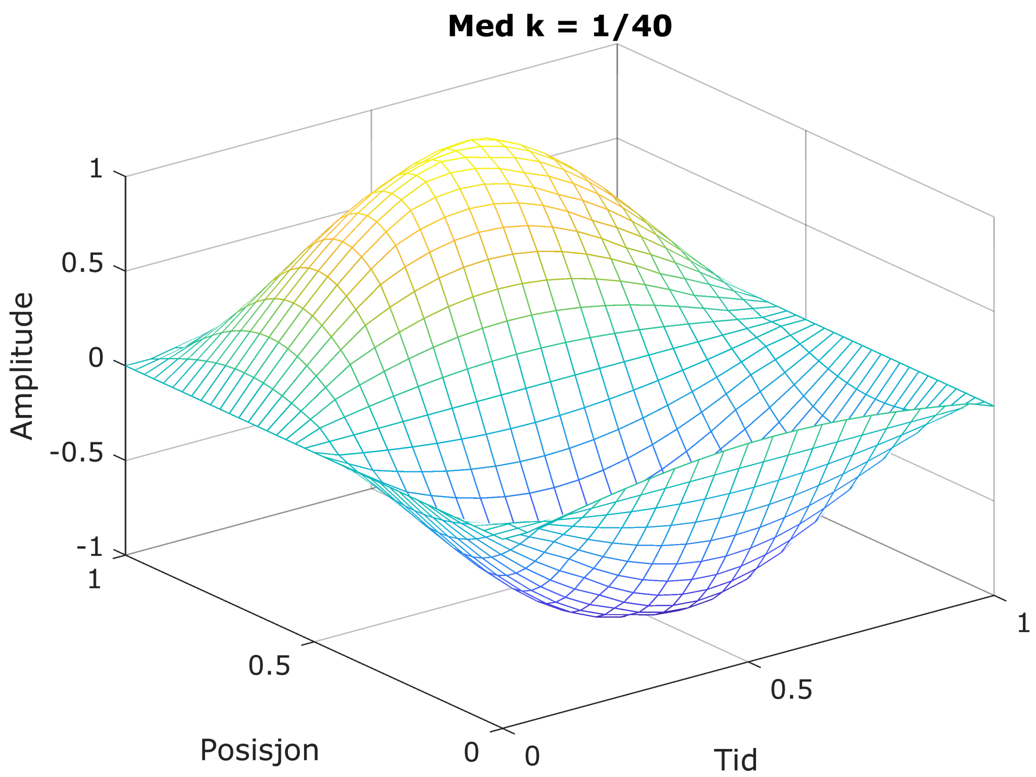
```
c = 2;
f = @(x) x*0;
g = @(x) 2*pi*sin(pi*x); % negativ versjon gir korrekt w...
l = f;
r = f;
```

Skal løses med $h = 0.05$ og en *k* som tilfredsstiller $\sigma = \frac{ck}{h} \le 1$ – altså med $\frac{2 \cdot k}{0.05} = 2 \cdot 20 \cdot k = 40k \le 1 \Rightarrow k \le \frac{1}{40}$

```
h = 0.05;
M = 1/h;
k = 1/40;
N = 1/k;
```

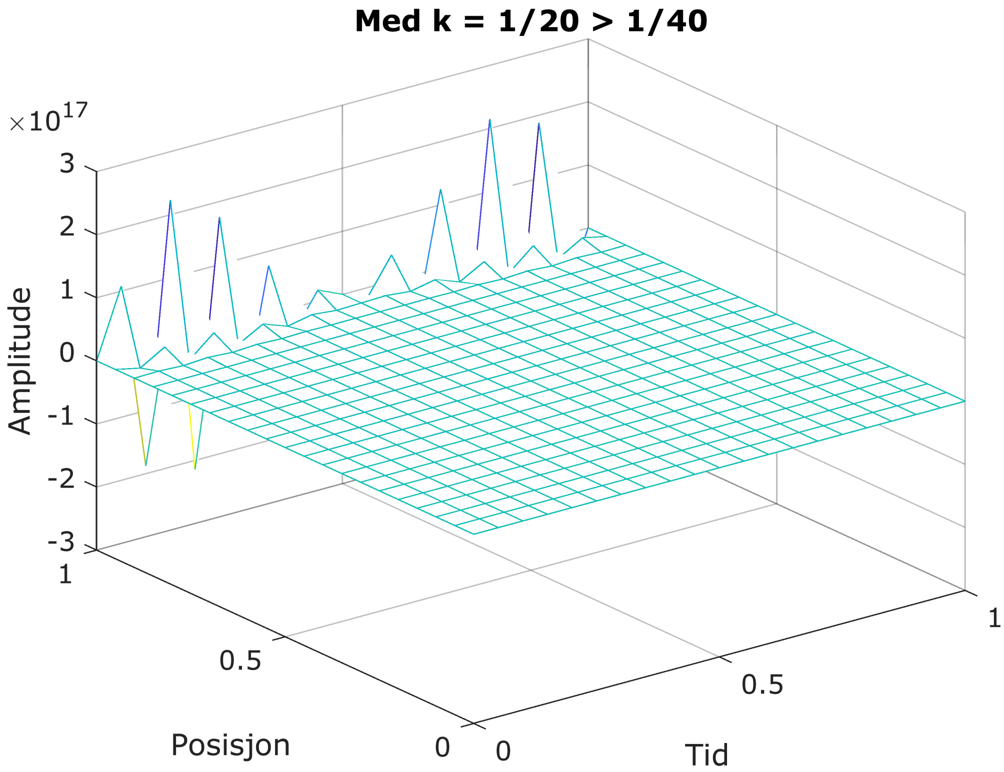
Så, med $k = 1/40$:

```
finiteDfference(0, 1, 1, M, N, c, f, g, l, r);
title("Med k = 1/40")
```



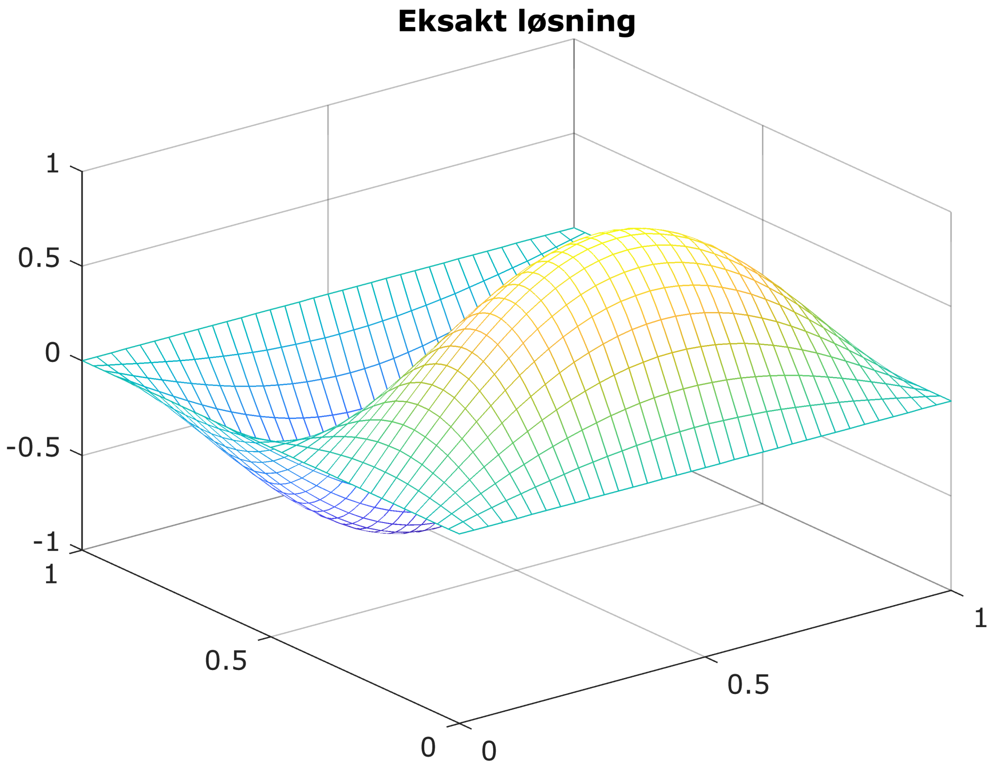

Med $k = 1/20 > 1/40$ ser vi at det er litt dårligere stilt:

```
finiteDfference(0, 1, 1, M, 20, c, f, g, l, r);
title("Med k = 1/20 > 1/40")
```




Tester med den eksakte løsningen $u(x,t) = \sin \pi x \sin 2\pi t$:

```
fmesh(@(x,t) sin(pi.*x) .* sin(2*pi.*t), [0 1])
title("Eksakt løsning");
```




Ser at min løsning er en flippet versjon av den eksakte – antar det skyldes en fortegnsfeil, men ser virkelig ikke hvor den feilen kan skje.

Om enten initalbetingelsen $g(x)$ eller $k \cdot g(x)$-leddet i initialvektoren settes negative, blir grafene like. Det har vi ikke grunnlag for å gjøre.

```
function finiteDfference(a, b, T, M, N, c, f, g, l, r)
%FINITEDIFFERENCE Finite Difference Method for the wave equation
%   Løser  u_tt = c² u_xx  numerisk
%   a og b er grensebetingelsene u(a,t)=l(t) og u(b,t)=r(t)
%   T utgjør øvre grense i tidsintervallet [0,T]
%   f(x) er initialbetingelsen for u(x,a)
%   g(x) er initialbetingelsen for u_t(x,a)
%   c er en bølgefartparameteren til den originale likningen
%   M og N er antall samplinger langs x- og t-aksen
%   NB: f, g, l og r er funksjoner

    % Beregner steglengder og sigma
    h = (b-a)/M;
    k = T/N;
    sigma = c*k / h;
    m = M-1;

    % Setter opp aksene
    x = linspace(a,b,M+1)'; % avstand (b-a)/(M+1-1) = h
    t = linspace(0,T,N+1);  % avstand T/(N+1-1) = k
```

```
    % Lager diagonalmatrisen A
    e = ones(m, 1);
    A = spdiags([sigma*sigma*e, (2-2*sigma*sigma)*e, sigma*sigma*e], [-1,0,1], m, m);

    % Lager kolonne 1 i w basert på initialbetingelsene
    w = zeros(m,N);
    s = 1/2 * sigma^2 * [l(0); zeros(m-2,1); r(0)];
    w(:,1) = 1/2 * A * f(x(1:m)) + k*g(x(1:m)) + s;
    % om vi isteden tar - k*g(x) her, får vi rett svar
    % men det har vi ikke grunnlag for

    % Regner ut resten av w
    for j = 2:N
        s = sigma^2 * [l(t(j)); zeros(m-2,1); r(t(j))];
        w(:,j+1) = A * w(:,j) - w(:,j-1) + s;
    end

    % Sleng grensene på w ("oppe og nede" == venstre og høyre i normalt koordinatsystem
    w = [l(t); w; r(t)];

    % Plott w
    [X,T] = meshgrid(x,t);
    mesh(X,T,w')
    xlabel('Tid')
    ylabel('Posisjon')
    zlabel('Amplitude')
end
```