UNIVERSITETET I OSLO

Det matematisk-naturvitenskapelige fakultet

Eksamen i:	MAT-INF2360 — Anvendelser av lineær algebra
Eksamensdag:	Torsdag 16. juni 2016
Tid for eksamen:	14.30 – 18.30
Oppgavesettet er på 7 sider.	
Vedlegg:	Ingen
Tillatte hjelpemidler:	Godkjent kalkulator

Kontroller at oppgavesettet er komplett før
du begynner å besvare spørsmålene.

Eksamenssettet inneholder 10 deloppgaver, og alle deloppgaver teller like mye.

Oppgave 1 Fourierrekker

Vi lar f være funksjonen med periode T definert ved

$$f(t) = \cos(4\pi t/T) + \sin(10\pi t/T + \pi/2).$$

Skriv opp den komplekse Fourierrekka av orden $N = 2$ (d.v.s. $f_2(t)$). Finn også en M slik at $f_N(t) = f(t)$ for alle $N \geq M$.
Løsningsforslag: Det er her ikke nødvendig å regne ut Fourierintegralene $\frac{1}{T}\int_0^T f(t)e^{-2\pi int/T}dt$ (selv om heller ikke dette gir mye regning), siden

$$\begin{aligned}&\cos(4\pi t/T) + \sin(10\pi t/T + \pi/2)\\ &= \cos(4\pi t/T) + \cos(10\pi t/T)\\ &= \frac{1}{2}(e^{2\pi i2t/T} + e^{-2\pi i2t/T}) + \frac{1}{2}(e^{2\pi i5t/T} + e^{-2\pi i5t/T}).\end{aligned}$$

Siden Fourierrekka til f har formen $f_N(t) = \sum_{n=-N}^{N} y_n e^{2\pi int/T}$, så ser vi at $y_2 = y_5 = y_{-2} = y_{-5} = 1/2$, og at alle andre y_n er 0. Dette betyr spesielt at

$$f_2(t) = \frac{1}{2}e^{2\pi i2t/T} + \frac{1}{2}e^{-2\pi i2t/T},$$

og at $f_N(t) = f(t)$ for alle $N \geq 5$.

Oppgave 2 DFT

Vi definerer vektorene $\boldsymbol{x}_1 \in \mathbb{R}^N$ og $\boldsymbol{x}_2 \in \mathbb{R}^N$ ved

$$\boldsymbol{x}_1 = (1, 1, 1, 1, ..., 1, 1) \qquad \boldsymbol{x}_2 = (1, -1, 1, -1, ..., 1, -1)$$

($\boldsymbol{x}_1$ er altså vektoren der alle komponenter er 1, $\boldsymbol{x}_2$ er vektoren med komponenter $(-1)^k$). Du kan anta at N er et partall. Regn ut $\mathrm{DFT}_N\boldsymbol{x}_1$

(Fortsettes på side 2.)

og $\text{DFT}_N \boldsymbol{x}_2$.
Løsningsforslag: Vi har at vektorene kan skrives $\boldsymbol{x}_1 = \sqrt{N}\phi_0$, og $\boldsymbol{x}_2 = \sqrt{N}\phi_{N/2}$. Siden $F_N(\phi_n) = \boldsymbol{e}_n$ for alle n, så har vi at

$$\text{DFT}_N \boldsymbol{x}_1 = \sqrt{N} F_N \boldsymbol{x}_1 = \sqrt{N} F_N(\sqrt{N}\phi_0) = N\boldsymbol{e}_0$$
$$\text{DFT}_N \boldsymbol{x}_2 = \sqrt{N} F_N \boldsymbol{x}_2 = \sqrt{N} F_N(\sqrt{N}\phi_{N/2}) = N\boldsymbol{e}_{N/2}.$$

Oppgave 3 Filtre

La S være filteret med kompakt filternotasjon $\frac{1}{16}(1, 4, \underline{6}, 4, 1)$.

3a

Skriv ned en 8×8 sirkulant Toeplitz matrise som svarer til å anvende S på vektorer som er periodiske med periode 8. Regn også ut og plott (den kontinuerlige) frekvensresponsen. Er filteret et lavpass- eller høypassfilter?
Løsningsforslag: Matrisen blir

$$\frac{1}{16}\begin{pmatrix} 6 & 4 & 1 & 0 & 0 & 0 & 1 & 4 \\ 4 & 6 & 4 & 1 & 0 & 0 & 0 & 1 \\ 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 \\ 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 \\ 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 \\ 1 & 0 & 0 & 0 & 1 & 4 & 6 & 4 \\ 4 & 1 & 0 & 0 & 0 & 1 & 4 & 6 \end{pmatrix}$$

Frekvensresponsen blir

$$\begin{aligned} \lambda_S(\omega) &= \frac{1}{16}(e^{2i\omega} + 4e^{i\omega} + 6 + 4e^{-i\omega} + e^{-2i\omega}) \\ &= \left(\frac{1}{2}(e^{i\omega/2} + e^{-i\omega/2})\right)^4 = \cos^4(\omega/2), \end{aligned}$$

der vi kjente igjen verdiene fra Pascals trekant, og det er klart at dette er et lavpassfilter (evt. regn ut $\lambda_S(0) = 1$, $\lambda_S(\pi) = 0$).

3b

Regn ut $S\boldsymbol{x}_1$ og $S\boldsymbol{x}_2$, der $\boldsymbol{x}_1$, $\boldsymbol{x}_2$ er vektorene fra oppgave 2.
Løsningsforslag: Siden Fourierbasisvektorene er egenvektorer for filtre så har vi at $S\boldsymbol{x}_1 = \lambda_{S,0}\boldsymbol{x}_1$ og $S\boldsymbol{x}_2 = \lambda_{S,N/2}\boldsymbol{x}_2$. S er her et lavpassfilter slik at $\lambda_S(\pi) = 0$, og det er lett å sjekke at $\lambda_S(0) = 1$. Dermed får vi $\lambda_S(0) = \lambda_{S,0} = 1$ og $\lambda_{S,N/2} = \lambda_S(2\pi(N/2)/N = \lambda_S(\pi) = 0$, og dermed blir

$$S\boldsymbol{x}_1 = \boldsymbol{x}_1 \qquad\qquad S\boldsymbol{x}_2 = \boldsymbol{0}.$$

(Fortsettes på side 3.)

Oppgave 4 Tensorprodukter og wavelets

4a

Vi definerer 256×256-matrisen X ved at $X_{ij} = 255$ når $(-1)^{i+j} = 1$, og 0 ellers ($0 \leq i, j \leq 255$), det vil si

$$X = \begin{pmatrix} 255 & 0 & \cdots & 255 & 0 \\ 0 & 255 & \cdots & 0 & 255 \\ 255 & 0 & \cdots & 255 & 0 \\ 0 & 255 & \cdots & 0 & 255 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 255 & 0 & \cdots & 255 & 0 \\ 0 & 255 & \cdots & 0 & 255 \end{pmatrix}.$$

Vi definer også filtrene $S_1 = \{1, \underline{-1}\}$ og $S_2 = \{\underline{1}, 1\}$. Regn ut $(S_1 \otimes S_2)X$ ($S_1 \otimes S_2$ er her tensorproduktet av de to lineære transformasjonene S_1 og S_2).

Løsningsforslag: Vi har lært at $(S_1 \otimes S_2)X = S_1 X (S_2)^T$, slik at vi kan starte med å anvende S_1 på alle søylene i X. S_1 kan regnes ut ved hjelp av formelen

$$z_n = x_{n+1} - x_n.$$

Vi får da

$$S_1 X = \begin{pmatrix} -255 & 255 & \cdots & -255 & 255 \\ 255 & -255 & \cdots & 255 & -255 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -255 & 255 & \cdots & -255 & 255 \\ 255 & -255 & \cdots & 255 & -255 \end{pmatrix},$$

der alle naboelementene har motsatt fortegn. Så anvender vi S_2 på radene i $S_1 X$ ved hjelp av formelen

$$z_n = x_n + x_{n-1},$$

og vi får da bare nuller på grunn av alternerende fortegn, slik at $(S_1 \otimes S_2)X = S_1 X (S_2)^T$ blir nullmatrisen.

4b

Forklar algoritmen for å regne ut en DWT med Haar-waveleten, og regn ut DWT over ett nivå for vektorene $(255, 0, 255, 0, ..., 255, 0)$ og $(0, 255, 0, 255, ..., 0, 255)$, der begge vektorene har lengde 256.

Løsningsforslag: En DWT over ett nivå splitter vektoren opp i lavresolusjonskoordinater og detaljkkoordinater ved hjelp av formlene

$$c_{0,n} = \frac{1}{\sqrt{2}}(c_{1,2n} + c_{1,2n+1})$$
$$w_{0,n} = \frac{1}{\sqrt{2}}(c_{1,2n} - c_{1,2n+1}).$$

For begge vektorene vil dermed lavresolusjonskoordinatene bli

$$\boldsymbol{c}_0 = (\underbrace{255/\sqrt{2}, ..., 255/\sqrt{2}}_{128}).$$

(Fortsettes på side 4.)

Detaljkoordinatene blir

$$\boldsymbol{w}_0 = \pm(\underbrace{255/\sqrt{2}, ..., 255/\sqrt{2}}_{128}),$$

der plusstegnet velges for den første vektoren, minustegnet for den andre. Dermed får vi

$$\text{DWT}((255, 0, 255, 0, ..., 255, 0)) = (255/\sqrt{2}, ..., 255/\sqrt{2})$$
$$\text{DWT}((0, 255, 0, 255, ..., 0, 255)) = (\underbrace{255/\sqrt{2}, ..., 255/\sqrt{2}}_{128}, \underbrace{-255/\sqrt{2}, ..., -255/\sqrt{2}}_{128}).$$

4c

Forklar algoritmen for å regne ut en 2-dimensjonal DWT med Haar-waveleten, og regn ut to-dimensjonal DWT over ett nivå for matrisen X fra a). Hvis du viser frem resultatet som et bilde, hva vil du da se i de fire hjørnene?

Løsningsforslag: Algoritmen for 2-dimensjonal DWT sier at vi skal kjøre DWT på rader, deretter på søyler. Fra b) har vi at DWT av søylene med partallsindeks er $(255/\sqrt{2}, ..., 255/\sqrt{2})$, og DWT av søylene med oddetallsindeks er

$$(\underbrace{255/\sqrt{2}, ..., 255/\sqrt{2}}_{128}, \underbrace{-255/\sqrt{2}, ..., -255/\sqrt{2}}_{128}).$$

Dermed blir første halvpart av radene på formen $(255/\sqrt{2}, ..., 255/\sqrt{2})$, og kjører vi en DWT på disse får vi

$$(\underbrace{255, ..., 255}_{128}, \underbrace{0, ..., 0}_{128}).$$

Andre halvpart av radene blir på formen

$$(255/\sqrt{2}, -255/\sqrt{2}..., 255/\sqrt{2}, -255/\sqrt{2}).$$

og kjører vi en DWT på disse får vi

$$(\underbrace{0, ..., 0}_{128}, \underbrace{255, ..., 255}_{128}).$$

Dette betyr at alle elementene i øvre høyre og nedre venstre hjørne blir 0 (hvitt), og alle elementene i øvre venstre og nedre høyre hjørne blir 255 (svart). Spesielt er det ikke nødvendig å korrigere for eventuelle pikselverdier som ligger utenfor $[0, 255]$.

Her kunne du også bruke filtertolkningen av wavelets: Hvis H_0 og H_1 er filtrene i Haar waveleten så skal du anvende disse på rader og søyler på alle mulige måter. Det er klart vi får S_1 og S_2 fra a) ved å gange H_0 og H_1 med $\sqrt{2}$, slik at man i a) faktisk verifiserer at nedre venstre hjørne blir nullmatrisen.

(Fortsettes på side 5.)

Oppgave 5 Konveksitet

Anta at $f_1(\boldsymbol{x})$, $f_2(\boldsymbol{x})$,...,$f_k(\boldsymbol{x})$ er konvekse funksjoner definert på $\mathbb{R}^n$. Vis først at

$$f(\boldsymbol{x}) = \max(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), ..., f_k(\boldsymbol{x}))$$

er en konveks funksjon. Vis deretter at funksjonen som summerer de r største komponentene i $\boldsymbol{x} \in \mathbb{R}^n$ er konveks (Hvis denne kalles g, så defineres g dermed som $g(\boldsymbol{x}) = x_{i_1} + x_{i_2} + ... + x_{i_r}$, der $x_{i_1} \geq x_{i_2} \geq ... \geq x_{i_n}$ gir komponentene i $\boldsymbol{x}$ i avtagende rekkefølge).
Hint: Prøv å finne funksjoner f_1,...,f_k slik at g kan skrives som et maksimum av disse. Andre del av denne oppgaven vil i så fall følge fra første del.
Løsningsforslag: Siden alle f_i er konvekse så har vi at

$$f_i((1-\lambda)\boldsymbol{x} + \lambda\boldsymbol{y}) \leq (1-\lambda)f_i(\boldsymbol{x}) + \lambda f_i(\boldsymbol{y}) \leq (1-\lambda)f(\boldsymbol{x}) + \lambda f(\boldsymbol{y}).$$

Siden dette gjelder for alle i har vi også at det gjelder for maksimum over alle i, slik at

$$f((1-\lambda)\boldsymbol{x} + \lambda\boldsymbol{y}) \leq (1-\lambda)f(\boldsymbol{x}) + \lambda f(\boldsymbol{y}),$$

slik at f er konveks. For andre del av oppgaven så kan vi velge r indekser $(s_1, ..., s_r)$ fra $1, ..., n$ der $s_1 < s_2 < ... < s_r$ på $k = \binom{k}{r}$ forskjellige måter. For hvert slikt valg kan vi definere funksjonen.

$$g_{s_1,...,s_r} = x_{s_1} + x_{s_2} + ... + x_{s_r},$$

og det er klart at $g(\boldsymbol{x}) = \max_{s_1<s_2<...<s_r} g_{s_1,...,s_r}$. Siden hver $g_{s_1,...,s_r}$ er konveks følger det fra første del av oppgaven at g er konveks.

Oppgave 6 Ikkelineær optimering

6a

Finn minimum for funksjonen

$$f(x, y, z) = x^2 + \frac{y^2}{4} + \frac{z^2}{9}$$

under betingelsene $x + y + z = 1$, $x \geq 0$, $y \geq 0$, $z \geq 0$. Du skal bruke Lagranges metode/sette opp KKT-betingelsene.
Løsningsforslag: KKT-betingelsene er

$$\begin{pmatrix} 2x \\ \frac{y}{2} \\ \frac{2z}{9} \end{pmatrix} + \lambda \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + \mu_1 \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix} + \mu_2 \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix} + \mu_3 \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix} = \mathbf{0},$$

samt $x + y + z = 1$, $\mu_i \geq 0$, og $\mu_j = 0$ når tilhørende ulikhet ikke er aktiv. Hvis ingen ulikheter er aktive så må $\mu_1 = \mu_2 = \mu_3 = 0$, og det er da klart at $2x = \frac{y}{2} = \frac{2z}{9}$, slik at $y = 4x$, $z = 9x$. Setter vi dette inn i likhetsbetingelsen får vi at $14x = 1$, slik at $x = 1/14$. Dette gir kandidaten $(x, y, z) = (1/14, 2/7, 9/14)$.
Det er klart at likhetsbetingelsen ikke er oppfylt når alle tre ulikhetene er

(Fortsettes på side 6.)

aktive.
Vi kan derfor anta at en av betingelsene ikke er aktiv, og at en er aktiv. Hvis ulikhet k ikke er aktiv så må $\mu_k = 0$, og det følger da fra rad k i gradientlikningen at $\lambda < 0$. Hvis ulikhet r er aktiv sier da rad r i gradientlikningen at $\mu_r = \lambda < 0$, som ikke er tillatt. Derfor får vi ikke flere kandidater ved å løse gradientlikningen.
Den eneste måten å lage en lineær avhengighet mellom de fire gradientene er å inkludere alle fire, og dette skjer bare hvis alle ulikhetene er aktive. Siden dette var umulig så er alle punkter regulære, og $(1/14, 2/7, 9/14)$ er dermed eneste kandidat til minimum (siden dette optimeringsproblemet er på en lukket, begrenset mengde, så er det klart at vi faktisk må ha et minimum også).

6b

Studer følgende Matlabkode

```
A = [1 1 1];
x0=[0.1; 0.2; 0.7];
mu=1;
r = 3;
alpha=0.1;
epsilon=10^(-3);
x = x0;
while (r*mu>epsilon)
    for numit=1:20
        dphi = -[1/x(1); 1/x(2); 1/x(3)];
        d2phi = [1/x(1)^2 0 0; 0 1/x(2)^2 0; 0 0 1/x(3)^2];
        df  = [2*x(1); x(2)/2; 2*x(3)/9] + mu*dphi;
        d2f = [2 0 0; 0 1/2 0; 0 0 2/9] + mu*d2phi;
        matr=[d2f A'; A 0];
        vect=[-df; 0];
        solvedvals=matr\vect;
        d = solvedvals(1:3);
        x = x + d;
    end
    mu=alpha*mu;
    x
end
```

eller følgende Pythonkode:

```
from numpy import *

A = [1, 1, 1]
x0 = [0.1, 0.2, 0.7]
mu = 1.
r = 3
alpha = 0.1
epsilon=1E-3
x = x0
while (r*mu>epsilon):
```

(Fortsettes på side 7.)

```
    for numit in range(20):
        dphi = -array([1/x[0], 1/x[1], 1/x[2]])
        d2phi = matrix([[1/x[0]**2, 0, 0], \
                        [0, 1/x[1]**2, 0], \
                        [0, 0, 1/x[2]**2]])
        df  = array([2*x[0], x[1]/2, 2*x[2]/9]) + mu*dphi
        d2f = matrix([[2., 0, 0], [0, 1/2., 0], [0, 0, 2/9.]]) \
              + mu*d2phi
        matr = zeros((4,4))
        matr[0:3, 0:3] = d2f
        matr[0:3, 3]=A
        matr[3, 0:3]=A
        vect = zeros(4)
        vect[0:3] = -df
        solvedvals = linalg.solve(matr, vect)
        d = solvedvals[0:3]
        x = x + d
    mu = alpha*mu
    print x
```

(legg merke til at Matlabkoden er litt enklere å forstå når det gjelder hvordan vi lager matrisen `matr`). Kommenter koden linje for linje. Spesielt skal du kommentere

1. hvilken algoritme som blir kjørt
2. hva `x`-verdiene som blir skrevet ut på skjermen konvergerer mot
3. Hvilket resultat som blir brukt på linjen `while (r*mu>epsilon)`.
4. Hva parametrene `r`, `mu`, `x0`, og `alpha` representerer.

Kan du nevne en svakhet ved koden over?

Løsningsforslag: Koden kjører indrepunkts-barriermetoden for å finne minimum til f, slik denne ble definert i a). Barrierfunksjonen her er $\phi(x,y,z) = -\mu\ln x - \mu\ln y - \mu\ln z$, og barrierproblemet er å minimere $f(x,y,z) - \mu\ln x - \mu\ln y - \mu\ln z$ under betingelsen $h(x,y,z) = z+y+z = 1$, Denne har gradient $(2x, y/2, 2z/9) - \mu(1/x, 1/y, 1/z)$ og Hesse-matrise

$$\begin{pmatrix} 2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 2/9 \end{pmatrix} - \mu \begin{pmatrix} -1/x^2 & 0 & 0 \\ 0 & -1/y^2 & 0 \\ 0 & 0 & -1/z^2 \end{pmatrix},$$

og begge disse blir regnet ut i den innerste for-løkka, der Newtons metode blir kjørt på barrierproblemet. En svakhet ved koden er at den ikke betrakter steglengden for Newtons metode, og den kjører et fiksert antall iterasjoner av denne, uten bruk av noen stoppbetingelse. `r` representerer antall ulikhetsbetingelser. Et resultat fra boka garanterer at verdien vi får fra barrierproblemet skiller seg med mindre enn $r\mu$ fra løsningen på det opprinnelige problemet. `mu` er barrier-parameteren, `alpha` er reduksjonsfaktoren for `mu` for hver iterasjon, og `x0` er startpunktet.

SLUTT