LinAlgDat

2024/2025

Projekt C

Projektet består af fire opgaver. Opgave 1 og 2 er rene matematikopgaver (ligesom dem i den skriftlige prøve). Opgave 3 har fokus på anvendelser af lineær algebra. Opgave 4 drejer sig om at implementere metoder og algoritmer fra lineær algebra i F# eller Python.

Besvarelsen af projektet skal bestå af følgende to filer. Filerne må ikke zippes og skal afleveres i Absalon.

- Én pdf-fil, skrevet i LaTeX, med løsninger til opgaverne 1, 2 og 3. Første side i pdf-filen skal være en forside indeholdende forfatterens fulde navn, KU-id og holdnummer.
- Én F#- eller Python-fil med løsninger til Opgave 4 (se opgaveformuleringen for detaljer).

Ved bedømmelsen af projektet lægges naturligvis vægt på korrekthed, men det er også vigtigt, at fremstillingen er klar og overskuelig. Mellemregninger skal medtages og koden skal kommenteres i passende omfang. Projektet skal laves individuelt. Afskrift betragtes som eksamenssnyd.

Programmeringsdelen rettes bl.a. ved at koden bliver afprøvet på hemmeligholdt testdata. Der vil blive udleveret tilsvarende testscripts, som man selv kan teste koden på før aflevering.

Der er adgang til hjælp ved projekthjælp-øvelserne og studiecaféerne på DIKU. Tidsfrister for aflevering, retning, mm. af projektet er beskrevet i dokumentet *Kursusoversigt*. Man er selv ansvarlig for at holde sig orienteret herom.

Besvarelser der er afleveret for sent vil som udgangspunkt ikke blive rettet. Der er ikke mulighed for genaflevering. Aflevér derfor i god tid, også selvom der er dele af opgaverne man ikke har nået.

Opgave 1 (25%)

Betragt matricen

$$\mathbf{A} = \begin{pmatrix} 3 & 1 & 19 \\ 0 & 2 & 14 \\ -2 & 6 & -15 \\ 6 & -2 & 10 \end{pmatrix}.$$

(a) Bestem en QR-faktorisering af $\mathbf{A}$.

Betragt nu underrummet (hyperplanen) $\mathcal{U} \subseteq \mathbb{R}^4$ udspændt af søjlerne i matricen $\mathbf{A}$.

(b) Bestem projektionsmatricen $\mathbf{P}$ for underrummet $\mathcal{U}$.

Gør rede for, at $\mathbf{P}^n = \mathbf{P}$ for alle $n \geqslant 1$.

(c) Betragt vektoren $\mathbf{x} = \mathbf{e}_3 = (0, 0, 1, 0)^\mathsf{T} \in \mathbb{R}^4$.

Bestem den ortogonale projektion af $\mathbf{x}$ på underrummet $\mathcal{U}$.

Bestem spejlingen af $\mathbf{x}$ i underrummet $\mathcal{U}$.

(d) Bestem en ortonormal basis for underrummet $\mathcal{U}^\perp$ (det ortogonale komplement til $\mathcal{U}$).

Lad $\mathbf{Q} = (\mathbf{q}_1|\mathbf{q}_2|\mathbf{q}_3)$ være Q-matricen i QR-faktoriseringen fra delspørgsmål (a) og lad $\{\mathbf{q}_4\}$ være den ortonormale basis for underrummet $\mathcal{U}^\perp$ fundet i delspørgsmål (d). Betragt så matricen

$$\mathbf{B} = (\mathbf{q}_1|\mathbf{q}_2|\mathbf{q}_3|\mathbf{q}_4) \qquad \text{samt vektoren} \qquad \mathbf{v} = \begin{pmatrix} \sqrt{5} \\ \sqrt{7} \\ \sqrt{11} \\ \sqrt{13} \end{pmatrix}.$$

(e) Bestem $\mathbf{B}^{-1}$, altså den inverse til matricen $\mathbf{B}$.

Bestem $\|\mathbf{Bv}\|$, altså normen af vektoren $\mathbf{Bv}$.

Vink: Man behøver ikke at regne ret meget.

Opgave 2 (25%)

Betragt matricen

$$\mathbf{M} = \begin{pmatrix} -2 & -6 & -3 \\ 3 & 7 & 3 \\ -6 & -12 & -5 \end{pmatrix}.$$

(a) Bestem det karakteristiske polynomium for $\mathbf{M}$.

Vink: Udfør først rækkeoperationerne $\mathbf{r}_2 + \mathbf{r}_1 \to \mathbf{r}_1$ og $2\mathbf{r}_2 + \mathbf{r}_3 \to \mathbf{r}_3$ på $\lambda\mathbf{I} - \mathbf{M}$ og bemærk, at 1. og 3. rækkevektor i den derved fremkomne matrix er multipla af $\lambda - 1$.

(b) Bestem alle egenværdierne for $\mathbf{M}$.

(c) Bestem for hver egenværdi for $\mathbf{M}$ en basis for det tilhørende egenrum.

Angiv for hver egenværdi for $\mathbf{M}$ dens geometriske multiplicitet.

(d) Diagonalisér $\mathbf{M}$, dvs. bestem en invertibel matrix $\mathbf{P}$ og en diagonalmatrix $\mathbf{D}$ så $\mathbf{P}^{-1}\mathbf{M}\mathbf{P} = \mathbf{D}$.

(e) Bestem et generelt udtryk for $\mathbf{M}^n$ hvor $n \in \mathbb{N}$.

(Det er en god idé et tjekke, om dit udtryk er korrekt for fx $n = 1, 2, 3, 4, 5$.)

Opgave 3 (25%)

En computers regnekraft måles i FLoating-point Operations Per Second (FLOPS). Vi vil benytte lineær algebra til at beskrive hvordan (super)computernes regnekraft har udviklet sig med tiden.

Frontier (supercomputer) 2022

TABEL 1 (http://en.wikipedia.org/wiki/History_of_supercomputing) viser, for udvalgte år, hvor mange FLOPS verdens bedste supercomputer kunne præstere i det givne år. Vi betegner med t tiden (målt i år) og med $y = y(t)$ det antal FLOPS som verdens bedste supercomputer kunne præstere i år t. Værdierne i første og anden søjle i TABEL 2 er således blot overført fra TABEL 1, mens tredje søjle er beregnet ved at tage den naturlige logaritme (ln) til værdierne i anden søjle.

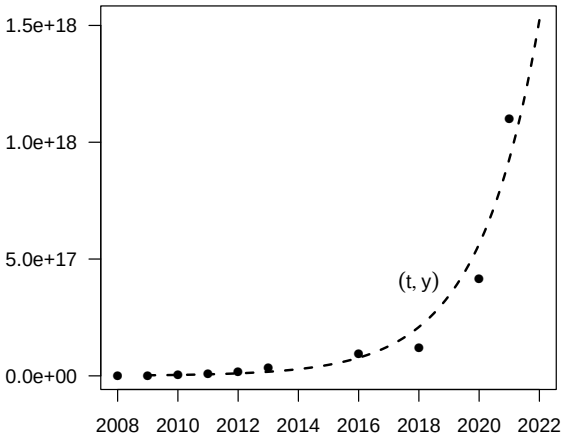
År	**Supercomputer**	**FLOPS**
2008	IBM Roadrunner	$1.026 \cdot 10^{15}$
2009	Cray Jaguar	$1.759 \cdot 10^{15}$
2010	Tianhe-IA	$2.566 \cdot 10^{15}$
2011	Fujitsu K computer	$10.51 \cdot 10^{15}$
2012	IBM Sequoia	$16.32 \cdot 10^{15}$
2013	NUDT Tianhe-2	$33.86 \cdot 10^{15}$
2016	Sunway TaihuLight	$93.01 \cdot 10^{15}$
2018	IBM Summit	$122.3 \cdot 10^{15}$
2020	Fugaku	$415.5 \cdot 10^{15}$
2021	Frontier	$1.102 \cdot 10^{18}$

TABEL 1

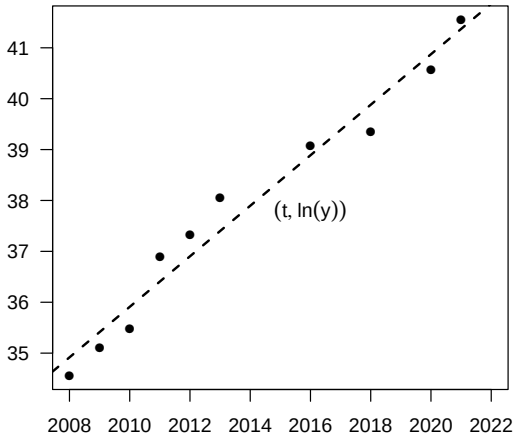
t	y	$\ln y$
2008	$1.026 \cdot 10^{15}$	34.564
2009	$1.759 \cdot 10^{15}$	35.104
2010	$2.566 \cdot 10^{15}$	35.481
2011	$10.51 \cdot 10^{15}$	36.891
2012	$16.32 \cdot 10^{15}$	37.331
2013	$33.86 \cdot 10^{15}$	38.061
2016	$93.01 \cdot 10^{15}$	39.071
2018	$122.3 \cdot 10^{15}$	39.345
2020	$415.5 \cdot 10^{15}$	40.568
2021	$1.102 \cdot 10^{18}$	41.544

TABEL 2

I nedenstående to koordinatsystemer er punkterne (t, y) hhv. $(t, \ln y)$ fra TABEL 2 indtegnet (sammen med stiplede grafer for nogle i første omgang ukendte funktioner):



FIGUR 1: Punkterne (t, y)



FIGUR 2: Punkterne $(t, \ln y)$

Det fremgår af FIGUR 2, at punkterne $(t, \ln y)$ tilnærmelsesvist ligger på en ret linje.

(a) Benyt mindste kvadraters metode (eng: *method of least squares*) til at bestemme forskriften for den bedste rette linje, $\ln y \simeq at + b$, gennem punkterne $(t, \ln y)$ fra TABEL 2.

Vink: Se §4.4.1 Example 4 i lærebogen. Det er i orden at benytte fx en af funktionerne fra Projekt A til matrixmultiplikation.

Den stiplede graf på FIGUR 2 *er grafen for den lineære funktion* $t \mapsto at + b$*, hvor* a *og* b *er de konstanter, som er bestemt ovenfor.*

(b) Begrund, at der gælder følgende tilnærmede forskrift for funktionen $y = y(t)$:

$$y = y(t) \simeq 1.46 \cdot 10^{15} \cdot e^{0.496\,(t-2008)} \qquad (*)$$

Vink: Man har tilnærmelsen $\ln y \simeq at + b$ for de i delspørgsmål (a) fundne konstanter a og b. Tag nu eksponentialfunktionen på begge sider af lighendstegnet. Regn med alle decimaler.

Den stiplede graf på FIGUR 1 *er grafen for eksponentialfunktionen* $t \mapsto 1.46 \cdot 10^{15} \cdot e^{0.496\,(t-2008)}$ *fundet i* $(*)$ *ovenfor.*

(c) Benyt tilnærmelsen $(*)$ til at give et estimat på hvor mange FLOPS verdens bedste supercomputer kunne præstere i år 2000.

Det historiske faktum er, at verdens bedste supercomputer i år 2000 var IBM ASCI White, og denne kunne præstere $7.226 \cdot 10^{12}$ *FLOPS.*

Benyt tilnærmelsen $(*)$ til at give et estimat på hvor mange FLOPS verdens bedste supercomputer kan præstere i år 2030.

Opgave 4 [Programmering i F# eller Python] **(25%)** Din opgave er at afslutteuimplementerede metoder i forbindelse med determinanter og Gram-Schmidt-faktorisering.

For at komme i gang med opgaven skal du først downloade projektfilerne. F#- og Python-filer er tilgængelige på Absalon: F#-filer er tilgængelige via linket `https://absalon.ku.dk/files/9531688/` og Python-filer via linket `https://absalon.ku.dk/files/9531689/`. For det andet man skal få et overblik over projektfilerne, prøve at åbne filerne og se på dem. Bemærk, at de fleste af filerne er identiske med dem, der er angivet i det foregående projekt. Der er kun få nye eller ændrede filer. Her er en kort oversigt over dem:

Du skal afslutte de ikke-implementerede metoder i `projectC/AdvancedExtensions.{fs|py}` hvor `projectC/AdvancedExtensions.{fs|py}` refererer til enten

- F#-filen `projectC/AdvancedExtensions.fs` eller
- Python-filen `projectC/AdvancedExtensions.py`.

Du er velkommen til at tilføje yderligere hjælpemetoder i `AdvancedExtensions.{fs|py}`, men du må ikke omdøbe eller på anden måde ændre typesignaturen for en af de eksisterende metoder. Når indsende din løsning til programmeringsdelen af Projekt C har du kun lov til upload filen `AdvancedExtensions.{fs|py}`. Upload kun denne fil, intet andet, komprimer den ikke.

Når du kopierer projektfilerne til din computer, bedes du kun bruge alfanumeriske ASCII-tegn (og måske understregningen '_' og prikken '.')*, ikke nationale sprogspecifikke (såsom dansk æ, ø, å) i din mappes navne, kan/vil det forhindre `dotnet` for at køre korrekt, hvis overhovedet!

File content

ProjectC/AdvancedExtensions.{fs|py} Denne fil indeholder flere ufærdige metoder. *Dette er den eneste fil du har lov til at ændre og den eneste fil du kan indsende til programmeringsdelen af Projekt C.* Du skal implementere følgende funktioner

- `SquareSubmatrix` som skaber en kvadratisk submatrix givet en kvadratisk matrix som samt række- og kolonneindekser for at fjerne fra det,
- `Determinant` som beregner determinanten af en given kvadratmatrix,
- `GramSchmidt` som beregner Gram-Schmidt-processen på en given matrix og returnerer en "(Q,R)"-dekomponering af matrixen.

Du finder også i filen en uimplementeret funktion, `SetColumn`, som kopierer en vektor v som en kolonne af en matrix A på en given position. Dens gennemførelse er ***ikke en del af opgaven og vil ikke blive testet under evalueringen af Projekt C***. *Vi dog foreslå at implementere det, så du kan bruge det i implementeringen af Gram-Schmidt-processen og en test for den leveres til dig, så du kan kontrollere rigtigheden af din implementering.*

*ASCII: American Standard Code for Information Interchange, 26 x 2 standard alfabetiske tegn (store og små bogstaver), tal samt tegnsætning, klammer og andre almindelige tegn. Ingen accent eller sprogspecifikke tegn. De alfanumeriske tegn er 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789'.

ProjectC/TestProjectC.{fs|py} Denne fil indeholder koden for testfunktionerne.

(kun F#) ProjectC/RunTest.fsx Denne fil indeholder data til selvtestning og koden, der kører testene.

F#: **Byg og kør projektet.** Scripts leveres til at beregne og udføre koden med dotnet. Deres brug er kort beskrevet i filen README.md.
Gå ikke i panik, hvis ingen af bygge/kompileringsmetoderne nævnt ovenfor lyder bekendt for dig. Kontakt venligst TA'erne og/eller François og dem vil hjælpe dig.

Python: Du behøver ikke nogen kompilering. For at køre projektet

- gå til mappen ProjectC.
- Windows:
  `C:\path_to\ProjectC> python TestProjectC.py`
  Linux/MacOs
  `/path_to/ProjectC$ python TestProjectC.py`

Henrik Holm (holm@math.ku.dk)
Henrik Laurberg Pedersen (henrikp@math.ku.dk)
Jon Sporring (sporring@di.ku.dk)