LinAlgDat

2021/2022

Projekt C

Projektet består af fire opgaver. Opgave 1 og 2 er rene matematikopgaver (ligesom dem i de skriftlige prøver). 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 jeres kode skal kommenteres i passende omfang. Projektet skal laves individuelt. Afskrift betragtes som eksamenssnyd.

Programmeringsdelen rettes bl.a. ved at jeres løsning bliver afprøvet på hemmeligholdt testdata. Der vil blive udleveret tilsvarende testscripts, som I selv kan teste jeres kode på før I afleverer.

Der er adgang til hjælp ved projekthjælp-øvelserne og (evt. i lidt mindre udstrækning) ved studiecaféerne på DIKU. Tidsfrister for aflevering, retning, mm. af projektet er beskrevet i dokumentet *Kursusoversigt*. I er selv ansvarlige for at holde jer 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 I ikke har nået.

Opgave 1 (25%)

Betragt matricen

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & -5 \\ 5 & -2 & -13 \\ -3 & 3 & 15 \\ 1 & -1 & 7 \end{pmatrix}.$$

(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 $\mathcal{U}$.

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

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

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

(d) Bestem en 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

$$\mathbf{B} = (\mathbf{q}_1|\mathbf{q}_2|\mathbf{q}_3|\mathbf{q}_4) \qquad \text{samt vektoren} \qquad \mathbf{v} = \begin{pmatrix} \sqrt{2} \\ \sqrt{3} \\ \sqrt{5} \\ \sqrt{6} \end{pmatrix}.$$

(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

$$\mathbf{A} = \begin{pmatrix} -1 & 2 & 1 \\ -2 & 3 & 1 \\ -2 & 2 & 2 \end{pmatrix}.$$

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

(*Vink:* Udfør først rækkeoperationerne $-\mathbf{r}_2 + \mathbf{r}_1 \to \mathbf{r}_1$ og $-\mathbf{r}_2 + \mathbf{r}_3 \to \mathbf{r}_3$ på $\lambda\mathbf{I} - \mathbf{A}$ og bemærk, at 1. og 3. rækkevektor i den derved fremkomne matrix kan skrives som $(\lambda - 1)$ gange henholdsvis $[1 \ -1 \ \ 0]$ og $[0 \ -1 \ \ 1]$.)

(b) Bestem alle egenværdierne for $\mathbf{A}$ og deres algebraiske multipliciteter.

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

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

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

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

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.

RIKEN's Fugaku (2020)

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.

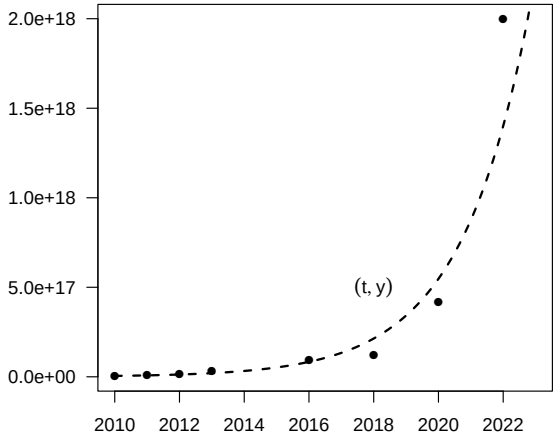
År	**Supercomputer**	**FLOPS**
2010	Tianhe-IA	$2.566 \cdot 10^{15}$
2011	Fujitsu K computer	$10.51 \cdot 10^{15}$
2012	IBM Sequoia	$16.32 \cdot 10^{15}$
2013	NUDT Tianhe-2	$33.86 \cdot 10^{15}$
2016	Sunway TaihuLight	$93.01 \cdot 10^{15}$
2018	IBM Summit	$122.3 \cdot 10^{15}$
2020	Fugaku	$415.5 \cdot 10^{15}$
2022	Aurora (expected)	$2.000 \cdot 10^{18}$

TABEL 1

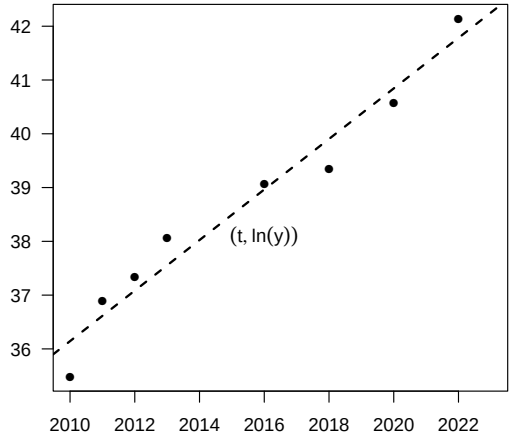
t	y	$\ln y$
2010	$2.566 \cdot 10^{15}$	35.481
2011	$10.51 \cdot 10^{15}$	36.891
2012	$16.32 \cdot 10^{15}$	37.331
2013	$33.86 \cdot 10^{15}$	38.061
2016	$93.01 \cdot 10^{15}$	39.071
2018	$122.3 \cdot 10^{15}$	39.345
2020	$415.5 \cdot 10^{15}$	40.568
2022	$2.000 \cdot 10^{18}$	42.140

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 linie.

(a) Benyt mindste kvadraters metode (eng: *method of least squares*) til at bestemme forskriften for den bedste rette linie, $\ln y \simeq at + b$, 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 $y = y(t)$:

$$y = y(t) \simeq 4.98 \cdot 10^{15} \cdot e^{0.470(t-2010)} \qquad (*)$$

(*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* $t \mapsto 4.98 \cdot 10^{15} \cdot e^{0.470(t-2010)}$ *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%) Your assignment is to finish the unimplemented methods retaled to determinants and Gram-Schmidt factorisation.

To get started on the assignment, first download the project files. F# and Python files are available on Absalon: https://absalon.ku.dk/courses/56839/files/folder/Projekt%20C?.

You have to finish the unimplemented methods in `projectC/AdvancedExtensions.{fs|py}` where `projectC/AdvancedExtensions.{fs|py}` refers to either

- the F# file `projectC/AdvancedExtensions.fs` or
- the Python file `projectC/AdvancedExtensions.py`.

You are welcome to add additional helper methods in `AdvancedExtensions.{fs|py}`, but you are not allowed to rename or otherwise alter the type signature of any of the existing methods. When submitting your solution to the programming part of Project C you are only allowed to upload the file `ProjectC/AdvancedExtensions.{fs|py}`.

When copying the project files to your computer, please, only use alphanumerical ASCII characters (and maybe the underscore '_' and dot '.')*, not language national specific ones (such as Danish æ, ø, å) in your folder's names, it might/will prevent `mono.exe` to run properly, if at all!

File content

ProjectC/AdvancedExtensions.{fs|py} This file contains several unfinished methods. *This is the only file you are allowed to modify and the only file you may submit for the programming part of Project C.* You will have to implement the following functions

- `SquareSubmatrix` which creates a square submatrix given a square matrix as well as row and column indices to remove from it,
- `Determinant` which computes the determinant of a given square matrix,
- `GramSchmidt` which computes Gram-Schmidt process on a given matrix and returns a "(Q,R)" decomposition of the matrix.

You will also find in the file an unimplemented function, `SetColumn`, which copies a vector v as a column of a matrix A at a given position. Its implementation is ***not part of the assignment and will not be tested during the veluation of Project C***. We however suggest to implement it so as to be able to use it in the implementation of the Gram-Schmidt process and a test for it is provided to you, so you can check the correctness of your implementation.

ProjectC/TestProjectC.{fs|py} This file contains the code of the test functions.

(F# only) ProjectC/RunTest.fsx This file contains data for self-testing and the code which runs the tests.

*American Standard Code for Information Interchange, 26 x 2 standard alphabetical characters (upper- and lowercase), numbers, as well as punctuation, braces, and other "standard" characters. No accented or language specific characters. The alphanumerical characters are 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789'.

F#: Build and run the project. Some build scripts are provided for building and/or running the code. You should have all a running installation of `mono`, used in Programering og Problemsløsning (PoP). Compilation scripts are provided for Windows and `bash` (MacOS/Linux).

- Go to ProjectC folder.
- Windows:
  `C:\path_to\ProjectC> compileAndRun`
  Linux/MacOs
  `/path_to/ProjectC$ ./compileAndRun.sh`

These commands produce a executable program called `RunTest.exe`. It can be run from the command line.

```
C:\PathToFolder\>RunTest (Windows)
username:PathToFolder$ ./RunTest.exe (Linux, MacOS)
```

It is also possible just to compile and not run, using the `compile.*` script

```
C:\PathToFolder\>compile (Windows)
username:PathToFolder$ ./compile.sh (Linux, MacOS)
```

Do not panic if none of the build/compile methods mentioned above sound familiar to you. Please contact the TAs and they will help you.

Python: You do not need any compilation. To run the project

- go to ProjectC folder.
- 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)
Francois Bernard Lauze (francois@di.ku.dk)