



LinAlgDat

2017/2018

Projekt B

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 C#.

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

- En pdf-fil, genereret af \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.
- Netop en C#-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 C#-kode skal kommenteres i passende omfang. Projektet laves og bedømmes individuelt.

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.

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. Aflever derfor i god tid, også selvom der er dele af opgaverne I ikke har nået.

Opgave 1 (25%)

Betragt den lineære transformation $T: \mathbb{R}^4 \rightarrow \mathbb{R}^4$ givet ved

$$T \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} -x_1 + 2x_3 + 2x_4 \\ 2x_1 - x_2 + 7x_3 + 3x_4 \\ 2x_1 + x_2 + 5x_3 - x_4 \\ x_1 - 2x_2 + 2x_4 \end{pmatrix} \quad \text{for } \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \in \mathbb{R}^4.$$

(a) Bestem den matrix \mathbf{A} som opfylder $T(\mathbf{x}) = \mathbf{Ax}$ for alle $\mathbf{x} \in \mathbb{R}^4$.

(b) Bestem en basis for $\text{ran } T$ (billedet af T).

Bestem en vektor $\mathbf{y} \in \mathbb{R}^4$ som ikke tilhører $\text{ran } T$. Er T surjektiv?

(c) Bestem en basis for $\ker T$ (kernen af T).

Bestem to forskellige vektorer \mathbf{x}_1 og \mathbf{x}_2 i \mathbb{R}^4 som opfylder $T(\mathbf{x}_1) = \mathbf{0} = T(\mathbf{x}_2)$. Er T injektiv?

(d) Bestem en vektor $\mathbf{x} \neq \mathbf{0}$ i \mathbb{R}^4 som opfylder $(T \circ T)(\mathbf{x}) = \mathbf{0}$.

Opgave 2 (25%)

Betragt følgende vektorer i \mathbb{R}^4 :

$$\mathbf{u}_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \end{pmatrix}, \mathbf{u}_2 = \begin{pmatrix} 0 \\ -2 \\ -1 \\ 1 \end{pmatrix}, \mathbf{u}_3 = \begin{pmatrix} -1 \\ -1 \\ 0 \\ 1 \end{pmatrix} \quad \text{og} \quad \mathbf{v}_1 = \begin{pmatrix} 2 \\ 2 \\ 1 \\ -1 \end{pmatrix}, \mathbf{v}_2 = \begin{pmatrix} 2 \\ 0 \\ 1 \\ 1 \end{pmatrix}, \mathbf{v}_3 = \begin{pmatrix} 5 \\ -1 \\ 3 \\ 4 \end{pmatrix}.$$

Det oplyses, at $\mathcal{B} = \{\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3\}$ og $\mathcal{C} = \{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$ begge er baser for det samme underrum.

(a) Bestem basisskift-matricen $\mathbf{P}_{\mathcal{B} \leftarrow \mathcal{C}}$ (fra \mathcal{C} til \mathcal{B}).




(*Vink:* Det er nyttigt at bringe matricen $(\mathbf{u}_1 | \mathbf{u}_2 | \mathbf{u}_3 | \mathbf{v}_1 | \mathbf{v}_2 | \mathbf{v}_3)$ på reduceret rækkeechelonform.)

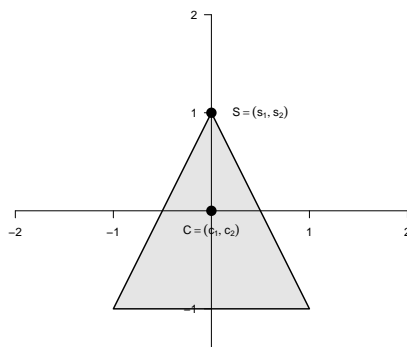
(b) Bestem basisskift-matricen $\mathbf{P}_{\mathcal{C} \leftarrow \mathcal{B}}$ (fra \mathcal{B} til \mathcal{C}).

(c) Lad λ være et (ukendt) tal og betragt vektoren $\mathbf{x} = 2\mathbf{u}_1 + \mathbf{u}_2 + \lambda\mathbf{u}_3$.




Bestem tal a , b og c (udtrykt ved λ) således at $a\mathbf{v}_1 + b\mathbf{v}_2 + c\mathbf{v}_3 = \mathbf{x}$.




Opgave 3 (25%)

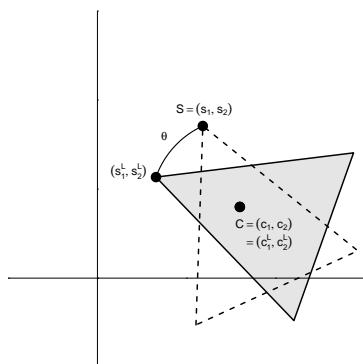
I et computerspil – som fx *Asteroids* (Atari Inc, 1979) – styrer spilleren et trekantformet rumskib i et 2D-koordinatsystem ved hjælp af piletasterne ,  og . Rent grafisk ser rumskibet ud som på figuren nedenfor. Fra et matematisk synspunkt er rumskibets position i koordinatsystemet bestemt ved positionen af to punkter: rumskibets *centrum* $C = (c_1, c_2)$ og *spids* $S = (s_1, s_2)$.




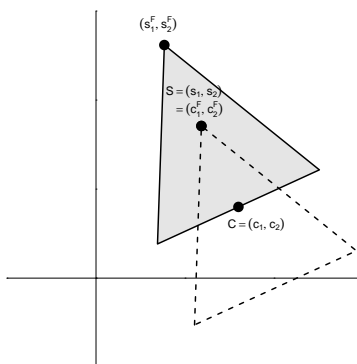
Rumskibets position ved spillets start


Ved spillets start er rumskibet placeret som på figuren ovenfor, dvs. $C = (0,0)$ og $S = (0,1)$. Effekten ved tryk på piletasterne  (Left rotate),  (Forward) og  (Right rotate) er:

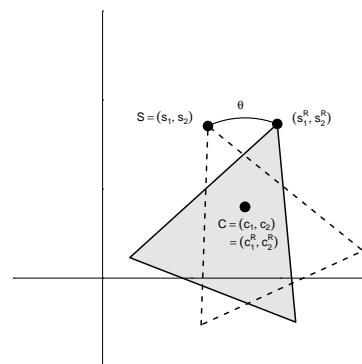
- Ved tryk på  /  roteres rumskibet omkring sit centrum en vis vinkel θ mod / med uret.
- Ved tryk på  parallelforskydes rumskibet efter vektoren \overrightarrow{CS} (vektoren fra C til S).




Ved tryk på 






Ved tryk på 



Ved tryk på 


Antag nu, at rumskibets centrum og spids har koordinatsæt hhv. $C = (c_1, c_2)$ og $S = (s_1, s_2)$. Vi indfører følgende betegnelser:

- (c_1^L, c_2^L) og (s_1^L, s_2^L) er koordinaterne for hhv. rumskibets centrum og spids efter tryk på 
 (c_1^F, c_2^F) og (s_1^F, s_2^F) er koordinaterne for hhv. rumskibets centrum og spids efter tryk på 
 (c_1^R, c_2^R) og (s_1^R, s_2^R) er koordinaterne for hhv. rumskibets centrum og spids efter tryk på 

Vi samler de fire tal c_1, c_2, s_1, s_2 , som fastlægger rumskibets position, i én vektor $(c_1, c_2, s_1, s_2) \in \mathbb{R}^4$.

(a) Bestem en 4×4 matrix \mathbf{F} som opfylder

$$\begin{pmatrix} c_1^F \\ c_2^F \\ s_1^F \\ s_2^F \end{pmatrix} = \mathbf{F} \begin{pmatrix} c_1 \\ c_2 \\ s_1 \\ s_2 \end{pmatrix}.$$

(Vink: Ved tryk på  parallelforskydes rumskibet med vektoren $\overrightarrow{CS} = \begin{pmatrix} s_1 - c_1 \\ s_2 - c_2 \end{pmatrix}$.)



(b) Gør rede for, at der gælder formlerne

$$\begin{pmatrix} c_1^L \\ c_2^L \\ s_1^L \\ s_2^L \end{pmatrix} = \mathbf{L}_\theta \begin{pmatrix} c_1 \\ c_2 \\ s_1 \\ s_2 \end{pmatrix} \quad \text{og} \quad \begin{pmatrix} c_1^R \\ c_2^R \\ s_1^R \\ s_2^R \end{pmatrix} = \mathbf{R}_\theta \begin{pmatrix} c_1 \\ c_2 \\ s_1 \\ s_2 \end{pmatrix}$$

hvor \mathbf{L}_θ og \mathbf{R}_θ er følgende 4×4 matricer:

$$\mathbf{L}_\theta = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 - \cos \theta & \sin \theta & \cos \theta & -\sin \theta \\ -\sin \theta & 1 - \cos \theta & \sin \theta & \cos \theta \end{pmatrix} \quad \text{og} \quad \mathbf{R}_\theta = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 - \cos \theta & -\sin \theta & \cos \theta & \sin \theta \\ \sin \theta & 1 - \cos \theta & -\sin \theta & \cos \theta \end{pmatrix}.$$

Følgende vink/oplysninger må frit benyttes:

- Ved tryk på  er rumskibets centrum uændret, så $c_1^L = c_1$ og $c_2^L = c_2$.
- Rotationsmatricen $\begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$ drejer som bekendt en vektor θ grader **mod uret** omkring **origo**. Ved tryk på  drejes vektoren $\overrightarrow{CS} = \begin{pmatrix} s_1 - c_1 \\ s_2 - c_2 \end{pmatrix}$, som går fra rumskibets centrum til dets spids, θ grader mod uret omkring **rumskibets centrum**. Dette betyder at $\begin{pmatrix} s_1^L \\ s_2^L \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} + \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} s_1 - c_1 \\ s_2 - c_2 \end{pmatrix}$. Regn nu videre på dette udtryk og bestem s_1^L og s_2^L udtrykt ved c_1, c_2, s_1, s_2 .
- Matricen \mathbf{R}_θ kan bestemmes når først \mathbf{L}_θ er fundet ved at benytte, at $\mathbf{R}_\theta = \mathbf{L}_{-\theta}$.

(c) Lad $\theta = 20^\circ$ og antag, at rumskibet er i sin startposition, dvs. $C = (0, 0)$ og $S = (0, 1)$.

Bestem positionen af rumskibets centrum og spids efter følgende tastkombination (der læses fra venstre mod højre):



(Man får brug for at multiplicere seks 4×4 matricer i en passende rækkefølge. Hertil kan man fx benytte C#-funktionen fra Opgave 4 i Projekt A eller en lommeregner.)

Opgave 4 [Programming i C#] (25%)

In this project you have to implement a number of operations related to Gauss Elimination. This requires a solid basic understanding of elementary row operations, forward reduction, backward reduction, and reduced Echelon form. To get started on the assignment, one has to first download the project files from <https://absalon.instructure.com/courses/25079/files/folder/Projekt%20B>. Secondly, one should get an overview of the project files, try and open the files and have a look at them. Observe that most of the files are identical to those provided in the previous project. There are only a few new or modified files. Here is a brief summary of them:

ProjectB/GaussExtensions.cs 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 B.*

ProjectB/MainClass.cs This file contains data for self-testing.

ProjectB/Program.cs This file is used to test your implementation in `GaussExtensions.cs` against the test data in `MainClass.cs`.

Your assignment is to finish the unimplemented methods in `ProjectB/GaussExtensions.cs`. On the next page you will find a description of the algorithms you must follow when implementing most of these methods. Carefully study these algorithms and read the comments above each method before you begin. Notice that the method `AugmentRight` from Project A is provided as it may come in handy when implementing `GaussElimination`. You are welcome to add additional helper methods in `GaussExtensions.cs`, 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 B you are only allowed to upload the file `ProjectB/GaussExtensions.cs`.

Build and run the project in JetBrains Rider using `ProjectB/ProjectB.csproj`. Alternatively, we provide a Makefile and if you have Mono installed you can run the following commands:

- `make build` – This command builds the project using `msbuild`. The output executable is named `ProjectB.exe` and is located in `ProjectB/bin/Debug/`.
- `make run` – This simply runs the executable using `mono`.
- `make clean` – This one does what the command name says.

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.

Elementary row operations.

Replacement (corresponding to row numbers i and j and a real number m): $\mathbf{r}_i + m\mathbf{r}_j \rightarrow \mathbf{r}_i$.

Interchange (corresponding to row numbers i and j): $\mathbf{r}_i \leftrightarrow \mathbf{r}_j$.

Scaling (corresponding to a row number i and a nonzero real number c): $c\mathbf{r}_i \rightarrow \mathbf{r}_i$.

Note that in row replacement use “ $+m$ ” (as in the lectures) and not “ $-m$ ” (as in the textbook).

Algorithm: Forward reduction.

Let \mathbf{M} be an $m \times n$ matrix.

- Step 1** Locate the first nonzero column from the left in \mathbf{M} and call it column j . This is the *pivot column*.
- Step 2** Choose the first (from the top) nonzero entry p_j in column j . The number p_j is called the *pivot* for column j and the row in which the pivot appears is called the *pivot row*. Interchange the pivot row and the first row and then use replacements to reduce all nonzero entries below p_j to zero.
- Step 3** Cover up the first row to form a submatrix of the current matrix with one less row. If the submatrix has no rows or all zero rows, then stop; else apply **Step 1** and **Step 2** to the submatrix. Continue in this way, always covering up the first row in the current submatrix to form a new submatrix with one less row.

Algorithm: Backward reduction.

Let \mathbf{U} be an echelon form for an $m \times n$ matrix. Select the last pivot column to the right in \mathbf{U} .

- Step 1** Scale the pivot row to give pivot value 1 and use the pivot row in replacements to reduce all entries above the pivot to zero.
- Step 2** Repeat **Step 1** for each pivot column in \mathbf{U} to the left of the one just processed. Stop after all the pivot columns have been processed.

Henrik Holm (holm@math.ku.dk)
Henrik Laurberg Pedersen (henrikp@math.ku.dk)
Francois Bernard Lauze (francois@di.ku.dk)