Optimering

Lineær programmering og Simplex-metoden

Projektrapport P2
Gruppe A400a

Aalborg Universitet
Føreste Studieår v/ Det Ingeniør- og Naturvidenskabelige Fakultet
Strandvejen 12-14 • 9000 Aalborg

Aalborg Universitet - Basis
Matematik
Strandvejen 12-14
DK-9000 Aalborg
http://www.math.aau.dk

AALBORG UNIVERSITET
STUDENTERRAPPORT

Titel:
Lineær programmering og Simplex-metoden

Tema:
Optimering

Projektperiode:
Fra februar til juni 2019 (2. semester)

Projektgruppe:
Gruppe A400a

Deltagere:
Thomas Dam
Jonas Faartoft Jensen
Signe Marie Madsen
Silje Bo Fischer Nielsen
Bartal Eyðfinsson Veyhe
Jimmy Bredal Wolfsen

Vejledere:
Lisbeth Fajstrup og Kirsten Krogh Hansen

Sidetal: 90

Afleveringsdato:
27. maj 2019

Synopsis:

Projektet undersøger hvorfor lineær programmering og mere specifikt Simplex-metoden virker ved optimering, og for hvilke optimeringsproblemer det giver mening at anvende Simplex-metoden.
Derfor redegøres der for væsentlige definitioner indenfor lineær algebra, der er nødvendige for at beskrive teorien bag Simplex-metoden. Derudover præsenteres longhand- og matrix-notation i forhold til opstilling af lineære programmeringsproblemer, og hvordan disse laves om til standard form, hvilket kræves af modellen, før Simplex-metoden kan anvendes. En grafisk løsningsmetode anvendes i denne sammenhæng til at vise, hvilke fire udfald der kan være i forbindelse med løsning af lineære optimeringsproblemer. Der følger et kapitel om den geometriske og algebraiske forståelse af lineær programmering, som beskriver teorien om konvekse kombinationer og konvekse mængder, og desuden studeres polyeder, da disse er essentielle i forståelsen af sammenhængen mellem ekstremumspunkter og basal mulige løsninger. Disse teoretiske områder leder frem til Simplex-metoden, hvor argumenterne for, hvorfor denne virker præsenteres og implementeres i en algoritme baseret på Simplex-metoden på tabelform.
Algoritmen bruges på diæt-problemet, hvor der herefter diskuteres problemer ved brug af Simplex-metoden i et samspil med modelleringen af virkeligheden.

Abstract

This project investigates why linear programming and more specifically the Simplex-method work. Furthermore, it examines which type of optimisation problems, set up from a real-life problem and then scaled down to mathematical modelling, the Simplex-method can be used on. Thus, concepts and notions concerning linear algebra, geometric and algebraic understanding of linear programming and the implementation of the Simplex-method are defined and explained.

The linear algebra chapter defines vectors and matrices, and fundamental properties concerning these. The notations of linear programming problems are introduced by longhand- and matrix-notation, and the process of making a linear programming problem into a standard form problem is accounted for. Furthermore, this chapter demonstrates a graphical solution which summarises the four possible outcomes. The next chapter concerning a geometric and algebraic understanding of linear programming covers the theory of convex combinations and convex sets, and moreover defines the quantity of a polyhedron which is essential in the understanding of the connection between extreme points and basic feasible solutions. These three chapters lead to the chapter about the Simplex-method where the argument as to why the Simplex-method works is constructed. Important notions such as the feasible direction $\mathbf{d}$, the reduced cost $\hat{c}_j$ and the longest distance $\mathbf{d}$ can go without leaving the polyhedron, noted by θ^*, are introduced which lead to a general iteration of the Simplex-method described in pseudocode. Pivot selection is described, and in this project the rules of the most negative reduced cost and furthermore the smallest subscript rule are applied. These notions and rules result in the full tableau of the Simplex-method described by pseudocode.

The full tableau of the Simplex-method is then implemented in an algorithm, which is applied to the diet problem. Lastly, a discussion is made regarding the problems about the Simplex-method and mathematical modelling where the linearity of a problem has significant implications on how well linear programming can be applied to the problem. Thus, it is always important to compare the results of the Simplex-method with the modelling and the constraints given from the problem to insure that the results have a practical application.

Forord

Dette projekt er skrevet af studerende i gruppe A400a på andet semester Matematik ved det Ingeniør- og Naturvidenskabelige fakultet på Aalborg Universitet. Projektet er udarbejdet i foråret 2019.

Temaet for projektet er optimering, hvor fokus er på optimeringsmetoden Lineær Programmering og herunder den matematisk systematiske Simplex-metode. Projektet henvender sig til læsere ved starten af andet semester af de matematiske uddannelser.

Projektet er udarbejdet over flere måneder, hvor der er arbejdet med teori i forhold til et problem - altså problembaseret læring. Nysgerrigheden faldt på, hvordan man kan optimere praktiske problemer, hvor flere variable indgår, og hvornår matematiske modeller overholder linearitetsbetingelser. Her faldt interessen især på Simplex-metoden, hvor det interessante var at finde ud af, hvorfor metoden virker. Projektet er afgrænset til Lineær Programmering, hvorfor ikke-linære modeller ikke vil blive betragtet.

Formålet med projektet er at skabe mere kendskab til lineær programmering og Simplex-metoden og ud fra disse skabe teoretiske argumenter for, hvorfor Simplex-metoden virker til optimering og skabe en algoritme der implementerer Simplex-metoden til løsning af lineære programmeringsproblemer. Denne skal så bruges til at foretage en matematisk analyse af diæt-problemet.

I forbindelse med udarbejdelsen af dette projekt bruges Vancouver-metoden til kildehåndtering og udarbejdelse af bibliografien. Alle kilder nummereres med et tal [n] og kan altså findes i litteraturlisten under dette nummer. Sætninger, definitioner, lemmaer, figurer, tabeller og eksempler nummereres løbende efter kapitelnummeret. Efter indholdsfortegnelsen forekommer et symbolregister. Det er desuden væsentligt at bemærke, at nye begreber eller udtryk kursiveres i teksten første gang, de bliver introduceret i teksten, hvorfor det er forskelligt fra tekststykke til tekststykke, om kursivering sker i brødteksten eller i definitionerne selv.

Slutteligt ønsker projektets forfattere at rette en tak til projektets vejledere Lisbeth Fajstrup og Kirsten Krogh Hansen for at have stået til rådighed gennem hele projektet, når gruppen har haft behov for vejledernes input både i forhold til det faglige indhold og i forhold til gruppesamarbejdet og projektplanlægning.

Indhold

Symbolregister

$\mathbf{0}$	Nul-vektoren
$*$	Anvendes i sammenhæng med matrix-formen $A\mathbf{x} * \mathbf{b}$ af betingelserne i et lineært programmeringsproblem, hvor $*$ enten betyder $\geq$, $\leq$ eller $=$
θ^*	Den mindste rate af $-x_i/d_i$ for $i \in \{B(1), B(2), \ldots, B(m)\}$ hvor $d_i < 0$
ℓ	Indeks for den basale variabel som forlader basen
λ	Anvendes som symbol for et reelt tal i intervallet $[0, 1]$ i forbindelse med konvekse kombinationer og konvekse mængder
$\begin{bmatrix} A & B \end{bmatrix}$	Sammensætning af to matricer A og B
$\begin{bmatrix} A & \mathbf{b} \end{bmatrix}$	En total-matrix bestående af en matrix A og en vektor $\mathbf{b}$
A	En generel betegnelse for en matrix (bemærk at vi anvender et stort bogstav og at dette bogstav *ikke* gøres fed)
$\mathbf{A_j}$	En søjle-vektor med indgange i den j'te søjle i matricen A
$A^\intercal$	En transponeret matrix - har samme betydning for vektorer
$\mathbf{a_i}$	En søjle-vektor med indgange i den i'te række i matricen A
a_{ij}	Indgangen i den i'te række og den j'te søjle af en matrix
$B(i)$	Indeksering for basale variable og tilhørende søjle-vektorer fra en matrix A som er i basis-matricen B
$\mathbf{c}$	Kostkoefficient-vektoren som en søjle-vektor
$\hat{\mathbf{c}}$	En søjle-vektor der indeholder de reducerede kostkoefficienter $\hat{c}_i$ for $i \in \{1, 2, \ldots, n\}$
$\mathbf{d}$	Den j'te mulige basisretning
$\mathbf{d_B}$	Den j'te mulige basisretning tilhørende de basale variable x_i for $i \in \{B(1), B(2), \ldots, B(m)\}$
E	En elementær-matrix
$\mathbf{e_i}$	Den i'te standard-vektor med indgange $e_j = 0$ for alle $j \neq i$ og én indgang $e_j = 1$ for $j = i$
I_n	Identitets-matricen bestående af n distinkte standard-vektorer $\mathbf{e_i}$ for $i \in \{1, 2, \ldots, n\}$
P	Anvendes for mængder beskrevet som polyeder
Q	Anvendes som symbol for den matrix der fås ved multiplikation af elementær-matricer, så $Q = E_k E_{k-1} \cdots E_1$
R	Anvendes som den reducerede trappeform af en matrix A

r_i Anvendes som betegnelse for den i'te række i forbindelse med udførelse af elementære rækkeoperationer på en matrix eller ved tabelformen af Simplex-metoden

$\mathbf{x_B}$ En vektor indeholdende værdierne af de basale variable

1 | Problemanalyse

Dette kapitel har til formål først at beskrive begrebet *lineær programmering* og hvilke typer af problemer, der karakteriseres som lineære programmeringsproblemer. Dernæst indebærer dette en analyse af karakteristika for de typer af problemstillinger, man kan løse med metoder inden for lineær programmering, samt hvilke problemstillinger der *ikke* umiddelbart kan løses med lineær programmering, medmindre der foretages de rette antagelser og modelleringer. Slutteligt leder dette frem til en beskrivelse af problemformuleringen til projektet.

1.1 Historisk introduktion

Under Anden Verdenskrig opstod der et akut behov for at optimere transporten af krigsvåben- og midler til soldater i udsatte områder [1]. Til dette opfandt man lineær programmering, som er en optimeringsmetode i matematikken skabt af blandt andet Leonid Vitalievich Kantorovich og Tjalling Charles Koopmans i årene omkring Anden Verdenskrig.

Kantorovich, der var matematiker fra Sovjet Unionen, begyndte at interessere sig for lineær programmering i 1939 [2]. På dette tidspunkt blev han kontaktet af en fabrik, der lavede planker, fordi de havde brug for en videnskabelig rådgiver til at hjælpe med at optimere deres produktion af planker. Dog fandt Kantorovich ud af, at værktøjet lineær programmering kunne bruges til mere end bare optimering på fabrikker.

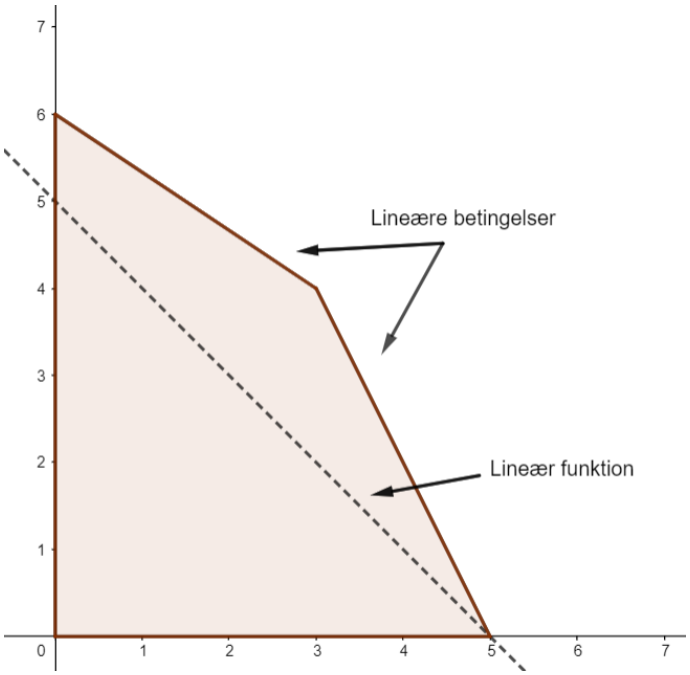
På nogenlunde samme tid som Kantorovich begyndte den amerikanske matematiker Koopmans at arbejde med lineær programmering i USA. Han var ansat som statistiker i militæret under Anden Verdenskrig, hvor han arbejdede for "the Allied Shipping Adjustment Board", og han var bekymret for deres transportmodel, hvilket fik ham til at søge efter en bedre løsning, hvorfor han kom frem til lineær programmering . I 1975 vandt Kantorovich og Koopmans sammen Nobelprisen i økonomi for deres bidrag til optimering og dets anvendelser [3].

Udover Kantorovich og Koopmans er den amerikanske matematiker George B. Dantzig også kendt for at have studeret lineær programmering. Han arbejdede videre med Koopmans' teorier indenfor blandt andet økonomiske problemstillinger og er i dag kendt som ophavsmand til lineær programmering i den vestlige verden, da han er opfinder af Simplexmetoden, som viste sig meget anvendelig indenfor lineær programmering [2].

1.2 Lineær programmering og dets anvendelser

Lineær programmering er en optimeringsmetode inden for matematik, som bruges til at bestemme den mest optimale løsning for et antal variable ud fra en funktion, som beskriver det pågældende problem. Variablene vil komme til udtryk i nogle lineære begrænsninger i det givet problem, som ikke må overskrides for at nå den optimale løsning til funktionen. Det kan blandt andet bruges til at belyse mange forskellige emner indenfor eksempelvis produktion, transport og økonomi [4]. Selvom det hedder lineær programmering er det

ikke programmering i en datalogisk forstand. I stedet dækker begrebet over matematiske og systematiske metoder, som ofte bruges til modellering især i økonomi til at finde den maksimale profit eller minimale omkostning. Dog kan man overføre lineære programmeringsproblemer til en algoritme, hvilket er nyttigt for især store problemer, da vi her kan udføre flere aritmetiske operationer både hurtigere og nemmere. I ordet lineær ligger der, at det problem der skal optimeres og de betingelser, som begrænser problemet, skal være af en lineær form, hvilket intuitivt og uden en videre matematisk præcision betyder, at der ikke må være led i større potens end 1 eller indgå logaritme og andre ikke-lineære former. Grafisk kan man derfor anskue lineære programmeringsproblemer ud fra figur 1.1.



Figur 1.1: *Et lineært programmeringsproblem med en lineær funktion der ønskes optimeret underlagt lineære betingelser.*

Et eksempel på et lineært programmeringsproblem er diæt-problemet, hvor en person ønsker at få opfyldt sit daglige optag af næringsstoffer til den mindste pris [5]. Hver vare har en pris pr. enhed af varen, og personen ønsker at finde ud af, hvor mange enheder denne skal købe af hvert produkt, for at det bliver billigst muligt og det daglige optag af næringsstoffer stadig er opfyldt. Det daglige optag af næringsstoffer er begrænset, altså er der et krav om minimumsindtag. I problemet har vi så fået givet nogle forskellige madvarer, som hver indeholder en bestemt mængde af de pågældende næringsstoffer pr. enhed af varen, sådan at hvis vi spiser mere af den pågældende vare, så får vi tilsvarende mere af næringsstoffet pr. enhed af varen. Heraf følger en intuitiv forståelse for, at mængden af næringsstoffer man indtager stiger ligefrem proportionalt med mængden af mad, man indtager.

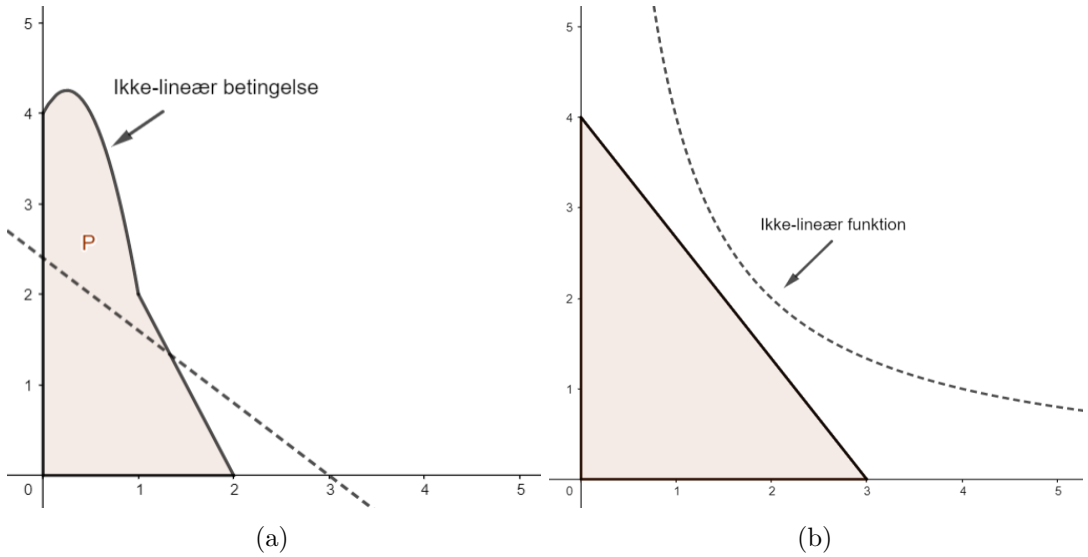
En anden måde at anskue lineære programmeringsproblemer på er ved, at virksomheder generelt ønsker at finde måder at maksimere deres profit på eller måder at minimere omkostninger til produktion eller transport på, og til dette er det muligt at anvende lineær programmering [5]. Hvis en virksomhed ønsker at optimere deres fortjeneste på et bestemt antal produkter eksempelvis i form af borde og stole, kan det undersøges, hvordan profitten optimeres for disse produkter i forhold til nogle begrænsninger i form af

eksempelvis materialer, arbejdskraft og lagerplads. Dette er netop området, vi bevæger os inden for med lineær programmering. Tilsvarende hvis virksomheden ønsker at minimere deres omkostninger til produktionen, kan det undersøges, hvordan materialet af borde og stole bruges optimalt igen i forhold til nogle betingelser som tilgængeligt materiale, arbejdskraft og lagerplads. Lineær programmering dækker så over en metode til at finde den sammensætning af produktionen, hvor virksomheden får mest ud af prisen, men hvor begrænsningerne samtidig er opfyldt. Med andre ord ønsker virksomheden at finde frem til produktionen af det antal af produkter, som skaber det største overskud ud fra mængden af begrænsninger. Intuitivt forstås det også, at man ikke kan sælge eller producere et negativt antal produkter, så en generel begrænsning for de fleste lineære programmeringsproblemer er, at alle variablene skal være ikke-negative. Dette er kun nogle få eksempler på, hvad lineær programmering kan bruges til.

Det er interessant, at lineær programmering forholdvis sent er blevet en essentiel del af løsningsmetoder for optimeringsproblemer, og projektet ønsker at undersøge anvendeligheden af lineær programmering, og mere specifikt Simplex-metoden, som er en konkret løsningsmetode til lineære programmeringsproblemer, som undersøger et problem systematisk for at finde frem til den optimale løsning [6].

1.3 Problemer ved anvendelsen af lineær programmering

I og med at lineær programmering skal kunne løse problemer på lineær form, er det naturligt afgrænset fra at kunne løse nogle typer af problemer. Det er sjældent, at praktiske problemer opfører sig lineært, og ofte vil der opstå problemer på ikke-lineære former. Dette sker eksempelvis, hvis det vi ønsker at optimere ikke kan beskrives lineært, eller hvis betingelserne for problemet viser sig ikke at kunne beskrives lineært. Grafisk kan man vise dette ved figur 1.2.



Figur 1.2: *(a) Problem der ikke er lineært på grund af ikke-lineære betingelser. (b) Problem der ikke er lineært på grund af en ikke-lineær funktion som ønskes optimeret.*

For at kunne anvende lineær programmering til at løse problemstillinger, der i virkeligheden er ikke-lineære, er man derfor nødt til at modellere problemerne som lineære, hvorfor

man skal være opmærksom på, hvor meget resultatet giver mening i forhold til det praktiske problem, når man har foretaget denne modellering.

Alt i alt leder ovenstående beskrivelser nu frem til opstillingen af den overordnede problemformulering for dette P2-projekt.

1.4 Problemformulering

Hvorfor virker lineær programmering og mere specifikt Simplex-metoden, og for hvilke optimeringsproblemer giver det mening at anvende Simplex-metoden?

1.5 Læsevejledning

I dette projekt vil fokus være at undersøge, hvorfor Simplex-metoden virker, og for hvilke problemer det giver mening at bruge den. For at kunne forklare dette er det nødvendigt at beskrive væsentlige dele af lineær algebra, og hvordan vi kan bruge det til at beskrive lineære programmeringsproblemer og baggrunden for Simplex-metoden. Det vil derfor også blive forklaret nærmere, hvad lineær programmering er, og hvilken geometrisk og algebraisk forståelse i forhold til konveksitet der ligger til grund for sådanne problemer og for funktionaliteten af Simplex-metoden. Mere specifikt i forhold til Simplex-metoden vil det blive gennemgået, hvordan metoden er bygget op, og hvordan den fungerer som algoritmisk implementation. Dertil vil Simplex-metoden blive anvendt på en konkret problemstilling i form af diæt-problemet, hvorefter relevante diskussioner og perspektiveringer vil blive taget op for slutteligt at konkludere på projektet ud fra den opstillede problemformulering.

2 | Introduktion til lineær algebra

Dette kapitel har til formål at introducere *matricer*, *vektorer* og basal aritmetik inden for lineær algebra, der er fundamentet for analysen af de senere optimeringsproblemer i dette projekt. Kapitlet tager udgangspunkt i [7, p. 3-250], medmindre andet fremgår af teksten.

2.1 Matricer og vektorer

Definition 2.1.1. ***Matricer***
En matrix A er en rektangulær tabel. *Størrelsen* af en matrix siges at være $m \times n$, hvis matricen består af m horisontale *rækker* og n vertikale *søjler*. Elementer i en matrix kaldes også for *indgange*. En indgang i den i'te horisontale række og j'te vertikale søjle noteres som a_{ij} for $i \in \{1, 2, \ldots, m\}$ og $j \in \{1, 2, \ldots, n\}$.

Bemærk af ovenstående definition at matricer generelt noteres med store bogstaver, mens indgange noteres med tilsvarende små bogstaver. Bemærk også at positionen af en indgang a_{ij} findes ved først at finde den i'te række og derefter den j'te søjle, som indgangen tilhører. Der gælder endvidere, at $a_{ij} \in \mathbb{R}$. En generel skitsering af en matrix A ses i figur 2.1.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

Figur 2.1: *En generel skitsering af en $m \times n$ matrix A med indgange a_{ij}.*

Eksempel 2.1.2. ***Forskellige matricer***
På figur 2.2 findes eksempler på 3 forskellige matricer A, B og C.

$$A = \begin{bmatrix} 2 & 1 & 5 \\ 3 & 1 & 3 \end{bmatrix} \quad \text{(a)} \qquad B = \begin{bmatrix} 2 & 3 \\ 1 & 1 \\ 5 & 3 \end{bmatrix} \quad \text{(b)} \qquad C = \begin{bmatrix} 2 & 4 & 6 & 8 \\ 1 & 3 & 5 & 7 \\ 10 & 5 & 1 & 7 \\ 1 & 4 & 2 & 2 \end{bmatrix} \quad \text{(c)}$$

Figur 2.2: *3 forskellige matricer.*

Her er A en 2×3 matrix, B er en 3×2 matrix og C er en 4×4 matrix. Bemærk her at C er et eksempel på en matrix, hvor $m = n$, hvorfor den kaldes for en *kvadratisk* matrix. Indgangene i matricerne kan beskrives ved hjælp af notationen fra definition 2.1.1, så a_{12} svarer til indgangen i første række og anden søjle i matrix A med værdien 1. Tilsvarende

vil c_{31} svare til indgangen i tredje række og første søjle i matrix C med værdien 10.

En basal aritmetisk operation for matricer er *skalar multiplikation*, noteret som kA, hvor $k \in \mathbb{R}$. kA er da defineret som en $m \times n$ matrix, hvor alle indgange er blevet multipliceret med k. Fra eksempel 2.1.2 kan et skalar multiplum for $k = -4$ af matricen B derfor bestemmes som

$$-4B = -4\begin{bmatrix} 2 & 3 \\ 1 & 1 \\ 5 & 3 \end{bmatrix} = \begin{bmatrix} -4 \cdot 2 & -4 \cdot 3 \\ -4 \cdot 1 & -4 \cdot 1 \\ -4 \cdot 5 & -4 \cdot 3 \end{bmatrix} = \begin{bmatrix} -8 & -12 \\ -4 & -4 \\ -20 & -12 \end{bmatrix} \tag{2.1}$$

Definition 2.1.3. ***Transponering af matricer***
Givet en $m \times n$ matrix A er *transponeringen* af denne matrix noteret som $A^\intercal$ en $n \times m$ matrix, hvorom det gælder, at indgangen a_{ij} i $A^\intercal$ matrix svarer til indgangen a_{ji} i A.

Bemærk af ovenstående definition at transponeringen af en matrix A findes ved at vende rækkerne i A, så den i'te række i A svarer til den j'te søjle i $A^\intercal$ og tilsvarende omvendt. Af eksempel 2.1.2 ser vi derfor, at $A = B^\intercal$, hvorfor B er transponeringen af A og ligeledes er $A^\intercal = B$, hvor A er transponeringen af B .

For *summen af matricer* gælder, at matricerne skal have samme størrelse $m \times n$, og at summen noteret som $A + B$ da giver en ny $m \times n$ matrix, hvis (ij)-indgange svarer til $a_{ij} + b_{ij}$. Bemærk af eksempel 2.1.2 at matricerne A og B har forskellige størrelser, hvorfor summen af dem ikke er defineret. Men som bemærket kan matricen B transponeres til en ny 2×3 matrix $B^\intercal$ med samme størrelse som A, hvorfor summen af disse to matricer nu er givet ved

$$A + B^\intercal = \begin{bmatrix} 2 & 1 & 5 \\ 3 & 1 & 3 \end{bmatrix} + \begin{bmatrix} 2 & 1 & 5 \\ 3 & 1 & 3 \end{bmatrix} = \begin{bmatrix} 2+2 & 1+1 & 5+5 \\ 3+3 & 1+1 & 3+3 \end{bmatrix} = \begin{bmatrix} 4 & 2 & 10 \\ 6 & 2 & 6 \end{bmatrix}.$$

Vi er nu kommet til vektorer, som er en speciel form for matricer, hvorfor disse nu defineres.

Definition 2.1.4. ***Vektorer***
En vektor $\mathbf{a}$ med n indgange kan enten være givet som en $n \times 1$ matrix benævnt som en *søjle-vektor* eller som en $1 \times n$ matrix benævnt som en *række-vektor*. En indgang a_i i en vektor benævnes også som en *komponent*. En vektor med n komponenter siges at tilhøre $\mathbb{R}^n$.

Bemærk af definition 2.1.4 at vektorer noteres med et lille bogstav med fed for unikt at illustrere, at det er vektoren, der henvises til, mens indgange i den samme vektor benævnes med a_i. Desuden bemærkes det, at vi noterer en søjle-vektor som $\mathbf{a} = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix}^\intercal$ i teksten gennem resten af projektet.

Eksempel 2.1.5. ***En søjle-vektor og en række-vektor***
I figur 2.3 er to vektorer givet; henholdsvis vektoren $\mathbf{a}$ som en søjle-vektor og vektoren $\mathbf{b}$ som en række-vektor.

$$\mathbf{a} = \begin{bmatrix} 2 \\ 3 \\ 7 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 2 & 3 & 7 \end{bmatrix}$$

(a) (b)

Figur 2.3: *(a) En søjle-vektor og (b) en række-vektor.*

I relation til matricer bemærkes det, at $\mathbf{a}$ er en 3×1 matrix, $\mathbf{b}$ er en 1×3 matrix og begge tilhører $\mathbb{R}^3$.

I forbindelse med matricer er det nyttigt at kunne beskrive hver søjle som en vektor, hvorfor en søjle-vektor i den j'te søjle af en $m \times n$ matrix A noteres som

$$\mathbf{A_j} = \begin{bmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{mj} \end{bmatrix} \tag{2.2}$$

for $j \in \{1, 2, \ldots, n\}$ [6, p. 27]. Tilsvarende noteres en vektor $\mathbf{a_i}$ som en søjle-vektor med den i'te rækkes indgange i matricen A. Der gælder derfor, at

$$\mathbf{a_i} = \begin{bmatrix} a_{i1} \\ a_{i2} \\ \vdots \\ a_{in} \end{bmatrix} \tag{2.3}$$

for $i \in \{1, 2, \ldots, m\}$[6, p. 27]. Bemærk at denne vektor noteres som $\mathbf{a_i}^\intercal$, hvis vi ønsker vektoren som en række-vektor taget fra matricen A.

Vektorer i $\mathbb{R}^n$ på formen

$$\mathbf{e_1} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \mathbf{e_2} = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix} \quad \cdots \quad \mathbf{e_n} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \tag{2.4}$$

kaldes *standard-vektorer*, og sammensættes de n standard-vektorer fra $\mathbb{R}^n$ i en $n \times n$ matrix noteret som I_n er dette *identitets-matricen* givet ved

$$I_n = \begin{bmatrix} \mathbf{e_1} & \mathbf{e_2} & \cdots & \mathbf{e_n} \end{bmatrix}. \tag{2.5}$$

Definition 2.1.6. ***Linearkombinationer***
Givet vektorer $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}$ i $\mathbb{R}^n$ og skalarer $c_1, c_2, \ldots, c_k \in \mathbb{R}$ er en *linearkombination* af vektorerne givet ved en ny vektor på formen

$$c_1\mathbf{v_1} + c_2\mathbf{v_2} + \cdots + c_k\mathbf{v_k}. \tag{2.6}$$

Bemærk at resultatet af en linearkombination af vektorer i $\mathbb{R}^n$ er en ny vektor i $\mathbb{R}^n$. Derfor giver det kun mening at snakke om linearkombinationer mellem vektorer, hvor antallet af komponenter er det samme, da additionen af vektorerne ellers ikke er defineret.

Definition 2.1.7. ***Matrix-vektor produkt***
Givet en $n \times 1$ vektor $\mathbf{v}$ og en $m \times n$ matrix A er *matrix-vektor produktet* af $\mathbf{v}$ og A en linearkombination af vektorens komponenter som koefficienter til hver søjle-vektor $\mathbf{A_j}$ i A. Matrix-vektor produktet noteres som $A\mathbf{v}$ og er derfor givet ved

$$A\mathbf{v} = v_1\mathbf{A_1} + v_2\mathbf{A_2} + \cdots + v_n\mathbf{A_n} \tag{2.7}$$

Bemærk af definition 2.1.7 at matrix-vektor produktet $A\mathbf{v}$ kun er defineret, hvis antallet af komponenter i vektoren $\mathbf{v}$ svarer til antallet af søjler i matrix A, hvilket vil resultere i en vektor i $\mathbb{R}^m$.

Eksempel 2.1.8. ***Et matrix-vektor produkt***
På figur 2.4 er der givet en 3×3 matrix A og en 3×1 vektor $\mathbf{v}$.

$$A = \begin{bmatrix} -1 & 5 & 4 \\ -3 & -4 & 3 \\ 6 & 1 & -2 \end{bmatrix} \qquad \mathbf{v} = \begin{bmatrix} 7 \\ 3 \\ -3 \end{bmatrix}$$

(a) (b)

Figur 2.4: *(a) En* 3×3 *matrix* A *og (b) en* 3×1 *vektor* $\mathbf{v}$ *.*

Matrix-vektor produktet $A\mathbf{v}$ er nu givet ved linearkombinationen af søjle-vektorerne i A og komponenterne i vektor $\mathbf{v}$ som

$$A\mathbf{v} = \begin{bmatrix} -1 & 5 & 4 \\ -3 & -4 & 3 \\ 6 & 1 & -2 \end{bmatrix} \begin{bmatrix} 7 \\ 3 \\ -3 \end{bmatrix} = 7 \begin{bmatrix} -1 \\ -3 \\ 6 \end{bmatrix} + 3 \begin{bmatrix} 5 \\ -4 \\ 1 \end{bmatrix} - 3 \begin{bmatrix} 4 \\ 3 \\ -2 \end{bmatrix} = \begin{bmatrix} -4 \\ -42 \\ 51 \end{bmatrix} \tag{2.8}$$

Vi nævner her nogle vigtige regneregler for matrix-vektor produkter uden beviser, hvor der henvises til [7, p. 24] for beviserne for disse.

2.1.9. Regneregler for matrix-vektor produkter
Vi lader A og B være matricer af størrelsen $m \times n$ og $\mathbf{u}$ og $\mathbf{v}$ være vektorer i $\mathbb{R}^n$. Tilsvarende lader vi $k \in \mathbb{R}$. Da gælder så, at

1. $A(\mathbf{u} \pm \mathbf{v}) = A\mathbf{u} \pm A\mathbf{v}$
2. $A(k\mathbf{u}) = k(A\mathbf{u})$
3. $(A \pm B)\mathbf{u} = A\mathbf{u} \pm B\mathbf{u}$
4. $A\mathbf{e_j} = \mathbf{A_j}$ hvor $j \in \{1, 2, \ldots, n\}$ og $\mathbf{e_j}$ er den j'te standard-vektor i $\mathbb{R}^n$
5. hvis $B\mathbf{w} = A\mathbf{w}$ for alle $\mathbf{w}$ i $\mathbb{R}^n$, så er $B = A$
6. $I_n\mathbf{u} = \mathbf{u}$.

2.2 Lineære ligningssystemer og rækkereduktioner

Definition 2.2.1. ***Lineære ligningssystemer***
Givet m ligninger med n forskellige variable er et *lineært ligningssystem* givet ved

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m, \end{aligned} \tag{2.9}$$

hvor $a_{ij}, b_i \in \mathbb{R}$ og x_j er *variable* for $i \in \{1, 2, \ldots, m\}$ og $j \in \{1, 2, \ldots, n\}$. En løsning til det lineære ligningssystem er da en vektor $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^\intercal$. Forskellige lineære ligningssystemer er *ækvivalente*, hvis de har samme *løsningsmængde*.

I forhold til definition 2.2.1 er et lineært ligningssystem *konsistent*, hvis der eksisterer mindst én vektor $\mathbf{x}$, der løser ligningssystemet. Hvis ingen løsninger findes, er det lineære ligningssystem *inkonsistent*. Det lineære ligningssystem som præsenteret i definition 2.2.1 kan tilsvarende beskrives som et matrix-vektor produkt ud fra ligningerne i 2.9 givet ved

$$A\mathbf{x} = \mathbf{b} \implies \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \tag{2.10}$$

Bemærk da af matrix-vektor produktet i ligning 2.10 at $m \times n$ matricen A med rette kaldes for *koefficient-matricen* til det lineære ligningssystem, da indgangene a_{ij} i A svarer til koefficienter for x_j for $j \in \{1, 2, \ldots, n\}$. Ligeledes indføres en $m \times (n+1)$ *total-matrix* noteret som $\begin{bmatrix} A & \mathbf{b} \end{bmatrix}$ der indeholder koefficient-matricen A samt vektoren $\mathbf{b}$ som søjle $n+1$ i $\begin{bmatrix} A & \mathbf{b} \end{bmatrix}$.

Eksempel 2.2.2. ***Et lineært ligningssystem***
Et lineært ligningssystem er givet ved

$$\begin{aligned} -2x_1 + 3x_2 \qquad - 7x_4 &= 3 \\ -x_2 + x_3 + x_4 &= 1 \\ 6x_1 \qquad - 2x_3 \qquad &= 0 \end{aligned} \tag{2.11}$$

Koefficient-matricen A og total-matricen $\begin{bmatrix} A & \mathbf{b} \end{bmatrix}$ for det lineære ligningssystem 2.11 er illustreret i figur 2.5.

$$A = \begin{bmatrix} -2 & 3 & 0 & -7 \\ 0 & -1 & 1 & 1 \\ 6 & 0 & -2 & 0 \end{bmatrix} \qquad \begin{bmatrix} A & \mathbf{b} \end{bmatrix} = \begin{bmatrix} -2 & 3 & 0 & -7 & 3 \\ 0 & -1 & 1 & 1 & 1 \\ 6 & 0 & -2 & 0 & 0 \end{bmatrix}$$

(a) (b)

Figur 2.5: *(a) Koefficient-matricen A og (b) total-matricen* $\begin{bmatrix} A & \mathbf{b} \end{bmatrix}$ *for det lineære ligningssystem 2.11 .*

Bemærk af eksemplet at $a_{ij} = 0$, hvis x_j ikke optræder i den i'te ligning i både koefficient-matricen og total-matricen.

Vi definerer nu de *tre elementære rækkeoperationer*.

Definition 2.2.3. ***De tre elementære rækkeoperationer***
De tre elementære rækkeoperationer som kan udføres på en matrix A for at opnå en anden matrix B er defineret ved

1. *Rækkeombytning* - enhver række r_i kan byttes med en anden række r_j (noteret som $\overset{r_i \leftrightarrow r_j}{\longrightarrow}$).
2. *Skalar rækkemultiplikation* - i en række r_i multipliceres hver indgang med det samme tal $k \neq 0$ (noteret som $\overset{kr_i \to r_i}{\longrightarrow}$).
3. *Rækkeaddition* - i en række r_i multipliceres hver indgang med det samme tal $k \neq 0$ og lægges til en anden række r_j (noteret som $\overset{kr_i + r_j \to r_j}{\longrightarrow}$).

Bemærk at de tre elementære rækkeoperationer ikke ændrer på løsningsmængden af total-matricen $\begin{bmatrix} A & \mathbf{b} \end{bmatrix}$ for et lineært ligningssystem, og bemærk desuden at vi i forbindelse med rækkeoperationer anvender notationen r_i for rækker i en matrix for at gøre operationerne mere overskuelige.

Eksempel 2.2.4. ***Elementære rækkeoperationer på en matrix*** A
Givet en matrix A og en matrix B i figur 2.6 kan elementære rækkeoperationer anvendes for at opnå B ud fra A (og ligeså omvendt).

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & \frac{3}{2} & \frac{1}{2} \\ 1 & -4 & 0 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 & 6 & 3 \\ 0 & 2 & 3 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Figur 2.6: *To matricer A og B.*

Med rækkeoperationerne i 2.12 kan A derfor omformes til B.

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & \frac{3}{2} & \frac{1}{2} \\ 1 & -4 & 0 & 1 \end{bmatrix} \overset{2r_2 \to r_2}{\longrightarrow} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2 & 3 & 1 \\ 1 & -4 & 0 & 1 \end{bmatrix} \overset{r_1 \leftrightarrow r_3}{\longrightarrow} \begin{bmatrix} 1 & -4 & 0 & 1 \\ 0 & 2 & 3 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \overset{2r_2 + r_1 \to r_1}{\longrightarrow} \begin{bmatrix} 1 & 0 & 6 & 3 \\ 0 & 2 & 3 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.12}$$

De 3 elementære rækkeoperationer er grundlæggende for mange af de aritmetiske operationer, der vil blive udført på matricer senere i projektet. Vi introducerer nu et vigtigt begreb, som får betydning for den kommende definition. *En ledende indgang* er den første indgang forskellig fra 0 i en række set fra venstre. Vi er så klar til at definere *trappeform* og *reduceret trappeform*.

Definition 2.2.5. ***Trappeform og reduceret trappeform***
En matrix A er på trappeform, hvis følgende gælder.

1. Rækker hvor alle indgange er 0, betegnet som *nul-rækker*, ligger nederst i matricen, så eventuelle rækker med indgange forskellige fra 0 er placeret over.
2. Den ledende indgang i en række er placeret til højre for den ovenstående rækkes ledende indgang.
3. Hvis en række har en ledende indgang, vil indgangene i rækkerne under i samme søjle være 0.

Matricen A er på reduceret trappeform, hvis der yderligere gælder følgende.

4. I tilfælde af at en søjle indeholder en ledende indgang for en given række, er alle andre indgange i samme søjle 0.
5. Alle ledende indgange er 1.

Vi benævner generelt en matrix på reduceret trappeform som R.

Bemærk at positionen af en ledende indgang i en matrix R på reduceret trappeform kaldes for en *pivot-position*, hvor den tilsvarende indgang i A også er en pivot-position. Tilsvarende kaldes en søjle indeholdende en pivot-position for en *pivot-søjle*. Det betyder, at den oprindelige matrix A også har pivot-søjler i tilsvarende søjler som for R på reduceret trappeform. Bemærk desuden at løsningsmængden af det oprindelige lineære ligningssystem givet ved total-matricen $\begin{bmatrix} A & \mathbf{b} \end{bmatrix}$ er bevaret ved total-matricen $\begin{bmatrix} R & \mathbf{k} \end{bmatrix}$ på reduceret trappeform, da vi udelukkende anvender elementære rækkeoperationer på $\begin{bmatrix} A & \mathbf{b} \end{bmatrix}$ for at opnå $\begin{bmatrix} R & \mathbf{k} \end{bmatrix}$.

Eksempel 2.2.6. ***Trappeform og reduceret trappeform***
På figur 2.7 er givet to matricer, hvor A er på trappeform og R er på reduceret trappeform.

$$A = \begin{bmatrix} 2 & 2 & 3 & 6 \\ 0 & 1 & -1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad R = \begin{bmatrix} 1 & 0 & 0 & -\frac{1}{2} \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Figur 2.7: *To matricer A og R hvor A er på trappeform og R er på reduceret trappeform.*

Ud fra de tre første elementer i definition 2.2.5 bemærkes det, at matricen A på figur 2.7 er på trappeform, da række 4, som er en nul-række, befinder sig nederst i matricen. Hver ledende indgang i hverenkelt række er placeret til højre for den ovenstående rækkes ledende indgang, mens indgangene under den ledende indgang i hver pivot-søjle er 0. Bemærk at indgang $a_{11} = 2 \neq 1$, hvor definition 2.2.5 ikke kræver, at hver ledende indgang skal være 1, så A er stadig på trappeform. Matricen R på figur 2.7 er på reduceret trappeform, da den opfylder alle fem betingelser i definition 2.2.5. Kravet om at indgangen r_{22} eksempelvis er den eneste indgang i 2. søjle forskellig fra 0 er opfyldt, så $r_{12} = r_{32} = r_{42} = 0$. Det ses endvidere, at $r_{22} = 1$, så andet krav for den reducerede trappeform er også opfyldt for dette eksempel. Det samme gælder for de resterende pivot-søjler. Derfor er matrix R på figur 2.7 på reduceret trappeform.

Trappeform og reduceret trappeform har en stor anvendelse i forhold til at skulle gennemskue, hvor mange løsninger et bestemt lineært ligningssystem har. Hvis matrix A på figur 2.7 som eksempel er en total-matrix for et lineært ligningssystem, betyder det, at det lineære ligningssystem består af fire ligninger ud fra antallet af rækker, hvor de tre første søjler viser koefficientmatricen af størrelse 4×3, mens den sidste søjle er et udtryk for, hvad kombinationen af variablene skal være lig med, altså en vektor af størrelse 4×1. Hvis en søjle er en pivot-søjle i koefficient-matricen, vil den tilhørende variabel til koefficienterne i denne søjle betegnes som en *bunden variabel*, mens variable der hører til ikke-pivot-søjler kaldes for *frie variable*.

I forhold til antallet af løsninger til et givet lineært ligningssystem gælder da følgende.

1. Det lineære ligningssystem er inkonsistent, hvis den sidste søjle i total-matricen er en pivot-søjle.
2. Det lineære ligningssystem er konsistent og har én løsning, hvis alle søjler på nær den sidste søjle i total-matricen er pivot-søjler. Altså er samtlige variable bundne.
3. Det lineære ligningssystem er konsistent og har uendeligt mange løsninger, hvis der findes én eller flere frie variable.

Bemærk her at argumentet for et inkonsistent lineært ligningssystem kommer af, at total-matricen $\begin{bmatrix} R & \mathbf{k} \end{bmatrix}$ på reduceret trappeform i tilfælde af pivot-søjle i sidste søjle medfører en ligning på formen

$$r_{m1}x_1 + r_{m2}x_2 + \cdots + r_{mn}x_n = k_m, \tag{2.13}$$

hvor $r_{m1} = r_{m2} = \cdots = r_{mn} = 0$ og $k_m \neq 0$. Uanset værdien af variablene $x_1, x_2, \ldots, x_n$ vil venstresiden i ligning 2.13 blive 0, hvorfor ligheden ikke er opfyldt, og der eksisterer derfor ikke en løsning til det givne lineære ligningssystem. I forhold til argumentet for at et lineært ligningssystem har præcis én løsning bemærkes det, at de bundne variable er de variable, som man løser for, når man skal beskrive en generel løsning for et lineært ligningssystem. Da fremgår det tydeligt, at hvis der ingen frie variable eksisterer, vil total-matricen på reduceret trappeform være på formen

$$\begin{bmatrix} R & \mathbf{k} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 & k_1 \\ 0 & 1 & \cdots & 0 & k_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & k_m \end{bmatrix}, \tag{2.14}$$

hvor $k_1, k_2, \ldots, k_m \in \mathbb{R}$, så $x_1 = k_1, x_2 = k_2, \ldots, x_m = k_m$ for løsningen til det lineære ligningssystem. Modsat gælder det i argumentet for, at et lineært ligningssystem har uendeligt mange løsninger, da en generel løsning for de bundne variable afhænger af én eller flere frie variable, hvor de frie variable kan vælges vilkårligt, så ligningssystemet stadig er opfyldt. Det medfører derfor eksistensen af et uendeligt antal løsninger.

Der introduceres nu to begreber, der kan være med til at beskrive antallet af pivot-søjler i en given matrix.

Definition 2.2.7. ***Rang og nullitet af matricer***
Givet en $m \times n$ matrix A er *rangen* af A defineret som antallet af ikke-nul rækker i den reducerede trappeform af A, og det noteres som rang A. *Nulliteten* af samme matrix er da defineret som $n-$rang A.

Bemærk at der findes en simpel sammenhæng mellem rang og nullitet af en matrix og antallet af pivot-søjler i den samme matrix.

Sætning 2.2.8. Givet en $m \times n$ matrix A er rang$A$ det samme som antallet af pivot-søjler i A. For samme matrix gælder, at nulliteten af A er antallet af ikke-pivot-søjler.

Bevis. Beviset følger af definition 2.2.7, da der i ikke-nul-rækker i en $m \times n$ matrix A på reduceret trappeform eksisterer en pivot-position i hver af disse rækker. Vi benævner her sådanne rækker som *pivot-rækker*. Ud fra definition 2.2.5 vil det første tal $a_{ij} = 1$ i den i'te pivot-række da være den eneste indgang forskellig fra 0 i den j'te pivot-søjle, for ellers er matricen ikke på reduceret trappeform. Det beviser derfor, at antallet af ikke-nul-rækker i matricen på reduceret trappeform svarer til antallet af pivot-søjler. Definitionen af nulliteten i 2.2.7 medfører så, at nulliteten svarer til differencen mellem antal søjler n og antal pivot-søjler givet ved rang$A$, hvilket beviser, at nulliteten er lig med antallet af ikke-pivot-søjler. ■

2.3 Lineær afhængighed og lineær uafhængighed

Definition 2.3.1. ***Lineær afhængighed og lineær uafhængighed***
Givet en mængde af k vektorer $\{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}\} \subseteq \mathbb{R}^n$ siges vektorerne at være *lineært uafhængige*, hvis ligningen

$$k_1\mathbf{v_1} + k_2\mathbf{v_2} + \cdots + k_k\mathbf{v_k} = \mathbf{0}, \tag{2.15}$$

kun har løsningen $k_1 = k_2 = \cdots = k_k = 0$. Hvis ligning 2.15 modsat har en løsning hvor mindst ét $k_i \neq 0$ for $i \in \{1, 2, \ldots, k\}$, siges de k vektorer $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}$ at være *lineært afhængige*.

Bemærk at enhver mængde af vektorer hvor nulvektoren er indeholdt er lineært afhængige, da

$$k_1\mathbf{0} + k_2\mathbf{v_2} + \cdots + k_k\mathbf{v_k} = \mathbf{0}, \tag{2.16}$$

hvor vektorerne $\mathbf{v_2}, \mathbf{v_3}, \ldots, \mathbf{v_k}$ kan vælges vilkårligt for $k_1 \neq 0$ og $k_2 = \cdots = k_k = 0$, så ligning 2.16 stadig er opfyldt.

Den følgende sætning samler ækvivalente påstande om, hvornår en mængde af vektorer er lineært uafhængige.

Sætning 2.3.2. Givet en $m \times n$ matrix A er nedenstående udsagn vedrørende søjle-vektorerne i A ækvivalente.

1. Søjle-vektorerne i A er lineært uafhængige.
2. Rang A svarer til antallet af søjler n.
3. Nulliteten af A er 0.
4. Søjle-vektorerne i den reducerede trappeform R af A er standard-vektorer $\mathbf{e_1}, \mathbf{e_2}, \ldots, \mathbf{e_n} \in \mathbb{R}^m$.
5. $A\mathbf{x} = \mathbf{0}$ har kun løsningen $\mathbf{x} = \mathbf{0}$.
6. Ligningssystemet $A\mathbf{x} = \mathbf{b}$ har højst én løsning for enhver vektor $\mathbf{b} \in \mathbb{R}^m$.

Bevis. Vi beviser denne sætning ved at kigge på ækvivalensen af de enkelte udsagn. Bemærk da først at (4) $\implies$ (5), da $R = \begin{bmatrix} \mathbf{e_1} & \mathbf{e_2} & \cdots & \mathbf{e_n} \end{bmatrix}$ medfører jævnfør regel 6 i 2.1.9, at den eneste løsning til $R\mathbf{x} = \mathbf{0}$ er $\mathbf{x} = \mathbf{0}$, fordi

$$x_1\mathbf{e_1} + x_2\mathbf{e_2} + \ldots + x_n\mathbf{e_n} = \mathbf{0} \tag{2.17}$$

kun er opfyldt for $x_1 = x_2 = \cdots = x_n = 0$. Da R er opnået fra A ved brug af elementære rækkeoperationer, vil $R\mathbf{x} = \mathbf{0}$ og $A\mathbf{x} = \mathbf{0}$ have den samme løsningsmængde, så der tilsvarende gælder, at $A\mathbf{x} = \mathbf{0}$ kun har løsningen $\mathbf{x} = \mathbf{0}$.

For at vise at (5) $\implies$ (6) antages det, at $\mathbf{u}$ og $\mathbf{v}$ begge er løsninger til $A\mathbf{x} = \mathbf{b}$, så $A\mathbf{u} = A\mathbf{v} = \mathbf{b}$. Der gælder så, at

$$A\mathbf{u} - A\mathbf{v} = \mathbf{b} - \mathbf{b} = \mathbf{0} \tag{2.18}$$

Da $A\mathbf{u} - A\mathbf{v} = A(\mathbf{u} - \mathbf{v})$ jævnfør 2.1.9 regel 1 betyder det, at $\mathbf{u} - \mathbf{v}$ er en løsning til $A\mathbf{x} = \mathbf{0}$, hvor den eneste løsning er $\mathbf{0}$, så $\mathbf{u} - \mathbf{v} = \mathbf{0}$, og derfor gælder at $\mathbf{u} = \mathbf{v}$. $A\mathbf{x} = \mathbf{b}$ har derfor kun én løsning for enhver vektor $\mathbf{b} \in \mathbb{R}^m$.

(6) $\implies$ (2) da der ikke eksisterer frie variable, når $A\mathbf{x} = \mathbf{b}$ højst har én løsning. Da er der pivot i alle søjler i A, hvilket betyder, at rang $A = n$.

(2) $\iff$ (3) da nulliteten er $n-$rang A, hvor rang $A = n$, så $n - n = 0$. Der gælder derfor tilsvarende, at hvis nulliteten er 0, så er rang $A = n$.

(2) $\implies$ (4) da R har en ledende indgang som er 1 i hver søjle, hvis der er pivot i alle søjler i A. Derfor vil søjlerne i R være standard-vektorer $\mathbf{e_1}, \mathbf{e_2}, \ldots, \mathbf{e_n} \in \mathbb{R}^m$.

(5) $\iff$ (1) da den eneste løsning til $A\mathbf{x} = \mathbf{0}$ er $\mathbf{x} = \mathbf{0}$, hvorfor søjlerne i A er lineært uafhængige jævnfør linearkombinationen i ligning 2.15 i definition 2.3.1.

■

2.4 Lineære afbildninger

Vi definerer nu en vigtig type af funktioner kaldet *lineære afbildninger*, som har en vigtig rolle for den senere definition af lineære programmeringsproblemer.

Definition 2.4.1. ***Lineære afbildninger***
En funktion $f : \mathbb{R}^n \to \mathbb{R}^m$ kaldes en lineær afbildning, hvis den bevarer vektoraddition og skalar multiplikation for alle vektorer $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ og alle $k \in \mathbb{R}$. Der gælder derfor, at

1. $f(\mathbf{x} + \mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})$
2. $f(k\mathbf{x}) = kf(\mathbf{x})$

To vigtige egenskaber for en lineær afbildning $f(\mathbf{x})$ som er bevist i [7, p. 171] er, at $f(\mathbf{0}) = \mathbf{0}$ og $f(a\mathbf{x} + b\mathbf{y}) = af(\mathbf{x}) + bf(\mathbf{y})$ for alle vektorer $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ og for alle $a, b \in \mathbb{R}$, hvilket kommer til at spille en stor rolle for forståelsen af lineære programmeringsproblemer senere, hvor funktionen vi ønsker at optimere skal være en lineær afbildning [8, p. 70].

2.5 Span af vektorer, underrum og baser

Definition 2.5.1. ***Span***
Givet en ikke-tom mængde $S = \{\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_k}\} \subseteq \mathbb{R}^n$ siges det, at Span af S svarer til mængden af alle linearkombinationer af $\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_k}$ i $\mathbb{R}^n$. Vi noterer desuden denne mængde som Span S eller Span$\{\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_k}\}$.

For en mængde $S = \{\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_k}\} \subseteq \mathbb{R}^n$ kan vi for en vektor $\mathbf{v} \in \mathbb{R}^n$ undersøge, om $\mathbf{v} \in$ Span S ved at kigge på matricen A, hvis søjler er vektorerne i S. Da gælder der, at vektoren $\mathbf{v}$ er i Span S, hvis ligningssystemet $A\mathbf{x} = \mathbf{v}$ er konsistent.

Eksempel 2.5.2. ***Vektorer i et Span***
I figur 2.8 er givet en mængde af vektorer S samt to vektorer $\mathbf{u}, \mathbf{v} \in \mathbb{R}^4$.

$$S = \left\{ \begin{bmatrix} 1 \\ 2 \\ -1 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ -1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 2 \\ 0 \\ 3 \end{bmatrix} \right\} \quad \mathbf{u} = \begin{bmatrix} 4 \\ 0 \\ 5 \\ 8 \end{bmatrix} \quad \mathbf{v} = \begin{bmatrix} -1 \\ 2 \\ -3 \\ 5 \end{bmatrix}$$

Figur 2.8: *En mængde af vektorer i $\mathbb{R}^4$ givet ved S og to vektorer $\mathbf{u}, \mathbf{v} \in \mathbb{R}^4$.*

Vi vil nu undersøge, om $\mathbf{u}$ og $\mathbf{v}$ ligger i Span S. Vi undersøger derfor, om ligningssystemerne $A\mathbf{x} = \mathbf{u}$ og $A\mathbf{x} = \mathbf{v}$ er konsistente, hvor

$$A = \begin{bmatrix} 1 & 2 & -1 \\ 2 & -1 & 2 \\ -1 & 1 & 0 \\ 1 & 0 & 3 \end{bmatrix}. \tag{2.19}$$

Total-matricen $\begin{bmatrix} A & \mathbf{u} \end{bmatrix}$ og den reducerede trappeform $\begin{bmatrix} R & \mathbf{u}^* \end{bmatrix}$ kan ses i figur 2.9.

$$\begin{bmatrix} A & \mathbf{u} \end{bmatrix} = \begin{bmatrix} 1 & 2 & -1 & 4 \\ 2 & -1 & 2 & 0 \\ -1 & 1 & 0 & 5 \\ 1 & 0 & 3 & 8 \end{bmatrix} \qquad \begin{bmatrix} R & \mathbf{u}^* \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 4 \\ 0 & 0 & 1 & 3 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

(a) (b)

Figur 2.9: *(a) Totalmatricen* $\begin{bmatrix} A & \mathbf{u} \end{bmatrix}$ *og (b) den reducerede trappeform* $\begin{bmatrix} R & \mathbf{u}^* \end{bmatrix}$.

Af den reducerede trappeform (b) i figur 2.9 fremgår det, at der eksisterer en løsning til $A\mathbf{x} = \mathbf{u}$, da der ikke er pivot i sidste søjle, hvorfor ligningssystemet er konsistent. Det medfører da, at $\mathbf{u} \in$ Span S.

Tilsvarende findes total-matricen $\begin{bmatrix} A & \mathbf{v} \end{bmatrix}$ og den reducerede trappeform $\begin{bmatrix} R & \mathbf{v}^* \end{bmatrix}$ i figur 2.10.

$$\begin{bmatrix} A & \mathbf{v} \end{bmatrix} = \begin{bmatrix} 1 & 2 & -1 & -1 \\ 2 & -1 & 2 & 2 \\ -1 & 1 & 0 & -3 \\ 1 & 0 & 3 & 5 \end{bmatrix} \qquad \begin{bmatrix} R & \mathbf{v}^* \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(a) (b)

Figur 2.10: *(a) Totalmatricen* $\begin{bmatrix} A & \mathbf{v} \end{bmatrix}$ *og (b) den reducerede trappeform* $\begin{bmatrix} R & \mathbf{v}^* \end{bmatrix}$.

Af den reducerede trappeform (b) i figur 2.10 fremgår det, at der er pivot i sidste søjle, hvorfor der ikke eksisterer en løsning til ligningssystemet $A\mathbf{x} = \mathbf{v}$. Altså er $A\mathbf{x} = \mathbf{v}$ inkonsistent, og $\mathbf{v}$ ligger derfor ikke i Span S.

Bemærk af R i eksempel 2.5.2 at vektorerne i S er lineært uafhængige med den konsekvens, at de tre vektorer ikke er en linearkombination af hinanden, hvorfor de tilsammen udspænder $\mathbb{R}^3$. Der gælder derfor, at Span $S = \mathbb{R}^3$ i dette eksempel. Vi siger også, at S er en *genererende mængde* for $\mathbb{R}^3$.

På baggrund af ovenstående introduktion til et Span af vektorer er det nu væsentligt at præsentere nogle egenskaber for lineært afhængige og lineært uafhængige mængder.

2.5.3. Følgende er egenskaber for lineært uafhængige og lineært afhængige mængder.

1. Givet en mængde $S = \{\mathbf{v}\}$ hvor $\mathbf{v} \neq \mathbf{0}$ er S lineært uafhængig. Hvis $S = \{\mathbf{0}\}$ er S lineært afhængig.
2. Givet en mængde $S = \{\mathbf{v_1}, \mathbf{v_2}\}$ med to vektorer er S lineært afhængig, hvis enten $\mathbf{v_1} = \mathbf{0}$ eller $\mathbf{v_2} = \mathbf{0}$ eller den ene vektor er et multiplum af den anden.
3. Givet en delmængde $S = \{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}\} \subseteq \mathbb{R}^n$ og en vektor $\mathbf{u} \in \mathbb{R}^n$. Da vil mængden $K = \{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}, \mathbf{u}\}$ være lineært uafhængig, hvis og kun hvis $\mathbf{u} \notin$ Span S.

4. For alle delmængder af $\mathbb{R}^n$ indeholdende mere end n vektorer gælder, at disse mængder er lineært afhængige.
5. Givet en mængde $S = \{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}\}$ hvor ingen af de k vektorer kan fjernes uden at ændre Spannet af S, da vil S være lineært uafhængig.

Vi er nu klar til at definere *underrum* og *baser* for underrum.

Definition 2.5.4. ***Underrum***
Givet en mængde S af vektorer i $\mathbb{R}^n$ siges S at være et underrum af $\mathbb{R}^n$, hvis der gælder følgende.

(a) $\mathbf{0} \in S$

(b) Hvis $\mathbf{v_1}, \mathbf{v_2} \in S$, så er $\mathbf{v_1} + \mathbf{v_2} \in S$.

(c) Hvis $\mathbf{v_1} \in S$ og $k \in \mathbb{R}$, så er $k\mathbf{v_1} \in S$.

Bemærk af definition 2.5.4 at en mængde S af vektorer i $\mathbb{R}^n$ er et underrum af $\mathbb{R}^n$, hvis og kun hvis vi kan vise, at de tre egenskaber i definition 2.5.4 er opfyldt.

Eksempel 2.5.5. ***Et underrum*** S ***af*** $\mathbb{R}^4$
Givet mængden

$$S = \left\{ \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \end{bmatrix} \in \mathbb{R}^4 \mid v_1 + 2v_2 - 3v_3 - 4v_4 = 0 \right\} \tag{2.20}$$

kan vi vise, at dette er et underrum af $\mathbb{R}^4$ ved at undersøge, om S har de tre egenskaber i definition 2.5.4. Bemærk da først at $\mathbf{0} \in S$, da $0 + 2 \cdot 0 - 3 \cdot 0 - 4 \cdot 0 = 0$. Antag nu at $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^\intercal \in S$ og $\mathbf{y} = \begin{bmatrix} y_1 & y_2 & y_3 & y_4 \end{bmatrix}^\intercal \in S$ så $x_1 + 2x_2 - 3x_3 - 4x_4 = 0$ og $y_1 + 2y_2 - 3y_3 - 4y_4 = 0$. Da gælder der, at $\mathbf{x}+\mathbf{y} = \begin{bmatrix} x_1 + y_1 & x_2 + y_2 & x_3 + y_3 & x_4 + y_4 \end{bmatrix}^\intercal \in S$, fordi

$$\begin{aligned} &(x_1 + y_1) + 2(x_2 + y_2) - 3(x_3 + y_3) - 4(x_4 + y_4) = \\ &\quad x_1 + 2x_2 - 3x_3 - 4x_4 + y_1 + 2y_2 - 3y_3 - 4y_4 = 0 + 0 = 0 \end{aligned} \tag{2.21}$$

Antag så til sidst at $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^\intercal \in S$ og $k \in \mathbb{R}$. Så gælder der, at $k\mathbf{x} = \begin{bmatrix} kx_1 & kx_2 & kx_3 & kx_4 \end{bmatrix}^\intercal \in S$, da

$$\begin{aligned} (kx_1) + 2(kx_2) - 3(kx_3) - 4(kx_4) &= k(x_1 + 2x_2 - 3x_3 - 4x_4) \\ &= k \cdot 0 = \mathbf{0} \end{aligned} \tag{2.22}$$

Da de tre egenskaber for et underrum er opfyldt for S, er det derfor blevet vist, at S er et underrum af $\mathbb{R}^4$.

Givet en matrix A er det nu vigtigt at få defineret to typer af underrum.

Definition 2.5.6. ***Nulrum og søjlerum***
Givet en $m \times n$ matrix A er nulrummet, noteret som Null A, givet ved løsningsmængden til $A\mathbf{x} = \mathbf{0}$. Der gælder derfor, at

$$\text{Null } A = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{0}\} \tag{2.23}$$

Givet den samme $m \times n$ matrix A er søjlerummet, noteret som Col A, givet ved Spannet af søjlerne i A. Der gælder derfor, at

$$\text{Col } A = \text{Span}\{\mathbf{A_1}, \mathbf{A_2}, \ldots, \mathbf{A_n}\} \tag{2.24}$$

Eksempel 2.5.7. ***Et nulrum og et søjlerum for en matrix*** A
Givet en 3×4 matrix A, hvor

$$A = \begin{bmatrix} 2 & 1 & 2 & -1 \\ 1 & 0 & 3 & -4 \\ 2 & 5 & 0 & -1 \end{bmatrix}$$

ønskes en genererende mængde for henholdsvis Null A og Col A. Bemærk da som det første, at

$$R = \begin{bmatrix} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -2 \end{bmatrix}$$

er den reducerede trappeform af A. Løsningsmængden for $A\mathbf{x} = \mathbf{0}$ kan da findes ud fra $\begin{bmatrix} R & \mathbf{0} \end{bmatrix}$, hvor der er pivot i de tre første søjler med det til følge, at løsningsmængden kan beskrives ved den frie variabel x_4 tilhørende den fjerde søjle i R, hvor der ikke er pivot. Det medfører derfor, at

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -2x_4 \\ x_4 \\ 2x_4 \\ x_4 \end{bmatrix} = x_4 \begin{bmatrix} -2 \\ 1 \\ 2 \\ 1 \end{bmatrix} \tag{2.25}$$

løser $A\mathbf{x} = \mathbf{0}$.
Heraf følger det, at løsningsmængden til $A\mathbf{x} = \mathbf{0}$ er Null $A = \text{Span}\left\{\begin{bmatrix} -2 & 1 & 2 & 1 \end{bmatrix}^\intercal\right\}$, og en genererende mængde for Null A er derfor givet ved $S = \left\{\begin{bmatrix} -2 & 1 & 2 & 1 \end{bmatrix}^\intercal\right\}$. Tilsvarende kan en genererende mængde for Col A findes ved blot at aflæse søjle-vektorerne i matricen A, så

$$\text{Col } A = \text{Span}\left\{\begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 5 \end{bmatrix}, \begin{bmatrix} 2 \\ 3 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ -4 \\ -1 \end{bmatrix}\right\}$$

og en genererende mængde for Col A er derfor givet ved

$$\left\{\begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 5 \end{bmatrix}, \begin{bmatrix} 2 \\ 3 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ -4 \\ -1 \end{bmatrix}\right\}$$

Bemærk af eksempel 2.5.7 at der godt må eksistere vektorer i genererende mængder for underrum, som er linearkombinationer af andre vektorer i den samme mængde. Hertil defineres nu en *basis* for et underrum, hvor dette ikke kan lade sig gøre.

Definition 2.5.8. ***Baser for underrum***
Givet en mængde S af vektorer i $\mathbb{R}^n$ siges S at være en basis for et underrum V af $\mathbb{R}^n$, hvis

(a) S er en genererende mængde for V og

(b) vektorerne i S er lineært uafhængige

I eksempel 2.5.7 fandt vi en genererende mængde for søjlerummet af den givne 3×4 matrix A som

$$\left\{\begin{bmatrix}2\\1\\2\end{bmatrix}, \begin{bmatrix}1\\0\\5\end{bmatrix}, \begin{bmatrix}2\\3\\0\end{bmatrix}, \begin{bmatrix}-1\\-4\\-1\end{bmatrix}\right\}$$

Denne genererende mængde er *ikke* en basis for søjlerummet, da den reducerede trappeform R af A viste, at der kun var pivot i de tre første søjler. Og af sætning 2.3.2 betyder det så, at vektorerne i den genererende mængde ikke er lineært uafhængige, hvorfor det ikke kan være en basis for søjlerummet jævnfør definition 2.5.8.

Bemærk dog at vi generelt kan reducere antallet af vektorer i en genererende mængde for et underrum for at opnå en basis for dette underrum, hvilket gengives i den følgende sætning.

Sætning 2.5.9. Givet at S er en genererende mængde med et endeligt antal vektorer for et ikke-tomt underrum V af $\mathbb{R}^n$. Da vil S kunne reduceres til en basis for V ved at fjerne vektorer fra S.

Bevis. Vi lader nu V være et underrum af $\mathbb{R}^n$, $S = \{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}\}$ tilsvarende er en genererende mængde for V. Konstruerer vi så matricen $A = \begin{bmatrix}\mathbf{v_1} & \mathbf{v_2} & \ldots & \mathbf{v_k}\end{bmatrix}$ gælder der ud fra definition 2.5.6, at Col$A$ = Span $\{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}\} = V$, da S netop er en genererende mængde for V. Af definition 2.5.8 gælder der så, at pivot-søjlerne i A giver en basis for Col A, hvilket ligeledes medfører, at disse pivot-søjler udgør en basis for V. Da disse pivot-søjler er lineært uafhængige og er indeholdt i S, er det derfor bevist, at vi kan fjerne overskydende vektorer i S, indtil vi har en lineær uafhængig mængde af vektorer, der er genererende for V. ■

På samme vis kan man udvide en lineært uafhængig mængde af vektorer til en basis for et underrum ud fra følgende sætning.

Sætning 2.5.10. Givet at S er en lineær uafhængig delmængde af et ikke-tomt underrum V af $\mathbb{R}^n$, kan S udvides til en basis for V ved at tilføje vektorer til S, så S bliver en generende mængde for V, og sådan at vektorerne i S stadig er lineært uafhængige. Mere specifikt eksisterer der en basis for alle ikke-tomme underrum.

Bevis. Vi lader $S = \{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}\}$ være en lineær uafhængig delmængde af underrummet V af $\mathbb{R}^n$. Hvis Span $S = V$ udgør S en basis for V, og vi behøver derfor ikke at tilføje vektorer til S. Hvis ikke dette er opfyldt, eksisterer der en vektor $\mathbf{u_1} \in V$ som ikke tilhører Span S. Ifølge tredje egenskab i 2.5.3 er vektorerne i $S' = \{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}, \mathbf{u_1}\}$ derfor lineært uafhængige. Hvis Span $S' = V$, så er S' en basis for V der indeholder S, og vi skal derfor igen ikke tilføje flere vektorer til S. Hvis S' heller ikke er en basis for V, eksisterer der endnu en vektor $\mathbf{u_2} \in V$, som ikke tilhører Span S'. Tilføjer vi da $\mathbf{u_2}$ til S', så vi får en ny mængde $S'' = \{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}, \mathbf{u_1}, \mathbf{u_2}\}$, vil vektorerne i denne mængde være lineært uafhængige. Processen fortsættes, indtil vi får en genererende mængde for V bestående af lineært uafhængige vektorer og dermed en basis for V, der indeholder vektorerne i S. Bemærk at der højst kan tilføjes n vektorer, da en delmængde af $\mathbb{R}^n$ med mere end n vektorer vil være lineær uafhængig ifølge fjerde egenskab i 2.5.3. For tilsvarende at bevise, at ethvert ikke-tomt underrum V har en basis, lader vi $\mathbf{u} \neq \mathbf{0}$ være en vektor i V. Konstruerer vi så $S = \{\mathbf{u}\}$, som er en lineær uafhængig delmængde af V jævnfør første egenskab i 2.5.3, kan vi blot udvide S til en lineær uafhængig og genererende mængde for V, så det bliver en basis for V, og det beviser derfor, at der eksisterer en basis for ethvert ikke-tomt underrum V.

■

2.6 Matrix-matrix produkt og invertible matricer

Definition 2.6.1. ***Matrix-matrix produkt***
Lad A være en $m \times n$ matrix og lad B være en $n \times k$ matrix. Vi definerer så matrix-matrix produktet AB til at være ny $m \times k$ matrice, hvor søjlerne i AB er givet ved $A\mathbf{B_j}$ for $j \in \{1, 2, \ldots, k\}$, så

$$AB = \begin{bmatrix} A\mathbf{B_1} & A\mathbf{B_2} & \cdots & A\mathbf{B_k} \end{bmatrix}. \tag{2.26}$$

Bemærk af definition 2.6.1 at antallet af rækker i B skal tilsvare antallet af søjler i A, hvis matrix-matrix produktet AB skal være defineret.

Eksempel 2.6.2. Lad A være en 3×4 matrix og lad B være en 4×5 matrix givet ved

$$A = \begin{bmatrix} 2 & 4 & 5 & 1 \\ 3 & 1 & 4 & 3 \\ 5 & 3 & 2 & 4 \end{bmatrix} \quad B = \begin{bmatrix} 4 & 2 & 3 & 5 & 2 \\ 3 & 3 & 2 & 4 & 1 \\ 4 & 1 & 4 & 2 & 2 \\ 5 & 3 & 2 & 1 & 1 \end{bmatrix}$$

AB vil da bestå af søjlerne givet ved

$$\mathrm{AB}_1 = \begin{bmatrix} 2 & 4 & 5 & 1 \\ 3 & 1 & 4 & 3 \\ 5 & 3 & 2 & 4 \end{bmatrix} \begin{bmatrix} 4 \\ 3 \\ 4 \\ 5 \end{bmatrix} = \begin{bmatrix} 45 \\ 46 \\ 57 \end{bmatrix} \qquad \mathrm{AB}_2 = \begin{bmatrix} 2 & 4 & 5 & 1 \\ 3 & 1 & 4 & 3 \\ 5 & 3 & 2 & 4 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \\ 1 \\ 3 \end{bmatrix} = \begin{bmatrix} 24 \\ 22 \\ 33 \end{bmatrix}$$

$$\mathrm{AB}_3 = \begin{bmatrix} 2 & 4 & 5 & 1 \\ 3 & 1 & 4 & 3 \\ 5 & 3 & 2 & 4 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \\ 4 \\ 2 \end{bmatrix} = \begin{bmatrix} 36 \\ 33 \\ 37 \end{bmatrix} \qquad \mathrm{AB}_4 = \begin{bmatrix} 2 & 4 & 5 & 1 \\ 3 & 1 & 4 & 3 \\ 5 & 3 & 2 & 4 \end{bmatrix} \begin{bmatrix} 5 \\ 4 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 37 \\ 30 \\ 45 \end{bmatrix}$$

$$\mathrm{AB}_5 = \begin{bmatrix} 2 & 4 & 5 & 1 \\ 3 & 1 & 4 & 3 \\ 5 & 3 & 2 & 4 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 19 \\ 18 \\ 21 \end{bmatrix},$$

så AB bliver 3×5 matricen givet ved

$$AB = \begin{bmatrix} 45 & 24 & 36 & 37 & 19 \\ 46 & 22 & 33 & 30 & 18 \\ 57 & 33 & 37 & 45 & 21 \end{bmatrix}$$

Bemærk at for to vilkårlige matricer A og B vil AB i de fleste tilfælde ikke være det samme som BA.

Man kan midlertidigt beregne matrix-matrix produkter på en anden måde. (ij)-indgangen i et matrix-matrix produkt AB kan nemlig findes som prikproduktet mellem den i'te række i A og den j'te søjle i B. For at illustrere dette laves et kort eksempel.

Eksempel 2.6.3. Lad A og B være 2×2 matricer givet ved

$$A = \begin{bmatrix} 3 & 5 \\ 4 & 2 \end{bmatrix} \qquad B = \begin{bmatrix} 4 & 1 \\ 2 & 4 \end{bmatrix}$$

Vi anvender her prikproduktet mellem rækker i A og søjler i B til at beregne matrix-matrix produktet AB, hvorfor vi får, at

$$AB = \begin{bmatrix} 3 \cdot 4 + 5 \cdot 2 & 3 \cdot 1 + 5 \cdot 4 \\ 4 \cdot 4 + 2 \cdot 2 & 4 \cdot 1 + 2 \cdot 4 \end{bmatrix} = \begin{bmatrix} 22 & 23 \\ 20 & 12 \end{bmatrix}$$

En vigtig type af matricer der kommer til at spille en stor rolle i forbindelse med løsning af lineære programmeringsproblemer er *invertible matricer*.

Definition 2.6.4. ***Invertible matricer***
Givet en $n \times n$ matrix A siges A at være invertibel, hvis der eksisterer en anden $n \times n$ matrix B, hvorom det gælder, at

$$AB = BA = I_n \tag{2.27}$$

Her kaldes B for den *inverse af* A, og vi betegner det på den måde, at $A^{-1} = B$.

Bemærk at ligning 2.27 i definition 2.6.4 er ækvivalent med ligningen

$$AA^{-1} = A^{-1}A = I_n$$

Eksempel 2.6.5. ***En invertibel matrix*** A ***og dens inverse*** B
Givet matricer A og B som

$$A = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} \quad B = \begin{bmatrix} -\frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ 0 & -1 & 2 \\ \frac{1}{4} & \frac{1}{2} & -\frac{5}{4} \end{bmatrix}$$

kan vi vise, at B er den inverse matrix til A og derfor at $B = A^{-1}$ ved at undersøge, om ligningen $AB = BA = I_3$ er opfyldt, hvilket er tilfældet her, da

$$AB = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} -\frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ 0 & -1 & 2 \\ \frac{1}{4} & \frac{1}{2} & -\frac{5}{4} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I_3$$
$$BA = \begin{bmatrix} -\frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ 0 & -1 & 2 \\ \frac{1}{4} & \frac{1}{2} & -\frac{5}{4} \end{bmatrix} \begin{bmatrix} 1 & 3 & 5 \\ 2 & 1 & 2 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = I_3$$

Har man et ligningssystem på formen $A\mathbf{x} = \mathbf{b}$, hvor A er en invertibel $n \times n$ matrix, hvor $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$, kan man finde en unik løsning $\mathbf{x}$ ved at bruge den inverse af A for at løse $A\mathbf{x} = \mathbf{b}$ ved følgende observation og regneregel 6 fra 2.1.9

$$\begin{aligned} A\mathbf{x} = \mathbf{b} &\implies \\ A^{-1}(A\mathbf{x}) = A^{-1}\mathbf{b} &\iff \\ (A^{-1}A)\mathbf{x} = A^{-1}\mathbf{b} &\iff \\ I_n\mathbf{x} = \mathbf{x} = A^{-1}\mathbf{b} & \end{aligned}$$

I forhold til eksempel 2.6.5 opstilles nu et lineært ligningssystem givet ved

$$A\mathbf{x} = \mathbf{b}, \tag{2.28}$$

hvor matricen A svarer til 3×3 matricen A fra eksempel 2.6.5 og $\mathbf{b} = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}^\intercal$. Her kan den unikke løsning $\mathbf{x}$ derfor findes ved at lave matrix-vektor produktet $B\mathbf{b} = A^{-1}\mathbf{b}$, hvor B er den inverse matrix til A fra eksempel 2.6.5.

$$\begin{aligned} \mathbf{x} = B\mathbf{b} &= \begin{bmatrix} -\frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ 0 & -1 & 2 \\ \frac{1}{4} & \frac{1}{2} & -\frac{5}{4} \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \\ &= \begin{bmatrix} -\frac{1}{4} + 1 + \frac{3}{4} \\ 0 - 2 + 6 \\ \frac{1}{4} + 1 - \frac{15}{4} \end{bmatrix} = \begin{bmatrix} \frac{3}{2} \\ 4 \\ -\frac{5}{2} \end{bmatrix} \end{aligned}$$

Der eksisterer en vigtig type af $n \times n$ invertible matricer kaldet *elementær-matricer* som har den egenskab, at de er fremkommet ved at udføre én rækkeoperation på identitets-matricen I_n. Som eksempel kan nævnes matricen

$$E = \begin{bmatrix} 1 & 0 & -4 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.29}$$

som er en elementær-matrix, der er fremkommet ved rækkeadditionen $-4r_3 + r_1 \to r_1$. Det medfører derfor også, at laver man matrix-matrix produktet EA for en $m \times m$ elementær-matrix E og en $m \times n$ matrix A svarer det til at udføre den pågældende elementære

rækkeoperation på A. Vi husker på at for at løse lineære ligningssystemer anvendes elementære rækkeoperationer, og at udføre disse elementære rækkeoperationer på en given $m \times n$ matrix A for at opnå den reducerede trappeform R kan via elementær-matricer oversættes til matrix-matrix produkter på formen

$$E_k E_{k-1} \cdots E_1 A = R, \tag{2.30}$$

hvilket leder frem til følgende sætning, der er bevist i [7, p. 127-128].

Sætning 2.6.6. Lad A være en $m \times n$ matrix og lad R være den reducerede trappeform af A. Da eksisterer der en invertibel $m \times m$ matrix Q sådan, at

$$QA = R \tag{2.31}$$

hvor Q er produktet af k elementær-matricer så $Q = E_k E_{k-1} \cdots E_1$.

Det undersøges nu hvordan man afgøre om en matrix er invertibel. Dette udtrykkes i den følgende sætning.

Sætning 2.6.7. Givet en $n \times n$ matrix A er A invertibel, hvis og kun hvis den reducerede trappeform af A giver identitets-matricen I_n.

Bevis. Vi antager først, at A er invertibel og skal så vise, at den reducerede trappeform af A giver identitets-matricen I_n. Vi betragter nu en vektor $\mathbf{x} \in \mathbb{R}^n$ som opfylder, at $A\mathbf{x} = \mathbf{0}$. Fordi A er invertibel gælder så, at $A\mathbf{x} = \mathbf{0} \implies A^{-1}A\mathbf{x} = A^{-1}\mathbf{0} \implies \mathbf{x} = \mathbf{0}$, hvorfor den eneste løsning til det lineære ligningssystem er $\mathbf{x} = \mathbf{0}$. Vi husker så, at dette nødvendigvis må betyde, at rank $A = n$ jævnfør sætning 2.3.2 udsagn 2, hvilket betyder, at der er pivot i alle søjler jævnfør sætning 2.2.8. Da der skal være pivot i alle n søjler, følger det derfor, at den reducerede trappeform af A kun kan være I_n.

Vi antager så, at I_n er den reducerede trappeform af A og skal så vise, at A er invertibel. Vi ser da, at der må eksisterer en invertibel matrix Q med dimension $n \times n$, som er et produkt af elementær-matricer, sådan at

$$QA = I_n \tag{2.32}$$

Der må så gælde, at

$$A = I_n A = (Q^{-1}Q)A = Q^{-1}(QA) = Q^{-1}I_n = Q^{-1}$$

så $A = Q^{-1}$. Vi ved, at Q er invertibel fra sætning 2.6.6, hvorfor Q^{-1} også er det, og vi får derfor resultatet, at A er invertibel. ■

Bemærk af sætning 2.6.7 at det kan undersøges, om en givet $n \times n$ matrix A er invertibel ved at finde den reducerede trappeform af A og se, om $R = I_n$. Denne idé kan udbygges en smule, idet vi kan lave en anden vigtig observation. Antag at vi finder den reducerede trappeform af matricen $\begin{bmatrix} A & I_n \end{bmatrix}$. Da må så gælde, at der eksisterer en invertibel matrix Q sådan, at

$$Q \begin{bmatrix} A & I_n \end{bmatrix} = \begin{bmatrix} QA & QI_n \end{bmatrix} = \begin{bmatrix} R & Q \end{bmatrix} \tag{2.33}$$

Hvis $R \neq I_n$ ved vi, at A ikke er invertibel, men gælder der modsat, at $R = I_n$, så $QA = I_n$, må det nødvendigvis betyde, at $Q = A^{-1}$, hvorfor vi nu også har bestemt den inverse matrix til A. Vi gengiver her metoden i et kort eksempel.

Eksempel 2.6.8. Givet

$$A = \begin{bmatrix} -1 & 5 \\ 2 & 0 \end{bmatrix}$$

ønsker vi at vise, at A er invertibel samt finde den inverse A^{-1}. Vi skaber derfor matricen $\begin{bmatrix} A & I_2 \end{bmatrix}$ som derefter rækkereduceres til reduceret trappeform.

$$\begin{bmatrix} -1 & 5 & 1 & 0 \\ 2 & 0 & 0 & 1 \end{bmatrix} \xrightarrow{2r_1+r_2 \to r_2} \begin{bmatrix} -1 & 5 & 1 & 0 \\ 0 & 10 & 2 & 1 \end{bmatrix} \xrightarrow{\frac{1}{10}r_2 \to r_2} \begin{bmatrix} -1 & 5 & 1 & 0 \\ 0 & 1 & \frac{1}{5} & \frac{1}{10} \end{bmatrix}$$
$$\xrightarrow{-5r_2+r_1 \to r_1} \begin{bmatrix} -1 & 0 & 0 & -\frac{1}{2} \\ 0 & 1 & \frac{1}{5} & \frac{1}{10} \end{bmatrix} \xrightarrow{-r_1 \to r_1} \begin{bmatrix} 1 & 0 & 0 & \frac{1}{2} \\ 0 & 1 & \frac{1}{5} & \frac{1}{10} \end{bmatrix} \tag{2.34}$$

Heraf ser vi både, at A er invertibel, og at

$$A^{-1} = \begin{bmatrix} 0 & \frac{1}{2} \\ \frac{1}{5} & \frac{1}{10} \end{bmatrix}$$

hvilket vi hurtigt kan verificere ved at se, at $AA^{-1} = A^{-1}A = I_2$

$$AA^{-1} = \begin{bmatrix} -1 & 5 \\ 2 & 0 \end{bmatrix} \begin{bmatrix} 0 & \frac{1}{2} \\ \frac{1}{5} & \frac{1}{10} \end{bmatrix} = \begin{bmatrix} (-1)0 + (\frac{1}{5})5 & (-1)\frac{1}{2} + (\frac{1}{10})5 \\ (2)0 + (\frac{1}{5})0 & (\frac{1}{2})2 + (\frac{1}{10})0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I_2$$
$$A^{-1}A = \begin{bmatrix} 0 & \frac{1}{2} \\ \frac{1}{5} & \frac{1}{10} \end{bmatrix} \begin{bmatrix} -1 & 5 \\ 2 & 0 \end{bmatrix} = \cdots = I_2$$

En vigtig egenskab for invertible matricer som følger af ovenstående beskrivelser er gengivet i det følgende.

2.6.9. Givet en invertibel matrix A er søjlerne i A lineært uafhængige.

Vi er nu klar til at anvende værktøjerne fra dette kapitel til at beskrive den geometriske og algebraiske forståelse af lineær programmering og baggrunden for *Simplex-metoden*.

3 | Lineær programmering

I dette kapitel vil grundbegreberne i lineær programmering blive præsenteret med tilhørende notation samt en grafisk løsningsmetode for lineære programmeringproblemer med højst tre uafhængige variable. Dette vil lede videre til en generalisering af lineære programmeringsproblemer i flere dimensioner med tilhørende løsningsmetoder i det efterfølgende kapitel. Kapitlet her tager udgangspunkt i [5, p. 651-687], medmindre andet fremgår af teksten.

3.1 Introduktion til lineær programmering

Lineær programmering er som nævnt en nyere metode indenfor matematisk optimering. Navnet lineær programmering refererer til, at alle ligninger i systemet er lineære, hvilket i en algebraisk forstand betyder, at funktionen der ønskes optimeret samt de givne begrænsninger er lineære afbildninger jævnfør definition 2.4.1.

For at opstille et lineært programmeringsproblem introduceres først to vigtige grundbegreber.

- *Objektfunktion* - funktionen, som udspringer fra et givet problem, der ønskes optimeret ved maksimering eller minimering. I tilfælde af lineære programmeringsproblemer er objektfunktionen en lineær afbildning, og koefficienterne for denne lineære afbildning er på forhånd givet. Variablene noteres som x_j hvor $j \in \{1, 2, \ldots, n\}$, og koefficienterne noteres ved c_j hvor $j \in \{1, 2, \ldots, n\}$. Antallet af variable n afgør, hvor mange dimensioner det pågældende programmeringsproblem bevæger sig i, så $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^\intercal$ er i $\mathbb{R}^n$.
- *Lineære betingelser* - en eller flere ligninger der indeholder de samme variable som objektfunktionen, men med nye koefficienter som afhænger af de pågældende betingelser fra den praktiske problemstilling. Der stilles umiddelbart det samme krav til disse lineære betingelser som for objektfunktionen; nemlig at de skal være lineære.

3.2 Notation til lineær programmering

I dette afsnit vil notationsmetoder til lineære programmeringproblemer blive præsenteret, da der findes flere forskellige måder at notere lineære programmeringsproblemer på. I dette projekt vil der primært tages udgangspunkt i *longhand-notation* og *matrix-notation*. I hver af disse notationsmetoder tages der udgangspunkt i n-variable og m-betingelser, hvor de to grundelementer - objektfunktionen og de lineære betingelser - danner grundlaget for notationen. Derudover vil forskellen på lineære programmeringsproblemer på generel form og lineære programmeringsproblemer på standard form blive præsenteret.

3.2.1 Longhand-notation

Den første notationsmetode er *longhand-notationen*, hvor ligningen for objektfunktionen og de lineære betingelser skrives ud. Med longhand-notation vil et *maksimeringsproblem* inden for lineær programmering da være på formen

$$\begin{aligned} &\textit{Maksimer} && f(x_1, x_2, \ldots, x_n) = c_1x_1 + c_2x_2 + \ldots + c_nx_n \\ &\textit{I forhold til} && a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n * b_1 \\ & && a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n * b_2 \\ & && \vdots \\ & && a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n * b_m \end{aligned}$$

Her betegner $f(x_1, x_2, \ldots, x_n) = c_1x_1 + c_2x_2 + \cdots + c_nx_n$ som sagt objektfunktionen, hvor f er den størrelse, der skal maksimeres ved optimering. Som nævnt er variablene noteret ved x_j, hvor $j \in \{1, 2, \ldots, n\}$ og koefficienterne noteret ved c_j, hvor $j \in \{1, 2, \ldots, n\}$.

Uligheder og/eller lighederne der følger efter objektfunktionen viser de lineære betingelser, der er med til at afgrænse det område, som objektfunktionen skal optimeres inden for. Bemærk at $*$ angiver, at det enten er $\geq$, $\leq$ eller $=$ [9, p. 19-23]. I disse betingelser er variablene stadig noteret ved x_j, mens b_i for $i \in \{1, 2, \ldots, m\}$ er de konstanter, som uligheder og/eller lighederne skal overholde. Koefficienterne for x_j i betingelserne er noteret ved a_{ij}, hvor $i \in \{1, 2, \ldots, m\}$ og $j \in \{1, 2, \ldots, n\}$. At a_{ij} er beskrevet ved både i og j får en betydning for den tilsvarende matrix-form af maksimeringsproblemet, hvor a_{ij} her svarer til en indgang i en $m \times n$ matrix A. Dette følger nærmere i det kommende afsnit.

Et tilsvarende *minimeringsproblem* inden for lineær programering vil med longhand-notationen være på formen

$$\begin{aligned} &\textit{Minimer} && f(x_1, x_2, \ldots, x_n) = c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ &\textit{I forhold til} && a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n * b_1 \\ & && a_{21}x_1 + a_{22}x_2 \cdots + a_{2n}x_n * b_2 \\ & && \vdots \\ & && a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n * b_m \end{aligned}$$

Til forskel fra maksimeringsproblemet er det blot et spørgsmål om, at objektfunktionen nu ønskes minimeret, hvor der gælder følgende vigtige observation.

3.2.1. Man kan gå fra et maksimeringsproblem til et minimeringsproblem (og omvendt) ved at multiplicere objektfunktionen med -1.

Eksempel 3.2.2. Et maksimeringsproblem som et lineært programmeringsproblem er givet på formen

$$\begin{aligned}
&\textit{Maksimer} && f(x_1,x_2,x_3) = 2x_1 - 3x_2 + 7x_3,\\
&\textit{I forhold til} && x_1 + x_2 - x_3 \geq 3\\
& && 2x_1 - 5x_2 = 6\\
& && 4x_1 + 3x_2 + 5x_3 \leq 10\\
& && x_1 \geq 0\\
& && x_2 \leq 0\\
& && x_3 \geq 0
\end{aligned}$$

Bemærk at dette maksimeringsproblem svarer til et minimeringsproblem på formen

$$\begin{aligned}
&\textit{Minimer} && f(x_1,x_2,x_3) = -2x_1 + 3x_2 - 7x_3,\\
&\textit{I forhold til} && x_1 + x_2 - x_3 \geq 3\\
& && 2x_1 - 5x_2 = 6\\
& && 4x_1 + 3x_2 + 5x_3 \leq 10\\
& && x_1 \geq 0\\
& && x_2 \leq 0\\
& && x_3 \geq 0
\end{aligned}$$

hvor objektfunktionen netop er blevet multipliceret med -1.

3.2.2 Matrix-notation

Som præsenteret i det foregående afsnit findes en tilsvarende *matrix-notation* for opstillingen af lineære programmeringsproblemer, hvilket senere får en betydning for overskueligheden ved løsning af optimeringsproblemerne. Matrix-notationen kræver da først 4 generelle matricer og vektorer givet på formerne

$$\mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

Figur 3.1

$\mathbf{c}$, $\mathbf{x}$ og $\mathbf{b}$ er søjle-vektorer, hvor $\mathbf{c}$ og $\mathbf{x}$ er af størrelsen $n \times 1$ og er derfor vektorer i $\mathbb{R}^n$. Vi benævner indgangene i $\mathbf{c}$ som *kostkoefficienterne* for objektfunktionen, og vektoren $\mathbf{c}$ benævnes som *kostkoefficient-vektoren.* $\mathbf{b}$ er *betingelsesvektoren* af størrelsen $m \times 1$ og er derfor en vektor i $\mathbb{R}^m$, hvor m er antallet af betingelser. A er da en koefficient-matrix af størrelsen $m \times n$.

Ud fra disse vektorer og matricer kan maksimeringsproblemet fra longhand-notation formuleres ved matrix-notation på formen

$$\begin{aligned} &\textit{Maksimer} && f(\mathbf{x}) = \mathbf{c}^\intercal \mathbf{x} \\ &\textit{I forhold til} && A\mathbf{x} * \mathbf{b} \end{aligned}$$

Bemærk her at objektfunktionen øverst er defineret som $f(\mathbf{x}) = \mathbf{c}^\intercal \mathbf{x}$, hvor søjle-vektoren $\mathbf{c}$ er blevet transponeret og derfor har størrelsen $1 \times n$, så matrix-vektor produktet $\mathbf{c}^\intercal \mathbf{x}$ bliver en vektor af størrelsen 1×1 - eller blot et reelt tal.

Betingelserne fylder betydeligt mindre i matrix-notationen, når de er på formen $A\mathbf{x} * \mathbf{b}$, heraf kommer fordelen ved denne notation, da det kort og præcist illustrerer det lineære programmeringsproblem.

Tilsvarende kan minimeringsproblemet fra longhand-notationen opstilles ved matrix-notation på formen

$$\begin{aligned} &\textit{Minimer} && f(\mathbf{x}) = \mathbf{c}^\intercal \mathbf{x} \\ &\textit{I forhold til} && A\mathbf{x} * \mathbf{b} \end{aligned}$$

3.2.3 Lineære programmeringsproblemer på standard form

Vi specificerer nu et lineært programmeringsproblem på standard form, da det er en gennemgående form af lineære programmeringsproblemer gennem resten af projektet. Bemærk at vi nu introducerer *ikke-negative begrænsninger* for variablene, som gør, at variablene er begrænset til ikke at være negative. Et lineært programmeringsproblem på standard form er da mere specifikt givet ved longhand-notationen

$$\begin{aligned} &\textit{Maksimer} && f(x_1, x_2, \ldots, x_n) = c_1x_1 + c_2x_2 + \ldots + c_nx_n \\ &\textit{I forhold til} && a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ & && a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ & && \vdots \\ & && a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n = b_m \\ &\textit{Og} && x_j \geq 0 \qquad j \in \{1, 2, \ldots, n\} \end{aligned}$$

Bemærk at alle betingelser fra generelle lineære programmeringsproblemer er blevet omformet til lighedsbetingelser. Tilsvarende gælder der også, at *alle* variablene $x_1, x_2, \ldots, x_n$ som elementer i $\mathbf{x}$ skal opfylde, at $x_j \geq 0$ for $j \in \{1, 2, \ldots, n\}$. Den tilsvarende matrix-notation er da givet ved

$$\begin{aligned} &\textit{Minimer} && f(\mathbf{x}) = \mathbf{c}^\intercal \mathbf{x} \\ &\textit{I forhold til} && A\mathbf{x} = \mathbf{b} \\ &\textit{Og} && \mathbf{x} \geq \mathbf{0} \end{aligned}$$

En vigtig del af forståelsen for lineære programmeringsproblemer på standard form er, at ethvert lineært programmeringsproblem på generel form kan omformes til standard form ved at kigge på betingelserne og de ikke-negative begrænsninger for variablene. Mere generelt sker omformningen ved to trin, når man skal fra lineære programmeringsproblemer på generel form til standard form.

1. *Elimination af "frie" variable* - hvis en variabel (eller flere variable) x_j for $j \in \{1, 2, \ldots, n\}$ ikke er begrænset af $x_j \geq 0$ udskiftes den pågældende variabel x_j med $x'_j - x''_j$ i objektfunktionen og i de resterende betingelser. Der tilføjes så yderligere, at $x'_j \geq 0$ og $x''_j \geq 0$, hvorfor den pågældende variabel x_j nu er blevet skiftet ud med de to andre variable $x'_j \geq 0$ og $x''_j \geq 0$. Bemærk at denne operation altid er gyldig, da ethvert reelt tal uanset fortegn kan beskrives som en difference mellem to ikke-negative tal.

2. *Elimination af ulighedsbetingelser* - hvis et lineært programmeringsproblem indeholder en ulighedsbetingelse på formen

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \geq b_i \tag{3.1}$$

substraheres en *overskudsvariabel* s_1 på venstre side af ulighedstegnet, så uligheden bliver til en lighed på formen

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n - s_1 = b_i, \tag{3.2}$$

hvor $s_1 \geq 0$. Tilsvarende gælder der, at en ulighedsbetingelse på formen

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n \leq b_i \tag{3.3}$$

omformes til en lighedsbetingelse, men her ved at addere en *underskudsvariabel* s_2 på venstre side af ulighedstegnet, så lighedsbetingelsen nu er på formen

$$a_{i1}x_1 + a_{i2}x_2 + \cdots + a_{in}x_n + s_2 = b_i, \tag{3.4}$$

hvor $s_2 \geq 0$.

Af eksempel 3.2.2 kan det generelle maksimeringsproblem omformes til standard form ved brug af ovenstående trin. Bemærk da som det første at x_3 ikke er begrænset til at være ikke-negativ, hvorfor vi udskifter x_3 med $x'_3 - x''_3$ og tilføjer $x'_3 \geq 0$ og $x''_3 \geq 0$. Tilsvarende omformes den første ulighedsbetingelse til en lighedsbetingelse ved at trække en overskudsvariabel s_1 fra på venstre side af ulighedstegnet, hvor $s_1 \geq 0$. Ligeledes omformes den sidste ulighedsbetingelse til en lighedsbetingelse ved at lægge en underskudsvariabel s_2 til på venstre side af ulighedstegnet, hvor $s_2 \geq 0$. Alt i alt medfører det maksimeringsproblemet på standard form givet ved longhand-notationen

$$\begin{aligned}
&\textit{Maksimer} && f(x_1, x_2, x'_3, x''_3, s_1, s_2) = 2x_1 - 3x_2 + 7x'_3 - 7x''_3 + 0s_1 + 0s_2, \\
&\textit{I forhold til} && x_1 + x_2 - (x'_3 - x''_3) - s_1 = 3 \\
& && 2x_1 - 5x_2 = 6 \\
& && 4x_1 + 3x_2 + 5x'_3 - 5x''_3 + s_2 = 10 \\
&\textit{Og} && z \geq 0, \quad z \in \{x_1, x_2, x'_3, x''_3, s_1, s_2\}
\end{aligned}$$

3.3 Grafisk løsningsmetode

I dette afsnit vil der blive præsenteret en grafisk løsningsmetode til lineære programmeringsproblemer i to dimensioner, der senere vil blive generaliseret.

Den grafiske metode undersøger som sagt løsninger til et givet lineært programmeringsproblem, hvilket kræver først at indtegne de givne betingelser i et passende koordinatsystem. De angivne betingelser afgrænser da *mulighedsområdet*, hvor variable x_1 og x_2 i

hjørnepunkter vil give optimale værdier af objektfunktionen $f(x_1, x_2)$ givet, at der eksisterer en løsning i mulighedsområdet. At dette rent faktisk er tilfældet, vil blive bevist senere i rapporten. En optimal løsning som enten maksimerer eller minimerer værdien af objektfunktionen kan da findes grafisk ved at indtegne *niveaukurver* $f(x_1, x_2) = k$ for objektfunktionen, hvor $k \in \mathbb{R}$ og forskyde disse niveaukurver i den retning, der optimerer værdien af objektfunktionen, indtil et hjørnepunkt opnås. Da vil den pågældende værdi af k være den optimale værdi af objektfunktionen, og størrelsen af variablene i dette hjørnepunkt enten maksimerer eller minimerer derfor objektfunktionen [5].

Den grafiske metode til at finde løsninger til linære programmeringsproblemer er begrænset af antallet af variable, der ønskes optimeret for i det pågældende problem. Det vil altså være antallet af variable, der bestemmer, hvilken dimensionen afbildningen kan laves i. Bemærk at den grafiske metodes anvendelse kun giver mening i to og tre dimensioner, men hvor der i dette afsnit kun vil blive præsenteret et eksempel med to variable x_1 og x_2.

3.3.1 Et Eksempel der viser den grafiske løsningsmetode

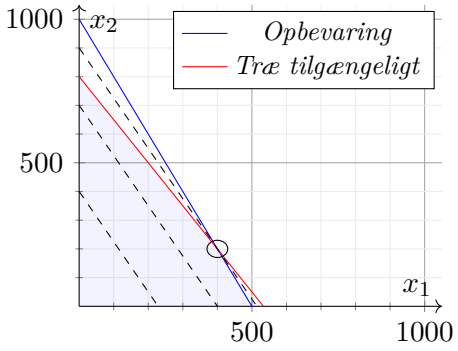
Givet et virkelighedsnært problem handler det i første omgang om at få det oversat til en matematisk model - i dette tilfælde et lineært programmeringsproblem. Objektfunktionen der skal optimeres, opstilles ud fra antallet af variable der repræsenterer problemet, og yderligere skal betingelserne opstilles for at repræsentere begrænsningerne i problemet. Problemet kan så repræsenteres ved longhand-notation (eller matrix-notation), hvorefter problemet kan afbildes grafisk. Bemærk af det følgende eksempel at det *ikke* afspejler en reel produktion.

En produktion af borde og stole

Lad problemet være et ønske om størst mulig profit for en virksomhed der producerer borde og stole. Virksomheden har en pladsbegrænsning på $1000m^3$ til opbevaring af de færdigproducerede borde og stole og har $1600m^3$ træ tilgængeligt til produktionen. Lad så x_1 være antallet af producerede borde og x_2 være antallet af producerede stole. Bordene giver 175 kr. i overskud pr. bord, og stolene giver 100 kr. i overskud pr. stol. Bordene optager $2m^3$ plads, og stolene optager $1m^3$ plads i opbevaring pr. enhed. I produktionen af borde og stole anvendes der henholdsvis $3m^3$ træ og $2m^3$ træ som materiale. Bemærk at træ ikke nødvendigvis er det eneste produktionsmateriale til borde og stole i reelle praktiske problemer, men dette illustrerer blot et forsimplet eksempel. Formuleringen af disse betingelser gør det nu muligt at opstille det lineære programmeringsproblem ved longhand-notation på figur 3.2a samt illustrere mulighedsområdet grafisk som gjort i grafen på figur 3.2b, ved det blå område. Bemærk her at i en praktisk sammenhæng giver det ikke mening at producere et negativt antal borde og stole, hvorfor betingelserne $x_1 \geq 0$ og $x_2 \geq 0$ tilføjes.

$$
\begin{aligned}
&\textit{Maksimer} && f(x_1, x_2) = 175x_1 + 100x_2 \\
&\textit{I forhold til} && 2x_1 + x_2 \leq 1000 \\
& && 3x_1 + 2x_2 \leq 1600 \\
& && x_1 \geq 0 \\
& && x_2 \geq 0
\end{aligned}
$$

(a)



(b)

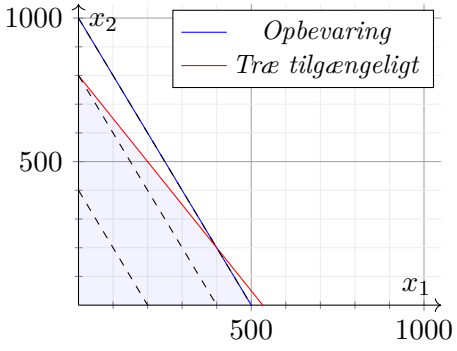
Figur 3.2: *(a) Problemet opskrevet med longhand-notation. (b) En grafisk afbildning.*

Bemærk af figur 3.2 at betingelserne danner et mulighedsområde for det opstillede eksempel. Et hjørnepunkt fremkommer her af skæringen mellem linjerne for de to betingelser. De stiplede linjer repræsenterer niveaukurver for objektfunktionen givet ved $f(x_1, x_2) = 40.000$, $f(x_1, x_2) = 70.000$ og $f(x_1, x_2) = 90.000$, hvor forskydningen af niveaukurverne mod skæringen mellem de to betingelsesslinjer for opbevaring og mængden af træ tilgængeligt for produktionen her maksimerer værdien af objektfunktionen, da værdien af objektfunktionen øges i denne retning. Bemærk at forskydningen af niveaukurverne for objektfunktionen fortsættes, så længe niveaukurverne forbliver i mulighedsområdet. Hjørnepunktet, markeret med en cirkel på figur 3.2b, som i dette tilfælde maksimerer værdien af objektfunktionen i det givne lineære programmeringsproblem repræsenterer præcis én løsning til maksimeringsproblemet. I hjørnepunktet er $x_1 = 400$ og $x_2 = 200$, og disse værdier kan efterfølgende indsættes i objektfunktion for at finde den pågældende maksimale værdi af objektfunktionen, så $175 \cdot 400 + 100 \cdot 200 = 90.000$ bliver den maksimale profit.

Forestiller man sig nu, at overskuddet pr. solgt bord ændrer sig fra 175 kr. til 200 kr, fås et lineært programmeringsproblem som givet på figur 3.3.

$$
\begin{aligned}
&\textit{Maksimer} && f(x_1, x_2) = 200x_1 + 100x_2 \\
&\textit{I forhold til} && 2x_1 + x_2 \leq 1000 \\
& && 3x_1 + 2x_2 \leq 1600 \\
& && x_1 \geq 0 \\
& && x_2 \geq 0
\end{aligned}
$$

(a)



(b)

Figur 3.3: *(a) Problemet opskrevet med longhand-notation. (b) En grafisk afbildning. Bemærk at objektfunktionen er blevet ændret i forhold til figur 3.2.*

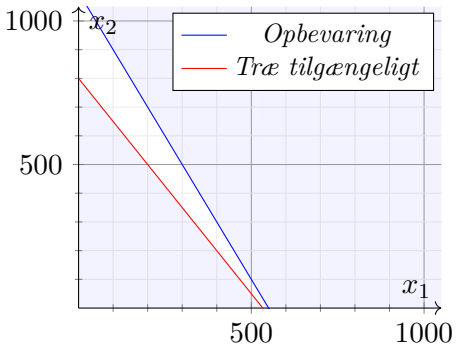
Bemærk af objektfunktionen i longhand-notationen på figur 3.3a at det er koefficienten tilhørende x_1 i objektfunktionen, der har ændret sig. Den tilhørende grafiske afbildning af problemet vises på figur 3.3b, hvor det tydeligt fremgår af de stiplede linjer, at niveaukurven $f(x_1, x_2) = 100.000$ og den blå begrænsningslinje for opbevaring ligger parallel med

hinanden, hvilket nu viser, at der matematisk set findes et uendeligt antal løsninger, der maksimerer værdien af objektfunktionen. Bemærk at det ikke giver mening i praksis at lave halve borde (eller stole), hvilket begrænser os til løsninger der giver hele tal i denne praktiske sammenhæng.

Antag nu at produktionen rent hypotetisk ikke er begrænset opadtil for opbevaringen af de færdigproducerede borde og stole, så der er et minimum for, hvor meget der skal produceres. Dette minimum sættes nu til $1100m^3$, mens der stadig er en begrænsning på mængden af træ tilgængeligt. Det vil da give et lineært programmeringsproblem som illustreret på figur 3.4.

$$
\begin{aligned}
&\textit{Maksimer} && f(x_1, x_2) = 200x_1 + 100x_2 \\
&\textit{I forhold til} && 2x_1 + x_2 \geq 1100 \\
& && 3x_1 + 2x_2 \leq 1600 \\
& && x_1 \geq 0 \\
& && x_2 \geq 0
\end{aligned}
$$

(a)



(b)

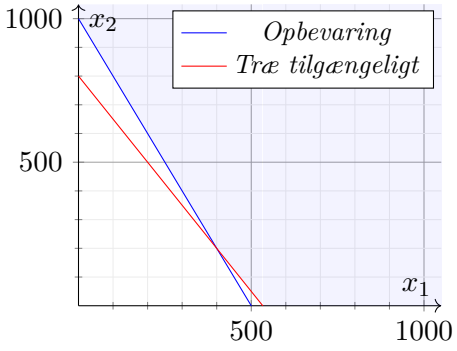
Figur 3.4: *(a) Problemet opskrevet med longhand-notation. (b) En grafisk afbildning. Bemærk at uligheden for opbevaringen nu er begrænset nedadtil med et minimum på* $1100m^3$.

Her illustreres da et lineært programmeringsproblem, hvor mulighedsområdet er en tom mængde, da der ikke findes en løsning, som opfylder betingelserne samtidig. Det er derfor ikke muligt at finde en løsning til det givne lineære programmeringsproblem.

Lad nu mængden af tilgængeligt træ være tilsvarende ubegrænset opadtil, så der minimalt set skal anvendes $1600m^3$. Da vil det lineære programmeringsproblem kunne illustreres som på figur 3.5.

$$
\begin{aligned}
&\textit{Maksimer} && f(x_1, x_2) = 200x_1 + 100x_2 \\
&\textit{I forhold til} && 2x_1 + x_2 \geq 1000 \\
& && 3x_1 + 2x_2 \leq 1600 \\
& && x_1 \geq 0 \\
& && x_2 \geq 0
\end{aligned}
$$

(a)



(b)

Figur 3.5: *(a) Problemet opskrevet med longhand-notation. (b) En grafisk afbildning. Bemærk at mulighedsområdet er blevet ubegrænset opadtil.*

Her vises et lineært programmeringsproblem, hvor betingelserne ikke begrænser, hvor stor værdien af objektfunktionen kan blive, eftersom vi kan fortsætte med at øge antallet af producerede borde og stole uden at bryde nogle af betingelserne.

Alt i alt afspejler disse fire eksempler fire overordnede muligheder der er for, hvilken type problem et givet lineært programmeringsproblem beskriver - og i hvilke tilfælde der findes en optimal løsning. Bemærk at disse muligheder også kan afspejles i minimeringsproblemer. Dette kan opsummeres ved, at et lineært programmeringsproblem generelt kan have følgende udfald.

- *Præcis én løsning*
- *Et uendeligt antal løsninger og begrænset*
- *Ingen løsninger*
- *Ubegrænset*

4 | En geometrisk og algebraisk forståelse af lineær programmering

En måde at kunne beskrive og generalisere mulighedsområdet for lineære programmeringsproblemer er gennem *konvekse mængder* og karakteristika for disse, hvorfor dette vil blive introduceret i dette afsnit. Det har endvidere rod i en *geometrisk* forståelse af lineære programmeringsproblemer, hvilket også leder frem til et argument for, at optimale løsninger findes i hjørnerne af et *polyeder*. Dette leder så senere frem til en algebraisk måde at finde optimale løsninger på. Dette kapitel tager udgangspunkt i [6, p. 1-79], medmindre andet fremgår af teksten.

4.1 Konvekse kombinationer og konvekse mængder

Konvekse mængder bygger i første omgang på *konvekse kombinationer*.

Definition 4.1.1. ***Konvekse kombinationer***
En konveks kombination af vektorer $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_k}$ er en ny vektor $\mathbf{x}$ givet ved

$$\mathbf{x} = \sum_{i=1}^{k} \lambda_i \mathbf{x_i}, \tag{4.1}$$

hvor $\lambda_i \in [1, 0]$ for $i \in \{1, 2, \ldots, k\}$ og $\sum_{i=1}^{k} \lambda_i = 1$.

Denne definition leder så frem til definitionen af konvekse mængder og disses egenskaber.

Definition 4.1.2. ***Konvekse mængder***
En mængde S af vektorer siges at være konveks, hvis der for ethvert par af vektorer $\mathbf{x_i}, \mathbf{x_j} \in S$ gælder, at

$$\mathbf{x} = \lambda \mathbf{x_i} + (1 - \lambda)\mathbf{x_j} \tag{4.2}$$

også tilhører S for ethvert $0 \leq \lambda \leq 1$.

Bemærk af definition 4.1.2 at en mængde er konveks, hvis enhver vektor $\mathbf{x}$ på linjen mellem $\mathbf{x_i}$ og $\mathbf{x_j}$ også er indeholdt i mængden. En intuitiv forståelse af konvekse mængder kan eksempelvis gives ud fra en kugle i rummet, da man for to vilkårlige vektorer der er indeholdt i kuglen har, at linjen mellem disse vektorer også vil være indeholdt i kuglen. Dog er mængden kun bestående af vektorer på kuglens overflade ikke en konveks mængde, da konvekse kombinationer af to vilkårlige vektorer på kuglens overflade ikke er indeholdt i kuglens overflade. Mulighedsområdet for lineære programmeringsproblemer kan beskrives ved en sådan konveks mængde, hvorfor det her er væsentligt at få den geometriske forståelse for dette mulighedsområde på plads.

4.2 Polyeder

Definition 4.2.1. ***Polyeder***
En mængde P udgør et *polyeder*, hvis der gælder, at $P = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} * \mathbf{b}\}$, hvor $\mathbf{b} \in \mathbb{R}^m$ og A er en $m \times n$ matrix. Et polyeder $P \subset \mathbb{R}^n$ siges at være *bundet*, hvis der eksisterer et tal $M \in \mathbb{R}$ sådan, at der for enhver vektor $\mathbf{x} \in P$ gælder, at $|\mathbf{x}| \leq M$.

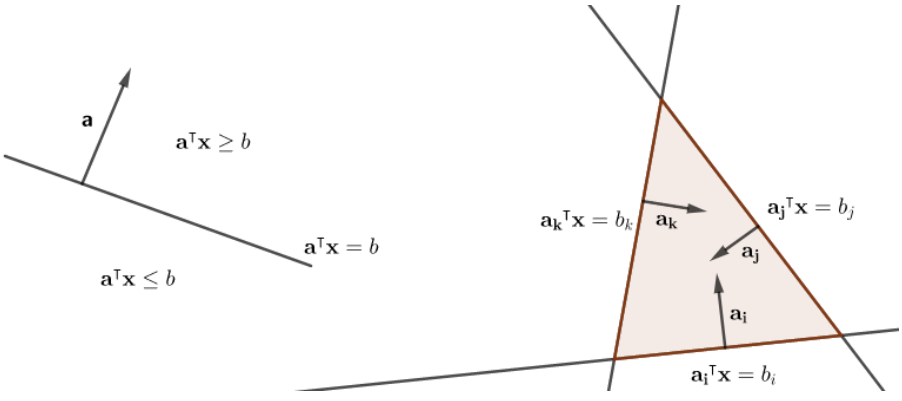
Bemærk som det første af definition 4.2.1 at et polyeder her beskriver en mængde, der svarer til mulighedsområdet for lineære programmeringsproblemer på generel form. Bemærk desuden ud fra figur 3.5 i afsnit 3.3 at polyederet her ikke er bundet, fordi der ikke findes et vilkårligt stort tal $M \in \mathbb{R}$ sådan, at der for enhver vektor $\mathbf{x} \in P$ gælder, at $|\mathbf{x}| \leq M$. For lineære programmeringsproblemer på standard form kan det tilsvarende *polyeder på standard form* beskrives ved mængden $P = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$. Desuden skal det bemærkes, at der ikke er nogen begrænsning på dimensionen n, som det pågældende polyeder kan beskrive, hvorfor det for større n bliver vanskeligt at forestille sig disse mængder.

Vi definerer nu to vigtige geometriske størrelser, som indgår i beskrivelsen af et polyeder.

Definition 4.2.2. ***Hyperplaner og halvrum***
Givet en vektor $\mathbf{a} \neq \mathbf{0}$ i $\mathbb{R}^n$ og $b \in \mathbb{R}$ så gælder der, at

1. Et *hyperplan* er givet ved mængden $\{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{a}^\intercal\mathbf{x} = b\}$.
2. Et *halvrum* er givet ved mængden $\{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{a}^\intercal\mathbf{x} \geq b\}$.

Bemærk at et halvrum også kan være givet ved mængden $\{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{a}^\intercal\mathbf{x} \leq b\}$, men vi bruger i det følgende definitionen af et halvrum som givet i definition 4.2.2. I en geometrisk forstand er et halvrum derfor en mængde af vektorer $\mathbf{x} \in \mathbb{R}^n$, der opfylder en givet ulighedsbetingelse $\mathbf{a}^\intercal\mathbf{x} \geq b$, og hvor det tilhørende hyperplan givet ved $\mathbf{a}^\intercal\mathbf{x} = b$ er grænseplanet, der deler $\mathbb{R}^n$ i to, så den ene halvdel netop udgør halvrummet. Et polyeder er da fællesmængden af de halvrum, der bliver skabt ud fra betingelserne givet i et lineært programmeringsproblem som vist i figur 4.1 herunder.



Figur 4.1: *(til venstre) Et hyperplan* $\mathbf{a}^\intercal\mathbf{x} = b$ *der deler rummet i to halvrum. (til højre) En fællesmængde af* 3 *halvrum der giver et polyeder* $P = \{\mathbf{x} \mid \mathbf{a_i}\mathbf{x} \geq b_i, \mathbf{a_j}\mathbf{x} \geq b_j, \mathbf{a_k}\mathbf{x} \geq b_k\}$.

Et vigtigt resultat af den intuitive geometriske visualisering af et polyeder som en fællesmængde af halvrum er, at mængden udgjort af et polyeder er konvekst.

Sætning 4.2.3. Ethvert polyeder er en konveks mængde.

Bevis. Vi vil først bevise, at ethvert halvrum er en konveks mængde, da et polyeder kan beskrives som fællesmængden af et endeligt antal halvrum. Antag derfor at $\mathbf{u}$ og $\mathbf{v}$ begge er indeholdt i det samme halvrum, så $\mathbf{a}^\intercal\mathbf{u} \geq b$ og $\mathbf{a}^\intercal\mathbf{v} \geq b$. Da skal vi nu vise, at enhver konveks kombination $\mathbf{w} = \lambda\mathbf{u} + (1-\lambda)\mathbf{v}$ for $\lambda \in [0,1]$ også er indeholdt i det samme halvrum, hvor vi da får, at

$$\begin{aligned}\mathbf{a}^\intercal(\lambda\mathbf{u} + (1-\lambda)\mathbf{v}) &= \lambda\mathbf{a}^\intercal\mathbf{u} + (1-\lambda)\mathbf{a}^\intercal\mathbf{v} \\ &\geq \lambda b + (1-\lambda)b \\ &= \lambda b + b - \lambda b \\ &= b\end{aligned}$$

Så ethvert halvrum er en konveks mængde, da enhver konveks kombination af to vektorer i et halvrum også er indeholdt i halvrummet.

Vi vil nu vise, at fællesmængden af et endeligt antal konvekse mængder er konvekst. Givet konvekse mængder H_i for $i \in I$ hvor I er en givet indeksmængde antager vi, at $\mathbf{u}$ og $\mathbf{v}$ er vektorer, som er indeholdt i fællesmængden af de givne konvekse mængder, så $\mathbf{u}, \mathbf{v} \in H_1 \cap H_2 \cap \cdots \cap H_n$ for $I = \{1, 2, \ldots, m\}$. Da enhver konveks mængde H_i er konveks gælder der så, at $\lambda\mathbf{u} + (1-\lambda)\mathbf{v} \in H_i$, eftersom $\mathbf{u}, \mathbf{v} \in H_i$, og det beviser derfor, at fællesmængden af de konvekse mængder også er konvekst, da det er blevet antaget, at $\mathbf{u}$ og $\mathbf{v}$ er indeholdt i fællesmængden. Da ethvert polyeder er en fællesmængde af et endeligt antal halvrum, der alle er konvekse, følger det derfor, at ethvert polyeder er konvekst. ■

Dette er et gennemgående resultat, som spiller en stor rolle for definitionen af optimale løsninger, der vil blive introduceret senere i dette afsnit. Ligeledes er det følgende resultat også vigtigt i denne sammenhæng.

Sætning 4.2.4. I en konveks mængde vil konvekse kombinationer af et endeligt antal elementer også være i mængden.

Bevis. Vi skal bevise, at konvekse kombinationer af mere end to punkter (eller mere generelt et endeligt antal punkter) er indeholdt i mængden. Dette gør vi nu ved induktion. Basisskridt: Ud fra definition 4.1.2 vides det, at en konveks kombination af to punkter i en konveks mængde P også er indeholdt i mængden, og dermed er basisskridtet opfyldt. Induktionshypotesen: Vi antager, at en konveks kombination af k elementer $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_k}$ er indeholdt i mængden, så der gælder, at

$$\sum_{i=1}^{k} \lambda_i \mathbf{x_i} \in P,$$

hvor $\lambda_i \in [0,1]$ for $i \in \{1, 2, \ldots, k\}$ og $\sum_{i=1}^{k} \lambda_i = 1$.

Induktionsskridt: Vi ser på $k+1$ elementer $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_{k+1}}$ i en konveks mængde P og lader $\lambda_1, \lambda_2, \ldots, \lambda_{k+1}$ være ikke-negative skalarer, så $\sum_{i=1}^{k+1} \lambda_i = 1$. Vi går her ud fra, at $\lambda_{k+1} \neq 1$, da der gælder, at hvis $\lambda_{k+1} = 1$ så er $\lambda_i = 0$ for $i \in \{1, 2, \ldots, k\}$, hvorfor den konvekse kombination giver $\mathbf{x_{k+1}}$, der allerede er indeholdt i mængden P. Fra induktionshypotesen ved vi, at en konveks kombination af k elementer er indeholdt i P, og vi vil bevise, at en konveks kombinationen af $k+1$ elementer også er i P, så

$$\sum_{i=1}^{k+1} \lambda_i \mathbf{x_i} \in P. \tag{4.3}$$

Vi kigger nu på summen $\sum_{i=1}^{k+1} \lambda_i$, som vi ved skal være lig med 1, hvorfor vi kan udlede følgende vigtige argument.

$$\sum_{i=1}^{k+1} \lambda_i = 1 \Leftrightarrow \sum_{i=1}^{k} \lambda_i + \lambda_{k+1} = 1 \Leftrightarrow \sum_{i=1}^{k} \lambda_i = 1 - \lambda_{k+1} \Leftrightarrow \sum_{i=1}^{k} \frac{\lambda_i}{1-\lambda_{k+1}} = 1$$

Vi betragter så igen summen i ligning 4.3. For denne sum tages det $k+1$'te element ud af summen, så

$$\sum_{i=1}^{k+1} \lambda_i \mathbf{x_i} = \lambda_{k+1}\mathbf{x_{k+1}} + \sum_{i=1}^{k} \lambda_i \mathbf{x_i}$$

Vi ved her, at $\lambda_{k+1} \neq 1$ ud fra vores antagelse og vil da gerne have summen $\sum_{i=1}^{k} \lambda_i \mathbf{x_i}$ på formen $\sum_{i=1}^{k} \frac{\lambda_i}{1-\lambda_{k+1}} \mathbf{x_i}$, hvilket gøres ved at multiplicere $\sum_{i=1}^{k} \lambda_i \mathbf{x_i}$ med $\frac{1-\lambda_{k+1}}{1-\lambda_{k+1}}$ sådan, at

$$\sum_{i=1}^{k+1} \lambda_i \mathbf{x_i} = \lambda_{k+1}\mathbf{x_{k+1}} + \frac{1-\lambda_{k+1}}{1-\lambda_{k+1}} \sum_{i=1}^{k} \lambda_i \mathbf{x_i} \iff$$

$$\sum_{i=1}^{k+1} \lambda_i \mathbf{x_i} = \lambda_{k+1}\mathbf{x_{k+1}} + (1-\lambda_{k+1}) \sum_{i=1}^{k} \frac{\lambda_i}{1-\lambda_{k+1}} \mathbf{x_i} \tag{4.4}$$

Da $\sum_{i=1}^{k} \frac{\lambda_i}{1-\lambda_{k+1}} = 1$ ved vi fra induktionshypotesen, at $\sum_{i=1}^{k} \frac{\lambda_i}{1-\lambda_{k+1}} \mathbf{x_i} \in P$. Det medfører da, at ligning 4.4 er en konveks kombination af $\mathbf{x_{k+1}}$ og $\sum_{i=1}^{k} \frac{\lambda_i}{1-\lambda_{k+1}} \mathbf{x_i}$ som begge er indeholdt i den konvekse mængde P, hvorfor $\sum_{i=1}^{k+1} \lambda_i \mathbf{x_i} \in P$, og induktionsskridtet er fuldendt, hvilket afslutter beviset. ■

Som en afslutning på dette afsnit defineres nu et *konvekst hylster*.

Definition 4.2.5. ***Konvekse hylstre***
Givet en mængde af vektorer $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_k}$ er det konvekse hylster mængden af alle konvekse kombinationer af denne mængde af vektorer.

Et vigtigt resultat som er bevist i [6, p. 68-70] er, at et bundet polyeder er det konvekse hylster af "hjørnepunkterne" i polyederet, hvilket leder frem til introduktionen af *ekstremumspunkter*.

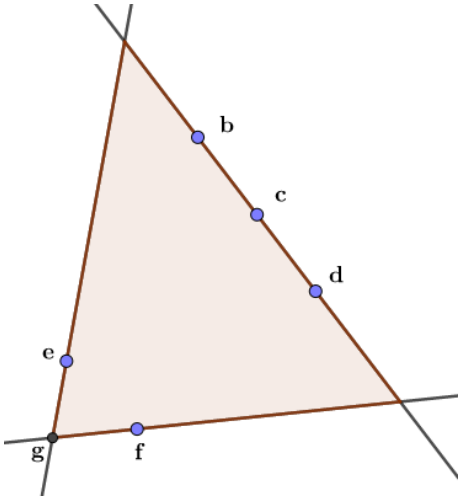
4.3 Ekstremumspunkter og basale løsninger

I det følgende vil resultatet af sætning 4.2.3 blandt andet være en del af argumentationen for eksistensen af optimale løsninger for objektfunktionen i ekstremumspunkter, hvorfor disse først vil blive defineret.

Definition 4.3.1. ***Ekstremumspunkter***
Givet et polyeder P, da er et ekstremumspunkt givet ved en vektor $\mathbf{x} \in P$, hvorom der gælder, at hvis $\mathbf{u}, \mathbf{v} \in P$ og $\mathbf{x} = \lambda\mathbf{u} + (1-\lambda)\mathbf{v}$ for $\lambda \in [0,1]$ så er $\mathbf{u} = \mathbf{x}$ eller $\mathbf{v} = \mathbf{x}$.

Bemærk at $\mathbf{x}$ er et ekstremumspunkt i et polyeder P, hvis man ikke kan beskrive $\mathbf{x}$ som en konveks kombination af to andre vektorer i P, der begge er forskellige fra $\mathbf{x}$.

Givet et polyeder der repræsenterer et endeligt antal betingelser vil antallet af ekstremumspunkter i polyederet også være endeligt, hvilket der kommer et argument for senere i det indeværende afsnit. Bemærk desuden af definition 4.3.1 at et ekstremumspunkt svarer til den intuitive forståelse af et "hjørnepunkt" i polyederet, som det fremgår af figur 4.2.



Figur 4.2: $\mathbf{g}$ *er et ekstremumspunkt, eftersom* $\mathbf{g}$ *ikke kan beskrives som en konveks kombination af andre vektorer i* P *såsom* $\mathbf{e}$ *og* $\mathbf{f}$*.* $\mathbf{c}$ *er ikke et ekstremumspunkt, eftersom* $\mathbf{c}$ *kan beskrives som en konveks kombination af andre vektorer i* P *såsom* $\mathbf{b}$ *og* $\mathbf{d}$*.*

En vigtig egenskab for ekstremumspunkter i et polyeder er, at objektfunktionen vil antage en optimal værdi i én eller flere af disse ekstremumspunkter, hvis det pågældende polyeder er bundet. Dette er centralt for forståelsen af Simplex-metoden senere i projektet, hvorfor det her gengives i en sætning [8, p. 72-73].

Sætning 4.3.2. For en søjle-vektor $\mathbf{c} = \begin{bmatrix} c_1 & c_2 & \cdots & c_n \end{bmatrix}^\intercal$ og $\mathbf{x} \in \mathbb{R}^n$ vil objektfunktionen $f(\mathbf{x}) = \mathbf{c}^\intercal\mathbf{x}$ for et lineært programmeringsproblem antage sit optimum ved et ekstremumspunkt, når polyederet P er bundet.

Bevis. Vi afgrænser her til kun at anskue et minimeringsproblem, da argumentet for et maksimeringsproblem er det samme. Vi antager desuden, at polyederet P er bundet, så der eksisterer et endeligt antal ekstremumspunkter i polyederet. Ekstremumspunkterne noteres som $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_k}$, og minimumsværdien af objektfunktionen findes ved den optimale løsning $\mathbf{x_0}$. Da gælder det så, at $f(\mathbf{x_0}) \leq f(\mathbf{x})$ for alle $\mathbf{x} \in P$.

Hvis $\mathbf{x_0}$ er et ekstremumspunkt, er sætningen allerede bevist. Vi antager så, at $\mathbf{x_0}$ ikke er et ekstremumspunkt og vil så komme frem til, at objektfunktionen stadig antager sit minimum i et ekstremumspunkt $\mathbf{x_m} \in \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_k}\}$. Under antagelse af at $\mathbf{x_0}$ ikke er et ekstremumspunkt, kan $\mathbf{x_0}$ beskrives som en konveks kombination af ekstremumspunkterne $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_k}$, da P netop udgøres af det konvekse hylster af ekstremumspunkterne under antagelse af, at P er bundet. Det medfører altså, at $\mathbf{x_0}$ kan beskrives som

$$\mathbf{x_0} = \sum_{i=1}^{k} \lambda_i \mathbf{x_i}$$

for $\lambda_i \in [0, 1[$ og $\sum_{i=1}^{k} \lambda_i = 1$. Da vi endvidere ved, at objektfunktionen $f(\mathbf{x})$ er en lineær afbildning [8, p. 70] gælder der, at

$$\begin{aligned} f(\mathbf{x_0}) &= f\left(\sum_{i=1}^{k} \lambda_i \mathbf{x_i}\right) \\ &= \lambda_1 f(\mathbf{x_1}) + \lambda_2 f(\mathbf{x_2}) + \cdots + \lambda_k f(\mathbf{x_k}) = m, \end{aligned} \tag{4.5}$$

hvor m er minimumsværdien af objektfunktionen. Da $f(\mathbf{x_0})$ giver minimumsværdien i polyederet og $\lambda_i \in [0, 1[$ for $i \in \{1, 2, \ldots, k\}$, kan ligning 4.5 ændres, så ekstremumspunktet $\mathbf{x_m}$ hvor $f(\mathbf{x_m}) = \min(f(\mathbf{x_i}))$ for $i \in \{1, 2, \ldots, k\}$ substitueres for alle $\mathbf{x_i} \neq \mathbf{x_m}$ i ligning 4.5. Der må så gælde, at

$$f(\mathbf{x_0}) \geq \lambda_1 f(\mathbf{x_m}) + \lambda_2 f(\mathbf{x_m}) + \cdots + \lambda_k f(\mathbf{x_m}) = f(\mathbf{x_m}), \tag{4.6}$$

da $\sum_{i=1}^{k} \lambda_i = 1$, og hvor det bemærkes, at $f(\mathbf{x_0}) \geq f(\mathbf{x_m})$, da substitueringen med $\mathbf{x_m}$ ikke gør udtrykket på højresiden af ulighedstegnet i ligning 4.6 større. Da $f(\mathbf{x_0}) \geq f(\mathbf{x_m})$ og der samtidig gælder, at $f(\mathbf{x_0}) \leq f(\mathbf{x_m})$ ud fra konstruktionen af $\mathbf{x_0}$, så skal $f(\mathbf{x_0}) = f(\mathbf{x_m}) = m$. Vi har da bevist, at objektfunktionen også antager sit optimum i et ekstremumspunkt, selvom $\mathbf{x_0}$ ikke er et ekstremumspunkt. ■

At definere ekstremumspunkter som gjort i definition 4.3.1 er en geometrisk måde at forklare, hvor objektfunktionen vil antage optimale værdier, hvis disse eksisterer, ud fra et givent lineært programmeringsproblem. For senere at kunne beskrive den systematiske Simplex-metode, der undersøger optimum ved bestemte ekstremumspunkter i polyederet, er det dog nødvendigt også at komme med en algebraisk definition af ekstremumspunkter. Dertil er det specielt de givne betingelser i det lineære programmeringsproblem, der skal være med til at kunne definere, hvornår en løsning er optimal.

Definition 4.3.3. ***Aktive betingelser***
Givet et polyeder P med betingelser på formen $\mathbf{a_i}^\intercal \mathbf{x} * b_i$ er en bestemt betingelse *aktiv* for en løsning $\mathbf{x}$, hvis der gælder, at

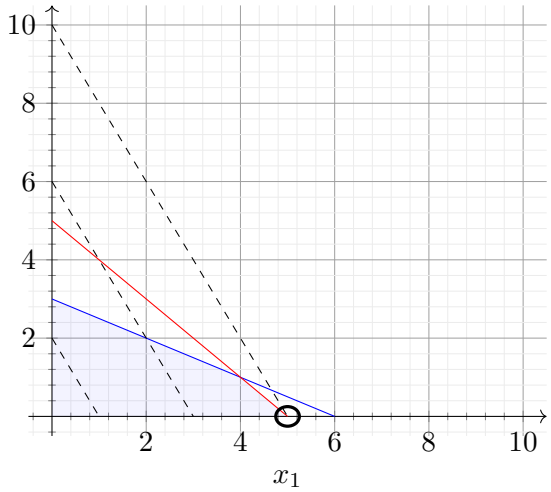
$$\mathbf{a_i}^\intercal \mathbf{x} = b_i \tag{4.7}$$

Bemærk at ikke-negative begrænsninger på formen $x_i \geq 0$ for $i \in \{1, 2, \ldots, n\}$ er aktive, hvis $x_i = 0$.

Eksempel 4.3.4. For en visuel forståelse af definition 4.3.3 illustrerer figur 4.3 et eksempel, hvor $\mathbf{x} = \begin{bmatrix} 5 & 0 \end{bmatrix}^\intercal$ er markeret med en cirkel i grafen.

Maksimer	$f(x_1, x_2) =$	$x_1 + 2x_2$
I forhold til		$2x_1 + x_2 \leq 3$
		$x_1 + x_2 \leq 5$
Og		$x_1, x_2 \geq 0$

(a)



(b)

Figur 4.3: *(a) Et problem opskrevet med longhand-notation. (b) En grafisk afbildning.*

Her ses det, at de aktive betingelser for $\mathbf{x}$ er $x_2 \geq 0$ og $x_1 + x_2 = 5$, da disse betingelser er opfyldt med ligheder jævnfør definition 4.3.3. Bemærk af definition 4.3.3 at dette svarer til, at hvis $\mathbf{x} \in \mathbb{R}^n$ har n forskellige aktive betingelser hvis vektorer $\mathbf{a_i}$ er lineært uafhængige, så opfylder $\mathbf{x}$ et lineært ligningssystem bestående af n ligninger. Der gælder så følgende sætning.

Sætning 4.3.5. Givet $\mathbf{x} \in \mathbb{R}^n$ og en indeksmængde $I = \{i \mid \mathbf{a_i}^\intercal \mathbf{x} = b_i\}$ for de aktive betingelser for $\mathbf{x}$ er følgende udsagn ækvivalente.

1. For mængden af vektorer $\{\mathbf{a_i} \mid i \in I\}$ findes der n vektorer, som er lineært uafhængige.
2. $\text{Span}\{\mathbf{a_i} \mid i \in I\} = \mathbb{R}^n$
3. For $i \in I$ har det lineære ligningssystem givet ved betingelserne $\mathbf{a_i}^\intercal \mathbf{x} = b_i$ præcis én løsning.

Bevis. For at vise at (1) $\iff$ (2) antages det først, at der eksisterer n vektorer i mængden $\{\mathbf{a_i} \mid i \in I\}$, som er lineært uafhængige. Da følger det tydeligt, at spannet af disse n vektorer er hele $\mathbb{R}^n$, eftersom de n vektorer udgør en basis for $\mathbb{R}^n$ jævnfør definition 2.5.8. Antages det modsat, at $\text{Span}\{\mathbf{a_i} \mid i \in I\} = \mathbb{R}^n$ betyder det, at $\{\mathbf{a_i} \mid i \in I\}$ kan udtyndes til en basis for $\mathbb{R}^n$ bestående af n lineært uafhængige vektorer jævnfør sætning 2.5.9.

(2) $\iff$ (3) vises ved kontraposition, så det antages først, at et lineært ligningssystem givet ved betingelserne $\mathbf{a_i}^\intercal \mathbf{x} = b_i$ for $i \in I$ har to forskellige løsninger $\mathbf{x_1}, \mathbf{x_2} \in \mathbb{R}^n$, og vi

skal så komme frem til, at Span$\{\mathbf{a_i} \mid i \in I\} \neq \mathbb{R}^n$. Givet de to løsninger $\mathbf{x_1}, \mathbf{x_2} \in \mathbb{R}^n$ gælder der for $\mathbf{v} = \mathbf{x_1} - \mathbf{x_2}$, at $\mathbf{v} \neq \mathbf{0}$, $\mathbf{v} \in \mathbb{R}^n$ og $\mathbf{a_i}^\intercal \mathbf{v} = 0$ for alle $i \in I$, da $\mathbf{a_i}^\intercal(\mathbf{x_1} - \mathbf{x_2}) = \mathbf{a_i}^\intercal \mathbf{x_1} - \mathbf{a_i}^\intercal \mathbf{x_2} = b_i - b_i = 0$. Så er $(\sum_{i \in I} c_i \mathbf{a_i})^\intercal \mathbf{v} = 0$ for ethvert valg af c_i for $i \in I$, hvorfor $\mathbf{v} \notin$ Span$\{\mathbf{a_i} \mid i \in I\}$, da $\mathbf{v}^\intercal \mathbf{v} \neq 0$. Derfor må der gælde, at Span$\{\mathbf{a_i} \mid i \in I\} \neq \mathbb{R}^n$.

Hvis det modsat antages, at Span$\{\mathbf{a_i} \mid i \in I\} \neq \mathbb{R}^n$ eksisterer der en vektor $\mathbf{v} \in \mathbb{R}^n$ sådan, at $\mathbf{a_i}^\intercal \mathbf{v} = 0$ for alle $i \in I$. Da må det betyde, at hvis en vektor $\mathbf{x}$ opfylder alle betingelser på formen $\mathbf{a_i}^\intercal \mathbf{x} = b_i$ for $i \in I$, så vil $\mathbf{a_i}^\intercal(\mathbf{x} + \mathbf{v}) = b_i$ også være opfyldt, hvilket viser, at der eksisterer flere løsninger til det lineære ligningssystem givet ved betingelserne $\mathbf{a_i}^\intercal \mathbf{x} = b_i$ for $i \in I$. ■

Bemærk af sætning 4.3.5 (1), at der skal eksistere n lineært uafhængige vektorer $\mathbf{a_i}$ for $i \in I$, før disse aktive betingelser giver præcis én løsning $\mathbf{x}$. Dette gør det nu muligt at definere *basale løsninger* og *basal mulige løsninger* i et polyeder.

Definition 4.3.6. ***Basale løsninger og basal mulige løsninger***
Givet et polyeder $P = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} * \mathbf{b}\}$, hvor A er en $m \times n$ matrix og $\mathbf{b} \in \mathbb{R}^m$. Da gælder så, at

1. $\mathbf{x}$ er en basal løsning, hvis alle lighedsbetingelser er aktive, og der i alt eksisterer n lineært uafhængige betingelser $\mathbf{a_i}$, som er aktive ved $\mathbf{x}$.
2. $\mathbf{x}$ er en basal mulig løsning, hvis $\mathbf{x}$ er en basal løsning og opfylder alle betingelser i P.

I grafen på figur 4.3 i eksempel 4.3.4 ser vi da, at $\mathbf{x} = \begin{bmatrix} 6 & 0 \end{bmatrix}^\intercal$ er et eksempel på en basal løsning, som ikke er mulig, da betingelsen $x_1 + x_2 \leq 5$ ikke er opfyldt. Tilsvarende er $\mathbf{x} = \begin{bmatrix} 5 & 0 \end{bmatrix}^\intercal$ et eksempel på en basal mulig løsning, eftersom alle betingelserne i polyederet er opfyldt. Vi laver nu en præcisering af basale og basal mulige løsninger for polyeder på standard form, hvilket vi gør ud fra følgende sætning.

Sætning 4.3.7. Givet et polyeder $P = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ hvor det antages, at rækkerne i $m \times n$ matricen A er lineært uafhængige. Da gælder det, at $\mathbf{x}$ er en basal løsning, hvis og kun hvis alle lighedsbetingelser er opfyldt, samt at der eksisterer indekstal $B(1), B(2), \ldots, B(m)$ sådan, at

1. $\mathbf{A_i}$ for $i \in \{B(1), B(2), \ldots, B(m)\}$ er lineært uafhængige.
2. $x_i = 0$ for $i \notin \{B(1), B(2), \ldots, B(m)\}$.

Bevis. Vi antager, at (1) og (2) samt $A\mathbf{x} = \mathbf{b}$ er opfyldt og skal så vise, at $\mathbf{x}$ er en basal løsning. Vi har da, at $x_i = 0$ for $i \notin \{B(1), B(2), \ldots, B(m)\}$, hvorfor der må gælde, at

$$A\mathbf{x} = \sum_{i=1}^{n} \mathbf{A_i} x_i = \sum_{i=1}^{m} \mathbf{A_{B(i)}} x_{B(i)} \iff$$

$$\sum_{i=1}^{m} \mathbf{A_{B(i)}} x_{B(i)} = \mathbf{b} \tag{4.8}$$

Da vi samtidig har antaget, at $\mathbf{A_i}$ for $i \in \{B(1), B(2), \ldots, B(m)\}$ er lineært uafhængige, har ligning 4.8 præcis én løsning jævnfør sætning 2.3.2, og $\mathbf{x}$ er derfor entydigt bestemt, når $x_i = 0$ for $i \notin \{B(1), B(2), \ldots, B(m)\}$. Sætning 4.3.5 medfører så, at der eksisterer n lineært uafhængige aktive betingelser, hvoraf nogle af de aktive betingelser kommer af de ikke-negative begrænsninger ved $x_i = 0$ for $i \notin \{B(1), B(2), \ldots, B(m)\}$. Jævnfør definition 4.3.6 er $\mathbf{x}$ da en basal løsning.

Vi antager nu, at $\mathbf{x}$ er en basal løsning, og vi vil så vise, at (1) og (2) er opfyldt. Lad $x_{B(1)}, x_{B(2)}, \ldots, x_{B(k)}$ være komponenter af $\mathbf{x}$ som er forskellig fra 0, sådan at $x_i = 0$ for $i \notin \{B(1), B(2), \ldots, B(k)\}$. Da $\mathbf{x}$ er en basal løsning, så vil $A\mathbf{x} = \mathbf{b}$ give en entydig løsning for de aktive betingelser, og derfor vil $\sum_{i=1}^{k} \mathbf{A_{B(i)}} x_{B(i)} = \mathbf{b}$ også give en entydig løsning. Det må nødvendigvis medføre, at søjlerne $\mathbf{A_{B(1)}}, \mathbf{A_{B(2)}}, \ldots, \mathbf{A_{B(k)}}$ er lineært uafhængige jævnfør sætning 2.3.2. Givet at A har m lineært uafhængige rækker, vil den også have m lineært uafhængige søljer, der udspænder hele $\mathbb{R}^m$, og derfor er $k \leq m$. Hvis $k < m$ kan vi jævnfør sætning 2.5.10 finde $m - k$ søjler $\mathbf{A_i}$ for $i \notin \{B(1), B(2), \ldots, B(k)\}$ sådan, at $\mathbf{A_{B(1)}}, \mathbf{A_{B(2)}}, \ldots, \mathbf{A_{B(k)}}, \mathbf{A_{k+1}}, \ldots, \mathbf{A_m}$ er lineært uafhængige, og (1) er dermed opfyldt.

For at vise at (2) også er opfyldt husker vi på, at $k \leq m$ medfører, at hvis $i \notin \{B(1), B(2), \ldots, B(m)\}$ så er $i \notin \{B(1), B(2), \ldots, B(k)\}$, og samtidig er $x_i = 0$ for $i \notin \{B(1), B(2), \ldots, B(k)\}$, hvorfor (1) og (2) i sætning 4.3.7 er opfyldt. ■

Bemærk at sætning 4.3.7 medfører, at vi kan finde en basal mulig løsning for et standard form polyeder $P = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ ved at sætte $n - m$ *ikke-basale* variable til 0 for derefter at løse for de resterende m *basale* variable, givet at de tilhørende søjler $\mathbf{A_i}$ for $i \in \{B(1), B(2), \ldots, B(m)\}$ er lineært uafhængige. Af dette får vi mindst n aktive betingelser i form af de $n - m$ ikke-negative begrænsninger samt de m lighedsbetingelser, der skal være opfyldt for $\mathbf{x} \in P$. En tilhørende basal mulig løsning for et polyeder på standard form vil da være en løsning, hvor alle $x_i \geq 0$ for $i \in \{1, 2, \ldots, n\}$. Denne definition af basale og basal mulige løsninger til polyeder på standard form overholder definition 4.3.6 under den antagelse, at rækkerne i A er lineært uafhængige [6, p. 53].

Gennem resten af projektet vil vi nu tage udgangspunkt i definitionen af basale løsninger og basal mulige løsninger ud fra polyeder på standard form, eftersom dette er formen, som Simplex-metoden senere kommer til at fungere ud fra. Vi vil da først bruge definitionen af basale løsninger og basal mulige løsninger for et polyeder på standard form til at bevise, at ekstremumspunkter og basal mulige løsninger er det samme [8, p. 73-75].

Sætning 4.3.8. Givet et ikke-tomt polyeder $P = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$, hvor A er en $m \times n$ matrix og $\mathbf{x} \in P$. Da er følgende udsagn ækvivalente.

1. $\mathbf{x}$ er en basal mulig løsning.
2. $\mathbf{x}$ er et ekstremumspunkt.

Bevis. Vi viser først, at (1) $\implies$ (2). Det antages derfor først, at $\mathbf{x}$ er en basal mulig løsning, så der eksisterer m lineært uafhængige vektorer $\mathbf{A_i}$ for $i \in \{B(1), B(2), \ldots, B(m)\}$ tilhørende de basale variable $x_{B(1)}, x_{B(2)}, \ldots, x_{B(m)}$, og hvor der derfor gælder, at

$$\sum_{i=1}^{m} \mathbf{A_{B(i)}} x_{B(i)} = \mathbf{b}$$

Her er $x_{B(1)}, x_{B(2)}, \ldots, x_{B(m)}$ givet ud fra de m lineært uafhængige søjler $\mathbf{A_i}$ for $i \in \{B(1), B(2), \ldots, B(m)\}$ og de resterende $n-m$ ikke-basale variable $x_i = 0$ for $i \notin \{B(1), B(2), \ldots, B(m)\}$, så den basal mulige løsning er givet ved

$$\mathbf{x} = \begin{bmatrix} x_{B(1)} & x_{B(2)} & \cdots & x_{B(m)} & 0 & 0 & \cdots & 0 \end{bmatrix}^\mathsf{T},$$

hvor alle $x_i \geq 0$ for $i \in \{1, 2, \ldots, n\}$. Bemærk at vi her har antaget, at det er de første m søjler, der er lineært uafhængige. For at vise at $\mathbf{x}$ er et ekstremumspunkt, antages det, at $\mathbf{x}$ ikke er et ekstremumspunkt, og vi skal så komme frem til en modstrid. Det må nødvendigvis betyde, at $\mathbf{x}$ kan skrives som en konveks kombination af to andre vektorer $\mathbf{u}, \mathbf{v} \in P$, sådan at $\mathbf{x} = \lambda\mathbf{u} + (1-\lambda)\mathbf{v}$ for $0 \leq \lambda \leq 1$. Her bemærkes det, at da de $n-m$ ikke-basale variable $x_i = 0$ for $i \notin \{B(1), B(2), \ldots, B(m)\}$, skal de tilsvarende $n-m$ variable u_i og v_i for $i \notin \{B(1), B(2), \ldots, B(m)\}$ i henholdsvis $\mathbf{u}$ og $\mathbf{v}$ også være 0, før den konvekse kombination er opfyldt. Da både $\mathbf{u}$ og $\mathbf{v}$ ligger i polyederet, opfylder de samtidig, at

$$A\mathbf{u} = \sum_{i=1}^{n} \mathbf{A_i} u_i = \sum_{i=1}^{m} \mathbf{A_{B(i)}} u_{B(i)} = \mathbf{b}$$
$$A\mathbf{v} = \sum_{i=1}^{n} \mathbf{A_i} v_i = \sum_{i=1}^{m} \mathbf{A_{B(i)}} v_{B(i)} = \mathbf{b}.$$

Men da det vides, at vektorerne $\mathbf{A_i}$ for $i \in \{B(1), B(2), \ldots, B(m)\}$ er lineært uafhængige, kan $\mathbf{b}$ udtrykkes som en entydig linearkombination af disse vektorer jævnfør sætning 2.3.2. Det medfører så, at $\mathbf{x} = \mathbf{u} = \mathbf{v}$, hvorfor $\mathbf{x}$ ikke kan udtrykkes som en konveks kombination af to andre vektorer $\mathbf{u}$ og $\mathbf{v}$, så $\mathbf{x}$ er et ekstremumspunkt jævnfør definition 4.3.1, og modstriden opnås heraf.

Vi vil nu vise, at (2) $\implies$ (1) [10], så vi antager, at $\mathbf{x} \in P$ er et ekstremumspunkt, hvor $k \leq n$ af indgangene i $\mathbf{x}$ er positive, og hvor de resterende $n-k$ indgange er 0. Vi omrokerer så indgangene i $\mathbf{x}$ og de tilhørende søljer i A, sådan at indgangene x_i og søljerne $\mathbf{A_i}$ for $i \in \{1, 2, \ldots, k\}$ er de positive indgange, mens $x_i = 0$ for $i \in \{k+1, k+2, \ldots, n\}$. Med andre ord har vi så konstrueret $\mathbf{x}$ sådan, at $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_k & 0 & 0 & \cdots & 0 \end{bmatrix}^\mathsf{T}$. Der må så gælde, at

$$A\mathbf{x} = \sum_{i=1}^{n} \mathbf{A}_i x_i = \sum_{i=1}^{k} \mathbf{A}_i x_i \iff$$
$$\sum_{i=1}^{k} \mathbf{A}_i x_i = \mathbf{b}$$

hvor vi skal bevise, at $\mathbf{A}_i$ for $i \in \{1, 2, \ldots, k\}$ er lineært uafhængige.

Vi laver et modstridsbevis, hvor vi antager, at $\mathbf{A}_i$ for $i \in \{1, 2, \ldots, k\}$ er linært afhængige, og der eksisterer derfor l_i for $i \in \{1, 2, \ldots, k\}$, hvor en eller flere l_i er forskellig fra 0 jævnfør definition 2.3.1 sådan, at

$$\sum_{i=1}^{k} \mathbf{A_i} l_i = \mathbf{0} \tag{4.9}$$

Vi lader så $l_i = 0$ for $i \in \{k+1, k+2, \ldots, n\}$ og bruger så vektoren

$$\mathbf{l} = \begin{bmatrix} l_1 & l_2 & \cdots & l_k & 0 & 0 & \cdots & 0 \end{bmatrix}^\mathsf{T}$$

til at konstruere to forskellige mulige vektorer $\overline{\mathbf{x}}_\mathbf{1}$ og $\overline{\mathbf{x}}_\mathbf{2}$. Vi ønsker da at vælge et ϵ sådan, at $\overline{x}_i = x_i \pm \epsilon l_i > 0$ for $i \in \{1, 2, \ldots, k\}$, og sådan at $\overline{x}_i \neq x_i$ for mindst et $i \in \{1, 2, \ldots, k\}$. Vi kan sætte $\epsilon > 0$, for så bliver $\overline{x}_i \neq x_i$ for mindst et $i \in \{1, 2, \ldots, k\}$, og for at sikre at $\epsilon > 0$ tager vi absolutværdien af l_i. Vi finder så det størst mulige ϵ ud fra

$$\overline{x}_i > 0 \iff x_i - \epsilon|l_i| > 0 \iff x_i > \epsilon|l_i| \iff \frac{x_i}{|l_i|} > \epsilon.$$

Vi skal her være opmærksom på, at dette samtidig skal gælde for alle $i \in \{1, 2, \ldots, k\}$, og derfor anvendes det mindste ϵ, så vi får, at

$$0 < \epsilon < \min_{l_i \neq 0} \frac{x_i}{|l_i|}.$$

Vi konstruerer så $\overline{\mathbf{x}}_\mathbf{1} = \mathbf{x} + \epsilon\mathbf{l}$ og $\overline{\mathbf{x}_\mathbf{2}} = \mathbf{x} - \epsilon\mathbf{l}$, hvor det bemærkes, at $\overline{\mathbf{x}}_\mathbf{1}$ og $\overline{\mathbf{x}}_\mathbf{2}$ tilhører polyederet, da

$$\begin{aligned} A\overline{\mathbf{x}}_\mathbf{1} &= A(\mathbf{x} + \epsilon\mathbf{l}) = A\mathbf{x} + A(\epsilon\mathbf{l}) = A\mathbf{x} + \epsilon(A\mathbf{l}) = A\mathbf{x} + \mathbf{0} = \mathbf{b} \\ A\overline{\mathbf{x}}_\mathbf{2} &= A(\mathbf{x} - \epsilon\mathbf{l}) = A\mathbf{x} - A(\epsilon\mathbf{l}) = A\mathbf{x} - \epsilon(A\mathbf{l}) = A\mathbf{x} - \mathbf{0} = \mathbf{b}, \end{aligned}$$

da $A\mathbf{l} = \mathbf{0}$ ud fra ligning 4.9. Men vi har også, at

$$\frac{1}{2}\overline{\mathbf{x}}_\mathbf{1} + \frac{1}{2}\overline{\mathbf{x}}_\mathbf{2} = \frac{1}{2}(\mathbf{x} + \epsilon\mathbf{l} + \mathbf{x} - \epsilon\mathbf{l}) = \frac{1}{2}(2\mathbf{x}) = \mathbf{x},$$

og derfor kan $\mathbf{x}$ beskrives som en konveks kombination af $\overline{\mathbf{x}}_\mathbf{1}$ og $\overline{\mathbf{x}}_\mathbf{2}$ for $\lambda_1 = \lambda_2 = \frac{1}{2}$. Da vi har antaget, at $\mathbf{x}$ er et ekstremumspunkt og nu har fundet frem til, at $\mathbf{x}$ kan beskrives ved en konveks kombination af $\overline{\mathbf{x}}_\mathbf{1}$ og $\overline{\mathbf{x}}_\mathbf{2}$, fremkommer modstriden jævnfør definition 4.3.1, og $\mathbf{x}$ er altså ikke et ekstremumspunkt, hvis søjlerne $\mathbf{A}_i$ for $i \in \{1, 2, \ldots, k\}$ er lineært afhængige. Derfor må søjlerne $\mathbf{A}_i$ for $i \in \{1, 2, \ldots, k\}$ være lineært uafhængige, og vi har derfor en basal mulig løsning jævnfør sætning 4.3.7.

■

Vi kan nu tilsvarende komme med et argument for, at et polyeder på standard form beskrevet ved et endeligt antal betingelser og begrænsninger indeholder et endeligt antal basal mulige løsninger og derfor også et endeligt antal ekstremumspunkter. Bemærk da at en øvre grænse for antallet af basale løsninger er givet ved [8, p. 76-77]

$$\binom{n}{m} = \frac{n!}{m!(n-m)!} \tag{4.10}$$

da det svarer til antallet af måder, hvorpå vi kan udvælge m basale variable ud af de n variable. Læg da mærke til at binomialkoefficienten i ligning 4.10 i værste tilfælde er antallet af basale løsninger, vi skal undersøge for at verificere den optimale løsning, givet at der ikke findes en anden systematisk måde at gøre dette på. Dette er et vigtigt argument for, at Simplex-metoden som beskrives i det følgende kapitel er en mere effektiv måde at finde den optimale basal mulige løsning på.

Et vigtigt resultat ud fra de foregående beskrivelser af basale løsninger og basal mulige løsninger gengives i det følgende.

4.3.9. Givet et ikke-tomt bundet polyeder eller et ikke-tomt polyeder på standard form vil der altid eksistere mindst én basal mulig løsning.

For senere at kunne systematisere hvilke variable der er basale, og hvilke der er ikke-basale i en basal mulig løsning, introduceres her en *basis-matrix* B givet ved de pågældende m lineært uafhængige søjler $\mathbf{A_i}$ i A som

$$B = \begin{bmatrix} \mathbf{A_{B(1)}} & \mathbf{A_{B(2)}} & \cdots & \mathbf{A_{B(m)}} \end{bmatrix}$$

for de basale variable $x_{B(i)}$ for $i \in \{1, 2, \ldots, m\}$. Vi definerer endvidere en vektor $\mathbf{x_B} \in \mathbb{R}^m$ som den vektor, der indeholder de pågældende basale variable tilhørende søjlerne i basismatricen B. Og $\mathbf{x_B}$ er da givet ved

$$\mathbf{x_B} = \begin{bmatrix} x_{B(1)} \\ x_{B(2)} \\ \vdots \\ x_{B(m)} \end{bmatrix}$$

Givet en basal løsning $\mathbf{x}$ i et polyeder på standard form ved vi, at de $n - m$ ikke-basale variable er 0, hvorfor løsningen $\mathbf{x}$ kan findes ved blot at bestemme $\mathbf{x_B}$ ud fra ligningen

$$B\mathbf{x_B} = \mathbf{b},$$

hvor $\mathbf{b}$ er en givet betingelsevektor for det lineære programmeringsproblem. Bemærk da at B er en invertibel matrix jævnfør sætning 2.6.7, da vektorerne i B er lineært uafhængige, hvorfor $\mathbf{x_B}$ kan findes som den entydige løsning til den tilsvarende ligning

$$\mathbf{x_B} = B^{-1}\mathbf{b}$$

I Simplex-metoden der præsenteres i det næste kapitel er en vigtig egenskab, at vi systematisk kan gennemgå basal mulige løsninger til et givet lineært programmeringsproblem ved at skifte basale variable ud med ikke-basale variable én ad gangen, så længe det optimerer værdien af objektfunktionen.

4.4 Degenereret og ikke-degenereret basale løsninger

Dette afsnit har til formål at præcisere de to overordnede typer af basale løsninger - *degenereret* basale løsninger og *ikke-degenereret* basale løsninger - da det kommer til at spille en stor rolle i forbindelse med anvendeligheden af den senere Simplex-metode til løsning af lineære programmeringsproblemer.

Bemærk da af definition 4.3.6 at der ikke eksisterer en øvre grænse for antallet af betingelser, der er aktive ved en given basal løsning $\mathbf{x}$. Dette leder netop frem til skelnen mellem degenereret basale løsninger og ikke-degenereret basale løsninger givet ved følgende definition.

Definition 4.4.1. ***Degenereret basale løsninger og ikke-degenereret basale løsninger***
Givet en basal løsning $\mathbf{x} \in \mathbb{R}^n$ til et lineært programmeringsproblem er $\mathbf{x}$ en degenereret basal løsning, hvis der eksisterer mere end n aktive betingelser for $\mathbf{x}$. Tilsvarende er $\mathbf{x}$ en ikke-degenereret basal løsning, hvis der eksisterer nøjagtigt n aktive betingelser for $\mathbf{x}$.

Vi husker på fra definition 4.3.6, at der skal eksistere mindst n aktive betingelser, som er lineært uafhængige, før $\mathbf{x}$ er en basal løsning, hvilket derfor også skal være opfyldt i forhold til definition 4.4.1. Bemærk desuden at for et polyeder $P = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ på standard form, hvor $\mathbf{b} \in \mathbb{R}^m$ og A er en $m \times n$ matrix, medfører det, at $\mathbf{x}$ er en degenereret basal løsning, hvis mere end de $n-m$ ikke-basale variable er 0 - og at der derfor eksisterer mindst én basal variabel $x_i = 0$ for $i \in \{B(1), B(2), \ldots, B(m)\}$. Tilsvarende vil $\mathbf{x}$ være en ikke-degenereret basal løsning, hvis det kun er de $n-m$ ikke-basale variable, der er lig med 0, da der så i alt vil være n aktive betingelser givet ved de m lighedsbetingelser og de $n-m$ ikke-negative begrænsninger.

Eksempel 4.4.2. Vi lader et polyeder være givet ved betingelserne

$$\begin{aligned} x_1 + 2x_2 + 4x_3 &= 10 \\ 2x_2 + x_3 &= 4 \\ x_3 &\leq 2 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \\ x_3 &\geq 0 \end{aligned}$$

og vi bemærker da først, at $\mathbf{x} = \begin{bmatrix} 0 & 1 & 2 \end{bmatrix}^\intercal$ er en degenereret basal mulig løsning, eftersom der findes fire aktive betingelser i form af de to lighedsbetingelser samt $x_3 \leq 2$ og $x_1 \geq 0$. Bemærk da at tre af disse betingelser skal være lineært uafhængige, før vi har en basal mulig løsning jævnfør 4.3.6, hvilket også er tilfældet for de fire aktive betingelser her. Tilsvarende er $\mathbf{x} = \begin{bmatrix} 6 & 2 & 0 \end{bmatrix}^\intercal$ en ikke-degenereret basal mulig løsninger, da der findes tre aktive betingelser i form af de to lighedsbetingelser og $x_3 \geq 0$. Igen gælder der her, at de tre aktive betingelser er lineært uafhængige.

Degenereret og ikke-degenereret basal mulige løsninger har betydning for, hvordan en optimal basal mulig løsning senere vil blive fundet gennem Simplex-metoden. Dette uddybes nærmere i det næste kapitel vedrørende Simplex-metoden.

5 | Simplex-metoden

Dette kapitel har til formål at beskrive Simplex-metoden til at løse lineære programmeringsproblemer, og implementationen som vi kigger på her har til formål at løse minimeringsproblemer. Grundprincippet består da i at undersøge basal mulige løsninger på en systematisk måde, så værdien af objektfunktionen bliver formindsket for hver gang, vi flytter os til en ny basal mulig løsning. Dette vil senere føre til konstruktionen af en algoritme for implementationen af Simplex-metoden. Beskrivelserne i dette afsnit bygger desuden videre på den geometriske og algebraiske forståelse af lineære programmeringsproblemer som givet i kapitel 4. Bemærk at vi i dette kapitel noterer overskuds- og underskudsvariable som x_i fremfor s_i som præsenteret i afsnit 3.2.3 for at simplificere mængdenotationen. Yderligere, vil fokus i dette kapitel være på ikke-degeneret løsninger, og degeneret løsninger vil blive behandlet i et afsnit for sig. Dette kapitel tager udgangspunkt i [6, p. 83-108], medmindre andet fremgår af teksten.

5.1 Introducerende eksempel til Simplex-metoden

For at give et indblik i hvordan Simplex-metoden fungerer, starter vi ud med at gennemgå et eksempel, hvor principperne i Simplex-metoden anvendes til at løse et givet lineært minmeringsproblem. Det ønskes ud fra eksemplet, at læseren får en intuitiv forståelse for, hvad det er, der foregår for hvert trin i en iteration af Simplex-metoden.

Vi får givet et problem ved longhand-notationen

$$
\begin{aligned}
&\textit{Minimer} && f(x_1, x_2, x_3) = x_1 - 3x_2 + 2x_3 \\
&\textit{I forhold til} && 3x_1 - x_2 + 2x_3 \leq 9 \\
& && -2x_1 + 4x_2 + x_3 \leq 14 \\
& && -4x_1 + 4x_2 + 8x_3 \leq 10 \\
& && x_1 \geq 0 \\
& && x_2 \geq 0 \\
& && x_3 \geq 0
\end{aligned}
$$

Vi ønsker nu at finde en optimal løsning ved brug af principperne bag Simplex-metoden. Den første opgave består i, at problemet omskrives til standard form. Vi adderer derfor først underskudsvariable på hver af ulighederne for at opnå lighedsbetingelser samt tilføjer ikke-negative begrænsninger for disse underskudsvariable.

$$
\begin{aligned}
&\textit{Minimer} && f(x_1, x_2, x_3, x_4, x_5, x_6) = x_1 - 3x_2 + 2x_3 + 0x_4 + 0x_5 + 0x_6 \\
&\textit{I forhold til} && 3x_1 - x_2 + 2x_3 + x_4 = 9 \\
& && -2x_1 + 4x_2 + x_3 + x_5 = 14 \\
& && -4x_1 + 4x_2 + 8x_3 + x_6 = 10 \\
&\textit{Og} && x_j \geq 0, \quad j \in \{1, 2, 3, 4, 5, 6\}
\end{aligned}
$$

I det første skridt lader vi nu de tre underskudsvariable være de basale variable, hvor vi bemærker, at de tilhørende søjler er lineært uafhængige, så vi har en begyndende basis-matrix B. Basis-matricen B bliver da I_3, mens de ikke-basale variable x_1, x_2, x_3 sættes til 0, og vi ønsker så at løse ligningen $B\mathbf{x_B} = \mathbf{b}$ her givet ved

$$I_3 \begin{bmatrix} x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 9 \\ 14 \\ 10 \end{bmatrix}$$

Ved at løse denne ligning får vi da, at $\mathbf{x_B}$ er givet ved

$$\mathbf{x_B} = B^{-1}\mathbf{b} = I_3 \begin{bmatrix} 9 \\ 14 \\ 10 \end{bmatrix} = \begin{bmatrix} 9 \\ 14 \\ 10 \end{bmatrix},$$

så den første basal mulige løsning er givet ved $\mathbf{x} = \begin{bmatrix} 0 & 0 & 0 & 9 & 14 & 10 \end{bmatrix}^\intercal$. Ud fra dette opstilles nu den tabel, som kommer til at udgøre starten på første iteration af Simplex-metoden og som er givet ved

		x_1	x_2	x_3	x_4	x_5	x_6
	0	1	−3	2	0	0	0
$x_4 =$	9	3	−1	2	1	0	0
$x_5 =$	14	−2	4	1	0	1	0
$x_6 =$	10	−4	4	8	0	0	1

Tabel 5.1

Her skal det bemærkes, at den røde cirkel i tabel 5.1 viser den negative værdi af objektfunktionen i den nuværende basal mulige løsning, og vi benævner her den tilhørende række for *den nulte række*. Idéen med Simplex-metoden er så, at vi ud fra denne begyndende basal mulige løsning ønsker at undersøge, om der eksisterer en anden basal mulig løsning, der formindsker værdien af objektfunktionen mere. Dette gør vi ved at skifte én basal variabel ud med en ikke-basal variabel og dermed finde en ny basis-matrix. I dette tilfælde vil vi have x_2 ind i basen, da den negative indgang tilhørende x_2 i den nulte række i tabel 5.1 sikrer os, at vi formindsker værdien af objektfunktionen ved at lave dette basis-skifte. I den tilhørende indgangssøjle til x_2 kigger vi kun på de positive indgange, så derfor kan vi se bort fra værdien -1 i den første række efter den nulte række. For de resterende positive indgange i indgangssøjlen danner vi så brøkerne 14/4 og 10/4 og vælger den mindste af de to brøker. I tilfældet her er brøken 10/4 mindst, og den tilhørende indgang med værdien 4 i den blå cirkel i tabel 5.1 er så den indgang, der arbejdes videre med. Vi ønsker da at udføre elementære rækkeoperationer, så denne indgang bliver lavet om til 1, mens de resterende indgange i indgangssøjlen bliver 0. I eksemplet her kommer det til at betyde, at rækkeoperationerne der udføres er givet ved

$$\begin{aligned} \frac{3}{4}r_3 + r_0 &\to r_0 \\ \frac{1}{4}r_3 + r_1 &\to r_1 \\ -r_3 + r_2 &\to r_2 \\ \frac{1}{4}r_3 &\to r_3 \end{aligned} \tag{5.1}$$

Rækkeoperationerne medfører så, at vi har fået lavet en ny tabel givet ved tabel 5.2 herunder.

		x_1	x_2	x_3	x_4	x_5	x_6
	$15/2$	-2	0	8	0	0	$3/4$
$x_4 =$	$23/2$	2	0	4	1	0	$1/4$
$x_5 =$	4	2	0	-7	0	1	-1
$x_2 =$	$5/2$	-1	1	2	0	0	$1/4$

Tabel 5.2

Her bemærkes det, at den røde cirkel i tabel 5.2 viser indgangsvariablen for den første iteration. Vi har nu en ny basis-matrix, der indeholder de oprindelige søjler tilhørende x_4, x_5 og x_2, og hvor den nye basal mulige løsning er givet ved $\mathbf{x} = \begin{bmatrix} 0 & 5/2 & 0 & 23/2 & 4 & 0 \end{bmatrix}^\mathsf{T}$, som aflæses af de sidste tre indgange i den første søjle i tabellen. Af tabellen kan vi desuden aflæse værdien af objektfunktionen i denne nye basal mulige løsning som den negative værdi af den første indgang i den nulte række, så $f(\mathbf{x}) = -15/2$, hvilket bekræfter, at vi har bevæget os til en ny basal mulig løsning, der har formindsket værdien af objektfunktionen fra 0 til $-15/2$. Vi er så klar til en ny iteration af Simplex-metoden, så det igen ønskes undersøgt, om der findes en anden basal mulig løsning, der formindsker værdien af objektfunktionen yderligere. Af den nulte række i tabel 5.2 undersøger vi derfor igen, om der findes en negativ indgang, hvilket der gør for den ikke-basale variabel x_1, og vi ønsker derfor at bringe den tilhørende søjle til x_1 ind i basen. Dette gør vi ved at udføre elementære rækkeoperationer på indgangen i den blå cirkel i tabel 5.2 ud fra samme fremgangsmåde som før.

$$\begin{aligned} r_2 + r_0 &\to r_0 \\ -r_2 + r_1 &\to r_1 \\ \frac{1}{2}r_2 + r_3 &\to r_3 \\ \frac{1}{2}r_2 &\to r_2 \end{aligned} \tag{5.2}$$

Uden at gå i dybden med detaljerne for den nye basal mulige løsning opstiller vi så den nye tabel efter denne anden iteration givet ved tabel 5.3 herunder.

		x_1	x_2	x_3	x_4	x_5	x_6
	$23/2$	0	0	1	0	1	$-1/4$
$x_4 =$	$15/2$	0	0	11	1	-1	$5/4$
$x_1 =$	2	1	0	$-7/2$	0	$1/2$	$-1/2$
$x_2 =$	$9/2$	0	1	$-3/2$	0	$1/2$	$-1/4$

Tabel 5.3

Da der stadig eksisterer en negativ indgang i den nulte række i tabel 5.3, udfører vi igen passende elementære rækkeoperationer, som fører indgangssøjlen for x_6 ind i basen og

fjerner udgangssøjlen tilhørende x_4, hvilket gøres ud fra rækkeoperationerne i ligning 5.3.

$$\begin{aligned} \frac{1}{5}r_1 + r_0 &\to r_0 \\ \frac{4}{10}r_1 + r_2 &\to r_2 \\ \frac{1}{5}r_1 + r_3 &\to r_3 \\ \frac{4}{5}r_1 &\to r_1 \end{aligned} \tag{5.3}$$

For den fjerde og sidste iteration får da vi en opdateret tabel givet ved tabel 5.4 herunder.

		x_1	x_2	x_3	x_4	x_5	x_6
	13	0	0	16/5	1/5	4/5	0
$x_6 =$	6	0	0	44/5	4/5	−4/5	1
$x_1 =$	5	1	0	9/10	2/5	1/10	0
$x_2 =$	6	0	1	7/10	1/5	3/10	0

Tabel 5.4

Her ses det, at der ikke findes negative indgange i den nulte række, hvilket betyder, at vi for den nuværende basal mulige løsning ikke kan bevæge os i nogen retning, der formindsker værdien af objektfunktionen yderligere. Det konkluderes, at den optimale løsning til problemet er givet ved $\mathbf{x} = \begin{bmatrix} 5 & 6 & 0 & 0 & 0 & 6 \end{bmatrix}^\intercal$, hvor den optimale værdi af objektfunktionen er $f(\mathbf{x}) = 1 \cdot 5 - 3 \cdot 6 + 2 \cdot 0 = -13$, hvilket i forhold til tabel 5.4 igen svarer til den negative værdi af den første indgang i den nulte række. I de følgende afsnit vil baggrunden for operationerne udført i dette eksempel blive beskrevet nærmere.

5.2 Introducerende beskrivelser og notationer

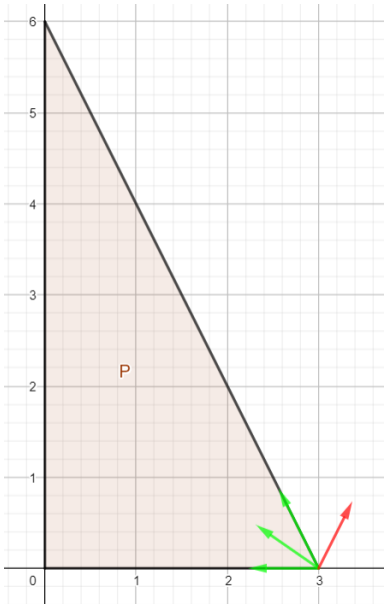
Før selve Simplex-metoden kan beskrives trin for trin, er det nødvendigt først at få nogle notationer på plads, som er gennemgående for funktionaliteten af denne metode. Vi husker på, at vi kun studerer minimeringsproblemer på standard form i det indeværende kapitel, da det er ud fra denne antagelse, at Simplex-metoden vil blive beskrevet og bygget op. Fra afsnit 3.2.3 fremgår det da, at de lineære programmeringsproblemer vi vil løse er på formen

$$\begin{aligned} &\textit{Minimer} && f(\mathbf{x}) = \mathbf{c}^\intercal \mathbf{x} \\ &\textit{I forhold til} && A\mathbf{x} = \mathbf{b} \\ &\textit{Og} && \mathbf{x} \geq \mathbf{0} \end{aligned}$$

Ud fra eksemplet i afsnit 6.1 ønsker vi nu at præcisere, hvad baggrunden er for de forskellige operationer. Og målet med Simplex-metoden er da generelt, at vi ud fra en nuværende basal mulig løsning med en tilhørende basis-matrix B ønsker for hver iteration at finde de basal mulige løsninger, hvor netop én søjle er blevet skiftet ud i basis-matricen sådan, at værdien af objektfunktionen formindskes ved dette basis-skifte. For at kunne gøre dette har vi da i første omgang brug for at definere en retning, som vi kan bevæge os i, væk fra en nuværende basal mulig løsning.

Definition 5.2.1. ***Mulig retningsvektor***
Givet en vektor $\mathbf{x}$ i polyederet $P = \{\mathbf{x} \in \mathbb{R}^n \mid A\mathbf{x} = \mathbf{b}, \mathbf{x} \geq \mathbf{0}\}$ siges $\mathbf{d} \in \mathbb{R}^n$ at være en *mulig retningsvektor*, hvis der findes et tal $\theta > 0$ sådan, at $\mathbf{x} + \theta\mathbf{d} \in P$. Konveksiteten af et polyeder medfører da, at $\mathbf{x} + t\mathbf{d} \in P$ for $0 \leq t \leq \theta$.

Bemærk af definition 5.2.1 at det *ikke* som udgangspunkt er størrelsen af retningsvektoren $\mathbf{d}$, der siger noget om, hvor langt vi kan bevæge os, før vi forlader polyederet - det er i stedet skalaren θ. Vi laver her en illustration, der viser, hvordan retningsvektoren $\mathbf{d}$ kan se ud for et givet polyeder.



Figur 5.1: *Et polyeder P hvor de grønne vektorer alle er mulige retningsvektorer og den røde vektor ikke er en mulig retningsvektor.*

Vi kigger så på en basal mulig løsning $\mathbf{x} = \begin{bmatrix} x_{B(1)} & x_{B(2)} & \dots & x_{B(m)} & 0 & \dots & 0 \end{bmatrix}^\mathsf{T}$ til et givet polyeder. Vi ønsker da at finde en retningsvektor, sådan at vi kan flytte os fra $\mathbf{x}$ til en ny basal mulig løsning $\overline{\mathbf{x}}$, hvor værdien af objektfunktionen i det givne lineære programmeringsproblem formindskes. Det betyder, at vi gerne vil øge en ikke-basal variabel x_j for $j \notin \{B(1), B(2), \dots, B(m)\}$ til så stor en positiv værdi θ som muligt, uden at den nye basale løsning bevæger sig udenfor polyederet. For at nå til denne nye basal mulige løsning konstruerer vi $\mathbf{d}$ sådan, at $d_j = 1$ og $d_i = 0$ for $i \notin \{j, B(1), B(2), \dots, B(m)\}$. Vi mangler da blot at undersøge, hvad der gælder for $\mathbf{d_B} = \begin{bmatrix} d_{B(1)} & d_{B(2)} & \dots & d_{B(m)} \end{bmatrix}^\mathsf{T}$, som tilsvarer den del af retningsvektoren, der siger noget om ændringen af de nuværende basale variable. Den nye basal mulige løsning er da givet ved $\overline{\mathbf{x}} = \mathbf{x} + \theta\mathbf{d}$, og på grund af linearitetsbetingelserne for objektfunktionen har vi desuden, at $f(\overline{\mathbf{x}}) = f(\mathbf{x} + \theta\mathbf{d}) = f(\mathbf{x}) + f(\theta\mathbf{d})$.

Selvom vi bevæger os til en ny basal løsning, vil vi stadig gerne sikre, at denne nye basale løsning er mulig, hvorfor den skal opfylde, at $A\overline{\mathbf{x}} = A(\mathbf{x} + \theta\mathbf{d}) = \mathbf{b}$ sådan, at

$$A\mathbf{x} + \theta A\mathbf{d} = \mathbf{b} \iff \theta A\mathbf{d} = \mathbf{b} - A\mathbf{x} \iff \theta A\mathbf{d} = \mathbf{0} \iff A\mathbf{d} = \mathbf{0}, \tag{5.4}$$

Bemærk at da $d_j = 1$ og $d_i = 0$ for $i \notin \{j, B(1), B(2), \ldots, B(m)\}$ må der så gælde, at

$$\mathbf{0} = A\mathbf{d} = \sum_{i=1}^{n} \mathbf{A}_i d_i = \mathbf{A_j} + \sum_{i=1}^{m} \mathbf{A}_{B(i)} d_{B(i)} = \mathbf{A_j} + B\mathbf{d_B}$$

Og givet at basis-matricen B er invertibel jævnfør sætning 2.6.7 gælder så, at

$$\mathbf{d_B} = -B^{-1}\mathbf{A_j}$$

Vi er nu klar til at konkretisere retningsvektoren $\mathbf{d}$.

5.2.2. Vektoren $\mathbf{d}$ hvor

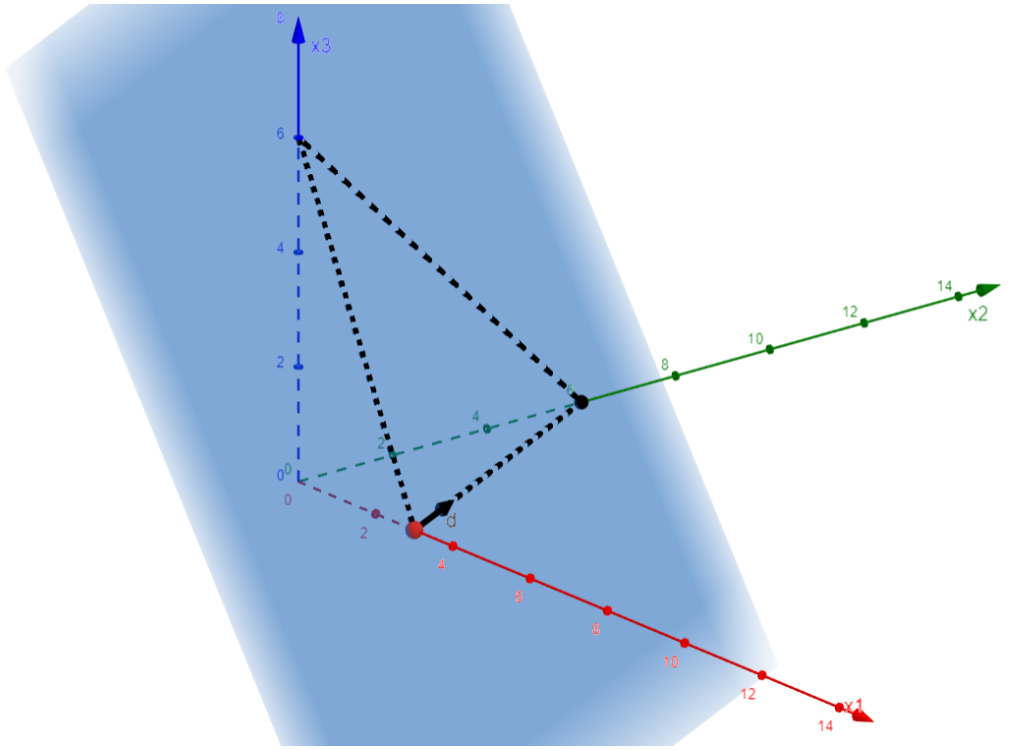
$$\mathbf{d_B} = -B^{-1}\mathbf{A_j} \tag{5.5}$$

samt $d_j = 1$ og $d_i = 0$ for $i \notin \{j, B(1), B(2), \ldots, B(m)\}$ kaldes for *den j'te mulige basisretning.*

Eksempel 5.2.3. Vi lader et polyeder være givet ved betingelserne og begrænsningerne

$$2x_1 + x_2 + x_3 = 6$$
$$x_i \geq 0, \quad i \in \{1, 2, 3\},$$

og vi kigger så på en grafisk illustration af dette polyeder i figur 5.2 herunder.



Figur 5.2: *En grafisk afbildning af polyederet, hvor vi ønsker at bevæge os væk fra den basal mulige løsning i det røde punkt i retning af vektoren* $\mathbf{d}$ *mod en ny basal mulig løsning i det sorte punkt.*

Vi vil så gerne bevæge os væk fra den basal mulige løsning $\mathbf{x} = \begin{bmatrix} 3 & 0 & 0 \end{bmatrix}^\intercal$ med den basale variabel x_1 og de ikke-basale variable x_2 og x_3 ved at følge retningsvektoren $\mathbf{d}$ på figur 5.2. Retningsvektoren $\mathbf{d}$ konstrueres derfor ud fra, at vi gerne vil øge x_2 og beholde x_3 som en ikke-basal variabel, hvorfor $\mathbf{d} = \begin{bmatrix} d_{B(1)} & 1 & 0 \end{bmatrix}^\intercal$. For at finde ud af hvad $d_{B(1)}$ skal være anvender vi så ligning 5.5, hvor basis-matricen her er givet ved søjlen tilhørende den basale variabel x_1, så der gælder, at $B = \begin{bmatrix} 2 \end{bmatrix}$, og vi får så, at

$$d_{B(1)} = -B^{-1}A_2 = -\begin{bmatrix} \frac{1}{2} \end{bmatrix}\begin{bmatrix} 1 \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} \end{bmatrix}$$

Og retningsvektoren i dette eksempel er derfor givet ved $\mathbf{d} = \begin{bmatrix} -1/2 & 1 & 0 \end{bmatrix}^\intercal$.

Spørgsmålet er så, hvordan værdien af objektfunktionen ændrer sig, når vi bevæger os i retningen af $\mathbf{d}$. Bemærk da at denne ændring kan beskrives som $\mathbf{c}^\intercal\mathbf{d}$, eftersom det giver et udtryk for, hvordan kostkoefficient-vektoren $\mathbf{c}$ ændrer sig i retning af $\mathbf{d}$. Bemærk da at $d_j = 1$ og $d_i = 0$ for $i \notin \{j, B(1), B(2), \ldots, B(m)\}$, hvorfor vi kan skrive $\mathbf{c}^\intercal\mathbf{d}$ som

$$\mathbf{c_B}^\intercal\mathbf{d_B} + c_j$$

Vi husker da på, at $\mathbf{d_B} = -B^{-1}\mathbf{A_j}$ jævnfør ligning 5.5, hvorfor vi nu kan definere størrelsen $\mathbf{c}^\intercal\mathbf{d}$ som følger.

5.2.4. Tallet

$$\hat{c_j} = c_j - \mathbf{c_B}^\intercal B^{-1}\mathbf{A_j} \tag{5.6}$$

kaldes *den reducerede kostkoefficient* for variablen x_j.

Her svarer $\mathbf{c_B}$ til kostkoefficienterne for de basale variable i objektfunktionen. Bemærk desuden at det kun giver mening at snakke om $\hat{c_j}$ for ikke-basale variable, hvilket der vil komme et argument for her. Vi husker da på, at $B = \begin{bmatrix} \mathbf{A_{B(1)}} & \mathbf{A_{B(2)}} & \cdots & \mathbf{A_{B(m)}} \end{bmatrix}$, hvorfor det naturligt følger af definition 2.6.4, at $B^{-1}B = I$. Heraf følger det så også, at $B^{-1}\mathbf{A_{B(i)}}$ for $i \in \{1, 2, \ldots, m\}$ alle er standard-vektorer, hvilket medfører, at

$$\begin{aligned} \hat{c}_{B(i)} &= c_{B(i)} - \mathbf{c_B}^\intercal B^{-1}\mathbf{A_{B(i)}} \\ &= c_{B(i)} - \mathbf{c_B}^\intercal\mathbf{e_{B(i)}} \\ &= c_{B(i)} - c_{B(i)} = 0 \end{aligned} \tag{5.7}$$

Med andre ord er $\hat{c_j}$ konstrueret ud fra, at ønskes den reducerede kostkoefficient for en basal-variabel, får vi altid 0, hvorfor dette er et argument for, at vi kun anskuer $\hat{c_j}$ for ikke-basale variable.

Ud fra konstruktionen af retningsvektoren $\mathbf{d}$ og de reducerede kostkoefficienter $\hat{c_j}$ har vi nu brug for at kunne udtrykke, hvornår vi har opnået en optimal basal mulig løsning og derfor ikke behøver at fortsætte med at kigge efter en bedre basal mulig løsning. Hertil gælder følgende sætning.

Sætning 5.2.5. Lad $\mathbf{x}$ være en basal mulig løsning med en tilhørende basis-matrix B. Lad tilsvarende $\hat{\mathbf{c}}$ være vektoren bestående af de reducerede kostkoefficienter for de givne variable i $\mathbf{x}$. Da gælder der så, at

1. Hvis $\hat{\mathbf{c}} \geq \mathbf{0}$, så er $\mathbf{x}$ en optimal basal mulig løsning.
2. Hvis $\mathbf{x}$ er en ikke-degenereret optimal basal mulig løsning, så er $\hat{\mathbf{c}} \geq \mathbf{0}$.

Bevis. For at bevise (1) antager vi, at $\hat{\mathbf{c}} \geq \mathbf{0}$ samt at $\mathbf{u}$ er en vilkårlig mulig løsning i standard polyederet P sådan, at retningsvektoren $\mathbf{d}$ er givet ved $\mathbf{d} = \mathbf{u} - \mathbf{x}$. Vi skal så vise, at $\mathbf{x}$ er en optimal basal mulig løsning. Bemærk først at da både $\mathbf{u}$ og $\mathbf{x}$ er mulige løsninger skal der gælde, at $A\mathbf{x} = A\mathbf{u} = \mathbf{b}$, hvorfor vi får, at

$$\mathbf{0} = \mathbf{b} - \mathbf{b} = A\mathbf{u} - A\mathbf{x} = A(\mathbf{u} - \mathbf{x}) = A\mathbf{d} \tag{5.8}$$

Ud fra konstruktionen af $\mathbf{d}$ betyder det så, at vi kan omskrive ligning 5.8 til

$$\mathbf{0} = B\mathbf{d_B} + \sum_{i \in I} \mathbf{A_i} d_i$$

hvor I er indeksmængden for de ikke-basale variable. Vi husker da på, at basis-matricen B er invertibel jævnfør sætning 2.6.7, og derfor får vi så, at

$$\mathbf{d_B} = -B^{-1} \sum_{i \in I} \mathbf{A_i} d_i = -\sum_{i \in I} B^{-1} \mathbf{A_i} d_i$$

De reducerede kostkoefficienter er som beskrevet tidligere konstrueret ud fra $\mathbf{c}^\intercal\mathbf{d}$, hvor vi af ovenstående da får, at

$$\begin{aligned} \mathbf{c}^\intercal\mathbf{d} = \mathbf{c_B}^\intercal\mathbf{d_B} + \sum_{i \in I} c_i d_i &= -\mathbf{c_B}^\intercal \sum_{i \in I} B^{-1}\mathbf{A_i} d_i + \sum_{i \in I} c_i d_i \\ &= \sum_{i \in I} (c_i - \mathbf{c_B}^\intercal B^{-1} \mathbf{A_i}) d_i = \sum_{i \in I} \hat{c}_i d_i \end{aligned} \tag{5.9}$$

Da $\mathbf{x}$ er en basal mulig løsning ud fra antagelsen i sætning 5.2.5 skal der gælde, at $x_i = 0$ for $i \in I$, og da vi samtidig har, at $\mathbf{u}$ er en mulig løsning i P, skal der gælde, at $u_i \geq 0$ for $i \in I$. Ud fra konstruktionen af $\mathbf{d}$ får vi så, at $d_i \geq 0$, og da der ud fra antagelsen gælder, at $\hat{\mathbf{c}} \geq \mathbf{0}$ medfører det, at $\hat{c}_i d_i \geq 0$ for $i \in I$. Af ligning 5.9 følger det så, at $\mathbf{c}^\intercal\mathbf{d} \geq 0$, hvorfor det betyder, at ligegyldigt hvilken retningsvektor $\mathbf{d}$ vi konstruerer ud fra $\mathbf{x}$ til en vilkårlig mulig løsning $\mathbf{u}$, så formindskes værdien af objektfunktionen ikke i den pågældende retning, og (1) er dermed bevist.

Vi beviser nu (2) ved modstrid, så vi antager, at $\mathbf{x}$ er en ikke-degenereret optimal basal mulig løsning, samt at der eksisterer $\hat{c}_i < 0$, som så skal føre til en modstrid. Vi husker da først ud fra bemærkningen til konstruktionen af $\hat{c}_i$, at $\hat{c}_i = 0$ for $i \notin I$ hvor I igen er indeksmængden for de ikke-basale variable, så de reducerede kostkoefficienter er 0 for alle basale variable, og x_i må derfor være en ikke-basal variabel med en tilhørende mulig basisretning $\mathbf{d}$, hvis $\hat{c}_i < 0$. Da $\mathbf{x}$ er en ikke-degenereret optimal basal mulig løsning, vil $\mathbf{d}$ være en mulig retningsvektor, eftersom vi ud fra definition 5.2.1 kan vælge $\theta > 0$ tilpas lille, så en ny løsning $\mathbf{x} + \theta\mathbf{d}$ stadig ligger i P. Da $\hat{c}_i < 0$ og vi kan bevæge os i retningen af $\mathbf{d}$, vil objektfunktionen blive formindsket i denne retning, og vi får så, at $\mathbf{x}$ ikke kan være en optimal basal mulig løsning, hvilket leder frem til en modstrid, og derfor er $\hat{\mathbf{c}} \geq \mathbf{0}$. ■

Bemærk af sætning 5.2.5 at optimalitetsbetingelserne *kun* er alsidigt gældende for ikke-degenereret basal mulige løsninger, hvorfor der i kommende afsnit vil blive kommenteret på den mulige situation, at $\mathbf{x}$ er en degenereret basal mulig løsning. Bemærk desuden at optimalitetsbetingelserne lægger op til, at vi blot skal undersøge, om $\hat{c}_i \geq 0$ for $i \in I$ hvor I er indeksmængden for de ikke-basale variable, da det vil betyde, at den nuværende basal mulige løsning er optimal.

5.3 Opbygning af Simplex-metoden

Vi anvender nu de værktøjer og notationer der er blevet beskrevet i det foregående afsnit til at konstruere principperne i Simplex-metoden. Idéen er, at vi gerne vil have en systematisk måde at undersøge basal mulige løsninger på ved at flytte fra basal mulig løsning til basal mulig løsning i vores algoritme, sådan at værdien af objektfunktionen formindskes for hver iteration, indtil den optimale løsning er fundet. Bemærk at den optimale værdi af objektfunktionen godt kan være $-\infty$, som det vil blive beskrevet i det indeværende afsnit.

Lad nu $\mathbf{x}$ være en basal mulig løsning og $\hat{\mathbf{c}}$ være vektoren bestående af de reducerede kostkoefficienter. Hvis $\hat{c}_j \geq 0$ for alle $j \notin \{B(1), B(2), \ldots, B(m)\}$, så vil $\mathbf{x}$ være en optimal basal mulig løsning jævnfør sætning 5.2.5 (1), og derfor skal algoritmen returnere $\mathbf{x}$ som den optimale basal mulige løsning. Hvis der derimod eksisterer mindst ét $\hat{c}_j < 0$ for alle $j \notin \{B(1), B(2), \ldots, B(m)\}$, så findes der en anden basal mulig løsning $\overline{\mathbf{x}}$ som formindsker værdien af objektfunktionen jævnfør sætning 5.2.5 (2). I dette tilfælde skal algoritmen så vælge en af de j'te mulige basisretninger $\mathbf{d}$, hvor $\hat{c}_j < 0$, sådan at vi bevæger os mod en ny basal mulig løsning $\overline{\mathbf{x}}$ som formindsker værdien af objektfunktionen. Vi husker på, at x_j for $j \notin \{B(1), B(2), \ldots, B(m)\}$ forøges i retningen af $\mathbf{d}$, da $d_j = 1$, mens x_i for $i \notin \{j, B(1), B(2), \ldots, B(m)\}$ forbliver 0. Når vi har fulgt den j'te mulige basisretning så meget som muligt siger vi så, at x_j eller $\mathbf{A_j}$ bliver *bragt ind i basen.*

Spørgsmålet er så, hvor langt vi kan bevæge os i retningen af $\mathbf{d}$, så den nye basal mulige løsning stadig er indeholdt i P. Vi husker da på, at $\overline{\mathbf{x}} = \mathbf{x} + \theta\mathbf{d}$ er den nye basal mulige løsning, hvor $\theta > 0$, og da vi ved, at objektfunktionens værdi formindskes i retningen af $\mathbf{d}$, ønsker vi derfor at finde det størst mulige θ sådan, at

$$\theta^* = \max\{\theta > 0 \mid \overline{\mathbf{x}} \in P\}.$$

Vi kan nu undersøge nærmere, hvad θ^* skal være. Givet at $A\mathbf{d} = \mathbf{0}$ jævnfør ligning 5.4 ved vi, at $A(\mathbf{x} + \theta\mathbf{d}) = \mathbf{b}$ for alle $\theta > 0$, og derfor er $\overline{\mathbf{x}} = \mathbf{x} + \theta\mathbf{d}$ en basal mulig løsning, medmindre en eller flere indgange i $\overline{\mathbf{x}}$ er negative. Lad os nu kigge på hvilken rolle $\mathbf{d}$ har for de to tilfælde, hvor $\mathbf{d} \geq \mathbf{0}$ eller $d_i < 0$ for mindst ét $i \in \{B(1), B(2), \ldots, B(m)\}$.

- Hvis $\mathbf{d} \geq \mathbf{0}$ kan vi vælge $\theta^* = \infty$, eftersom at $\mathbf{x_B} > \mathbf{0}$ og $x_i = 0$ for $i \notin \{B(1), B(2), \ldots, B(m)\}$, hvorfor $\theta^* > 0$ aldrig forårsager, at indgange i den nye basal mulige løsning $\overline{\mathbf{x}} = \mathbf{x} + \theta\mathbf{d}$ bliver negative.
- Hvis $d_i < 0$ for mindst ét $i \in \{B(1), B(2), \ldots, B(m)\}$, skal det stadig være opfyldt, at den nye basale løsning er en mulig løsning, sådan at

  $$x_i + \theta d_i \geq 0 \iff \theta \leq -\frac{x_i}{d_i}$$

  Bemærk at valget af θ^* skal tilfredsstille $x_i + \theta^* d_i \geq 0$ for alle i hvor $d_i < 0$, så vi vælger θ^* ved

  $$\theta^* = \min_{\{i \mid d_i < 0\}} \left\{ -\frac{x_i}{d_i} \right\}. \tag{5.10}$$

Ud fra konstruktionen af $\mathbf{d}$ er $d_j = 1$ for den ikke-basale variabel x_j der bliver bragt ind i basen og $d_i = 0$ for $i \notin \{j, B(1), B(2), \ldots, B(m)\}$, og det betyder, at vi kun kigger på d_i for basale variable. Det giver en ækvivalent formel for θ^* ud fra ligning 5.10 givet ved

$$\theta^* = \min_{\{i \in \{B(1),B(2),\ldots,B(m)\} | d_i < 0\}} \left\{ -\frac{x_i}{d_i} \right\}. \tag{5.11}$$

Givet at θ^* er valgt og $x_j = 0$ samt $d_j = 1$ gælder så, at $\overline{x}_j = \theta^*$. Lad tilsvarende ℓ være det indeks, der giver θ^* fra ligning 5.11 sådan, at

$$\theta^* = -\frac{x_{B(\ell)}}{d_{B(\ell)}}$$

Vi ved så, at $d_{B(\ell)} < 0$ ud fra ligning 5.11, hvorfor $\overline{x}_{B(\ell)}$ i den nye løsning bliver

$$\overline{x}_{B(\ell)} = x_{B(\ell)} + \theta^* d_{B(\ell)} = x_{B(\ell)} - \frac{x_{B(\ell)}}{d_{B(\ell)}} \cdot d_{B(\ell)} = x_{B(\ell)} - x_{B(\ell)} = 0 \tag{5.12}$$

Vi ser da, at $\overline{x}_{B(\ell)} = 0$, når vi har flyttet os fra den nuværende basal mulige løsning $\mathbf{x}$ til den nye basal mulige løsning $\overline{\mathbf{x}}$, mens at x_j er blevet positiv ud fra θ^*, og vi siger derfor, at x_j er blevet bragt ind i basen, mens $x_{B(\ell)}$ er blevet fjernet fra basen. Det betyder tilsvarende, at vi kan opdatere B sådan, at søjlen $\mathbf{A_{B(\ell)}}$ fjernes fra basen, mens $\mathbf{A_j}$ bringes ind i basen, hvorfor den opdaterede basis-matrix $\overline{B}$ bliver

$$\overline{B} = \begin{bmatrix} \mathbf{A_{B(1)}} & \mathbf{A_{B(2)}} & \ldots & \mathbf{A_{B(\ell-1)}} & \mathbf{A_j} & \mathbf{A_{B(\ell+1)}} & \mathbf{A_{B(\ell+2)}} & \ldots & \mathbf{A_{B(m)}} \end{bmatrix}, \tag{5.13}$$

og vi får tilsvarende, at den nye indeksmængde $\overline{B}(i)$ for basal-variablene i den nye basal mulige løsning opdateres ud fra

$$\overline{B}(i) = \begin{cases} j, & i = \ell \\ B(i), & i \neq \ell \end{cases} \tag{5.14}$$

Vi mangler så blot at vise, at $\overline{B}$ er en basis-matrix, og for at gøre dette har vi i første omgang brug for at vise, at givet en mængde af lineært uafhængige vektorer, vil mængden af de samme vektorer med basis-matricen multipliceret på også være lineært uafhængige.

Lemma 5.3.1.
Hvis B er en $n \times n$ invertibel matrix, og $V = \{\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_k}\}$ er en mængde af vektorer i $\mathbb{R}^n$, så er V en lineær uafhængig mængde, hvis og kun hvis mængden $S = \{B\mathbf{v_1}, B\mathbf{v_2}, \ldots, B\mathbf{v_k}\}$ er lineær uafhængig.

Bevis. Vi viser først den ene vej ved kontraposition, så vi antager, at mængden S er lineær afhængig, og vi da skal komme frem til, at mængden V også er lineær afhængig. Ved antagelsen af lineær afhængighed for mængden S gælder der jævnfør definition 2.3.1, at $\exists l_i \neq 0$ sådan, at

$$B \sum_{i=1}^{k} \mathbf{v_i} l_i = \mathbf{0} \tag{5.15}$$

Fordi B er en invertibel matrix, kan vi så multiplicere B^{-1} på begge sider af ligning 5.15, hvilket medfører, at

$$B^{-1} B \sum_{i=1}^{k} \mathbf{v_i} l_i = B^{-1}\mathbf{0} \iff \sum_{i=1}^{k} \mathbf{v_i} l_i = \mathbf{0}$$

hvor vi ved, at $\exists l_i \neq 0$, så mængden V er tilsvarende lineær afhængig. Vi har da bevist, at hvis V er en lineær uafhængig mængde af vektorer, så er S også en lineær uafhængig mængde af vektorer.

Vi viser så tilsvarende ved kontraposition, at hvis S er en lineær uafhængig mængde af vektorer, så er V også en lineær uafhængig mængde af vektorer. Vi antager derfor, at mængden V er lineær afhængig og skal så vise, at mængden S også er lineær afhængig. Da gælder der tilsvarende, at der $\exists l_i \neq 0$ sådan, at

$$\sum_{i=1}^{k} \mathbf{v_i} l_i = \mathbf{0} \tag{5.16}$$

Her multiplicerer vi så B på begge sider af lighedstegnet i ligning 5.16 ud fra regnereglerne (1) og (2) i 2.1.9, hvilket medfører, at

$$\begin{aligned} B\sum_{i=1}^{k} \mathbf{v_i} l_i = B\mathbf{0} \iff \\ B(\mathbf{v_1} l_1 + \mathbf{v_2} l_2 + \ldots + \mathbf{v_k} l_k) = \mathbf{0} \iff \\ (B\mathbf{v_1} l_1 + B\mathbf{v_2} l_2 + \ldots + B\mathbf{v_k} l_k) = \mathbf{0} \iff \\ (l_1 B\mathbf{v_1} + l_2 B\mathbf{v_2} + \ldots + l_k B\mathbf{v_k}) = \mathbf{0} \end{aligned}$$

Her ved vi, at $\exists l_i \neq 0$ ud fra antagelsen af lineær afhængighed, hvorfor S også er en lineær afhængig mængde. Vi har da vist, at hvis S er en lineær uafhængig mængde, så er V også en lineær uafhængig mængde. Lemmaet er da bevist. ■

Vi er nu klar til at vise, at $\overline{B}$ jævnfør ligning 5.13 er en basis-matrix, hvilket er gengivet i den følgende sætning.

Sætning 5.3.2.

1. Søjlerne $\mathbf{A_{B(i)}}$ for $i \neq \ell$ og $\mathbf{A_j}$ er lineært uafhængige, og det følger derfor, at $\overline{B}$ er en basis-matrix.
2. For basis-matricen $\overline{B}$ er vektoren $\overline{\mathbf{x}} = \mathbf{x} + \theta^* \mathbf{d}$ en basal mulig løsning, der er knyttet til $\overline{B}$.

Bevis. Vi beviser (1) direkte ved at kigge på sammenhængen mellem $B^{-1}\mathbf{A_{B(i)}}$ for $i \neq \ell$ og $B^{-1}\mathbf{A_j}$, hvor vi af lemma 5.3.1 har vist, at hvis søjlerne $B^{-1}\mathbf{A}_{\overline{\mathbf{B}}\mathbf{(i)}}$ for $i \in \{1, 2, \ldots, m\}$ er lineært uafhængige, så er søjlerne $\mathbf{A}_{\overline{\mathbf{B}}\mathbf{(i)}}$ for $i \in \{1, 2, \ldots, m\}$ også lineært uafhængige. Bemærk da at $B^{-1}B = I$ og $\mathbf{A_{B(i)}}$ er den i'te søjle i basis-matricen B. Det medfører så, at $B^{-1}\mathbf{A_{B(i)}}$ for $i \neq \ell$ giver standard-vektorer $\mathbf{e_i}$, hvorfor disse standard-vektorer per definition har 0 som deres ℓ'te indgang. Vi har da som det første, at $B^{-1}\mathbf{A_{B(i)}}$ for $i \neq \ell$ er lineært uafhængige, da det er forskellige standard-vektorer. Desuden ved vi, at $B^{-1}\mathbf{A_j} = -\mathbf{d_B}$ jævnfør ligning 5.5, og ud fra konstruktionen af $\mathbf{d_B}$ gælder der så, at $d_{B(\ell)} \neq 0$, så den ℓ'te indgang i $B^{-1}\mathbf{A_j}$ er forskellig fra 0. Derfor er $B^{-1}\mathbf{A_j}$ lineært uafhængig af $B^{-1}\mathbf{A_{B(i)}}$ for $i \neq \ell$, så $\overline{B}$ er en ny basis-matrix, og (1) er dermed bevist.

For at bevise (2) husker vi på, at $\overline{\mathbf{x}}$ er en basal mulig løsning, da $\overline{x}_{B(\ell)} = 0$ jævnfør ligning 5.12, mens vi har konstrueret $\mathbf{d}$ og θ^* sådan, at $\overline{\mathbf{x}} \geq \mathbf{0}$. Mere specifikt gælder der derfor, at $\overline{x}_i = 0$ for $i \notin \{\overline{B}(1), \overline{B}(2), \ldots, \overline{B}(m)\}$. Vi ved tilsvarende, at $A\overline{\mathbf{x}} = \mathbf{b}$ er opfyldt jævnfør ligning 5.4, hvorfor $\overline{\mathbf{x}}$ er en basal mulig løsning. Desuden har vi ud fra (1) vist, at $\mathbf{A}_{\overline{\mathbf{B}}(\mathbf{1})}, \mathbf{A}_{\overline{\mathbf{B}}(\mathbf{2})}, \ldots, \mathbf{A}_{\overline{\mathbf{B}}(\mathbf{m})}$ er lineært uafhængige, hvilket medfører, at $\overline{\mathbf{x}}$ er en basal mulig løsning jævnfør sætning 4.3.7 tilhørende basis-matricen $\overline{B}$, og dermed er (2) bevist. ■

Bemærk at vi fremover betegner $-\mathbf{d_B}$ som vektoren $\mathbf{u}$, så

$$\mathbf{u} = -\mathbf{d_B} = B^{-1}\mathbf{A_j}, \tag{5.17}$$

da det viser sig at være nyttigt for konstruktionen af algoritmen for Simplex-metoden, og hvor $\mathbf{A_j}$ her svarer til den søjle-vektor, som bringes ind i basen. Da vi har indført vektoren $\mathbf{u} = -\mathbf{d_B}$, erstattes denne også i udtrykket for θ^*, som nu er på formen

$$\theta^* = \min_{\{i \in \{B(1), B(2), \ldots, B(m)\} | u_i > 0\}} \left\{ \frac{x_i}{u_i} \right\}. \tag{5.18}$$

Alt i alt leder ovenstående beskrivelser frem til, at vi nu kan konstruere pseudo-algoritmen for iterationer i Simplex-metoden.

5.3.3. Pseudokode for Simplex-metoden

1. Simplex-metoden skal først bruge en basis-matrix B, der indeholder basis-søjler $\mathbf{A}_{\mathbf{B}(\mathbf{1})}, \mathbf{A}_{\mathbf{B}(\mathbf{2})}, \ldots, \mathbf{A}_{\mathbf{B}(\mathbf{m})}$ fra A og en tilhørende basal mulig løsning $\mathbf{x}$.
2. De reducerede kostkoefficienter $\hat{c_j}$ beregnes for $j \notin \{B(1), B(2), \ldots, B(m)\}$. Hvis det viser sig, at $c_j \geq 0$ for alle j er den nuværende basal mulige løsning $\mathbf{x}$ optimal, og algoritmen skal stoppe. Ellers vælger vi et j, sådan at $c_j < 0$.
3. Vi beregner så $\mathbf{u} = B^{-1}\mathbf{A_j}$, og hvis $\mathbf{u} \leq \mathbf{0}$ vælges $\theta^* = \infty$, hvorfor det medfører, at den optimale værdi af objektfunktionen bliver $-\infty$, og algoritmen skal derfor stoppe.
4. Hvis der eksisterer $u_i > 0$, vælges θ^* sådan, at
$$\theta^* = \min_{\{i = B(1), B(2), \ldots, B(m) | u_i > 0\}} \left\{ \frac{x_i}{u_i} \right\} \tag{5.19}$$
5. Vi lader så ℓ være indeks sådan, at $\theta^* = x_{B(\ell)}/u_{B(\ell)}$. Ud fra dette konstruerer vi en ny basis-matrix $\overline{B}$, hvor der for den tilhørende nye basal mulige løsning $\overline{\mathbf{x}}$ gælder, at $\overline{x}_j = \theta^*$, mens de resterende basale variable $\overline{x}_{B(i)} = x_{B(i)} - \theta^* u_{B(i)}$ for $i \neq \ell$ og de ikke-basale variable $\overline{x}_i = 0$ for $i \notin \{j, B(1), B(2), \ldots, B(m)\}$.
6. Gå tilbage til trin 2.

Det er væsentligt her at huske på, at et ikke-tomt polyeder på standard form *altid* har mindst én basal mulig løsning jævnfør 4.3.9, hvorfor Simplex-metoden altid vil starte, så længe vi blot allerede har denne basal mulige løsning til rådighed. Vi mangler så et argument for, at den konstruerede pseudo-algoritme for Simplex-metoden stopper indenfor et endeligt antal iterationer, hvilket er gengivet i den følgende sætning.

Sætning 5.3.4. Lad P være et ikke-tomt polyeder på standard form hvor alle basal mulige løsninger er ikke-degenereret. Da vil algoritmen for Simplex-metoden stoppe efter et endeligt antal iterationer, og da vil resultatet af Simplex-metoden være givet ved én af følgende muligheder.

1. Givet en afsluttende vektor $\mathbf{u}$ sådan, at $\mathbf{u} \leq \mathbf{0}$ og $\hat{c}_j < 0$ vil den optimale værdi af objektfunktionen være $-\infty$.
2. Givet en afsluttende basis-matrix $\overline{B}$ med en tilhørende basal mulig løsning $\overline{\mathbf{x}}$, hvor $\hat{\mathbf{c}} \geq \mathbf{0}$, vil $\overline{\mathbf{x}}$ være en optimal basal mulig løsning i P.

Bevis. Vi ser først, at hvis $\hat{c}_j \geq 0$ for alle $j \notin \{B(1), B(2), \ldots, B(m)\}$ ved den nuværende basal mulige løsning $\mathbf{x}$, vil trin to i pseudo-algoritme 5.3.3 stoppe algoritmen, da ingen retning vil formindske objektfunktionens værdi, og $\mathbf{x}$ er derfor en optimal basal mulig løsning.

Hvis det modsat gælder, at vi har fundet en basal mulig løsning $\mathbf{x}$, hvor der eksisterer en ikke-basal variabel x_j sådan, at $\hat{c}_j < 0$ mens $\mathbf{u} \leq \mathbf{0}$, ser vi af trin tre i pseudo-algoritme 5.3.3, at Simplex-metoden stopper og giver den optimale værdi af objektfunktionen som $-\infty$. Dette følger af, at $\overline{\mathbf{x}} = \mathbf{x} - \theta^* \mathbf{u} \in P$ for alle $\theta^* \geq 0$ hvis $\mathbf{u} \leq \mathbf{0}$, så vi kan vælge θ^* vilkårlig stor og dermed gøre værdien af objektfunktionen vilkårligt lille, mens $\overline{\mathbf{x}}$ stadig ligger i P.

Givet at alle basal mulige løsninger er ikke-degenereret, er det ikke muligt at finde $\theta^* = 0$ ud fra konstruktionen af θ^*, og derfor vil vi bevæge os til en ny basal mulig løsning for hver iteration, hvor værdien af objektfunktionen bliver mindre, så $f(\overline{\mathbf{x}}) < f(\mathbf{x})$. Det medfører så, at den samme basal mulige løsning aldrig kan blive besøgt flere gange. Og da P indeholder et endeligt antal basal mulige løsninger som beskrevet i kapitel 4, vil algoritmen derfor stoppe efter et endeligt antal iterationer. ■

5.4 Degenereret løsning og Simplex-metoden

I afsnit 5.3 er algoritmen for en iteration af Simplex-metoden beskrevet i pseudokode 5.3.3. Denne algoritme for Simplex-metoden er dog kun konstrueret for ikke-degenereret basal mulige løsninger, hvor der i tilfælde af degenereret basal mulige løsninger kan opstå nogle alternative situationer. Algoritmen kan dog stadig bruges i tilfælde af degenereret basal mulige løsninger, hvilket vil blive beskrevet i det følgende afsnit.

En situation der kan opstå er, at hvis den nuværende basal mulige løsning $\mathbf{x}$ i trin 1 i pseudokoden 5.3.3 til Simplex-metoden er degenereret, så bliver $\theta^* = 0$ i trin 4, og ud fra konstruktionen af $\overline{\mathbf{x}}$ betyder det så, at denne nye basal mulige løsning er den samme som den forrige. Så selvom der eksisterer $x_{B(i)} = 0$ i den nuværende basal mulige løsning, og indgangen $d_{B(i)}$ i retningsvektoren $\mathbf{d}$ er negativ, vil vi ikke følge denne retningsvektor mod en ny basal løsning, som ikke er mulig, da algoritmen automatisk vælger $\theta^* = 0$. Det ses dog, at sætning 5.3.2 stadig er opfyldt, da iterationen fuldendes ved at definere en ny basis-matrix $\overline{B}$ ved at fjerne $\mathbf{A}_{\mathbf{B}(\ell)}$, som er søjlen for udgangsvariablen, fra basen og bringe $\mathbf{A_j}$, som er søjlen tilhørende den ikke-basale variabel x_j, ind i basen.

En anden situation der kan opstå er, at selvom θ^* er positiv i trin 4, så kan det stadig ske, at mere end én af de oprindelige basale variable bliver 0 i den nye basal mulige løsning $\overline{\mathbf{x}}$. Og det medfører så, at den nye basal mulige løsning bliver degenereret, eftersom kun én af disse basale variable vil forlade basen efter en iteration.

Når man for en degenereret basal mulig løsning skal finde en ny basal mulig løsning kan man ende med, at man efter et vilkårligt antal iterationer vender tilbage til den oprindelige basal mulige løsning med den samme tilhørende basis. Dette resulterer i, at algoritmen går ind i en *løkke*, så algoritmen aldrig stopper, hvilket ikke ønskes for den praktiske implementation af algoritmen. Det følgende afsnit har til formål at beskrive de valg, man kan træffe, som kan have en betydning for, om man eksempelvis undgår, at algoritmen går ind i en løkke.

5.5 Regler for at vælge pivot-indgang

Som nævnt i forrige afsnit skal der i algoritmen for Simplex-metoden tages nogle valg undervejs, og man kan som nævnt komme ud for uendelige løkker. For at undgå disse løkker beskrives nogle systematiske valg i dette afsnit.

Det ses i beskrivelsen af Simplex-algoritmen i afsnit 5.3, at der i trin 2 og trin 5 i pseudokode 5.3.3 er mulighed for, at flere valg kan tages. I trin 2 er det eksempelvis muligt at vælge et hvilket som helst indeks j, hvor den tilsvarende reducerede kostkoefficient er negativ. I trin 5 kan det ske, at der er flere indekstal ℓ, der giver det samme mindste θ^*, og man kan derfor vælge udgangsvariablen x_ℓ på mange forskellige måder ud fra disse muligheder. For at tage systematiske valg i trin 2 og trin 5 findes der forskellige regler kaldet *pivot-regler*.

For indgangssøjlen $\mathbf{A_j}$ til basen findes der nogle oplagte regler for, hvordan man skal vælge $\mathbf{A_j}$ ud fra de muligheder, der er. Den første regel går ud på, at den søjle $\mathbf{A_j}$ med den mest negative reducerede kostkoefficient $\hat{c}_j$ udvælges som indgangssøjlen. Som beskrevet tidligere svarer de reducerede kostkoefficienter til, hvor meget værdien af objektfunktionen formindskes ved at følge den pågældende retningsvektor $\mathbf{d}$, og vi ønsker med denne regel derfor at følge den retning, hvor værdien af objektfunktionen umiddelbart formindskes mest.

Den næste regel bygger så videre på den foregående regel, da vi nu vælger den søjle $\mathbf{A_j}$ med $\hat{c}_j < 0$ for hvilken $\theta^*|\hat{c}_j|$ er størst som indgangssøjlen, fordi det giver det korrekte billede af, hvilken retning der reducerer værdien af objektfunktionen mest. Man skal altså beregne hver $\hat{c}_j$ og hver θ^* for alle de mulige indgangssøjler, hvor $\hat{c}_j < 0$. Ved at medtage θ^* i denne vurdering ser vi altså, at man umiddelbart kan komme frem til den optimale basal mulige løsning hurtigere, men hvor vi dog bemærker, at antallet af aritmetiske regneoperationer er større ved hver iteration, netop fordi hvert $\hat{c}_j$ og hvert θ^* skal beregnes for hver eneste mulige indgangssøjle. Dog viser det sig, at beregningstiden ikke forbedres ved denne regel, hvorfor det ikke er en pivot-regel, som vi vil fokusere mere på i dette projekt.

For problemer med mange variable er der en stor sandsynlighed for, at antallet af aritmetiske regneoperationer er stort, også for reglen der vælger den mest negative $\hat{c}_j$, fordi den skal udregne alle $\hat{c}_j$. Derfor bruger man i praksis ofte mere intuitive regler i form af eksempelvis *mindste-indeks-reglen* - en regel der vælger det første beregnede $\hat{c}_j < 0$. Heraf følger det, at ikke alle reducerede kostkoefficienter nødvendigvis behøver at blive beregnet,

hvis blot vi finder en reduceret kostkoefficient, som er negativ. Der findes mange andre regler som er med til at forbedre beregningstiden, men i projektet tages der udgangspunkt i reglen om den mest negative $\hat{c}_j$ ved valg af indgangssøjlen. Reglen om mindste indeks bruges generelt, når valget står mellem flere ens elementer i forhold til valg af indgangssøjlen $\mathbf{A_j}$ og udgangssøjlen $\mathbf{A_{B(\ell)}}$.

5.6 Revidering af Simplex-metoden

Bemærk af konstruktionen af Simplex-metoden i pseudokode 5.3.3 at vi skal beregne B^{-1} ved hver iteration for at konstruere vektoren $\mathbf{u}$ og beregne de reducerede kostkoefficienter $\hat{c}_j$. At udregne B^{-1} på ny for hver iteration som gjort i pseudokode 5.3.3 er umiddelbart meget ineffektivt, da det kræver mange aritmetiske operationer, hvorfor vi nu ønsker en simpel måde at opdatere B^{-1} på, så vi for hver iteration nemt kan finde frem til den nye inverse basis-matrix $\overline{B}^{-1}$. Fra de tidligere beskrivelser ved vi, at en ny basal mulig løsning findes ved at lade en ikke-basal variabel blive bragt ind i basen sådan, at $\mathbf{A_j}$ bliver bragt ind i basis-matricen, og der tilsvarende fjernes en søjle $\mathbf{A_{B(\ell)}}$ fra basis-matricen. Intuitivt betyder det, at alle søjler på nær én forbliver de samme i den nye basis-matrix $\overline{B}$ i forhold til den gamle basis-matrix B, da vi kun udskifter én søjle ved hver iteration. Vi vil nu forklare, hvorfor og hvordan en nemmere opdatering af den inverse basis-matrix kan lade sig gøre. Bemærk da igen at $B^{-1}\mathbf{A_{B(i)}}$ for $i \neq \ell$ giver standard-vektorer, så der gælder, at

$$B^{-1}\overline{B} = \begin{bmatrix} \mathbf{e_1} & \mathbf{e_2} & \cdots & \mathbf{e_{\ell-1}} & \mathbf{u} & \mathbf{e_{\ell+1}} & \cdots & \mathbf{e_m} \end{bmatrix}. \tag{5.20}$$

Bemærk at vi kan omforme matricen i ligning 5.20 til identitets-matricen I ud fra sætning 2.6.6, da vi kan finde en matrix $Q = E_k E_{k-1} \cdots E_1$ sådan, at $QB^{-1}\overline{B} = I$. Dette svarer til, at vi kan udføre rækkeoperationer på matricen $B^{-1}\overline{B}$ og heraf opnå I. Heraf følger det så, at

$$QB^{-1}\overline{B}\overline{B}^{-1} = I\overline{B}^{-1} \iff QB^{-1} = \overline{B}^{-1},$$

og det medfører, at vi kan udføre de samme rækkeoperationer på B^{-1} for at opnå $\overline{B}^{-1}$. Kigger vi igen på matricen $B^{-1}\overline{B}$ ser vi, at for at opnå I udføres passende elementære rækkeoperationer sådan, at indgangene over og under den ℓ'te indgang i $\mathbf{u}$ bliver 0 og den ℓ'te indgang tilsvarende bliver 1, så $\mathbf{u}$ reduceres til $\mathbf{e}_\ell$. At udføre rækkeoperationer der tilsvarer opdateringen af B^{-1} til $\overline{B}^{-1}$ vil blive anvendt i den følgende implementation af *Simplex-metoden på tabelform.*

5.7 Simplex-metoden på tabelform

Simplex-metoden på tabelform er den implementation, som projektet vil gøre brug af i den senere analyse af diæt-problemet. Denne metode tager udgangspunkt i først og fremmest at vedligeholde og opdatere matricen

$$B^{-1}\begin{bmatrix} \mathbf{b} & A \end{bmatrix} \tag{5.21}$$

af størrelsen $m \times (n+1)$. I denne matrix ses det, at søjlerne bliver $\mathbf{x_B} = B^{-1}\mathbf{b}$ som løsningen for de basale variable ved den nuværende basis-matrix B og $B^{-1}\mathbf{A}_1, B^{-1}\mathbf{A}_2, \ldots, B^{-1}\mathbf{A}_n$. Denne matrix opstilles ved, at man får givet et problem på standard form, hvor man opstiller total-matricen $\begin{bmatrix} \mathbf{b} & A \end{bmatrix}$ bestående af betingelsesvektoren $\mathbf{b}$ og koefficient-matricen A. Man får så givet en basis-matrix B, hvis inverse skal multipliceres på fra venstre, så

formen bliver $B^{-1}\begin{bmatrix}\mathbf{b} & A\end{bmatrix}$. Desuden udbygger vi tabellen med en nulte række, der har til formål at holde styr på værdien af objektfunktionen i den nuværende basal mulige løsning samt opdatere de reducerede kostkoefficienter for hver iteration. Dette medfører så, at vi kan opstille tabelformen af Simplex-metoden på følgende måde.

$-\mathbf{c_B}^\intercal\mathbf{x_B}$	$\hat{c}_1$	$\hat{c}_2$	$\dots$	$\hat{c}_n$
$x_{B(1)}$				
$x_{B(2)}$	$B^{-1}\mathbf{A_1}$	$B^{-1}\mathbf{A_2}$	$\cdots$	$B^{-1}\mathbf{A_n}$
$\vdots$				
$x_{B(m)}$				

Tabel 5.5: *Opstilling af den generelle form af Simplex-metoden på tabelform.*

Rækkerne i tabellen opstilles som sagt ud fra matricen A og den tilhørende betingelsesvektor $\mathbf{b}$. Disse repræsenterer betingelserne fra standard problemet, som er givet på matrix-formen $\mathbf{b} = A\mathbf{x}$. Når den inverse af basis-matricen multipliceres på fra venstre, kan betingelserne da skrives på formen $B^{-1}\mathbf{b} = B^{-1}A\mathbf{x}$, hvilket giver tabel 5.5 på nær den nulte række, hvor rækkerne i tabellen altså repræsenterer koefficienterne for disse betingelser.

Søjlen indeholdende $\mathbf{x_B} = B^{-1}\mathbf{b}$ kaldes den *nulte søjle* og viser værdierne af basalvariablene ud fra den nuværende basis-matrix. En bestemt søjle $B^{-1}\mathbf{A_i}$ for $i \in \{1, 2, \dots, n\}$ kaldes tilsvarende for den i'te søjle. Vi husker på fra tidligere, at $\mathbf{u} = B^{-1}\mathbf{A_j}$, som vi her betegner som pivot-søjlen, hvis tilhørende variabel er indgangsvariablen. Tilsvarende betegnes den ℓ'te række som pivot-rækken, hvis tilhørende variabel er udgangsvariablen. Den indgang hvor pivot-rækken og pivot-søjlen krydser hinanden kaldes pivot-indgangen u_ℓ. Den øverste række i tabellen kaldes som sagt den nulte række, hvor den negative værdi af objektfunktionen $-\mathbf{c_B}^\intercal\mathbf{x_B} = -\mathbf{c_B}^\intercal B^{-1}\mathbf{b}$ i den nuværende basal mulige løsning vises som den første indgang. At den første indgang i den nulte række er den negative værdi af objektfunktionen i den nuværende basal mulige løsning kommer af, at rækkeoperationerne der udføres på den nulte række skal passe. De andre indgange i den nulte række viser de reducerede kostkoefficienter givet ved $\hat{\mathbf{c}}^\intercal = \mathbf{c}^\intercal - \mathbf{c_B}^\intercal\mathbf{B}^{-1}\mathbf{A}$.

Som beskrevet tidligere er målet for hver iteration at opdatere basis-matricen, så vi flytter os til en ny basal mulig løsning (hvis den nuværende løsning er ikke-degenereret), og her betyder det, at vi gerne vil opdatere matricen $B^{-1}\begin{bmatrix}\mathbf{b} & A\end{bmatrix}$ til $\overline{B}^{-1}\begin{bmatrix}\mathbf{b} & A\end{bmatrix}$. Som beskrevet i afsnit 5.6 gør vi dette ved at udføre elementære rækkeoperationer, der tilsvarer at multiplicere en matrix Q på fra venstre sådan, at vektoren $\mathbf{u}$ bliver lavet om til $\mathbf{e}_\ell$.

Når vi så skal bestemme, hvilken basis-vektor der er udgangssøjlen $\mathbf{A_{B(\ell)}}$, skal vi kigge på θ^*, der som beskrevet tidligere er givet ved

$$\theta^* = \min_{\{i \in \{B(1), B(2), \dots, B(m)\} | u_i > 0\}} \left\{ \frac{x_i}{u_i} \right\}.$$

Her svarer x_i/u_i som sagt til forholdet mellem den i'te indgang i den nulte søjle og den i'te indgang i pivot-søjlen $\mathbf{u}$. For det i hvor θ^* er mindst findes så den basis-vektor, der bliver udgangssøjlen $\mathbf{A_{B(\ell)}}$, idet søjlen, hvor pivot-indgangen befinder sig i, bearbejdes således at $\mathbf{A_j}$ kommer ind i basen ud fra rækkeoperationerne.

Ved brug af rækkeoperationer der tilsvarer multiplikation med Q fra venstre har vi tidligere vist, at dette går godt for opdateringen af $B^{-1}\begin{bmatrix}\mathbf{b} & A\end{bmatrix}$, hvorfor vi blot mangler at kigge på opdateringen af den nulte række. Og for at opdatere den nulte række anvendes da de samme elementære rækkeoperationer som for resten af tabellen, så $\hat{c}_j$ bliver 0 for indgangsvariablen. Vi skal nu vise, at disse rækkeoperationer giver det ønskede resultat for den nulte række, så opdateringen ved brug af rækkeoperationerne også går godt her.

Bemærk da at vi ud fra konstruktionen af de reducerede kostkoefficienter har, at den nulte række for hver iteration er på formen

$$\begin{bmatrix}0 & \mathbf{c}^\intercal\end{bmatrix} - \mathbf{g}^\intercal\begin{bmatrix}\mathbf{b} & A\end{bmatrix}. \tag{5.22}$$

Her har vi konstrueret $\mathbf{g}^\intercal$ sådan, at $\mathbf{g}^\intercal = \mathbf{c_B}^\intercal B^{-1}$, hvorfor den nulte række netop kommer til at bestå af den negative værdi af objektfunktionen i den nuværende basal mulige løsning som den første indgang, og hvor de resterende indgange svarer til de reducerede kostkoefficienter. Bemærk her ligeledes at størrelserne på de forskellige vektorer og matricer passer, så differencen i ligning 5.22 er defineret. Når vi så udfører elementære rækkeoperationer ud fra pivot-rækken ℓ og skaber 0 i den nulte række i pivot-søjlen, får vi en tilsvarende form af den nulte række givet ved

$$\begin{bmatrix}0 & \mathbf{c}^\intercal\end{bmatrix} - \mathbf{p}^\intercal\begin{bmatrix}\mathbf{b} & A\end{bmatrix},$$

hvor $\mathbf{p}^\intercal$ her er en ny vektor. Hvis opdateringen af den nulte række skal passe, skal der da gælde, at $\mathbf{p}^\intercal = \mathbf{c}_{\overline{\mathbf{B}}}{}^\intercal \overline{B}^{-1}$, hvilket vi nu vil vise er tilfældet.

Vi husker da i første omgang på, at $\hat{c}_j = 0$ for indgangen i den nulte række i pivot-søjlen, efter vi har udført passende elementære rækkeoperationer, hvilket medfører at

$$c_{\overline{B}(\ell)} - \mathbf{p}^\intercal \mathbf{A}_{\overline{\mathbf{B}}(\ell)} = c_j - \mathbf{p}^\intercal \mathbf{A_j} = 0$$

jævnfør ligning 5.14. Vi skal så tilsvarende undersøge, om dette også gælder for de andre søjler i den nye basis-matrix, så vi kigger nu på basis-indeks $\overline{B}(i)$ for $i \neq \ell$ efter iterationen. Bemærk da først at indgangene i den nulte række for søjlerne i den oprindelige basis-matrix er 0, da $\hat{c}_{B(i)} = 0$ for de basale variable jævnfør ligning 5.7. Heraf følger det så, at søjlerne $B^{-1}\mathbf{A_{B(i)}}$ er standard-vektorer $\mathbf{e_i}$ for $i \neq \ell$, hvorfor indgangen i pivot-rækken for disse søjler nødvendigvis også er 0. Da $B^{-1}\mathbf{A_{B(i)}}$ for $i \neq \ell$ har 0 som indgang i den nulte række og i pivot-rækken, påvirker de elementære rækkeoperationer ikke indgangene i den nulte række, hvorfor disse forbliver 0. Altså skal vektoren $\mathbf{p}^\intercal$ opfylde, at $c_{\overline{B}(i)} - \mathbf{p}^\intercal \mathbf{A}_{\overline{\mathbf{B}}(\mathbf{i})} = 0$ for alle søjler $\mathbf{A}_{\overline{\mathbf{B}}(\mathbf{i})}$ i den nye basis-matrix. Det medfører så, at $\mathbf{c}_{\overline{\mathbf{B}}}{}^\intercal - \mathbf{p}^\intercal \overline{B} = \mathbf{0}$ er opfyldt i forhold til indgangene i den nulte række ved den nye basis-matrix, og fordi $\overline{B}$ er en basis-matrix jævnfør sætning 5.3.2 og derfor også invertibel har vi så, at $\mathbf{p}^\intercal = \mathbf{c}_{\overline{\mathbf{B}}}{}^\intercal \overline{B}^{-1}$. Altså gælder der, at når vi anvender de tilsvarende elementære rækkeoperationer, så indgangen i den nulte række i pivot-søjlen bliver 0, opnår vi en ny opdateret nulte række af tabellen, der er lig med

$$\begin{bmatrix}0 & \mathbf{c}^\intercal\end{bmatrix} - \mathbf{c}_{\overline{\mathbf{B}}}{}^\intercal \overline{B}^{-1}\begin{bmatrix}\mathbf{b} & A\end{bmatrix},$$

og heraf observeres det så, at vi er nået tilbage til udgangspunktet for en ny basal mulig løsning givet ved den nye basis-matrix $\overline{B}$, som er det ønskede udgangspunkt for den næste iteration.

Vi kan nu samle op på beskrivelsen af Simplex-metoden på tabelform ved at konstruere nedenstående pseudo-algoritme.

5.7.1. Pseudokode for Simplex-metoden på tabelform

1. Man får givet en basis-matrix B, den inverse basis-matrix B^{-1} og en tilhørende basal mulig løsning $\mathbf{x}$. Vi opstiller da tabelformen for Simplex-metoden ud fra tabel 5.5.

2. Vi undersøger alle indgange $\hat{c}_i$ for $i \in \{1, 2, \ldots, n\}$ i den nulte række af tabellen i forhold til pivot-reglerne beskrevet i afsnit 5.5. Hvis alle indgangene er positive, er den nuværende basal mulige løsning optimal, og algoritmen stopper. Ellers anvender vi en pivot-regel til at udvælge $\hat{c}_j < 0$.

3. Vi betragter nu vektoren $\mathbf{u} = B^{-1}\mathbf{A_j}$ og tjekker, om indgangene er positive. Hvis ingen af indgangene er positive, er værdien af objektfunktionen for den optimale basal mulige løsning $-\infty$, og algoritmen stopper.

4. Hvis der findes indgange $u_i > 0$ beregnes brøkerne $x_{B(i)}/u_i$, og vi finder da $\theta^* = x_{B(\ell)}/u_\ell$ som den mindste af disse brøker. Bemærk at i tabellen udføres udregningen af θ^* ikke. Vi har nu fundet udgangssøjlen $\mathbf{A_{B(\ell)}}$, der skal forlade basen, og indgangssøjlen $\mathbf{A_j}$ som vil blive bragt ind i basen.

5. Der udføres elementære rækkeoperationer således, at pivot-indgangen bliver 1 og de resterende indgange i pivot-søjlen bliver 0 inklusiv indgangen i pivot-søjlen i den nulte række.

6. Gå til trin 2.

5.8 Starteksempel ud fra tabelform

For at afrunde dette kapitel vil starteksemplet blive gennemgået med de værktøjer, som er blevet introduceret i forbindelse med udviklingen af Simplex-metoden på tabelform. Dette er med henblik på at give en bedre forståelse for Simplex-metoden, og hvordan argumenterne for Simplex-metoden på tabelform afspejles i de operationer, der udføres. Det bemærkes fra det tidligere eksempel af fremgangsmåden for tabelformen i afsnit 5.1, at det kræver fire iterationer i pseudokode 5.7.1 for at opnå den optimale løsning til dette eksempel.

Vi husker på, at tabelformen fra starteksemplet i afsnit 5.1 er på formen

	x_1	x_2	x_3	x_4	x_5	x_6
0	1	−3	2	0	0	0
9	3	−1	2	1	0	0
14	−2	4	1	0	1	0
10	−4	4	8	0	0	1

Tabel 5.6

Det bemærkes her af tabel 5.6, at de tre nederste rækker er $B^{-1}\begin{bmatrix}\mathbf{b} & A\end{bmatrix}$. Den nulte søjle er resultatet af matrix-vektor produktet $B^{-1}\mathbf{b}$ og de restrerende søjler er resultatet af matrix-matrix produktet $B^{-1}A$ givet ved søjlerne $B^{-1}\mathbf{A_1}$,$B^{-1}\mathbf{A_2}$, ...,$B^{-1}\mathbf{A_6}$. Bemærk at for den begyndende basal mulige løsning er basis-matricen B givet ved søjlerne tilhørende underskudsvariablene, hvorfor B her svarer til identitets-matricen I_3. Den inverse af basis-matricen B giver da i dette tilfælde den samme matrix, så $B = B^{-1}$, hvorfor vi nu har nok oplysninger til, at Simplex-metoden kan påbegynde. Den nulte række findes over de tre nederste rækker, og vi bemærker da, at denne række er delt i to, som også stemmer overens med beskrivelserne fra tidligere. Den første indgang i øverste venstre hjørne svarer til den negative værdi af objektfunktionen i den nuværende basal mulige løsning, hvilket her er 0, eftersom underskudsvariablene selv ikke bidrager til værdien af objektfunktionen. Den anden del af den nulte række er kostkoefficient-vektoren givet ved $\hat{\mathbf{c}}^\intercal = \mathbf{c}^\intercal - \mathbf{c_B}^\intercal B^{-1}A$. Vi ser da, at $\hat{\mathbf{c}}^\intercal = \mathbf{c}^\intercal$ ved første iteration, eftersom $\mathbf{c_B}^\intercal = \mathbf{0}$. Som opsamling ved første iteration har vi da, at basis-matricen B for den nuværende tabel og den tilhørende begyndende basal mulige løsning $\mathbf{x}$ er givet ved

$$B = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix}0 & 0 & 0 & 9 & 14 & 10\end{bmatrix}^\intercal$$

Tabellen er nu blevet opstillet, hvilket vil sige, at vi er nået til trin 2 i pseudokode 5.7.1. Her kigger vi så på indgangene i den nulte række, som er de reducerede kostkoefficienter $\hat{c}_i$. Bemærk af tabel 5.6 at der kun findes ét $\hat{c}_i < 0$, hvorfor søjlen tilhørende denne værdi vælges som pivot-søjle. Den valgte søjle i tabellen bliver derfor $\mathbf{u} = B^{-1}\mathbf{A_2}$, som her er givet ved

$$\mathbf{u} = B^{-1}\mathbf{A_2} = \begin{bmatrix}-1\\ 4\\ 4\end{bmatrix}$$

Valget af denne pivot-søjle afslutter da trin 2. I trin 3 skal vi nu betragte $\mathbf{u}$ og se, om denne vektor har positive indgange. Af tabel 5.6 ser vi, at de positive indgange for dette eksempel er indgangene $u_2 = 4$ og $u_3 = 4$. Da vi har fundet $u_i > 0$, fortsætter vi til trin 4 i pseudokode 5.7.1. I dette trin skal vi finde frem til θ^*, hvilket vi gør ved at beregne brøkerne $x_{B(i)}/u_i$ for at se, hvilken brøk der giver den mindste værdi, så vi sikrer, at bevægelsen ud fra dette θ^* ikke forårsager, at den nye basal mulige løsning ligger uden for polyederet. θ^* er da givet ved

$$\theta^* = \min\left\{\frac{14}{4}, \frac{10}{4}\right\} = \frac{10}{4}$$

Ud fra valget af θ^* får vi så, at $x_{B(3)}$ bliver udgangsvariablen, så $\mathbf{A_6}$ bliver udgangssøjlen.

Nu er vi kommet til det sidste trin for en iteration af Simplex-metoden på tabelform. Her skal der udføres elementære rækkeoperationer på hele tabellen for at lave pivot-søjlen

$\mathbf{u}$ om til standard-vektoren $\mathbf{e_3}$. Bemærk at de nødvendige rækkeoperationer er givet ved

$$\begin{aligned}\frac{3}{4}r_3 + r_0 &\to r_0\\ \frac{1}{4}r_3 + r_1 &\to r_1\\ -r_3 + r_2 &\to r_2\\ \frac{1}{4}r_3 &\to r_3\end{aligned}$$

Det afslutter så den første iteration, hvilket giver tabellen herunder.

	x_1	x_2	x_3	x_4	x_5	x_6
15/2	−2	0	8	0	0	3/4
23/2	2	0	4	1	0	1/4
4	2	0	−7	0	1	−1
5/2	−1	1	2	0	0	1/4

Ved denne nye tabel kan det aflæses, at den nye basal mulige løsning er

$$\mathbf{x} = \begin{bmatrix}0 & 5/2 & 0 & 23/2 & 4 & 0\end{bmatrix}^\intercal$$

hvor værdien af objektfunktionen er $f(\mathbf{x}) = -15/2$, hvilket vi ser af den første indgang i den nulte række i tabellen.

Herefter vil næste iteration starte for at undersøge, om der eksisterer andre løsninger, der optimerer objektfunktionen yderligere. I vores eksempel kommer der tre iterationer mere, hvor de to næste iterationer følger samme argumentation. Den sidste iteration tager udgangspunkt i følgende tabel.

	x_1	x_2	x_3	x_4	x_5	x_6
13	0	0	16/5	1/5	4/5	0
6	0	0	44/5	4/5	−4/5	1
5	1	0	9/10	2/5	1/10	0
6	0	1	7/10	1/5	3/10	0

Her starter vi i trin 2 i pseudokode 5.7.1, hvor vi undersøger alle $\hat{c}_i$-indgangene i den nulte række. Det ses, at alle indgangene er positive, og derfor er den nuværende basal mulige løsning optimal, og algoritmen stopper i dette trin. Vi har da den optimale løsning givet ved

$$\mathbf{x} = \begin{bmatrix}5 & 6 & 0 & 0 & 0 & 6\end{bmatrix}^\intercal,$$

og den optimale værdi af objektfunktionen er da $f(\mathbf{x}) = -13$. I det næste kapitel vil vi anvende tabelformen af Simplex-metoden som beskrevet i de foregående afsnit på et mere praktisk eksempel og diskutere, hvad man skal være opmærksom på ved anvendelsen af Simplex-metoden i forhold til modelleringen af praktiske problemer og evalueringen af resultaterne ved brug af metoden.

6 | Anvendelse af Simplex-metoden på diæt-problemet

Dette kapitel har til formål at præsentere et udvalgt tekstbogseksempel, der kan modelleres som et lineært programmeringsproblem og herudfra løses ved brug af implementationen af Simplex-metoden, som beskrevet i kapitel 5. Dette kapitel tager udgangspunkt i [11], medmindre andet fremgår af teksten.

6.1 Diæt-problemet - matematisk modellering og standard form

Vi vil kigge på et tekstbogseksempel kendt som *diæt-problemet*, som består i, at vi som udgangspunkt får givet n forskellige madvarer $K_1, K_2, \ldots, K_n$, der hver indeholder forskellige mængder af m forskellige næringsstoffer $N_1, N_2, \ldots, N_m$, som er essentielle i det daglige indtag af mad. Mere konkret skal der en vis mængde b_i af et næringsstof N_i til, før vi overholder et dagligt indtag, som er sundt for helbredet. Vi ved desuden, hvad mængden a_{ij} af et pågældende næringsstof N_i er pr. enhed af madvaren K_j, samt hvad prisen p_j er pr. enhed af madvaren K_j. Og det er så vores opgave at minimere omkostningen ved indtag af de forskellige madvarer, så vi får dækket det daglige behov af næringsstoffer på den billigst mulige måde.

For at kunne optimere på dette problem opstilles først en matematisk model for problemet ud fra den givne information. Vi bemærker da først, at objektfunktionen vi ønsker at minimere er den samlede pris for mængden af mad som skal indtages, hvorfor vi kan beskrive objektfunktionen ved

$$f(\mathbf{x}) = \mathbf{p}^\intercal \mathbf{x}$$

Her er $\mathbf{p}$ en $n \times 1$ vektor med indgange p_j, der netop viser prisen pr. mængde af madvaren K_j, og $\mathbf{x}$ er tilsvarende en $n \times 1$ vektor med indgange x_j, der angiver mængden af de forskellige madvarer K_j, vi indtager. Bemærk at $\mathbf{p}$ derfor svarer til kostkoefficient-vektoren. Desuden ved vi så, at mængden af indtaget næringsstof N_i skal være mindst b_i, hvorfor der skal gælde, at

$$A\mathbf{x} \geq \mathbf{b},$$

hvor indgange a_{ij} i A svarer til mængden af det i'te næringsstof N_i for $i \in \{1, 2, \ldots, m\}$ pr. enhed af madvaren K_j for $j \in \{1, 2, \ldots, n\}$. Ved longhand-notationen får vi derfor opstillet dette lineære minimeringsproblem som

$$
\begin{aligned}
&\textit{Minimer} && f(x_1, x_2, \ldots, x_n) = p_1x_1 + p_2x_2 + \cdots + p_nx_n \\
&\textit{I forhold til} && a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \geq b_1 \\
& && a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \geq b_2 \\
& && \vdots \\
& && a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \geq b_m
\end{aligned}
$$

Bemærk dog at mængden af indtaget mad ikke kan være negativ i en praktisk sammenhæng, hvorfor vi tilføjer de ikke-negative begrænsninger for variablene. Og på standard form giver det så følgende longhand-notation

$$
\begin{aligned}
&\textit{Minimer} && f(x_1, x_2, \ldots, x_n) = p_1x_1 + p_2x_2 + \cdots + p_nx_n + 0s_1 + 0s_2 + \cdots + 0s_m \\
&\textit{I forhold til} && a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n - s_1 = b_1 \\
& && a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n - s_2 = b_2 \\
& && \vdots \\
& && a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n - s_m = b_m \\
&\textit{Og} && k \geq 0, \quad k \in \{x_1, x_2, \ldots, x_n, s_1, s_2, \ldots, s_m\},
\end{aligned}
$$

hvor vi også har tilføjet overskudsvariable $s_i \geq 0$ for $i \in \{1, 2, \ldots, m\}$ for at opnå denne standard form. Med udgangspunkt i ovenstående model af diæt-problemet kan vi så konkretisere de enkelte størrelser, så vi har en konkret problemstilling at optimere på med implementationen af Simplex-metoden på tabelform. Data for denne konkretisering fremgår af appendiks A, hvor resultatet ved brug af Simplex-metoden gengives i det følgende afsnit

6.2 Resultat for diæt-problemet

I eksemplet der analyseres her anvendes enheden gram for mængder af forskellige næringsstoffer og mængden af tilhørende madvarer, der skal købes ind. Derfor tager vi ikke højde for, hvilke mængder at de forskellige madvarer rent faktisk sælges i. Valget af data for næringsstoffer og madvarer samt tilhørende begrænsninger kan ses i appendiks A. I denne analyse har vi taget udgangspunkt i anbefalingerne for en 21-årig mand angående indtag af udvalgte næringsstoffer [12], samt hvad indholdet af de pågældende næringsstoffer er i de valgte madvarer [13], hvor de valgte madvarer og næringsstoffer er gengivet i følgende tabel.

Næringsstoffer	**Madvarer**
Fibre	Havregryn, x_1
Vitamin B6	Mælk, x_2
Vitamin B12	Rugbrød, x_3
Vitamin C	Banan, x_4
Thiamin (B1)	Leverpostej, x_5
Riboflavin (B2)	Pasta, x_6
Niacin (B3)	Oksekød, x_7
Folinsyre (B9)	Broccoli, x_8

Tabel 6.1: *En tabel der viser de næringsstoffer og madvarer, der studeres i analysen til eksemplet på dette diæt-problem. Bemærk desuden at madvarerne er noteret ved x_i for $i \in \{1, 2, \ldots, 8\}$ som de variable.*

Ved anvendelse af Simplex-metoden på problemet som opstillet i appendiks A, hvor vi benytter Python-koden for Simplex-metoden i appendiks B, får vi da en optimal løsning $\mathbf{x}$ og optimal værdi af objektfunktionen $f(\mathbf{x})$ givet ved

$$\mathbf{x} = \begin{bmatrix} 380.10 & 0 & 0 & 187.96 & 80.53 & 0 & 0 & 1.5 \end{bmatrix}^\mathsf{T} \tag{6.1}$$

$$f(\mathbf{x}) = 9.22 \text{ kr.} \tag{6.2}$$

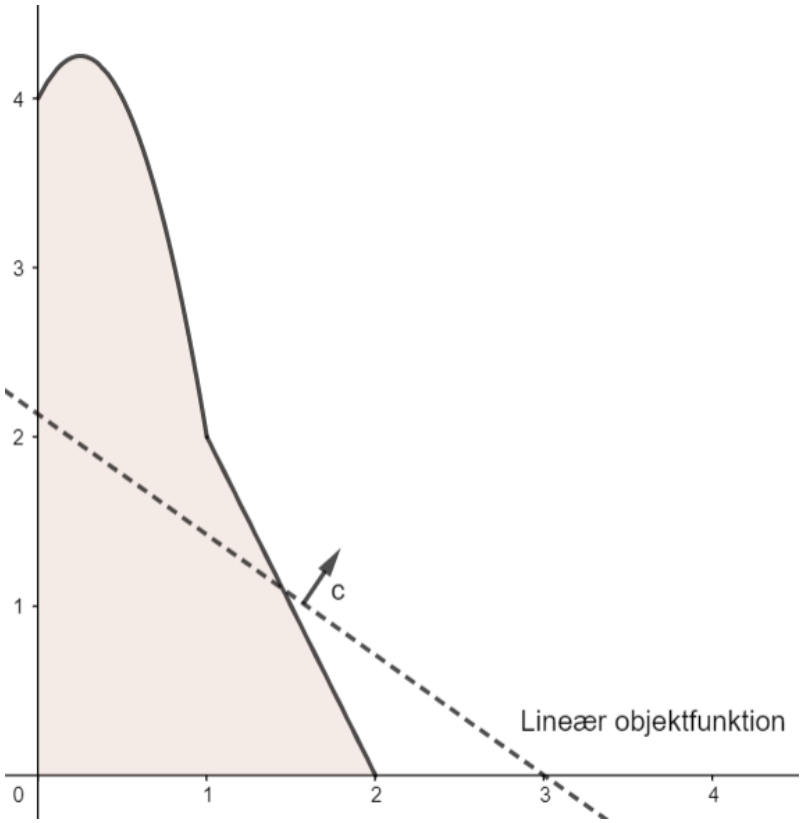
Ved modellering af disse ganske få næringstoffer ses der tydeligt et kulinarisk problem. Det viser nemlig, at Simplex-metoden giver et svar for en optimal løsning ud fra vores modellering, men at dette resultat ikke giver så meget mening for den kost, som ønskes anbefalet. Dette vil blive diskuteret nærmere i det følgende diskussions- og perspektiverings kapitel.

7 | Diskussion og perspektivering

Dette afsnit har til formål at diskutere nogle af de problemstillinger, der opstår i forbindelse med implementationen af Simplex-metoden både generelt og specifikt i forhold til det analyserede diæt-problem.

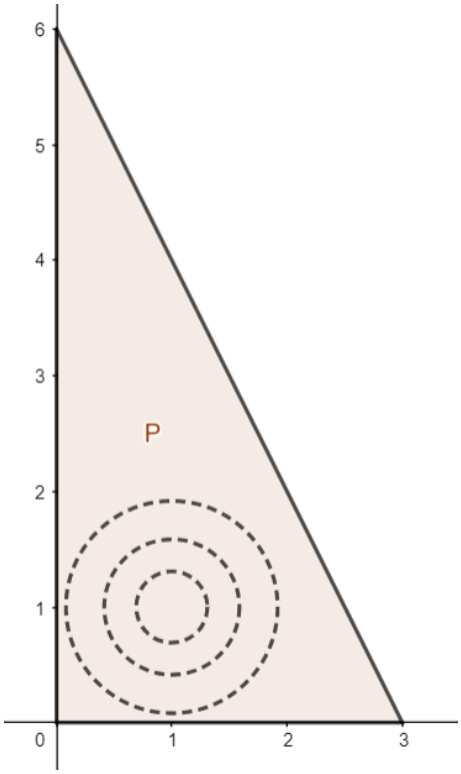
7.1 Ikke-lineære problemer og Simplex-metoden

Èn af de væsentligste punkter at komme omkring i forhold til Simplex-metoden er, at implementationen af Simplex-metoden i dette projekt *kun* kan løse lineære programmeringsproblemer, eftersom idéen med Simplex-metoden netop er en systematisk undersøgelse af basal mulige løsninger (eller det der svarer til hjørnepunkter i det givne polyeder). Det betyder også, at Simplex-metoden ikke kan løse ikke-lineære programmeringsproblemer, hvilket generelt følger af to konkrete omstændigheder for denne type problemer. Den ene omstændighed er, at et ikke-lineært programmeringsproblem kan være ikke-lineært i den forstand, at en *betingelse ikke er lineær*, hvilket kan få den betydning, at en optimal løsning ikke nødvendigvis eksisterer i et hjørnepunkt. En illustration af dette er vist i figur 7.1 herunder.



Figur 7.1: *Et polyeder med en ikke-lineær betingelse hvor værdien af en lineær objektfunktion ønskes optimeret, men hvor den optimale løsning ikke eksisterer i et hjørnepunkt.*

Af ovenstående figur skal man bemærke, at hvis vi optimerer værdien af objektfunktionen ved at forskyde niveaukurven for objektfunktionen i retningen af vektoren $\mathbf{c}$, vil det betyde, at den optimale basal mulige løsning ikke findes i et hjørnepunkt af polyederet, hvorfor vi ikke kan finde frem til den optimale løsning ved brug af Simplex-metoden. Den anden omstændighed er, at *objektfunktionen ikke er lineær*, hvilket er illustreret på figur 7.2 herunder.



Figur 7.2: *Et polyeder med lineære betingelser og niveaukurver for en ikke-lineær objektfunktion, der viser, at en optimal løsning ikke findes nær hjørnepunkter.*

I dette tilfælde betyder det igen, at Simplex-metoden ikke vil kunne finde den optimale løsning, eftersom den optimale løsning ikke eksisterer i et hjørnepunkt. Det er derfor vigtigt at pointere, at funktionaliteten af Simplex-metoden afhænger af, at modellen for det givne problem er lineært, hvilket vil blive beskrevet nærmere i det følgende afsnit.

7.2 Den matematiske modellering og Simplex-metoden

Af resultatet for det konkretiserede diæt-problem ved brug af Simplex-metoden bemærkes det, at det i denne model ikke er optimalt hverken at indtage mælk, rugbrød, pasta eller oksekød for at opnå det daglige indtag af de nødvendige næringsstoffer. I praksis giver det selvfølgelig ikke så meget mening, da sammensætningen af mad også spiller ind i forhold til måltider, og hvilke madvarer der naturligt skal spises sammen. Eksempelvis spiser man nok ikke havregryn uden mælk. I forhold til anvendelsen af Simplex-metoden viser dette resultat, at der findes en begrænsning på valg af optimum, hvis vi samtidig skal opfylde, at vores kost er varieret. Bemærk desuden at vi mangler mange andre næringsstoffer at undersøge i forhold til det daglige indtag, hvilket også kan få betydning for, at kosten bliver mindre varieret, da vi mangler at indtage madvarer, som sikrer andre næringsstoffer.

Som en sidste ting er det også vigtigt at være opmærksom på, at diæt-problemet i virkeligheden ikke er et lineært problem, hvilket eksempelvis kan ses ved optaget af jern i forhold til calcium. Eksperimenter har vist, at hvis man indtager 165 mg. calcium, eksempelvis ved at drikke et glas mælk, så optager man 50 % mindre jern i samme måltid [14, p. 547]. Ud fra dette ses det, at jern og calcium afhænger af hinanden indbyrdes, hvilket kommer til at betyde, at begrænsningerne ikke længere er lineære.

Et andet problem er også, at vi i vores diæt-problem ikke har en øvre grænse for næringsstofferne, hvilket kan blive et problem for eksempelvis indtagelsen af vitamin-B6. For vitamin-B6 er den øvre grænse 25 mg/dag, og hvis den overskrides over en længere periode, risikerer man at få neurologiske symptomer eller forgiftning af nervesystemet [14, p. 430]. Dette kan dog nemt løses ved at tilføje en øvre grænse for, hvor meget vitamin-B6 man højst må indtage dagligt.

Et tredje problem, som også gør at diæt-problemet ikke er lineært i praksis, er at dagligvarerbutikkerne kan have mængderabat på madvarer. Dette vil sige at prisen ikke er konstant i forhold til mængde, hvilket vil gøre at objektfunktionen ikke vil blive lineær. Et eksempel på dette er at man nogle gange kan købe tre for prisen af to, her vil prisen per gram ændre sig fra om man køber to eller om man køber tre.

Selvom diæt-problemet ikke er lineært i virkeligheden, kan Simplex-metoden stadig anvendes til at finde en optimal løsning, da metoden løser den givne lineære model der opstilles - man skal blot være opmærksom på, om det resultat man får giver mening i en praktisk sammenhæng og da eventuelt tilpasse sin model. Anvendelsen af Simplex-metoden afhænger derfor af, om problemet man analyserer på kan modelleres til at være lineært, for så vil Simplex-metoden altid virke [5] jævnfør kapitel 5.

7.3 Problemer ved kodning af Simplex-metoden

Det viser sig, at implementationen af Simplex-metoden som et program i Python også kan medføre problemer. Eksempelvis foreligger der en naturlig præcisionsfejl med hensyn til afrunding af tal ved de aritmetiske operationer, som Python benytter. Her er vi begrænset af antallet af betydende cifre i repræsentationen af tal, hvorfor vi kan forvente at få akkumulerende fejl i større problemer, hvor der skal bruges langt flere iterationer for at opnå en optimal løsning. Specifikt i forhold til algoritmen for Simplex-metoden på tabelform kan dette blive et problem, når vi anvender rækkeoperationer på tal, der er en smule fejlbehæftet i henhold til antallet af betydende cifre, vi kan repræsentere, eftersom dette vil akkumulere fejlen og skabe en usikkerhed i forbindelse med opdateringen af den inverse basis-matrix B^{-1} [6, p. 107-108]. For større problemer i praksis kan man undgå dette ved at beregne $\overline{B}^{-1}$ på ny efter et vist antal iterationer af Simplex-metoden på tabelform ud fra det faktum, at vi undervejs holder styr på indeks for basis-indgangene. Med andre ord kan vi altid finde frem til den egentlige basis-matrix ved at bruge basis-indeks og herudfra beregne den inverse basis-matrix til det pågældende tidspunkt i algoritmen. I forhold til eksemplet der studeres i dette projekt kommer det dog ikke til at have en så stor betydning, eftersom diæt-problemet indeholder forholdsvis få variable, og der tilsvarende ikke skal bruges så mange iterationer, før vi har fundet en optimal løsning.

Bemærk desuden af konstruktionen af algoritmen for Simplex-metoden på tabelform at der kræves, at vi på forhånd kender en basal mulig løsning med en tilhørende basis-matrix.

Det er kun i tilfælde af, at vores problem indeholder en identitets-matrice, at vi nemt kan finde frem til en begyndende basal mulig løsning og den tilhørende basis-matrix blot ved at kigge på opstillingen af problemet. En måde at udbygge algoritmen på vil derfor være at kigge på, om vi aritmetisk kan finde frem til en begyndende basal mulig løsning og en tilhørende basis-matrix, som vil blive beskrevet kort i afsnit 7.4.

For den implementation af Simplex-metoden der er blevet konstrueret i dette projekt findes den første basal mulige løsning naivt ved at afprøve forskellige baser og tjekke, om de givne søjler rent faktisk er lineært uafhængige, samt om den tilhørende entydigt bestemte løsning er mulig. Når en sådan løsning er blevet fundet, anvendes denne som den begyndende basal mulige løsning i implementationen, og Simplex-metoden på tabelform kan påbegyndes. Bemærk at dette er meget ineffektivt, eftersom algoritmen kan bruge lang tid på at undersøge mange baser jævnfør binomialkoefficienten i ligning 4.10 i afsnit 4.3, før en basal mulig løsning rent faktisk findes, og der er derfor mulighed for forbedring af algoritmen på dette punkt.

Desuden skal man lægge mærke til af implementationen af Simplex-metoden i dette projekt, at valg af pivot-søjle i dette projekt vælges ud fra den reducerede kostkoefficient $\hat{c}_j$ som er mest negativ. Dette vil ikke nødvendigvis forhindre algoritmen i at gå ind i en løkke på samme måde, som hvis vi modsat havde lavet implementeringen med mindsteindeks-reglen [6, p. 111]. Dog har vi valgt implementeringen med valg af pivot ud fra den reducerede kostkoefficient, som er mest negativ ud fra ønsket om, at vi skal nå frem til en optimal løsning ved at bevæge os i den retning, hvor værdien af objektfunktionen reduceres mest muligt for på den måde at opnå den optimale løsning hurtigere.

7.4 Udvidelser af Simplex-metoden

I Simplex-metoden på tabelform som implementeret i dette projekt har vi som sagt brug for en begyndende basis-matrix B og en tilhørende basal mulig løsning, før vi kan påbegynde iterationer af Simplex-metoden. I henhold til Python-koden i appendiks B findes denne begyndende basis-matrix naivt ved at undersøge kombinationer af søjler, indtil der findes en basis-matrix, hvor den tilhørende basale løsning er mulig. For at undgå at skulle lave denne naive undersøgelse af forskellige baser findes der en udvidelse af Simplex-metoden på tabelform benævnt som *to-fase-metoden* [6, p. 116-117]. To-fase-metoden består, som fremgår af navnet, af to faser, hvor anden fase er det samme som Simplex-metoden på tabelform, men første fase er en udvidelse af metoden, hvor vi løser et udvidet problem ved at tilføje nogle kunstige variable.

En anden udvidelse til Simplex-metoden er at anvende *dualitet* [5], hvor man omskriver det givne problem ved at ændre objektfunktionen og betingelserne (detaljerne for denne omskrivning vil ikke blive beskrevet her). Fordelen ved at anvende dualitet er, at man eksempelvis kan opnå færre variable, således problemet er mere simpelt at løse.

8 | Konklusion

Formålet med dette kapitel er at besvare problemformuleringen på baggrund af indholdet i projektet. For at besvare problemformuleringen er der i projektet tre overordnede teriområder. Det første område er lineær algebra, hvor væsentlige definitioner og sætninger indgår for at kunne beskrive teorien bag Simplex-metoden. Her er vigtige begreber såsom matricer, vektorer og lineær uafhængighed blevet introduceret.

Dette leder så frem til andet teoretiske område omhandlende opstillingen af lineære programmeringsproblemer på standard form ved hjælp af longhand- og matrix-notation, hvor vi har n variable og m betingelser. Desuden beskrives den grafiske løsning af lineære programmeringsproblemer ved et eksempel af produktionen af borde og stole, som fører til en opsummering af de fire udfald, et lineært programmeringsproblem kan have - præcis én optimal løsning, et uendeligt antal optimale løsninger, ingen løsninger eller at problemet er ubegrænset. Dernæst beskrives den geometriske og algebraiske forståelse af lineær programmering, hvor konvekse kombinationer og konvekse mængder spiller en stor rolle. I dette afsnit beskrives det da, hvordan et programmeringsproblem kan beskrives ved et polyeder, som tilsvarende er afgørende for forståelsen af ekstremumspunkter og disses sammenhæng med basal mulige løsninger, hvor vi skal have mindst n aktive lineært uafhængige betingelser for at have en basal løsning.

Det sidste teoretiske område er selve beskrivelsen af Simplex-metoden, hvor et eksempel bruges som introduktion til metoden for at give en intuitiv forståelse. Herefter introduceres blandt andet vigtige notationer som den j'te mulige basisretning $\mathbf{d}$ og den reducerede kostkoefficient $\hat{c}_j$. Dette leder frem til opbygningen af Simplex-metoden, hvor en generel iteration af Simplex-metoden i form af pseudokode præsenteres. Her introduceres blandt andet θ^*, som fortæller, hvor langt vi kan bevæge os i retningen af $\mathbf{d}$ til en ny basal mulig løsning, så vi stadig er i polyederet. Det beskrives kort, hvilke problemer der kan opstå i forbindelse med degenereret basal mulige løsninger, og det leder frem til regler for at vælge pivot-placering, hvor dette projekt anvender reglen om at vælge den mest negative $\hat{c}_j$ og ellers mindste-indeks-reglen. Alle disse notationer og regler bruges så til at implementere Simplex-metoden på tabelform i en iteration beskrevet ved pseudokode 5.3.3. Til slut gennemgås starteksemplet igen med fokus på at forklare notationerne og reglerne for Simplex-metoden, som beskrevet i de foregående afsnit.

Derefter har vi implementeret Simplex-metoden på tabelform i en algoritme, som vi afprøver på et eksempel i form af tekstbogseksemplet diæt-problemet, hvor opstilling og resultater for dette minimeringsproblem vises i afsnit 6. Selve data til diæt-problemet og algoritmen kan ses i appendix A og B.

Slutteligt diskuteres problemer i forhold til Simplex-metoden og den matematiske modellering, hvor lineariteten af et problem har stor betydning for, hvor godt et problem kan modelleres som værende lineært. Det kan altså konkluderes, at i anvendelsen af Simplex-metoden er det *altid* vigtigt at sammenholde de resultater, man får med de tilhørende modelleringer og afgrænsninger, man har lavet af den givne problemstilling, for ellers vil resultaterne i mange tilfælde ikke have en praktisk anvendelse. Derudover diskuteres nogle

problemer ved kodning af Simplex-metoden i forhold til afrundningsfejl, valget af en begyndende basal mulig løsning og valget af pivot-regler for at undgå degeneration. Dertil vil komme en perspektivering til udvidelser af Simplex-metoden.

Bibliografi

[1] Forfatterne og Systime. „Kapitel 6 - Lineær Programmering“. Dansk. I: (2011), s. 32. URL: https://systime.dk/fileadmin/indhold/SupplerendeMaterialer/Matematik_hhx/Matematik_B_hhx/Forside/Matematik_B_06_LineaerProgrammering.pdf (sidst set 20.02.2019).

[2] *Traces and emergence of nonlinear programming*. en. Basel: Birkhäuser, 2014. ISBN: 978-3-0348-0438-7.

[3] *The Sveriges Riksbank Prize in Economic Sciences in Memory of Alfred Nobel 1975*. en-US. URL: https://www.nobelprize.org/prizes/economic-sciences/1975/koopmans/lecture/.

[4] Mikulás Luptácik. *Mathematical Optimization and Economic Analysis*. Engelsk. Vienna University of Economics og Business Administration, Austria: Springer, 2010. ISBN: 978-0-387-89551-2. (Sidst set 17.04.2019).

[5] Alpha C. Chiang. *Fundamental methods of mathematical economics*. 3rd ed. New York: McGraw-Hill, 1984. ISBN: 978-0-07-010813-4.

[6] Dimitris Bertsimas og John N. Tsitsiklis. *Introduction to Linear Optimization*. Engelsk. Massachusetts Institute of Technology: Athena Scientific, 1997. ISBN: 1-886529-19-1. (Sidst set 17.03.2019).

[7] Lawrence E. Spence, Arnold J. Insel og Stephen H. Friedberg. *Elementary Linear Algebra 2015*. Engelsk. 5. udg. UK: Pearson, 2015. ISBN: 978-0-13-187141-0. (Sidst set 05.02.2019).

[8] Saul I. Gass. *Linear Programming - Methods and Applications*. Engelsk. 5. udg. University of Maryland, New York: Dover Publications, 2003. ISBN: 978-0-486-43284-7. (Sidst set 17.03.2019).

[9] Gregory Gutin. *Computational Optimisation*. self, jan. 2018.

[10] Marco Cuturi. „Linear Programming and Convex Analysis“. en. I: (). Marco Cuturi er professor i statistik ved Institut Polytechnique de Paris, s. 32. URL: http://marcocuturi.net/Teaching/ORF522/lec4v2.pdf (sidst set 24.03.2019).

[11] Thomas S. Ferguson. „Linear Programming - A Concise Introduction“. Engelsk. I: URL: https://www.math.ucla.edu/~tom/LP.pdf (sidst set 29.04.2019).

[12] Australian Government Department of Health. *Daily nutrient requirements calculator*. en. Maj 2019. URL: https://www.eatforhealth.gov.au/page/eat-health-calculators/calculated/1556700855.

[13] DTU Fødevareinstituttet Afdeling for Risikovurdering og Ernæring. *Frida - Database med fødevaredata udgivet DTU Fødevareinstituttet*. 2019. URL: https://frida.fooddata.dk/?.

[14] *Nordic nutrition recommendations 2012*. English. OCLC: 922600316. Place of publication not identified: Nordic Council Of Ministe, 2014. ISBN: 978-92-893-2670-4.

[15] rema 1000. *REMA 1000*. da. 2019. URL: https://rema1000.dk/ (sidst set 14.05.2019).

Appendiks

A | Data til eksempel for diæt-problemet

I det følgende vil der være en beskrivelse af den konkrete opstilling af eksemplet til diæt-problemet, som er blevet løst ved hjælp af Simplex-metoden. Vi kigger her på 8 forskelligt udvalgte mikro-næringsstoffer samt 8 forskellige madvarer, der indtages i løbet af dagen. Mere specifikt fokuserer vi på følgende næringsstoffer og madvarer.

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
	Havregryn	**Mælk**	**Rugbrød**	**Banan**	**Leverpostej**	**Pasta**	**Oksekød**	**Broccoli**
Fibre	1120000	0	850000	160000	40000	200000	0	330000
Vitamin B6	14.4	4.92	15.2	30.5	21.0	1.0	26.1	20.0
Vitamin B12	0	0.04	0	0	0.99	0	0.204	0
Vitamin C	0	130	0	1120	2900	0	0	3960
Thiamin (B1)	42.4	4.63	17.6	3.99	13.0	4.0	4.41	6.3
Riboflavin (B2)	12.5	17.4	6.54	1.41	102	1.0	16.6	12.3
Niacin (B3)	80	9.48	61	59.4	440	30	393	55.3
Folinsyre (B9)	5.0	1.15	2.41	3.8	17	0.20	0.963	10.8

Tabel A.1: *Tallene i tabellen viser mængden af næringsst offer i gram pr. gram af hver madvarer med en faktor* 10^{-7}.

I forhold til diæt-problemet har vi da undersøgt minimumsgrænsen for det daglige indtag af disse næringsstoffer for en 21-årig mand [12], samt hvad indholdet af næringsstofferne er i madvarerne pr. gram af de tilhørende madvarer [13] (dette er indgangene i A i matrix-notationen herunder). Tilsvarende er der blevet undersøgt priser af de forskellige madvarer i forhold til mængde [15]. Alt i alt medfører det, at vi kan opstille følgende matrix-notation for det generelle minimeringsproblem

$$\begin{aligned} &\textit{Minimer} && f(\mathbf{x}) = \mathbf{p}^\intercal \mathbf{x} \\ &\textit{I forhold til} && A\mathbf{x} \geq \mathbf{b} \end{aligned}$$

hvor der gælder følgende om de enkelte matricer og vektorer

$$\mathbf{p} = \begin{bmatrix} 0.01 \\ 0.0085 \\ 0.024 \\ 0.02 \\ 0.02 \\ 0.009 \\ 0.03 \\ 0.03 \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_8 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 30 \\ 0.0013 \\ 2.4 * 10^{-6} \\ 0.045 \\ 0.0012 \\ 0.0013 \\ 0.016 \\ 0.0004 \end{bmatrix}$$

Figur A.1: *Kostkoefficient-vektoren* $\mathbf{p}$*, variabelvektoren* $\mathbf{x}$ *og betingelsesvektoren* $\mathbf{b}$ *for dette konkrete eksempel på diæt-problemet.*

$$A = 10^{-7} \begin{bmatrix} 1120000 & 0 & 850000 & 160000 & 40000 & 200000 & 0 & 330000 \\ 14.4 & 4.92 & 15.2 & 30.5 & 21.0 & 1.0 & 26.1 & 20 \\ 0 & 0.04 & 0 & 0 & 0.99 & 0 & 0.204 & 0 \\ 0 & 130 & 0 & 1120 & 2900 & 0 & 0 & 3960 \\ 42.4 & 4.63 & 17.6 & 3.99 & 13.0 & 4.0 & 4.41 & 6.3 \\ 12.5 & 17.4 & 6.54 & 1.41 & 102 & 1.0 & 16.6 & 12.3 \\ 80 & 9.48 & 61 & 59.5 & 440 & 30 & 393 & 55.3 \\ 5.0 & 1.15 & 2.41 & 3.8 & 17 & 0.20 & 0.963 & 10.8 \end{bmatrix}$$

Figur A.2: *Matricen A med indgange a_{ij} der viser mængden af det i'te næringsstof N_i der er til stede i den j'te madvare K_j pr. gram af denne madvare [13].*

Bemærk her at værdien af objektfunktionen regnes i *kr*, indgangene i $\mathbf{p}$ regnes i *kr.* pr. gram af en pågældende madvare, variablene i $\mathbf{x}$ regnes i gram af madvaren, indgangene i A regnes i gram af næringsstof pr. gram af hver madvare og indgangene i $\mathbf{b}$ regnes i gram af næringsstof. Ud fra denne opstilling af problemet finder vi først standard formen ved at tilføje overskudsvariable s_i for $i \in \{1, 2, \ldots, 8\}$ til hver betingelse, og vi er så klar til at anvende Simplex-metoden til at finde den optimale løsning $\mathbf{x}$ og den tilhørende optimale værdi af objektfunktionen $f(\mathbf{x})$. Detaljerne for denne anvendelse vil ikke blive beskrevet her, men implementationen med Python-koden for Simplex-metoden til løsning af dette problem findes i appendiks B. Løsningen vi får ved at anvende Simplex-metoden på dette eksempel er præsenteret i kapitel 6, hvor resultatet desuden diskuteres nærmere i kapitel 7.

B | Python-kode for Simplex-metoden

B.1 Brute-force af begyndende basis-matrix med tilhørende basal mulig løsning

```
def findSoultion(coefficientMatrix, b): #en funktion der skal finde
    en begyndende basal mulig løsning og tager koefficient-matricen
    og betingelsesvektoren som input
    n = coefficientMatrix.shape[1] #vi gemmer antallet af søjler i
        koefficient-matricen som n
    m = coefficientMatrix.shape[0] #vi gemmer antallet af rækker i
        koefficient-matricen som m

    for basis in combinations(range(n), m): #vi tjekker baser ud
        fra de kombinationer, vi kan lave for valg af basale
        variable ud af de n variable
        try:
            solution = np.linalg.inv(coefficientMatrix[:, basis]) *
                b.transpose() #vi beregner en løsning for de basale
                variable med den nuværende basis
            if np.all(solution >= 0): #vi tjekker, om alle
                indgangene i x_B er positive
                solution = solution.transpose()
                solutionFix = np.zeros((1, n)) #vi laver en vektor
                    med n indgange, der i starten kun består af 0'er
                for i, index in enumerate(basis):
                    solutionFix[0, index] = solution[0, i] #vi
                        tilpasser basis-indgangene i x_B til x (den
                        nye vektor vi har lavet med n indgange)
                return basis, solutionFix #vi returnerer basis-
                    indeks og vektoren x som den første basal mulige
                    løsning
        except Exception as e:
            #anvendes i tilfælde af at søjlerne ud fra basis-indeks
                ikke er lineært uafhængige, så udregningen af x_B
                ikke er defineret, og så skal koden afprøve med nye
                basis-indeks
    return None, None #i tilfælde af at der ikke eksisterer nogen
        basale mulige løsning skal algoritmen ikke returnerer noget
```

B.2 Implementation af pseudokoden for iterationer af Simplex-metoden

```
import numpy as np

#funktion til at generere tabelformen, hvor vi som input kræver en
    begyndende basal mulig løsning, koefficient-matricen A,
    basisindgangene B(i) og kostkoefficient-vektoren i
    objektfunktionen - bemærk at alle vektorer som udgangspunkt er r
    ække-vektorer med numpy-modulet i Python
def generateTable(solution, coefficientMatrix, basisEntry,
    costCoefficient):
    basis = coefficientMatrix[:, basisEntry] #vi finder basis-
        matricen $B$
    basisInv = np.linalg.inv(basis) #vi finder den inverse basis-
        matrix

    costResult = costCoefficient * solution.transpose() #værdien af
         objektfunktionen i den nuværende løsning
    reduseCostVectors = costCoefficient - costCoefficient[:,
        basisEntry] * basisInv * coefficientMatrix #den reducerede
        kostkoefficient-vektor
    InvMultipyCefficientMatrix = basisInv * coefficientMatrix #B
        ^{-1} * A

    nulrækken = np.concatenate((costResult, reduseCostVectors),
        axis=1) #nulrækken i tabellen
    totalMatrix = np.concatenate((solution[:, basisEntry].transpose
        (), InvMultipyCefficientMatrix), axis=1) #total-matricen i
        tabelformen

    tabel = np.concatenate((nulrækken, totalMatrix)) #den færdige
        tabel
    return tabel

#tabelformen for Simplex-metoden med samme input
def simplextabelForm(solution, coefficientMatrix, basisEntry,
    costCoefficient):
    m = coefficientMatrix.shape[0] #antallet af betingelser i det
        lineære programmeringsproblem
    tabel = generateTable(solution, coefficientMatrix, basisEntry,
        costCoefficient) #vi genererer tabellen
    while True:
        if (np.all(tabel[0, 1:] >= 0)): #vi tjekker, om alle
            reducerede kostkoefficienter er positive
            return True, tabel, basisEntry #den nuværende løsning
                er optimal

        j = pivotSelection(tabel) #vi vælger, hvilken regel for
            valg af pivot, vi ønsker at bruge

        if np.all(tabel[1:, j] < 0): #vi tjekker, om alle indgange
            i u-vektoren er negative
            return False, tabel, basisEntry #i så fald vil den
                optimale løsning være -uendelig
```

```
        minValue, l = None, None
        for i in range(1, m + 1):
            if tabel[i, j] <= 0: #vi tjekker indgangene i u-
                vektoren, og hvis den nuværende indgang er negativ,
                går vi videre til næste indgang
                 continue
            value = tabel[i, 0] / tabel[i, j] #vi udregner brøken
                theta = x_B(i)/u_i for positive u_i
            if l is None or value < minValue: #hvis l ikke har
                nogen værdi, eller den beregnede værdi er mindre end
                 den nuværende, opdaterer vi l
                 l, minValue = i, value #vi gemmer værdien af l og
                     brøken (theta* = x_B(l)/u_l)

        for i in range(m + 1): #vi laver en for-løkke som gennemgår
             alle rækker
            pivotindgang = tabel[l, j] #vi definerer pivot-
                indgangen
            indgang = tabel[i, j] #den i'te indgang i u-søjlen
            if i == l: #hvis indgangen er pivot-indgangen sikrer vi
                , at værdien bliver 1 ved at dividerer hele rækken
                med pivot-indgangen
                 tabel[l, :] = (1 / pivotindgang) * tabel[l, :]
            else: #hvis ikke det er pivot-rækken, vi er kommet til,
                 skal vi gøre følgende
                 tabel[i, :] = tabel[i, :] - tabel[l, :] * (indgang
                    / pivotindgang) #vi sørger for, at indgangene i
                    u-søjlen bliver 0 for alle andre rækker end l
        basisEntry[l - 1] = j - 1 #vi trækker én fra i rækken og én
             fra u-søjlen for at få basis-indeks til at passe med
            basis-matricen


def pivotSelection(tabel): #valg af pivot ud fra mindste reducerede
     kostkoefficient
    j = None
    for i in range(1, tabel.shape[1]): #svarer til at vi gennemgår
        alle reducerede kostkoefficienter (bemærk at shape[1] giver
        størrelsen på tabellen i forhold til antal søjler, så n+1)
         if tabel[0, i] >= 0: #hvis den nuværende reducerede
            kostkoeffeicient er større end 0, går vi videre til næ
            ste værdi
             continue
         if j is None or tabel[0, i] < tabel[0, j]: #hvis den nuvæ
            rende reducerede kostkoefficient er mere negativ end den
             forrige, opdaterer vi j
             j = i
    return j #vi returnerer indgangen c_j med den mindste værdi


def pivotSelectionSmallestSubscript(tabel): #valg af pivot ud fra
    mindste-indeks regel
     for i in range(1, tabel.shape[1]): #svarer igen til at vi skal
        gennemgå alle reducerede kostkoefficienter
         if tabel[0, i] >= 0: #hvis den nuværende reducerede
```

```
            kostkoefficient er positiv, går vi videre til den næste
             continue
        return i #så snart vi kommer til et indeks i for en
            reduceret kostkoefficient c_i < 0, vælger vi den


def main(): #eksempel til at blive gennemgået i koden (svarer til
    eksemplet i Simplex-afsnittet) og input defineres herunder
    coefficientMatrix = np.mat([[3, -1, 2, 1, 0, 0],
                                [-2, 4, 1, 0, 1, 0],
                                [-4, 4, 8, 0, 0, 1]], dtype=np.
                                   float64)
    basisEntry = [3, 4, 5]
    solution = np.mat([0, 0, 0, 9, 14, 10], dtype=np.float64)
    costCoefficient = np.mat([1, -3, 2, 0, 0, 0], dtype=np.float64)
    print(simplextabelForm(solution, coefficientMatrix, basisEntry,
        costCoefficient))


main()
```