Masteroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for samfunns- og utdanningsvitenskap
Institutt for lærerutdanning

Tuva Liisa Jacobsen Sevildal

Programmeringsoppgaver i lærebøker i matematikk

En lærebokanalyse om muligheter for algoritmisk tenkning, problemløsning og arbeid med matematiske kunnskapsområder på Vg1

Masteroppgave i Master i fagdidaktikk, retning matematikk
Veileder: Anders Sanne
Desember 2023

NTNU
Kunnskap for en bedre verden

Tuva Liisa Jacobsen Sevildal

Programmeringsoppgaver i lærebøker i matematikk

En lærebokanalyse om muligheter for algoritmisk tenkning, problemløsning og arbeid med matematiske kunnskapsområder på Vg1

Masteroppgave i Master i fagdidaktikk, retning matematikk
Veileder: Anders Sanne
Desember 2023

Norges teknisk-naturvitenskapelige universitet
Fakultet for samfunns- og utdanningsvitenskap
Institutt for lærerutdanning

NTNU
Kunnskap for en bedre verden

Sammendrag

Denne masterstudien har undersøkt programmeringsoppgavene i trykte lærebøker i faget matematikk 1T. Formålet med masteroppgaven er å vurdere hvilke muligheter elevene får til å arbeide med algoritmisk tenkning, problemløsning og matematikk, gjennom programmeringsoppgavene i de trykte lærebøkene. Disse fokusområdene er valgt, fordi de programmering nylig er blitt en del av matematikkfaget i norsk skole, og at programmering nevnes i tett sammenknytting med algoritmisk tenkning og problemløsning i læreplanen (Kunnskapsdepartementet, 2019b). For å undersøke programmeringsoppgavene er følgende forskningsspørsmål utviklet:

1. Hvilke muligheter for algoritmisk tenkning finnes i programmeringsoppgavene i de trykte lærebøkene i matematikk 1T?
2. Hvilke muligheter for kreativ resonnering finnes i programmeringsoppgavene i lærebøkene i matematikk 1T?
3. Hvilke muligheter for arbeid med matematiske kunnskapsområder finnes i programmeringsoppgavene i lærebøkene for matematikk 1T?

Masterstudien er en kvalitativ dokumentanalyse av de trykte lærebøkene i faget matematikk 1T (MAT09-01).

For å samle data og svare på forskningsspørsmålene har kvalitative og kvantitative metoder blitt benyttet. Som en del av arbeidet med forskningsspørsmål 1 har jeg undersøkt hvilken definisjon for algoritmisk tenkning som kan ha blitt benyttet i utarbeidelsen av læreplanen.

Studiens bidrag er et rammeverk utviklet for å undersøke programmering i en norsk skolekontekst, i tråd med ordlyden i læreplanen for matematikk 1T. Funnene viser at elevene gis få muligheter til å arbeide med programmering, men at de programmeringsoppgavene som finnes gir muligheter til kreativ resonnering i omtrent 40% av oppgavene, både med et matematisk og et programmeringsteknisk fokus. Det ble funnet at 91% av programmeringsoppgavene ga mulighet for arbeid med algoritmebehandling, mens samtlige av bøkene mangler oppgaver om abstrahering. Elevene får flest muligheter til å arbeide med det matematiske kunnskapsområdet *formler og variable størrelser.*

Sentrale begreper i studien er lærebok, matematikkundervisning, programmering, kreativ og imitativ resonnering, algoritmisk tenkning og læreplan.

Abstract

Programming has recently been implemented in the mathematics subject in norwegian curriculum. This master's thesis has examined the programming tasks in three printet textbooks in a norwegian 11th grade mathematics subject, "theoretical mathematics", matematics 1t. The purpose of this thesis is to assess the opportunities students get to engage with computational thinking, problem solving and mathematic, through the programming tasks in the printed textbooks. These three areas of study have been chosen because they are closely related to programming, in the new norwegian mathematics curriculum. To investigate the programming tasks, the following research questionshave been developed:

In the programming tasks in the printed textbooks in a 11th grade mathematics subject…

1. …what possibilities are given to engage with computational thinking?
2. … what possibilities are given for creative reasoning?
3. … what possibilities are given for working with mathematical content knowledge?

The master's study is a qualitative document analysis of the selected textbooks.

In order to collect data and answer the research questions, qualitative and quantitative methods have been used. As part of the work on research question 1, I have investigated which definition of computational thinking may have been used in the design of the Norwegian national curriculum.

The study's contribution is a framework developed to investigate programming in a Norwegian school context, in line with the intended curriculum for mathematics 1T. The findings show that students are given few opportunities to work with programming, but that the programming tasks that do exist provide opportunities for creative reasoning in approximately 40% of the tasks, with both a mathematical and a programming perspective. It was found that 91% of the programming tasks gave the opportunity to work with algorithmic thinking, while all three of the books lack tasks on abstraction. The students get the most opportunities to work with the mathematical knowledge area of formulas and variable quantities.

Key words: mathematics text book, programming, creative and imitative reasoning, computational thinking and curriculum.

Forord

```
print("Takk!!")
```

Denne masteroppgaven markerer slutten på tiden som student i fagdidaktikk, retning matematikk, ved NTNU i Trondheim - og for en tid det har vært!

Masteroppgaven min er et resultat av en prosess hvor jeg har mottatt så mye god hjelp og støtte underveis. Jeg er så takknemlig!

Jeg ønsker å rette en stor «takk for hjelpa» til min veileder Anders Sanne, for fleksibel veiledning og gode innspill, gjennom hele denne prosessen, og til Morten Munthe, for å ha tatt seg tid til faglige diskusjoner med meg. Takk til verdens beste kolleger ved Ringsaker videregående skole, for et utrolig raust samarbeid gjennom min tid som student. Takk til ledelsen ved skolen, som ville satse på meg, gjennom videreutdanningsprogrammet Kompetanse for Kvalitet, og spesielt til Grete Øksdahl, for gode diskusjoner rundt mulige retninger for oppgaven min underveis.

Oda, Ranveig, Vigdis og mamma har hjulpet meg i skriveprosessen, det setter jeg veldig stor pris på! Jeg må også rette en stor takk til Ingrids besteforeldre og tante Torunn, for barnevakttjeneste, så jeg har kunnet skrive litt hjemme i mammapermisjonen min, og etter arbeidstid mens Andreas renoverer førsteetasjen.

Helt til sist må jeg få takke deg, kjæreste Andreas, for at du har passet barn, hund og hjem, så jeg har kunnet sitte foran PCen og skrive. Nå gleder jeg meg til å legge vekk denne oppgaven og nyte tiden som en familie på tre, før vi plutselig blir fire.

Jeg gikk gravid med vår første datter, Ingrid, når arbeidet med denne masteroppgaven ble startet, høsten 2022. Et år senere kan jeg levere inn oppgaven, med en liten lillesøster eller lillebror i magen.

For en tid!

Desember, 2023

Tuva Liisa Jacobsen Sevildal

Innhold

Figurer

Tabeller

1 Innledning

Programmering og algoritmisk tenkning har fått en sentral plass i matematikkfaget i norsk skole, gjennom innføring av læreplanen Kunnskapsløftet LK20 (Kaufmann & Stenseth, 2023, s. 119). Norge føyer seg dermed i rekken av vestlige land som innfører programmering og algoritmisk tenkning i undervisning i skolen (Jensen et al., 2023, s. 1; Sanne et al., 2016, s. 38). De fleste land som har innført programmering i skolen, begrunner det med at programmering kan bidra til økte evner til logisk tenkning og problemløsning (Balanskat & Engelhardt, 2015, s. 4), og at programmering og algoritmisk tenkning er viktige kunnskaper i det 21. århundret (Jensen et al., 2023, s. 1; Weintrop et al., 2016, s. 129). Tanken er at elever som behersker programmering kan bruke denne evnen til å utvikle seg innen matematikk (Weintrop et al., 2016, s. 139).

Det er i dag stor etterspørsel etter forskningsbasert kunnskap om programmering i skolen (Blikstein, 2018, s. 35; Forsström & Kaufmann, 2018, s. 28). Etterspørselen skyldes den pågående implementeringen av programmering i skolepensum i Vesten (Forsström & Kaufmann, 2018, s. 28). Denne masterstudien er et bidrag til å kartlegge og vurdere programmeringsinnholdet i lærebøker i faget matematikk 1T (MAT09-01). Studien vil analysere programmeringsoppgavene i et utvalg nylig utgitte, fysiske lærebøker i matematikk 1T (MAT09-01), etter innføring av læreplanen Kunnskapsløftet 2020.

1.1 Bakgrunn for valg av tema

Denne masteroppgaven undersøker hvordan programmering legges frem og forklares i nye trykte lærebøker i matematikk som svarer til den nye læreplanen Kunnskapsløftet LK20. Det er første gang lærebøkene i matematikk inkluderer programmeringsrelaterte oppgaver (Hammerø & Schmeding, 2023, s. 1). Fordi programmering ikke tradisjonelt har vært en del av læreplanen i skolen i Norge, er det behov for å evaluere undervisningsressurser som er egnet for å undervise programmering i skolen (Jensen et al., 2023, s. 8). Formålet med studien er å foreta en innholdsanalyse av programmeringsinnholdet i lærebøker i faget matematikk 1T (MAT09-01), for å undersøke hvilke muligher elevene har til å arbeide med algoritmisk tenkning, problemløsning og matematiske kunnskapsområder, i arbeidet med programmeringsoppgavene i bøkene.

Tradisjonelt sett har lærebokens posisjon vært sterk i matematikkfaget i Norge (Kongelf, 2019, s. 23). Lærebøkene er et viktig verktøy i både planlegging og gjennomføring av undervisning, særlig i matematikk (Johansson, 2006; Kongelf, 2019). I matematikkundervisningen fungerer læreboka som en bro mellom læreplanen og undervisningen i timene (Schmidt et al., 1997, s. 53). Undersøkelser rundt bruken av undervisningsressurser i norske klasserom viser at en stor del av norske lærere anser at de, ved å følge læreboken, «sikrer» at kompetansemålene i et fag blir dekket (Gilje et. al., 2016, s. 27). Sannheten er at læreboken representerer forfatternes fortolkning av læreplanen, med tilhørende kompetansemål (Gilje et al., 2016, s. 27). Fordi det ikke finnes noen nasjonal kvalitetssikring av lærebøker i Norge, er det opp til lærebokforfatterne å sammenfatte læringsressurser som skal dekke læreplanen (Askeland, 2023).

Læreplanen inneholder kompetansemål der programmering er nevnt eksplisitt (Utdanningsdirektoratet, 2019). Allikevel kommer det ikke klart frem i kompetansemålene hvilken rolle programmering er tiltenkt i faget (Kaufmann & Stenseth, 2023, s. 110). Læreplanen spesifiserer ikke når programmeringen skal brukes og overlater slik mange valg til den enkelte lærer (Munthe, 2022, s. 10).

Kaufmann og Stenseth (2023) mener vi ikke kan forvente at lærere med lav kompetanse i programmering skal designe økter som går ut over å lære programmering som en ferdighet (s. 119). Formålet med programmering i matematikkfaget i skolen beskrives som noe annet enn ren programmeringsinnlæring (Utdanningsdirektoratet, 2023). I læreplanen for matematikk 1T omtales programmering sammen med algoritmisk tenkning og problemløsning (Kunnskapsdepartementet, 2019, s. 5). I en artikkel av Utdanningsdirektoratet spesifiseres det at når programmeringen brukes til problemløsning, så vil den kunne bidra til økt forståelse for matematikk (Utdanningsdirektoratet, 2023).

Siden flere matematikklærere i dag underviser i programmering, uten å selv ha utdanning i emnet (Jensen et al., 2023; Kaufmann & Stenseth, 2023; Hammerø & Schmeding, 2023) er det grunn til å tro at matematikklærerne i stor grad baserer sin undervisning av programmering i matematikkfaget på innholdet i lærebøkene (Bråting & Kilhamn, 2022, s. 596). Derfor ønsker jeg å undersøke innholdet i de nye lærebøkene. Gir programmeringsinnholdet elevene anledning til å øke sin kompetanse i algoritmisk tenkning, problemløsning og matematikk, slik det uttrykkes i læreplanen og Utdanningsdirektoratets (2019) artikkel?

1.2 Forskningsspørsmål og avgrensing

I læreplanen for matematikk 1T (MAT09-01), omtales programmering i sammenheng med algoritmisk tenkning og problemløsning (Kunnskapsdepartementet, 2019, s. 5). I denne studien ønsker jeg å undersøke hvordan lærebøkene adresserer algoritmisk tenkning, problemløsning og matematiske kunnskapsområder, via programmeringsoppgavene sine. Disse områdene for undersøkelse er valgt, fordi programmering omtales i læreplanen i sammenheng med problemløsning og algoritmisk tenkning, og fordi Utdanningsdirektoratet uttrykket at når programmering brukes til problemløsning i matematikk, oppstår mulighetene til økt matematikklæring (Kunnskapsdepartementet, 2019b; Utdanningsdirektoratet, 2023).

Formålet med denne studien er å vurdere hvilke muligheter elevene får til å arbeide med algoritmisk tenkning, problemløsning og matematikk, gjennom programmeringsoppgavene i de trykte lærebøkene. På bakgrunn av sammenkoblingen i læreplanen mellom de nevnte tre områdene og programmering i matematikkfaget, er følgende tre forskningsspørsmål er utviklet:

1. Hvilke muligheter for algoritmisk tenkning finnes i programmeringsoppgavene i de trykte lærebøkene i matematikk 1T?
2. Hvilke muligheter for kreativ resonnering finnes i programmeringsoppgavene i lærebøkene i matematikk 1T?
3. Hvilke muligheter for arbeid med matematiske kunnskapsområder finnes i programmeringsoppgavene i lærebøkene for matematikk 1T?

For å undersøke og besvare forskningsspørsmåelene studerer jeg hvilke aspekter ved algoritmisk tenkning, hvilken type resonnering og hvilke matematiske kunnskapsområder som programmeringsoppgavene i tre trykte lærebøker i faget matematikk 1T gir muligheter til arbeid med. I tillegg studeres frekvensen av disse aspektene ved algoritmisk tenkning, resonneringen og de matematiske kunnskapsområdene. Ordet *mulighet* anvendes i denne masterstudien som et tilbud, som måles i både frekvens og innhold.

I arbeidet med å finne svar på forskningsspørsmål 1 er det også undersøkt hvilken mening som kan tenkes brukt om algoritmisk tenkning i Læreplanen for Matematikk (Utdanningsdirektoratet, 2019). Forskningsspørsmål 1 undersøkes så med utgangspunkt i det samme teoretiske rammeverket.

Det finnes tre trykte lærebøker tiltenkt faget matematikk 1T: Lærebøkene som vil analyseres er Sinus 1T av forlaget Cappelen Damm (Maus, G. & Smestad, B.-T., 2020), Matematikk 1T av Aschehoug (Frydenlund, C. & Holst, L., 2020) og Mønster 1T av Gyldendal (Raustøl, A. & Lund, H.-S., 2020). Alle disse lærebøkene ble gitt ut i 2020, klare for bruk i skolen samtidig som innføringen av Kunnskapsløftet LK20. Samtlige av bøkene analyseres i denne studien. Studien avgrenses til å omhandle programmeringsoppgavene i disse tre lærebøkene. Se kapittel 3 om metode for ytterligere spesifikasjon av hva som defineres som en oppgave i denne studien, med tilhørende eksempler fra lærebøkene.

For å unngå forveksling i navnene på matematikkfaget matematikk 1T (MAT09-01) og Aschehoug sin lærebok Matematikk 1T (Frydenlund, C. & Holst, L., 2020), anvendes fagkoden MAT09-01 når faget omtales i de neste kapitlene i denne studien.

1.3 Leserveiledning og disposisjon

I kapittel 2 vil jeg presentere teori og tidligere forskning innen programmering i matematikk, og lærebokanalyser av programmeringsinnhold i lærebøker i matematikk. Kapittel 3 gjennomgår oppgavens metode og analyseprosess. I kapittel 4 presenteres lærebokanalysens resultater. Disse diskuteres opp mot presentert teori i kapittel 5, hvor jeg også peker på denne studiens implikasjoner for praksis og videre forskning. Kapittel 6 er oppgavens avslutning.

2 Teori

2.1 Programmering

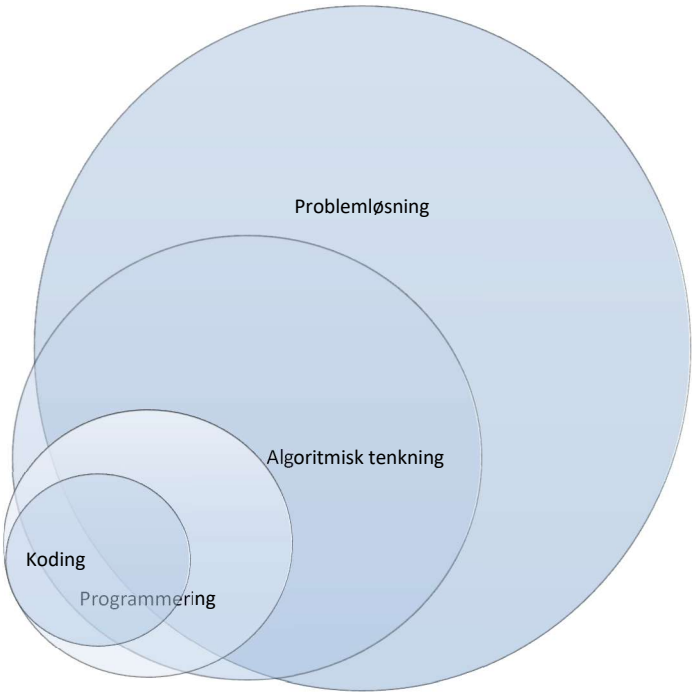
Begrepene koding, programmering og algoritmisk tenkning brukes noen ganger om hverandre (Boconi et al., 2019, s. 190). De skiller seg derimot fra hverandre, ved at koding består av det å skrive instruksjoner som en datamaskin skal utføre, i et bestemt programmeringsspråk (Balanskat & Engelhardt, 2015). Programmering er et videre begrep, som også involverer å analysere og utvikle fremgangsmåter for å løse et problem, ved bruk av datamaskiner (Lye & Koh, 2014; Bocconi et al., 2016). Andre inkluderer også analoge aktiviteter som en del av begrepet programmering (Sun et al., 2021). Avhengig av anvendt definisjon behøver dermed programmeringen ikke nødvendigvis å omhandle bruk av datamaskiner over hodet (Sun et al., 2021). Programmering uten bruk av datamaskin omtales som *unplugged programming* (Sun et al., 2021, s. 2). Slike aktiviteter uten datamaskin kan være å utforske programmeringskonsepter (Bell & Vahrenhold, 2018, s. 507). Instruksjoner om å tegne eller bygge noe av legoklosser kan gi innsikt om hvor nøyaktig man bør formulere seg i et program. Det å jobbe seg gjennom en kode, uten å kjøre den på datamaskinen, kan hjelpe elever med å forstå programmet de skal arbeide med (Bell & Vahrenhold, 2018, s. 507). Bell og Vahrenhold (2018) argumenterer for at elever bør gis muligheter til å arbeide med både unplugged programmering, og programmering på datamaskinen, såkalt plugged-in-programmering (s. 507).

Programmering ved bruk av en datamaskin kan foregå på en rekke programmeringsspråk (Drange, 2023). De tradisjonelle programmeringsspråkene er tekstbaserte, som Python og C++ (Drange, 2023). Det finnes også programmeringsspråk som består av visuelle komponenter, heller enn tekst (Drange, 2023). Disse språkene kalles blokkprogrammering (Drange, 2023). I blokkprogrammering bygger man programmet sitt ved å plassere kodeblokker etter hverandre.

Programmering, og særlig tekstbasert programmering, stiller strenge krav til syntaks (Vihovde, 2019). I blokkprogrammering er det enklere å unngå syntaksfeil (Drange, 2023). De visuelle blokkene er fargekodet, og blokkenes spesifikke former gjør at man ikke får satt sammen blokker som ikke følger syntaks (Drange, 2023). Jeg antar dette er årsaken til at blokkprogrammering er mye brukt for å lære barn å programmere (Drange, 2023). I tillegg kan konkreter som roboten Micro:bit eller Lego Mindstorms brukes sammen med blokkprogrammering (Drange, 2023). Syntaksfeil er blant de mest vanlige feilene elevene møter i arbeid med programmering, og en type feil som ikke bidrar til matematikklæring (Munthe, 2022).

Algoritmisk tenkning kan inkludere både programmering og koding, avhengig av hvilken definisjon av begrepet man anvender. Som vi skal se i kap. 3.2 er ikke begrepet algoritmisk tenkning entydig definert i forskningslitteraturen. I læreplanen tydeliggjøres det at algoritmisk tenkning er en problemløsningsstrategi (Utdanningsdirektoratet, 2019). Også dette utsagnet nyanseres noe i forskningslitteraturen, som det fremkommer i kap. 3.3. Allikevel kan vi se for oss en tenkt sammenheng mellom koding, programmering, algoritmisk tenkning og problemløsning, som jeg forsøker å presentere i figur 2.1. Koding, selve aktiviteten med å skrive programkode, inngår i det videre

begrepet programmering. Programmering kan inngå i algoritmisk tenkning, eller ikke, avhengig av hvilket teoretisk rammeverk man anvender (Elicer & Tamborg, 2022, s. 45). Algoritmisk tenkning omtales ofte som en problemløsningsstrategi (Selby & Woollard, 2013, s. 5).



Figur 2.1: Sammenhengen mellom problemløsning, algoritmisk tenkning, programmering og koding

2.1.1 Programmering og algoritmisk tenkning i skolen

Programmering i skolekontekst er ikke et nytt fenomen (Selby & Woollard, 2014, s. 2). Allerede på 1960-tallet laget Papert det første programmeringsspråket for barn, kalt Logo, med tanke på undervisning (Papert, 1980). Selv om begrepet algoritmisk tenkning var knyttet til feltet computer science, begynte en idé om at tankesettet kunne være nyttig for andre fagfelt å spire (Selby og Woollard, 2014, s. 2). På 1980-tallet ble programmering tilgjengelig i norsk skole, gjennom valgfaget elektronisk databehandling (Meld. St. 37 (1988-1989), s. 37). Mot slutten av 1980-tallet begynte man å anse algoritmisk tenkning som mer enn aktiviteter knyttet til bruk av maskiner, og algoritmisk tenkning linkes til problemløsning og en måte å tenke på (Selby og Woollard, 2014, s. 2). Papert introduserer programmering i skolen, med en konstruktivistisk idé om at programmeringen skal bidra til problemløsning og utforskning på tvers av fagfelt (Selby og Woollard, 2014, s. 2). I Paperts bok «Mindstorms» fra 1980, beskriver forfatteren at programmeringen skal bidra til å utvikle barns kognitive ferdigheter innen planlegging og problemløsning, og evnen til metakognisjon (Papert, 1980).

Til tross for stor suksess med boka «Mindstorms» i pedagogiske miljøer, møtte Papert også kritikk (Pea & Kurland, 1984). Enkelte hevder at overføringsverdien fra barns arbeid med programmeringsspråket Logo til læring i andre fag, er liten (Pea & Kurland, 1984).

Som vi vil se nærmere i kap. 2.4.3, har man i flere studier forsøkt å undersøke hvorvidt det finnes noen overføringsverdi mellom programmering og kognitive evner som er nyttige også i andre fagfelt, men har funnet at det er utfordrende å påvise noen slik sammenheng (Lye & Koh, 2014; Forsström & Kauffman, 2018). Denne første bølgen med programmering i skolen forsvinner, og programmering blir ikke innført som et fagfelt for alle (Munthe, 2022).

I dag er algoritmisk tenkning og programmering igjen aktuelle begreper i skolen (Forsström & Kaufmann, 2018). Blant årsakene finner vi en stor tilgang på digital teknologi, som ikke var til stede i samme grad på 1980-tallet (Voogt et al., 2015, s. 715; Lodi & Martini, 2021, s. 895), og nye uttalelser om programmeringens nytteverdi. Mest kjent er kanskje Wings utsagn om at algoritmisk tenkning burde læres i skolen, på linje med lesing, skriving og aritmetikk (Wing, 2006).

I England og Danmark har man valgt å innføre programmering i skolen ved å opprette nye fag; computation i England, og teknologifag i Danmark, (Jensen et al., 2023; Kilhamn og Bråting, 2022). Norge og Sverige, på sin side, har valgt å legge programmeringen som en del av matematikkfaget (Utdanningsdirektoratet, 2019; Bråting & Kilhamn, 2022). Begge deler synes å kunne medføre utfordringer. Sanne-utvalget peker på erfaringer med implementering av teknologifag i skolen i andre vestlige land, hvor det fremkommer at svak lærerkompetanse, manglende spesifisering av timeantall og uklare definisjoner om hva et slik fag i skolen skal inneholde, bidrar til at faget blir nedprioritert (Sanne et al., 2016).

I Sverige ble programmering nylig innført som en del av matematikkfaget (Bråting & Kilhamn, 2022). Innføringen skjedde med kort forvarsel, og uten at lærerne fikk et nasjonalt tilbud om relevant kompetanseheving på forhånd (Bråting & Kilhamn, 2022, s. 596). Tidligere har ikke programmering vært en del av lærerutdanningen i Sverige (Bråting & Kilhamn, 2022, s. 596), og svenske matematikklærere underviser derfor i dag elevene sine i programmering, uten å selv nødvendigvis være trygge på feltet (Kilhamn et al., 2021; Misfeldt et al., 2019; Mozelius et al., 2019). Dagens situasjon i Norge kan sammenliknes med den svenske: De nye læreplanene undervises i matematikk-klasserom av lærere med manglende kunnskap om programmering (Jensen et al., 2023, s. 1).

Til tross for de mulige utfordringene ved å opprette et eget teknologi-fag anbefalte Sanne-utvalget innføring av et slikt fag også i Norge, heller enn at programmering og algoritmisk tenkning implementeres i allerede-eksisterende fag (Sanne et al., 2016). Som vi skal se nærmere i neste delkapittel ble ikke denne anbefalingen fulgt. Med innføring av LK20 ble undervisning i programmering lagt til fagene kunst og håndverk, naturfag, musikk og matematikk (Kunnskapsdepartementet, 2019a; Kunnskapsdepartementet, 2019d; Kunnskapsdepartementet, 2019c; Kunnskapsdepartementet, 2019b).

2.1.2 Programmering i matematikk 1T

Matematikkfaget fikk hovedansvaret for programmeringsopplæringen i LK20, i strid med anbefalinger fra ekspertgruppa Sanne-utvalget (Sanne et al., 2016) Dette førte til debatt rundt hva som var bakgrunnen for, og intensjonen med, implementeringen av programmering i matematikkfaget i skolen (Dahl et al., 2017). Jeg vil ikke forsøke å finne noen nærmere forklaring på dette i denne studien, men sammenfatter hensikten

med programmering i matematikkfaget med læreplanens, og Utdanningsdirektoratets, ord:

Læreplanen i matematikk 1T (MAT09-01) inneholder ett kompetansemål som omhandler programmering: «Mål for opplæringen er at eleven skal kunne formulere og løse problemer ved hjelp av algoritmisk tenkning, ulike problemløsingsstrategier, digitale verktøy og programmering» (Kunnskapsdepartementet, 2019b).

Kompetansemålene i Kunnskapsløftet 2020 skal leses og forstås i lys av kjerneelementene og overordnet del (Kunnskapsdepartementet, 2019b). I overordnet del omtales algoritmisk tenkning og programmering i kjerneelementet «Utforsking og problemløysing»:

> Utforsking i matematikk T handlar om at elevane leiter etter mønster, finn samanhengar og diskuterer seg fram til ei felles forståing. Elevane skal leggje meir vekt på strategiane og framgangsmåtane enn på løysingane. Problemløysing i matematikk T handlar om at elevane utviklar ein metode for å løyse eit problem dei ikkje kjenner frå før. Algoritmisk tenking er viktig i prosessen med å utvikle strategiar og framgangsmåtar for å løyse problem og inneber å bryte ned eit problem i delproblem som kan løysast systematisk. Vidare inneber det å vurdere om delproblema best kan løysast med eller utan digitale verktøy. Problemløysing handlar òg om å analysere og forme om kjende og ukjende problem, løyse dei og vurdere om løysingane er gyldige (Kunnskapsdepartementet, 2019b).

Programmering omtales også som en del av den grunnleggende ferdigheten «Digitale ferdigheter». Også der i forbindelse med utforskning og problemløsning: «Digitale ferdigheiter i matematikk T inneber å kunne bruke grafteiknar, rekneark, CAS, dynamisk geometriprogram og programmering til å utforske og løyse matematiske problem. Vidare inneber det å finne, analysere, behandle og presentere informasjon ved hjelp av digitale verktøy.» (Kunnskapsdepartementet, 2019b).

Programmering og algoritmisk tenkning knyttes altså sammen med å utforske. I Utdanningsdirektoratets verktøy «støtte til læreplanen» gis følgende forklaring av verbet «utforske»:

> Å utforske handler om å oppleve og eksperimentere og kan ivareta nysgjerrighet og undring. Å utforske kan bety å sanse, søke, oppdage, observere og granske. I noen tilfeller betyr det å undersøke ulike sider av en sak gjennom åpen og kritisk drøfting. Å utforske kan også bety å teste eller prøve ut og evaluere arbeidsmetoder, produkter eller utstyr. (Utdanningsdirektoratet, u.å).

Det siste stedet i læreplanen for MAT09-01 som programmering nevnes, er i teksten om underveisvurdering:

> […] Læraren skal vere i dialog med elevane om utviklinga deira i programmering og strategiar for å løyse problem. Elevane skal få høve til å prøve og feile. Med utgangspunkt i kompetansen elevane viser, skal dei få høve til å setje ord på kva dei opplever at dei får til, og reflektere over si eiga faglege utvikling. Læraren skal gi rettleiing om vidare læring og tilpasse opplæringa slik at elevane kan bruke rettleiinga for å utvikle kompetansen sin i å sjå samanhengar mellom matematikk og teoretiske anvendingar. (Kunnskapsdepartementet, 2019b).

Programmering og algoritmisk tenkning i læreplanen knyttes sammen med ordene utforsking og problemløsning. Å utforske er, ifølge Utdanningsdirektoratet, å

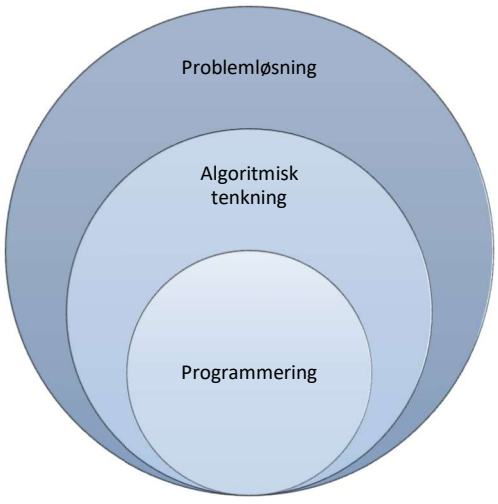
eksperimentere, søke, oppdage, observere, granske, teste, prøve ut og evaluere, mens problemløsning handler om å utvikle metoder for å løse matematiske problemer de ikke kjenner fra før, ved å omformulere matematiske problemer og/eller bryte ned problemene i delproblemer, jobbe systematisk, løse problemene, og vurdere om løsningene er gyldige.

Utdanningsdirektoratet har også publisert en tekst hvor de forklarer hva som er nytt i læreplanen om matematikk. I artikkelen «Kva er nytt i matematikk?» (fra 1-10. trinn, i tillegg til 1P, 1P-Y, 1T og 2P), tydeliggjør de at algoritmisk tenkning er en viktig problemløsningsstrategi, og skriver at «Når elevene bruker programmering til å utforske og løse problemer, kan det være et godt verktøy for å utvikle matematisk forståelse» (Utdanningsdirektoratet, 2023).

Under artikkelen ligger en video med samme navn, hvor Utdanningsdirektoratet presiserer at elevene trenger å trene på å jobbe med åpne oppgaver. I tillegg uttrykkes følgende:

> […] det å vurdere (arbeidet man har gjort) må gjennomsyre hele utdanningsløpet […]
>
> Algoritmisk tenkning er en problemløsningsstrategi som er tydeliggjort i læreplanen. Problemløsningen kan skje med kropp, konkreter, tegning og digitale verktøy. Programmering er lagt inn i læreplanen på en måte som kan gi rom for kreative måter å løse problemer på. […]
>
> Elevene skal utforske hvordan algoritmer kan lages, testes og forbedres, og de skal bruke programmering til å utforske matematiske egenskaper og sammenhenger. (Utdanningsdirektoratet, 2023).

Begrepene fra figur 2.1 har følgende sammenheng med hverandre, slik begrepen brukes i læreplanen: Programmering er den del av algoritmisk tenkning, som igjen er en problemløsningsstrategi. Se figur 2.2 for en modifisert versjon av figur 2.1, for å vise sammenhengen mellom begrepene, slik jeg tolker dem i læreplanen.



Figur 2.2: Min tolkning av sammenhengene mellom programmering, algoritmisk tenkning og problemløsning, i læreplanen for matematikk (Kunnskapsdepartementet, 2019b).

Oppsummert uttrykker læreplanen i matematikk MAT09-01 og de nevnte artiklene hos Utdanningsdirektoratet, at hensikten med programmering i matematikkfaget i skolen generelt, og i MAT09-01 spesielt, er å bidra til at elevene utvikler evne til algoritmisk tenkning og problemløsning. I læreplanen tydeliggjøres det også at når programmeringen benyttes til problemløsning, så bidrar det til matematikklæring. Som jeg vil vise i denne oppgaven, er ikke algoritmisk tenkning entydig definert hverken i forskningslitteraturen eller i Kunnskapsløftet LK20 og norske styringsdokumenter.

2.2 Algoritmisk tenkning

Før jeg velger en definisjon av algoritmisk tenkning som jeg vil anvende som teoretisk rammeverk i min lærebokanalyse, ønsker jeg å undersøke hvilken tolkning av algoritmisk tenkning som kan være tatt i bruk i læreplanen for MAT09-01. For å gjøre dette vil jeg presentere ulike definisjoner av algoritmisk tenkning i forskningslitteraturen. Deretter vil jeg redegjøre for anvendte definisjoner av algoritmisk tenkning i tekster av Utdanningsdirektoratet, norske styringsdokumenter, og deres kilder, i et forsøk på å finne sammenfallende definisjoner. I utarbeidelsen av denne studien har denne prosessen ikke vært lineær, slik det kan fremstå i teorikapittelet. Flere runder med innsamling av teori fra forskningslitteraturen og søken etter, og sammenlikning med, Utdanningsdirektoratets kilder er blitt foretatt.

2.2.1 Ulike definisjoner

Bruken av begrepet algoritmisk tenkning er ikke entydig. Boconi et al. (2018) undersøkte hvilke begreper som ble anvendt for å beskrive algoritmisk tenkning i skolen i Europa (s. 23). De fant at begreper som pensée informatique (informatisk tenkning), algorithmic thinking, computational thinking, koding, technological literacy, programming, language of technology, informatics og den svenske «datalogisk tenkende», brukes om hverandre, for å beskrive algoritmisk tenkning (Boconi et al., 2018, s. 23).

I forskningslitteraturen finnes det heller ikke konsensus rundt begrepet algoritmisk tenkning. Shute et al. (2017) definerer algoritmisk tenkning som et fundament av konsepter som behøves for å løse problemer på en effektiv måte, med eller uten datamaskiner, med løsninger som er gjenbrukbare i andre kontekster (s. 142). Andre argumenterer for at algoritmisk tenkning heller burde defineres som en oversikt over handlingene som den algoritmiske tenkningen består av (Elicer & Tamborg, 2022, s. 45). Som vist i figur 2.1 er det ikke en gang enighet i hvorvidt programmering bør inngå i det teoretiske rammeverk for algoritmisk tenkning (Elicer & Tamborg, 2022, s. 45). Mange av definisjonene av algoritmisk tenkning er bygget opp av sett med praksiser, sammenflettet med måter å se på problemer på (Selby og Woollard, 2013; Weintrop et al., 2016).

Selby og Woolland (2013) gjennomgikk forskningslitteraturens definisjoner av algoritmisk tenkning fra 2006 frem til 2013 (s. 1). De kom frem til en definisjon av begrepet algoritmisk tenkning ved å inkludere aspekter ved algoritmisk tenkning som hadde konsensus i forskningslitteraturen, og forkaste øvrige idéer (Selby & Woollard, 2013). Deres definisjon ble slik: Algoritmisk tenkning er en tankeprosess som involverer logisk resonnering og som består av abstrahering, dekomponering, algoritmebehandling, evaluering og generalisering (Selby & Woollard, 2013). Selby og Woollards (2013) har den anvendte definisjonen av algoritmisk tenkning i engelsk skolesetting, og algoritmisk tenkning (og programmering) inngår i teknologifaget i England (Csizmadia et al., 2015).

Woollard har senere deltatt i arbeidet til Csizmadia et al. (2015). Der beskriver de kategoriene som utgjør algoritmisk tenkning (Csizmadia et al., 2015). I tillegg

operasjonaliserer de kategoriene på en slik måte at læreren kan kjenne igjen algoritmisk tenkning som elevaktiviteter som kan observeres i klasserommet (Csizmadia et al., 2015). For eksempel kan det dreie seg om algoritmebehandling når en elev skriver instruksjoner som gjentar seg, og om generalisering når en elev tilpasser deler av løsninger slik at de gjelder for en hel klasse med lignende problemer (s. 14). Operasjonaliseringen presenteres til læreren i form av en mer utfyllende forklaring av de ulike kategoriene, samt en detaljert liste med elevaktiviteter som kan tilhøre det enkelte aspektet ved algoritmisk tenkning (Csizmadia et al., 2015). En forenklet versjon av denne listen gjengis under:

Abstrahering: Abstrahering er prosessen med å gjøre en gjenstand mer forståelig, ved å redusere unødvendige detaljer. Abstrahering handler om å skjule de rette detaljene, uten å miste viktig informasjon. (Csizmadia et al., 2015, s. 15).

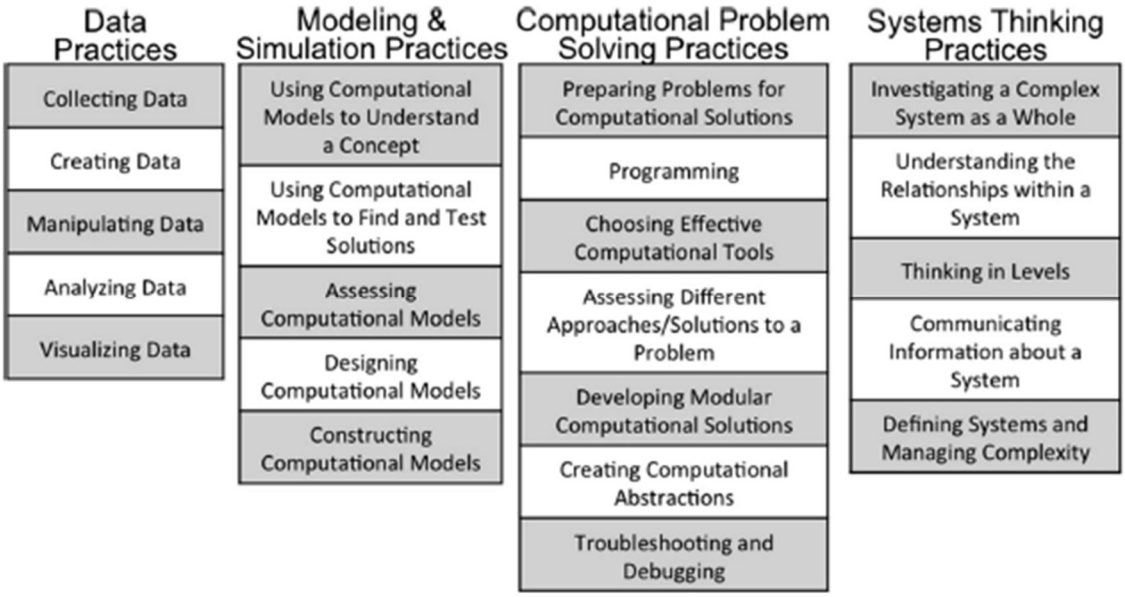
Dekomponering: Dekomponering er en måte å tenke på gjenstander på, i form av deres komponenter. Komponentene kan deretter forstås, løses, utvikles og evalueres separat. (Csizmadia et al., 2015, s. 14).

Algoritmebehandling: Algoritmebehandling er evnen til å tenke i sekvenser og regler som en måte å løse problemer på. Både det å lage algoritmer, formulere instruksjoner og å bruke passende notasjoner for å skrive kode inngår som eksempler på algoritmebehandling. (Csizmadia et al., 2015, s. 14).

Evaluering: Evaluering er prosessen for å sikre at en løsning er god, og egnet til formålet. Evaluering i algoritmisk tenkning er spesifikk, og detaljfokusert, knyttet til feilsøking. Evaluering kan også være en stegvis vurdering av en algoritme eller kode, for å finne ut hva den gjør. (Csizmadia et al., 2015, s. 15).

Generalisering: Generalisering er en måte å løse nye problemer på, basert på tidligere problemløsninger. Det innebærer å identifisere og utnytte mønstre, og tilpasse løsninger eller deler av løsninger, slik at de gjelder en hel klasse med liknende problemer, eller å overføre idéer og løsninger fra ett type problem til et annet. (Csizmadia et al., 2015, s. 14).

Mens Selby og Woollard (2013) har utviklet et generelt rammeverk for algoritmisk tenkning i skolen, har andre laget definisjoner av algoritmisk tenkning knyttet til spesielle fagområder, som for eksempel matematikk og realfag (Weintrop et al., 2016). Weintrop et al. (2016) har utarbeidet en taksonomi over praksiser som er knyttet til algoritmisk tenkning i realfagklasserom (s. 135). De kom frem til sitt rammeverk gjennom en større litteraturstudie, en analyse av undervisningsaktiviteter i matematikk og andre realfag, i tillegg til intervjuer med realister som biokjemikere, fysikere og dataingeniører (Weintrop et al., 2016, s. 134). De forsøkte å samle helheten av arbeidsmåter som inngår i algoritmisk tenkning i realfagene og matematikk, for å kunne skape en taksonomi og definisjon av algoritmisk tenkning i matematikk og realfag som man kunne samles og enes om (Weintrop et al., 2016). Taksonomien er inndelt i de fire kategoriene datapraksiser, modellering og simulering, problemløsning med algoritmisk tenkning og praksiser for systemtenking (Weintrop et al., 2016). Hver kategori er igjen delt inn i praksiser for algoritmisk tenkning, og forklart gjennom en beskrivende tekst, for å være konkret og operasjonaliserbar for effektiv implementering i klasserommet (Weintrop et al., 2016, s. 129).



Tabell 2.1: Weintrop et al. (2016) sin taksonomi over algoritmisk tenkning (Weintrop et al., 2016, s. 135).

Hurt et al. (2023) er kritiske til den typen teoretiske rammeverk som kategoriserer aktiviteter som kan føre til algoritmisk tenkning (s. 3). Selv om disse rammeverkene bidrar til å tydeliggjøre aktiviteter hvor algoritmisk tenkning kan oppstå, sier de ikke noe om hvordan disse aktivitetene skal bidra til den kognitive prosessen som algoritmisk tenkning er (Hurt et al., 2023, s. 3). Med andre ord stiller de spørsmål ved hvordan man kan vite elevene bedriver algoritmisk tenkning eller kun for eksempel følger en fremgangsmåte innenfor en gitt aktivitet (Hurt et al., 2023, s. 3). For å kunne si noe om dette trengs en modell over de kognitive prosessene elevene utfører i arbeidet med aktivitetene som tilknyttes algoritmisk tenkning (Weintrop et al., 2021). Hurt et al. (2023) tilbyr en slik modell over algoritmisk tenkning i realfagsundervisningen, ved å bygge videre på Weintrop et al. (2016) sin taksonomi.

Voogt et al. (2015) sin definisjon av algoritmisk tenkning er mindre kategorisk enn de overnevnte. De presenterer en bredere forståelse av algoritmisk tenkning, uten bruk av absolutte kriterier. Deres definisjon er mer flytende. De henviser til eksempler av momenter som kan inngå i algoritmisk tenkning. På denne måten kan man se enkelte likheter mellom definisjonen til Voogt et al. (2015) og Hurt et al. (2023), til tross for forskjeller i hvor rigide rammer forskerne har brukt til å definere algoritmisk tenkning i artiklene sine. Begge forskerteamene understreker at det ikke er aktiviteten i seg selv som avgjør hvorvidt elevene deltar i algoritmisk tenkning eller ei. Voogt et al. (2015) ønsker å se etter likheter i diskusjonene rundt algoritmisk tenkning, det være seg innen matematikk, naturfag eller kunst og håndverk (s. 723). Ved å se etter disse sammenhengene vil man få en mer konsis forståelse av hva som er det viktige innen algoritmisk tenkning, fremfor de perifere aktivitetene, og hvordan man skal integrere algoritmisk tenkning i en grunnskolesetting (Voogt et al., 2015).

Flere studier har i senere år forsøkt å syntetisere eksisterende litteratur om algoritmisk tenkning (Brennan & Resnick, 2012; Grover & Pea, 2014; Weintrop et al., 2016). Resultatet har vært varierende, og det gjenstår fremdeles en manglende konsensus om hvordan algoritmisk tenkning skal forstås og skilles fra andre tankeprosesser (Hurt et al., 2023; Shute et al., 2017, s. 145).

2.2.2 Computational thinking på norsk

Gjøvik og Torkildsen (2019) påpeker at det er utfordrende å oversette engelske «computational thinking» til et norsk, tilsvarende begrep (s. 32). Mange definisjoner av computational thinking inneholder «algorithmic thinking» (Bocconi & Chiccariello, 2018; Selby & Woollard, 2013). Dette kan skape forvirring, da uttrykket likner på den mye brukte norske oversettelsen av computational thinking: algoritmisk tenkning. Gjøvik og Torkildsen (2019) tok for seg denne problematikken i en artikkel i tidsskriftet Tangenten, hvor de forsøkte å nøste opp i mulige tvetydigheter i begrepsbruken innen algoritmisk tenkning, og overgangen fra engelske begreper til norske (s. 33). Artikkelen inkluderer et begrepsapparat som kan anvendes når begrepene omkring algoritmisk tenkning diskuteres på norsk. De oversetter begrepene slik:

Engelsk begrep	Norsk oversettelse
Computational thinking	Algoritmisk tenkning
Automation	Automatisering
Abstraction	Abstrahering
Algorithmic thinking	Algoritmebehandling
Generalization	Generalisering
Decomposition	Dekomponering

Tabell 2.2: Oversettelse av begreper knyttet til algoritmisk tenkning (Gjøvik og Torkildsen, 2019, s. 33)

Det er disse oversettelsene jeg bruker i denne studien. I tillegg bruker jeg begrepet algoritmisk resonnering, som en del av studiens undersøkelser av kreativ og imitativ resonnering (Lithner, 2008). Begrepene algoritmisk tenkning, algoritmebehandling og algoritmisk resonnering må ikke sammenblandes. Når algoritmisk resonnering omtales, er det med tanke på type resonnering knyttet til problemløsning.

Begrepene algoritmisk tenkning, algoritmebehandling og algoritmisk resonnering er allikevel ikke uten relasjon: Som vi vil se i kap. 2.3.2 er algoritmisk resonnering en form for imitativ resonnering, ved hjelp av kjente løsningsmetoder, eller algoritmer (Lithner, 2008). Dermed kan vi si at algoritmisk resonnering er en undergruppe av algoritmebehandling (Gjøvik & Torkildsen, 2019, s. 33), som igjen er en del av algoritmisk tenkning (Selby & Woollard, 2013, s. 5).

2.2.3 Algoritmisk tenkning i læreplanen for matematikk

Læreplanen for matematikk 1T og læreplanen for matematikk i grunnskolen, inneholder ingen kildehenvisninger som kan forklare hvordan algoritmisk tenkning skal forstås i læreplanen. Her vil jeg undersøke uttalelser om programmering og algoritmisk tenkning i styringsdokumentene som kom forut for innføringen av ny læreplan, for å forsøke å finne den teoretiske definisjonen av algoritmisk tenkning Utdanningsdirektoratet kan ha benyttet seg av i utarbeidelsen av læreplanen. Målet med denne delen av teksten er å finne en tolkning av læreplanens bruk av begrepet algoritmisk tenkning som ligger så tett opp mot begrepet slik det anvendes av læreplanen og læreplanens kilder som mulig. Det er også viktig at forskningen min er etterprøvbar. Derfor vier jeg denne delen av teksten så stor plass som jeg gjør, for å gjøre prosessen med å finne det teoretiske rammeverket for algoritmisk tenkning gjennomsiktig.

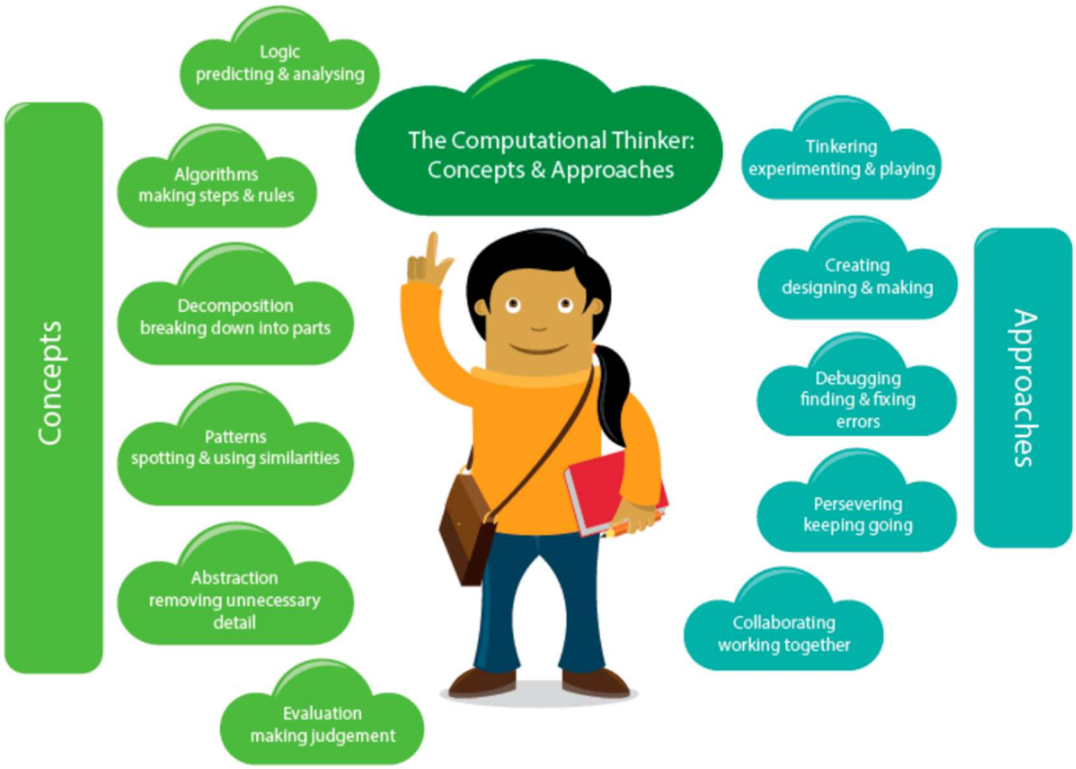
Utdanningsdirektoratet har publisert en artikkel med tittelen «Algoritmisk tenkning», hvor de forklarer at det finnes flere ulike definisjoner av algoritmisk tenkning, men at hovedtrekkene er sammenfallende (Utdanningsdirektoratet, 2019).

Sammen med teksten publiserer Utdanningsdirektoratet følgende figur:

Figur 2.3: Utdanningsdirektoratets "Den algoritmiske tenkeren» (Utdanningsdirektoratet, 2019).

Figuren følges av en kildehenvisning til Barefoot Computing.

Barefoot Computing er en nettside med læringsressurser til bruk i undervisning i engelsk skolekontekst. Et litteratursøk etter Barefoot Computing i databasen Eric brakte meg til artikkelen til Csizmadia et al. (2015), hvor denne figuren fremkommer:



Figur 2.4: "The Computational Thinker: Concepts & Approaches" (Csizmadia et al., 2015, s. 8)

Likhetene mellom figuren i Csizmadia et al. (2015) og Utdanningsdirektoratets «Den algoritmiske tenkeren» overbeviser meg om at det er Csizmadia et al. (2015) som er Utdanningsdirektoratets kilde for utarbeidelsen av «Den algoritmiske tenkeren». I tillegg til at figurene likner hverandre, rent visuelt, er nøkkelbegrepene og arbeidsmåtene i Utdanningsdirektoratets «den algoritmiske tenkeren» oversettelser av Barefoot Computings «The Computational Thinker: Concepts and approaches». Ved bruk av snøballmetoden som metode for litteratursøk fant jeg Selby og Woollards (2013) litteraturstudie i litteraturlisten til Csizmadia et al. (2015).

Kanskje er det Selby og Woollards (2013) definisjon av algoritmisk tenkning som er anvendt hos Utdanningsdirektoratet? Jeg ønsker å lete videre etter en tydelig definert anvendelse av begrepet algoritmisk tenkning i læreplanen. I de neste avsnittene vil jeg undersøke uttalelser om programmering og algoritmisk tenkning i styringsdokumentene som ble publisert i tiden forut for innføringen av Kunnskapsløftet 2020.

Norges Offentlige Utgreiinger har i flere omganger omtalt behovet for økt digital kompetanse i norsk skole (NOU 2013: 2; NOU 2015: 8; NOU 2020: 2). Selv om digitale ferdigheter har vært en av de grunnleggende ferdighetene i norsk læreplan siden innføringen av Kunnskapsløftet LK06, har det vært for stort fokus på å bruke digital teknologi i skolen, fremfor å utvikle og forstå den digitale teknologien (NOU 2013: 2, s. 99). Digitutvalget konkluderer i 2013 med at innsatsen for å lære elever digitale ferdigheter må endres og styrkes (2013: 2, s. 1). De foreslår å innføre programmering som valgfag i ungdomsskolen, og mener nettopp den manglende oppmerksomheten på grunnleggende programmeringsferdigheter er årsaken til at barn og unge vokser opp som konsumenter av digital teknologi (NOU 2013: 2, s. 10). Læreplanens overordnede definisjon av digitale ferdigheter i LK06 omtales som god, men kritiseres for å snevres for

mye inn i forklaringen av de digitale ferdighetsområdene (NOU 2013: 2, s. 101). Kritikken går ut på at fokuset med digitale ferdigheter blir på kommunikasjon, fremfor utvikling, programmering, beregning, analysering og teknisk forståelse (NOU 2013: 2, s. 101).

I 2013-2015 arbeider Ludvigsen-utvalget med å vurdere grunnopplæringens fag opp mot fremtidens krav til kompetanse i samfunn og arbeidsliv (NOU 2015: 8, s. 3). Digital kompetanse trekkes frem som en «sentral del av fagområdene i skolen», og som relevant på tvers av fagområder, i tillegg til innad i det enkelte skolefag (NOU 2015:8, s. 26). Ludvigsen-utvalget presiserer at det vil variere mellom fagene hvilke verktøy som er relevante å bruke, og hva elevene skal kunne bruke kompetansen til (NOU 2015: 8, s. 26). Rapporten tydeliggjør behovet for digital kompetanse i skolen (NOU 2015: 8, s. 26), uten å foreslå detaljerte tiltak eller en konkret reform (Meld. St. 28 (2015-2016), s. 15). Programmering og algoritmisk tenkning er ikke nevnt i Ludvigsen-utvalgets rapport (NOU 2015: 8).

NOU-rapportene følges opp med melding til Stortinget 28 (2015/16), hvor det legges føringer for en fornyelse av Kunnskapsløftet (Meld. St. 28 (2015-2016)). Her nevnes programmering ved to anledninger: i en henvisning til Lysne-utvalgets ønske om et valgfag i programmering (NOU 2015: 13), og i en henvisning til at det er satt i gang forsøk med valgfag i programmering på ungdomstrinnet (Meld. St. 28 (2015-2016), s. 54). Lysne-utvalget peker på at programmering i skolen vil kunne bidra til problemløsning, utvikling av analytisk tenkning, samt fungere som et støtteverktøy i andre fag, ved å stimulere til kreativitet og gruppearbeid (NOU 2015: 13). Algoritmisk tenkning nevnes ikke (NOU 2015: 13).

I 2016 oppnevner Utdanningsdirektoratet Sanne-utvalget til å gjennomgå teknologi i grunnopplæringen (Sanne et al., 2016, s. 3). Deres mandat inkluderer å formidle et kunnskapsgrunnlag om hvilken kompetanse fremtidens elever skal ha i digitale ferdigheter, herunder programmering (Sanne et al., 2016, s. 3). I rapporten til Sanne-utvalget (2016) omtales algoritmisk tenkning som kjernekompetansen i digital teknologi (s. 14). Sanne et al. (2016) oppsummerer algoritmisk tenkning som «en kompetanse i å abstrahere, kunne gjennomgå informasjon systematisk, lære å tilegne seg og å forstå forskjellige representasjonsformer, modulisere (dekomponere) problemer og problemstillinger, resonnere i iterative og parallelle strukturer» (s. 38).

I NOU 2020: 2 dukker begrepet algoritmisk tenkning opp, for første gang i styringsdokumentene jeg har undersøkt. Begrepet defineres derimot ikke i rapporten. I NOU 2020: 2 henvises det til OECD 2019, når begrepet algoritmisk tenkning tas i bruk. OECD 2019 anvender i sin tur henvisninger til Boconi et al. (2016), Lye & Koh (2014), Voogt et al. (2015), Bell (2016) og Paniague & Istance (2018), når det kommer til algoritmisk tenkning.

Av disse kan jeg umiddelbart utelukke Bell (2016), i letingen etter min tolkning av Utdanningsdirektoratets anvendte definisjon av algoritmisk tenkning. Bell (2016) definerer ikke algoritmisk tenkning, men skriver om programmering og koding. Da gjenstår fire kandidater: Boconi et al. (2016), Lye & Koh (2014), Voogt et al. (2015), og Paniague og Istance (2018). Voogt et al. (2015) sin definisjon av algoritmisk tenkning ble presentert i kapittel 2.2.1. De øvrige definisjonene fra OECD 2019 vil gjengis her, for sammenlikning med læreplanens beskrivelse av algoritmisk tenkning. Det er tatt et bevisst valg om å henvise til sekundærkilder der Lye & Koh (2014) og Paniague og Istance (2018) viser videre til Brennan og Resnick (2012) og Berry (2014). Årsaken til

dette er at jeg ønsker å tydeliggjøre at jeg anvender de samme kildene som det henvises til i OECD 2019, selv om de ikke er originalkildene i disse definisjonene av algoritmisk tenkning. Jeg mener det er sannsynlig at OECD 2019 ville oppgitt originalkildene dersom definisjonene av algoritmisk tenkning som de refereres til hadde vært hentet tilbake i de originale tekstene til Brennan og Resnick (2012) og Berry (2014).

Boconi et al. (2016) definerer algoritmisk tenkning som sammensatt av ferdighetene abstraksjon, algoritmebehandling, generalisering, automatisering og dekomponering (s. 18). Lye & Kohs (2014) definisjon bygger på Brennan og Resnick (2012) sine tre dimensjoner computational concepts, computational practices og computational perspectives (Brennan & Resnick, 2012, s. 1). Konseptene, den første dimensjonen, er knyttet til programmering, som variabler og løkker (Lye & Koh, 2014, s. 53). Dimensjon to består av problemløsningspraksiser som oppstår i programmeringsprosessen, som det å teste og feilsøke, eller abstrahere ut essensiell informasjon (Lye & Koh, 2014, s. 53). Siste dimensjon består av elevenes forståelse av deres relasjon med andre mennesker og den teknologiske verden rundt dem (Lye & Koh, 2014, s. 53). Paniague og Istance beskriver algoritmisk tenkning som en prosess i to steg (Paniague & Istance, 2018, s. 102). Først ser man for seg måter å løse problemet på, og deretter jobber man med å få datamaskiner til å kunne løse problemet (Paniague & Istance, 2018, s. 102). Algoritmisk tenkning består av logisk resonnering, dekomponering, algoritmer, abstrahering og mønstre (Berry, 2014, referert i Paniague & Istance, s. 102).

2.2.4 Denne studiens rammeverk for algoritmisk tenkning

Som det fremkommer av kapittel 2.2.3, anvender kildene som Kunnskapsløftet 2020 bygger på, flere ulike forskningsbaserte definisjoner av algoritmisk tenkning. Blant disse finner vi definisjonene til Boconi et al. (2016), Selby og Woollard (2013) og Voogt et al. (2015). Mens de to første er av typen definisjoner som kategoriserer handlinger som utgjør den algoritmiske tenkningen skiller Voogt et al. (2015) sin definisjon seg fra disse, ved å ikke tydeliggjøre hva algoritmisk tenkning består av, rent spesifikt, men heller omfavne en bredere, mindre kategorisk forståelse av algoritmisk tenkning som et tankesett som kan anvendes i musikk og kunst og håndverk, like fullt som i matematikk (Voogt et al., 2015). Undersøkelsen av de anvendte definisjonene for algoritmisk tenkning i dokumentene som danner grunnlaget for læreplanen Kunnskapsløftet 2020, viser derfor flere kilder som utfyller hverandre, ved at de fremmer ulike aspekter ved algoritmisk tenkning.

I denne studien ønsker jeg å anvende samme definisjon av algoritmisk tenkning som læreplanen. Siden læreplanen ikke inkluderer kildehenvisninger, og Utdanningsdirektoratets tekster om algoritmisk tenkning også mangler forskningsbaserte referanser, ser jeg meg nødt til å gjennomføre en tolkning av denne anvendelsen. Jeg baserer min tolkning på dokumentene som er presentert i kapittel 2.2.3.

Som vist i kap. 2.2.3, har Utdanningsdirektoratet publisert en artikkel på sine nettsider hvor de spesifiserer hvordan algoritmisk tenkning skal forstås i den norske læreplanen (Utdanningsdirektoratet, 2019). Denne artikkelen bygges rundt en figur som er en oversettelse av den engelske «The computational thinker, concepts and approaches" (Csizmadia et al., 2015, s. 8). Den engelske figuren figurerer i Csizmadia et al. (2015) sin artikkel «Computational thinking – a guide for teachers», hvor Selby og Woollard er medforfattere. Artikkelen til Csizmadia et al. (2015) bygger igjen på Selby og Woollards (2013) definisjon av algoritmisk tenkning.

Jeg har ikke gjort mer overbevisende funn i gjennomgangen av læreplanen, styringsdokumentene og deres kilder, i leting etter læreplanens definisjon av algoritmisk tenkning. Derfor mener jeg det er plausibelt å tolke læreplanens anvendelse av begrepet algoritmisk tenkning i samme retning som Selby og Woollards (2013) definisjon. Algoritmisk tenkning i læreplanen for matematikk 1T, og i de øvrige matematikkfagene i norsk grunnskole og videregående skole, tolkes til å bety at algoritmisk tenkning er en tankeprosess som involverer logisk resonnering, og som består av abstrahering, dekomposisjon, algoritmebehandling, evaluering og generalisering (Selby & Woollard, 2013). Disse kategoriene operasjonaliseres i Csizmadia et al. (2015), se kap. 2.2.1.

Jeg vil videre i denne studien anvende Selby og Woollard (2013) sin definisjon når jeg anvender begrepet algoritmisk tenkning, med beskrivelser og operasjonaliseringer gitt av Csizmadia et al. (2015).

2.3 Problemløsning

En matematikkoppgave kan være en rutineoppgave, eller et problem (Schoenfeld, 1992). Her vil disse to typene oppgaver beskrives og defineres. Deretter presenteres Lithners (2008) begreper kreativ og imitativ resonnering og hvordan disse knyttes til problemløsning.

2.3.1 Matematisk problem eller rutineoppgave?

Begrepene problem og problemløsning, brukes på ulike måter, i skolematematikken og i hverdagslivet for øvrig (Lithner, 2005; Schoenfeld, 1992). Problemløsning har blitt anvendt om alt fra å løse rutineoppgaver til å utføre matematikk som en profesjonell matematiker (Schoenfeld, 1992, s. 334).

Webster (1979) illustrerer motsigelsene som finnes i ulike definisjoner av begrepet problem, ved å presentere følgende to definisjoner:

1: "In mathematics, anything required to be done, or requiring the doing of something."

2: "A question … that is perplexing or difficult."

(Webster, 1979, s. 1434)

Forskningslitteraturen beskriver at problemer tradisjonelt sett har blitt anvendt i matematikkundervisningen om en hvilken som helst matematisk oppgave som skal utføres, lik Websters første definisjon (Schoenfeld, 1992, s. 337). Schoenfeld (1992) peker på at mange av disse oppgavene er langt fra Websters andre definisjon på problemer, da de har som formål å trene elevene i en spesifikk matematisk teknikk eller fremgangsmåte, som typisk sett nettopp er blitt modellert for eleven (s. 337). Björkqvist har funnet at ordet problem i skolematematikken typisk har blitt brukt om tekstoppgave (Björkqvist, 2003). En kan spørre seg om denne problematikken er knyttet til språk, og at man på svensk og engelsk anvender ordet «problem» om både rutineoppgaver og matematikkproblemer, der man på norsk ville brukt ordet «oppgave» om disse begge. Allikevel er det hensiktsmessig å tydelig avklare hva som menes med ordet problem i min studie, da ordet anvendes både i dagligtale og i matematikkundervisningen.

Som i matematikkundervisningen, finnes det også ulike definisjoner på et problem, i forskning på matematikkdidaktikk (Björkqvist, 2003; Schoenfeld, 1992; Boesen, 2006). I dag er det vanlig å definere et problem som en matematisk oppgave som skal utføres,

men hvor det innledningsvis i tillegg er uklart for problemløseren hvilke metoder som kan løse oppgaven (Björkqvist, 2003, s. 54). Schoenfeld (1992) definerer matematiske problemer som en oppgave som oppfyller to kriterier: Det er en oppgave som eleven er interessert og engasjert i, og som eleven ønsker å finne en løsning på (s. 71). I tillegg er oppgaven slik at eleven ikke allerede har en ferdig matematisk fremgangsmåte som kan anvendes for å finne løsningen (Schoenfeld, 1992, s. 71). Boesen (2006) definerer et problem som en oppgave hvor man ikke vet hvordan man skal gå frem for å løse den, og hvor det ikke foreligger noen kjent fremgangsmåte som kan anvendes for å nå løsningen (s. 31).

Polya betrakter problemløsning som en ferdighet, og hevder at denne ferdigheten utvikles via imitasjon og praksis (Polya, 2004). Polya definerer det å arbeide med problemer som en «bevisst søken etter en handling som er passende for å oppnå et klart formulert mål, som ikke er umiddelbart oppnåelig» (Polya, 1981, s. 117, egen oversettelse). Den som er god på problemløsning evner å konsentrere seg om problemet, og ønsker inderlig å finne løsningen (Polya, 2004, s. 207).

Definisjonene av matematikkproblemer skiller seg fra hverandre i måten de presenterer forholdet mellom problemet og den som skal løse det. Mens Polya og Schoenfeld beskriver problemløserens ønske om å løse problemet (Polya, 2004, s. 207; Schoenfeld, 1992, s. 71), nevnes ikke dette av Boesen (2006). Schoenfeld går videre og inkluderer interesse og engasjement om problemet fra problemløserens side som regelrette kriterier for at matematikkoppgaven kvalifiserer som et problem. Også Björkqvist (2003) fremhever det som et viktig tilleggsaspekt, hvorvidt problemløseren opplever problemet som «sitt eget», eller ei (Björkqvist, 2003, s. 55). Et eierskap og interesse for problemet har innvirkninger på motivasjonen til å gå løs på arbeidet med problemet.

Felles for definisjonene av et problem, er en forståelse av at løsningen ikke skal være umiddelbart åpenbar for problemløseren, og at en må arbeide med matematikkoppgaven uten en forutbestemt algoritme som vil føre dem til løsningen. Hvorvidt matematikkoppgaven er et problem eller ikke, avhenger altså av personen som skal løse oppgaven (Schoenfeld, 1992; Björkqvist, 2003). Det er ikke et problem for en elev, dersom eleven øyeblikkelig ser en algoritme som sannsynligvis vil føre til løsningen (Polya, 1981). Matematikkoppgaver som ikke er problemer, kalles øvelser (Schoenfeld, 1992; Solvang, 1992), eller rutineoppgaver (Solvang, 1992). I følge Schoenfeld (1992) kan de fleste oppgaver i lærebøker i matematikk løses ved å bruke algoritmer som presenteres i tilhørende kapittel (s. 338). Dermed er ikke disse problemer, men rutineoppgaver.

En utfordring med forskning på problemløsning, er å avgjøre hva som utgjør et problem. Definisjonen av et problem er individrelatert. Om oppgaven er et problem eller ikke, avhenger både av oppgaven og av eleven som skal løse den (Boesen, 2006, s. 3). I tillegg er det vanskelig å avgjøre om problemløsning er anvendt, fordi en elevs tankemåte og prosess med å løse oppgaven kan være vanskelig å identifisere (Kilpatrick, 1969, s. 526). At en elev har funnet løsningen på problemet ikke er noen god indikator på hvorvidt problemløsningsprosesser er anvendt for å komme frem til løsningen (Kilpatrick, 1969, s. 526). Disse utfordringene får enda en dimensjon når studien er en lærebokanalyse, som her.

Uten detaljer om konteksten som oppgaven gis i, er det vanskelig å vurdere hvor kognitivt krevende en gitt matematikk oppgave er (Charalambous, 2010, s. 145). En av utfordringene med lærebokanalyse som metode, er nettopp at den ikke gir innblikk i

konteksten som oppgavene gis i, og elevene som arbeider med dem (Charalambous, 2010, s. 118). Når jeg skal undersøke oppgavene i lærebøkene, uten å være i kontakt med, eller observere, elevene som skal løse oppgavene, kan jeg ikke anvende definisjonene av problemer til hjelp i kategoriseringen av rutineoppgaver og matematiske problemer. Jeg behøver derfor et annet teoretisk rammeverk for å kunne vurdere hvorvidt oppgavene er problemer eller ei.

2.3.2 Kreativ og imitativ resonnering

Lithner har forsket på elevers strategivalg i arbeidet med problemløsning (Lithner, 2008). Han bruker begrepet resonnering om den tankerekken som brukes til å skape argumenter og komme frem til konklusjoner i arbeidet med problemløsning (Lithner, 2008, s. 257). Lithner skiller mellom imitativ og kreativ resonnering (Lithner, 2008, s. 256).

Kreativ resonnering er en form for resonnering som oppfyller følgende tre kriterier

1. Den er nyskapende for eleven.
2. Den er troverdig.
3. Den har et matematisk fundament.

(Lithner, 2008, s. 266)

Her oppstår tilsynelatende samme problematikk som tidligere nevnt: Hvordan kan jeg, i en lærebokanalyse, avgjøre hvorvidt en oppgave bidrar til resonnering som er nyskapende for eleven? Hva betegnes som nyskapende for en tenkt elev jeg ikke kjenner til? Dette vil bli tatt hensyn til i det teoretiske rammeverket om kreativ og imitativ resonnering.

Resonnering som ikke oppfyller kravene til kreativ resonnering er imitativ resonnering (Boesen, 2006, s. 20). Imitativ resonnering preges av strategier hvor elevene søker etter eksempler og fremgangsmåter som likner oppgaven de står ovenfor, og kopierer disse for å løse oppgaven (Boesen, 2006, s. 21). Eleven kan ha flaks og komme frem til rett svar ved bruk av imitativ resonnering, men kommer sjelden noen vei ved forsøk på bruk av imitativ resonnering i møte med matematiske problemer (Boesen, 2006, s. 4). Imitativ resonnering deles inn i memorert resonnering og algoritmisk resonnering (Boesen, 2006, s. 18). Den memorerte resonneringen består av å huske et svar, og anvende dette i oppgaven, mens den algoritmiske resonneringen går ut på å huske en fremgangsmåte for å løse oppgaven, og anvende denne i arbeide med å løse oppgaven eleven står ovenfor (Boesen, 2006, s. 18). Med imitativ resonnering elevene komme frem til konklusjoner i problemløsning ved å imitere, eller gjenbruke og kopiere (Lithner, 2008).

I motsetning til ved imiterende resonnering, så kjennetegnes kreativ resonnering ved at problemløseren skaper sin argumentasjonsrekke og trekker sine konklusjoner på egenhånd (Lithner, 2008). Å utføre kreativ resonnering vil danne denne studiens definisjon av problemløsning, lik Boesens definisjon (2006, s. 17).

Ved å anvende Lithners (2008) rammeverk over kreativ og imitativ resonnering, kan jeg operasjonalisere kategoriseringen av matematikkoppgaver som problemer eller rutineoppgaver. Dersom en tenkt elev kan løse en gitt oppgave i læreboken ved å

kopiere et eksempel i boken, kan oppgaven besvares ved hjelp av imitativ resonnering, og er dermed en rutineoppgave. Det samme gjelder dersom oppgaven som studeres følger en eller flere liknende oppgaver, hvor det holder at fremgangsmåten kopieres, for å kunne løse oppgaven. Dersom oppgaven ikke likner hverken noe trykt eksempel eller tidligere gitt oppgave i læreboken, kvalifiserer oppgaven som et matematisk problem. Unntaket er hvis det matematiske nivået på oppgaven er av et slikt nivå at eleven forventes å kunne løse denne oppgaven allerede før han eller hun påbegynte undervisning i faget matematikk 1T. Dette for å luke ut oppgaver hvor det er programmeringen som er det nye i oppgaven, og ikke matematikken selv som skaper et problem. Årsaken til at jeg kategoriserer oppgavene på denne måten er for å fremme læreplanens intensjon om at programmeringen skal være et ledd i problemløsning i matematikkfaget.

2.4 Tidligere forskning

Forskning på programmeringens tilknytning til matematikklæring, og programmeringsinnhold i trykte lærebøker i matematikkfaget, er i sin spede begynnelse (Forsström & Kaufmann, 2018). Her presenterer jeg tidligere forskning for å belyse hva vi allerede kjenner til om lærebokens rolle i matematikkfaget, programmeringsinnhold i trykte lærebøker i matematikk, og programmering for matematikklæring.

2.4.1 Lærebokens rolle i matematikkfaget

De trykte lærebøkene har en sentral rolle i norske klasserom (Gilje et al., 2016). I matematikkfaget spesielt har læreboken en sentral rolle når det kommer til planlegging og gjennomføring av undervisning (Kongelf, 2019). Lærere synes å benytte seg av lærebokens oppbygging av et fag og rekkefølgen den tilbyr (Pingel, 2010, s. 47). Læreboken muliggjør effektiv planlegging, som er tidsbesparende for læreren (Pingel, 2010, s. 47). Vanlig praksis for matematikkundervisningen synes å være en tavlegjennomgang, gitt av læreren, etterfulgt av at elevene arbeider med oppgaver i læreboken (Gilje et al., 2016, s. 68). Lærerens undervisning påvirkes både av hva som presenteres i læreboken, hvordan det presenteres, og hvor mye plass et tema tildeles i læreboken (Kongelf, 2019).

At læreboken har hatt en sentrall rolle i planlegging og gjennomføring av undervisningen i matematikkfaget til nå, behøver ikke bety at den vil fortsette å ha det. Kaufmann og Stenseth (2023) peker på at læreplanen Kunnskapsløftet LK20 bringer med seg en overgang fra lærerstyrt til elevstyrt undervisning (s. 111). De antyder at denne overgangen kan være krevende å gjennomføre i praksis (Kaufmann & Stenseth, 2023, s. 111). Allikevel kan kanskje en slik overgang medføre at matematikkundervisningen distanserer seg noe fra læreboken?

Med innføringen av læreplanen Kunnskapsløftet 2020 ble det gitt ut nye lærebøker i matematikk 1T høsten 2020 (Gyldendal, Aschehoug Undervisning, Cappelen Damm). Nytt i disse bøkene er blant annet at de har et innhold som inkluderer programmering. Drijvers (2015) trekker frem læreren, undervisningskonteksten og oppgavenes design som tre avgjørende faktorer for å lykkes med bruk av digital teknologi i undervisningen. Kaufmann og Stenseth (2023) mener vi ikke kan forvente at lærere med lav kompetanse i programmering skal designe økter som går ut over å lære programmering som en ferdighet (s. 119). Siden læreren i dag befinner seg i en situasjon hvor vedkommende skal undervise i programmering, uten nødvendigvis å være dreven på området (Jensen et

al., 2023; Kaufmann og Stenseth, 2023), så antar jeg at læreren vil fortsette å støtte seg på lærerboken i en periode fremover.

2.4.2 Programmering i lærebøker og andre undervisningsressurser

På grunn av lærebokens sentrale rolle i undervisningsplanleggingen i matematikkfaget, vil en lærebokanalyse kunne gi innblikk i hvilke læringsmuligheter elevene gis i matematikkundervisningen (Jablonka og Johanson, 2010). Flere forskere har utviklet rammeverk for å undersøke sammenhengen mellom programmeringsoppgaver, algoritmisk tenkning og matematiske konsepter i lærebøker i matematikk (Bråting & Kilhamn, 2022; Elicer & Tamborg, 2022; Jensen et al., 2023).

Bråting og Kilhamn (2022) utviklet et rammeverk basert på Brennan og Resnicks (2012) rammeverk for algoritmisk tenkning og Benton et al. (2017) sitt rammeverk for prinsipper ved design av programmeringsaktiviteter for matematikklæring. Bråting og Kilhamn (2022) ønsket å oppnå en oversikt over programmeringsinnholdet i svenske lærebøker på barnetrinn, og diskuterte hvordan innholdet kan påvirke elevenes forutsetninger til å lære matematikk (Bråting & Kilhamn, 2022, s. 595). I Bråting og Kilhamns (2022) artikkel blir programmering sett på som en del av algoritmisk tenkning og programmeringsoppgaver som oppgaver som kan utvikle den algoritmiske tenkningen (s. 594).

I deres studie anvendes feilsøking om debugging og det å finne feil, mens «forestille seg», handler om å forutsi hva som vil skje, eller reflektere over mulige utfall når man endrer verdier eller forutsetninger i programmet (Bråting & Kilhamn, 2022, s. 599). Med «utforsk»-oppgaver menes det i det teoretiske rammeverket Kilhamn og Bråting baserer studien sin på, en oppgave hvor man utforsker programmeringsverktøyet i seg selv, og idéene som presenteres i oppgaven (Benton et al., 2016, referert i Bråting & Kilhamn, 2022, s. 597). Å utforske i programmeringsoppgavene lar elevene prøve ut ting på egenhånd, jobbe interaktivt og rette opp i feil som oppstår underveis (Bråting & Kilhamn, 2022, s. 597).

Kilhamn og Bråtings rammeverk kritiseres for å være noe overfladisk, og for å begrense seg til å undersøke samtidig forekomst av algoritmisk tenkning og matematikk, ikke sammenhengen mellom disse (Elicer & Tamborg, 2022, s. 50). Elicer og Tamborg (2022) peker på at Kilhamn og Bråting (2022) sitt rammeverk og analyseverktøy ikke klarer å skille mellom oppgaver som innbyr til et integrert arbeid med algoritmisk tenkning og matematikk og oppgaver hvor algoritmisk tenkning eller matematikk er en kontekst for læring av den andre. Sammenhengen mellom matematikk og programmering i analysen av enkeltoppgaver kan være tynn, og allikevel identifiseres oppgavene som «brobyggende» mellom matematikk og programmering (Elicer & Tamborg, 2022, s. 50).

Elicer og Tamborg (2022) undersøkte selv 14 undervisningsopplegg utviklet for å knytte sammen matematikk og det nye, danske faget teknologiforståelse, på grunnskolenivå i Danmark (s. 45). Gjennom analyser av undervisningsoppleggene utviklet de seks kategorier som beskriver sammenhengen mellom programmering, algoritmisk tenkning og matematisk kompetanse i de didaktiske ressursene (Elicer & Tamborg, 2022, s. 45). Kategoriene beskrives som «ikke noe matematikk involvert», «ikke noe algoritmisk tenkning involvert», «matematikk som en kontekst», «algoritmisk tenkning som en kontekst», «konseptuell integrasjon» og «operasjonell integrasjon». Undervisningsoppleggene inneholdt oppgaver, som ble plassert i en av disse seks kategoriene.

For å kunne ta hensyn til små variasjoner i oppgavene skiller Elicer og Tamborg (2022) mellom handlinger og konsepter i oppgavene de analyserer (s. 51). Slik kan to oppgaver som ville blitt plassert i samme kategori av Bråting og Kilhamn (2022), havne i hver sin kategori hos Elicer og Tamborg (2022). Oppgaver som omhandler et matematisk konsept, kan i Elicer og Tamborgs (2022) artikkel kategoriseres som matematikk som kontekst. Dersom elevene skal utføre en matematisk handling kan kategorien operasjonell integrasjon brukes. (Elicer & Tamborg, 2022, s. 50). Slik kan Elicer og Tamborgs (2022) rammeverk beskrive programmeringsoppgavenes tilknytning til matematikken på et mer nærgående nivå enn Bråting og Kilhamns rammeverk (2022).

Kategorien konseptuell integrasjon viser til en sammenheng mellom programmering, algoritmisk tenkning og matematiske konsepter, som for eksempel et geometrisk objekt. Operasjonell integrasjon viser til en sammenheng mellom programmering, algoritmisk tenkning og matematiske handlinger, som for eksempel konstruksjon av et geometrisk objekt (Elicer & Tamborg, 2022).

Til forskjell fra i konseptuell integrasjon kan ikke oppgaver med operasjonell integrasjon løses dersom et konsept fra programmering, algoritmisk tenkning eller matematikk ble byttet ut med et annet konsept. En oppgave av denne typen kan for eksempel være «hvordan ser en algoritme for å tegne et kvadrat ut?» (Elicer & Tamborg, 2022, s. 50). Å programmere en robot til å kunne tegne kvadratet innebærer bruk av handlinger innen både matematikk og algoritmisk tenkning (Elicer & Tamborg, 2022, s. 50).

I Norge har Jensen et al. (2023) utviklet et rammeverk for analyse av programmeringsoppgaver i lærebøker i matematikk på ungdomsskolen. Rammeverket fungerer som analyseverktøy for å studere programmeringsoppgavenes sammenheng med matematiske emner og algoritmisk tenkning.

Som en del av den pågående studien «Programmering for å forstå matematikk» undersøkte Berge (2022) i hvilken grad programmeringsoppgavene i lærebøkene til Cappelen Damm og Gyldendal i matematikk 1T og matematikk R1 har potensiale for å bidra til læring av matematiske konsepter og konsepter innen programmering (s. 3). De utvidet rammeverket til Boston og Smith (2009) for analyse av matematikkoppgaver, til å også inkludere programmeringsaspektet ved oppgavene. De fant at den største delen av oppgavene ble klassifisert som mindre kognitivt utfordrende, både med tanke på matematikken og programmeringen (Berge, 2022, s. 6). Til tross for dette beskriver de at mange av oppgavene inkluderer relativt komplekse konsepter innen både matematikk og programmering (Berge, 2022, s. 6). Årsaken til at oppgavene allikevel klassifiseres som lite kognitivt utfordrende er at de etterfølger eksempler som er svært like oppgavene elevene skal løse (Berge, 2022, s. 6). På denne måten holder det at elevene modifiserer eksemplene. Berge (2022) fant at kun enkelte av oppgavene ber elevene utforske eller evaluere svaret de får i programmeringsoppgavene, og stiller derfor spørsmålstegn ved hvorvidt det er nok kognitivt utfordrende programmeringsoppgaver i bøkene, tatt i betraktning at programmering er implementert i matematikkfaget for å forbedre elevenes evne til problemløsning og algoritmisk tenkning (Berge, 2022, s. 8).

Berge (2022) understreker også at ikke samtlige oppgaver i en lærebok i matematikk skal være problemløsningsoppgaver, selv om programmering er innført i matematikkfaget for å trene elevene i problemløsning og algoritmisk tenkning (s. 8). Berge (2022) stiller derimot spørsmål ved hvor mange rutineoppgaver det bør være per problemløsningsoppgave, og hvorvidt programmeringsoppgavene er mer eller mindre kognitivt krevende for elevene enn oppgavene som ikke er knyttet til programmering (s.

8). Berge (2022) fant at programmeringsoppgavene i lærebøkene de undersøkte i stor grad var av typen hvor elevene ble bedt om å etterlikne et gitt eksempel (s. 6).

Siden programmeringen skal bidra til problemløsning er det nærliggende å tenke at det burde være minst like mange problemløsningsoppgaver blant programmeringsoppgavene som blant de øvrige oppgavene i læreboken. Berge (2022) peker på dette som et interessant område for videre forskning.

Tidligere forskning på programmering og algoritmisk tenkning i undervisningsressurser i matematikk har funnet oppgaver som i liten grad er kognitivt krevende med tanke på matematikken (Berge, 2022, s. 8; Bråting & Kilhamn, 2022, s. 601; Elicer & Tamborg, 2022, s. 50). Disse funnene virker gjentakende enten studien omhandler lærebøker på barnetrinnet (Bråting & Kilhamn, 2022), øvrige undervisningsressurser utviklet for grunnskolen (Elicer & Tamborg, 2022), eller, mest relevant for denne studien, i lærebøker for teoretisk og realfaglig matematikk i videregående skole (Berge, 2022, s. 8).

Forskerne trekker frem et stort uutnyttet potensial i lærebøkene når det kommer til programmering for matematikklæring (Bråting & Kilhamn, 2022, s. 607). I en svensk barneskolekontekst konkluderes det med at elevene ville hatt større muligheter til å utnytte sammenhenger mellom programmering og matematikk dersom bøkene hadde inneholdt flere oppgaver av typen «utforsk», «forestill deg» og «finn feil» (Bråting & Kilhamn, 2022, s. 607).

At en stor del av oppgavene gir muligheter for arbeid med matematikk på et begrenset nivå, kan virke å være i strid med intensjonen ved innføring av programmering i matematikkfaget. Elicer og Tamborg (2022) peker på at oppgaver som ikke kobler sammen programmering eller algoritmisk tenkning og matematikk kan virke problematiske (s. 50). Det samme inntrykket kan gis i oppgaver hvor matematikken er en kontekst for å lære programmering (Elicer & Tamborg, 2022, s. 50). Derimot kan slike oppgaver være nødvendige (Berge, 2022, s. 8), eller til og med viktige, med tanke på læring av både programmering og matematikk (Elicer & Tamborg, 2022, s. 50).

For innlæring av den nye programmeringen er det nødvendig med rutineoppgaver, og slike oppgaver har derfor sin plass i lærebøkene i matematikk (Berge, 2022, s. 8). Det å få stoppe opp i det utforskende arbeidet for å gjennomføre en rutineoppgave, kan være det som skal til for at et matematisk eller programmeringsteknisk konsept virkelig fester seg hos eleven (Elicer & Tamborg, 2022, s. 50).

2.4.3 Overføringsverdier ved programmering i matematikk

I læreplanen kan man lese at intensjonen med å innføre programmering som del av matematikkfaget er at det skal bidra til problemløsning og økt matematikkforståelse (Kunnskapsdepartementet, 2019). Utsagnet er ikke ledsaget av noen kildehenvisning, og har variert støtte i forskningslitteraturen. Scherer et al. (2019) fant i sin metaanalyse av 105 studier om programmering, at kompetanse i programmering gir læringsutbytte innen kreativ tenkning, matematiske ferdigheter og metakognisjon (s. 28). Sett i sammenlikning med studier som inneholdt en kontrollgruppe som ikke fikk programmeringsundervisning, men heller jobbet systematisk med læringsstrategier og problemløsning, også digitalt, så fant de derimot ikke noen signifikant forskjell på gruppene (Scherer et al., 2019). Dette antyder at det ikke er programmeringen som digitalt hjelpemiddel i seg selv som bidrar til økt matematisk forståelse, men arbeidsmåten som anvendes i programmeringsarbeidet. Som Utdanningsforbundet også

sier det i artikkelen «Kva er nytt i matematikk?» kan programmering, når det brukes til problemløsning, bidra til økt matematikkforståelse (Scherer et al., 2018; Utdanningsdirektoratet, 2023, s. 1). Nøkkelordet her synes altså å være problemløsning.

Shute et al. (2017) presenterer i sin artikkel hvordan algoritmisk tenkning sammenfaller og skiller seg fra ulike andre kognitive prosesser, blant annet matematisk tenkning (Shute et al., 2017, s. 145). Matematisk tenkning består av "beliefs about math, problem solving processes, and justification for solutions" (Shute et al., 2017, s. 145). Den største fellesnevneren mellom algoritmisk tenkning og matematisk tenkning er problemløsning (Wing, 2008). Weintrop et al. (2016, s. 139) hevder at elever som kan forstå, bearbeide og lage dataprogrammer kan bruke disse ferdighetene til å utvikle seg innen matematikk. Weintrop et al. (2016) klassifiserer programmering som en del av nettopp computational problem solving, en del av problemløsning med algoritmisk tenkning.

For en samlet oversikt over likheter og forskjeller mellom matematisk tenkning og algoritmisk tenkning, se figur 2.2.

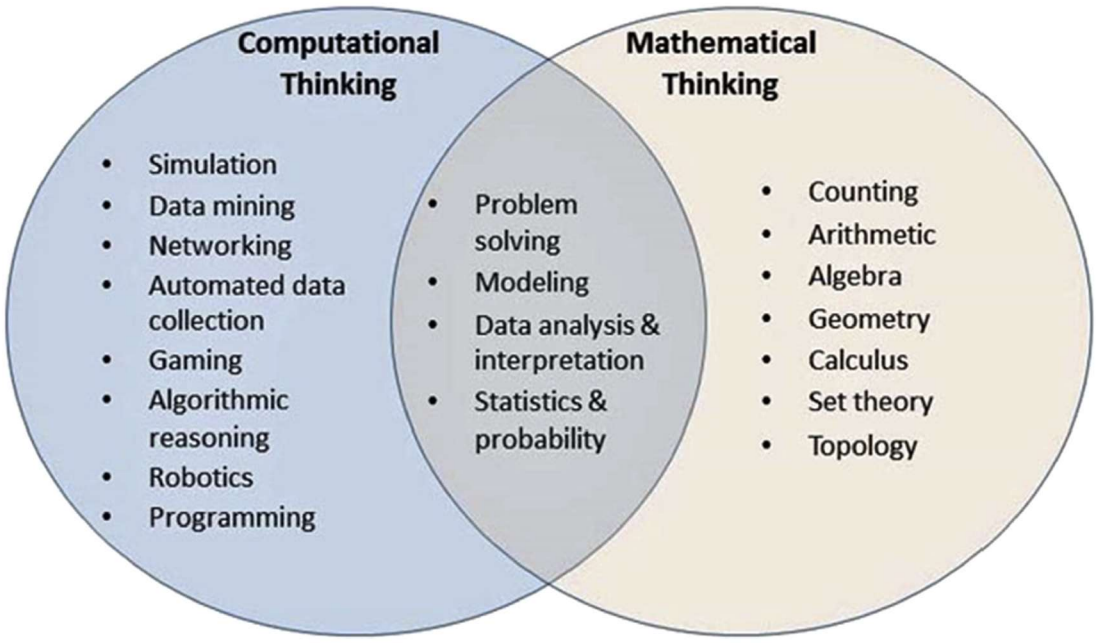


Fig. 1. Similarities and differences between CT and mathematical thinking. Adapted from Sneider et al. (2014).

Figur 2.2: Likheter og forskjeller mellom computational thinking og matematisk tenkning. (Shute et al., 2017, s. 145, bearbeidet fra Sneider er al., 2014).

Lye og Koh (2014) har gjennomført en litteraturstudie, for å undersøke hvilket læringsutbytte elevene sitter igjen med etter gjennomført undervisning i programmering, med tanke på algoritmisk tenkning (s. 54). De fant at artiklene de undersøkte rapporterte om økte kunnskaper og ferdigheter hos elevene innen konsepter knyttet til programmering, men ikke innen problemløsningspraksiser (Lye og Koh, 2014, s. 58). Med andre ord ble det oppfattede læringsutbyttet rent programmeringsteknisk, og forskningsartiklene bidro ikke til å forstå hvordan programmering kan fremme algoritmisk tenkning, som tankeprosess og problemløsningsstrategi.

Forsström og Kaufmann (2018) har gjennomført en litteraturstudie hvor de undersøker programmeringens potensiale i matematikkundervisningen. De fant at det var gjort lite forskning på feltet (Forsström & Kaufmann, 2018, s. 28). Den forskningen som fantes viste en sammenheng mellom programmering i matematikkfaget og en bedring i elevenes matematikkprestasjoner, samt økt motivasjon til å lære matematikk (Forsström & Kaufmann, 2018, s. 28). Derimot var ikke studiene de undersøkte generaliserbare: De positive utfallene knyttet til matematikk var bare synlige i enkelte grupper av deltakerne i forskningsstudiene (Forsström & Kaufmann, 2018, s. 26). Hver enkeltstudie i litteraturstudien var bygget på ulike parametere, og resultatene kunne dermed ikke sammenliknes med hverandre (Forsström & Kaufmann, 2018, s. 26). I tillegg var de forbedrede ferdighetene og holdningene i matematikk kun knyttet til enkelte matematiske emner, fordi ikke alle emnene ble ansett å være like programmerbare (Forsström & Kaufmann, 2018, s. 26).

Det etterlyses mer forskning på sammenhengen mellom algoritmisk tenkning i undervisning og elevenes problemløsningsferdigheter (Lye & Koh, 2014) og for å undersøke hvilket potensial som ligger i implementeringen av programmering i matematikkfaget, samt forskningsbaserte begrunnelser for en slik implementering (Forsström & Kaufmann, 2018, s. 28).

3 Metode

Det skilles ofte mellom kvalitativ og kvantitativ metode i forskning (Tjora, 2023, s. 26). Tradisjonelt sett skilles disse to metodene fra hverandre ved at kvalitativ forskning har vekt på forståelse og genererer empiri i form av tekst, mens den kvantitative forskningen vektlegger forklaring og har data i form av tall (Tjora, 2023, s. 27). Tjora (2023) påpeker at selv om disse kjennetegnene er beskrivende, så er den kvalitative forskningen mangfoldig (s. 27). En kombinasjon av kvalitative og kvantitative metoder vil ofte være gunstig, så sant man har ressurser til det (Tjora, 2023, s. 26).

I denne masterstudien gjennomfører jeg en dokumentanalyse av programmeringsoppgaver i trykte lærebøker i faget matematikk 1T. Jeg undersøker oppgavene med fokus på hvilke aspekter ved algoritmisk tenkning, type resonnering og hvilke matematiske kunnskapsområder de inkluderer. I tillegg undersøker jeg frekvensen av forekomsten av disse ulike aspektene ved oppgavene. Dermed er min forskning av både kvalitativ og kvantitativ art. Å anvende både kvalitativ og kvantitativ tilnærming i samme undersøkelse kalles metodetriangulering, eller mixed methods (Pingel, 2010).

I min oppgave ønsker jeg å undersøke hvilke muligheter programmeringsoppgavene i lærebøkene i matematikk 1T gir for arbeid med algoritmisk tenkning, problemløsning og matematiske kunnskapsområder, fordi disse begrepene anvendes i læreplanen for å begrunne og beskrive programmeringens rolle i matematikkfaget. Det finnes flere teoretiske rammeverk som omhandler algoritmisk tenkning, problemløsning og rammeverk for lærebokanalyser knyttet til matematikk-emner (se kap. 2). Mange av rammeverkene omhandler derimot disse temaene hver for seg. Jeg ønsker å studere samtlige av disse tre aspektene ved programmeringsoppgavene i lærebøkene i matematikk 1T. Jeg har derfor kombinert ulike teoretiske rammeverk fra forskningslitteraturen, for å danne rammeverket for min lærebokanalyse.

I dette kapittelet vil jeg redegjøre for innholdsanalysen av lærebøker, som metode. Deretter presenteres utvalget av lærebøker og oppgaver i disse, etterfulgt av en presentasjon av analyseverktøyet som anvendes i denne studien. Analyseprosessen beskrives så i detalj, før jeg redegjør for studiens kvalitet ved å si noe om dens validitet og reliabilitet.

3.1 Innholdsanalyse av lærebøker

Forskning på lærebøker i matematikk kan handle om lærebokens innflytelse, læreboken i seg selv og bruken av, og innvirkningen til, læreboken (Rezat & Strässer, 2012). Jeg posisjonerer meg innenfor det andre forskningsområdet til Rezat og Strässer (2012) med denne studien.

Mesa (2004) ser på lærebokanalysen som en hypotetisk undersøkelse av et sannsynlig læringsutbytte, med visse betingelser (s. 255). Den undersøker hva elevene ville ha lært, dersom de arbeidet med samtlige deler av læreboken i matematikkundervisningen, og hva elevene ville ha lært dersom de hadde jobbet med å løse alle oppgavene i læreboken (Mesa, 2004, s. 256).

Forskere som analyserer lærebøker møter utfordringer i den hypotetiske undersøkelsen (Charalambous et al., 2010, s. 118). Det at et emne er inkludert i læreboken er ingen garanti for at emnet blir undervist, selv om sannsynligheten for det øker (Valverde et al., 2002). Lærebokens rolle i undervisningen avhenger av hvordan elevene og læreren anvender læreboken (Charalambous et al., 2010, s. 118), og en lærebokanalyse kan derfor ikke si noe om hvilken læring læreboken bidrar til.

Lærebokanalyser undersøker kun *det intenderte pensum* – intensjonen med hva elevene skal lære i faget, ut fra innholdet i lærebøkene, og ikke det *implementerte pensumet*, altså hva elevene altså får presentert av matematikkinnhold i undervisningen sin (Charalambous et al., 2010, s. 118). Allikevel kan lærebokanalysene belyse en side av elevenes muligheter til å lære matematikk (Charalambous et al., 2010, s. 118).

Norge har nasjonale kompetansemål, fastsatt av læreplanen (Kunnskapsdepartementet, 2019). Charalambous et al. (2010, s. 119) mener da at lærebøkene her sannsynligvis reflekterer læreplanen. På hvilken måte og i hvilken grad, ønsker jeg å undersøke nærmere i min masterstudie, når det kommer til implementeringen av programmering i MAT09-01.

Det finnes ikke enighet i forskningslitteraturen rundt én ideell fremgangsmåte for lærebokanalyse (Charalambous et al., 2010, s. 118). Charalambous et al. (2010) har utviklet et rammeverk for lærebokanalyse i matematikk, som inkluderer en horisontal og en vertikal analyse (s. 123). Med horisontal analyse menes en oversikt over læreboken, med informasjon om strukturen i boka (Charalambous et al., 2010, s. 122). Den vertikale analysen gir en dypere analyse med detaljer som beskriver hvordan lærebokforfatterne behandler og presenterer matematiske emner i læreboken, hva som er forventet av elevene, og hvilke sammenhenger som eksplisitt belyses mellom de matematiske emnene (Charalambous et al., 2010, s. 122). Jeg vil anvende dette rammeverket i min studie, med egne tilpasninger når det kommer til den vertikale analysen. For en nærmere gjennomgang av analyseverktøyet i denne studien, se kap. 3.3.

3.2 Utvalg

Det finnes tre trykte lærebøker spesielt utviklet for faget MAT09-01. Disse er Sinus 1T av forlaget Cappelen Damm (Maus, G. & Smestad, B.-T., 2020), Matematikk 1T av Aschehoug (Frydenlund, C. & Holst, L., 2020) og Mønster 1T av Gyldendal (Raustøl, A. & Lund, H.-S., 2020). Alle disse lærebøkene ble gitt ut i 2020, klare for bruk i skolen samtidig som innføringen av Kunnskapsløftet LK20.

Utvalget av empiri til denne studien baserer seg på programmeringsoppgaver fra samtlige av disse tre trykte lærebøkene. En oppgave defineres i denne studien som «en instruksjon eller et spørsmål som krever respons fra eleven […]» (Haladyna, 1997, s. 36). Denne definisjonen innebærer at lærebokens angitte oppgaver kan bestå av det som klassifiseres som flere oppgaver i denne studien: Hver respons fra eleven kategoriseres som en egen oppgave, etter Haladynas definisjon.

Med programmeringsoppgave menes i denne studien en oppgave som spesifikt inkluderer ordene program, programmering eller Python. Det refereres aldri til noe annet programmeringsspråk i de analyserte lærebøkene. Derfor holder disse tre ordene som utvalgskriterier. I tillegg har jeg inkludert samtlige oppgaver i siste kapittel i Mønster 1T. Dette kapittelet er dedikert til programmering. Oppgavene her har ingen ordlyd som gjør

at de kan klassifiseres som programmeringsoppgaver. For eksempel er oppg. 4a på s. 409 som følger: «Avgjør om 167 335 er delelig med 3» (Raustøl, A. & Lund, H.-S., 2020, s. 409). Fordi oppgaven tilhører programmeringskapittelet tolker jeg den dithen at den skal løses ved hjelp av programmering.

Jeg har tatt utgangspunkt i de trykte lærebøkene, fordi disse er konstante, og ikke risikerer å oppdateres underveis i min analyse, slik oppgaver på en nettside kan gjøre. Alle de tre læreverkene tilbyr også elevnettsteder med oppgaver. Disse er ikke studert.

3.3 Presentasjon av analyseverktøyet

Følgende analyseverktøy anvendes i denne studien:

	Aspekter ved algoritmisk tenkning	Type resonnering	Matematisk kunnskapsområde
Oppgaven gir mulighet til arbeid med	Abstrahering	Kjent algoritmisk resonnering	Matematiske bevis
	Dekomponering	Guidet algoritmisk resonnering	Formler og variable størrelser
	Algoritmebehandling	Memorert resonnering	Likninger og ulikheter
	Evaluering	Annen imiterende resonnering	Funksjoner
	Generalisering	Kreativ resonnering	Modellering
			Identifisering og formulering av problemer
			Vekstfart og derivasjon
			Polynomdivisjon
			Trigonometri
			Annet matematisk kunnskapsområde
			Programmering

Tabell 3.1: Studiens analyseverktøy

Analyseverktøyets innholdskomponenter presenteres under.

3.3.1 Aspekter ved algoritmisk tenkning.

Som begrunnet i kapittel 2.2.7 anvendes Selby og Woollard (2013) sin definisjon av algoritmisk tenkning i denne studien. Komponentene abstrahering, dekomponering, algoritmebehandling, evaluering og generalisering defineres slik Woollard beskriver dem sammen med resten av forskerteamet i Csizmadia et al. (2015), se kap. 2.2.4.

3.3.2 Type resonnering

Her undersøker jeg om oppgavene kan bidra til problemløsning, ved å anvende begrepene kreativ og imitativ resonnering (Boesen, 2006; Lithner 2008; Palm et al., 2005).

Denne delen av analyseverktøyet er opprinnelig utarbeidet av Palm et al. (2005), med enkelte modifiseringer gjort av meg. Endringene spesifiseres i teksten under.

Kjent algoritmisk resonnering (KAR): Programmeringsoppgaven vurderes som svært lik minimum tre *forutgående* øvrige oppgaver og/eller eksempler fra læreboken. Sammenlikningen foregår basert på oppgavevariablene som presentert under.

Oppgavene må høre til forutgående delkapitler. At jeg vurderer om det finnes tre *forutgående* eksempler og/eller oppgaver som likner, er en spesifisering som er lagt til av meg. Jeg antar at sekvensene i bøkene anvendes slik de er satt opp, slik Charalambous et al. (2010) også legger vekt på sekvensene og rekkefølgen stoff presenteres i sin lærebokanalyse. Dermed vil en oppgave ikke være kjent for elevene første gang de møter på den, selv om det finnes liknende oppgaver og/eller eksempler lenger bak i boka. Jeg antar at elevene jobber med oppgavene i den rekkefølgen de er gitt. Se eksempel på oppgaveanalyse.

For oppgaver som ikke tilhører noe spesifikt delkapittel, som programmeringsoppgavene bakerst i Mønster 1T-boka (Raustøl, A. & Lund, H.-S., 2020)., ser jeg bort fra rekkefølgen, da jeg ikke kan si noe om sannsynligheten for når i opplæringsløpet disse oppgavene introduseres for elevene.

Eksemplene kan befinne seg hvor som helst i læreboken, da jeg antar det som mer sannsynlig at elevene skummer gjennom eksemplene i boka etter svar dersom de leter etter hvordan de skal gå frem for å løse en oppgave.

Guidet algoritmisk resonnering (GAR): Oppgaven inkluderer en fremgangsmåte eller formel som kan brukes til å løse oppgaven. Det holder med én tilsvarende *forutgående* oppgave i lærebøkene for at en oppgave skal kunne kategoriseres som løsbar ved guidet algoritmisk resonnering.

Memorert resonnering (MR): En programmeringsoppgave klassifiseres som mulig å løse ved memorert resonnering, dersom det finnes tre svar på oppgaver, eksempler eller i teoriteksten i lærebøkene som kan løse oppgaven.

Kreativ resonnering (KR): Oppgaven vurderes som ikke-løsbar ved bruk av overfladisk resonnement. Mulige svar eller algoritmer som løser oppgaven er ikke beskrevet i læreboken. Innholdskunnskapen er inkludert i læreboken, slik at det vurderes at oppgaven er mulig å løse for i det minste noen av elevene.

For å kunne skille mellom problemløsningsoppgaver knyttet til programmering og problemløsningsoppgaver knyttet til matematikk, har jeg inndelt den kreative resonneringen i tre underkategorier:

Kreativ resonnering innen matematikk (KRM): Det er matematikken i programmeringsoppgaven som gjør at oppgaven klassifiseres som løsbar ved bruk av kreativ resonnering.

Kreativ resonnering innen matematikk og programmering (KRMP): Både matematikken og programmeringen i programmeringsoppgaven inneholder aspekter som gjør at oppgaven klassifiseres som løsbar ved bruk av kreativ resonnering.

Kreativ resonnering innen programmering (KRP): Det er kun programmeringen i oppgaven som gjør at den er løsbar ved bruk av kreativ resonnering.

Ingen av delene: Oppgaver som ikke er mulige å løse, basert på innholdskunnskapen som gis i læreboken. De kvalifiseres verken som mulig å løse ved imitativ resonnering, men heller ikke ved kreativ resonnering.

Utførte endringer i analyseverktøyet:

Fordi elevene benytter seg av læreboka når de arbeider med disse oppgavene endrer jeg kravene til hva læreboken må inneholde for at det skal være sannsynlig at elevene anvender de ulike typene resonnering. For eksempel behøves ikke tre like oppgaver for å kategorisere som kjent algoritmisk resonnering. Til forskjell fra i Palm et al. (2005) trenger ikke elevene å huske oppgavene i læreboken, fordi de har den tilgjengelig. Derimot tar jeg hensyn til hvor i boka oppgavene står i forhold til hverandre og eksemplene de likner på. Oppgaver som følger rett eller tett etter et eksempel det går an å kopiere fremgangsmåten til kategoriseres som kjent algoritmisk resonnering, selv om det ikke finnes flere av denne typen oppgaver i boka. Dette fordi jeg anser det sannsynlig at elevene studerer det forutgående eksempelet før de løser oppgaven som følger rett etter. Presiseringer om at liknende eksempler og/eller oppgaver skal være forutgående, altså forekomme i læreboken før den gjeldende oppgaven, er lagt til av meg. Dette for å ta hensyn til at elevene sannsynligvis arbeider med læreboken i den rekkefølgen den er satt opp i (Charalambous et al., 2010).

Å dele opp den kreative resonneringen i kategoriene kreativ resonnering matematikk (KRP), kreativ resonnering matematikk og programmering (KRMP) og kreativ resonnering programmering (KRP), er en av mine tilpasninger av rammeverket, for å bedre passe som analyseverktøy i min studie. Jeg ønsker å kunne avgjøre hvor mange av oppgavene som er problemløsende med tanke på den involverte matematikken, og dermed sile ut oppgaver som klassifiseres som problemløsende med tanke på programmeringen (KRP). Dette er et grep jeg har tatt for å kunne undersøke mulighetene for problemløsning innen matematikk i programmeringsoppgavene.

I arbeidet med analysen føyde jeg til kategorien *ingen av delene*, for å kunne kategorisere samtlige av programmeringsoppgavene. Et av læreverkene inneholdt flere oppgaver som ikke var løsbare verken med kreativ eller imitativ resonnering, fordi elevene ikke har kunnskapen om programmering som trengs for å løse oppgaven, og heller ikke kan finne denne i læreboka. Her trengs ytterligere kilder, for at elevene skal ha mulighet til å løse oppgaven.

Oppgavevariablene som oppgavene sammenliknes ut fra er som følger:

1. Oppdrag
2. Eksplisitt informasjon om situasjonen
3. Representasjoner
4. Språklige egenskaper
5. Eksplisitt formulerte hint
6. Responsformat

De ulike formene for imitativ resonnering baserer seg på at elevene kjenner igjen løsningsmetoder eller svar fra liknende oppgaver, som de kan forsøke å anvende i oppgaven de jobber med, for å finne et svar. For at elevene skal anse oppgaver som «liknende» hverandre, må de kjenne igjen oppgavevariabler i oppgaven i jobber med, og disse liknende oppgavene i boken. Likhet i oppgavevariablene kan føre elevene til å tro at

oppgavene kan løses ved samme svar, eller samme algoritme (Palm et al., 2011. s, 231). Under følger en beskrivelsene av oppgavevariablene.

1. *Oppdrag:* I oppgavene gis elevene, implisitt eller eksplisitt, et oppdrag de skal utføre, i form av et spørsmål eller en instruks (Palm et al., 2011, s. 231). Et slikt oppdrag i en programmeringsoppgave kan være «Utvid programmet til å…», «Løs likningen i Python», eller «Plott grafen i Python».

2. *Eksplisitt informasjon om situasjonen*: Oppgaven gis ofte i en beskrivelse av en situasjon (Palm et al., 2011, s. 232). Beskrivelsen kan bestå av eksplisitt informasjon om matematiske komponenter, som begreper, verdier og liknende, og eksplisitt informasjon om en reell situasjon (real-life event) (Palm et al., 2011, s. 232). Hvis en programmeringsoppgave inneholder informasjon om en reell situasjon, som elevene kjenner igjen fra en annen oppgave, kan de anta at programmeringsoppgaven er løsbar på samme måte. For eksempel handler både oppg. 1.59 a og b, på s. 31, samt et eksempel på s. 30 i Sinus 1T om vindkraft på Storheia (Maus, G. & Smestad, B.-T., 2020, s. 30-31).

3. *Representasjoner*: Situasjonen i en programmeringsoppgave kan bli presentert ved hjelp av forskjellige representasjoner, som for eksempel et flytskjema, kode, symboler, tabeller, tekst eller grafer. Valget av representasjoner kan avgjøre hvor enkelt det er for elevene å kjenne igjen sammenhengen mellom oppgavene i boka (Palm et al., 2011, s. 232). Like representasjoner kan gjøre at elevene enklere ser sammenhengen mellom oppgaver, mens forskjellige representasjoner gjør det mer utfordrende (Palm et al., 2011, s. 232).

4. *Språklige egenskaper:* Oppgavetekstene kan ha forskjellige språklige egenskaper, som ulik semantikk (betydningen til ord og setninger) eller ulik syntaks (for eksempel kan man velge å skrive 20 000 eller «tjue tusen») (Palm et al., 2011, s. 232). Med semantikk som del av oppgavevariabelen menes det at elevene kan tillegge ulike matematiske ord en algoritme, som de har sett brukt i oppgaver med lik semantikk. Palm et al. (2011) bruker et eksempel som er passende med tanke på elever som gjennomfører faget matematikk 1T: Hvis oppgavene i lærebøkene ofte knytter ordet «toppunkt» med algoritmen å finne når den deriverte til en funksjon er lik null, så vil ordet «toppunkt» kunne fungere som en assosiasjon som får elevene til å ta i bruk denne algoritmen (Palm et al., 2011, s. 232). I programmeringsoppgavene i lærebøkene i matematikk 1T observerer jeg at *tilnærmingsverdi* kan være et eksempel på et ord som kan knyttes til algoritmer for numerisk likningsløsning og Newton-Raphson-metoden, fordi mange av programmeringsoppgavene som anvender *tilnærmingsverdi* er løsbare ved denne metoden.

5. *Eksplisitt formulerte hint*. Eksplisitte hint kan lede elevene mot en bestemt algoritme for å løse en oppgave (Palm et al., 2011, s. 232). For eksempel «bruk halveringsmetoden til å…».

6. *Svarformat*: Formatet på svaret som kreves av elevene kan være en variabel som assosierer en oppgave med en annen (Palm et al., 2011, s. 232). Eksempler på svarformat kan være å velge mellom flere gitte svaralternativer eller å forklare fremgangsmåten i et gitt løsningsforslag (Palm et al., 2011, s. 232). Knyttet til programmering kan svarformat være å beskrive programkode i naturlig språk, lage en algoritme for å utføre en handling, gjøre endringer i en gitt kode, eller lage programkode selv.

3.3.3 Matematisk kunnskapsområde

De matematiske kunnskapsområdene i denne oppgaven er ment å gjenspeile matematikken elevene skal tilegne seg kunnskap om, gjennom opplæringen i faget MAT09-01. For å utarbeide kategoriene for disse matematiske kunnskapsområdene, har jeg undersøkt læreplanen i MAT09-01, og hva denne inneholder av informasjon om matematiske temaer, i tekstene om fagets kompetansemål, fagets relevans og sentrale verdier, samt fagets kjerneelementer. Begrepet matematiske kunnskapsområder syntes å beskrive innholdskunnskapen elevene skal inneha etter endt opplæring bedre enn alternative begreper som matematiske temaer eller emner. Jeg har utelatt ferdigheter og fokusert på matematikkfaglig teori. Begrepet matematiske kunnskapsområder må ikke forveksles med kjerneelementet i læreplanen for MAT09-01, med samme navn.

Det er til sammen 14 kompetansemål i MAT09-01 (Kunnskapsdepartementet, 2019). Jeg har kategorisert sammen kompetansemål som tilhører under samme kunnskapsområde. Gjennom analyseprosessen ble det lagt til to kategorier; annet matematisk kunnskapsområde og programmering. Årsaken til dette var at ikke alle oppgavene i lærebøkene lot seg plassere innen kategoriene dannet på bakgrunn av læreplanens kompetansemål.

Se tabell 3.1 for en oversikt over de matematiske kunnskapsområdene og deres tilhørende kompetansemål.

Matematisk kunnskapsområde	**Tilhørende kompetansemål**
Matematiske bevis	Kompetansemål 2: Lese og forstå matematiske bevis og utforske og utvikle bevis i relevante matematiske emner.
Formler og variable størrelser.	Kompetansemål 3: Identifisere variable størrelser i ulike situasjoner, sette opp formler og utforske disse ved bruk av digitale verktøy.
Likninger og ulikheter	Kompetansemål 4: Utforske strategier for å løse likninger, likningssystemer og ulikheter og argumentere for løsningene sine. Kompetansemål 6: Utforske sammenhenger mellom andregradslikninger, andregradsulikheter, andregradsfunksjoner, kvadratsetningene og bruke sammenhengene i problemløsning.
Funksjoner	Kompetansemål 5: Forklare forskjellen mellom en identitet, en likning, et algebraisk uttrykk og en funksjon. Kompetansemål 6: Utforske sammenhenger mellom andregradslikninger, andregradsulikheter, andregradsfunksjoner, kvadratsetningene og bruke sammenhengene i problemløsning. Kompetansemål 9: Utforske og beskrive egenskapene ved polynomfunksjoner, rasjonale funksjoner, eksponentialfunksjoner og potensfunksjoner
Modellering	Kompetansemål 7: Modellere situasjoner knyttet til ulike temaer, drøfte, presentere og forklare resultatene og argumentere for om modellene er gyldige
Identifisering og formulering av problemer	Kompetansemål 8: Lese, hente ut og vurdere matematikk i relevante tekster og presentere relevante beregninger og analyser av resultatene Kompetansemål 1: Formulere og løse problemer ved hjelp av algoritmisk tenkning, ulike problemløsningsstrategier, digitale verktøy og programmering

Vekstfart og derivasjon	Kompetansemål 10: Bruke gjennomsnittlig og momentan vekstfart i konkrete tilfeller og gjøre greie for den deriverte.
Polynomdivisjon	Kompetansemål 11: Bruke polynomdivisjon til å gjøre om algebraiske uttrykk, drøfte funksjoner og løse likninger og ulikheter
Trigonometri	Kompetansemål 12: Gjøre greie for definisjonene av sinus, cosinus og tangens. Bruke trigonometri til å beregne lengder, vinkler og areal i vilkårlige trekanter. Kompetansemål 13: Grunngi sinus-, cosinus- og arealsetningen Kompetansemål 14: Bruke trigonometri til å analysere og løse sammensatte teoretiske og praktiske problemer med lengder, vinkler og areal
Annet matematisk kunnskapsområde	Matematisk kunnskapsområde som ikke inngår i kompetansemålene i faget, eller i teksten om fagets kjerneelementer.
Programmeringsteknisk	Oppgaver som ikke inneholder tilknytning til noe matematisk kunnskapsområde, og som regnes som rent knyttet til programmering.

Tabell 3.1: Matematiske kunnskapsområder

3.3.4 Analyseprosessen

I dette delkapittelet vil jeg beskrive analyseprosessen i denne studien. Målet med analysen er å undersøke hvor mange, og hvilke muligheter elevene får til å arbeide med algoritmisk tenkning, problemløsning og matematikk i programmeringsoppgavene i de tre trykte lærebøkene i faget MAT09-01.

For å analysere oppgavene med tanke på muligheter for problemløsning vil jeg anvende begrepene kreativ og imitativ resonnering (Lithner, 2008). Hensikten er å avgjøre hvorvidt det er mulig og sannsynlig at elevene kan løse programmeringsoppgavene ved bruk av imitativ resonnering, basert på fagstoffet og oppgavene som presenteres i læreboken, samt rekkefølgen stoffet og oppgavene presenteres i. Dersom ingen form for imitativ resonnering fremstår som sannsynlig og mulig, basert på informasjonen og rekkefølgen på informasjonen som finnes i læreboken, vil jeg vurdere om kreativ resonnering er mulig. Som nevnt i kapittel 2.3.2 vil oppgaver som er løsbare ved bruk av kreativ resonnering regnes som problemløsningsoppgaver i denne studien.

Analyseprosessen består av følgende trinn:

1. Horisontal analyse
    a. Kartlegge antall kapitler og oppbyggingen av disse i de respektive bøkene.
    b. Kartlegge av antall oppgaver totalt i de respektive bøkene.
    c. Identifisere oppgaver som kategoriseres som programmeringsoppgaver i denne studien. For hver av disse oppgavene gjennomføre steg 2a-d.
2. Vertikal analyse
    a. Identifisere svar eller løsninger som kan løse programmeringsoppgaven, se figur 3.2.
    b. Algoritmisk tenkning
        i. Avgjøre hvilket eller hvilke aspekter av algoritmisk tenkning elevene sannsynligvis vil arbeide med når de jobber med å løse programmeringsoppgaven. Merk at det kan være flere per programmeringsoppgave.
    c. Problemløsning
        i. Beskrive oppgaven ved hjelp av oppgavevariablene.

ii. Lete etter informasjon i lærebøkene som kan løse programmeringsoppgaven, og notere hvor i boken denne informasjonen finnes, i forhold til den aktuelle programmeringsoppgaven.
iii. Søke gjennom programmeringsoppgavene i læreboken for å identifisere øvrige oppgaver eller eksempler som kan løses med samme svar eller algoritme som den aktuelle programmeringsoppgaven som er gjenstand for analyse.
iv. Beskrive disse oppgavene som kan løses med samme svar eller algoritme ved hjelp av oppgavevariablene.
v. Sammenlikne oppgavevariablene på programmeringsoppgaven som er gjenstand for analysen, med oppgavene eller eksemplene som kan løses ved hjelp av samme svar eller algoritme, for å undersøke hvorvidt oppgavene kan oppfattes som like eller ei.
vi. På bakgrunn av steg 2ci-v, kategorisere programmeringsoppgaven etter rammeverket for kreativ og imitativ resonnering. Dersom verken kreativ eller imitativ resonnering kan anvendes kategoriseres oppgaven som «ingen av delene».
vii. Dersom programmeringsoppgaven kategoriseres som løsbar ved bruk av kreativ resonnering vurderes hvorvidt den kreative resonneringen er knyttet til matematikk, programmering eller en kombinasjon av disse.

d. Matematisk kunnskapsområde
   i. Identifisere hvilket matematisk kunnskapsområde programmeringsoppgaven bidrar til arbeid med.

Som det vil fremkomme av resultatene i kapittel 4, vil ikke alle kategoriene inneholde oppgaver fra lærebøkene etter gjennomført lærebokanalyse. Jeg har allikevel valgt å beholde analyseverktøyet slik det fremstår her, uten å endre det for å passe til de faktiske funnene i bøkene. Årsaken til dette er at jeg ønsker å undersøke hvilke muligheter til det jeg mener er læreplanens intensjon med programmering i matematikkfaget elevene faktisk får, i arbeidet med programmeringsoppgavene i lærebøkene. Der hvor enkelte oppgaver ikke har vært kategorisertbare ved hjelp av rammeverket, har jeg lagt til kategorier, slik at alle programmeringsoppgavene tilhører en kategori for aspekt ved algoritmisk tenkning, type resonnering og matematisk kunnskapsområde. Se kap. 3.3 for en oversikt over rammeverket slik det anvendes i denne studien.

3.3.5 Eksempel på oppgaveanalyse

Mønster 1T, reflekter og diskuter, s. 263:
Reflekter og diskuter! Hvordan kan du utvide programmet i eksempelet til å telle hvor mange ganger halveringsmetoden kjøres?

Oppgaven følger etter et eksempel 19, som løser likningen $x^2 - 3 = 0$ for $1 \leq x \leq 2$, ved hjelp av et program som anvender halveringsmetoden i Python, ved en while-løkke som

kjøres til nullpunktet er funnet med en nøyaktighet på 0,001. Se figur 3.1. En while-løkke er en del av en programkode som gjentas så lenge en betingelse er sann, her så lenge man ikke har kommet til nullpunktet med en tusendels nøyaktighet.

EKSEMPEL 19

Skriv et program i Python som løser likningen $x^2 - 3 = 0$ for $x \in [1, 2]$.
Bruk halveringsmetoden med en tusendels nøyaktighet.

Løsning:

Halveringsmetoden:

```
1  def f(x):
2      return x**2 - 3
3
4  nedregrense = 1
5  øvregrense = 2
6  nøyaktighet = 0.001
7
8  a = nedregrense      #startverdi venstre grense
9  b = øvregrense       # startverdi høyre grense
10 m = (a + b)/2        #midtpunkt
11
12 while b-a > nøyaktighet:
13     if f(a)*f(m) <0:  #skal forkaste høyre halvdel
14         b = m
15     else:             # skal forkaste venstre halvdel
16         a = m
17     m = (a + b)/2     # m oppdateres til å være
                         midtpunktet i det nye intervallet
18
19 print (f'Løsningen til likningen er x ≈ {m:.3f}.')
```

Dette programmet tar ikke høyde for at endepunktene eller midtpunktet kan være selve nullpunktet.

Når vi kjører programmet, får vi:

```
Løsningen til likningen er x ≈ 1.732
```

Figur 3.1: Eksempel 19 i Mønster 1T (Raustøl, A. & Lund, H.-S., 2020, s. 263).

Aspekt ved algoritmisk tenkning: I denne oppgaven vil elevene måtte gå gjennom deler av koden trinn for trinn for å finne ut hva den gjør, for å vite hvordan de kan endre programmet til å også telle antall halveringer. Å systematisk analysere deler av kode på denne måten, er en del av evalueringsaspektet ved algoritmisk tenkning. I tillegg bes elevene lage og legge til denne utvidelsen av koden, slik at den teller antall halveringer. Denne delen av oppgaven, hvor elevene skal lage og legge til ny kode er algoritmebehandling. Dermed gir oppgaven mulighet for arbeid med både evaluering og algoritmebehandling.

Type resonnering: Oppgaven løses ved å legge en teller i while-løkken. Oppgaven er klassifisert som en oppgave som kan løses ved kreativ resonnering. I boka finnes det to eksempler med koder hvor man legger inn en variabel i en while-løkke, som teller hvor mange ganger løkken kjøres. Eksemplene finnes på s. 422 i Mønster 1T (Raustøl, A. &

Lund, H.-S., 2020, s. 422). I tillegg finnes tre nesten identiske oppgaver, basert på oppgavevariablene; oppgave 4.36, 4.95b og 4.95 c. Disse svært like oppgavene følger derimot *etter* denne aktuelle oppgaven. Derfor klassifiseres «reflekter og diskuter»-oppgaven på s. 263 i Mønster 1T som kreativ resonnering.

Fordi oppgaven kan løses ved kreativ resonnering regnes den som å kunne bidra til problemløsning. Det er derimot ikke problemløsning med tanke på matematikken. Matematikken er knyttet til telling, som er langt fra problemløsning på dette nivået. Det er hvordan få programmet til å telle som er problemet. Dermed kategoriseres oppgaven som kreativ resonnering innen programmering.

Matematisk kunnskapsområde: Matematikken i programkoden forut for oppgaven er knyttet til numerisk likningsløsning, og det matematiske kunnskapsområdet likninger og ulikheter. Derimot handler ikke oppdraget elevene gis i oppgaven om likninger. Elevene bes finne et tillegg til programkoden som kan få programmet til å telle hvor mange ganger while-løkken kjøres. I prosessen med å få finne ut hvordan programmet skal kunne telle gir elevene direkte mulighet til arbeid med formler og variabler. Det matematiske kunnskapsområdet blir derfor *annet matematisk kunnskapsområde.*

Mønster 1T, oppgave 4.36, 4.95 b og 4.95 c: Disse oppgavene kommer senere i boka enn *reflekter og diskuter, s. 263.* I disse oppgavene etterspørres det samme; *hvor mange halveringer må programmet foreta?* Det matematiske kunnskapsområdet og aspektene ved algoritmisk tenkning blir dermed identisk som i reflekter-og-diskuter-oppgaven på s. 263. Oppgave 4.36, 4.95b og 4.95c skiller seg dermed fra oppgaven over når det kommer til type resonnering. Fordi oppgavene likner på to eksempler og den *reflekter og diskuter, s. 263*, oppfylles kravene om kjent algoritmisk resonnering.

Mønster 1T, oppg. 1.47:
Skriv et program som trekker et tilfeldig heltall mellom 1 og 100 og lar brukeren gjette hvilket tall maskinen har trukket ut. Så lenge brukeren gjetter feil, skal programmet oppgi om brukeren har gjettet for høyt eller for lavt. Tips: Kommandoen `randint(0,10)` fra biblioteket `random` trekker ut et tilfeldig tall mellom 0 og 10.

Aspekt ved algoritmisk tenkning: Å skrive et program inngår i algoritmebehandling. Oppgaven inneholder flere bestillinger til programmet elevene bes skrive. Elevene må bryte ned informasjonen de får i oppgaven, og behandle dem hver for seg, før de setter sammen delene til svaret på oppgaven. Dette er dekomponering.

Type resonnering: For å kunne løse oppgaven trenger elevene å vite hvordan de importerer kommandoen `randint(0,10)` fra biblioteket random. Dette står beskrevet i programmeringsdelen bakerst i boka, under en egen overskrift «Eksterne bibliotek».

I tillegg trenger elevene å hente inn informasjon fra brukeren, via input-kommandoen, og å gjøre om dette til tall. Dette beskrives også i programmeringsdelen bakerst i læreboka.

Av matematisk informasjon trenger elevene å komme frem til en måte å få datamaskinen til å undersøke hvorvidt tallet som gjettes er høyere eller lavere enn tallet som er valgt ut, og hvordan man kan få programmet til å fortsette å la brukeren gjette nye verdier så

lenge man ikke har gjettet riktig (while-løkke). Boka presenterer både vilkår og while-løkker i programmeringskapittelet, og inkluderer et nyttig eksempel for elevene når de arbeider med oppg. 1.47:

Boka forklarer i programmeringskapittelet hvordan man kan få et program til å undersøke hvilken verdi som er størst, mellom to variabler. Eksempelet bruker koden

```
If tall1 > tall 2:
      Print(f'{tall1} er større enn {tall2}')
Else:
      Print(f'{tall1} et ikke større enn {tall2}')
```

(Raustøl, A. & Lund, H.-S., 2020, s. 417).

I tillegg kan eksemplene på numerisk likningsløsning gi noen hint om hvordan man bruker while-løkker til å fortsette å gjette, frem til man har funnet riktig løsning:

Ved å kombinere eksempelet fra s. 263 og informasjonen om vilkår kan elevene komme frem til løsningen. Derimot finnes det ingen oppgaver eller eksempler i boka hvor løsningen allerede eksisterer. Dermed kvalifiserer denne oppgaven til kreativ resonnering.

Matematisk kunnskapsområde: Denne programmeringsoppgaven knyttes til enkel aritmetikk, og å sammenlikne om en størrelse er større eller mindre enn en annen. Det er ikke en del av kompetansemålene i læreplanen til MAT09-01. Hadde dette vært en unplugged-aktivitet hvor en elev kunne gjettet tall, og den andre skulle svart om det tallet som var blitt trukket ut var høyere eller lavere, ville oppgaven blitt kategorisert som *annet matematisk kunnskapsområde.* Derimot bes elevene om å skrive et program som utfører algoritmen. Da kreves bruk av variabler for å få programmet til å fungere slik det er tenkt. Derfor kategoriserer jeg oppgavene som innen det matematiske kunnskapsområdet *formler og variabler.* Se mine to forslag til løsninger på oppgaven i figur 3.2.

Mine utarbeidede løsningsforslag til oppgave 1.47 i Mønster 1T, med to forskjellige fremgangsmåter:

```python
# -*- coding: utf-8 -*-
"""
Created on Sat Nov 11 15:08:52 2023

@author: tuvsev
"""

from random import randint

a = randint(1, 100)
b = int(input("Vi har valgt et tall mellom 1 og 100. Hvilket tall gjetter du at det er?"))



while a-b !=0:
    if a-b < 0:
        print("Tallet er lavere enn", b)
    if a-b > 0:
        print("Tallet er høyere enn", b)
    b = int(input("Gjett på nytt!"))

if a-b == 0:
    print("Riktig! Tallet var", b)

```

```python
# -*- coding: utf-8 -*-
"""
Created on Sat Nov 11 15:29:35 2023

@author: tuvsev
"""

from random import randint

a = randint(1, 100)
b = int(input("Vi har valgt et tall mellom 1 og 100. Hvilket tall gjetter du at det er?"))



while a!=b:
    if a < b:
        print("Tallet er lavere enn", b)
    if a > b:
        print("Tallet er høyere enn", b)
    b = int(input("Gjett på nytt!"))

if a == b:
    print("Riktig! Tallet var", b)

```

Figur 3.2 Løsningsforslag oppg. 1.47 I Mønster 1T

Mønster 1T, oppg. 5.27
La f være funksjonen gitt ved $f(x) = 3x^3 - 3x^2 + 1$. Lag et program som regner ut tilnærmingsverdi for f'(2).

Oppgaven ledsages av et eksempel med kode som regner ut f'(5) når $f(x) = x^2 + 3x + 6$.

Aspekt ved algoritmisk tenkning: Oppgaven gir mulighet for algoritmebehandling.

Type resonnering: Oppgaven klassifiseres som guidet algoritmisk resonnering. Eleven kan følge eksempelet, og bytte ut verdier med de aktuelle i oppgaven, for å endre funksjonen og x-verdien i den deriverte. Fremgangsmåten er gitt.

Matematisk kunnskapsområde: Oppgaven gir mulighet for arbeid med vekstfart og derivasjon.

3.4 Studiens kvalitet

Som forsker er det vanlig å gjøre en kvalitetsvurdering av studien sin, ved å stille spørsmål angående studiens validitet, reliabilitet og generaliserbarhet (Dalland, 2020). Med generaliserbarhet menes det at resultatene i en situasjon kan overføres til en andre situasjoner (Kvale & Brinkmann, 2021, s. 355). Innen kvalitativ forskning stilles det ikke krav til at forskningen skal være generaliserbar (Kvale & Brinkmann, 2021, s. 289). I denne studien er jeg interessert i det spesifikke tilfellet programmeringsoppgaver i lærebøker i MAT09-01, uten ambisjon om å generalisere til øvrige matematikkfag, eller matematikkbøker i skolen generelt. Derfor vil jeg i dette kapittelet holde meg til å redegjøre for studiens validitet og reliabilitet.

3.4.1 Validitet

I samfunnsvitenskapene handler validitet om i hvilken grad en metode er hensiktsmessig til å undersøke det den er tenkt å undersøke (Kvale & Brinkmann, 2021, s. 276). Validiteten vurderes ved å undersøke i hvilken grad empiri og analysen leder til resultater som er gyldige for å svare på forskningsspørsmålene (Grønmo, 2016). Analyseverktøyet og analyseprosessen i min oppgave er presentert i kapittel 3.3, og resultatene disse fører til finnes i kapittel 4. Resultatene viser mulighetene i programmeringsoppgavene, når det kommer til algoritmisk tenkning, type resonnering og tilknytninger til matematiske kunnskapsområder, og resultatene oppgis i form av frekvens, som antall oppgaver og prosentandel av oppgavene. Forskningsspørsmålene går ut på å finne muligheter for algoritmisk tenkning, kreativ resonnering og matematiske kunnskapsområder i programmeringsoppgavene, målt i både innhold og frekvens. Analyse og resultater er relevante for forskningsspørsmålene.

I kvalitativ forskning, hvor forskeren selv inngår i undersøkelsene, påvirker forskerens egne forventninger og forutinntatte ideer forskningens validitet (Cohen et al., 2007). I denne studien har jeg benyttet meg av begreper hentet fra læreplanen i MAT09-01. Deretter har jeg anvendt forskningsteoretiske definisjoner av begrepene. Der det ikke har vært tydelig hvordan begrepet kan forstås, har jeg lett i dokumenter som støtter læreplanen, for kildehenvisninger som kan gi svar. Jeg har ikke funnet en entydig definisjon av algoritmisk tenkning i læreplanen, og har derfor foretatt en tolkning, basert

på læreplanen og tilknyttede styringsdokumenter og deres kilder. Hele denne prosessen har jeg dokumentert i kapittel 2.2.3, for gjennomsiktighet. At jeg har holdt begrepsbruken nær læreplanen, som lærebøkene er en implementering av, er med på å styrke det som kan kalles studiens indre validitet. Med indre validitet menes i hvilken grad begrepene som anvendes står i samsvar med datamaterialet som studeres (Postholm & Jacobsen, 2018), her lærebøkene i MAT09-01. Allikevel har jeg foretatt en tolkning, hvor min egen forutinntatthet og erfaring kan ha vært med på å farge resultatene (Grønmo, 2016, s. 248). I neste kapittel redegjør jeg for egen bakgrunn, for å skape transparens og la leseren kunne reflektere over min rolle i forskningen.

3.4.2 Reliabilitet

Reliabilitet sier noe om hvor pålitelige forskningens funn er (Grønmo, 2016). Reliabiliteten knyttes til en refleksjon rundt hvordan forskeren eller analysen kan ha påvirket studiens resultater (Postholm & Jacobsen, 2018). Her vil jeg derfor redegjøre for egen bakgrunn, samt reflektere rundt analyseprosessen og analyseverktøyet som er anvendt i denne studien.

Motivet mitt med denne studien er å undersøke om programmeringsoppgavene som gis i lærebøkene gir elevene muligheter til arbeid som er i tråd med læreplanens intensjoner med innføringen av programmering i matematikkfaget. Svaret er av særlig interesse for meg, fordi jeg selv jobber i en videregående skole, og ukentlig skal velge ressurser å benytte meg av i planlegging og gjennomføring av egen undervisning. Jeg underviste i faget MAT09-01 sist i skoleåret 2022/2023. På skolen hvor jeg jobber, anvendes læreboken Sinus 1T (Maus, G. & Smestad, B.-T., 2020). Boken er allerede kjøpt inn til både lærere og elever ved skolen, og valget av læreverk ble gjennomført før jeg ble ansatt ved skolen. Jeg kjenner ingen av lærebokforfatterne som har bidratt i arbeidet av de tre bøkene som studeres i denne masteroppgaven, og jeg har ikke noen økonomisk, profesjonell eller personlig interesse av å skulle stille en av bøkene i et bedre lys enn de andre, eller på noen annen måte påvirke resultatene i denne studien.

Reliabiliteten er høy, dersom uavhengige målinger gir samme resultat (Guba, 1981, s. 81). I denne studien har jeg selv gjennomført lærebokanalysen to ganger, på to ulike tidspunkt i arbeidet med studien. Ønsket bak dette valget var å øke reliabiliteten til min studie. Som forsker vil analysen og fortolkningen jeg foretar meg påvirkes av min situasjon og erfaring (Grønmo, 2016, s. 248). At jeg gjennomfører analysen selv to ganger, kan derfor ikke kvalifiseres som to uavhengige målinger. Mine to analyserunder ga meg allikevel muligheten til å finne inkonsekvent koding i min egen analyseprosess, og reflektere over hvilke resultater jeg skulle beholde og hvilke jeg skulle forkaste.

Ved andre analyserunde fant jeg 7 avvik fra første analyse, når det kom til kategorisering av matematisk kunnskapsområde. Jeg sendte 3 av disse oppgavene til en kollega, med spørsmål om hvilken av de matematiske kunnskapsområdene hun syntes oppgaven tilhørte. En av oppgavene var reflekter-og-diskuter-oppgaven som ble presentert på s. 45. Opprinnelig hadde jeg kodet denne som en oppgave med mulighet for arbeid med likninger og ulikheter. Min kollega foreslo samme kategori som jeg hadde kommet frem til i min andre analyserunde, formler og variable størrelser. Ut fra dette har jeg konkludert med at jeg har plassert oppgaver i feil kategori i første omgang, og beholdt resultatene fra andre analyserunde.

Kanskje skyldes avvikene fra første analyserunde min forutinntatthet over hva jeg antok oppgavene kom til å handle om, basert på min kunnskap om temaet og kapittelet

oppgaven hørte til under, og at jeg kategoriserte i tråd med funn jeg forventet å finne, heller enn med et mer objektivt blikk på språket i oppgavene. Fordi analyseverktøyet for matematisk kunnskapsområde er laget på de vidt formulerte kompetansemålene i MAT09-01 kunne jeg ha økt reliabiliteten til oppgaven med å lage et tillegg til verktøyet, et ekstra sett med analysekriterier, for å gjøre det mer operasjonaliserbart. Jeg ville kunnet styrke reliabiliteten i oppgaveanalysene ved å få hele analysen gjennomført og samtlige oppgaver kategorisert av flere enn meg selv, og deretter sammenliknet resultatene. Det har det ikke vært rom for i gjennomføringen av denne studien, med kun én forfatter, i en begrenset periode med tid.

Analyseverktøyet anvender tydelige kriterier for oppgaveanalysen når det kommer til type resonnering, ved en presentasjon av det teoretiske rammeverket til Palm et al. (2005), med tilpasninger. Analyseverktøyet er også tydelig i kategorisering av aspekter ved algoritmisk tenkning, på grunn av beskrivelsene til Csizmadia et al. (2015) som er lagt ved Selby og Woollard (2013) sin definisjon.

I kapittel 5.6 diskuterer jeg metodiske og analytiske styrker og begrensninger ved denne studien, i lys av den gjennomførte analyseprosessen og funnene som gjøres. Disse refleksjonene rundt min påvirkning på datamaterialet, og synliggjøringen av forskningsprosessen, er en av måtene jeg viser reliabilitet på (Postholm & Jacobsen, 2018, s. 224).

4 Resultater

Her vil jeg presentere resultatene av lærebokanalysene. Først gjøres det rede for resultatene av den horisontale analysen, og deretter den vertikale analysen, med fokus på henholdsvis algoritmisk tenkning, type resonnering og matematiske kunnskapsområder. Til slutt i kapittelet oppsummeres analysens funn.

4.1 Horisontal analyse

Bakgrunnsinformasjon og overordnet struktur:

Lærebok	**Sinus 1T**	**Matematikk 1T**	**Mønster 1T**
Klassetrinn	VG1	VG1	VG1
Fag	MAT09-01	MAT09-01	MAT09-01
Tilgjengelige tilleggsressurser	Elevnettsted. Nettsted for læreren, med forslag til kapittelprøver og årsplan.	Elevnettsted. Nettsted for læreren, med forslag til kapittelprøver og årsplan. Forenklet lærebok for elevene.	Elevnettsted. Nettsted for læreren, med forslag til kapittelprøver og årsplan.
Utgiver og utgivelsesår	Cappelen Damm AS, 2020	H. Aschehoug & Co, 2020	Gyldendal Norsk Forlag AS, 2020
Totalt antall oppgaver	3307	2725	3105
Antall programmeringsoppgaver	28	32	87
Prosentandel programmeringsoppgaver	0,84%	1,17%	2,80%
Antall kapitler	7	8	7
Antall delkapitler	55	45	51
Gjennomsnittlig antall oppgaver per delkapittel	60	61	61
Antall utforsk-oppgaver	37	158	184
Antall utforsk-oppgaver med programmering	0	5	0

Tabell 4.1: Bakgrunnsinformasjon og overordnet struktur av de tre læreverkene Sinus 1T, Matematikk 1T og Mønster 1T

Samtlige av eksemplene i de tre lærebøkene skrives i det tekstbaserte programmeringsspråket Python. I programmeringsoppgavene spesifiseres det at elevene skal bruke Python, ellers nevnes det ikke noe spesielt programmeringsspråk. I enkelte oppgaver står det ikke en gang at elevene skal programmere, det står bare «Bruk Python til å…».

Ingen av programmeringsoppgavene inneholder bruk av konkreter eller unplugged-aktiviteter. Ingen av oppgavene som nevner programmering eller Python spesifikt, gir

elevene valget om å løse oppgaven på annet vis: Avgjørelsen om å bruke programmering er allerede tatt og lagt som et premiss i oppgavebeskrivelsen.

Alle lærebøkene inneholder en type oppgaver som de har valgt å kalle *Utforsk-oppgaver.* Mønster 1T spesifiserer ikke hva de mener med navngivningen av disse oppgavene (Raustøl, A. & Lund, H.-S., 2020). I Matematikk 1T presenteres disse oppgavene som en type oppgaver som «får elevene til å gå i dybden og se sammenhenger i faget» (Frydenlund, C. & Holst, L., 2020). Sinus 1T anvender utforsk-oppgavene i læreboken som en inngang til større utforskende opplegg som finnes på deres nettsider (Maus, G. & Smestad, B.-T., 2020). Jeg har kun studert den innledende delen av oppgavene, som finnes i læreboken. I Sinus 1T presiseres det at man nok ikke har tid til å arbeide med samtlige utforsk-oppgaver i løpet av et skoleår, men at «noe slikt arbeid må alle gjøre for å oppfylle kravene i læreplanen» (Maus, G. & Smestad, B.-T., 2020).

Kapitlenes struktur og rekkefølge:

Sinus 1T:	Matematikk 1T:	Mønster 1T:
1. Formler og likninger 1.1. Regnerekkefølge 1.2. Variabler og parenteser 1.3. Likninger og identiteter 1.4. To lineære likninger med to ukjente **1.5. Formler** 1.6. Rette linjer 1.7. Å finne likningen for ei linje 1.8. Digital graftegning 1.9. Grafisk avlesning 2. Faktorisering 2.1. Brøkregning 2.2. Kvadratsetningene 2.3. Faktorisering 2.4. Heltallsmetoden 2.5. Fullstendig kvadrat-metoden 2.6. Rasjonale uttrykk 3. Andregradslikninger 3.1. Funksjonsbegrepet 3.2. Andregradsfunksjoner 3.3. Grafisk løsning av andregradslikninger 3.4. Kvadratrøtter og røtter av høyere orden 3.5. Andregradslikninger **3.6. Andregradsformelen** 3.7. Faktorisering ved hjelp av nullpunktene 3.8. Ikke-lineære likningssett 4. Tredjegradslikninger og ulikheter 4.1. Lineære ulikheter 4.2. Andregradsulikheter 4.3. Polynomfunksjoner 4.4. Polynomdivisjon	1. Tall 1.1. Tallmengder **1.2. Tallmønstre** **1.3. Algoritmer** 1.4. Potenser 1.5. Store og små tall 1.6. Bevis 2. Algebra **2.1. Bokstavuttrykk** 2.2. Kvadratsetningene 2.3. Faktorisering 2.4. Faktorisering med kvadratsetningene 2.5. Brøkregning **2.6. Formler** 3. Likninger 3.1. Lineære likninger 3.2. Formelregning 3.3. Andregradslikninger **3.4. abc-formelen** 3.5. Ekvivalens ved løsning av likninger 3.6. Nullpunktfaktorisering 3.7. Polynomdivisjon 4. Funksjoner 4.1. Funksjonsbegrepet **4.2. Lineære funksjoner** 4.3. Polynomfunksjoner 4.4. Rasjonale funksjoner 4.5. Potensfunksjoner **4.6. Eksponentialfunksjoner** 4.7. Vekstfart 4.8. Den deriverte 5. Likningssystemer og ulikheter **5.1. Lineære likningssystemer** 5.2. Likninssystemer med flere enn to ukjente 5.3. Ikke-lineære likningssystemer 5.4. Lineære ulikheter 5.5. Polynomulikheter	1. Tall og regning 1.1. Tall, tallmengder og regning 1.2. Regning med brøk 1.3. Prosentregning 1.4. Vekstfaktor 1.5. Bevis og sammenhenger **1.6. Problemløsning** 2. Likninger og andregradsuttrykk 2.1. Førstegradslikninger 2.2. Kvadratsetningene 2.3. Fullstendig kvadrat 2.4. Andregradslikninger 2.5. Løsningsformel for andregradslikninger 2.6. Praktisk bruk av andregradslikninger 2.7. Faktorisering og forkorting 3. Funksjoner 3.1. Funksjonsbegrepet 3.2. Skjæringspunkter 3.3. Lineære funksjoner 3.4. Polynomfunksjoner 3.5. Eksponentialfunksjoner 3.6. Potensfunksjoner og rotfunksjoner 3.7. Rasjonale funksjoner 3.8. Modellering og regresjon 4. Algebra 4.1. Polynomdivisjon 4.2. Anvendelse av polynomdivisjon 4.3. Likningssystem 4.4. Ulikheter **4.5. Numerisk løsning av likninger** 5. Vekstfart og derivasjon 5.1. Vekst og vekstfart 5.2. Gjennomsnittlig vekstfart 5.3. Momentan vekstfart

4.5. Resten ved en polynomdivisjon 4.6. Faktorisering av polynomer 4.7. Likninger og ulikheter av tredje grad 4.8. Rasjonale likninger 4.9. Rasjonale ulikheter 5. Modeller og funksjoner 5.1. Lineære modeller 5.2. Lineær regresjon 5.3. Polynomregresjon 5.4. Potenser **5.5. Prosentregning** 5.6. Eksponentialfunksjoner 5.7. Potensfunksjoner 5.8. Rasjonale funksjoner 6. Vekstfart og derivasjon 6.1. Gjennomsnittlig vekstfart 6.2. Momentan vekstfart 6.3. Grenseverdier for ubestemte uttrykk 6.4. Vekstfart som grenseverdi 6.5. Derivasjon 6.6. Noen derivasjonsregler 6.7. Funksjonsdrøfting **6.8. Numerisk likningsløsning** 7. Trigonometri 7.1. Sinus og cosinus 7.2. Å finne vinkler 7.3. Tangens 7.4. Arealsetningen 7.5. Ubestemte trekanter 7.6. Sinussetningen 7.7. Cosinussetningen	5.6. Rasjonale ulikheter 6. Modellering 6.1. Matematiske modeller 6.2. Regresjon 6.3. Modellering i praksis 7. Trigonometri 7.1. Pytagorassetningen 7.2. Formlikhet 7.3. Tangens, sinus og cosinus 7.4. Generell definisjon av sin v og cos v 7.5. Arealsetningen 7.6. Sinussetningen 7.7. Cosinussetningen 8. Eksamenstrening 8.1. Uten hjelpemidler 8.2. Med hjelpemidler	5.4. Den deriverte – stigningstallsfunksjonen **5.5. Numerisk derivasjon** 5.6. Praktisk bruk av derivasjon 5.7. Derivasjon av polynomfunksjoner 6. Trigonometri 6.1. Trigonometriske forhold 6.2. Finne lengder i trekanter 6.3. Finne vinkler i trekanter 6.4. Enhetssirkelen 6.5. Arealsetningen 6.6. Sinussetningen **6.7. Cosinussetningen** 7. Innføring i Python 7.1. Digitalt verktøy for Python **7.2. Grunnleggende kommandoer i Python** **7.3. Variabler** **7.4. Datatyper** **7.5. Hente inn opplysninger fra brukeren** **7.6. Sannhetsverdier og vilkår** **7.7. Løkker – repeterende kode** **7.8. Funksjoner** **7.9. Lister** 7.10. Feilsøking og hjelp **7.11. Eksterne bibliotek**

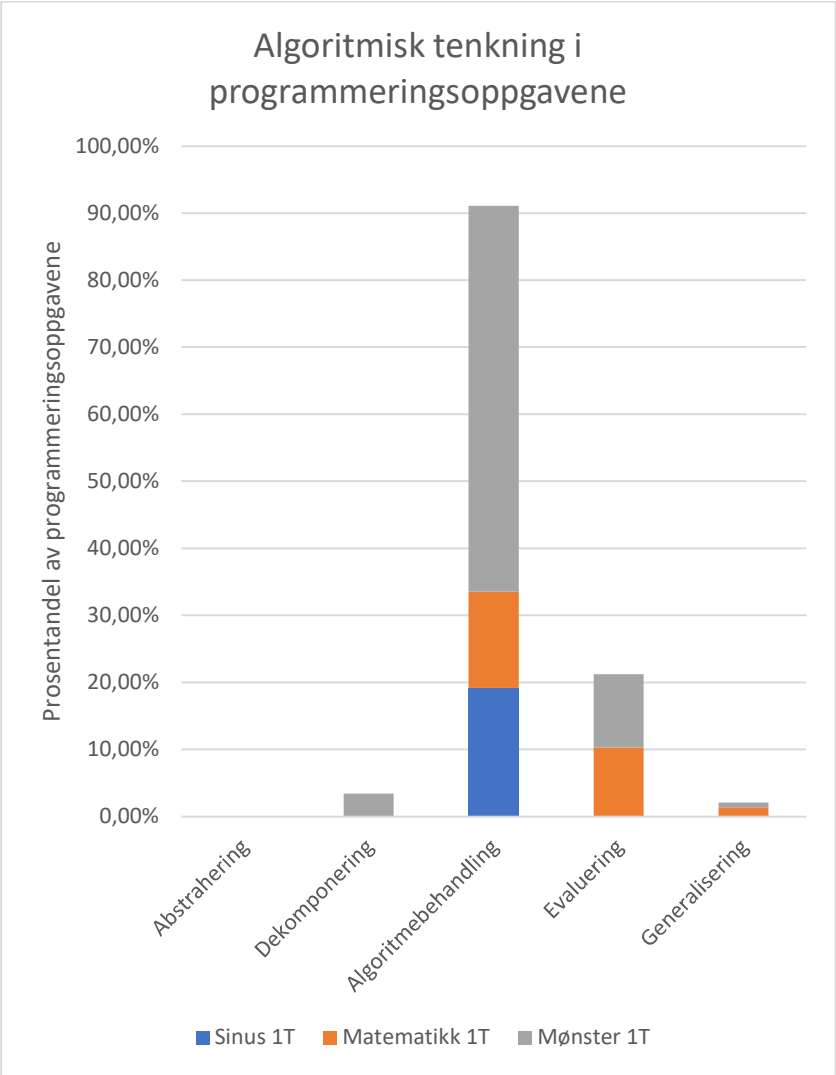
Tabell 4.2: Oversikt over struktur og rekkefølge på kapitlene i de tre lærebøkene.

Kapitlene som inneholder eksempler i programmering og/eller programmeringsoppgaver er uthevet med fet skrift.

4.2 Vertikal analyse

Her presenteres resultatene fra den vertikale analysen.

4.2.1 Algoritmisk tenkning

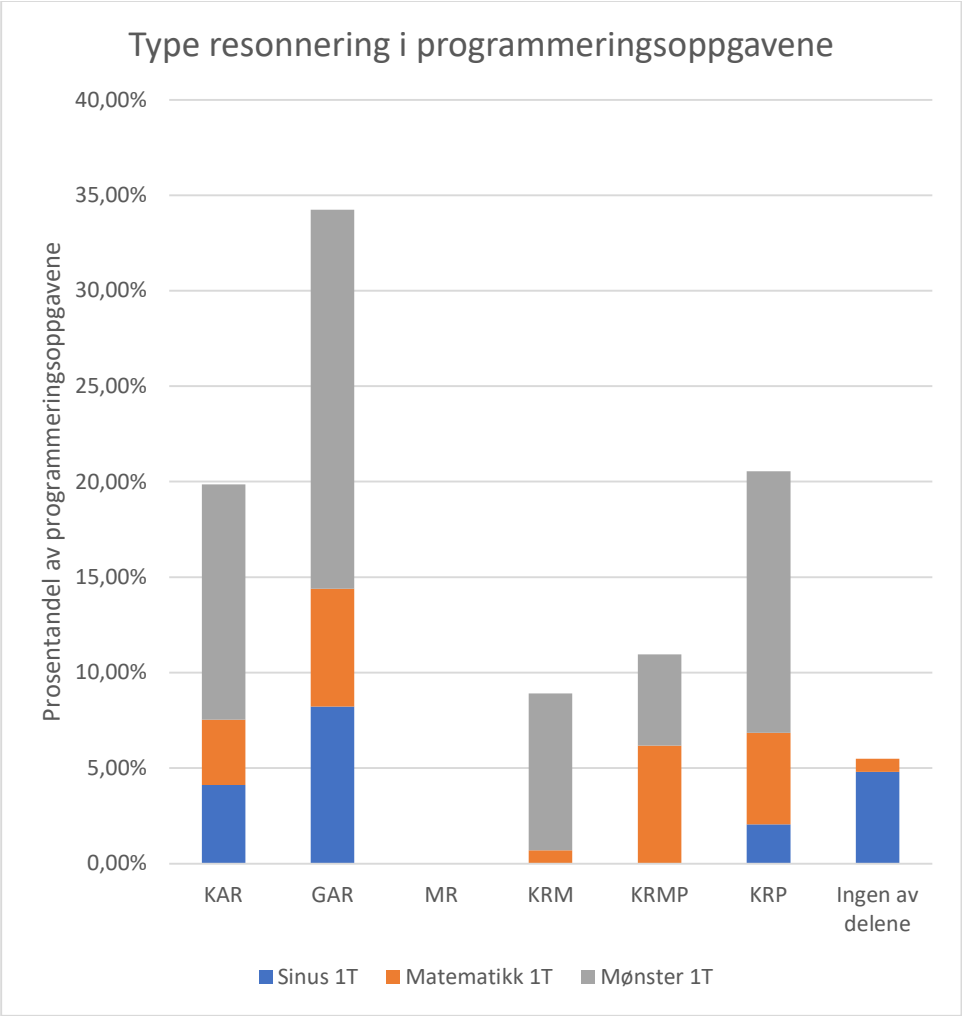


Figur 4.1: Aspekter ved algoritmisk tenkning i programmeringsoppgavene.

Som det fremkommer av figur 4.1 får elevene mulighet til å arbeide med en form for algoritmebehandling i hele 91% av programmeringsoppgavene. 21% består av evaluering, 3% består av dekomponering, 2% av generalisering. Ingen av programmeringsoppgavene gir elevene eksplisitt muligheten til å arbeide med abstrahering.

Det er også en skjevfordeling mellom lærebøkene. Figur 4.1 viser at mens Mønster og Matematikk 1T har oppgaver med aspektene algoritmebehandling, evaluering og generalisering, er programmeringsoppgavene i Sinus 1T utelukkende knyttet til algoritmebehandling. Dette betyr ikke at elevene ikke kan evaluere eller generalisere i arbeidet med programmeringsoppgavene i Sinus 1T, men det er ikke dette elevene blir bedt om å gjøre i oppgavene, et slikt arbeid nevnes aldri eksplisitt. Kun Matematikk 1T-boka har oppgaver som knyttes til dekomponering (3% av oppgavene).

4.2.2 Problemløsning



Figur 4.2: Type resonnering i programmeringsoppgavene

De fleste programmeringsoppgavene er løsbare ved en form for imitativ resonnering: 20% av programmeringsoppgavene kan løses ved bruk av kjent algoritmisk resonnering, mens 34% av programmeringsoppgavene kan løses ved bruk av guidet algoritmisk resonnering. 5% av de studerte oppgavene lar seg ikke løse basert på innholdskunnskapen presentert i den aktuelle læreboken. Resterende 40% av programmeringsoppgavene kan løses ved bruk av kreativ resonnering.

Av oppgavene som kunne løses ved bruk av kreativ resonnering identifiserte jeg hvorvidt denne resonneringen var knyttet til matematikk, programmering, eller en kombinasjon. Diagrammet under viser en oversikt over fordelingen mellom disse problemløsningsoppgavene knyttet til matematikk, programmering og en kombinasjon av disse.

	Antall oppgaver	Prosentandel av problemløsningsoppgavene med programmering
KRM	11	19,6%
KRMP	16	28,6%
KRP	29	51,8%

Tabell 4.3: Fordeling av problemløsningsoppgaver innen matematikk, programmering eller en kombinasjon

38% av programmeringsoppgavene (56 av 146 oppgaver) kan løses ved bruk av kreativ resonnering, og kvalifiserer som problemløsningsoppgaver. Av disse problemløsningsoppgavene gir 52% anledning til å bruke kreativ resonnering kun til programmeringsteknisk kunnskap 19,6% av oppgavene gir mulighet til kreativ resonnering i matematikk, mens 28,6% av oppgavene gir mulighet til kreativ resonnering i både matematikk og programmering. Totalt kan 48,2% av problemløsningsoppgavene bidra til kreativ resonnering innen matematikk. Dette medfører at totalen av alle programmeringsoppgavene i de tre studerte lærebøkene som kan bidra til kreativ resonnering innen matematikk er 18,3% (48,2% av 38%).

Av det totale antallet oppgaver i disse tre lærebøkene er 1,6% programmeringsoppgaver. 0,3% av det totale antallet oppgaver i lærebøkene er av typen som gir mulighet for kreativ resonnering og problemløsning knyttet til matematikk, i form av en programmeringsoppgave.

4.2.3 Matematiske kunnskapsområder

Matematisk kunnskapsområde	**Sinus 1T**	**Matematikk 1T**	**Mønster 1T**
Matematiske bevis	0% av oppgavene	6,23% (2/32)	1,1% (1/87)
Formler og variable størrelser.	32,1% av oppgavene (9/28)	34,4% (11/32)	27,6% (24/87)
Likninger og ulikheter	46,4% av oppgavene, hvorav samtlige handler om andregradslikninger (13/28) Hvorav numerisk likningsløsning: 28,6% av oppgavene (8/28)	18,8% (6/32) Hvorav numerisk likningsløsning: 0%	19,5% (17/87) Hvorav numerisk likningsløsning: 18,4% (16/87)
Funksjoner	0% av oppgavene	9,4% (3/32)	10,3% (9/87)
Modellering	17,9% av oppgavene (5/28)	0%	2,2% (2/87)
Identifisering og formulering av problemer	0% av oppgavene	3,1% (1/32)	1,1% (1/87)
Vekstfart og derivasjon	0% av oppgavene	0%	4,6% (4/87)
Polynomdivisjon	0% av oppgavene	0%	0%

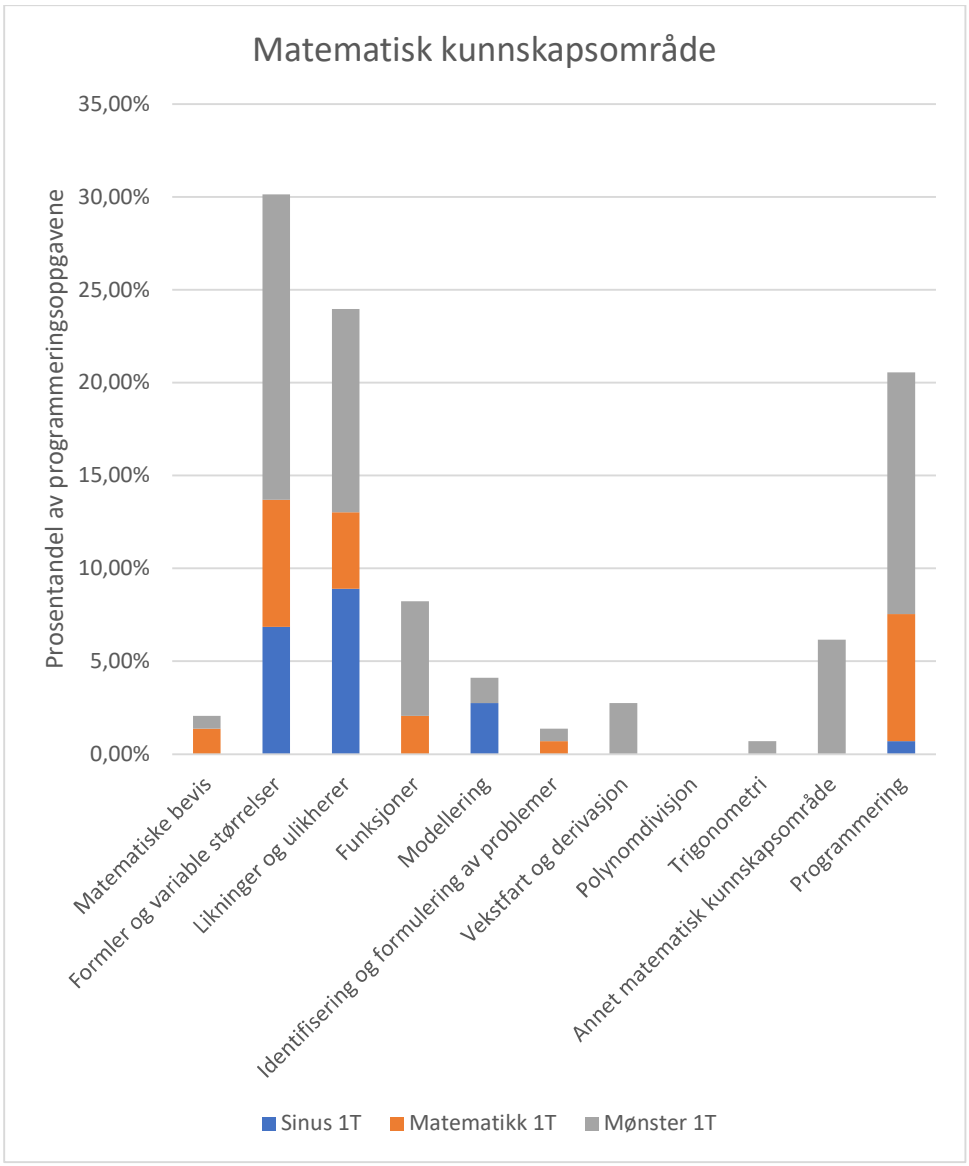
Trigonometri	0% av oppgavene	0%	1,1% (1/87)
Annet matematisk kunnskapsområde	0%	0%	10,3% (9/87)
Programmerings-teknisk, ikke knyttet til noe matematisk kunnskapsområde	3,6% av oppgavene (1/28)	28,1% (9/32)	22,8% (19/87)

Tabell 4.4: Programmeringsoppgavene fordelt på de matematiske kunnskapsområdene

Se tabell 3.1 for en oversikt over kompetansemålene som inngår i de ulike kunnskapsområdene.

80% av programmeringsoppgavene er knyttet til et matematisk kunnskapsområde, mens resterende 20% er rent programmeringstekniske. 6% av oppgavene som kan knyttes til matematikk handler derimot om tallregning og aritmetikk, og er på et slikt nivå at de ikke kan regnes å bidra mot målet for matematikkopplæringen i faget Matematikk 1T, fastsatt av kompetansemålene i faget. 74% av programmeringsoppgavene er knyttet til relevante matematiske kunnskapsområder for opplæring i faget Matematikk 1T.

Flesteparten av oppgavene knyttes til formler og variable størrelser, og likninger og ulikheter, 30% og 24% av alle oppgavene, respektivt.



Figur 4.3: Matematiske kunnskapsområder i programmeringsoppgavene.

Forskningslitteraturen viser ikke tydelig hvordan programmering kan bidra til økt kunnskap og kompetanse innen matematikk (Forsström & Kaufmann, 2018). Utdanningsdirektoratet anvender derimot et premiss i videoen i sin artikkel «Hva er nytt i matematikk?». Der sies det at *når* programmeringen i matematikkfaget brukes til problemløsning, kan det bidra til økt forståelse innen matematikk (Utdanningsdirektoratet, 2023). Det er derfor interessant å undersøke oppgavene som kan bidra til kreativ resonnering, i lys av hvilke matematiske kunnskapsområder de gir muligheter til arbeid med.

Figur 4.4: Matematiske kunnskapsområder i problemløsningsoppgavene med programmering.

Totalt er det 27 programmeringsoppgaver til sammen, i de tre undersøkte lærebøkene, som er av typen som kan løses ved bruk av kreativ resonnering innen matematikk, eller matematikk og programmering. Disse oppgavene er fordelt på 7 av de matematiske kunnskapsområdene; formler og variable størrelse, likninger og ulikheter, funksjoner, vekstfart og derivasjon, matematiske bevis, identifisering og formulering av problemer og modellering. Legger vi Utdanningsdirektoratets premiss om at problemløsende programmering bidrar til matematikklæring til grunn, betyr dette at programmeringsoppgavene i lærebøkene kan bidra til økte matematikk-kunnskaper innen disse syv temaene. Mens 22% av oppgavene kan bidra til økt forståelse innen likninger og ulikheter, ser vi av figur 4.6 at flesteparten av oppgavene, 55%, handler om formler og variable størrelser. Ingen av disse oppgavene handler om polynomdivisjon og trigonometri.

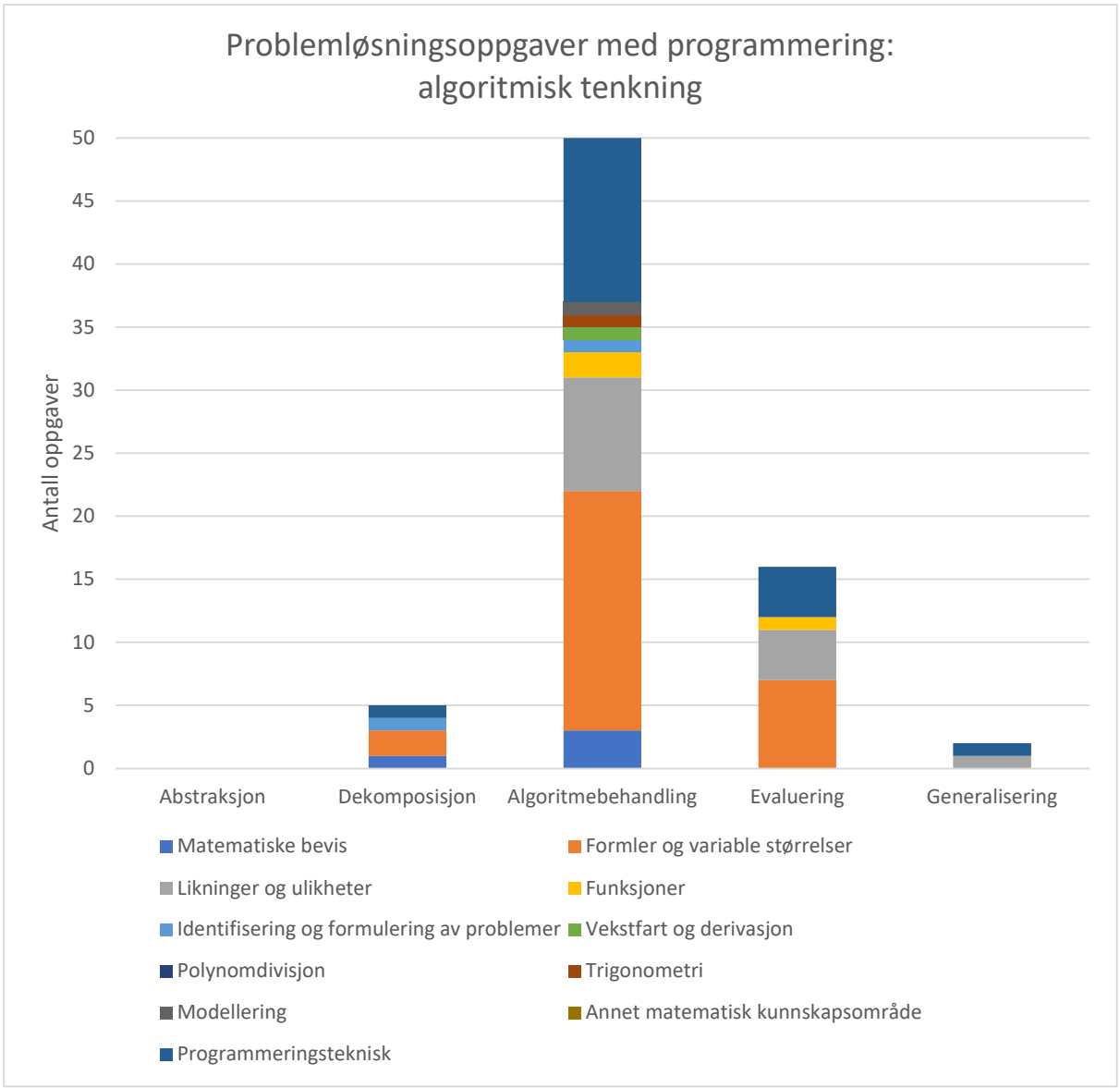
Matematisk kunnskapsområde	Antall programmeringsoppgaver
Fomler og variable størrelser	15
Matematiske bevis	1
Likninger og ulikheter	6
Identifisering og formulering av problemer	1
Funksjoner	2
Modellering	1
Vekstfart og derivasjon	1

Tabell 4.5: Oversikt over oppgavene som er løsbare ved bruk av kreativ resonnering, kategorisert etter matematisk kunnskapsområde

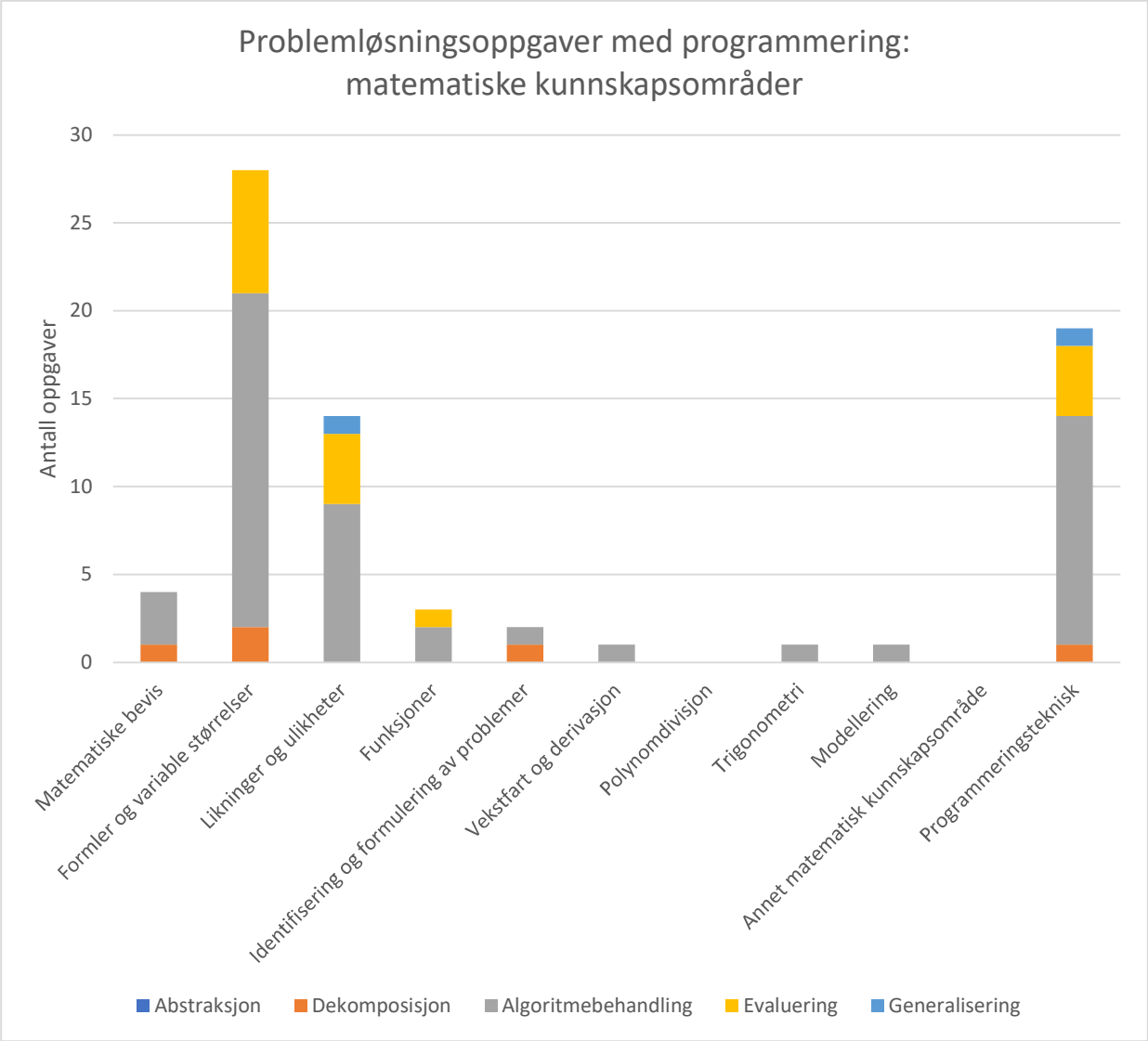
4.3 Oppsummering av funn fra analysen

Algoritmisk tenkning omtales både i forskningslitteraturen og i læreplanen til MAT09-01 som en problemløsningsstrategi (Weintrop et al., 2016; Kunnskapsdepartementet, 2019b). Derfor er det interessant å se på hvilke aspekter ved algoritmisk tenkning som kommer fram i de programmeringsoppgavene som kvalifiserer som problemløsningsoppgaver.

Antallet problemløsningsoppgaver blant de totale 146 programmeringsoppgavene i de tre lærebøkene til sammen, er på 59 oppgaver, eller 40,4%. Blant disse 59 oppgavene får elevene mulighet til å arbeide med følgende aspekter ved algoritmisk tenkning. Figur 4.8 viser oversikt over hvilke aspekter ved algoritmisk tenkning elevene får mulighet til å jobbe med, i arbeidet med de programmeringsoppgavene som kategoriserer som problemer. Figuren viser hvilke matematiske kunnskapsområder som inngår i arbeidet med disse oppgavene.



Figur 4.5: Algoritmisk tenkning i problemløsningsoppgavene i programmering



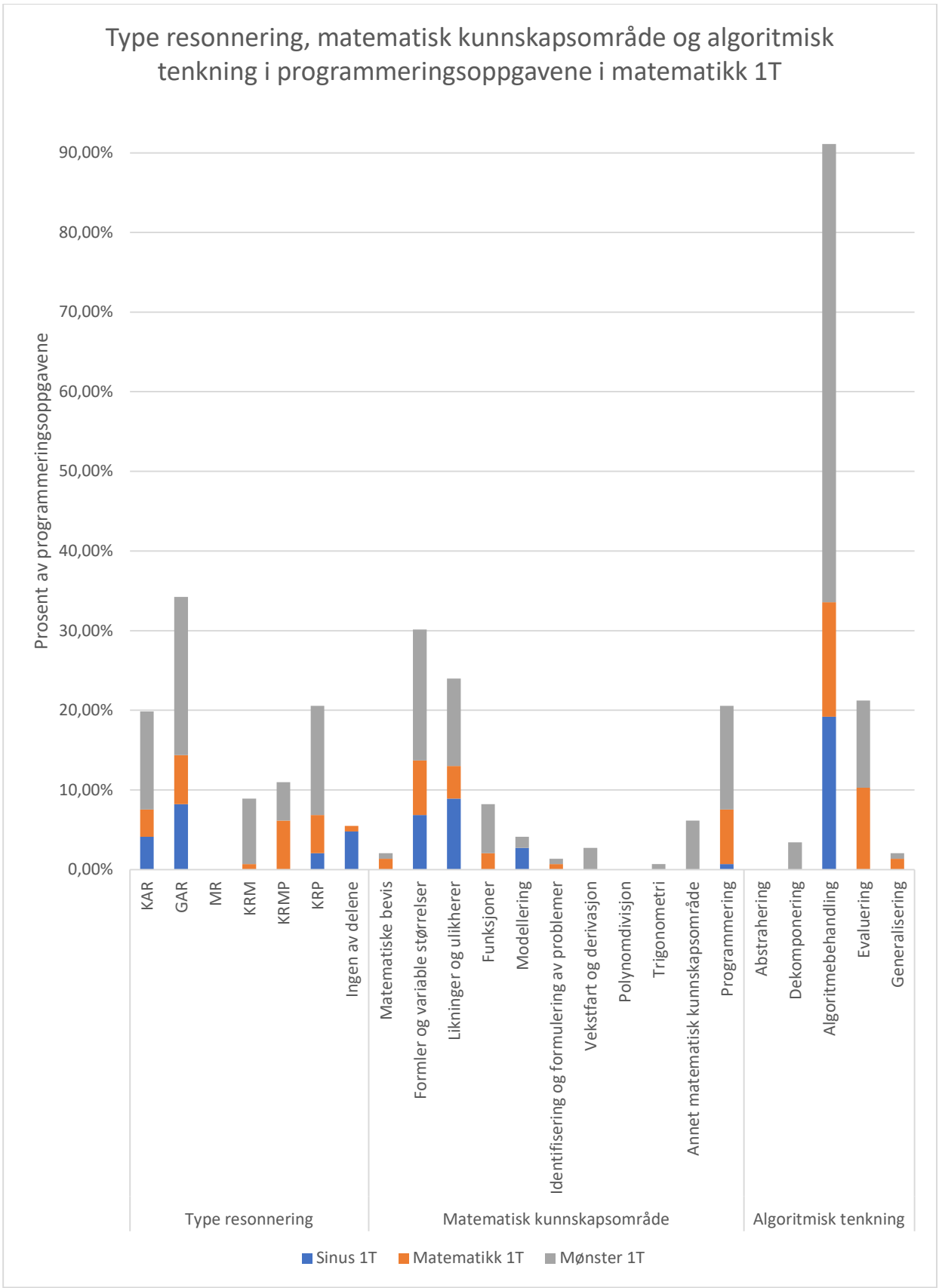
Figur 4.6: Matematiske kunnskapsområder i problemløsningsoppgaver med programmering

Figur 4.7 viser oversikt over hvilke matematiske kunnskapsområder elevene får mulighet til å jobbe med, i arbeidet med de programmeringsoppgavene som kategoriseres som problemer. Figuren viser også hvilke aspekter ved algoritmisk tenkning som inngår i arbeidet med disse oppgavene.

Det er i stor grad samtidig forekomst av algoritmebehandling og det matematiske kunnskapsområdet formler og variable størrelser, som det fremkommer av figur 4.7 og figur 4.8.

		Sinus 1T	Matematikk 1T	Mønster 1T
Type resonnering	KAR	6	5	18
	GAR	12	9	29
	MR	0	0	0
	KRM	0	1	12
	KRMP	0	9	7
	KRP	3	7	20
	Ingen av delene	7	1	0
Matematisk kunnskapsområde	Matematiske bevis	0	2	1
	Formler og variable størrelser	10	10	24
	Likninger og ulikheter	13	6	16
	Funksjoner	0	3	9
	Modellering	4	0	2
	Identifisering og formulering av problemer	0	1	1
	Vekstfart og derivasjon	0	0	4
	Polynomdivisjon	0	0	0
	Trigonometri	0	0	1
	Annet matematisk kunnskapsområde	0	0	9
	Programmerings-teknisk, ikke knyttet til noe matematisk kunnskapsområde	1	10	19
Algoritmisk tenkning	Abstrahering	0	0	0
	Dekomponering	0	0	5
	Algoritmebehandling	28	21	84
	Evaluering	0	15	16
	Generalisering	0	2	1

Tabell 4.6: Tabell som sammenfatter analysens funn, muligheter målt i antall oppgaver



Figur 4.7: Diagram som sammenfatter analysens funn, muligheter målt i prosentandel av programmeringsoppgavene

5 Diskusjon

I denne studien har jeg undersøkt hvilke muligheter programmeringsoppgavene i de trykte lærebøkene i MAT09-01 gir til å arbeide med algoritmisk tenkning, kreativ resonnering og matematiske kunnskapsområder. Som vist i kap. 4.1, er mulighetene få. Blant totalt 9137 oppgaver i de tre læreverkene til sammen, er 146 stykker, eller 1,6% av oppgavene, programmeringsoppgaver. I en læreplan hvor programmering inngår som en sentral del i 1 av 14 kompetansemål, er jeg overrasket over hvor få muligheter elevene gis til arbeid med programmeringsoppgaver i lærebøkene.

I dette kapittelet vil jeg diskutere resultatene av lærebokanalysene i lys av tidligere forskning og teori, som er definert i kapittel 2. Slik kan resultatene sammenliknes med tidligere forskning. Jeg vil først se nærmere på mulighetene for algoritmisk tenkning, deretter kreativ resonnering og problemløsning, og til sist hvilke muligheter programmeringsoppgavene gir for arbeid med matematiske kunnskapsområder. Mot slutten av dette kapittelet vil jeg drøfte styrker og svakheter ved valg av metode og studiens analytiske gjennomførelse, før jeg redegjør for mulige implikasjoner for praksis og videre forskning.

5.1 Muligherer for algoritmisk tenkning

Elevene får i stor grad muligheter til å jobbe med algoritmebehandling i programmeringsoppgavene i lærebøkene i MAT09-01. Hele 91% av programmeringsoppgavene gir muligheter til arbeid med algoritmebehandling. Elevene gis vesentlig færre muligheter til arbeid med de øvrige aspektene ved algoritmisk tenkning, og ingen av oppgavene i lærebøkene gir mulighet til arbeid med flere enn to aspekter ved algoritmisk tenkning av gangen.

Mønster 1T inkluderer ingen oppgaver i programmeringskapittelet med navn «feilsøking og hjelp». Dette er overraskende, siden programmeringen skal bidra til algoritmisk tenkning, og en viktig del av den algoritmiske tenkningen er evaluering (Selby & Woollard, 2013). Derimot er det i tråd med tidligere forskning på programmeringsoppgaver i lærebøker i matematikk (Bråting & Kilhamn, 2022).

Mitt teoretiske rammeverk inneholder ikke de samme kategoriene som Bråting og Kilhamns (2022). Dermed forsvinner muligheten til å drive en inndelt sammenlikning av funn på tvers av studiene. Derimot finner jeg likheter mellom kategoriene på tvers av studiene, se kap. 3.3.2. Oppgaver som enten ville kategoriseres som «feilsøking» eller «forestill deg»-oppgaver i Kilhamn og Bråting (2022) havner begge i «evaluering»-kategorien i min studie. Hvis jeg legger sammen kategoriene «feilsøke» og «forestill deg» i Bråting og Kilhamn (2022) vil jeg kunne se deres funn opp mot mine.

«Forestill deg» dukker opp i 8 av 390 oppgaver i Bråting og Kilhamns (2022) studie (s. 601), eller i 2% av oppgavene. «Feilsøke»-kategorien innehar mindre enn 20 av de 390 programmeringsoppgavene (Bråting & Kilhamn, 2022, s. 601). Nøyaktig tall oppgis ikke, men ut fra grafikken på s. 601 vil jeg anta det er snakk om 19 av 390 oppgaver, eller 4,8% (Bråting & Kilhamn, 2022, s. 601). Til sammen er altså 27 av 390 oppgaver, 6,9%

av oppgavene, av typen som kan sammenliknes med kategorien som i min studie kalles «evaluering».

I analysen av lærebøkene i MAT09-01 fant jeg muligheter for arbeid med evaluerings-aspektet ved algoritmisk tenkning i 21% av programmeringsoppgavene. Det er vesentlig større enn i et utvalg svenske lærebøker i matematikk på barnetrinn (Bråting & Kilhamn, 2022). At delkapittelet «feilsøking og hjelp» ikke gir elevene oppgaver å arbeide med, betyr altså ikke at elevene ikke får mulighet til å arbeide med evaluering gjennom programmeringsoppgavene i lærebøkene.

Selby og Woollards (2013) rammeverk viser at det er handlinger innen abstrahering, dekomponering, algoritmebehandling, evaluering og generalisering til sammen som utgjør den algoritmiske tenkningen. Som vist i resultatdelen av denne studien, i kapittel 4.2.1, gis elevene muligheter til arbeid med kun fire av disse fem områdene, i programmeringsoppgavene i lærebøkene. Det finnes ingen programmeringsoppgaver i lærebøkene i MAT09-01 som gir elevene muligheter for arbeid med abstrahering.

Det er viktig å påpeke at jeg kun har undersøkt programmeringsoppgavene i lærebøkene. Algoritmisk tenkning kan trenes uten å arbeide med programmering, og elever kan arbeide med programmering også i oppgaver som ikke krever det spesifikt, og som dermed ikke er undersøkt i denne studien. Derimot er det et tankekors at ingen av programmeringsoppgavene direkte knyttes til abstrahering. Skal elevene gis en direkte mulighet til å abstrahere, trengs oppgaver hvor det finnes en stor mengde data som elevene kan strukturere eller representere på en annen måte, overflødige detaljer som elevene kan skrelle vekk, eller andre typer oppgaver hvor eleven må arbeide med å finne og ta vare på det essensielle i oppgaven.

Ser vi tilbake til taksonomien over algoritmisk tenkning, av Weintrop et al. (2016), finner vi både datapraksiser og modellerings- og simuleringspraksiser som to av fire hovedkategorier i taksonomien. Dette tydeliggjør data og modellers viktige plass i algoritmisk tenkning i realfagene. Den typen oppgaver som er knyttet til store datasett og modellering er også oppgaver som vil gi muligheter til abstrahering. Når denne typen oppgaver ikke tilbys elevene, mister de også muligheten til å trene på abstrahering.

I læreplanen for MAT09-01 er modellering 1 av 14 kompetansemål, men også 1 av 6 kjerneelementer i faget. Modelleringens sentrale rolle i faget burde være tydelig. Så hvorfor mangler det programmeringsoppgaver knyttet til temaet? Som nevnt i teorikapittelet er det ikke spesifisert i de norske læreplanene for matematikk hvor og hvordan programmeringen skal anvendes (Munthe, 2022). Dermed overlates mange valg til læreren og lærebokforfatteren, når det kommer til implementeringen av programmering i matematikkfaget. Med en mer spisset formulering av læreplanen kunne en slik problematikk vært unngått. Som vi skal se i kapittel 5.3 og 5.5 kunne kanskje innholdet i læreplanen vært justert også, for å legge til rette for en integrering mellom programmering og matematikk i større grad.

Mønster 1T inkluderer 2 modelleringsoppgaver blant programmeringsoppgavene sine. Dette tilsvarer 1,4% av programmeringsoppgavene i de tre bøkene. Derimot har Mønster 1T, som eneste av de tre læreverkene, viet tre sider til en algoritme for hvordan man kan importere og behandle store datasett fra filer (Raustøl, A. & Lund, H.-S., 2020, s. 437-439). Elevene gis derfor mulighet til å følge algoritmen og laste inn og behandle store datasett som de eventuelt finner utenfor boken, i Mønster 1T. Denne inkluderingen vitner

om at også lærebokforfatterne anser det som nyttig at elevene har denne programmeringskunnskapen for programmering i matematikkfaget.

5.2 Muligheter for problemløsning

Algoritmisk resonnering går igjen i 54% av programmeringsoppgavene. Da algoritmebehandling er en del av algoritmisk resonnering, samsvarer dette funnet godt med mulighetene elevene gis til arbeid med algoritmisk tenkning. Der er det nettopp aspektet algoritmebehandling som forekommer hyppigst, i 91% av oppgavene. Jeg gjør oppmerksom på at en oppgave kan gi mulighet til arbeid med flere aspekter ved algoritmisk tenkning, mens oppgaven kun har blitt kategorisert som løsbar ved én type resonnering.

Den typen resonnering som elevene gis flest muligheter til arbeid med, er imitativ resonnering (54% av programmeringsoppgavene). Dette samsvarer godt med Berges (2022) funn om at de fleste programmeringsoppgavene i et utvalg lærebøker i MAT09-01 og matematikk R1 er ledsaget av et eksempel som elevene skal følge. Selv om det medfører at en mindre del av programmeringsoppgavene kan bidra til kreativ resonnering og problemløsning, så behøver ikke dette være problematisk. Som det er fremhevet i flere forskningsartikler har rutineoppgavene også sin plass i programmeringsoppgavene i matematikkfaget (Elicer & Tamborg, 2022; Berge, 2022).

Det som kan være problematisk er at andelen oppgaver som gir mulighet til kreativ resonnering innen matematikk (KRM og/eller KRMP) er på 32%. Når det er så få programmeringsoppgaver i lærebøkene i MAT09-01 i utgangspunktet, gis elevene veldig få muligheter til kreativ resonnering innen matematikk i programmeringsoppgavene. Det er snakk om 29 oppgaver, fordelt på tre lærebøker. I tillegg er det ikke sannsynlig at elevene arbeider med samtlige av oppgavene i lærebøkene. Hvilke muligheter elevene reelt får til å arbeide problemløsende i programmeringen i matematikk, avhenger av læreren, anvendelsen av læreboken, og hvilke øvrige ressurser som anvendes i undervisningen.

Konklusjonen på forskningsspørsmål 2 er at mulighetene for å arbeide med kreativ resonnering i tilknytning til programmering og matematikk i hver enkelt lærebok er få. De oppgavene som gir mulighet til problemløsning innen matematikk er knyttet til de matematiske kunnskapsområdene formler og variable størrelser, likninger og ulikheter, funksjoner, vekstfart og derivasjon, matematiske bevis, identifisering og formulering av problemer og modellering. Se figur 4.6 og tabell 4.5.

Hver av lærebøkene inkluderer en type oppgaver de har valgt å kalle for utforsk-oppgaver. Siden algoritmisk tenkning omtales i kjerneelementet «Utforsking og problemløsing» i læreplanen for MAT09-01, og algoritmisk tenkning er tett sammenkoblet med programmering i Utdanningsdirektoratets tekster (Utdanningsdirektoratet, 2019), forventet jeg å finne programmeringsoppgaver blant disse utforsk-oppgavene. Denne forventningen ble ikke innfridd. Av de tre læreverkene er det kun Matematikk 1T som inkluderer programmeringsoppgaver som en del av utforsk-oppgavene, se tabell 4.1.

Mønster 1T og Sinus 1T har ikke valgt å løse det slik, og tilbyr ingen muligheter til arbeid med utforsk-oppgaver og programmering samtidig. I denne studien har jeg ikke undersøkt hvilke aspekter ved algoritmisk tenkning elevene gis mulighet til arbeid med i utforsk-oppgavene som ikke er av typen programmeringsoppgave. Dermed er det mulig

at alle tre læreverk har integrert algoritmisk tenkning og utforsk-oppgaver, slik jeg forventer når jeg leser læreplanen, selv om de har holdt programmeringen utenfor. Som vist i figur 2.1 og 2.2, behøver ikke programmering å være en del av algoritmisk tenkning. Matematikk 1T er det eneste av disse læreverkene som har markert oppgaver de mener skal løses ved hjelp av algoritmisk tenkning. Flesteparten av disse er programmeringsoppgaver.

Programmeringsoppgavene inkluderer ingen oppgaver som er løsbare ved bruk av memorert resonnering. Antakeligvis forklares dette ved at programmeringsoppgavene i denne oppgaven i stor grad ber elevene lage programmer, og at denne typen oppgave ikke inneholder svar som er enkle å memorere. Programmene er oppbyggede koder, som kan la seg kopiere, men da heller ved å imitere en fremgangsmåte, slik at oppgaven kategoriseres som kjent eller guidet algoritmisk resonnering. Det er lite sannsynlig at elevene husker programkode som et svar, uten å ha koden som visuell støtte når svaret skal skrives ned.

Sinus 1T inneholder 7 oppgaver som ikke er løsbar med noen av de typene resonnering som presenteres i rammeverket. Årsaken til dette er at programmeringskunnskapen elevene trenger for å løse oppgaven ikke er inkludert i boken. Her er det viktig at læreren supplerer med nødvendig informasjon, eller benytter seg av flere kilder i planleggingen og gjennomføringen av undervisningen i programmering for matematikkfaget.

I neste kapittel vil jeg diskutere hvilke muligheter disse matematiske problemløsningsoppgavene gir elevene med tanke på læring av de matematiske kunnskapsområdene som er pekt ut for MAT09-01.

5.3 Muligheter for arbeid med matematiske kunnskapsområder

Som vi ser av tabell 4.2, varierer det fra lærebok til lærebok i hvilket delkapittel programmeringsoppgavene er plassert. Dette er ikke overraskende, da det ikke finnes klare føringer i læreplanen for MAT09-01 på hvor i matematikken man skal arbeide med programmering. Allikevel kommer det til syne flere likheter mellom læreverkene.

Både Sinus 1T og Mønster 1T inkluderer oppgaver med numerisk likningsløsning. Dette til tross for at numerisk likningsløsning ikke nevnes eksplisitt i læreplanen for MAT09-01. Den store andelen av oppgavene som kategoriseres innunder dette temaet, 28,6% og 18,4% i Sinus 1T og Mønster 1T respektivt, vitner om at lærebokforfatterne har funnet det hensiktsmessig å koble programmering til matematikken innen dette matematikkemnet.

Blant de øvrige matematiske kunnskapsområdene fant jeg svake sammenhenger mellom matematikken og programmeringen, slik som Bråting og Kilhamn (2022) observerte det også i en svensk barneskolekontekst (s. 607). For eksempel er kompetansemålet som danner grunnlaget for kategorien *formler og variable størrelser*, vidt, og kan tolkes til å omfavne mange av programmeringsoppgavene i de tre lærebøkene. Kompetansemålet lyder som følger: Mål for opplæringa er at elevene skal kunne […] identifisere variable størrelser i ulike situasjoner, sette opp formler og utforske disse ved bruk av digitale verktøy (Kunnskapsdepartementet, 2019). Som vist i kapittel. 3.3.5 har flere av programmeringsoppgavene i lærebøkene blitt plassert i denne kategorien, til tross for at matematikken har vært svært enkel, og omhandlet enkel aritmetikk. Så fort en oppgave har krevd at elevene anvender variabler og en formel, om enn en enkel en, så har

oppgaven havnet under kategorien *formler og variable størrelser.* Dersom elevene skulle løst oppgavene uten bruk av programmering ville de derimot ikke blitt klassifisert som innenfor kompetansemålene i MAT09-01 over hodet, men som matematikkoppgaver på et langt lavere nivå. Derfor kan det diskuteres hvor egnede disse oppgavene er til å lære matematikk på et nivå som elevene skal i faget MAT09-01. Dersom variabler og formler er så enkle at de ikke dukker opp noe annet sted i lærebøkene enn i oppgavene med programmering, er dette da den mest hensiktsmessige måten å knytte sammen programmering og matematikk på, på dette nivået? Programmeringen bør tilføre matematikken noe (Munthe, 2022). Som lærer og forsker ville det vært enklere å vurdere hvilke muligheter en programmeringsoppgave gir for matematikken lik læreplanens definisjon, dersom matematikken i læreplanen var mer konkret definert.

Hver av de studerte lærebøkene i MAT09-01 inkluderer en type oppgaver de har valgt å kalle for utforsk-oppgaver. Siden algoritmisk tenkning omtales i kjerneelementet «Utforsking og problemløsing» i læreplanen for MAT09-01 (Kunnskapsdepartementet 2019b), og algoritmisk tenkning er tett sammenkoblet med programmering i Utdanningsdirektoratets tekster (Utdanningsdirektoratet, 2019; Utdanningsdirektoratet, 2023), forventet jeg å finne programmeringsoppgaver blant disse utforsk-oppgavene. Denne forventningen ble ikke innfridd. Av de tre læreverkene er det kun Matematikk 1T som inkluderer programmeringsoppgaver i enkelte av utforsk-oppgavene, se tabell 4.1.

Bråting og Kilhamn (2022) fant at dersom lærebøkene de undersøkte på svensk barnetrinn i matematikk hadde tilbudt flere oppgaver av typen «utforsk», «forestill deg seg» og «feilsøke», så ville elevene hatt større mulighet til å utnytte sammenhenger mellom programmering og matematikkfaget til å bedrive matematikklæring (s. 607). Det samme uutnyttede potensialet kan sies å finnes i lærebøkene for MAT09-01.

19% av programmeringsoppgavene av typen problemløsningsoppgave med matematikk. Jeg opplever et uutnyttet potensial i disse oppgavene. Modellering er et av temaene som finnes i krysningen mellom algoritmisk tenkning og matematisk tenkning (Shute et al., 2017, s. 145). Blant temaene som jeg kategoriserer som matematiske kunnskapsområder i min studie, finnes ingen andre i dette krysningsfeltet (se kap. 2.4.3 og kap. 3.3.4). Modellering er et sentralt fagområde i MAT09-01, og modellering og anvendelser er et av 6 kjerneelementer i MAT09-01. Modelleringen er med andre ord viktig både i algoritmisk tenkning og matematisk tenkning, herunder også i faget MAT09-01. Derfor overrasker det meg at det finnes så få modelleringsoppgaver blant programmeringsoppgavene i lærebøkene i MAT09-01.

Blant oppgavene i de trykte lærebøkene i MAT09-01 finner vi muligheter for arbeid med matematisk modellering i kun 5% av oppgavene. Ser vi nærmere på fordelingen av disse oppgavene blant de tre lærebøkene, ser vi at elevenes muligheter for arbeid med modellering i programmeringsoppgaver i stor grad avhenger av anvendt lærebok: Mens Mønster 1T gir muligheter for arbeid med modellering i 2,2% av programmeringsoppgavene, og Matematikk 1T ikke tilbyr noen programmeringsoppgaver med mulighet for arbeid med modellering, gir Sinus 1T mulighet for arbeid med modellering i 14,3% av programmeringsoppgavene i boken. Dette er mer i tråd mine forutinntatte antakelser, etter å ha lest Shute et al. (2017) sin modell over krysninger mellom matematisk tenkning og algoritmisk tenkning (s. 145), og læreplanen i MAT09-01, hvor 1 av 6 kjerneelementer, eller 16,7% av kjerneelementene, omhandler modellering.

Jeg mener de få mulighetene elevene får til arbeid med modellering i programmeringsoppgavene i de studerte lærebøkene kan ses i sammenheng med fraværet av oppgaver som gir elevene muligheter til å jobbe med aspektet abstrahering, innen algoritmisk tenkning. Det gis ikke programmeringsoppgaver med reelle datasett som kan representeres på annet vis, komplekse modeller som skal forenkles, eller oppgaver hvor elevene må hente ut det essensielle blant overflødige detaljer. Dette er en sentral arbeidsmåte i arbeidet med modellfremkallende aktiviteter, hvor elevene selv lager modeller over en situasjon (Lesh et al., 2003). Dersom elevene var blitt gitt muligheten til å arbeide med denne typen oppgaver blant programmeringsoppgavene i lærebøkene, ville de fått muligheten til å jobbe med abstrahering, og modellering. I tillegg ville det være en type oppgave hvor programmeringen kunne vært et hensiktsmessig verktøy for læring av matematikk, i større grad enn det fremstår i programmeringsoppgavene i lærebøkene i dag.

I de tre læreverkene er programmering utelatt fra kapitlene om trigonometri, polynomdivisjon og vekstfart og derivasjon, med unntak av 5 oppgaver (3,4 % av alle programmeringsoppgavene i de tre læreverkene) i boken Mønster 1T. Denne enigheten mellom forlagene om å tilby ingen eller svært få programmeringsoppgaver innen disse temaene, til tross for at disse tre matematiske kunnskapsområdene representerer 5 av 14 kompetansemål i faget, tror jeg ikke er tilfeldig. Når læreplanen ytrer at elevene skal tilegne seg kompetanse innen disse kunnskapsområdene er ikke det ikke gitt at det er programmering som er verktøyet elevene best kan tilegne seg denne kompetansen med.

Mønster 1T og Sinus 1T inneholder en stor andel oppgaver hvor elevene gis muligheten til numerisk likningsløsning og numerisk tilnærming til eksakte verdier, som $\sqrt{3}$. Læreplanen nevner ikke numeriske metoder med ett ord. Slik jeg ser det tolker læreverkene kompetansemål 4 om likningsløsning til å også skulle inneholde numeriske metoder. Årsaken til at de tolker det dithen faller sannsynligvis sammen med implementeringen av programmering i matematikkfaget, da numeriske metoder ikke er en del av læreverkene forut for innføring av Kunnskapsløftet 2020. Jeg antar at numeriske metoder innføres i lærebøkene fordi dette er et emne innen matematikken hvor programmeringen kan «tilføre matematikken noe» (Munthe, 2022). Det finnes altså matematiske temaer hvor programmeringen kan bidra til matematikklæring i større grad enn det ser ut til at den gjør i programmeringsoppgavene slik de presenteres i dagens lærebøker i MAT09-01. Kanskje er dette knyttet til oppgavene, og at de ikke omfavner mulighetene som for eksempel kunnskapsområdet modellering gir. Kanskje kunne også programmeringen gitt større muligheter for læring av matematisk kompetanse og kunnskap, dersom matematikkinnholdet i læreplanen hadde vært annerledes?

5.4 Metodiske og analytiske styrker og begrensninger

En viktig del av forskningsprosessen går ut på å være kritisk til metodevalg og kvaliteten på empirien i forskningen (Grønmo, 2016, s. 237). Her vil jeg diskutere styrker og begrensninger knyttet til denne studiens metode og analyse.

5.4.1 Diskusjon omkring metoden

Lærebokanalysen som metode lar meg undersøke mulighetene elevene *kan* få til å lære algoritmisk tenkning, problemløsning og matematiske kunnskapsområder. Derimot sier ikke lærebokanalysen noe om muligheter elevene *faktisk* får i undervisningen. Lærebokanalysen er kun en hypotetisk analyse. Den sier heller ikke noe om det eventuelle læringsutbyttet elevene sitter igjen med etter endt undervisning.

I forskningsprosjektet «Programmering for å forstå matematikk» planlegges det å undersøke hvordan lærebøkene anvendes i praksis av læreren når det kommer til programmering i matematikkfaget (Berge, 2022, s. 8). I fremtiden vil vi altså sannsynligvis ha noen flere svar knyttet til bruken av læreboken, men vi vil allikevel ikke vite spesifikt hvordan den enkelte lærer anvender læreboka i sin undervisning. Lærebokanalysen som metode vil fortsette å være hypotetisk (Pingel, 2010).

Denne studiens rammeverk er utviklet for å undersøke hvilke muligheter programmeringsoppgavene i lærebøkene i MAT09-01 gir, med tanke på læreplanens intensjon om at programmeringen skal bidra til algoritmisk tenkning, problemløsning og læring av matematikk.

Et aspekt ved rammeverket, slik det utvikles for, og anvendes i denne studien, er oppgavenes koblinger til algoritmisk tenkning. Selby og Wollards (2013) rammeverk for algoritmisk tenkning blir noe grovinndelt for å kunne beskrive mulighetene som ligger i programmeringsoppgavene i lærebøkene i MAT09-01 når det kommer til utvikling av algoritmisk tenkning. Som vist i kapittel 5.2 er rammeverket tilstrekkelig for å konkludere med at programmeringsoppgavene alene, slik de fremstår i lærebøkene i dag, ikke gir elevene muligheter til å jobbe eksplisitt med samtlige aspekter ved algoritmisk tenkning, etter Selby og Woollards (2013) definisjon.

En annen inndeling og anvendelse av et annet rammeverk for algoritmisk tenkning kunne gitt er mer nyansert bilde av programmeringsoppgavenes tilknytning til mer findelte aspekter ved algoritmisk tenkning. Det kan for eksempel tenkes at bruken av Weintrop et al. (2016) sitt rammeverk kunne gitt flere svar. Som vist i kapittel 2.2.1 har Weintrop et al. (2016) sitt rammeverk flere kategorier, basert på arbeidsmetoder observert i realfagsundervisning, og disse er igjen inndelt i flere undernivåer (s. 135) Slik ville de 133 oppgavene som i denne studien havnet under kategorien «algoritmebehandling» kunne gitt mer informasjon, for eksempel. Er disse oppgavene av typen «les og forstå en kode», «lag en kode», «lag en fremgangsmåte», «følg en algoritme», «finn et mønster», eller annet? Modifiseringer i rammeverket ville gitt en annen innsikt.

Valget falt på Selby og Woollards definisjon av algoritmisk tenkning, fordi denne studien tolker læreplanens bruk av begrepet algoritmisk tenkning til å sammenfalle med Selby og Woollards (2013) definisjon, se kapittel 2.2.7. Valget om å anvende samme definisjon i denne studien bygger på ønsket om transparens, og å undersøke hvilke muligheter programmeringsoppgavene gir elevene til problemløsning, matematikklæring og algoritmisk tenkning, lik min tolkning av læreplanens hensikt. Fordi ordlyden i læreplanen er sentral i studiens forskningsspørsmål var det viktig å anvende samme teoretiske definisjon på algoritmisk tenkning som den jeg tolker er brukt av Utdanningsdirektoratet i artikler på www.udir.no og i læreplanen. Se kapittel 2.2.4. På den måten kan man si at studiens rammeverk er vellykket.

Læreplanen i matematikk skiller ikke på definisjonen av algoritmisk tenkning i grunnskolen eller på videregående nivå. Det er forskjell på første klasse i grunnskolen og første klasse på videregående skole. Det kunne vært interessant å analysere oppgavene i MAT09-01 med rammeverket til Weintrop et al. (2016) også, for å undersøke om dette ville gitt en annen innfallsvinkel på informasjonen om programmeringsoppgavenes tilknytning til algoritmisk tenkning. Weintrop et al. (2016) sin forskning, som ledet frem til deres rammeverk, er utført på universitetsnivå. Ved å anvende både Selby og Woollard (2013) og Weintrop et al. (2016) sine definisjoner, kunne perspektivene fra respektivt grunnskolen og universitetsnivået muligens bidratt til interessante funn som

kan belyse programmeringsoppgavene i MAT09-01, som befinner seg mellom nettopp grunnskolenivå og universitetsnivå.

Ikke alle kategoriene av algoritmisk tenkning inneholdt oppgaver, etter endt gjennomføring av lærebokanalysen. Ingen av oppgavene ble plassert i kategorien «abstrahering». Jeg har allikevel beholdt kategorien, for å tydeliggjøre forskjellen mellom teorien, og mangelen på muligheter elevene får til å jobbe med dette aspektet av algoritmisk tenkning i lærebøkenes programmeringsoppgaver.

Kilhamn og Bråting (2022) fant i sin studie at flere av programmeringsoppgavene i lærebøkene de analyserte ikke passet inn i noen av kategoriene i sitt teoretiske rammeverk, og ble derfor nødt til å gjøre endringer i rammeverket for å få kategorisert samtlige oppgaver. Dette var ikke nødvendig for aspektene ved algoritmisk tenkning i min studie. Derimot har jeg inkludert to kategorier for å kategorisere programmeringsoppgavene etter tilknytninger til matematiske kunnskapsområder, for å favne om også oppgavene som ikke ga muligheter til matematikklæring innen læreplanens oppgitte temaer for faget MAT09-01. I tillegg har jeg lagt til kategoriene KRM, KRMP og KRP i analyseverktøyets del om type resonnering. Det er hensiktsmessig for min studie å skille mellom disse tre typene kreativ resonnering, for å kunne identifisere mulighetene elevene får til kreativ resonnering og matematikklæring i programmeringsoppgavene.

Kilhamn og Bråting (2022) påpeker at en årsak til at man trenger å endre på rammeverket underveis i prosessen med oppgaveanalyse av lærebøker, kan være at rammeverket er basert på teori, mens oppgavene i praksis kan være av en annen type (s. 599). Mens forskerne i teorien presenterer en «ideell situasjon», der elevene helst burde fått mulighet til å engasjere seg i og tilegne seg kunnskap innen, så inneholder lærebøkene også andre typer oppgaver, som også trenger kategoriseringer (Bråting & Kilhamn, 2022, s. 599). Innen aspektene ved algoritmisk tenkning vises en tydelig forskjell fra den ideelle situasjonen beskrevet i teorien, og det jeg finner i bøkene i MAT09-01, når lærebøkene ikke inneholder noen programmeringsoppgaver som gir mulighet for abstrahering.

Selv om programmering etter læreplanens intensjon skal bidra til problemløsning, kan ikke alle oppgavene være problemløsningsoppgaver, for eksempel. Det er nødvendig med rutineoppgaver for innlæring av programmering også (Berge, s. 8). Selv om denne typen oppgaver ikke er læreplanens mål, er de et nødvendig steg på veien til målet.

I min studie undersøker jeg, i likhet med Kilhamn og Bråting (2022), samtidig forekomst av aspekter ved algoritmisk tenkning og matematiske kunnskapsområder. I kapittel 4.2.3 presenterer jeg fordelingen av programmeringsoppgavene etter hvilket matematisk kunnskapsområde de kan tilknyttes. Disse resultatene er kun koblet til samtidig forekomst av matematiske kunnskapsområder, og oppgaver som kan eller skal løses ved bruk av programmering. Hvorvidt programmeringsoppgavene bidrar til økt forståelse av disse matematiske kunnskapsområdene forteller denne samtidige forekomsten ikke noe om.

Derimot mener jeg at undersøkelsene gjort rundt type resonnering og problemløsning kan ta min studie et steg videre fra Bråting og Kilhamns (2022). I tillegg til å studere samtidig forekomst av algoritmisk tenkning og matematisk kunnskapsområde i programmeringsoppgavene, undersøker jeg hvorvidt oppgavene gir mulighet til å bedrive

kreativ resonnering. Dette gir innsikt i hvilke oppgaver som gir elevene mulighet til å trene på problemløsning.

De programmeringsoppgavene som gir elevene mulighet til å trene på problemløsning vil, ifølge Utdanningsdirektoratet (2023), kunne bidra til økt forståelse for matematikken i oppgavene. Ved å også undersøkte type resonnering mener jeg at min studie har en styrke, selv om den undersøker samtidig forekomst, som Elicer og Tamborg (2022) kritiserer i sin artikkel.

5.4.2 Diskusjon omkring analysen

I analyseprosessen har jeg måttet foreta valg knyttet til kategoriseringen av programmeringsoppgaver innen de matematiske kunnskapsområdene. Valgene baseres på en fortolkning av oppgavene, og hvilket matematisk kunnskapsområde de tilhører. I kap. 3.4.2 diskuterte jeg min studies reliabilitet, og pekte på analyseverktøyets ikke-entydige rammer for kategorisering av matematisk kunnskapsområde som en mulig begrensning ved mitt analyseverktøy. Den ikke-entydige definisjonen stammer fra læreplanens kompetansemål, som kan tolkes i vid forstand, og det faktum at jeg ikke valgte å legge til et ekstra sett med operasjonaliserbare kriterier for å tydeliggjøre kategoriene ytterligere. Her vil jeg presentere et eksempel fra oppgaveanalysen, som viser noe av problematikken, knyttet til hvordan en kan tolke det matematiske kunnskapsområdet *formler og variabler* i min studie. Holder det at elevene må ta i bruk en variabel i en programkode, eller er det et krav at variabelen ville vært brukt også i arbeid med oppgaven om den hadde vært uten tilknytning til programmering som gjør at en oppgave hører inn i kategorien *formler og variabler?*

I utvelgelsen av empiri til min studie har jeg foretatt andre valg enn Berge (2022). Der de kun har undersøkt programmeringsinnholdet i kapitlene frem til, men ikke inkludert programmeringskapittelet i Mønster 1T, har jeg inkludert samtlige programmeringsoppgaver i boken. Dermed har jeg analysert oppgave 12, s. 415 i Mønster 1T, mens de har latt være. De begrunner sitt valg med at oppgavene i dette kapittelet er programmeringsoppgaver for programmeringsinnlæring og ikke programmeringsoppgaver for matematikklæring (Berge, 2022, s. 5). Jeg har derimot både inkludert oppgaven i min analyse, og kategorisert den som en oppgave som gir muligheter til læring innen det matematiske kunnskapsområdet formler og variabler.

Mønster 1T, oppg. 12, s. 415
Be om et heltall fra brukeren. Regn ut hvor mange hele ganger dette tallet er større enn 5. Skriv svaret til skjerm.

Jeg mener oppgaven kan knyttes til kompetansemål 3, hvor formler og variable størrelser omtales. Som diskutert i kap. 5.5 er dette kompetansemålet i liten grad spesifisert. Det mener jeg denne oppgaven gir elevene mulighet til å arbeide med. Utover det å arbeide med variabel-konseptet gir oppgaven trening i helt grunnleggende aritmetikk, nemlig heltallsdivisjon. Dette kan antas å være kjent matematikk for elevene, siden barneskolen.

Problemløsningsoppgavene innen matematikk med programmering brukes i liten grad til å utforske matematiske egenskaper som er nye for elevene i faget MAT09-01. Det «nye»

synes i oppgave 12, s. 415 i Mønster 1T å være variabelbegrepet, men selv dette er ikke nytt, men noe elevene har jobbet med gjennom ungdomsskolen. Selv om en oppgave som oppgave 12, s. 415 i Mønster 1T kategoriseres som en mulighet for elevene til å lære matematikk knyttet til formler og variabler, kan man spørre seg hvor god denne muligheten er. At jeg har anvendt en så romslig fortolkning av kompetansemålene i MAT09-01 kan anses å være en svakhet ved min analyse. Jeg har derimot vært konsekvent, og anvendt denne vide fortolkningen i samtlige oppgaver. Dersom oppgaven kan sies å gi elevene eksplisitt mulighet til å arbeide med variabler, så er oppgaven knyttet til det matematiske kunnskapsområdet formler og variabler.

Et eksempel på oppgave som ikke har havnet i kategorien formler og variabler, men annet matematisk kunnskapsområde, er oppgaver hvor programmeringen skal anvendes som en kalkulator. Dersom elevene bes tilordne verdier til variabler som de ikke trenger å manipulere videre senere, har det ikke vært nok til å beregnes som arbeid med variabler å gjøre, i en kontekst som faget MAT09-01 er.

Som tidligere nevnt, gir lærebokanalysen kun et hypotetisk svar på hvilke muligheter elevene gis arbeid med læreboken (Charalambous et al., 2010). I denne studien av programmeringsoppgavene i lærebøkene er dette hypotetiske aspektet tydelig. I utvalget av empiri har jeg valgt å kun studere oppgaver som eksplisitt bruker ord som «program», «programmering» eller «Python», eller oppgaver som er skrevet inn i et eget programmeringskapittel i læreboken. Det er derimot ikke kun disse oppgavene som kan være egnet å løse ved hjelp av programmering. Disse ble valgt i denne studien, fordi jeg ikke har innblikk i hvordan oppgavene i læreboken blir anvendt i klasserommet, av læreren og av elevene. Dermed var det enkelt for meg å velge oppgaver som eksplisitt nevner programmering. Min studie sier derfor ikke noe om den generelle muligheten til arbeid med programmering som lærebøkene gir, kun mulighetene disse utvalgte programmeringsoppgavene gir.

Som følge av utvelgelsen av empiri blir et viktig aspekt ved evaluering innen algoritmisk tenkning, umulig å undersøke i min studie: I disse oppgavene er det forutbestemt at elevene skal bruke programmering i arbeidet med å finne løsningen på oppgaven. Elevene får derfor ikke mulighet til å vurdere hvilket digitalt hjelpemiddel, hvis noe, som er mest egnet til å løse oppgaven, i tråd med evaluerings-kategorien i Selby og Woollards (2013) rammeverk.

I lærebøkene kunne det, på den annen side, stått for eksempel «vurder hvorvidt du vil løse oppgaven ved bruk av programmering, Geogebra eller et annet hjelpemiddel». Ingen formulering som denne, eller formuleringer med et liknende budskap, er funnet i denne oppgavens arbeid med å analysere oppgavene i de trykte lærebøkene i MAT09-01.

5.5 Implikasjoner for praksis

Læreren bør ikke basere sin programmeringsundervisning utelukkende på oppgaver av typen programmeringsoppgaver hentet fra lærebøkene. Det vil begrense elevenes muligheter til å arbeide med problemløsning og helheten av aspekter ved algoritmisk tenkning.

I resultatene rundt muligheter for arbeid med algoritmisk tenkning i programmeringsoppgavene i lærebøkene, finner vi et avvik mellom teori og de faktiske forholdene i lærebøkene. Mens det teoretiske rammeverket for algoritmisk tenkning

inkluderer fem kategorier med handlinger som til sammen utgjør algoritmisk tenkning, gir programmeringsoppgavene i de studerte lærebøkene kun muligheter for arbeid med fire av disse.

I denne studiens resultater synliggjøres et avvik mellom teori rundt algoritmisk tenkning, og de faktiske mulighetene elevene gis for aktiviteter knyttet til algoritmisk tenkning i programmeringsoppgavene i lærebøkene. I kapittel 5.1 drøftes det hvorvidt mangelen på muligheter knyttet til abstrahering kan være koblet sammen med en lite hensiktsmessig tolkning av læreplanen, eller om læreplanens innhold med fordel burde vært justert for å treffe intensjonen om at matematikklæringen bør bidra til algoritmisk tenkning. Denne diskusjonen medfører potensielt implikasjoner for både lærebokforfatteren og de som utvikler læreplanen i MAT09-01.

Når data og modellering har en så sentral rolle i algoritmisk tenkning, slik Weintrop et al. (2016) har observert det i litteraturen og ute i praksis blant realister, ville det vært naturlig om denne typen oppgaver ble viet større plass også i matematikkfaget.

Det ville også vært hensiktsmessig for å kunne implementere programmering i matematikkfaget som lettere kan bidra til fagets relevans. I oppgavene som er studert i lærebøkene i MAT09-01 gis matematikken inntrykk av å være en kontekst for læring av programmering.

Dagens vidt formulerte læreplan gir rom for tolkning, og man kan fint implementere data og modellering i dagens matematiske kunnskapsområde modellering. Denne muligheten benyttes ikke i lærebøkene slik de fremstår i dag. Her har lærebokforfatterne anledning til å tilby elevene enda flere muligheter til å få jobbet matematisk og i tråd med læreplanen, ved å inkludere flere programmeringsoppgaver om modellering og behandling av datasett

Enda større læringsutbytte med tanke på integrering av programmering og matematikkfaget kunne man kanskje fått, om kompetansemålene i MAT09-01 var annerledes. Det er ikke rart at lærebøkene ikke integrerer programmering i kapitlene om polynomdivisjon. Her finnes det veletablerte former for å lære bort denne typen matematikk, og programmering savnes ikke her, fordi det kanskje ikke føles hensiktsmessig å utforske polynomdivisjon gjennom programmering. Derimot ville det vært naturlig å integrere programmering og matematikk i emner som diskret matematikk eller representasjoner av store datasett. Kanskje kan en justering i de matematiske emnene i læreplanen føre til at programmeringen bidrar til enda flere muligheter til matematikklæring, algoritmisk tenkning og problemløsning i matematikkfaget? Her finnes et uutnyttet potensial som læreplanutviklerne kan se nærmere på i neste revisjon av læreplanen. Ved tydeligere formidlet intensjon bak kompetansemålene i faget, vil lærebokforfatternes behov for å tolke læreplanen bli mindre, og lærebøkene kan lettere bli i tråd med læreplanens hensikt.

Fra høsten 2017 ble det innført et nytt fag i norsk videregående skole: Programmering og modellering X (Utdanningsdirektoratet, 2021). I kompetansemålene til dette faget kan man blant annet lese om visualisering av data ved hjelp av programmering, bruk av data for å simulere virkelige systemer og å bruke strategier for testing, feilsøking og feilhåndtering (Utdanningsdirektoratet, 2021). Jeg har ikke satt meg inn i bakgrunnen for opprettelsen av dette faget. Derimot kunne jeg ønske meg at dette ikke var beskrivelsen av et valgfag for den spesielt interessert elev, men at denne eksplisitte integreringen av

programmering og modellering kom til synet også i øvrige matematikkfag i videregående skole, der det er tenkt at elevene skal lære matematikk gjennom programmering.

5.6 Implikasjoner for videre forskning

Studien min har flere bidrag inn til det matematikkdidaktiske forskningsfeltet. Forskning på programmeringsinnholdet i lærebøker i matematikk er et område innen matematikkdidaktisk forskning som er under utvikling (Jensen et al., 2023). Tidligere forskning på feltet har i stor grad fokusert på lærebøker på grunnskoletrinn (Bråting & Kilhamn, 2022; Jensen et al., 2023; Hammerø & Schmeding, 2023; Elicer & Tamborg, 2022). Forskningen jeg har funnet som gjennomfører lærebokanalyser på videregående nivå, med hensyn til programmeringsinnholdet i matematikk, har hatt et øvrig fokus enn min studie (Berge, 2022). Nytt i min studie er at undersøkelsen tar utgangspunkt i Utdanningsdirektoratet og læreplanen LK20 sine uttalelser om at programmeringen skal bidra til algoritmisk tenkning, problemløsning og matematikklæring. Dette muliggjør undersøkelser av hvilke muligheter elevene gis til arbeid i tråd med læreplanens uttalte intensjon for programmering i matematikkfaget i skolen.

Studien har vist anvendelsen av et mulig rammeverk for vurdering av programmeringsinnholdet i trykte lærebøker i matematikk, opp mot læreplanens intensjon med programmering i matematikkfaget. Studien viser at anvendelse av Lithner (2008) og Boesens (2006) begreper kreativ og imitativ resonnering kan gi interessant informasjon også i en lærebokanalyse, selv om det originale rammeverket er utviklet for forskning på nasjonale prøver og vurderingssituasjoner (Boesen, 2006). Anvendelsen av dette rammeverket ga innblikk i muligheter som lå i programmeringsoppgavene i lærebøkene, med tanke på problemløsning, og fungerte etter hensikten; å vurdere mulighetene for kreativ resonnering i et utvalg programmeringsoppgaver, uten å ha tilgang til informasjon om personene som skal løse oppgavene.

Denne studien er et bidrag i å kartlegge lærebøker i en norsk undervisningskontekst på videregående skole. Det anvendte teoretiske rammeverket kan derimot godt fungere til å analysere øvrige programmeringsressurser i øvrige kontekster, det være seg andre land, oppgaver via øvrige undervisningsressurser enn læreboken, eller oppgaver gitt på eksamen. Dersom det teoretiske rammeverket skal anvendes utenfor læreboken anbefaler jeg å endre nivå 3 av rammeverket tilbake til Palm et al. sine opprinnelige beskrivelser, slik at mine endringer som er utført spesifikt med hensyn til læreboken og dens oppbygging og rekkefølge utelates. Spesielt godt vil rammeverket fungere til å analysere norske lærebøker i matematikk på lavere trinn, fordi læreplanen synes å benytte seg av det samme teoretiske rammeverk for algoritmisk tenkning som det som anvendes i denne studien, på tvers av trinn (Utdanningsdirektoratet, 2019). Slik kan rammeverket vurdere hvilke muligheter programmeringsoppgaver gir mot læreplanens intensjon på både høyere og lavere trinn, gitt at delen om matematiske kunnskapsområder tilpasses det enkelte trinns kompetansemål i matematikk.

Denne studien har ikke undersøkt andelen problemløsningsoppgaver blant programmeringsoppgavene kontra andelen problemløsningsoppgaver blant det totale antallet oppgaver i lærebøkene. Gitt læreplanens intensjon om at programmering skal bidra til problemløsning, vil det være nærliggende å tro at problemløsningsoppgavene innen programmering bør være minst like mange i forhold til problemløsningsoppgavene i de øvrige oppgavene i læreboken (Berge, 2022). Som påpekt av Berge (2022) vil et

interessant område for videre forskning være å undersøke hva som synes å være det ideelle forholdet mellom rutineoppgaver og problemløsningsoppgaver innen programmering i matematikkfaget.

Et annet mulig område for videre forskning er å anvende denne studiens rammeverk på de sentralgitte, skriftlige eksamenene i MAT09-01 etter innføring av Kunnskapsløftet 2020. Hvordan er programmeringsoppgavene som gis der, sammenliknet med de elevene får muligheten til å arbeide med i lærebøkene i faget? Det ville det vært interessant å undersøke. Jeg har lekt med tanken om å inkludere dette som et forskningsspørsmål i min studie, men slått den fra meg grunnet oppgavens omfang. Den som ønsker å forske videre på dette bør reversere de av endringene jeg har utført på Palm et al. (2011) sitt rammeverk, for å få det *fra* det opprinnelige rammeverket tilpasset undersøkelser av skriftlig vurdering, *til* lærebokanalysen i denne studien.

6 Avlutning

Denne studien viser hvilke muligheter de tre trykte lærebøkene i faget matematikk 1T kan gi elevene i arbeid med programmeringsoppgaver i matematikkfaget.

Lærebokanalysen som er foretatt i denne studien har vist at programmeringsoppgavene i lærebøkene i matematikk 1T gir mulighet til arbeid med enkelte aspekter ved algoritmisk tenkning. De fleste programmeringsoppgavene gir mulighet til algoritmebehandling. Omtrent en av fem programmeringsoppgaver gir mulighet til evaluering. Enkelte oppgaver gir mulighet til arbeid med generalisering og dekomponering, mens programmeringsoppgavene er uten mulighet for arbeid med abstrahering.

Det er også stor variasjon i mulighetene for algoritmisk tenkning i programmeringsoppgavene, på tvers av lærebøkene. To av bøkene muliggjør arbeid med både algoritmebehandling, evaluering og generalisering, én av bøkene gir kun muligheter for arbeid med algoritmebehandling, mens kun en bok tilbyr programmeringsoppgaver som gir muligheter innen både dekomponering, algoritmebehandling, evaluering og generalisering. Dette medfører som implikasjon for læreren i matematikk 1T at flere ressurser bør benyttes i arbeidet med planlegging og gjennomføring av undervisningen. Det er ikke tilstrekkelig å anvende programmeringsoppgavene i læreboken alene, for å gi elevene muligheter for arbeid med algoritmisk tenkning. Dette er særlig sant om læreren anvender læreboken Sinus 1T. Studien fant at Sinus 1T mangler innholdskunnskapen som elevene behøver for å besvare flere av programmeringsoppgavene i boken. Kombinert med de ensformige mulighetene for algoritmisk tenkning er det viktig at læreren oppfordrer elevene til å løse også andre oppgaver i læreboken med programmering, enn de som er undersøkt i denne studien, eller at læreren benytter seg av flere ressurser i arbeidet med planlegging og gjennomføring av undervisningen knyttet til programmering i matematikkfaget. Å anvende flere kilder til informasjon og oppgaver er en generell anbefaling til lærere og elever, basert på resultatene av denne studien.

Når det kommer til problemløsning, finnes det muligheter for kreativ resonnering i om lag 40% av programmeringsoppgavene. Av disse gir omtrent halvparten mulighet for kreativ resonnering i kun programmering, mens halvparten gir mulighet for kreativ resonnering innen matematikk, i tillegg til eventuell problemløsning knyttet til programmering. At flesteparten av oppgavene kan løses med imitativ resonnering kan ses i sammenheng med den store andelen oppgaver som anvender algoritmebehandling, og stemmer overens med funn fra tidligere forskning.

Programmeringsoppgavene i de studerte lærebøkene gir mulighet for arbeid med de fleste av de matematiske kunnskapsområdene som er relevante for kompetansemålene i matematikk 1T (MAT09-01). Flest muligheter gis til arbeid med formler og variable størrelser, og likninger og ulikheter. I to av lærebøkene gis et vesentlig antall muligheter til numeriske metoder, uten at dette nevnes eksplisitt i læreplanen for matematikk 1T (MAT09-01). Arbeid med numeriske metoder er hensiktsmessig utført med bruk av programmering. Dette fører til en drøfting av kombinasjonen av programmeringsoppgaver og matematikk: Kunne programmeringen gitt større potensiale for integrering med matematikk, dersom læreplanen for matematikk var noe justert?

Mulighetene elevene gis til algoritmisk tenkning, problemløsning og arbeid med matematiske kunnskapsområder er veldig varierte i lærebøkene i matematikk 1T (MAT09-01). Hvilke muligheter som gis varierer i stor grad fra lærebok til lærebok. Med læreplanens vide formuleringer omkring kompetansemålene generelt, og programmeringens plass i faget matematikk 1T spesielt, oppstår et stort rom for tolkning. Det er ikke rart at tre ulike team med lærebokforfattere tolker noe ulikt fra hverandre.

Studien foreslår for læreplanutviklere å vurdere en innføring av diskret matematikk med differenslikninger og numeriske metoder som en del av læreplanen, og å fremheve programmeringens muligheter til arbeid med modellering og behandling av store datasett. Ved tydeligere å peke på hvor det er tenkt at matematikken og programmeringen skal kobles sammen, ville lærere og lærebokutvikleres behov for tolkning, og rom for feiltolkninger, blitt mindre. Kanskje ville dette ført til en mer operasjonaliserbar implementering av programmering i matematikkfaget, og en implementering som gir større potensiale til matematikklæringen i faget.

Studiens fremgangsmåte er satt sammen av flere steg. Innledningsvis ble det gjort en systematisk tolkning av læreplanens anvendelse av begrepet algoritmisk tenkning. Den konkluderte med at Selby og Woollards (2013) definisjon av algoritmisk tenkning sammenfaller med læreplanens. Deretter ble Lithner (2008) sitt rammeverk for kreativ og imitativ resonnering tilpasset konteksten lærebokanalyse innen programmering i matematikkfaget. Kompetansemålene i matematikk 1T (MAT09-01) dannet grunnlaget for 11 matematiske kunnskapsområder. Til sammen utgjorde disse matematiske kunnskapsområdene, Selby og Woollards (2013) og Lithners (2008) rammeverk det teoretiske rammeverket og denne studien.

Det teoretiske rammeverket fungerte hensiktsmessig for å studere mulighetene som programmeringsoppgavene gir for algoritmisk tenkning, problemløsning og arbeid med matematikk, lik ordlyden i læreplanen. En nærmere fininndeling av kategoriene innen algoritmisk tenkning kunne gitt flere svar angående mulighetene alle programmeringsoppgavene innen algoritmebehandling gir, men dette ble valgt bort til fordel for å beholde rammeverket slik det er, for å skape høyest mulig sammenheng mellom undersøkelsen i denne studien, og ordlyden i læreplanen rundt hensikten med programmering i matematikk 1T (MAT09-01). Videre forskning på programmeringsinnholdet i undervisningsressurser i matematikk kan bygge videre på arbeidet som er gjennomført i denne studien. Rammeverket kan, med noen tilpasninger, anvendes på både høyere og lavere trinn.

7 Referanser

Askeland, N. (2023, 20. mars). Lærebok. I *Store norske leksikon.* https://snl.no/l%C3%A6rebok

Balanskat, A., & Engelhardt, K. (2015). Computing our future. Computer programming and coding. Priorities, school curricula and initiatives across Europe. European Schoolnet, Brussels. http://www.eun.org/documents/411753/817341/Computing+our+future_final_2015.pdf/d3780a64-1081-4488-8549-6033200e3c03

Bell, T. (2016). What's all the fuss about coding?

Bell, T. & Vahrenhold, J. (2018). CS Unplugged – How Is It Used, and Does It Work? I H.-J. Böckenhauer, D. Komm & W. Unger (Red.), *Adventures Between Lower Bounds and Higher Altitudes: Essays Dedicated to Juraj Hromkovič on the Occasion of His 60th Birthday* (s. 497-521). Springer. https://doi.org/10.1007/978-3-319-98355-4

Benton, L., Hoyles, C., Kalas, I., & Noss, R. (2017). Bridging primary programming and mathematics: Some findings of design research in England. Digital Experiences in Mathematics Education, 3(2), 115–138. https://doi.org/10.1007/s40751-017-0028-x

Berge, R. L. (2022, 19.-22. september). Integrated programming and mathematics in schools – A solid foundation for a future engineering education? [Paperpresentasjon]. SEFI 50th Annual conference of the European Society for Engineering Education. Barcelona. https://www.researchgate.net/publication/369089477_Integrated_programming_and_mathematics_in_schools_-_A_solid_fundation_for_a_future_engineering_education

Berry, M. (2014). *Computational thinking in primary schools.* Teach primary. http://milesberry.net/2014/03/computational-thinking-in-primary-schools/.

Björkqvist, O. (2003). Matematisk problemløsning. I B. Grevholm (Red.), Matematikk for skolen (s. 51- 57). Bergen: Fagbokforlaget.

Blikstein, P. (2018). Pre-College Computer Science Education: A Survey of the Field. Mountain View, CA: Google LLC. Retrieved from https://goo.gl/gmS1Vm

Bocconi, S. & Chioccariello, A. (2018). The Nordic approach to introducing computational thinking and programming in compulsory education. Hentet fra https://www.itd.cnr.it/doc/CompuThinkNordic.pdf

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K. (2016). Developing computational thinking in compulsory education – Implications for policy and practice. doi:10.2791/792158

Boesen, J. (2006). *Assessing mathematical creativity : Comparing national and teachermade tests, explaining differences and examining impact.*

[Doktorgradsavhandling, Umeå universitet]. https://umu.diva-portal.org/smash/get/diva2:144670/FULLTEXT01.pdf

Bråting, K. & Kilhamn, C. (2022). The Integration of Programming in Swedish School Mathematics: Investigation Elementary Mathematics Textbooks. *Scanidinavian Journal of Education Research, 66*(4), 594-609. https://doi.org/10.1080/00313831.2021.1897879

Munthe, M. (2022). *Press «Run» to Improve Mathematical Expertise (PRIME).* [Doktorgradsavhandling, Norges miljø- og biovitenskapelige universitet]. NMBU Brage. https://nmbu.brage.unit.no/nmbu-xmlui/bitstream/handle/11250/3045236/115012_NMBU_Munthe_finished.pdf?sequence=1&isAllowed=y

Boston, M. D. & Smith, M. S. (2009), Transforming Secondary Mathematics Teaching: Increasing the Cognitive Demands of Instructional Tasks Used in Teachers' Classrooms, *J. Res. Math. Educ. 40*(2), 119–156.

Brennan, K. & Resnick, M. (2012, 13.-17. april). New frameworks for studying and assessing the development of computational thinking [Paperpresentasjon]. Proceedings of the 2012 American Educational Research Association, Vancouver. https://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf

Charalambous, C. Y., Delaney, S., Hsu, H.-Y., & Mesa, V. (2010). A Comparative Analysis of the Addition and Subtraction of Fractions in Textbooks from Three Countries. Mathematical Thinking & Learning, 12(2), 117–151.

Cohen, L., Manion, L. & Morrison, K. (2007). Research methods in education Oxon: Routledge, 6. https://gtu.ge/Agro-Lib/RESEARCH%20METHOD%20COHEN%20ok.pdf

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). Computational thinking—A guide for teachers.

Dalland, O. (2020). *Metode og oppgaveskriving* (7. utg.). Gyldendal.

Drange, P. G. (2023, 15. Oktober). Blokkprogrammering. I *Store norske leksikon.* https://snl.no/blokkprogrammering

Drijvers, P. (2015). Digital technology in mathematics education: Why it works (or doesn't). I S. Cho (Red.), *Selected Regular Lectures from the 12th International Congress on Mathematical Education*, (s. 135-151). Springer: Cham.

Elicer, R. & Tamborg, A. L. (2022). Nature of the relations between programming and computational thinking and mathematics in Danish teaching resources. I U. F. Jankvist, R. Elicer, A. Clark-Wilson, H.-G. Weigand & M. Thomsen (Red.). *Proceedings of the 15th International Conference on Technology in Mathematics Teaching (ICTMT 15)* (s. 45-52). University. https://doi.org/10.7146/aul.452

Forsström, S. E. & Kaufmann, O. T. (2018). A literature review exploring the use of programming in mathematics education. *International Journal of Learning, Teaching and Educational Research*, *17*(12), 8-32. https://doi.org/10.26803/ijlter.17.12.2

Frydenlund, C. & Holst, L. (Red.). (2020). *Matematikk 1T.* Ashcehoug.

Gilje, Ø., Ingulfsen, L., Dolonen, J. A., Furberg, A., Rasmussen, I., Kluge, A., Knain, E., Mørch, A., Naalsund, M. & Granum, K. (2016). Med ARK&APP*. Bruk av læremidler*

og ressurser for læring på tvers av arbeidsformer. Universitetet i Oslo. https://www.uv.uio.no/iped/forskning/prosjekter/arkapp/arkapp_syntese_endelig_til_trykk.pdf

Gjøvik, Ø., Torkildsen, H. A. (2019). Algoritmisk tekning. *Tangenten – tidsskrift for matematikkundervisning, 30*(3). 31–37. http://www.caspar.no/tangenten/2019/Tangenten%203%202019%20Gj%C3%B8vik%20Torkildsen.pdf

Grover, S. & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field*. Educational Researcher, 42*(1), 38–43.

Grønmo, S. (2016). *Samfunnsvitenskapelige metoder* (2.utg.). Fagbokforlaget.

Guba, E. G. (1981). ERIC/ECTJ annual review Paper: Criteria for assessing the trustworthiness of naturalistic inquiries. *Educational Communication and Technology, 29*(2), 75–91. https://www.jstor.org/stable/30219811

Haladyna, T. M. (1997). *Writing test items to evaluate higher order thinking.* Allyn and Bacon

Hammerø, J. & Schmeding, A. (2023). *An instrument theoretic analysis of programming tasks in Norwegian mathematics textbooks.* [Preprint]. https://doi.org/10.13140/RG.2.2.19392.71680

Dahl, G., Ranestad, K. & Hole, A. (2017, 25. oktober). Programmering rammer dybdelæring i matematikk. *Aftenposten.* https://www.aftenposten.no/meninger/kronikk/i/E0zga/programmering-rammer-dybdelaering-i-matematikk-geir-dahl-kristian-ranestad-og-arne-hole

Hurt, T., Greenwald, E., Allan, S., Cannady, M. A., Krakowski, A., Brodsky, L., Collins, M. A., Montgomery, R. & Dorph, R. (2023). The computational thinking for science (CT-S) framework: operationalizing CT-S for K-12 science education researchers and educators. International Journal of STEM Education, 10(1). https://doi.org/10.1186/s40594-022-00391-7

Jablonka, E., & Johansson, M. (2010). Using texts and tasks: Swedish studies on mathematics textbooks. I B. Sriraman, C. Bergsten, S. Goodchild, G. Palsdottir, B. Dahl, B. D. Söndergaard, & L. Haapasalo (Red.), *The first source-book on Nordic research in mathematics education* (s. 363–372). Information Age Publishing.

Jensen, M. S., Julien A. L., Rafiepour, A. & Schmeding, A. (2023). *An analytical framework for programming tasks in mathematics textbooks* [Preprint]. FLU, Nord University. https://www.researchgate.net/publication/369334923_An_analytical_framework_for_programming_tasks_in_mathematics_textbooks

Johansson, M. (2006). *Teaching mathematics with textbooks: A classroom and curricular perspective* [Doktorgradsavhandling, Luleå University of Technology]. http://ltu.diva-portal.org/smash/get/diva2:998959/FULLTEXT01.pdf

Kaufmann, O. T. & Stenseth, B. (2023). Programmering som del av matematikkfaget. *Norsk pedagogisk tidsskrift, 107*(2), 109-123. https://doi.org/10.18261/npt.107.2.2

Kilhamn, C., Bråting, K., & Rolandsson, L. (2021). Teachers' arguments for including programming in mathematics education. I G. A. Nortvedt, N. F. Buchholtz, J.

Fauskanger, F. Hreinsdóttir, M. Hähkiöniemi, B. E. Jessen, J. Kurvits, Y. Liljekvist, M. Misfeldt, M. Naalsrud, H. K. Nilsen, G. Palsdottir, P. Portaankorva-Koivisto, J. Radisic & A. Werneberg (Red.), *Bringing Nordic mathematics education into the future. Preceedings of Norma 20 The ninth Nordic Conference on Mathematics Education.* SMDF, Svensk Förening för MatematikDidaktisk Forskning.

Kilpatrick, J. (1969). Problem Solving in Mathematics. *Science and Mathematics Education, 39*(4). 523-534. https://doi.org/10.2307/1169713

Kongelf, T. R. (2019). *Matematisk innhold og matematiske metoder i lærebøker brukt på ungdomstrinnet i Norge: Gullgruve eller fallgruve for utvikling av matematisk kompetanse i problemløsning og algebra?* [Doktorgradsavhandling]. Brage. http://hdl.handle.net/11250/2616700

Kunnskapsdepartementet, (2019a). *Læreplan i kunst og håndverk (KHV01-02).* Fastsatt som forskrift. Læreplanverket for Kunnskapsløftet 2020. https://www.udir.no/lk20/KHV01-02

Kunnskapsdepartementet, (2019b). *Læreplan i matematikk fellesfag vg1 teoretisk (matematikk T) (MAT09-01).* Fastsatt som forskrift. Læreplanverket for Kunnskapsløftet 2020. https://www.udir.no/lk20/MAT09-01

Kunnskapsdepartementet, (2019c). *Læreplan i musikk (MUS01-02).* Fastsatt som forskrift. Læreplanverket for Kunnskapsløftet 2020. https://www.udir.no/lk20/MUS01-02

Kunnskapsdepartementet, (2019d). *Læreplan i naturfag (NAT01-04).* Fastsatt som forskrift. Læreplanverket for Kunnskapsløftet 2020. https://www.udir.no/lk20/NAT01-04

Kvale, S. & Brinkmann, S. (2015). *Det kvalitative forskningsintervju*. (3. utg.). Gyldendal.

Lesh, R., Cramer, K., Doerr, H. M., Post, T. & Zawojewski, J. S. (2003). Model development sequences. I R. Lesh & H. M. Doerr (Red.), *Beyond constructivism: models and modeling perspectives on mathematics problem solving, learning and teaching* (s. 35–58). Lewrence Erlbaum.

Lithner, J. (2008). A reasearch framework for creative and imitative reasoning. *Educational Studies in Mathematics, 67*(3), 255-276. https://doi.org/10.1007/s10649-007-9104-2

Lodi, M. & Martini, S. (2021). Computational Thinking, Between Papert and Wing. Science & education, 30(4), 883-908. https://doi.org/10.1007/s11191-021-00202-5

Lye, S. Y. & Koh, J. H. L. (2014). Review on Teaching and Learning of Computational Thinking through Programming: What Is Next for K-12? *Computers in Human Behavior, 41.* 51-61. https://doi.org/10.1016/j.chb.2014.09.012

Maus, G. & Smestad, B.-T. (Red.). (2020). *Sinus 1T: Matematikk.* Cappelen Damm.

Meld. St. 37. (1988-1989). *Stortingsmelding 37 - Om datateknologi i skole og opplæring*. Kirke- og undervisningsdepartementet. https://www.stortinget.no/nn/Saker-og-publikasjonar/Stortingsforhandlingar/Lesevisning/?p=1987-88&paid=3&wid=c&psid=DIVL373&pgid=c_0253

Meld. St. 28 (2015-2016). *Fag – Fordyping – Forståelse – En fornyelse av Kunnskapsløftet.* Kunnskapsdepartementet. https://www.regjeringen.no/no/dokumenter/meld.-st.-28-20152016/id2483955/

Mesa, V. (2004). Characterizing practices associated with functions in middle school textbooks: An empirical approach*. Educational Studies in Mathematics, 56*(2), 255–286.

Misfeldt, M., Szabo, A., & Helenius, O. (2019). Surveying teachers' conception of programming as a mathematical topic following the implementation of a new mathematics curriculum. I U. Jankvist, M. Van den Heuvel-Panhuizen, & M. Veldhuis, M. (Red.). *Proceedings of CERME11* (s. 2713–2720). Utrecht University

Mozelius, P., Ulfenborg, M., & Persson, N. (2019, 11.-13. mars). *Teacher attitudes towards the integration of programming in middle school mathematics*. [Paperpresentasjon]. 13th International Technology, Education and Development Conference, Valencia. https://doi.org/10.21125/inted.2019.0249

Munthe, M. (2022). *Press «Run» to Improve Mathematical Expertise (PRIME).* [Doktorgradsavhandling, Norges miljø- og biovitenskapelige universitet]. NMBU Brage. https://nmbu.brage.unit.no/nmbu-xmlui/bitstream/handle/11250/3045236/115012_NMBU_Munthe_finished.pdf?sequence=1&isAllowed=y

NOU 2013: 2. (2013). *Hindre for digital verdiskaping.* Kommunal- og distriktsdepartementet. https://www.regjeringen.no/no/dokumenter/nou-2013-2/id711002/

NOU 2015: 8. (2015). *Fremtidens skole. Fornyelse av fag og kompetanser*. Kunnskapsdepartementet. https://www.regjeringen.no/no/dokumenter/nou-2015-8/id2417001/

NOU 2015: 13. (2015). *Digital sårbarhet – sikkert samfunn – Beskytte enkeltmennesker og samfunn i en digitalisert verden.* Justis- og beredskapsdepartementet. https://www.regjeringen.no/no/dokumenter/nou-2015-13/id2464370/

NOU 2020: 2. (2020). *Fremtidige kompetansebehov III: Læring og kompetanse i alle ledd.* Kunnskapsdepartementet. https://www.regjeringen.no/contentassets/053481d65fb845be9a2b1674c35d6d14/no/pdfs/nou202020200002000dddpdfs.pdf

OECD. (2019). *OECD Skills Outlook 2019: Thriving in a Digital World.* OECD. https://doi.org/10.1787/df80bc12-en

Palm, T., Boesen, J., & Lithner, J. (2005). *The requirements of mathematical reasoning in upper secondary level assessment.* Umeå universitet.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas.* Basic Books, Inc.

Pea, R. D., & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New ideas in psychology*, *2*(2), 137-168.

Pingel, F. (2010). *UNESCO Guidebook on Textbook Research and Textbook Revision* (2. utg). United Nations Educational, Scientific and Cultural Organization. https://unesdoc.unesco.org/ark:/48223/pf0000117188/PDF/117188eng.pdf.multi

Polya, G. (1981). *Mathematical discovery: On understanding, learning, and teaching problem solving.* John Wiley & Sons, Inc.

Polya, G. (2004). *How to solve it: A new aspect of mathematical method*. Princeton University Press.

Popat, S. & Starkey, L. (2019). Learning to code or coding to learn? A systematic review. *Computers & Education, 128*(365-378)

Postholm, M. B., & Jacobsen, D. I. (2018) *Forskningsmetode for masterstudenter i lærerutdanningen.* Cappelen Damm AS.

Raustøl, A. & Lund, H.-S. (Red.). (2020). *Mønster: Matematikk 1T.* Gyldendal.

Sanne, A., Berge, O., Bungum, B., Jørgensen, E. C., Kluge, A., Kristensen, T. E., Mørken, K. M., Svorkmo, A.-G., & Voll, L. O. (2016). Teknologi og programmering for alle. Utdanningsdirektoratet, 91.

Scherer, R., Siddiq, F. & Viveros, B. (2019). The cognitive benefits of learning computer programming: A meta-analysis of transfer effects. *Journal of educational psychology*, 111(5), 764-792. https://10.1037/edu0000314

Schmidt, W. H., McKnight, C. C., Valverde, G. A., Houang, R. T. & Wiley, D. E. (1997). *Many Visions, Many Aims: A Cross-national Investigation of Curricular Intentions in School Mathematics.* Springer.

Schoenfeld, A. H. (1992). *Teaching mathematical thinking and problem solving*. [Paperpresentasjon]. Sånn, ja! Rapport fra en konferanse om matematikk-didaktikk og kvinner i matematiske fag, Kristiansand. https://www.nb.no/items/URN:NBN:no-nb_digibok_2008011001014

Selby, C. & Woollard, J. (2013). *Computational thinking: the developing definition.* http://eprints.soton.ac.uk/356481/ [Accessed 01-04-2014].

Selby, C. & Woollard, J. (2014, 5.-8. mars). Computational thinking: the developing definition [Paperpresentasjon]. Special Interest Group on Computer Science Education (SIGCSE) 2014, Atlanta. https://www.researchgate.net/publication/299450690_Computational_thinking_the_developing_definition

Shute, V. J., Sun, C. & Asbell-Clarke, J. (2017). Demystifying computational thinking. Educational Research Review, 22, 142-158. https://doi.org/10.1016/j.edurev.2017.09.003

Sneider, C., Stephenson, C., Schafer, B., & Flick, L. (2014). Computational thinking in high school science classrooms. *The Science Teacher, 81*(5), 53-59.

Solvang, R. (1992). *Matematikkdidaktikk.* Oslo: NKI Forlaget.

Sun, L., Hu, L. & Zhou, D. (2021). Improving 7th-graders' computational thinking skills through unplugged programming activities: A study on the influence of multiple fators. *Thinking Skills And Creativity, 42.* https://doi.org/10.1016/j.tsc.2021.100926

Tjora, A. (2023). *Kvalitative forskningsmetoder: I praksis.* (4. utg). Gyldendal.

Utdanningsdirektoratet. (2019, 27. mars). *Algoritmisk tenkning.* https://www.udir.no/kvalitet-og-kompetanse/digitalisering/algoritmisk-tenkning/

Utdanningsdirektoratet. (2021). *Læreplan i programmering og modellering X (PRM01-02).* Læreplanverket for Kunnskapsløftet 2020. https://www.udir.no/lk20/PRM01-02

Utdanningsdirektoratet. (2023, 9. juni). *Kva er nytt i matematikk?* https://udir.no/laring-og-trivsel/lareplanverket/fagspesifikk-stotte/nytt-i-fagene/hva-er-nytt-i-matematikk/

Utdanningsdirektoratet. (u.å). *Støtte til læreplanen.* https://www.udir.no/lk20/mat09-01/kompetansemaal-og-vurdering/kv42?Verb=true

Valverde, G., Bianchi, L. J., Wolfe, R., Schmidt, W. H., & Hounang, R. T. (2002). *According to the book: Using TIMSS to investigate the translation of policy into practice through the world of textbooks.* Springer Science

Vihovde, E. H. (2019, 6. november). Programmeringsspråk. I *Store norske leksikon.* https://snl.no/programmeringsspr%C3%A5k

Voogt, J., Fisser, P., Good, J., Mishra, P. & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Educ Inf Technol, 20.* 715-728. https://doi.org/ 10.1007/s10639-015-9412-6

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. Journal of Science Education and Technology, 25(1), 127–147.

Voogt, J., Fisser, P., Good, J., Mishra, P. & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. Education and Information Technologies, 20(4), 715-728. https://doi.org/10.1007/s10639-015-9412-6

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L. & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. Journal of Science Education and Technology, 25(1), 127-147. https://doi.org/10.1007/s10956-015-9581-5

Wing, J. (2006). Computational Thinking. Communications of the ACM, 49, 33–35. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2008). Computational Thinking and Thinking about Computing. Philosophical Transactions: Mathematical, Physical and Engineering Sciences, 366(1881), 3717–3725.

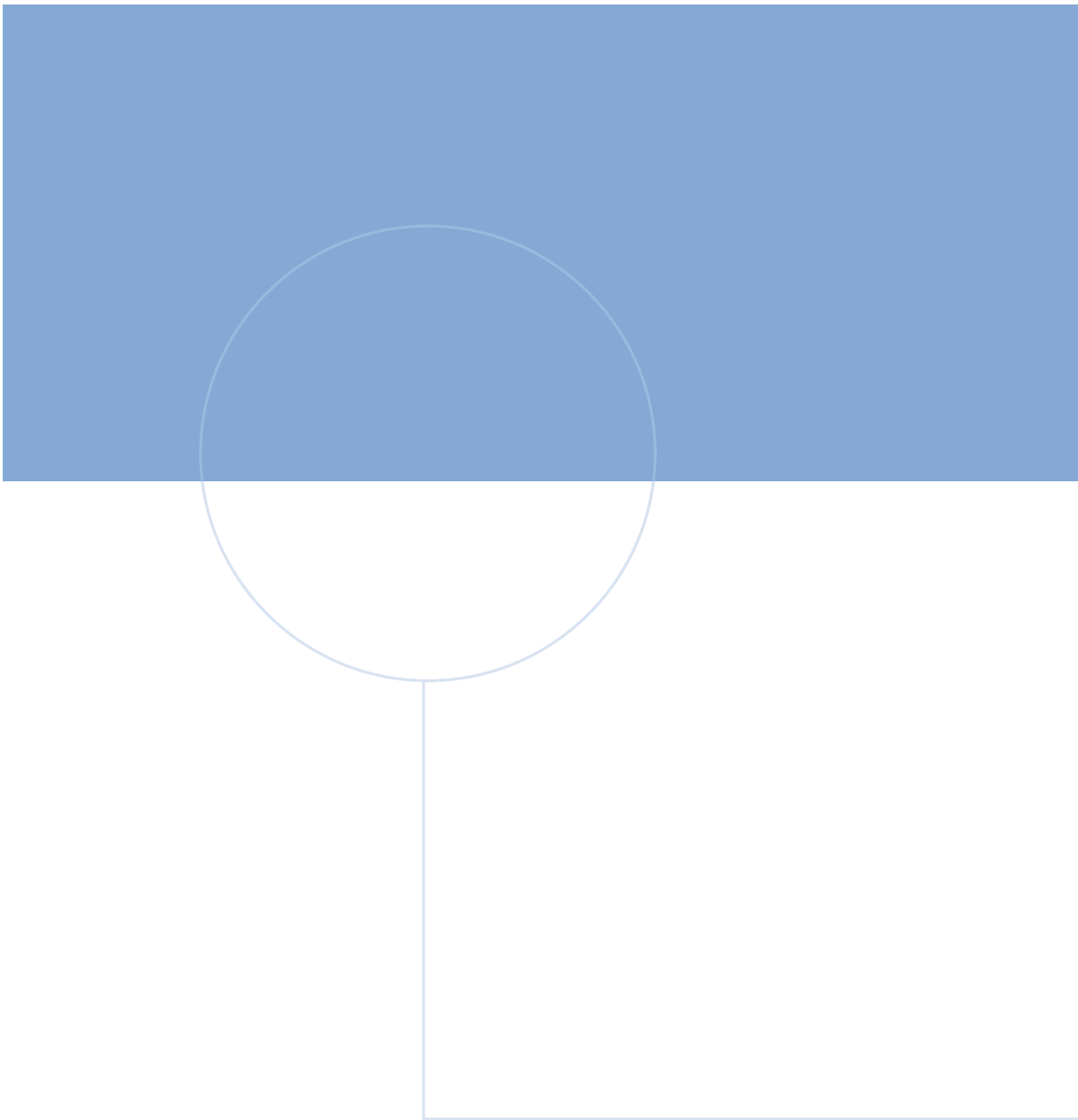
Vedlegg

Vedlegg 1: Utklipp av excel-fil brukt til oppgaveanalysen

	A	B	C	D	E	F	G
1	Lærebok	Oppgave nr	Aspekt ved AT	Type resonnering	Matematisk kunnskapsområde	Oppdrag	
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							
25							
26							
27							
28							

	H	I
1		
2		
3		
4	**Algoritmisk tenkning:**	
5	Abstrahering	=ANTALL.HVIS(C:F; "Abstrahering")
6	Dekomponering	=ANTALL.HVIS(C:F; "Dekomponering")
7	Algoritmebehandling	=ANTALL.HVIS(C:F; "Algoritmebehandling")
8	Evaluering	=ANTALL.HVIS(C:F; "Evaluering")
9	Generalisering	=ANTALL.HVIS(C:F; "Generalisering")
10		
11	**Type resonnering:**	
12	KAR	=ANTALL.HVIS(C:F; "KAR")
13	GAR	=ANTALL.HVIS(C:F; GAR)
14	MR	=ANTALL.HVIS(C:F; "MR")
15	AIR	=ANTALL.HVIS(C:F; "AIR")
16	KRM	=ANTALL.HVIS(C:F; "KRM")
17	KRMP	=ANTALL.HVIS(C:F; "KRMP")
18	KRP	=ANTALL.HVIS(C:F; "KRP")

	H	I
19		
20	**Matematisk kunnskapsområde:**	
21	Matematiske bevis	=ANTALL.HVIS(C:F; "Matematiske bevis")
22	Formler og variable størrelser	=ANTALL.HVIS(C:F; "Formler og variable størrelser")
23	Funksjoner	=ANTALL.HVIS(C:F; "Likninger og ulikheter")
24	Modellering	=ANTALL.HVIS(C:F; "Funksjoner")
25	Identifisering og formulering av problemer	=ANTALL.HVIS(C:F; "Modellering")
26	Vekstfart og derivasjon	=ANTALL.HVIS(C:F; "Identifisering og formulering av problemer")
27	Polynomdivisjon	=ANTALL.HVIS(C:F; "Vekstfart og derivasjon")
28	Trigonometri	=ANTALL.HVIS(C:F; "Polynomdivisjon")
29	Annet matematisk kunnskapsområde	=ANTALL.HVIS(C:F; "Trigonometri")
30	Programmeringsteknisk	=ANTALL.HVIS(C:F; "Annet matematisk kunnskapsområde")
31		
32	**Oppdrag:**	
33	Utvid programmet	=ANTALL.HVIS(A:A; "Utvid programmet")
34	Lag et program	=ANTALL.HVIS(C:F; "Lag et program")
35	Kjør programkoden med disse ver	=ANTALL.HVIS(A:A;"Kjør programkoden med disse verdiene")
36	Forklar hvordan programmet virk	=ANTALL.HVIS(A:A; "Forklar hvordan programmet virker")
37	Gjøre endringer i eksisterende pro	=ANTALL.HVIS(A:A;"Gjøre endringer i eksisterende progra")
38	Vurder effekten av en endring i pr	=ANTALL.HVIS(A:A;"Vurder effekten av en endring i programkoden")
39	Vurder resultatet	=ANTALL.HVIS(A:A;"Vurder resultatet")
40	Bruk et program	=ANTALL.HVIS(A:A;"Bruk et program")
41		



NTNU

Kunnskap for en bedre verden