Masteroppgave

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

Håvar Engen

Programmeringsinnhold i lærebøker

En kvalitativ innholdsanalyse av programmeringsinnholdet i lærebøker på mellomtrinnet

Masteroppgave i Matematikkdidaktikk (5-10)
Veileder: Anders Sanne
November 2022

NTNU
Kunnskap for en bedre verden

Håvar Engen

Programmeringsinnhold i lærebøker

En kvalitativ innholdsanalyse av programmeringsinnholdet i lærebøker på mellomtrinnet

Masteroppgave i Matematikkdidaktikk (5-10)
Veileder: Anders Sanne
November 2022

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

NTNU
Kunnskap for en bedre verden

Sammendrag

Programmering ble fra og med skoleåret 20/21 innført som en del av matematikkfaget gjennom fagfornyelsen. Denne studien har undersøkt programmeringsinnholdet i norske lærebøker i matematikk på mellomtrinnet. Hensikten har vært å bidra til mer kunnskap om programmering i skolen, og kartlegge styrker og svakheter med programmeringsoppgaver i lærebøker. Gjennom økt kunnskap vil læreres bevissthet kunne øke omkring programmeringsinnholdet og hvor godt oppgavene tilrettelegger for å utvikle elevenes kompetanse i matematikk og programmering. Studiens to forskningsspørsmål er: *1) Hva karakteriserer programmeringsinnholdet i norske matematikkbøker på mellomtrinnet? 2) På hvilke måter knytter programmeringsoppgavene programmering og matematikk sammen?*

Studien har benyttet metoden kvalitativ innholdsanalyse med kvantitative elementer, og har analysert oppgaver i lærebøker for hele mellomtrinnet. Datamaterialet ble analysert ved bruk av et analytisk verktøy utviklet av Bråting og Kilhamn (2021). Det ble gjort noen forandringer i deres rammeverk for å tilpasse det til en norsk kontekst. Analysen består av tre delanalyser, med både deduktiv og induktiv tilnærming, avhengig av om det ble brukt forhåndsdefinerte koder og kategorier eller om det var behov for å utarbeide nye. Gjennom prosessen med tre delanalyser kan man identifisere både det matematiske innholdet og programmeringsinnholdet i hver enkelt oppgave, og man kan undersøke på hvilke måter oppgaven knytter matematikk og programmering sammen.

Analysen viser at programmeringskompetansen som formidles i læreplanen, i mindre grad gjenspeiles i lærebøkene. Koblingen mellom matematikk og programmering er svak, på det vis at oppgavene ikke øker muligheten til å utforske matematikk gjennom programmering. Det finnes eksempler fra datainnsamlingen hvor det er god korrelasjon mellom det som formidles i læreplanen og innholdet i oppgaven, men dette gjelder kun rundt tjue prosent av oppgavene. Førti prosent av programmeringsoppgavene har blitt kategorisert som oppgaver uten matematisk innhold. Det ser også ut til at lærebøkene har en tendens til å vektlegge noen programmeringsbegrep mer enn andre. Funn fra analysen viser vider at «forme og skape» er den handlingen som forekommer flest ganger. Ettersom denne studien viser at lærebøkene ikke ivaretar alle aspekter ved den kompetansen som formidles i læreplanen, indikerer det at lærere har en viktig jobb med å tilrettelegge for gode oppgaver og tilpasse oppgaver slik at manglene i lærebøkene blir kompensert for.

Nøkkelord: lærebøker i matematikk, programmering, lærebokanalyse

Abstract

Programming was introduced from the school year 20/21 as part of the mathematics subject through the subject renewal. This study has examined the programming content in Norwegian mathematics textbooks at intermediate level. The purpose has been to contribute to more knowledge about programming in schools, and to map strengths and weaknesses with programming tasks in textbooks. Through increased knowledge, teachers' awareness can increase regarding the programming content and how well the tasks facilitate the development of pupils' competence in mathematics and programming. The study's two research questions are: 1) What characterizes the programming content in Norwegian mathematics books at intermediate level? 2) In what ways do the programming tasks bridge programming and mathematics?

The study has used the method of qualitative content analysis with quantitative elements, and has analyzed tasks in textbooks at the entire middle level. The data was analyzed using an analytical tool developed by Bråting and Kilhamn (2021). Some changes were made to their framework to adapt it to a Norwegian context. The analysis consists of three sub-analyses, with both a deductive and an inductive approach, depending on whether predefined codes and categories were used or whether there was a need to make new ones. Through the process of three partial analyses, one can identify both the mathematical content and the programming content in each individual task, and one can search in which ways the task bridge mathematics and programming.

The analysis shows that the programming skills imparted in the curriculum are to a lesser extent reflected in the textbooks. Bridging mathematics and programming is considered weak, becuase the tasks does not increase the opportunity to explore mathematics through programming. There are examples from the data collection where there is a good correlation between what is conveyed in the curriculum and the content of the assignment, but this only applies to around twenty percent of the analysis units. Forty percent of the programming tasks have been categorized as tasks without mathematical content. It also seems that the textbooks tend to emphasize some programming concepts more than others. The results uncovers "form and create" as the dominating action. This study shows further that the textbooks don’t include all aspects of the competence conveyed in the curriculum. This indicates that teachers have an important job of facilitate good tasks and adapting tasks which compensate for the deficiencies in the textbooks.

Keywords: textbooks in mathematics, programming, textbook analysis

Forord

Min eldste bror har i mange år forsøkt å få meg interessert i programmering, noe jeg ikke viste noen spesiell interesse for før jeg plutselig tok valget om å skrive master om nettopp programmering. Det var først da jeg oppdaget hvor mye glede og mestringsfølelse en kan få av å tukle med koder for å lage enkle dataprogram, eller programmere en Lego-robot til å bevege seg slik jeg ønsker. Jeg, blant de første masterstudentene etter innføring av ny læreplan, har vært så heldig å få bruke masteroppgaven til å utvikle kompetanse om noe som er helt nytt i fagfornyelsen. Jeg håper at den kunnskapen jeg har tilegnet meg gjennom denne prosessen, er noe jeg får bidratt med inn på arbeidsplassen når jeg selv begynner å jobbe som lærer.

Selv om det er jeg som har stått på for å skrive denne masteroppgaven, er det andre som fortjener en liten oppmerksomhet etter over fem år som lærerstudent i Trondheim.

Først og fremst vil jeg takke de nærmeste medstudentene mine, som har bidratt til et godt læringsmiljø og mange gode stunder de siste årene. Vi har stått sammen gjennom krevende eksamensperioder, og kost oss masse sammen på de nydeligste vårdagene. Jeg setter umåtelig stor pris på dere!

Jeg vil også takke min veileder Anders Sanne, som har hjulpet meg på rett spor, og vist forståelse for de valgene jeg har tatt gjennom prosessen. Når alt kommer til alt, så har du vært akkurat den personen jeg trengte som veileder.

Til slutt vil jeg rette en takk til NTNUI-håndball og alle de herlige menneskene jeg har vært så heldig å bli kjent med gjennom klubben de siste årene. Det å leve som student, med lite penger og alltid skolearbeid hengende over meg, har vært en leik med dere til stede i hverdagen! Masteroppgaven har kanskje ikke vært en leik for det, men nå er jeg uansett endelig i mål.

November, 2022

Håvar

INNHOLD

Figurer

Tabeller

1 INNLEDNING

Etter hvert som verden blir mer og mer digitalisert, har programmering de siste årene vokst fram som en viktig ferdighet. Programmering har tradisjonelt sett kun vært relevant for IT-bransjen, men de senere årene har synet på dette forandret seg. Slik digitaliseringen foregår i verden på dette tidspunkt, inngår software og teknologi i de fleste områder i samfunnet og de livene vi lever (Nouri et al., 2020). I flere land, deriblant Norge, har programmering blitt en del av læreplanen, fra småtrinnet til ut videregående opplæring. Dette betyr at det ikke lenger kun er de som velger programmering i utdanningsløpet som skal få kunnskap innenfor dette fagfeltet. I fagfornyelsen (LK20) er programmering et gjennomgående tema i matematikkfaget. Allerede fra 2. trinn er det kompetansemål knyttet til programmering med "trinnvise instruksjoner", og fra fjerde trinn er det ett kompetansemål på hvert trinn hvor programmering, eller begrep innenfor programmering, nevnes eksplisitt i læreplanen (Utdanningsdirektoratet, 2020). Fra 4. trinn skal elevene kunne lage algoritmer og uttrykke dem ved bruk av variabler, løkker og vilkår, og i 6. trinn skal elevene bli introdusert for funksjoner i programmering (Utdanningsdirektoratet, 2020).

I tillegg til at programmering har sin plass som konkrete læreplanmål, kommer *algoritmisk tenkning* (Computational Thinking) frem i kjerneelementutvalget til faget under *utforsking og problemløsing* (Utdanningsdirektoratet, 2019). Kjerneelementene er ideer om ting som skal være gjennomgående i matematikkopplæringen i grunnskolen. Koding og programmering er nært beslektet til algoritmisk tenkning (Gjøvik og Torkildsen, 2019). Gjennom programmering bli man utsatt for algoritmisk tenkning (Wing, 2006), og det er nettopp programmering som er den vanligste tilnærming til arbeid med algoritmisk tenkning i skolen (Lye og Koh, 2014). Med tanke på at algoritmisk tenkning er en del av kjerneelementene kan det tyde på at programmering er tenkt til å kunne ha en større plass i matematikkundervisning enn kun arbeid med de spesifikke programmerings-målene på hvert trinn.

I Norge har man valgt å implementere programmering i matematikkfaget slik at programmeringens egenart skal læres gjennom matematikkundervisningen. I tillegg kommer det tydelig frem i læreplanmålene at gjennom grunnskolen skal programmering anvendes til ulike spesifikke ting innenfor matematikkfeltet. Et eksempel på dette er på 6. trinn med læreplanmålet: "bruke variabler, løkker, vilkår og funksjoner i programmering til å utforske geometriske figurer og mønstre" (Utdanningsdirektoratet, 2020). Det er altså ikke kun mål om å lære å programmere, men elevene skal bruke programmering til å utforske og lære matematikk. Dette betyr i praksis at programmering skal kunne være et verktøy for å jobbe med og lære om nye matematisk objekt, som for eksempel vinkelsummen til ulike geometriske figurer.

Når vi ser på hvordan nabolandene våre Danmark og Sverige har implementert programmering i skolen, ser vi at Danmark har fag som heter "Technology comprehension" hvor programmering er en del av faget (Bocconi et al., 2018). På denne måten læres programmeringsfagets egenart gjennom et eget teknologifag. I Sveriges læreplan kan vi derimot se at valget har vært å gjøre programmering til et aspekt ved algebra gjennom læreplanmål i matematikk på alle trinn (Swedish National Agency of Education, 2018). Det finnes altså ulike måter å gjøre programmering til en del av skolepensum, men uansett om programmering går innenfor et IT-fag eller matematikk, så vil programmering likevel være en disiplin som er nært knyttet til matematikk

(Misfeldt & Ejsing-Dunn 2015; Wing, 2006). På grunn av programmeringens nære forhold til matematikk kan det være et poeng å designe programmeringsundervisning etter Benton et al.'s (2016) prinsipper slik at matematikk og programmering blir sett i sammenheng. På denne måten kan programmering bli tatt i bruk som et verktøy for å lære matematikk på en utforskende måte slik det formidles gjennom kjerneelementene i fagfornyelsen (Utdanningsdirektoratet, 2019).

Læreboka har stor innvirkning på de didaktiske valgene som blir gjort (Lepik et al., 2015), og lærebøker blir ansett som å være noe av det som har sterkest innvirkning på skolematematikken (Mullis, Martin & Foy, 2008; Valverde, Bianchi, Wolfe, Schmidt & Houang, 2002, referert i Lepik et al., 2015). Til tross for de enorme mengdene digitale ressurser som er tilgjengelig på internett, står fortsatt læreboka sterkt i planleggingsarbeidet til lærere og når det kommer til valg av oppgaver elevene skal jobbe med. Lærebøker i matematikk anses som en viktig ressurs både for elevenes læring av matematikk, og lærerens planlegging og undervisning i faget (Lepik et al., 2015; Rezat, 2012). Nå som skolen har vært, og er, i en overgangsfase til ny læreplan, kan gode lærebøker som er skrevet for å dekke de nye kompetansemålene og ta hensyn til fagfornyelsen, være et nyttig hjelpemiddel for lærere. Dette spesielt med tanke på programmering som er et nytt tema i læreplanen, med kompetansemål på hvert trinn fra og med 2. trinn.

1.1 Forskningsspørsmål og formål med undersøkelsen

Som det kommer frem i forrige avsnitt har læreboka stor innflytelse på hva som faktisk undervises i matematikkfaget, og er en nyttig ressurs både for elever og lærere (Lepik et al., 2015; Rezat, 2012). Derfor er det interessant å undersøke hvor godt lærebøkene kan bidra i elevers utvikling av kompetanse og ferdigheter innenfor programmering, og undersøke hvordan programmeringsinnholdet i lærebøkene sammenfaller med det som formidles i fagfornyelsen.

Sammen med ny læreplan følger nye læreverk, både i form av lærebøker og nettressurser. Noe som er interessant å undersøke er hvorvidt programmerings-oppgaver i noen nye lærebøker er designet slik at programmering og matematikk knyttes sammen, eller om det er mer eller mindre rene programmeringsoppgaver uten matematisk innhold. Det er gjort en studie i Sverige av Bråting og Kilhamn (2021) etter innføring av nye læreplan hvor også det matematiske innholdet i nye lærebøker ble analysert. I min studie benytter jeg meg av et liknende rammeverk som Bråting og Kilhamn (2021) bruker i sin studie. Det analytiske verktøyet som er brukt er en kombinasjon av Brennan and Resnick's (2012) rammeverk om Computational Thinking, sammen med Benton et al.'s rammeverk for handling, som blir kalt "5E's". Disse rammeverkene hver for seg egner seg ikke automatisk for en lærebokanalyse, men inspirert av begge har Bråting og Kilhamn (2021) konstruert et nytt analytisk verktøy som jeg har beskrevet detaljert i teorikapittelet. I et forsøk på å finne ut hvordan programmeringsoppgaver i nye lærebøker knyttes til matematikk som egenart stiller jeg de samme forskningsspørsmålene som Bråting og Killhamn (2021):

(1) Hva karakteriserer programmeringsinnholdet i norske matematikkbøker på mellomtrinnet?
(2) På hvilke måter knytter programmeringsoppgavene programmering og matematikk sammen?

Formålet med studien er å bidra til mer kunnskap om programmering i skolen, og kartlegge styrker og svakheter med oppgaver i lærebøker. Dette er for å skape en bevisstgjøring om lærebokas begrensninger, slik at man som lærer er i stand å gjøre de tilpasningene i programmeringsoppgaver som skal til, for å tilrettelegge for best mulig undervisning.

Videre i kapittel to, teorikapittelet, vil jeg beskrive det teoretiske rammeverk i mer detalj og begrunne valg av rammeverk. Kapittel 3 vil metode for datainnsamling og analyse bli beskrevet, i tillegg til at jeg vurderer studiens troverdighet. I kapittel 4 vil funnene mine bli presentert sammen med eksempler fra datamaterialet. Kapittel 5 er diskusjonsdelen av studien, hvor funnene blir diskutert opp mot relevant litteratur. Her vil det være naturlig å gjøre sammenligninger med studien til Bråting og Kilhamn (2021). Helt til slutt vil det være en avslutning.

2 TEORI

Denne studien har til hensikt å undersøke hva som karakteriserer innholdet til programmeringsoppgaver i lærebøker på mellomtrinnet, og på hvilke måter programmeringsoppgavene legger til rette for å konstruere matematisk kunnskap. To sentrale begreper i oppgaven er *algoritmisk tenkning* og *programmering*, og i tillegg er det noen andre sentrale begreper som jeg skal gjøre rede for senere i kapittelet. Jeg starter med å beskrive hva programmering og algoritmisk tenkning er, sammen med en redegjørelse av Brennan og Resnick (2012) sitt rammeverk for algoritmisk tenkning (AT). Videre vil jeg gjøre rede for rammeverket "The 5E's" utviklet av Benton et al. (2016). Mot slutten av kapittelet beskriver jeg rammeverket for denne studien og begrunne valg av rammeverket.

2.1 HVA ER ALGORITMISK TENKNING?

Begrepet algoritmisk tenkning har i fagfornyelsen dukket opp i kjerneelementene i matematikkfaget under "utforsking og problemløsning" og er dermed et nytt begrep i læreplanen som skal være gjennomgående i matematikkopplæringen i grunnskolen.

> Algoritmisk tenkning er viktig i prosessen med å utvikle strategier og framgangsmåter for å løse problemer og innebærer å bryte ned et problem i delproblemer som kan løses systematisk. Videre innebærer det å vurdere om delproblemene best kan løses med eller uten digitale verktøy (Utdanningsdirektoratet, 2019).

Internasjonalt brukes begrepet Computational Thinking (CT) om det vi omtaler som algoritmisk tenkning på norsk. Det er ikke noen klar definisjon på algoritmisk tenkning (Barr & Stephenson, 2011; Coulter et al., 2010), men Cuny et al. (2010) definerer algoritmisk tenkning som «tankeprosessene involvert i å formulere problemer og deres løsninger slik at løsningene er representert i en form som effektivt kan utføres av en informasjonsbehandler». I et forsøk på å forklare hva dette betyr i praksis, kan algoritmisk tenkning innebære å løse problemer, designe systemer, tenke rekursivt, gjøre vanskelige problem til noe vi enklere kan håndtere, og bruke abstraksjon og dekomponering når man angriper en stor og kompleks oppgave (Wing, 2006). Innenfor paraplyen algoritmisk tenkning deles det vanligvis inn i fem nøkkelbegrep; «*abstraction», «algorithmic thinking», «generalization», «automation»* og *«decomposition»*, som Gjøvik og Torkildsen (2019) har oversatt til norsk og endt opp med begrepene: A*bstraksjon, algoritmebehandling, generalisering, automatisering* og *dekomponering.*

Abstraksjon innebærer å kunne se bort ifra opplysninger som ikke er relevante og trekke ut selve essensen i de ulike tilfellene (Gjøvik og Torkildsen, 2019). Med andre ord så handler det om å gjøre noe enklere, men at den samme jobben likevel blir gjort. I programmering vil abstraksjon for eksempel kunne være å bruke funksjoner eller egendefinerte blokker for å gjøre koden kortere og enklere å lese.

Når det kommer til *algoritmebehandling*, som forvirrende nok på engelsk er "algorithmic thinking", innebærer dette begrepet et utvalg av ferdigheter som er forbundet med å forstå og kunne skape algoritmer (Futschek, 2006; Gjøvik og Torkildsen, 2019). Oppgaver knyttet til algoritmebehandling vil kunne innebære å tolke, forklare, sammenligne, forbedre og lage algoritmer.

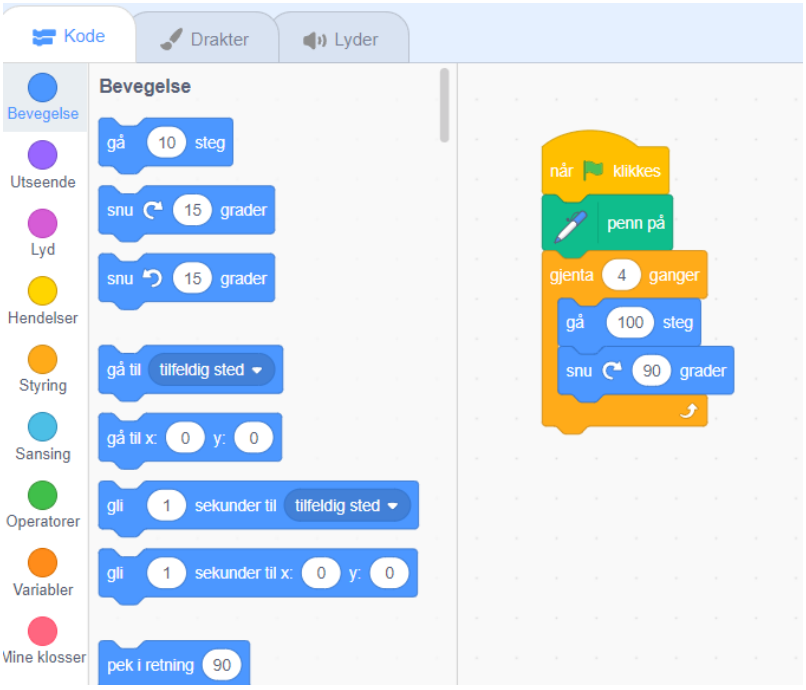
Generalisering handler om å gjenkjenne mønstre og sammenhenger man oppdager, og lage regler og metoder som gjelder på en hel klasse av tilfeller (Gjøvik og Torkildsen, 2019).

Automatisering kan innebære det å gjøre det menneskelige bidraget minimalt (Gjøvik & Torkildsen, 2019). For å tydelig forklare dette begrepet vil jeg vise til et eksempel med en innendørs dyrkestasjon for en liten plante, hvor man ønsker at det menneskelige bidraget skal være minimalt, og at planten likevel skal overleve. Det første man kan gjøre er å sørge for at planten får riktig mengde lys gjennom døgnet, uten at noen trenger å skru av og på plantelyset. Videre vil man kanskje installere en fuktmåler i jorda, og koble opp denne mot et automatisk vanningssystem. På denne måten automatiserer man dyrkestasjonen slik at behovet for menneskelig bidrag minimeres.

Dekomponering innebærer å kunne bryte opp et problem i mindre deler som hver for seg er enklere å håndtere. Wing (2006) omtaler dette som seperasjon av problemer. For eksempel vil designing av et dataspill være et komplekst problem. Her vil det være nyttig å dele opp i mindre problemer eller oppgaver som er enklere å løse hver for seg, før det settet sammen til en helhet.

2.1.1 Programmering og algoritmisk tenkning

"Et naturlig miljø for å implementere algoritmisk tenkning er i programmering" (Gjøvik og Torkildsen, 2019, s. 34). Det har opp gjennom blitt utviklet ulike programmeringsverktøy som har til hensikt å være tilpasset barn, som for eksempel *LegoMindstorm, Micro:bit, Etoys* og *Scratch.* Felles for disse programmeringsverktøyene er at de er blokk-baserte verktøy som vil si at brukeren setter sammen ferdig koda blokker for å lage et program. Når man opererer innenfor disse verktøyene, er det ikke behov for å lære seg ulike syntakser og programmeringsspråk for å utvikle kompetanse innenfor algoritmisk tenkning og programmering. Under viser figur 2.1 hvordan et blokk-basert programmeringsverktøy kan se ut.



Figur 2.1 Skjermbilde fra Scratch

Det sistnevnte verktøyet, Scratch, har ifølge sitt eget nettsted, 95 millioner registrerte brukere over hele verden, hvor over 16 millioner av disse er barn som er 11 eller 12 år gamle (Scratch, 2022). Scratch har til hensikt at brukeren skal kunne lage prosjekter som de blir personlig engasjert i, motivert og føle at er meningsfullt (Maloney et al., 2010). Årsaken til at jeg nevner Scratch er at Brennan og Resnick (2012) har utviklet et rammeverk for algoritmisk tenkning med Scratch som programmeringsverktøy. Dette rammeverket skal jeg gjøre rede for i neste delkapittel, da jeg bruker prinsipper fra dette i min analyse.

2.1.2 Rammeverk for algoritmisk tenkning

Brennan og Resnick (2012) har gjennom sine studier av programmeringsaktivitet i interaktive medier utviklet et rammeverk for algoritmisk tenkning. I deres studie er Scratch kontekst for programmering. Brennan og Resnick (2012) sitt rammeverk for algoritmisk tenkning er i hovedsak et rammeverk for designing av prosjekter i Scratch. Likevel er det noen prinsipper og begrep som er aktuelt å bruke i min analyse av oppgaver i lærebøker. Dette vil bli utdypet i metodekapittelet. Jeg skal i dette delkapittelet beskrive det Brennan og Resnick (2012) kaller for nøkkeldimensjoner ved deres rammeverk for algoritmisk tenkning: *begrep innenfor algoritmisk tenkning* (begrepene designere engasjerer seg når de programmerer, slik som iterasjon, parallellisme, etc.), *praksis for algoritmisk tenkning* (praksisen designere utvikler når de engasjerer seg i begrepene, som å feilsøke prosjekter eller remikse andres arbeid), og *perspektiver i algoritmisk tenkning* (perspektivene designere danner om verden rundt dem og om seg selv).

Begrep innenfor algoritmisk tenkning

Brennan og Resnick (2012) identifiserte i sin studie syv begrep som de kvalifiserer som høyt nyttige i et bredt spenn av Scratch-prosjekter, og som kan overføres til andre programmeringskontekster. De identifiserte begrepene var: *sekvenser, løkker*, *vilkår, handlinger, parallellisme, operatorer* og *data.*

Praksis for algoritmisk tenkning

Denne dimensjonen i rammeverket hjelper for å beskrive prosessene i elevers arbeid med AT-oppgaver. Brennan og Resnick (2012) observerte fire praksiser for å beskrive disse prosessene: *være inkrementell og iterativ*, *testing og feilsøking*, *gjenbruk og remiksing*, og *abstrahering og modulisering*.

Perspektiver i algoritmisk tenkning

Dimensjonen om «perspektiver» ble lagt til i rammeverket for å beskrive endringene i perspektiv som de observerte hos unge mennesker som jobbet med Scratch (Brennan og Resnick, 2012). De kom frem til tre perspektiver. Det første var at de observerte elevers behov for å uttrykke seg, dermed perspektivet *uttrykke.* Det andre perspektivet var «connecting», som handler om at elevene nyttiggjøre seg av å ha tilgang til andre, andres perspektiver og andres prosjekter via internett (Brennan og Resnick, 2012). Det siste perspektivet Brennan og Resnick (2012) inkluderte i sitt rammeverk var «questioning». Dette perspektivet innebærer å tørre og stille spørsmål om det som er tatt for gitt (Brennan og Resnick, 2012).

2.2 Å BRUKE PROGRAMMERING FOR Å KONSTRUERE MATEMATISK KUNNSKAP

De siste årene har programmering fått en plass i læreplaner i ulike land, og er blitt kompetanse alle elever som går gjennom ordinært utdanningsløp skal ha opplæring i allerede fra småtrinnet. I England ble alle skoler i 2014 pålagt å undervise et IT-fag, hvor elevene skal lære om hvordan datasystemer fungerer, de skal kunne bruke teknologi på en sikker og forsvarlig måte, samt å kunne utvikle sine egne dataprogrammer (Benton et al., 2016). Etter hvert som programmering tar form i læreplanen vokser det fram et større behov for programmeringsverktøy som bedre tilpasset barn som brukerbase. Gjennom flere år har forskere i England arbeidet med å utvikle et prosjekt kalt Scratchmaths hvor målet er å bygge matematisk kunnskap gjennom programmering i Scratch (Benton et al., 2016). Scratchmaths består av flere sett med ulike moduler som er ment til å gjennomføres over et toårs perspektiv på 5. og 6. trinn i England. Det første året er fokuset på algoritmisk tenkning med en implisitt matematisk komponent. Det andre året fokuseres det mer på nøkkelbegrep og sentrale ideer innenfor matematikk, ved bruk av digitale verktøy (Benton et al., 2016). I Scratchmaths er aktivitetene og oppgavene designet ut fra fem ulike designprinsipper, som jeg skal forklare i mer detalj i de neste avsnittene.

2.2.1 De fem 5 E'ene

De fem prinsippene i rammeverket til Benton et al. (2016) handler om ulike handlinger oppgaver eller læringsaktiviteter burde inneholde for å fremme læring. Hver av handlingene har sine styrker, og det er en variasjon og kombinasjon av handlingene som utfordrer elevene på nye områder, i tillegg til at kunnskapen kan blir mer tilgjengelig for elevene ettersom det blir flere ulike innfallsvinkler på utfordringene som skal løses. «De fem E'ene» ble utviklet for å veilede om hvilke pedagogiske strategier lærere kan bruk for å lykkes med implementeringen av ulike aspekter ved Scratchmaths (Benton et al. 2016). Rammeverket jeg bruker i min studie bygger på de prinsippene jeg beskriver i de kommende avsnittene.

Explore [Utforske]

Papert (1980) tror at barn burde bruke pc for å utforske tankeprosessene sine, og foreslår (med Logo som kontekst) at den viktigste erfaringen knyttet til læring er "å bli kjent med skilpadden" ved å utforske hva skilpadden kan gjøre og hva den ikke kan gjøre (Benton et al., 2016), s. 5). Dette er en konstruktivistisk tilnærming til læring, som ifølge Brennan (2015) tilrettelegger for at elevene skal kunne utforske måter å håndtere ulike begrensninger og tvetydigheter gjennom å bruke ferdigheter som iterativ tenkning, problemløsning og kreativitet (Benton et al., 2016). Gjennom dette læringssynet er det viktig å tilrettelegge for at elevene selv skal ta kontroll over egen læring. Læringsaktiviteter og oppgaver burde være slik at de får utforsket ting på egenhånd, undersøkt nye ideer, og feilsøke konseptuelle og tekniske feil når det er nødvendig (Benton et al., 2016).

Explain [Forklare]

Et avgjørende aspekt av det å forstå ideer er å være i stand til å kunne forklare hva som har blitt lært og grunnen til at en har valgt den gitte tilnærmingen (Benton et al., 2016). Det å uttrykke seg muntlig kan gi positiv effekt når det kommer til å avklare ideer. Flere teoretikere har fremhevet de kognitive fordelene ved å skape muntlige forklaringer, og en av de er Brown (1980) som har vist at det å bli oppmuntret til å forklare og representere kunnskap på ulike måter kan øke motivasjon og forståelse (Benton et al., 2016). Bråting og Kilhamn (2021) skriver at det å forklare et skript og hva det er ment til

å gjøre, steg for steg, vil være en oppgave som legger til rette for egen refleksjon om hvordan man tenker. Definere og navngi blokker er også en viktig del av forklaring i en Scratch-kontekst (Bråting og Kilhamn, 2021).

Envisage [Forestille seg]
I programmering er det viktig å ha et mål i bakhodet om hvordan programmet skal bli når man utvikler det, og kunne forutse hva et utfall blir før man tester det (Benton et al., 2016; Bråting og Kilhamn, 2021). Ettersom programmeringsverktøyene som er tilpasset barn tar hånd om mange av syntaks-feilene, vil behovet for feilsøking som regel kun oppstå når man har et klart mål, men at man har frembrakt et utfall som ikke er det planlagte utfallet (Rader et al., 1997). Papert (1980) mener at for å virkelig forstå en ide er det viktig å bruke tid til å forutse utfallet før dataprogrammet blir utviklet, og deretter sammenligne det faktiske utfallet med det som ble forutsett. Derfor er det viktig at eleven blir utfordret på å se for seg og skape et bilde av hvordan programmet eller kodesekvensen skal fungere i forkant, slik at det blir tilrettelagt for å lære denne evnen innenfor programmering.

Exhange [Dele]
Å engasjere seg i forklaring av egne strategier, stille spørsmål til hverandre og klarering av uenigheter i samhandling med andre for å fremme kreativitet og utforskertrang (Bråting og Kilhamn, 2021). Benton et al. (2017) skriver at meningsfulle muligheter til å dele og bygge på hverandres ideer burde bli inkludert i elevaktiviteter og oppgaveløsning. De skriver videre at det ble gjort observasjoner av at elever med høyere måloppnåelse innenfor temaet bidro til å støtte medelever og gi en inngang for å ta opp problemene og utfordringene de møtte, ved å uttrykke og dele det som opplevdes som vanskelig med andre elever (Benton et al., 2017).

bridgE [Knytte sammen]
«Bridge» eller «å knytte sammen» innebærer å finne forbindelser mellom programmering og store matematiske ideer. Ideer blir sett på som store, eller «*powerful*», delvis gjennom deres forbindelse til andre disipliner (Benton et al., 2017). I dette tilfellet matematikk. For å utvikle disse forbindelsene må ideene settes i ny kontekst og bli konstruert gjennom matematisk språk (Bråting og Kilhamn, 2021). «Knytte sammen» krever at læringsaktivitetene eller læreren legger til rette for å lage eksplisitte koblinger til en annen kontekst, i dette tilfellet skolematematikk (Benton et al., 2016).

2.3 RAMMEVERK FOR ANALYSE AV PROGRAMMERINGSOPPGAVER

I Bråting og Kilhamn (2021) sin studie forsøkte de først å bruke de to rammeverkene som jeg har beskrevet tidligere hver for seg i analyseprosessen. Problemet med dette var at ingen av rammeverkene kunne beskrive innholdet på en tilstrekkelig omfattende måte (Bråting & Kilhamn, 2021). AT-rammeverket (Brennan & Resnick, 2012) vektlegger ikke matematiske ideer, og 5E's (Benton et al., 2017) har ikke et tilstrekkelig fokus på begreper innenfor algoritmisk tenkning. I tillegg fant de mange oppgaver fra lærebøkene som ikke inkluderte noen av praksisene fra algoritmisk tenkning eller «5E's» prinsipper. Bråting og Kilhamn (2021) foreslår at årsaken til dette kan være at disse rammeverkene beskriver en ideell situasjon. De beskriver en situasjon som forskere gjerne vil at elevene engasjerer seg i og utvikler kunnskap om. Ettersom rammeverkene ikke fungerte tilstrekkelig hver for seg, endte de opp med å kombinere de to nevnte rammeverkene for å lage et analytisk verktøy bestående av *handlinger* og *begreper*, og deretter diskutere

på hvilke måter matematikk og programmering knyttes sammen som en overordnet idé (Bråting & Kilhamn, 2021, s. 6).

2.3.1 Handlinger

Bråting og Kilhamn (2021) kategoriserte de ulike handlingene oppgavene krever at elevene skal utføre. I noen av oppgavene fant de spor av praksis innenfor algoritmisk tenkning som feilsøking, og å være inkrementell og iterativ, og «5E's» prinsipper; utforske, forestille seg og følge en prosedyre. Under er en oversettelse av Bråting og Kilhamn (2021) sine kategorier for handlinger og en kort beskrivelse av disse.

(a) Følge en prosedyre - følg trinnvise instruksjoner, repeter eller fortsette et mønster
(b) Finne regel – finn ut prosedyren, regelen eller mønsteret som generer et utfall, for eksempel en tallsekvens.
(c) Feilsøke - feilsøk en kode. Her kan det undersøkes om hva som gjør at programmet ikke fungerer slik det skal eller er tenkt
(d) Forme og skape – gi instruksjoner, lag et mønster, skriv kode, representer med symboler.
(e) Forklare – forklar ved bruk av naturlig språk, ved bruk av ord for å beskrive en prosedyre, en regel, et mønster eller et konsept.
(f) Forestille seg – forutse hva som vil skje, reflekter rundt mulige utfall når betingelser eller verdier blir forandret.

2.3.2 Begrep

Det analytiske verktøyet inkluderer begrep innenfor to ulike grener: *matematiske begrep* og *programmeringsbegrep. B*egrepene i oppgavene blir identifisert og klassifisert gjennom en deskriptiv analyse (Bråting og Kilhamn, 2021). På denne måten er det mulig å skaffe en oversikt over det matematiske innholdet og AT-innholdet i programmeringsoppgaver i lærebøkene.

2.4 RAMMEVERK FOR DENNE STUDIEN

I denne studien har jeg valgt å benytte meg av Bråting og Kilhamn (2021) sitt rammeverk for analyse av programmeringsoppgaver ettersom jeg ønsker å finne ut (1) hva som karakteriserer programmeringsinnhold i norske matematikkbøker på mellomtrinnet, og (2) på hvilke måter programmeringsoppgavene knytter matematikk og programmering sammen. Dette er interessant å undersøke dette blant annet fordi programmering i skolen har vist seg å ha potensial til å utvikle høyere nivåer av matematisk tenkning i forhold til aspekter ved tall knyttet til multiplikativt resonnement, matematisk abstraksjon inkludert algebraisk tenkning, og evner knyttet til problemløsning (Benton et al., 2016). Dette kan oppnås gjennom arbeid med programmeringsoppgaver med matematisk innhold på en utforskende måte i henhold til prinsippene i «5E's».

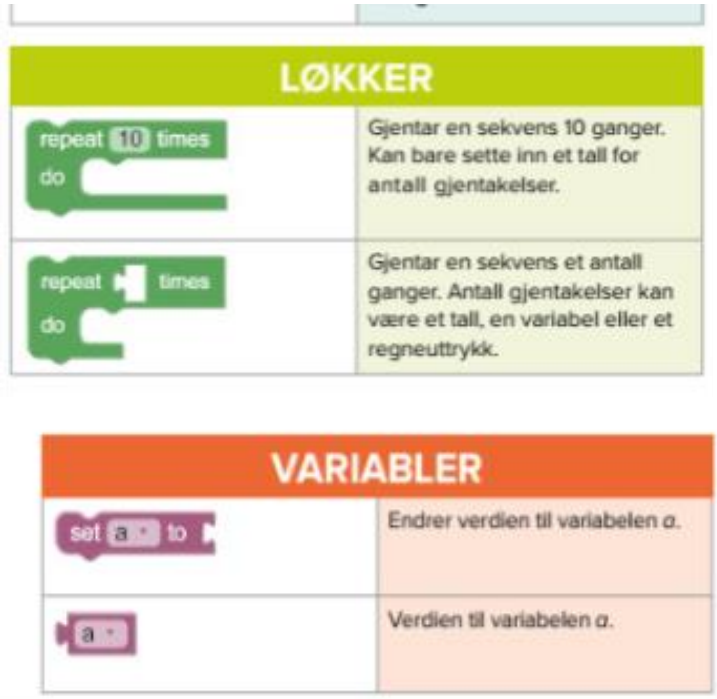
Ettersom Bråting og Kilhamn (2021) sin studie er gjort i Sverige med ulik læreplan og lærebøker enn i Norge, har jeg gjort noen forandringer i kodene både innenfor

programmeringsbegrep og matematiske begrep og endt opp med: *Algoritme, program, løkker, vilkår, variabler og funksjoner.* Dette blir forklart i mer detalj i metodedelen.

Programmeringsbegrepene Bråting og Kilhamn (2021) forholdt seg til i sin analyse var *stegvise instruksjoner, algoritmer, løkker, regel, kode, vilkår* og *feil og feilsøking*. Variabler i en programmeringskontekts er en del av pensum allerede på fjerde trinn, hvor elevene skal kunne lage algoritmer og uttrykke de ved hjelp av variabler, løkker og vilkår (Utdanningsdirektoratet, 2020). Funksjoner er en del av læreplanen fra 6. trinn gjennom læreplanmål som uttrykker at elevene skal bruke variabler, løkker, vilkår og funksjoner i programmering til å utforske geometriske figurer og mønstre (Utdanningsdirektoratet, 2020). Funksjonsbegrepet har ulik betydning gitt om det er i en matematisk eller en programmeringskontekst.

Når det kommer til programmeringsbegrepene er forskjellen fra analysen til Bråting og Kilhamn (2021) at *rotasjoner* er fjernet, og *symmetri* er en ny kode. Begrepet rotasjoner er ikke sentralt i fagfornyelsen, men kunne kommet inn som et relevant begrep dersom det jeg gjorde analyse av oppgaver knyttet til for eksempel kongruensavbildning. Symmetri er et begrep som eksplisitt blir nevnt i læreplanen på 6. trinn (Utdanningsdirektoratet, 2019). Dette inkluderes som et matematisk begrep i rammeverket.

I Bråting og Kilhamn (2021) står det at programmeringsbegrep og matematiske begrep som eksplisitt blir nevnt i oppgaven, eventuelt i tilhørende eksempler og forklaringer, blir identifisert og kategorisert. Mange tilfeller i lærebøkene jeg har analysert forklares oppgavene gjennom en kombinasjon av tekst og en sekvens blokk-kode. De ulike blokkene representerer ulike programmeringsbegrep slik figuren under viser. Det betyr at innholdet i oppgavene blir identifisert også gjennom blokkene som illustreres i oppgavene ettersom disse er definerte representasjoner for begrepene innenfor dette blokkprogrammerings-verktøyet.



Figur 2.2: Skjemaer om løkker og variabler (Kongsnes et al., 2021b).

2.4.1 Begrunnelse for valg av rammeverk

Ettersom jeg ønsket å gjøre en lignende studie som Bråting og Kilhamn gjorde i Sverige, virket det fornuftig å ta utgangspunkt i et ferdig rammeverk utviklet av forskere som har erfaring med innholdsanalyse av lærebøker. Rammeverket er veldig spesifikt utviklet i den hensikt å skulle undersøke programmeringsinnhold i matematikkoppgaver. Alternativet hadde vært og tatt utgangspunkt i et mer generelt rammeverk for innholdsanalyse og utviklet mitt eget spesifikke rammeverk. Etter å ha undersøkt de to rammeverkene Bråting og Kilhamn (2021) sitt rammeverk er utviklet fra, er det tydelig at det er gjennomtenkt hvordan de har utviklet sitt eget analytiske verktøy for å gjøre det best mulig tilpasset analyse av programmeringsoppgaver.

3 METODE

I arbeidet med å besvare forskningsspørsmålet, har jeg tatt utgangspunkt i analyseverktøyet utviklet av Bråting og Kilhamn (2021). Jeg har utviklet skjemaer for innsamling og koding av datamateriell, for å gjøre en oversiktlig analyse av de ulike delene av hver analyseenhet. Analysen er både deduktiv og induktiv, avhengig av hvilke aspekt av analyseenhetene som undersøkes. Jeg vil i dette kapittelet beskrive metode, utvalg av datamateriale, analysens struktur og analyseprosessen. Til slutt i kapittelet har jeg vurdert studiens troverdighet og etiske betraktninger.

3.1 FORSKNINGSMETODE

Hovedandelen av forskning gjort på lærebøker viser seg å være innholdsanalyse-studier som har til hensikt å undersøke hvordan et bestemt matematisk innhold fremstilles eller behandles (Fan et al., 2013). Rezat og Strässer (2017) viser til at forskning bygget på innholdsanalyser kan gi svar på spørsmål om innhold i lærebøker. Mine forskningsspørsmål handler om hva som karakteriserer programmeringsinnholdet i norske matematikkbøker på mellomtrinnet, og på hvilke måter disse oppgavene knytter matematikk og programmering sammen. Derfor faller min studie også innenfor kategorien beskrevet av Fan et al. (2013). På bakgrunn av dette har jeg valgt å bruke metoden innholdsanalyse.

I litteraturen blir begrepet innholdsanalyse brukt forskjellig. Drisko og Maschi (2016) skriver at mange forskere tenker på "grunnleggende innholdsanalyse" som en kvalitativ forskningsmetode, noe som er en korrekt men begrenset forståelse. Forskere bruker ordteller som en viktig analytisk teknikk i grunnleggende innholdsanalyse. Likevel bruker forskere innholdsanalyse uten statistiske analyser i fortolkende innholdsanalyser og kvalitative innholdsanalyser (Drisko og Maschi, 2016). I utgangspunktet ble innholdsanalyser brukt i kvantitativ forskning, men har også blitt utviklet som analyseverktøy i kvalitativ forskning (Fauskanger & Mosvold, 2015).

I denne studien har det blitt brukt en kvalitativ innholdsanalyse med noen kvantitative elementer (Bryman, 2012). Koding av kvalitativ data innebærer som regel å skrive korte notater ved siden av den innsamlede dataen og gradvis omgjøre disse notatene om til koder. På denne måten vil den gitte delen av datamaterialet høre til visse navn eller etiketter (Bryman, 2012).

Bryman (2012) beskriver at koding av kvalitativ data bør gjøres gjennom å følge fire steg. Det første forskeren gjør er å bli kjent med datamaterialet (1). Her skal kode-prosessen starte ved at forskeren går gjennom datamaterialet uten å ta notater eller komme med tolkninger. Det andre steget innebærer at forskeren skal lese gjennom datamaterialet på nytt og skrive notater (2). Forskeren skal ta så mange notater om betydningsfulle bemerkninger eller observasjoner som mulig. Ved å gjøre dette har forskeren begynt å kode. På det tredje steget skal forskeren gjennomgå kodene som ble laget i forrige steg (3). Forskeren gjennomgår for å finne ut om det er flere koder som beskriver samme fenomen, om noen av kodene er relatert til begreper og kategorier i eksisterende litteratur, og om det er sammenhenger mellom kodene. På det siste steget

skal forskeren vurdere mer generelle teoretiske ideer i forhold til koder og data (4). Forskeren skal forsøke å skissere sammenhenger mellom begrepene og kategoriene som blir utviklet. Forskeren skal i mer detalj ta i betraktning hvordan de forholder seg til eksisterende litteratur.

Bryman (2012) poengterer at kodinga ikke er analysen, men at det er en viktig del av analysen. Koding er en måte å tenke på meningen med dataen og redusere den enorme mengden data man har samlet (Huberman og Miles, 1994, sitert i Bryman, 2012, s. 535).

Ettersom jeg har gjort en studie inspirert av en studie gjort i Sverige av Bråting og Kilhamn (2021) har deler av analysen blitt gjort med en deduktiv tilnærming med allerede eksisterende koder og kategorier. Andre deler er tilnærmingen mer induktiv når kodene må endres og erstattet for å passe bedre til denne studien.

Bryman (2012) skriver at en av de største styrkene med kvalitativ forskning er dybden og rikdommen til dataen som produseres, men disse positive aspektene medfører også til at det kan være veldig omfattende å analysere. En annen styrke ved koding av kvalitativ data er at selve kodeprosessen vil hjelpe med å utvikle en detaljert forståelse av datamaterialet (Bryman, 2012). Bryman (2012) trekker frem tre svakheter eller problemer som er assosiert med koding av kvalitativ data. Det første er at prosessen er arbeids- og tidskrevende. Det andre er at fragmentering av data kan føre til at man går glipp av viktig innhold. Og det tredje problemet er at det kan være vanskelig å redusere kodene man lager til et håndterlig antall.

3.2 UTVALG AV DATAMATERIALE

I min innholdsanalyse av lærebøker samlet jeg inn datamaterialet fra to lærebokserier: Multi, 5.-6. trinn, publisert av Gyldendal, og Matemagisk, 5.-7. trinn, publisert av Cappelen Damm. Ettersom forskningsspørsmålene i studien tar for seg mellomtrinnet og programmeringsoppgaver, er populasjonen allerede innskrenket til matematikkbøker på mellomtrinnet. Det er kun etter fagfornyelsen at programmering har blitt en del av læreplanen, så det var naturlig å velge ut nye lærebøker som baserer seg på kompetansemål fra fagfornyelsen. For å kunne samle inn tilstrekkelig mengde data ønsket jeg å samle inn data fra mer enn ett læreverk, og helst fra læreverk fra alle tre trinnene på mellomtrinnet. Ved å bruke flere læreverk fra ulike trinn i datainnsamlingen kunne jeg innhente tilstrekkelig data for å best mulig svare på forskningsspørsmålene i studien.

3.2.1 Avgrensning

Populasjonen var allerede bestemt ut fra forskningsspørsmålene mine, men videre måtte jeg gjøre en ytterligere avgrensning. Læreverk kan bestå av lærerveiledning, grunnbøker, oppgavebøker og digitale ressurser. Jeg valgte å avgrense slik at jeg kun innhenter data fra grunnbøkene i læreverkene. Hovedårsaken til dette var at grunnboka var noe som var felles for alle læreverkene, og dessuten var det praktisk enklere å kun

måtte ordne tilgang til denne delen av læreverket. Et annet argument for å velge å avgrense til akkurat grunnboka er at læreboka elevene bruker er noe av det som har størst innflytelse for hva som undervises i matematikkfaget på skolen (Mullis, Martin & Foy, 2008; Valverde, Bianchi, Wolfe, Schmidt & Houang, 2002, referert i Lepik et al., 2015, s. 129). I tillegg ut fra egen erfaring fra praksis har jeg erfart at lærere ofte planlegger undervisningen ut ifra innholdet i grunnboka, og at lærerveiledningen blir brukt som en støtte i planleggingsarbeidet og i undervisningen. Jeg har valgt å ikke inkludere læreverkenes digitale ressurser fordi det ville gjort studien for omfattende.

Da jeg skulle velge læreverk tok jeg utgangspunkt i de største forlagene basert på markedsandel i 2021 (Medienorge, 2022) ettersom dette gir en indikasjon på hvilke læreverk som blir mest brukt i norsk skole. De tre største var Multi fra Gyldendal, Matemagisk fra Aschehoug og Matematikk fra Cappelen Damm. En felles komponent for de tre læreverkene er at alle har grunnbøker. Erfaring fra praksis er at undervisningen i stor grad bygger på innhold fra grunnboka, med støtte i de digitale ressursene som tilhører læreverket. På bakgrunn av dette har jeg valgt å ta utgangspunkt i grunnbøkene i datainnsamlingen. Jeg valgte å ikke inkludere digitale ressurser fordi, som nevnt tidligere, at arbeidet ville blitt for omfattende og tidkrevende for en studie av dette omfang. De digitale ressursene er i noen læreverk svært uoversiktlige blant annet fordi det kan være et randomisert utvalg av oppgaver som er tilgjengelig, og det er vanskelig å vite om man får hentet inn alle aktuelle analyseenheter. I tillegg kan de digitale oppgavene og læringsaktivitetene skiftes ut ofte. Et interessant funn som ble gjort umiddelbart var at *Matematikk* fra Cappelen Damm ikke hadde programmeringsoppgaver i sine grunnbøker. I tillegg hadde ikke *Multi 7* blitt gitt ut da jeg arbeidet med datainnsamlingen, og er derfor ikke en del av datagrunnlaget. Mitt datagrunnlag er derfor Matemagisk 5B (Raen et al., 2020), Matemagisk 6B (Kongsnes et al., 2021a), Matemagisk 7A (Kongsnes et al., 2021b), Multi 5B, (Alseth et al., 2021a) og Multi 6B (Alseth et al., 2021b), ettersom disse grunnbøkene inneholdt kode- og programmeringsoppgaver. Datamaterialet ble innhentet ved å ordne meg tilgang til de aktuelle grunnbøkene, enten ved å låne på biblioteket, få tilgang gjennom elektroniske lærebøker på nett, eller ved å kjøpe lærebøkene.

I noen av lærebøkene kunne utvalget avgrenses til kapittelet om programmering ettersom alle programmeringsoppgavene var samlet i det aktuelle kapittelet. I andre lærebøker finner man programmeringsoppgaver i ulike kapitler med innhold som samsvarer med kapittelets tema. Derfor gir det ikke mening å avgrense til spesifikke kapitler, men i stedet avgrense til sider hvor man finner programmeringsinnhold i de ulike grunnbøkene. I neste delkapittel forklares hvilke analyseenheter som blir valgt ut fra programmeringsinnholdet i lærebøkene som analyseres.

3.2.2 Analyseenheter

I en innholdsanalyse kan det være aktuelt å velge analyseenheter som oppgaver, eksempler, forklaringer og tekst i tilknytning til illustrasjoner. I min studie har jeg valgt å begrense analyseenheter til kun oppgaver fra kapitlene. Grunnen til dette var at oppgavene i stor grad gjenspeilte innholdet i eksemplene i lærebøkene, slik at det å inkludere eksemplene fra kapitlene ville i liten grad tilført datamaterialet noe nytt. Jeg

har heller ikke gjort en analyse av forklaringen og figurtekst fordi det ville gjort studien for omfattende ettersom jeg har valgt å samle inn data fra flere lærebøker.

Oppgavene jeg har analysert er kun nummererte oppgaver. Det vil si at oppgaver som opptrer i form av *samtaleruter*, *utforsk sammen*, *spill* eller liknende ikke er en del av analysen. Deloppgaver er ikke definert som en egen analyseenhet. Gjennom rammeverket til Bråting og Kilhamn (2021) gir det mest mening å se på én oppgave (eventuelt bestående av flere deloppgaver) som en analyseenhet, ettersom man blant annet ønsker å kartlegge hvilke handlinger oppgaven som en helhet krever av elevene.

3.3 Analysens struktur

Til sammen har 101 analyseenheter blitt analysert. Måten dette har blitt gjort på er gjennom ulike delanalyser for å gjøre analysen mer strukturert og oversiktlig, slik at det er enklere å arbeide seg gjennom datamateriale på en måte som gjør at alle analyseenhetene blir behandlet likt. Den første delanalysen handler om å identifisere hvilken eller hvilke handlinger oppgaven krever av elevene. Delanalyse 2 handler om å identifisere programmeringsbegrep og matematiske begrep i oppgavene som blir analysert. Dette blir gjort ved bruk av skjemaer jeg har utviklet for å ha god systematikk og oversikt over analyseenhetene, og oversikt over hvilke koder og kategorier de tilhører. Den tredje og siste delanalysen spisser seg mer inn på å utforske det andre forskningsspørsmålet som handler om på hvilke måter programmeringsoppgavene knytter matematikk og programmering sammen. Her har jeg også utviklet skjema for å enklere kategorisere analyseenhetene.

3.4 Analyseprosessen

Etter å ha oversatt kodene til Bråting og Kilhamn (2021) og utarbeidet nye koder i tråd med læreplanen, ble datamaterialet fra én av lærebøkene kodet. Jeg så på delene delanalyse 1 og 2 samtidig på hver analyseenhet. I løpet av denne prosessen gjorde jeg tilpasninger i kodene for å tilrettelegge best mulig for koding av det resterende datamaterialet. Forandring i kodene er normalt i en teoretisk tilnærming (Fauskanger & Mosvold, 2015). Videre i kodeprosessen oppsto det behov for nye koder for innhold fordi de ulike lærebøkene tar i bruk ulike begreper. Da jeg var helt ferdig med kodinga for delanalyse 1 og 2, jobbet jeg videre med koding for delanalyse 3. I denne delen ble analyseenhetene kategorisert etter forhåndsbestemte koder fra Bråting og Kilhamn (2021).

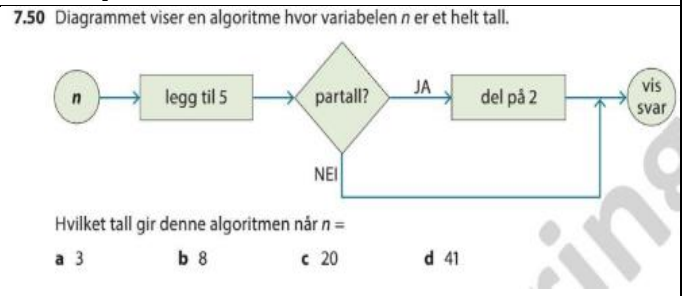
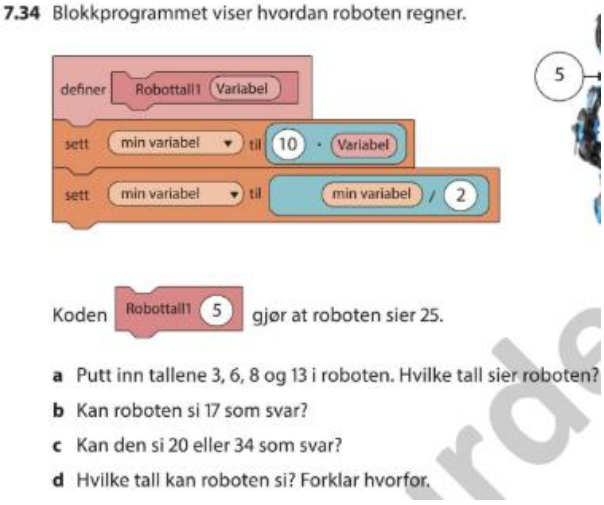
En utfordring i prosessen med koding av datamaterialet var å være konsekvent og presis slik at kodene ble brukt helt likt på alle analyseenheter. Det var noen avklaringer som ble gjort underveis i prosessen som resulterte i endringer i kodene. Dette medførte at jeg ble nødt til å gå gjennom hele datamaterialet flere ganger for å sikre at alle analyseenhetene var analysert på samme måte.

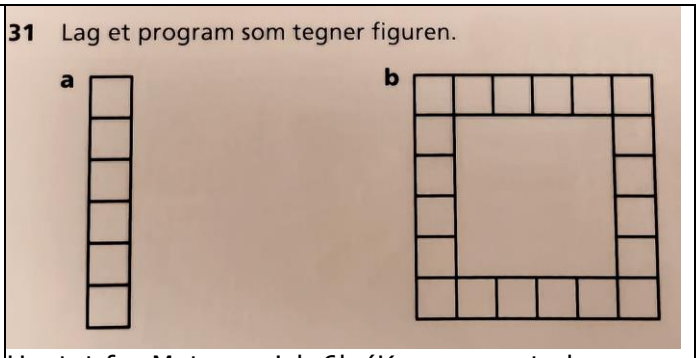
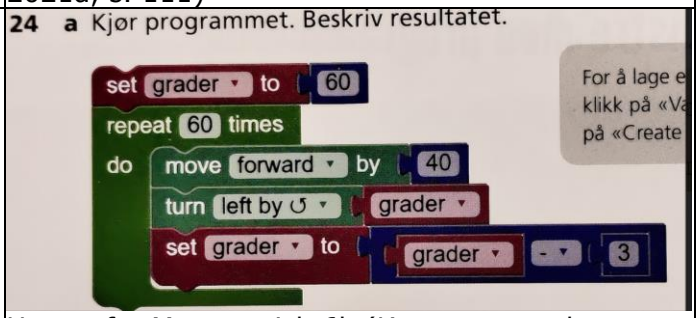
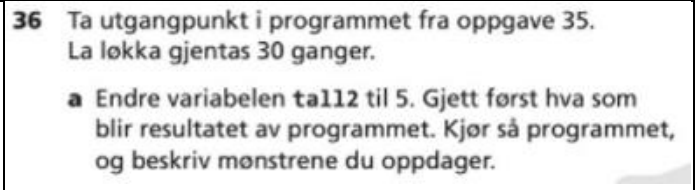
Etter at alle analyseenhetene var kodet, utførte jeg analysen. De tre delene av analysen ble utført, og aktuelle funn ble presentert.

Analyseenhetene i delanalyse 1 og 2 kan potensielt kodes med flere koder innenfor samme kategori. En oppgave kan eksempelvis både kreve at eleven skal *(a) følge en prosedyre* og *(e) forklare*. Ettersom det i mange tilfeller var flere koder innenfor samme kategori i delanalyse 1 og 2, brukte jeg en tabell i regneark som gjorde det enkelt å håndtere analyseenheter hvor dette var tilfellet.

3.4.1 Delanalyse 1 – Handlinger

I arbeidet med å undersøke (1) hva som karakteriserer programmeringsinnholdet i norske lærebøker i matematikk, og (2) på hvilke måter programmeringsoppgavene knytter programmering og matematikk sammen, undersøkte jeg først hvilke handlinger oppgavene krever at elevene skal gjøre for å løse oppgaven. Ulike handlinger kan representere ulike tilnærminger for å tilegne seg kunnskap. Noen av handlingene legger bedre til rette for utforsking enn andre, noe som kan ha betydning med tanke på elevenes mulighet til å konstruere ny kunnskap. I analysen av handlinger ble kategoriene og kodene utviklet av Bråting og Kilhamn (2021) i deres rammeverk og var forhåndsbestemt i rammeverket jeg bruker. Delanalyse 1 er derfor deduktiv. Kodene som blir brukt i analysen er beskrevet og eksemplifisert i tabell 3.2.

Handlinger	**Beskrivelse**	**Eksempel fra datamaterialet**
(a) Følge en prosedyre	Følg trinnvise instruksjoner, repeter eller fortsette et mønster.	 Hentet fra Multi 5B, (Alseth et al., 2021a)
(b) Finne ut	Finn ut prosedyren, regelen eller mønsteret som frembringer et utfall.	 Hentet fra Multi 6B (Alseth et al., 2021b)
(c) Feilsøke	Feilsøk en kode. Finne ut hva som er feil.	Finnes ikke eksempel i datamaterialet.
(d) Forme og skape	Gi instruksjoner, lag et mønster, skriv kode, representer	

	med symboler.	 Hentet fra Matemagisk 6b (Kongsnes et al., 2021a, s. 111)
(e) Forklare	Forklar ved bruk av naturlig språk, ved bruk av ord for å beskrive en prosedyre, en regel, et mønster eller et begrep	 Hentet fra Matemagisk 6b (Kongsnes et al., 2021a)
(d) Forestille seg	Forutse hva som vil skje, reflekter rundt mulige utfall når betingelser eller verdier blir forandret.	 Hentet fra Matemagisk 7a (Kongsnes et al., 2021b)

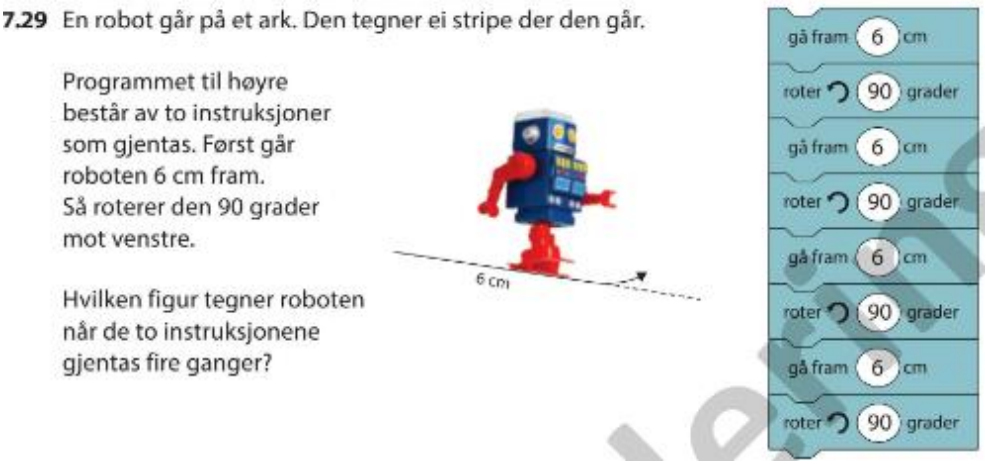
Tabell 3.1 Koder for handlinger. Eksempler hentet fra Alseth et al., 2021a, Alseth et al., 2021b, Kongsnes et al., 2021a og Kongsnes et al., 2021b

3.4.2 Delanalyse 2 – Matematiske begrep og programmeringsbegrep

For å finne ut hva som karakteriserer innholdet i programmeringsoppgaver i lærebøkene ser jeg videre på begrep knyttet til matematikk og begrep knyttet til programmering i analyseenhetene. For å klassifiseres som et matematisk begrep må det være et begrep innenfor tradisjonell skolematematikk og formidle en viktig matematisk ide (Bråting & Kilhamn, 2021). Det kan i mange tilfeller være vanskelig å vite hvilke matematiske ideer som oppgavene har til hensikt å formidle, og derfor kan det være enklere å støtte seg til mer veldefinerte begreper som vi finner i matematikken. Et eksempel på dette kan være oppgaven som er vist i figur 3.1. Denne oppgaven handler om å finne ut hvilken geometrisk figur roboten tegner ut fra gitte instruksjoner i et blokk-programmeringsverktøy. Denne beskrivelsen på hvordan programmet konstruerer figuren ved å fire ganger gå frem 6 cm, deretter rotere 90 grade, samt at oppgaven går ut på å finne ut at det tegnes den geometriske figuren kvadrat, gjør at det kan bli kategorisert innenfor geometriske begrep.

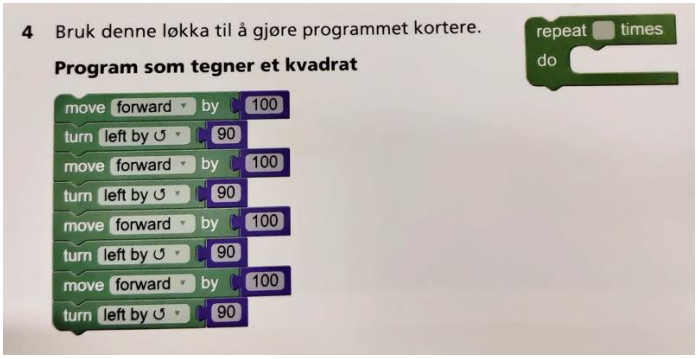
Når det kommer til programmeringsbegrep har jeg tatt utgangspunkt i det litteraturen sier at er grunnleggende begreper innenfor programmering, og sett på hvilke

programmeringsbegrep som blir bruk i læreplanen. Jeg har også sett på programmeringsbegrepene som blir brukt i Bråting og Kilhamn (2021) sin studie.



Figur 3.1 Programmeringsoppgave med geometriske begrep (Alseth et al., 2021b, s. 87)

Når de matematematiske begrepene og programmeringsbegrepene er kodet på hver enkelt analyseenhet, kan man få en oversikt over det matematiske innholdet og programmeringsinnholdet i de aktuelle oppgavene. På denne måten kan man finne ut hvilke begrep, både innenfor matematikk og programmering, som forekommer flest og færrest ganger. Under viser jeg et eksempel på hvordan matematiske begrep og programmeringsbegrep blir identifisert i en oppgave fra datamaterialet mitt.



Figur 3.2 Oppgave fra Matemagisk 5 (Raen et al., 2020, s. 102)

I oppgaveteksten står det «Bruk denne løkka til å gjøre programmet kortere». Her er *løkke* og *program* to programmeringsbegrep som eksplisitt nevnes i oppgaven. I tillegg er det også en grafisk fremstilling av blokk-programmet. Det matematiske begrepet som eksplisitt kommer til syne i oppgaven er *kvadrat,* som vi finner innenfor «geometriske begrep».

3.4.3 Delanalyse 3 – Knytte sammen matematikk og programmering

I den siste delen av analysen ser jeg på kombinasjonen av handlinger og begreper for å se på hvilke måter koblinger mellom programmering og matematikk er gjort, eller potensielt kunne blitt gjort. I analysen til Bråting og Kilhamn (2021) blir det diskutert

hvordan eksplisitte koblinger mellom matematikk og programmering kan føre til læring av matematikk. Det er aktuelt å se på om elevene er bedt om å omformulere AT-ideer ved å bruke matematisk terminologi eller tradisjonell matematisk notasjon, og om oppgaven lar dem utforske matematiske ideer som er nye for dem (Bråting & Kilhamn, 2021, s. 7). I denne delen av analysen ønsker man å diskutere og finne ut av om de ulike programmeringsoppgavene i lærebøkene legger til rette for å konstruere matematisk kunnskap. Bråting og Kilhamn (2021) valgte i sin studie å ekskludere eksempler og forklaringer fra lærebøkene, slik at utvalget kun består av programmeringsoppgaver tatt fra lærebøkene de analyserte. Begrunnelsen for dette var at introduksjon og beskrivelse av nye begrep i eksemplene i lærebøkene var veldig korte, og at de påfølgende oppgavene reflekterte eksemplene i stor grad. Det ble derfor vurdert til at oppgavene var mer passende som analyseenheter. Jeg gjør samme vurdering i mitt tilfelle, og velger i likhet med Bråting og Kilhamn (2021) kun oppgaver fra lærebøkene som analyseenheter. Hver analyseenhet ble karakterisert ut fra hvilke handlinger som inngår i oppgaven, for eksempel om elevene er bedt om å *(a) følge en prosedyre*, om de skal *(d) forme og skape* eller *(e) forklare*. Videre blir alle programmeringsbegrepene og de matematiske begrepene som eksplisitt blir nevnt i oppgaven identifisert. I noen tilfeller finnes det instruksjoner eller et løst eksempel etterfulgt av oppgaver uten at instruksjonene gjentas. I disse tilfellene ble instruksjonene og generell informasjon fra eksempelet inkludert som en del av beskrivelsen til oppgavene. I denne delen av analysen ser jeg i hovedsak på om oppgavene krever omformulering av AT-ideer til matematisk terminologi eller tradisjonell matematisk notasjon, og om oppgavene gir elevene mulighet til å utforske matematiske ideer som er nye for dem (Bråting & Kilhamn, 2021). Under er et utklipp fra tabellen jeg har utformet og tatt i bruk i delanalyse 3.

Oppgave	**1. Omformulering av AT-ideer til matematisk terminologi eller tradisjonell matematisk notasjon**		**2. Utforske matematiske ideer som er nye for dem?**		**Forklaring**
	JA	NEI	JA	NEI	
1, s. 99	X			X	Omforme fra blokk-programmering til geometri (kvadrat). Kvadrat skal være kjent fra før av, så oppdager ikke "nye" matematiske ideer.

Tabell 3.2 Eksempel på skjema brukt i analysen

Omformulering av ideer innenfor algoritmisk tenkning til matematisk terminologi eller tradisjonell matematisk notasjon innebærer et skifte mellom programmering og matematikk. Et eksempel på dette er å skape en kode som tegner en geometrisk figur, eller at man tar utgangspunkt i en geometrisk figur for å lage en kode.

Nå det kommer til det andre punktet; «utforske matematiske ideer som er nye for dem», tar jeg utgangspunkt i kompetansemål fra læreplanen på det gjeldende trinnet. Dersom det er samsvar mellom innhold i læreplanmål og det eleven får mulighet til å utforske i den aktuelle oppgaven, kan det krysses av i kolonnen for «ja». Et eksempel på dette er en oppgave i en lærebok på 6. trinn som legger til rette for at eleven får utforske matematiske begrep som formidles i kompetansemål fra 6.trinn. I tillegg til at innholdet i

oppgaven og læreplanen må samsvare, spiller det en viktig rolle hvilke handlinger oppgaven krever. For eksempel at oppgaven krever at eleven skal forme og skape i en kombinasjon med å forklare, vil kunne skape bedre mulighet for å utforske enn ved en oppgave som krever at eleven kun skal følge en prosedyre.

3.5 TROVERDIGHET

Målet med troverdighet i en kvalitativ undersøkelse er å støtte argumentet om at undersøkelsens funn er «verdt å ta hensyn til» (Lincoln & Guba, 1985). Guba (1981) utviklet et rammeverk for å sikre forskningens troverdighet, og tar for seg fire aspekter ved troverdighet som en forsker må forholde seg til.

Kredibilitet er det første aspektet, som omhandler en intern validitet (Guba, 1981). God praksis skal være fulgt og man forholder seg til rammeverket som er satt. For å sikre god kredibilitet i den type studie jeg har gjort, må man sørge for at datainnsamlingen gir data som er relevante for problemstilling. Forskningsspørsmålene mine har omhandlet programmeringsoppgavers karakteristikk og i lærebøker for mellomtrinnet, og på hvilke måter programmering og matematikk knyttes sammen i oppgavene. Datagrunnlaget mitt har derfor naturligvis vært hentet fra lærebøker i matematikk på mellomtrinnet. Jeg har også forsøkt å gjøre forskningen min så gjennomsiktig som mulig ved å beskrive forskningsprosessen så nøyaktig som mulig og lagret alt av skjemaene jeg har brukt i datainnsamlingen og analysen. For å tilstrebe at alle analyseenheter har blitt behandlet likt, har jeg gått gjennom datamaterialet på nytt når det har dukket opp nye ting.

Det neste aspektet *overførbarhet* handler om en ekstern validitet (Guba, 1981). Det vil si at det skal være rike beskrivelser av konteksten funnene kommer fra slik at det eventuelt kan være overførbart til andre sammenhenger (Stahl & King, 2020). I studien min er funnene mine godt beskrevet og illustrert med eksempler fra datamaterialet, slik at det kan være overførbart til andre sammenhenger dersom det ses på som hensiktsmessig.

Aspektet *pålitelighet* omhandler at studien skal kunne bli ettergått (Guba, 1981). Derfor har jeg jeg tatt vare på alt av dokumenter, tabeller, notater og analyser gjort i løpet av forskningsperioden. På denne måten kan noen andre få full tilgang til hvordan jeg har kodet, kategorisert og analysert datamaterialet. I tillegg har jeg forsøkt å forklare nøye hvordan jeg har gjennomført kodinga både med beskrivelser og eksempler. Beskrivelse burde være tilstrekkelig for at andre kan gjøre det på samme måte som meg.

Det siste aspektet i rammeverket til Guba (1981) er *bekreftbarhet.* Bekreftbarhet innebærer at forskeren skal være objektiv og vise at man har handlet i god tro, og ikke la personlige verdier eller annet påvirke arbeidet (Guba, 1981). Jeg mener selv at jeg har jeg ikke har hatt noen sterke forutinntatte holdninger eller formeninger om oppgaver fra lærebøker, eller lærebøker i seg selv. Personlig ser jeg på lærebøker som gode ressurser å støtte seg på som lærer, og at det er viktig å ha gode lærebøker som har et innhold i samsvar med læreplanen. Når jeg i denne studien har samlet inn data, analysert og diskutert den har det vært gjennom et rammeverk som er designet ut fra relevant litteratur og læreplanen, for å finne ut hva som karakteriserer programmeringsinnhold, og hvordan matematikk og programmering knyttes sammen. Jeg har ingen personlig interesse av å avdekke hvilke oppgaver som er gode eller dårlig i henhold til

rammeverket jeg har brukt, men det er et nyttig verktøy for meg videre inn i læreryrket med tanke på å gjøre vurderinger om hva som eventuelt burde tilføres i programmeringsoppgaver for å berike dem.

3.5.1 Etiske betraktninger

Ettersom jeg har gjort en innholdsanalyse av lærebøker er det naturligvis færre etiske begrensninger enn i studier som for eksempel involverer intervjuobjekter, hvor man må ta høyde for personvernet til de deltakende. Det er ingen personopplysninger man må forholde seg til, og det er ikke noe krav om å melde inn prosjektet til NSD. Ettersom jeg gjør en analyse av oppgaver fra lærebøker, er lærebokforfatterne en tredjepart jeg må ta hensyn til. «Forskeren har uansett ansvar for å sikre personvernet til dem som direkte eller indirekte er berørt av forskningsprosjektet» (NESH, 2016, s. 19). I min studie gjør jeg en analyse av et utvalg av oppgaver fra flere lærebøker for å finne ut hva som karakteriserer disse oppgavene, og mer spesifikt, på hvilke måter programmering og matematikk knyttes sammen i disse oppgavene. Intensjonen min har aldri vært å vurdere kvaliteten på de spesifikke læreverkene. Det er viktig å forholde seg objektiv i analysen, og være nøye på hvordan setningene formuleres når lærebøkene omtales. En annen ting som er viktig å huske på er at lærebokforfattere ikke uten videre kan kategoriseres som det NESH (2016) omtaler som sårbare tredjeparter. Det er ifølge NESH (2016) viktig at hensynet til forskningens kritiske funksjon og sannhetssøken veies opp mot hensynet til belastningen for tredjepart. I min studie overskrider jeg ingen grenser med tanke på hensynet for tredjepart. I tillegg til hensyn til tredjepart er også «vitenskapelig redelighet» og «etterprøving og deling av data» retningslinjer, fra NESH (2016), en forsker skal forholde seg til. I min studie har jeg vært nøye med å oppgi referanser til teori og data som har blitt brukt, slik at alle henvisninger skal kunne spores tilbake til kilden. Dessuten har jeg tatt vare på alt av datainnsamling og hvordan datamaterialet har blitt kodet og analysert, slik at studien skal være etterprøvbar.

4 ANALYSE

Forskningsspørsmålene i denne studien er: (1) Hva karakteriserer programmeringsinnholdet i norske matematikkbøker på mellomtrinnet? (2) På hvilke måter knytter programmeringsoppgavene programmering og matematikk sammen? For å svare på dette har jeg tatt utgangspunkt i Bråting og Kilhamn (2021) sitt analytiske verktøy for å analysere programmeringsoppgaver, inspirert av rammeverkene til Brennan and Resnick (2012) om algoritmisk tenkning og Benton et al.'s (2016) rammeverk for handling. Jeg har forsøkt å anvende metoden på så like måte som Bråting og Kilhamn (2021) gjorde, men jeg har gjort de tilpasningene som var nødvendige eller hensiktsmessige i konteksten for min studie. Dette er beskrevet i metoden. Analysekapittelet er strukturert slik de tre delanalysene blir presentert hver for seg, med respektive funn og eksempler fra datamaterialet. Til slutt i kapittelet er det et sammendrag av funnene fra de tre delanalyse.

4.1 ANALYSE

Analysen består av tre delanalyser for å undersøke hva som karakteriserer programmeringsinnholdet, og på hvilke måter programmeringsoppgavene knytter programmering og matematikk sammen i matematikkbøker på mellomtrinnet. Eksempler og funn fra hver av delanalysene presenteres i de kommende delkapitlene. Totalt sett identifiserte jeg programmeringsinnhold i 101 oppgaver, med analyseenheter fra to lærebøker for 5. trinn, to på 6. trinn og én på 7. trinn.

4.1.1 Delanalyse 1 - Handlinger

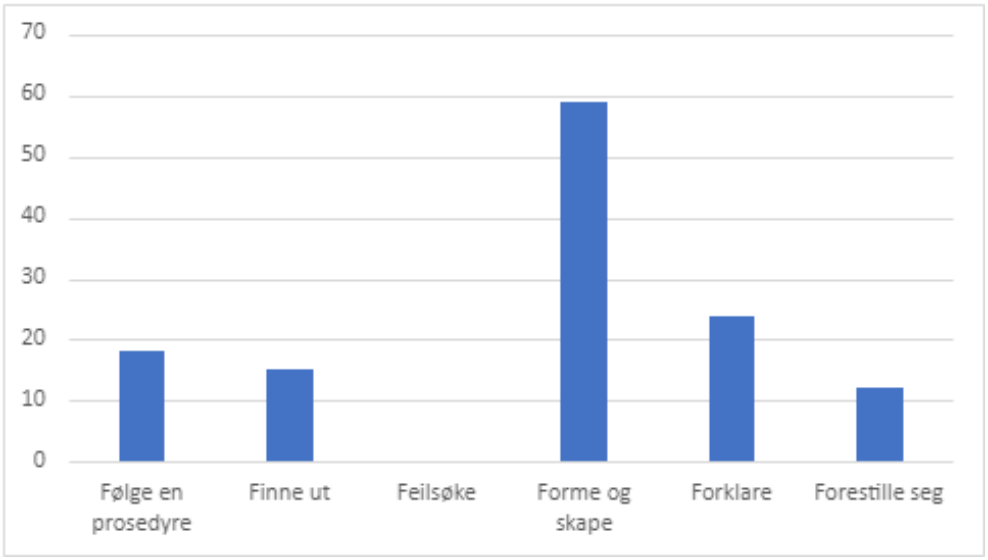
I delanalyse 1 har jeg undersøkt hvilke handlinger oppgavene i lærebøkene krever at elevene gjør i oppgaveløsningen. Oppgavene ble kodet med fem ulike handlinger; *(a) følge en prosedyre, (b) finne ut, (c) feilsøke, (d) forme og skape, (e) forklare* og *(f) forestille seg* (se tabell 4.1). Merk at en enkelt oppgave kan bli kategorisert med mer enn en handling. I denne delen av analysen er det aktuelt å kartlegge hvilke handlinger som kreves flest til færrest ganger og finne ut om oppgavene krever mer enn én handling. Funnene jeg velger å trekke frem i denne delanalysen er:

Funn 1: Fordeling av handlingene
Funn 2: Oppgaver med mer enn én handling

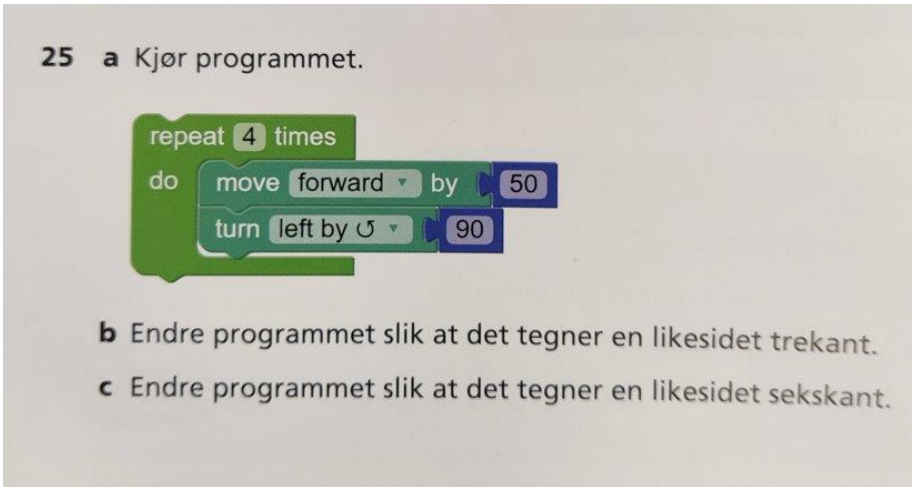
Funn 1: Fordeling av handlingene

Figur 4.1 viser fordelingen av handlinger oppgaven krever av eleven. Ut fra diagrammet ser man at den vanligste handlingen er «forme og skape», som karakteriserer aktiviteten i mer enn halvparten av oppgavene. Et typisk eksempel på «forme og skape» vises i figur 4.2 hvor eleven blir bedt om å endre programmet slik at det gjør noe annet. I dette tilfellet skal eleven gjøre forandringer i koden slik at (i deloppgave *b) programmet tegner en likesidet trekant* og (deloppgave *c) programmet tegner en likesidet sekskant*. Illustrasjoner av blokk-kode er ofte en del av oppgavene i lærebøkene. Oppgavene legger i mange tilfeller opp til at elevene skal gjøre forandringer og legge til kode i allerede påbegynte kode-sekvenser. I tillegg finnes det mange eksempler på oppgaver i lærebøkene som innebærer at elevene skal skape programmer ved bruk av blokk-

programmering, uten hjelp fra illustrasjoner av blokk-kode. Et eksempel på dette er oppgaver som sier «Lag et program som tegner denne figuren", hvor den aktuelle figuren er tegnet i tilknytning til oppgaven.



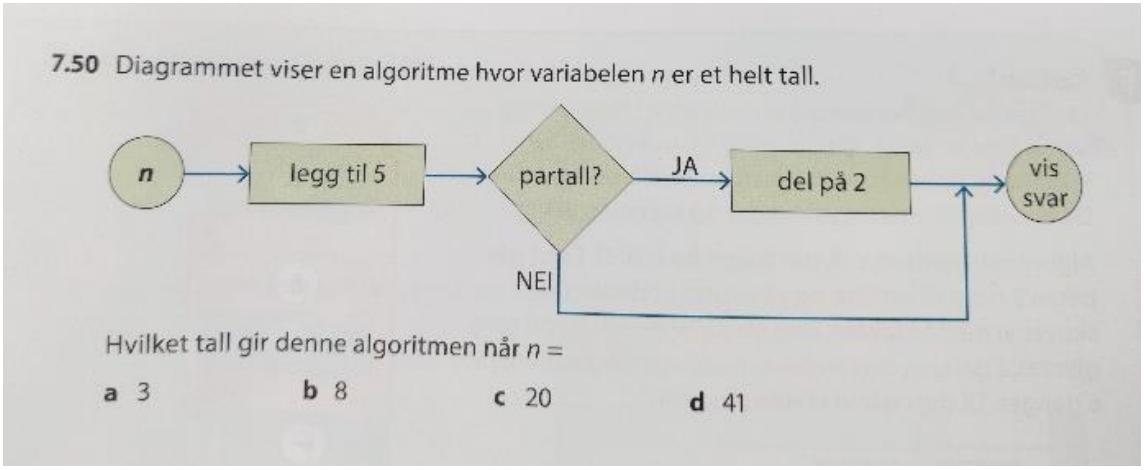
Figur 4.1 Diagram som viser fordeling av handlingene



Figur 4.2 Eksempel på «Forme og skape» i en oppgave på 6. trinn (Kongsnes et al., 2021a, s. 106)

Handlingen som forekommer nest flest ganger som identifiseres i tjuefire av oppgavene er «Forklare». Som regel innebærer oppgavene innenfor denne kategorien å forklare hva blokk-koden som er illustrert i oppgaven gjør.

Den tredje og fjerde vanligste handlingen er «Følge en prosedyre» og «Finne ut», som karakteriserer aktiviteten i henholdsvis sytten og femten av oppgavene. Oppgavene identifisert som «Følge en prosedyre» innebærer i de fleste tilfeller å bruke en algoritme bestående av stegvise instruksjoner. Figur 4.3 viser et typisk eksempel hvor eleven skal bruke algoritmen for teste hvilket svar algoritmen gir dersom man starter med en verdi n.

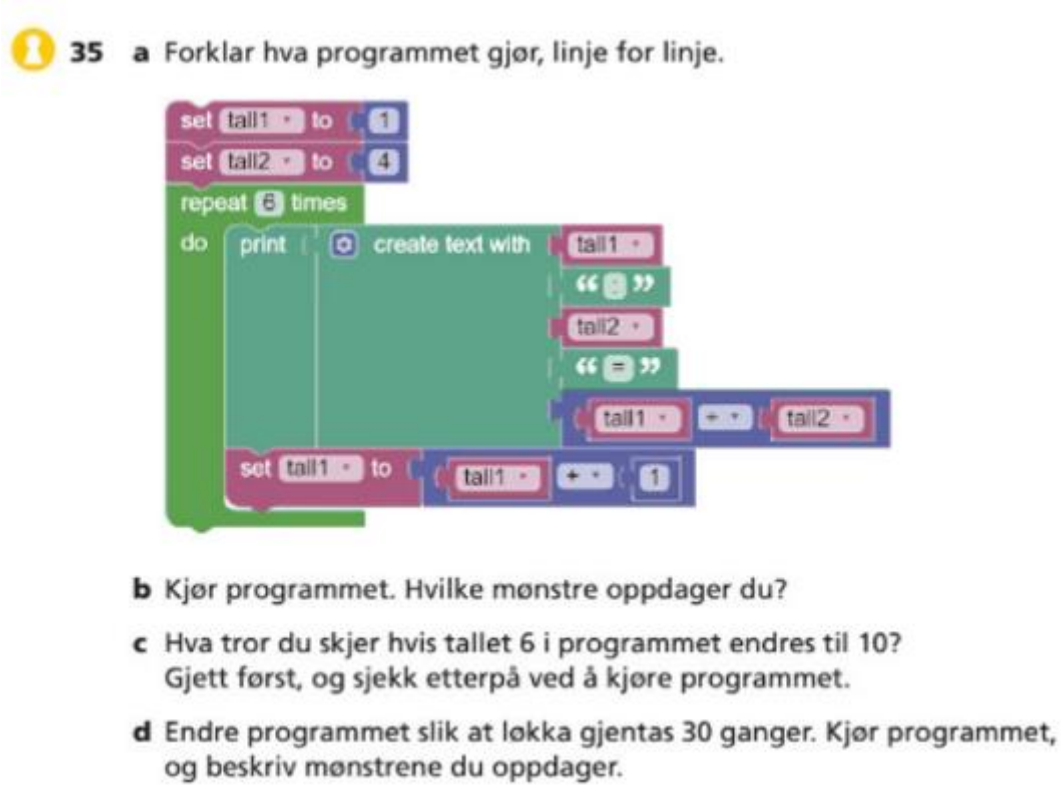


Figur 4.3 Algoritme i form av et flytdiagram (Alseth et al., 2020a)

Oppgavene identifisert som «finne ut» innebærer i de fleste tilfeller å finne ut hvilke verdier en algoritme kan gi som svar, eventuelt å finne ut hvilke verdier som kan bli brukt i algoritmen for at den skal fungere slik den er ment. Den handlingen som forekommer i færrest tilfeller, sett bort i fra «feilsøke» med null tilfeller, er «forestille seg» som kreves i tolv av til sammen 101 oppgaver. Denne handlingen viser seg som regel i deloppgaver hvor eleven blir bedt om å se for seg hva en blokk-kode gjør, før hen tester det ut i et programmeringsverktøy.

Funn 2: Oppgaver med mer enn én handling

I analysen oppdaget jeg at 35 av til sammen 101 programmeringsoppgaver krever mer enn én handling av eleven. I de fleste av tilfellene var det en kombinasjon av «Forme og skape» og enten «Forklare» eller «Forestille seg». I en stor andel av oppgavene som krever at elevene skal «følge en prosedyre» var denne handlingen kombinert med en annen. Merk at «følge en prosedyre» og «forme og skape» i kombinasjon med hverandre kun er identifisert i én av analyseenhetene. Figur 4.4 viser et eksempel på en oppgave med kombinasjonen «forme og skape», «forklare og «forestille seg».



Figur 4.4 Oppgave med mer enn én handling (Kongsnes et al., 2021b, s. 108)

Figuren over viser en oppgave fra datamaterialet. I første deloppgave står det: «Forklar hva programmet gjør, linje for linje», og oppgaven krever dermed handlingen «forklare». Den samme handlingen gjelder i b-oppgaven, men i denne situasjonen skal eleven beskrive mønsteret som programmet lager. I deloppgave c, spør oppgaven etter hva eleven tror skjer dersom en gitt endring i koden blir gjort. Eleven blir bedt om å gjette først, før hen sjekker ved å kjøre programmet. Denne deloppgaven forventer først og fremst at eleven skal «forestille seg» hva som skjer dersom det blir en endring i koden. I både deloppgave c og d forventes det at eleven gjør en forandring i koden. Dette er ikke en stor forandring, men går likevel under kategorien «forme og skape». Oppgaven som helhet krever tre ulike handlinger av eleven ved at hen skal forklare og beskrive hva programmet gjør og resultatet, forestille seg hva som skjer dersom en endring gjøres, og aktivt gå inn i koden og forme den.

4.1.2 Delanalyse 2 - Begrep

Delanalyse 2 inneholder både en analyse av programmeringsinnholdet og det matematiske innholdet i analyseenhetene. Fordelingen av de identifiserte programmeringsbegrepene og de matematiske begrepene blir i dette delkapittelet fremstilt i tabeller for å gi en tydelig oversikt. Måten jeg har gått frem for å identifisere begrepene illustreres gjennom eksempler fra datamaterialet.
Funnene jeg har valgt å trekke frem i delanalyse 2 er:

1. Identifiserte programmeringsbegrep
2. Identifiserte matematiske begrep

Funn 1: Identifiserte programmeringsbegrep
Jeg begynner med å se på programmeringsbegrepene identifisert i oppgavene. Som tabell 4.1 viser, er *program* det begrepet som forekommer flest ganger. Begrepet brukes i de fleste oppgaver hvor blokk-programmering er en del av oppgaven. Eksempler på dette er oppgaver hvor elevene blir bedt om å kjøre, utforske eller lage et program. I Multi 5 brukes *algoritme* om det som ville blitt omtalt som *program* i Matemagisk-serien. Et eksempel på algoritme er vist i figur 4.3 hvor eleven er bedt om å bruke algoritmen på ulike tallverdier for n. *Algoritme* og *program* er ved kun to anledninger nevnt i samme oppgave, og et av de to begrepene finner man i de fleste av oppgavene i datamaterialet.

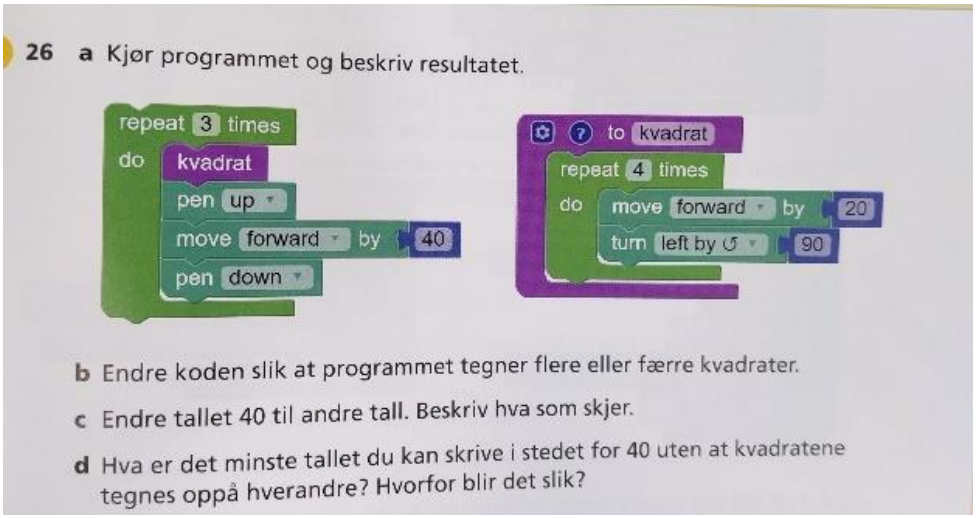
Programmeringsbegrep	Oppgaver
Algoritme	31 (31%)
Løkker, repetisjon	49 (49%)
Program	63 (62%)
Vilkår	19 (19%)
Variabler	37 (37%)
Funksjoner	16 (16%)

Tabell 4.1 Fordeling av programmeringsbegrep

Tabell 4.1 viser at *løkker* forekommer i nesten halvparten av oppgavene. Løkker er nevnt i læreplanmål på både fjerde-, femte- og sjettetrinn og er et viktig redskap i programmering for å gjøre koden mer effektiv og enklere å lese. I mange av oppgavene er løkker en av flere programmeringsbegrep som brukes i en og samme oppgave. Jeg har også identifisert oppgaver hvor forfatterne i større grad har valgt å isolere begrepene slik at det for eksempel er algoritmeoppgaver med løkker i fokus, og algoritmeoppgaver med vilkår i fokus.

Variabler og *vilkår* er en del av læreplanen fra femtetrinn og vises i henholdsvis 37 og 19 av oppgavene som ble analysert. I oppgaver med variabler får elevene ofte mulighet til å teste hva som skjer dersom variablene endres. Når det kommer til *vilkår*, viser figur 4.3 et typisk eksempel på en oppgave fra lærebøkene hvor vilkår blir brukt. Eksempelet viser en algoritme med et vilkår som sier at hvis tallet er et partall så skal det deles på to, hvis ikke skal den gå rett til å vise svar. Det finnes også oppgaver i lærebøkene som bruker vilkår i blokk-koding.

Programmeringsbegrepet som forekommer færrest ganger er *funksjoner.* Det er til sammen seksten oppgaver om funksjoner i hele datamaterialet. Funksjoner (som programmerings-begrep) er en del av læreplanen fra sjettetrinn, hvor elevene skal bruke programmering til å utforske geometriske figurer og mønstre. Oppgavene om funksjoner handler som regel om å enten forandre på en funksjon slik at den tegner annen figur, eller bruke funksjonen for å tegne repeterende mønstre. Figur 4.6 viser et eksempel på en oppgave fra Matemagisk hvor funksjoner blir brukt for å tegne kvadrater i et program.



Figur 4.5 Oppgave om funksjoner (Kongsnes et al., 2021a, s. 107)

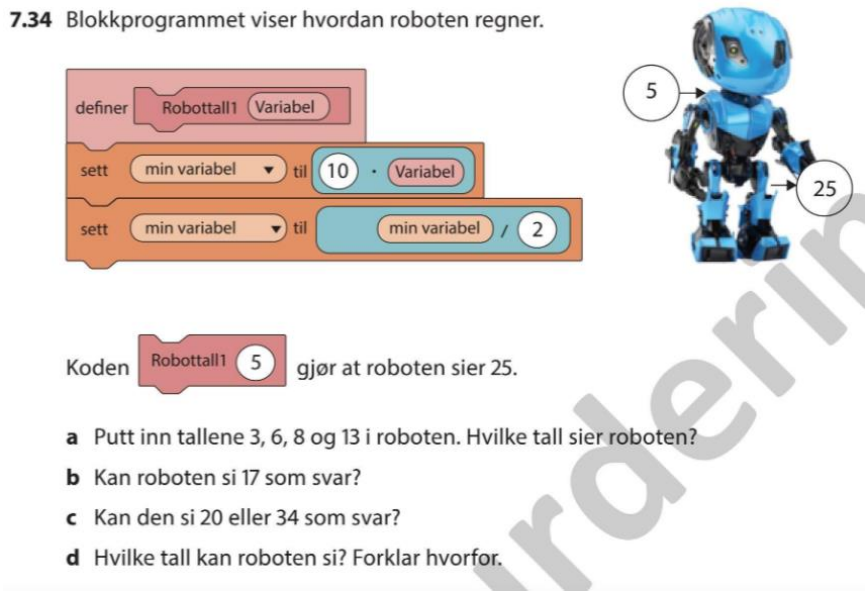
Funn 2: Identifiserte matematiske begrep

Jeg går videre til de matematiske begrepene i lærebøkene. Totalt sett ble det identifisert matematisk begrep i 61 av 101 programmeringsoppgaver, altså i 60 prosent av oppgavene. Tabell 4.2 viser antall oppgaver og prosentandel på de til sammen 61 programmeringsoppgavene hvor matematiske begrep er identifisert.

Matematiske begrep	Oppgaver
Mønstre (inkludert tallsekvenser)	26 (43%)
Geometriske begreper	23 (38%)
Aritmetiske begreper	35 (57%)
Symmetri	1 (2%)
Koordinatsystem	0 (0%)

Tabell 4.2 Fordeling av matematiske begrep på programmeringsoppgaver med identifisert matematiske begrep

Tabell 4.2 viser at aritmetiske begreper inngår i over halvparten av oppgavene med identifisert matematiske begrep. Ofte er det algoritmer eller programmer som bruker aritmetiske begreper i utregninger eller til å frembringe tallsekvenser. Figur 4.6 er hentet fra datamaterialet og viser et typisk eksempel på en oppgave kodet som «aritmetiske begreper». Multiplikasjon og divisjon inngår i blokkprogrammet for å gi en svarverdi basert på hvilket tall som blir puttet inn i algoritmen.



Figur 4.6 Programmeringsoppgave med «aritmetiske begrep» fra Multi 6b (Alseth et al., 2021b, s.91)

Ut fra tabell 4.2 ser man at symmetri kun er nevnt eksplisitt én gang i det innsamlede datamaterialet. I tillegg har valgt å inkludere *koordinatsystem* i tabell 4.2 selv om det ikke har blitt identifisert i noen av analyseenhetene. Årsaken til at jeg likevel tar det med i tabellene vil jeg gå nærmere inn på i delanalyse 3 når jeg tar for meg programmeringsoppgaver uten matematematisk innhold.

4.1.3 Delanalyse 3 - Knytte sammen matematikk og programmering

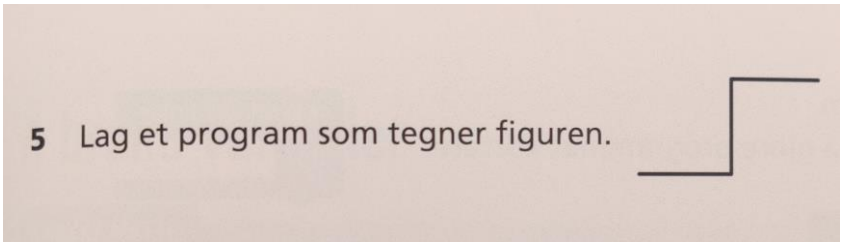
I delanalyse 3 ser jeg på kombinasjonen av handlinger og begreper for å undersøke hvordan programmeringsoppgavene i lærebøkene knytter matematikk og programmering sammen. Spesielt ser jeg på om oppgavene krever at elevene omformulerer ideer fra algoritmisk tenkning til matematisk terminologi eller tradisjonell matematisk notasjon, og om elevene får mulighet til å utforske matematiske ideer som er nye for dem. De funnene jeg har gjort som jeg ønsker å trekke frem i delanalyse 3 er:

Funn 1: Programmeringsoppgaver uten matematisk innhold
Funn 2: Matematikk som kontekst for programmeringsoppgaver
Funn 3: Utforske matematiske ideer med programmering

Funn 1: Programmeringsoppgaver uten matematisk innhold

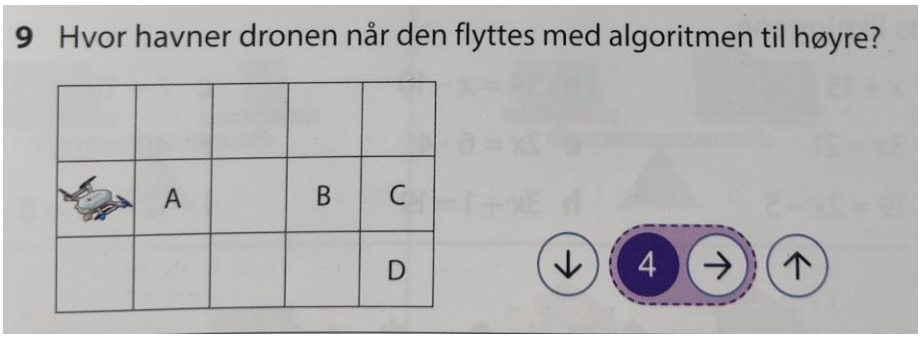
I delanalyse 2 kommer det frem at det ikke identifiseres matematiske begrep i 40 av analyseenhetene. Dette tilsvarer nesten halvparten av programmeringsoppgavene fra lærebøkene jeg har analysert. I dette delkapittelet ønsker jeg å vise til eksempler på programmeringsoppgaver uten identifiserte matematiske begrep, og se på hvilke mulige koblinger mellom matematikk og programmering som kunne blitt gjort, og hvordan dette kan føre til læring av matematikk. Programmeringsoppgavene uten matematisk innhold er i hovedsak to type oppgaver: Oppgaver som krever at man skal lage et program som tegner en figur (se figur 4.7), og oppgaver som innebærer navigasjon i rutenett, slik det er vist i figur 4.8).

Oppgaven i figur 4.7 er et av mange eksempler fra datamaterialet hvor oppgaven innebærer å lage et program som tegner en figur eller bokstaver. Denne typen oppgaver har jeg karakterisert som oppgaver uten matematisk innhold ettersom matematiske begrep ikke nevens eksplisitt. Det er slike oppgaver som forekommer i flest tilfeller av oppgaver uten matematisk innhold samlet sett i hele datamaterialet. En oppgave som dette vil kunne legge til rette for at eleven skal kunne forbedre programmeringsferdighetene sine, men man kan stille spørsmål om hvor stort det potensielle utbyttet er for å utvikle kunnskap og ferdigheter innenfor matematikk. Det vil kunne være et behov for å ha kunnskap om vinkler for å lage program seg tegner de ulike figurene, men det er likevel ingen tydelig kobling mellom matematikk og programmering.



Figur 4.7 Programmeringsoppgave – Lag et program som tegner en figur. Fra Matemagisk 5 (Raen et al., 2020, s. 103)

Det neste jeg vil trekke frem er oppgaver med bruk av rutenett og hvorfor disse kategoriseres som oppgaver uten matematisk innhold. Bruk av rutenett identifiseres i åtte av elleve oppgaver uten matematisk innhold i Multi 5. Det kan argumenteres for at rutenett er et matematisk begrep som kan brukes for å lære om koordinatsystemer. Likevel er rutenett ikke et begrep som nevnes i læreplanen, og det er noen klare forskjeller mellom de begrepene som må utdypes. Det første er at i et koordinatsystem er fokuset rettet mot punktene, mens i rutenett vil det være rettet mot selve rutene. Det andre er at det vil være nærliggende å bruke koordinater for å beskrive punktenes posisjon i et koordinatsystem. Figur 4.8 viser et eksempel på navigasjon i rutenett fra Multi 5b.



Figur 4.8 Navigasjon av drone i rutenett (Alseth et al., 2021a)

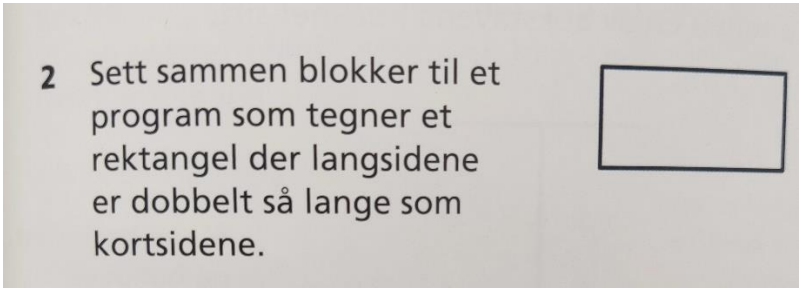
I oppgavene med rutenett fra datamateriale er fokuset i alle tilfeller rettet mot selve rutene. Som regel innebærer oppgavene forflytning rundt i rutenettet etter en gitt algoritme, eller ved å lage en algoritme. I en av oppgavene (se figur 4.9) er kolonnene navngitt med bokstaver og radene med tall (samme prinsipp som et sjakkbrett). Dette er

til hjelp med å navigere seg rundt i rutenett, og kan i dette tilfellet fungere litt på samme måte som koordinater i et koordinatsystem. Likevel har forfatterne også i dette tilfellet gjort at fokuset er rettet mot rutene fremfor punktene, og oppgaven kan derfor ikke direkte knyttes til koordinatsystem.

Figur 4.9 Rutenett-oppgave fra datamaterialet (Alseth et al., 2021a)

Funn 2: Kjent matematikk som kontekst for programmeringsoppgaver
Det andre funnet i delanalyse 3 som jeg ønsker å trekke frem er at matematikk som allerede er kjent for elevene, ofte blir brukt for å konstruere en kontekst for programmering. Det jeg legger i dette er at matematiske ideer som ifølge læreplanen skal være kjent gjennom mål fra tidligere trinn blir brukt som et utgangspunkt for programmeringsoppgavene. Med andre ord blir det i mange av oppgavene tatt matematiske ideer som er kjente for elevene som et utgangspunkt for å skape en kontekst for programmering. Dette fremfor å bruke programmering for å lære og utforske matematisk ideer som ifølge læreplanen skal være nye for dem. Jeg skal i de neste avsnittene vise til eksempler fra datamaterialet som støtter funnet, og se på hvilke måter koblinger mellom matematikk og programmering er gjort eller potensielt kunne blitt gjort, og hvordan dette kan føre til læring av matematikk.

Det første eksempelet er hentet fra Matemagisk 5 (se figur 4.10). Figuren viser en oppgave hvor man skal sette sammen blokker til et program som tegner et rektangel. Det kan være fristende å konkludere med at matematikk og programmering knyttes sterkt sammen i denne oppgaven ettersom et geometrisk begrep som «rektangel» blir brukt, og eleven må gi en presis instruksjon på hvordan denne geometriske figuren konstrueres i blokk-programmet. I tillegg krever oppgaven handlingen «Forme og skape» som ofte legger til rette for utforskende læring. Likevel tar oppgaven utgangspunkt i en todimensjonal figur som eleven ifølge læreplanen skal ha kunnskaper om fra 4. trinn. I dette tilfellet kan det se ut til rektangelet kun blir brukt for å skape en kontekst for programmering. Eleven får ingen videre mulighet til å utforske matematiske ideer som er nye for hen. Det kan derfor argumenteres for at koblingen mellom matematikk og programmering i dette tilfellet er svak.



Figur 4.10 Programmeringsoppgave – Lag et program som tegner et rektangel (Raen et al., 2020, s. 99)

Dersom oppgaven hadde vært i en lærebok på 4. trinn, kunne den enkelt blitt utvidet for å styrke koblingen mellom matematikk og programmering. Ved at eleven i tillegg blir bedt om å lage et program for en annen todimensjonal figur, og beskrive de to figurene, kunne oppgaven lagt til rette for at eleven jobber direkte med læreplanmål på 4. trinn. På denne måten kunne programmering blitt brukt for å utforske, beskrive og sammenligne egenskaper ved todimensjonale figurer slik læreplanmål på 4. trinn formidler (Utdanningsdirektoratet, 2019).

Det siste eksempel jeg vil trekke frem er en type oppgaver, «Algoritmer med flytdiagram», som utgjør en tredjedel av programmeringsoppgavene i en av lærebøkene. Figur 4.3 viser et eksempel på en slik oppgave. Dette er oppgaver hvor man starter med en verdi som brukes i en ferdig algoritme for å få et svar ut av algoritmen. Alle flytdiagram-oppgavene er kodet med *aritmetiske begrep*, og i samtlige tilfeller er det enkel addisjon, subtraksjon, multiplikasjon eller divisjon. Oppgavene krever i alle tilfeller handlingen «Følge en prosedyre», og noen har deloppgaver som krever andre handlinger i tillegg. «Følge en prosedyre» i seg selv legger ikke godt til rette for utforskende læring. Ettersom oppgavene i flere tilfeller går ut på å følge en prosedyre og bruke matematikk som allerede skal være kjent, er det en svak kobling mellom matematikk og programmering. Koblingen kan være noe sterkere i de tilfeller hvor elevene blir bedt om å for eksempel forklare algoritmen i tillegg. I alle tilfellene av oppgaver med «Algoritmer med flytdiagram» er matematikk som skal være kjent for elevene brukt for å lage en kontekst for programmering.

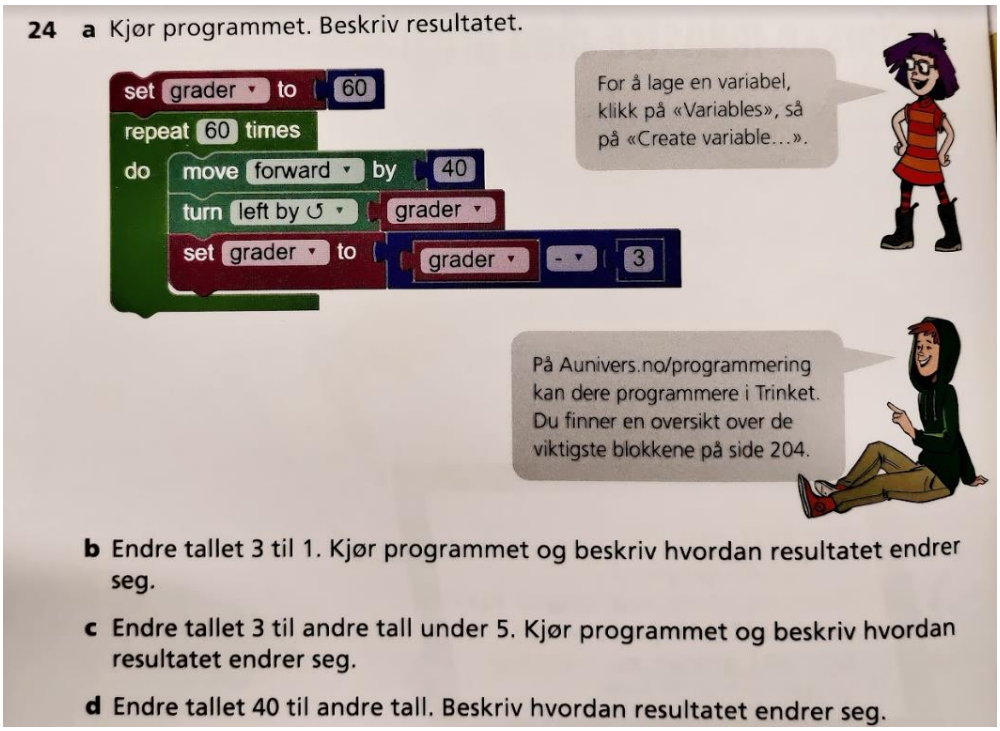
Funn 3: Utforske matematiske ideer med programmering

Noen av programmeringsoppgavene gir elevene mulighet til å utforske matematiske ideer som er nye for dem. Til sammen har jeg plassert 22 av oppgavene i denne kategorien. Disse oppgavene har matematisk innhold som er tett knyttet opp mot læreplanen på de aktuelle trinnene, og er satt i en programmeringskontekst. Det ble blant annet identifisert oppgaver i lærebok på 7. trinn med matematisk innhold som er presentert i kompetansemål på tidligere trinn. Disse blir ikke klassifisert som oppgaver som gir elevene muligheter til å utforske matematiske ideer som er nye for dem. Under presenteres en tabell som viser fordelingen av hvilke matematiske begrep som inngår i de aktuelle oppgavene. Tabell 4.6 viser at *mønstre* er det matematiske begrepet som forekommer flest ganger. To av oppgavene inneholder både *mønstre* og *geometriske begrep,* derfor blir summen av tallene i tabellen høyere enn antall oppgaver i den gitte klassifiseringen.

Matematiske begrep som kan utforskes	Antall
Mønstre (inkludert tallrekker)	14
Geometriske begrep	6
Aritmetiske begrep	4

Tabell 4.6 Fordelingen av matematiske begrep som inngår i de aktuelle oppgavene

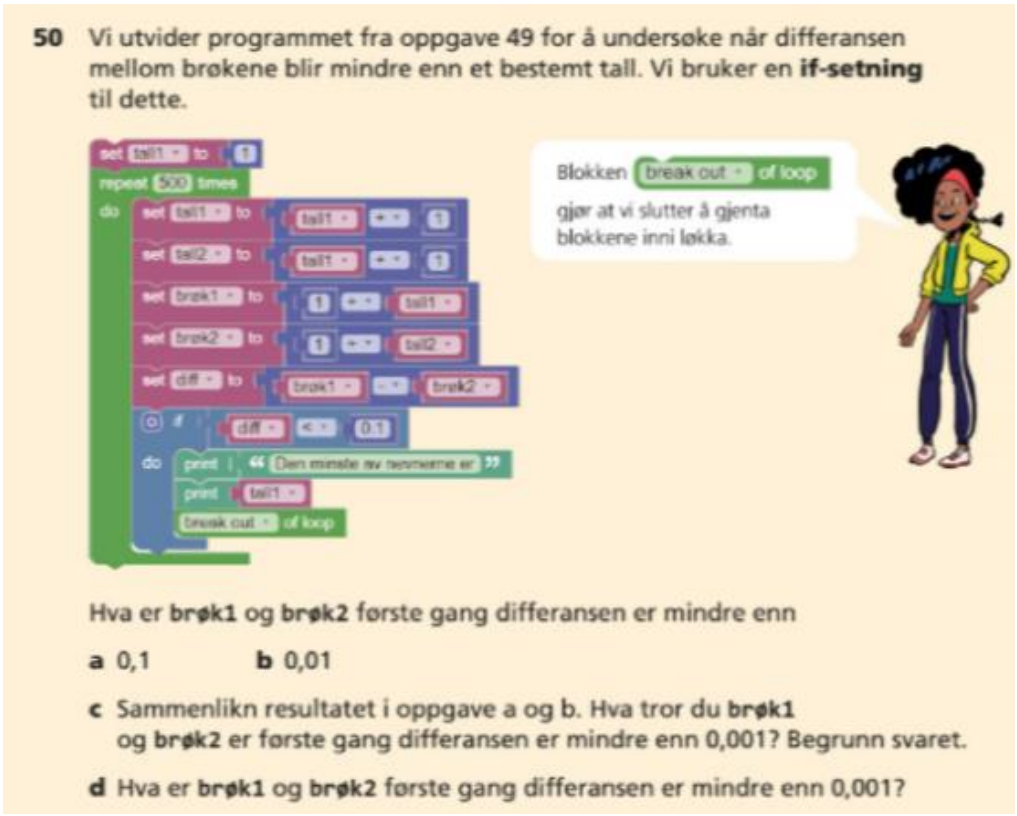
Først vil jeg trekke frem et eksempel på en oppgave hvor eleven får mulighet til å utforske mønstre i en programmeringskontekts. Figur 4.11 viser en oppgave tatt fra en lærebok på 6. trinn. Denne oppgaven har fire deloppgaver, hvor deloppgave a, b og c gir eleven mulighet til å utforske tre forskjellige geometriske mønstre ved en liten forandring i koden. Den siste deloppgaven skal elevene utforske og beskrive hva som skjer når tallet 40 endres til noe annet, altså at størrelsen på figurene vil forandres ettersom dette tallet bestemmer avstanden mellom retningsforandringene. Oppgaven som helhet krever at elevene skal kunne bruke det angitte programmeringsverktøyet for å utforske symmetri og geometriske mønstre. I tillegg krever hver deloppgave at elvene skal kunne beskrive (handlingen «forklare») hvordan resultatene forandrer seg i takt med forandringer i koden.



Figur 4.11 Utforske mønstre med programmering (Kongsnes et al., 2021b, s. 106).

Det andre eksempelet jeg vil trekke frem er fra lærebok på 7. trinn med matematisk tema *brøk* (se figur 4.12). Oppgaveteksten vist i figuren under sier eksplisitt at programmet skal brukes for å undersøke noe matematisk. I dette tilfellet skal eleven representere og bruke brøk og desimaltall, noe som er tematikken i et kompetansemål på 7. trinn. I oppgaven tar man utgangspunkt i to brøker fra oppgaven før, og skal i en programmeringskontekts undersøke når disse brøkene har en differanse som er mindre enn a) 0.1, b) 0.01 og c) 0.001. Eleven må derfor ha kunnskap om blokk-koding slik at

hen kan gjøre små forandringer i koden for å undersøke det matematiske problemet, og gjennom programmering utvikle matematisk kunnskap om brøk.



Figur 4.12 Oppgave med aritmetisk begrep (Kongsnes et al., 2021b)

4.1.4 Oppsummering av funn i analyse

I delanalyse 1 trekker jeg frem to funn knyttet til *handlinger*. Det først funnet er fordelinger av handlinger, hvor det kommer frem at «forme og skape» er den handlingen som blir identifisert flest ganger i datamaterialet. De andre handlingene er relativt likt fordelt, sett bort ifra «feilsøke» som ikke blir identifisert i noen av analyseenhetene. Det andre funnet i delanalyse 1 viser at 35 av de 101 analyserte oppgavene krever mer enn én handling, altså to eller flere.

Videre i delanalyse 2 trekkes frem funn om hvilke programmeringsbegrep og matematiske begrep som blir identifisert i oppgavene som ble analysert. Her kommer det frem at «algoritme» og «program» er begrepene som forekommer flest ganger, som oftest i den hensikt å beskrive oppgaven. Av *begrepene løkker, variabler, vilkår* og *funksjoner* er det løkker som blir identifisert flest ganger, i tett opp mot halvparten av programmeringsoppgavene. Vilkår og funksjoner er de to begrepene som forekommer færrest ganger. Når det kommer til matematiske begrep viser funn 2 i delanalyse 2 at *aritmetiske begreper*, *mønstre* og *geometriske begreper,* utgjør til sammen 98 prosent av alle oppgaver med identifisert matematisk innhold. Det er kun én oppgave med annet identifisert matematisk innhold. Aritmetiske begrep er det som identifiseres i flest tilfeller.

Siste delanalyse er det gjort tre funn som kobles spesielt mot det andre forskningsspørsmålet i denne studien; på hvilke måter programmeringsoppgavene knytter programmering og matematikk sammen. Det første funnet viser at omtrent 40 prosent av oppgavene er uten identifisert matematisk innhold, og at dette i hovedsak er oppgaver som går ut på å bruke eller lage et program som tegner en figur, eller oppgaver med rutenett. Funn 2 i denne delanalysen viser at i mange av programmeringsoppgavene brukes matematikk som allerede skal være kjent for elevene ifølge læreplanen. Allerede kjent matematikk blir brukt som kontekst for å drive med programmering, i stedet for at programmering kunne blitt brukt som et verktøy for å utvikle kunnskap og ferdigheter innenfor matematikk. Det siste funnet, funn 3, viser at 22 av oppgavene fra datamaterialet lar elevene utforske ny matematikk ved bruk av programmering. I fjorten av tilfellene var det mønstre som var det matematiske begrepet som kunne utforskes.

5 DISKUSJON

Analyseenhetene har blitt analysert gjennom tre delanalyser som resulterte i funn som ble presentert i forrige kapittel. I dette kapittelet skal jeg diskutere funnene som ble gjort opp mot teori og læreplanen for å kunne svare på forskningsspørsmålene: (1) Hva som karakteriserer programmeringsoppgavene i lærebøkene, og (2) På hvilke måter programmeringsoppgavene knytter matematikk og programmering sammen

5.1 HVILKE HANDLINGER OPPGAVENE KREVER AV ELEVENE

Det første som ble undersøkt var hvilke handlinger programmeringsoppgavene i lærebøkene krever at elevene gjør i oppgaveløsningen. Analysen viser at fem av de seks definerte handlingene blir identifisert i lærebøkene, hvor det kun er «feilsøking» som ikke forekommer i oppgavene fra datamaterialet. I denne delen vil jeg diskutere funnene gjort i delanalyse 1 opp mot relevant litteratur.

Funn fra delanalyse 1 viser at «forme og skape» er den handlingen som karakteriserer flest oppgaver i lærebøkene. Denne handlingen krever at eleven skal gjøre endringer, videreutvikle eller skape egne koder, og kan legge til rette for å utforske og undersøke på egenhånd (Benton et al., 2016). Papert (1980) tror at barn burde bruke pc for å utforske tankeprosessene sine, og at arbeid med programmeringsoppgaver kan vise seg hensiktsmessige. Oppgavene som har handlingen «forme og skape» i lærebøkene legger som regel opp til at elevene selv skal kunne utføre koding og få et program til å gjøre det oppgaven har instruert. I disse tilfellene får eleven mulighet til å prøve seg frem, feile og utarbeide en løsning. Med andre ord er dette utforsking og problemløsning, som kommer frem som et av de seks kjerneelementene som er nedfelt i læreplanen (Utdanningsdirektoratet, 2020). Jeg tenker derfor at det er positivt at «forme og skape» inngår i 59 av 101 analyseenheter, ettersom utforsking og problemløsning skal være sentralt i matematikkopplæringen. Selv om denne handlingen inngår i over halvparten av programmeringsoppgavene i lærebøkene, utelukker ikke dette at én oppgave kreve flere handlinger som også kan være gunstige i elevers læring.

Noe nokså oppsiktsvekkende som kommer frem i delanalyse 1 er at handlingen «feilsøke» ikke blir identifisert i noen av analyseenhetene. Bråting og Kilhamn (2021) kommenterer i sin diskusjon at denne handlingen ble identifisert overraskende få ganger. Feilsøking er nevnt både i rammeverket til Benton et al. (2016) og i Brennan og Resnick (2012) sitt rammeverk om algoritmisk tenkning, som begge er inspirasjon til det analytiske verktøyet brukt i Bråting og Kilhamn (2021) sin forskning. Med tanke på ideen om at elever kan lære matematikk gjennom testing og feilsøking (Papert, 1980), er det som Bråting og Kilhamn (2021) påpeker nærliggende å tro at feilsøking ville blitt vektlagt når programmering ble inkludert i lærebøkene. Oppgaver hvor eleven skal feilsøke kan skape en større variasjon i oppgavetyper, og i stor grad legge til rette for at eleven skal utforske og drive med problemløsning i matematikk. I tillegg til at feilsøking kan virke positivt på elevers læring av matematikk (Papert, 1980), er feilsøking helt elementært innenfor feltet programmering. Selv om programmeringsverktøyene tilpasset barn og unge fjerner mange av utfordringene knyttet til syntaks-feil, vil det likevel være nødvendig med feilsøking av koder for å finne ut av hvorfor programmet ikke fungerer

slik det er tenkt til å fungere (Rader et al., 1997). Det vil derfor være svært relevant for elevene å arbeide med feilsøkings-oppgaver med tanke på de reelle problemene som vil oppstå i programmering.

Litt over en tredjedel av oppgavene fra lærebøkene krever mer enn én handling av elevene, noe som kommer frem i funn 2 av delanalyse 1. Jeg oppdaget at dette gjelder i 35 av 101 tilfeller fra datamaterialet. I analysen kom det frem at i de fleste tilfellene var den kombinasjon av «forme og skape» og enten «forklare» eller «forestille seg». Bråting og Kilhamn (2021) ser stort potensiale i videreutviklingen av de svenske lærebøkene ved å spesielt ta i bruk flere oppgaver som krever handlinger som «utforske», «forestille seg» og ikke minst «feilsøke». De mener at dette kan være verktøy for å berike programmeringsinnholdet og styrke koblingen til matematikk. Benton et al. (2016) uttrykker styrkene ved «utforske», «forklare», «forestille seg», «dele» og «knytte sammen» i deres rammeverk for handlinger. På bakgrunn av dette mener jeg at oppgavene kan berikes ytterligere dersom det kreves flere ulike handlinger i samme oppgave. Et annet aspekt ved å ha oppgaver som krever ulike handlinger, er at de kan legge til rette for et bredere utvalg av tilnærminger for å løse oppgaven. Når man tenker tilpasset opplæring tar man utgangspunkt i at elevene i elevmassen har ulike styrker og svakheter, og at én mal ikke nødvendigvis passer alle. Ved å ha rike oppgaver som krever ulike handlinger kan man spille på elevenes styrker slik at de får en inngang inn i oppgaven, for å deretter utfordre de på svakhetene sine ved at andre handlinger kreves av dem. I min analyse kom det som nevnt tidligere i avsnittet frem at i de fleste av tilfellene var det en kombinasjon av «forme og skape» og enten «forklare» eller «forestille seg». Det vil si at i de fleste tilfellene er det en kombinasjon av handlinger som både Benton et al. (2016) og Bråting og Kilhamn (2021) trekker frem som handlinger med stort pedagogisk potensiale, og som kan berike oppgavenes innhold.

5.2 PROGRAMMERINGSBEGREP OG MATEMATISKE BEGREP SOM BLIR BRUKT I LÆREBØKENE

Hvilke programmeringsbegrep og matematiske begrep som blir brukt i lærebøkene, kan fortelle oss noe om innholdet i læreverkene, noe som tar meg et steg nærmere det å finne ut hva som karakteriserer programmeringsinnholdet i norske lærebøker i matematikk. I dette delkapittelet vil jeg starte med å diskutere funnene som er gjort knyttet til programmeringsbegrep, før jeg går videre i diskusjonen om de matematiske begrepene.

«Program» og «algoritme» er to begreper som blir identifisert i henholdsvis sekstitre og trettien av oppgavene, noe som kommer frem i funn 1 i delanalyse 2. Disse to begrepene blir brukte om det samme konseptet, men det er forfatterne av de ulike læreverkene som har gjort et valg om hvilket av de to begrepene som blir brukt i lærebøkene. I Multi-serien har forfatterne hovedsakelig valgt å bruke algoritme-begrepet, mens i Matemagisk-serien blir begrepet program brukt. I læreplanen står det at: «Mål for opplæringen er at elevene skal kunne lage og programmere algoritmer med bruk av variabler, vilkår og løkker» (Utdanningsdirektoratet, 2019). Dette er kompetansemål fra femtetrinn, og allerede året før skal elevene være introdusert for algoritme-begrepet i programmering. Algoritme har litt forskjellig betydning i en matematisk kontekst og i en

programmeringskontekts, og kan derfor bli behandlet som ulike begreper (Bråting og Klihamn, 2020). I matematikk er en algoritme en generell metode for å løse en gitt klasse med problemer (Brouseau, 1997 i Bråting og Kilhamn, 2021). I programmering på barne- og mellomtrinnet vil algoritmer bestå av stegvise instruksjoner for å oppnå et spesifikt mål (Bråting og Kilhamn, 2021). Det at man har samme ordet for to begreper som skiller seg fra hverandre ut ifra om det brukes i en programmeringskontekts eller en matematisk kontekst, kan være et argument for å ta i bruk begrepet algoritme fremfor program i opplæringen, slik at elevene tidlig får begynt å arbeide med begrepsforståelsen. På en annen side kan det være nyttig å bruke begrepet program for å unngå forvirring som fort vil kunne oppstå når ordet algoritme blir brukt både i en matematisk kontekst og i en programmeringskontekst. Likevel er algoritme et såpass sentralt begrep innenfor programmering, noe som betyr at det på et tidspunkt vil oppstå et behov for en begrepsforståelse for algoritme-begrepet både i en matematisk kontekst og i en programmeringskontekst.

I læreplanen blir det formidlet at elevene skal kunne bruke løkker, vilkår, variabler og funksjoner til å lage algoritmer, og til å utforske geometriske figurer og mønstre (Utdanningsdirektoratet, 2019). Analysen viser at det er mange av oppgavene som inneholder begrepet «løkker», men at det kun er 16 oppgaver fra datamaterialet som inneholder programmeringsbegrepet «funksjoner». Det er naturlig at løkker inngår i flere oppgaver enn de tre andre begrepene ettersom løkker er nyttig selv i en veldig enkel kode, men det er likevel viktig å understreke at de andre begrepene også er fundamentale i programmering. I programmering er det er helt grunnleggende å kunne definere og navngi blokker (Bråting og Kilhamn, 2021), noe Benton et al. (2016) understreker. Ettersom det er et stort sprik i antall tilfeller de ulike programmeringsbegrepene blir identifisert, kan det tyde på at noen av begrepene blir vektlagt mer enn andre. I tillegg til at funksjoner blir identifisert så lite som 16 ganger, så er det kun et fåtall av disse oppgavene som utfordrer elevene til å definere og navngi egne blokker. Bråting og Kilhamn (2021) gjorde lignende funn som meg i sin studie. De kommenter at lærebøkene har en tendens til å ha et sterkt fokus på noen få begreper, og nesten utelater andre grunnleggende begreper innenfor programmering (Bråting og Kilhamn, 2021).

Det andre funnet i delanalyse 2 viser at aritmetiske begreper er det matematiske innholdet som forekommer flest ganger i programmeringsoppgavene i lærebøkene. Mønstre og geometriske begreper utgjør det matematiske innholdet i resten av programmeringsoppgavene hvor matematisk innhold er identifisert, sett bort ifra én oppgave hvor begrepet symmetri er identifisert. Dette gjenspeiler kompetansemålene fra læreplanen som spesifikt omhandler programmering. Likevel synes jeg at man kan stille spørsmål til at begrepet symmetri ikke forekommer flere ganger i oppgaver knyttet til geometriske figurer og mønstre, da det er flere oppgaver som har potensiale til å inkludere elementer i kompetansemål som omhandler symmetri i mønstre. Læreplanen formidler et mål for opplæringen på 6. trinn, at elevene skal utforske og beskrive symmetri i mønstre (Utdanningsdirektoratet, 2019) Selv om begrepet «symmetri» kun blir brukt i en av oppgavene betyr det likevel ikke at det ikke finnes programmeringsoppgaver i lærebøkene som i utgangspunktet kan gi elevene mulighet til å utforske og beskrive symmetri i mønstre. En utfordring er at i begrepsutviklingen er det et behov for å ha ord knyttet til de matematiske begrepene (Roos & Trygg, 2018). Ettersom ordet «symmetri» ikke blir brukt i oppgavene kan dette begrense begrepsutviklingen for selve begrepet.

Funnene Bråting og Kilhamn (2021) gjorde i sin forskning på svenske lærebøker, skiller seg litt fra funnene gjort i min studie. Lærebøkene de analyserte fra fjerde- til sjettetrinn hadde en jevn fordeling mellom begrepene mønster, geometriske begreper, rotasjoner, koordinatplan og aritmetiske begreper, hvor det sistnevnte lå noen få prosent over de andre begrepene. I min studie kan man se ut fra funn 2 i delanalyse 2 at begreper som symmetri og koordinatsystem faller utenom de tre matematiske begrepene som preger resten av programmeringsoppgavene med matematisk innhold. Noe som er litt overraskende er at i mange av oppgavene fra datamaterialet benyttes rutenett, men likevel er det ingen av oppgavene hvor elevene får mulighet til å jobbe med koordinatsystem. Dette kommer jeg tilbake til i neste delkapittel hvor jeg diskuterer programmeringsoppgaver uten matematisk innhold.

5.3 Det å knytte sammen matematikk og programmering

«ScratchMaths-prosjektets mål er å gjøre det mulig for elevene å engasjere seg i og utforske viktige matematiske ideer gjennom å lære å programmere» (Benton et al., 2016). I analysen trakk jeg frem tre funn relatert til det å knytte sammen matematikk og programmering i programmeringsoppgaver. Funn 1 viser at nesten 40 prosent av programmeringsoppgavene i lærebøkene blir definert som oppgaver uten matematisk innhold. I læreplanen står det i kompetansemål etter 6. trinn at elevene at «mål for opplæringen er at elevene skal kunne bruke variabler, løkker, vilkår og funksjoner i programmering til å utforske geometriske figurer og mønstre», og i kompetansemål i 7. trinn formidles det at elevene skal kunne bruke programmering til å utforske data i tabeller og datasett (Utdanningsdirektoratet, 2019). Under grunnleggende ferdigheter i faget matematematikk står det at digitale ferdigheter innebærer blant annet å kunne bruke programmering til å utforske og løse matematiske problemer (Utdanningsdirektoratet, 2019). Læreplanen formidler at programmering ikke kun skal læres for programmeringens del, men at programmering også skal brukes som et verktøy for læring av matematikk. Gjennom programmering skal elevene få mulighet til å utforske og løse matematiske problemer. Med tanke på at læreplanen formidler dette, er det et nokså overraskende funn at nesten 40 prosent av programmeringsoppgavene blir definert som oppgaver uten matematisk innhold.

Flesteparten av oppgavene uten matematisk innhold var oppgaver med *rutenett eller program som tegner en figur*. Bråting og Kilhamn (2021) oppdaget også i sin studie at mange av oppgavene omhandlet rutenett, hvor matematisk innhold ikke ble identifisert. Begrepet rutenett er ikke nevnt i læreplanen, men koordinatsystem er et begrep vi finner i læreplanen som er nært beslektet. En forskjell på rutenett og koordinatsystem er at i rutenett er fokuset gjerne rettet mot selve rutene, mens i et koordinatsystem er det punktene som er relevante. Figur 4.9 i delanalyse 3 viser et typisk eksempel på rutenett-oppgaver fra lærebøkene. For å knytte programmering og matematikk sammen kunne man gjort et enkelt grep og benyttet seg av punktene i stedet for rutene i oppgavene. I stedet for at oppgaven handlet om forflytning rundt i rutenettet, kunne det handlet om forflytning til ulike punkter (koordinater) i koordinatsystem. Funn fra analysen viser også at det er mange oppgaver som går ut på å lage et program eller en logaritme som tegner en enkel figur eller bokstaver. Ettersom matematiske begrep er helt fraværende i disse oppgavene, og at oppgavene ikke legger til rette for å gjøre noe matematisk, har disse

oppgavene blitt kategorisert som oppgaver uten matematisk innhold. I datamaterialet finner jeg andre oppgaver som innebærer å lage programmer som tegner geometriske figurer eller mønstre. Dette er nokså like oppgaver som figur 4.6 i analysen illustrerer, både med tanke på at handlingen som kreves er «Forme og skape» og at eleven skal bruke programmering for å tegne en figur. Det som i hovedsak skiller oppgavene, er matematiske begreper som eksplisitt brukes. Et eksempel kan være en oppgave som innebærer at eleven skal lage et program som tegner en likesidet trekant, med en gitt sidelengde. På denne måten blir *geometriske begrep* en del av oppgaven, og det vil være en eksplisitt kobling mellom matematikk og programmering. Denne oppgaven krever også handlingen «Forme og skape», som kan legge til rette for at eleven får utforske matematikk. Det er med andre ord små forandringer i oppgaven som kan gi den et matematisk innhold som kan arbeides med på en utforskende måte.

Det andre funnet i delanalyse 3 viser at mange av programmeringsoppgavene inneholder matematikk som ifølge læreplanen allerede skal være kjent for elevene. Med andre ord blir allerede kjent matematikk brukt som kontekst for mange av programmeringsoppgavene, i stedet for at programmering blir et verktøy for å lære ny matematikk. Bråting og Kilhamn (2021) har gjort liknende funn i sin studie. De skriver at de matematiske begrepenes rolle i flere tilfeller ser ut til å være for å skape en kontekst for at elever kan lære programmering, heller enn å bruke programmering til å oppdage matematiske begrep (Bråting og Kilhamn, 2021). Selv om mange av oppgavene i min studie blir vurdert som at de har svak kobling mellom matematikk og programmering, kan de likevel være gode oppgaver for å lære både programmering og matematikk når man tar i betraktning elevenes nivåforskjell i matematikk i et vanlig klasserom. Det at analysen viser at det er svak kobling mellom matematikk og programmering, betyr ikke nødvendigvis at matematiske begrep er fraværende. Det kan også bety at det matematiske innholdet hører til kompetansemål fra lavere trinn, og blir derfor ikke definert som ny matematikk innenfor det rammeverket jeg forholder meg til i denne studien. Når man tar høyde for hvilken differanse det kan være på elevers utvikling og kompetanse innenfor faget i et vanlig klasserom, vil det være svært sannsynlig at flere av oppgavene som har matematisk innhold som hører til et lavere trinn, kan legge til rette for at enkeltelever skal kunne oppdage matematikk som er ny for dem. For noen elever vil for eksempel læringsmål fra 5. trinn oppleves helt ukjent og nytt selv om de går i trinnet over. Da vil disse oppgavene for noen av elevene likevel kunne brukes for å lære og oppdage ny matematikk.

Jeg har tidligere referert til Benton et al. (2016) som formidler ScratchMaths sitt mål å gjøre det mulig å utforske matematikk gjennom å lære å programmere, og læreplanen som formidler liknende mål gjennom kompetansemål og under grunnleggende ferdigheter i matematikk. På de totalt 101 programmeringsoppgavene jeg har analysert, har 22 av oppgavene blitt kategorisert som oppgaver som gir elevene mulighet til å utforske matematiske ideer som er nye for dem. Ulike kriterier må oppfylles dersom en oppgave skal kunne plasseres innenfor denne kategorien. Det første kriteriet er at oppgaven eller aktiviteten skal kreve handlinger som legger til rette for utforskende læring. Et annet kriteria er at matematiske begrep fra læreplanen kan identifiseres i programmeringsoppgaven. Det tredje kriteriet er at det skal være omformulering av ideer innenfor algoritmisk tenkning til matematisk terminologi eller tradisjonell matematisk notasjon. Og det siste kriteriet er at det er samsvar mellom det matematiske innholdet i oppgaven og kompetansemål fra tilhørende (eller høyere) alderstrinn. Funn fra delanalyse 1 viser at over halvparten av oppgavene krever handlingen «forme og

skape», som ifølge Bråting og Kilhamn (2021), basert på kunnskap fra Benton et al. (2016; 2017), kan legge til rette for utforskende læring. Handlinger som «finne ut», «forklare» og «forestille seg» blir identifisert i oppgavene, men det kunne vært hensiktsmessig at disse handlingene hadde blitt krevd i flere av oppgavene. Argumentet for dette er at de sistnevnte handlingene utfordrer elevene på ulike måter med tanke på det å utforske gjennom oppgavene. Det samme gjelder «feilsøke» som ikke er identifisert i noen av oppgavene. Likevel er det antall oppgaver med matematisk innhold, og ikke minst matematisk innhold i samsvar med kompetansemål på det gjeldene trinnet, som begrenser antall oppgaver som gir elevene mulighet til å utforske matematiske ideer som er nye for dem.

Det er viktig å presisere det at en oppgave som ikke fyller flere eller alle av kriteriene fra rammeverket, ikke nødvendigvis er en dårlig oppgave. Det kan være mulig å få godt utbytte av å jobbe med oppgaven, selv om rammeverkets kriterier ikke oppfylles. Likevel kan det være viktig å være klar over hvordan oppgavene i lærebøkene er i samsvar med mål og ønsket kompetanse fra læreplanen, slik at læreren kan gjøre grep og sikre at dette kommer frem i andre deler av undervisningen. Ofte kan læreren gjøre små endringer i oppgaven, slik at elevene blir utfordret på for eksempel andre handlinger eller matematisk innhold tilpasset deres trinn. Vi vet at læreboka har stor innflytelse på hva som undervises i skolematematikken (Lepik et al., 2015), derfor er det ekstra viktig at læreren kan kompensere for de aspektene ved programmering i matematikkfaget som oppgavene i lærebøkene ikke dekker.

5.4 Studiens begrensninger

I denne studien har jeg undersøkt hva som karakteriserer programmeringsinnholdet i lærebøker for mellomtrinnet, og hvordan programmering om matematikk knyttes sammen i oppgavene. Dette har jeg gjort ved å analysere oppgaver i lærebøker fra Multi- og Matemagisk-serien. Intensjonen har hele veien vært å undersøke forskningsspørsmålene mine – ikke å vurdere kvaliteten på læreverkene. Ettersom at jeg har analysert oppgaver, gjøres det en vurdering av hver enkel analyseenhet. Man kan si at jeg på denne måten indirekte vurderer kvaliteten på deler av læreverkene. Dette vil være vanskelig å unngå når man gjør en innholdsanalyse.

For å systematisere datainnsamlingen og analysen, tok jeg for meg læreverkene hver for seg i ulike skjemaer. På denne måten blir det tydelig for forskeren hvilke forskjeller det er på læreverkene. Hver analyseenhet ble behandlet som én av alle, uavhengig av læreverk, slik at jeg tar utgangspunkt i en felles samling av oppgaver for å svare på forskningsspørsmålene. En svakhet med dette er at læreverkene hver for seg kan ha store mangler eller lite variasjon, men når man ser på læreverkene kombinert så dekker de et bredere utvalg av *handlinger* og matematiske temaer. Dette kan gi et inntrykk av at oppgavene i lærebøkene har større variasjon enn det som er realiteten. Ettersom datamaterialet mitt er nokså begrenset, noe som jeg ser på som den største svakheten ved studien, vil ikke forskningen min si noe generelt om programmeringsoppgaver i norske lærebøker. Likevel vil det kunne gi en indikasjon på hvilke typer programmeringsoppgaver som finnes i lærebøkene. I en ideell situasjon for å kunne svare best mulig på forskningsspørsmålene mine hadde man tatt for seg alle læreverkene som blir brukt i Norge, og analysert alle eksempler, oppgaver og læringsaktiviteter både i lærerveiledning, lærebøkene og ikke minst i de digitale ressursene. På denne måten

kunne man fått et mer generelt bilde av hva som karakteriserer programmeringsinnholdet i lærebøker for mellomtrinnet i Norge.

En annen svakhet med studien er at det kun var en person som analyserte analyseenhetene. Dette gjorde at det ikke var noen å diskutere med dersom jeg var usikker på hvilke kategorier analyseenhetene skulle plasseres i, og det var heller ingen til å kontrollere at rammeverket ble brukt likt i alle tilfellene. I studien til Bråting og Kilhamn (2021) var de to forskere som kunne kontrollere hverandres analyser, og de hadde muligheten til å drøfte sammen i situasjoner hvor de var usikre på hvordan de skulle bruke det analytiske verktøyet. Ettersom det kun var jeg som gjorde analysen i denne studien, medførte dette at jeg kategoriserte og kodet analyseenhetene mange ganger for å øke sjansen for at metoden ble anvendt likt på alle analyseenhetene. Jeg har selv kontaktet forskeren Kajsa Bråting for avklaring angående usikkerheter jeg hadde ved riktig bruk av metoden, men det er likevel ingen garanti for at jeg benyttet deres analytiske verktøy på samme måte som de gjorde i sin forskning.

6 AVSLUTNING

Studien hadde som formål å bidra til mer kunnskap om programmering i skolen, og kartlegge styrker og svakheter med programmeringsoppgaver i lærebøker. For å utforske dette, valgte jeg ut to forskningsspørsmål:

(1) Hva karakteriserer programmeringsinnholdet i norske matematikkbøker på mellomtrinnet?
(2) På hvilke måter knytter programmeringsoppgaven programmering og matematikk sammen

Funn i analysen viser at det finnes programmeringsoppgaver i lærebøkene hvor kompetansemål og kjerneelementer (fra læreplanen) knyttet til programmering blir ivaretatt. Allikevel viser funnene at dette ikke gjelder majoriteten av oppgavene, noe som antyder at man ikke kan belage seg fullt og helt på læreboka for å drive undervisning som er i fullstendig dekkende for det læreplanen formidler. Fagfornyelsen er tydelig på formålet med programmering, men samlet sett så gjenspeiles ikke dette i lærebøkene. Det at lærebøkene ikke ivaretar den kompetansen som er tiltenkt matematikkfaget, kan i ytterste konsekvens føre til at elvene går glipp av muligheten til å tilegne seg viktig kompetanse, ettersom vi vet at læreboka har stor innflytelse på hva som blir undervist (Lepik et al., 2015). Læreren har derfor en viktig rolle med å tilføre elementer og tilpasse oppgavene slik at elevene kan drive med utforskende læring og bruke programmering for å lære matematikk, slik læreplanen formidler (Utdanningsdirektoratet, 2020). Funnene viser for eksempel at en stor andel av oppgavene i lærebøkene er uten matematisk innhold, og mange av oppgavene har matematisk innhold som er forventet at elevene skal ha lært tidligere i utdanningsløpet. Denne studien har belyst viktigheten av at lærere er bevisst på hvilke kompetanse som formidles i læreplanen og hvordan dette blir ivaretatt i lærebøkene. Ved å være bevisst på dette, har læreren mulighet til å kompensere for lærebokas mangler.

Noe annet denne studien kan ha bidratt med er å videreformidle et analytisk verktøy for å vurdere styrkene og svakhetene til programmeringsoppgaver. Dette kan være et nyttig verktøy for en lærer både når man skal velge ut oppgaver elevene skal jobbe med, hva som eventuelt burde tilføres eller byttes ut i oppgavene, og ikke minst når man designer egne programmeringsaktiviteter for elevene. Ettersom programmering nylig har blitt en obligatorisk del av læreplanen, er det et stort behov for økt kompetanse hos lærere for å kunne gi elevene best mulig undervisning i faget.

Noe jeg vil foreslå som en forlengelse av studien, er å kartlegge programmeringsinnholdet i digitale ressurser som brukes i skolen. Ettersom mye av oppgavematerialet ligger tilgjengelig for elevene på læreverkenes digitale ressurser, vil det være svært relevant å finne ut hvordan disse oppgavene tilrettelegger for at elevene skal utvikle matematisk kunnskap og ferdigheter gjennom programmering. De digitale ressursene ser jeg for meg at blir mer og mer brukt i skolen fremover, og vil derfor kunne ha stor innflytelse på hva som blir undervist i matematikkfaget.

Litteraturliste

Alseth, B., Arnås, A.-C., Røsseland, M., & Nordberg, G. (2021a). Multi 5b Grunnbok. Gyldendal.

Alseth, B., Arnås, A.-C., Røsseland, M., & Nordberg, G. (2021b). *Multi 6b Grunnbok. Gyldendal.*

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *Acm Inroads*, *2*(1), 48-54.

Benton, L., Hoyles, C., Kalas, I., & Noss, R. (2016). Building mathematical knowledge with programming: insights from the ScratchMaths project.

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).

Bocconi, S., Chioccariello, A., & Earp, J. (2018). The Nordic approach to introducing computational thinking and programming in compulsory education. Report prepared for the Nordic@BETT2018 Steering Group

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada.

Brennan, K., & Resnick, M. (2012). Using artifact-based interviews to study the development of computational thinking in interactive media design. Paper presented at annual American Educational Research Association meeting, Vancouver, BC, Canada

Bråting, K., & Kilhamn, C. (2021). The integration of programming in Swedish school mathematics: Investigating elementary mathematics textbooks. *Scandinavian Journal of Educational Research*.

Coulter, B., Lee, I., & Martin, F. (2010). Computational thinking for youth. In: Citeseer.

Cuny, J., Snyder, L., & Wing, J. M. (2010). Demystifying computational thinking for non-computer scientists. *Hentet fra: http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing. pdf*.

De nasjonale forskningsetiske komiteene. (2016). Forskningsetiske retningslinjer forsamfunnsvitenskap, humaniora, juss og teologi. . Hentet fra Forskningsetiske retningslinjer forsamfunnsvitenskap, humaniora, juss og teologi: https://www.etikkom.no

Drisko, J. W., & Maschi, T. (2016). *Content analysis*. Pocket Guide to Social Work Re.

Fan, L., Zhu, Y., & Miao, Z. (2013). Textbook research in mathematics education: development status and directions. *Zdm*, *45*(5), 633-646.

Fauskanger, J., & Mosvold, R. (2015). En metodisk studie av innholdsanalyse–med analyser av matematikklæreres undervisningskunnskap som eksempel. *Nordic Studies in Mathematics Education*, *20*(2).

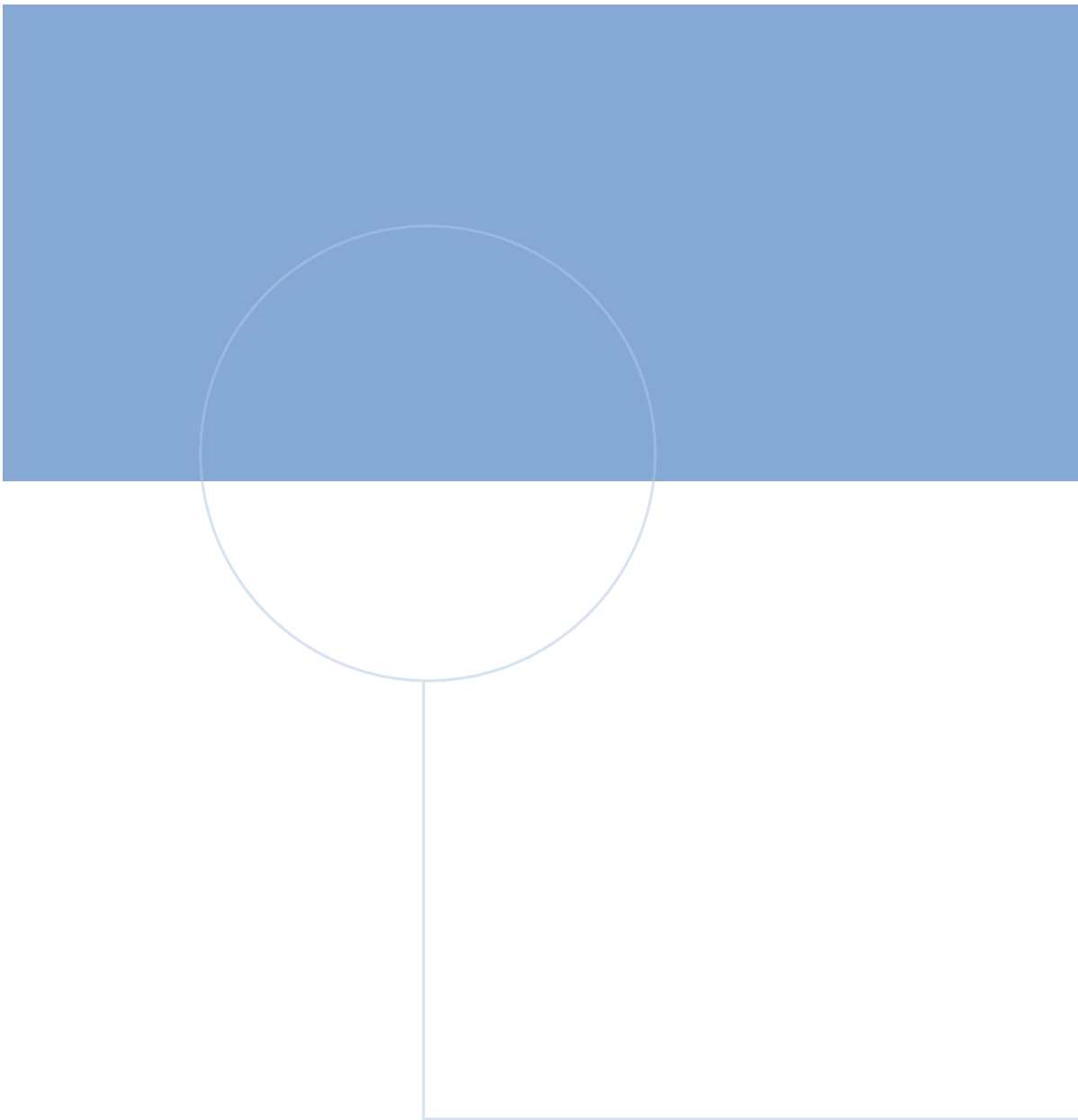
Futschek, G. (2006, November). Algorithmic thinking: the key for understanding computer science. In International conference on informatics in secondary schools-evolution and perspectives (pp. 159-168). Springer, Berlin, Heidelberg.

Gjøvik, Ø., & Torkildsen, H. A. (2019). Algoritmisk tenkning. *Tangenten–tidsskrift for matematikkundervisning*, *30*(3).

Guba, E. G. (1981). Criteria for assessing the trustworthiness of naturalistic inquiries. *Ectj*, *29*(2), 75-91.

Kongsnes, A. L., Raen, K. M., Sørdal, M., (2021a) *Matemagisk 6B.* Aschehoug.

Kongsnes, A. L., Raen, K. M., Sørdal, M., (2021b) *Matemagisk 7A. Aschehoug.*
Lepik, M., Grevholm, B., & Viholainen, A. (2015). Using textbooks in the mathematics classroom–the teachers' view. *Nordic Studies in Mathematics Education*, *20*(3-4), 129-156.
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.
Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. ACM Transactions on Computing Education (TOCE), 10(4), 1-15.
Medienorge. (2022. 24. september). Største forlagsgrupper i Norge. Hentet fra: https://medienorge.uib.no/statistikk/148
Misfeldt, M., & Ejsing-Duun, S. (2015). Learning Mathematics through Programming: An Instrumental Approach to Potentials and Pitfalls. In K. Krainer & N. Vondrová (Eds.), Proceedings of the Ninth Congress of the European Society for Research in Mathematics Education (pp. 2524–2530). Prague, Czech Republic: Charles University in Prague.
Mullis, I. V. S., Martin, M. O. & Foy, P. (2008). TIMSS 2007 international mathematics report: findings from IEA's Trends in international mathematics and science study at the fourth and eighth grades. Chestnut Hill: Boston College
Nouri, J., Zhang, L., Mannila, L., & Norén, E. (2020). Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. Education Inquiry, 11(1), 1–17. https://doi.org/10.1080/20004508.2019.1627844
Rader, C., Brand, C., & Lewis, C. (1997, March). Degrees of comprehension: Children's understanding of a visual programming environment. In Proceedings of the ACM SIGCHI Conference on Human factors in computing systems (pp. 351-358).
Roos, H., & Trygg, L. (2018). Begrepp och representationer:[ingår i Lärportalens modul Matematik-Specialpedagogik, Matematikdidaktik och specialpedagogik, Del 2: Begrepp och representationer, årskurs 4-6].
Scratch. (2022, 19. september). *How many people use Scratch?* Hentet fra: https://en.scratch-wiki.info/wiki/How_many_people_use_Scratch%3F
Stahl, N. A., & King, J. R. (2020). Expanding approaches for research: Understanding and using trustworthiness in qualitative research. *Journal of Developmental Education*, *44*(1), 26-28.
Swedish National Agency of Education. (2018). Curriculum for the compulsory school, preschool class and school-age educare 2011. Elanders Sverige AB.
Valverde, G. A., Bianchi, L. J., Wolfe, R. G., Schmidt, W. H. & Houang, R. T. (2002). According to the book. Using TIMSS to investigate the translation of policy into practice through the world of textbooks. Dordrecht: Kluwer Academic Publishers
Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35.



NTNU

Kunnskap for en bedre verden