MASTEROPPGAVE

M5GLU

Mai 2022

Programmering i norske lærebøker i matematikk

Programming in Norwegian Mathematics Textbooks

Vitenskapelig masteroppgave

30 sp oppgave

Johannes Habbestad Stokkenes

OSLOMET

OsloMet – storbyuniversitetet

Fakultet for lærerutdanning og internasjonale studier

Institutt for grunnskole- og faglærerutdanning

Sammendrag

Denne masteroppgaven handler om programmering i matematikkfaget. I arbeidet med Fagfornyelsen ble det fastsatt nye læreplaner i norsk skole, og det ble vedtatt at programmering skulle bli en integrert del av matematikkfaget. Programmering skal fungere som et verktøy for å utforske og løse problemer i matematikkfaget, samt være et viktig verktøy for å sikre elevenes opplæring i algoritmisk tenkning.

I denne oppgaven ser jeg nærmere på hva som karakteriserer programmeringsinnholdet til et utvalg læreverk i matematikk, der i hovedsak oppgavene er grunnlag analysen. Læreverkene er laget i henhold til Fagfornyelsen, og bøkene som er inkludert i analysen strekker seg mellom 5-10.trinn. Analysen tar utgangspunkt i følgende to forskningsspørsmål:

1. Hva karakteriserer programmeringsinnholdet i norske lærebøker i matematikk på mellom-og ungdomstrinnet?
2. På hvilken måte legger programmeringsoppgavene til rette for å få frem eller skape sammenhenger mellom programmering og matematikk?

For å besvare forskningsspørsmålene har jeg analysert lærebøkene ved hjelp av et analyseverktøy, inspirert av Bråting & Kilhamns (2021) analyse av programmeringsinnholdet i svenske lærebøker, som baserer seg på to rammeverk innenfor programmering og algoritmisk tenkning. I oppgaven presenteres teori om programmering, algoritmisk tenkning og programmering i matematikk. Datagrunnlaget for analysen består av 276 oppgaver fra totalt elleve bøker. Resultatene og diskusjonen viser at handlingen *følge* og *forme og skape* er de mest fremtredende i lærebøkene. Og at begrepene *steg-for-steg instruksjoner, kode* og *algoritme* var oftest inkludert i oppgavene. Enkelte sentrale ferdigheter og begreper innenfor algoritmisk tenkning og programmering var overraskende lite inkludert i oppgavene, som *feilsøking/debugging*. Lærebøkene inneholder oppgaver som kan gi elevene mulighet til å bruke programmering til å utforske matematiske begreper og idéer, men slike oppgaver er det relativt sjeldne.

Abstract

This master´s thesis is about programming in the school subject mathematics. In the renewal of the national curricula in 2020 in Norway, it were decided to implement programming as a part of mathematics. Programming should be used as a tool to explore and solve problems in the subject of mathematics. In addition to work as an important tool to teach computational thinking.

In this thesis I take a closer look at what characterizes the programming content in a selection of Norwegian mathematics textbooks from grades 5-10, where the tasks are mainly the foundation for the analysis. All the textbooks included in the analysis is made after the new curricula implemented in 2020. The following two research questions will be discussed:

1. What characterizes the programming content in Norwegian textbooks in school mathematics?
2. In what way does the programming tasks create a bridge between programming and mathematics?

In order to answer these research questions I've analyzed the textbooks using a analytical tool, inspired by Bråting & Kilhamn´s (2021) analysis of the programming content in Swedish mathematics textbooks, which is based on two theoretical frameworks which includes actions and concepts that are important for teaching in computational thinking and programming. The thesis presents theory about programming, computational thinking and programming in mathematics. The data basis for the analysis consists of a total of 276 tasks from the in total eleven books. The results and discussion reveals that the action *follow* and *form and create* are the two most prominent in the textbooks. And the concepts *stepwise instructions* and *algorithm* were most often included in the tasks. In addition, the results show that important and basic computational concepts, such as debugging, are rarely included in the tasks. The textbooks contains some tasks that gives the students the opportunity to use programming to explore mathematical concepts and ideas. Although, these types of tasks are relatively rare.

Forord

Denne masteroppgaven utgjør siste del av et femårig utdanningsløp som har vært svært lærerike, til tross for at vi har vært til dels prøvekaniner i innføringen av femårig lærerutdanning. I løpet av utdanningen har vi hatt mange dyktige forelesere, gjesteforelesere og praksislærere som underveis har formet meg på veien til å bli utdannet lærer. Selve utdanningen med å bli lærer mener jeg ikke stopper her, det blir en livslang utdanning ute i skolen for å etterstrebe best mulig opplæring for de fremtidige generasjonene.

I valget av tema til denne oppgaven har jeg valgt noe som jeg i utdanningsløpet har ønsket mer kunnskap om før jeg skal ut i skolen. Programmering og algoritmisk tenkning er og vil være viktige ferdigheter i samfunnet, og med god opplæring kan vi i skolen sikre at elevene blir digitalt kompetente for fremtiden.

I arbeidet med denne masteroppgaven vil jeg takke min veileder Henrik Forssell for råd, tips og konstruktive tilbakemeldinger gjennom denne prosessen. En takk er også på plass til alle fantastiske studievenner for fem innholdsrike år.

En stor takk til min trofaste samboer som har støttet meg i denne perioden og generelt i hele utdanningsforløpet. Og takk for å ha lettet på de fleste husarbeidsoppgavene dette halvåret slik at jeg har kunnet fokusere på denne oppgaven.

Sist, men ikke minst. Takk til min fantastiske familie som har støttet meg, bidratt, kommet på besøk og latt gutterommet stå åpent for hyggelige ferier gjennom alle disse fem årene.

Innholdsfortegnelse

Figurliste

Tabeller

1 Innledning

Samfunnet vi lever i blir stadig mer digitalisert. Med ny teknologi og utvikling i samfunnet spiller skolen en viktig rolle ved å gi fremtidens generasjon mulighet til å tilegne seg kunnskap og ferdigheter for fremtiden (Kunnskapsdepartementet, 2017; Sevik et al., 2016). Elevene vil i større eller mindre grad være involvert i utvikling av ny teknologi i fremtiden. Digitale ferdigheter har vært i fokus i skolen tidligere, men det har blitt kritisert for å være forbrukerorientert. Elevene må ha bredere forståelse og innblikk i hvordan teknologi fungerer og hvordan den utvikles (Johansson, 2020; Forsstrøm & Kaufmann, 2018; Sanne et al., 2016). Skolen har et sosialt mandat og skal gi elevene mulighet til å delta og bli medborgere i samfunnet. Programmering er en ferdighet som skal bidra til at elevene får en bredere forståelse for teknologi, ta stilling til etiske dilemmaer i møte med ny teknologi, samt tilegne seg kunnskap om hvordan teknologi kan fremmer eller hemmer bærekraftig utvikling (Sanne et al., 2016).

Ferdigheter innenfor programmering har blitt en stadig viktigere kompetanse i det som omtales som «21st-century skills», som er ferdigheter og kompetanser som er og vil bli sentrale i fremtiden. Programmering er ofte sett på som et pedagogisk verktøy for å utvikle elevers *computational thinking* (CT). CT kan sammenlignes med prosessen der vi forklarer, analyserer og løser problemer slik at en datamaskin kan løse oppgaven (Bråting & KIlhamn, 2021; Wing, 2006). Dybdelæring er et sentralt element i fagfornyelsen. Programmering kan være med som et ledd i dybdelæringen ved at elevene kan utforske kjente og ukjente konsepter på flere måter (Haraldsrud, Sveinsson & Løvold, 2020). I matematikkfaget kan programmering bidra som et verktøy for å utforske nye fenomener, effektivisere beregninger og generalisere løsningsstrategier.

1.1 Bakgrunn for studien

De fleste europeiske land har inkludert programmering i læreplanen for å kunne bidra til at elevene tilegner seg ferdigheter som logisk tenkning og problemløsning (Bocconi, Chioccariello & Earp, 2018; Balanskat, Engelhardt & Ferrari, 2017). Hvordan landene har implementert programmering i skolen varierer fra å danne et eget fag, ha det som en tverrfaglig kompetanse eller at det blir en del av et eller flere allerede eksisterende fag. I den norske læreplanen er det inkludert gjennom en blanding av tverrfaglig og i allerede

eksisterende fag. I norsk skole skal programmering primært innføres i matematikkfaget, men er også nevnt i læreplanen i musikk, kunst og håndverk og naturfag.

I denne studien vil jeg analysere innholdet av programmering som nylig er blitt en del av norske lærebøker i matematikk for grunnskolen. Målet med studien er å få en oversikt over hva som karakteriserer innholdet som er knyttet til programmering i lærebøkene, og diskutere hvordan innholdet kan påvirke elevenes muligheter til å lære matematikk med programmering som verktøy. Lærebøker er et viktig verktøy for lærere i planlegging av undervisning (Johansson, 2006; Kongelf, 2019). I tillegg blir lærebøker i matematikk ofte brukt til å gi elevene individualisert undervisning, ved at elevene arbeider med oppgavene i lærebøkene i sitt eget tempo (Neuman et al., 2015). Undersøkelser fra norske klasserom viser at lærebøker brukes svært aktivt i matematikkundervisningen (Gilje et al., 2016). På denne måten kan vi gjennom å undersøke innholdet i matematikklærebøkene få et innblikk i hvilke læringsmuligheter elevene har tilgang til (Jablonka & Johansson, 2010).

1.2 Forskningsspørsmål

Interessen for å skrive om programmering i matematikkfaget stammer fra blant annet debatten om implementeringen av programmering i matematikkfaget og ut fra en egen interesse om å opparbeide meg mer kunnskap om hva programmering kan tilføye skolen, og mer spesifikt matematikkfaget. Diskusjonen rundt innføringen av programmering har satt fokus på manglende forkunnskaper og kompetanse på dette fagområdet i den norske lærerstaben (Johansen, 2020). Og det settes i gang tiltak for å øke kompetansen blant lærere gjennom videreutdanning og nettbaserte introduksjonskurs som «kompetansepakken for programmering og algoritmisk tenkning» (Utdanningsdirektoratet, 2020b). I prosessen med å finne ut spesifikt hva jeg skulle skrive om leste jeg en svensk studie der de undersøkte innholdet av programmering i svenske lærebøker i matematikk (Bråting & Kilhamn, 2021). Denne studien skapte grunnlaget for det jeg ønsker å undersøke, og jeg ønsket gjennomføre en lignende studie på norske læreverk i matematikk.

Lærebøker står sterkt i matematikkfaget, blir mye brukt. Det kan derfor tenkes at lærere med lite kompetanse innenfor programmering vil benytte seg av innholdet i lærebøkene i planlegging og gjennomføring av undervisning i programmering. I tillegg viser forskning at norske elever arbeider relativt mye med oppgaver fra lærebøker i matematikkundervisningen,

sett opp mot det internasjonale gjennomsnittet (Grønmo et al., 2004; Grønmo & Onstad, 2009; Klette, 2003). Derfor er det interessant å undersøke innholdet av programmering i lærebøkene, for å få et innblikk i hvilke muligheter elevene har for å lære seg programmering, og til å lære seg å bruke programmering til å utforske og lære matematikk, . Jeg stiller følgende to forskningsspørsmål:

1. Hva karakteriserer programmeringsinnholdet i norske lærebøker i matematikk på mellom-og ungdomstrinnet?
2. På hvilken måte legger programmeringsoppgavene til rette for å få frem eller skape sammenhenger mellom programmering og matematikk?

For å undersøke disse forskningsspørsmålene har jeg benyttet meg av samme analyseverktøy som i studien til Bråting & Kilhamn (2021). Deres analyseverktøy baserer seg på en kombinasjon av Brennan & Resnick´s (2012) rammeverk for *computational concepts* (begreper innenfor algoritmisk tenkning) og Benton et al.´s (2016) rammeverk for handlinger, omtalt som *the 5E´s.* Analyseverktøyet består av de tre hovedkategoriene, handlinger, begreper og det å undersøke sammenhengene mellom programmering og matematikk (Bråting & Kilhamn, 2021). I sammenheng med forskningsspørsmål 1 vil handlingene og begrepene fra analyseverktøyet brukes til å undersøke hva som karakteriserer programmeringsinnholdet, det samme for forskningsspørsmål 2 i tillegg til notater underveis i analyseprosessen. I analyseprosessen har jeg også valgt å skille mellom 5-7 og 8-10 for å undersøke om det er noen karakteristiske forskjeller mellom innholdet elevene møter på mellom- og ungdomstrinnet.

1.3 Oppgavens struktur

Kapittel 2 starter med en innledning for programmering i skolen etterfulgt av et historisk tilbakeblikk på programmering i skolesammenheng, i tillegg til erfaringer fra noen europeiske land med implementering av programmering i skolen. Videre vil jeg i teorikapittelet ta for meg relevant litteratur som omhandler algoritmisk tenkning og programmering i matematikkfaget. I metodekapittelet vil jeg først å fremst ta for meg metodiske valg, redegjøre for analyseverktøyet som er benyttet og hvorfor, presentere utvalget og diskutere oppgavens validitet og relabilitet. I det neste kapittelet, 4 – resultat, vil jeg presentere funn fra analysen. Og deretter diskutere disse funnene, og knytte det opp mot relevant teori.

Avslutningsvis oppsummeres hovedfunnene i oppgaven og forskningsspørsmålene besvares på bakgrunn av studiens funn og relevant teori.

2 Teori

I dette kapittelet vil jeg presentere relevant litteratur og forskning. Kapittelet starter med en generell beskrivelse av programmering, før jeg går over på å gi et historisk tilbakeblikk på programmering i skolesammenheng. Videre ser jeg nærmere på programmering i dagens skole, samt erfaringer med programmering i skolen fra andre land. I kapittel 2.2 beskrives algoritmisk tenkning, og forholdet mellom programmering og algoritmisk tenkning. Deretter presenteres litteratur som omhandler programmering i matematikkfaget, lærebokens rolle i skolematematikken og tidligere forskning.

2.1 Programmering

Programmering blir for mange kun forbundet med å skrive koder for å få en datamaskin eller annen teknologi til å gjøre det man ber den om, men begrepet omhandler mer enn dette. Programmering er mer enn bare koding. Koding er selve aktiviteten med å skrive en kode, mens programmering blant annet omhandler hele prosessen som innebærer tankeprosessen med å identifisere og løse et problem, som en kan løses med hjelp av en datamaskin. Det assosieres blant annet med utforskning, feilsøking og problemløsning (Sevik et al., 2016). For å kunne analysere oppgavene i matematikkbøkene er det viktig å danne seg et helhetlig og moderne bilde av hva programmering er og hva det innebærer. Programmering utover det å skrive en programkode inneholder hele prosessen med å komme frem til en kode. Fra det å identifisere problemet og tenke ut mulige løsninger på problemet, til prosessen der man skriver en kode, som kan forstås av en datamaskin og som skal løse problemet. Og til å gjennomføre feilsøking og kontinuerlig utbedring av denne koden (Sevik et al., 2016). Koding derimot er selve aktiviteten med å formulere dette inn i et bestemt programmeringsspråk (Gjøvik & Torkildsen, 2019).

I skolesammenheng er programmering et redskap for å arbeide med problemløsning med hjelp av teknologi, digital kompetanse, kommunikasjon og ansvarsfull bevissthet i den digitale verden (Manilla, 2018; Sevik et al., 2016). Manilla (2018) mener at siden programmering er relativt nytt i skolensammenheng så finnes det ikke noe fastsatt progresjon i hvordan man bør arbeide med programmering i skolen og i de ulike klassetrinnene. Hun trekker frem at man bør begynne med det grunnleggende når det kommer til å lære seg programmering, men at styringsdokumentene må ligge til grunn for hvilke pedagogiske valg man gjør. Manilla (2018) legger til at det fremdeles ikke er klart hvordan det skal undervises i

programmering i skolen. Det trengs mer erfaring om hvordan man skal undervise programmering i skolen, og at denne erfaringen kan gi et bedre bilde på hvordan progresjonen bør være (Manilla, 2018).

2.1.1 Historisk tilbakeblikk

Seymour Papert (1980) blir ofte omtalt som pioneren når det kommer til å inkludere programmering i undervisning med barn og unge. Hensikten var å lage et programmeringsspråk der yngre elever kunne lære matematikk, og da spesielt utforske geometri (Papert, 1980). Det tekstbaserte programmeringsspråket Logo ble lansert i 1967 gjennom "The Logo Project" av Seymour Papert og hans kolleger på MIT (Massachusetts Institute of Technology) (Feurzieg, Papert, Bloom, Grant & Solomon, 1970). Logo ble laget som et verktøy for å programmere en virtuell skilpadde som kunne utføre ulike kommandoer. Skilpadden ble beskrevet som "objects to think with" (Papert, 1980). Og tanken var at elevene skulle kunne arbeide med programmering og matematikk samtidig.

I Paperts (1980) forsøk viste han at det var mulig for elever i ung alder å lage algoritmer. Elevene kunne i større grad arbeide utforskende gjennom Logo ved at datamaskinen gjorde det lettere og mer tilgjengelig for å elevene å utforske, prøve og feile, og finne løsninger på problemer. Papert selv mente at han hadde stor suksess når elevene arbeidet på denne måten, og mente at denne typen opplæring burde være et ideale i skolen. Elevene fikk tilgang til faget på en ny måte, som kunne hjelpe de å få en bredere og dypere forståelse. Han argumenterte også for at det å arbeide på denne måten over tid hadde overføringsverdier til andre områder utenfor programmering (Papert, 1980).

Selv om Papert selv hevdet at arbeidsmetoden med programmering kunne utvikle elvenes kognitive evner, hadde han noen kritikere. Pea & Kurland (1984) stilte seg særlig kritisk til påstanden om at programmering i Logo ville gi bedre kognitive ferdigheter. De pekte på studier som viste til at programmering ikke førte til bedre problemløsning eller matematisk forståelse (Pea & Kurland, 1984). Likevel erkjente de at programmering og problemløsning har en del like komponenter som potensielt kan fremme tenkeferdigheter, men at det var behov for mer forskning på området for å kunne konkludere med at programmering gir faglig utbytte og overføringsverdi (Pea & Kurland, 1984). En annen forskningsstudie stilte seg også kritisk til Paperts påstand om at læring i programmering gav forbedret tenkning i andre fag.

Men at det kunne gi en effekt på elevenes tenkeferdigheter i relasjon til språket som blir lært (Mayer, Dyck & Vilberg, 1986).

Paperts ideal med at programmering skulle bli en viktig del i skolen ble ikke en realitet av flere ulike årsaker. Forskningen på 1980-tallet pekte mer på at programmering ikke hadde den antatte overføringsverdien til andre fag. Drijvers et al (2009) peker også på begrensninger innen teknologien og tilgang på denne. I tillegg til manglende tilgang på kompetente lærere på fagområdet.

I Mønsterplanen M87 fikk programmering noe plass i matematikkfaget. I læreplanen M87 ble «datalære» knyttet opp mot matematikkfaget, og skulle ta utgangspunkt i algoritmebegrepet. Videre blir det også knyttet nært opp mot emne problemløsning (Bueie, 2019). Om man ser nærmere på M87s beskrivelse av datalære ser vi at den beskriver arealberegning, tilnærmingsmetoder, simulering og ligningsløsning, som emner der datamaskiner kan fungere som et nyttig verktøy (Bueie, 2019). Til tross de gode intensjonene i M87 så fikk ikke programmering en sentral rolle i matematikkfaget, eller i skolen generelt. Mulige forklaringer for dette er nok kanskje at det teknologiske utgangspunktet den gangen ikke er det samme som i dag (Bueie, 2019). Dette kombinert med tilgangen på utstyr, og tilpassede brukergrensesnitt. I de neste læreplanen L97 og K06 ble programmering lite vektlagt (Bueie, 2019).

2.1.2 Programmering i dagens skole

Sammenlignet med situasjonen på 1980-tallet er tilgjengeligheten på teknologi totalt annerledes. Vi lever i et samfunn der vi konstant har den digitale verden i umiddelbar nærhet. Dermed har mulighetene for programmering også blitt mer tilgjengelig, gjennom flere programmerbare objekter og språk. Og både de programmerbare objektene og språkene har hatt en svært stor utvikling. I tillegg til flere programmeringsspråk som er spesifikt rettet til opplæring av barn, som Scratch, Blockly og micro.bit (Haraldsrud et al., 2020)

De siste to tiårene har debatten om programmering i skolen fått økt fokus, spesielt i sammenheng med algoritmisk tenkning (AT) (Bueie, 2019; Wing, 2006). Programmering ble mye diskutert i forbindelse med utformingen av LK20 og spesielt den nye læreplanen i matematikk. Det ble diskutert mye rundt hvorvidt programmering skulle få et eget fag i skolen, eller om det skulle bli implementert inn i andre fag. Der det særlig ble diskutert om

programmering skulle inn i matematikkfaget (Stenseth, Kaufmann & Forsstrøm, 2019). Slik Bueie (2019) ser det er det naturlig at matematikkfaget får hovedansvaret for programmering. I den nye læreplanen er det fastsatt at elevene skal lære å programmere i matematikk, naturfag, musikk, og kunst og håndverk, der matematikk er tildelt hovedansvaret for programmeringsopplæringen (Flø, 2020).

Gjøvik og Torkildsen (2019) skriver om to sider av debatten om programmering i LK20 der den ene siden omtales som teknologioptimister og den andre teknologipessimister. Samtidig er det relativt bred enighet i dagens samfunn om at elevene bør undervises i ulike digitale ferdigheter. Innunder de digitale ferdighetene skal elevene lære seg grunnleggende ferdigheter i programmering. Dette for å kunne bli medborgere i et samfunn preget av mer digital og teknologisk hverdag. I tillegg til å kunne bidra til videre teknologisk utvikling (Stenseth et al., 2019). Men det er viktig å påpeke at implementering av programmering i læreplanen alene ikke oppnår disse målene eller automatisk gir positiv læringseffekt. Det er nødvendig å ha lærere både med god kompetanse i programmering og faglig kompetanse (Drijvers et al., 2009).

Waite (2018) peker på manglende kompetanse og gode pedagogiske ressurser når det kommer til undervisning i programmering i skolen. Innføringen av et fagområde som er nytt og ukjent for store deler av lærerbestanden følger med en rekke utfordringer. Utdanningsdirektoratet anerkjenner disse utfordringene og har kommet med en rekke tiltak som skal hjelpe skoler og lærere i overgangsfasen. De har blant annet gitt ekstra midler til skoler som prioriterer videreutdanning og kompetanseheving i programmering. Midlene er ment å gå til utstyr til undervisning og videre-/etterutdanning av lærere (Utdanningsdirektoratet, 2020b).

Utdanningsdirektoratet (2020b) er tydelig på at de vet at mange lærere er usikre på hvordan de skal undervise i programmering og algoritmisk tenkning i sine fag. For å støtte lærerne har de laget en kompetansepakke som er rettet mot de ulike fagene i skolen som har kompetansemål knyttet til programmering. Her vektlegges det at undervisning i programmering skal gi elevene flere verktøy til å løse problemer og oppgaver. Kompetansepakken er delt inn i en fellesmodul og fagspesifikke moduler slik at lærerne skal få oversikt og spesialkompetanse i programmeringsundervisning i sitt fag (Utdanningsdirektoratet, 2020b). Pakken legger opp til samarbeid med kollegaer, og består av forarbeid, utprøving og etterarbeid. Ser vi nærmere på kompetansepakken for programmering

i matematikk legges det frem fire argumenter for hvorfor programmering skal styrke undervisning i matematikk:

1. **Problemløsning*:*** Elevene skal lære å bli problemløsere. Med programmering kan de gå løs på helt nye og mer virkelighetsnære problemstillinger, med flere strategier og effektive verktøy. For eksempel kan elevene samle egne datasett med mobilen og deretter utforske datasettene ved hjelp av programmering.
2. **Eksperimentering, prøving og feiling:** Programmering åpner for eksperimentering, prøving og feiling på en annen måte enn vi kanskje er vant med fra før. Hvis programmet ikke fungerer, får du vite det med en gang. Da er det bare å lete etter egne feil og prøve på nytt. Slike umiddelbare tilbakemeldinger kan føre til større utholdenhet hos elevene dine.
3. **Læring for flere:** Elevene skal få lov til å finne sin vei inn i matematikken. Noen elever liker å tenke og abstrahere, andre liker å fikle, teste og eksperimentere. Programmering og algoritmisk tenkning åpner for mange ulike tilnærminger til matematikk. Hvis du som lærer behersker mange ulike metoder, kan du lettere tilpasse opplæringen til hver enkelt elev.
4. **Dybdelæring:** En av nøklene til dybdelæring er bruk av ulike representasjonsformer. Matematiske problemer kan ofte løses med og uten programmering. Å klare begge deler kan gi en dypere forståelse for matematikken. Samtidig utvikler elevene en bevissthet om hvilke verktøy som egner seg for hvilke problemer.

(Utdanningsdirektoratet, 2020b)

I rapporten «programmering i skolen» av Sevik et al. (2016) ser de nærmere på hvordan og hvorfor programmering skal inkluderes i skolen. I notatet fremkommer det at prosessen med å programmere har tydelige likhetstrekk med problemløsning (Sevik et al., 2016). Og at programmering kan være med på å gi kompetanse i å lære. Elevene lærer seg å lære gjennom strukturerte aktiviteter med prøving og feiling, systematisk feilsøking og ved å finne de beste løsningene (Sevik et al., 2016; Utdanningsdirektoratet, 2019a).

Sevik et al. (2016) trekker frem at implementering i eksisterende fag ikke nødvendigvis er den beste løsningen i et lengre tidsperspektiv, men at de anbefaler at det på sikt blir et eget teknologi og programmeringsfag i skolen for å sikre kvalitet og god læring. Som nevnt

tidligere er det matematikk som har fått hovedansvaret for programmeringsopplæringen i norsk skole. Matematikkfaget skal gi elevene kunnskaper om grunnstrukturene i programmering for å kunne anvende programmering i andre fag, og i tillegg utforske matematikk gjennom programmering (Kunnskapsdepartementet, 2019). I tillegg skal elevene lære å bruke algoritmisk tenkning som en problemløsningsstrategi og vurdere når det er hensiktsmessig å bruke digitale hjelpemidler, inkludert programmering (Flø, 2021). Hvis vi ser på kompetansemålene i matematikk, er det omlag et kompetansemål i snitt hvert år i grunnskolen som er knyttet til programmering:

2. Trinn - lage og følge regler og trinnvise instruksjoner i lek og spill
3. Trinn - lage og følge regler og trinnvise instruksjoner i lek og spill knyttet til koordinatsystemet
4. Trinn - lage og programmere algoritmer med bruk av variabler, vilkår og løkker
5. Trinn - lage og programmere algoritmer med bruk av variabler, vilkår og løkker
6. Trinn - bruke variabler, løkker, vilkår og funksjoner i programmering til å utforske geometriske figurer og mønstre
7. Trinn - bruke programmering til å utforske data i tabeller og datasett
8. Trinn - utforske hvordan algoritmer kan skapes, testes og forbedres ved hjelp av programmering
9. Trinn - simulere utfall i tilfeldige forsøk og beregne sannsynligheten for at noe skal inntreffe, ved å bruke programmering
10. Trinn - utforske matematiske egenskaper og sammenhenger ved å bruke programmering

(Kunnskapsdepartementet, 2019)

Ser vi nærmere på kompetansemålene ser vi at elevene skal lære seg grunnleggende begreper innenfor programmering som løkker, variabler, funksjoner, vilkår o.l. Slike begreper kan være krevende for elever som ikke har tidligere programmeringskunnskaper, og det er viktig at lærere forklarer disse begrepene eksplisitt i sin undervisning (Grover et al., 2015). En velkjent metode for å lære bort disse begrepene er gjennom analog programmeringsaktivitet, altså uten digitale hjelpemidler (Flø, 2020). Man ser at kompetansemålene legger til rette for å drive med analog programmering. Samtidig er det viktig å være oppmerksom på at lærere og elever ofte har et snevert syn på opplæring i programmering. Det ses ofte i sammenheng med mye steg for steg instruksjoner (Benton et al., 2016). Derfor er det viktig med kompetanseheving og tilgang på pedagogiske verktøy for å løfte programmeringsundervisningen. Lærerne må se verdien i programmering ved at det kan styrke elevenes algoritmiske tenkning (Benton et al., 2016).

I de ulike kompetansemålene ser vi også flere mål som kan knyttes til algoritmisk tenkning. I tillegg finner vi under kjerneelementer, fagets relevans og grunnleggende ferdigheter, både indirekte og direkte henvisninger som kan knyttes til programmering. Under kjerneelementet utforskning og problemløsning beskrives det at elevene skal lære seg å se etter mønstre, sammenhenger og diskutere seg frem til felles forståelse. I tillegg til at fremgangsmetodene og strategier ilegges større verdi enn selve svaret (Kunnskapsdepartementet, 2019). Her kan algoritmisk tenkning være et nyttig verktøy for å oppnå nettopp dette, og programmering ses på som et naturlig miljø for å implementere algoritmisk tenkning (Gjøvik & Torkildsen, 2019).

2.1.3 Dybdelæring

Erfaringer fra den forrige reformen LK06 viste et behov for å kutte ned antall temaer og mål i læreplanene i de ulike fagene (Gilje, Landfald & Ludvigsen, 2018). En av utfordringene til lærerne var at mengden av ulike temaer innenfor fagene førte til at det ble utfordrende å legge til rette for dybdelæring. Essensen i dybdelæring er å lære seg noe så godt at du forstår sammenhenger og kan bruke dette i nye situasjoner (Utdanningsdirektoratet, 2019b). Refleksjon over egen læring er viktig for å bli kjent med læringsmetoder som fungerer godt for den enkelte, og for å utvikle egen evne i å lære (Utdanningsdirektoratet, 2019b). Hvordan elevene lærer er altså like viktig som det man lærer om. Det stilles også krav om at læreren innehar nok kompetanse og fagdidaktisk forståelse om tema som undervises (Gilje et al., 2018).

I den forrige læreplanreformen ble det mer vektlagt at elevene skulle lære å bruke digitale hjelpemidler som et verktøy (Bueie, 2019; Haraldsrud et al., 2020). I motsetning legger fagfornyelsen mer til rette for at elevene skal lære seg å bruke digitale hjelpemidler mer aktivt i undervisningen med programmering (Utdanningsdirektoratet, 2020b;c). Elevene skal får et innblikk i hvordan dataprogrammer er bygd opp gjennom en mer faglig tilnærming som skal styrke elevenes digitale ferdigheter (Sevik et al., 2016; Tellefsen, 2018). Gjennom programmering kan elevene undersøke kjente og ukjente konsepter på ulike måter, på tvers av fag, noe som legger til rette for dybdelæring (Haraldsrud et al., 2020).

2.1.4 Erfaringer med programmering fra noen europeiske land

I europeisk sammenheng var Norge relativt sent ute med å inkludere programmering i de nasjonale læreplanene (Bocconi et al., 2018). Det er derfor interessant å se på andre lands erfaringer med innføring av programmering i skolen. England var en av de første landene til å inkludere programmering i læreplanene. Dette gjennom et eget obligatorisk fag, *Computing,* fra og med første klasse på barneskolen. Erfaringer fra England viser at britiske lærere støtte på en rekke utfordringer etter innføringen. Flere lærere meldte om manglende selvtillit i å undervise, selv etter å ha deltatt på flere kurs (Sentance & Csizmadia, 2017). Dette finner vi også igjen i en gjennomgang av utdanningsreformen i Storbritannia av The Royal Society, der de trekker frem at lærere selv sier at de ikke innehar nødvendig kompetanse på fagområdet (Crick, 2017; Larke, 2019). Variasjoner i tolkninger av læreplanen blant lærerne trekkes også frem som en utfordring.

Sverige innførte programmering i læreplanene som trådde i kraft høsten 2018. Sveriges implementering kan i stor grad sammenlignes med den norske ved at programmering i hovedsak er ilagt innunder matematikkfaget, men også noe i andre fag som det svenske teknologifaget *teknik* og i formgivingsfaget, i likhet med naturfag og kunst og handverk i norsk skole. I Sverige blir programmering introdusert i hovedsak gjennom algebra, som er relativt unikt, i tillegg til et fokus på problemløsning (Skolverket, 2019).

I en svensk casestudie følges fire lærere det første halvåret etter implementeringen av programmering. Studien viser til at lærerne i stor grad gjennomfører opplegg som stiller krav til ferdigheter som samarbeid, kommunikasjon og problemløsning (Stigberg & Stigberg, 2019). Lærerne erfarte også at elevene ble gitt muligheter til å øve på matematikkferdigheter i tråd med læreplanen gjennom programmering. Samtidig pekte lærerne på noen utfordringer som manglende læremateriale og pedagogiske opplegg som er tilpasset matematikkfaget (Stigberg & Stigberg, 2019). En lignende studie intervjuet åtte lærere på ungdomskoler i Sverige hvor det kom frem at majoriteten av lærerne var positive til innføringen av programmering (Mozelius, Ulfenborg & Persson, 2019). Selv om de var positive til innføringen gav også disse lærerne uttrykk for en del utfordringer. De fremhevet spesielt sin egen mangelfulle kompetanse på området og tilgjengelighet på tilrettelagte opplegg. I tillegg nevnte noen av lærerne utfordringer knyttet til tid, og syntes det var vanskelig å inkludere programmering i en allerede travel timeplan.

Finland innførte nye læreplaner høsten 2016 der programmering ble implementert i flere fag (Sevik et al., 2016). Programmering er inkludert i Finland som et tverrfaglig element i alle fag, som en del av elevenes digitale kompetanse. Det er en generell kompetanse som skal utvikles i alle fag, men som er eksplisitt del av matematikkfaget og kunst- og håndverksfaget (Opetushallitus, 2014). I matematikkfaget knyttes programmering tett opp mot problemløsning. Men det er i liten grad beskrevet hvordan programmering skal implementeres i finsk skole. Mannila (2018) presenterer noen erfaringer, fra et lærerperspektiv, i sin studie etter implementeringen av programmering i finsk skole. De aller fleste lærerne i studien ser viktigheten og verdien av et økt fokus på digitale ferdigheter som programmering. Men det er tydelig at det er behov for mer kunnskap om innholdet og mer pedagogisk bruk av digitale verktøy (Mannila, 2018).

2.2 Algoritmisk tenkning/Computational thinking

Algoritmisk tenkning består av de to ordene *algoritme* og *tenkning.* En algoritme kan sammenlignes med en matoppskrift, en strikkeoppskrift eller en regneoppskrift i matematikk. I matematikk er en algoritme en systematisk og nøyaktig beskrivelse av en fremgangsmåte for en løsning av en beregningsoppgave. Algoritmen beskriver de ulike stegene med ord, matematisk symbolikk eller skjematisk fremstilling av arbeidsgangen (Dahlum, 2018). En algoritme i et datamaskinprogram er beskrevet med hjelp av et programmeringsspråk der man gir presise sekvenser av kommandoer som tolkes av datamaskinen (Sanne et al., 2016). Vi bruker ofte algoritme synonymt med oppskrift i dagligtalen. For å kunne benytte seg av en algoritme trenger man nødvendigvis ikke å forstå det logiske fundamentet bak oppskriften, men bare evnen til å følge stegene i den (Biggs, 1990). Tenkning omfatter noe mer generelt enn tenking («det å tenke») og tankegang («en bestemt tankerekke») (Gjøvik & Torkildsen, 2019).

I faglitteraturen er det noe omdiskutert hva som er den norske oversettelsen av det engelske begrepet *Computational Thinking (CT)* (Gjøvik & Torkildsen, 2019). I denne oppgaven vil jeg bruke den norske oversettelsen «algoritmisk tenkning» (AT), fremfor «algoritmisk tankegang» som også blir benyttet. Gjøvik & Torkildsen (2019) mener algoritmisk tenkning er en mer presis oversettelse. De mener «algoritmisk tankegang» er en oversettelse av det engelske begrepet «algorithmic thinking», som kan ses på som en underkategori av AT, og som handler mer om å kunne tolke og forklare algoritmer (Gjøvik & Torkildsen, 2019).

I denne oppgaven vil den norske oversettelsen, algoritmisk tenkning (AT), av *computational thinking* brukes med forståelsen og innholdet i det engelske begrepet. AT blir beskrevet i litteraturen som en grunnleggende ferdighet, på lik linje med skriving, lesing og regning, som ikke bare er for programmerere (Wing, 2006). Under kjerneelementet *Utforskning og problemløsning* i matematikkfaget beskrives AT slik:

> «Algoritmisk tenkning er viktig i prosessen med å utvikle strategiar og framgangsmåtar for å løyse problem og inneber å bryte ned eit problem i delproblem som kan løysast systematisk. Vidare inneber det å vurdere om delproblema best kan løysast med eller utan digitale verktøy»
> (Kunnskapsdepartementet, 2019)

AT handler om å løse et problem ved å spesifisere en presis sekvens av kommandoer (Sanne et al., 2016). Videre beskriver de i rapporten at AT er å kunne abstrahere, gjennomgå informasjon systematisk, lære å lese og forstå ulike representasjonsformer, modulisere problemer og problemstillinger, og å resonnere i iterative og parallelle strukturer (Sanne et al., 2016). I rapporten fremkommer det at slike ferdigheter er særlig viktig for å kunne møte fremtidens behov (Sanne et al., 2016).

I følge Wing (2006) kan AT knyttes opp mot matematisk tenkning på bakgrunn av at datavitenskap tar utgangspunkt i og er formet med grunnlag i matematikken. AT forekommer også i matematikk på den måten at tall og andre komponenter må settes sammen i spesifikke rekkefølger for at det skal gi et gyldig resultat og gi mening. Samtidig kan det være forskjell på å tenke algoritmisk i matematikk og i programmering. I programmering vil rekkefølgen på en kode alltid tolkes i kronologisk rekkefølge, mens i matematikken er det mer rom for fleksibilitet. I tillegg er språket ulikt. I programmering består språket ofte av bokstaver, ord eller andre koder, mens det i matematikken som oftest består av tall og andre symboler.

2.2.1 Computational thinking i Internasjonal forskning

Wing (2006) sier at AT er måten mennesker løser problemer, det er ikke meningen at mennesker skal tenke som datamaskiner. I en senere artikkel kommer Wing (2017) med en mer bearbeidet beskrivelse, der AT er tankeprosessen der vi formulerer et problem og

uttrykker en løsning som en datamaskin, et menneske eller en maskin kan utføre. AT blir definert som en metode for problemløsning, som inkluderer fem kognitive prosesser;

1. **Reformulere problemet** – rekonstruere et problem til et løsbart og kjent et
2. **Rekursjon** – lage et system basert på tidligere informasjon
3. **Bryte ned problemet** – dele opp problemet til mindre og mer håndterlige deler
4. **Abstraksjon** – modellere kjernen i komplekse problemer eller systemer
5. **Systematisk testing** – ta meningsfulle handlinger for å komme frem til løsninger (Shute, Sun & Asbell-Clarke, 2017; Wing, 2006).

Lodi & Martini (2021) gjennomførte en analyse av definisjoner og rammeverk av AT de siste ti årene for å forsøke å skape en oversikt over innholdet i begrepet AT. Det Lodi & Martini (2021) fant ut av var at definisjonene inneholdt i stor grad de samme begrepene. AT er en måte å tenke på, eller en tankeprosess og en type problemløsning-strategi. En problemløsning-strategi som ofte involverer noe eksternt, for eksempel en datamaskin, som utfører oppgaven. Som i følge Lodi & Martini (2021) karakteriseres ut i fra fire hovedkategorier med underkategorier;

- **Mental processes** (Logisk tenkning; modellering; abstraksjon; algoritmisk tenkning; generalisering)
- **Methods** (datainnsamling; analyse og representasjon; programmering; automatisering; vurdere)
- **Practices** (eksperimentering; tesing og feilsøking; iterasjon; gjenbruk)
- **Transversal skills** (design; kommunikasjon og samarbeid; refleksjon; læring; meta-refleksjon)

Innholdet i begrepene har likhetstrekk med andre disipliner og er veldig generelle, derfor understreker han at innholdet i de må ses i sammenheng med *computer science* (informatikk). Hvis ikke vil det alltid være en risiko for at ideene med AT kunne virke uklare og vanskelige å skille fra andre *21st century skills* (Lodi & Martini, 2021).

2.2.2 Forholdet mellom AT, programmering og koding

Som nevnt tidligere er koding og programmering ofte brukt om hverandre for å beskrive prosessen med å skrive instruksjoner som en datamaskin skal utføre. Begrepet programmering omhandler mer enn det å skrive en kode, i en bredere forstand inkluderer det å analysere problemer, utvikle løsninger og implementere disse. Og koding er den delen der man implementerer løsningen med hjelp av et programmeringsspråk. Dette stadiet også inkluderer også feilsøking og testing.

AT og programmering er generelt sett ikke sett på som to overlappende områder. Det å tenke som en informatiker ("computer scientists") omhandler mer enn det å kunne programmere på en datamaskin (Hsu, Chang & Hung, 2018; Wing, 2006). Det er viktig å bemerke seg at koding og programmering er en viktig del av AT, men samtidig også være bevisst på at AT inneholder flere og andre elementer som; det å analysere problemer, bryte ned problemer og en tankeprosess uavhengig fra teknologi (Bocconi et al., 2018; Wing, 2006). Programmering kan fungere som et verktøy eller en aktivitet for å konkretisere og lære seg sentrale AT begreper og ferdigheter (Shute, Sun & Asbell-Clarke, 2017; Bocconi, 2018). Undervisning i programmering handler ikke utelukkende om at elevene skal lære seg å forstå datamaskiner eller få kompetanse i hvordan de bruker digitale verktøy. Det handler også om å gi elevene gode verktøy for å kunne løse oppgaver og problemer (Bråting & Kilhamn, 2021; Sevik et al, 2016).

2.3 Programmering og matematikk

Som nevnt tidligere startet interessen for å lære matematikk gjennom programmering på 70/80-tallet, hovedsakelig potensialet med programmeringsspråket Logo (Benton et al., 2016;17; Papert, 1980). I dag er det blokkbaserte programmeringsspråket Scratch et av de mest populære verktøyene, utviklet av MIT og inspirert av Logo, som er designet og utviklet til barn og unge (Resnick et al., 2009). I følge Resnick et al. (2009) legger Scratch til rette for å kunne lære sentrale matematiske og algoritmiske begreper (*Computational consepts*). I tillegg til at programmeringsspråket gir rom for øvelse i kreativitet, systematisk argumentasjon og samarbeid. Programmering i dag blir løftet mer frem som en metode eller et pedagogisk verktøy for å utvikle elevenes AT. I Norge har vi i læreplanen, i likhet med Sverige og Finland, gitt matematikkfaget hovedansvaret for å lære de grunnleggende prinsippene for programmering.

Mye av den tidligere forskningen rundt programmering i matematikk stammer for det meste fra matematikkemnet på videregående og universitetsnivå (Grover & Pea, 2013). Samtidig er det pågående og publisert forskning rundt erfaringer etter implementering av programmering i grunnskolen (Bocconi et al., 2018). Kilhamn et al. (2021) presenterer i sin studie en samling av matematikklærers erfaringer og meninger, i svensk og finsk skole, rundt programmering og matematikk. Det fremkommer at de fleste lærerne ser på programmering som et pedagogisk verktøy for å skape engasjement i matematikkundervisningen. I tillegg knytter de det til det å følge instruksjoner, aspekter innenfor logisk tenkning, og samarbeid og kommunikasjon (Kilhamn et al., 2021). Lærerne knytter i liten grad programmering opp mot spesifikke matematiske temaer.

2.3.1 Erfaringer med programmering i matematikkundervisning

I en annen studie av Kilhamn et al. (2021) undersøker de også 32 undervisningsleksjoner i matematikk der programmering er en del av undervisningen. Ut i fra analysen laget de fire ulike kategorier for matematikkundervisning med programmering; 1. *Enbart programmering* (undervisningen handler ensidig om programmering), 2. *matematik som kontext for programmering* (matematikken utgjør konteksten for programmering for å lære seg programmering. Utforsker ikke nye matematiske ideer, men programmering brukes til repitisjon), 3. *programmering som ett verktyg för att effektivisera beträkningar inom matematiken (*et tydelig matematisk innhold. Programmering for å utføre regneoperasjoner) og 4. *programmering som ett verktyg för att utforska matematik* (programmering for å utforske matematiske begrep eller sammenhenger. Få en dypere forståelse for matematikk) (Kilhamn et al., 2021).

Resultatene fra studien viser at det er kategori 1 og 2 som var mest vanlig, henholdsvis 31% og 32% av alle undervisningene. Kategori 3 og 4 fordelte seg på 18% og 13%. Innholdet av matematikk er hovedsakelig knyttet til geometri og talloppfatning. Selv om den svenske læreplanen sier at algebra skal være sentralt i sammenheng med programmering (Kilhamn et al., 2021). For at programmering skal bli sett på som en del av matematikkfaget understreker de at programmeringen må ha et matematisk innhold eller kontekst. En tolkning av resultatet av mye undervisning i kategori 1 er at siden programmering også er nytt for lærerne så er det en nødvendighet med eksklusiv undervisning i programmering. I analyse av undervisningene innunder kategori 4 retter de et kritisk blikk mot bruken av programmering som pedagogisk

verktøy for å utforske det gitte matematiske begrepet. Andre verktøy, som Geogebra, kunne gitt like gode eller bedre muligheter til utforskning. Derfor fremheves behovet for bedre kunnskap i å finne meningsfulle sammenhenger mellom matematikk og programmering (Kilhamn et al., 2021).

I arbeid med matematikk og programmering er det viktig at lærerne har god oversikt over begreper, da det ofte brukes samme begreper og symboler uten at disse alltid har samme betydning. Lærere må inneha disse kunnskapene for å ikke skape begrepsforvirring hos elevene (Grover, Pea & Cooper, 2015; Kilhamn et al., 2021). Et eksempel på dette er funksjonsbegrepet. I matematikk innebærer det en relasjon mellom to mengder, mens i programmering kan det også likestilles med en prosedyre.

2.3.2 Forsstrøm & Kaufmann (2018) – *Noen elementer fra deres artikkel om potensialet programmering har i matematikkfaget*

I artikkelen til Forsstrøm & Kaufmann (2018) diskuterer de fire ulike temaer som de mener er sentrale for å se på potensialet programmering har i matematikkundervisning. Det første de diskuterer er *programmering og motivasjon til å lære matematikk.* Etter å ha gjennomgått en del studier som ser på elevenes motivasjon i møtet med programmering og matematikk, mer spesifikt programmering med roboter (LEGO mindstorms), så de ikke noen tydelige endringer i elevenes motivasjon før og etter perioden med programmering med roboter. De påpeker at elevgruppen som ble undersøkt hadde generell høy motivasjon for faget. De legger til at perioden med programmering var kort og trekker frem at studiene ikke kan generaliseres, men at de har interessante funn. Det andre temaet de tar for seg er *elevers prestasjon i matematikk.* Her ser de nærmere på fem kvalitative studier som undersøker elevers prestasjoner etter en periode med programmering. Alle studiene trakk frem noe positivt om resultatene etter perioden, selv om det ikke i alle tilfellene var bedring på testene. En av studiene så spesielt på utvikling i elevenes forståelse for omkrets og areal, samt økning i resultater i problemløsning og logisk tenkning. Resultatene her er heller ikke generaliserbare og studiene var ikke godt egnet til sammenligning da de fokuserte på ulike ferdigheter og egenskaper. Forbedringene var bare synlige hos enkelte grupper.

I gjennomgangen av det tredje og fjerde tema, *samarbeid mellom elever og rollen til læreren,* fant de ut at samarbeid var en svært vanlig arbeidsmetode. Lærerens rolle var å være en

veileder og gi støtte fremfor å være en underviser, noe som la til rette for samarbeid i grupper. Læreren opptrer som en konfliktløser i klasserommet, og må være til stede og løse konflikter når de oppstår, for eksempel når elevene møter utfordringer som gjør at samarbeidet ikke fungerer. Klasseromskulturen er viktig for denne typen samarbeidsbaserte undervisningen. Der det er viktig at elevene respekterer andres synspunkter og lytter til hverandre, dette vil styrke samarbeidet. Et slikt fritt miljø i klasserommet gir elevene muligheter til å innta andre roller enn det som er standard i ordinære matematikktimer. Programmering kan gi elever, som normalt er sett på som lavt-presterende, mulighet til å lede gruppene og utforske mer sofistikerte matematiske ideer. Slik lærerens rolle i klasserommet påvirker elevenes læring, vil samarbeidet mellom elevene i programmeringsaktiviteter påvirke deres læring. Gjennom samarbeid og kunnskapsdeling gis elevene mulighet til å lære av hverandre, men dette forutsetter nødvendigvis at oppgavene legger til rette for samarbeid. Elevene deler deres kunnskap i og på tvers av gruppene, og kunnskapsdelingen kan påvirke elevenes valg i deres problemløsningsstrategier.

I konklusjonen trekker de frem ansvaret skolen har for å utdanne og forme elevene til å bli borgere som kan støtte og utvikle samfunnet. Skolen derfor må tilpasse seg og inkludere ny teknologi som påvirker samfunnet. De understreker at det trengs mer kunnskap om hvordan man skal inkludere programmering i skolen, og hvilke arbeidsmetoder som kan bidra til lære matematikk gjennom programmering (Forsstrøm & Kaufmann, 2018). Studiene viste at programmering kan gi bedre prestasjoner og høyere motivasjon i matematikkfaget, men på den andre siden er generaliserbarheten til studiene begrenset. Avslutningsvis sier Forsstrøm & Kaufmann (2018) at det er liten tvil om at programmering er viktig og vil bli viktigere i fremtiden, men at det er behov for tydeligere retningslinjer for hvilken rolle programmering skal ha i skolen og i matematikkfaget.

2.4 Lærebøkenes rolle i matematikkfaget

Lærebøker i matematikk som et pedagogisk verktøy for undervisning og læring i matematikk har lang tradisjon. I denne sammenheng mener jeg bøkene som brukes til undervisning og læring av matematikk, som er tradisjonelle fysiske-klassetrinns-spesifikke bøker. Forskning rundt lærebøkene i matematikk derimot har ikke en lang historie (Fan et al., 2013). Forskning på tema har ikke før de siste tre tiårene fått økt oppmerksomhet og vekst (Fan, 2013). Samtidig er forskningsfeltet i en tidlig fase i utviklingen, og det filosofiske grunnlaget,

teoretiske rammeverkene og forskningsmetodene er underutviklet (Fan et al., 2013; Kongelf, 2019). Forskning på lærebøker faller innunder det store forskningsfeltet matematikkdidaktikk (Kongelf, 2019).

Lærebøker har fortsatt en sentral rolle i moderne skole, og forskere er generelt samstemte om at lærebøker er en av hovedformidlerne læreplanen (Fan et al., 2013). Ifølge Fan et al. (2013) brukes lærebøker i matematikkfaget mer sammenlignet med lærebøker i andre fag, og har større påvirkning på undervisningen. I tillegg kan lærebøkene ha en innvirkning på lærerstilen og undervisningsmetodene som blir brukt i klasserommet. Det kan dermed ha en innvirkning på lærerens pedagogikk, og på elevenes læring i matematikk (Fan, 2013; Fan et al., 2013; Kongelf, 2019).

Forskning på 2000-tallet viser at matematikkundervisningen i stor grad er organisert rundt elevenes oppgaveløsning (Hiebert et al., 2003; Shimizu et al., 2010). Ser vi nærmere på norsk matematikkundervisning ser vi at norske elever arbeider relativt mye med oppgaver fra lærebøker sett opp mot det internasjonale gjennomsnittet (Grønmo et al., 2004; Grønmo & Onstad, 2009; Klette, 2003). I studien til Lepik et al. (2015) presenterer de funn som viser at 69% av norske lærere bruker oppgaver fra læreboken som eneste kilde i hver eller annenhver undervisningstime.

Oppgaven i lærebøkene inneholder tre faser. Det første går ut på hvordan oppgaven fremstår slik forfatteren har skrevet den. Det andre er hvordan læreren bruker oppgaven i klasserommet, og den tredje er elevens bruk og tolkning (Kongelf, 2019). Disse tre fasene menes å ha en innvirkning på elevenes læringsprosess.

I min studie tar jeg for meg oppgavene som er i selve læreboken, og er avgrenset til det jeg tolker at forfatteren har planlagt at elevene, og til dels lærerne skal gjøre i undervisning og læring i programmering. Til dels lærerne, da lærerne kan velge å justere enkelte oppgaver eller velge ut oppgaver som kan utvides å jobbes videre med. Den typiske strukturen i bøkene er at oppgavene er jevnt fordelt i kapitlene med forklaringer eller presentasjoner av idéer eller løsningsmetoder i starten og underveis i kapitlene. Oppgavene er hovedsakelig nummererte, mens noen gjerne større oppgaver er presentert som "oppdrag", "aktivitet" eller "utforsk". Det varierer mellom bøkene om programmeringsoppgavene utelukkende er plassert i et underkapittel, om de er spredt utover, eller at det er en kombinasjon av underkapitler og

fordeling. I min studie velger jeg ikke ut eller skiller mellom oppgavetypene, alle oppgaver som har et innhold knyttet til programmering er tatt med i analysen. Det som er interessant er innholdet i oppgavene og hvordan dette blir presentert.

2.5 Tidligere forskning

Lærebøkene som er inkludert i denne analysen er utgitt i tilknytning til den nye læreplanen som trådte i kraft høsten 2020. Programmering i tilknytning til matematikkfaget i norsk skole er nytt for de fleste lærere i dagens skole, til tross for at idéen om programmering i matematikk stammer fra 1970-tallet (Papert, 1980). Dette medfører at det er lite tidligere forskning på programmering i matematikklærebøker, spesielt i grunnskolen (Grover & Pea, 2013). I studien til Bråting & Kilhamn (2021) gjennomførte de en analyse av programmeringsinnholdet i svenske lærebøker i matematikk på barnetrinnet, etter implementeringen av programmering i den svenske læreplanen i 2018. Der de ønsket å se på hva som karakteriserte programmeringsinnholdet i lærebøkene og hvordan programmeringsoppgavene la til rette for å skape bro mellom programmering og matematikk. Som nevnt tidligere designet de et eget analyseverktøy som baserer seg på Brennan & Resnick (2012) sitt rammeverk for *computational concepts* og Benton et al. (2016) sitt rammeverk the 5E´s, som jeg også vil benytte meg av i dette prosjektet.

Resultatene fra analysen av de svenske lærebøkene på barnetrinnet viste at noe av det mest karakteristiske med oppgavene var den hyppige bruken av stegvise instruksjoner. I tillegg til at den mest vanlige handlingen i oppgavene var det å følge en prosedyre. De trekker frem at lærebøkene ser ut til å fokusere mye på enkelte begreper, mens de omtrent utelukker andre begreper. *Condition (betingelser), debugging (feilsøke) og rule (regel)* er blant de computational concepts (AT-begrepene) som blir lite inkludert i programmeringsoppgavene. Videre konkluderer Bråting & Kilhamn (2021) at programmeringsinnholdet i liten grad lykkes i å skape bro mellom programmering og matematikk, ved at oppgavene i liten grad gir mulighet til å utforske matematiske begreper og idéer.

3 Metode

Analysen i denne studien undersøker hva som karakteriserer programmeringsinnholdet i et utvalg av lærebøker i matematikk, og fokuserer hovedsakelig på innholdet i oppgavene i bøkene. I dette kapittelet vil jeg presentere hvilket analyseverktøy jeg har brukt i analysen, og begrunne hvorfor jeg har valgt dette analyseverktøyet. Videre vil jeg gi en presentasjon av hvilke lærebøker som er valgt ut i studien, hvorfor disse er valgt ut og hvorfor enkelte bøker ikke er inkludert i analysen. Til slutt beskriver jeg validitet og reliabilitet knyttet til studien.

3.1 Innholdsanalyse av lærebøker

Analysen ser nærmere på innholdet som kan knyttes til programmering i norske læreverk på mellom- og ungdomstrinnet, dette ved hjelp av et analyseverktøy hentet fra studien til Bråting & Kilhamn (2021). Utvalget som ligger til grunn for analysen er utgitt av tre ulike forlag, de ulike forlagene er representert med ulikt antall læreverk. Utgangspunktet for studien er ikke å sammenligne de ulike læreverkene, men å undersøke hva som karakteriserer programmeringsinnholdet elevene møter i norske matematikklærebøker, i tillegg til å se på hvordan dette innholdet legger til rette for å skape bro mellom programmering og matematikken. Lærebøkene som er inkludert i analysen ses på som dokumenter. For å kunne besvare forskningsspørsmålene i denne studien har jeg gjennomført en innholdsanalyse av disse dokumentene (Christoffersen & Johannesen, 2012). Programmeringsinnholdet i lærebøkene vil bli bearbeidet og kategorisert ved hjelp av analyseverktøyet for å skape et datagrunnlag for analysen. Metoden karakteriseres som en innholdsanalyse da fokuset er på dokumentenes innhold og kategorisering av innholdet (Grønmo, 2016).

Det finnes ulike typer lærebokanalyse, enkelte studier ønsker å undersøke hvordan lærebøkene benyttes i skolen, og andre studier retter fokuset mot selve læreboken (Rezat & Strässer, 2017). I denne studien ser jeg utelukkende på bøkenes innhold, og analysen ses da på som en summativ innholdsanalyse ved at fokuset er på hvor ofte og i hvilken betydning ord og innhold forekommer (Fauskanger & Mosvold, 2015). I sammenheng med forskningsspørsmålene for denne studien vil jeg se nærmere på innholdet i bøkene som knyttes til programmering. Det er i hovedsak oppgavene som er knyttet til programmering som skal analyseres. Tekst, bilder, aktiviteter eller illustrasjoner før eller etter oppgavene som kan knyttes til oppgaven vil også bli undersøkt. Oppgavene vil bli kategorisert med hjelp av analyseverktøyet til Bråting & Kilhamn (2021). Hensikten med en slik kategorisering er for å

få en oversikt over hva som kjennetegner programmeringsinnholdet i lærebøkene, samt undersøke i hvilken grad oppgavene bidrar til å skape sammenhenger mellom programmering og matematikk. Kategoriseringen og registreringen i min studie er hovedsakelig kvalitativ, men inkluderer noen kvantitative elementer. Kvalitativ forskning trenger nødvendigvis ikke være svarthvitt, og forskningen kan inneholde elementer av både kvantitativ og kvalitativ metode i samme undersøkelse (Christoffersen & Johannesen, 2012; Grønmo, 2016). Det kvalitative i min analyse blir tydeligere ved at jeg selv tolker oppgavene, illustrasjoner og annen tekst i lærebøkene. Min rolle som forsker, samt mine valg og mine tolkninger, vil påvirke datainnsamlingen og den videre analysen (Krumsvik et al., 2019).

Datamaterialet i denne analysen er hovedsakelig skriftlige lærebøker, dette gir en mulighet til å se på konkrete temaer presentert i bøkene. Samtidig er det viktig å bemerke seg at innholdsanalyser av lærebøker gir en begrenset forståelse av kontekst. Det gir et begrenset innblikk i interaksjonen mellom elev og lærer og læringsituasjonen til elevene (Cohen, Manion & Morrison, 2011). Samtidig kan analyse av lærebøker gi et innblikk i elevenes læringsprosess da det er tradisjonelt sett et viktig verktøy i planlegging og gjennomføring av undervisning (Johansson, 2006; Kongelf, 2019). Oppgavene som er inkludert i analysen er oppgaver som er merket som programmeringsoppgaver, oppgaver der elevene får valg om å bruke programmering som løsningsmetode og oppgaver som innebærer begreper innenfor programmering.

3.2 Utvalg

Jeg har valgt ut elleve lærebøker i matematikk som brukes i matematikkundervisning fra 5-10.trinn. Matematikklærebøkene som analyseres i denne studien er *Multi 5B, Multi 6B, Matematikk 5, Matematikk 6, Matemagisk 5B, Maximum 8, Maximum 9, Maximum 10, Matemagisk 8, Matemagisk 9 og Matemagisk 10.* Alle bøkene som analyseres er laget etter Fagfornyelsen (2020), og inkluderer programmering. Det er en del lærebøker som har lanseringsdato våren 2022 som ikke blir en del av analysen, og enkelte bøker jeg ikke har hatt tilgang til. Dette gjelder spesielt lærebøkene for 7.trinn, legg merke til at det ikke er inkludert bøker fra 7.trinn. I noen bokserier er det to bøker pr skoleår, her har jeg inkludert de bøkene som inneholder programmering.

Enkelte av bøkene som er analysert, *Matematikk*-serien fra Cappelen Damm, har hoveddelen av programmeringsinnholdet digitalt. *Matematikk 5 og 6* fra Cappelen Damm inneholder hvert sitt delkapittel om programmering og er derfor analysert. Det digitale innholdet til Cappelen Damm 5-10 om programmering innunder matematikk er inkludert i analysen, men på grunn av tidsrammen er ikke det digitale innholdet til de andre bøkene tatt med i analysen. Jeg har valgt å gjøre dette unntaket for å kunne inkludere bokserier fra flere forlag. I tillegg til digitale ekstramateriale har også noen av lærebøkene andre tilleggsmateriale som lærerveiledning og oppgavebøker. Disse blir ikke inkludert i oppgaven grunnet tid og relevansen til forskningsspørsmålene, selv om det ville vært interessant å sett på lærerveiledningene i sammenheng med forskningsspørsmål 2.

3.2.1 Multi og Maximum

Multi og Maximum er en lærebokserie med bøker fra 5-10 trinn, gitt ut av Gyldendal forlag. Jeg har valgt ut de bøkene som inneholder programmering, bøkene med lanseringsdato i 2022 er ikke inkludert i analysen. Bøkene som er analysert fra denne serien er Multi 5B, Multi 6B, Maximum 8, Maximum 9 og Maximum 10. Alle lærebøkene, utenom Maximum 9 og 10, egne kapitler eller underkapitler om programmering. I de to nevnte er programmeringsinnholdet spredt mellom flere temaer.

3.2.2 *Matemagisk 5-10*

Matemagisk er en lærebokserie fra Aschehoug Univers. Alle bøkene har egne kapitler eller underkapitler om programmering. I tillegg har bøkene på ungdomstrinnet oppgaver om programmering i tilknytting til flere kapitler. Bøkene fra denne serien som er analysert er Matemagisk 5B, Matemagisk 6B, Matemagisk 8, Matemagisk 9 og Matemagisk 10.

3.2.3 *Matematikk 5-10*

Matematikk 5-10 er en serie lærebøker gitt ut av Cappelen Damm. I analysen har jeg inkludert bøkene for 5 og 6 trinn, i tillegg til digitale ressurser om programmering som er rettet mot 5-10. trinn. Det var svært lite programmeringsinnhold i de trykte lærebøkene. Jeg tok derfor kontakt med Cappelen Damm, de begrunnet lite programmeringsinnhold i de trykte læremidlene med at de mener at måten programmering undervises i skolen vil endres mye i løpet av lærebøkenes brukstid (H.M. Bjørklund, personlig kommunikasjon, 19. april 2022). Jeg fikk også tilsendt et oppgavehefte som er inkludert i analysen, dette ligger tilgjengelig for

allmennheten under navnet; «Utdrag fra hefte om programmering i matematikk 8-10 Digital lærerressurs».

3.3 Presentasjon av analyseverktøyet

Som nevnt innledningsvis i oppgaven ønsker jeg å gjennomføre omtrent samme analyse som Bråting & Kilhamn (2021), bare på norske læreverk i matematikk. I valget av analyseverktøy har jeg hentet inspirasjon fra en svensk studie av Bråting & Kilhamn (2021) som analyserer programmeringsinnholdet i svenske lærebøker i matematikk. I deres studie forsøkte de to ulike rammeverk for å analysere programmeringsinnholdet i lærebøkene. Det ene var Brennan & Resnick (2012), to av personene bak Scratch miljøet, sitt rammeverk som beskriver bruken av *computational consepts.* Det andre var Benton et al. (2016) sitt rammeverk «the 5E´s». Bråting & Kilhamn (2021) endte opp med å designe sitt eget analyseverktøy, som de beskriver som en kombinasjon av de to øvrige. I avsnittene under vil jeg gi en kort presentasjon av rammeverkene til Brennan & Resnick (2012) og Benton et al. (2016), før jeg presenterer Bråting & Kilhamns (2021).

3.3.1 Computational concepts

Rammeverket til Brennan & Resnick (2012) bestod i utgangspunktet av begrepene: *sequences, loops, parallelism, events, conditionals, operators and data*. Brennan & Resnick (2012) syntes ikke at dette beskrev elevenes læring i stor nok grad og det ble derfor lagt til fire *conceptual practices* for å styrke rammerverket: (i) *being incremental and iterative*, (ii) *testing and debugging*, (iii) *reusing and remixing*, og (iv) *abstracting and modularizing*. I tillegg ble en tredje dimensjon lagt til som gikk ut på å beskrive elevenes endring av perspektiver underveis i arbeid med Scratch. Bråting & Kilhamn (2021) inkluderte ikke denne dimensjonen da det ville være vanskelig å få tilgang til i en lærebokanalyse.

3.3.2 The 5E´s

Etter at Storbritannia la til et nytt skolefag, kallet Computing, ville forskere se på hvordan arbeid med programmering i Scratch kunne bygge opp elevenes kunnskap i matematikk. Forskere, sammen med lærere, utviklet et rammeverk som omtales som «the 5E´s» (Benton et al., 2016; 2017). Målet med rammeverket var å gi pedagogiske verktøy for å kunne designe programmeringsaktiviteter i matematikk fra et konstruktivistisk perspektiv. Rammeverket består av de fem punktene: *Explore, Explain, Envisage, Exchange og BridgE:*

- **Explore** (**utforske**): Å utforske både verktøyet i seg selv og ideene som er presentert i aktiviteten. Noe som tillater elevene å prøve ut nye ting, jobbe interaktivt og se på feilmeldinger/feilsøke.
- **Explain** (**forklare**): Å forklare og representere kunnskap på flere ulike måter. Forklare et script og hva det skal gjøre, samtidig som man reflekterer over egen tenkning. I Scratch programmering så vil det å definere og navngi blokker være en viktig del av det å forklare.
- **Envisage** (**forutse**): Det å forutse mulige utfall, å se for seg hva som vil skje før det skjer og starte refleksjonen før man prøver det på datamaskinen.
- **Exchange** (**dele**): Det å forklare egne strategier, stille spørsmål til andre og løse uenigheter i par/gruppe arbeid, samarbeide og helklasse diskusjoner for å fremheve kreativitet og utforskning.
- **BridgE** (**brobygging**): Det å finne koblinger mellom programmering og matematiske ideer. For å kunne utvikle/oppdage disse koblingene må ideer bli satt i en ny kontekst og bygget opp på ny med et matematisk språk.

3.3.3 Bråting & Kilhamns (2021) analyseverktøy

Det er både likheter og ulikheter i de to nevnte rammeverkene nevnt ovenfor; *computational concepts* utviklet av Brennan & Resnick (2012) og «the 5E´s» utviklet av Benton et al. (2016; 2017). Begge tar utgangspunkt i Scratch. Man ser tydelige likheter mellom den andre dimensjonen i CT-rammeverket og punktet om Explore/utforske i de 5E´s. Noen av de *computational consepts* (AT-begreper) som vi finner i CT-rammeverket finner vi ikke igjen i de 5E´s, som fokuserer mer på elevenes handlinger som å forklare, dele og forutse. Den største forskjellen mellom rammeverkene er punktet om bridgE i de 5E´s, som plasserer programmeringsaktiviteter tydelig i en matematisk kontekst. Matematiske ideer må bli implementert i aktivitetene og uttrykt med matematiske begreper og notasjoner, hvis matematisk læring skal være til stede.

I den svenske studien trakk de frem at de søkte et rammeverk for å analysere programmeringsinnholdet i matematikk lærebøker som tok for seg både sentrale ideer innenfor AT og som inkluderte matematikk eksplisitt. Etter å ha forsøkt å benytte seg av de to rammeverkene beskrevet tidligere hver for seg, kom de frem til at ingen av dem kunne beskrive innholdet i lærebøkene på en god nok måte. Derfor tok de inspirasjon fra begge rammeverkene og konstruerte deres eget analyseverktøy, som er en kombinasjon av begge rammeverkene. Under vil jeg beskrive dette sammenslåtte analytiske verktøyet som består av

handlinger (actions) og *begreper* (consepts), og som i tillegg diskuterer forholdet mellom matematikk og programmering som en overordnet idé.

3.3.3.1 Handlinger

Som et resultat av sammenslåingen av rammeverkene til Brennan & Resnick (2012) og Benton et al. (2016) kom Bråting & Kilhamn (2021) frem til følgende handlinger;

a. **Følge** – følge stegvise instruksjoner, repetere eller fortsette på et mønster.
b. **Finne regel** – Jobbe gjennom en prosedyre, finne en regel eller et mønster som generer et utfall, for eksempel et tallmønster
c. **Feilsøke** – feilsøke en kode
d. **Forme og skape** – gi instruksjoner, lage mønster, skrive koder, representere med symboler
e. **Forklare** – forklare med et naturlig språk, bruke ord til å beskrive fremgangsmåten, en regel eller et begrep
f. **Forestille seg** – forutse hva som vil skje, reflektere over mulige utfall når betingelser eller verdier endres

I Bråting & Kilhamn´s (2021) handlinger finner vi igjen elementer fra rammeverket til Brennan & Resnick (2012) med feilsøking/debugging (c) og det å være *incremental* og *iterative* (b,d) og punkter fra the 5E´s explore (b, c), *explain* (e) og *envisage* (f). Den første (a), *følge*, er lagt til av Bråting & Kilhamn (2021).

3.3.3.2 Begreper

Analyseverktøyet ser som sagt på innholdet i oppgavene gjennom å se på hvilke matematiske begreper og hvilke AT-begreper oppgavene inneholder. Det kan i en oppgave registreres flere begreper, og det samme gjelder for handlingene. I analysen av begrepene i oppgavene bruker jeg en beskrivende analyse hvor begrepene identifiseres og klassifiseres. Når jeg identifiserer begrepene ser jeg nærmere på ordene og meningen disse ordene har i den gitte konteksten. I analysen vil oppgavene kategoriseres ut i fra følgende begreper innenfor AT;

- **Steg-for-steg instruksjoner** – Følge stegvise instruksjoner for å lage et dataprogram, eller følge stegene i en gitt eksempel oppgave.
- **Algoritme** – En algoritme er en presis beskrivelse av en endelig serie operasjoner som skal utføres for å løse et problem eller et sett med flere problemer.

- **Løkke** – løkke brukes for handlinger som skal gjentas flere ganger (for-løkker), enten et gitt antall eller gjenta til noe er sant (while-løkke).
- **Regel** – I likhet med handlingen *finne regel*, handler dette om å finne en regel eller et gitt mønster som genererer et utfall.
- **Kode** – programkode, ofte omtalt kun som kode, er de ulike bestanddelene i et dataprogram. En kode kan for eksempel en «turn-blokk» i blokk programmering.
- **Betingelser/vilkår** – en betingelse sjekker om noe er sant eller usant eller formuleres som er ja/nei spørsmål.
- **Feilsøking** – Feilsøking eller debugging er prosessen der man identifiserer og retter opp i eventuelle feil i et dataprogram.

Og kategoriseres ut i fra følgende matematiske begreper;

- **Mønster** (inkludert tallmønster) – utforske og finne egenskaper ved ulike mønstre, og fortsette på disse.
- **Geometriske begreper** – oppgaver som inkluderer geometriske begreper og idéer, for eksempel kjennetegn ved geometriske figurer, eller utregning av areal, omkrets og volum.
- **Aritmetiske begreper** – oppgaver der én eller flere av de fire regneartene benyttes i oppgavene, og oppgaver der elevene får øving og kunnskap om tallegenskaper.
- **Rotasjon** – rotering av geometriske figurer, speiling av figurer og forskyving.
- **Koordinatplan/funksjoner** – oppgaver som involverer koordinatplan eller funksjoner/funksjonsutrykk.
- **Statistikk og sannsynlighet** – oppgaver som involverer statistikk og sannsynlighet.

Der kategorien statistikk og sannsynlighet er lagt til av meg, grunnet relevansen til kompetansemålene etter syvende og niende trinn. Analyseverktøyet klassifiserer et begrep som matematisk dersom det er i tradisjonell skole-matematikk og formidler en viktig matematisk idé. En matematisk idé kan være vanskelig å definere og i denne sammenheng er det uttrykt gjennom de matematiske begrepene og representasjonene som er benyttet. Matematiske begrep er enklere å definere og er knyttet til de spesifikke ordene og representasjonene som er brukt i oppgavene. I denne analysen vil de matematiske begrepene som fremkommer eksplisitt i oppgavene først bli registrert for å så bli kategorisert. Et eksempel kan være en oppgave som handler om å tegne et kvadrat, inkludert instruksjoner om

å repetere fire ganger og snu 90 grader. Siden et matematiske begrep er brukt for å beskrive og konstruere et geometrisk objekt, vil oppgaven bli klassifisert under geometriske begreper. I analyseverktøyet har jeg valgt å legge til statistikk og sannsynlighet som et eget begrep, da det i læreplanen er egne kompetansemål som knytter statistikk og sannsynlighet til programmering.

Et begrep er klassifisert som AT hvis det er et begrep som fremhever en spesifikk mening innenfor den presenterte programmeringskonteksten. Eksempler på slike begreper kan være for eksempel *kode, løkker, loop, variabler, bug/feil/feilsøking og vilkår/betingelser*. Noen begreper som for eksempel algoritme, kan ha ulik mening i en matematisk og programmeringskontekst, og vil derfor behandles noe annerledes. En algoritme vil i analysen bli kategorisert som et AT-begrep hvis det er definert og/eller referert som en algoritme for å oppnå et spesifikt mål, uten å kunne brukes til å løse generelle matematiske problemer. I noen tilfeller vil denne definisjonen sammenfatte med en matematisk algoritme, som subtraksjonsalgoritmen, men disse tilfellene vil ikke bli inkludert i analysen. I tillegg vil naturligvis matematiske algoritmer være nevnt andre steder i lærebøkene uten å ha være merket som programmering og vil heller ikke være en del av denne analysen.

3.3.3.3 Brobygging mellom matematikk og programmering

I tillegg til å se på handlinger og begreper i analysen av oppgavene, vil jeg også se på hvordan kombinasjonen av disse gir tydelige linker mellom programmering og matematikk. Samt diskutere hvordan disse sammenhengene kan føre til læring i matematikk. Jeg ønsker å se om elevene blir spurt om å reformulere AT-ideer ved at de bruker matematiske begreper eller tradisjonell matematisk notasjon. Og om oppgavene tillater elevene å utforske kjente eller ukjente matematiske ideer på en ny måte. Det er viktig å påpeke at når jeg undersøker lærebøker så vil ikke lærerens rolle komme tydelig frem. Læreren kan fremheve sammenhenger som ikke er eksplisitt nevnt i lærebøkene. Jeg har i denne oppgaven ikke sett på lærerveiledningene til bøkene, delvis grunnet tid og at ikke alle lærerveiledningene er lansert.

3.4 Validitet og Reliabilitet

I forskning handler validitet og relabilitet, i grove trekk, om å se på hvor gyldig og pålitelig forskningen og dens resultater er (Bratberg, 2021). I dette delkapittelet vil jeg beskrive validiteten og reliabiliteten i denne studien.

3.4.1 Validitet

I forskning som benytter seg av kvalitative metoder bruker forskeren seg selv i stor grad i innhentingen av data (Cohen et al., 2011). I sammenheng med validitet vil forskerens forventninger, fordommer og forutinntatte oppfatninger kunne ha en innvirkning på forskningen, og dermed også validiteten (Cohen et al., 2011). Framgangsmåten og funnene skal i størst mulig grad reflektere formålet med studien og gi en representasjon av virkeligheten. Validitet handler om i hvor stor grad forskeren oppnår dette (Johannessen, Tufte & Christoffersen, 2010).

Overførbarhet, troverdighet, pålitelighet og bekreftbarhet er fire viktige aspekt for validiteten i kvalitativ forskning (Choen et al., 2011). Overførbarhet handler om forskningens resultater kan overføres til et lignende fenomen. Om forskningen kan bekreftes gjennom en tilsvarende undersøkelse av en annen forsker er forskningen bekreftbar. Troverdighet, beskrevet ovenfor, handler om at studiens formål skal gi god representasjon av virkeligheten (Choen et al., 2011; Johannessen et al., 2010) Pålitelighet vil bli mer beskrevet under relabilitet.

Min studie tar for seg innholdet av programmering i lærebøker i matematikk. Her vil mine forkunnskaper og min forståelse av hva programmering innebærer kunne prege analysen. For å minimere denne faktoren har jeg i forkant av analysen redegjort og satt meg inn i relevant teori for å få et utgangspunkt og noen holdepunkter for analysen. I tillegg til at jeg benytter meg av et analytisk verktøy for å analysere innholdet i lærebøkene. Analyseverktøyet og dens definerte kategorier for oppgavene som skal analyseres gjør at min forståelse blir mindre tilstedeværende i analysen. Det at det er en analyse av lærebøker, som er permanente trykte bøker, gjør det mulig for andre forskere å kunne etterprøve mine funn gjennom en ny undersøkelse. Samtidig kan jeg ikke påstå at mitt utvalg av matematikklærebøker vil være representative. Resultatene vil først å fremst gjelde de bøkene og kapitlene som er undersøkt. Funnene kan dermed ikke generaliseres, men dette betyr nødvendigvis ikke at det vil påvirke validiteten. I følge Kvale & Brinkmann (2009) er det ikke krav om at all forskning nødvendigvis må være universell og gyldig for alle mennesker eller alle lærebøker til enhver tid. Som nevnt tidligere er ikke læreverkenes digitale ressurser (med unntak av Cappelen Damm) og lærerveiledningene inkludert i analysen, derfor vil ikke analysen kunne gi et helhetlig bilde på elevenes læringssituasjon i møte med oppgavene.

3.4.2 Relabilitet

Relabilitet kan ses på som et paraplybegrep for pålitelighet, kontinuitet og replikerbarhet (Cohen et al., 2011). I grove trekk handler det om troverdigheten og påliteligheten til resultatene i forskningen. I tillegg til replikerbarheten som handler om i hvilken grad ny lik forskning vil få like resultater, dette vil spille inn på relabiliteten. Relabilitet handler om hvilke data som brukes, hvordan de samles inn og hvordan dataen bearbeides (Cohen et al., 2011; Johannessen et al., 2010). I kvantitativ forskning er relabilitet kritisk og kan testes på ulike måter (Johannessen et al., 2010). Mens i den kvalitative forskningen er det flere faktorer som spiller inn som tid og sted, som gjør det utfordrende å gjenskape de samme dataene i en lik studie. Dette er en del av premissene til naturalistiske studier som undersøker unike situasjoner, og ses på som en styrke i stedet for en svakhet. Kvalitativ forskning kan være pålitelig selv om dataen ikke kan repliseres på samme måte som i kvantitativ forskning. For å styrke relabiliteten i kvalitativ forskning må forskeren gi leseren et innblikk i alle deler av prosessen, gjennom en beskrivelse av konteksten, og en åpen og detaljert forklaring av framgangsmåten i hele forskningsprosessen (Cohen et al., 2011; Johannessen et al., 2010).

I denne studien gir jeg en tydelig oversikt over hvor jeg har hentet dataen min, altså hvilke lærebøker som er analysert. Analyseverktøyet jeg benytter meg av er grundig beskrevet og det er gitt begrunnelse for hvorfor jeg har valgt dette analyseverktøyet. Selve analyseverktøyet tar utgangspunkt i to anerkjente rammeverk og er blitt brukt tidligere. Det har tydelige kategorier og særtrekk for å identifisere ulike oppgavetyper, noe som minimerer risikoen for at egne meninger og betraktninger har innvirkning. Samtidig kan det ikke utelukkes at mine tolkninger og meninger vil ha innvirkning på analysen. Analyse av lærebøker har gjerne høyere grad av reliabilitet da det er skriftlig datamateriale som kan undersøkes flere ganger og som ikke endres over tid.

4 Resultater

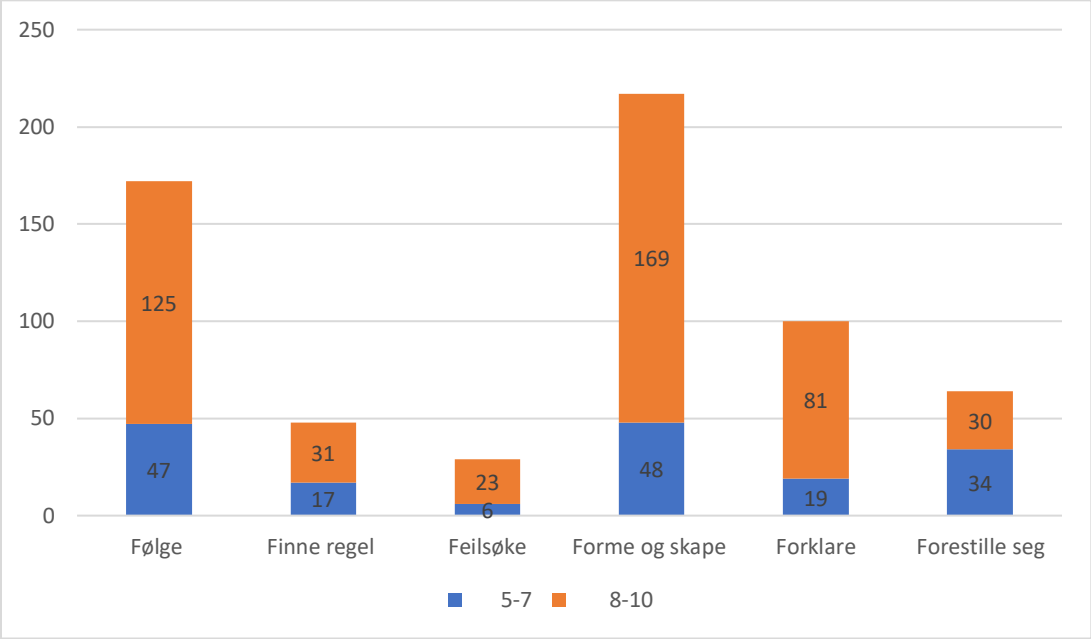
Hensikten med analysen i denne studien er som sagt å se på hva som karakteriserer programmeringsinnholdet i et utvalg lærebøker i matematikk på mellom- og ungdomstrinnet. Der fokuset er å analysere oppgavene i bøkene med hjelp av analyseverktøyet som ble presentert tidligere. I tillegg vil jeg i prosessen se på hvordan programmeringsoppgavene bidrar til læring i matematikk, det de på engelsk omtaler som «*bridging*». Denne studien har ikke som hensikt å sammenligne lærebøkene i noen utstrakt grad, men heller se nærmere på hva elvene i norsk skole vil møte av innhold knyttet til programmering i matematikkfaget.

Dette kapittelet består av tre deler. Først kommer jeg til å presentere resultatene som handler om hvilke handlinger elevene møter eller må gjennomføre i oppgavene. Deretter vil jeg vise til resultatene for hvilke begreper innen for AT og matematikk som er inkludert i oppgavene. Til slutt tar jeg for meg forholdet mellom programmeringsoppgavene og matematikk. Underveis vil det vises til tabeller og utvalgte eksempeloppgaver fra læreverkene.

Før jeg presenterer resultatene vil jeg skrive noe om strukturen i lærebøkene, og gi et bilde på hvordan de ulike lærebøkene har inkludert programmering. Bøkene på mellomtrinnet er relativt like ved at de har et eller flere underkapittel om programmering, som regel innunder kapittel om algebra eller geometri. Bøkene på ungdomstrinnet har i tillegg til egne underkapittel også programmeringsoppgaver spredt rundt i andre kapitler. Mengden av programmeringsinnhold er noe varierende, alt fra 2-30 sider i de analyserte bøkene. Lærebøkene av Cappelen Damm har valgt å ikke plassere hovedinnholdet om programmering i de trykte læringsmidlene, da de tror og mener at måten vi underviser i programmering vil endre seg mye i løpet av bøkenes brukstid (H.M. Bjørklund, personlig kommunikasjon, 19. april 2022).

4.1 Handlinger

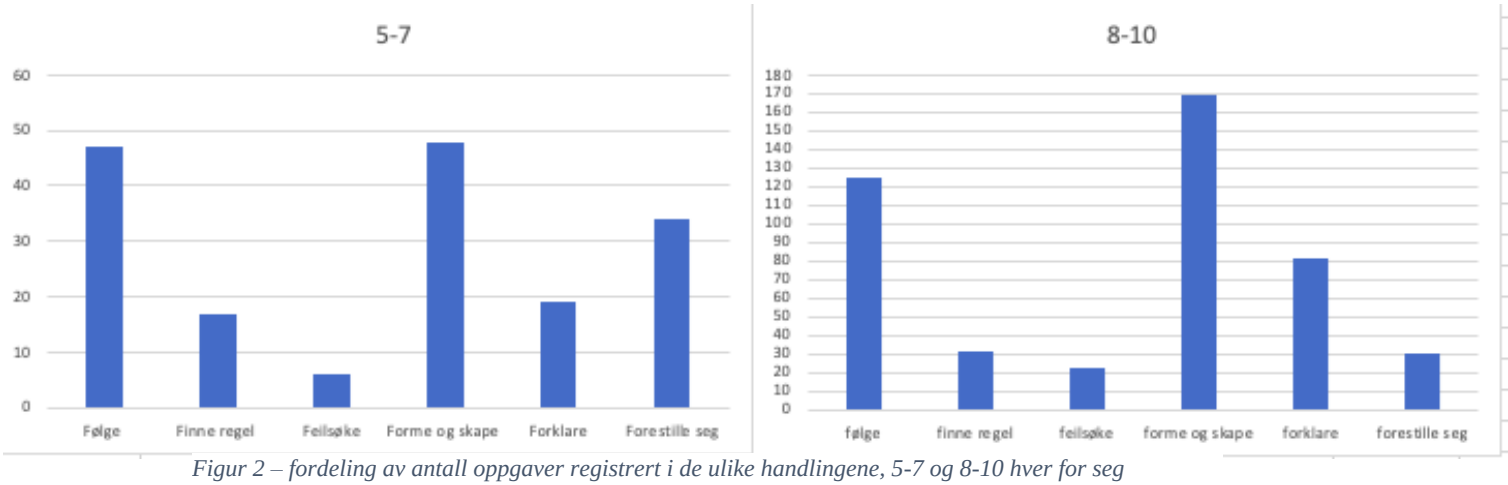
Totalt identifiserte jeg programmeringsinnhold i 276 oppgaver (79 i 5-7.trinn og 197 på 8-10.trinn). I Figur 1 ser vi fordelingen av de seks ulike handlingene i oppgavene, en oppgave kan inneholde flere handlinger. Blå representerer 5-7.trinn og oransje 8-10.trinn.



Figur 1 – fordeling av antall oppgaver registrert i de ulike handlingene

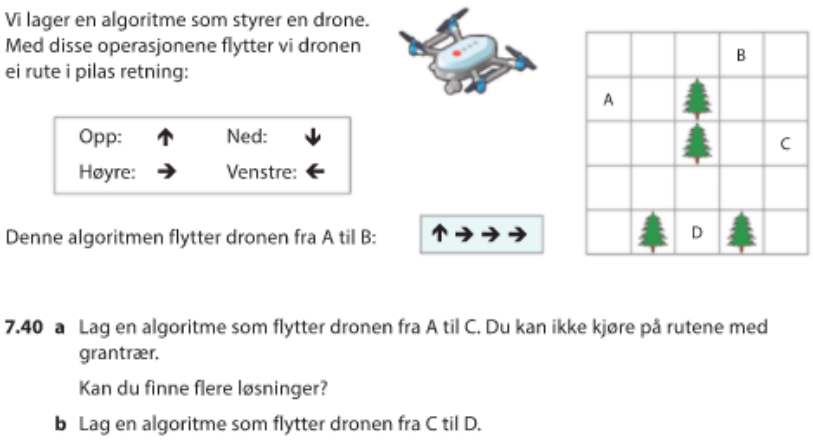
Ser vi på Figur 1 viser den at handlingene *følge* og *forme og skape* er de to mest vanlige handlingene både på mellom- og ungdomstrinnet. Legg merke til at handlingen elevene møter minst er å *feilsøke*, dette er gjeldende for både 5-7 og 8-10. De få oppgavene som inneholder feilsøking er ofte strukturert slik at elevene blir presentert et sett kodelinjer som inneholder en feil, eller at de blir bedt om å lage et program som de skal feilsøke og eventuelt rette opp feilkoder. Det kan argumenteres for at elevene også må drive med feilsøking i oppgaver der elevene skal lage egne programmer, slik som i Figur 4. Slike oppgaver er ikke registrert som *feilsøke*, kun oppgaver der elevene eksplisitt blir bedt om å feilsøke faller under denne kategorien.

Vi finner et skille mellom 5-7 og 8-10 i handlingene *forklare* og *forestille seg* (Figur 2). Oppgavene i 5-7 inneholder flere oppgaver der elevene blir bedt om å se for seg mulige utfall når man for eksempel endrer på enkelte variabler, kodelinjer eller rekkefølgen på en algoritme. Mens det på 8-10.trinn er flere oppgaver der elevene blir bedt om å forklare i større grad, for eksempel gi en forklaring til gitte kodelinjer eller en algoritme som en helhet. Figur 2 viser forskjellene i fordelingen av handlingene mellom 5-7 og 8-10 tydeligere.



Figur 2 – fordeling av antall oppgaver registrert i de ulike handlingene, 5-7 og 8-10 hver for seg

Handlingen *følge* er som sagt en av de to mest vanlige, og er identifisert i over halvparten av oppgavene som er analysert. Oppgaver blir registrert innenfor denne kategorien når enten elevene blir presentert en eksempeloppgave som en «oppskrift» rett i forkant av en oppgave, eller når oppgaven inneholder steg-for-steg instruksjoner i oppgaven. Eksempelet i Figur 3 ser vi en typisk oppgave der elevene får en «oppskrift» i forkant av oppgaven, og deretter følger disse instruksjonene for å løse oppgaven 7.40.


Vi lager en algoritme som styrer en drone.
Med disse operasjonene flytter vi dronen
ei rute i pilas retning:

Opp: ↑ Ned: ↓
Høyre: → Venstre: ←

Denne algoritmen flytter dronen fra A til B: ↑ → → →

A
B
C
D

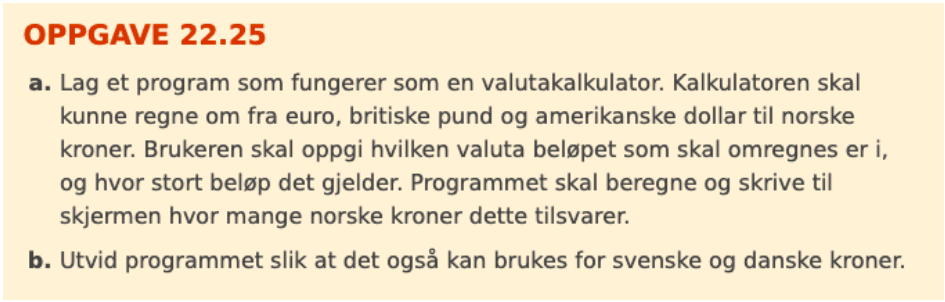
7.40 **a** Lag en algoritme som flytter dronen fra A til C. Du kan ikke kjøre på rutene med grantrær.

Kan du finne flere løsninger?

b Lag en algoritme som flytter dronen fra C til D.

Figur 3 – eksempel på en oppgave med handlingen «følge» på 5.trinn (Multi 5B, s. 83)

Oppgaven i Figur 3 er i tillegg et eksempel på den mest vanlige handlingen, *forme og skape*, der elevene i dette tilfelle skal lage ulike algoritmer som flytter dronen til spesifikke ruter. Analysen viser at handlingene *forme og skape* og *følge* ofte er inkludert i samme oppgave. I eksempelet i Figur 3 blir elevene bedt om å lage egne stegvise instruksjoner, på samme måte som i eksempelet i forkant av oppgaven. Samtidig er det også oppgaver som inneholder handlingen *forme og skape* som ikke inneholder handlingen *følge.* Dette er spesielt gjeldende for 8-10.trinn, der elevene blir gitt oppgaver der de skal lage et dataprogram uten at de får et eksempel i forkant eller en tydelig beskrivelse for hvordan de skal gå frem for å løse oppgaven. Figur 4 er et eksempel på en slik oppgave.



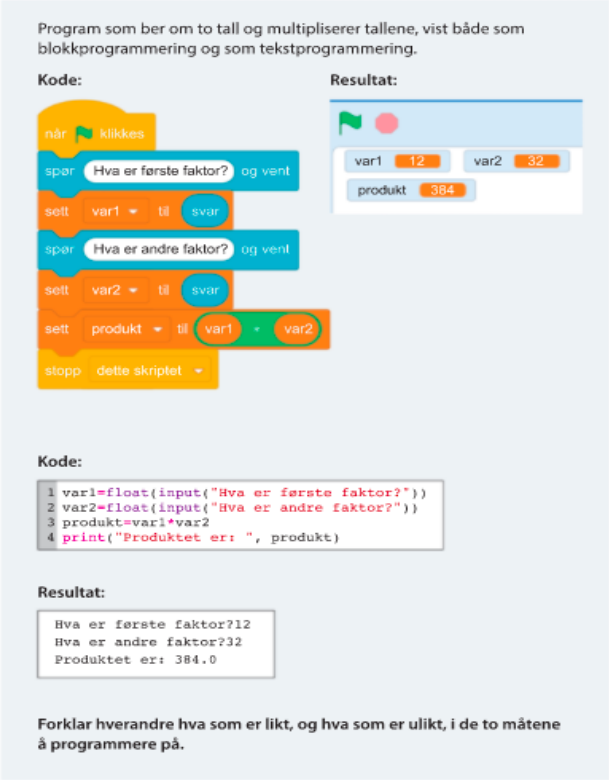
OPPGAVE 22.25

a. Lag et program som fungerer som en valutakalkulator. Kalkulatoren skal kunne regne om fra euro, britiske pund og amerikanske dollar til norske kroner. Brukeren skal oppgi hvilken valuta beløpet som skal omregnes er i, og hvor stort beløp det gjelder. Programmet skal beregne og skrive til skjermen hvor mange norske kroner dette tilsvarer.

b. Utvid programmet slik at det også kan brukes for svenske og danske kroner.

Figur 4 – eksempel på en oppgave med handlingen «forme og skape» (Matemagisk 10, s. 135)

I sammenheng med handlingen *forme og skape* er det interessant å se på hvilke ulike programmeringsspråk lærebøkene anbefaler elevene å bruke i oppgavene. På mellomtrinnet varierer lærebøkene mellom å ha innføring i analog programmering uten koblinger til programmeringsspråk, til å ha innføring i blokk-programmering. Trinket og Scratch er de blokk-baserte programmeringsspråkene som benyttes av lærebøkene. Trinket har i tillegg en funksjon som gjør det enkelt for elevene å kunne se koden bak blokkene de benytter, noe som kan være nyttig for elevene å ta med seg videre i utdanningsløpet da de på ungdomstrinnet som regel benytter seg av tekstbaserte programmeringsspråk. På ungdomstrinnet brukes Python som programmeringsspråk i alle lærebøkene, enkelte bøker har også innslag av blokk-programmering. I en av lærebøkene (Maximum 8) blir elevene presentert en oppgave der de skal diskutere sammenhenger mellom blokk- og tekstbasert programmering (Figur 5).


Program som ber om to tall og multipliserer tallene, vist både som blokkprogrammering og som tekstprogrammering.

Kode:

Resultat:

når klikkes
spør Hva er første faktor? og vent
sett var1 til svar
spør Hva er andre faktor? og vent
sett var2 til svar
sett produkt til var1 * var2
stopp dette skriptet

var1 12 var2 32
produkt 384

Kode:

```
var1=float(input("Hva er første faktor?"))
var2=float(input("Hva er andre faktor?"))
produkt=var1*var2
print("Produktet er: ", produkt)
```

Resultat:

```
Hva er første faktor?12
Hva er andre faktor?32
Produktet er: 384.0
```

Forklar hverandre hva som er likt, og hva som er ulikt, i de to måtene å programmere på.

Figur 5 – eksempel på en oppgave med handlingen «forklare» (Maximum 8, s. 135)

Oppgaver som inneholder handlingen *finne regel* er de nest minst vanlige. De oppgavene som inneholder denne handlingen går som regel ut på at elevene skal finne en regel for å fortsette på et mønster. Det mest vanlige er at elevene skal finne en regel for å fortsette et tallmønster og deretter lage et program som fortsetter på mønsteret. Enkelte oppgaver handler om at elevene skal fortsette på mønster som består av geometriske figurer. Figur 6 viser et eksempel på en slik oppgave.


2.72 **Programmer tallmønsteret**

«Gjenta til»-blokken er fin å bruke for å teste når en skal arbeide med å løse matematiske problemer, for eksempel å finne tallmønstre. Nedenfor er det laget et lite blokkbasert program som kan finne tre tall som følger etter hverandre og som har en viss sum.

Bruk et blokkbasert program, og se om du kan finne fem andre heltall som kan være summen av tre tall som følger etter hverandre.

Finn en regel for hvilke tall dette programmet fungerer for.

Figur 6 – eksempel på en oppgave med handlingen «finne regel» (Maximum 8, s. 144)

4.2 Begreper

I dette delkapittelet vil jeg presentere resultater fra analysen knyttet til kategoriseringen av AT-begrepene og de matematiske begrepene som ble identifisert i oppgavene. Én oppgave kan inneholde flere begreper, både flere matematiske- og AT-begreper.

4.2.1 AT-begreper

Tabell 1 - Frekvens og prosentandel der de ulike AT-begrep er inkludert i oppgavene.

AT - begreper	Oppgaver på 5-7.trinn	Oppgaver på 8-10.trinn
Steg for steg instruksjoner	45 (57%)	104 (54%)
Algoritmer	44 (56%)	115 (58%)
Løkker, iterasjon, repetisjon	35 (44%)	75 (38%)
Regel	17 (22%)	31 (16%)
Kode	33 (42%)	171 (87%)
Betingelser	18 (23%)	66 (34%)
Bug/feil, feilsøking	6 (8%)	23 (12%)

Jeg begynner med å ta for meg resultatene knyttet til AT-begrepene som ble identifisert i analysen av oppgavene. Ut i fra Tabell 1 ser vi at den tydeligste forskjellen mellom 5-7 og 8-10 er at AT-begrepet *kode* fremtrer omtrent dobbelt så ofte på ungdomstrinnet. Dette kan ha noe sammenheng med at det på ungdomstrinnet i større grad er mer komplekse oppgaver, der elevene oftere må utvikle egne programmer. I tillegg til at det på ungdomstrinnet er flere oppgaver som innebærer handlingen *forme og skape*, noe som i større grad krever at elevene må skrive egne kodelinjer for å løse det spesifikke problemet. I tillegg er det også på mellomtrinnet, spesielt 5.trinn, oppgaver og aktiviteter med analog programmering der begrepet *kode* ikke blir relevant. Videre er det på mellomtrinnet oppgaver der elevene introduseres for algoritmer, der de skal lage egne algoritmer som løser en oppgave. Et eksempel kan være «lag en algoritme for å lage et glass sjokolademelk». Der elevene skal lage egne skriftlige instruksjoner fremfor å bruke et blokk- eller tekstbasertprogrammeringsspråk.

Ser vi på de to begrepene *steg-for-steg instruksjoner* og *algoritme* ser vi at disse er omtrent like vanlige på tvers av trinnene. *Steg-for-steg instruksjoner* er tilstede i litt over halvparten av oppgavene, noe som man kan se i sammenheng med handlingen *følge*. Oppgaven i Figur 3

viser en typisk *steg-for-steg* oppgave på mellomtrinnet. Oppgaver der elevene enten får presentert et eksempel i forkant av oppgaven og deretter løser oppgaven basert på instruksjonene gitt i eksempelet, er svært vanlig i lærebøkene. I lærebøkene er de fleste oppgavene med *steg-for-steg instruksjoner* i starten av kapitlene eller når de presenterer et nytt begrep. Oppgaver som i større grad krever at elevene må tenke selv og utforske kommer mot slutten av kapitlene.

Algoritme er som nevnt et av de begrepene elevene møter oftest i oppgavene. I Figur 7 ser vi en typisk oppgave fra tidlig mellomtrinn som involverer begrepet *algoritme*. I forkant av oppgaven har elevene fått en introduksjon om hva en algoritme og et flytdiagram er, der flytdiagram blir eksplisitt trukket frem som et nyttig verktøy i planleggingsfasen av det å lage et dataprogram.

Figur 7 – eksempel på oppgave med begrepet algoritme (Multi 5B, s. 84)

Felles for lærebøkene er hvordan de beskriver en algoritme i programmeringssammenheng. Det som går igjen er fokus på at rekkefølgen er avgjørende for resultatet. Det er noe varierende hvorvidt lærebøkene skiller mellom matematiske algoritmer og algoritmer i programmeringssammenheng. Et eksempel fra Maximum 8; «En algoritme er en nøyaktig beskrivelse av trinn i en prosess som skal utføres for å løse ett eller flere problemer. Algoritmene må følges nøyaktig, og trinnenes rekkefølge er avgjørende for resultatet...» (Tofteberg et al., 2020, s 128)

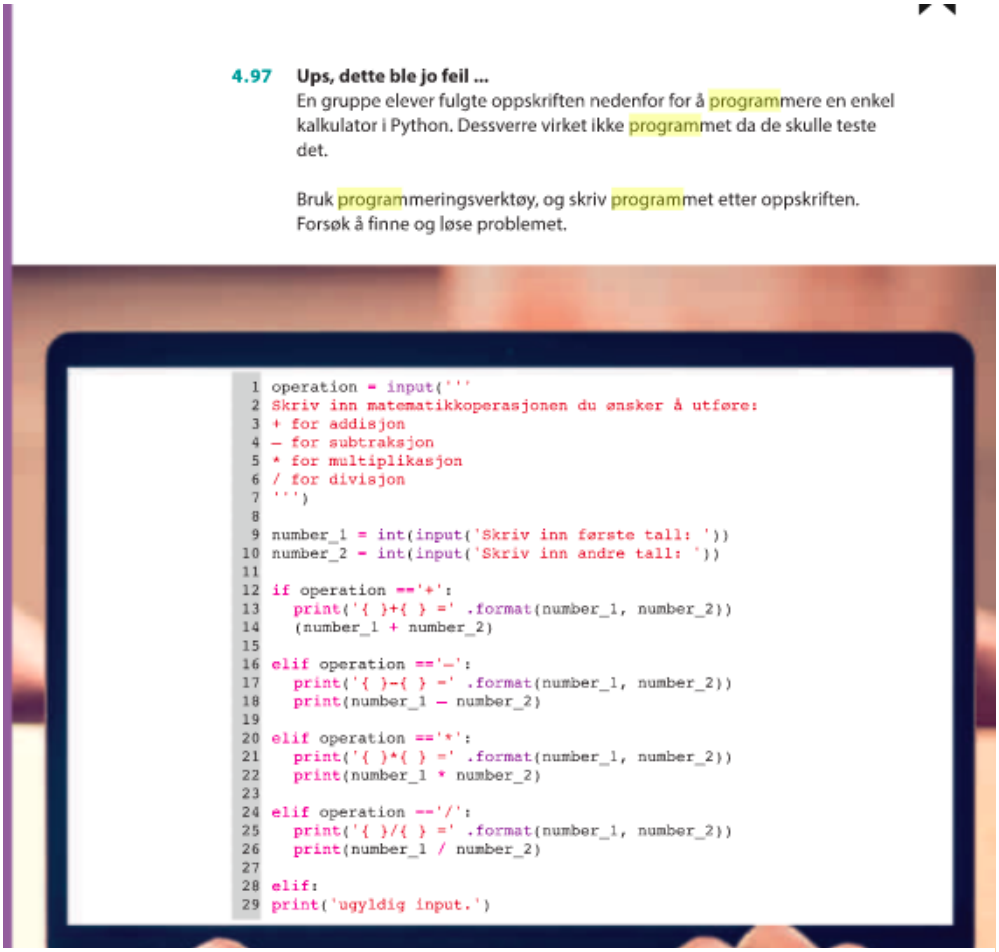
Begrepet *løkker* er del i henholdsvis 44% og 38% av oppgavene på mellomtrinnet og ungdomstrinnet (Tabell 1), men hvordan det brukes og i hvilken sammenheng varierer

mellom trinnene. På mellomtrinnet blir løkker ofte brukt i sammenheng med å effektivisere algoritmer for å tegne geometriske figurer, tegne mønster eller i oppgaver som ligner på den i figur 3 for å flytte en drone/robot med færre kodelinjer eller blokker. Hvilket det på ungdomstrinnet som regel brukes i sammenheng med å gjennomføre simuleringer eller skrive/finne tallrekker eller tallmønster. Et annet funn er at på mellomtrinnet utelukkende brukes *for-løkker,* som anvendes når vi vet hvor mange ganger noe skal gjentas. Mens på ungdomstrinnet, i tillegg til *for-løkker*, bruker de også *while-løkker* (Figur 6). While-løkker brukes når vi vil at noe skal gjentas så lenge noe er sant, altså ved at vi legger til en betingelse for når løkken skal slutte. Det er også varierende mellom lærebøkene i hvilken grad de tar for seg overgangen fra løkker med blokkbasert- til tekstbasert programmering. Lærebøkene, foruten om Multi og Maximum fra Gyldendal, inkluderer ikke oppgaver der elevene får mulighet til å se og diskutere forskjeller mellom blokk- og tekstbasertprogrammering slik som oppgaven i Figur 5.

Oppgaver som inneholder begrepet *betingelser* øker fra mellom- til ungdomstrinnet (Tabell 1). På mellomtrinnet brukes det som regel i sammenheng med for eksempel å avgjøre om noe er sant eller ikke. Dette er også tilfelle på ungdomstrinnet. Men i tillegg brukes det også på ungdomstrinnet i mer komplekse oppgaver, som for eksempel til å lage programmer som løser likninger. Begrepet *regel* er blant de AT-begrepene som er minst til stede i oppgavene som er analysert (Tabell 1). Typiske oppgaver som inkluderer dette begrepet er oppgaver der elevene skal undersøke et mønster (tegning/geometriske mønster eller tallrekker) for å lage et program som tegner eller fortsetter på mønsteret som i Figur 6. Ut i fra tabell 1 ser vi at begrepet *kode* oftere er inkludert i oppgavene på ungdomstrinnet. Dette henger sammen med at det på ungdomstrinnet er flere oppgaver der elevene må bruke et programmeringsverktøy.

I likhet med resultatene i Figur 1 er også begrepet *feilsøke* sjeldent inkludert i oppgavene (Tabell 1). I tillegg til at det er lite fokus på feilsøking i oppgavene så er det heller ingen av bøkene som tar opp det å feilsøke som en viktig del av programmering, foruten om å nevne at rekkefølgen i en algoritme er viktig. Det er imidlertid noen oppgaver der elevene eksplisitt blir bedt om å feilsøke (Figur 8 og 9). Samtidig ser vi ut i fra Figur 1 og Tabell 1 at både handlingen og begrepet *feilsøke* i liten grad er inkludert i oppgavene. For å undersøke om bruken av *feil/feilsøking* blir brukt som et pedagogisk virkemiddel i oppgaver på andre fagområder i lærebøkene gjennomførte jeg et søk på ordet «feil» i en tilfeldig lærebok

(Matemagisk 10). I denne læreboken alene var det i underkant av 30 oppgaver der feil ble brukt som et virkemiddel for å skape forståelse for et matematisk begrep.



4.97 **Ups, dette ble jo feil ...**

En gruppe elever fulgte oppskriften nedenfor for å programmere en enkel kalkulator i Python. Dessverre virket ikke programmet da de skulle teste det.

Bruk programmeringsverktøy, og skriv programmet etter oppskriften. Forsøk å finne og løse problemet.

```
operation = input('''
Skriv inn matematikkoperasjonen du ønsker å utføre:
+ for addisjon
- for subtraksjon
* for multiplikasjon
/ for divisjon
''')

number_1 = int(input('Skriv inn første tall: '))
number_2 = int(input('Skriv inn andre tall: '))

if operation =='+':
   print('{ }+{ } =' .format(number_1, number_2))
   (number_1 + number_2)

elif operation =='-':
   print('{ }-{ } =' .format(number_1, number_2))
   print(number_1 - number_2)

elif operation =='*':
   print('{ }*{ } =' .format(number_1, number_2))
   print(number_1 * number_2)

elif operation =='/':
   print('{ }/{ } =' .format(number_1, number_2))
   print(number_1 / number_2)

elif:
print('ugyldig input.')
```

Figur 8 – eksempel på oppgave med feilsøking (Maximum 8, s. 280)

2.57 I koden er målet at figuren kommer tilbake til utgangspunktet, i samme posisjon. Når vi kjører koden, oppdager vi at noe ikke er helt riktig.

Bruk et blokkbasert programmeringsspråk, og test koden. Finn feilen, og gjør de nødvendige endringene for å rette feilen.

Figur 9 – eksempel på oppgave med feilsøking (Maximum 8, s. 136)

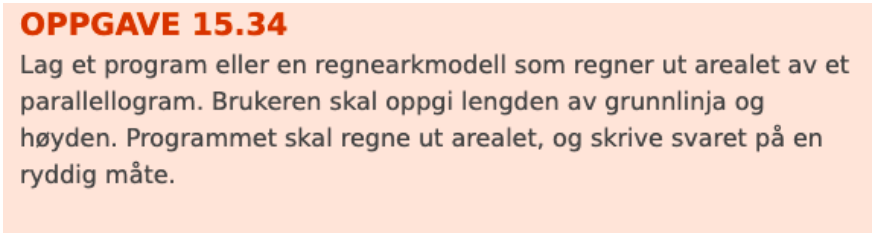
4.2.2 Matematiske begreper

Jeg går nå over på å se på de matematiske begrepene som ble identifisert i oppgavene. På mellomtrinnet var det 17 av totalt 79 oppgaver som ble registrert som uten matematisk innhold, og på ungdomstrinnet 30 av 197 oppgaver (Tabell 2).

Tabell 2 - Frekvens og prosentandel der matematiske begrep er inkludert.

Matematiske - begreper	Oppgaver på 5-7.trinn	Oppgaver på 8-10.trinn
Mønster (inkludert tallmønster)	9 (11%)	27 (14%)
Geometriske begreper	16 (20%)	27 (14%)
Aritmetiske begreper	26 (33%)	72 (37%)
Rotasjon	2 (3%)	1 (1%)
Koordinatplan, funksjoner	11 (14%)	24 (12%)
Statistikk og sannsynlighet	0 (0%)	29 (15%)
Uten matematisk innhold	17 (22%)	30 (15%)

Statistikk og sannsynlighet finner vi utelukkende på ungdomstrinnet, noe som henger sammen med kompetansemål etter 9.trinn som spesifikt knytter programmering til dette. Oppgaver som inneholder *geometriske begreper* er registrert omtrent like ofte på mellom- og ungdomstrinnet, noe mer på mellomtrinnet. Dette kan ha sammenheng med kompetansemålet etter 6.trinn, der elevene skal utforske geometriske figurer og mønstre. Hovedsakelig handler oppgavene innenfor geometri om å tegne enkle geometriske figurer med blokkprogrammering på mellomtrinnet. Mens det i oppgavene på ungdomstrinnet er oppgaver der elevene må tegne geometriske figurer med spesifikke mål og bruke programmering til å regne ut areal og/eller omkrets. Eller, som i Figur 10, lage programmer som kan regne ut arealet etter input fra en bruker.

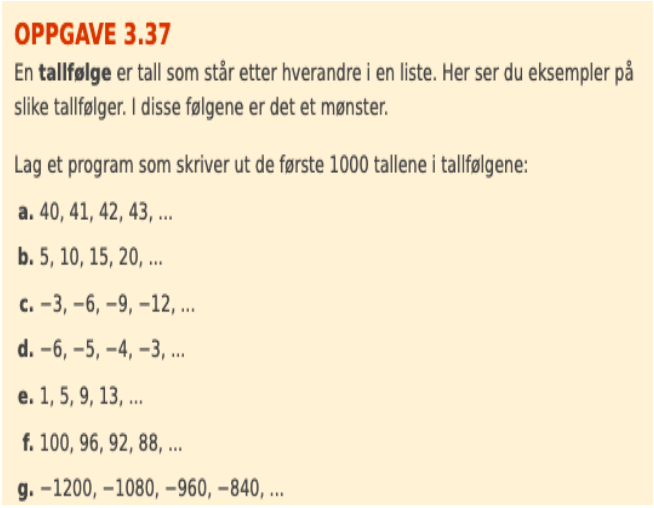


Figur 10 – eksempel på oppgave med geometri (Matemagisk 9, s. 166)

I analysen har jeg hatt en bred forståelse for kategorien *aritmetiske begreper*, dette for å kunne kategorisere matematiske begrep som ligger nær aritmetikken, som talls

delingsegenskaper. Oppgaver der én eller flere av de fire regneartene har en relativt sentral rolle for å løse oppgaven er blitt registrert innenfor denne kategorien. Som vi ser ut i fra tabell 2 er det dette matematiske begrepet som oftest er brukt i programmeringsoppgavene.

Andelen programmeringsoppgaver som inneholder begrepet *mønster* er relativt lik mellom 5-7.trinn og 8-10.trinn. På mellomtrinnet handler som regel oppgavene om å lage programmer som kan tegne et gitt mønster, altså reprodusere et gitt mønster. Mens på ungdomstrinnet er det et mer variert utvalg av oppgaver som kan handle om å lage programmer som skriver bestemte tallrekker eller å bruke programmering til å finne og løse oppgaver med figurtall (figur 11 og 12)



OPPGAVE 3.37

En **tallfølge** er tall som står etter hverandre i en liste. Her ser du eksempler på slike tallfølger. I disse følgene er det et mønster.

Lag et program som skriver ut de første 1000 tallene i tallfølgene:

a. 40, 41, 42, 43, ...

b. 5, 10, 15, 20, ...

c. −3, −6, −9, −12, ...

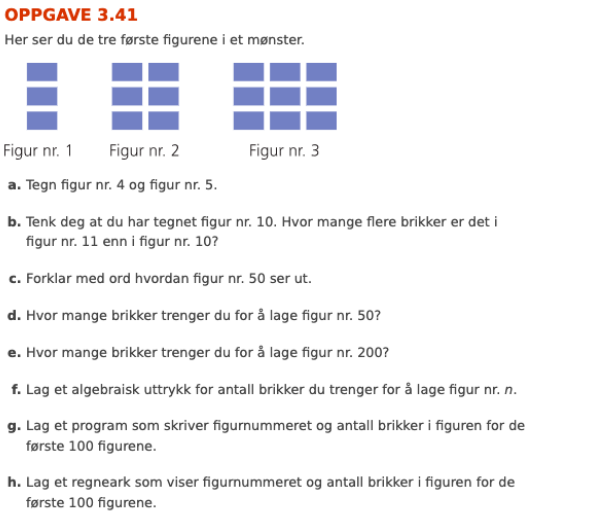
d. −6, −5, −4, −3, ...

e. 1, 5, 9, 13, ...

f. 100, 96, 92, 88, ...

g. −1200, −1080, −960, −840, ...

Figur 11 – oppgave kategorisert som mønster (Matemagisk 8, s. 115)



OPPGAVE 3.41

Her ser du de tre første figurene i et mønster.

Figur nr. 1 Figur nr. 2 Figur nr. 3

a. Tegn figur nr. 4 og figur nr. 5.

b. Tenk deg at du har tegnet figur nr. 10. Hvor mange flere brikker er det i figur nr. 11 enn i figur nr. 10?

c. Forklar med ord hvordan figur nr. 50 ser ut.

d. Hvor mange brikker trenger du for å lage figur nr. 50?

e. Hvor mange brikker trenger du for å lage figur nr. 200?

f. Lag et algebraisk uttrykk for antall brikker du trenger for å lage figur nr. *n*.

g. Lag et program som skriver figurnummeret og antall brikker i figuren for de første 100 figurene.

h. Lag et regneark som viser figurnummeret og antall brikker i figuren for de første 100 figurene.

Figur 12 – oppgaver kategorisert som mønster (Matemagisk 8, s.117)

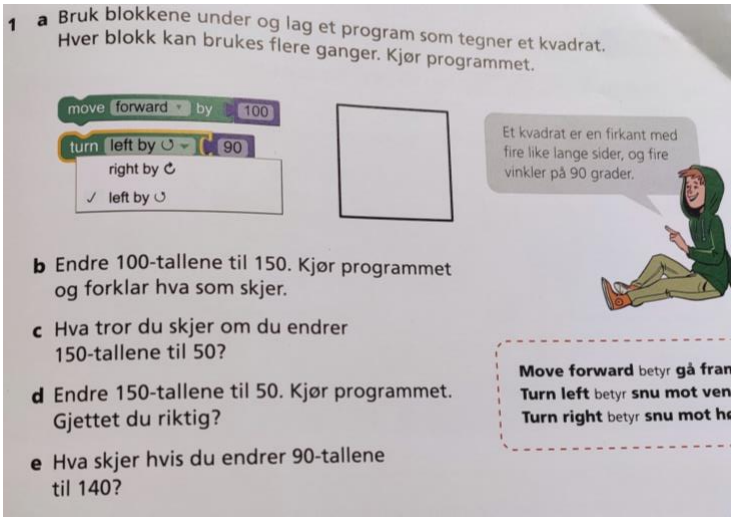
Oppgavene som inkluderer *koordinatplan og funksjoner* kjennetegnes av oppgaver der elevene skal gi instruksjoner eller lage algoritmer for å flytte objekter i et koordinatplan på mellomtrinnet. Oppgavene på ungdomstrinnet fokuserer mer på å lage programmer som kan regne ut funksjonsverdier. I tillegg er det et fåtall oppgaver der elevene skal tegne grafer ved hjelp av programmering, men som regel oppfordres det i oppgavene til å bruke mer egnete verktøy som Geogebra. Lærebøkene tar også opp et viktig skille mellom funksjoner, slik vi kjenner det fra matematikken, og funksjoner som begrep i programmeringssammenheng. Begrepet *rotasjon* var svært lite tilstedeværende i oppgavene, tre oppgaver totalt, disse oppgavene gikk ut på å bruke programmering til å speile geometriske figurer.

4.3 Brobygging mellom programmering og matematikk

Med utgangspunkt i både analysen av handlingene og begrepene vil jeg nå presentere funn fra lærebøkene som går på den overordnede idéen om å undersøke i hvilken grad oppgavene legger til rette for at elevene kan se sammenhenger mellom programmering og matematikk. Forskningsspørsmål 2 handler om å undersøke hvordan oppgavene i lærebøkene legger til rette for å få frem forholdet mellom matematikk og programmering. Ser vi tilbake til Tabell 2 er det relativt mange oppgaver som er kategorisert uten et matematisk innhold (22%, 5-7 og 15%, 8-10).

Lærebøkene på de ulike trinnene knytter som regel programmeringsinnholdet til de respektive kompetansemålene til de ulike trinnene, for eksempel er det meste programmeringsinnholdet i lærebøker på 9.trinn knyttet til sannsynlighet, og på 6.trinn knyttet til geometri. De resterende kompetansemålene gir mer rom for lærebokforfatterne til å knytte programmeringsinnholdet til flere områder, noe som spesielt blir utnyttet i lærebøkene for åttende og tiende trinn, der programmeringsinnholdet knyttes til flere ulike matematiske begreper. Lærebøkene på 10.trinn inneholder i tillegg oppgaver som knyttes til det tverrfaglige temaet «Folkehelse og livsmestring», der programmering blir brukt til blant annet å lage budsjett og regne ut lånekostnader.

I lærebøkene, spesielt på mellomtrinnet, er oppgavene i starten av kapitlene ofte preget av handlingene *følge*. Samtidig som oppgavene ofte enten utelukkende inneholder programmering eller der matematikk er kontekst for å lære seg programmering. Altså der det brukes kjente matematiske begrep for å lære seg programmering, og ikke motsatt, lære seg matematikk gjennom programmering. Deloppgave a i Figur 13 er et eksempel på hvordan slike oppgaver ofte kan se ut, ofte også med en eksempeloppgave i forkant. Elevene blir presentert et sett med kodeblokker for å tegne et kvadrat, som er et kjent begrep på 5/6.trinn, det matematiske begrepet kvadrat blir brukt for å lære seg å programmere med blokker.



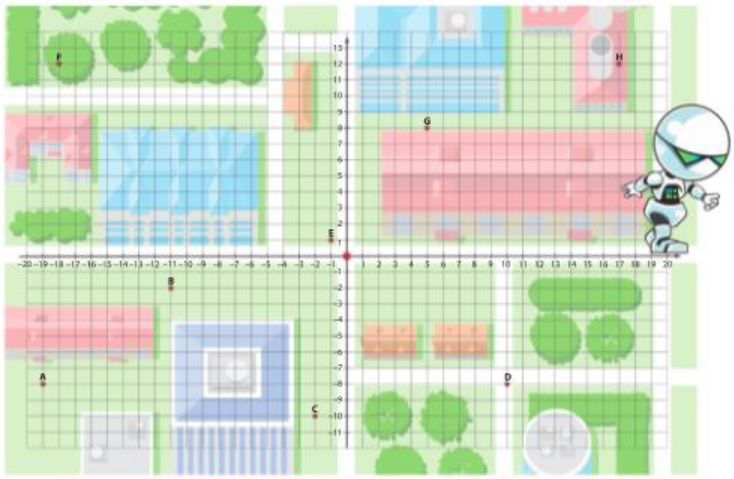
Figur 13 – oppgave med handlingene forklare og forestille seg (Matemagisk 5B, s 99)

I de påfølgende oppgavene derimot blir forholdet snudd, ved at elevene da skal bruke kunnskapen om blokkprogrammering til å tegne andre geometriske figurer. I tillegg til at de neste oppgavene innebærer handlingen *forme og skape* blir koblingen mellom programmeringen og det matematiske begrepet tydeligere. Elevene må ta i bruk eller søke kunnskap om hvilke kriterier som må oppnås for å for eksempel tegne en likebeint trekant. Dette altså uten handlingen *følge*, der de har tydelige instruksjoner eller en eksempeloppgave i forkant. Her vil elevene også potensielt oppdage sammenhenger mellom verdien til blokkene «turn» og vinklene i de ulike geometriske figurene. I tillegg har noen lærebøker oppgaver der elevene starter med et program som tegner et kvadrat, men blir bedt om å se for seg (*forestille seg*) hva som skjer når de endrer på verdiene i koden, slik som i deloppgavene i Figur 13. I denne oppgaven (Figur 13) er det spesifikke instruksjoner for hvilke endringer som skal gjøres i hver deloppgave. I tillegg er det lagt opp til at elevene skal forestille seg hva som skal skje før de gjør en endring i de gitte blokkene.

Ser vi videre på oppgaver som inkluderer handlingen *forme og skape*, uten rigide *steg-for-steg instruksjoner*, ser det ut til at oppgaver med denne handlingen i større grad kan fungere som oppgaver der programmering blir brukt som et verktøy for å effektivisere beregninger innenfor matematikken. Oppgaver som i tillegg suppleres med én eller flere av handlingene *forklare, finne regel eller forestille seg* ser ut til å skape en tydeligere sammenheng mellom programmering og matematikken. Oppgavene legger til rette for å bruke programmering til å utforske matematiske begrep og idéer. Det er relativt få oppgaver i lærebøkene som inkluderer handlingene *forestille seg* og *forklare* (Figur 1). I tillegg er det få oppgaver som handler om

det å forklare enkelte kodelinjer eller hele algoritmer, som kan fremheve de matematiske aspektene ved en programkode.

I Figur 3 ser vi et eksempel på en oppgave der elevene skal gi instruksjoner til en drone for at den skal forflytte seg til en gitt rute i et rutenett. Denne typen oppgave var vanlig i introduksjonen av programmering i lærebøkene for 5-7.trinn. Hensikten med slike oppgaver ser ut til å være opplæring i programmeringsprinsipper, uavhengig av matematikk. Noen vil kanskje argumentere for at rutenett vil kunne fungere som en introduksjon i en overgang til koordinatplanet. Men, i et rutenett er det fokus på rutene og ikke koordinatene i et koordinatplan. En programmeringsoppgave som i større grad kan brukes til innføring i koordinatplan er oppgaven i Figur 14. Her spiller det matematiske innholdet en tydeligere rolle i det å lære seg programmering, og kan fungere som en oppgave for å introdusere eller repetere egenskaper i et koordinatsystem.



6.34 Roboten er programmert til å følge ei bestemt rute fra A til H.

a Følg ruta og skriv koordinatene for hvert stopp den gjør.

b Beskriv ruta mellom hvert stoppunkt. Lag tabell.

Fra punkt	Til punkt	Bevegelse
A (–19, –8)	B (–11, –2)	8 Ø, 6 N
B (–11, –2)	C (–2, –10)	
C (–2, –10)		

6.35 Programmer roboten til å gå ei ny rute fra A til H.

Start i koordinat A (–19, –8).

Fra punkt	Til punkt	Bevegelse
A (–19, –8)	B (…)	

Figur 14 – Oppgave med koordinatplan 6.trinn (Multi 6B, s. 52)

Oppgaver med kombinasjonen av handlingen *følge* og begrepet *steg-for-steg instruksjoner* ser ut til å være de oppgavene som i minst grad inkluderer matematiske begreper og idéer, og som i minst grad skaper en kobling mellom matematikk og programmering. Det er viktig å påpeke at dette ikke er gjeldende for alle oppgaver med denne kombinasjonen. Rollen til disse oppgavene handler ofte mer om å gi elevene innføring i et nytt begrep eller verktøy innenfor programmering som de senere vil få bruk for. Dette kan ses i sammenheng med at matematikkfaget har fått hovedansvaret for opplæringen i programmering. Og denne typen oppgaver kan være nødvendig for å opparbeide seg nødvendige ferdigheter for å kunne bruke programmering til å utforske i matematikkfaget. I analyseprosessen er det tydelig at ferdighetene og begrepene som presenteres i slike oppgaver blir tatt i bruk senere i oppgaver der det er en tydeligere sammenheng mellom programmering og matematikken.

Gjennom analyseprosessen observerte jeg at det på ungdomstrinnet var flere oppgaver der programmering brukes til å utforske matematiske begreper og idéer, og/eller der programmering brukes til å effektivisere matematiske beregninger. Kontra oppgavene på mellomtrinnet der det er flere oppgaver som utelukkende handler om programmering eller oppgaver der matematikk er konteksten for å lære seg programmering. I Figur 15 ser vi et eksempel på en slik oppgave fra ungdomstrinnet. Vær likevel oppmerksom at denne typen oppgaver er relativt sjeldne.

2.56 Samarbeid to og to. I dette programmet leter vi etter tilnærmede verdier for nullpunktene til funksjonen $f(x) = x^2 - 2x - 1$.

```
from pylab import *

def f(x):
  return x**2-2*x-1

x=linspace(-10,10,100)
y=f(x)

ylim(-3,6)
xlim(-4,4)
akser=gca()
akser.spines["bottom"].set_position("zero")
akser.spines["left"].set_position("zero")
akser.spines["top"].set_visible(False)
akser.spines["right"].set_visible(False)

plot(x,y)
grid()
show()

antall=int(input("Hvor mange nullpunkter skal vi finne?"))

for i in range(0,antall,1):
  print("\nVi finner nullpunkt nr.",i+1)
  under=float(input("Oppgi nedre grense:"))
  over=float(input("Oppgi øvre grense:"))

  for n in range(0,6,1):
    x_verdi=(under+over)/2
    if f(under)*f(x_verdi)<0:
      over=x_verdi
    else: under=x_verdi

  print("Nullpunkt",i+1, ":", round(x_verdi,4))
```

a Kjør programmet og finn ut hvordan det fungerer. Hvorfor er grafplotting tatt med når målet er å finne nullpunktene?

b I programmet brukes «halveringsmetoden» for å finne nullpunkter. Forklar hvordan metoden fungerer, og hvorfor dette er et godt navn på metoden.

c Skriv inn forklaringstekster til hver del av programmet (#-linjer), slik at funksjonaliteten i programmet kan forstås bedre.

d Forbedre programmet, slik at nullpunktene finnes med større nøyaktighet. Forklar hvordan dere velger å forbedre, og hvorfor.

e Forbedre programmet slik at det
1. blir mer generelt og kan finne nullpunkter for en kvadratisk funksjon som velges av brukeren
2. blir mer robust overfor feil inntastinger av brukeren

Figur 15 – oppgave med tydelig matematisk innhold (Maximum 10, s. 121)

I denne oppgaven fungerer programmering som et verktøy slik at elevene kan utforske kvadratiske funksjoner. Der kan elevene samtidig øve seg å kritisk vurdere hvilke digitale verktøy som er hensiktsmessige å bruke i gitte situasjoner. Geogebra kunne i dette tilfellet potensielt vært et mer effektivt verktøy. Men på den andre siden vil elevene kunne få mer innsikt og kunnskap ved å se oppbyggingen i et program, samt ved å selv forbedre programmet slik som i deloppgave *e*. Oppgaven legger også opp til, gjennom handlingen *forklare*, at elevene skal forklare hvordan programkoden fungerer. Dette kan igjen bidra til at elevene ser sammenhengene mellom kodelinjene og matematikken, og kan identifisere det matematiske innholdet i programkoden.

5 Diskusjon

Jeg vil i dette kapittelet diskutere resultatene fra analysen opp mot tidligere presentert teori, for å besvare forskningsspørsmålene med forankring i teori og resultatene. Strukturen i diskusjonen følger de to forskningsspørsmålene.

Hensikten med studien er, som nevnt tidligere, ikke å sammenligne lærebøkenes innhold. Men å undersøke hva som kjennetegner programmeringsinnholdet norske elever kan møte i matematikkfaget, og i hvilken grad oppgavene skaper eller får frem sammenhenger mellom programmering og matematikken. Det vil imidlertid være interessant, i noen tilfeller, å ta opp forskjeller mellom bøkene for å fremheve viktige poeng. I tillegg til å ta opp forskjeller mellom programmeringsinnholdet på mellom- og ungdomstrinnet. Diskusjonen vil også komme inn på i hvilken grad lærebøkenes programmeringsinnhold og oppgaver legger opp til utvikling av elevenes AT, på grunn av relevansen dette har som en problemløsningsstrategi og til programmering.

5.1 Forskningsspørsmål 1: Hva karakteriserer programmeringsinnholdet i norske lærebøker i matematikk på mellom- og ungdomstrinnet?

Forskningsspørsmål 1 ser nærmere på hva som karakteriserer programmeringsinnholdet i lærebøkene. Jeg vil i dette delkapittelet se nærmere på hvilke handlinger og begreper som karakteriserer innholdet i lærebøkene, og se dette i sammenheng med hva forskningen sier om programmeringens rolle i skolen og matematikkfaget.

5.1.1 Karakteristiske trekk ved oppgavene

Noe av det mest karakteristiske på tvers av lærebøkene var at store deler av oppgavene inneholdt handlingene *følge* og *forme og skape*, og i et relativt stort utvalg av oppgavene en kombinasjon av disse. Som nevnt tidligere fremkom det i analyseprosessen at det oftere var oppgaver på mellomtrinnet med kombinasjonen av disse to handlingene, mens oppgavene med handlingen *forme og* skape på ungdomstrinnet oftere var løsrevet fra handlingen *følge.* Dette ser ut til å gi mer rom for at elevene kunne ta i bruk og øve på ferdigheter innenfor AT, fordi oppgavene ikke inneholdt en oppskrift eller stegvise instruksjoner for hvordan den skulle løses. For eksempel oppgaven med valutakalkulatoren (Figur 4). Slike oppgaver skaper mer rom for at elevene får øvelse i å gjennomgå informasjonen systematisk (lage et program som tilfredsstiller oppgavekravene), modulisere problemer (lage en plan for hvordan de kan

løse problemet), lese og bruke ulike representasjonsformer (skifte mellom å løse oppgaven rent matematisk, for å så overføre det til et programmeringsspråk) (Lodi & Martini, 2021; Sanne et al., 2016; Wing, 2006). Denne typen oppgaver er i større grad inkludert i lærebøkene for ungdomstrinnet, noe som naturligvis kan ha bakgrunn i progresjonen elevene vil ha fra mellom- til ungdomstrinnet. Samtidig er det få oppgaver på mellomtrinnet der programmering blir brukt til utforskning i matematikk, selv om det er gode muligheter for å bruke programmering med relativt lite kunnskap om programmering, til å utforske matematiske begrep og idéer (Benton et al., 2016;17; Brennan & Resnick, 2012).

I resultatene fremkommer det at kombinasjonen av handlingen *følge* og begrepet *steg-for-steg instruksjoner* er de oppgavene med minst matematisk innhold. Selv om dette ikke er gjeldende for alle oppgavene med denne kombinasjonen. Som regel finner vi denne kombinasjonen i oppgaver som kommer rett etter et nytt begrep blir presentert i lærebøkene. Oppgavene er ofte rene programmeringsoppgaver og om de har et matematisk innhold er dette ofte for å lære seg ferdigheter innenfor programmering (Figur 3 og 7). Altså ikke oppgaver der programmering brukes som et verktøy for effektivisering eller utforskning. Lignende resultater kan vi se i studien til Kilhamn et al. (2021) som analyserte 32 undervisningstimer om programmering i matematikkfaget. Deres resultater viste at undervisning i kategori 1 og 2 (se kapittel 2.3.1) var den vanligste måten å undervise i programmering i matematikk i de undervisningsøktene de observerte.

Lærebøkene bruker mye stegvise instruksjoner i opplæringen av programmering (Figur 1, Tabell 1), oppgaver med mye stegvise instruksjoner finner vi ofte rett etter introduksjonen av nye begreper. Dette kan gi et bilde på at lærebokforfatterne mener at stegvise instruksjoner er nødvendig eller nyttig i sammenheng med opplæring i programmering. Verken rammeverket til Brennan & Resnick (2012) eller rammeverket til Benton et al. (2016) inneholder eller løfter frem stegvise instruksjoner i programmeringssammenheng. Som nevnt tidligere kan dette ses i sammenheng med at matematikkfaget er gitt hovedansvaret for opplæring i programmering (Forstrøm & Kaufmann, 2018). Selv om dette kun kan ses på som en observasjon og ikke et tydelig funn, finner vi igjen dette snevre synet på opplæring i programmering i andre studier (Kilhamn et al., 2021). Opplæring i programmering blir ofte sett i sammenheng med mye steg-for-steg instruksjoner, noe som kan bidra til å gi lærere og elever et snevert syn på potensialet til AT (Benton et al., 2016). Det understrekes at dette kan hemme lærere å se verdien programmering har for å utvikle elevenes algoritmiske tenkning (Benton et al., 2016).

5.1.2 Introduksjon av nye begreper

Alle lærebøkene presenterer forklaringer og/eller eksempeloppgaver når de introduserer nye begreper innenfor programmering som variabler, betingelser, løkker, funksjoner o.l. Elever med lite tidligere kunnskap om programmering kan ha utfordringer med å lære seg slike begreper, spesielt begreper som algoritme og funksjoner, som har ulike betydninger i matematikk- og programmeringssammenheng (Grover et al., 2015). Læreverkene er tydelige i denne begrepsforklaringen, og beskriver skillene mellom begrepene på tvers av de to fagområdene. Et eksempel er beskrivelsen av funksjoner i Multi-serien; "I programmering er en funksjon en gruppe med koder som alltid utføres sammen... I dagliglivet er knyte sko en tilsvarende funksjon. Den består av disse operasjonene:″ (Alseth et al., 2021, s 90) Der de viser en illustrasjon på å knytte sko med tekst til hvert bilde. Og skriver under illustrasjonen "hver gang vi tar på oss sko med lisser, kan vi tenke på funksjonen "knyte sko". Vi trenger ikke tenke på alle operasjonene som inngår″ (Alseth et al., 2021, s 90).

Samtidig som det er viktig at læreboken inkluderer en beskrivelse for hva funksjoner i programmering er, kan det være utfordringer knyttet til hvordan det er presentert. Tidligere i samme lærebokserie beskrives en algoritme i programmering slik; "... En algoritme er en samling operasjoner. De må gjennomføres i riktig rekkefølge." (Alseth et al., 2021, s 86). Også her blir forklaringen støttet opp med en illustrasjon, i dette tilfelle det å pusse tenner, med forklaringer under hvert bilde. Måten lærebøkene presenterer begrepene algoritme og funksjon er svært like. Dette kan potensielt føre til begrepsforvirring, ved at forskjellene mellom disse to begrepene ikke blir tydelig beskrevet (Grover et al., 2015; Haraldsrud et al., 2020).

5.1.3 Ensidig fokus

Noe av det mest overraskende funnet i analyseprosessen var at enkelte begreper og ferdigheter som er fundamentale for læring i programmering, var overraskende lite inkludert i lærebøkene. Den mest sentrale av disse var feilsøking (debugging), som er vektlagt i begge rammeverkene som analyseverktøyet er inspirert av (Benton et al., 2016;17; Brennan & Resnick, 2012). I flere av oppgavene i lærebøkene kan det argumenteres for at elevene må bedrive aktiv feilsøking når de arbeider med å løse problemer og lage programmer. Samtidig er det overraskende at lærebøkene ikke introduserer feilsøking som en viktig del av det å

programmere, slik de introduserer andre begreper innenfor programmering. Elevene er garantert å møte på feil i programmene de lager i sammenheng med oppgavene i lærebøkene, men likevel blir dette i liten grad diskutert i lærebøkene. Og da spesielt hva elevene skal gjøre i møte med feilmeldinger.

Elevene skal lære seg å lære gjennom aktiviteter med prøving og feiling, og systematisk feilsøking (Sevik et al., 2016; Utdanningsdirektoratet, 2020b). I arbeid med programmering vil elevene møte på feilmeldinger, og her vil kunnskap og erfaringer om hvordan de kan eller skal løse den aktuelle feilmeldingen være viktig. Vi ser som sagt ut i fra resultatene at lærebøkene sjeldent legger opp til aktiviteter og oppgaver der elevene får opplæring i hva de skal/kan gjøre i møte med feilkoder (Tabell 1, Figur 1). Ferdigheter som vil være sentrale for elevene i møte med feilmeldinger blir i liten grad tatt opp i lærebøkene som for eksempel; å lese feilmeldinger for å undersøke hvilken type feilmelding det er, vite hvor du kan søke for å lete etter løsninger og hvilke grep du kan ta for å finne feilen selv (Bueie, 2019; Haraldsrud et al., 2020). Om vi ser vi tilbake på et av de fire argumentene i kompetansepakken til Utdanningsdirektoratet som omhandler *eksperimentering, prøving og feiling*, argumenteres det for at nettopp denne erfaringen med prøving og umiddelbare feilmeldinger kan føre til større utholdenhet hos elevene i matematikkfaget (Utdanningsdirektoratet, 2020b).

5.1.4 Utvidelse av oppgavene

Oppgavene i lærebøkene har, som beskrevet i resultat kapittelet 4.1, et potensiale når det kommer til å inkludere handlingen *forklare* og *forestille seg*. Ved å stille krav til at elevene i større grad må forklare, enten ovenfor seg selv eller sammen med andre, kan det bidra til økt forståelse for begrepene innenfor AT (Benton et al., 2016; Brennan & Resnick, 2012). Dette kan være med på å utvikle elevenes ferdigheter og kunnskaper innen programmering slik at de i større grad kan bruke programmering som et verktøy for å effektivisere og utforske matematikk (Kilhamn et al., 2021). I tillegg vil også oppgaver der elevene må forklare innholdet i kodelinjer, eller i programmet som en helhet, kunne bidra til at elevene får erfaringer med flere representasjonsformer i matematikken. Dette kan igjen bidra til å legge til rette for dybdelæring i matematikk (Haraldsrud et al., 2020; Tellefsen, 2018).

Videre vil flere oppgaver som legger opp til at elevene må *forestille seg* kunne bidra til å styrke elevenes algoritmiske tenkning. Benton et al. (2016) beskriver evnen til å forestille seg i sammenheng med programmering som noe fundamentalt, det å se for seg et potensielt utfall

før du prøver programmet kan bidra til økt forståelse innenfor programmering og matematikk. Arbeid med programmering bygger opp elevenes evne til refleksjon og intuitive kunnskap, som kan sette i gang fruktbare refleksjoner rundt idéen eller begrepet som utforskes (Benton et al, 2016; Sevik et al., 2016; Papert, 1980). Ved at elevene får erfaring med å forutse eller forestille seg mulige utfall kan elevene få et mer bevisst forhold til det å planlegge og bryte ned oppgaver eller problemer som de skal løse. Og muligens se og gjennomføre endringer før de tester programmet, som kan bidra til elevene opplever programmering som et mer effektiv verktøy for utforskning (Benton et al., 2016). Selv om det er relativt få oppgaver som inneholder disse handlingene, og enda sjeldnere en kombinasjon av disse handlingene, kan det være at lærerveiledningene inneholder viktige poenger og grep som ikke er inkludert i analysen (Kongelf, 2019).

5.1.5 Overgang fra tekst til blokk

Haraldsrud et al. (2020) beskriver et scenario mange lærere vil møte i overgangen fra blokk- til tekstbasert programmering, spesielt i overgangen til ungdomstrinnet. Selv om elevene har opparbeidet seg kunnskap og kjennskap til begreper som *betingelser* og *løkker* i Scratch eller Trinket, vil overgangen potensielt sett bli stor for mange. I følge Haraldsrud et al. (2020) vil den største, og muligens første, utfordringen elever og lærere kommer til å møte er det å skrive riktig. Når elevene arbeider med blokkbasert programmering er det lite de kan gjøre feil, blokker som ikke passer sammen vil ikke heller henge sammen, og riktig bruk av ord og kommandoer, parenteser og kolon er heller ikke en bekymring. Anbefalingene til Haraldsrud et al. (2020) i denne overgangsfasen er todelt. Den ene anbefalingen er at elevene lager programmer, eller bruker tidligere prosjekter, i Scratch eller Trinket som de skriver om til Python. Den andre anbefalingen er å fokusere på gruppearbeid, og at læreren identifiserer elever som har erfaring med tekstprogrammering fra før.

Resultatene i kapittel 4.2 viser at det er varierende i hvilken grad lærebøkene vektlegger denne overgangen. I Figur 5 ser vi et eksempel på en oppgave der elevene blir oppfordret til å reflektere og diskutere forskjeller og likheter mellom blokk- og tekstbasert programmering. Enkelte lærebøker (Matemagisk-serien) benytter seg som nevnt tidligere av blokk programmeringsverktøyet Trinket. En sentral funksjon i Trinket er funksjonen der brukeren kan veksle mellom blokk- og tekstbasert programmering. Denne funksjonen kan potensielt gjøre overgangen lettere for elevene ved at de kan observere og reflektere over likheter og ulikheter mellom programmeringsspråkene, i likhet med det Haraldsrud et al. (2020) løfter

frem med å bruke tidligere prosjekter i Scratch/Trinket og oversette det til tekstbaserte programmeringsspråk. Jeg registrerte ingen oppgaver der denne funksjonen ble introdusert eller forklart. Jeg har, som nevnt tidligere, ikke sett på lærerveiledningene til lærebøkene i denne oppgaven. Lærerveiledningene kan potensielt inneholde betraktninger og råd til læreren i overgangen fra blokk til tekst, i tillegg kan læreren bruke sin pedagogiske kunnskap til å justere oppgaver hentet fra lærebøkene (Kongelf, 2019).

5.2 Forskningsspørsmål 2: På hvilken måte legger programmeringsoppgavene til rette for å få frem eller skape sammenhenger mellom programmering og matematikk?

Forskningsspørsmål 2 dreier seg om å undersøke hvordan oppgavene i lærebøkene legger til rette for at elevene kan se sammenhenger mellom programmering og matematikk. I Tabell 2 (kapittel 4.2.2) ser vi oversikten av hvilke matematiske begreper som er identifisert i de ulike oppgavene. Tabellen viser at det er en relativt jevn fordeling mellom de ulike matematiske begrepene, der *rotasjon* og *aritmetiske begreper* skiller seg ut som henholdsvis minst og mest tilstedeværende i oppgavene. I tillegg ser vi at *statistikk og sannsynlighet* ikke er identifisert på mellomtrinnet, dette kan skyldes fraværet av lærebøker på 7.trinn i utvalget, da kompetansemålet etter 7.trinn kobler programmering til statistikk. Tabellen viser også at det er et ganske stort utvalg oppgaver uten et matematisk innhold (22%, 5-7 og 15%, 8-10). Det er altså færre oppgaver uten matematisk innhold i lærebøkene på ungdomstrinnet, noe som potensielt kan forklares med at oppgavene på mellomtrinnet i større grad tar høyde for at programmering er nytt for elevene.

5.2.1 Hvordan de ulike handlingene legger til rette for å få frem sammenhenger

Ut i fra mine data inneholder store deler av oppgavene i lærebøkene handlingen *følge* og *forme og skape*, og ofte en kombinasjon av disse. Det er et begrenset utvalg av oppgaver som inneholder handlingene *forestille seg, feilsøke, finne regel* og *forklare*. Disse handlingene blir også sjeldent kombinert. Selv om disse handlingene blir løftet frem som viktige aspekter for å kunne lære og øve på AT (Benton et al., 2016;17; Wing, 2017). Jeg har observert er at det i oppgavene ligger et potensiale for å kunne inkludere én eller flere av disse handlingene i oppgavene, noe som kan bidra til å både styrke elevenes AT og skape bro mellom programmering og matematikk (Benton et al., 2016;17). Det Kongelf (2019) skriver om oppgavens andre fase i lærebøker vil imidlertid spille en rolle for elevenes læringsprosess.

Lærere med kunnskap om hvordan undervise programmering i matematikk vil kunne identifisere oppgaver som kan utvides ved å legge til spørsmål eller lignende, og ta dette i bruk i undervisningssammenheng. Men dette krever lærere med kompetanse i og om programmering i matematikk, og ser vi på studien til Kilhamn et al. (2021) var det få lærere knyttet programmering opp mot spesifikke matematiske begreper.

Samtidig er det oppgaver i lærebøkene der disse handlingene blir inkludert i oppgavene, som kan bidra til at elevene får brukt programmering som et verktøy for å utforske matematiske begreper. Altså i tråd med kategori 4 i studien til Kilhamn et al. (2021), der elevene effektivt og raskt kan se og gjøre endringer for å utforske matematiske begreper. Et eksempel på dette er i Figur 13, der elevene effektivt kan eksperimentere og utforske egenskaper ved geometriske figurer og få en forståelse for dette gjennom prøving og feiling. I oppgaven (Figur 13) er det gitt spesifikke instruksjoner for hvilke endringer som skal foretas, selv om endingene ser ut til å være bevisst valgt ut kunne det blitt gitt en deloppgave der elevene selv gjør endringer. Ved å gi elevene muligheten til selv gjøre endringer kan de få mer erfaring med å bruke programmering som et verktøy til å utforske matematiske begreper, og potensielt oppdage sammenhenger med det matematiske begrepet (Benton et al., 2016). Resultatene viser at 20% av programmeringsoppgavene på mellomtrinnet inneholder geometriske begreper (Tabell 2). Oppgavene handler som regel om å tegne geometriske figurer ved hjelp av blokkprogrammering slik som i Figur 13, men da ofte uten deloppgaver som vi ser i dette eksempelet. På mellomtrinnet skal, i følge kompetansemålene i læreplanen, disse geometriske figurene være kjent for elevene (Kunnskapsdepartementet, 2019). I slike oppgaver vil matematikken fungere som kontekst for å lære seg programmering, eller som repetisjon for elevene.

Flere oppgaver med handlingene *forestille seg og forklare* vil også kunne bidra til at arbeid med algoritmer i programmering vil skape tydeligere koblinger til matematikkfaget. Ved å for eksempel stille spørsmål som i oppgaven i Figur 13, i flere og andre typer oppgaver; "Hva skjer om vi endrer variablene?", "Med hvilke betingelser eller verdier i variablene vil algoritmen produsere det vi ønsker?". Slike spørsmål kan bidra til å hjelpe elevene til å se sammenhenger mellom programmering og matematikken (Bråting & Kilhamn, 2021; Benton et al., 2016;17). Målet med programmering i matematikk, slik det er uttrykt i læreplanen, er ikke nødvendigvis å lære seg programmering i seg selv. Men å bruke programmering som et

verktøy for å utforske matematiske idéer og begreper, til å gjøre effektive beregninger og presentere og vurdere datamaterialer (Utdanningsdirektoratet, 2020b).

5.2.2 Læreplanen og programmeringsinnholdet

Når det kommer til hvilke matematiske begreper lærebøkene knytter programmering til baserer de seg som regel på de respektive kompetansemålene til de ulike klassetrinnene der det er knyttet til et spesifikt tema. Enkelte kompetansemål, 6, 7 og 9.trinn, knytter programmering til spesifikke matematiske begreper som geometri, og statistikk og sannsynlighet. De øvrige kompetansemålene åpner mer opp for at lærebøkene selv kan bestemme hvordan de vil knytte programmeringsinnholdet til matematikken. Lærebøkene varierer hvordan de velger å plassere programmeringsinnholdet, bøkene på åttende og tiendetrinn har som regel egne kapitler som handler om programmering, samt at de i tillegg har innspill av programmeringsoppgaver knyttet til flere matematiske begreper. På mellomtrinnet derimot har lærebøkene kun egne programmeringskapiteler, ofte underkapitler i kapitelet om algebra. Som jeg har diskutert tidligere er det varierende i hvilken grad programmeringsoppgavene bidrar til at elevene får erfaringer der programmering kan brukes som et verktøy for å utforske matematiske begreper eller effektivisere beregninger (Kilhamn et al., 2021). Det er oppgaver, spesielt med handlingen *følge*, der det matematiske innholdet enten ikke er tilstede eller der det matematiske innholdet er kontekst for å lære seg begreper og ferdigheter innenfor programmering.

Videre viser resultatene at oppgavene innenfor kategorien *finne regel* ofte legger til rette for at elevene, i tråd med læreplanen, kan utforske egenskaper med tall, figurer og se etter mønstre for å finne sammenhenger (Figur 6, 11 og 12). I disse oppgavene er det relativt lite instruksjoner (spesielt Figur 11) der elevene bruker programmering til å utforske matematiske begreper, og der de må oversette det matematiske til et programmeringsspråk. Elevene får dermed øving i å skifte mellom representasjonsformer, og kan se sammenhenger mellom matematikken og programmeringen. Oppgavene legger også til rette for at elevene får øving i AT, ved at de får øving i å dele opp problemer, abstrahere og modulere problemet, og drive med systematisk testing (Forstrøm & Kaufmann, 2018; Kunnskapsdepartementet, 2019; Sevik et al., 2016; Wing, 2017). Noe som igjen kan gi elevene erfaringer med programmering som et verktøy for å effektivisere beregninger og utforske matematiske begreper (Kilhamn et al., 2021). Programmering kan bidra til at elevene kan eksperimentere, prøve og feile på en mer effektiv måte, med umiddelbare tilbakemeldinger, noe som i følge kompetansepakken for

programmering i matematikk kan føre til økt utholdenhet hos elevene (Utdanningsdirektoratet, 2020b). Det å visualisere og dynamisk representere abstrakte konsepter er til stor hjelp i matematikkundervisning (Egeberg, Hultin & Berge, 2016). Oppgaver med kombinasjonen *finne regel* og *forklare* ville kunne gi elevene erfaringer innenfor argumentasjon i matematikken (Benton et al., 2016). Ved at elevene underveis blir bedt om å argumentere for fremgangsmåten og løsninger i arbeid med programmering i matematikk, kan også sammenhengene komme tydeligere frem (Benton et al., 2016). Denne typen oppgaver var det svært få av i lærebøkene.

Enkelte lærebokserier (Maximum 8-10 og Matemagisk 8-10) fokuserer mer på å bevisstgjøre elevene på at de i møte med ulike problemer bør foreta en vurdering for hvilke digitale verktøy som er best egnet for å løse det gitte problemet. Ved at elevene får valg mellom å løse oppgaven med graftegner, regneark eller programmering. I enkelte oppgaver gis også instruksjonen om å løse oppgaven med to ulike digitale verktøy, for å så vurdere hvilket verktøy elevene opplevde som mest egnet. Slike oppgaver kan tenkes at vil bidra til at elevene selv, i møte med nye oppgaver, vil foreta kritiske vurderinger for hvilke verktøy som er mest egnet til det gitte problemet (Kilhamn et al., 2021; Kunnskapsdepartementet, 2019; Sevik et al., 2016). Elevene vil med denne erfaringen løse matematiske oppgaver ved hjelp av flere ulike representasjonsformer. Noe som er en av nøklene i sammenheng med dybdelæring i matematikk (Gilje et al., 2018; Haraldsrud et al., 2020; Utdanningsdirektoratet, 2019b).

5.2.3 Samarbeid og kommunikasjon

Handlingen *forklare* er relativt sjeldent inkludert i oppgaver med programmering, spesielt på mellomtrinnet (19/79, 5-7 og 81/197, 8-10, Figur 1). I rammeverket til Benton et al. (2016), the 5E´s, løftes handlingen *forklare* (Explain) frem som noe sentralt for å skape forståelse og øke motivasjonen til elevene i arbeidet med programmering i matematikk. For at programmering skal fungere som et effektivt verktøy for å utforske matematikk, er det nødvendig at elevene blir oppfordret til å diskutere og forklare hva de ulike kodelinjene eller blokkene representerer som deler og i en helhet (Bueie, 2019; Benton et al., 2016;17). Som igjen vil sette i gang en refleksjon der elevene kan identifisere hvilke deler av de matematiske idéene eller begrepene som er viktige, og for å opparbeide seg kunnskap i hvilke muligheter som ligger i programmering (Benton et al., 2016).

Videre la jeg merke til i analyseprosessen, uten at dette er kategorisert, at det i de fleste lærebokseriene er lagt opp til lite samarbeid i oppgavene knyttet til programmering. I læreplanen beskrives kommunikasjon og samarbeid som en viktig ferdighet i matematikk, samt i sammenheng med programmering og AT (Kunnskapsdepartementet, 2019; Lodi & Martini, 2021). Enkelte lærebøker (Maximum 8-10) legger i stor grad opp til samarbeid, ved at det i oppgavene presiseres at elevene skal arbeide i par eller grupper. I sammenheng med utvikling av ferdigheter innenfor AT kan vi se tilbake til en av de fire hovedkategoriene, *transversal skills*, som Lodi & Martini (2021) presenterte. Der samarbeid, refleksjon og kommunikasjon blir beskrevet som noe svært viktig i utviklingen av AT. Oppgaver slik som i Maximum 8-10 kan bidra til at elevene kan få god øving i å argumentere, lytte og snakke i matematikk, i tråd med læreplanen (Kunnskapsdepartementet, 2019).

I arbeid med programmering kreves det at man er presis i språket for at algoritmene og programmene skal fungere som ønsket (Haraldsrud et al., 2020). Rammeverket til Benton et al. (2016) the 5E´s inneholder handlingen Exchange, denne handlingen går ut på at elevene deler erfaringer og argumenterer for sine løsningsmetoder i arbeid med programmering. De mener også at dette kan ha en overføringsverdi til matematikkfaget ved at elevene gjennom erfaringer med programmering se hensikten med et presist og riktig matematisk språk. I tillegg vil ofte samarbeid som arbeidsform føre til at elevene blir eksponert for ulike løsningsmetoder eller kommer frem til ulike svar (Benton et al., 2016). I analyseverktøyet til Bråting & Kilhamn (2021) er ikke handlingen Exchange fra the 5E´s inkludert, og ble dermed ikke kategorisert i analyseprosessen.

6 Konklusjon

I denne masteroppgaven har jeg forsøkt å besvare de to forskningsspørsmålene, presentert innledningsvis. Før jeg går inn på å oppsummere de mest sentrale funnene ut i fra resultatene og diskusjonen vil jeg trekke frem at resultatene i denne studien først og fremst er gjeldende for dette utvalget av lærebøker. I tillegg vil ikke nødvendigvis elevene møte alle oppgavene som er inkludert i studien slik de er presentert. Lærerveiledningene, og læreren selv, kan justere oppgaver og aktiviteter når de blir brukt i undervisningssammenheng, slik Kongelf (2019) påpeker. Lærebøkene er likevel en sentral del av lærerens praksis, i planlegging og gjennomføring av undervisning, og kan dermed ha innvirkning på elevenes læring (Fan et al., 2013; Kongelf, 2019). Undersøkelser viser også at oppgaver fra lærebøker ofte blir brukt til elevenes individuelle arbeid i matematikkundervisning i norske klasserom (Lepik et al., 2015). Samtidig, vet jeg ikke sikkert i hvilken grad lærere bruker programmeringsinnholdet i trykte lærebøker, spesielt med tanke på det voksende utvalget av læringsmateriale som er tilgjengelig digitalt.

6.1 Forskningsspørsmål 1

For å svare på forskningsspørsmålet er bøkenes programmeringsinnhold, med fokus på oppgavene, analysert ut i fra hvilke handlinger og begreper som inkluderes. Generelt sett er det mest karakteristiske ved lærebøkene at en stor andel av oppgavene inneholder handlingene *følge* og *forme og skape*. Kombinasjonen av disse og oppgaver som baserer seg utelukkende på handlingen *følge*, er de oppgavetypene som legger minst til rette for læring i matematikk gjennom programmering. Slike oppgaver vil kunne plasseres i kategori 1 og 2 ifra studien til Kilhamn et al. (2021), der undervisningen enten dreier seg utelukkende om opplæring i programmering eller at matematikk kun fungerer som en kontekst for å lære seg programmering. Likevel, er det viktig å se dette i sammenheng med at programmering i matematikk er nytt for både lærer og elever. Og at det kan være nødvendig med slike oppgaver for å opparbeide essensiell kunnskap i programmering for å kunne bruke det som et effektivt verktøy i matematikkfaget.

Resultatene viser at oppgavene i mindre grad inneholder handlingene *forestille seg, feilsøke, finne regel* og *forklare*, og at det er forskjeller mellom 5-7 og 8-10.trinn, spesielt mellom handlingene *forestille seg* og *forklare* (Figur 1 og 2). I følge teorien burde oppgavene inneholde flere handlinger, da det kan bidra til at elevene kan se sammenhenger mellom

programmering og matematikkfaget, skape bedre forståelse for programmering og AT som en problemløsningsmetode og for å utforske matematikk ved hjelp av programmering (Benton et al., 2016;17). Mer i tråd med kategori 3 og 4 fra Kilhamn et al., (2021) sin studie. Lærebøkene inneholder et utvalg oppgaver med én eller flere av disse handlingene, men utvalget av slike oppgaver er relativt lite. Det er dermed viktig at læreren er bevisst på og innehar nødvendig kunnskap og kompetanse i matematikk og programmering for å kunne velge ut, justere eller legge til elementer i oppgavene når de brukes i undervisningssammenheng. Lærerens rolle og kompetanse i undervisning av programmering i matematikk er altså sentral. Det ser ut til at oppgaver som inkluderer flere av disse handlingene i større grad legger til rette for læring i matematikk der programmering fungerer som et verktøy for å effektivisere beregninger og utforske. Mer i tråd med slik programmering er ment i matematikkfaget (Sanne et al., 2016; Sevik et al., 2016; Utdanningsdirektoratet, 2020a, 2020b).

Når det kommer til AT-begrepene viser resultatene etter analysen at de fire mest fremtredende begrepene i oppgavene er *steg-for-steg instruksjoner*, *algoritme, løkker* og *kode*, med en relativt jevn fordeling av de nevnte. Begrepet *kode* blir i større grad inkludert i oppgavene på ungdomstrinnet, da det på mellomtrinnet er flere oppgaver med analog-programmering, og på ungdomstrinnet nesten utelukkende oppgaver der elevene må bruke et programmeringsverktøy. Et relativt stort innslag av *steg-for-steg instruksjoner* kan ses i sammenheng med handlingen *følge*, som ble diskutert ovenfor. Stegvise instruksjoner løftes ikke frem i noen av rammeverkene som analyseverktøyet baserer seg på (Benton et al., 2016;17; Brennan & Resnick, 2012; Bråting & Kilhamn, 2021).

Resultatene viser at elevene møter på og blir presentert for en rekke sentrale begreper innenfor programmering i lærebøkene. Samtidig er fordelingen av disse er noe ujevn (Tabell 1). Begrepene *regel, betingelser* og spesielt *feilsøking* er i mindre grad inkludert i oppgavene enn de øvrige. Lærebøkene ser ut til å fokusere mer på enkelte begreper, og utelukker andre. Noe av det som var mest overraskende er i hvor liten grad *feilsøking* ble løftet frem i lærebøkene både som begrep og som handling. Til tross for at det å feilsøke, i sammenheng med programmering, blir løftet frem som noe svært sentralt i litteraturen (Benton et al., 2016;17; Brennan & Resnick, 2012; Bueie, 2019; Haraldsrud et al., 2020). Elevene bør få opplæring og erfaring i det å håndtere feilmeldinger. I tillegg til at feilsøking som arbeidsmetode kan bidra til at elevene får øving i algoritmisk tenkning og se sammenhenger,

gjennom å systematisk gå gjennom programkodene og gjøre nødvendige endringer (Bueie, 2019; Haraldsrud et al., 2020; Sevik et al., 2016).

6.2 Forskningsspørsmål 2

Resultatene viser at de aller fleste programmeringsoppgavene i lærebøkene har et matematisk innhold (78%, 5-7, 82%, 8-10). Ut i fra resultatene og diskusjonen ser det ut til at det varierer i hvilken grad oppgavene har et matematisk innhold som utgjør konteksten for å lære seg programmering, eller om programmering brukes som et verktøy for å effektivisere beregninger i matematikk og/eller for å utforske matematikken. Det ser ut til at oppgavene som har et matematisk innhold som regel bruker det matematiske innholdet som kontekst for å lære programmering, der det matematiske innholdet allerede er kjent for elevene på det gitte klassetrinnet. Enkelte oppgaver innenfor geometri på mellomtrinnet legger opp til at elevene kan bruke programmering til å utforske matematikk, dette ofte i sammenheng med handlingene *forklare eller forestille seg*. Men slike oppgaver er det relativt lite av, både på mellom- og ungdomstrinnet. Oppgaver som inneholder én eller flere av handlingene *finne regel, forklare, forestille seg* og *feilsøke* ser ut til å oftere fremme sammenhenger mellom programmering og matematikk (Benton et al., 2016;17). Videre viser diskusjonen og resultatene at oppgaver med handlingen *følge,* og i kombinasjon med *forme og* skape, ser ut til å i minst grad fremme sammenhenger mellom programmering og matematikk. Disse handlingene er de som er registrert oftest i analysen av oppgavene.

Målet med programmering i matematikk er ikke at elevene skal blir «programmerere», men at de skal kunne bruke programmering som et verktøy for å lære og få en dypere forståelse for matematikk. Flere oppgaver med handlingene *forestille seg, feilsøke, finne regel* og *forklare,* som diskutert i 5.1 og 5.2, kan potensielt bidra til å oppnå dette i større grad. Lærebøkene, spesielt på ungdomstrinnet, ser ut til å legge noe mer til rette for at elevene vil få erfaringer med å løse oppgaver med hjelp av flere ulike digitale verktøy. Og i tillegg få valg om hvilke de vil bruke til en gitt oppgave, eller at de løser samme oppgave med to ulike digitale verktøy og vurderer hvilke som var mest effektive. Denne erfaringen kan bidra til at elevene evner å kunne ta kritiske vurderinger for hvilke digitale verktøy som er hensiktsmessig å bruke i møte med ulike problemer. Og i tillegg tilegne seg en forståelse for hvordan digitale hjelpemidler fungerer og er strukturert. Lærebøkene inneholder oppgaver som bidrar til å skape sammenhenger mellom programmering og matematikk, spesielt på ungdomstrinnet, men disse

oppgavene er det relativt få av i det store bildet. Oppgavene inneholder ofte et matematisk innhold for å lære seg programmering, fremfor å bruke programmering til å utforske matematikken.

6.3 Kommentar til studien og veien videre

Datagrunnlaget i denne studien består utelukkende av programmeringsoppgaver, og disse programmeringsoppgavene er de oppgavene som er identifisert av meg som forsker. Det kan ikke utelukkes at enkelte oppgaver ikke er inkludert i studien, i tillegg er ikke lærerveiledningene og alle lærebøkenes digitale læringsressurser inkludert i analysen. Dette kan medføre at studiens resultater kan avvike noe fra hvordan lærebokseriene totalt sett vektlegger programmering, og hvordan elevene møter programmeringsinnholdet til lærebøkene i skolen. Til tross for dette, kan resultatene bidra til å vise hva som karakteriserer programmeringsinnholdet i lærebøkene, hvordan nye lærebøker har valgt å implementere programmering og på hvilken måte de lykkes å få frem sammenhenger mellom programmering og matematikken.

Denne studien ser kun på lærebøkenes fremstilling og hvordan oppgavenes fremstilling kan påvirke elevene, og vil derfor ikke kunne gi et bilde på hvilket læringsutbytte bøkene gir. I tillegg til at lærerveiledningene kan gi instruksjoner og variasjoner for bruken av oppgavene, vil også læreren kunne justere og endre oppgavene. Analyseverktøyet og mine tolkninger, som igjen er påvirket av relevant teori, har bidratt til kategoriseringen av oppgavene. Andre som gjennomfører lærebokanalyser, innenfor samme tema og som benytter seg av samme analyseverktøy, vil kunne tolke oppgavene annerledes. Siden resultatene påvirkes av mine tolkninger har det vært viktig for meg å tydeliggjøre valg som er tatt for å gjøre forskningen mest mulig transparent.

I videre forskning innen programmering i lærebøker ville det vært interessant å gjennomført en analyse som inkluderer lærerveiledningene og de digitale læringsressursene, for å få dypere innsikt i programmeringsinnholdet. I tillegg ville det vært interessant å intervjue og observere hvordan lærere og elever bruker lærebøkene, samt å undersøke hva som kjennetegner oppgavetyper som legger til rette for å bruke programmering som et verktøy for å effektivisere beregninger og utforske i matematikkfaget.

7 Litteratur

Alseth, B., Alseth, B., Røsseland, M., Arnås, A.-C. & Nordberg, G. (2021). *Multi 5B, 3.utg. : matematikk for barnetrinnet : Elevbok* (Elevbok, 3. utgave. utg.). Oslo: Gyldendal.

Alseth, B., Arnås, A.-C., Nordberg, G. & Røsseland, M. (2021). *Multi 6B, 3. utg. : matematikk for barnetrinnet: Elevbok* (Bokmål[utgave], 3. utgave. utg.). Oslo: Gyldendal.

Balanskat, A., Engelhardt, K., & Ferrari, A. (2017). The integration of Computational Thinking (CT) across school curricula in Europe. Hentet fra

Benton, L., Hoyles, C., Kalas, I., & Noss, R. (2016). Building mathematical knowledge with programming: Insights from the ScratchMaths project. In A. Sipitakiat & N. Tutiyaphuengprasert (Eds.), Constructionism in Action 2016, Conference Proceedings (pp. 25–32). Suksapattana Foundation.

Benton, L., Hoyles, C., Kalas, I., & Noss, R. (2017). Bridging primary programming and mathematics: Some findings of design research in England. Digital Experiences in Mathematics Education, 3(2), 115–138.

Biggs, N. L. (1990). Discrete Mathematics. Oxford: Oxford University Press.

Bratberg, Ø. (2021). *Tekstanalyse for samfunnsvitere*. Cappelen Damm akademisk.

Cappelen Damm. (u.å.). *Matematikk 8-10 fra Cappelen Damm*. Hentet 4. april 2022 fra https://www.cappelendammundervisning.no/verk/matematikk- 8-10-fra-cappelen-damm-153429

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In Proceedings of the 2012 Annual Meeting of the American Educational Research Association (pp. 1–25). American Educational Research Association.

Bråting, K. & Kilhamn, C. (2021). The Integration of Programming in Swedish School Mathematics: Investigating Elementary Mathematics Textbooks. *Scandinavian Journal of Educational Research, ahead-of-print*(ahead-of-print), 1-16. doi: 10.1080/00313831.2021.1897879

Bocconi, S., Chioccariello, A. & Earp, J. (2018). The Nordic approach to introducing Computational Thinking and programming in compulsory education. Rapport skrevet for Nordic@ BETT2018 Steering Group. doi: 10.17471/54007

Bueie, H. (2019). *Programmering for matematikklærere*. Oslo: Universitetsforlaget.

Christoffersen, L. & Johannesen. (2012). Forskningsmetode for lærerutdanningene. Abstrakt forlag.

Cohen, L., Manion, L. & Morrison, K. (2017). *Research Methods in Education*. London: London: Taylor & Francis Group.

Crick, T. (2017). Computing Education: An Overview of Research in the Field (s. 40)

Dahlum, S. (2018). Algoritme. Store Norske Leksikon. Hentet 6. februar.2022 fra https://snl.no/algoritme

Drijvers, P., Kieran, C., Mariotti, M.-A., Ainley, J., Andresen, M., Chan, Y. C., . . . Meagher, M. (2009). Integrating Technology into Mathematics Education: Theoretical Perspectives. In C. Hoyles & J.-B. Lagrange (Eds.), Mathematics Education and Technology-Rethinking the Terrain (pp. 89-132). Boston, MA: Boston, MA: Springer US.

Egeberg, G., Hultin, H. & Berge, O. (2016). Monitor skole 2016 – Skolens digitale tilstand.Senter for IKT I utdanningen. Hentet 21.04.22 fra https://www.udir.no/globalassets/filer/tallogforskning/rapporter/2016/monitor_2016_bm_-_2._utgave.pdf

Fan, L. (2013). Textbook research as scientific research: towards a common ground on issues and methods of research on mathematics textbooks. ZDM – The international Journal on Mathematics Education, 45, 765-777.

Fan, L., Zhu, Y. & Miao, Z. (2013). Textbook research in mathematics education: Development status and directions. ZDM – The international Journal on Mathematics Education, 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), 79-96.

Feurzeig, W., Papert, S., Bloom, M., Grant, R., & Solomon, C. (1970). Programming-languages as a conceptual framework for teaching mathematics (s. 5). Hentet fra https://dl.acm.org/doi/pdf/10.1145/965754.965757

Forsström, S. E. & Kaufmann, O. T. (2018). A literature review exploring the use of

programming in mathematics education. International Journal of Learning, Teaching and Educational Research, 17(12), 8-32. doi: https://doi.org/10.26803/ijlter.17.12.2

Fløe, Ellen Egeland. (2020). Programmering I LK20. Tangenten – Tidsskrift for matematikkundervisning 32(1), 3-9. Hentet fra http://www.caspar.no/tangenten/2021/tangenten_1_2021_Flo.pdf

Gilje, Ø., Ingulfsen, L., Dolonen, J. A., Furberg, A., Rasmussen, I., Kluge, A., Knain, E., Mørch, A., Naalsund, M. & Skarpaas, K. G. (2016). *Med ARK&APP: Bruk av læremidler og ressurser for læring på tvers av arbeidsformer* (ARK&APP). Universitetet i Oslo. https://www.udir.no/globalassets/filer/tall-og-forskning/forskningsrapporter/arkapp_syntese_endelig_til_trykk.pdf

Gilje, Ø., Landfald, Ø. & Ludvigsen, S. (2018). Dybdelæring – historisk bakgrunn og teoretiske tilnærminger. Bedre Skole -tidsskrift for lærere og skoleledere, 30(4), 22-27.

Gjøvik, Ø., & Torkildsen, H. A. (2019). Algoritmisk tenkning. Tangenten - Tidsskrift for matematikkundervisning(3), 31-37. Hentet fra http://www.caspar.no/tangenten/2019/Tangenten%203%202019%20Gj%C3%B8vik%20Torkildsen.pdf?fbclid=IwAR3lLaDiUI54BzIZEA7nNyqsYGn6xNj1uU31-J6gpc2rJNS6aYDOPUJTAxA

Grover, S., & Pea, R. (2013). Computational thinking in K—12: A review of the state of the field. Educational Researcher, 42(1), 38–43. doi: 10.3102/0013189X12463051

Grover, S., Pea, R. & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. Computer Science Education, 25(2), 199–237.

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

Gulbrandsen, J. E., Gulbrandsen, J. E., Løchsen, R., Måleng, K., Olsen, V. S., Skogstad, H. & Mathisen, L. (2020a). *Matematikk 5 fra Cappelen Damm : Grunnbok* (Grunnbok, Bokmål[utgave], utgave 1. utg.). Oslo: Cappelen Damm.

Gulbrandsen, J. E., Gulbrandsen, J. E., Løchsen, R., Måleng, K., Olsen, V. S., Skogstad, H. & Mathisen, L. (2020b). *Matematikk 6 fra Cappelen Damm : Grunnbok* (Grunnbok, Bokmål[utgave], utgave 1. utg.). Oslo: Cappelen Damm.

Grønmo, L. S., Bergem, O. K., Kjærnsli, M., Lie, S. & Turmo, A. (2004). Hva i all verden har skjedd i realfagene? Norske elevers prestasjoner i matematikk og naturfag i TIMSS 2003. Oslo: Universitetet i Oslo.

Grønmo, L.S. & Onstad, T. (red.) (2009). Tegn til bedring: norske elevers prestasjoner i matematikk og naturfag i TIMSS 2007. Oslo: Unipub.

Haraldsrud, A. D., Sveinsson, H. A., & Løvold, H. H. (2020). Programmering i skolen. Universitetsforlaget

Hjardar, E., Hjardar, E., Pedersen, J.-E. & Sidorowicz, M. (2020). *Matematikk 9 fra Cappelen Damm : Grunnbok* (Grunnbok, Bokmål[utgave], utgave 1. utg.). Oslo: Cappelen Damm.

Hjardar, E., Pedersen, J.-E. & Sidorowicz, M. (2020). *Matematikk 8 fra Cappelen Damm : Grunnbok* (Grunnbok, Bokmål[utgave], utgave 1. utg.). Oslo: Cappelen Damm.

Hiebert, J., Gallimore R., Garnier H., Givvin K. B., Hollingsworth H., Jacobs J., … Stigler, J. (2003). Teaching mathematics in seven countries: Results from the TIMSS 1999 Video Study (NCES 2003-013, U.S. Department of Education). Washington, DC: National Center for Education Statistics.

Hsu, T.-C., Chang, S.-C. & Hung, Y.-T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers and education, 126*, 296-310. doi: 10.1016/j.compedu.2018.07.004

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

Johannessen, A., Christoffersen, L. & Tufte, P. A. (2010). *Introduksjon til samfunnsvitenskapelig metode* (4. utg. utg.). Oslo: Abstrakt.

Johansen, A.-K. (2020, 11.07.2020). Programmering vil bli en utfordring for lærere. *Forskning.no*. Hentet fra https://forskning.no/barn-og-ungdom-hogskolen-i-ostfold-matematikk/programmering-vil-bli-en-utfordring-for-laerere/1711838

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

Kaufmann, O. T., Stenseth, B. & Holone, H. (2018). Programmering i matematikkundervisningen. I A. Norstein & F. Haara (Red.), Matematikkundervisning i en digital verden, (s. 73-96) Oslo: Cappelen Damm Akademisk.

Kilhamn, C., Bråting, K. & Rolandsson, L. (2021). *Teachers' arguments for including programming in mathematics education. Hentet fra https://www.diva-portal.org/smash/get/diva2:1541426/FULLTEXT01.pdf*

Kilhamn, C., Rolandsson, L. & Bråting, K. (2021). Programmering i svensk skolmatematik: Programming in Swedish school mathematics. *LUMAT, 9*(1), 283. doi: 10.31129/LUMAT.9.2.1457

Kirke- og undervisningsdepartementet (1987) Mønsterplan for grunnskolen. Aschehoug, Oslo.

Kirke-, utdannings- og forskningsdepartementet (1996) Læreplanverket for den 10-årige grunnskolen. Aschehoug, Oslo.

Klette, K. (2003). Klasserommets praksisformer etter Reform 97. Oslo, Norge: Pedagogisk Tidsskrift, 91(4), 344-358

Kongelf, T. R., Universitetet i Agder Fakultet for teknologi og, r. & Universitetet i, A (2019). *Matematisk innhold og matematiske metoder i lærebøker brukt på ungdomstrinnet i Norge : gullgruve eller fallgruve for utvikling av matematisk kompetanse i problemløsning og algebra?* (241). Universitetet i Agder, Fakultet for teknologi og realfag, Kristiansand

Kongsnes, A. L., Kongsnes, A. L., Wallace, A. K., Ødegaard, E., Sortland, K. & Hvattum, M. M. (2020). *Matemagisk 9* (Bokmål[utgave], 1. utgave. utg.). Oslo: Aschehoug undervisning.

Kongsnes, A. L. & Wallace, A. K. (2021). *Matemagisk 10* (Bokmål[utgave], 1. utgave. utg.). Oslo: Aschehoug undervisning.

Kongsnes, A. L., Wallace, A. K., Moholt, A., Ødegaard, E., Sortland, K. & Hvattum, M. (2020). *Matemagisk 8* (Bokmål[utgave], 1. utgave. utg.). Oslo: Aschehoug undervisning.

Krumsvik, R. J., Jones, L. Ø. & Røkenes, F. M. (2019). *Kvalitativ metode i lærarutdanninga*. Fagbokforlaget.

Kunnskapsdepartementet. (2019). Læreplan i matematikk 1.-10. trinn (MAT01-05). Fastsatt som forskrift. Læreplanverket for Kunnskapsløftet 2020. https://www.udir.no/lk20/mat01-05?lang=nno

Kvale, S., Brinkmann, S., Anderssen, T. M. & Rygge, J. (2009). *Det kvalitative forskningsintervju* (InterView[s] learning the craft of qualitative research interviewing, 2. utg. utg.). Oslo: Gyldendal akademisk.

Larke, L. R. (2019). Agentic neglect: Teachers as gatekeepers of England's national computing curriculum. British Journal of Educational Technology, 50(3), 1137–1150. https://doi.org/10.1111/bjet.12744

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J. & Werner, L. (2011). Computational thinking for youth in practice. Acm Inroads, 2(1), 32-37.

Lepik, M., Grevholm, B. & Viholainen, A. (2015). Using textbooks in mathematics classrooms – the teachers' view. Nordic Studies in Mathematics Education, 20(3-4), 129-156.

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

Mannila, L. (2018). Digitally competent schools: Teacher expectations when introducing digital competence in Finnish basic education. Seminar.Net: Media, Technology & Life-Long Learning, 14(2), 1–15.

Mayer, R. E., Dyck, J. L., & Vilberg, W. (1986). Learning to program and learning to think: What's the connection? Communications of the ACM, 29(7), 605–610. https://doi.org/10.1145/6138.6142

MIT. (2021). Scratch—Imagine, Program, Share. Hentet 20. mars 2022, fra https://scratch.mit.edu/

Mozelius, P., Ulfenborg, M., & Persson, N. (2019). Teatcher attitudes towards the integration of programming in middle school mathematics. Presentert på INTED 2019, Valencia, Spania.

Neuman, J., Hemmi, K., Ryve, A., & Wiberg, M. (2015). Mathematics textbooks' impact on classroom instruction: Examining the views of 278 Swedish teachers. In H. Silfverberg, T. Kärki, & M. Hannula (Eds.), Proceedings of the 7th Nordic Conference on Mathematics Education, NORMA 14 (pp. 215–224). University of Turku.

Opetushallitus. (2014). Perusopetuksen opetussuunnitelman perusteet 2014. Hentet 03.03.2022 fra https://www.oph.fi/sites/default/files/documents/perusopetuksen_opetussuunnitelman_perusteet_2014.pdf

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

Papert, S. (1993). The children's machine: Rethinking school in the age of the computer. ERIC.

Pea, R. D., & Kurland, D. M. (1984). On the cognitive effects of learning computer

programming. New ideas in psychology, 2(2), 137-168.

Raen, K. M., Kongsnes, A. L., Lang-Ree, H. L., Nyhus, G., Ødegaard, E., Sortland, K., . . . Frati, S. V. (2020). *Matemagisk 5B : Grunnbok* (Grunnbok, 2. utgave, bokmål[utgave]. utg.). Oslo: Aschehoug undervisning.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., & Kafai, Y. (2009). Scratch: Programming for all. Communications of the ACM, 52(11), 60–67.

Rezat, S. & Strässer, R. (2017). Methodological issues and challenges in research on mathematics textbooks. I B. Grevholm (Red.), *Mathematics textbooks, their content, use and influences: Reaserch in Nordic and Baltic countries* (s. 495-514). Cappelen Damm Akademisk.

Sanne, A., Berge, O., Bungum, B., Jørgensen, E. C., Kluge, A., Kristensen, T. E. & Voll, L. O. (2016). Teknologi og programmering for alle - En faggjennomgang med forslag til endringer i grunnopplæringen. Oslo: Utdanningsdirektoratet. Hentet fra https://www.udir.no/tallog-forskning/finn-forskning/rapporter/teknologi-og-programmering-for-alle

Sevik, Kristine et al. (2016). Programmering i skolen. Notat fra Senter for IKT i utdanningen. Hentet fra https://www.udir.no/globalassets/filer/programmering_i_skolen.pdf

Sentance, S., & Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspective. Education and Information Technologies, 22(2), 469–495. https://doi.org/10.1007/s10639-016-9482-0

Shimizu, Y., Kaur, B., Huang, R. & Clark, D. (2010). The role of mathematical tasks in different cultures. I Y. Shimizu, B. Kaur, R. Huang & D. Clark (red.) Mathematical tasks in classrooms around the world. The learner's perspective study (s. 1-14). Rotterdam, Nederland: Sense Publishers.

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

Skolverket. (2019). Läroplan (Lgr11) för grundskolan samt för förskoleklassen och fritidshemmet. Hentet 17. mars 2022, fra https://www.skolverket.se/publikationsserier/styrdokument/2019/laroplan-for-grundskolan-forskoleklassen-och-fritidshemmet-reviderad-2019

Stenseth, B., Kaufmann, O. T., & Forsström, S. E. (2019). Programmering og matematikk. Tangenten - Tidsskrift for matematikkundervisning(2), 7-12. Hentet fra

http://www.caspar.no/tangenten/2019/tangenten%202%202019%20Stenseth%20et%20al.pdf

Stigberg, H., & Stigberg, S. (2019). Teaching programming and mathematics in practice: A case study from a Swedish primary school. Policy Futures in Education, 18(4), 483–496. https://doi.org/10.1177/1478210319894785

Tellefsen, C. W. (2018). Realfaglig programmering. Hentet 8. februar 2022, fra https://www.uv.uio.no/forskning/satsinger/fiks/kunnskapsbase/realfagligprogrammering/realfaglig-programmering_tellefsen.pdf

Tofteberg, G. N., Alseth, B., Stedøy, I., Tangen, J., Tofteberg, G. N. & Bråthe, L. T. (2021). *Maximum 9, 2. utg. : matematikk for ungdomstrinnet* (Maximum 9 : matematikk, Bokmål[utgave], 2. utgave. utg.). Oslo: Gyldendal.

Tofteberg, G. N., Tangen, J., Bråthe, L. T., Stedøy, I. & Alseth, B. (2020). *Maximum 8, 2. utg.: matematikk for ungdomstrinnet* (2. utgave, bokmål[utgave]. utg.). Oslo: Gyldendal.

Tofteberg, G. N., Tofteberg, G. N., Stedøy, I., Tangen, J. & Bråthe, L. T. (2021). *Maximum 10, 2. utg. : matematikk for ungdomstrinnet* (Maximum 10 : matematikk, Bokmål[utgave], 2. utgave. utg.). Oslo: Gyldendal.

Torkildsen, H. A., & Gjøvik, Ø. (2019). Algoritmisk tenkning. Tangenten: Tidsskrift for matematikkundervisning. H. A., 7

Utdanningsdirektoratet. (2019a). Algoritmisk tenkning. Hentet fra https://www.udir.no/kvalitet-og-kompetanse/profesjonsfaglig-digital-kompetanse/algoritmisk-tenkning/

Utdanningsdirektoratet. (2019b). Dybdelæring. Hentet fra https://www.udir.no/laring-og-trivsel/dybdelaring/

Utdanningsdirektoratet. (2020a). Hva er nytt i matematikk?. Hentet fra https://www.udir.no/laring-og-trivsel/lareplanverket/fagspesifikk-stotte/nytt-i-fagene/hva-er-nytt-i-matematikk/

Utdanningsdirektoratet. (2020b). Kompetansepakken for lærere – Programmering og algoritmisk tenkning. Hentet fra https://bibsys.instructure.com/courses/387

Waite, J. (2018). Literature review: pedagogy in teaching. Hentet fra https://royalsociety.org/- /media/policy/projects/computing-education/literature-review-pedagogy-inteaching.pdf

Wing, J. (2006). Computational Thinking. Communications of the ACM, 49, 33-35.

doi:10.1145/1118178.1118215
Wing, J. M. (2011). Research Notebook: Computational Thinking—What and Why? 8.

Vedlegg 1

Noteringsark til analyse – Handlinger

Bok ->	Multi 6B Totalt 11 oppgaver	Matematikk 5-7 Totalt 9 oppgaver	Maximum 10 Totalt 17 oppgaver	Matemagisk 8 Totalt 66 oppgaver	Matemagisk 9 Totalt 25 oppgaver
Følge	9	9	14	41	16
Finne regel	4	0	4	10	5
Feilsøke	0	1	4	6	1
Forme og skape	3	9	16	51	22
Forklare	6	4	7	27	15
Forestille seg	3	1	3	6	3

Vedlegg 2

Noteringsark til analyse – Handlinger

Bok ->	Multi 5B Totalt 26 oppgaver	Matematikk 8-10 Totalt 8 oppgaver	Maximum 8 Totalt 27 oppgaver	Maximum 9 Totalt 25 oppgaver	Matemagisk 10 Totalt 29 oppgaver	Matemagisk 5B Totalt 33 oppgaver
Følge	13	6	24	8	16	16
Finne regel	4	1	5	1	5	9
Feilsøke	1	2	6	3	1	4
Forme og skape	12	8	25	24	23	24
Forklare	2	5	5	5	17	7
Forestille seg	10	1	7	7	3	20

Vedlegg 3

Forklaring til noteringsark – Begreper

AT-Begreper	Kode
Steg-for-steg instruksjoner	A
Algoritme	B
Løkke, iterasjon, repetering	C
Regel	D
Kode	E
Betingelser	F
Bug/feil, feilsøking	G

Matematiske-Begreper	Kode
Mønster (inkludert tallmønster)	1
Geometriske konsept	2
Aritmetiske konsept	3
Rotasjon	4
Koordinatplan	5
Statistikk og sannsynlighet	6
Uten matematisk innhold	7

Vedlegg 4

Noteringsark til analyse - Begreper **Bok:**

Oppgave	AT- begreper	Matematiske-begreper

Vedlegg 5

Oversikt - fordeling av begreper etter analyse i de ulike lærebøkene

Bok	AT- begreper	Matematiske-begreper
Multi 5B	**A-19, B-17, C-6, D-4, E-12, F-11, G-1**	**1-5, 2-0, 3-12, 4-0, 5-1, 6-0, 7-8**
Matemagisk 5B	**A-14, B-19, C-21, D-5, E-14, F-4, G-4**	**1-4, 2-14, 3-7, 4-2, 5-0, 6-0, 7-8**
Multi 6B	**A-8, B-7, C-6, D-4, E-7, F-3, G-1**	**1-0, 2-1, 3-5, 4-0, 5-6, 6-0, 7-0**
Matematikk + CDU 5-7	**A-4, B-1, C-2, D-4, E-0, F-0, G-0**	**1-0, 2-1, 3-2, 4-0, 5-4, 6-0, 7-2**
Maximum 8	**A-15, B-15, C-4, D-6, E-18, F-3, G-5**	**1-5, 2-5, 3-12, 4-0, 5-4, 6-0, 7-2**
Maximum 9	**A-7, B-16, C-10, D-5, E-19, F-6, G-5**	**1-0, 2-9, 3-0, 4-1, 5-0, 6-14, 7-2**
Maximum 10	**A-13, B-11, C-4, D-8, E-16, F-8, G-4**	**1-1, 2-1, 3-6, 4-0, 5-11, 6-0, 7-1**
Matematikk + CDU 8-10	**A-4, B-5, C-3, D-1, E-3, F-5, G-3**	**1-0, 2-3, 3-3, 4-0, 5-2, 6-0, 7-3**
Matemagisk 8	**A-38, B-36, C-25, D-7, E-64, F-17, G-4**	**1-17, 2-3, 3-25, 4-0, 5-3, 6-0, 7-21**
Matemagisk 9	**A-12, B-16, C-15, D-1, E-25, F-15, G-2**	**1-3, 2-5, 3-5, 4-0, 5-0, 6-15, 7-1**
Matemagisk 10	**A-15, B-16, C-14, D-2, E-26, F-12, G-1**	**1-1, 2-1, 3-21, 4-0, 5-4, 6-0, 7-0**