Masteroppgave

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

Silje Hagerup Gurigard

Print("Programmering i matematikk")

En kvalitativ innholdsanalyse av programmeringsoppgaver i norske lærebøker for matematikk på 10.trinn med fokus på programmering som verktøy i matematikk.

Masteroppgave i matematikkdidaktikk 5. – 10.trinn
Veileder: Solomon Abedom Tesfamicael
Mai 2024

NTNU
Kunnskap for en bedre verden

Silje Hagerup Gurigard

Print("Programmering i matematikk")

En kvalitativ innholdsanalyse av programmeringsoppgaver i norske lærebøker for matematikk på 10.trinn med fokus på programmering som verktøy i matematikk.

Masteroppgave i matematikkdidaktikk 5. – 10.trinn
Veileder: Solomon Abedom Tesfamicael
Mai 2024

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

NTNU
Kunnskap for en bedre verden

Sammendrag

Print(“Programmering i matematikk”)

...

Med innføringen av kompetanseløftet, LK20, ble programmering integrert i flere fag, spesielt i matematikk. Denne masteroppgaven undersøker hvordan programmering er implementert i lærebøker for matematikk på 10. trinn. Studien fokuserer spesielt på implementeringen av kompetansemålet for 10.trinn, der elevene skal bruke programmering som et verktøy for å utforske matematiske egenskaper og sammenhenger. Forskningsspørsmålet er: Hvordan har norske lærebøker for matematikk på 10.trinn programmering som et verktøy i matematikk?

Tidligere forskning på læringsutbytte av programmering i matematikkundervisning har vist varierende resultater. Likevel er det mangel på forskning som undersøker hvordan programmering skal anvendes og hvilket læringsutbytte programmering har i matematikkundervisning. Hensikten med denne studien er å få innsikt i hvordan programmering blir brukt i matematikk, og løfte diskusjonen om i hvilken grad programmering kan være et verktøy for å lære matematikk.

Forskningsmetoden som er benyttet er en kvalitativ innholdsanalyse av programmeringsinnholdet i lærebøkene Maximum 10, Matemagisk 10 og Matematikk 10. Datamaterialet ble analysert ved å bruke en tilpasset versjon av analyseskjema til Charalambous et al. (2010). Analyseskjema består av en horisontal og en vertikal analyse. I den horisontale analysen ble omfanget og struktureringen av programmeringsoppgavene analysert. Den vertikale analysen var teoridrevet, og baserte seg på to rammeverk; rammeverk for kobling forholdet mellom programmering og matematikk utviklet av Kilhamn et al. (2021), og PRIMM (Coleman, 2021), for å analysere hva oppgaven ba elevene gjøre. De to rammeverkene, sammen med teori om algoritmisk tenkning og programmering i skolen, vil danne det teoretiske grunnlaget for studien.

Den horisontale analysen viser at det er stor forskjell på vektleggingen av programmering i lærebøkene. Matemagisk 10 har dobbelt så mange oppgaver med programmering som Maximum 10 og Matematikk 10. Til tross for ulik mengde er den dominerende praksisen å benytte programmering i sammenheng med funksjoner. Den vertikale analysen viser at programmeringsoppgavene er koblet til matematikk, og at den dominerende praksisen er å bruke programmering som et verktøy for å effektivisere matematiske handlinger. Det er dog få oppgaver som kategoriseres som programmering som verktøy for å utforske matematikk, noe som er interessant fordi kompetansemålet for matematikk etter 10. trinn tilsier at elevene skal utforske matematiske egenskaper og sammenhenger gjennom programmering.

Nøkkelord: *Programmering i matematikk, programmering, lærebøker i matematikk, lærebokanalyse, Algoritmisk tenkning.*

Abstract

Print("Programming mathematics")

...

With the introduction of the Knowledge Promotion Reform (2020), programming has been integrated into several subjects, especially mathematics. This master's thesis investigates how programming is implemented in mathematics textbooks for the 10th grade. The study focuses on the competence aim that requires pupils to use programming as a tool to explore mathematical properties and connections, and how this is potentially implemented in textbooks. The research question is: How have Norwegian mathematics textbooks for the 10th grade implemented programming with a focus on programming as a tool?

Research on the learning outcomes of programming in mathematics education has shown mixed results. However, there is still a lack of research on how to use programming in mathematics education and its effects. This study's purpose is to gain insight into how programming is used in mathematics and to promote the discussion of whether programming can be a medium to learn mathematics.

The research method used was a qualitative content analysis of the programming content in the textbooks *Maximum 10, Matemagisk 10,* and *Matematikk 10*. The data were analyzed using an adapted version of the analysis framework by Charalambous et al. (2010). The framework consists of a horizontal and a vertical analysis. In the horizontal analysis, the scope and structure of the programming tasks were analyzed. The vertical analysis was theory-driven and based on two frameworks: the framework for connecting programming and mathematics developed by Kilhamn et al. (2021), and PRIMM (Coleman, 2021).

These two frameworks, along with theories of computational thinking and programming in education, form the theoretical foundation for the study.

The horizontal analysis implies significant differences in the emphasis on programming in the textbooks. *Matemagisk 10* has twice as many programming tasks as *Maximum 10* and *Matematikk 10*. Despite the varying quantities, the dominant practice is to use programming in relation to functions. The vertical analysis indicates that the programming tasks are connected to mathematics and that the dominant practice is to use programming as a tool to make mathematical operations efficient. However, there are few tasks categorized as using programming as a tool to explore mathematics. This finding is intriguing as it does not align with the competence aim for mathematics after the 10th grade.

Keywords: *Programming in mathematics, programming, mathematics textbooks, textbook analysis, computational thinking.*

Forord

```
def takk(relasjon):
    if relasjon == "veileder":
      print("Tusen takk til Solomon for god veiledning, og ikke minst takk
      for tålmodigheten du har vist ovenfor en rastløs student som meg.")
    elif relasjon == "roomie" :
      print("Tusen takk til verdens beste kollektiv, SCHOT! Takk for at
      dere opprettholder mitt sosiale liv, og at jeg etter en lang dag på
      lesesal kan komme til et hjem fylt med dårlige vitser, musikk og gode
      samtaler<3")
    elif relasjon == "medstudent":
      print("Tusen takk for alle faglige og ufaglige samtaler på
      masterplass, spesielt takk til kaffehjørnet. Takk for all latter,
      sosiale happenings og ikke minst is-pauser! Dere har gjort
      masterskriving en god del morsommere!")
    elif relasjon == "forelder":
      print("Tusen takk for at dere tar telefonen hver gang jeg ringer, og
      at dere stiller opp til sparring av master, korrekturlesing og ren
      utblåsning.")
    else:
      print("Tusen takk for at du er du!")
relasjon = input("Skriv inn din relasjon:")
relasjon = relasjon.lower()
takk(relasjon)
print("Silje Hagerup Gurigard, Trondheim, Mai 2024")
```

Innholdsfortegnelse

Figurer

Tabeller

1 Introduksjon

1.1 Bakgrunn for tema

Teknologi har blitt en så integrert del av samfunnet vårt, at det er vesentlig å ha en grunnleggende teknologisk forståelse for å holde tritt med sosiale utfordringer og digital utvikling (Kafai & Burke, 2013;Haraldsrud et al.,2020). I Opplæringsloven (2006, §1-1) står det at elevene skal utvikle kunnskap, ferdigheter og holdninger for å mestre eget liv, og delta i arbeid og fellesskap i samfunnet. En av skolens oppgaver er dermed å gi elevene ferdigheter i og forståelse for teknologi. Den teknologiske utviklingen kom tydelig frem i Kunnskapsløftet LK20 der programmering fikk en sentral rolle, spesielt i matematikk (Utdanningsdirektoratet, 2019; Sevik et al.,2016; Vinnervik & Bungum, 2022).

Programmering i skolesammenheng er ikke nytt. En av de første pionerne innen dette, Seymour Papert (1980), utviklet programmeringsverktøyet Logo allerede på 80-tallet. Han mente at programmering hadde et stort potensialt der yngre elever kunne lære matematikk, og da spesielt utforske geometri (Papert, 1980). Gjennom bruk av Logo i undervisningen, forsket han på elevers læring av matematikk gjennom programmering og fant at Logo gjorde det lettere for elevene å anvende algoritmer og å utforske og drive problemløsning (Papert, 1980). Dette startet en programmeringsbølge, og flere europeiske skoler innførte programmering på 1980-tallet.

På 1990-tallet stoppet imidlertid bølgen med en økende skepsis til det faglige læringsutbytte ved bruk av programmering. Flere studier slet med å finne en korrelasjon mellom bruk av programmering i matematikk og økt forståelse eller ferdigheter i matematikk. Debatten endte med at flere nasjoner trakk programmering ut av skolen (Kafai & Burke, 2013).

20 år senere er programmering tilbake igjen ved flere europeiske skoler; i England er programmering enn del av et eget fag kalt «Computing», i Estland er programmering innført i skolen gjennom det tverrfaglige tema «Teknologi og innovasjon» (Sevik et al.,2016). I Finland, Norge og Sverige er programmering implementert i fagene gjennom læreplanen, og matematikk er faget som har fått hovedoppgaven med å lære elevene programmering (Vinnervik & Bungum, 2022; Sevik et al, 2016). Selv om programmering har blitt implementert i skolesystemer over hele verden, er det fremdeles en betydelig mangel på forskning som undersøker den mest effektive anvendelsen av programmering i grunnskolen, både på nasjonalt og internasjonalt nivå (Crick, 2017).

Eksisterende forskning peker på manglende kompetanse i programmering hos lærere som en av de største utfordringene ved integrering av programmering i skolen (Kilhamn et al., 2021; Manilla, 2018). Vinnervik(2023) sin studie understreker dette, der samtlige lærere i studien hevder at de har manglende kompetanse om og forståelse for programmering. Etter fem år på lærerutdanningen har jeg, til tross for hovedfag i matematikk, og fordypning i naturfag, hatt lite undervisning i programmering, og den undervisningen jeg har hatt har vært gjennom tilleggsemnet «Programmering i skolen». Dette fremhever to faktorer; lærerutdanningens innsats for å transformere, fremme og praktisere profesjonell digital kompetanse er begrenset, og at til tross for lik utdannelse,

er det forskjeller i læreres kompetanse basert på hvor interessert læreren er (Aagård et al.,2022). Læreres grad av engasjement i det digitale feltet gjenspeiles i undervisningen, og dette engasjementet korrelerer med deres digitale kompetanse, som igjen påvirker kvaliteten på elevers opplæring (Aagård et al., 2022).

Manglende kompetanse hos lærere gjør at de i større grad baserer undervisningen på læreverkene (Haggarty & Pepin, 2002). Ved lite kompetanse gir lærere enda større tillit til læringsressurser og de blir primærkilden i undervisningen. Dette skjer gjennom at elevene arbeider med oppgaver hentet fra lærebøkene, og læreren støtter seg på og henter eksempler fra lærebøkene (Johansson, 2006). Lærebøkene har derfor en sentral rolle i undervisningspraksis, og studier viser at tematikk som ikke er inkludert i lærebøkene, heller ikke trekkes inn i undervisningen (Johansson, 2006).

Institutt for Lærerutdanning (ILU) ved NTNU arrangerte våren 2024 en minikonferanse om programmering i lærerutdanningen og skolen (PILS). På konferansen holdt leder for forskergruppa «læring og undervisning av matematikk med digitale verktøy», Øistein Gjøvik (2024), et innlegg om statusen til programmering i matematikk. Et av perspektivene han trakk frem var om det faktisk er et potensial for læring i matematikk gjennom programmering. Han mente at elevene absolutt kan bruke programmering for å øve på anvendelse av matematikk, men at det er et ambisiøst og tidkrevende mål å skulle utforske og lære ny matematikk gjennom programmering. Diskusjonen fra 90-tallet er dermed fremdeles relevant, og i denne oppgaven skal jeg se på hvordan programmering blir brukt i matematikkundervisning, om programmering er et middel eller et mål i seg selv.

1.2 Forskningsspørsmålet og formålet

Det er flere aspekter ved programmering i skolesammenheng som er av interesse for forskning. Denne studien fokuserer spesifikt på hvordan programmering kan anvendes som et verktøy for å fremme læring av matematikk, heller enn å undersøke det tekniske aspektet ved programmering isolert sett. Jeg vil undersøke implementeringen av kompetansemålet for 10. trinn: «utforske matematiske egenskaper og sammenhenger ved å bruke programmering» (Kunnskapsdepartementet, 2019). Kompetansemålet vektlegger utforskning av matematiske egenskaper og sammenhenger ved bruk av programmering. Dette er særlig interessant fordi kompetansemålet tydeliggjør programmeringens rolle som et middel for å utforske matematikk.

For å undersøke implementeringen av kompetansemålet, vil jeg analysere norske lærebøker i matematikk for 10. trinn. Lærebøker er en primær kilde for mange lærere når det gjelder matematikkundervisning, spesielt i områder der lærere har manglende kompetanse (Rezat & Strässer, 2017; Johansson, 2006). På bakgrunn av lærebøkenes sentrale rolle i undervisningen kategoriseres lærebøker som den potensielt implementerte læreplanen (Valverde et al.,2002). Derfor kan lærebøker være en god indikator på hvordan kompetansemålet er blitt implementert i praksis.

Det overordnede forskningsspørsmålet er: **Hvordan har norske lærebøker for matematikk på 10.trinn implementert programmering med fokus på programmering som verktøy i matematikk?**

For å besvare forskningsspørsmålet vil jeg benytte to rammeverk. Et rammeverk utviklet av Kilhamn et al. (2021), som fokuserer på forholdet mellom programmering og matematikk. I tillegg vil jeg benytte PRIMM (Coleman, 2021), for å få innsikt i hvordan

elevene blir bedt om å bruke programmering. Disse rammeverkene, sammen med teori om algoritmisk tenkning og programmering i skolen, vil danne det teoretiske grunnlaget for studien.

1.3 Forskerens bakgrunn

Denne studien er kvalitativ og det er derfor viktig at jeg som forsker presenterer min bakgrunn og motivasjon, for å øke studiens validitet (Creswell & Creswell, 2018).

«Silje Gurigard er fire år og dataprodusent» slik lød ingressen til TV2 nyhetene 21. september 2002, se figur 1.1. Bakgrunnen var at barnehagen jeg gikk deltok i et prosjekt med Høgskolen i Oslo (i dag: OsloMet), der vi utviklet dataspill. Mitt favorittspill var søppelspillet der vi hadde tatt bilder av søppel og redigert det inn i barnehagens uteområde. Spillet gikk ut på å sortere søpla i riktig søppelkasse. Om jeg lærte noe av å produsere spill i barnehagen som har vært styrende for mine senere utdanningsvalg, har jeg ikke grunnlag for å si noe om, utover at jeg i innslaget sier at det var ”kjempegøy”.

Figur 1.1: Bilder hentet fra nyhetssendingen på TV2 21.09.2002 (TV2, 2002).

To år senere begynte jeg på skolen, og jeg gikk gjennom 10 års grunnskole uten å produsere flere program, dermed ble jeg nok også tidenes yngste pensjonerte dataprodusent.

Da jeg begynte på NTNU og fikk muligheten for flere teknologiske fag, kom i alle fall interessen tilbake I tillegg til det ordinære studieløpet ved lærerutdanningen har tatt fagene «Programmering i Skolen», og «Informasjonsteknologi, grunnkurs». Gjennom fagene så jeg et stort potensial i bruk av programmering i skolen, og ønsket å undersøke programmering i matematikk nærmere. Ved å skrive masteroppgaven om programmering ønsket jeg å rette fokus mot forskningsfeltet, og øke egen kompetanse både didaktisk og faglig. For i en tid der diskusjonen om bruk av skjem i skolesammenheng står høyt på agendaen hos politikerne, savner jeg mer didaktisk forskning om bruk av digitale verktøy, spesielt programmering, i skolen.

1.4 Oppgavens struktur

Studien er strukturert i seks kapitler. Første kapittel inneholder bakgrunn for studien og presentasjon av forskningsspørsmålet. Kapittel 2 fokuserer på teori og tar for seg programmering i europeisk skolesammenheng, ulike former for programmering, samt programmeringens rolle i matematikkundervisningen. Videre utforskes LK20-læreplanen og læreboks teori. I kapittel 3 blir det gjort rede for det vitenskapelige paradigmet og forskningsmetoden. Dette inkluderer presentasjon av utvalget, analysemetoder og en

diskusjon om oppgavens validitet, reliabilitet og etiske betraktninger. Kapittel 4 presenterer funnene fra analysen, som deretter drøftes opp mot teorien i kapittel 5. Avslutningsvis oppsummeres studiens hovedfunn i lys av forskningsspørsmålet i kapittel 6.0.

2 Teori

I dette kapittelet presenteres relevant teori og tidligere forskning. Kapittelet er strukturert slik at det begynner med en innføring i programmering i skolesammenheng med spesiell vekt på algoritmisk tenkning. Videre blir ulike former for programmering, både analog og digital, presentert. Deretter programmering i undervisningspraksis ved PRIMM-modellen, og i matematikkfaget med rammeverk for relasjon mellom matematikk og programmering, samt programmering i læreplanen og utforskende matematikk. Til slutt, diskuteres lærebøkene og deres rolle i undervisning.

2.1 Programmering i skolesammenheng

Programmering er ikke bare et verktøy for læring, men en måte å tenke på (Forsstöm & Kaufmann, 2018; Papert, 1980). I datautvikling, og dagligtale er programmering et begrep som ofte blir snevret inn til koding. Programmering inkluderer ofte kode, men skal ikke begrenses til kode. Vinnervik (2022) studerte definisjoner av programmering i skolesammenheng, og så at felles for definisjonene er at programmering er en flertrinns prosess der en løser et problem ved hjelp av en datamaskin. Ifølge Gjøvik og Torkildsen (2019) handler programmering om å løse et problem ved hjelp av en algoritme eller et program. Det vil si forarbeid, planlegge, skrive, tegne, bearbeide, undersøke, eksperimentere, revurdere, og generalisere slik at algoritmen er i stand til å løse en hel klasse av problemer. Koding er mer spesifikt enn programmering, og er selve aktiviteten der en transkriberer en algoritme eller en oppskrift inn i et programmeringsspråk, på fagspråket kalt transponering (Vinnervik, 2022; Kallia et al.,2021). I følge Scherer et al. er programmering er en prosess med utvikling og implementering av instruksjonssett for at en datamaskin skal kunne utføre en gitt oppgave, løse et problem og gi menneskelig samspill (2020). Programmering innebærer derfor minst tre elementer; et problem, en datamaskin og en problemløser, som sammen utgjør en prosess i flere steg.

2.1.1 Algoritmisk tenkning

Algoritmisk tenkning er en ferdighet som har fått økt oppmerksomhet de siste årene (Chytas et al.,2024). *Computational thinking*, eller Algoritmisk tenkning som Utdanningsdirektoratet (Udir) bruker, er en grunnleggende ferdighet som kan bli praktisert gjennom programmering (Kaufmann & Stenseth, 2019). Oversettelsen av *Computational thinking* til Algoritmisk tenkning noe omdiskutert da det kan minne om det engelske begrepet Algorithmic thinking. Gjøvik og Torkildsen (2019) løser oversettelsesproblematikken ved å bruke algoritmebehandling for algoritmical thinking, og benytter algoritmisk tenkning som norsk oversettelse av begrepet *computational thinking*, på samme måte som Utdanningsdirektoratet (Udir). Begrepene algoritmisk tenkning og algoritmebehandling må ikke forveksles da algoritmebehandling er en snever del av algoritmisk tenkning.

Litteraturen er ikke enig om én definisjon av algoritmisk tenkning, likevel finnes det noen fellestrekk. Utdanningsdirektoratet definerer begrepet som "[..] å vurdere hvilke steg som skal til for å løse et problem, og å kunne bruke sin teknologiske kompetanse for å få en datamaskin til å løse (deler av) problemet. I dette ligger også en forståelse av hva slags problemer/oppgaver som kan løses med teknologi og hva som bør overlates til

mennesker. [...]" - (Udir, 2019). En slik forståelse av begrepet stemmer overens med litteraturen, der Algoritmisk tenkning både defineres som en tenkemåte og en problemløsningsstrategi (Bocconi et al.,2018). Algoritmisk tenkning er ifølge Kallia et al. (2018) «en aktivitet, ofte produktorientert, som ofte er assosiert med, men ikke begrenset til problemløsning». Algoritmisk tenkning kan ses på som en type problemløsning som inkluderer teknologi (Kaufmann & Stenseth, 2019; Kallia et al., 2021; Bocconi et al., 2018). Algoritmisk tenkning kjennetegnes derfor ved tre prosesser: problemløsning, kognitiv prosess og transposisjon. I problemløsning består algoritmisk tenkning av fem elementer; abstraksjon, dekomposisjon, algoritmer, evaluering og generalisering (Kallia et al., 2018). Med unntak av generalisering er fire av elementene i Utdanningsdirektoratet sin figur på algoritmisk tenkning, se figur 2.1. Den kognitive prosessen er en tankeprosess som inkluderer, men ikke begrenses til de fem elementene fra problemløsningen. Det siste kjennetegnet er transponering, som kobler algoritmisk tenkning til teknologi, ved å formulere en løsning på et problem slik at den kan overføres til et annet menneske, eller oftest til en maskin eller et program (Kallia et al., 2018).

I utdanningsforskning blir programmering ofte koblet til algoritmisk tenkning (Kafai & Burke, 2013). Koblingen tydeliggjøres i Kunnskapsløftet, LK20, som legger vekt på elevenes utvikling av algoritmisk tenkning (Sevik et al., 2016). Algoritmisk tenking er dermed en sentral del av programmering, og jeg har derfor valgt å utdype kjennetegnene til Den algoritmiske tenkeren(Udir, 2019a), se figur 2.1, og dens nøkkelbegrep og arbeidsmåter.

Figur 2.1 Den algoritmiske tenkeren hentet fra Utdanningsdirektoratet (2019a).

2.1.1.1 Nøkkelbegreper

Figur 2.1 tar for seg seks nøkkelbegrep for algoritmisk tenkning. Prosessen starter med å analysere problemet og planlegge prosessen for problemløsning (1). Nøkkelbegrep 2, algoritmer, handler om å lage regler og en trinnvis plan for å komme frem til en løsning. Algoritmisk tenkning handler om å dekomponere problemet til konkrete og håndterlige delproblemer som er løsbare (3). Det handler om å finne mønstre (4) og bruke disse til å lage en hensiktsmessig modell ved hjelp av matematisk språk og/eller modeller (5). Til

slutt skal løsningen evalueres (6) og tenkeren skal vurdere hvorvidt problemet er løst (Udir, 2019). Lignende nøkkelbegreper finnes hos Gjøvik og Torkildsen (2019), som beskriver kjennetegnene for algoritmisk tenkning ved; Abstraksjon, Generalisering, automatisering, dekomposisjon, algoritme-behandling, og automatisering. Noen av disse nøkkelbegrepene likner på Udir sine, mer abstraksjon, dekomposisjon, algoritmer, og generalisering som likner på mønster (Udir, 2019). Det som skiller seg ut i Gjøvik og Torkildsens (2019) definisjon er algoritme-behandling og automatisering. Algoritme-behandling minner om koding, og handler om å implementere algoritmen slik at en kan finne resultatene en vil ha. Dersom en benytter en maskin så vil automatisering være å få maskinen ti å kjøre programmet. Disse to stegene skiller algoritmisk tenkning fra vanlig problemløsning, og er sentrale for den teknologiske delen av algoritmisk tenkning, som vist i Udirs definisjon av begrepet (Udir, 2019)

2.1.1.2 Arbeidsmåter

Udir presenterer i figur 2.1 fem arbeidsmåter som kjennetegner algoritmisk tenkning. Første arbeidsmåte er å fikle, og handler om å utforske og eksperimentere. Det krever at eleven har en systematisk og analytisk tilnærming til prosessen, samtidig som eleven er nysgjerrig. Den neste er å skape, og handler om å designe og lage løsningen. Det krever at elevene er skapende og åpne for alternative løsninger. Feilsøking handler om å oppdage og korrigere feil. Til forskjell for matematiske prosesser er feil en naturlig og viktig del av algoritmisk tankeprosess, og det er viktig at Den algoritmiske tenkeren har en plan for hvordan oppdage og løse feil. Neste punkt handler om å holde ut, og feilsøkingsprosessen kan være lang. Da behøver man en kognitiv utholdenhet, slik at en ikke gir opp underveis (Udir, 2019). Videre trekker Udir frem samarbeid som metode. Det likner på sosiokulturell læringsteori, der det finnes en grense for hva en elev kan klare uten noe annen sosialinput. Det er derfor viktig for progresjonen at elevene deler og jobber sammen (Vygotsky, 1978; Udir, 2019).

2.1.1.3 Algoritmisk tenkning i programmering

Den algoritmiske tankemåten er essensiell i programvareutvikling og for teknologisk kompetanse generelt (Sevik et al., 2016). Arbeid med programmering kan bidra til å lære algoritmisk tenkning (Mayer et al., 1986; Scherer et al., 2020). De mener at elevene ved å lære seg programmering tilegner seg ferdigheter innen algoritmisk tenkning. Koblingen mellom algoritmisk tenkning og programmering kommer frem i Den algoritmiske tenkeren der elevene skal anvende algoritmer, for å designe program (Udir, 2019). Scherer et al. (2020) trekker frem tre andre nøkkelementer ved algoritmisk tenkning; konsepter, praksiser og perspektiver. Teknologiske konsepter (*competutional consepts*) omfatter programmeringsteknikk, som løkker og variabler. Teknologiske praksiser (*competutional practice)* handler om problemløsningsprosessen i programmering som testing og debugging, tilsvarende Udirs feilsøking. Teknologiske perspektiver (*competutional perspectives)* handler om elevens forståelse av prosessen og evne til å reflektere og stille seg kritisk til egen posisjon, interaksjonen med datamaskinen og hvorvidt teknologi er nødvendig for å løse problemet (Scherer et al., 2020).

En studie utført av Afre et al. (2019) viser at elever som engasjerer seg intensivt med programmering over en periode, opplever forbedret evne til dekomponering og abstraksjon på nivå med, eller bedre enn, elever som har gjennomgått 7 måneders standard undervisning. Kaufmann og Stenseth (2019) peker på essensen av algoritmisk tenkning i programmering som evnen til å dekomponere en løsningsstrategi (et av Udirs

nøkkelbegrep innen algoritmisk tenkning). Dekomposisjon i programmering, kommer til uttrykk ved at elevene må formulere kode og systematisere elementer som variabler, betingelser og løkker (Kaufmann & Stenseth, 2019).

En av arbeidsmetodene i Den algoritmiske tenkeren er samarbeid (Udir, 2019). I programmering anses samarbeid som en sentral del, og studier viser at par-programmering har stor effekt på elevenes læringsutbytte (Hanks, 2004; Crick, 2017). Ved å programmere i par eller i grupper stopper elevene sjeldnere opp i arbeidet, og progresjonen er dermed jevnere enn ved individuelt arbeid. Når det er sagt viser samme studie at elevene får større eierskap til programmet dersom de arbeider alene, og de kommer ofte fortere i gang med arbeidet (Hanks, 2004; Crick, 2017). Når elevene arbeider alene bruker de ofte medstudenter som ressurs, og på den måten får elevene en kombinasjon (Hanks, 2004; Crick, 2017). En sentral del av algoritmisk tenkning er videreutvikling ved å fikle og feilsøke. Et program er aldri perfekt den første gangen det kjøres enten fordi det møter på en syntaksfeil, logiske feil eller overflødig kode. Derfor er feilsøking en sentral del av programmering (Kaufmann & Stenseth, 2019; Vinnervik, 2023). Innen programmering er samarbeids- og delingskulturen stor. Kopiering av kode oppfattes ikke som juks, men som en mulighet for samarbeid, og videreutvikling med mål om å kunne løse mer komplekse problemer. En betingelse er at man gjør seg kjent med koden og forstår innholdet, før man anvender den (Kaufmann & Stenseth, 2019).

2.2 Analog og digital programmering

Algoritmisk tenkning og programmering kan implementere i skolen på flere måter. I denne delen presenteres digitale og analoge måter å programmere på. Digitalt gjennom tekst- og blokkprogrammering, og analogt gjennom instruksjoner og pseudokode (Berg, 2021).

2.2.1 Analog programmering

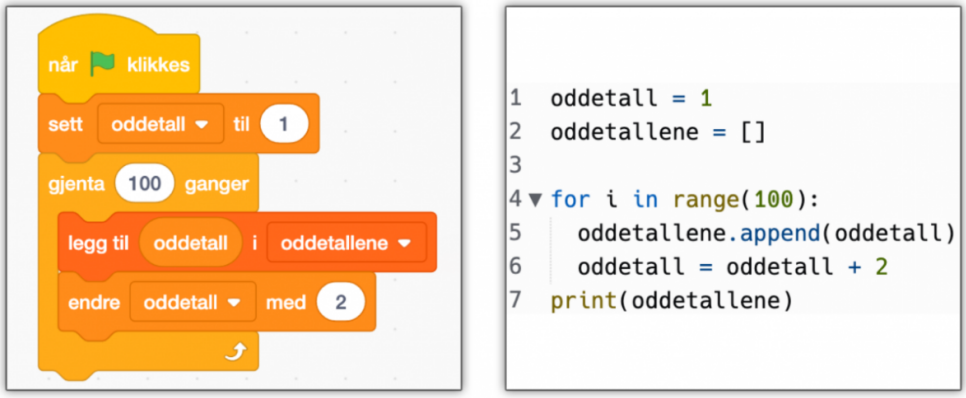
Analog programmering, også kjent som unplugged programmering, er en metode hvor elever engasjerer seg i programmeringsaktiviteter uten bruk av datamaskiner (Berg, 2021). Dette innebærer at elever utformer prosedyrer eller løsninger på problemer uten digital støtte (Berg, 2021). Typiske eksempler på analog programmering kan være å instruere en medelev til å utføre en konkret oppgave, følge en oppskrift/instruksjon, eller skissere et program gjennom flytskjema eller pseudo-kode (Berg, 2021; Looiet al., 2018). Berg fremhever flere fordeler med analog programmering, blant annet som bidrag til variasjon, trening av algoritmisk tenkning, økt motivasjon, og overføringsverdi til digital programmering (Berg, 2021) (Matematikksenteret, u.å.c). Cortina, referert i Berg (2021), fremhever analog programmering som en tilnærming som appellerer til en bredere gruppe elever enn digital programmering. Analog programmering krever ikke spesifikke tekniske ferdigheter, og dette kan bidra til å fjerne den tekniske barrieren for læring, samtidig som datamaskinen ikke blir en distraksjon (Berg, 2021). Analog programmering inkluderer terminologien på en måte som ikke kommer like tydelig frem i digital programmering. Flø (2021) trekker frem flere studier som viser at elever synes begreper i programmering, f.eks. løkker, variabler, funksjoner og vilkår, er vanskelige og krevende for elevene å lære. Derfor er det viktig at læreren eksplisitt inkluderer begrepene i sin undervisning, og analog programmering er en populær tilnærming til begrepslæring i programmering (Flø,2021).

En undersøkelse av læreres tilnærming til programmering viser at lærere har tro på at analog programmering bidrar til økt algoritmisk tekning ved at det trener elevene på å

forklare en prosess, og dekomponering av et problem til løsbare delproblem (Berg, 2021). Vinnervik (2023) viser at lærere får økt selvtillit i undervisning av algoritmisk tenkning gjennom unplugged aktiviteter.

2.2.2 Blokkprogrammering

Digital programmering deles i skolesammenheng inn i to kategorier blokkprogrammering og tekstprogrammering. Et blokkprogrammeringsspråk har ferdigskrevede kommandoer i blokker/klosser, og elevene kan dermed bygge kode ved å sette sammen blokker (Gjøvik & Høyland, 2022). Tekstprogrammering er ren kode, og krever at elevene kan forstå og anvende tekstprogrammeringsspråk. Figur 2.2 viser eksempler på de to typene programmeringsspråk, der kan en se at ved blokkprogrammering er klossene ferdig kodet slik at elevene kun trenger å fylle ut detaljene, mens i tekstprogrammering må elevene skrive hele koden selv.



Figur 2.2: To program som printer de første 100 oddetallene. T.v. er programmet kodet i Scratch og t.h. er Python. Figuren er hentet fra Matematikksenteret (u.å.b)

I blokkprogrammering er det lettere å bygge kode, da det ikke er mulig å sette sammen klosser som ikke passer sammen, og dermed vanskeligere å gjøre syntaksfeil. Gjøvik og Høyland (2022) anbefaler Scratch for nybegynnere; Scratch har stort handlingsrom, og har nok funksjonalitet til at elevene kan lære sentrale og viktige programmeringsteknikker. Blokkprogrammering er effektivt og brukervennlig, da det er veldig visuelt med lett brukergrensesnitt. Blokkprogrammering gjennom Scratch gir elevene programmeringsferdigheter som er overførbare til tekstprogrammering. Blokkprogrammering er basert på samme prinsipper som tekstprogrammering, med vilkår, funksjoner, variabler, og løkker (Gjøvik & Høyland, 2022). Både tekst- og blokkprogrammering kan brukes til å programmere roboter og mikrokontrollere, f.eks. Micro:bit. Micro:bit er en mikrokontroller som kan kobles til motorer og sensorer, som gir mange muligheter for å bruke den i undervisning. Micro:bit kan kodes med blokk- og tekstprogrammering, og gjennom MakeCode kan elevene skifte mellom blokker og tekstspråkene JavaScript og Python (Gjøvik & Høyland, 2022).

2.2.3 Tekstprogrammering

Det finnes ca. 1000 programmeringsspråk, og Python er blant de vanligste (Kristensen & Kirfel, 2022). Valg av programmeringsspråk i skolesammenheng har vært en diskusjon helt fra programmering ble innført i skolesammenheng tidlig på 70-tallet (Circk, 2017). En internasjonal studie gjort i 2017, blant annet i Australia, Storbritannia og USA, viser at Python er det foretrukket programmeringsspråket i undervisningssammenheng (Circk,

2017). I læreplanen er det ingen spesifikasjoner for hvilke språk eller verktøy som skal anvendes i skolen, men kompetansepakken til Utdanningsdirektoratet og eksamen for 10.trinn i matematikk bruker Scratch og Python (Udir, 2019b; Udir, 2024).

Haraldsrud et al. (2020) trekker frem flere pedagogiske, faglige og didaktiske grunner til å bruke tekstprogrammet Python i undervisningen. For det første er Python ett av de enkleste tekstbaserte programmeringsspråkene både å lese og skrive. Det er lett overførbart, og passer dermed godt for transponering både fra flytskjema, og fra matematikk da syntaksen i Python er relativt intuitiv. Python er hyppig brukt både i akademia og i næringslivet. Flere peker på nærheten til virkeligheten som en av de mest motiverende faktorene ved programmering, og da er det å bruke et ekte programmeringsspråk viktig (Haraldsrud et al., 2020; Crick, 2017). Til slutt trekker de frem Pythons bibliotek, der det finnes mye ferdigskrevet kode som kan benyttes, spesielt i realfagene. Bibliotekene gjør det enklere for elevene å anvende programmering i realfaglig kontekst (Haraldsrud et al., 2020).

2.3 PRIMM

I kontrast til læring av fremmedspråk, der elever vanligvis lærer å lese før de begynner å skrive, går elevene i programmeringsundervisning umiddelbart i gang med å skrive kode (Crick, 2017; Looi et al., 2018). Flere forskere har påpekt at denne tilnærmingen til å lære programmering kan være problematisk. Som et svar på dette har forskere utviklet PRIMM-modellen. PRIMM, står for «Predict, Run, Investigate, Modify, Make» og er en forskningsbasert tilnærming som tar utgangspunkt i et sosiokulturelt perspektiv. Utviklingen av PRIMM-modellen er basert på en studie som viser at elever trenger støtte for å forstå programmeringsteknikker (Coleman, 2021). Formål med PRIMM er å gi veiledning i planleggingen av programmeringsundervisning (Matematikksenteret, u.å.c; Coleman, 2021). PRIMM gir elevene en forståelse av koding ved å analysere og utforske eksisterende kode, noe som gir dem innsikt i hvordan kode fungerer før de selv begynner å skrive (Crick, 2017; Gjøvik & Høyland, 2022).

Flere forskere har fremhevet at evnen til å feilsøke, og analysere kode er avgjørende og nødvendig for å kunne mestre programmering (Crick, 2017; Vinnervik, 2023; Kaufmann & Stenseth, 2019). Denne ferdigheten understreker viktigheten av å inkludere aktiviteter som gir elevene mulighet til å undersøke og eksperimentere med kode i programmeringsundervisningen, i motsetning til å alltid skulle lage fra bunn av.

I tabell 2.1 har jeg beskrevet hver komponent i PRIMM-modellen, basert på Gjøvik og Høylands oversettelse (Gjøvik & Høyland, 2022; Coleman, 2021).

Kategorier	Forklaring
Forutse (Predict)	Elevene får utdelt en kode, og skal forklare hva som skjer når programmet kjøres, de kan enten skrive eller tegne outputen. I denne fasen er fokuset på funksjon til elementene i koden.
Kjøre (Run)	Elevene blir bedt om å kjøre programmet.
Undersøke (Investegate)	Elevene utforsker koden gjennom deloppgaver gitt av lærer/lærebok/lignende. Her kan elevene spore koden ved å følge en variabel gjennom en kode og se hvordan variabelen endrer seg. Elevene kan forklare koden, linje for linje og hele sammenhengende. Elevene kan kommentere hver linje i koden og skrive hva hver linje med kode gjør. Dersom programmet inneholder en feil eller unødvendig kode kan elevene undersøke og markere forbedringer
Endre (modify)	Elevene redigerer koden, enten ved å rette opp en feil, legge til en ekstra funksjon, eller fjerne unødvendig kode.
Lage (make)	Elevene skal lage et helt nytt program fra bunn av. Programmet involverer de samme strukturene som i det andre programmet, men skal løse et nytt problem

Tabell 2.1: Fasene i PRIMM oversatt og konkretisert fra Coleman (2021).

Et av hovedpoengene i PRIMM-modellen er at elevene først blir eksponert for ferdigskrevet kode som de skal undersøke grundig. Gjennom å forutse koden, får elevene trening i å tolke og forstå kodestrukturer, noe som gir dem et innblikk i hvordan kode fungerer. Deretter oppfordres elevene til å kjøre koden for å bekrefte eller avkrefte deres hypotese om hva programmet gjør. Undersøkelse utgjør en betydelig del av PRIMM. Å spore kode (tracing); å lese kode linje for linje eller følge en variabel, er en viktig del av å skrive kode (Looi et al., 2018; Crick, 2017). Det samme er å tolke koden gjennom flytskjema, eller feilsøke programmet. Feilsøking etter syntaks og/eller logiske feil er en kritisk del av programmering og en sentral del av Den algoritmiske tenkeren (Vinnervik, 2023; Kaufmann & Stenseth, 2019; Udir, 2019).

Det sosiale aspektet av PRIMM-modellen, der elevene oppfordres til å forklare og diskutere koden sammen, kan bidra til å styrke begrepsforståelsen til elevene ved å kreve at de prater om kodens komponenter (Coleman, 2021). Det sosiale aspektet kan kobles til Den algoritmiske tenkeren gjennom samarbeid (Udir, 2019).

I endringsfasen får elevene muligheten til å skrive egen kode. Dette gir elevene en følelse av eierskap til koden, noe som kan føre til økt selvtillit og engasjement (Coleman, 2021; Matematikksenteret, u.å.c).

Til slutt skal elevene utvikle et helt nytt program, enten med en ny funksjon eller knyttet til et annet problem. Dette nye programmet kan være inspirert av det første programmet som elevene ble introdusert for, slik at de kan dra nytte av elementer fra den eksisterende koden. Evnen til å navigere mellom ulike tilnærminger til kode, som å forutse, undersøke, endre og lage kode, kan resultere i en økt generell forståelse av programmering hos elevene (Sentance et al., 2019).

PRIMM representerer en effektiv tilnærming til å lære programmering, da store deler av programmering dreier seg om feilsøking, forståelse av kode og analytisk tenkning, aspekter som kan være utfordrende å lære (Kaufmann & Stenseth, 2019). Videre vil PRIMM gjøre det enklere for elever å komme i gang med programmering ved å gi dem en eksisterende kode å jobbe ut fra, og dermed redusere kravene til forkunnskaper innen

programmering og senke inngangsterskelen (Kaufmann & Stenseth, 2019). PRIMM kan anvendes på alle nivå, og en kan differensiere oppgavene ved variere vanskelighetsgraden på koden elevene tar utgangspunkt i.

2.4 Programmering i matematikk

Det å kunne anvende programmering som et middel for læring er en viktig del av læreplanen (Kunnskapsdepartementet, 2019). Papert (1980) beskrev flere positive effekter av programmeringsverktøyet Logo. Han observerte at elever forbedret sine kognitive evner ved å utforske matematikk gjennom programmering. I Scherer et al. (2019) sammenlignes effekten av programmering med spill aktiviteter og tradisjonell undervisning. Resultatene indikerer at programmeringsarbeid har overføringsverdi til matematikk og fremmer økt matematisk og logisk tenkning (Scherer et al., 2019; Mayer et al.,1986). Flere studier av elever og studenters arbeid med programmering i matematikk viser en positiv effekt på elevers holdning til matematikkfaget (Lambic, 2011; Forsström & Kaufmann, 2018). Dette gjenspeiles i læreres erfaring, der elever som tradisjonelt har slitt med matematikk, mestrer programmering og viser interesse for dette området (Berntsen & Holone, 2016). Dette indikerer at programmering kan motivere en annen elevgruppe enn tradisjonell undervisning og ved å diversifisere undervisningen kan programmering bidra til forbedret matematisk kompetanse for en bredere elevgruppe (Forsström & Kaufmann, 2018).

Andre studier på samme felt har gitt tvetydige resultater, og flere studier har ikke funnet korrelasjon mellom programmeringsundervisning og evnen til resonnering (Mayer et al.,1986). En kritikk av implementeringen av programmering i skolen er manglende kobling til fag, og lite faglig utbytte (Kafai & Burke, 2013). Kaufmann og Stenseth (2019) mener at læreres kompetanse og bevissthet er sentral for å gjøre kobling mellom matematikk og programmering. Kilhamn et al. (2021) har utviklet et rammeverk for å analysere relasjonen mellom matematikk og programmering.

2.4.1 Relasjon mellom matematikk og programmering

Programmering kan bidra til økt matematikkforståelse (Gjøvik & Høyland, 2022). Det finnes ulike måter å implementere programmering på i undervisning, og programmering kan ha ulike roller i undervisningen. Når en implementerer programmering i undervisningen er det viktig å være bevisst på hvilken rolle programmering har i undervisning, og hensikten med å implementere programmering (Gjøvik & Høyland, 2022). Kilhamn et al. (2021) har utviklet fire kategorier for relasjonen mellom matematikk og programmering. Gjøvik og Høyland (2022) har laget en lignende kategorisering. Kategoriene i tabell 2.1, kan brukes som et verktøy i undervisning, for å være oppmerksom på hvilken rolle programmering har.

Programmering **uten kobling** til matematikk.	Matematikk som **en kontekst** for programmering	Programmering som **et verktøy** for å **effektivisere** matematiske beregninger.	Programmering som **et verktøy** for å **utforske** matematikk
Disse leksjonene handler kun om programmering.	Disse leksjonene tar for seg matematikk, og bruker matematikk for å lære seg programmering. Det læres ingen ny matematikk, men programmering kan utnyttes for å repetere eller styrke matematikk kunnskaper.	Disse leksjonene har et tydelig matematikkinnhold. En datamaskin programmeres og anvendes som et effektivt sett for å utføre beregninger.	Disse leksjonene anvender programmering for å utforske matematiske konsepter og sammenhenger. Programmering tilfører ny tilnærming til matematikk og angir informasjon slik at en kan få en dypere matematisk forståelse.

Tabell 2.2: Viser kategoriene for relasjon mellom programmering og matematikk i undervisning (Kilhamn et al., 2021, s. 293)

Undervisning uten kobling til matematikk handler om å lære seg det programmeringstekniske, og et eksempel kan være en undervisningsøkt der elevene undersøker komponenter som løkker, variabler og funksjoner, og lager et program som krever bruk av slike kontrollstrukturer. Læringsutbytte i slik undervisning er programmering og ikke matematikk (Kilhamn et al., 2021; Gjøvik & Høyland, 2022).

De tre andre kategoriene inkluderer både matematikk og programmering. Matematikk som en kontekst for programmering handler om å bruke matematikk for å lære programmering. Læringsutbytte i programmering med matematikk som kontekst er hovedsakelig knyttet til programmeringstekniske mål, men økte ferdigheter og kunnskap innen matematikk kan være et biprodukt da slik undervisning repeterer allerede eksisterende kunnskap i matematikk. Likevel gir denne kategorien mest læringsutbytte i programmering.

De to neste kategoriene løfter matematikk inn i programmeringen. I disse kategoriene blir programmering brukt som et verktøy i matematikk. Programmering som et verktøy for å effektivisere matematiske beregninger bygger på kategori 2, men har et tydelig matematikkinnhold. Denne kategorien er mer spesifikk, og handler om å programmere en datamaskin til å utføre matematiske beregninger. Ved å ta i bruk et ferdig program, som elevene, læreren eller noen eksterne har laget for å f.eks. øve på multiplikasjon vil programmering bli brukt som et verktøy (Gjøvik & Høyland, 2022).

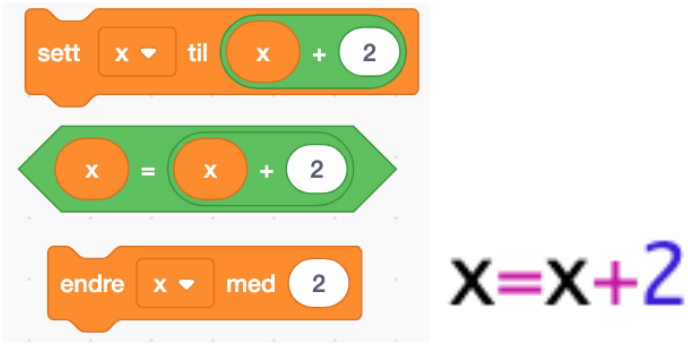
Ved å bruke programmering som et verktøy for å utforske matematikk får elevene en ny tilnærming til matematikk, som kan bidra til å få en dypere forståelse av det matematiske konseptet (Kilhamn et. al., 2021). Et konkret eksempel kan da være å lage et program som tegner mangekanter og lar elevene utforske vinkelsummen i mangekanter. Undervisning med programmering som verktøy for matematiske beregninger har ikke den utforskende-delen og et der eksempel kan være mengdetreningsspill i regneartene. Selve undervisningen med programmering har ikke

mål om å lære noe programmeringsteknisk (Kilhamn et al., 2021; Gjøvik & Høyland, 2022).

Kilhamn et al. (2021) brukte denne modellen for å analysere matematikkundervisning, de så da at kategori 2, matematikk som en kontekst for programmering inngikk i både kategori 3 og 4, slik at det var vanskelig å skille 2 fra 3 og 4. Studiens funn er presentert i 2.8.

2.4.2 Utfordringer med programmering og matematikk

Gjøvik og Høyland (2022) trekker frem flere utfordringer ved å koble matematikk og programmering. For det første har enkelte begreper en annen betydning i programmering enn i matematikk. Likhetstegnet er et godt eksempel på et konsept som har en helt annen betydning i programmering versus matematikk. I matematikk er elevene vant med at verdien av høyre siden for likhetstegnet skal tilsvare verdien på venstre siden av likhetstegnet, i programmering er ikke dette nødvendigvis sant da likhetstegnet brukes for å sette verdien av en variabel. Dermed kan en i programmering skrive x= x+2, dersom en ønsker å øke x med 2, mens dette i matematisk kontekst vil være ukorrekt (Gjøvik & Høyland, 2022).

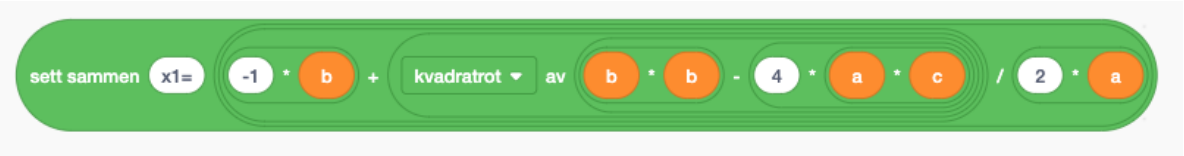


Figur 2.3: Viser hvordan en endrer variabelen x slik at den øker med 2. t.v. i blokkprogrammeringsprogrammet Scratch og t.h. i tekstprogrammeringsprogrammet Python. (Hentet fra Matematikksenteret u.å.c)

I Scratch finnes det alternativer til å bruke likhetstegnet, ved å bruke klosser som endre eller sett variabelen til, mens i tekstprogrammering må man anvende likhetstegnet for å skrive samme kommando (se figur 2.3). Det er viktig at læreren er bevisst på bruk av likhetstegnet i tekstprogrammering (Gjøvik & Høyland, 2022).

For det andre benyttes andre representasjoner for regneoperasjoner. Elevene er muligens vant med å skrive potenser slik a^b eller a^b, mens potenser i Python skrives med dobbelt gangetegn: **, mens potenser i Scratch løses med gjentatt multiplikasjon/divisjon (Gjøvik & Høyland, 2022). I Python er det mulig å benytte seg av ulike bibliotek med forhåndslagre kommandoer som inneholder noen gitte konstanter, f.eks. pi, π. Disse er ikke lagt inn i Scratch, og en må dermed manuelt skrive inn tallet, og dermed i tilfeller med pi må elevene gjøre avrundinger.

En av styrkene i matematiske notasjoner er at den kan presse sammen en stor mengde informasjon i kompakt form, pga. bruk av symboler. Dette er vanskeligere i blokkprogrammering da hver regneoperasjon har en kloss, og noen kommandoer velges i en dropdown-meny og består av ord, ikke symboler. Et godt eksempel på at formler kan bli mindre kompakt i blokkprogrammering er abc-formelen, gitt med kun én løsning: $\frac{-b\pm\sqrt{b^2-4ac}}{2a}$, se figur 2.4.



Figur 2.4: Viser hvordan abc-formelen kan settes sammen i Scratch. (Hentet fra Gjøvik & Høyland, 2022, s. 109)

Gjøvik og Høyland(2022) poengterer at selv om programmering har fått økt fokus bør man ikke glemme andre digitale ressurser som Geogebra og Excel som kan være mer hensiktsmessig enn programmering i enkelte tilfeller f.eks. regnskapsføring og tegning av grafer (Gjøvik & Høyland, 2022).

2.5 Programmering i LK20

Programmering ble implementert i læreplanen gjennom kompetanseløftet i 2020 (Kunnskapsdepartementet, 2019). Der står det at elevene skal bruke programmering i matematikk, naturfag, musikk og kunst og håndverk (Flø, 2021; Vinnervik & Bungum, 2022; Kunnskapsdepartementet, 2019). Selv om programmering er innført i flere fag har matematikk hovedansvaret for elevenes kompetanse i programmering (Flø, 2021; Vinnervik & Bungum, 2022). I læreplanen står det at elevene både skal lære programmering, altså det programmeringstekniske, og anvende programmering som verktøy for å utforske faglige problemstillinger (Flø, 2021).

Programmering inngår i flere deler av læreplanen. Programmering fremmer digitale ferdigheter, og gir et verktøy for å løse matematiske problemer ved hjelp av digitale hjelpemidler (Flø, 2021; Vinnervik & Bungum, 2022). I tillegg til grunnleggende ferdigheter, blir programmering eksplisitt nevnt i kompetansemålene, og algoritmisk tenkning blir nevnt i kjerneelementer i matematikk.

2.5.1 Kjernelementer i matematikk

Kjerneelementene er fagspesifikke og er det viktigste faglige innholdet. Kjerneelementene består av begreper, metoder, og tenkemåter som elevene må lære for å kunne mestre og å anvende faget (Udir, 2019b).

I matematikk er det fem kjerneelementer. Algoritmisk tenkning blir eksplisitt nevnt i kjerneelementet *utforskning og problemløsning* (Kunnskapsdepartementet, 2019). Algoritmisk tenkning defineres av Udir som en problemløsningsmetode med bruk av digitale hjelpemidler, og det er derfor naturlig at algoritmisk tenkning inngår under problemløsning (Kristensen & Kirfel, 2022). Flere av læreplanmålene (tabell 2.3) går ut på å bruke programmering til å utforske matematikk. Udir trekker dermed frem programmering som et verktøy for utforsking og problemløsning.

En kan argumentere for at programmering inngår i flere av kjerneelementene, blant annet trekker Haraldsrud et al. (2020) frem programmering som et modelleringsverktøy, som kan vise et problem over tid, f.eks. gjennom simuleringer, en kan derfor argumentere for at programmering er en del av kjerneelementet modellering og anvendelser (Haraldsrud et al., 2020). Slik Udir definerer algoritmisk tenkning er abstraksjon og generalisering en viktig del av prosessen, og en kan derfor argumentere for at programmering også inngår i dette kjerneelementet.

2.5.2 Programmering i matematikk

Kompetansemålene i matematikk inkluderer programmering eksplisitt fra 5.trinn, men prinsippene for programmering er med allerede fra 2.trinn. Det er totalt ni programmeringsrelaterte kompetansemål fra LK 20, se tabell 2.3 (Gjøvik & Høyland, 2022)

Trinn	Kompetansemål	Relasjon til matematikk
Kompetansemål etter 2. trinn	Lage og følge regler og trinnvise instruksjoner i lek og spill	Programmering **uten kobling** til matematikk.
Kompetansemål etter 3. trinn	Lage og følge regler og trinnvise instruksjoner i lek og spill og knytte til koordinatsystemet	
Kompetansemål etter 4. trinn	Lage algoritmer og utrykke dem ved bruk av variabler, vilkår og løkker	
Kompetansemål etter 5. trinn	Lage og **programmere** med bruk av variabler, vilkår og løkker	Programmering **uten kobling** til matematikk.
Kompetansemål etter 6. trinn	Bruke variabler, løkker, vilkår og funksjoner i **programmering** til å utforske geometriske figurer og mønster	Programmering som **et verktøy** for å **utforske** matematikk
Kompetansemål etter 7. trinn	Bruke **programmering** til å utforske data tabeller og datasett.	Programmering som **et verktøy** for å **utforske** matematikk
Kompetansemål etter 8. trinn	Utforske hvordan algoritmer kan skapes, testes og forbered ved hjelp av **programmering.**	Matematikk som **en kontekst** for programmering
Kompetansemål etter 9. trinn	Simulere utfall i tilfeldige forsøk og beregne sannsynligheten for at noe skal inntreffe ved å bruke **programmering**.	Programmering som **et verktøy** for å **utforske** matematikk
Kompetansemål etter 10. trinn	Utforske matematiske egenskaper og sammenhenger ved å bruke **programmering.**	Programmering som **et verktøy** for å **utforske** matematikk

Tabell 2.3: Kompetansemålene i matematikk relevante for programmering hentet fra LK20(Kunnskapsdepartementet, 2019).

For å forstå innholdet i kompetansemålene, har jeg valgt å kategorisere dem i henhold til Kilhamn et al. (2021). Kompetansemålene til og med 5.trinn handler om programmeringsteknikk, og faller derfor under kategorien programmering uten kobling til matematikk. Gjøvik og Høyland (2022) trekker frem at det som skiller programmering på 5.trinn fra 6.trinn og oppover, er at kompetansemålene knytter programmering til tradisjonell matematikk. Kompetansemålet etter 8.trinn skiller seg fra de resterende målene på 6-10.trinn fordi det handler om å utforske algoritmer, et begrep som eksisterer både i programmering og matematikk, så på den måten kan en argumentere

for at dette kompetansemålet kun handler om programmering. De resterende kompetansemålene er kategorisert som programmering som et verktøy for å utforske matematikk, og det tydeliggjøres i selve kompetansemålene der formålet med programmering blir spesifisert (Vinnervik & Bungum, 2022). For 6-9.trinn er programmering koblet til en del av matematikken, og det er naturlig å se kompetansemålene med programmering i sammenheng med resten av kompetansemålene for matematikk. Eksempelvis for 9.trinn hvor kompetansemålet med programmering er koblet til simulering og sannsynlighet. En slik sammenkobling av to fagfelt kan være utfordrende. Det krever at elevene har nødvendig kompetanse i programmering for å kunne få læringsutbytte i matematikk, derfor er det viktig at elevene har grunnleggende ferdigheter i programmering, noe læreplanen legger til rette for ved undervisning i programmeringsteknikk på 2-5.trinn (Gjøvik & Høyland,2022; Vinnervik & Bungum, 2022).

Målet poengterer det samme som Kilhamn et.al. (2021): programmering som et verktøy for å undersøke matematikk. Det spesifiserer ikke hvilke matematiske konsepter som skal utforskes, og bør derfor ses i sammenheng med resten av læreplanen etter 10.trinn. Gjøvik og Høyland (2022) har tatt for seg de programmeringsrelevante kompetansemålene, og i kompetansemålet etter 10.trinn peker de på at dette målet krever at elevene kobler matematikk til programmering. Selv om enkelte elever ser sammenhengen og nytteverdien av feltene, kan svakere elever synes at koblingen er vanskelig (Gjøvik & Høyland,2022). Gjøvik og Høyland (2022) mener lærere må være bevisste på hva slags matematikk de kobler programmering med. Enkelte matematiske konsepter passer bedre med programmering enn andre; Scratch er passende for å utforske vinkelsum i mangekanter mens Scratch ikke er spesielt egnet for å utforske Thales setning eller Pascals trekant (Gjøvik & Høyland,2022).

Avslutningsvis setter Gjøvik og Høyland spørsmål ved hva det vil si å "bruke programmering"; er det å skrive kode, eller kan en bruke programmering ved å diskutere og/eller bruke allerede eksisterende program (Gjøvik & Høyland,2022). Læreplanen er relativt åpen, så det er opp til hver enkelt lærer hvordan de velger å tolke «bruke programmering».

Læreplanen LK20 legger ingen føringer om krav til programmeringsspråk, og legger derfor opp til at lærere kan tolke målene og bruke de programmeringsverktøyene de finner best passende. Gjøvik og Høyland (2022) benytter blokkprogrammering på alle trinn, mens Kristensen og Kirfel (2022) anvender kun Python. Utdanningsdirektoratet ønsker ikke å legge noen ytterligere føringer utover det som står i læreplanen, men de har laget en kompetansepakke for programmering og algoritmisk tenkning med tips og forslag til oppgaver/opplegg lærere kan bruke i undervisning av programmering (Udir, u.å.). I kompetansepakken benytter Udir analoge aktiviteter på 2.trinn og Scratch på 3.-7.trinn (Udir, u.å.). På ungdomstrinnene benytter Udir Python, og kobler programmering til geometri, gjentakende strukturer og funksjoner og modellering (Udir, u.å.).

2.5.3 Utforskende matematikk

En sentral del av relasjon mellom matematikk, programmering og kompetansemålet for 10.trinn er utforskende matematikk (Kilhamn et al., 2021; Kunnskapsdepartementet, 2019). Utforskning ble en større del av kunnskapsløfte 2020, og Udir definerer begrepet «utforske» i vedlegget til læreplanen slik; «Å utforske handler om å oppleve og eksperimentere og kan ivareta nysgjerrighet og undring. Å utforske kan bety å sanse, søke, oppdage, observere og granske. I noen tilfeller betyr det å undersøke ulike sider av

en sak gjennom åpen og kritisk drøfting. Å utforske kan også bety å teste eller prøve ut og evaluere arbeidsmetoder, produkter eller utstyr» (Udir, 2019). En slik forståelse av utforskning involverer en mer elevstyrt tilnærming, som avviker fra den tradisjonelle undervisningen der elevene noterer av lærerens arbeid. Definisjonen av utforskning er ikke fagspesifikk, og en trenger derfor å spesifisere den til matematikk.

Tradisjonell matematikkundervisning har fokusert på det abstrakte og bygd på tanken om at matematikk undervisning skal gi elevene lærdom som de senere kan anvende i ulike kontekster og arbeidslivet. Nyere tankegang, spesielt vist i LK20 er at utbytte av matematikk undervisning skal være at elevene utvikler evnen til å bli tenkende, kreativ og kritisk, med formål om å gi elevene innsikt i meningsfull bruk av matematikk, og skape motivasjon og gode holdninger i faget (Sikko & Grimmeland, 2020; Engeln et al., 2013; Artigue & Blomhøj, 2013). Det nye synet på matematikk speiler slik man ser utforskende matematikk.

I matematisk kontekst er utforskende matematikk ofte koblet til inquiry-based learning (IBL). IBL strekker seg helt tilbake til Deweys «Learning by doing» og det finnes et bredt spekter av definisjoner. De fleste handler om utvikling av et spørrende sinn og en vitenskapelig holdning (Sikko & Grimmeland,2020; Engeln et al.,2013; Artigue & Blomhøj, 2013). I denne artikkelen brukes modellen utviklet av det EU-finansierte PRIMAS prosjektet (Sikko & Grimmeland,2020; Artigue & Blomhøj, 2013). Modellen tar for seg kjennetegn ved IBL med fokus på fem aspekter (figur 2.5); Lærerens veiledning, klasseromskultur, ønsket utbytte, oppgaver og elevene. I denne studien er de tre siste aspektene mest sentrale.

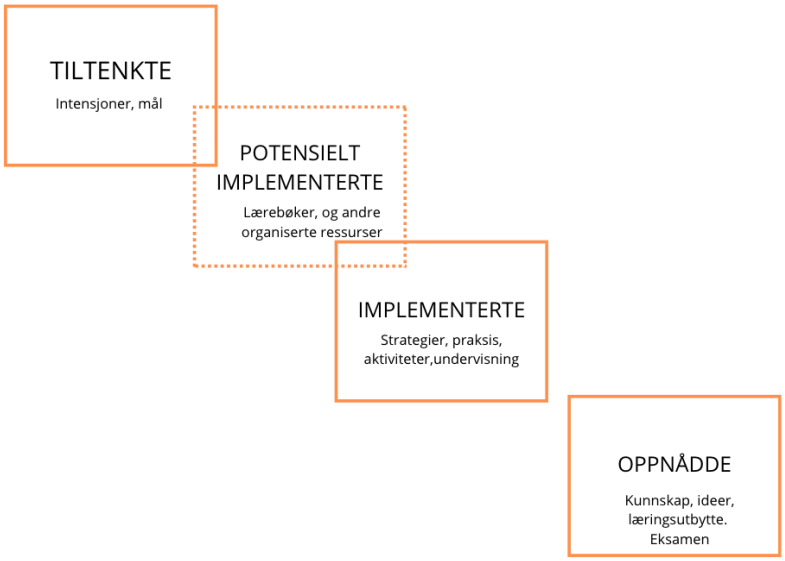
Figur 2.5: Kjennetegn ved IBL (Oversatt og rekonstruert fra Maaß & Reitz-Koncebovski, 2013 s. 8).

Utforskende oppgaver i matematikk kjennetegnes ved at de er åpne, og med det menes at de kan løses med flere fremgangsmåter. De er også virkelighetsnære og oppleves av elevene som realistiske problemer enten til deres omverden eller til vitenskapen (Maaß & Reitz-Koncebovski, 2013; Sikko & Grimmeland,2020; Artigue & Blomhøj, 2013). Oppgavene skal inneholde et problem der elevene kan stille spørsmål, og dekomponere problemet inn i løsbare delproblem. Samarbeid er en del av utforskende matematikk, og det er ønskelig at elevene gir hverandre input og sammen løser problemene. De 5 e-ene er et verktøy for å utforskende matematikk, der elevene skal engage, explore, explain, elaborate og evaulate (Benton et al, 2016;2017). De 5 e-ene minner om en problemløsningsprosess, og har likhetstrekk med Den algoritmiske tenkeren.

2.6 Lærebøker

Lærebøker utgjør hovedkilden til matematikkundervisning for mange lærere (Valverde et al., 2002; Rezat & Strässer, 2017; Johansson, 2006; Haggarty & Pepin, 2002). De brukes som inspirasjon og basis for undervisningen, og elevene tilbringer mye tid med å løse oppgaver fra lærebøker (Pepin, 2010; Rezat & Strässer, 2017). Studier viser at lærebøkene anses som en autoritet på kunnskap og veiledning til læring, og noen lærere støtter seg utelukkende til lærebøkene (Pepin, 2010). Hiebert et al. (i Pepin., 2010) argumenterer for at oppgavene i lærebøkene spiller en avgjørende rolle i formingen av elevenes oppfatninger om matematikk og deres læring av faget. Selv i dagens digitale verden er de fysiske lærebøkene sentrale i matematikkundervisningen (Rezat & Strässer, 2017). Viktigheten av lærebøker ble tydeliggjort i mai 2024 da den norske regjeringen varslet en millionsatsing på fysiske lærebøker i grunnskolen (Utdanningsnytt, 2024).

Valverde et al. (2002) hevder at lærebøker er designet for å konkretisere den abstrakte læreplanen inn i operasjoner og representasjoner som lærere, elever og foreldre kan bruke. Det er derfor svært relevant å studere lærebøker og dens tolkning av læreplanen, spesielt da det ikke finnes en statlig kontroll av innholdet i lærebøkene. Valverde et al. (2002) bygger videre på TIMMS modell for hvordan læreplanen blir implementert. Modellen til TIMMS inneholder tredeler: den tiltenkte læreplanen (intended) altså læreplanens intensjon, den implementerte læreplanen, som omfatter undervisningen og den praktiserte læreplanen, og den siste delen er den oppnådde læreplanen(attained), altså det elevene sitter igjen med av kunnskap fra den tiltenkte læreplanen (Valverde et al., 2002; Pepin et al., 2001). Valverde et al. (2002) argumenterer for at det er et element til i denne modellen, nemlig den potensielt implementerte læreplanen (se figur 2.5), som lærebøker er en del av.



Figur 2.6: Den potensielt implementerte læreplanen (Oversatt og rekonstruert fra Valverde et al., 2002, s.13).

Valverde et al. (2002) og Pepin et al. (2001) mener at lærebøker har en medierende rolle for den implementerte læreplanen. Rollen er kompleks da lærere og elever ikke er passive til lærebøkene, og lærere og elever vil derfor påvirke lærebøkene i praksis. Likevel argumenterer Valverde et al. for at bøkene har såpass stor tillit i de nordiske landene, siden de blir så mye brukt, og at de blir brukt som om de var en konkretisering av læreplanen (Valverde et al., 2002; Pepin et al., 2001; Rezat & Strässer, 2012).

Lærebøker som den potensielt implementerte læreplanen kommer til syne i bøkene ved at alle lærebøkene som er blitt analysert i denne oppgaven hevder at de er utviklet etter og på bakgrunn av kompetansemålene i matematikk (Hjardar & Pedersen, 2020; Kongsnes & Wallace,2020; Tofteberg et al., 2021). Når det er sagt er ikke lærebøkene kvalitet sikret av Kunnskapsdepartementet, de legger ingen ytterligere føringer til læreplanen. Tidligere sto det i opplæringsloven at alle lærebøkene som ble brukt i skolen måtte bli godkjent av kunnskapsdepartementet, slik er det ikke nå lenger (Ot.prp. 44, 1999-2000). Lærebøkene er forlagenes tolkning av læreplanen, og er ikke blitt kvalitetssjekket etter læreplanen og læreplanens intensjon.

2.7 Tidligere forskning

Bråting & Kilhamn gjennomførte i 2021 en studie av programmering i lærebøker ble gjennomført i Sverige. Studien tok for seg alle oppgavene med programmering i fire lærebøker for matematikk på 1-6.trinn (Bråting & Kilhamn, 2021). Studien var todelt: den første delen brukte kjennetegn ved algoritmisk tenkning til å studere karakteristikk ved programmeringsinnholdet i bøkene. Den andre delen så på broen mellom matematikk og programmering (Bråting & Kilhamn, 2021). For å analysere koblingen benyttet studien «the 5E's» av Benton et al. (2016; 2017); Explore, Explain, Envisage, Exchange og bridgE. Der det siste konseptet evaluerte elevenes evne til å koble matematiske konsepter og programmering.

Resultatet var at oppgavene ofte ga elevene en problemstilling og steg-for-steg-instruksjoner, som introduserte elevene til koding, men resulterte i at eleven ikke så helheten, og dermed lite til ingen utvikling i algoritmisk tenkning. Programmering ble

presentert i lærebøkene i sammenheng med geometri og algebra. I oppgavene om algebra så Bråting og Kilhamn (2021) ingen kobling til matematikk, bare oppgaver der elevene skulle repetere et mønster. I geometri så de en kobling mellom matematikk og programmering, der matematikk fungerte som bakteppe for programmeringen, tilsvarende programmering med matematikk som kontekst (Kilhamn et al.,2021). Det var derimot ingen kobling andre veien. Bråting og Kilhamn (2021) ser et potensiale der elevene kan bruke programmering som kontekst eller verktøy for å lære/anvende matematikk, til tross for at dette potensiale ikke ble utnyttet i de svenske lærebøkene.

3 Metode

I metodekapittelet skal jeg gi en oversikt over forskningsdesignet, paradigmevalget, metodikken og utvalget. Kapittelet er strukturert slik at jeg først beskriver hvordan studien ble gjennomført, inkludert en demonstrasjon av datanalysen og utbredelsen av analyseskjemaet. Deretter vil en refleksjon av studiens validitet og reliabilitet, før jeg til slutt redegjør for de etiske betraktningene som ble gjort i denne studien.

3.1 Vitenskapelig paradigme

Vitenskapelige paradigmer gir et rammeverk som definerer forskerens syn på verden og legger grunnlaget for studiens tilnærming og analyse (Creswell & Creswell, 2018). Et paradigme består av fire elementer; ontologi, som omhandler grunnleggende spørsmål om virkelighetens natur og menneskets vesen. Epistemologi, som utforsker forholdet mellom det kjente og det ukjente, samt forskerens tilnærming til verden og forskning. Metodologi, som definerer hvilke metoder som er hensiktsmessige for å tilegne seg kunnskap om verden. Etikk, som tar for seg hvordan forskeren ivaretar moralske hensyn i forskningssammenheng (Creswell & Creswell, 2018). Innen utdanningsforskning identifiseres fire hoved paradigmer: positivisme og post-positivisme, interpretivisme, pragmatisme og transformativt perspektiv (Creswell & Creswell, 2018).

Denne studien tar utgangspunkt i en interprevistisk tilnærming, også kjent som fortolkningsparadigmet. Interpretivismen betrakter verden som en sosial konstruksjon og anerkjenner dermed at det eksisterer flere sannheter, ettersom individer tolker verden på ulike måter (Creswell & Creswell, 2018). Interpretivismen mener at mennesker ikke kan skilles fra sine erfaringer og kunnskap, og derfor er det menneskelige aspektet essensielt. I forskningen innebærer dette at forskeren alltid bringer med seg sine egne erfaringer og kunnskaper, og forskningen vil derfor alltid være preget av forskerens subjektive tolkning. Interpretivismen benytter kvalitative metoder for å oppnå en grundig forståelse av forskningsobjektet (Creswell & Creswell, 2018).

I denne studien anvender jeg en interprevistisk tilnærming, da jeg gjennomfører en kvalitativ studie med mål om å oppnå en dypere forståelse av programmeringens rolle i matematikklærebøker. Med en interprevistisk tilnærming til analysen, er det viktig å bemerke at tolkningen er basert på min kunnskap og mine erfaringer, og vil dermed reflektere min subjektive forståelse av datamaterialet.

3.2 Forskningsmetodikk

Forskningsmetode er den overordnede tilnærmingen og prosessen for studien (Mackenzie & Knipe. 2006). Forskningsmetoden i denne studien er kvalitativ da formålet er å få en dypere forståelse for programmering i matematikkundervisning. Forskningsmetoden jeg anvender er en kollektiv kasusstudie (*Multiple casestudy*). Kasusstudier kjennetegnes ved å ha tydelige avgrenset enheter (*cases*) som skal analyseres og beskrives, og i denne studien er det de tre grunnbøkene som er det avgrensede materialet (Creswell & Poth, 2018). For å sikre helhet i forskningsprosessen vil studien ta utgangspunkt i Creswell og Poths (2018) modell for forskningsprosedyre av en kasusstudie, se figur 3.1.

Figur 3.1: Forskningsprosedyren for kasusstudie. (Oversatt og rekonstruert fra Creswell & Poth, 2018, s.101).

Ved på følge modellen for forskningsprosessen sikrer jeg som forsker kontinuitet gjennom hele forskningsarbeidet. Den første fasen involverer formulering av forskningsspørsmål. Forskningsspørsmålet fungerer som en retningsgivende målsetning for studien. Et kvalitativt forskningsspørsmål kjennetegnes ofte av sin åpne natur, med et fokus på «hva» og «hvordan» (Creswell & Creswell, 2018). I denne studien er forskningsspørsmålet formulert som «Hvordan har norske lærebøker for matematikk på 10. trinn implementert programmering med fokus på programmering som verktøy?» Dette spørsmålet ble utledet med bakgrunn i tidligere forskning som antyder at programmering ofte benyttes uten et betydelig matematisk utbytte (Kilhamn et al., 2021). For å undersøke implementeringen av programmering i skolesammenheng, ble tre lærebøker for 10. trinn valgt som studieobjekt, og en kasusstudie ble ansett som mest hensiktsmessig. Fase to av kasusstudier omfatter identifiseringen av studiens intensjon, altså hva forskeren søker å oppnå med studien. Herunder faller valg av utvalg for analyse. Fase en og to er grundig utdypet i introduksjonen av masteroppgaven (kap.1). Fase tre involverer utviklingen av en prosedyre for datainnsamling, denne fasen vil være mer utfordrende dersom studien tar i bruk intervjuer eller observasjoner. I en lærebokanalyse dreier denne fasen seg om å velge materiale som skal analyseres, altså hvilke lærebøker og deler av disse som skal analyseres. Dette blir ytterligere undersøkt i kapittel 3.3. Fase fire fokuserer på forberedelse av analysen, siden dette er en deduktiv studie, benyttes et eksisterende rammeverk. I fase fire presiseres kodekategoriene, slik at analyseverktøyet består av klart definerte og lukkede kategorier. Dette vil bli tydeligere belyst i kapittel 3.2 om dataanalyse og 3.4 om analyseskjema. Til slutt, fase fem som innebærer rapportering av analysen og tolkning av funnene, dette utgjør resultatdelen av forskningsoppgaven.

3.3 Utvalg

I dette delkapittelet vil jeg presentere utvalget for analysen, og gjøre reder for hvilke avgrensninger som er gjort med tanke på valg av datamaterialet.

3.3.1 Trinn

Forskningsspørsmålet fokuserer på kompetansemålet for 10. trinn, som understreker bruk av programmering for å utforske matematiske egenskaper og sammenhenger, av vesentlig betydning, da det klart definerer formålet med programmering (Kunnskapsdepartementet, 2019). Kompetansemålet er ikke knyttet til et bestemt matematisk tema, slik som kompetansemålet på 9.trinn. Det er derfor interessant å se hvilke temaer lærebøkene benytter programmering til.

En interessant tilnærming vil være å undersøke programmeringens rolle i matematikkundervisningen fra 1. til 10. trinn for å få et innblikk i bruk av programmering i hele grunnskolen. Imidlertid har begrensninger knyttet til omfanget av oppgaven nødvendiggjort en avgrensning, og derfor er fokuset primært rettet mot matematikkundervisning på 10. trinn.

3.3.2 Lærebøker

Lærebøkene anses som en integrert del av den potensielt implementerte læreplanen og har tradisjonelt hatt betydelig innflytelse på undervisningen (Valverde et al., 2002; Pepin et al., 2001; Fan & Pepin, 2021). Lærebøkene fungerer ofte som primærkilde for undervisning, både fordi elevene ofte arbeider med oppgaver fra lærebøkene og fordi lærerne ofte benytter eksempler og støtter seg på materiale fra lærebøkene (Johansson, 2006; Rezat, Fan & Pepin, 2021). Derfor er det av sentral betydning å analysere lærebøkene for å få innsikt i hvordan læreplanen er implementert (Valverde et al., 2001; Fan & Pepin, 2021).

De valgte studieobjektene (*cases*), lærebøkene, representerer de tre største lærebokforlagene i Norge, som angitt av Den norske forleggerforening (2022). Ifølge Forleggerforeningens bransjestatistikk har Cappelen Damm, Aschehoug og Gyldendal en samlet markedsandel på 98,2% innen læreboksegmentet i grunnskolen (Den norske forleggerforening, 2022). Selv om det ikke finnes eksplisitte data som forteller antall skoler som benytter lærebøkene fra hvert enkelt forlag, antas det, basert på markedsandelen, at lærebøkene utgitt av Aschehoug, Cappelen Damm og Gyldendal er de mest utbredte lærebøkene i den norske skolesektoren.

Innholdsanalysen tar utgangspunkt i grunnbøkene; Maximum 10, Matemagisk 10, og Matematikk 10 Grunnbok (Tofteberg et al., 2021; Kongsnes & Wallace, 2021; Hjardar & Pedersen, 2021), se figur 3.2. Alle disse bøkene ble utgitt i 2021 og angir at de er basert på LK20, og det er derfor relevant å analysere innholdet i lys av læreplanmålene for matematikk på 10. trinn i LK20.

	Maximum 10	Matemagisk 10	Matematikk 10 Grunnbok
Forfattere	Tofteberg, G.N., Tangen, J., Bråthe, L.T., & Stedøy, I.	Kongsnes, A.L., & Wallace, A.K.	Hjardar, E. & Pedersen, J.
Forlag	Gyldendal	Aschehoug Undervisning	Cappelen Damm
Utgitt	2021	2021	2021
Utgave og opplag	2.utgave, 1.opplag	1.utgave, 1.opplag	1.utgave, 1.opplag

Tabell 3.1: Oversikt over de tre lærebøkene som utgjør utvalget (Tofteberg et al., 2021; Kongsnes & Wallace, 2021; Hjardar & Pedersen, 2021)

En ytterligere motivasjon til å studere disse lærebøkene er at jeg av egen erfaring i praksis har sett at bøkene har ulike tilnærminger til implementeringen av programmering. For eksempel inkluderer Matemagisk 10 (Kongsnes & Wallace, 2021) oppgaver med programmering i hvert kapittel uavhengig av tema, mens Maximum 10 (Tofteberg et al., 2021) har dedikert en egen del til programmering på slutten av boken, og virker som har mindre programmering utover i boken. På den annen side har Matematikk 10 (Hjardar & Pedersen, 2021) integrert programmering i et separat digitalt hefte. Cappelen Damm Undervisning har informert på sine nettsider at formålet med å plassere programmeringsoppgavene i en ekstern digital del er å muliggjøre regelmessige oppdateringer, noe de hevder er nødvendig for å sikre kontinuerlig høy kvalitet på innholdet. Den versjonen av kapittel 0 Programmering (Hjardar, u.å.) som jeg baserer meg på, ble lastet ned 02. april 2024.

Samtlige programmeringsrelaterte oppgaver i grunnbøkene vil bli analysert. Dette innebærer flere gjennomlesninger av bøkene. Først vil bøkene bli skumlest i sin helhet, og oppgavene som inkluderer programmering vil bli notert. Deretter vil bøkene gjennomgås på nytt for å forsikre at ingen programmeringsoppgaver er oversett. Disse programmeringsoppgavene vil utgjøre utvalget og danne grunnlaget for analysen.

Det er viktig å erkjenne at lærebøker bare utgjør en del av det større samspillet i undervisningen. Lærere har tilgang til flere læreverk og andre ressurser, og gjennom å kombinere disse kan de tilpasse undervisningen for å dra nytte av det beste fra hver ressurs (Valverde et al., 2002; Pepin et al., 2001). Til tross for at det eksisterer flere skolerelevante ressurser innen programmering, har jeg i denne studien valgt å avgrense oppgaven til lærebøkene fra de tre største lærebokforlagene. Da de fysiske lærebøkene fremdeles utgjør en sentral rolle i undervisningen (Valverde et al., 2002; Rezat et al., 2021). Lærebøkenes sentrale rolle i norsk skole ble tydeliggjort gjennom regjeringens millionsatsing på fysiske lærebøker (Utdanningsnytt, 2024).

3.4 Dataanalysen

Det finnes to typer kvalitative dataanalyser; tematisk og innholdsanalyse. I denne studien vil datamaterialet analyseres gjennom en kvalitativ innholdsanalyse. I en innholdsanalyse vil dataanalysen gå ut på å organisere mengden tekst inn i et par kategorier. Kategorier er mønstre eller temaer. Teksten kan enten direkte nevne tema eller tolkes til å falle under tema (Hsieh & Shannon, 2005). Creswell og Creswell (2018) har utarbeidet en prosedyre for kvalitativ dataanalyse, fra rå data til studiens funn, og den vil være det overordnede utgangspunktet for dataanalysen.

Prosedyren for kvalitativ dataanalyse starter med å organisere og forberede datamaterialet til analysen. Dette steget inkluderer transkribering av intervjuer, dokumentskanning, og arrangering av datamaterialet (Creswell & Creswell, 2018). Deretter skal all dataen leses over og det skrives notater på makronivå: generelle ideer tekstene bygger på, førsteinntrykk, kredibilitet og brukbarhet. Deretter starter kodeprosessen. Å kode data vil si å organisere materialet del for del, og merke den basert på forhåndsbestemte kategorier (Creswell & Creswell, 2018). Kodeprosessen brukes til å genere en beskrivelse og temaer basert på datamaterialet. Beskrivelsen inkluderer en detaljert gjengivelse av informasjon om dokumentet. Temaer derimot er utgangspunktet for hovedfunnene, og identifiseres ut fra kodingen (Creswell & Creswell, 2018). I en kasusstudie blir temaene analysert for hvert dokument. Siste steget er å finne en oversiktlig måte å representer beskrivelsen og temaene på. Representasjonen kan være visuell gjennom figurer, diagrammer og tabeller eller ved tekst (Creswell & Creswell, 2018).

3.4.1 Innholdsanalyse

Den vanligste metoden for læreboks forskning er innholdsanalyse (Rezat & Strässer, 2017). Innholdsanalyse er en overordnet betegnelse for flere analytiske tilnærminger (Hsieh & Shannon, 2005). Målet med en innholdsanalyse er å skaffe kunnskap og forståelse for et fenomen gjennom å analysere et dokument, en bok eller et annet skriftlig materiale (Rezat & Strässer, 2017; Hsieh & Shannon, 2005). En innholdsanalyse kan være både kvantitativ og kvalitativ. Kvalitativ innholdsanalyse defineres som en forskningsmetode for subjektive tolkninger av dokumentet, gjennom systematiserte klassifiseringer av koding og ved å identifisere temaer og mønstre (Hsieh & Shannon, 2005). Målet med kvalitativ innholdsanalyse er å oppnå en dypere forståelse av forskningsobjektet (Creswell & Creswell, 2018). Kvantitativ innholdsanalyse derimot har formål om å kartlegge omfanget av et tema, et begrep eller en metode.

I denne studien er målet å forstå hvordan kompetansemålet for 10.trinn «utforske matematiske egenskaper og sammenhenger ved å bruke programmering» er forstått og implementert i norsk lærebøker (Kunnskapsdepartementet, 2019). Creswell og Creswell (2018) fremhever flere fordeler ved bruk av innholdsanalyse. For det første gir innholdsanalyse mulighet til å observere et fenomen i sin naturlige form, der forskeren ikke påvirker datamaterialet som for eksempel ved intervju eller observasjon, der forskerens tilstedeværelse kan påvirke intervjuobjektet/observasjonen. Innholdsanalyse gir forskeren mulighet til å analysere råmaterialet, uten transkripsjoner, noe som bidrar til å gjøre prosessen mer transparent, ettersom leseren kan få tilgang til datamaterialet, og dermed validere funnene. Å analysere råmaterialet er fleksibelt, siden materialet er tilgjengelig til enhver tid, samt at det er tidsbesparende fordi forskeren unngår

transkripsjon (Creswell & Creswell, 2018). I denne studien vil jeg begrense analysen til oppgaver som omhandler programmering, faren med det er å ignorere sammenhengen med andre tematiske områder (Charalambous et al., 2010). For å imøtekomme denne utfordringen har jeg vurdert min tilnærming til innholdsanalysen, og dette vil bli utdypet i de påfølgende delkapitlene.

3.4.2 Tilnærminger til innholdsanalyse

Hiseh og Shannon (2005) presenterer tre tilnærminger til innholdsanalyse; konvensjonell, teoridrevet og summativ innholdsanalyse, her med Fauskanger og Mosvolds oversettelser (Hiseh & Shannon, 2005; Fauskanger & Mosvold, 2014).

Den første tilnærmingen; konvensjonell innholdsanalyse, brukes i studier med mål om å beskrive et fenomen, der det eksisterer begrenset med tidligere forskning og teori. I en konvensjonell innholdsanalyse brukes ofte en induktiv tilnærming der kategoriene for koding av data blir utviklet i analyseprosessen. Analyseprosessen i en konvensjonell innholdsanalyse fokuserer på refleksjoner heller enn nøkkelbegreper (Hiseh & Shannon, 2005). Resultatet av en konvensjonell innholdsanalyse kan være en modell eller et konsept (Hiseh & Shannon, 2005; Fauskanger & Mosvold, 2014). Utfordringen ved en konvensjonell tilnærming er at det er vanskelig å få en fullstendig forståelse for konteksten, og at en kan overse viktige sammenhenger og kategorier (Fauskanger & Mosvold, 2014).

Den andre tilnærmingen, summativ innholdsanalyse, starter med å identifisere begreper/innhold i teksten med mål om å skape forståelse for en kontekstuell bruk av begrepene eller innholdet (Hiseh & Shannon, 2005). I denne tilnærmingen teller forskeren opp antall ganger begrepet blir nevnt, i ett forsøk på å undersøke bruken av begrepet i praksis. En summativ innholdsanalyse kan ved første øyekast virke kvantitativ, men summative innholdsanalyser er kvalitative fordi analysen går forbi tallene og har som mål å skape forståelse av innholdet. Denne tilnærmingen kan være utfordrende fordi den gir innsikt i hvordan begrepene blir brukt, men er begrenset med tanke på å forstå betydningen av begrepene på dypere plan. En slik tilnærming kan derfor alene bli overfladisk (Hiseh & Shannon, 2005; Fauskanger & Mosvold, 2014).

Den tredje tilnærmingen, teoridrevet innholdsanalyse, baserer seg på eksisterende teori. Formålet med teoridrevet innholdsanalyse er å validere eller utvide et teoretiske rammeverk/teori eller å se teorien i praksis. Teoridrevet innholdsanalyse bidrar til utvikling av forskningsfeltet. I en teoridrevet innholdsanalyse er kodingskategoriene basert på eksisterende teori, noe som indikerer at metoden er deduktiv. Likevel er teoridrevet innholdsanalyse åpen for at forskeren kan endre kodingskategoriene i analysen, og dermed få en mer abduktiv tilnærming (Fauskanger, J. & Mosvold, R., 2014). Utfordringene med teoridrevet innholdsanalyse er at forskerens fokus på teorien gjør at forskeren kan få et bias og med det overse viktige sammenhenger og aspekter (Fauskanger & Mosvold, 2014).

De tre tilnærmingene har alle sine begrensinger, så for å optimalisere innholdsanalysen kreves en kombinasjon av de tre tilnærmingene (Hiseh & Shannon, 2005; Fauskanger & Mosvold, 2014). Denne studien har tatt utgangspunkt i en kombinasjon, med hovedvekt på en teoridrevet tilnærming. Den teoridrevne tilnærmingen brukes gjennom å benytte forhåndsbestemte kategorier for koding basert på eksisterende teori, i denne studien: PRIMM (Coleman, 2021) og relasjonen mellom programmering og matematikk (Kilhamn et al., 2021). Videre har benyttet en summativ tilnærming ved å kvantifisere funnene og

presentere dem sammenfattet i form av tabeller. Dette vil gi en systematisk oversikt over fordelingen av oppgaver innenfor de teoridrevne kategoriene.

3.4.3 Horisontal og vertikal analyse

For å strukturere analysen har jeg benyttet et analyseverktøy for læreboksforskning utviklet av Charalambous et al. (2010). Analyseverktøyet har formål å undersøke læringsmulighetene i læreverket. En kan si at læreboksforskning er en mer spesifikk form for innholdsanalyse rettet mot læreverk brukt i læringssituasjoner. Analyseverktøyet består av to deler; horisontal og vertikal analyse. Charalambous et al. (2010) mener at et rammeverk bør inneholde begge typer analyse for å se helheten av læreverket.

Horisontal analyse tar for seg den brede delen av analysen, og handler om å få en oversikt over læreverket i form av tittel, forfatter, kapittelinndeling, antall sider per tema, og antall oppgaver (Charalambous et al., 2010). En slik analyse er overfladisk og ser ikke på kvaliteten av innholdet i boken. Likevel er det en viktig del av analysen for å få oversikt over strukturen i læreverkene og for å se helheten i bøkene (Charalambous et al., 2010). I denne oppgaven har jeg gjort noen justeringer i rammeverket, så den horisontale analysen har tatt for seg to hoveddeler: Informasjon om læreboken og lærebokens struktur, se tabell 3.2.

HORISONTAL ANALYSE AV LÆREBØKER	
Lærebokens informasjon	**Lærebokens struktur**
• Tittel • Forfatter og redaktører • Forlag • Utgivelsesår • Utgave og opplag • Tilleggsmateriale	• Antall kapitler • Kapittelnavn • Antall sider • Antall programmeringsoppgaver per kapittel • Andre vedlegg

Tabell 3.2: En tilpasset og norsk oversatt versjon av den horisontale delen av rammeverket fra Charalambous et al. (2010) s. 123.

Den mer grundige gjennomgangen skjer i den vertikale analysen. I den vertikale analysen setter forskeren seg inn i hvordan et emne eller konsept presenteres i læreverket. I denne studien; hvordan programmering implementeres i læreverket. Det er denne delen av analysen som studerer hvordan læreverket er skaper læring, og analysen har som mål å se på læringsmulighetene i læreboken (Charalambous et al., 2010). Den vertikale analysen er delt inn i todeler: kommunisert til elevene og forventet av elevene. Den kommuniserte delen av analysen handler om hvordan læreboken er formulert. Denne delen er delt i tre; matematisk innhold, matematiske praksiser og holdninger (Charalambous et al., 2010). Denne studien har sett på det matematiske innholdet, hvilke matematiske temaer og konsepter som er koblet til programmering. Den andre delen av det kommuniserte tar for seg praksisene, hvilke(t) programmeringsspråk lærebøkene tar i bruk. Del to tar for seg forventningene lærebøkene har til elevene, altså hvilke forkunnskaper elevene forventes å ha, så kalt kognitive krav, og hva slags respons oppgavene krever av elevene (Charalambous et al., 2010). Se tabell 3.3.

VERTIKAL ANALYSE AV LÆREBØKER	
Kommunisert til elevene	**Forventet av elevene**
Matematisk innhold • Relasjon mellom matematikk og programmering	• Typer respons (PRIMM) • Samarbeid
Matematiske praksiser • Programmeringsspråk	

Tabell 3.3: En tilpasset og norsk oversatt versjon av den vertikale delen av rammeverket fra Charalambous et al. (2010) s. 123.

Med de tilpasningene som er vist i tabell 3.2 og 3.3, har jeg tilpasset et overordnet rammeverk for strukturen av analysen. Den vertikale analysen vil være teoridrevet, og derfor har jeg i analysen av matematisk innhold benyttet rammeverket til Kilhamn et al. (2021) for å se på relasjonen mellom matematikk og programmering. Under «forventet av elever» har jeg anvendt PRIMM (Coleman, 2021) for å analysere hvilke krav oppgavene stiller til elevene og hvilken respons oppgaven forventer. Når det gjelder matematiske praksiser, undersøker jeg hvilke(t) programmeringsspråk oppgavene benytter, denne analysen er gjort induktivt.

Charalambous et al. (2010), har en tredje kategori: koblinger, som handler om hvordan læreboken blir brukt i praksis, både i undervisning og til omverdenen. Denne studien har ikke tatt for seg bruken av lærebøkene i praksis, og det aspektet faller derfor bort.

3.5 Analyseskjema

Analyseskjema er notater fra innholdsanalysen, og baserer seg på de to rammeverkene over; Relasjon mellom matematikk og programmering og PRIMM. Oppsettet til analyseskjema er likt per lærebok. Den første delen av analyseskjema består av generell informasjon om oppgaven slik at det er mulig å finne tilbake. Videre noterte jeg kapittelnavn for å se i hvilken matematisk sammenheng oppgaven er gitt. Videre er analyseskjema delt inn i en kolonne for programmeringsspråk, en kolonne for relasjon mellom matematikk og programmering og en kolonne for PRIMM. I kapittel 3.5 presenteres konkrete eksempler fra analysen og bruk av analyseskjema.

Den vertikale analysen består av tre delanalyser. Delanalyse 1 tar for seg relasjonen mellom programmering og matematikk i henhold til Kilhamn et al. (2021) sin kategorisering (se 2.2). Delanalyse 2: hva oppgavene spør om i forhold til PRIMM. Delanalyse 3: hvilke(t) programmeringsspråk oppgavene benytter.

3.5.1 Delanalyse 1: Relasjon mellom matematikk og programmering

For å undersøke forholdet mellom matematikk og programmering, har jeg presisert og spesifisert rammeverket fra Kilhamn et al. (2021) med mål om å etablere fire distinkte

og uavhengige kategorier. Bråting og Kilhamn (2021) brukte rammeverket til å utføre en lignende studie i Sverige. De møtte da på en utfordring ved at alle oppgavene med programmering som ble kategorisert som verktøy, også falt under kategorien *matematikk som en kontekst.* For å optimalisere rammeverket slik at det kan fungere som distinkte kategorier for koding, har jeg inkludert to ekstra elementer for hver kategori, nemlig kjennetegn og læringsutbytte. Disse elementene er basert på inspirasjon fra Kilhamn et al. (2021) og Gjøvik og Høyland (2022). Se tabell 3.4.

Kategori	Forklaring	Kjennetegn	Læringsutbytte
Programmering **uten kobling** til matematikk.	Disse leksjonene handler kun om programmering	Programmeringsteknisk	Programmering
Matematikk som **en kontekst** for programmering	Disse leksjonene tar for seg matematikk, og bruker matematikk for å lære seg programmering. Det læres ingen ny matematikk, men programmering kan utnyttes for å repetere eller styrke matematikk-kunnskaper.	Programmeringsteknisk med en matematisk kontekst, eks. bruk vilkår og løkker til å lage et program som skriver ut partall	Programmering
Programmering som **et verktøy** for å **effektivisere** matematiske handlinger.	Disse leksjonene har et tydelig matematikkinnhold. En datamaskin programmeres og anvendes som et effektivt sett for å utføre beregninger.	Lage/bruke et program for å regne eller modellere matematikk	Mengdetrening i matematikk
Programmering som **et verktøy** for å **utforske** matematikk	Disse leksjonene anvender programmering for å utforske matematiske konsepter og sammenhenger. Programmering tilfører ny tilnærming til matematikk og angir informasjon slik at en kan få en dypere matematisk forståelse.	Lage/bruke programmering for å utforske et matematisk konsept	Dybde læring i matematikk

Tabell 3.4: En videreutvikling av tabell 2.2. Et rammeverk for å analysere relasjonen mellom programmering og matematikk i undervisning (Kilhamn et al., 2021, s. 293)

For at analyseskjema skal romme hele datamaterialet, måtte jeg gjøre en endring i kategori 3. Kategorien *Programmering som et verktøy for å effektivisere matematiske beregninger* har jeg endret til *Programmering som et verktøy for å effektivisere matematiske handlinger.* Dette på bakgrunn av at jeg i test av analyseverktøyet så at flere oppgaver brukte programmering som et verktøy for å utføre matematiske handlinger som graftegning, og modellering uten å gjøre beregninger. Slik oppgaver som ikke kvalifiserer til utforskning slik det er forstått av Maaß & Reitz-Koncebovski (2013), bør også kategoriseres og derfor har jeg valgt å utvide kategorien til *Programmering som et verktøy for å effektivisere matematiske handlinger.*

3.5.2 Delanalyse 2: PRIMM

For å få innblikk i hvilke ferdigheter oppgavene krever av elevene og hvilke ferdigheter de lærer bort, har jeg gjennomført en analyse i henhold til PRIMM. Rammeverket PRIMM er beskrevet av Coleman (2021), Matematikksenteret (u.å.a) og Gjøvik og Høyland (2022). Deres tolkning av PRIMM er utgangspunktet for analyseverktøyet vist i tabell 3.5

Kategorier	Forklaring	Kjennetegn	Elevenes besvarelse
Forutse	Elevene blir spurt om å forutse hva programmert gjør. En slik oppgave krever et ferdigskrevet program/kode.	Ferdigskrevet kode. «forklar», «Hva skjer når koden kjøres?»	Skriftlig forklaring, eller en tegning
Kjøre	Elevene blir bedt om å kjøre programmet	«kjør»	–
Undersøke	Elevene utforsker koden ved å forklare koden linje for linje, følge en variabel eller feilsøke.	«Utforsk», «Feilsøke», «forklar koden», «hva skjer med variabelen a?»	Skriftlig forklaring, eller en tegning
Endre	Elevene redigerer koden, enten for å optimalisere eller legge til en ekstra funksjon.	«Endre», «Legg til en funksjon», «Fiks feilen»	Koding
Lage	Elevene skal lage et helt nytt program fra bunn av.	«Lag et program som ...»	Koding

Tabell 3.5: En videreutvikling av tabell 2.1. Et rammeverk for å analysere programmeringsoppgaver etter PRIMM.

3.4.3 Delanalyse 3: Programmeringsspråk

For å undersøke hvilke programmeringsspråk oppgavene benytter seg av, samt hvordan fordelingen er mellom oppgaver som angir et spesifikt programmeringsspråk og de som gir elevene valgmuligheter, vil delanalyse 3 være en induktiv analyse av programmeringsspråk. Analysen blir induktiv ettersom kategoriene blir definert i analysen. Dette fordi det finnes flere tusen programmeringsspråk (Kristensen & Kirfel, 2022).

3.6 Analyseprosessen

Etter å ha utviklet et analyseskjema for den vertikale analysen, gikk jeg gjennom de tre lærebøkene, oppgave for oppgave. Analyseprosessen startet med å gjøre en horisontal analyse. Lærebokens informasjon ble lest av fra hver lærebok, og lagt inn i analyseskjema tilsvarende tabell 3.2. Deretter så jeg på lærebøkenes struktur. Den analyseringen ble gjort ved å lese lærebok for lærebok og markere alle oppgavene med og om programmering. Oppgavene ble så telt opp kapittel for kapittel. Den vertikale analysen ble gjort ved å kategorisere alle deloppgavene med programmering etter analyseskjema tilsvarende tabell 3.3. Bøkene ble analysert hver for seg med likt analyseskjema. Den vertikale analysen er tre-delt, der del 1 ser på relasjon mellom matematikk og programmering etter Kilhamn et al. (2021), del 2 ser på PRIMM, og del 3 ser på programmeringsspråk.

Jeg møtte på to utfordringer i analyseprosessen. Først og fremst at kategorien i noen tilfeller var overlappende, og derfor vanskelig å plassere. Det er ikke en uvanlig utfordring å møte på i en deduktiv analyse (Creswell & Poth, 2018). For å sikre at alle oppgavene ble analysert med samme utgangspunkt, gikk jeg over alle oppgavene på nytt, hver gang jeg endret en kode. Underveis i analysen oppdaget jeg at det var interessante detaljer om oppgavene som ikke ble fanget opp av de forhåndsbestemte kategoriene, og jeg la derfor til en delanalyse underveis samarbeid. Samarbeid er en sentral del av arbeidsmetoder for algoritmisk tenkning, slik Udir (2019) beskriver det.

Under vil jeg presentere eksempler på hvordan oppgavene er analysert. Dette for at leseren skal få innblikk i hele prosessen, som kan bidra til økt validitet og relabilitet (Creswell & Creswell, 2018).

3.6.1 Delanalyse 1: Relasjon mellom matematikk og programmering

I delanalyse 1 har jeg sett på relasjonen mellom matematikk og programmering i deloppgavene. For å få et klart innblikk i relasjonen må deloppgavene ses i sammenheng, med det sagt kan en oppgave bestå av ulike grader av relasjon, så alle deloppgavene er ikke nødvendigvis kategorisert som det samme. I delanalyse 1 presenteres en oppgave fra hver kategori, med en begrunnelse på hvorfor oppgaven er kategorisert slik den er.

3.6.1.1 Programmering uten kobling til matematikk

Denne kategorien tar for seg oppgaver om programmering, det vil si *uten kobling til matematikk,* og uten *matematikk som kontekst* (Gjøvik & Høyland, 2022; Kilhamn et al., 2021). Oppgavene gir ikke læringsutbytte i matematikk, kun programmering. Et eksempel på slik oppgave er vist i figur 3.2. Denne kategorien kan være vanskelig å skille fra programmering med matematikk som kontekst, fordi til tross for at deloppgave 3 er rent programmeringsteknisk tar programmet som det henvises til for seg matematikk. Programmet regner ut veksten av et innskudd etter x antall år, så om en ser hele oppgaven som en helhet, og ikke bare på deloppgave 3, vil en kunne argumentere for at det finnes en matematisk kontekst. Det mest sentrale med relasjon mellom programmering og matematikk er læringsutbytte (Gjøvik & Høyland, 2022; Kilhamn et al., 2021), og i deloppgave 3 vil læringsutbytte være programmeringsteknisk og ikke matematisk, dermed kategoriseres oppgaven herunder.

2. Her ser du et Python-program.

```
1   innskudd = 50000
2   vekstfaktor = 1.035
3   for x in range(1, 11):
4       sum = innskudd * vekstfaktor**x
5       print(f"Etter {x} år har {innskudd} kr vokst til {sum} kr.")
6
```

*I linje 5 bruker vi Python-funksjonen **f** (streng-funksjon) for å printe ut en rekke med verdier et gitt antall ganger.*

Skriv inn programmet ovenfor og test det.

3. Forklar hvorfor vi må skrive **range(1, 11)** og ikke **range(1, 10)** i linje 3 for å få programmet til å kjøre løkken ti ganger.

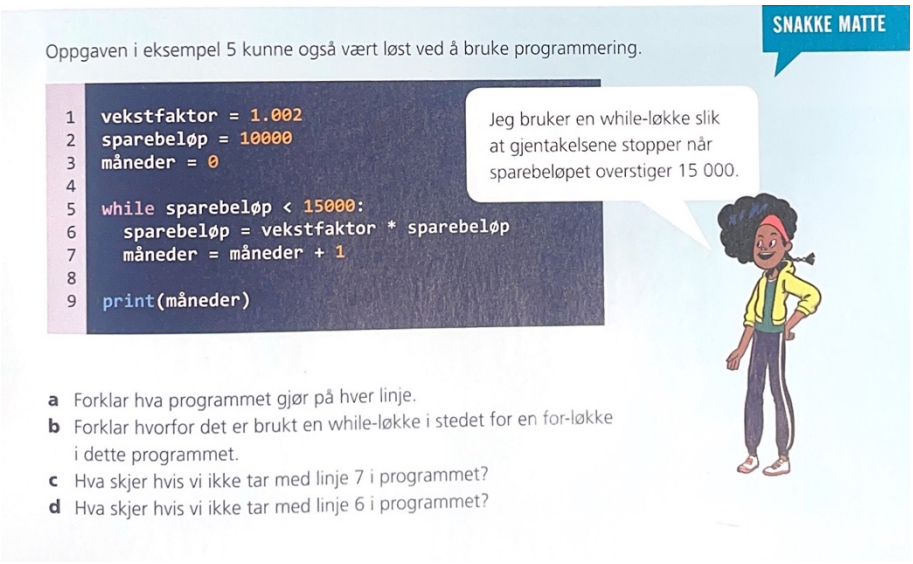
Forklar her:

Figur 3.2: Oppgave fra Matematikk 10 (Hardar, 2021, s.61)

3.6.1.2 Matematikk som en kontekst for programmering

Denne kategorien har som programmering uten kobling til matematikk fokus på å utvikle ferdigheter i programmering. I oppgaven vist i figur 3.3 er konteksten økonomi i form av renter, og underliggende; eksponentiallikninger. Oppgaveteksten lyder som følger: «Michal setter inn 10 000kr på sparekonto. Renten på sparekontoen er 0,2% per måned. Hvor mange måneder går det før beløpet har vokst til 15 000kr» (Kongsnes & Wallace, 2021, s.138). Oppgaven er løst med et program som elevene kan kjøre for å finne svaret. Deloppgavene tar for seg det programmeringstekniske, men med sparing som kontekst. Elevene kan derfor svare på oppgavene i konteksten den er gitt i, altså ikke bare kommentere koden programmeringsteknisk, men se det i sammenheng med Michals sparekonto.

Hvorvidt hele oppgaven er i matematisk kontekst kan diskuteres, og oppgave b) kan minne om oppgaven i figur 3.2, som jeg kategoriserte som *programmering uten kobling til matematikk*. Helt umiddelbart vil en tenke at de to oppgavene burde kategoriseres som det samme, men svaret på oppgave b kan kobles til matematikk ved å inkludere en økonomisk forståelse av at pengene vokser så lenge de er på konto, og renters rente. Gitt at elevene besvarer deloppgavene i sammenheng med Michals sparekonto så vil denne oppgaven falle under kategorien *matematikk som en kontekst for programmering*.



SNAKKE MATTE

Oppgaven i eksempel 5 kunne også vært løst ved å bruke programmering.

```
1  vekstfaktor = 1.002
2  sparebeløp = 10000
3  måneder = 0
4
5  while sparebeløp < 15000:
6    sparebeløp = vekstfaktor * sparebeløp
7    måneder = måneder + 1
8
9  print(måneder)
```

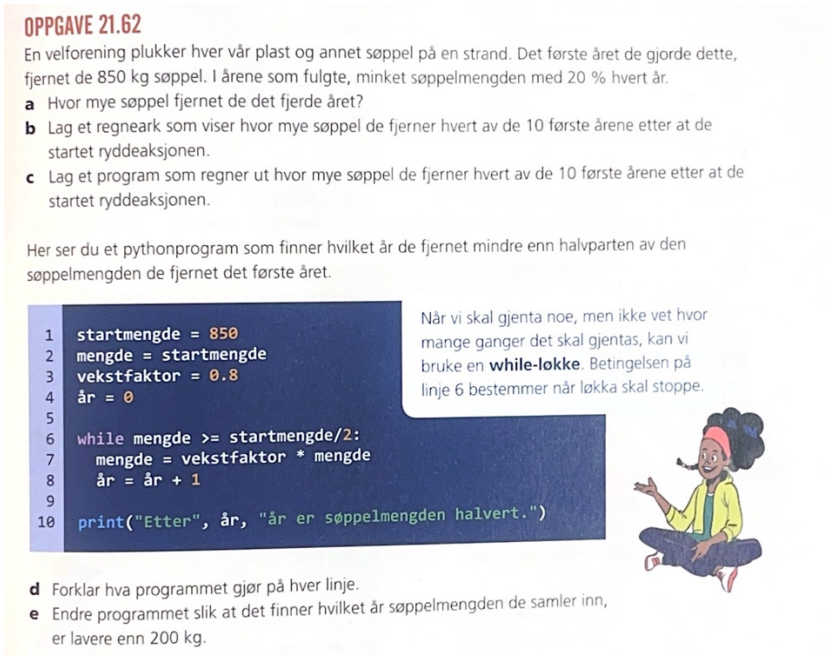
Jeg bruker en while-løkke slik at gjentakelsene stopper når sparebeløpet overstiger 15 000.

a Forklar hva programmet gjør på hver linje.
b Forklar hvorfor det er brukt en while-løkke i stedet for en for-løkke i dette programmet.
c Hva skjer hvis vi ikke tar med linje 7 i programmet?
d Hva skjer hvis vi ikke tar med linje 6 i programmet?

Figur 3.3: Oppgave fra Matemagisk 10 (Kongsnes & Wallace, 2021, s.139)

3.6.1.3 Programmering som et verktøy for å effektivisere matematiske handlinger

Denne kategorien skiller seg fra de to over fordi læringsutbytte ikke er programmeringsteknisk. Oppgaver som bruker *programmering som et verktøy for å effektivisere matematiske handlinger* kjennetegnes ved at de legger opp til å bruke programmering for å finne et matematisk svar (Gjøvik & Høyland, 2022; Kilhamn et al., 2021). Oppgave 21.62, vist i figur 3.4 er et eksempel på en slik oppgave. Oppgaven ber elevene lage et program som regner ut hvor mye søppel de fjerner gjennom søppelaksjonene. Elevene lager da et program som er et verktøy for å beregne søppeltømming. Programmet bidrar til å lette utregningene for elevene. Grunnen til at denne oppgaven ikke kategoriseres som et *verktøy for å utforske matematikk* er at oppgaven ikke fører til at elevene får en dypere matematisk forståelse. Elevene anvender matematikk, men utforsker ingen matematiske konsepter eller sammenhenger.



OPPGAVE 21.62

En velforening plukker hver vår plast og annet søppel på en strand. Det første året de gjorde dette, fjernet de 850 kg søppel. I årene som fulgte, minket søppelmengden med 20 % hvert år.

a Hvor mye søppel fjernet de det fjerde året?
b Lag et regneark som viser hvor mye søppel de fjerner hvert av de 10 første årene etter at de startet ryddeaksjonen.
c Lag et program som regner ut hvor mye søppel de fjerner hvert av de 10 første årene etter at de startet ryddeaksjonen.

Her ser du et pythonprogram som finner hvilket år de fjernet mindre enn halvparten av den søppelmengden de fjernet det første året.

```
1  startmengde = 850
2  mengde = startmengde
3  vekstfaktor = 0.8
4  år = 0
5
6  while mengde >= startmengde/2:
7    mengde = vekstfaktor * mengde
8    år = år + 1
9
10 print("Etter", år, "år er søppelmengden halvert.")
```

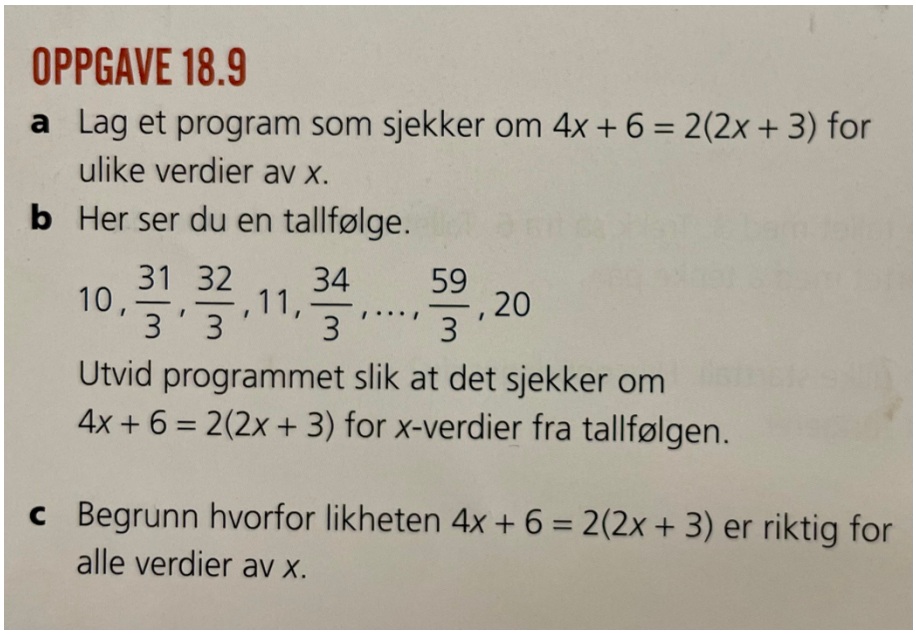
Når vi skal gjenta noe, men ikke vet hvor mange ganger det skal gjentas, kan vi bruke en **while-løkke**. Betingelsen på linje 6 bestemmer når løkka skal stoppe.

d Forklar hva programmet gjør på hver linje.
e Endre programmet slik at det finner hvilket år søppelmengden de samler inn, er lavere enn 200 kg.

Figur 3.4: Oppgave fra Matemagisk 10 (Kongsnes & Wallace, 2021, s.117)

3.6.1.4 Programmering som et verktøy for å utforske matematikk.

Den siste kategorien likner på kompetansemålet for 10.trinn om å bruke programmering til å utforske matematiske sammenhenger og egenskaper (Kunnskapsdepartementet, 2019). For at en oppgave skal kunne kategoriseres som et *verktøy for å utforske matematikk*, må oppgaven legge opp til at elevene undersøker et problem, og gjerne generaliserer. I oppgave 18.9 (figur 3.6) skal elevene utforske den distributive loven, som er en av *the big ideas* i matematikk. I oppgave 18.9a+b skal elevene lage et program som sjekker hvilke x-verdier-likningen $4x + 6 = 2(2x + 3)$ stemmer for. Disse to deloppgavene faller under *programmering som verktøy for å effektivisere matematiske handlinger*, da programmet gjør utregningen for elevene. 18.9c derimot er kategorisert som programmering som et verktøy for å utforske matematikk. Med bakgrunn i beregningene fra 18.9a+b, ber 18.9c elevene om å begrunne hvorfor likheten $4x + 6 = 2(2x + 3)$ stemmer for alle verdier av x. Oppgaven krever at eleven evner å begrunne, en ferdighet som kan tyde på høy matematisk forståelse. Enkelte elever med god kjennskap til den distributive loven og faktorisering, vil relativt lett kunne se at uttrykket til venstre er likt utrykket høyre, og at forskjellen er at uttrykke til høyre er faktorisert ved å trekke 2 utenfor parentesen. For elever som ser denne sammenhengen raskt, vil oppgaven være relativt lett, og dermed ikke bidra til en dypere matematisk forståelse. Likevel tror jeg at ved å legge opp til begrunnelse vil elevene kunne utfordre seg selv til å formulere et godt bevis eller en god begrunnelse og det kan bidra til økt matematisk forståelse.



OPPGAVE 18.9

a Lag et program som sjekker om $4x + 6 = 2(2x + 3)$ for ulike verdier av x.

b Her ser du en tallfølge.

$$10, \frac{31}{3}, \frac{32}{3}, 11, \frac{34}{3}, \ldots, \frac{59}{3}, 20$$

Utvid programmet slik at det sjekker om $4x + 6 = 2(2x + 3)$ for x-verdier fra tallfølgen.

c Begrunn hvorfor likheten $4x + 6 = 2(2x + 3)$ er riktig for alle verdier av x.

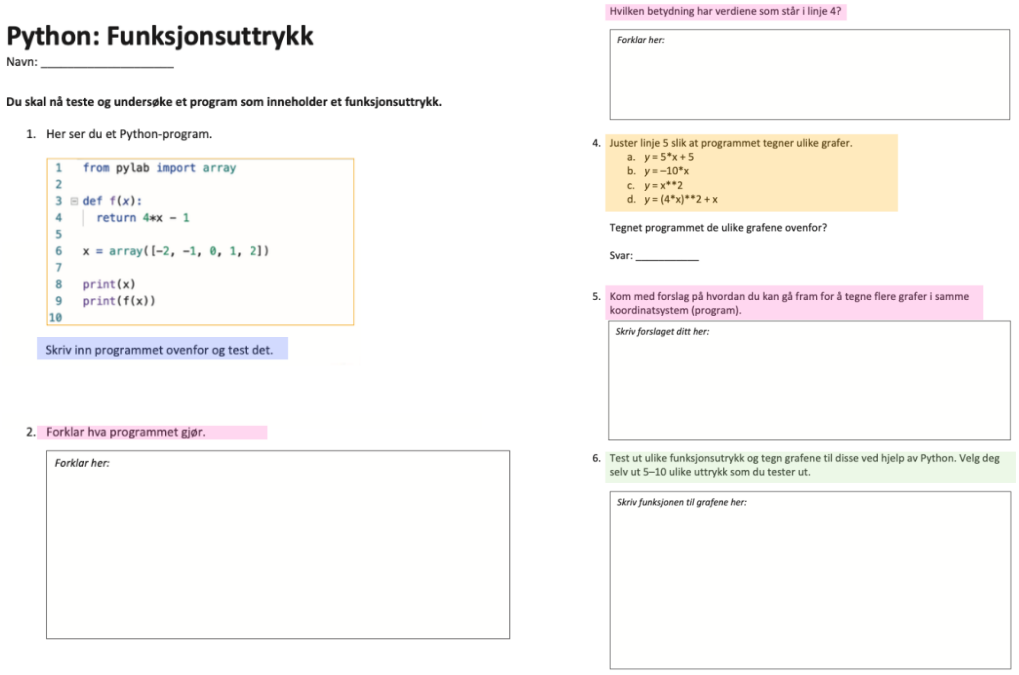
Figur 3.5: Oppgave fra Matemagisk 10(Kongsnes & Wallace, 2021, s.14)

3.6.2 Delanalyse 2: PRIMM

Delanalyse 2 tar utgangspunkt i tabell 3.5 med mål å få oversikt over hva lærebøkene krever av elevene i programmering. Under presenteres to eksempler fra analysen av datamaterialet. Fargene betyr følgende; oransje – forutse, blå –kjøre, rosa – undersøke, gul – endre og grønn – lage. Denne analysen er mer intuitiv enn relasjon mellom matematikk og programmering, da kjennetegnene er lett gjenkjennelige, som gjør det lettere å plassere oppgavene, se tabell 3.5.

3.6.2.1 Eksempel 1

Figur 3.6 er et godt eksempel på hvordan jeg kategoriserte deloppgavene. Oppgave 1 handler om å kjøre koden, og er derfor markert blått, som i *kjør*. Oppgave 2 spør om en forklaring av koden, og hadde denne kommet før eleven kjørte programmet ville den kunne kategoriseres som *forutse*, men siden elevene har kjørt koden vil det være en del av å *undersøke*, det samme gjelder oppgave 3. Oppgave 4 går kategoriseres som *endre*, da den ber eleven justere programmet til å tegne fire andre grafer. Oppgave 5 er kategorisert som *undersøke*, her skal elevene komme med forslag til hvordan de kan videreutvikle programmet til å tegne flere grafer i samme koordinatsystem. Dersom elevene hadde blitt bedt om å utføre disse endringene, ville oppgaven falt under kategorien *endre*. Oppgave 6 har jeg kategorisert som *lag.* Det står ikke eksplisitt at elevene skal bruke programmet oppgitt og de står dermed fritt til å lage et nytt program, gjerne med elementer fra den forhåndsskrevede koden. Dersom oppgaveteksten hadde bedt elevene bygge videre på koden, ville oppgave 6 blitt kategorisert som *endre*.



Python: Funksjonsuttrykk

Navn: _____

Du skal nå teste og undersøke et program som inneholder et funksjonsuttrykk.

1. Her ser du et Python-program.

```
1  from pylab import array
2
3  def f(x):
4      return 4*x - 1
5
6  x = array([-2, -1, 0, 1, 2])
7
8  print(x)
9  print(f(x))
10
```

Skriv inn programmet ovenfor og test det.

2. Forklar hva programmet gjør.

Forklar her:

Hvilken betydning har verdiene som står i linje 4?

Forklar her:

4. Juster linje 5 slik at programmet tegner ulike grafer.
   a. $y = 5*x + 5$
   b. $y = -10*x$
   c. $y = x**2$
   d. $y = (4*x)**2 + x$

Tegnet programmet de ulike grafene ovenfor?

Svar: _____

5. Kom med forslag på hvordan du kan gå fram for å tegne flere grafer i samme koordinatsystem (program).

Skriv forslaget ditt her:

6. Test ut ulike funksjonsutrykk og tegn grafene til disse ved hjelp av Python. Velg deg selv ut 5–10 ulike uttrykk som du tester ut.

Skriv funksjonen til grafene her:

Figur 3.6: Oppgave hentet fra Matematikk 10, og analysert med PRIMM (Hardar, 2021, s.49-50)

3.6.2.2 Eksempel 2

Eksempelet fra Maximum 10, vist i figur 3.7, viser godt hvordan en deloppgave kan bestå av flere kategorier. 4.30a starter med at eleven skal tolke koden, og siden det skjer før koden kjøres kan en si at dette er *forutse*. Neste del av oppgaven er å undersøke koden, der elevene skal gjøre et dypdykk i koden. Videre skal elevene videreutvikle programmet, og det å undersøke utviklingsmulighetene til programmet kategoriseres som *undersøke*, mens selve endringene i koden kategoriseres som *endring*. Deloppgave 4.30a kategoriseres dermed som *forutse*, *undersøke* og *endre*.

Deloppgave 4.30b er nok et eksempel på oppgaver som tilhører flere kategorier. Oppgaven ber elevene lage et nytt program om et kortspill. Å lage et nytt program kategoriseres som *lage*. Diskusjonen etterpå er *undersøke*, og utvidelsen av programmet er *endre*. Den siste oppgaven 4.30c handler om å lage et helt eget program basert på et spill elevene velger, og er derfor også *lage*. Noe som er verdt å merke seg i denne

oppgaven er progresjonen. PRIMM-modellens progresjon kommer godt frem i denne oppgaven, der elevene først setter seg inn i et gitt program, så endrer det, før de anvender elementene i et eget program.

4.30 Digitaliser et analogt spill

a Denne koden viser starten på simulering av spillet ludo. Tolk koden og forklar hva programmet gjør, hvilke regler i ludo som følges, og hvilke som ikke følges. Velg minst én regel til, og utvid eller endre programmet slik at det følger denne regelen i tillegg.

```
from pylab import *

S1=0
T=0
S2=0
tur=0

while S1<56 and S2<56:
  T= randint(1,7)
  if tur==0:
    print("Jeg fikk",T)
    S1=S1+T
    if T==6:
      tur=0
    else: tur=1
  else:
    print("Du fikk",T)
    S2=S2+T
    if T==6:
      tur=1
    else: tur=0

if S1>=56:
  print("Jeg vant med", S1-S2, "skritt.")
else: print ("Du vant med", S2-S1, "skritt.")
```

b Et enkelt kortspill går ut på at to spillere trekker tre kort hver fra en kortstokk. Du trekker uten tilbakelegging. Spilleren med høyest sum av de tre verdiene, vinner runden. Et dataprogram skal simulere et slikt spill. Planlegg hvordan programmet kan bygges opp ved å bruke et flytskjema. Lag et slikt program i Python. Diskuter hvordan dere kan utvide spillet, og utvid programmet på samme måte for å simulere de nye reglene.

c Velg et annet analogt spill dere kan, enten kortspill eller brettspill, og lag program som simulerer dette spillet. Det kan være lurt å starte med et forenklet spill for så å bygge opp programmet til å likne mer og mer på det kjente spillet.

Figur 3.7: Oppgave hentet fra Maximum 10, analysert etter PRIMM (Tofteberg et al., 2021, s. 240).

3.7 Validitet og reliabilitet

I denne delen skal jeg redegjøre for validitet og relabilitet i forskningen. Dette er en sentral del av studien da betraktningene gjort i henhold til å opprettholde validitet og reliabilitet forteller leseren hvorvidt studien er pålitelig, og i hvilken grad funnene er realistiske (Creswell & Creswell, 2018).

3.7.1 Validitet

Creswell og Creswell (2018) har identifisert åtte strategier for å opprettholde validitet i kvalitative studier, hvorav fire er spesielt relevante for en lærebokanalyse. For å sikre validitet i studien har jeg først og fremst lagt vekt på en detaljert beskrivelse av funnene og vært transparent gjennom hele prosessen ved å inkludere konkrete eksempler fra datamaterialet for å støtte opp under mine funn. Videre har jeg presentert eksempler fra analysen i kapittel 3.5 for å gi leseren innsikt i analyseprosessen. For det tredje har jeg i kapittel 1.3 vært åpen om min bakgrunn og hvordan dette potensielt kan ha påvirket studien. For det fjerde har jeg forsøkt å vise flere perspektiver og inkludert alle funn, selv om de avviker fra hovedtemaet. Ved å presentere ulike synspunkter blir funnene mer realistiske og dermed også mer gyldige. For det femte har jeg underveis i prosessen konsultert med veileder og gjennomgått teori og analyse med ham. I tillegg har jeg deltatt i diskusjoner om metode med andre studenter som også skriver lærebokanalyse og diskutert teori og resultater med studenter som også skriver om programmering. På den måten har jeg brukt profesjonsfellesskapet blant medstudentene. En slik peer-debriefing bidrar til økt validitet ved å gi et sikkerhetsnett utenfor forskerens egen vurdering (Creswell & Creswell, 2018).

3.6.2 Reliabilitet

Kvalitativ reliabilitet vurderer forskningens pålitelighet og konsistens i forskerens tilnærming, med formål om å sikre at resultatene vil bli de samme dersom studien ble gjentatt (Creswell & Creswell, 2018). Creswell og Creswell (2018) identifiserer seks strategier for å opprettholde pålitelighet i forskning, hvorav tre er særlig relevante for en lærebokanalyse utført av en enkeltforsker. For det første krever det at forskeren dokumenterer prosessen detaljert, noe jeg har forsøkt å gjøre i metodekapittelet, der jeg beskriver både datainnsamling og analyse trinn for trinn, se 3.5 analyseprosess. For det andre er det avgjørende å sikre at det ikke forekommer åpenbare feil i transkribering. I mitt tilfelle innebærer dette å sikre at min gjengivelse av materiale er så likt det opprinnelige datamaterialet som mulig (Creswell & Creswell, 2018). For det tredje anbefales det å ha en annen person til å gjennomgå kodene i analyseskjemaet for å sikre klarhet i kategoriene, unngå overlapp og forsikre at oppgavene kan klassifiseres innenfor de definerte rammene (Creswell & Creswell, 2018). I denne studien har min veileder vært involvert i gjennomgangen av analyseskjemaet, og sammen har vi analysert et par oppgaver.

3.8 Etiske betraktninger

I en studie av lærebøker det ikke nødvendig å søke godkjenning hos Sikt, da studien ikke innebærer behandling av personopplysninger eller direkte involverer elever, lærere eller andre personer (De nasjonale forskningsetiske komiteene (NESH), 2021). Til tross for at studien ikke involverer noen personer direkte, vil forskningen basere seg på og ta i bruk eksisterende teori og læreverk. Det er derfor viktig som forsker å vurdere hvordan studien kan påvirke forfatterne bak lærebøkene og teorien (NESH,2021). For å unngå ulemper for forfatterne har jeg som forsker vært bevisst på hvordan jeg omtaler lærebøkene og litteraturen, og forsøkt etter beste evne å analysere lærebøkene objektivt. På denne måten håper jeg at konsekvensene for forfatterne og forlagene er minimert, uten at det står i veien for forskningsresultatene. Det er viktig å være åpen om at forskere tolker materiale ulikt ut ifra egne erfaringer, og det har derfor vært viktig for meg å være åpen om mine tolkninger, med riktige henvisninger til teori og data, slik at det er mulig for leseren og finne tilbake til original tekst (NESH,2021).

Som forsker har jeg et ansvar for å være åpen om mine roller og interesser (NESH, 2021). Samtidig som denne studien er gjort har jeg arbeidet som studentkonsulent i Cappelen Damm Undervisning. Arbeidet går ut på å utvikle oppgaver til deres digitale læreverk skolen.cdu.no. Det å være en del av Cappelen Damm Undervisning kan påvirke min analyse av læreverkene, da jeg kjenner til de interne retningslinjene. Når det er sagt er jeg ikke innblandet i prosessen bak de fysiske lærebøkene, kun den digitale plattformen. Likevel for å unngå en subjektiv analyse, har det vært viktig å utvikle et tydelig analyseverktøy med tydelige kategorier. Jeg har prøvd å være transparent om min metode og mine tolkninger. I tillegg har jeg forsøkt å være ærlig i fremstillingen av prosess og funn gjennom hele studien (Creswell & Creswell, 2018).

Forlagene har opphavsrett på bøkene, og jeg har derfor bedt om tillatelse til å bruke utdrag fra bøkene i oppgaven. Aschehoug og Gyldendal ga meg tillatelse til å bruke utklipp fra bøkene, og jeg har dermed hentet oppgaver rett fra lærebøkene. Cappelen Damm kunne ikke tillate, bruk av illustrasjoner fra bøkene, da disse illustrasjonene er kjøpt til lærebøkene eksklusivt, og jeg har derfor ikke rettigheter til å gjenbruke illustrasjonene i oppgaven. Oppgavene fra Matematikk 10 er derfor redigert slik at

illustrasjonene er fjernet og erstattet med en bildebeskrivelse. I analysen har jeg ikke tatt i bruk noen oppgaver der illustrasjonen er essensiell for oppgaven, dersom dette hadde vært tilfelle ville jeg gjenskapt illustrasjonen etter beste evne.

I innledningen av oppgaven inkluderte jeg et sitat fra en konferanse (Gjøvik, 2024). Av respekt for den relevante personen og for å sikre korrekt gjengivelse, ba jeg om tillatelse til å bruke deres innlegg. Jeg ga også vedkommende muligheten til å lese gjennom sitatet for å sjekke at han var titulert og sitert på riktig måte. Dette ble gjort med tanke på å sikre korrekt og respektfull bruk av kilder i oppgaven.

4 Analysen

I dette kapittelet presenteres analysen av grunnbøkene Maximum 10, Matemagisk 10 og Matematikk 10. Analysen ble gjennomført ved hjelp av et analyseskjema basert på rammeverket til Charalambous et al. (2010), som jeg har tilpasset til forskningsspørsmålet. Kapittelet er strukturert slik at den horisontale analysen av hver enkelt lærebok presenteres først. Deretter funnene fra den vertikale analysen, som i motsetning til den horisontale analysen vil bli inndelt i delanalyser og ikke etter lærebøker.

4.1 Horisontal analyse

I den horisontale analysen har jeg analysert lærebøkenes informasjon og struktur med et spesielt fokus på oppgaver knyttet til programmering. Selve strukturen i lærebøkene ble nøye gjennomgått med tanke på distribusjonen av programmeringsoppgaver, og analysen av lærebøkenes struktur omfatter en gjennomgang av programmeringsoppgavene og en oversikt over hvilke kapitler disse er plassert i. Den horisontale analysen presenteres for hver enkelt lærebok, med en avsluttende oppsummering som identifiserer felles trekk og forskjeller mellom lærebøkene.

4.1.1 Maximum 10

Maximum 10 grunnbok er læreverket til Gyldendal, og er skrevet av Tofteberg, Bråthe, Tangen og Stedøy (2021). En kort oversikt over den horisontale analysen er presentert i tabell 4.1. Grunnboken er en del av en kolleksjon med tilhørende regelsamling, lærerrom og det digitale læreverket skolestudio.no (Gyldendal, u.å). Gyldendal presenterer Maximum 10 som en grunnbok bestående av «[…] varierte aktiviteter, med utforskende, rike og åpne oppgaver. Slik kan elevene gå i dybden og se sammenhenger mellom fagområder og temaer. Elevene møter en variert matematikkopplæring som legger vekt på relasjonell forståelse og lar dem arbeide med faget innenfor læringsfellesskapet.» (Gyldendal, u.å.). Maximum 10 er bygd opp slik at hvert kapittel starter med teori og et eksempel før elevene gjør oppgaver. Blant oppgavene er det noen aktiviteter og oppdrag. Aktivitetene er ofte samarbeidsoppgaver der gruppene ofte trenger litt utstyr for å utforske matematikk på en kreativ måte. Oppdragene likner på aktivitetene, men krever mindre utstyr og er ofte mindre åpne enn aktivitetene. Oppgavene i Maximum 10 utgjør store deler av læreboken. Oppgavene er differensiert i ulike vanskelighetsgrader, og varierer mellom individuelle og samarbeidsoppgaver. Grunnboken Maximum 10 består av fire kapitler; Likninger og algebra, Funksjoner, Økonomi og Se flere sammenhenger. Fordelingen av sider per kapittel er omtrentlig lik, og kapitelene er bygd opp unisont. Det siste kapittelet skiller seg fra de andre ved at det tar for seg flere matematiske temaer. Blant annet ved å sette søkelys på kjerneelementene i matematikk som utforskning og problemløsning, og modellering, samt tverrfaglige temaer og en mer praktisk og virkelighetsorientert tilnærming til matematikk (Tofteberg et al.2021).

HORISONTAL ANALYSE AV MAXIMUM 10

Lærebokens informasjon		**Lærebokens struktur**			
Forfattere	Tofteberg, G.N., Tangen, J., Bråthe, L.T., & Stedøy, I.	***Antall sider***	304		
Forlag	Gyldendal	***Antall kapitler***	4	**Antall sider**	**Antall prog.opg (deloppgaver)**
Utgitt	2021	***Oversikt over kapitler***	1. Likninger og algebra	54	0
Utgave og opplag	2.utgave, 1.opplag		2. Funksjoner	70	5 (13)
Tilleggsmateriale	Regelsamling		3. Økonomi	54	0
	Digitalt lærerrom (lærerveiledning)		4. Se flere sammenhenger	52	4 (8)
				Totalt	9 (21) prog.opg.
	Skolestudio.no	***Annet***	*Ordbibliotek*		

Tabell 4.1: En horisontalanalyse av læreverket Maximum 10 (Tofteberg et al.,2021)

Programmering blir brukt i totalt 21 deloppgaver, og er plassert i kapitelene Funksjoner og Se flere sammenhenger. Oppgavene er ikke differensiert, og er en del av fellesopplæringen. De ni oppgavene nevner programmering eksplisitt, mens flere av oppgavene ber elevene bruke digitale hjelpemidler uten å spesifisere hvilke hjelpemidler de henviser til. Slike oppgaver er ikke inkludert i datamaterialet, selv om elevene kan benytte programmering til å løse noen av oppgavene. Av andre digitale hjelpemidler benytter Maximum dynamiske graftegnere, dynamiske geometriprogram og regneark.

4.1.2 Matemagisk 10

Matemagisk 10 grunnbok er læreverket til Aschehoug Undervisning, og er skrevet av Wallace og Kongsnes (2021). En kort oversikt over den horisontale analysen er presentert i tabell 4.2. Grunnboken er en del av en kolleksjon med tilhørende parallellbok, elevhåndbok, digital lærerveiledning, samt den digitale plattformen Aunivers (Kongsnes & Wallace, 2021). Aschehoug presenterer Matemagisk 10 som en grunnbok som «inneholder en stor variasjon av oppgavetyper, spill og aktiviteter som engasjerer og gjør matematikkundervisningen meningsfull for lærere og elever. Differensieringsmodellen lar elevene lære matematikk på sitt nivå, men likevel i takt med hverandre. Med Snakke matte får elevene gjøre aktiviteter som oppfordrer til dem til å argumentere med egne ord og utvikle kritisk tenkning. Elevene vil utvikle algoritmisk tenkning og lære programmering på fagets premisser» (Kongsnes & Wallace, 2021). Fokuset på algoritmisk tenkning og programmering kommer frem i antall oppgaver med programmering, der Matemagisk 10 har tre ganger så mange oppgaver med programmering sammenlignet med Maximum 10 (Tofteberg et al.,2021; Wallace &

Kongsnes, 2021). Grunnboken er strukturert slik at hvert kapittel begynner med en fellesløype bestående av teori og eksempeloppgaver, som er designet for å arbeides med i plenum. Differensieringsmodellen består av fire løyper, der «følg stien» er den mest grunnleggende, og elevene trener i disse oppgavene på én ting av gangen. «Terrengløypa» tar for seg flere sammensatte utfordringer, «Topptur» er enda mer utfordrende og elever anbefales å gå videre til «Topptur» dersom «Terrengløypa» var veldig overkommelig, da «Topptur» tar for seg pensum noe over 10.trinns nivå. Det siste nivået er «Ekspedisjon» som rommer totalt fire oppgaver, og gir særlig god trening i problemløsning, abstraksjon og generalisering (Wallace & Kongsnes, 2021). I tillegg til de vanlige oppgavene har Matemagisk 10 en seksjon som heter snakke matte. Snakke matte oppgaver er samarbeidsoppgaver der elevene arbeider med å uttrykke matematikk muntlig, gjennom argumentasjon og kritisk tenkning (Wallace & Kongsnes, 2021).

Grunnboken Matemagisk 10 består av åtte kapitler. Nummereringen av kapitlene kommer av at tellingen begynner i Matemagisk 8, så de tre læreverkene Matemagisk 8,9 og 10 utgjør totalt 25 kapitler. Slik som Maximum 10 reflekteres kjerneelementene i Matemagisk 10, der modellering er et eget kapittel, og i Geometritårnet der elevene driver utforskning og problemløsning knyttet til geometri.

HORISONTAL ANALYSE AV MATEMAGISK 10

Lærebokens informasjon		**Lærebokens struktur**			
Forfattere	Kongsnes, A.L., & Wallace, A.K.	***Antall sider***	320		
Forlag	Aschehoug Undervisning	***Antall kapitler***	8	**Antall sider**	**Antall prog.opg (deloppgaver)**
Utgitt	2021	***Oversikt over kapitler***	18. Utforske matematiske sammenhenger	11	11 (23)
Utgave og opplag	1.utgave, 1.opplag		19. Algebrastiger	44	0
Tilleggsmateriale	Pararellbok		20. Likningssett	22	0
	Elevhåndbok		21. Prosentregning	36	4 (8)
	Digital lærerveiledning		22. Personlig økonomi	46	4 (14)
	Aunivers.no		23. Funksjoner	58	5 (13)
			24. Modellering	42	1 (5)
			25.Geometritårnet	32	0
				Totalt	26 (63) prog.opg.

Annet	
Fasit	
Programmering i Python	*4 sider*

Tabell 4.2: En horisontalanalyse av læreverket Maximum 10 (Tofteberg et al.,2021)

Oppgavene med programmering er fordelt over fem kapitler, og som i Maximum 10 er flest programmeringsoppgaver i Utforske matematiske sammenhenger. Matemagisk 10 har programmeringsoppgaver noenlunde jevnt fordelt ut over grunnboken. Av de 58 deloppgavene med programmering er åtte av dem i felles delen, 4 i «snakke matte». Når det kommer til nivå, er de fleste oppgavene med programmering i «terrengløypa» (50%). På det høyeste nivået er det én programmeringsoppgave bestående av fire deloppgaver. Av andre digitale hjelpemidler benytter Matemagisk 10 graftegneren i Geogebra, og regneark i Excel.

4.1.3 Matematikk 10

Matematikk 10 er læreverket til Cappelen Damm, og er skrevet av Hjardar og Pedersen (2021). En generell oversikt over læreverket er presentert i tabell 4.3. Grunnboken er en del av en kolleksjon med tilhørende oppgavebok, alternativ oppgavebok, lærerens bok og det digitale læreverket skolenmin.cdu.no (Cappelen Damm, u.å.). Cappelen Damm presenterer grunnbøkene Matematikk 8-10 ved at de «[..] legger til rette for dybdelæring ved å ta kjerneelementene på alvor, samtidig som elevene øver på grunnkompetansen og er lenger i hvert tema. Gjennom tydelig struktur og progresjon, bruk av ulike læringsstrategier, utforsking og problemløsing får elevene utvikle sin matematiske forståelse og sine ferdigheter i faget. Med Matematikk 8-10 fra Cappelen Damm får elevene anledning til å utforske, resonnere og argumentere for egne løsninger, samt oppøve evnen til kritisk tenkning» (Cappelen Damm, u.å.). I likhet med de to andre bøkene anser Matematikk 10 kjerneelementene som sentrale i lærebøkene, dette kommer tydelig frem i Matematikk 10 kapittel 4 «utforskende arbeid» der elevene arbeider med problemer på tvers av matematiske temaer.

Grunnboken er strukturert slik at den består av fire kapitler helt uten programmering. Programmeringen er plassert i et digitalt vedlegg kalt kapittel «0. Programmering». Cappelen Damm skriver på sine nettsider at programmering er et felt som er i kontinuerlig utvikling, og at ved å ha programmering separert og digitalt, vil de kunne oppdatere innholdet oftere enn ved en fysisk bok. (Cappelen Damm, u.å.). Gitt progresjonen av programmering beskrevet i LK20, vil elevene får større kompetanse i programmering på barnetrinnet, og det kan derfor være hensiktsmessig å revidere opplegget basert på elevenes forkunnskaper i programmering. I grunnboken står det ikke eksplisitt at det finnes et kapittel om programmering digitalt, dette opplyses om på nettsidene. Heftet er også tilgjengelig i den digitale lærerressursen skolenmin.cdu.no. Den fysiske grunnboken har ingen programmering i seg, og er derfor ikke inkludert i den vertikale analysen. Grunnboken tar for seg andre digitale verktøy som regneark og Geogebra, og helt bakerst i læreboken er det en brukerveiledning til de to digitale hjelpemidlene.

HORISONTAL ANALYSE AV MATEMATIKK 10

Lærebokens informasjon		**Lærebokens struktur**			
Forfattere	Hjardar, E. & Pedersen, J.	***Antall sider***	339 + 64		
Forlag	Cappelen Damm	***Antall kapitler***	4+1	**Antall sider**	**Antall prog.opg**
Utgitt	2021	***Oversikt over kapitler***	0. Programmering *Hefte lastet ned 02.02.24*	15	5 (33 deloppgaver)
Utgave og opplag	1.utgave, 1.opplag		1.Algebra	82	0
Tilleggsmateriale	Oppgavebok Alternativ oppgavebok Lærerens bok Skolenmin.cdu.no		2.Funksjoner og grafer	98	0
			3.Økonomi	80	0
			4.Utforskende arbeid	73	0
				Totalt	5 (33) prog.opg.
		Annet	*Manual for digitale verktøy* *Fasit* *Stikkord*		

Tabell 4.3: En horisontalanalyse av læreverket Matematikk 10 (Hjardar & Pedersen, 2021)

Programmeringsheftet dekker matematikk for hele ungdomsskolen, altså 8-10.trinn. Totalt er heftet på 64 sider. Programmeringsheftet følger ikke samme differensiering som den fysiske grunnboken, Matematikk 10. Programmeringsheftet består av fire deler. Den første delen består av en generell innføring programmering, altså oppgaver om programmeringsteknikk. De resterende delene er fag- og trinnspesifikke, der del fire tilhører programmering i matematikk for 10.trinn. Det er oppgavene for 10.trinn som danner utgangspunktet for analysen. Delen av programmeringsheftet for 10.trinn har et omfang på fem oppgaver. Oppgavene til er fordelt etter temaene; funksjonsuttrykk, grafer, kvadrattall, Pytagoras og sparing. I tabell 4.4 har jeg koblet temaene i programmeringsheftet til kapitlene i den fysiske grunnboken. Oversikten i tabell 4.4 viser at Matematikk 10 har flest oppgaver knyttet til funksjoner og utforskende arbeid, og ingen knyttet til algebra.

Tema	Antall opg. m/programmering	Navn på oppgavene
1.Algebra	0	-
2.Funksjoner og grafer	2(10)	Funksjonsutrykk, Grafer
3.Økonomi	1(5)	Sparing
4.Utforskende arbeid	2(10)	Pytagoras, Kvadrattall

Tabell 4.4: Oversikt over temaene i programmeringshefte sett i lys av kapittelinndelingen fra grunnboken (Hardar, 2021; Hjardar & Pedersen,2021)

4.1.4 Oppsummering av den horisontale analysen.

Resultatene fra den horisontale analysen indikerer at lærebøkene deler visse likheter når det gjelder plasseringen av programmeringsoppgaver. De fleste programmeringsoppgavene er gitt i sammenheng med temaene funksjoner og utforskende arbeid/se flere sammenhenger, mens ingen av bøkene inkluderer programmering i sammenheng med algebra. Imidlertid er det betydelige forskjeller i antall programmeringsoppgaver i de tre grunnbøkene. Selv om lærebøkene er omtrent like lange i form av antall sider, har Matemagisk 10 over dobbelt så mange programmeringsoppgaver som de to andre lærebøkene. Den vertikale analysen vil gi ytterligere innsikt i hvordan programmering blir anvendt i lærebøkene.

4.2 Vertikal analyse

I den vertikale analysen har jeg undersøkt det matematiske innholdet i lærebøkene, praksiser og de forventningene som stilles til elevene. Hovedfokuset når det gjelder det matematiske innholdet har vært på forholdet mellom matematikk og programmering i oppgavene, og dette er analysert i samsvar med rammeverket presentert av Kilhamn et al. (2021). Når det gjelder forventningene til elevene, er oppgavene analysert i henhold til PRIMM-modellen, og i forhold til matematiske praksiser har jeg undersøkt hvilke programmeringsspråk som benyttes i oppgavene.

Den vertikale analysen er strukturert slik at funnene fra de tre delanalysene presenteres først. Etter dette presenteres andre relevante funn som er av interesse for forskningsspørsmålet. Delanalyse 1 og 2 er derfor deduktive, mens de andre funnene er induktive. De induktive funnene er ikke kodet basert på et teoretisk rammeverk, men kategoriene er utledet fra datamaterialet.

4.2.1 Delanalyse 1: Relasjon mellom matematikk og programmering

Delanalyse 1 har undersøkt relasjon mellom matematikk og programmering i oppgavene. Samtlige oppgaver som omhandler eller bruker programmering fra de tre lærebøkene er analysert og kategorisert etter rammeverket vist i metodekapittel 3.5.1. En fremstilling etter opptelling av kategoriseringen er vist i tabell 4.5.
I delanalyse 1 har jeg lyst til å trekke frem følgende funn:

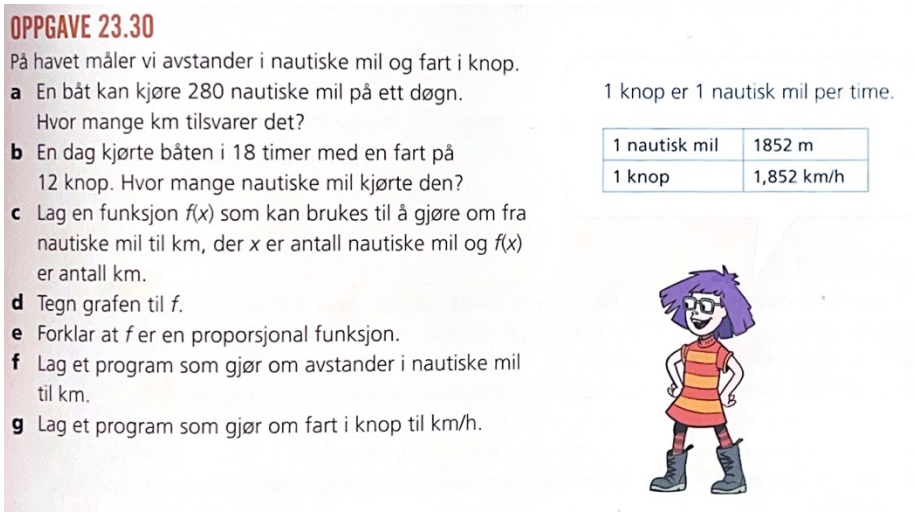
1. Flest oppgaver bruker programmering som verktøy for å effektivisere matematiske handlinger.
2. Programmering uten kobling til matematikk eksisterer nesten ikke i lærebøkene.
3. Det er få oppgaver som bruker programmering som et verktøy for å utforske matematikk.

Kategori	Maximum 10	Matemagisk 10	Matematikk 10	Totalt
Programmering **uten kobling** til matematikk.	-	-	1	3
Matematikk som **en kontekst** for programmering	5	6	5	13
Programmering som **et verktøy** for å **effektivisere** matematiske handlinger.	**14**	**54**	**25**	**93**
Programmering som **et verktøy** for å **utforske** matematikk	1	7	1	9

Tabell 4.5: Oversikt over kategoriseringen av relasjon mellom matematikk og programmering i de tre læreverkene (Kilhamn et al., 2021; Tofteberg et al., 2021; Kongsnes & Wallace, 2021; Hjardar & Pedersen, 2021).

Funn 1: Flest oppgaver bruker programmering som verktøy for å effektivisere matematiske handlinger

I lærebøkene både sett samlet og hver for seg brukes programmering mest som et verktøy for å effektivisere matematiske handlinger. Over 70% av oppgavene med programmering ble kategorisert som et verktøy for å effektivisere matematiske handlinger (se tabell 4.5). Oppgavene som faller innenfor denne kategorien ber elevene ofte bruke, lage eller endre et program for å besvare et matematisk spørsmål. Eksempelvis i figur 4.1, der programmering brukes som et verktøy, mer spesifikt som en konverterings-kalkulator mellom nautiske mil og km, og knop til km/t.



OPPGAVE 23.30

På havet måler vi avstander i nautiske mil og fart i knop.

a En båt kan kjøre 280 nautiske mil på ett døgn. Hvor mange km tilsvarer det?

b En dag kjørte båten i 18 timer med en fart på 12 knop. Hvor mange nautiske mil kjørte den?

c Lag en funksjon $f(x)$ som kan brukes til å gjøre om fra nautiske mil til km, der x er antall nautiske mil og $f(x)$ er antall km.

d Tegn grafen til f.

e Forklar at f er en proporsjonal funksjon.

f Lag et program som gjør om avstander i nautiske mil til km.

g Lag et program som gjør om fart i knop til km/h.

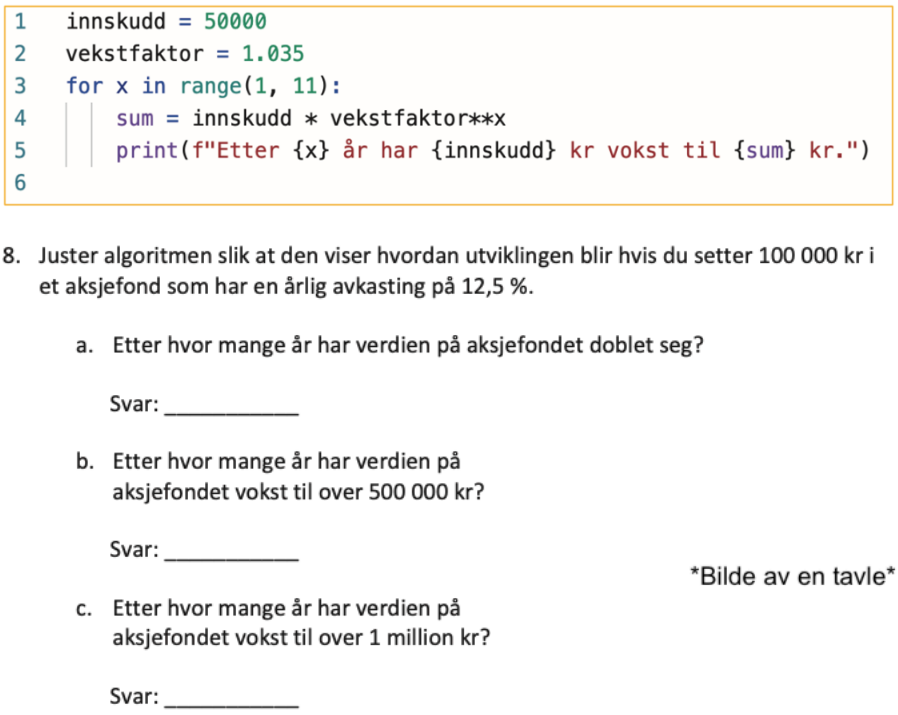
1 knop er 1 nautisk mil per time.

1 nautisk mil	1852 m
1 knop	1,852 km/h

Figur 4.1: Oppgave hentet fra Matemagisk 10 om funksjoner (Kongsnes & Wallace, 2021, s.189)

Ved å bruke programmering som verktøy benytter elevene seg av velkjent matematikk og programmering, uten å oppnå en dypere forståelse i noen av dem. De oppdager ingen nye perspektiver eller sammenhenger som de ikke allerede kjenner til. Denne tilnærmingen tillater ikke utforskning av forholdet mellom knop og nautiske mil, og km og km/t. Oppgaven oppfyller derfor ikke kravene til et verktøy for å utforske matematikk gjennom programmering. Til gjengjeld gir oppgaven elevene mulighet til å praktisere bruken av matematikk i en realistisk sammenheng.

I figur 4.2, blir elevene presentert med kode som de skal tilpasse for å lage et verktøy som kan beregne avkastningen på aksjefond. I denne oppgaven blir det forventet at elevene bruker det utviklede verktøyet til å besvare videre spørsmål, i motsetning til figur 4.1 der elevene lager et program uten å anvende det videre.



```
1  innskudd = 50000
2  vekstfaktor = 1.035
3  for x in range(1, 11):
4      sum = innskudd * vekstfaktor**x
5      print(f"Etter {x} år har {innskudd} kr vokst til {sum} kr.")
6
```

8. Juster algoritmen slik at den viser hvordan utviklingen blir hvis du setter 100 000 kr i et aksjefond som har en årlig avkasting på 12,5 %.

   a. Etter hvor mange år har verdien på aksjefondet doblet seg?

      Svar: _____

   b. Etter hvor mange år har verdien på aksjefondet vokst til over 500 000 kr?

      Svar: _____

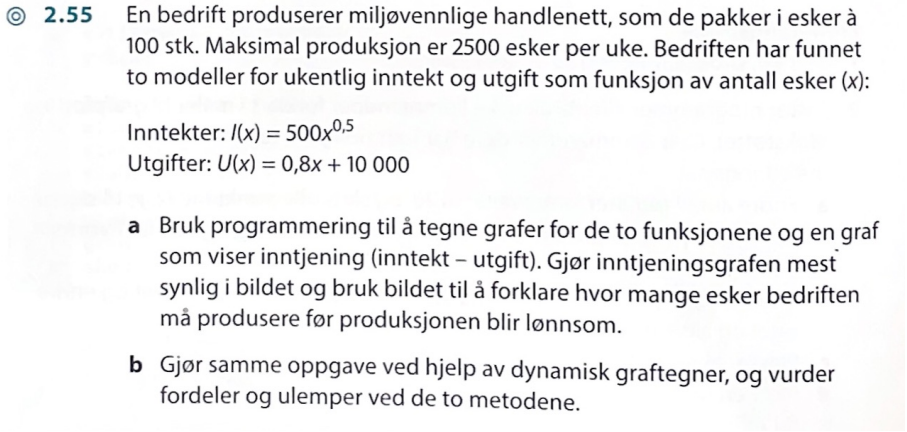
   c. Etter hvor mange år har verdien på aksjefondet vokst til over 1 million kr?

      Svar: _____

Bilde av en tavle

Figur 4.2: Oppgave hentet fra Matematikk 10 om Sparing (Hardar, 2021, s.62)

Oppgaven vist i figur 4.3 ber elevene lage et program som tegner grafen til inntekt og utgifts funksjonen til en bedrift. Programmet blir brukt som et verktøy for å utføre en matematisk handling, nemlig graftegning. Elevene skal videre bruke grafen til å finne optimal produksjon.



◎ **2.55** En bedrift produserer miljøvennlige handlenett, som de pakker i esker à 100 stk. Maksimal produksjon er 2500 esker per uke. Bedriften har funnet to modeller for ukentlig inntekt og utgift som funksjon av antall esker (x):

Inntekter: $I(x) = 500x^{0,5}$
Utgifter: $U(x) = 0{,}8x + 10\,000$

a Bruk programmering til å tegne grafer for de to funksjonene og en graf som viser inntjening (inntekt – utgift). Gjør inntjeningsgrafen mest synlig i bildet og bruk bildet til å forklare hvor mange esker bedriften må produsere før produksjonen blir lønnsom.

b Gjør samme oppgave ved hjelp av dynamisk graftegner, og vurder fordeler og ulemper ved de to metodene.

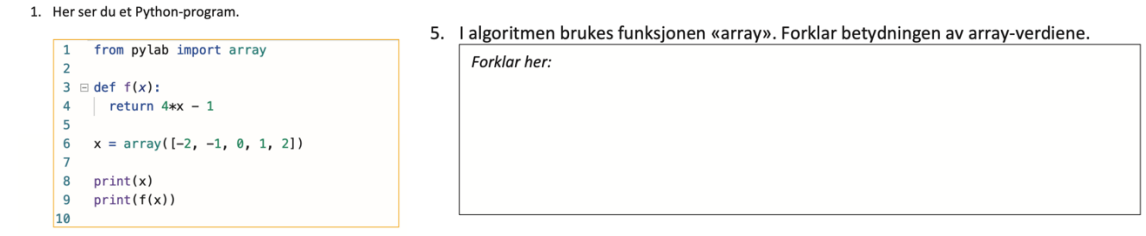
Figur 4.3: Oppgave hentet fra Maximum 10 om økonomi (Tofteberg et al., 2021, s.120)

For å oppsummere funnet, ble programmering i majoriteten av oppgavene brukt som et verktøy for å effektivisere en matematisk handling. Programmering ble brukt som et verktøy for å gjøre matematiske handlinger. I oppgave 4.1 blir programmering brukt til å effektivisere matematiske beregninger, i oppgave 4.2 som et modelleringsverktøy, og i 4.3 som en graftegning. Programmering blir dermed brukt som et verktøy til å utføre en rekke matematiske handlinger.

Funn 2: Programmering uten kobling til matematikk eksisterer nesten ikke i lærebøkene.

I datamaterialet identifiserte jeg en deloppgave som kunne kategoriseres som ren programmering uten tilknytning til matematikk, se figur 3.2. Selv om denne deloppgaven isolert sett omhandler programmeringstekniske aspekter, inngår den likevel i en større oppgave som omhandler sparing. Det avgjørende for om oppgaven kan knyttes til matematikk eller ikke, avhenger derfor av hvordan elevene velger å svare på den. Dersom elevene fokuserer utelukkende på de tekniske aspektene ved oppgaven, for eksempel ved å diskutere løkker isolert fra konteksten, vil oppgaven forbli uten tilknytning til matematikk. Imidlertid, hvis elevene inkluderer matematiske prinsipper, som for eksempel funksjoner, i løsningen av oppgaven, vil den kunne klassifiseres som programmering med matematikk som kontekst. En mulig løsning på oppgaven kan være å vurdere at range (1,11) vil inkludere verdiene fra 1 til 10, da det siste tallet er eksklusivt og derfor ikke inkorporert i løkken. Derfor vil range (1,10) gjenta løkken 9 ganger, mens range (1,11) vil resultere i 10 repetisjoner. En slik løsning fokuserer utelukkende på det programmeringstekniske aspektet uten å inkludere matematiske prinsipper, derfor vil oppgaven bli betraktet som programmering uten tilknytning til matematikk.

En liknende oppgave fra samme lærebok, Matematikk 10, er presentert i figur 4.4. Denne oppgaven deler likhetstrekk med oppgaven i figur 3.2 ved at den fokuserer på et programmeringsteknisk element, array. Deloppgaven i figur 4.4 utgjør en del av en større oppgave om funksjonsuttrykk, tilsvarende som oppgave i 3.2 gjør med sparing. Det som skiller oppgavene er at oppgaven i figur 4.4 krever en forklaring på betydningen av array-verdiene, og dermed ber eleven om å se array i sammenheng med funksjonen. Av den grunn er oppgaven i figur 4.4 klassifisert som programmering med matematikk som kontekst.


1. Her ser du et Python-program.

```
from pylab import array

def f(x):
    return 4*x - 1

x = array([-2, -1, 0, 1, 2])

print(x)
print(f(x))
```

5. I algoritmen brukes funksjonen «array». Forklar betydningen av array-verdiene.

Forklar her:

Figur 4.4: Oppgave hentet fra Matematikk 10 om funksjoner (Hardar, 2021, s.50)

For å oppsummere funnet, var det kun én oppgave som ble kategorisert som programmering uten kobling til matematikk. Funnet kan tyde på at lærebøkene i matematikk har implementert programmering på en fagspesifikk måte. Enkelte av oppgavene har fokus på programmeringsteknikk, men majoriteten kobler programmeringen til matematikk ved å bruke matematikk som kontekst.

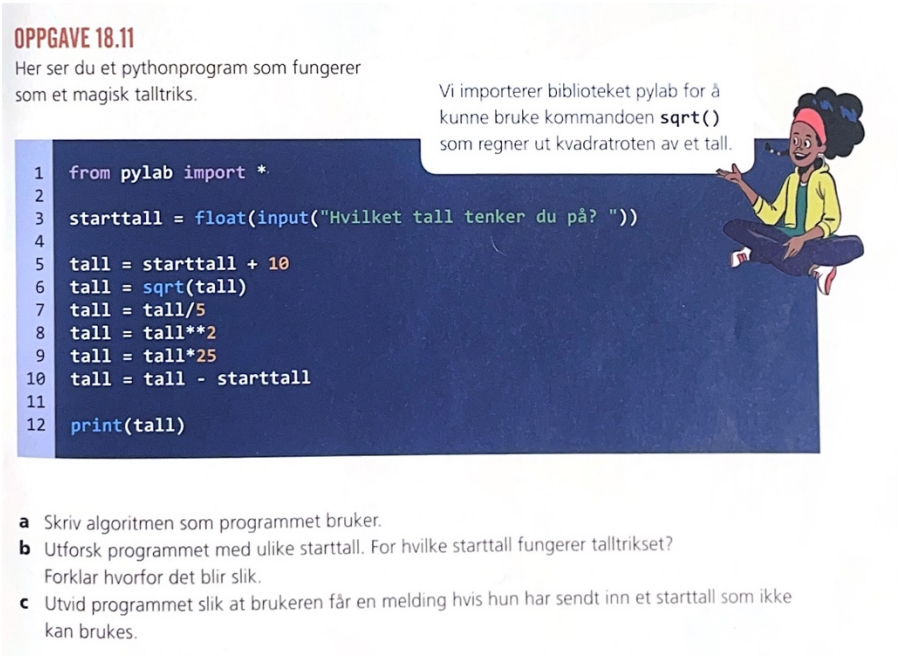
Funn 3: Det er få oppgaver som bruker programmering som et verktøy for å utforske matematikk

Kategorien programmering som et verktøy for å utforske matematikk er smalere enn de andre kategoriene, og det kreves mer av oppgavene for å kvalifisere til denne kategorien. I totalt 9 av 117 deloppgaver blir programmering brukt som et verktøy for å utforske matematikk (se tabell 4.5). Alle oppgavene i denne kategoriseringen er presentert i sammenheng med funksjoner og utforske/se matematiske sammenhenger. Et eksempel på en slik oppgave er vist i figur 4.5. I denne oppgaven skal elevene lage et program som tegner grafer til andregradsuttrykk basert på brukerens valg av koeffisienter. De skal lage et generalisert program der brukeren kan tegne spesifikke grafer. Ved at oppgaven baserer seg på det generelle minner oppgaven om utforskning da utforskende matematikk ofte kjennetegnes ved å gå fra noe spesifikt til en generalisering. Oppgavene legger opp til at elevene kan utforske hvordan endringer i koeffisientene påvirker egenskapene til funksjonen, for eksempel posisjonen til ekstremalpunktet og stigningen til funksjonen. Oppgaven ber ikke elevene eksplisitt om å teste programmet, men en kan anta at i forsøket på å lage et program så vil elevene teste programmet for å feilsøke det, og dermed få innblikk i hvordan koeffisientene påvirker grafen. For at oppgaven skulle vært enda mer utforskende, kunne det vært et oppfølgingsspørsmål.

2.53 Lag et dataprogram som tegner grafen til en andregradsfunksjon når brukeren oppgir verdiene til a, b og c i det generelle uttrykket $f(x) = ax^2 + bx + c$, så lenge grafen vises i tallområdet [–10, 10] for både x og y.

4.5: Oppgave hentet fra Maximum 10 om funksjoner (Tofteberg et al., 2021, s.120)

I Matemagisk 10 ble begrepet utforskning eksplisitt nevnt tre ganger i sammenheng med programmering. Begrepet ble ikke brukt i programmeringssammenheng i Matematikk 10 eller Maximum 10. Et eksempel er oppgave 18.11 i Matemagisk 10, er vist i figur 4.6. Oppgaven oppfordrer elevene til å utforske et talltriks som er gitt i form av et program. Oppgaven går ut på at elevene skal utforske uttrykket $(\frac{\sqrt{x+10}}{5})^2 \cdot 25 - x$, og oppdage at det kan forenkles til $(\frac{\sqrt{x+10}}{5})^2 \cdot 25 - x = 10$. Videre skal elevene undersøke hvilke starttall talltrikset fungerer for, de kan bruke ulike tilnærminger, enten ved å teste programmet med en rekke starttall eller ved å undersøke uttrykket. Ved å analysere uttrykket vil elevene oppdage at kvadratroten er avgjørende, siden det ikke er mulig å ta roten av et negativt tall. Derfor vil talltrikset kun fungere for $x \geq -10$, det vil si verdier som er større eller lik -10. Dersom elevene velger et mindre startall enn -10, for eksempel -11 vil man i linje 6 få $\sqrt{-1}$, som vil resultere i en feilmelding, ettersom ingen reelle tall multiplisert med seg selv gir et negativt produkt. Ved å inkludere programmering i oppgaven vil elevene motta en feilmelding i programmet, og dermed internalisere prinsippet om betingelser ved kvadratrøtter og overføre det til sine egne beregninger. Dette kan resultere i diskusjoner eller det kan fungere som en påminnelse om viktigheten av å huske at kvadratroten bare kan tas av positive tall. Denne erfaringen gjennom programmering kan bidra til å forbedre elevenes forståelse av matematiske begreper og øke deres bevissthet omkring betingelsene for matematiske operasjoner.



OPPGAVE 18.11

Her ser du et pythonprogram som fungerer som et magisk talltriks.

Vi importerer biblioteket pylab for å kunne bruke kommandoen `sqrt()` som regner ut kvadratroten av et tall.

```
1  from pylab import *
2
3  starttall = float(input("Hvilket tall tenker du på? "))
4
5  tall = starttall + 10
6  tall = sqrt(tall)
7  tall = tall/5
8  tall = tall**2
9  tall = tall*25
10 tall = tall - starttall
11
12 print(tall)
```

a Skriv algoritmen som programmet bruker.
b Utforsk programmet med ulike starttall. For hvilke starttall fungerer talltrikset? Forklar hvorfor det blir slik.
c Utvid programmet slik at brukeren får en melding hvis hun har sendt inn et starttall som ikke kan brukes.

Figur 4.6: Oppgave hentet fra Matemagisk 10 fra kapittelet Utforske matematiske sammenhenger (Kongsnes & Wallace, 2021, s.15)

Felles for oppgavene der utforskning er nevnt, er at oppgavene omhandler et generelt uttrykk. Figur 4.7 viser et eksempel til fra Matemagisk 10 med kvadratsetningene. I oppgave 18.3a skal elevene lage et program som tester hvilke x-verdier utrykket $x^2 + 25 = (x+5)^2$ stemmer for. Deretter tester elevene programmet for ulike x-verdier. Elevene vil i 18.3b se at uttrykket stemmer for x=0, og i 18.3c se at det ikke stemmer for x=3. Videre vil noen elever kunne oppdage at det første uttrykket er den første kvadratsetningen $(x+5)^2 = x^2 + 10x + 5^2$ og det andre uttrykket er konjugatsetningen $x^2 + 25 = (x+5)(x-5)$. I 18.3d skal elevene endre uttrykket slik at programmet stemmer for alle verdier av x, og de vil da ende opp med å teste uttrykket $(x+5)^2 = x^2 + 10x + 5^2$, som stemmer for alle verdier av x.

OPPGAVE 18.3

a Lag et program som kan brukes til å utforske om $(x+5)^2$ har samme verdi som $x^2 + 25$ for ulike heltallige verdier av x.
b Kjør programmet med $x = 0$. Stemmer sammenhengen for denne verdien av x?
c Kjør programmet med $x = 3$. Stemmer sammenhengen for denne verdien av x?
d Oppdater programmet ved å bytte ut uttrykket $x^2 + 25$ med et annet uttrykk. Uttrykket skal ha samme verdi som $(x+5)^2$ for alle verdier av x. Kjør programmet for ulike verdier av x og sjekk at de to uttrykkene har den samme verdien for disse x-verdiene.

Figur 4.7: Oppgave hentet fra Matemagisk 10 fra kapittelet Utforske matematiske sammenhenger (Kongsnes & Wallace, 2021, s.13)

Oppgaven presentert i figur 4.7 kan bidra til å fremme en dypere forståelse av kvadratsetningene blant elevene. Selv om dette ikke fremgår klart i selve figuren, er det verdt å merke seg at elevene blir oppfordret til å bruke modeller, for eksempel firkanter, som et hjelpemiddel. Dette anbefales ikke bare for å bevise resultatet, men også for å hjelpe elevene med å komme frem til svaret. Denne tilnærmingen ved å lage program, teste hvilke verdier uttrykket stemmer for, for og så endre slik at det stemmer vil for noen elever være kun anvendelse av kjent matematikk, men oppgaven er bygd slik at noen elever vil kunne oppdage nye sammenhenger. Uansett vil læringsutbytte være

matematisk, derfor kategoriseres denne oppgaven som programmering som verktøy for å utforske matematikk.

For å oppsummere funnet, var det kun ni oppgaver der programmering ble brukt som et verktøy for å utforske matematikk, hvor av syv av de var i Matemagisk 10. Fem av oppgavene brukte begrepet utforske eksplisitt. Oppgavene med utforskning i lærebøkene hadde som fellestrekk at ber elevene undersøke et generelt uttrykk, enten ved å starte i det spesifikke med mål om å lage et generelt utrykk eller et bevis på hvorfor uttrykket stemmer/ikke stemmer i det generelle, eller ved å starte i det spesifikke for og så generalisere.

4.2.2 Delanalyse 2: PRIMM

Delanalyse 2 har undersøkt hva som er forventet av elevene med tanke på hva oppgavene ber eleven gjøre. Jeg har undersøkt alle oppgavene som omhandler eller bruker programmering og kategorisert dem etter rammeverket vist i metodekapittel 3.5.2. Kategoriseringen er blitt gjort lærebok for lærebok, og i tabell 4.6 er en fremstilling av resultatet av delanalyse 2.
I delanalyse 2 har jeg lyst til å trekke frem to funn:

1. Flere av oppgavene fulgte PRIMM-modellen
2. En del av oppgavene ber elevene lage et program

Kategori	Maximum 10	Matemagisk 10	Matematikk 10	Totalt
Forutse	2	3	0	5
Kjøre	2	7	6	15
Undersøke	5	16	**16**	**38**
Endre	5	**21**	4	30
Lage	**9**	18	4	30

Tabell 4.6: Antall deloppgaver kategorisert etter PRIMM, der det høyeste tallet er fetet ut (Tofteberg et al., 2021; Kongsnes & Wallace, 2021; Hjardar & Pedersen, 2021).

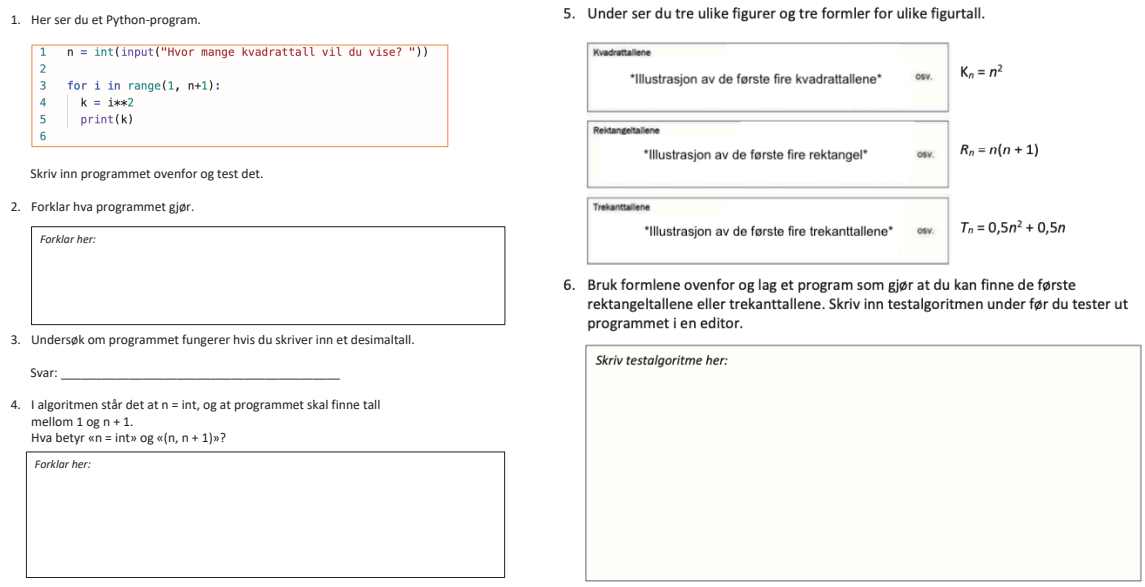
Funn 1: Alle oppgavene i Matematikk 10 fulgte PRIMM-modellen.

I lærebøkene ble flere av oppgavene strukturert i samsvar med PRIMM-modellen. I Matematikk 10 var alle oppgavene utformet etter PRIMM-modellen, og startet med en analyse av en ferdigskrevet kode. I Matemagisk begynte 3 av oppgavene (33%) med en ferdigskrevet kode, mens tilsvarende gjaldt to oppgaver i Maximum 10 (22%).

Et eksempel på en oppgave strukturert etter PRIMM-modellen er oppgave 4.30 fra Maximum 10, vist i figur 3.7. I 4.30a skal elevene først undersøke koden (forutse) og analysere den grundig, så vurdere hvilke endringer som kan gjøres for å forbedre spillet (undersøke). Deretter skal de implementere disse endringene (endre). I den første deloppgaven blir elevene introdusert for et program som simulerer ludo, og dette gir dem et innblikk i de ulike elementene som de senere kan anvende når de skal utvikle sitt eget program i oppgavene 4.30b og c. Det er verdt å merke seg at elevene ikke blir bedt om å utføre programmet (kjøre) i denne delen av oppgaven.

Strukturen på oppgavene til Cappelen Damm er alle lik oppgaven om kvadrattall, vist i figur 4.8. Oppgavene starter med en kode, som elevene skal kjøre, deretter skal de undersøke noe, før de til slutt skal lage et eget program. I oppgaven om kvadrattall får elevene et program som skriver ut en rekke med kvadrattall, fra det første og så langt brukeren vil. Elevene skal videre forklare koden, før de skal undersøke om brukeren kan skrive desimaltall, og hva range (1, n+1) innebærer. I deloppgave 5 får elevene

informasjon om kvadrat-, rektangel- og trekanttall, før de skal lage et program tilsvarende det fra oppgave 1 med rektangel- eller trekanttallene. Strukturen likner på PRIMM, bortsett fra at oppgaven ikke ber elevene endre koden, og elevene blir ikke bedt om å forutse. I noen oppgaver i Matematikk 10 blir elevene bedt om å endre koden, dette skjer da før elevene selv lager kode. Selve strukturen er inspirert av PRIMM i den forstand at elevene undersøker et forhåndsskrevet program før de selv lager sitt eget. En slik tilnærming gjør at elevene har en kode med elementer de kan bruke, noe som gjør at inngangsterskelen til å kode minimeres.



1. Her ser du et Python-program.

```
n = int(input("Hvor mange kvadrattall vil du vise? "))

for i in range(1, n+1):
    k = i**2
    print(k)

```

Skriv inn programmet ovenfor og test det.

2. Forklar hva programmet gjør.

Forklar her:

3. Undersøk om programmet fungerer hvis du skriver inn et desimaltall.

Svar: _____

4. I algoritmen står det at n = int, og at programmet skal finne tall mellom 1 og n + 1.
Hva betyr «n = int» og «(n, n + 1)»?

Forklar her:

5. Under ser du tre ulike figurer og tre formler for ulike figurtall.

Kvadrattallene
Illustrasjon av de første fire kvadrattallene osv. $K_n = n^2$

Rektangeltallene
Illustrasjon av de første fire rektangel osv. $R_n = n(n+1)$

Trekanttallene
Illustrasjon av de første fire trekanttallene osv. $T_n = 0{,}5n^2 + 0{,}5n$

6. Bruk formlene ovenfor og lag et program som gjør at du kan finne de første rektangeltallene eller trekanttallene. Skriv inn testalgoritmen under før du tester ut programmet i en editor.

Skriv testalgoritme her:

Figur 4.8: Oppgave hentet fra Matematikk 10 om kvadrattall (Hjardar, 2021, s. 55-56).

For å oppsummere funnet, blir ikke PRIMM-modellen benyttet direkte i lærebøkene, men flere av oppgavene, og samtlige oppgaver i Matematikk 10 gir elevene et ferdiglaget program først slik at de har noen elementer å ta utgangspunkt i når de skal endre eller lage et eget program.

Funn 2: Oppgavene som avviker fra PRIMM-modellen, handler for det meste om å lage et program.

Oppgavene som avvek fra den typiske strukturen til PRIMM-modellen, startet med eller utelukkende handlet om at elevene skulle utvikle et program fra grunnen av, uten at de ble presentert for eksisterende kode. I analysen av lærebøkene ble det identifisert en rekke slike oppgaver vist i figur 4.1, 4.3, 4.5, og 4.7. Totalt var det 17 slike oppgaver; tre av dem i Maximum 10, som tilsvarer omtrent en tredjedel av oppgavene, og 13 oppgaver i Matemagisk 10, tilsvarende halvparten av oppgavene.

Et konkret eksempel er vist i figur 4.9 med en oppgave 1 fra Topptur i kapitelet om Økonomi. I den første deloppgaven blir elevene bedt om å lage et program. Programmet skal simulere prisendringer på en vare. Programmet blir i deloppgavene b-h brukt for å svare på spørsmål om prisendringen, og undersøke ulike scenarioer der prisen på varen endres.

a Lag et regneark eller et program som lar deg skrive inn prisen G som en vare har i utgangspunktet. Du skal deretter regne ut hva det koster å

- sette prisen opp med a % og deretter ned med b %
- sette prisen ned med b % og deretter opp med a %

Du skal kunne endre tallene for G, a og b.

b Forklar hva du oppdager når du prøver deg fram med ulike tall for G, a og b.

c Forklar at når noe øker med p %, kan vekstfaktoren skrives som $1+\frac{p}{100}$.

d Forklar at når noe minker med p %, kan vekstfaktoren skrives som $1-\frac{p}{100}$.

Vi lar nå prisen G øke og minke med samme prosent, p.

e Forklar hvorfor du får samme svar enten du starter med å øke mengden med p % eller du starter med å minke mengden med p %.

f Vis ved regning at prisen etter de to endringene er $G\left(1-\frac{p^2}{100^2}\right)$.

Vi fortsetter med to og to slike prisendringer mange ganger.

g La n være et naturlig tall. Forklar at prisen etter $2n$ prisendringer, er $G\left(1-\frac{p^2}{100^2}\right)^n$.

h Hvilken pris vil varen få i det lange løp? Er prisen den ender opp med i det lange løp, avhengig av hva p er? Er den avhengig av hva G er? Begrunn svaret.

Figur 4.9: Oppgave hentet fra Matemagisk 10 om økonomi (Kongsnes & Wallace, 2021, s. 121)

Oppgaven gir elevene valgfrihet til å velge mellom å bruke regneark eller å utvikle et program, noe som reduserer kravene for programmeringstekniske forkunnskaper. Dette bidrar til å redusere inngangsterskelen for oppgaven, som er spesielt viktig når de neste syv oppgavene baserer seg på at elevene må klare oppgave a. Likevel er det verdt å nevne at oppgaven hadde fått en enda lavere terskel dersom elevene hadde kunne sett et liknende program, eller en halvferdig kode.

Oppgaven vist i figur 4.10 er strukturer motsatt, der skal elevene først regne for hånd, før de skal utvide oppgaven ved å lage et program. Dette tilrettelegger for at elever som ikke behersker programmering like godt fortsatt kan oppnå en form for mestring ved å fullføre deloppgave a. Deloppgave b introduserer imidlertid et ekstra nivå av kompleksitet, ettersom elevene må utvikle en generell metode for å løse problemet, i motsetning til bare å sjekke enkelttilfeller. En slik tilnærming kan stimulere til en annen form for tenkning hos elevene.

OPPGAVE 18.5

a Trekk sammen uttrykket $5x + 4 - 3x$ så mye som mulig.

b Lag et program som kan sjekke om verdien av det opprinnelige uttrykket og verdien av det forenklede uttrykket er den samme for ulike heltallige verdier av x.

Figur 4.10: Oppgave hentet fra Matemagisk 10 fra kapittelet å utforske matematiske sammenhenger (Kongsnes & Wallace, 2021, s. 13)

Oppsummert, er det en stor andel av oppgavene i Maximum 10 og Matemagisk 10 som avviker fra PRIMM-modell strukturen. Disse oppgavene presenterer ingen kode, og ber elevene utvikle et program. Programmet skal enten brukes for å besvare matematiske spørsmål, eller mot slutten av en oppgave for å undersøke i en større skala med flere tilfeller.

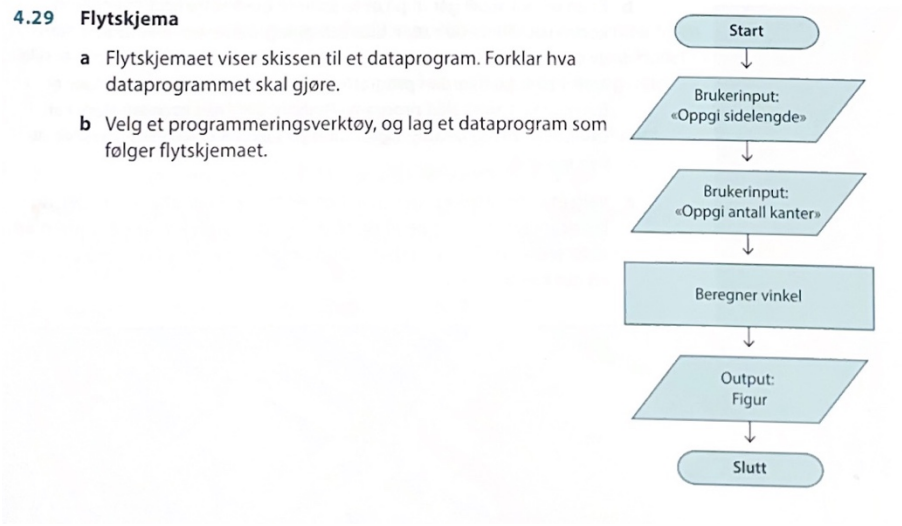
4.2.3 Delanalyse 3: Programmeringsspråk

I delanalyse 3 analyserte jeg hvilke programmeringsspråk bøkene brukte, og resultatene er vist i tabell 4.7. Oversikten viser at de fleste oppgavene benytter Python. Dette kommer frem ved at oppgavene viser til en kode i Python eller eksplisitt poengterer at elevene skal bruke det programmeringsspråket i oppgaveteksten. Oppgaver der språk ikke blir spesifisert kjennetegnes ved at ingen programmeringsspråk blir vist eller nevnt i oppgaven, se f.eks. figur 4.10 og 4.11.

Kategori	Maximum 10	Matemagisk 10	Matematikk 10	Totalt
Python	**7**	**17**	**5**	**29**
Pseudokode	1	-	-	1
Ikke spesifisert	1	9	0	10

Tabell 4.7: Antall oppgaver kategorisert etter programmeringsspråk

Én oppgave i Maximum 10 tar i bruk pseudokode, se figur 4.11. Oppgaven viser en skisse av et program skrevet i et flytskjema. Programmet tegner likesidede mangekanter basert på brukerens valg av sidelengde og antall kanter.



Figur 4.11: Oppgave med flytskjema fra Maximum 10 (Tofteberg et al., 2021, s.239)

Ved å analysere programmet som er skissert i flytskjemaet, blir elevene eksponert for pseudokode, og de får anledning til å øve på overgangen fra pseudokode til et konkret programmeringsspråk. I deloppgave b blir elevene bedt om å utvikle et dataprogram basert på flytskjemaet.

For å oppsummere delanalyse 3, benytter lærebøkene hovedsakelig tekstprogrammeringsspråket Python. Lærerbøkene spesifiserer sjelden programmeringsspråk i oppgaveteksten, men ber ofte elevene forutse, endre eller undersøke kode skrevet i Python. Eksemplene benyttet i lærebøkene er utelukket skrevet i Python, noe som kan indikere at Python også skal anvendes av elevene når de blir bedt

om å lage program. Det var ett tilfelle av pseudokode i form av et flytskjema, der elevene skulle undersøke flytskjema, før de skulle overføre det til et dataprogram.

4.2.4 Samarbeid

Ifølge Utdanningsdirektoratet (Udir, 2019) er samarbeid en viktig måte å utvikle algoritmisk tenkning på. Derfor merket jeg oppgavene som oppfordret til samarbeid. I læreboken Maximum 10 oppfordret fire av åtte programmeringsoppgaver elevene til å jobbe sammen. Dette blir tydeliggjort ved at oppgaveteksten starter med «samarbeid to og to», og som vist i figur 3.7 referer læreboken til leseren som «dere». Verken Matemagisk 10 eller Matematikk 10 hadde oppgaver som fremmet samarbeid.

4.2.5 Oppsummering av den vertikale analysen

For å oppsummere den vertikale analysen har jeg plassert hovedfunnene i tabell 4.8.

VERTIKAL ANALYSE AV LÆREBØKER

Kommunisert til elevene	**Forventet av elevene**
Matematisk innhold • Majoriteten av oppgaver bruker programmering som et verktøy for å effektivisere matematiske handlinger: enten ved å utføre beregninger eller tegne grafer. • Det eksisterer nesten ikke oppgaver uten kobling til matematikk. • Det er få oppgaver som bruker programmering som verktøy for å utforske matematikk	• I Matematikk 10 får elevene presentert en kode som de kan ta utgangspunkt i, dermed er kravene for forkunnskaper i programmering lavere, sammenlignet med Matemagisk 10 og Maximum 10 der flere oppgaver ber elevene lage program helt fra grunn av. • Elevene blir for det meste bedt om å undersøke, endre og lage kode. • Elevene blir sjeldent bedt om å forutse program. • Elevene ble i halvparten av oppgavene i Maximum 10 bedt om å samarbeide. I de to andre lærebøkene ble ikke samarbeid nevnt.
Matematiske praksiser • Bøkene anvender tekstprogrammeringsspråket Python. • Én oppgave i Maximum 10 benyttet pseudokode.	

Tabell 4.8: Oppsummering av den vertikale analysen.

5 Diskusjon

Målet med studien er å undersøke hvordan har norske lærebøker for matematikk på 10.trinn implementert programmering med fokus på programmering som verktøy. I denne delen skal samles teori og analyse i forsøk på å svare på forskningsspørsmålet. For å best mulig svare på forskningsspørsmålet har jeg strukturert diskusjonen slik at jeg først diskuterer funnene fra den horisontale analysen, deretter fra den vertikale analysen. Videre trekker jeg frem to biprodukt fra analysen før jeg tilslutt i gjør rede for studiens begrensninger.

5.1 Omfanget av programmering i lærebøkene

Den horisontale analysen viser en ulikhet i vektleggingen av programmering i lærebøkene. Matemagisk 10 har dobbelt så mange oppgaver med programmering enn det Matematikk 10 og Matemagisk 10 har. Matemagisk 10 har en jevn fordeling av oppgaver med programmering, og har programmeringsoppgaver tilhørende 5 temaer. Disse ligger i eller etter hvert tema, og er fordelt ut over alle nivåene. Maximum 10 har minst programmering, og har kun ni oppgaver fordelt på kapitelene om funksjoner og se flere sammenhenger, disse oppgavene er også plassert på flere nivåer. På bakgrunn av variasjonen i vektlegging av programmering i lærebøkene, og fordi læreplanen ikke har spesifisert omfanget, kan en anta at variasjonen i omfanget av bruken av programmering også gjenspeiles i undervisningen. Johannson(2006) viser at tematikk som ikke er beskrevet i lærebøkene sjeldent blir inkludert i undervisningen, dermed kan valg av lærebok kunne avgjøre i hvilken grad elevene får kjennskap til programmering.

Matematikk 10 har et eget hefte om programmering. Bakgrunnen for at de har valgt å ha programmering separat er at de da har muligheten til å oppdatere det (Cappelen Damm, u.å.). Heftet finnes i trykt eksemplar, men fordelen med å kunne oppdatere innholdet forsvinner dersom elevene over lenger tid tar utgangspunkt i det opptrykkede eksemplaret. Å ha et separat hefte med programmering kan være en ulempe, da læreren må skrive ut eller henvise til heftet i tillegg til grunnboken. Flere bøker, eller nettsider kan føre til at elever sliter med å holde styr på hvilke oppgaver de skal gjøre siden noen oppgaver er i grunnboken og andre i det digitale heftet. Programmeringsheftet tar for seg hele ungdomsskolen, altså trinnene 8 -10, mens grunnbøkene er delt etter trinn. Fordelen med det er at heftet gir en introduksjon i programmeringsteknikk som kan være til hjelp dersom elevene står fast uavhengig av trinn. En ulempe med at programmeringsheftet gjelder for flere trinn enn grunnboken kan være at enkelt elever opplever det som uryddig.

Den horisontale analysen viser at alle lærebøkene benytter programmering i sammenheng med funksjoner og utforske flere sammenhenger. I kapitlene bruker elevene programmering for å tegne grafer, og utforske ulike x-verdier i funksjonsuttrykk. En slik bruk av programmering stemmer overens med Haraldsrud et al. (2020) der programmering kobles til modellering, og gir elevene en modell å utforske matematikk gjennom. Et slikt modelleringsperspektiv kan koble programmering til kjerneelementene, og ved å bruke gode modeller får elevene fokusert på å utforske matematikken (Haraldsrud et al.,2020).

Ingen av lærebøkene benytter programmering i sammenheng med algebra. Dette kan komme av ulikhetene i representasjon i matematikk versus programmering. Gjøvik og Høyland (2022) peker på at elevene kan misforstå matematiske konsepter dersom de overfører betydningen fra programmering til matematikk. Likhetstegnet er et godt eksempel på en slik misoppfatning. I programmering bruker man likhetstegnet for å endre variablenes verdi, og man kan skrive x=x+2, noe som i matematikk er grunnleggende feil. Algebra krever ofte at elevene benytter matematisk notasjon, og i noen programmeringsspråk er det avvik i hvordan en representerer regneoperasjoner (Gjøvik & Høyland, 2022). Når det er sagt, har Python et bibliotek med regneoperasjoner elevene kan anvende. Den vertikale analysen viser at flere oppgaver anvender programmering for å effektivisere regneoperasjoner, men ingen oppgaver bruker programmering til å forkorte et algebraisk uttrykk eller løse en likning.

5.2 Kjennetegn ved programmeringsoppgavene

Den dominerende praksisen i de tre lærebøkene er å benytte tekstbasert programmeringsspråk, spesielt Python. Ingen av oppgavene eller eksemplene i bøkene anvender blokkprogrammering. Dette samsvarer med Udirs kompetansepakke for programmering og algoritmisk tenkning, som foreskriver bruk av tekstprogrammering på ungdomsskolenivå (Udir, u.å.). Valget av programmeringsspråk stemmer også overens med eksisterende litteratur (Matematikksenteret II, u.å.; Kristensen & Kirfel, 2022).

Den vertikale analysen viser at oppgavene i Matematikk 10 delvis er bygget opp etter PRIMM-modellen. Samtlige oppgaver begynner med å presentere en ferdigskrevet kode, som elevene undersøker før de blir bedt om å lage et eget program. Ved å alltid presentere et ferdigskrevet program som elevene kan ta utgangspunkt i, reduseres inngangsterskelen i oppgaven. PRIMM er utviklet for undervisning i programmeringsteknikk, mens målet med programmeringsoppgavene i lærebøkene for matematikk på 10. trinn er å bruke programmering i matematikkundervisningen (Coleman, 2021; Gjøvik & Høyland, 2022). Studier viser imidlertid at elever ofte står fast når de blir bedt om å lage et program, og for å lære matematikk gjennom programmering er det vesentlig at elevene klarer å lage programmet (Coleman, 2021). Derfor er PRIMM-metoden også aktuell når elevene skal lære matematikk gjennom programmering. Å ta utgangspunkt i en ferdigskrevet kode kan oppleves mer realistisk, da dataprogrammerere ofte benytter andres kode for å videreutvikle eller tilpasse koden til sitt formål (Kaufmann & Stenseth, 2019). Den vertikale analysen viser at oppgavene i Matemagisk 10 og Maximum 10 sjelden presenterer en ferdigskrevet kode før de ber elevene lage et program. Dette kan føre til at oppgavene i Matemagisk 10 og Maximum 10 muligens krever flere forkunnskaper i programmeringsteknikk, og ha høyere inngangsterskel enn Matematikk 10, hvor hver oppgave starter med å presentere et ferdigskrevet program.

Majoriteten av deloppgavene ba elevene undersøke enten et gitt program eller et program elevene selv hadde laget. Oppgavene som ba elevene undersøke, handlet om å forklare, feilsøke, se etter muligheter for å utvide programmet, fjerne unødvendig kode eller kjøre programmet med ulike verdier. Tidligere studier viser at feilsøking og analysering av kode er viktige evner for å mestre programmering (Crick, 2017; Vinnervik, 2023; Kaufmann & Stenseth, 2019). Delingskulturen i programmering er høy, og siden dataprogrammerere ofte videreutvikler og tilpasser andres kode, er det viktig at elevene kan lese og analysere kode (Kaufmann & Stenseth, 2019). Feilsøking er en arbeidsmetode i Den algoritmiske tenkeren, så ved å feilsøke kode kan elevene forbedre

evnen til algoritmisk tenkning (Udir, 2019a). Logikk i Den algoritmiske tenkeren er definert som å analysere og forutse, og gjennom å undersøke vil elevene få trent opp disse evnene, samt vurdere koden, som minner om nøkkelbegrepet evaluering. Oppgaver som ber elevene utforske kode, kan derfor bidra til å utvikle elevenes evner i algoritmisk tenkning.

En del oppgaver, spesielt i Maximum 10 og Matemagisk 10, ba elevene lage et program som kunne utføre en oppgave. Å lage et program fra bunnen av krever mer av elevene, spesielt når det gjelder programmeringsteknikk. Likevel peker Bråting & Kilhamn (2021) i sin studie av svenske lærebøker på at elever som jobber med steg-for-steg-oppgaver ikke ser helheten i prosessen og dermed mister muligheten til å forbedre evnen til algoritmisk tenkning. En sentral del av programmering er problemløsning, og den prosessen omhandler planlegging og evnen til å reflektere over hvorvidt løsningen er den beste mulige og om det er nødvendig å anvende teknologi for å løse problemet (Scherer et al., 2020). Elever som lager et program fra bunnen av, vil ta del i hele prosessen og gjennom det anvende flere aspekter ved algoritmisk tenkning. Ifølge Den algoritmiske tenkeren er det å skape en arbeidsmåte som kan fremme algoritmisk tenkning (Udir, u.å.). Ved å lage et program må elevene dekomponere problemet i mindre delproblemer, abstrahere ved å fjerne unødvendige detaljer, og foreta evaluering og feilsøking underveis (Udir, 2019a; Kaufmann & Stenseth, 2019). Ved å lage et program brukes flere nøkkelbegreper og arbeidsmåter som kan forbedre algoritmisk tenkning (Udir, 2019a; Mayer et al., 1986; Scherer et al., 2020). Det er derfor viktig at elevene får lage program for å løse et problem, men det betyr ikke at man ikke kan anvende PRIMM-metoden. For den siste fasen i PRIMM er å lage, og selv om elevene i de foregående fasene har fått sett og undersøkt et ferdigskrevet program, vil det nye programmet elevene skal lage ta for seg et annet problem, og de vil fremdeles kunne ta del i hele skape-prosessen.

Maximum 10 skiller seg ut ved å være den eneste læreboken som nevner samarbeid. I Maximum 10 oppmuntres elevene i halvparten av oppgavene til å samarbeide i par eller grupper. Der oppgavene ikke gir elevene ferdigskrevet kode, kan det være lurt å tillate elevene å samarbeide, da de sjeldnere står fast ved gruppearbeid (Hanks, 2004; Crick, 2017). Tidligere forskning viser at par-programmering har stor effekt på elevenes læringsutbytte, og i en matematikkbok der programmering kun skal benyttes som et verktøy for å lære matematikk, vil det å tillate samarbeid kunne redusere inngangsterskelen i oppgavene (Hanks, 2004; Crick, 2017). Med det sagt har ikke denne studien sett på lærebøkene i praksis, og det kan hende at læreren oppfordrer til samarbeid selv om bøkene ikke nevner det eksplisitt. Motsatt kan det også hende at elevene løser oppgavene individuelt, selv om Maximum 10 oppfordrer til samarbeid.

5.3 Programmering som middel for å utforske matematikk

Den vertikale analysen viser at de aller fleste oppgavene, med ett unntak, har en kobling til matematikk. Dette funnet skiller seg fra en lærebokanalyse gjort i Sverige, hvor oppgavene utelukkende handlet om programmeringsteknikk uten en matematisk kontekst (Bråting & Kilhamn, 2021). En mulig forklaring på denne forskjellen er at Bråting og Kilhamn (2021) studerte lærebøker for barneskolen. I henhold til den norske læreplanen skal elevene på 2.–5. trinn i matematikk lære programmeringsteknikk, og det er ingen krav om kobling til matematikk før 6. trinn.

For at en oppgave skal kunne kategoriseres med kobling til matematikk, kreves det at oppgaven tar utgangspunkt i matematikk. Oppgaven kan derimot handle om

programmeringsteknikk mens programmet utfører en matematisk handling. Et eksempel er en oppgave der elevene skal se hvordan while-løkken fungerer i en sparekalkulator. Et slikt eksempel kategoriseres som programmering med matematikk som kontekst.

Den dominerende praksisen i lærebøkene er å bruke programmering som et verktøy for å effektivisere matematiske handlinger. Analysen viser at lærebøkene bruker programmering til å effektivisere matematiske beregninger, tegne grafer og lage modeller. En slik anvendelse av programmering er til tross for at det ikke nevnes i læreplanen for 10. trinn, sentralt i flere av kjerneelementene, blant annet modellering og anvendelser. I dette kjerneelementet skal elevene få innsikt i hvordan de kan bruke matematikk i ulike situasjoner, både i faget og i elevenes liv (Udir, 2019b). At elevene kan anvende og utøve matematikk er nødvendige evner for å oppnå god matematisk kompetanse. Det å kjenne til mulighetene og begrensningene i digitale hjelpemidler anses som viktig for elevenes teknologiske kompetanse og algoritmiske tenkning (Udir, 2019b; Scherer et al., 2020). Det er derfor relevant for elevene å arbeide med programmering som et verktøy for å effektivisere matematiske handlinger.

Kompetansemålet for 10. trinn tilsier at elevene skal bruke programmering for å utforske matematiske sammenhenger og egenskaper. Den vertikale analysen viste at det var få slike oppgaver. Et naturlig spørsmål er da om der er mulig å utforske matematikk i arbeid med lærebøker alene? Utforskning slik det er forstått i Maaß & Reitz-Koncebovski (2013) omhandler en omfattende prosess der samarbeid står sentralt, og hvor det finnes flere fremgangsmåter. Oppgavene må ha stor takhøyde og være åpne. Analysen viser derimot at oppgavene i lærebøkene er relativt konkrete, noe som begrenser antall fremgangsmåter. De fleste av oppgavene som ble kategorisert som programmering som verktøy for utforskning nevnte begrepet eksplisitt og krever derfor at elevene kjenner til begrepet og prosessen. Slik utforskning er definert i denne oppgaven, er samtale en sentral del, enten med medstudenter eller ved veiledning av lærer. En slik forståelse av utforskende arbeid stemmer overens med det sosiokulturelle læringssynet (Vygotsky, 1978), hvor det finnes en grense for hva elevene klarer å lære uten innspill fra andre. Samarbeid er dermed et element som kan bidra til at elevene får en dypere matematisk forståelse, noe som oppfordres til i Maximum 10, men ikke i Matemagisk 10 eller Matematikk 10. Flere studier peker på at lærebøkene blir brukt til individuelt arbeid, ofte som mengdetrening etter en fellesundervisning (Pepin, 2010; Rezat & Strässer, 2017). Derfor gir det mening at de fleste oppgavene legger opp til nettopp individuelt arbeid, der elevene anvender programmering som et verktøy for å effektivisere matematiske handlinger. Likevel kan undervisningen anvende lærebøkene annerledes, og det kan hende at oppgavene som i denne analysen er blitt kategorisert som utforskende, faktisk bidrar til å gi elevene en dypere matematisk forståelse.

5.4 Programmering som verktøy

Et biprodukt av denne studien er spørsmålet om programmering brukes fordi det er det mest passende digitale verktøyet, eller om andre verktøy ville vært mer hensiktsmessige. Lærebøkene har totalt flest programmeringsoppgaver i sammenheng med funksjoner. Der skal elevene undersøke funksjonsuttrykk og tegne grafer. Den digitale graftegneren Geogebra er brukt i samtlige av lærebøkene, og er en intuitiv og brukervennlig graftegner. Programmet har knapper med kommandoer, og elevene kan gå inn i graftegneren og justere og undersøke punkter på grafen. Et av argumentene for programmering i skolen er at det fremmer algoritmisk tenkning. Det gjør også Geogebra i følge Chytas et al. (2024), og en kan derfor stille spørsmål ved hvorfor læreplanen

akkurat fremmer programmering og ikke digitale verktøy generelt. Av egen erfaring fra praksis opplever jeg at elever har større kjennskap til og erfaringer med Excel og Geogebra enn programmering. Excel og Geogebra har i sammenligning med programmering mer intuitive og brukervennlige grensesnitt. I læringssammenheng kan derfor bruk av disse gi lav inngangsterskel og stor takhøyde. Dette kommer frem i lærebøkene, spesielt i Matematikk 10 der Geogebra og Regneark er inkludert i grunnboken, mens programmering er plassert i et separat hefte. I datamaterialet var det flere oppgaver, for eksempel 4.3 og 4.6, der elevene skulle tegne grafer ved hjelp av programmering. I oppgaver med graftegning kan det hende at elevene ville fått et bedre matematisk utbytte av å bruke Geogebra enn programmering, siden Geogebra gir elevene mulighet til å bevege grafene og opprette punkter i koordinatsystemet.

Lærebøkene veksler mellom ulike digitale verktøy som elevene skal bruke, noe som er positivt og gir elevene kjennskap til flere verktøy. Et eksempel fra Maximum 10, ber elevene først benytte regneark og så programmering, og på den måten får elevene kjennskap til begge, og kan senere bestemme hvilket verktøy de selv foretrekker. Derimot er det veldig få oppgaver der eleven får velge hvilket verktøy de skal benytte. Oppgaver der elevene skal velge er ofte karakterisert som større, som «snakke sammen» eller «aktivitet». Chytas et al. (2024) hevder at innsikt i digitale verktøy er en vesentlig del av matematisk kompetanse. De argumenterer for at elevene skal prøve flere verktøy, og få oversikt over verktøyenes funksjoner og begrensinger (Chytas et al., 2024). Med det perspektivet er det positivt at lærebøkene ber elevene anvende programmering, Geogebra og regneark litt om hverandre, slik at de blir kjent med flere digitale verktøy.

5.5 Programmering som mål

Et annet biprodukt av denne studien er om programmering kan være et mål i seg selv. Innledningsvis ble det poengtert hvor viktig teknologi er i dag og vil være i fremtiden. Haraldsrud et al. (2020) og Kafia og Burke (2013) mener at evnen til å lese kode i dag er nødvendig for å lese verden med tanke på å forstå og se muligheter innen teknologi. Hver gang en datamaskin åpnes og noen går inn på en nettside kjøres det mange linjer med kode (Haraldsrud et al., 2020). Skolen skal i henhold til opplæringsloven gi elevene den kompetansen de trenger for å mestre eget liv og aktivt delta i samfunnet (Opplæringsloven, 2006, §1-1). I en tid der teknologi tar stadig større plass, er det derfor viktig at elever har kompetanse i programmering. Fremtidens arbeidsmarked vil forandre seg, og mange av dagens skoleelever vil arbeide innen datavitenskap (Kafia & Burke, 2013). Det er mangel på mangfold innen datavitenskapelige yrker, noe som kan være problematisk, spesielt ettersom kunstig intelligens har potensiale til å kunne overta flere viktige arbeidsoppgaver, som ansettelser. Det er derfor viktig at disse systemene utvikles av en mangfoldig gruppe for å unngå bias (Kafia & Burke, 2013). Skolen kan ved å innføre programmering bidra til å engasjere flere inn i datavitenskapen, slik at mangfoldet i skolen kan gjenspeiles på datavitenskapelige arbeidsplasser. Selv for elever som ikke vil jobbe innen data, vil forståelse for programmering og algoritmisk tenkning være nyttig, da nesten alle vil være brukere av digitale teknologier. Evnen til å forstå og kritisk vurdere beslutninger bak teknologiutvikling er derfor sentralt (Kafia & Burke, 2013). I England er programmering blitt integrert i skolen som et eget fag, noe som signaliserer viktigheten av at elever kan programmere (Sevik et al., 2016). Et forslag om teknologi og programmering som eget fag ble foreslått av en rapport om teknolog og programmering i skolen bestilt av Utdanningsdirektoratet (2016). Arbeidsgruppen mente at å lære programmering kan ha en verdi for elevene i seg selv. I tillegg til at

programmering kan brukes som et verktøy i matematikk, og for å utvikle algoritmisk tenkning.

5.6 Begrensinger med studien

Studien har tatt utgangspunkt i programmeringsoppgavene i grunnbøkene Matematikk 10, Maximum 10 og Matemagisk 10. Oppgavene med programmering er identifisert av meg som forsker, og det kan hende det er enkelte oppgaver som er blitt oversett. Ved å kun studere en liten del av lærebøkene er det vanskelig for meg som forskere å se kontekstene oppgavene er gitt i, og lærebøkene som en helhet. Derfor er ikke målet med studien å vurdere kvaliteten på lærebøkene, men å undersøke hvordan bøkene anvender programmering.

Utvalget er begrenset til de tre grunnbøkene på 10.trinn, så lærerveiledninger og tilleggsmaterialet er ikke tatt med i betraktningen i denne studien. Dermed kan studiens implikasjoner av implementeringen av programmering i lærebøkene avvike fra totalpakken med undervisningsmateriell fra de tre forlagene. Det kan derfor hende at funnene i denne studien ikke reflekterer undervisningsmateriellet i sin helhet. Med det sagt viser teorien i denne studien at de fysiske lærebøkene har en viktig rolle i undervisningen, og at tematikk som ikke presenteres i lærebøkene, ikke blir inkludert i undervisningen. Utvalget er avgrenset til 10.trinn, og funnene kan kun indikere hvordan programmering er inkludert på det trinnet, og er dermed ikke representativt for læreverkets bruk av programmering i sin helhet.

Studien inkluderer ikke læreres eller elevers erfaringer og synspunkter på bruken av programmeringsoppgaver i undervisningen, noe som kan gi en skjev framstilling av hvordan disse oppgavene gjennomføres i praksis. Studien baserer seg utelukkende på en innholdsanalyse av lærebøkene, derfor kan ikke studien si noe om hvordan oppgavene blir brukt i undervisning, eller hvilket læringsutbytte lærebøkene gir i praksis. Innholdsanalysen er gjort med utgangspunkt i teorien vist i kapittel 2, og med min analyse. Det kan derfor hende at med utgangspunkt i annen teori ville jeg fått andre funn, og dersom en annen forsker hadde gjentatt studien vil også funnene kunne avvike fra denne studien.

6 Avslutning

Som avslutning på masteroppgaven ønsker jeg i korte trekk svare på forskningsspørsmålet: Hvordan har norske lærebøker for matematikk på 10.trinn implementert programmering med fokus på programmering som et verktøy i matematikk? Samt se på videre forskning av programmering i lærebøker.

6.1 Konklusjon

Studien viser at de tre lærebøkene i matematikk har variert omfang av programmeringsoppgaver. Matemagisk 10 har oppgaver med programmering knyttet til nesten hvert tema, mens Maximum 10 kun har inkludert programmering i kapitlene om funksjoner og utforskning av matematiske sammenhenger. Ingen av bøkene inkluderer programmering i algebra, sannsynligvis på grunn av forskjellene i matematisk notasjon og programmeringsspråk. Lærebøkene anvender utelukkende tekstbasert programmering, hvor Python er den dominerende praksisen, noe som samsvarer med Utdanningsdirektoratets kompetansepakke i programmering og algoritmisk tenkning for ungdomsskolen.

Koblingen mellom matematikk og programmering er tilstede i lærebøkene. Studien viser at lærebøkenes dominerende praksis er å bruke programmering som et verktøy for å effektivisere matematiske handlinger som beregninger, graftegning og modellering. Det det er dog få oppgaver der elevene bruker programmering til å utforske matematiske sammenhenger og egenskaper. Studien stiller spørsmål ved om det er mulig å bruke programmering til å utforske matematikk kun ved bruk av individuelt arbeid med lærebøker.

Programmeringsoppgavene i lærebøkene veksler mellom å gi elevene et ferdiglaget program og å lage program selv. Når elevene får utdelt et ferdigskrevet program, skal de ofte analysere og foreta endringer i programmet for å tilpasse det til det aktuelle problemet. Studien belyser flere fordeler ved oppgaver som er strukturert etter PRIMM-modellen, der elevene blir presentert en ferdigskrevet kode. Når elever analyserer kode får de arbeidet med sentrale arbeidsmåter for algoritmisk tenkning som feilsøke og fikle. Samt at når de selv skal lage et program, så vil inngangsterskelen i oppgaven reduseres dersom elevene kan ta utgangspunkt i ferdigskrevet kode og bruke elementer fra den i deres egne program. Kravet om forkunnskaper i programmeringsteknikk minimeres ved bruk av PRIMM, og programmering blir i PRIMM strukturerte oppgaver ikke et hinder fra elevenes matematiske utbytte av oppgavene.

6.2 Videre forskning

Et av spørsmålene som ble løftet frem i denne studien, men som studien ikke klarer å besvare, er om det er mulig å utforske matematikk i lærebøkene. For å kunne besvare dette spørsmålet, ville det vært interessant å gjennomføre en studie med et rammeverk for utforskende matematikk for eksempel 5E-modellen (Benton et al., 2016; 2017). I tillegg er det nødvendig å undersøke lærebøkene i praksis og se hvordan de faktisk blir brukt i undervisningen. Flere studier peker på at lærebøkene hovedsakelig blir brukt som mengdetrening etter fellesundervisning. Hvis mengdetrening er formålet med

lærebøkene og det er slik de blir brukt i undervisningen, kan de, som denne studien indikerer at de gjør, beholde fokuset på oppgaver der elevene anvender kjent matematikk i programmeringsoppgavene.

Ved å studere lærebøker på flere trinn og i praksis kan en få et dypere innblikk i hvordan programmering er implementert i matematikkundervisningen. Man vil da kunne undersøke om progresjonen i programmering, slik den er skissert trinnvis i læreplanen, er realistisk. Dersom man studerer et trinn over tid, kan man undersøke hvordan oppgavene i Matematikk 10, Matemagisk 10 og Maximum blir løst ulikt basert på hvor lenge elevene har fått undervisning i programmering. Kullet som begynte i første klasse i 2020 vil ha en helt annen kompetanse i programmering enn kullene som går ut nå. Det kan derfor være interessant å se på hvilke revideringer Cappelen Damm gjør i det digitale programmeringsheftet til Matematikk 10.

Referanser

Arfe, B., Lavanga, M., Montuori, C. & Vardanega, T. (2019). Coding in Primary Grades Boosts Children's Executive Functions. Frontiers in Psychology, https://doi.org/10.3389/fpsyg.2019.02713

Benton, L., Hoyles, C., Kalas, I., & Noss, R. (2016). Building mathematical knowledge with programming: Insightsfrom the ScratchMaths project. In A. Sipitakiat & N. Tutiyaphuengprasert (Eds.), Constructionism in Action2016, Conference Proceedings (pp. 25–32). Suksapattana Foundation.

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

Berg, T.K. (2021). Analog programmering. Tangenten – tidsskrift for matematikkundervisning, 32(3), 42–52.

Berntsen, K.R., & Holone.J, (2016). Programmering som et pedagogisk verktøy i matematikkundervisningen. Presentert på Norsk Informatikkonferanse 2016.

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

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

Cappelen Damm (u.å.). *Matematikk 8-10 fra Cappelen Damm*. Cappelendamm.no. Hentet 28.april 2024 fra https://utdanning.cappelendamm.no/verk/matematikk-8-10-fra-cappelen-damm-153429?omtale

Charalambous, C., Delaney, S., Hsu, H.-Y. & Mesa, V. (2010). A Comparative Analysis of the Addition and Subtraction of Fractions in Textbooks from Three Countries. Mathematical Thinking and Learning. 12. 117-151. https://doi.org/10.1080/10986060903460070

Chytas, C., van Borkulo, S.P., Drijvers, P. et al. (2024). Computational Thinking in Secondary Mathematics Education with GeoGebra: Insights from an Intervention in Calculus Lessons. Digit Exp Math Educ. https://doi.org/10.1007/s40751-024-00141-0

Coleman, G. (2021). Hello World – the big book of computing pedagogue- The Raspberry Pi Foundation. https://www.raspberrypi.org/hello-world

Creswell, J. W., & Creswell, J. D. (2018). Research Design: Qualitative, Quantitative & Mixed Methods Approaches. SAGE.

Creswell, J. W., & Poth, C. N. (2018). *Qualitative inquiry & research design : choosing among five approaches* (4th edition.). Sage.

Crick, T. (2017). Computing education: An overview of research in the feld. Cronfa - Swansea University Open Access Repository, (April), 1–38. Retrieved from http://cronfa.swan.ac.uk/Record/cronfa43589.

Den nasjonale forskningsetiske komité for samfunnsvitenskap og humaniora [NESH] (2021). Forskningsetiske retningslinjer for samfunnsvitenskap, humaniora, juss og teologi.https://www.forskningsetikk.no/globalassets/dokumenter/4-publikasjoner-som-pdf/forskningsetiske-retningslinjer-forsamfunnsvitenskap-og-humaniora

Fauskanger, J. og Mosvold, R. (2014) Innholdsanalysens muligheter i utdanningsforskning. Norsk Pedagogisk. Tidsskrift, 98(2), pp. 127-139

Flø, E. E. (2021). Programmering i LK20 Tangenten – tidsskrift for matematikkundervisning, 32(1), 3–9.

Forsström, S. E. & Kaufmann, O. T. (2018). A literature review exploring the use of 64 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

Gjøvik, Ø. (2024, Mai) *Matematikk og programmering*. Seksjon for naturfag, ILU PILS-konferansen, Trondheim.

Gjøvik, Ø., & Høyland, J. (2022). Kloss for kloss: blokkprogrammering for lærere. Universitetsforlaget.

Gjøvik, Ø., & Torkildsen, H. A. (2019). Algoritmisk tenkning. Tangenten – tidsskrift for matematikkundervisning, 30(3). 31–37.

Gyldendal (u.å.). *Maximum 10, 2.utgave, Grunnbok*. Gyldendal.no. Hentet 28.april 2024 fra https://www.gyldendal.no/grs/maximum/10/maximum-10-2-utgave-grunnbok/p-10026488-no/

Haggarty, L., & Pepin, B. (2002). An investigation of mathematics textbooks and their use in English, French and German classrooms: Who gets an opportunity to learn what? British Educational Research Journal, 28(4), 567–590. https://doi.org/10.1080/0141192022000005832

Hanks, B., McDowell, C., Draper, D., & Krnjajic, M. (2004). Program quality with pair programming in CS1. In Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE '04) (pp. 176–180).

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

Hjardar, E. & Pedersen, J.E., (2020). Matematikk 10 fra Cappelen Damm: Grunnbok (Bokmål[utgave], utgave 1.). Cappelen Damm.

Hsieh, H. & Shannon, S. E. (2005). Three approaches to qualitative content analysis. Qualitative Health Research, 15 (9), s. 1277–1288.

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

Kafai, Y. & Burke, Q. (2013). Computer Programming Goes Back to School. Phi Delta Kappan. 95. 61– 65. https://doi.org/.1177/003172171309500111.

Kallia, M., van Borkulo, S., Drijvers, P., Barendsen, E., & Tolboom, J. (2021). Characterisingcomputational thinking in mathematics education: A literature-informed Delphi-study.RESEARCH IN MATHEMATICS EDUCATION21Research in Mathematics Education,23(2), 159–187.https://doi.org/10.1080/14794802.2020.1852104

Kaufmann, O.T., & Stenseth, B. (2020). Programming in mathematics education. International Journal of Mathematical Education in Science and Technology. https://doi.org/10.1080/0020739X.2020.1736349

Kilhamn, C., Bråting, K., & Rolandsson, L. (2021). Teachers'arguments for including programming in mathematicseducation. In G. A. Nortvedt, N. F. Buchholtz, J. Fauskanger, F. Hreinsdóttir, M. Hähkiöniemi, B. E. Jessen,A.Werneberg (Eds.), Bringing Nordic mathematics education into the future. Preceedings of Norma 20 The ninthNordic Conference on Mathematics Education. SMDF, Svensk Förening för MatematikDidaktisk Forskning, Nr 14.

Kongsnes, A. L., & Wallace, (2020). Matemagisk 10 (Bokmål[utgave], 1. utgave.). Aschehoug undervisning.

Kristensen, T. E., & Kirfel, C. (2022). Programmering i matematikkfaget (1. utgave.). Cappelen Damm akademisk.

Kunnskapsdepartementet (2019). Læreplan i matematikk fellesfag 1.–10. trinn. Fastsatt som forskrift. Læreplanverket for Kunnskapsløftet 2020.

Lambic, D. (2011). Presenting practical application of mathematics by the use of programming soft-ware with easily available visual components.Teaching Mathematics and Its Applications,30(1),10–18. https://www.researchgate.net/publication/270821026_Presenting_practical_application_of_Mathematics_by_the_use_of_programming_software_with_easily_available_visual_components

Looi, M.L.H.C, Longkai, W., Seow, P., & Liu, L. (2018). Analysis of linkages between an unplugged activity and the development of computational thinking. Journal of Educational Computing Research, 58(2), 255–279. https://doi.org/10.1080/08993408.2018.1533297

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.

Matematikksenteret a (u.å). *Programmering i LK20.* Matematikksenteret.no. https://www.matematikksenteret.no/l%C3%A6ringsressurser-og-undervisningsopplegg/programmering/programmering-i-lk20 (Hentet mars 2024)

Matematikksenteret b (u.å). *Om programmeringsverktøyene.* Matematikksenteret.no. Hentet fra: https://www.matematikksenteret.no/l%C3%A6ringsressurser-og-undervisningsopplegg/programmering/om-programmeringsverkt%C3%B8yene

Matematikksenteret c (u.å.). *Didaktisk grunnlag.* Matematikksenteret.no. Hentet fra: https://www.matematikksenteret.no/l%C3%A6ringsressurser-og-undervisningsopplegg/programmering/anbefalt-arbeidsmetodikk

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

Opplæringsloven. (2006). Lov om grunnskolen og den videregående opplæringen (LOV-1998-07-17-61). Lovdata. https://lovdata.no/forskrift/2006-06-23-724

Ot.prp. 44 (1999-2000). Godkjenning av lærebøker. Krav om språkleg kvalitet. Kultur- og likestillingsdepartementet. https://www.regjeringen.no/no/dokumenter/otprp-nr-44-1999-2000-/id586147/?ch=4

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

Pepin, B. (2010). Mathematical tasks and learner dispositions: A comparative perspective. In V.Durand-Guerrier, S. Soury-Lavergne, & F. Arzarello (Eds.), Proceedings of the Sixth Congress of the European Society for Research in Mathematics Education (pp. 2504−2512). Lyon, France: INRP.

Pepin, B., Haggarty, L. & Keynes, M. (2001). Mathematics textbooks and their use in English, French and German classrooms: Zentralblatt für Didaktik der Mathematik 33, 158–175. https://doi.org/10.1007/BF02656616

Rezat, S., Fan, L. & Pepin, B. (2021). Mathematics textbooks and curriculum resources as instruments for change. ZDM Mathematics Education 53, 1189–1206. https://doi.org/10.1007/s11858-021-01309-3

Rezat, S., & Strässer, R.(2017). Methodological issues and challenges in research on mathematics textbooks. Grevholm, B., & Nordic Graduate School in Mathematics Education. (Red.). Mathematics textbooks, their content, use and influences : research in Nordic and Baltic countries. Cappelen Damm akademisk.

Scherer, R., Siddiq, F., & Sánchez Viveros, B. (2019). The cognitive benefits of learning computer programming: A meta-analysis of transfer effects. Journal of Educational Psychology, 111(5), 764–792. https://doi.org/10.1037/edu0000314

Sentance, S., Waite, J. and Kallia, M. (2019). Teaching computer programming with PRIMM: a sociocultural perspective, Computer Science Education.https://doi.org/10.1080/08993408.2019.1608781

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

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

TV2 (2002). 21.Nyhetene [Nyhetssending]. TV2

Utdanningsdirektoratet (2024). Eksamensoppgave Matematikk 10.årstrinn (MAT0015). Hentet fra https://sokeresultat.udir.no/eksamensoppgaver.html#:~:text=Matematikk-,10,-.%20%C3%A5rstrinn%2C%20del%201

Utdanningsdirektoratet (2019a). Den algoritmiske tenkeren. Hentet fra https://www.udir.no/kvalitet-og-kompetanse/digitalisering/algoritmisk-tenkning/

Utdanningsdirektoratet (2019b). Hva er kjerneelementer? Hentet fra https://www.udir.no/laring-og-trivsel/lareplanverket/stotte/hva-er-kjerneelementer/

Utdanningsidrektoratet (2016). Teknologi og programmering for alle – En faggjennomgang med forslag til endringer i grunnopplæringen- august 2016. https://www.udir.no/globalassets/filer/tall-og forskning/forskningsrapporter/teknologi-og-programmering-for-alle.pdf

Utdanningsdirektoratet (u.å.). Programmering og algoritmisk tenkning. https://www.udir.no/kvalitet-og-kompetanse/kompetansepakker/

Utdanningsnytt. (2024, 08.mai). Regjeringen vil gi 300 millioner kroner til papirbøker i skolen. Hentet fra: https://www.utdanningsnytt.no/ntb-skoleboker/regjeringen-vil-gi-300-millioner-kroner-til-papirboker-i-skolen/400236

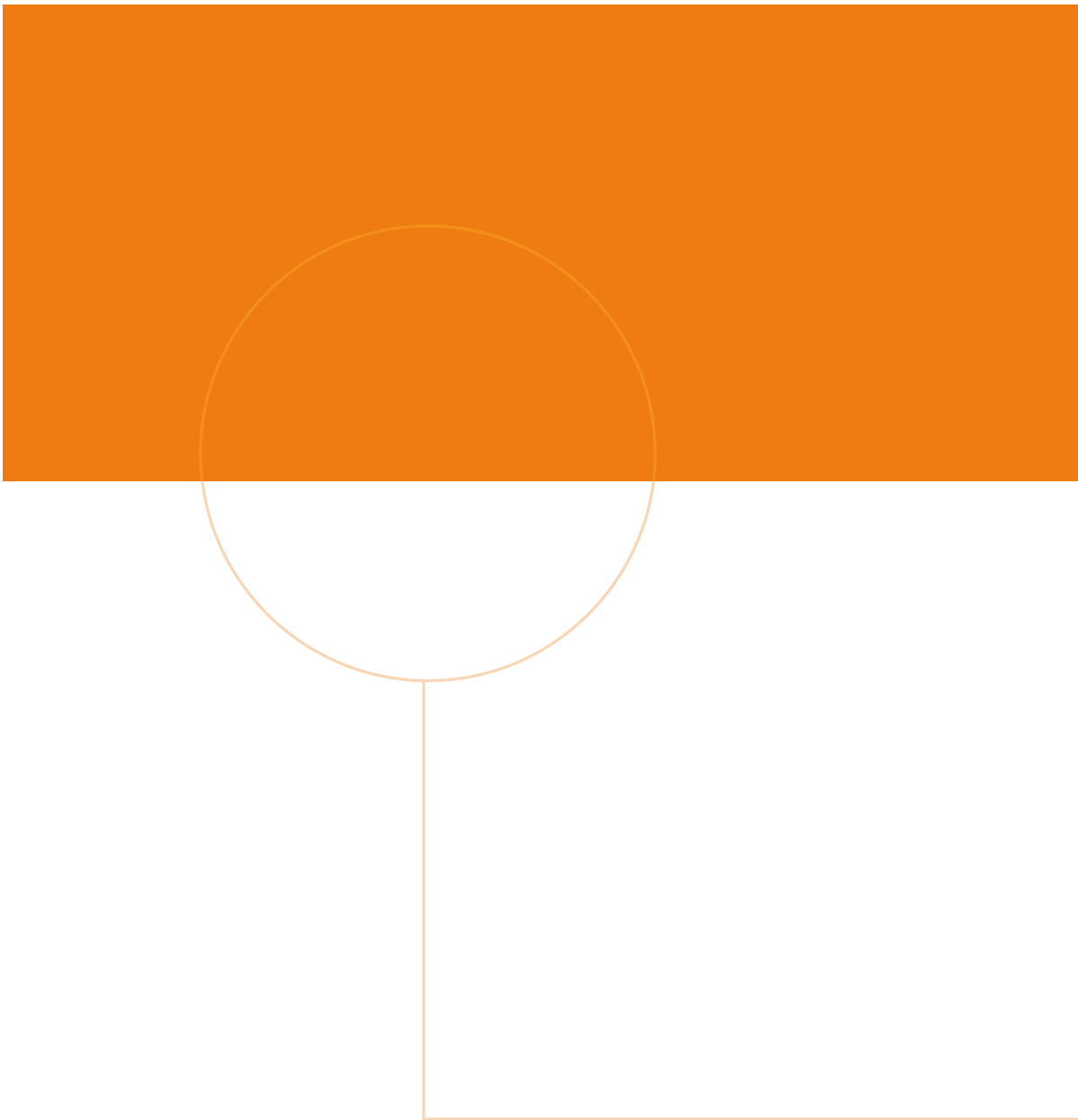
Valverde, G.A., Bianchi, L.J., Wolfe, R.G., Schmidt, W.H. & Hoang, R.T. (2002). According to the book. Springer. https://doi.org/10.1007/978-94-007-0844-0

Vinnervik, P. (2023). Programming in school technology education: the shaping of a new subject content. International Journal of Technology and Design Education, 33(5), 1449–1470. https://doi.org/10.1007/s10798-022-09773-y

Vinnervik, P. (2022). An in-depth analysis of programming in the Swedish school curriculum—rationale, knowledge content and teacher guidance. Journal of Computer Education, 10(3), 237–271. https://doi.org/10.1007/s40692-022-00230-2

Vinnervik, P., & Bungum, B. (2022). Computational thinking as part of compulsory education: How is it represented in Swedish and Norwegian curricula? Nordic Studies in Science Education, 18(3), Artikkel 3. https://doi.org/10.5617/nordina.9296

Vygotsky, L. S. (1978). Mind in Society: Development of Higher Psychological Processes (M. Cole, V. Jolm-Steiner, S. Scribner, & E. Souberman, Eds.). Harvard University Press. https://doi.org/10.2307/j.ctvjf9vz4



NTNU

Kunnskap for en bedre verden