Programmering i matematikkundervisning

Intervju med tre lærere om matematisk kompetanse gjennom programmering

Tommy Odeh

Masteroppgave i matematikk ved NLA Høgskolen Bergen

Våren 2023

Forord

Mine fem år på lærerutdanningen ved NLA fullføres ved ferdigstillingen av denne oppgaven. Det har vært en innholdsrik utdannelse hvor jeg har blitt kjent med mange flotte mennesker som har vært med å utfordre, utfolde og utvikle meg som lærer. Jeg har gjennom denne oppgaven fått muligheten til å dykke dypere i noe som fasinerer meg, og jeg håper at dette bidraget kan inspirere andre til å gjøre programmering til en større del av matematikkundervisningen.

I forbindelse med dette arbeidet ønsker jeg å rette en takk til min veileder, Torbjørn Aadland, som har vært en god ressurs fra dag én. Han har pekt meg i riktig retning, hjulpet meg med formuleringer og funnet god litteratur til prosjektet.

Videre ønsker jeg å takke Kristin og Paul Reidar som har hjulpet meg å komme i havn med oppgaven. Takk, Kristin for lån av arbeidsplass, slik at jeg fikk sitte i rolige omgivelser og jobbe med oppgaven i innspurten. Og takk til Paul Reidar som las gjennom oppgaven min og pekte ut punkter jeg kunne forbedre og endre på før innleveringen.

Til slutt ønsker jeg å takke min fantastiske familie. Takk til min datter Kornelia, for at du har gitt meg så mye å le av og glede meg over. Hver dag har jeg kunne glede meg til å se smilet ditt når jeg kommer hjem. Og takk til verdens beste mamma og kone, Bertine, som har vært der og støttet meg gjennom hele oppgaven. Hadde det ikke vært for deg, hadde jeg aldri blitt ferdig med dette. Du gjorde en ekstra innsats hjemme når jeg ikke kunne, du har ofret din fritid, slik at jeg kunne få arbeide mer, og du har vært der for Kornelia når jeg ikke kunne. Du har vist forståelse, medlidenhet og gitt meg motivasjon. Så takk. For alt du er, og alt du gjør.

Tommy Odeh

Juni 2023

tommy.odeh@gmail.com

Abstract

This master's thesis focuses on mathematical competence and programming in mathematics education at the secondary school level. There have been ongoing discussions and arguments regarding the introduction of programming in schools, where students acquire new essential skills in preparation for the future. In 2020, programming was integrated into mathematics education, and many mathematics teachers now lack high competence in programming, needing to teach students at different levels how to use and master this tool. While there is only one specific competency goal mentioning programming at each grade level, programming is a skill and a tool that takes a significant amount of time to master. Therefore, there is a strong rationale for incorporating programming into other competency goals.

My research question was: "Which mathematical competencies do certain subject teachers aim to promote through programming in mathematics education at the secondary school level?"

In the theoretical part of this thesis, I explain various concepts related to programming and mathematics education, as well as examine the relationship between programming, mathematics, algorithmic thinking, and mathematical competence. Research indicates that programming can strengthen students' computational thinking, mathematical thinking, problem-solving abilities, and more.

To address my research question, I conducted interviews with three teachers who teach mathematics at the secondary school level. All three are comfortable integrating programming into their mathematics teaching and have either received further education in programming or teach programming as an elective subject. The interviews were transcribed, analyzed, and then examined in the context of eight mathematical competencies.

My research shows that these three teachers view programming as well-suited for problem-solving, modeling, and generalization. However, they often encounter a competence gap among students. If the programming aspect is appropriately challenging, the mathematics may be too simple, and if the mathematics is at an appropriate level, the programming becomes too advanced. Nevertheless, the teachers have a positive outlook on using programming in their teaching and believe that students will benefit greatly from it in the future, whether in professional contexts or through the skills they have learned through programming.

Sammendrag

Denne masteroppgaven handler om matematisk kompetanse og programmering i matematikkundervisning på ungdomsskolen. Det har lenge vært diskusjoner og argumenter rundt innføringen av programmering i skolen, hvor elevene lærer seg nye nødvendige ferdigheter i møte med framtiden. I 2020 ble programmering innført som en del av matematikkundervisningen, og mange matematikklærere sitter nå uten høy kompetanse i programmering, og må lære elever som ligger på ulike nivå til å bruke og mestre dette verktøyet. Det er kun et kompetansemål på hvert trinn som nevner programmering eksplisitt, selv om programmering er en ferdighet og et verktøy som kan ta lang tid å mestre. Derfor har vi en god grunn til å benytte oss av programmering inn mot andre kompetansemål. Problemstillingen min ble da:

«Hvilke matematiske kompetanser ønsker enkelte faglærere å fremme gjennom programmering i matematikkundervisning på ungdomsskolen?»

I teoridelen av denne oppgaven, gjør jeg rede for ulike begreper knyttet til programmering og matematikkundervisningen, samt ser nærmere på relasjonen mellom programmering, matematikk, algoritmisk tenkning og matematisk kompetanse. Forskningen viser at programmering kan være med å styrke elevers algoritmiske tenkning, matematiske tenkning, problemløsningferdigheter og mer.

For finne ut av problemstillingen min, har jeg intervjuet tre lærere som underviser matematikk på ungdomsskolen. Alle tre er komfortable med å integrere programmering i undervisningen sin, og har enten videreutdanning innen programmering, eller underviser også i valgfaget programmering. Intervjuene har blitt transkribert, deretter analysert og til slutt sett i lyset av åtte matematiske kompetanser.

Forskningen min viser at disse tre lærerne ser på programmering som godt egnet i møte med problemløsning, modellering, og generalisering. Men per i dag opplever lærerne ofte et kompetansegap hos elevene. For at elevene skal få skikkelig utbytte av den matematiske delen, blir programmeringen for avansert, og hvis programmeringen ligger på et passelig nivå, er matematikken for simpel. Men likevel ser lærerne positivt på å bruke programmeringen i undervisningen, og mener at elevene vil få godt nytte av dette i framtiden, enten i jobbsammenheng, eller av ferdigheter de har lært gjennom å programmere.

Innhold

Kapittel 1 - Innledning

Programmering har de siste årene blitt sett på som en ferdighet som er mer og mer nødvendig for den nye generasjonen (Bocconi et al., 2018, s. i). Flere av de nordiske landene har anerkjent dette gjennom å innføre programmering som en del av den generelle grunnskoleopplæringen. I Norge er det ikke lenger en ferdighet eller et fag som elever kan velge å ikke forholde seg til, for med LK20 har det blitt innført at alle skal lære programmering fra og med 5. klasse og ut grunnskolen (Utdanningsdirektoratet, 2019c). Andre land har endt med forskjellige løsninger, og løsningen her i Norge var å gjøre det til en del av matematikkopplæringen. Innføringen av programmering i matematikkundervisningen har åpnet dørene for en ny og spennende tilnærming til læring. Kombinasjonen av matematikk og programmering gir elevene muligheten til å utvikle problemløsningsferdigheter, abstrakt tenkning og logisk resonnering på en kreativ og praktisk måte. Dette er fordi programmering og matematikk begge bygger på logisk tenkning (Kaufmann & Stenseth, 2020, s. 2).

Denne integrasjonen av programmering i matematikkfaget har potensial til å styrke elevenes matematiske forståelse og forberede dem på en stadig mer teknologidrevet verden. Samtidig byr dette på mange nye utfordringer, blant annet at matematikklærere som ikke nødvendigvis selv kan programmere nå har fått ansvaret for å lære elevene sine det. Dette ser vi blir problematisert gjennom hva Forsström og Kaufmann skriver:

> «One relevant topic to consider regarding pedagogical solutions is the role of the teacher. When integrating programming with a mathematics curriculum, the role of the teacher may become challenging because the mathematics teacher may not have previous knowledge of programming» (Forsström & Kaufmann, 2018, s. 19).

I en faggjennomgang av matematikkfagene, foreslos det en etterutdanning av matematikklærere i digitale verktøy. Gjennom en etterutdanning og tilgang til flere ressurser, kan man øke nytten av digitale verktøy i undervisningen (Borge et al., 2014, s. 84). Programmering er et av mange digitale verktøy man kan ta i bruk i matematikkundervisningen.

En annen viktig utfordring, er at å lære seg å programmere kan ta lang tid og er en omstendelig prosess. Men det er bare ett kompetansemål på hvert trinn som handler om programmering. Om man bare skal bruke programmering til dette ene kompetansemålet per år, er det ikke tilstrekkelig med tid for elevene å mestre programmering som et effektivt

verktøy. For å bruke nok tid med programmering, blir mange lærere stående med to alternativ. Det ene er å integrere programmering i andre kompetansemål i matematikken, eller legge læreplanen og matematikk til side for en periode og fokusere bare på programmering. Jo lenger i skolegangen man kommer, jo mer komplisert blir programmeringen for elevene.

En av fordelene derimot, ved å innføre programmering i matematikkundervisningen, er at matematikk og programmering deler mange grunnleggende prinsipper. Begge fagfeltene handler om å løse problemer, identifisere mønstre og bruke logikk for å komme frem til løsninger (Kilhamn & Bråting, CERME 11/2019). Matematikken fokuserer på abstrakte konsepter og teorier, mens programmering gir et praktisk rammeverk for å anvende matematisk tenkning i virkelige situasjoner. Ved å kombinere disse to disiplinene kan lærere skape en mer praktisk læring som engasjerer elevene og gjør matematikk mer anvendelig og relevant for dem. Ved hjelp av programmering kan elevene også utforske komplekse matematiske ideer på en mer intuitiv måte. De kan skape interaktive simuleringer og modeller som gjør det mulig å utforske geometriske former, algebraiske relasjoner, numeriske metoder og mer. Dette gir elevene en dypere forståelse av matematiske konsepter ved å gi dem muligheten til å se dem i aksjon og eksperimentere med ulike scenarier.

Integreringen av programmering i matematikkundervisningen representerer et spennende skritt mot å forberede elevene på en fremtid som er sterkt avhengig av teknologi og digitale ferdigheter. Ved å kombinere matematikk og programmering får elevene muligheten til å utvikle viktige ferdigheter og tenkemåter som vil være avgjørende for deres suksess i det 21. århundre. Det gir også en mulighet til å gjøre matematikkundervisningen mer engasjerende, relevant og utforskende. Med riktig støtte og implementering kan programmering i matematikkundervisningen være et verdifullt verktøy for å styrke elevenes matematiske forståelse og forberede dem på fremtidens utfordringer.

1.1 Problemstilling

For å undersøke hvilke deler av matematikkundervisningen på ungdomsskolen man kan integrere programmering i, har jeg formulert denne problemstillingen:

«Hvilke matematiske kompetanser ønsker enkelte faglærere å fremme gjennom programmering i matematikkundervisning på ungdomsskolen?»

For å bedre kunne svare på denne problemstillingen, har jeg kommet med tre forskningsspørsmål.

Forskningsspørsmål 1:

> I hvilken grad brukes programmering som et verktøy for å lære matematikk?

Forskningsspørsmål 2:

> Hvilke holdninger har matematikklærere til om programmering er nyttig i undervisningen på ungdomstrinnet?

Forskningsspørsmål 3:

> Hva slags matematiske kompetanser ser læreren som mest relevante i forhold til programmering?

Disse forskningsspørsmålene knyttes til ulike deler av problemstillingen. Jeg kommer til å drøfte hvordan lærerne bruker programmering i undervisningen, hvilke holdninger og tanker de har rundt det, og hvilke kompetanser fra matematikken de anser som relevante til programmering.

1.2 Disposisjon

Til å begynne, vil jeg se på tidligere forskning gjort på dette fagfeltet i Kapittel 2 - Teori. Der tar jeg for meg hva programmering er og dens relasjon til algoritmisk tenkning. Videre ser jeg nærmere på relasjonen mellom programmering og matematisk tenkning. Som avslutning i det kapitelet, går jeg gjennom rammeverket og de åtte matematiske kompetansene den består av. I Kapittel 3 - Forskningsmetode, tar jeg for meg hvilken forskningsmetode som er mest hensiktsmessig i møte med problemstillingen min. Jeg kommer til å gå gjennom hva kvalitative og kvantitative metoder er, og hvorfor jeg endte opp med den spesifikke forskningsmetoden. Videre hvordan datainnsamlingen foregikk, dens kvalitet og hvilke etiske hensyn man er nødt til å ta i forbindelse med forskning. I Kapittel 4 - Resultat og analyse, presenterer jeg resultatene fra datainnsamlingen min. Jeg tar først for meg programmering og matematisk kompetanse, deretter kjerneelementene, og så kompetansemålene. Resultatene er kategorisert under programmering og matematisk kompetanse, kjerneelementene og kompetansemålene. I tillegg vil jeg presentere interessante innsikter og kommentarer som ikke faller inn under disse kategoriene. I Kapittel 5 - Drøfting, drøfter jeg resultatene, og ser dem i lys av rammeverket og dens matematiske kompetanser. Der tar jeg for meg den enkelte kompetanse, og forsøker å svare på problemstillingen. Kapittel 6 - Avslutning er det siste kapittelet i oppgaven min. Der kommer jeg med en oppsummering av oppgaven og kommentere hva som kan være relevant som videre forskning.

Kapittel 2 - Teori

I dette kapittelet kommer jeg først til å ta for meg hva programmering innebærer og forskjellen mellom begrepene «programmering» og «koding». Deretter vil jeg komme med begrepsavklaring til ulike termer som brukes i forbindelse med programmering og matematikk som «algoritmisk tenkning». Etter begrepsavklaringen skal jeg se hva tidligere forskning har funnet som kobler matematisk tenkning og programmering. Som avslutning på dette kapittelet skal jeg gjøre rede for hvordan man kan kategorisere matematisk kompetanse.

2.1 Hva er programmering?

Forsström og Kaufmann omtaler programmering som «the process related to the development and implementation of instructions for computer programs so the computer can perform specific tasks, solve problems, and support human interactions» (2018, s. 19). Programmering er da ifølge dem, en måte å kommunisere med en datamaskin for å få den til å utføre oppgaver for oss. Videre skriver de at programmering innebærer utvikling av algoritmer og logikk. Det handler om å løse problemer ved bruk av algoritmer. For å kommunisere med og gi kommandoer til en datamaskin er man nødt til å bruke et av programmeringsspråkene. Språkene kan være bygget opp på forskjellige måter, med ulike funksjoner, eller være rettet mot spesifikke bruksområder. Programmeringsspråk er ofte tekst-basert, som for eksempel Python, men det finnes også grafiske språk mer rettet mot barn, som Scratch (Nätt, 2022). Dette blir også ofte omtalt som blokk-programmering, da man koder ved å plassere blokker i et vindu, som gir forskjellig kommando.

2.2 Programmering eller koding?

Ofte bruker man ordene «programmering» og «koding» om hverandre, men det refererer ikke alltid til det samme. Noen ganger blir begrepene behandlet som synonymer (Holo et al., 2022, s. 2), men andre ganger blir koding definert som selve handlingen av å sette sammen programkoder, mens programmering er en prosess som handler som skjer mentalt og fysisk. I et notat fra Senter for IKT i utdanningen kommer det fram at programmering ikke bare er å skrive programkode, men også innebærer selve prosessen, fra å se et problem, komme med løsninger, og drive med feilsøking (Sevik, 2016, s. 9). Det nevnes at begrepet «koding» er blitt vanligere å bruke i møte med barn, og at det gir uttrykk for noe mye mindre farlig enn hva «programmering» hadde gitt. Dette tyder på at måten man ordlegger seg på kan være med å påvirke elevers holdninger til dette temaet. For elever kan «programmering» være noe som er vanskelig og fremmed som de kanskje ikke kommer til å mestre, i motsetning til

«koding» som kan være noe gøy og kjekt de kan leke med i timen. Forsström og Kaufmann (2018, s. 19) hevder at selve implementeringen av algoritmer ofte omtales som koding, mens programmering innebærer hele prosessen fra tanker, logikk, forbedring, verifisering og skriving. De differensierer altså mellom programmering og koding, der programmering omhandler hele den overordnede prosessen, mens koding er underordnet og omhandler selve skriveprosessen. Til forskjell fra koding har programmering derimot et mer kognitivt element ved seg. Denne differensieringen finner vi igjen andre steder, som i tidsskriftet Tangenten. Gjøvik og Torkildsen skriver at:

> «Programmering kan sies å være å kommunisere med logikk. Det kan vi også si er en del av matematikken. Koding, på den andre siden, er selve aktiviteten med å reformulere dette inn i et bestemt programmeringsspråk. Koding er altså mer snevert enn programmering» (2019, s. 34).

Det er ikke sikkert at man alltid er bevisst på hvordan man ordlegger seg når man snakker om programmering eller koding, men det er kanskje ikke så nødvendig heller. Om elever responderer bedre på begrepet «koding» enn på «programmering», så gir det selvfølgelig mening å bruke begrepet «koding». I samtale med kollegaer derimot, kan det noen ganger bli litt upresist om man bruker koding og programmering som synonymer. Det kan skape misforståelser eller usikkerhet i hva man mener med begrepet. Er det selve handlingen, å skrive programkode, eller er det hele prosessen rundt det vedkommende mener? For å være så presis så mulig i prosjektet mitt kommer jeg til å differensiere mellom ordene «programmering» og «koding» på en lignende måte som Forsström og Kaufmann (2018). Koding vil jeg bruke i referanse til selve plasseringen av blokkene eller skrivingen av teksten som brukes for å lage et program. Programmering derimot, vil bli brukt i referanse til hele prosessen, fra å forstå problemet og tenke ut hvordan man løser det, til å skrive instrukser til en datamaskin og rette på feilene som eksisterer i instruksene. Dette vil gjelde både for tekst-programmering og blokk-programmering.

2.3 Algoritmisk tenkning og programmering

Utdanningsdirektoratet definerer algoritmisk tenkning som en problemløsningsmetode. Dette innebærer blant annet å tenke logisk ved å analysere problemer og prøve å forutse hendelser, å tenke algoritmisk og bryte problemet opp i flere mer håndterlige biter, og vurdere om løsningen er god nok, eller om man må gjøre endringer i arbeidet (Utdanningsdirektoratet, 2019a). Dette nevnes også i kjerneelementet om «utforskning og problemløsning» i

matematikk, hvor «Algoritmisk tenkning er viktig i prosessen med å utvikle strategier og framgangsmåter for å løse problemer…» (Utdanningsdirektoratet, 2019b).

2.3.1 Er «algoritmisk tenkning» og «computational thinking» det samme?

Et begrep som er sterkt knyttet opp til programmering, er «computational thinking». Dette er et begrep Wing diskuterer i sin anerkjente artikkel Computational Thinking (2006). Utdanningsdirektoratet oversetter «computational thinking» med «algoritmisk tenkning» som det norske begrepet (Utdanningsdirektoratet, 2019a). Man ser med en gang forskjell i ordlyden, med «computational» versus «algoritmisk», hvor den norske oversettelsen virker snevere i ordets betydning. Wing (2006) bruker begrepet «algorithm» som en del av «computational», men på norsk oversettes «computational» med «algoritme». «Computational» er ikke det samme som «algoritme», selv om de har en del til felles. I tillegg eksisterer det allerede et engelsk begrep som kalles «algorithmic thinking», som er noe annet enn «computational thinking». Hoyles and Noss beskriver «computational thinking» som bestående av fire deler, der «algorithmic thinking» er en av dem (Hoyles & Noss, 2015, sitert i Kilhamn & Bråting, CERME 11/2019, s. 566). De tre andre delene av «computational thinking» er å dekomponere, gjenkjenne mønstre, og abstraksjon. Det er en utfordring at den norske oversettelsen av «computational thinking» ikke virker dekkende nok til å beskrive begrepet, samtidig som at ordlyden ligner mer på et annet engelsk begrep. Dette blir også problematisert av Gjøvik og Torkildsen, som viser til eksempler hvor disse begrepene har blitt brukt om hverandre:

> «Her påstås det altså at algoritmisk tankegang er det samme som Algorithmic Thinking, og at dette er en undergruppe av Computational Thinking. Som vi ser, er det ikke fullstendig overensstemmelse mellom norske og utenlandske termer, og en vil også finne forskjellige definisjoner i forskjellig litteratur» (Gjøvik & Torkildsen, 2019, s. 32).

De finner det også utfordrende å bruke «algoritmisk tenkning» for «computational thinking», samtidig som vi har det engelske begrepet «algorithmic thinking». Deres løsning var å oversette «algorithmic thinking» med «algoritmebehandling» og fortsette bruken av «algoritmisk tenkning» som den norske oversettelsen av «computational thinking» (Gjøvik & Torkildsen, 2019, s. 32-33). Man kan diskutere hvilke begrep og oversettelser som er best egnet for å formidle innholdet i ordet, men på et tidspunkt må man bare ta et valg. I denne oppgaven kommer jeg i likhet med Gjøvik og Torkildsen, til å bruke «algoritmisk tenkning» som den norske oversettelse av «computational thinking» (Gjøvik & Torkildsen, 2019, s. 32).

Hensikten er å ha et språkbruk i oppgaven som er i samsvar med Utdanningsdirektoratet sine dokumenter, og på den måten unngå flere forvirringer (Utdanningsdirektoratet, 2019a).

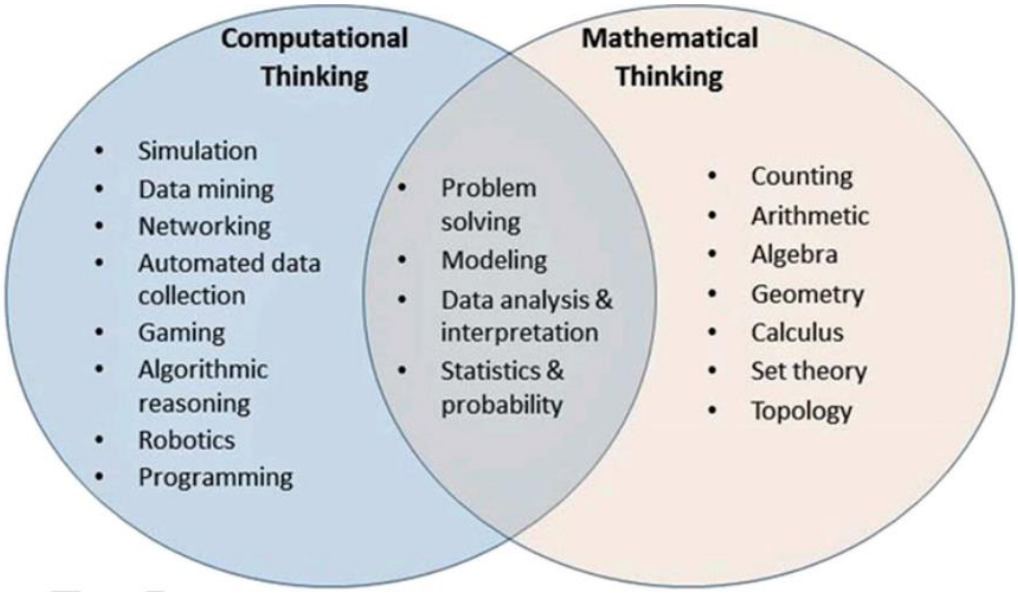
2.3.2 Er det en kobling mellom algoritmisk tenkning og programmering

Ifølge Shute et al. (2017, s. 2) stammer algoritmisk tekning fra Seymour Papert sitt arbeide innen konstruktivisme, men ble først brukt som et utrykk i den tidligere nevnte artikkel fra Wing (2006). Wing (2006) hevder at algoritmisk tenkning er en fundamental måte å tenke på, og burde være like sterkt trukket fram som lesing, skriving og regning. På samme måte som boktrykkerkunsten gjorde det enklere og nødvendig at befolkningen lærte seg å lese og skrive, er datamaskiner med på å spre algoritmisk tenkning ut til befolkningen. Algoritmisk tenkning involverer å løse problemer, designe systemer og forstå menneskelig oppførsel. Det handler om å tenke algoritmisk. Å tenke algoritmisk vil si å tenke systematisk og løse problemer på en bestemt måte (Kaufmann & Stenseth, 2020, s. 2). I artikkelen sin, går Wing gjennom mange måter å forstå algoritmisk tenkning på, og hvordan man bruker det i hverdagen. «Computational thinking is reformulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation» (Wing, 2006, s. 33). Algoritmisk tenkning er sterkt knyttet opp til datainformatikk og programmering, men det blir presisert svært sterkt at hvis man lærer seg å bruke algoritmisk tenkning, så kan det bli brukt i nesten alle sammenhenger. Wing hevder at man kan bruke algoritmisk tenkning i en rekke felt som for eksempel medisin, jus og politikk (Wing, 2006, s. 35). Dette er altså et begrep som er svært innholdsrikt med mange nyanser. I denne artikkelen kommer det fram at algoritmisk tenking bygger på evnene og grensene til maskiner. Det ligger en spenning mellom hva maskiner kan gjøre bedre enn mennesker, og hva mennesker kan gjøre bedre enn maskiner. Wing stiller spørsmålet om hva som er beregnbart, og skriver at vi bare vet deler av dette svaret i dag. Men algoritmisk tenkning er en fundamental egenskap som alle burde lære seg (Wing, 2006, s. 33). Det er altså ikke bundet opp i bruk av datamaskiner. Man kan også bruke algoritmisk tenkning uten datamaskiner. Det handler om å løse problemer på en rekursiv måte, å hele tiden gå tilbake og forbedre eller løse flere og flere deler av hele problemet. Man dekomponerer et større problem inn i mer håndterbare delproblemer som kan løses hver for seg. Videre kommer det fram at algoritmisk tenkning er en tenkemåte som kommer fra oss mennesker, og at det innebærer matematisk tenkning (Wing, 2006, s. 35). Her blir det koblet opp algoritmisk tenkning og matematikk, men ikke noe videre utdyping på noen annen måte enn at det er en del av de vitenskapelige studiene som har matematikk som et fundament.

Algoritmisk tenkning stammer fra programmering, men er ikke det samme som det. En grunnleggende forskjell er at algoritmisk tenkning kan tas i bruk i andre omstendigheter og situasjoner som ikke er programmering. Men samtidig blir man bedre til å programmere ved å kunne bruke algoritmisk tenkning (Shute et al., 2017, s. 5). I et studie fra Shute i 1991, konkluderes det at algoritmisk tenkning, programmering og problemløsning er sterkt knyttet til hverandre (Shute et al., 2017, s. 5).

2.4 Programmering og matematisk tenkning

Shute et al. (2017, s. 4) differensierer mellom ulike tenkemåter, deriblant algoritmisk og matematisk tenkning. Det blir presentert et venn-diagram som vi ser på Figur 1. Der ser vi hva som er fellesnevnere og hva som er ulikhetene mellom disse to tenkemåtene. Eksempler på ulikheter er at algoritmisk tenkning innebærer simulering og programmering, mens matematisk tenkning dreier seg om aritmetikk og algebra. Blant elementene som er til felles er den viktigste muligens problemløsning. I kjerneelementet «utforskning og problemløsning skriver Utdanningsdirektoratet at «Algoritmisk tenkning er viktig i prosessen med å utvikle strategier og framgangsmåter for å løse problemer …» (Utdanningsdirektoratet, 2019b). De har også sett på algoritmisk tenkning som en bra måte å drive problemløsning på. Videre viser figuren modellering, statistikk og sannsynlighet og analysering og tolkning. Med dette kan vi da bruke programmering og algoritmisk tenkning i matematikkundervisningen for å styrke problemløsning blant elever.



Figur 1 - CT og MT fra Shute et al (2017, s. 4)

Å innføre programmering i skolen er ikke noe nytt. Allerede i 1980 utviklet Seymour Papert programmeringsspråket «Logo» i håp om å bruke dette til motivasjon og læring av matematikk (Forsström & Kaufmann, s. 19-20). Ut fra dette prosjektet har det i senere tid blitt utviklet mange ulike språk som brukes særlig i skolen, deriblant «Scratch» og «Lego Mindstorms». I 1995 undersøkte Yelland om Logo kunne bli brukt i matematikk-undervisningen i forbindelse med problemløsning og samarbeid om matematiske problemer. Resultatet av undersøkelsen viste at i noen studier kunne programmering være med å styrke eleven i matematikk, mens i andre studier ble det ikke oppdaget noen forskjell i elevens matematiske oppnåelser eller problemløsnings-evner (Forsström & Kaufmann, 2018, s. 20). Det var da vanskelig å kunne konkludere at programmering var et godt middel å bruke for å lære bort matematikk på den tid. Ett annet poeng de fremmer, er at det ikke er så mange studier som studerer akkurat denne sammenhengen isolert, og at de viser delvis positive resultater innen deler av matematikken eller i spesielle klasser eller for spesielle typer elever, men at det er lite generaliserbart, og mer forskning trengs.

Siden den gang har Kaufmann og Stenseth (2020) studert hvordan programmering kan være med å bidra til matematikkundervisning. De ser på noen elever som skal samarbeide om en matematikkoppgave i programmering, og analyserer hvilke argumentasjon og problemløsningsstrategier de bruker i møte med utfordringen. De viste at elevene var i stand til å en fremdrift i sin argumentasjon tilsvarende Lavys fire argumentasjons kategorier. Samtidig tar de fram et relevant dilemma. Lærer man matematikk gjennom programmeringen, eller er det problemløsning i programmering man lærer gjennom matematikken? Dette dilemmaet kan fort komme opp i en undervisningssammenheng, hvor man har kompetansemål elevene skal kunne og kjerneelement man skal fremme. Er det programmering elevene lærer nå, eller er det matematikk? Kaufmann og Stenseth hevder at oppgaven de designet var med å fremme begge deler, og konkluderer at dette doble perspektivet er nødvendig for lærere å ta med seg (Kaufmann & Stenseth, 2020, s. 17).

Kilhamn og Bråting (2019) tar for seg overlappingen mellom algebraisk tenking og algoritmisk tenkning, og hvordan man kunne utvikle algebraisk tenkning gjennom programmering. Begge disse tenkemåtene har symbolbruk, struktur og generalisering, derfor kan man bruke programmering for å lære elever å tenke algebraisk. Både algebra og koding bruker variabler og likhetstegn i stor grad, men syntaksen og måten de blir brukt på er forskjellig. Innenfor algebraisk tenkning vil variabler alltid representere et tall, men i programmering kan variabler representere mer enn tall. Variabler kan representere ord og

setninger, eller strenger som det kalles, binære kategorier som "true/false" eller en liste, men mange elementer. Konseptet om hva en variabel kan være, er altså mye videre enn i algebraisk tenkning, som alltid bruker bokstaver til å representere tall (Kilhamn & Bråting, CERME 11/2019, s. 571). Det blir trukket fram at slike forskjeller kan føre til forvirring som gjør det vanskelig for elever å lære. Det er da viktig at læreren er klar over slike forskjeller, og bruker dem til å skape kontraster og viktigheten av nøyaktighet i arbeidet overfor elevene sine (Kilhamn & Bråting, CERME 11/2019, s. 571). Å angripe et problem fra to forskjellige vinkler eller å bruke to ulike strukturer i arbeidet med det, kan også være med å utvikle sin matematiske forståelse og tekning, men bare hvis dette blir lagt merke til og diskutert opp mot hverandre (Kilhamn & Bråting, CERME 11/2019, s. 572).

Calder (2018) undersøkte hvordan Scratch kunne bli brukt for å utvikle matematisk tenking. Studiet gikk over to uker hvor elevene arbeidet i par og skrev daglige blogger hvor de reflekterte over sin egen progresjon. Alle elevene hadde tilgang til en egen datamaskin, og skulle i løpet av den første uken bli kjent med å arbeide i «Scratch», ettersom dette var noe nytt for dem. Den andre uken skulle parene planlegge og gjennomføre sitt eget prosjekt. Alle parene var nødt til å arbeide ut ifra den samme oppgaven, som var å lage et matematikk-spill for 1. klassinger. Etter å ha analysert de innsamlede dataene, fant han at elevene var i stand til å bruke matematisk tenkning og problemløsning til å håndtere programmeringsoppgaven. De hadde utviklet sin matematiske tenkning innenfor problemløsing og geometri. Et høyere nivå av problemløsning etter studiet, kom fra elevenes behov for å prøve, feile og forbedre produktet sitt flere ganger. Flere ganger møtte elevene utfordringer hvor de måtte diskutere sammen og prøve ut ulike fremgangsmetoder for å løse problemet. Deres geometriske forståelse økte gjennom bruken av figurer i «Scratch». De er plassert i et koordinatsystem der man beveger dem ved å enten definere deres y- og x- koordinater, eller å addere og subtrahere fra verdien til koordinatene. Dette var med på å gi dem en bedre forståelse over geometriske konsepter som sted, rom og distanse. Programmering var et godt egnet instrument til å fostre matematiske samtaler og diskusjoner, samt se hvordan vinkler, posisjoner og rom-forståelse hang sammen (Calder, 2018, s. 55-56).

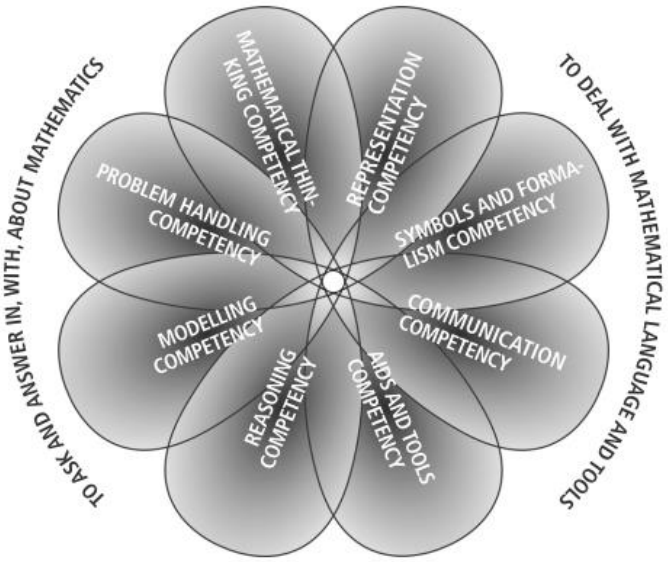
Som vi har sett, kan programmering være med å bidra til å øke matematisk tenkning blant elever. Forsström og Kaufmann (2018) viser til problemløsning, modellering, statistikk og sannsynlighet. Kaufmann og Stenseth (2020) kommer fram til at programmering kan hjelpe elever til å argumentere innenfor matematikken, samtidig som de belyser et aktuelt dilemma; Hvordan kan man vite om det er matematikken som fremmer problemløsning i

programmering, eller om det er programmering som øker forståelsen i matematikken? De hevder at man kan fremme begge disse ferdighetene samtidig med samme oppgave. Kilhamn og Bråting (2019) viser til sammenhengen mellom algebra og programmering, og at elver kan øke sin kompetanse innenfor algebra, selv om det ikke er en én til én korrelasjon. Calder kobler programmering i Scratch til økt kompetanse innenfor samtale og problemløsning i matematikk, samt geometri og koordinatsystem. Det er verdt å legge merke til at flere av studiene legger opp til mye samarbeid mellom elevene når de bruker programmering i matematikk. Dette kan tyde på at mye av verdien i å inkorporere programmering i matematikkundervisning, ligger i selve samarbeidet og resonneringen mellom elevene.

2.5 Kompetanseblomsten som rammeverk for matematisk kompetanse

For å kunne svare på problemstillingen min; «hvilke matematiske kompetanser ønsker enkelte faglærere å fremme gjennom programmering i matematikkundervisning på ungdomsskolen?», skal jeg bruke et rammeverk som beskriver de ulike matematiske kompetansene som eksisterer. Rammeverket jeg skal bruke i denne oppgaven er Niss og Højgaard (2002) sin kompetanseblomst. De skrev sin originale rapport i 2002, og en artikkel som oppfølgning i 2019. Disse to artiklene har mye av det samme innholdet, men Niss og Højgaard (2019) har oppdatert noe av terminologien, samt prøvd å være tydeligere og gi bedre forklaringer enn i den originale artiklene.

Niss og Højgaard prøver å sammenfatte hva det innebærer å ha kompetanse innenfor matematikk. Dette handler ikke om de ulike emnene eller temaene innenfor matematikk, slik som geometri og algebra, men det overordnende som er felles for all matematikk. De ulike temaene innenfor matematikk kan være svært forskjellige og de kan arbeides med på helt ulike nivå. Målet deres er å identifisere ulike komponenter av matematikk som alltid er til stede, uansett tema eller nivå (Niss & Højgaard, 2019, s. 10-11). De utviklet åtte matematiske kompetanser, fordelt på to forskjellige kategorier. Hver av disse kategoriene består av fire matematiske kompetanser, som til sammen utgjør de åtte kompetansene i kompetanseblomsten illustrert i Figur 2.



Figur 2 Representasjon av de åtte matematiske kompetansene [Hentet fra Niss & Højgaard, 2019, s. 19]

Som figuren viser, overlapper noen av disse kompetansene hverandre (Niss & Højgaard, 2019, s. 19). Hver av kompetansene er forbundet med de sju andre kompetansene, men har fremdeles sin egen identitet. Ingen av kompetansene kan oppsummeres av en annen kompetanse (Niss & Højgaard, 2002, s. 44). På samme måte er oppdelingen av kompetansene i to kategorier ikke et forsøk på å skille noen av kompetansene fra hverandre. Hver av kompetansene bidrar til begge kategoriene, på tross av at de ligger under én kategori (Niss & Højgaard, 2002, s. 44-46).

2.5.1 Kategori 1 – Spørsmål i og med matematikk

Den første kategorien handler om å «stille og svare på spørsmål i og med matematikk» (Niss & Højgaard, 2019, s. 15). Dette innebærer også å bedømme og forstå, samt å argumentere i matematiske spørsmål.

Den første kompetansen vi finner i denne kategorien er *«Matematisk tankegangskompetanse»*. I denne kompetansen inngår det å stille generelle spørsmål som passer i matematikken, og deriblant ta i bruk matematiske konsepter og begreper som definisjon, fomler, logikk og hvis-så påstander (Niss & Højgaard, 2019, s. 15). Gjennom spørsmål og svar i den matematiske konteksten, og bruk av begrepene, øker også muligheten til å generalisere. Tankegangskompetansen tar også for seg matematiske begrep sine begrensninger og innhold (Niss & Højgaard, 2002, s. 47).

Den neste kompetansen man trenger for å stille og svare på spørsmål i og med matematikk er *«matematisk problemløsningskompetanse»*. Denne kompetansen går ut på å identifisere, avgrense, formulere og løse ulike matematiske problemer (Niss & Højgaard, 2019, s. 15). Disse problemene kan finne seg i alle de ulike matematiske domenene. Men hva som er et matematisk problem, kommer an på hvem som skal løse den. Det som for en person er noe dagligdags og har blitt gjort mangfoldige ganger før, kan være et svært hodebry for en annen. På grunn av ulike erfaringer og nivåer til individer, må man definere hva som er et matematisk problem ut ifra hvem som skal løse det. Det inngår også å kunne analysere både sine egne og andres løsninger på problemene. Ikke alle spørsmål man stiller i matematikken er et problem, men det kan ta for seg andre aspekter av matematikken. For eksempel om man spør hva det betyr det når det står «0» i tallet «406», så går dette under begrepsforståelse, og ikke et matematisk problem. Fordi dette er ikke noe man må undersøke for å finne ut, men noe man må lære hva betyr (Niss & Højgaard, 2002, s. 50)

Videre har vi *«matematisk modelleringskompetanse»*, som går ut på å håndtere situasjoner utenfor matematikken. Dette gjør man ved å lage modeller hvor man kan bruker matematikk i ulike kontekster. Disse modellene kan bli analysert gjennom data, fakta og egenskaper (Niss & Højgaard, 2019, s. 16). Man kan gjerne se på dette som en praktisk bruk eller matematikk i hverdagen. Man tar situasjoner fra omverdenen, og representerer dette på en matematisk måte. For å bygge modeller av situasjoner utenfor matematikken, er man nødt til å oversette problemstillingen til et område av matematikken, slik at man ender opp med en matematisk modell (Niss & Højgaard, 2002, s. 52).

Den siste kompetansen i denne første kategorien, heter *«matematisk resonneringskompetanse»*. Kjernen av denne kompetansen går ut på å argumentere og begrunne svaret til matematiske påstander. Denne kommunikasjonen kan skje både i skriftlig eller muntlig form (Niss & Højgaard, 2019, s. 16). Resonneringskompetansen kommer til syne i mange former, blant annet å kunne rettferdiggjør matematiske setninger, og gjengi matematiske bevis. Men det er mye mer som inngår i denne kompetansen. Man kan også begrunne svar med mot-eksempler, logisk deduksjon fra aksiomer, og skille mellom hovedpunktene og mindre detaljer. Siden denne kompetansen innebærer å rettferdiggjøre svar og løsninger, knytter Niss og Højgaard problemløsningskompetansen og modelleringskompetansen sterkt opp til resonneringskompetansen (Niss & Højgaard, 2002, s. 54).

2.5.2 Kategori 2 – Språk, konstruksjoner og verktøy i matematikk

Kategori to tar for seg språket og redskapene til matematikk. Den første kompetansen heter *«matematisk representasjonskompetanse»* og går ut på å tolke og bevege seg mellom ulike former for representasjoner. Disse representasjonsformene kan være blant annet diagrammer, symboler og verbal. Et viktig aspekt ved denne kompetansen er å forstå styrkene og svakhetene til de ulike representasjonene. Dette er med på å velge det som er mest hensiktsmessig i ulike situasjoner og kontekster (Niss & Højgaard, 2019, s. 17). Selv om man kan representere noe i ulike former, er det viktig å vite at alle former ikke inneholder samme informasjon. Ved å gå fra en representasjonsform til en annen kan noe informasjon gå tapt. Samtidig kan ny informasjon komme fram ved et slik skifte. Representasjonskompetansen er også sterkt knyttet til to andre kompetanser. Symbolsk bruk i matematikken er muligens den viktigste formen for representasjon, som forbinder denne kompetansen opp mot den neste kompetansen, matematisk symbol- og formaliseringskompetanse. (Niss & Højgaard, 2002, s. 57).

Å kunne håndtere symboler i matematikken går inn under neste kompetanse, som er *«matematisk symbol- og formaliseringskompetanse»*. Symboler blir brukt hele tiden og har en tydelig definisjon på hva det de betyr for å kunne effektivt kommunisere matematikk. Man viser kompetanse ved å kunne håndtere disse symbolene i matematikken, samt oversette og se sammenhenger mellom det matematiske symbolspråket og et hverdagslig språk (Niss & Højgaard, 2019, s. 17). Denne kompetansen innebærer også det mer formelle og regler man har innenfor matematikken. For eksempel at $5 \cdot (3 + 4)$ ikke er det samme som $5 \cdot 3 + 4$. Forskjellen på representasjonskompetansen og denne kompetansen, er at her fokuseres det spesifikt på symboler, betydningen av dem, og reglene som inngår i systemet (Niss & Højgaard, 2002, s. 59).

Den neste kompetansen, *«matematisk kommunikasjonskompetanse»* handler om å formidle og kommunisere matematiske ideer. Denne formidlingen skjer på en rekke forskjellige måter, deriblant skriftlig, muntlig og visuelt (Niss & Højgaard, 2019, s. 17-18). For å kunne kommunisere matematiske ideer på en best mulig måte, er det viktig at man er i stand til å variere språket sitt og legge seg på ulike nivåer, tilpasset til den man kommuniserer med. Denne kommunikasjonen går begge veier, altså ikke bare at man er i stand til å formidle til andre, men også er i stand til å forstå andre sin formidling. Denne kompetansen er også sterkt forbundet med representasjonskompetasen og symbol- og formaliseringskompetansen, da alle

former for kommunikasjon i og med matematikk, foregår gjennom en eller annen form for representasjon, som oftest i form av symboler (Niss & Højgaard, 2002, s. 60).

I matematikken bruker man ulike hjelpemidler for å gjøre ting lettere, blant annet kalkulator, fysiske gjenstander, diagrammer og programmering. Her kommer den siste kompetansen inn, som er *«matematisk hjelpemiddelskompetanse»*. Denne kompetansen går ut på å kunne bruke slike hjelpemidler på en hensiktsmessig måte, og kunne se fordelene og ulempene knyttet til de ulike hjelpemidlene (Niss & Højgaard, 2019, s. 18). Ettersom et hjelpemiddel tar i bruk en eller annen form for representasjon, er det en kobling mellom disse to kompetansene. Og siden ulike hjelpemidler ofte er begrenset av sine egne regler og egenskaper, er hjelpemiddelskompetansen også i slekt med symbol- og formaliseringskompetansen (Niss & Højgaard, 2002, s. 62).

2.5.3 Kompetanseblomsten og kjerneelementene

I en rapport fra en ekstern arbeidsgruppe oppnevnt av Utdanningsdirektoratet, kom det fram at rammeverket for grunnleggende ferdigheter i regning, bygges på blant annet Niss og Højgaard sin kompetanseblomst (Borge et al., 2014, s. 74-75). Derfor har jeg laget en oversikt som sammenligner kjerneelementene som utdanningsdirektoratet bruker med kompetansene til Niss og Højgaard, vist i Tabell 1.

Tabell 1 - Matematiske kompetanser og kjerneelement

Kjerneelement → Kompetanser ↓	Utforskning og problemløsning	Modellering og anvendelser	Resonnering og argumentasjon	Represetasjon og kommunikasjon	Abstraksjon og generalisering	Matematiske kunnskapsområder
Kategori 1						
Tankegangs-kompetansen					■	
Problemløsnings-kompetansen	■					
Modellerings-kompetansen		■				
Resonnerings-kompetansen			■			
Kategori 2						
Representasjons-kompetansen				■		
Symbol- og formalisme-				■	■	
Kommunikasjons-kompetanse				■		
Hjelpemiddels-kompetanse	■					
Antall kompetanser per kjerneelement	2	1	1	3	2	

Som tabellen viser, er ikke kjerneelementet om matematiske kunnskapsområder forbundet med noen spesielle kompetanser. Det er ikke fordi ingen kompetanser er forbundet med det, men fordi dette kjerneelementet skiller seg noe ut fra de andre. Læreplanene ordlegger seg

slikt om det: «Dette kjerneelementet har ingen koblinger til kompetansemålene fordi alle kompetansemålene handler om ett eller flere av de matematiske kunnskapsområda» (Utdanningsdirektoratet, 2019b). Dette er et kjerneelement som inngår i alle kompetansemål, som gjør det noe unødvendig å ta det med. Da sitter vi igjen med fem kjerneelementer som skal fordeles på åtte matematiske kompetanser. Dette medfører til at det i noen tilfeller er flere av kompetansene som passer inn i et kjerneelement, eller at én kompetanse kommer til syne i flere av kjerneelementene. Jeg har koblet kjerneelementene med kompetansene ved å sammenligne språkbruk og nøkkelord. Noen koblinger er naturlige å finne, som problemløsningskompetansen og kjerneelementet «utforskning og problemløsnings». Andre kan være mindre opplagte, som for eksempel symbol- og formalismekompetansen og kjerneelementet «representasjon og kommunikasjon». Disse koblingene er ikke én til én sammenligning, men innholdet omtaler store deler av de samme elementene til hverandre. Kjerneelementene nevner aldri noe om hjelpemidler, og begrepet «verktøy» blir bare tatt opp én gang under «utforskning og problemløsning» (Utdanningsdirektoratet, 2019b). Selv om dette ikke tar stor plass i dokumentene, som blir tatt mye i bruk i undervisningen, og er nesten automatisk inkludert da prosjektet handler om programmering i matematikkundervisning.

Kapittel 3 - Forskningsmetode

For å samle inn data som er nødvendig i besvarelsen av mine forskningsspørsmål, er jeg nødt til å overveie fordelene og ulempene ved de ulike metodene, og velge den metoden som er mest hensiktsmessig og fordelaktig. I dette kapittelet kommer jeg derfor til å begrunne og gjøre rede for mitt metodiske valg for datainnsamling. Først vil jeg ta for meg kvalitativ og kvantitativ metode og drøfte fordelene og ulempene ved hver av dem i møte med problemstillingen min. Deretter kommer jeg nærmere inn på den spesifikke forskningsmetoden jeg skal bruke i oppgaven min og dens styrker og svakheter.

3.1 Hvilken metode skal man bruke?

Det har lenge vært en diskusjon rundt hvilken vitenskapelig metode og hvilke data som egner seg best til å beskrive det sosiale og menneskelige. Noe av denne diskusjonen går blant annet ut på hva som kan kategoriseres som vitenskapelig data (Postholm & Jacobsen, 2018, s. 90). Ifølge Postholm og Jacobsen (2018, s. 99-100) er det ikke en stor konflikt mellom de kvalitative og kvantitative tilnærmingene, men de omhandler ulike typer informasjon, hvor begge kan være nyttige i ulike situasjoner. En kvantitativ metode vil søke å kvantifisere, sette tall på, det en undersøker. For eksempel antall elever i en klasse som synes matematikk er gøy, eller terningkast på en film, mens en kvalitativ metode vektlegger ordene som blir sagt. Kvalitative metoder måler mer subjektive erfaringer. Om en velger det ene eller det andre, kommer an på problemstillingen man har valgt (Postholm & Jacobsen, 2018, s. 101). I noen problemstillinger vil en kvalitativ metode egne seg best til å samle inn data, men andre problemstillinger vil det være mer naturlig å bruke en kvantitativ metode.

3.2 Den kvalitative forskningsmetoden

Det er flere ulike kvalitative metoder som kan brukes, blant annet observasjon, intervju, casestudier og mange flere. Den kvalitative metoden er ofte forbundet med et konstruktivistisk virkelighetsperspektiv (Postholm & Jacobsen, 2018, s. 113). Det som er felles for alle disse forskningsmetodene, er deres fokus på ord, oppførsel og meninger, altså kvalitativ informasjon. Basert på min problemstilling, er jeg mest interessert i enkelte personer og deres erfaringer og tanker rundt matematisk kompetanse og dens tilknytning til programmering. Kvale og Brinkmann (2015, s. 20) skriver at et kvalitativt forskningsintervju har som mål å forstå seg på enkeltpersoners syn og opplevelser av verden på. Dette er noe som er vanskelig å fange opp gjennom tall og rangeringer, som ofte kan miste en del nyanser i svarene sine (Postholm & Jacobsen, 2018, s. 100). En av utfordringene ved en kvalitativ tilnærming er det begrensede utvalget. Ved å bruke intervju som metode må man holde seg til

et visst antall deltakere og kan risikere å få en snever innsikt i temaet når det ikke samles informasjon fra tilstrekkelig antall personer. For å få gode resultater må man bruke mye tid på hver enkelt deltaker, både til selve intervjuet, men også til kommunikasjonen før og i etterkant, samt transkriberingen av samtalen. Hvis man for eksempel heller går for en spørreundersøkelse, kan man få inn hundrevis av svar som kan være med på å gi et bedre bilde av lærere generelt. Men det vil da være vanskeligere for å plukke opp nyanser eller gå i dybden (Postholm & Jacobsen, 2018, s. 165).

3.3 Den kvantitative forskningsmetoden

Tanken bak den kvantitative metoden er at informasjonen som blir samlet inn, skal bli standardisert (Postholm & Jacobsen, 2018, s. 166). Gjennom å samle inn masse data om en spesifikk ting fra en rekke forskjellige personer, skal man kunne ende opp med et representativt bilde fra en gruppe. Dette kunne for eksempel ha vært av matematikk-lærere i Norge og deres holdninger rundt programmering i faget. Siden denne type metoden vil være mer fokusert, er man er i stand til å ha et større utvalg enn en kvalitativ metode. (Postholm & Jacobsen, 2018, s. 165). Ulempen ved å gjennomføre en spørreundersøkelse, er at man blir mer fastlåst. Siden den blir sendt til så mange ulike personer uten direkte kommunikasjon med dem, er det ikke alle typer spørsmål som vil være mulig å stille dem. Hvert eneste ord må nøye vektlegges, og hvert spørsmål må spisses. Man vil aldri kunne utelukke misforståelser helt, og personer vil kunne tolke spørsmålene på ulike måter. Men ved å utforme spørsmålene så tydelig som mulig og ved å ha et stort nok utvalg, reduserer man risikoen for misvisende svar (Postholm & Jacobsen, 2018, s. 167). Som en konsekvens av dette, blir informasjonen som man samler inn veldig spesifikk. Man får generell informasjon fra gruppen, men individene forsvinner blant all dataen og får ikke mulighet til å stikke seg litt ut. All dataen som skal samles inn, er nødt til å være forhåndsdefinert, og alt utenom denne dataen forsvinner. Det er heller ikke mulig å stille oppfølgingsspørsmål eller tolke kroppsspråk og stemmeleie, som kan være utfyllende informasjon (Postholm & Jacobsen, 2018, s. 166).

3.4 Metodevalg for denne oppgaven

Basert på fordelene og ulempene til de ulike forskingsmetodene, har jeg konkludert med at et intervju vil være mest hensiktsmessig for mitt prosjekt. Gjennom min problemstilling, ønsker jeg å fremheve individets erfaringer og tanker, selv om det kunne vært interessant å se på matematikk-lærere generelt. Informasjonen blir ikke så snever, og jeg får mulighet til å innhente data som ikke passer inn i en forhåndsdefinert kategori. Deltakerne får mulighet til å

ytre sine personlige erfaringer på en måte som ikke hadde vært mulig i en spørreundersøkelse. Gjennom et intervju kan jeg som forsker komme med oppfølgingsspørsmål for videre utdypning, eller få oppklaring i det som tidligere har blitt sagt. Det kan hende at lærerne ikke har tenkt gjennom eller tatt en stilling til noen av spørsmålene jeg kommer til å spørre om, som gjør det ekstra gunstig for dem at jeg også er tilgjengelig til å formulere spørsmålet på en annen mer forståelig måte. Derfor mener jeg at et kvalitativt forskningsintervju vil være den mest hensiktsmessige forskningsmetoden for å innhente data til prosjektet mitt.

3.5 Intervju som metode

3.5.1 Ulike intervjuformer

Et intervju kan legges opp på ulike måter. Strukturerte intervju har en svært lukket form. Man følger intervjuguiden ufravikelig, og respondenten har ikke mulighet til å påvirke intervjuet. Forskeren har en så nøytral rolle som mulig gjennom å alltid stille spørsmålene i samme rekkefølge, og alltid har samme ordlyden til alle respondentene (Postholm & Jacobsen, 2018, s. 120). Derimot har et ustrukturerte intervjuet en veldig åpen form, hvor ingen spørsmål er forberedt på forhånd. Denne intervjuformen er ofte forbundet med eller kombinert med observasjon. Lærere kan komme bort til forskeren under observasjonen og fortelle på eget initiativ om for eksempel undervisningen, mens forskeren observerer klasserommet (Postholm & Jacobsen, 2018, s. 120-121). På denne måten kan man få et mer utfyllende bilde og kan sammenligne ord med handling. Et semistrukturert intervju er en slags blanding, eller middelvei, mellom disse to ytterpunktene. Interaksjonen mellom partene vil ligne en samtale fra dagliglivet, men med et tydelig mål eller tema for samtalen (Kvale & Brinkmann, 2015, s. 46). Rammene for samtalen er satt gjennom intervjuguiden, som opptrer som en slags rettesnor. Intervjuguiden inneholder spørsmålene for temaet, men rekkefølgen på spørsmålene må ikke følges slavisk. Man kan starte samtalen ett sted, og komme med spørsmålene der hvor det er naturlig. Denne mer naturlige samtalen rundt et bestemt tema gjør det mulig for forskeren å «grave» dypere hvis den intervjuede bringer opp aspekter ved temaet som ikke er blitt tatt høyde for i intervjuguiden (Postholm & Jacobsen, 2018, s. 121). Dette virker å være den mest egnete intervjuformen for å hente fram erfaringer og tanker fra et individ, samtidig som man passer på å holde seg på et spesifikt tema. Et strukturert intervju kan fort bli litt fastlåst, mens i et åpent intervju kan det være vanskelig å holde seg til et bestemt tema. Det er da semistrukturert intervju som jeg anser egner seg best til mitt forskningsprosjekt, ettersom man kan samle informasjon om noe spesifikt, samtidig som man

er åpen for andre spørsmål og digresjoner om det virker interessant. Arbeidet med intervjuguiden startet allerede våren 2022, og har siden den gang blitt revidert og arbeidet med fram mot våren 2023. Utviklingen av intervjuguiden skjedde ettersom jeg las og bearbeidet tidligere forskning, og i samtale med veilederen min. Spørsmål ble lagt til, fjernet og omformulert med tanke på å spisse dem mer og mer inn mot problemstillingen. Slik tok intervjuguiden form over tid, og ble mer og mer tilpasset studiens formål.

3.5.2 Utvalg

Hvor mange personer som burde bli intervjuet er avhengig av formålet med undersøkelsen. I noen tilfeller kan være nok med så lite som én person, mens i andre tilfeller kan det være nødvendig med 1000 personer, om ikke mer (Kvale & Brinkmann, 2015, s. 148). I arbeid med intervju, er det en balanse mellom antall deltakere, og kvaliteten på analysen av intervjuene. Generelt anbefales det minimum tre personer. Noe mindre enn dette vil generelt påvirke reliabiliteten til studiet for mye. Mer enn 25 kan føre til en for stor arbeidsmengde, og kvaliteten på analysen vil minke (Postholm & Jacobsen, 2018, s. 118). For mitt formål virker det som 3-4 personer er en passelig mengde. Det gir en god balanse på nok datapunkt for å kunne sammenligne, uten at arbeidsmengden påvirker kvaliteten på studiet.

Postholm og Jacobsen (2018, s. 118) skriver i boken sin, at deltakerne burde være valgt ut fra de samme kriteriene og ha erfaringer fra samme type situasjon. Hvem man velger som informanter vil kunne påvirke resultatet på ulike måter. Ungdomsskoleelever, lærerutdannere eller skoleledere vil alle gi ulike perspektiv og innsikt enn det matematikklærerne kommer med. Mitt ønske er å høre hvilke kompetanser lærerne mener programmering kan være med å utvikle.

Et av kriteriene mine er at intervjupersonen har erfaring med å undervise i klasserommet. Jeg har ikke satt ikke noen grense på hvor lenge de hadde jobbet som lærere. Gjennom å ha opparbeidet seg erfaring, vil de kunne ha flere innsikter og tanker å komme med. Dette er noe de har erfart i møte med elever og klasser, og har noe mer å tilføye enn hva nyutdannede lærere har. Det er også naturlig at et av kriteriene er at de aktivt bruker programmering i undervisningen og er komfortable med det. Dette førte til at lærerne enten hadde utdanning innen programmering, eller underviste i valgfaget programmering. Selv om det ikke var et av kriteriene, førte det til at læreren kunne snakke om programmering i andre kontekster enn i matematikk-undervisningen. Forhåpentligvis fører dette til at matematikk-lærerne kan dele mer hvordan de oppfatter relasjonen mellom matematikk og programmering, og hvilke

situasjoner de mener det går an å bruke programmering, og hvilke kompetansemål man kan lære gjennom det.

En svakhet med utvalget mitt er at datamaterialet kun vil reflektere erfaringer fra lærere som allerede er positive til programmering. Dette kan føre til en mye mer positiv opplevelse av inkluderingen av programmering i matematikkundervisningen. Det kan være flere utfordringer som ikke blir belyst ut fra dette utvalget, som kunne ha kommet fram hvis utvalget inkluderte matematikklærere som ikke hadde erfaring innen programmering fra før av.

3.6 Gjennomføring av intervjuene

Intervjuene ble gjennomført i februar 2023 på den enkelte lærer sin skole. Jeg bestemte meg raskt for at det beste var å gjennomføre intervjuene fysisk der hvor de selv jobbet. Det var et alternativ å ha det gjennom et videomøte, men i frykt for å miste en del av kroppsspråket og en «ekte» interaksjon, ønsket jeg å gjennomføre dem fysisk. Kroppsspråket kan fortelle vel så mye som ordene i en samtale. Videomøter kan i noen tilfeller være svært praktisk, fordi den geografiske hindringer forsvinner, men for meg var det en fordel å kunne treffe personen ansikt til ansikt. Grunnen til at intervjuet ble holdt på den enkeltes arbeidsplass, var for at vedkommende skulle føle seg trygg og på «hjemmebane». I tillegg var det mer praktisk rent logistisk sett for intervjuobjektene. Ved at jeg møtte opp på avtalt tid, kunne vi gjennomføre intervjuet i en av fri-timene til læreren i deres arbeidstid. Da kunne vedkommende slippe å bruke tid på reise.

Lærerne fikk beskjed på forhånd om temaet til intervjuet, og mulighet til å se over kompetansemålene i matematikk på ungdomsskolen, ettersom store deler av intervjuet omhandlet dem. Jeg hadde med meg utskrifter av samtykkeskjema til prosjektet som vedkommende måtte lese og skrive under på før vi begynte intervjuet. Dette har blitt lagt til som Vedlegg 1 – Informasjonsskriv og samtykkeerklæring. Skrivet beskriver studiets formål og hvilke rettigheter deltakere har i forbindelse med personvern. Underveis tok jeg fram utskrifter av kompetansemålene, da det er lettere å snakke om og referere til dem når man har dem fysisk foran seg. Selv om de fikk mulighet til å gjøre seg noen tanker før intervjuet, er det lett å glemme, eller komme på noe i ettertid som man ikke nevnte under selve intervjuet. I tillegg hadde jeg med meg utskrift av en definisjon av algoritmisk tenkning. Jeg hadde ikke nevnt noe om algoritmisk tenkning i forkant av intervjuet, da det hadde vært interessant å se hva slags forhold de hadde til dette begrepet uten å kunne lese seg opp på det. I tilfelle de

ikke hadde noen koblinger eller tanker rundt selve begrepet «algoritmisk tenkning», tok jeg med meg utskrift av en definisjon, slik at de kunne snakke litt rundt det uten å nødvendigvis ha et bevist forhold til det. Dette viste seg til å være svært nyttig i noen av tilfellene, mens andre ganger tok læreren det opp selv uten at jeg nevnte noe.

3.6.1 Datainnsamling

Det er ulike måter å samle inn dataen fra et intervju, blant annet videoopptak, notater, lydopptak og hukommelsen. Kvale og Birkmann (Kvale & Brinkmann, 2015) skriver at lydopptak blir sett på som den vanligste måten. Det er også dette jeg valgte å bruke, supplert med notater og hukommelse. I noen tilfeller blir ikke alt plukket opp av lydopptaket, ettersom vi var på en skole i skoletid, og andre lydkilder kan ha blitt plukket opp, som støy fra gangen og ringeklokken. Dette var en av ulempene ved å gjennomføre intervjuet på skolen i skoletid. For å få høyere kvalitet på lydopptaket, kunne man ha gjennomført intervjuet etter skoletid. Men dette hadde gått på bekostning av deltakerne som måtte ha tatt av sin fritid for å delta på intervjuet. Hvis jeg skulle ha brukt noe bedre utstyr til lydopptak for å få bedre lydkvalitet, ville dette ha medført mer arbeid i forbindelse med sikker lagring av opptakene. En annen ulempe ved lydopptak er at man ikke fanger opp all informasjonen som kommer fram i samtalen. Hvis vedkommende refererer til noe fysisk foran oss, om for eksempel kode som en elev har arbeidet med, kommer dette ikke med på lydopptaket. Denne ulempen kunne man ha unngått ved ta opp video istedenfor. Dette hadde også fanget opp kroppsspråket til deltakeren. For å kompensere denne nedsiden ved lydopptak, tok jeg notater underveis i intervjuet. Lydopptaket kombinert med notater førte til at jeg kunne bevare et mer helhetlig bilde av interaksjonen mellom meg og deltakerne. Det var praktisk å kunne se tilbake på notatene, slik at det kunne bli med i transkripsjonen. Selve lydopptaket ble tatt opp på mobilen min via «diktafon»-appen, som var koblet opp mot nettskjema. Alle lydopptakene er lagret der, og vil bli slettet etter at prosjektet er ferdig og levert.

3.6.2 Vurderinger under intervjuene

Selve intervjuene varte i 45-60 minutter. Utgangspunktet var intervjuguiden som ble brukt underveis og i slutten av intervjuet, for å sikre at vi hadde vært innom alle de sentrale punktene i løpet av samtalen. Det begynte med at jeg introduserte prosjektet og informerte deltakeren om personvern, deres rettigheter og varigheten på studiet. Etter denne introduksjonen, gikk samtalen videre over til spørsmålene fra intervjuguiden. Her kunne jeg få både lange og korte svar, og måtte kontinuerlig ta avgjørelser som påvirket intervjuet. Skulle jeg stille et oppfølgingsspørsmål? Til hvilke utsagn skulle jeg stille det til? Skulle vi

gå over til neste punkt? Skal jeg si meg fornøyd med svaret, eller skal jeg prøve å grave litt til? Hva jeg vurderte som best i den enkelte situasjonen ville påvirke hvordan flyten i samtalen gikk og hvilke temaer vi gikk innom. I tillegg måtte jeg passe på at vi hadde nok tid på hvert punkt og rakk å komme oss gjennom alle de sentrale temaene. Alt dette gjorde at jeg raskt innså hvor vanskelig det ville være å notere ned alt jeg ønsket under samtalen, samtidig som jeg klarte å gi oppmerksomhet til personene uten å sette en stopper for flyten. Jeg bestemte meg da for å bare notere ned helt vesentlige ting, for eksempel kode som ble vist meg på en skjerm. I alle intervjuene prøvde jeg å omformulere utsagnene deres og stille spørsmålet tilbake i form av «så det du sier, er at …» Grunnen til dette var for å sikre seg at jeg har forstått dem riktig og gi dem en mulighet til å enten oppklare eller bekrefte min oppfatning. (Kvale & Brinkmann, 2015).

3.7 Kvalitative analysemetoden

3.7.1 Transkripsjon

Det er transkripsjonen av intervjuene som er grunnlaget for studiets datamateriale, og er det som har blitt analysert. For å besvare på problemstillingen, kommer jeg til å sammenligne resultatene fra transkripsjonen med kompetanseblomsten til Niss og Højgaard (2019).

Transkripsjon handler om å gjøre en samtale mellom to personer om til en skriftlig form (Kvale & Brinkmann, 2015, s. 204). Man går fra en form til en annen, og sitter igjen med en dekontekstualisert versjon av samtalen. Ettersom det er en fortolkningsprosess, står man ovenfor mange valg som påvirker sluttresultatet. En av de største utfordringene med transkripsjon er forskjellen mellom det skriftlige og muntlige språket. Det som høres ut som et velformulert utsagn, kan fort være en vanskelig og krevende setning å forstå (Kvale & Brinkmann, 2015, s. 204-205). Mye informasjon fra samtalen kan gå tapt i prosessen. Det handler ikke bare om ord og utsagn, men også kroppsspråk og ironi. Ved ironi stemmer ikke ordene ovenfor kroppsspråket, men kroppsspråket kommer ikke med i transkripsjonen. Andre utfordringer kan være tegnsetting som komma-plassering og punktum, eller at en person starter på en setning, men fullfører den ikke før vedkommende begynner på en ny setning. Det kan være blant annet en omformulering av tankene som kan være svært nyttig eller oppklarende i samtalen. Men i oversettelsen fra muntlig til skriftlig form, gjør slike setninger det svært vanskelig å lese og passer ikke inn i den mer formelle skriftlige formen.

Selve prosessen med å transkribere et intervju kan være svært lærerik, noe jeg selv erfarte.. I etterkant av dette arbeidet, har jeg blitt mye mer bevist og oppmerksom på formuleringer og

strukturen i dagligtaler. Det var ved flere anledninger underveis hvor læreren jeg intervjuet begynte på en setning, for så å ombestemme seg halvveis gjennom og begynne på en ny formulering, uten å ha fullført den første. Etter å ha hørt på meg selv om andre snakke sammen, prøver jeg å formulere meg mer presist, og droppe småord og «tenke-lyder» som «eum», «hmmm» og «ja, nei».

Alle tre intervjuene ble holdt på skolen hvor vedkommende arbeidet i et gruppe- eller møterom. Lydkvaliteten varierte noe, og ble påvirket av ringeklokker eller annen bakgrunnsstøy. Det ene intervjuet ble stoppet underveis, da en elev ønsket å snakke med læreren sin. Slike forstyrrelser kunne gjøre det vanskelig å høre alt som ble sagt på lydopptaket da jeg skulle transkribere, og kunne bruke mye tid på enkelte seksjoner av samtalen for å fange opp alt som ble sagt. Selve transkripsjonen ble skrevet med tidsstempel og nummert. Dette gjorde det enklere for meg å gå tilbake til lydopptaket for å høre på de spesifikke spørsmålene og interaksjonen, men også enkelt kunne referere til de ulike utsagnene i teksten.

3.7.2 Koder og tegnsetting i transkripsjon

På grunn av forskjellen mellom skriftlig og muntlig form som tidligere nevnt i 3.7.1 Transkripsjon, var jeg nødt til å bruke koder og tegnsettinger. Dessverre eksisterer det ikke noen standard måte å gjøre dette på (Kvale & Brinkmann, 2015, s. 208). Jeg endte opp med å bruke **fet skrift** på ord som personen la ekstra trykk på. På denne måten er det mulig å se hvilke ord vedkommende mente var ekstra viktig i en sammenheng. Ved flere anledninger omformulerte deltakeren seg midt i en setning, eller tok pauser i snakkingen. For å få med dette i transkripsjonen, brukte jeg tre punktum, «…» for å vise en avbrytelse eller pause i setningen. Det brukes klammer på hver ende av punktumene […] når jeg siterer en ufullstendig setning. Sitatet vil være del av en større sammenheng. For å bevare anonymiteten til individene og unngå misforståelser for lesere, valgte jeg å skrive på bokmål i transkripsjonen.

3.8 Evaluering av studiet sin kvalitet

For å kunne avgjøre kvaliteten på sin egen forskning, er man nødt til å se på det med et kritisk blikk. Postholm og Jacobsen peker på tre begreper som samlet sett inngår i et studies samlede troverdighet: indre gyldighet, ytre gyldighet og pålitelighet (2018, s. 223). Postholm og Jakobsen beskriver at forskning ikke bare et resultat, men også selve prosessen til resultatet (2018, s. 219). Dette er viktig å ta med når man skal snakke om gyldigheten og påliteligheten

til studier og prosjekter, fordi resultatet man får i dag kan bli utfordret en gang i framtiden gjennom nye perspektiv eller metoder. Derfor kan man ikke bare dømme en studie ut fra resultatet, men også hovedsakelig ut fra prosessen som produserte denne kunnskapen. Fordi selv om det er riktig i dag, kan dette endre seg over tid. Som forskere kan vi aldri være helt objektive. Vi kommer alle med våre egne subjektive meninger og vinklinger som påvirker, spesielt når det er snakk om et intervju. Vår oppgave er derfor å bli så bevisst som mulig over vår subjektivitet, så den påvirker resultatet i så liten grad som mulig (Postholm & Jacobsen, 2018, s. 220).

Tradisjonelt har det vært vanlig å måle påliteligheten til en studie gjennom dens evne til å bli gjenskapt på et annet tidspunkt, og produsere de samme resultatene. Denne standarden kan være problematisk i forbindelse med en samfunnsvitenskapelig og sosial sammenheng. For å kunne gjenskape et resultat, er man nødt til å ha full kontroll på alle variablene som er involvert. Dette er en utfordring når det kommer til mennesker, fordi det er snakk om individer som stadig endrer og utvikler seg. I tillegg vil ulike forskere ta med seg sin egen subjektive individuelle teori inn i studiet. I stedet for den tradisjonelle måten å måle pålitelighet kan man heller knytte påliteligheten opp mot hvordan forskeren selv reflekterer over sin påvirkning av studiet, og gjøre forskningsprosessen synlig for andre (Postholm & Jacobsen, 2018, s. 224). I mine intervju valgte jeg å vise lærerne kompetansemålene og kjerneelementene før jeg stilte dem spørsmål om dem. Samtlige lærere påpekte kompetansemålene som eksplisitt nevner programmering, og ble da muligens ledet på en tankerekke de ellers ikke hadde fått om de fikk muligheten til å tenke seg litt om selv, før kompetansemålene ble presentert. Hvordan jeg har transkribert samtalene vil også kunne påvirke påliteligheten til studiet. Dette arbeidet har jeg tidligere beskrevet i 3.7 Kvalitative analysemetoden. Min forskningsprosess har blitt synliggjort ved å blant annet legge ved intervjuguiden som ble brukt i møte med lærerne, og ta med utdrag fra transkripsjonen fra samtalene. På denne måten får andre mulighet til å vurdere om spørsmålene som ble brukt er ledende eller uklare, og om utsagnene som har kommet fram i datamaterialet er gyldige til å svare på forskningsspørsmålene til studiet.

Postholm og Jacobsen (2018, s. 229-241) beskrive studiets validitet som bestående av to deler, indre og ytre gyldighet. Indre gyldighet er videre delt inn i to punkter. Det første punktet i indre gyldighet går ut på i hvilken grad teorien og utrykkene som brukes samsvarer med virkeligheten som skal studeres. Begrepene må være meningsfulle, både for dem som blir intervjuet, og for dem som skal lese forskningen på et senere tidspunkt. Dette har jeg

forsøkt å gjøre gjennom teori-kapittelet, der jeg tok for meg sentrale begreper og gjorde rede for tidligere forskning som er nærliggende min egen. Det andre punktet under indre gyldighet handler om kausalitet, om årsak og virkning. Men som nevnt tidligere er det ikke nødvendigvis evnen til å reprodusere de samme resultatene som avgjør gyldigheten til en samfunnsvitenskapelig studie. Man kan gjøre evalueringer om kausalitet, men det innebærer en større usikkerhet Man kan da ikke generalisere funnen og trekke absolutte konklusjoner, ettersom resultatet ikke er reproduserbart. Kausaliteten innebærer en viss usikkerhet som man ikke kan gjøre rede foruten en større kvanta datamaterialet. I denne konteksten vil det da si at jeg kan komme med forslag basert på funnene mine, men ikke trekke absolutte konklusjoner eller at det vil stemme i alle situasjoner.

Ytre gyldighet kan bli beskrevet som overførbarhet. Postholm og Jacobsen (2018, s. 238) beskriver det på denne måten: «Overførbarhet går på i hvilken grad funn fra en kontekst kan overføres – *eller generaliseres* – til andre kontekster som ikke er studert». En viktig del av arbeidet mitt har vært å finne et rammeverk som beskriver matematisk kompetanse, og hvorvidt programmering kan passe inn i disse kompetansene. Vi kan da si at det er en ytre gyldighet, da dette rammeverket er anerkjent og kan brukes i andre situasjoner som ikke innebærer programmering. Man kan også bevege seg inn på spesifikke sider av integrering av programmering i matematikkundervisningen enn det jeg har gjort.

3.9 Etiske vurderinger og hensyn

Når man foretar seg studier er det ulike etiske hensyn man må ta (Kvale & Brinkmann, 2015, s. 95-96). For mange situasjon finnes det etiske retningslinjer man må følge, men man kan møte på utfordringer som det ikke finnes tydelige retningslinjer for. Da må man være i stand til å kunne gjøre egne etiske vurderinger. Når man intervjuer andre, ønsker man gjerne å komme så nær personen som mulig, men da kan man risikere å krenke individet som blir intervjuet. På andre siden om man er for forsiktig, skraper man bare overflaten og det empiriske materialet blir ikke tilstrekkelig. Dette er en fin balanse man må finne i møte med individet (Kvale & Brinkmann, 2015, s. 96).

For å sikre det etiske grunnlaget til studiet, fulgte jeg Sikt sine retningslinjer. Før jeg gjennomførte intervjuene, søkte jeg om godkjennelse fra Sikt, ettersom studiet innebær sensitiv informasjon. Alle deltakerne fikk informasjon om studiet de var med på i form av et samtykkeskjema. I skjemaet ble det presentert studiets formål, hvem som var ansvarlig for studiet og hvem som hadde tilgang til informasjonen som ble samlet inn. De fikk også vite

hva deres rettigheter er i forbindelse med datamaterialet som ble samlet inn og hvordan dette skulle bli lagret. Stemmen til en person er å anse som en personopplysning, derfor var det nødvendig å lagre dette på en sikker måte. Lydopptakene ble tatt opp via applikasjonen «diktafon» fra universitetet i Oslo. De ble lagret kryptert på mobiltelefonen og lastet opp til nettskjema. I transkripsjonen er intervjuobjektene blitt anonymisert, og refereres kun til som Lærer 1, Lærer 2 og Lærer 3. På denne måten har deltakernes anonymitet blitt bevart. Det refereres heller ikke til skolene lærerne arbeider på, og ingen navn på elever har blitt tatt med i studiet.

Kapittel 4 - Resultat og analyse

Jeg skal i dette kapittelet analysere intervjuene og presentere resultatene av dem. Datamaterialet mitt består av intervju med tre forskjellige matematikk-lærere og transkripsjonen min av disse samtalene. Utsagnene til lærerne er basert på deres egne erfaringer og meninger om hvilke matematiske kompetanser som kan fremmes gjennom programmering. For å gjøre det mer oversiktlig for leseren, kommer analysen til å være strukturert etter temaene fra intervjuguiden. Stort sett vil utsagnene komme i kronologisk rekkefølge, men om det blir sagt noe annet som er relevant i forhold til temaet senere i intervjuer, vil det bli presentert samtidig. Jeg har utført en oversiktsanalyse der jeg har kategorisert utsagnene fra transkripsjonen, for å gjøre det enklere å sammenligne utsagnene deres. For hver lærer tar jeg først for meg hva de tenkte rundt matematisk kompetanse og programmering. Deretter går jeg inn på hvilke kjerneelement og kompetansemål som er egnet å bruke programmering som undervisningsmetode. Til slutt, etter at jeg har tatt for meg alle lærerne, gir jeg en oppsummering av datamaterialet jeg har samlet inn.

Mot slutten av alle intervjuene, spurte jeg læreren om hva som var viktigst for dem at elevene satt igjen med etter en matematikktime der programmering var tatt i bruk. Dette skulle de svare på med å plassere seg på en skala fra 1 til 10, der 1 representerte at det ikke gjorde noe om elevene ikke forstod programmeringen så lenge de satt igjen med økt kompetanse i matematikk etter undervisningen, og 10 representerte at det viktigste var kompetanse i programmering, og det gjorde ikke noe om elevene ikke lærte noe nytt i matematikken. Om noen plasserte seg på 5, vil det si at de likestiller programmering med matematikk. Det er like viktig at elevene lærer seg programmering som matematikk etter en matematikkundervisning hvor programmering ble tatt i brukt.

4.1 Samtalen med Lærer 1

4.1.1 Lærer 1 om matematisk kompetanse og programmering

I intervjuet av Lærer 1, kommer det fram at programmering kan bli brukt til å lære mange forskjellige sider av matematikk. Det første vedkommende tar fram er bruken av variabler, og hvordan elevene kan lære hva det er og hvordan det fungerer på en annen måte ved å bruke programmering. Flere ganger i løpet av samtalen kommer variabler opp, og hvordan de har brukt det i klasserommet, blant annet ved å lage spill der elevene må komme med verdier på tall som skal brukes. For at spillet skal fungere, er elevene nødt til å kunne definere og bruke variabler når de skriver programmet.

Et annet aspekt ved matematikken som ble trukket fram, var algoritmer. I forbindelse med dette sier Lærer 1:

> «Og så er det jo dette med å forstå en algoritme, å kunne bryte ned et regnestykke i små operasjoner. Det synes de ofte er vanskelig og, å vise hva gjorde du faktisk nå? Jo, du gjorde først sånn og så sånn, så da er det godt egnet til».

Senere i samtalen kommer Lærer 1 med et eksempel på hvordan mange elever har lært seg en algoritme fra barneskolen på hvordan de skal regne ut oppstilte delestykker. Dette er noe de kan gjøre igjen og igjen på ulike regnestykker, uten å nødvendigvis måtte tenke seg om hver gang de skal regne ut et slik regnestykke. Når elevene skal programmere, blir de tvungen til å tenke gjennom steg for steg hvordan de faktisk skal utføre operasjonene, å tenke algoritmisk. Ettersom en datamaskin ikke forstår hva vi ønsker å gjøre med mindre problemet blir brutt ned til mindre biter, argumenterer Lærer 1 at programmering vil egne seg godt til å lære elevene algoritmisk tenkning.

4.1.2 Lærer 1 om kjerneelement

Det første Lærer 1 sier etter at jeg viste fram de seks kjerneelementene, var at «Programmering går jo nesten inn i alle disse, tenker jeg». Selv om alle kunne være aktuelle i programmering, blir likevel kjerneelementet «utforsking og problemløsing» trukket spesielt fram. Grunnen til dette var at strategier og hvordan man gikk fram var viktigere enn selve løsningen i både programmering og problemløsning, og at algoritmisk tenkning også er sterkt knyttet til dette. Dette gjenspeiler hvordan Lærer 1 tidlig i samtalen trakk fram algoritmisk tenkning som en del av matematikken man kan lære gjennom programmering.

På motsatt side, når jeg spurte Lærer 1 om det var noen av disse som pekte seg ut som lite relevant til programmering, ble «Resonnering og argumentasjon» trukket fram. Det ble forklart at det eksisterte en overlapp, og at programmering også var relevant i forhold til dette kjerneelementet. Men Lærer 1 mente at det ville være for mye å be elevene på ungdomsskolen å bruke programmering som argumentasjon. Elevene hadde nok med å bruke vanlige regnestykker som argumentasjon. Det vil da si at Lærer 1 ser en kobling mellom kjerneelementet og programmering, men elevene har som regel ikke tilstrekkelige ferdigheter og kunnskaper i programmering, til å kunne bruke dette som et verktøy for å resonnere og argumentere i matematikk.

Det ble ytret en bekymring ved å bruke digitale verktøy ukritisk i undervisningen. Derfor ønsket Lærer 1 å skape en kobling mellom programmering og matematiske

kunnskapsområder, for at det faktisk skulle være med å utvikle deres matematiske forståelse. Det var et ønske om å være bevisst ovenfor elevene at programmeringen alltid skulle bli knyttet opp til hva de lærte i matematikken, og ikke at dette var en pause fra undervisningen der man lærte programmering, for senere å komme tilbake til matematikken uten faktisk å ha kombinert disse to. I dette tilfellet ønsket Lærer 1 å vise elevene at man kunne bruke algebra når man programmerer, og arbeidet med en oppgave som omhandlet at en figur skulle navigere en labyrint. Lærer 1 prøvde å være veldig bevisst på dette med klassene sine, å alltid knytte programmering sammen med matematikken, slik at det ikke blir noe man holder på med på siden, men derimot er en integrert del av matematikkundervisningen.

4.1.3 Lærer 1 om kompetansemål

I intervjuet sier Lærer 1 i forbindelse med kompetansemålene fra 8. trinn:

> «Ja, beskrive og generalisere mønstre. Da trenger man jo gjerne en løkke sant, men se hvordan noe utvikler seg når vi øker opp … Jeg tenker og egentlig altså alt, sånn som her har du utforskning av potenser.»

Her nevnes det at man kan bruke programmering i alle kompetansemålene i matematikk på 8. trinn, og spesifikt at man kan bruke løkker i arbeid med generalisering av mønstre. Videre forteller Lærer 1 at programmering er et fint verktøy til å drive med utforskning og å knytte matematikk opp mot praktiske situasjoner i de forskjellige kompetansemålene.

Videre på kompetansemålene til 9. trinn, nevnes «beregne og vurdere sannsynlighet i statistikk og spill», og hvordan de har brukt programmering til å lage ulike typer spill. Elevene kan lage urettferdige spill og rettferdige spill, og diskutere sammen om sannsynligheten for å vinne i de ulike spillene. Lærer 1 var litt usikker, men tenkte at dette var det mest relevante kompetansemålet på 9. trinn, utenom å simulere utfall ved å bruke programmering. Kompetansemålet som handler om å simulere tilfeldige utfall, ville selvfølgelig være det mest relevante, da det nevner programmering spesifikt.

Mange av kompetansemålene i 9. trinn omhandler geometri, men det kommer fram i samtalen at Lærer 1 ikke har brukt programmering så mye til geometri, selv om vedkommende kan se potensialet, ettersom det er enkelt å endre på forutsetningene og raskt få tilbakemelding på det nye resultatet. Et eksempel var å bruke programmering i forbindelse med formel for utregning av volum. Der kan man se hva som skjer med volumet hvis man for eksempel dobler radiusen. Det gir en umiddelbar visuell representasjon av det abstrakte

matematiske konseptet, som kan hjelpe på forståelsen for elevene. Generelt var programmering godt egnet til å utforske formelsammenhenger, hevdet Lærer 1.

Lærer 1 hadde ikke undervist på 10. trinn, men utrykte at 8. og 10. trinn speiler hverandre litt i innholdet, med blant annet algebra, funksjoner og ligninger. Lærer 1 så på regneark som en form for programmering. Derfor kunne kompetansemålet «planlegge, utføre og presentere et utforskende arbeid knyttet til personlig økonomi» være svært relevant. Det kom fram i flere sammenhenger at programmering var godt egnet til oppgaver og kompetansemål der man skulle endre på forutsetningene eller utforske sammenhengene mellom noe. Ved å endre på noen variabler, får man kjapt tilbake resultatet og kan lett sammenligne og forstå effekten. Derfor kunne også kompetansemålet om å «utforske sammenhengen mellom konstant prosentvis endring, vekstfaktor og eksponentialfunksjoner» være godt egnet å bruke programmering i.

Man kunne også bruke programmering til å regne ut stigningstall til en lineær funksjon, men det var mer praktisk å utføre dette grafisk gjennom for eksempel Geogebra. Geogebra er et interaktivt grafisk kalkulatorprogram man kan bruke på en datamaskin eller laste ned som en applikasjon på mobiltelefonen. Lærer 1 peker også ut at man kunne bruke programmering til å modellere situasjoner knyttet til reelle datasett, men at datasettene man driver med på ungdomsskolen blir for små for at programmering skal være hensiktsmessig å bruke.

4.1.4 Andre interessante aspekter fra intervjuet med Lærer 1

Under intervjuet snakket vi om andre interessante momenter i forbindelse med programmering og matematikkundervisning, som kan være verdt å ta opp. Jeg utfordret Lærer 1 på om programmering var nødvendig for å bli kyndig i matematikk. Responsen var at man ikke trenger programmering for å lære matte, men at det er et veldig nyttig verktøy som elevene kan få bruk for senere i livet. Programmering har en verdi i seg selv, men det må ikke komme i veien for matematikk. Matematikk er viktigst, og det er en forskjell på det og programmering. Hvis man lærer seg programmering, så kan det ha en egen nytteverdi utover matematikkunnskapene. Selv om de ikke nødvendigvis blir bedre i matematikk, blir de bedre i «livets matte» gjennom å programmere, utforske problemer og se ting på forskjellige måter.

Lærer 1 syntes det var vanskelig å plassere seg på en skala. Det ble forklart at skolen var så tidlig i implementeringen av programmering i matematikkundervisningen. Lærerne var nødt til å ha masse fokus på programmering og skape et godt fundament, slik at programmering kunne bli brukt mer i undervisningen framover. Likevel plasserer Lærer 1 seg på 4.

Lærer 1 legger opp undervisningen på et lavt nivå, så kan elevene utforske selv og gjøre oppgaven mer avansert om de føler nivået blir for lett for dem. Det ble også kommentert at elevene lærte seg viktigheten av å være nøye og å kunne stå i ting når det ikke fungerer med en gang når de programmerer. Når man skriver en feil i koden sin, får man opp en feilmelding som man må undersøke. For at programmet skal fungere, må man finne denne feilen og rette opp på den. Enten så fungerer koden, eller så gjør den ikke det.

4.2 Samtale med Lærer 2

4.2.1 Lærer 2 om matematisk kompetanse og programmering

I samtale med Lærer 2, kommer det fram at vedkommende ser for seg at en gang i framtiden kommer programmering til å fase ut mange av de andre digitale verktøyene vi bruker i matematikk i dag, deriblant regneark og Geogebra. Tanken bak det er at alle disse verktøyene kjører på en kode, så hvis elevene på et tidspunkt blir kyndige nok i programmering, vil de kunne skrive det i Python og lage sine egne verktøy. I denne forbindelse hevdes det at det er noen kompetansemål som ikke kan oppnås uten å bruke programmering i dag. Eksempelet som ble trukket fram, var kompetansemålet fra 9. trinn «simulere utfall i tilfeldige forsøk og beregne sannsynligheten for at noe skal inntreffe, ved å bruke programmering» (Utdanningsdirektoratet, 2019d). Her står det jo konkret at det skal brukes programmering i dette kompetansemålet, men Lærer 2 utdyper dette og sier:

> «Den mest konkrete jeg kommer på i hodet nå, er jo den fine med «simulere forsøk». Det er helt u … Du **kan** gjøre det fysisk. Du kan ikke faktisk simulere så mange forsøk som du trenger for å ha tilfredsstillende simulert resolusjon. Du simulerer 10 ganger, så er det fortsatt sannsynlighet for at det kan være tilfeldig. Hvis du simulerer 10.000 forsøk, og det gjør du ikke for hånd liksom, da må du programmere».

Selv om det presiseres i kompetansemålet at man skal bruke programmering, vil det være svært vanskelig å gjennomføre dette for hånd. Altså man er nødt til å bruke programmering her, uavhengig av at det presiseres. Videre forteller Lærer 2 hvordan vedkommende håper at vi kommer til det punktet hvor man kan ta i bruk programmering i alle temaene i matematikk. At hver gang man begynner på noe nytt, lærer man det fysisk for hånd, og mot slutten av hvert emne blir det lagt til en programmeringsdel hvor elevene lærer seg hvordan man kan kombinere programmering og matematikken de nettopp har lært. Ikke bare lærer man å bruke programmering og matematikk sammen, men Lærer 2 påpeker at dette vil hjelpe elevene med

den logiske og strukturelle biten av matematikk, ettersom man er nødt til å være pinlig nøyaktig når man skriver et program.

4.2.2 Lærer 2 om kjerneelement

I samtalen når vi snakket om kjerneelementene i matematikk, nevner Lærer 2 «modellering og anvendelser» (Utdanningsdirektoratet, 2019b). Det utdypes ikke noe mer om det, annet enn at de bruker programmering for å lage modeller som settes inn i praktiske situasjoner. Videre snakker vedkommende litt spørrende om hva godkjent argumentering er. På grunn av måten man arbeider på når man programmerer, vil dette være nyttig i arbeid med sin egen resonnering og argumentasjon. Man er nødt til å sette ting i riktig rekkefølge, være ryddig når man skriver koden, og er nødt til å finne ut hva som er feilen når programmet ikke fungerer. Dette er argumentene Lærer 2 trekker fram for å vise hvordan programmering kan brukes i kjerneelementet om resonnering og argumentasjon.

Videre poengterer Lærer 2 at kjerneelementet om matematiske kunnskapsområder trolig var den minst relevante i forhold til bruken av programmering i undervisningen. Det kommer fram at Lærer 2 ser på programmering mer som et verktøy og en anvendelse av matematikken, og trekker et tydelig skille mellom programmering og matematikk. Det blir referert til en samtale Lærer 2 hadde med en kollega, hvor de snakket som om utfordringen med å lage gode oppgaver hvor både matematikken og programmering er på et tilfredsstillende nivå. Hvis programmeringen skulle være utfordrende nok, måtte matematikken nødvendigvis ligge på et mye lavere nivå enn hva elevene ligge på. Og hvis elevene skal bli utfordret av matematikken i oppgaven, vil programmeringen bli for avansert.

Lærer 2 kommenterer også kjerneelementet om «abstraksjon og generalisering», og sier at dette er et veldig vidt tema hvor mye kan passe inn. Men programmering kan brukes til å finne mønster og generalisere. Å generalisere kunne man gjøre gjennom å lage riktige modeller. Lærer 2 fortalte videre at dette ville muligens bli mer en problemløsningsoppgave, enn en oppgave om generalisering og abstraksjon.

4.2.3 Lærer 2 om kompetansemål

Under intervjuet spør jeg Lærer 2 om det er noen av kompetansemålene til 9. trinn som egner seg spesielt godt til programmering i matematikk. Responsen var:

> «Nå bare ser jeg kjapt over alle de for niende, og jeg vil egentlig snu om på det spørsmålet siden det er faktisk … Er det noen som **ikke** passer til å programmere? Jeg tror ikke det er noen som ikke passer […] ».

Etter å ha snakket litt om de ulike kompetansemålene, avslutter Lærer 2 med: «Hva er det som egentlig ikke kan brukes? Egentlig ingenting. […] Alt kan programmeres. Alle kompetansemålene kan bruke programmering». Læreren refererer her til alle kompetansemålene i matematikk på ungdomsskolen. Lærer 2 hevder at programmering kan brukes i alle aspektene ved matematikkundervisningen, men nevner spesifikt geometri som godt egnet. Tanken her er å la elevene erstatte Geogebra med programmering i for eksempel «Turtle», og la den tegne ulike former for elevene. Turtle er en modul i programmeringsspråket Python, hvor man får en skilpadde til å tegne på skjermen ved å skrive kommandoer i form av kode. På denne måten kan elevene utforske og sammenligne egenskapene til geometriske figurer ved å endre på verdiene deres.

«Utforske og argumentere for formler for areal og volum» var ifølge Lærer 2 det kompetansemålet som var minst egnet til å bruke programmering i, ettersom det var «rent matematisk». Det fantes lite spillerom rundt dette, men Lærer 2 mente det burde være mulig å tenke litt kreativt her for å tilpasse dette til programmering. Vedkommende kom imidlertid ikke med noen eksempler på dette. Videre påpeker Lærer 2 at mange av kompetansemålene på 10. trinn som omhandler økonomi, absolutt er mulige å oppfylle ved å bruke programmering. Når jeg bær om en utdypelse, kommer vi inn på regneark og hvordan det er mange som er svært dyktige, uten å vite at de faktisk programmerer i regnearket. Lærer 2 presiserer at det også er mulig å programmere et helt regnskap i Python, men ofte er det mer oversiktlig å bruke regneark. Det er gjerne mer intuitivt også, da det bruker et grafisk brukergrensesnitt, i motsetning til tekstbasert programmering som Python.

Når jeg utfordrer Lærer 2 på hvilke kompetansemål som egner seg spesielt godt til programmering, refereres det til det siste punktet på 9. trinn, som handler om å simulere utfall i tilfeldige forsøk, ved å bruke programmering. Grunnen til dette var fordi det kreves at man bruker programmering i dette kompetansemålet. Lærer 2 hadde planer om å bruke «random» funksjonen i Python for å simulere ulike forsøk med terninger. Vedkommende mente også at man kan bruke det til å lage ulike tallrekker, som for eksempel Fibonacci-rekken. Dette er noe som tar lang tid å gjøre for hånd, men hvis elevene får bruke programmering til å lage instruksene, er det mulig å produsere langt flere svar.

4.2.4 Andre interessante aspekter fra intervjuet med Lærer 2

Når Lærer 2 blir utfordret på å plassere seg på skalaen om hva som er viktigst å sitte igjen med etter en undervisningstime, ønsket ikke Lærer 2 å svare på det. Grunnen var at

programmering og matematikk er to forskjellige ting som ikke konkurrerer med hverandre. Begge er viktige og nødvendige. Lærer 2 ønsket at elevene skulle få økt kompetanse i både matematikk og programmering, og at dette ikke trengte å være motstridende. Dette henger sammen med at Lærer 2 skiller tydelig mellom programmering og matematikk. Det blir flere ganger presisert at programmering er en anvendelse av matematikk. Det blir hevdet at det er et for stort gap mellom elevenes kompetanse i matematikk og programmering. Mye tid blir kastet bort fordi elevene ikke kan det fra før, eller at læreren ikke er god nok i programmering. Men det kommer fram at Lærer 2 håper på en framtid hvor programmering har blitt en integrert del av all undervisning på skolen, i alle fag. Med dette følger det at lærere og elever har en mye større kompetanse innenfor programmering, som gjør det mulig å bruke dette som et verktøy til det fulleste.

4.3 Samtale med Lærer 3

4.3.1 Lærer 3 om matematisk kompetanse og programmering

I samtalen med Lærer 3, kommer det fram at elevene lærer seg å arbeide mer systematisk i matematikken ved å jobbe med programmering i undervisningen. Grunnen til dette er at når elevene skriver for hånd, trenger ikke regnestykkene og føringen deres nødvendigvis å se bra ut, så lenge eleven forstår det selv i hodet sitt. Eleven har ikke sett noe behov for å skrive regnestykkene opp på en oversiktlig og ryddig måte som gir mening, så lenge de klarer å komme fram til rett svar. Men når elevene møter programmering, fungerer ikke denne arbeidsmåten lenger. Lærer 3 sier det slik:

> «[…]de fører helt forferdelig og usystematisk fordi de forstår det i hode, sant? Mens i programmering så er du nødt for å systematisere arbeidet sitt veldig, for at det skal fungere, og gjerne luke ut feil, og feilsøke og finne … Altså virkelig arbeide mer med lite materie-stoff […]».

Elevene blir altså tvungne til å arbeide systematisk, og gjøre det forståelig. Ikke bare for seg selv, slik som for hånd, men for datamaskinen. Samtidig kan dette virke som et tveegget sverd. Lærer 3 forteller nemlig videre at dette er veldig utfordrende for mange, og god progresjon blant elevene krever tett én til én veiledning, noe som blir utfordrende i en klasse med 30 elever.

4.3.2 Lærer 3 om kjerneelement

I spørsmålet rundt programmering og kjerneelementene, forteller Lærer 3 at alle kjerneelementene er relevante, men peker seg inn på «modellering og anvendelser» som noe

utfordrende. Det er ikke et problem å bruke modeller i matematikken sammen med programmering, men det vanskelige er anvendelsen av dem. Elevene er i stand til å forstå koden som blir presentert for dem, og de kan gjøre endringer i koden. Men å gå fra den instrumentelle til en mer relasjonell forståelse av koden er mye vanskeligere. De er i liten grad stand til å gjøre om koden eller modellen de arbeider med, og vurdere om dette kan anvendes i andre situasjoner. Lærer 3 kunne for eksempel vise og forklare en kode til et spill for elevene. Elevene var i stand til å forstå koden og gjøre endringer der, men møtte fort en vegg om de skulle skrive koden fra bunnen av selv.

Når Lærer 3 blir videre utfordret på å utdype mer om kjerneelementene og programmering, responderer vedkommende slik:

> «Jeg kjenner det for å være helt ærlig … så jeg føler jeg det går veldig i hverandre uansett. Programmering, det er jo et språk, sant, og det er både matematisk, det er argumenterende, det er abstraksjonen, det er generalisering. Det er egentlig alt. Og det er kanskje det som gjør det litt utfordrende for elevene å forstå hvor de skal begynne».

Videre påpeker lærer 3 at programmering har potensial til å brukes innenfor alle kjerneelementer. Dette mente vedkommende fordi det tar for seg alle disse ulike elementene som tilhører matematikken. Men dette er også en av utfordringene til programmering, at det inneholder så mye at det blir vanskelig å bygge videre på tidligere kunnskap litt etter litt. Lærer 3 sammenligner programmering med matematikk, og forteller hvordan selv det å skrive en kort kode kan kreve ganske mye forståelse om hvordan alle delene henger sammen. I matematikken derimot, mener vedkommende at det er mye lettere å komme med en liten ny ting som bygger videre på eksisterende kunnskap til elevene.

4.3.3 Lærer 3 om kompetansemål

Når vi begynner å snakke om kompetansemålene i matematikk, nevner Lærer 3 med en gang det siste punktet på 8. trinn om «utforske hvordan algoritmer kan skapes, testes og forbedres ved hjelp av programmering», og forteller at dette er noe de har brukt mye tid på. Noen andre kompetansemål var «lage, løse og forklare ligninger knyttet til praktiske situasjoner» og «lage og forklarer regneuttrykk, med tall, variabler og konstanter, knyttet til praktiske situasjoner». Her forteller Lærer 3 at de har programmert små biler til å kjøre, og at dette er et eksempel på å knytte matematikken til praktiske situasjoner. Det forklares også at man bruker mye variabler og konstanter generelt når man programmerer. Ved å la elevene leke og teste ut disse bilene og hvordan de kan styre dem gjennom programmering, ser de hvordan variablene

påvirker den virkelige verden. Variabelen for fart er ikke bare et tall, men kan være avgjørende for om bilen klarer svingen eller ikke.

Senere i samtalen spør jeg om noen kompetansemål ikke er egnet til å innlemme programmering i. Da nevner Lærer 3 at flere kompetansemål går inn på økonomi og algebra, og at denne type matematikk ville være vanskelige å koble opp mot programmering. Det er mulig, men i en mye mindre grad enn de andre kompetansemålene som ble nevnt tidligere.

4.3.4 Andre interessante aspekter fra intervjuet med Lærer 3

Jeg ber Lærer 3 plassere seg på skalaen som nevnt tidligere, og vedkommende plasserer seg mellom 7 og 8. Elevene har mer enn nok med programmering for øyeblikket, og de har ikke kapasitet til å fokusere på matematikk i den grad man ønsker når programmering er involvert. Lærer 3 har ikke store ambisjoner om at elevene skal bruke mye av programmeringen sin i matematikken akkurat nå, men mer på sikt. Elevene har for lite tid til å mestre programmering. Det er utelukkende positivt med programmering, men man mangler tid. Her blir det sammenlignet med progammeringsvalgfaget. Det kommer blant annet frem at i det valgfaget er det færre elever, som gir læreren mer tid til individuell oppfølging, som er så viktig. Elevene lærer mye om struktur og å arbeide systematisk gjennom programmering. Mange elever blir frustrerte når datamaskinen ikke forstår hva de mener. Men på den andre siden får de veldig konkrete tilbakemeldinger når koden først fungerer. Enten så fungerer koden, eller så gjør den ikke det. Når koden først fungerer, opplever elevene at de får til noe i faget, og får en mestringsfølelse. Mot slutten av samtalen gir jeg Lærer 3 mulighet til å ta opp noe som vedkommende ønsket å si noe om i forbindelse med programmering og matematikkundervisning.

> «[…] så er det veldig slik at vi bruker mye tid på å «lage veien mens den blir til» da. Og vi vet egentlig ikke helt hvor veien går, hehe. Enn hvor lite gjennomtenkt det høres ut, så er det litt sånn da, at vår største utfordring er å finne ut hva programmering er for oss og for våre elever. Hva er nytteverdien av programmering, egentlig? I stedet for å bare bruke masse tid på det, hva er på en måte, gevinsten, eller læringen da. Ikke bare å gjennomføre, men hva er læringen i programmering? Det er på en måte der vi er nå da og prøve å finne ut av, som er enormt stort tema».

4.4 Oppsummering av analysen

I dette kapittelet har jeg presentert ulike utsagn som tre forskjellige lærere har kommet med i en intervjusetting. Gjennom sitat og gjenfortellinger av deres tanker og meninger, har vi sett

hvordan de forstår programmering sin plass i matematikkundervisningen, og knytter programmering med matematikk. Som vi ser, har Lærer 1, 2 og 3 mange like tanker, men også punkter hvor de skiller seg fra hverandre. Jeg har laget en oversikt i Tabell 2 som viser hvilke kjerneelement i matematikk Lærer 1, 2 og 3 mener var mest og minst egnet til programmering

Tabell 2 - Lærerne og kjerneelement

Lærere → Kjerneelement↓	Lærer 1	Lærer 2	Lærer 3		**Fargekode**
Utforskning og problemløsning					Spesielt egnet
Modellering og anvendelser					Egnet
Resonnering og argumentasjon					Utfordrende
Represetasjon og kommunikasjon					Ikke egnet
Abstraksjon og generalisering					
Matematiske kunnskapsområder					

Lærer 1 knyttet kjerneelementet «utforskning og problemløsning» sterkest til programmering, mens «resonnering og argumentasjon» var mer utfordrende for elevene å bli kompetent på ved å bruke programmering.

Lærer 2 derimot, koblet programmering mest til kjerneelementet «resonnering og argumentasjon», som Lærer 1 syntes var utfordrende. Tanken til Lærer 2 var at elevene må tenke logisk og resonnere seg fram hvordan koden kan se ut. Lærer 2 kom fram til at programmering var mindre nyttig i to av kjerneelementene. Dette gjaldt «abstraksjon og generalisering» og «matematiske kunnskapsområder». Det var mulig å kombinere programmering sammen med «abstraksjon og generalisering», men dette var et veldig vidt tema, og vedkommende kom ikke på noen spesifikke eksempler som man kunne bruke programmering i her. Det var utfordrende å kombinere «matematiske kunnskapsområder» med programmering, på grunn av nivåforskjellen til elevene. Enten ble matematikken for enkel, eller så ble programmeringen for vanskelig.

Lærer 3 pekte ut kjerneelementet «modellering og anvendelser» som minst relevant i forhold til programmering. Modelleringsbiten var grei, men det var utfordrende for elevene å kunne anvende programmeringen, å bruke det i andre sammenhenger enn akkurat det som ble vist.

De andre kjerneelementene gikk så mye inn i hverandre og i programmering, at ingen pekte seg spesielt ut for Lærer 3. Alle var relevante å lære gjennom programmering.

Kapittel 5 - Drøfting

Jeg skal i denne delen drøfte funnene fra analysen i lys av relevant teori for å svare på problemstillingen min. Funnene fra forrige kapittel blir presentert i hvert sitt delkapittel basert på Niss og Højgaard (2002) sin kompetanseblomst. Matematiske kunnskapsområder har ikke blitt tildelt noen spesiell plass i kompetanseblomsten, da Utdanningsdepartementet knytter dette kjerneelementet opp til alle kompetansemålene. Alle de andre kjerneelementet har spesifikke kompetansemål knyttet til seg i læreplanen.

5.1 Tankegangskompetanse

Tankegangskompetansen er knyttet til kjerneelementet om «abstraksjon og generalisering», da begge disse nevner abstraksjon og generalisering av matematiske resultater (Niss & Højgaard, 2002, s. 47). Et av kompetansemålene som tilhører både dette kjerneelementet og tankegangskompetansen, er «beskrive og generalisere mønstre med egne ord og algebraisk». Her har Lærer 1, 2 og 3 ulik forståelse av dens plass i forhold til programmering. Lærer 1 knytter dette kompetansemålet sterkt opp mot programmering, mens Lærer 3 konkluderer med at dette kompetansemålet er mer komplisert å oppfylle med programmering i undervisningen. Lærer 2 er et sted mellom Lærer 1 og 3, hvor det er fult mulig å knytte programmering til dette kompetansemålet, men det er andre kompetansemål som egner seg bedre. Denne forskjellen kan stamme fra hvor mye lærerne knytter matematikk opp mot programmering. Om man ser på programmering og matematikk som to veldig forskjellige ting, er det naturlig at man ikke lærer vekk et tema som algebra gjennom programmering. Lærer 1 er veldig tydelig på å alltid gjøre programmeringen relevant i temaet de jobber med i matematikken. Dette var også noe Lærer 2 tok opp, men vendte mer i retningen at vedkommende ønsket å komme dit en gang i framtiden. Lærer 3 vektla programmeringen mest skilt fra matematikken. Dette synet på programmeringens relevans i forhold til programmering, kan forklare det forskjellige synet disse lærerne har på programmering og algebra.

Flere av kompetansemålene som handler om geometri, faller også inn under dette kjerneelementet. Både Lærer 1 og 2 påpeker at geometri fungerer bra sammen med programmering, og at dette er et velegnet verktøy å bruke i undervisningen. Lærer 3 nevner ikke noe om hvordan programmering kan henge sammen med geometri. Flere programmeringsspråk eller moduler er rettet mot å lære geometri, som for eksempel «Turtle».

Calder (2018) fant også at «Scratch» kunne bli bruk for å fremme geometrisk tenkning. Mye tyder på at kompetansemål som omhandler utforskning og geometri kan arbeids med i programmering.

5.2 Problemløsningskompetanse

Problemløsningskompetansen er muligens den kompetansen som er sterkest knyttet til programmering gjennom bruken av algoritmisk tenkning, sammen med digital kompetanse (Bocconi et al., 2018, s. 7). Alle lærerne kom fram til at programmering og problemløsning gikk godt sammen, men bare Lærer 1 spesifiserte at denne koblingen var sterkere enn andre kompetanser. Dette kom også til syne ved at Lærer 1 var den eneste som uoppfordret tok opp algoritmisk tenkning som en del kjerneelementet «utforskning og problemløsning». Lærer 3 snakket om essensen til algoritmisk tenkning uten å bruke begrepet, da vedkommende sa:

> «[…] mange elever som blir frustrert over at koden ikke forstår hva de tenker. I det mener jeg at […] at elevene må lære seg å bryte opp sin egen tenking så mye at selv en datamaskin kan forstå det».

Ifølge artikkelen «demystifying computationl thinking» nevnes det fire komponenter av algoritmisk tenkning som går igjen i litteraturen. Den ene er dekomponering, å bryte noe ned i mindre biter, den andre er algoritmer som handler om instruksjoner for framgangsmåten, og den tredje er feilsøking, altså å finne ut hvor feilen ligger, og rette den opp. Den fjerde og siste komponenten er abstraksjon (Shute et al., 2017, s. 10). Slik som Lærer 3 beskriver det, utvikler elevene disse tre komponentene gjennom programmering.

Det er knyttet fire kompetansemål til dette kjerneelementet på 8. trinn, men bare kompetansemålet om «utforske, forklare og sammenligne funksjoner knyttet til praktiske situasjoner» ble nevnt av Lærer 1, til tross for fremhevelsen av kjerneelementet kompetansemålet tilhører. På 9. trinn er tre av kompetansemålene om geometri direkte knyttet til utforskning og problemløsning, sammen med å simulere utfall og hvordan framstillinger av data kan fremme ulike synspunkt. Ingen av lærerne koblet «fremstilling av tall og data kan brukes for å fremme ulike synspunkt» som relevante for programmering.

Det kan tyde på at dette kompetansemålet ikke er så godt egnet til å kombinere med programmering. En mulig grunn kan være at det er utfordrende å fremme ulike synspunkt gjennom programmering. På en annen side kunne den dynamiske fordelen programmering tilbyr, bli brukt i dette kompetansemålet. Ved å endre på akser eller filtrere data kan man illustrere hvordan fremstillingen av de samme tallene og dataene kan brukes for å få frem

ulike synspunkt. Dette trakk Lærer 1 og Lærer 2 frem som en fordel når det gjaldt geometri, men knyttet det ikke opp mot dette punktet. Men begge lærerne mente at å simulere utfall var veldig egnet til programmering. Det er veldig naturlig, da dette kompetansemålet spesifiserer at man skal bruke programmering. Men som tidligere skrevet, kommenterer Lærer 2 at dette kompetansemålet ikke er hensiktsmessig å gjennomføre med mindre man bruker programmering. Simulering krever så mange forsøk at det i praksis blir helt umulig å gjøre det for hånd.

Lærer 1 gav et generelt positivt inntrykk av å kombinere geometri og programmering, men det virket som om det ikke var det mest naturlige for vedkommende, ettersom Lærer 1 ikke hadde brukt det i undervisningen sin. Lærer 2 la vekt på geometri som godt egnet, og hadde selv hatt undervisning med det ved å bruke «Turtle». Men det er interessant at selv om Lærer 2 var mer positiv til programmering og geometri, var Lærer 2 uenig med Lærer 1 om utforskning av volum og areal. Grunnen til denne forskjellen er vanskelig å avgjøre, men det kan være at Lærer 2 ikke koblet volum og programmering fordi vedkommende ikke så på det som noe dynamisk man kan «leke» med ved å endre på verdier, i motsetning til Lærer 1. Lærer 3 påpekte aldri geometri, verken positivt eller negativt.

På 10. trinn er det seks kompetansemål knyttet opp mot dette kjerneelementet, hvorav Lærer 1 trakk fram to av disse. Disse kompetansemålene var «Utforske sammenhengen mellom konstant prosentvis endring, vekstfaktor og eksponentialfunksjoner» og «Planlegge, utføre og presentere et utforskende arbeid knyttet til personlig økonomi». Begge disse kompetansemålene er sterkt knyttet til programmering ifølge Lærer 1. Her tenker også Lærer 2 mye likt, ved at økonomi og programmering har mye til felles. Lærer 3 peker seg ut og hevder at programmering og økonomi ikke går så godt sammen. Denne forskjellen kan trolig kobles til lærerens syn på hva programmering er. Lærer 1 og Lærer 2 forklarer hvordan regneark fungerer som en form for programmering, ved at man bruker funksjoner og kobler celler sammen med verdier og bruker dem som variabler. Trolig har Lærer 3 koblet økonomi opp mot regneark, men ikke sett sammenhengen mellom programmering og regneark. Lærer 2 mente at man kunne føre regnskap i Python uten å bruke regneark, men dette ville være mindre intuitivt og hensiktsmessig. Det vil også kreve større evner innen programmering før man kan gjøre dette. I tillegg er regneark noe som brukes i mange sammenhenger, og har en verdi av at elevene lærer seg bruken av.

5.3 Modelleringskompetanse

Modelleringskompetansen henger sammen kjerneelementet om «modellering og anvendelser», da begge tar for seg modellering og bruken av matematikk i den virkelige verden (Niss & Højgaard, 2002, s. 52). Lærer 3 påpeker at programmering og modellering fint kan kombineres, men det er anvendelsen som er utfordrende. Utfordringen ligger ikke i egenskapene til programmering, men elevenes kompetanse i det. Hvis elevene blir bedre i programmering over tid, burde man kunne utvide bruken til anvendelsen også. Lærer 2 kommenterer ikke noen slike utfordringer i forbindelse med modellering og anvendelse. Samtidig opplever Lærer 2 en stor forskjell i elevenes matematiske kompetanse kontra deres kompetanse i programmering. Det virker som om både Lærer 2 og 3 opplever at elevene vil kunne bruke programmering i en større del av matematikken, om elevene hadde forbedret sine evner i programmering. Lærer 1 påpeker også at undervisningen med programmering blir lagt på et mye lavere nivå enn hva som forventes, så elevene får mulighet til å utforske mer selv. Alle tre lærerne opplever at elevene burde hatt et høyere kompetansenivå i programmering enn i dag. Dette kan forklares ut fra når programmering ble innført som obligatorisk del av grunnskolen. Det er enda ikke noen av elevene som nå er på ungdomsskolen som har hatt programmering fra 5. klasse, som er trinnet dagens elever begynner med programmering. Trolig vil elever være bedre i programmering når de begynner på ungdomsskolen de neste årene. For vært år som går, kommer det elever som har lært programmering fra en yngre alder enn kullet over dem.

5.4 Resonneringskompetanse

Resonneringskompetansen har mye til felles med kjerneelementet «resonnering og argumentasjon». Det var mange forskjeller meninger rundt denne kompetansen mellom lærerne. Igjen kommer det opp at det ikke er mangel på overlapp mellom programmering og resonnering, men nivået til elevene er ikke tilstrekkelige, i hvert fall ifølge Lærer 1. Lærer 3 kom fra et annet synsvinkel, og så på argumentasjon som noe felles mellom programmering og matematikk. Ved å programmere utviklet elevene sine evner til å argumentere. Lærer 2 var den som virket mest positiv til å kombinere programmering og resonnering. På grunn av hvordan programmering er bygget opp, vil det nødvendigvis føre til at elevene må resonnere og argumentere overfor sine valg når de koder. Likevel var det Lærer 3 som refererte til færrest kompetansemål knyttet til resonnering og argumentasjon. Lærer 1 og Lærer 2 snakket om flere kompetansemål som innebærer å argumentere, blant annet om geometri. Det er interessant å se hvor stort gap det er mellom lærerne her, når denne kompetansen er sterkt

knyttet til problemløsning og modellering (Niss & Højgaard, 2002, s. 54), noe som alle lærerne var positive til. Dette gapet kan forklares ut fra at Lærer 1 har aldri brukt programmering i forbindelse med geometri. Om denne læreren fått noe erfaring med for eksempel «Turtle» i undervisningssammenheng, kunne meningen rundt geometri og argumentasjon knyttet til programmering blitt mer positiv.

5.5 Representasjonskompetanse

Denne kompetansen har mye felles med deler av kjerneelementet «representasjon og kommunikasjon». Resten av dette kjerneelementet vil bli tatt opp i 5.7 Kommunikasjonskompetanse.

Dette var et tema som ble lite omtalt av lærerne. Lærer 1 snakket ikke direkte om representasjon, men koblet programmering med tre kompetansemål knyttet til kjerneelementet «representasjon og kommunikasjon». Det ene var å planlegge et arbeid knyttet til personlig økonomi, noe som alle tre lærerne kom inn på, som nevnt i 5.2 Problemløsningskompetanse. Ingen av lærerne nevnte noe om ulike måter å representere informasjon på, men tok det nesten som en selvfølge at regneark er noe man bruker i forbindelse med økonomi. Her kommer det opp mange måter å representere tallene på, for eksempel tabeller, diagram eller grafer. Lærer 1 snakket litt rundt om å la elevene se på noen små datasett, men at disse ofte ville være for små til å bruke i programmering. Her kunne man potensielt ha sammenlignet ulike måter å representere informasjon på, gitt mengden data i datasettet. Både Lærer 1 og Lærer 3 lot elevene lage spill i forbindelse med programmering. Dette faller inn under kompetansemålet «beregne og vurdere sannsynlighet i statistikk og spill». Når man lager spill, møter man på utfordringer knyttet til hvordan informasjonen skal representeres til spilleren. For eksempel når man lager instrukser til hvordan spillet fungerer, må man velge mellom en visuell forklaring, eller tekst. Lærer 2 nevnte ikke å lage spill i undervisningen. Det kan vær at Lærer 2 ser på programmering mer som et praktisk verktøy elevene skal lære seg uavhengig av matematikk. Lærer 2 var ikke fremmed med å la elevene lage programmering som kunne være nyttige, legger muligens mer vekt på det enn å lage spill for å lære matematikk.

5.6 Symbol- og formalisekompetanse

Kompetanse innen symbolbruk og formaliteter knyttet til matematikk kan ikke kategoriseres under et enkelt kjerneelement, men er fordelt over «representasjon og kommunikasjon» og «abstraksjon og generalisering», da det er disse kjerneelementene som tar for seg

symbolbruken. Kilhamn og Bråting (2019) kobler algebra og programmering sammen med å bruke symboler. Lærer 1 hadde brukt programmering en del til å undervise i algebra, og sa at elevene fikk en bedre forståelse av hva en variabel var. Men noen elever kunne bli litt frustrert når de gikk over til å regne på papir. Når de programmerte hadde alltid variablene og bokstavene et tall, en spesifikk verdi, men når de flyttet seg over til papiret, ble det litt mer abstrakt da en bokstav ikke lenger var forbundet med en spesifikk verdi. Noe som kan være utfordrende å kombinere programmering og matematikk, er nettopp den ulike bruken av symbolene. Som Kilhamn og Bråting (2019) tar opp, fungerer likhetstegnet annerledes i programmering og matematikk. Når man programmerer bruker man dette tegnet for å angi en verdi til en variabel, mens i algebra sier et enkelt likhetstegn noe om forholdet. Dette kan tyde på at Lærer 1 gikk fram for fort i programmeringen, før elevene hadde lært det tilsvarende konseptet i matematikken. En annen forklaring kan være at programmeringsoppgavene var for enkle i forhold til matematikken. Lærer 3 opplevde at elevene hadde en mer instrumentell forståelse av programmering, og klarte i mindre grad å tenke veldig selvstendig enda. På grunn av dette, hadde Lærer 3 enda ikke møtt på noe store utfordringer der elevene hadde misforstått den forskjellige symbolbruken mellom programmering og matematikk.

5.7 Kommunikasjonskompetanse

Denne kompetansen tar for seg den andre delen av kjerneelementet «representasjon og kommunikasjon». Niss og Højgaard argumenterer at siden alle former for kommunikasjon i matematikk nødvendigvis bruker en eller annen form for representasjon, er denne kompetansen svært nærliggende representasjonskompetansen (2002, s. 60). Denne type kommunikasjon vil også ofte bruke symboler og formaliteter i matematikken, som gjør at 5.5 Representasjonskompetanse og 5.6 Symbol- og formalisekompetanse henger nøye sammen med kommunikasjonskompetanse. Kompetansemålet om «presentere et utforskende arbeid knyttet til personlig økonomi» er svært relevant i forhold til denne kompetansen. Å kunne presentere og sette ord på sitt eget matematiske arbeid er noe elevene kan få god trening i.

Lærer 2 påpeker at elevene trenger å bli bedre på å kommentere sin egen kode som de leverer inn. Ved å legge inn kommentarer i koden som de skriver, trener de på å formidle ideer og logikken sin på en annen måte. De må sette ord på og forklare hva koden skal gjøre. Men per i dag er det «fritt fram hva folk leverer inn», og så lenge koden fungerer er det bra. Men problemet er at Lærer 2 ikke alltid vet om elevene forstår hva sin egen kode betyr. Og hvis koden ikke kjører, kan kommentarer hjelpe lærere til å se hva eleven hva forstått, og hva eleven har misforstått.

5.8 Hjelpemiddelskompetanse

Denne kompetansen er koblet opp mot kjerneelementet «utforskning og problemløsning», da dette innebærer «å vurdere om delproblemene best kan løses med eller uten digitale verktøy» (Utdanningsdirektoratet, 2019b). Hjelpemiddelskompetansen går ut på å ha kjennskap til og vurdere egnetheten til ulike hjelpemiddel i ulike situasjoner. De fleste kompetansemålene under «utforskning og problemløsning» vil kunne gå under denne kompetansen, da man hele tiden må vurdere hvilke hjelpemiddel som man bruker i møte med problemene man løser. Lærer 1 mente at programmering var et nyttig verktøy elevene kunne får bruk for i livet, og ikke bare et instrument for å lære matematikk. Lærer 2 så også på programmering som et verktøy, mens Lærer 3 var mer utydelig på dette området. Det blir aldri eksplisitt sagt noe om programmering som et verktøy, men Lærer 3 brukte det i forbindelse med praktiske situasjoner som å programmere biler, og er tydelig et verktøy som blir tatt aktivt i bruk i undervisningen.

Det virker som elevene kunne ha fått utviklet denne kompetansen mer i undervisningen. Som jeg oppfattet det, får ikke elevene noe stort valg i hvilke verktøy de skal bruke til å løse bestemte oppgaver. Når de skal lære om geometri, må de gjør det ved programmering, har elevene om økonomi og regnskap, skal det gjøres i regneark, osv. Hadde man lagt opp til at elevene måtte velge om regnskapet skulle føres i regneark eller i Python, kan det vær at de fikk et mer bevisst forhold til å velge verktøy ut fra egenskapene til nevnte verktøy.

5.9 Øvrig drøfting

Et av de mer interessante funnene mine, var den ulike holdningen til økonomi og programmering. Lærer 1 og Lærer 2 var svært positive til å kombinere programmering sammen med kompetansemålene om økonomi, mens Lærer 3 uttalte at økonomi «ikke er **veldig** relevant» til programmering. En forklaring her kan være hva lærerne regner for programmering. Lærer 1 er veldig tydelig når det blir sagt «[…] det er jo en form for programmering hver gang man bruker regneark […]», noe som Lærer 2 også punkterer i sitt intervju. Men Lærer 3 lager ingen slike koblinger mellom regneark og programmering. Dette kan tyde på at Lærer 3 tolker programmering mer som bare koding, at man programmerer ikke hvis man ikke bruker et dedikert programmeringsspråk. Samtidig kommer det fram at Lærer 3 ser på programmering som noe veldig sammensatt av mange forskjellige ferdigheter, blant annet argumenterende, generaliserende og abstraherende. Alle tre lærerne så på programmering som en måte å gjøre matematikken praktisk, enten om det var å styre biler, lage et spill eller skrive et nyttig program man kan bruke i hverdagen.

Lærer 1 og Lærer 2 var veldig tydelige på at programmering ikke var nødvendig for å bli kyndig i matematikk. Man kunne fint lære seg matematikk uten å lære seg å programmere, men det er fremdeles et viktig verktøy å lære seg for å mestre livet i samfunnet framover. Lærer 2 argumenterte at framover vil flere og flere kunne programmere, og programmeringsspråk vil snart kunne konkurrere med engelsk som det mest brukte språket i verden. Dette samsvarer med mye av litteraturen som sier at programmering vil være regnes som en grunnleggende ferdighet om ikke for mange år (Bocconi et al., 2018, s. 1). Lærer 1 tenker at elevene blir bedre samfunnsytere generelt, selv om de ikke ender opp med å bruke programmering direkte i livene sine. Men det blir poengtert at mange kommer til å arbeide i digitale yrker hvor programmering vil være relevant. Lærer 3 har erfart at mange elever strever med føring av matematikk, og at programmering har vært med å skjerpe dem. De lærer seg å strukturere arbeidet sitt. Videre forteller Lærer 3 at programmering ikke er nødvendig, men det er veldig hjelpsomt. Det virker som Lærer 3 har fokuset på selve undervisningen og matematikken generelt, mens Lærer 1 og Lærer 2 ser på det mer i et samfunnsperspektiv.

Kapittel 6 - Avslutning

6.1 Konklusjon

Problemstillingen for denne oppgaven var: «hvilke matematiske kompetanser ønsker enkelte faglærere å fremme gjennom programmering i matematikkundervisning på ungdomsskolen?». Grunnen til at jeg har arbeidet med denne problemstillingen var at programmering har blitt en del av matematikkundervisningen, samtidig som mange lærere ikke føler seg rustet til å bruke dette i undervisningen.

Målet mitt var å høre hvilke matematiske kompetanser som ble fremmet av et lite utvalg lærere som følte seg trygge i programmering. For å gjøre dette, utarbeidet jeg en intervjuguide som ble brukt i møte med tre forskjellige lærere, og analysert resultatet i lys av Niss og Højgaard (2002) sine åtte matematiske kompetanser.

Det kommer fram i intervjuene at alle tre lærerne ser på programmering som egnet eller godt egnet til å lære elevene problemløsningskompetanse. Dette samsvarer Kaufmann og Stenseth (2020), at programmering fører til bedre problemløsningsferdigheter. Utdanningsdirektoratet kobler problemløsning til algoritmisk tenkning (Utdanningsdirektoratet, 2019a), som igjen har sterke tilknytninger til programmering (Wing, 2006). Alle lærerne lærer elevene programmering som et hjelpemidlene elevene kan ha til rådighet. Dette kan komme i form av regneark, tekstbasert programmering eller et grafisk programmeringsspråk. Likevel virker det ikke som elevene lærer å aktivt velge hvilke hjelpemiddel som er mest egnet å bruke i ulike situasjoner. I en undervisningssituasjon bestemmer ofte læreren hvilke verktøy man skal arbeide med for å løse oppgaven man får. Det kan være at elevene kunne ha blitt bedre til å avgjøre hjelpemiddelets grenser og styrker, ved å aktivt måtte ta et valg. Programmering kan være et godt verktøy å lære seg å bruke. Men det kan være vanskelig å balansere matematikken og programmeringen i undervisningen. Hvis målet er at elevene skal lære matematikk, kan programmeringen bli for avansert for enkelte. Og på motsatt side kan matematikken bli for enkel hvis nivået på programmeringen er mer tilpasset elevene.

Det var også enighet blant lærerne om at programmering kunne styrke resonneringskompetansen til elevene. Men en av lærerne hevdet at dette var for avansert for elevene på dette tidspunktet. Det vil si at programmering virker å være godt egnet for å fremme logisk tenkning, men en av lærerne opplevde at elevene trenger mer kompetanse innen matematikk og programmering før det er gjennomførbart. Det virker som programmering i matematikken kan opptre som et tveegget sverd. På den ene siden er

strukturen og den strenge syntaksen med på å hjelpe elever til å få bedre struktur når man regner for hånd. Men på den andre siden kan elever som allerede strever med matematikk, ende opp med å bli enda mer forvirret. Dette kan også henge sammen med lærerens ulike erfaringer rundt programmering og kompetanse innen symbolbruk. Plutselig betyr ikke symbolene og tegnene det samme som når man skriver for hånd, og det blir enda et lag med forvirring og utfordringer.

En annen uenighet som dukket opp, var programmering sin relevans i forhold til økonomi og algebra. Den ene læreren mente at programmering var lite relevant i disse emnene, mens de to andre uttrykte at programmering ofte ble brukt til økonomi og algebra. Denne uenigheten kan stamme fra deres oppfattelse av hva programmering er. Ser man ikke på regneark som programmering, er det ikke naturlig å inkludere økonomi. Å føre regnskap med et programmeringsspråk som for eksempel Python kan være lite hensiktsmessig og mindre naturlig enn å føre det i et regneark.

Alle lærerne plasserte seg forskjellig på skalaen som ble beskrevet i Kapittel 4 - Resultat og analyse. Lærer 2 ønsket ikke å plassere seg på en slik skala, fordi vedkommende mente at læring i programmering ikke trengte å utelukke læring i matematikk og vise versa. Men basert på dette, kan man argumentere for at Lærer 2 kunne plassert seg midt på skalaen. Dette innebærer at programmering og matematikk likestilles i en undervisningstime. Lærer 1 som hadde mest erfaring med programmering, plasserte seg på 4, at matematikk var viktigst å lære, men programmering hadde fremdeles en egen verdi. Lærer 3 hadde minst erfaring med programmering av utvalget, og plasserte seg mellom 7 og 8. Ut fra dette, virker det som lærerens personlige ferdigheter innen programmering påvirker deres syn på kombinasjonen av programmering og matematikk. Er læreren selv kyndig i programmering, øker sannsynligheten for at læreren ønsker å fremme både programmering og matematikk samtidig.

Det er viktig å huske at dette har vært en kvalitativ studie, som ikke gjør det mulig å trekke generelle konklusjoner basert på funnene mine, men vi har fått et innblikk i hvordan dette utvalget av lærere tilnærmer seg programmering i matematikkundervisningen. På grunn av utvalget mitt, kan resultatet være mer positivt mot bruken av programmering i matematikkundervisning enn lærere generelt. Dette er fordi utvalget besto av lærere som allerede var komfortable med å bruke programmering og gjerne hadde erfaringer fra før av.

6.2 Videre forskning

I løpet av prosjektet har jeg støtt på mange ulike spørsmål som hadde vært interessante å undersøke nærmere. Som en forlengelse av dette studiet, kunne det vært verdt å se nærmere på undervisningsopplegget til en rekke lærere og sammenligne hvordan de legger opp undervisningen sin i de ulike temaene i programmering. Dette kunne ha kommet til nytte for lærere som ikke føler seg trygge i å bruke programmering i sin egen undervisning.

Det kan være verdt å ha se nærmere på om programmeringsferdighetene til lærerne påvirker deres tilnærming og holdning til bruken av programmering i undervisningen. Fokuset til en lærer med en instrumentell forståelse av programmering kan gjerne være noe helt annet enn en lærer med en relasjonell forståelse. Det kan også være høyst aktuelt å se hvordan innføringen av programmering, og dermed mer lesing og skriving i matematikkfaget, påvirker elever med lese- og skrivevansker. Å skrive kode krever en pinlig nøyaktighet, noe som kan være en unødvendig barriere for en rekke elever.

6.3 Avsluttende kommentar

Bakgrunnen til dette studiet kommer fra endringer i samfunnet vårt. Digitale ferdigheter og algoritmisk tenkning blir en større del av hverdagen framover (Bocconi et al., 2018). Algoritmisk tenkning er et verktøy som mange kan bruke i arbeidslivet, uavhengig om man programmerer eller ikke (Wing, 2006). Jeg har undersøkt hvilke matematiske kompetanser tre lærere har som intensjon å fremme gjennom å bruke programmering i undervisningen. Gjennom dette arbeidet har jeg blitt inspirert til å bruke programmering i store deler av matematikkundervisningen min, og ser tydeligere sammenhengen mellom de ulike matematiske kompetansene og kjerneelementene i matematikkfaget. Programmering er noe stort, og mye tyder på at vi kommer til å bruke det mer og mer, både i undervisningssammenheng og i arbeidslivet. Håper denne oppgaven også kan være en inspirasjon til andre, slik som den har vært for meg.

Referanser

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A. & Engelhardt, K. (2016). *Developing Computational Thinking in Compulsory Education - Implications for policy and practice.*

Bocconi, S., Chioccariello, A. & Earp, J. (2018). *The Nordic approach to introducing Computational Thinking and programming in compulsory education*. Institute for Educational Technology.

Bocconi, S., Chioccariello, A., Giuliana Dettori, A. F. & Engelhardt, K. (2016). *Developing computational thinking in compulsory education - Implications for policy and practice* JRC science for policy report.

Borge, I. C., Sanne, A., Nortvedt, G. A., Meistad, J. A., Skrindo, K., Ranestad, K., Maugesten, M., Lindstrøm, T. & Kristensen, T. E. (2014). *Matematikk i norsk skole anno 2014 - Faggjennomgang av matematikkfagene - Rapport fra ekstern arbeidsgruppe oppnevnt av Utdanningsdirektoratet*. Utdanningsdirektoratet. https://www.udir.no/tall-og-forskning/finn-forskning/rapporter/Matematikk-i-norsk-skole-anno-2014/

Calder, N. (2018). Using Scratch to facilitate mathematical thinking. *Waikato Journal of Education*, *23*(2), 43-58. https://doi.org/10.15663/wje.v23i2.654

Forsström, S. E. & Kaufmann, O. T. (2018). A Literature Review Exploring the use of Programming in Mathematics Education. *International Journal of Learning, Teaching and Educational Research*, *17*(12), 18-32. https://doi.org/10.26803/ijlter.17.12.2

Gjøvik, Ø. & Torkildsen, H. A. (2019). Algoritmisk tenkning. *Tangenten - tidsskrift for matematikkundervisning*, *30*(3), 7.

Holo, O. E., Kveim, E. N., Lysne, M. S., Taraldsen, L. H. & Haara, F. O. (2022). A review of research on teaching of computer programming in primary school mathematics: moving towards sustainable classrom action. *Education Inquiry*. https://doi.org/https://doi.org/10.1080/20004508.2022.2072575

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. (2019, 2019-02-06). Algebraic thinking in the shadow of programming. I J. Uffe Thomas, H.-P. Marja van den & V. Michiel, [Proceedings of the Eleventh Congress of the European Society for Research in Mathematics

Education (CERME11)]. Eleventh Congress of the European Society for Research in Mathematics Education, Utrecht, Netherlands. https://hal.archives-ouvertes.fr/hal-02429028

Kilpatrick, J., Swafford, J. & Findell, B. (2001). *Adding It Up: Helping Children Learn Mathematics*. National Academy Press.

Kvale, S. & Brinkmann, S. (2015). *Det kvalitative Forskningsintervjue* (T. M. Anderssen & J. Rygge, Overs.; 3. . utg.). Gyldendal.

Nätt, T. H. (2022). *Scratch*. Store norske leksikon. Hentet 28. juni fra https://snl.no/Scratch

Niss, M. & Højgaard, T. J. (2002). *Kompetencer og matematiklæring - Ideer og inspiration til udvikling af matematikundervisning i Danmark* (Uddannelsesstyrelsens temahæfteserie nr. 18 - 2002, Issue. Undervisningsministeriet.

Niss, M. & Højgaard, T. J. (2019). Mathematical competencies revisited. *Educational Studies in Mathematics*, *102*, 20. https://doi.org/https://doi.org/10.1007/s10649-019-09903-9

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

Sevik, K. (2016). *Programmering i skolen*. S. f. I. i. utdanningen.

Shute, V. J., Sun, C. & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, 17.

Utdanningsdirektoratet. (2019a). *Algortimisk tenkning*. Udir. https://www.udir.no/kvalitet-og-kompetanse/digitalisering/algoritmisk-tenkning/

Utdanningsdirektoratet. (2019b). *Kjerneelement*. https://www.udir.no/lk20/mat01-05/om-faget/kjerneelementer

Utdanningsdirektoratet. (2019c). *Læreplan i matemaikk 1.-10. trinn (MAT01-05)*. https://www.udir.no/lk20/mat01-05?lang=nno

Utdanningsdirektoratet. (2019d). *Læreplan i matematikk 1.-10.* https://www.udir.no/lk20/mat01-05

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM*, *49*(3), 3. https://doi.org/10.1145/1118178.1118215

Vedlegg

Vedlegg 1 – Informasjonsskriv og samtykkeerklæring

Vil du delta i forskningsprosjektet

«Programmering og matematisk kompetanse»

Dette er et spørsmål til deg om å delta i et forskningsprosjekt hvor formålet er å studere hvilke matematiske ferdigheter som er ønskelig å fremme gjennom programmering. I dette skrivet gir vi deg informasjon om målene for prosjektet og hva deltakelse vil innebære for deg.

Formål

Dette er en masteroppgave som skal ta for seg hvilke matematiske ferdigheter et lite utvalg lærere ønsker å fremme gjennom bruken av programmering i matematikkundervisningen sin. Målet er gjennom å intervjue matematikklærere, kunne svare på disse forskingsspørsmålene:

1. I hvilken grad brukes programmering som et verktøy i for å lære matematikk?
2. Hvilke holdninger har matematikklærere til om programmering er nyttig i undervisningen på ungdomstrinnet?
3. Hva slags matematiske kompetanser ser læreren som mest relevante i forhold til programmering?

.

Hvem er ansvarlig for forskningsprosjektet?

NLA Høgskolen er ansvarlig for prosjektet.

Hvorfor får du spørsmål om å delta?

Du har fått spørsmål om å delta på grunn av din stilling matematikklærer og erfaring med å bruke programmering som en del av matematikkundervisningen.

Hva innebærer det for deg å delta?

Hvis du velger å delta i prosjektet, vil det innebære et intervju på ca. 1 time. Intervjuet kommer til å handle om din erfaring som matematikklærer og dine holdninger og tanker rundt bruken av programmering som en del av matematikkundervisningen. Intervjuet vil være semistrukturert, som vil si at spørsmålene ikke er helt faste, selv om temaet er bestemt.

Opplysningene som samles inn i løpet av intervjuet vil registreres gjennom notater og lydopptak.

Det er frivillig å delta
Det er frivillig å delta i prosjektet. Hvis du velger å delta, kan du når som helst trekke samtykket tilbake uten å oppgi noen grunn. Alle dine personopplysninger vil da bli slettet. Det vil ikke ha noen negative konsekvenser for deg hvis du ikke vil delta eller senere velger å trekke deg.

Ditt personvern – hvordan vi oppbevarer og bruker dine opplysninger

Vi vil bare bruke opplysningene om deg til formålene vi har fortalt om i dette skrivet. Vi behandler opplysningene konfidensielt og i samsvar med personvernregelverket. Det er bare masterstudenten og veilederen hans som vil ha tilgang til informasjonen som blir samlet inn. I tillegg vil navnet ditt og dine kontaktopplysninger erstattes med en kode og lagres adskilt fra øvrige dokumenter. Du vil ikke være i stand til å gjenkjenne deg i publikasjonen.

Hva skjer med personopplysningene dine når forskningsprosjektet avsluttes?
Prosjektet vil etter planen avsluttes 22. mai, 2023. Etter prosjektslutt vil datamaterialet med dine personopplysninger anonymiseres. Dette skjer ved at lydfiler og koblingsnøkkel blir slettet.

Hva gir oss rett til å behandle personopplysninger om deg?
Vi behandler opplysninger om deg basert på ditt samtykke.

På oppdrag fra NLA Høgskolen har Personverntjenester vurdert at behandlingen av personopplysninger i dette prosjektet er i samsvar med personvernregelverket.

Dine rettigheter
Så lenge du kan identifiseres i datamaterialet, har du rett til:

- innsyn i hvilke opplysninger vi behandler om deg, og å få utlevert en kopi av opplysningene
- å få rettet opplysninger om deg som er feil eller misvisende
- å få slettet personopplysninger om deg
- å sende klage til Datatilsynet om behandlingen av dine personopplysninger

Hvis du har spørsmål til studien, eller ønsker å vite mer om eller benytte deg av dine rettigheter, ta kontakt med:

- Masterstudent ved NLA Høgskolen, Tommy Odeh
  E-post: tommy.odeh@gmail.com
  Tlf: 99 38 67 29

- Veileder, Torbjørn Aadland
  E-post: Torbjorn.Aadland@nla.no
  Tlf: 47 23 20 90

- Vårt personvernombud, Inger-Johanne Gamlem Njau
  E-post: personvernombud@nla.no
  Tlf: 55 54 07 49

Hvis du har spørsmål knyttet til Personverntjenester sin vurdering av prosjektet, kan du ta kontakt med:

- Personverntjenester på epost (personverntjenester@sikt.no) eller på telefon: 53 21 15 00.

Med vennlig hilsen
Høgskolelektor Torbjørn Aadland Masterstudent Tommy Odeh

Samtykkeerklæring

Jeg har mottatt og forstått informasjon om prosjektet «Programmering og matematisk kompetanse», og har fått anledning til å stille spørsmål. Jeg samtykker til:

- ☐ å delta i intervju
- ☐ at det blir tatt lydopptak

Jeg samtykker til at mine opplysninger behandles frem til prosjektet er avsluttet

---

(Signert av prosjektdeltaker, dato)

Vedlegg 2 – Intervjuguide

Problemstilling: «Hvilke matematiske kompetanser ønsker enkelte faglærere å fremme gjennom programmering i matematikkundervisning på ungdomsskolen?»

Forskningsspørsmål

1. I hvilken grad brukes programmering som et verktøy for å lære matematikk?
2. Hvilke holdninger har matematikklærere til om programmering er nyttig i undervisningen på ungdomstrinnet?
3. Hva slags matematiske kompetanser ser læreren som mest relevante i forhold til programmering?

Introduksjon til intervju

- Presentere kort masteroppgaven og hensikten bak intervjuet
- Forklare personvern og anonymitet
- Sørge for at informanten har forstått informasjonsskrivet

Innledende spørsmål

- Hvor lenge har du undervist i matematikk?
- Når begynte du å ta i bruk programmering i undervisningen?
- Hvordan har du lært deg å programmere?
- Bruker du noen spesielle læreverk?
- Hva slags språk og utstyr bruker dere når elevene skal programmere?

Hovedspørsmål

- Hvorfor begynte du å ta i bruk programmering i undervisningen?
    - Stemmer dette enda i dag?
- Hvilke matematiske kompetanser har du som mål å fremme når programmering blir brukt i matematikk-undervisningen?
- Hvilke av kjerneelementene er mest relevante i forhold til å bruke programmering i undervisningen?
    - Hvorfor? Hvorfor ikke de andre?
- Er det noen spesifikke kompetansemål i matematikkundervisningen du tenker programmering egner seg spesielt godt til å brukes i?
    - Hvorfor akkurat disse, og ikke andre?
- Er det nødvendig for elever å kunne programmere for å bli kyndige i matematikk?
- Hva er viktigst for deg at elevene sitter igjen med etter undervisningen, kompetanse i matematikk, eller kompetanse i programmering? Ranger dette på en skala fra 1-10 hvor 1 er bare matematikk, og 10 er bare programmering.
    - Hvorfor?
- Hvilket utbytte får elevene ut av programmering i timene?

    - Kunne man ha fått samme utbytte gjennom en annen undervisningsmetode som ikke innebærer programmering?
- Kan du fortelle om en undervisningstime du husker godt hvor programmering ble tatt i bruk?
    - Hva var formålet med dette undervisningsopplegget?
    - Hvilke tema hadde dere om?
    - Hva ble læringsutbyttet til elevene?
- Hva slags tanker, hvis noen, har du om algoritmisk tenkning og dens relasjon til programmering i matematikk?
- Hvilke utfordringer ser du ved å bruke programmering i undervisningen?
    - Har du noen eksempler på dette?
- Noe du til slutt ønsker å kommentere eller si?