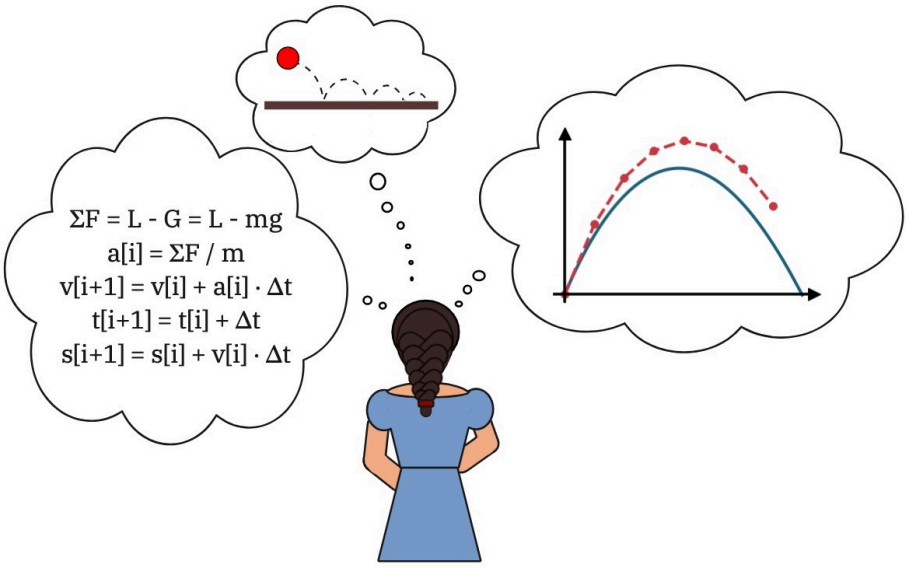
Masteroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for fysikk

Johanna Xiaoli Skøien

Numeriske metoder og programmering i fysikk 1

Masteroppgave i Lektorutdanning i realfag
Veileder: Berit Bungum
Medveileder: Trond Morten Thorseth
Mai 2023



Illustrasjon: Johanna Xiaoli Skøien

NTNU
Kunnskap for en bedre verden

Johanna Xiaoli Skøien

Numeriske metoder og programmering i fysikk 1

Masteroppgave i Lektorutdanning i realfag
Veileder: Berit Bungum
Medveileder: Trond Morten Thorseth
Mai 2023

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for fysikk

NTNU
Kunnskap for en bedre verden

SAMMENDRAG

Med fagfornyelsen ble det vedtatt at norske fysikkelever skal arbeide mer digitalt. Denne digitaliseringen kommer spesielt til syne i fysikk 1 gjennom kompetansemålet: *«... eleven skal kunne bruke numeriske metoder og programmering til å modellere og utforske bevegelse i situasjoner der akselerasjonen ikke er konstant»*. Kompetansemålet krever at et helt nytt faginnhold, uten særlig tradisjon i skolen, innlemmes i fysikkfaget. Det er ikke entydig hva det å «*kunne bruke numeriske metoder og programmering*» innebærer, noe som medfører at kompetansemålet er nokså åpent for tolkning. Formålet med denne oppgaven har derfor vært å utvikle et undervisningsopplegg som kan imøtekomme kompetansemålet om numeriske metoder og programmering i fysikk 1. Forskningsarbeidet har i hovedsak vært kvalitativ, og inneholder aspekter av designbasert forskning. Studiens datamateriale er firedelt og består av elevbesvarelser på oppgaveark, elevbesvarelser på spørreskjema, lydopptak fra elevsamtaler fra timen og lydopptak fra gruppeintervjuer, som har blitt analysert tematisk.

Resultatene viser at numeriske metoder og programmering fint kan inngå som en naturlig og konstruktiv del av undervisningen i fysikk 1. Både forskere og ingeniører bruker programmering til å gjøre numeriske beregninger. Å la elevene bli kjent med denne arbeidsmetoden, kan dermed bidra til å gjøre faget mer autentisk og føre til at elevene får en dypere fysikkforståelse. Arbeidsmetoden åpner også opp for at elevene kan arbeide med virkelighetsnære problemer, som ikke lar seg løse analytisk. Utprøvingen av undervisningsopplegget viser at elevene, uavhengig av programmeringserfaring, i stor grad forstår prinsippene bak numeriske metoder. Elevene sliter imidlertid med å oversette numeriske metoder til programkode. Årsaken til dette virker i hovedsak å være at det kan være utfordrende for elevene å overføre programmeringen de har lært i matematikken til å løse problemer i fysikkfaget. Studien viser også at det er fare for kognitiv overbelastning dersom det ikke tas hensyn til at læring er situert.

ABSTRACT

As decided by the Norwegian government, students will be introduced to numerical methods and programming in physics courses in upper secondary level. One of the curriculum objectives in physics indicates that students should be able to use these methods to model and explore motions with a non-constant acceleration. In order to satisfy the curriculum objective, a completely new subject matter must be incorporated into the physics curriculum. There is also no clear explanation of what being able to useentails. As a consequence, the curriculum objective seems to be rather ambiguous. The purpose of this thesis is to develop a good and well considered lesson plan that satisfies the benchmark regarding numerical methods and programming. The research has been a predominantly qualitative procedure, and includes aspects of design-based research. The data material consists of student answers on worksheets, student answers on a survey, audio recordings of student discussions from class and audio recordings of group interviews, which have been thematically analyzed.

The results imply that numerical methods and programming can be implemented in physics courses in a natural and constructive manner. By introducing the students to numerical methods and programming in a way that reflects how it is used by scientists and engineers, the physics course becomes more authentic. As a result students can be assigned real life problems that can not be solved analytically, which they may find more interesting and appealing. The results show that the students, regardless of their programming experience, understand the principles behind numerical methods. The students did however struggle to put the numerical method into code that could be interpreted by a computer. The main reason for this seems to be that the students find it hard to use the programming skills they learned in mathematics to solve problems in physics courses. The results also show that if the situated learning model is not considered, there is a chance of cognitive overload.

FORORD

Denne masteroppgaven markerer slutten på årene som lektorstudent i fysikk og matematikk ved NTNU. I løpet av de siste fem årene har jeg tilegnet meg en god porsjon kunnskap om fysikk, matematikk, didaktikk og pedagogikk. Jeg føler meg likevel på ingen måte utlært. Tvert om har jeg blitt oppmerksom på det jeg ikke kan. Arbeidet med denne masteroppgaven har også vært omtrent sånn. Selv om studietiden har inneholdt mange arbeidstimer og mye frustrasjon, er det i løpet av denne perioden jeg har fått flere av mine beste venner. Uten dem hadde studietiden ikke vært den samme.

Jeg vil rette en spesielt stor takk til veilederen min Berit Bungum som har hjulpet meg med idé til masteroppgaven, gitt meg verdifulle råd og tilbakemeldinger, og har tatt seg mer tid til å lese og diskutere oppgaven enn det noen kunne forvente. Jeg har følt meg svært godt ivaretatt under hele arbeidet med masteroppgaven. Takk også til Trond Morten Thorseth, som har vært medveileder og bidratt med programmeringstekniske råd, og hjulpet meg med å lage animasjonene i forbindelse med undervisningsopplegget. Læreren som «lånte» meg fysikklassen sin, og elevene som deltok i studien, fortjener også en takk. Takk til min roomie Juni Loennechen Magnussen som har vært en ressurs i datainnsamlingsprosessen og en super sparringspartner. Min siste takk rettes til familien min og Thomas for å ha hatt troen på meg hele veien, og som har stilt opp som både korrekturlesere og illustratør.

Med dette avslutter jeg min tid som lektorstudent, og trer inn i de voksnes rekker som fysikklærer. Jeg gleder meg til å skape nysgjerrighet og lærelyst, på tilsvarende måte som mine egne fysikklærere gjorde.

Trondheim, mai 2023

Johanna Xiaoli Skøien

INNHOLD

1 INTRODUKSJON

1.1 Bakgrunn

I takt med den teknologiske utviklingen, har digitale enheter blitt en stadig større del av livene våre. Vi omgir oss på daglig basis med en mengde digitale «duppeditter» som smarttelefoner, datamaskiner og nettbrett. Den teknologiske utviklingen har også gjort at numeriske metoder og programmering har blitt en naturlig del av fysikken. De fysiske beregningene er ikke lenger forbeholdt papir og tavler, men foregår også digitalt gjennom programkode på datamaskiner. I dag kan man gjøre store og tunge beregninger på en bærbar datamaskin med bare noen få tastetrykk, noe som tidligere krevde en hel del menneskelig arbeidskraft og datamaskiner i mammut-skala (Bowler & Morus, 2020). Det digitale har vært med på å utvikle samfunnet, og har dermed sterke røtter i dagens samfunn.

Både fysikere og ingeniører utnytter datamaskiners enorme regnekraft til å modellere fysiske fenomener, ved bruk av numeriske metoder og programmering (Malthe-Sørenssen et al., 2015). En byggingeniør kan eksempelvis modellere og simulere belastningen en bro utsettes for og dermed finne konstruksjonens svakeste punkt. Dette er informasjon som er svært verdifull når man skal avgjøre hvilket materiale broen skal lages av, og hvordan den skal dimensjoneres for å best hindre brudd eller knekking. Numeriske metoder og programmering som et nyttig og grunnleggende verktøy i forskning og industri har tidligere vært lite gjenspeilt i norsk fysikkundervisning (Utdanningsdirektoratet, 2006). Gjennom følgende kompetansemål i den nye læreplanen (LK20), har numerikk og programmering fått en plass i fysikkfaget:

> «*...eleven skal kunne bruke numeriske metoder og programmering til å modellere og utforske bevegelse i situasjoner der akselerasjonen ikke er konstant.*»
>
> (Utdanningsdirektoratet, 2021)

Undervisningsopplegget utviklet i denne masteroppgaven skal møte dette kompetansemålet.

1.2 Studiens formål og struktur

Kompetansemålet om numeriske metoder og programmering innebærer at et helt nytt fagstoff, uten særlig tradisjon i skolen, innlemmes i fysikk 1. Akkurat hvordan kompetansemålet skal tolkes, hva det innebærer og hvordan det kan inngå som en konstruktiv del av fysikkfaget, virker å være nokså åpent for tolkning. Mange fysikklærere kan av den grunn oppleve kompetansemålet som uklart, og dermed utfordrende å implementere fagstoffet i undervisningen. Å innlemme numeriske metoder og programmering i fysikkfaget har potensial til å endre faget med tanke på innhold, arbeidsmåter og oppgavetyper (Haraldsrud & Tellefsen, 2018). Selv om lærerne sannsynligvis har varierende erfaring med numeriske metoder og programmering, er de alle bundet av kunnskapsløftet til å implementere temaet i undervisningen.

Studien er ment å være en ressurs for lærere som skal undervise om numeriske metoder og programmering, og et eksempel på hvordan kompetansemålet *kan* tolkes og hvordan dette kan gjennomføres i praksis. Denne oppgaven er ment som et bidrag til et forskningsfelt som fortsatt er relativt nytt og i utvikling, og omfatter et utformet undervisningsopplegg, basert på relevant faglig og didaktisk teori, som har blitt gjennomført og i etterkant evaluert. Her har dataene fra elevene hatt en sentral rolle. Håpet er at oppgaven kan være et eksempel på hvordan numeriske metoder og programmering *kan* implementeres på en hensiktsmessig måte for å fremme fysikkforståelse. Oppgaven er avgrenset til hvordan numeriske metoder og programmering kan inngå som en naturlig del av undervisningen i mekanikk, men belyser i liten grad hvordan dette kan brukes i forbindelse med andre fysikktemaer som astrofysikk, termofysikk eller elektrisitetslære.

Oppgaven er avgrenset til utvikling og testing av et undervisningsopplegg, med et innhold som fortsatt er ganske nytt i fysikkfaget. Av den grunn bør det forskes mer på hva kompetansemålet kan innebære, og hvordan det kan imøtekommes i klasserommet. Denne studien er mitt bidrag til fagfeltet. Studien har aspekter av designbasert forskning og inkluderer metoder som forskning og utviklingsarbeid, og er forankret i en reell klasseromskontekst med en sterk kobling til praksis. Håpet er at oppgaven kan inngå som én av mange iterasjoner i en lengre prosess om god undervisningsforskning (Anderson & Shattuck, 2012).

Oppgavens struktur er firedelt. Den første delen presenterer relevant faglig og didaktisk litteratur (kapittel 2), som munner ut i en rekke konkrete prinsipper for hvordan undervisning

om numeriske metoder og programmering kan foregå (kapittel 3), og er brukt som grunnmur i utviklingen av undervisningsopplegget. Den andre delen omhandler undervisningsopplegget (kapittel 4), der valgene som har blitt gjort begrunnes. Videre beskrives utprøvingen av opplegget, og jeg deler erfaringer fra klasserommet. Den tredje delen gir leseren innblikk i studiens forskningsdesign, og det redegjøres for utvalg og datainnsamlingsmetodene som er brukt, med tilhørende litteratur, samt hvilke erfaringer jeg sitter igjen med (kapittel 5). Her gis en grundig beskrivelse av hvordan studiens datamateriale er analysert. Den fjerde delen presenterer resultatene fra analysearbeidet (kapittel 6). Jeg har valgt å inkludere relativt store mengder kvalitative data for å styrke studienes troverdighet, noe som nødvendigvis tar en del plass. Følgelig er oppgaven nokså voluminøs. Avslutningsvis diskuteres studiens resultater på bakgrunn av relevant teori (kapittel 7). Basert på dette kommer jeg også med anbefalinger til fysikklærere og læreplanforfattere, i forbindelse med undervisning om numeriske metoder og programmering i fysikkfaget.

1.3 Problemstilling og forskningsspørsmål

Oppgaven undersøker problemstillingen om hvordan numeriske metoder og programmering kan inngå som en konstruktiv del av fysikk 1. Dette gjøres ved å besvare følgende forskningsspørsmål i tilknytning til studiens utviklingsarbeid og gjennomføring:

1. Hvordan fungerer det utformede undervisningsopplegget, med tanke på gjennomføring i klasserommet, elevenes utbytte og utfordringer?
2. Hva er elevenes læringsopplevelse av å arbeide med numeriske metoder og programmering?

I tillegg diskuterer jeg veien videre gjennom forskningsspørsmålet:

3. Hvilke dilemmaer oppstår i forbindelse med gjennomføringen av det utformede undervisningsopplegget om numeriske metoder og programmering?

2 TEORETISK FORANKRING

I dette kapittelet vil jeg gjennomgå både faglige og didaktiske komponenter, som bakgrunn for undervisningsopplegget og utprøvingen. Undervisningsopplegget handler om numeriske metoder og programmering, som med fagfornyelsen 2020 ble del av fysikk 1 gjennom kompetansemålet:

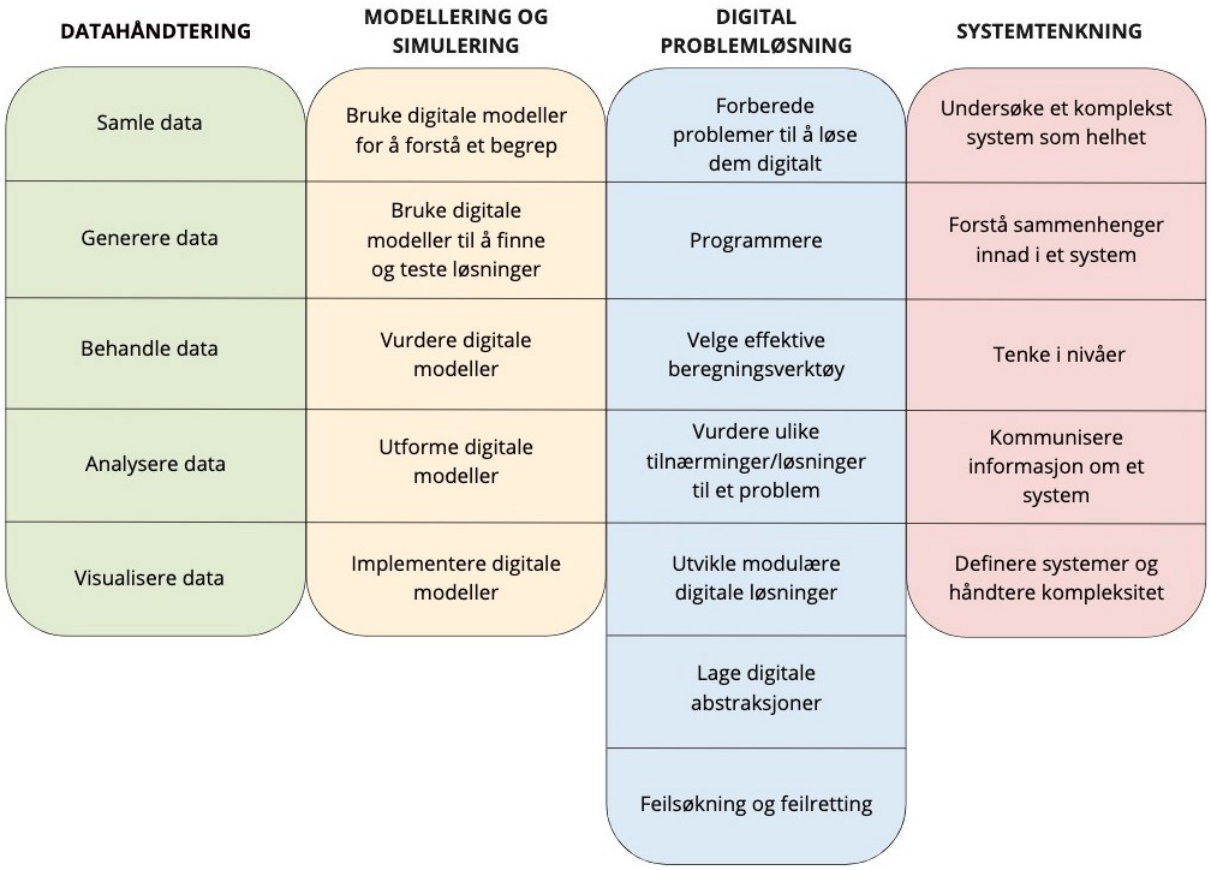
> «*...eleven skal kunne bruke numeriske metoder og programmering til å modellere og utforske bevegelse i situasjoner der akselerasjonen ikke er konstant.*»
>
> (Utdanningsdirektoratet, 2021)

Numeriske metoder og programmering er fortsatt et relativt nytt faginnhold i fysikkfaget, og det finnes derfor lite erfaring med å undervise om dette i skolen. Dette kapittelet presenterer litteratur om numeriske metoder og programmering, og ulike momenter ved å undervise om dette. Momentene munner ut i en håndfull konkrete prinsipper for hvordan undervisningen kan foregå. Dette presenteres i kapittel 3, og har fungert som holdepunkter i utformingen av undervisningsopplegget. Kapittelet innledes med at jeg gjør rede for begrepene numeriske metoder og programmering, og hva de kan innebære i fysikk 1, kapittel 2.1. Her presenteres også funnene fra en intervju-undersøkelse om fysikklæreres refleksjoner rundt de to temaene. Undersøkelsen ble gjort som et forprosjekt til masteroppgaven i emnet *RFEL3100 Forskningsmetoder i matematikk- og realfagsdidaktikk*, og har bidratt til utformingen av undervisningsforløpet. I kapittel 2.2 presenteres fysikkens ulike representasjonsformer. Søkelyset i kapittel 2.3 rettes mot begrepene modeller og modellering, og hva de innebærer. Her får leseren også innblikk i tidligere forskning om numeriske tilnærminger i fysikk 1. Kapittel 2.4 belyser at læring er situert, og presenterer leseren for begrepet kognitiv overbelastning, og hvorfor dette bør unngås. Avslutningsvis i kapittel 2.5 beskrives LIST-prinsippet, om hvordan undervisning kan tilpasses alle elever. Til sammen utgjør de ulike momentene forskjellige, men viktige, aspekter som jeg har valgt å bygge undervisningsopplegget på.

2.1 Numeriske metoder og programmering

Programmering omfatter det å utarbeide og implementere tydelige instruksjoner som en datamaskin kan tolke for å løse et gitt problem. Dermed innebærer programmering både å skrive programkode og å kunne bryte ned problemer i mindre, løsbare deler, drive feilsøk, tilpasse og videreutvikle programkode (Sanne et al., 2016, s. 18). Programmering foregår på ulike programmeringsspråk, som kan være blokk-basert eller tekstbasert. De yngste barna, typisk barnetrinnet, introduseres gjerne for programmering gjennom blokk-baserte programmeringsspråk som Scratch. Her arbeider de med blokker som kan settes sammen til større program, der de kan lage spill eller animasjoner. Programkoden settes altså sammen som et puslespill, der enkelte blokker passer sammen, mens andre ikke passer. Blokkenes kompatibilitet er forutbestemt av logikk. Et slikt programmeringsspråk gir brukeren øvelse i å programmere, uten å måtte forholde seg til syntaksfeil og eventuelle andre feilmeldinger. Med det nye kompetansemålet i fysikk 1 om numeriske metoder, er det naturlig at programmeringen foregår på et tekstbasert språk, da det innehar funksjonaliteten elevene behøver på dette faglige nivået, og fordi det ligger nærmere hvordan fysikere arbeider. Et tekstbasert programmeringsspråk som er nærliggende å benytte er Python. Her er hver kodelinje tekst som må oppfylle strenge krav, syntaks, for at datamaskinen skal forstå instruksene den blir gitt. Dette betyr at selv om elevene har lært programmering tidligere, har de ikke nødvendigvis erfaring med tekstbasert programmering.

I forbindelse med programmering er det naturlig å se nærmere på begrepet «computational thinking» (heretter referert til som CT). Wing (2006) sin definisjon tar utgangspunkt i datavitenskapens fundamentale prinsipper. Ved å bygge på disse prinsippene kan vi mennesker løse problemer, designe systemer og bedre forstå egen adferd (s. 33). Dermed er ikke CT forbeholdt datamaskinen, men det er også en tenkemåte blant mennesker. En lignende definisjon av begrepet finner vi hos Grover og Pea (2013), der CT forklares som det å tenke som en informatiker i møte med et problem (s. 39). Dermed kan CT ansees som en ferdighet samfunnsborgeren burde mestre på lik linje med andre grunnleggende ferdigheter som lesing og skriving (Wing, 2006, s. 33). I USA ble ferdighetene tilknyttet CT ansett som såpass viktige at det nasjonale forskningsrådet la det inn som del av grunnopplæringen i naturfag (NRC, 2012). I forbindelse med dette utviklet Weintrop et al. (2016) en taksonomi for CT-praksiser, med hovedfokus på hvordan det kan anvendes i klasserommet (se Figur 2-1 på neste side).



Figur 2-1: En modifisert gjengivelse av Nordbys (2019, s. 28) norske oversettelse av taksonomien for «computational thinking» presentert av Weintrop et al. (2016, s. 135).

«Computational thinking» har også fått plass i norsk skole, men omtales her som «algoritmisk tenkning». «Algoritmisk tenkning» brukes gjerne i forbindelse med programmering, og beskrives av Utdanningsdirektoratet (2019) som det å bryte ned et større problem i mindre, løsbare delproblemer. Dermed utgjør Utdanningsdirektoratets beskrivelse av «algoritmisk tenkning» kun en liten del av taksonomien presentert i Figur 2-1 (Vinnervik & Bungum, 2022, s. 385). Følgelig faller viktige aspekter, som databehandling og vurdering av digitale modeller, vekk i den norske oversettelsen (Nordby, 2019, s. 24). Selv om «algoritmisk tenkning» har en betydelig snevrere definisjon enn CT, er det dette begrepet som brukes av Utdanningsdirektoratet, og det er derfor denne definisjonen jeg vil benytte meg av videre.

I henhold til det nye kompetansemålet i fysikk, skal elevene kunne modellere bevegelser med ikke-konstant akselerasjon ved hjelp av programmering. Videre inkluderer kompetansemålet begrepet «numeriske metoder» som er nært beslektet med programmering, men ikke identisk. Mens programmering ifølge Figur 2-1 bare utgjør en bit av CT, rommer numeriske metoder

en større del av taksonomien, med hovedvekt innen digital problemløsning. I forbindelse med numeriske metoder, visualiseres gjerne data som posisjon, fart eller akselerasjon grafisk. På den måten inkluderer numeriske metoder et aspekt av datahåndtering (se Figur 2-1). For å kunne løse et problem numerisk, må problemet stykkes opp i mindre, løsbare deler ved å tenke i ulike nivåer. Siden numeriske beregninger gjerne utføres ved hjelp av programmering og datakraft, kan numeriske metoder også plasseres innenfor digital problemløsning i Figur 2-1. For at problemer skal kunne løses digitalt, må de forberedes slik at de kan forstås av en datamaskin. Dette gjøres ved å utarbeide tydelige og løsbare instruksjoner, noe som gjerne krever flere forsøk med ulike løsninger, feilsøking og feilretting.

Dahlquist og Björck (2003) beskriver numeriske metoder som relativt enkle ideer, kombinert på en snedig måte, for å løse et gitt problem (s. 1). Her benyttes numeriske approksimasjoner der arbeidsenheten er konkrete tallverdier, fremfor symboler som i analytiske metoder. Mens analytiske løsninger er eksakte, er numeriske løsninger approksimerte, og derfor mindre nøyaktige. Graden av nøyaktighet ligger i antall desimaler gitt i løsningen (Dahlquist & Björck, 2003, s. 2; Sauer, 2012). Eksempelvis er den analytiske løsningen av $\frac{2}{3}$ nettopp $\frac{2}{3}$, mens den numeriske løsningen er $0,67$ med to desimalers nøyaktighet og $0,6667$ med fire desimalers nøyaktighet. Her ser vi at hvordan vi forholder oss til problemet, altså hvor mange desimaler man ønsker, gir ulike løsninger. Desto flere desimaler i den numeriske løsningen, desto nærmere er man den analytiske, eksakte løsningen. Om det er ønskelig med to eller fire desimalers nøyaktighet, avhenger av hva resultatet skal brukes til, og er noe problemløseren selv må vurdere. En slik beskrivelse av numeriske metoder ligger tett opptil det som i taksonomien til Wientrop et al. (2016) omtales som å «vurdere ulike tilnærminger/løsninger til et problem».

Numeriske metoder er altså ikke ekvivalent med programmering. Dermed kan problemer løses numerisk, uten en datamaskin og programkode. Tilsvarende vil det å lage en nettside eller en enkel kalkulator i Python kunne kategoriseres som programmering, men ikke nødvendigvis som en numerisk metode. Numeriske metoder innebærer ofte det som kalles suksessive approksimasjoner (også kalt iterasjoner), simuleringer eller estimater for å løse den matematiske kjernen av praktiske problemer (Dahlquist & Björck, 2003, s. 2; Lorentz & Roşca, 2020, s. ix; Sauer, 2012). Ofte skal mye data behandles, og det medfører en så stor arbeidsmengde at det er hensiktsmessig å la en datamaskin utføre de numeriske beregningene ved hjelp av programkode. Imidlertid kan numeriske metoder også utføres med penn og papir.

Altså er numeriske metoder heller en fremgangsmåte enn noe som må gjøres på et bestemt format.

Numeriske metoder er en naturlig del av fysikken, og brukes både i forskning og industri (Malthe-Sørenssen et al., 2015, s. 303 - 310). Ved å inkludere numeriske metoder i fysikkfaget, og la elevene bli kjent med dem, vil faget gjøres mer autentisk. Numeriske metoder kan eksempelvis anvendes i læren om mekanikk, slik at elevene kan arbeide med oppgaver der luftmotstanden virker og akselerasjonen ikke lenger er konstant. Formålet med numeriske metoder i skolen er altså tosidig: På den ene side kan man behandle flere og mer komplekse situasjoner, og på den annen side kan man jobbe på en måte som bedre speiler hvordan fysikere arbeider i yrkeslivet, som påpekt av Malthe-Sørenssen et al. (2015).

Eulers metode

Akkurat hva numeriske metoder innebærer presiseres ikke i læreplanen. Jeg har derfor valgt å fokusere på Eulers metode som numerisk metode i mekanikk, da den egner seg for elevenes faglige nivå i fysikk 1. Den kommende beskrivelsen av Eulers metode baserer seg på det Sauer (2012, s. 283 - 287) skriver i læreboken «Numerical Analysis». Metoden går ut på å ta utgangspunkt i et startpunkt (t_0, s_0), der t_0 er den initielle tiden og s_0 er den initielle posisjonen, og å følge stigningstallet i det gitte punktet. Etter å ha fulgt stigningstallets spesifiserte retning i et kort intervall, revurderes stigningstallet og retningen i det nye punktet (t_1, s_1). Eulers metode kan også brukes på bevegelseslikningene fra mekanikken, til å finne posisjonen s_1 etter et kort tidssteg dt dersom både farten v_0 og startposisjonen s_0 er kjent. Da er $s_1 = s_0 + v_0 \cdot dt$. Ved å gjenta denne iterative prosessen mange nok ganger, har vi tilstrekkelig med posisjonsdata til å kunne beskrive en bevegelse. I situasjoner der vi kjenner til hvilke krefter som virker på et legeme, som for eksempel en ball i et vertikalt kast med kjent luftmotstand L, kan vi på tilsvarende måte regne ut posisjon, fart og akselerasjon på følgende måte (se neste side):

$$\Sigma F = L - G = L - mg$$

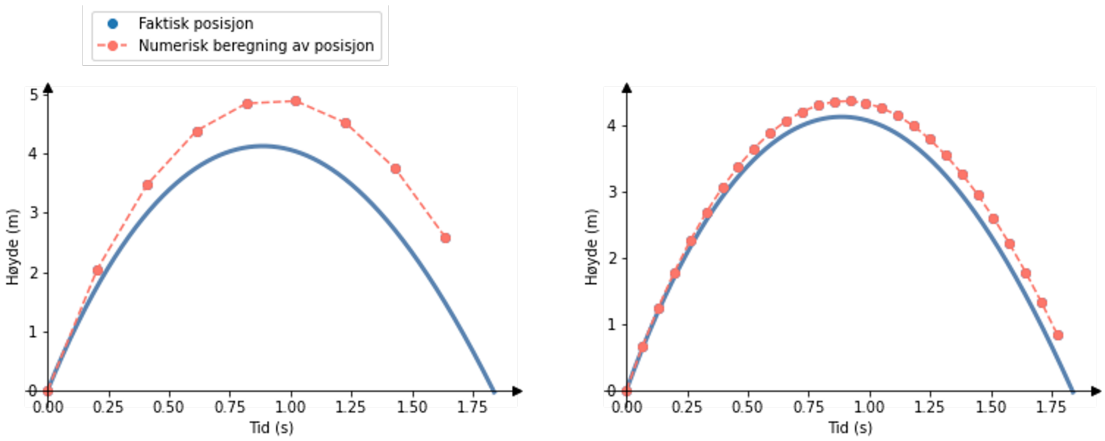
$$a_i = \frac{\Sigma F}{m}$$

$$v_{i+1} = v_i + a_i \cdot dt$$

$$t_{i+1} = t_i + dt$$

$$s_{i+1} = s_i + v_i \cdot dt$$

Figur 2-2 viser de numeriske beregningene av posisjon (stiplet linje) med Eulers metode, sammen med den faktiske posisjonsgrafen (heltrukken linje). Det er tydelig at den numeriske posisjonsgrafen avviker mindre fra den faktiske posisjonsgrafen i bildet til høyre, der tidssteget $dt = 0,1$, enn på bildet til venstre der $dt = 0,5$. Dette skyldes at desto kortere tidssteget dt er, desto nærmere vil de numeriske beregningene være de faktiske posisjonsdataene og den faktiske bevegelsen.



Figur 2-2: Numerisk beregning av posisjonsdata ved hjelp av Eulers metode med ulik steglengde dt. På bildet til venstre er tidssteget $dt = 0,5$, og på bildet til høyre er $dt = 0,1$.

I en mer hverdagslig setting vil Eulers metode minne om når man finner veien ved hjelp av Google Maps, og man forsøker å følge den oppgitte ruta. Følges riktig rute, vil retningspilen på kartet, som indikerer ens retning, sammenfalle med den oppmerkede veien på kartet. Om man er usikker på veien videre, kan man rotere og se hvordan pilspissen endrer retning, og deretter bestemme retningen som gjør at pilspissen passer best med ruta. Forenklet kan vi si at man for hver 100 meter (eller kvartal) peiler seg inn på en retning basert på hvor man er på kartet, går nye 100 meter i den gitte retningen, og gjentar prosessen til hele ruta er gått.

I denne analogien representerer hver 100 meter et nytt punkt. En fysikk 1- elev vil kanskje oppleve Sauers (2012) forklaring av Eulers metode som abstrakt, og dermed ha større vansker med å forstå den enn eksempelvis en masterstudent i fysikk. Dermed kan en forklaring som tar utgangspunkt i elevenes egne hverdagserfaringer være nyttig for at de skal forstå hva Eulers metode innebærer. Imidlertid er det viktig å være klar over at analogien til Google Maps er mindre presis enn lærebokforklaringen presentert av Sauer i «Numerical Analysis».

I henhold til kompetansemålet om numeriske metoder og programmering, er det naturlig at elevene utfører Eulers metode på datamaskiner ved hjelp av programmering. Som beskrevet tidligere er Eulers metode en iterativ algoritme der den nåværende posisjonen ble bestemt av den foregående, og den neste posisjonen bestemmes av den nåværende posisjonen. Denne fremgangsmåten følger samme logikk som *løkker* i programmering. Her utføres de samme kommandoene enten et gitt antall ganger, eller frem til en viss betingelse er oppfylt. Dette kalles henholdsvis en *for*- eller *while-løkke*. Følgelig vil det være en naturlig del av undervisningen at elevene får arbeidet med grunnleggende programmeringsferdigheter med *løkker* før de lærer og benytter Eulers metode.

Fysikklæreres syn på numeriske metoder og programmering

Som forprosjekt til masteroppgaven gjennomførte jeg en intervju-undersøkelse med tre fysikklærere ved en alminnelig norsk videregående skole (Skøien, 2022). Intervju-undersøkelsen handlet om lærernes refleksjoner rundt begrepet numeriske metoder, og hva det kan innebære i fysikk. I det følgende presenteres undersøkelsen og dens hovedfunn.

Lærerne som deltok i intervjuundersøkelsen arbeider alle ved den samme skolen som er brukt utprøvingen av opplegget. Utrøvingen ble også gjennomført i fysikklassen til en av lærerne som deltok i intervju-undersøkelsen. Nevnte fysikklærer ble rekruttert gjennom masterveileders kontakter i skolen, som igjen rekrutterte to andre kolleger med fysikk i fagkretsen. På den måten var undersøkelsens utvalg et bekvemmelighetsutvalg, med aspekter av «snowball sampling» og «purposive sampling» (Robson & McCartan, 2016, s. 160). Selv om samtlige informanter har høyere utdanning innen fysikk, har de varierende erfaring med numeriske metoder og programmering. Likevel utgjør informantene (Robson & McCartan, 2016, s. 301) en relativt homogen gruppe, som var interessert i temaet og var positive til å intervjues om det. Basert på dette kan informantene som deltok i intervjuundersøkelsen ha vært mer interesserte i numeriske

metoder og programmering enn andre fysikklærere.

Undersøkelsens resultater viser at intervjuet i hovedsak omhandlet praktiske utfordringer og elevutfordringer, tilknyttet implementasjon av numeriske metoder i fysikkfaget gjennom programmering. Av praktiske utfordringer ble tidsaspektet fremhevet som spesielt viktig. Tidsaspektet handler i hovedsak om at faginnholdet i fysikken opplevdes som større som følge av den nye læreplanen, mens fagets tidsramme forble den samme. Som resultat har elevene mindre tid til å lære de ulike modulene i faget enn de hadde før fagfornyelsen. Videre beskrev lærerne at det å lære elevene å programmere er tidkrevende, og at mange elever har problemer med å anvende det de kan av programmering i fysikken. Altså beskrives elevenes læring som situert (Brown et al., 1989) (mer om dette i kapittel 2.4). De mest sentrale elevutfordringene lærerne tok opp omhandlet (1) elevenes tidligere erfaring med programmering, (2) programmering som «et nytt språk» og dermed et ekstra moment i fysikkfaget. Her fremheves det at elevene har varierende erfaring med numeriske metoder, og spesielt programmering, selv om de alle har vært innom det i matematikken. Undersøkelsens funn indikerer også at idet elevene introduseres for numeriske metoder og programmering, introduseres de også for «et nytt språk», kodespråket, som de må beherske for å kunne anvende det i fysikken. Oppsummert får elevene tildelt for liten tid til å prosessere den presenterte informasjonsmengden, noe som kan resultere i kognitiv overbelastning (Rutkowski & Saunders, 2019)(mer om dette i kapittel 2.4).

Undersøkelsen viser også at lærerne har ulik oppfatning om hva kompetansemålet om numeriske metoder og programmering i fysikk innebærer. Mer spesifikt var de uenige om hva det ville si «*å kunne bruke numeriske metoder og programmering*». Den ene læreren mente at det «*å kunne bruke*» innebar at elevene programmerte selv. Dette ble begrunnet med at «det er enklere å forstå en programkode hvis du jobber deg gjennom og skriver den selv». Imidlertid påpekte læreren at det å la elevene programmere selv er svært tidkrevende. De to andre lærerne var noe uenige i denne tolkningen av kompetansemålet. De mente at det «*å kunne bruke*» innebar at elevene kunne anvende og tilpasse ferdiglagde programkoder, ved eksempelvis å endre variablene, slik at de passer med problemets initialverdier. Dersom programkoden ble gjennomgått med elevene, mente lærerne at elevene kunne forstå kodelinjene og programmet. De påpekte at denne tilnærmingen til kompetansemålet var mindre tidkrevende, og dermed enklere lot seg implementere i et fullpakket fysikkfag. Altså synes det å være et stort

tolkningsrom for hva kompetansemålet innebærer for lærere.

2.2 Fysikkens representasjonsformer

En stor del av fysikkfaget omhandler det å tyde ulike representasjonsformer og «oversette» mellom dem. For mange elever er dette en læringsmessig barriere (se for eksempel Dolin 2002). De forskjellige representasjonsformene beskriver den fysiske verden på ulike måter, og kan deles inn i fem hovedkategorier. Den første representasjonsformen er *fenomenologisk*, der det fysiske fenomenet beskrives gjennom hvordan det opptrer. Den andre representasjonsformen er *begrepsmessig*, og beskriver fysikken ved hjelp av veldefinerte begreper. Fysiske fenomener kan også representeres *eksperimentelt* gjennom undersøkelser med utstyr og måleinstrumenter, eller *grafisk* gjennom avbildninger, figurer, grafer eller tabeller. Den femte representasjonsformen er *matematisk-symbolsk*, der eksempelvis matematiske symboler og likninger benyttes. Å øve på å bruke flere representasjonsformer kan gjøre det enklere for elevene å få oversikt over oppgavene de skal løse og dermed oppleve det som enklere å løse dem, samt gi en dypere fysikkforståelse (Angell et al., 2019, s. 127).

Elevene behøver trening i å anvende de forskjellige representasjonsformene, i tillegg til å utvikle kompetanse i å navigere mellom dem. Dersom fysikklærere følger lærebokens oppbygging, slik som i Ergo 1 og Kraft 1, vil elevenes første møte med fysikkfaget være mekanikk og krefter. Her lærer elevene blant annet at det er svært fordelaktig å tegne en figur som tydelig beskriver situasjonen før de gyver løs på å regne oppgaven. Allerede her presenteres elevene for to representasjonsformer, nemlig *grafisk* og *matematisk-symbolsk*. Når numeriske metoder og programmering implementeres i undervisningen, introduseres elevene for enda en representasjonsform, et *fremmedspråk* som er selve kodingen. Man kan derfor si at dette er en sjette representasjonsform. For at elevene skal sitte igjen med kunnskapen spesifisert i kompetansemålet om numeriske metoder og programmering, bør de forstå og kunne oversette mellom representasjonsformene: grafisk, matematisk-symbolsk og kodespråk.

For mange elever kan det oppleves som overveldende å få tildelt oppgaver der flere representasjonsformer inngår, og hvor det forventes at de kan oversette mellom dem. En måte å redusere risikoen for at elevene overveldes mens de leser oppgaveteksten, kan være å animere situasjonen de skal undersøke. En animasjon kan sørge for at samtlige elever har et visst forhold til problemet før de arbeider med det, enten ved å trekke på elevenes forkunnskaper

eller simpelthen at de ser animasjonen i timen. Altså tilrettelegges det for at alle elevene på forhånd har et forhold til situasjonen i oppgaven. Resultatet av dette er at flere elever kan ta fatt på oppgaven (Wæge & Nosrati, 2018, s. 82-83).

Animasjon er per definisjon en grafisk representasjonsform, da en video er flere bilder satt sammen. Animasjon kan også minne om en fenomenologisk representasjonsform, siden den gir en beskrivelse som er et tett på det fysiske fenomenet. Likevel passer ikke animasjon ordentlig inn i noen av de presenterte representasjonsformene. Følgelig ønsker jeg å tilføye ytterligere en selvstendig representasjonsform, kalt *visuell* representasjon, der animasjoner kan få plass. På samme måte som at elevene kan skifte fra graf til likning, kan de oppleve det som krevende å se sammenhenger mellom animasjoner vist i timen og likningene som trengs for å besvare oppgavene. Av den grunn bør man være ekstra oppmerksom på utfordringene det kan medføre å skifte mellom representasjonsformene.

2.3 Modeller og modellering

Kompetansemålet om numeriske metoder og programmering innebærer også at elevene skal modellere. Det finnes mange forklaringer av ordet *modell*, men de fleste beskriver en *modell* som en refleksjon av virkeligheten (se for eksempel Angell et al., 2008; Etkina et al., 2006; Oh & Oh, 2011). En slik refleksjon kan kommuniseres gjennom ulike representasjonsformer som verbale forklaringer, matematiske uttrykk, grafer og animasjoner (Etkina et al., 2006; Gilbert, 2004; Oh & Oh, 2011). Modeller gjengir virkeligheten på en forenklet måte, og er dermed ikke perfekte speilbilder med 1-1 korrespondanse med virkeligheten (Harrison, 2002, sitert i Angell et al., 2019; Sørby & Angell, 2012). Det samme fenomenet kan også modelleres på forskjellige måter med ulike representasjonsformer, da det er modellmakeren som beslutter hvilke aspekter av fenomenet det skal fokuseres på og hvilke som skal neglisjeres. Antagelsene og beslutningene modellen er tuftet på, er med på å bestemme modellens gyldighetsområde, og hvor «god» den er som et forutbestemmende verktøy (se for eksempel Etkina et al., 2006; Oh & Oh, 2011). Rowley-Jolivet (2004) poengterer at når vi ser på fenomener som opptrer på jorda, brukes gjerne figurative og grafiske representasjoner (se Oh & Oh; 2011, s. 1118). For at elevene skal kunne arbeide med slike jordlige fenomener bør de få trening i å anvende de ulike representasjonsformene og i å «oversette» mellom dem (se for eksempel Angell et al., 2019).

Modellering er et kraftfullt verktøy i læringen av fysikk, og kan gjøre at elevene bedre forstår og

kan utforske sammenhenger mellom matematikken og fysikken (Angell et al., 2008, Angell et al., 2019). For at elevene skal oppleve dette er det nødvendig at læreren kommuniserer hvilken modell som benyttes og dens gyldighetsområde. På den måten har elevene grunnlag til å kunne si seg enig i modellen (Gilbert, 2004; Oh & Oh, 2011). Av Gilbert (2004) omtales dette som en «consensus model». Elevene bør også gjøres bevisste på at modeller som ikke passer perfekt med virkeligheten ikke nødvendigvis er feil, siden alle modeller har begrensninger. Elevene bør heller gjøres oppmerksomme på at modellers avvik fra fenomenet kan skyldes at enkelte aspekter ved fenomenet er utelatt til fordel for andre aspekter modellmakeren ønsket å fokusere på (Oh & Oh, 2011, s. 1120).

Tidligere forskning på numeriske tilnærminger og modellering i fysikk

Haraldsrud og Tellefsen (2018) beskriver at elevene er vant til å finne eksakte svar, gjennom analytiske løsninger, på problemer fra både matematikk- og fysikkfaget. Problemer som lar seg løse analytisk er gjerne sterkt idealiserte problemer fra virkeligheten. Siden virkelighetsnære problemer sjelden lar seg løse analytisk, blir elevene lite eksponert for dem. Dermed kan det sies at kravet om en analytisk løsning begrenser hvilke problemer elevene tildeles. Med programmering kan denne begrensingen imidlertid forsvinne. Programmering kan altså bidra til å gjøre fysikkfaget mer autentisk (Malthe-Sørenssen et al., 2015), ved å muliggjøre for elevene at de kan arbeide med mer virkelighetsnære problemstillinger, som også kan oppfattes som mer interessante (Gilbert, 2004; Haraldsrud & Tellefsen, 2018). Videre kan programmering gi elevene mulighet til å være kreative, siden et problem kan løses med mange ulike programkoder. I forbindelse med dette kan programmering, i form av simuleringer, også bidra til et økt fokus på drøfting, tolkning og vurdering av ulike modeller av fysiske fenomener. Det påpekes også at slike arbeidsmetoder kan åpne for at flere elevtyper kan beherske og føle mestring i fysikkfaget, samt gi elevene forståelse og kompetanse i den naturvitenskapelige tenkemåte og fysikkens egenart (Haraldsrud & Tellefsen, 2018).

I senere år er dette også blitt knyttet til begrepet «computational thinking». Landau (2006) legger vekt på at «computational thinking» (CT) innebærer å bruke sin kunnskap om teori, modellering og implementasjon til å vurdere, visualisere og utforske fysiske fenomener gjennom eksperimenter og simuleringer. Videre påpeker han at CT med programmering ikke

utelukkende kan brukes til å gjøre beregninger, men også benyttes til å lage visualiseringer som kan hjelpe elevene i å vurdere og analysere egne resultater under problemløsning (i Sørby & Angell, 2012). Dette synet på CT ligger tett opptil Weintrop et al. (2016) sin taksonomi av «computational thinking». Å bruke programmering og datamaskiner som hjelpemiddel, innebærer at elevene lærer å utvikle en algoritme for å løse problemet, samt å bruke numerisk matematikk i problemløsning. Av Sørby og Angell (2012) omtales dette som «computational modelling».

I 2019 gjorde Nordby (2019) en omfattende litteraturstudie av forskning på og erfaringer med algoritmisk tenkning og programmering i fysikkundervisning. Basert på en litteraturstudie, og en intervju-undersøkelse med fysikklærere om hvordan de tenker programmering kan implementeres i fysikkfaget, utformet han tre undervisningsopplegg om programmering i fysikk 1 og 2. Selv om utprøvingen av undervisningsoppleggene ikke ble analysert systematisk, deler Nordby (2019) egne erfaringer med utgangspunkt i observasjonsnotater, og elevenes svar på en spørreundersøkelse om deres opplevelse av timens innhold. Studiens resultater viser at elevene generelt opplevde det programmeringstekniske som mest utfordrende, men at de også syntes det å forstå utdelt programkode var vanskelig. Likevel ytret flere elever at de ønsket å forstå hele programkoden, og ikke bare kunne bruke den, selv om dette ikke var nødvendig for å løse oppgaven. Studien indikerer også at elevene er i stand til å bruke programmering til å løse nokså avanserte fysikkproblemer, men at dette forutsetter at elevene gis god støtte i arbeidet og får tett oppfølging fra lærer.

I en annen studie fremkom det at elevene hadde problemer med å trekke riktige konklusjoner da de prøvde å kommentere egne resultater (Sørby & Angell 2012). Sørby og Angell (2012) påpeker at dette trolig kommer av at elevene har liten bevissthet rundt hvordan modellens ulike parametere påvirker arbeidet deres. De viser også til at elevene ofte prøvde å forklare resultaters uventede oppførsel med spontane konsepter eller feil bruk av teori. Videre fikk feil syntaks også mye av skylden når programmet ikke virket eller ga uventet output. Elevene forsøkte ofte å rette opp i feilene ved å vilkårlig endre på kodefragmenter, noe som stjal verdifull tid, som heller burde vært brukt til å analysere og vurdere den matematiske modellen. En av de viktigste årsakene til at mye av fysikktimene går med til å lære programmering fremfor fysikk, er at elevene har for lite kunnskap om programmering (Taub et al., 2015). Her er det verdt å nevne at programmering kan brukes som lærestøtte, fremfor at elevene skal lære seg den bestemte

teknologien (se for eksempel Angell et al., 2019). Ved å ta i bruk programmering som verktøy i fysikkundervisningen, kan det altså dannes en link mellom ulike representasjonsformer av den samme fysiske ideen, som matematisk uttrykk og graf (Chabay & Sherwood, 2008).

Generelt virker det som om fysikkelevene behersker teknisk arbeid, som å anvende bevegelseslikningene, plotte funksjoner og forklare stigningstall. Imidlertid sliter de med å sette dette i en fysisk sammenheng (Erickson, 2006; Taber, 2006, sitert i Angell et al., 2019). Dette eksemplifiseres med at elevene kan identifisere stigningstallet til en tid-strekningsgraf, men ikke nødvendigvis forbinder stigningstallet med fart. Det påpekes at dette kan komme av at elevene ikke har nok trening i å tyde dataene som grafiske plott fremviser. Angell et al. (2019) viser også til at mange elever sliter med å forstå og tolke grafer, og poengterer at digital visualisering dermed kan være mer til hinder enn støtte for elevenes læring. Dette begrunnes med at programmeringen kan være i overkant krevende for elevene, siden de både må forholde seg til det fysikkfaglige og det digitale verktøyet samtidig (se for eksempel Malthe-Sørenssen et al., 2015). Imidlertid kan man oppnå en synergieffekt dersom elevene tildeles nok tid til å arbeide med oppgavene. For at datamaskinen skal være til hjelp i arbeidet deres, krever det at elevene utarbeider og gir datamaskinen tydelige instruksjoner. Et slikt forarbeid kan gjøre elevene mer bevisste på hva ulike begreper innebærer, og tydeliggjøre sammenhenger i fysikken (Angell et al., 2019).

Digital modellering åpner også opp for flerfoldige måter å formidle og fremstille fysiske fenomener på. Mork og Erlien (2017, s. 150-151) fremhever at animasjoner gjør at elevene lettere kan visualisere fenomener som involverer bevegelse. Ved å vise elevene animasjoner kan de lettere se for seg nye situasjoner, men også se kjente situasjoner fra nye perspektiver. Videre kan også simuleringer være svært nyttige, da elevene har anledning til å aktivt endre på modellens ulike parametere, og mer eller mindre umiddelbart se hvordan de påvirker hverandre, og dermed bedre forstå fysikken (Angell et al., 2019).

Det viser seg også at måten modelleringsoppgavene brukes på i undervisningen, kan påvirke fysikklæringen. Sins et al. (2005) beskriver at det er fordelaktig å gi elevene modelleringsoppgaver som innebærer delmål, som å identifisere kreftene som virker på det studerte legeme og bestemme kreftenes fortegn. Slike delmål hjelper elevene i å se fysikken i oppgavene de modellerer og bedre forstå hvordan modellstrukturen påvirker det studerte fenomenet, og dermed løse oppgaven (Sørby & Angell, 2012). I forlengelsen av dette kan

elevene bedre vurdere egne resultater og komme frem til korrekte fysikkfaglige konklusjoner. Som nevnt tidligere, kan modeller formidles gjennom ulike representasjonsformer. Studier tyder på at å bruke modeller med ulike representasjonsformer i undervisningen kan fostre effektiv læring, dersom de brukes på en «passende» måte. Hva som ansees som «passende» avhenger i stor grad av elevenes kunnskaper (se for eksempel Ainsworth, 2008, sitert i Oh & Oh, 2011, s. 1121).

2.4 Situert læring

Ifølge sosiokulturell læringsteori utvikles kunnskap i samhandling med andre (se for eksempel Vygotsky, 1934/2001), men også innen bestemte rammer eller kontekst. Følgelig kan læring sies å være situert, der det å lære noe i én sammenheng ikke nødvendigvis betyr at kunnskapen kan anvendes i andre kontekster (Brown et al., 1989). I skolen tilegner elevene seg kunnskap i de forskjellige fagene, og det forventes til en viss grad at elevene kan overføre kunnskapen mellom de ulike fagene. Det å eksempelvis lære derivasjon i matematikken, løkker innenfor programmering og bevegelseslikningene i fysikken, og forvente at elevene på egenhånd skal kombinere dette og bruke det i andre sammenhenger er problematisk. Dermed er det spesielt viktig å se på læring som situert når numeriske metoder og programmeirng skal forenes med «tradisjonell» fysikkundervisningen.

Ofte er gode matematiske ferdigheter en forutsetning for at elevene oppnår gode resultater i fysikk. Dermed forklares elevers dårlige resultater i faget gjerne med at vedkommende har lave matematiske ferdigheter (Karamet al., 2019, s. 37). Av den grunn er det rimelig å anta at mange elever opplever fysikkfaget som fullt av matematiske formler, regler og metoder som må pugges. Til tross for disse oppfatningene er måten matematikken blir benyttet på i fysikkfaget mer kompleks (Sørby & Angell, 2012, s. 284). Det elevene lærer i matematikken er ment å være verktøy de får bruk for i fysikken, siden fysiske relasjoner gjerne beskrives med det matematiske språket (Sørby & Angell, 2012). Utfordringen ligger i om elevene ser at det de lærer i matematikken, er verdifulle verktøy de får bruk for i fysikken, og om de klarer å overføre ferdighetene fra en kontekst til en annen.

At elevene strever med å anvende tidligere tilegnet kunnskap i nye sammenhenger, kommer til syne ved at de arbeider i ikke-overlappende roller. Eksempelvis vil en elev som jobber med en modelleringsoppgave i fysikk ofte enten arbeide som den konseptuelle fysikeren,

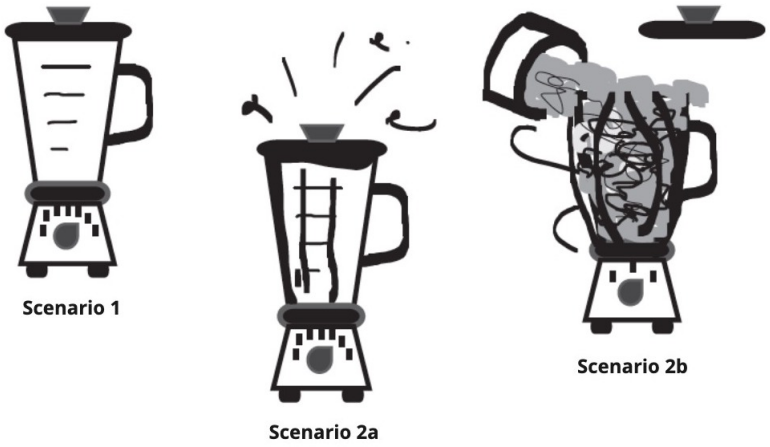
programmereren eller matematikeren (Sørby & Angell, 2012). Sørby og Angell (2012) peker på en fare ved at elevene arbeider i slike ikke-overlappende roller, nemlig at de går glipp av viktig fagforståelse. På den måten kan elevers usystematisk «prøving og feiling» når de arbeider med programmering i fysikk, gå på bekostning av fysikkforståelsen deres. En måte å få elevene til å arbeide som den konseptuelle fysikeren, programmereren og matematikeren samtidig, er å tilrettelegge for at elevene selv får oppleve at programmering og matematikk er direkte relevant for fysikkfaget.

Kognitiv overbelastning

Rutkowski og Saunders (2019) beskriver hvordan den menneskelige hjernen ikke kan prosessere en ubegrenset mengde med informasjon, noe som kan være en utfordring med digital teknologi. Dermed kan det å skulle tilegne seg store mengder kunnskap resultere i det de omtaler som «brain overload», oversatt til «kognitiv overbelastning» på norsk. Rutkowski og Saunders (2019) deler inn i tre ulike scenarioer. I scenario 1 ønsker man å blande ingredienser som er vanskelige å blande sammen. Dersom blenderens hastighet skrus opp tilstrekkelig, klarer vi likevel å blande sammen ingrediensene. Dette fungerer kun dersom mengden ingredienser ikke overskrider blenderens kapasitet. Skulle man ønske å blande sammen en mengde ingredienser som overskrider blenderens kapasitet, dukker det opp to andre scenarioer. Enten kan man porsjonere ingrediensene i mindre delmengder, som prosesseres hver for seg. På den måten kan alle ingrediensene blandes. Dette er scenario 2a. Dersom mengden ingredienser ikke stykkes opp inntreffer scenario 2b, der blenderen oversvømmes og ingen av ingrediensene blir sammenblandet. De tre scenarioene er illustrert i Figur 2-3 (se neste side).

Dersom elevene møter på problemer i fysikken som de enda ikke har fått verktøy til å løse fra matematikken, kan dette medføre at elevene opplever at de verken forstår det fysiske problemet eller matematikken som er ment å være et verktøy. En slik situasjon kan beskrives som scenario 2b av blendermetaforen. Her blandes ikke ingrediensene sammen uansett om vi setter blenderen på maksimal styrke. Dette vil si at elevene presenteres for såpass store mengder informasjon at de ikke klarer å prosessere det nye fagstoffet, og dermed ikke tilegner seg den ønskede kunnskapen.

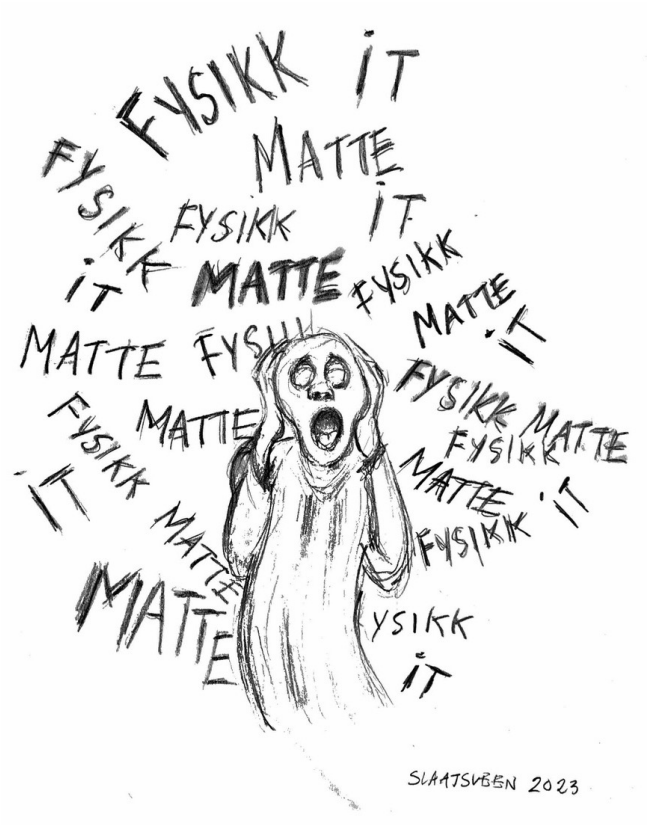
Det er viktig å fremheve en forskjell mellom blenderen og det faktiske mennesket. I metaforen antar vi at alle blendere har samme prosesseringsevne. Dette er ikke tilfellet hos mennesker.



Figur 2-3: Figuren illustrerer kognitiv overbelastning med en blendermetafor. Figuren er hentet fra Rutkowski og Saunders (2019, s. 5) og er gjengitt med tillatelse fra forlaget Routledge.

Rutkowski og Saunders (2019) beskriver hvordan forskjellige individer innehar ulike kognitive ferdigheter og erfaringer. Det er de individuelle forskjellene som avgjør hvorvidt vedkommende makter å prosessere informasjonen hen presenteres for. Altså antas det at enkelte individer har bedre prosesseringsevne enn andre, og dermed har bedre kognitive evner. I scenario 2a av blendermetaforen, ser vi også bort fra et viktig menneskelig aspekt. Mens en blender antas å kunne prosessere håndterbare mengder over en lang periode, er ikke situasjonen den samme for mennesker. Ethvert individ har en øvre grense for hvor mye vedkommende kan prosessere i løpet av en viss tid. Dermed er det ikke mulig for et individ å bearbeide ubegrensede mengder informasjon, uansett hvor mye informasjonen stykkes opp. Mennesker har rett og slett en grense for hvor mye de klarer å håndtere. Når denne grensen overskrides ender vi opp med «kognitiv overbelastning» (Rutkowski & Saunders, 2019, s. 6). Den overveldende følelsen av å motta mer informasjon enn man klarer å prosessere er illustrert i Figur 2-4.

I fysikken er numeriske metoder og programmering et verktøy som kan hjelpe elevene med å forstå fysiske fenomener. Elever som ikke behersker verktøyet står dermed i fare for å oppleve kognitiv overbelastning. Risikoen for kognitiv overbelastning kan reduseres ved å tilrettelegge for at elevene blir kjent med verktøyet, og kan anvende det i ulike kontekster.



Figur 2-4: Figuren illustrerer en følelse av overveldelse som kan oppstå hos elevene når de skal lære flere nye elementer samtidig, eller om de ikke er godt nok kjent med verktøyene som skal brukes, og hvordan de kan anvendes. Illustrasjon av Harald Slaatsveen.

Den proksimale utviklingssonen og selvbestemmelsesteorien

I forbindelse med kognitiv overbelastning, er det naturlig å trekke inn Vygotskys (1934/2001) syn på læring. Han fremhever blant annet at den pedagogiske praksisen bør foregå i elevens proksimale utviklingssone. Den proksimale utviklingssonen defineres som det elevene kan klare på egenhånd, og det de kan klare med hjelp fra en lærer (s. 166). Selv om elevene får til betraktelig mer når de får hjelp enn det de får til på egenhånd, kan de imidlertid ikke hjelpes til å løse ethvert vanskelig problem. Generelt er problemene elevene er best på å løse, de problemene som ligger tett opptil dem de klarer å løse på egenhånd (Vygotsky, 1934/2001, s. 166). Altså er det helt essensielt at problemene elevene tildeles er akkurat passe vanskelige, slik at elevene kan få dem til uten altfor mye hjelp, men likevel blir utfordret. På den måten ledes elevene inn i det de enda ikke får til helt på egenhånd. Dersom undervisningen følger en slik jevn progresjon, kan elevene en dag løse problemer på egenhånd, problemer som de tidligere behøvde hjelp med (Vygotsky, 1934/2001, s. 167). Dette kan medføre at elevene opplever motivasjon og mestring i takt med at den proksimale utviklingssonen utvides, etter hvert som elevene utvikler seg.

«Self-determination theory», oversatt til «selvbestemmelsesteorien» på norsk, presentert av Ryan og Deci (2017), forklarer også motivasjon som en dynamisk prosess. De beskriver at motivasjon kontinuerlig påvirkes av menneskers grunnleggende psykologiske behov for å føle *kompetanse*, *selvkontroll* og *tilhørighet*. Menneskers behov for *kompetanse* går ut på at man føler seg kompetent til å håndtere omgivelsene. Behovet for *selvkontroll* innebærer det å føle at man kan tenke og bestemme selv, og dermed er fri til å ta egne avgjørelser. Behovet for *tilhørighet* handler om det å føle tilhørighet eller tilknytning til andre mennesker. Det beskrives også at disse behovene må dekkes for at menneskelig utvikling kan skje (Ryan & Deci, 2017). Selvbestemmelsesteorien er relevant i undervisningssammenheng, siden skoleklima som støtter autonomi fremmer mer selvmotivasjon, utholdenhet og kvaliteten på læringen (Ryan & Deci, 2017, s. 18). Teorien skaper altså en sterk kobling mellom mestring og motivasjon. Her kan det å oppleve mestring, tilhørighet og selvkontroll på mange måter ansees som «oppskriften» på motivasjon, og dermed læring. Selvbestemmelsesteorien forklarer altså mestring og læring som opplevelsesbasert fremfor faktabasert.

I lys av at elevene skal kunne programmere og gjøre numeriske beregninger i fysikkfaget, bør de mestre representasjonsformene: matematisk-symbolsk og koding som fremmedspråk, samt kunne oversette mellom dem. Å hanskes med de ulike representasjonsformene er ikke enkelt. Følgelig bør elevene tildeles oppgaver de kan få til med en «passelig» mengde hjelp og veiledning fra lærer (Vygotsky, 1934/2001). På den måten kan risikoen for at elevene opplever «kognitiv overbelastning» reduseres (Rutkowski & Saunders, 2019, s. 5-7), slik at sannsynligheten for at elevene opplever mestring øker (Ryan & Deci, 2017).

En måte å forhindre at elevene opplever kognitiv overbelastning, er å sette av tid til å la dem bli kjent med numeriske metoder og grunnleggende programmering, uten å måtte forholde seg til fysikk de ennå ikke behersker. Slik får de mulighet til å opparbeide ferdigheter innen numerikk og programmering, ferdigheter som de senere kan anvende når de eksempelvis skal modellere et vertikalt kast der luftmotstand virker. Dersom grunnopplæring i programmering ikke gis i fysikkfaget, kan elevene oppleve programmering som et nytt faglig aspekt de skal beherske, og som de har liten forutsetning for å klare. Da er sannsynligheten stor for at læringen foregår utenfor elevenes proksimale utviklingssone (Vygotsky, 1934/2001), og at elevene opplever at de har for lite kompetanse til å løse oppgaven. Læringsopplevelsen kan dermed oppfattes som dårlig, noe som trolig medfører at elevene opplever lite motivasjon og mestring

(Ryan & Deci, 2017). Det er imidlertid viktig å påpeke at behovet for å lære opp elevene i grunnleggende programmering *kan* avta med årene, da elever fra senere årskull vil ha arbeidet med programmering fra en tidligere alder som resultat av innføring av nye læreplaner. For at elevene skal lykkes i fysikkfaget er det helt essensielt at de ser relevansen og sammenhengen mellom matematikk, programmering og fysikk, og dermed kan arbeide som den konseptuelle fysikeren, programmereren og matematikeren synkront (Sørby & Angell, 2012, s. 284). For å få til dette bør elevene ha den grunnleggende kompetansen på områdene som er nødvendige for å kunne se sammenhenger. Dette underbygger viktigheten av å la elevene bli kjent med numeriske metoder og programmering før disse ferdighetene for alvor skal anvendes på fysiske fenomener.

2.5 Tilpasse undervisningen til alle elevene

En typisk fysikklasse består av i underkant av 30 elever. Fysikkelever er gjerne faglig sterke og motiverte, men det finnes likevel store forskjeller både i forutsetninger og interesser (Angell et al., 2019; Wæge & Nosrati, 2018, s. 82-83). Skolen og lærere har som oppgave å tilfredsstille elevenes krav om tilpasset opplæring (Utdanningsdirektoratet, 2022). Altså skal enhver elev gis mulighet for læring og utvikling, ved at læreren tilrettelegger undervisningen slik at eleven har best mulig utgangspunkt for faglig utvikling uansett nivå (Kunnskapsdepartementet, 2017). Det som omtales som LIST-oppgaver, er oppgaver med Lav Inngangsterskel og Stor Takhøyde. LIST-oppgaver søker å gjøre at elevene enkelt kommer i gang med oppgaven, samtidig som de gis mange muligheter og ikke begrenses. Følgelig kan oppgavetypen ivareta mange ulike elevtyper (Wæge & Nosrati, 2018, s. 82-83), og sies å imøtekomme kravet om tilpasset opplæring (Kunnskapsdepartementet, 2017). LIST-oppgaver omtales gjerne i forbindelse med matematikk, men har også overføringsverdi til andre fag, deriblant fysikkfaget.

Som nevnt i kapittel 2.3 krever programmering i fysikkfaget en høy grad av presisjon, og at elevene klarer å holde styr på en rekke vanskelige elementer samtidig. Ved å gi elevene innføringsoppgaver tilknyttet grunnleggende programmering, tilrettelegges det for at flere elever kommer i gang med oppgaven. Med andre ord kan inngangsterskelen sies å ha blitt nedjustert. Programmeringsoppgaver kan også løses på mange forskjellige måter, og elevene har dermed stor frihet når det gjelder løsningsmetoder. Basert på dette kan bestemte typer programmeringsoppgaver kategoriseres som LIST-oppgaver.

3 PRINSIPPER FOR UNDERVISNINGSOPPLEGGET

Kompetansemålet om numeriske metoder og programmering forutsetter at elevene har en viss erfaring med programmering. I henhold til den nye læreplanen forventes det i naturfaget at elevene etter 10. trinn «*kan bruke programmering til å utforske naturfaglige fenomener*» (Utdanningsdirektoratet, 2020). Realiteten er at dagens fysikkelever fulgte en annen læreplan. Følgelig har de trolig arbeidet lite med programmering tidligere. Ser vi kompetansemålet:

> «*...eleven skal kunne bruke numeriske metoder og programmering til å modellere og utforske bevegelse i situasjoner der akselerasjonen ikke er konstant.*»
>
> (Utdanningsdirektoratet, 2021)

i sammenheng med de nye kompetansemålene for tidligere årstrinn, er det tydelig at det forventes at fysikkelevene behersker grunnleggende programmering før selve fysikkundervisningen begynner (Utdanningsdirektoratet, 2020). Siden fagfornyelsen ble innført for bare et par år siden, vil det for mange elever sannsynligvis være et gap mellom kunnskapen de sitter på, og kunnskapen det forventes at de innehar. Altså er det fare for at elevene opplever kognitiv overbelastning. Etterhvert som den nye læreplanen har vært gjeldende en stund, vil man kunne tro at fremtidens fysikkelever kan den grunnleggende programmering som kompetansemålet forutsetter.

Undervisningsopplegget er tiltenkt dagens elevgruppe, som er blant de første årskullene som får undervisning etter ny læreplan. Det blir tatt høyde for at elevene ikke nødvendigvis har arbeidet med programmering i skolen før, slik at de får mest mulig ut av undervisningen. For å utforme et best mulig opplegg har jeg utarbeidet flere prinsipper for undervisningen. Prinsippene er basert på teori fra forrige kapittel, og er listet opp og beskrives i det følgende.

- **Prinsipp 1:** Elevene skal få støtte i de ulike representasjonsformene og i å oversette mellom dem.
- **Prinsipp 2:** Opplegget skal ta høyde for at læring er situert. Det å lære noe i én sammenheng, betyr ikke nødvendigvis at kunnskapen kan anvendes i andre kontekster. Av den grunn er det ønskelig at elevene bruker numeriske metoder på kjente og enkle situasjoner, for å bli kjent med hva metodene faktisk gjør.
- **Prinsipp 3:** Opplegget skal søke å unngå kognitiv overbelastning så godt det lar seg gjøre. Kognitiv overbelastning kan unngås ved eksempelvis å dele opp problemet elevene arbeider med, eller legge opp til at elevene jobber med kjent fagstoff.
- **Prinsipp 4:** Undervisningen skal passe til samtlige fysikkelever. En måte å imøtekomme dette prinsippet på er med LIST-oppgaver, oppgaver med Lav Inngangsterskel og Stor Takhøyde.
- **Prinsipp 5:** Elevene skal være aktive i undervisningen, og bli kjent med fagstoffet på en interaktiv måte.

Et undervisningsopplegg basert på prinsippene ovenfor, kan både ivareta det fysikkfaglige og elevene. Undervisningsopplegg utformet etter *Prinsipp 1*, gir elevene trening i å oversette faginnhold mellom ulike representasjonsformer, noe som er en viktig del av fysikkfaget. Å oversette mellom representasjonsformer er en vanskelig øvelse som elevene trenger trening i, og støtte fra lærer, for å mestre. Ved å være oppmerksom på at læring er situert, *Prinsipp 2*, kan læreren bedre forstå utfordringene elevene har med å anvende allerede lært fagstoff, som de eksempelvis har lært i andre skolefag. En slik forståelse kan føre til at sammenhengene mellom forskjellige temaer innen fysikken, eller relasjonen mellom ulike fag, forklares tydeligere. Videre kan en lærer som innser at elevene mangler grunnleggende programmeringsferdigheter, legge opp til at elevene får arbeidet med dette før komplekse fysikkproblemer skal løses. Dermed får elevene opparbeidet seg programmeringsferdighetene de får bruk for i oppgaveløsningen.

Ved å tilrettelegge for at elevene er kjent med hjelpemidlene og faginnholdet som skal benyttes i undervisningen, kan man redusere risikoen for at de kognitivt overbelastes, som er i tråd med *Prinsipp 3*. Her legger læreren opp til at det som forventes av elevene ikke overskrider deres kapasitet til å tilegne seg ny informasjon. Ved å la elevene arbeide med kjent fagstoff

og hjelpe dem med å dele opp komplekse problemer i mindre løsbare deler, kan man minske risikoen for kognitiv overbelastning. I forlengelsen av dette er det helt essensielt at elevene forstår situasjonen som presenteres i oppgaven. Dersom elevene ikke forstår situasjonen de skal undersøke, er det fare for kognitiv overbelastning. Videre kan elevene oppleve det som utfordrende å forstå hvordan de skal anvende allerede lært kunnskap. Dette er beskrevet som scenario 1 av blendermetaforen (Rutkowski & Saunders, 2019, s. 5-7). En måte å forhindre denne typen kognitiv overbelastning på, er å sørge for at elevene er innforstått med fysikken i situasjonen som beskrives i oppgaven. Her er animasjoner et godt hjelpemiddel. Ved å bruke animasjoner til å forklare situasjonen som skal undersøkes, kan flere elever forstå premissene for oppgaven de skal løse.

Prinsipp 4 sørger for at oppgavene elevene gis er enkle og tydelige nok til at alle kan ta fatt på dem. Oppgavene kan også utvides til mer krevende oppgaver, noe som ivaretar hele elevgruppen (Wæge & Nosrati, 2018, s. 82-83). En måte å gjøre elevene kjent med programmering på, er å la dem delta aktivt i undervisningen, *Prinsipp 5*. Dette kan eksempelvis tolkes bokstavelig, der elevene settes i fysisk aktivitet for å simulere hvordan en datamaskin «tenker» når den sorterer data. Aktiv deltakelse kan også innebære at elevene repeterer grunnleggende programmering ved å gjøre interaktive oppgaver, istedenfor at læreren repeterer fagstoffet i plenum. I gjennomgang av eventuelle eksempler kan elevene også delta med forslag til løsningsmetoder. Det utformede undervisningsopplegget, som presenteres i det neste kapittelet, bygger på disse fem prinsippene.

4 UNDERVISNINGSOPPLEGG: Presentasjon, begrunnelse og utprøving

Dette kapittelet handler om det utformede undervisningsopplegget og utprøvingen av det. Leseren presenteres først for undervisningsopplegget i kapittel 4.1, hvor valgene som ble gjort i utformingen av opplegget begrunnes. I kapittel 4.2 beskriver jeg gjennomføringen av opplegget, og deler relevante erfaringer fra klasserommet. Undervisningsmaterialet som ble brukt i timene er tilgjengelig som vedlegg. Minioppgavene som ble brukt som repetisjon, og de to oppgavearkene elevene fikk utdelt, finnes i henholdsvis Vedlegg D, Vedlegg E og Vedlegg F, mens lysbildepresentasjonen som ble holdt i klasserommet finnes i Vedlegg I. Utdelt programkode og løsningsforslag finnes i Vedlegg J. Programkoden til animasjonene som ble vist for elevene, er laget i samarbeid med Trond Morten Thorseth, spesifikt for denne masteroppgaven, og finnes i Vedlegg H.

4.1 Modellering av vertikal bevegelse med luftmotstand

Undervisningsopplegget er utarbeidet etter følgende kompetansemål:

> «*...eleven skal kunne bruke numeriske metoder og programmering til å modellere og utforske bevegelse i situasjoner der akselerasjonen ikke er konstant.*»
>
> (Utdanningsdirektoratet, 2021)

Opplegget handler om å utforske vertikal bevegelse med luftmotstand, beskrevet ved bevegelseslikningene med numeriske metoder og programmering. Her får elevene innsikt

i fysiske fenomener gjennom numerikk, og hvordan dette kan visualiseres ved hjelp av grunnleggende programmering. I det følgende presenteres undervisningsoppleggets ulike momenter.

Undervisningsopplegget: Numeriske metoder som faginnhold

For å kunne oppfylle undervisningsoppleggets læringsmål, må elevene forstå hva numeriske metoder innebærer, og hvordan det skiller seg fra programmering. I opplegget presenteres elevene for hva numeriske metoder innebærer (se Lysbilde H-10). Her forklares numeriske metoder ved hjelp av elevenes kjennskap til bevegelseslikningene. Videre velges Eulers metode som et eksempel på en numerisk metode. Det er denne det blir lagt hovedfokus på videre. Eulers metode kan illustreres ved at læreren tar ett skritt fremover og spør klassen «hvis jeg tar enda et skritt rett frem, hvor er jeg da?». Ved å repetere denne sekvensen flere ganger, får elevene en følelse av hva Eulers metode gjør. Den matematiske metoden kan også illustreres gjennom å sammenligne den med det å navigere etter Google Maps.

Elevene skal også utføre Eulers metode med penn og papir, noe som kan føre til at de får en bedre forståelse av metoden. Her blir de i første omgang bedt om å bruke metoden i kombinasjon med vei-fart-tid-formelen, til å regne ut hvor langt en bil som kjører i 60 km/t har kommet etter 5 tidssteg, der hvert tidssteg er én time. Videre blir elevene bedt om å gjøre dette for tilstrekkelig mange tidssteg, si 15, slik at prosessen tar relativt lang tid. Ikke bare kan dette medføre at elevene får en dypere forståelse av hva som skjer i hvert steg i metoden, men også bedre forståelse av hvorfor numeriske metoder ofte gjennomføres av en datamaskin. Elevene tildeles også en programmeringsoppgave om den samme bilen, der de skal fullføre en programkode, ved å skrive noen kodelinjer, slik at bilens farts- og akselerasjonsgraf plottes (se Vedlegg E). På den måten får elevene også trening i å fremstille data grafisk.

I undervisningsoppleggets neste del, gis elevene en oppgave der de må bruke det de har lært om mekanikk og bevegelseslikningene, i kombinasjon med grunnleggende programmering. Her undersøker elevene en ball som kastes rett opp fra bakkenivå der luftmotstanden virker (se Lysbilde I-12 til I-16). Situasjonen presentert i oppgaven vil også illustreres visuelt ved hjelp av en animasjon (se Vedlegg H), som knytter oppgaven til elevenes egne erfaringer, og gir dem øvelse i å «bevege seg mellom» ulike representasjonsformer. Elevenes oppgave er å gjøre nødvendige beregninger av posisjon, fart og akselerasjon, samt å fremstille dette i separate plot.

De får også utdelt et tilhørende oppgaveark der de skal svare på spørsmål som i hovedsak går på fysikkforståelse. Når læreren går rundt og veileder elevene er det spesielt viktig at hen ikke hjelper for mye. Noen holdepunkter for veiledningen kan være å hjelpe elevene med å forstå situasjonen de skal undersøke, stille spørsmål som oppfordrer til høyere ordens tenkning, samt oppfordre og støtte dem i å bruke Google som hjelpemiddel. Oppgaven kan også utvides ved å eksempelvis be elevene om å bruke fysikkunnskapen deres til å vurdere om plottene deres virker rimelige.

I siste del av undervisningsopplegget får elevene utdelt en grubleoppgave som ligner på den forrige oppgaven, men som er mer kompleks. Her skal de undersøke bevegelsen til en sprettball som slippes fra en bestemt høyde der luftmotstand virker og energi går tapt i sprettet, når ballen er i kontakt med underlaget (se Lysbilde H-17 og H-18). I likhet med den forrige oppgaven, vil situasjonen i oppgaven illustreres for elevene ved hjelp av en animasjon (se Vedlegg H). I oppgaven gjøres det forenklinger. Vi antar at kontakttiden mellom ball og underlag er infinitesimalt. En annen antakelse er at energitapet i sprettøyeblikket fører til en prosentvis reduksjon i sprettballens hastighet, der hastigheten går fra v til eksempelvis $0,8v$. I realiteten vil sprettøyeblikket vare en tid dt, og ballens energitap i kontakt med underlaget vil blant annet avhenge av materialet ballen er laget av. Forenklingene gir en mindre presis modell av virkeligheten, men ivaretar fenomenets hovedmomenter. Elevenes oppgave er å gjøre nødvendige beregninger av sprettballens posisjon, fart og akselerasjon, og fremstille det grafisk. Også her får elevene utdelt et tilhørende oppgaveark som omhandler elevenes fysikkforståelse. Ved å be elevene undersøke hva som skjer hvis man antar null energitap i sprettøyeblikket, får oppgaven et ekstra element av fysikkforståelse. Også her er det viktig at elevene gis passende veiledning av læreren, slik at oppgaven ikke løses for dem eller at de opplever kognitiv overbelastning.

En skjematisk oversikt over undevisningsoppleggets rammer og innhold er gitt i Tabell 4-1 (se neste side). Det presenterte undervisningsopplegget vil trolig skille seg fra elevenes vante undervisning, da fokuset på programmering vil være større. Det gis derfor en introduksjon, der elevene introduseres for hva som vil skje og oppleggets mål presenteres. Undervisningsoppleggets mål er «*å kunne bruke numeriske metoder og programmering som et verktøy i fysikken*», og er basert på kompetansemålet: «*å kunne bruke numeriske metoder og programmering til å modellere og utforske bevegelse i situasjoner der akselerasjonen ikke er*

Tidsramme	2 dobbelttimer (à 90 minutter)
Utstyr for lærer	Datamaskin med animasjonsprogramvaren Blender
Utstyr for elever	• 1 datamaskin per elev • Skrivesaker og noe å skrive på
Læringsmål	• kunne bruke if-else-setninger og for-løkker og kunne forklare hva de gjør • kunne bruke liste/array og kunne forklare hva de er • kunne bruke numeriske metoder til å beregne fart, posisjon og akselerasjon for rettlinjede bevegelser • kunne fremstille algebraiske svar som plott • kunne drive feilsøk • forstå hensikten med å bruke numeriske metoder og programmering i fysikk • bli kjent med fysikkens arbeidsmåter
Struktur	1. **Oppstart og introduksjon**. Undervisningsoppleggets mål, algoritmisk tenkning og programmering presenteres. 2. **Minioppgaver**. Elevene tildeles oppgaver hvor de får repetert grunnleggende programmering, feilsøking og hvordan rette opp i feil. 3. **Repetisjon av bevegelseslikningene**. 4. **Gjennomgang av numeriske metoder**. Her introduseres elevene for hva numeriske metoder er, og blir kjent med Eulers metode. 5. **Eksempel med konstant akselerasjon**. Beregne posisjon med Eulers metode og fremstille det grafisk. 6. **Oppgave**. Beregne og visualisere posisjon, fart og akselerasjon for et vertikalt kast der luftmotstanden virker med Eulers metode. 7. **Grubleoppgave**. Beregne og visualisere posisjon, fart og akselerasjon for en sprettball der luftmotstanden virker med Eulers metode.

Tabell 4-1: En oversikt over undervisningsoppleggets tidsramme, nødvendig utstyr, læringsmål og stuktur.

konstant». Ved å formidle hensikten med opplegget til elevene, skapes både en struktur og en forutsigbarhet for hvordan undervisningen vil foregå den neste perioden.

For at elevene skal kunne bruke programmering som et verktøy er det essensielt at de vet hva programmering innebærer. Elevene blir presentert for viktige aspekter ved programmering, gjennom algoritmisk tenkning. Videre får de også kjenne på hva algoritmisk tenkning innebærer i en hverdagslig og praktisk situasjon. Her skal elevene innad i klassen sorteres i stigende rekkefølge etter fødselsdato, uten å si annet enn «når har du bursdag?», eller si bursdagen sin. Gjennom en slik aktivitet blir elevene kjent med hvordan en datamaskin «tenker».

Kognitiv overbelastning kan unngås ved å gi elevene en gjennomgang av fysikkfagstoffet om mekanikkens bevegelseslikinger. Av samme grunn bør de få tid til å arbeide med grunnleggende programmering for å kunne anvende det i faget. For å tilrettelegge for at elevene får størst mulig faglig utbytte av undervisningen er det derfor hensiktsmessig å gi dem en oppfriskning i

programmering. Hvis tilfellet er at elevene ikke har programmert noe særlig før, er det spesielt viktig å la dem bli kjent med det som ansees som grunnleggende programmering. En måte å imøtekomme elevenes behov for repetisjon av fysikkfagstoff på, er å skrive opp og forklare bevegelseslikningen på tavlen, og regne noen enkle eksempler i fellesskap (se Lysbilde H-9). Oppfriskningen av programmering kan foregå på en liknende måte, der de grunnleggende kommandoene «print», variabler, «if- og else-setninger», «for-løkker» , «arrays» og «plotting av grafer» presenteres og forklares. Kommandoene kan gjerne forklares ved å involvere elevene. Eksempelvis kan det å lagre variabler på en datamaskin illustreres ved å be enkeltelever huske ulike tall, før man deretter ber de samme elevene si tallet de ble bedt om å huske. Dette illustrerer på en hverdagslig måte hvordan det å lagre variabler simpelthen er å be datamaskinen huske data, eksempelvis tall, slik som elevene nettopp gjorde. En slik tilnærming til fagstoff, som for noen elever kan virke abstrakt, kan være med på å gi dem et nærere forhold til det som undervises i, og kanskje medføre at kunnskapen fester seg bedre. Imidlertid bør elevene også selv få trening i å anvende de ulike kommandoene, og det er dermed ønskelig å tildele dem små oppgaver der de får trening i å bruke dem (se minioppgavene i Vedlegg D).

Feilsøking og nettsøk som verktøy

Feilsøking og syntaksfeil er en stor del av programmeringsarbeidet, og elevene bør bli kjent med dette. Det er også viktig å formidle til elevene at det er nettopp feilsøking og syntaksfeil ingeniører og programmerere bruker mesteparten av tiden sin på. Å få innblikk i hvordan kode utvikles i yrkeslivet, og gjøre elevene oppmerksomme på at feilkode nesten alltid dukker opp, kan være med på å gjøre programmering mindre skremmende. Håpet med en slik tilnærming til temaet er at elevene innser at gode programmerere ikke kjennetegnes ved at de verken får feilkode eller syntaksfeil, men ved at de er gode på å identifisere feilene i koden og rette opp i dem.

Google er programmererens beste venn i arbeidet med programmering og feilsøking. Mange elever har trolig lite trening i å bruke Google, eller andre søkemotorer, til å finne fagspesifikk informasjon, slik eksempelvis studenter gjør. Det er ikke sånn at ingeniører går rundt og husker enhver kommando de har brukt når de koder. Realiteten er heller at Google blir flittig brukt til å finne kommandoene man er ute etter, eller for å forstå hva en feilmelding skyldes. Ved eksplisitt å fortelle elevene om verdien av å google og la dem øve seg, kan de oppleve å være

mer autonome i eget programmeringsarbeid. Ved å legge inn planlagte feil i programkoden som skal kjøres for elevene (se minioppgavene i Vedlegg D), kan læreren vise dem hvordan man driver feilsøk ved hjelp av nettsøk. Her kan man sammen finne ut hvor i koden feilen ligger, og komme med forslag om hva feilene betyr og hvordan de kan rettes opp i. I tillegg vil det å kode sammen med elevene på storskjerm, fremfor å vise ferdiglagd kode, gi dem inntrykk av hvordan kode skapes, og alle utfordringene man møter. Gjennom en slik tilnærming integreres aspektene ved feilsøking og nytten av søkemotorer naturlig i undervisningen.

Programkode kan også inneholde det som kalles logiske feil. Her ligger ikke problemet i at datamaskinen ikke forstår programkoden, men heller i at programmet ikke gjør det man ønsker. Et eksempel på en slik logisk feil er et program som skal printe arealet av en sirkel, men printer omkretsen. Programmet vil ikke gi feilmeldinger, selv om noe er galt. Her ligger feilen i at programmet er fôret med feil arealformel, og ikke i feil symbolbruk. Slike logiske feil er spesielt viktig å gjøre elevene oppmerksomme på i fysikkfaget, da feilene kan skyldes elevenes misoppfatninger i faget eller manglende fysikkforståelse.

4.2 Utprøving av undervisningsopplegget

Opplegget om numeriske metoder og programmering ble prøvd ut i en fysikklasse ved en alminnelig videregående skole i januar 2023. Undervisningsopplegget ble gjennomført av forfatteren, med faglærer til stede i klasserommet. I utgangspunktet var tidsrammen for opplegget 3 dobbelttimer (à 90 minutter). Grunnet uforutsette hendelser, måtte tidsrammen justeres ned til 2 dobbelttimer, kort tid før undervisningsopplegget skulle gjennomføres. Elevene hadde tidligere i skoleåret arbeidet en del med syntaks og grunnleggende programmering. Basert på dette, gjorde jeg nødvendige justeringer, og endte opp med det presenterte undervisningsopplegget, en nedkortet versjon av det opprinnelige opplegget, som kunne gjennomføres innenfor den gitte tidsrammen.

Oppleggets faktiske tidsramme ble 2 dobbelttimer (à 90 minutter) og 1 enkelttime, og fordelte seg over 3 økter. Økt 1 ble gjennomført med forfatteren og faglærer til stede, mens jeg under økt 2 og 3 også hadde med meg en medstudent som skulle gjøre lydopptak av elevdiskusjoner i timen. Elevene fikk utdelt alle oppgavene på papir. De ulike deloppgavene, med unntak av programmeringsoppgavene (oppgave 1a og 2d), skulle besvares direkte på papiret med penn eller blyant. Elevene ble også bedt om å la være å skrive navnet sitt på oppgavearkene, slik at

besvarelsene kunne samles inn som datamateriale og at elevens anonymitet ble ivaretatt. Videre følger en beskrivelse av innholdet i de tre undervisningsøktene.

Økt 1: Minioppgaver og «Bil med konstant akselerasjon»

I den første undervisningsøkten ble elevene presentert for undervisningsoppleggets mål og hva som var forventet utbytte. Elevene fikk en innføring i begrepene programmering og algoritmisk tenkning, og hva de innebærer. De fikk også tildelt syv mindre oppgaver, hvor de fikk repetert grunnleggende programmering, drevet feilsøk og rettet opp i syntaksfeil (se minioppgavene i Vedlegg D). Siden undervisningsopplegget omhandlet rettlinjet bevegelse av ulike legemer, fikk elevene også en rask oppsummering av bevegelseslikningene, hva de kan brukes til og hvilke begrensninger de har. Den lærerstyrte gjennomgangen munnet ut i en aktivitet der elevene med penn og papir skulle bestemme farten og den tilbakelagte strekningen til en bil, ved en rekke tidspunkter, ut fra et sett med initialbetingelser og informasjon om bilen. Videre ble elevene introdusert for begrepet *numeriske metoder*, og fikk kjennskap til *Eulers metode*. I forlengelsen av dette fikk de tildelt oppgaven «Bil med konstant akselerasjon» (se Vedlegg E).

Økt 2: «Vertikalt kast med luftmotstand»

I undervisningsoppleggets andre økt skulle elevene arbeide med oppgaven «Vertikalt kast med luftmotstand» (se Vedlegg F), og hvis de fikk tid arbeide med grubleoppgaven «Sprettball med luftmotstand» (se Vedlegg G). Som en innledning til oppgaven «Vertikalt kast med luftmotstand», ble elevene vist en animasjon av situasjonen. En tilsvarende inngang til oppgaven «Sprettball med luftmotstand» var også planlagt. Underveis i timen oppdaget jeg at bare et mindretall av elevene rakk å besvare alle deloppgaver, altså at oppgavearket var mer omfattende enn forventet. Likevel ble slutten av timen satt av til at elevene fikk svart på spørreskjemaet (se Vedlegg B) om hvordan de opplevde undervisningsopplegget, både økt 1 og økt 2. Beslutningen ble tatt på bakgrunn av at spørreskjemaet kunne gi bedre innsikt i elevenes opplevelse av undervisningsopplegget enn det oppgavearkene kunne. Spørreskjemaet ble besvart anonymt (mer om dette kapittel 5).

Økt 3: Mer om «Vertikalt kast med luftmotstand»

I løpet av økt 2 fikk et mindretall av elevene arbeidet skikkelig med deloppgavene, som omhandlet det nye fagstoffet. Basert på dette ble faglærer og jeg enige om å bruke ytterligere 1 enkelttime på undervisningsopplegget. Som resultat ble oppleggets tidsramme 2 dobbelttimer (à 90 minutter) og 1 enkelttime. Enkelttimen gikk med til å la elevene arbeide videre med oppgavearket «Vertikalt kast med luftmotstand» (se Vedlegg F).

Erfaringer fra klasserommet

Egne observasjoner fra økt 1, der elevene arbeidet med minioppgaver om grunnleggende programmering, og oppgavearket «Bil med konstant akselerasjon», tyder på at oppgavene var passe vanskelige. Samtlige elever fikk løst oppgavene i løpet av timen, men med ulik grad av veiledning fra lærer. Selv om de første oppgavene som ble gitt i økt 2 virket å være «passe» vanskelige, ble det også gitt oppgaver som var i overkant vanskelige for flere av elevene. Mot slutten av økt 2, ble det tydelig at flere av elevene slet med programmeringsoppgaven 2d (se Vedlegg F), og ikke rakk å arbeide med de neste deloppgavene som omhandlet nytt fagstoff. For at elevene skulle kunne gå videre i oppgavesettet, og få arbeidet med oppgavene om forståelse av tidssteget dt, besluttet jeg å gi elevene den ferdiglagde programkoden til oppgave 2d. Hvordan elevene brukte programkoden og i hvilken grad den ble benyttet i deres videre arbeid varierte. Mens noen elever brukte egenlagd eller utdelt programkode til å starte på de neste deloppgavene, fortsatte andre elever på oppgave 2d, både med og uten hjelp fra den utdelte programkoden.

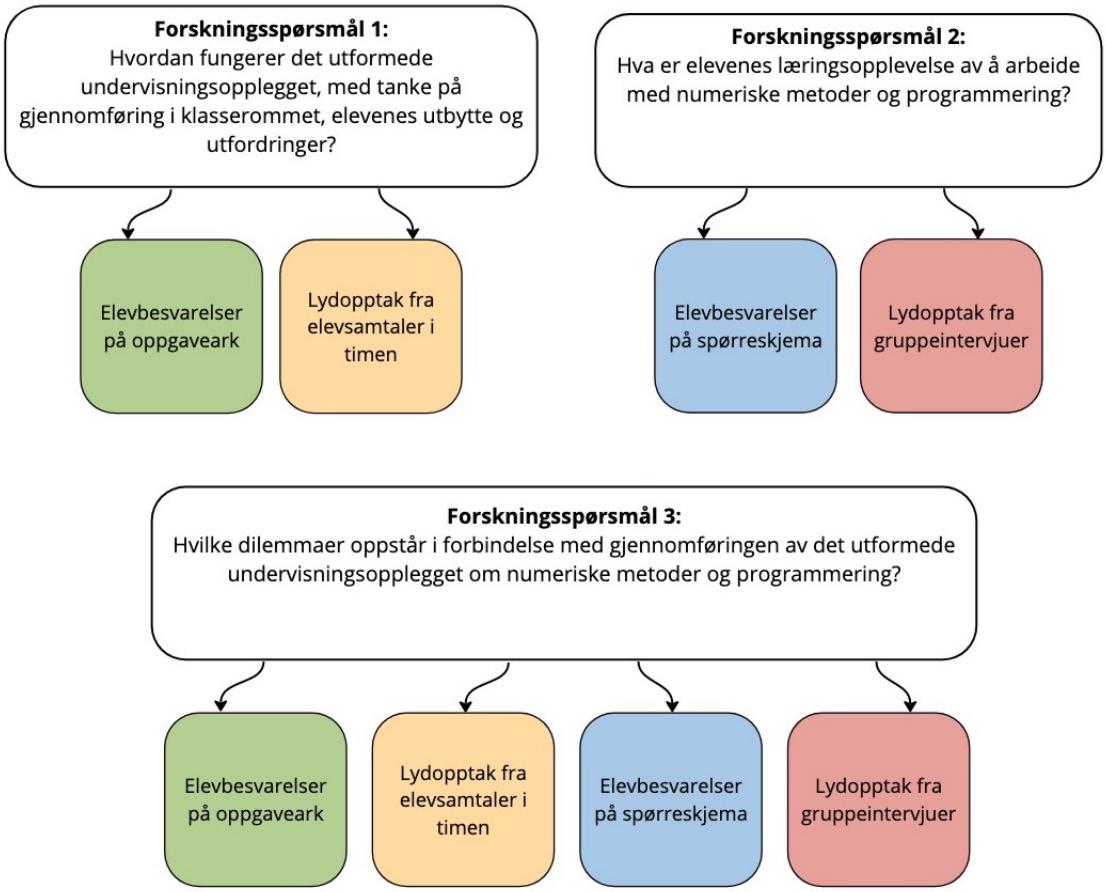
I økt 3 ble elevene eksplisitt bedt om å bruke den utdelte programkoden til å besvare de neste deloppgavene. Basert på dette var det behov for å gjøre små justeringer på oppgavearket de skulle få utdelt i økt 3. Et eksempel på dette er at oppgave 2e, der elevene skulle skrive programkode som regnet ut ballens maksimale høyde, ikke ble gitt. Som følge av at elevene brukte mer tid på oppgaven «Vertikalt kast med luftmotstand» enn forventet, rakk ingen av dem å begynne på grubleoppgaven «Sprettball med luftmotstand» (se Vedlegg G).

5 METODE

Dette kapittelet handler om studiens design og metodiske valg. I første omgang presenteres leseren for studiens forskningsdesign, kapittel 5.1. I kapittel 5.2 gis en beskrivelse av det innhentede datamaterialet, studiens utvalg, og personvernet til undersøkelsens deltakere. Her presenteres datainnsamlingsmetodene som er brukt, med tilhørende litteratur. I tillegg deler jeg erfaringene jeg hadde med de ulike metodene. Til slutt beskrives jeg i kapittel 5.3 hvordan studiens datamateriale er analysert.

5.1 Forskningsdesign

Studien er av den kvalitative sorten, der fokuset har vært informantenes tanker og resonnementer, fremfor tall og statistikk (se for eksempel Braun & Clarke, 2022; Robson & McCartan, 2016, Tjora, 2018). For å besvare masteroppgavens forskningsspørsmål, og dermed problemstillingen, har jeg benyttet data- og metodetriangulering (se for eksempel Braun & Clarke, 2022, s. 278; Robson & McCartan, 2016). Det har blitt brukt to overordnede metoder for datainnsamling: (1) elevbesvarelser og (2) lydopptak. Datainnsamlingsmetodene delte seg igjen i to undergrupper. Datamaterialet består dermed av (1a) elevbesvarelser på oppgaveark, (1b) elevbesvarelser på spørreskjema, (2a) lydopptak fra elevsamtaler i timen og (2b) lydopptak fra gruppeintervjuer. Hensikten med et såpass omfattende datamateriale, var at de ulike delene kunne bidra til å besvare ett eller flere av forskningsspørsmålene. Helt konkret var det tenkt at forskningsspørsmål 1 skulle besvares med datamateriale (1a) og (2a), forskningsspørsmål 2 med datamateriale (1b) og (2b), og forskningsspørsmål 3 med datamateriale (1a), (1b), (2a) og (2b). En skjematisk fremstilling av hvilket datamateriale som skulle besvare de ulike forskningsspørsmålene er vist i Figur 5-1 (se neste side).



Figur 5-1: Diagrammet viser en oversikt over studiens forskningsspørsmål, og hvilke datakilder som er ment å besvare dem.

Studien har aspekter av både fast og fleksibelt design. Oppgavearkene og spørreskjemaet innhentet i hovedsak kvalitativ informasjon gjennom både lukkede og åpne spørsmål, og hadde dermed et fast forskningsdesign. På den måten fikk jeg innsikt i alle elevenes opplevelse av undervisningsopplegget, noe som hadde vært betraktelig mer tidkrevende å få tilgang på gjennom et mer fleksibelt forskningsdesign som intervju (Kvale & Brinkmann, 2015; Postholm, 2010; Robson & McCartan, 2016). For at lydopptakene skulle produsere gode data, var det naturlig å velge et fleksibelt forskningsdesign, slik at undersøkelsen kunne formes i takt med datainnsamlingen (Postholm, 2010). Dette gjorde det mulig å stille oppfølgingsspørsmål for å bedre forstå informantenes svar og dermed få bedre innsikt i refleksjonene deres. På grunn av studiens kvalitative natur, og utvalget som er gjort, er resultatene og kunnskapen som er produsert situasjonsavhengig, og kan derfor ikke nødvendigvis generaliseres i statistisk forstand. Studien kan imidlertid være nyttig, da den viser hva som *kan* være tilfellet. Dermed kan studien ha overføringsverdi til lignende situasjoner, kalt analytisk generaliserbarhet (se for eksempel Postholm, 2010).

5.2 Datainnsamling

Det innhentede datamaterialet er femdelt, og inkluderer skriftlige elevbesvarelser, svar på spørreskjema, observasjon med tilhørende notater fra gjennomføringen av undervisningsopplegget, transkriberte lydopptak av elevsamtaler fra timen og gruppeintervjuer.

Utvalg

Undervisningsopplegget er testet ut én gang i en fysikklasse ved en alminnelig og morderne norsk videregående skole, både med tanke på størrelse og tilsøkning. Totalt var det 33 elever i klassen, og kjønnsfordelingen var 15 jenter og 18 gutter. Faglærer og fysikklassen ble rekruttert gjennom masterveileders nettverk i skolesektoren, og kan dermed sies å være et bekvemmelighetsutvalg (Robson & McCartan, 2016, s. 160). Læreren har høyere utdanning i fysikk og lang undervisningserfaring i fysikk, matematikk og naturfag. Han har imidlertid mindre erfaring med å undervise i programmering i fysikk, men har kunnet støtte seg til en kollega. Denne kollegaen er satt inn som en ekstra ressurs fordi fysikklassen er noe større enn normalt. Ressurslæreren har mer erfaring med programmering, og får derfor gjerne sitte i førersete når et slikt tema behandles i fysikken. Det er også verdt å nevne at faglæreren og ressurslæreren var to av lærerne som deltok på intervjuundersøkelsen gjort som et forprosjekt til masteren. Klassen opplegget ble gjennomført i kan kategoriseres som en alminnelig norsk fysikklasse, både når det gjelder måloppnåelse og kjønnsfordeling, som vanligvis er på omtrent 60% gutter og 40% jenter.

Personvern

I forkant av utprøvingen av undervisningsopplegget besøkte jeg fysikklassen og informerte elevene om opplegget og studien. Her ble elevene kjent med eget personvern og hva det ville innebære for dem å delta i studien, samt at de når som helst kunne trekke seg i henhold til retningslinjer (Kvale & Brinkmann, s. 97, NESH, 2021, NTNU, 2019). Elevene ble også gjort oppmerksomme på at innsamlet data, elevarbeider, lydopptak av diskusjoner og intervjuer ville anonymiseres, samt at faglæreren deres ikke kunne bruke det som vurderingsgrunnlag i faget. Det ble også presisert at elevene ikke fikk fritak fra undervisningen dersom de ikke ønsket å delta i studien, og at alle måtte levere inn arbeid fra timen og utfylt spørreskjema. Imidlertid var det frivillig om de ønsket å bli tatt lydopptak av mens de arbeidet med oppgaver

i timen, og å delta på gruppeintervju. Dette ble formidlet muntlig, men også skriftlig gjennom et samtykkeskjema (se Vedlegg A). Ved å signere samtykkeskjemaet, samtykket elevene til å bli tatt lydopptak av i timen og/eller delta på et gruppeintervju med lydopptaker. Studien ble meldt til Norsk senter for forskningsdata (NSD) og godkjent i forkant av gjennomføringen i klasserommet.

Siden to av datainnsamlingsmetodene: lydopptak av klassediskusjoner og gruppeintervju, inkluderte innsamling av personopplysninger fra elevene, fikk de betenkningstid til å avgjøre om de ønsket å delta i denne delen av studien. Ved gjennomgang av signerte samtykkeskjema, fremkom det at 31 av 33 elever samtykket til å bli tatt lydopptak av i timen. I tillegg ønsket 7 elever å delta på gruppeintervju, hvorav 3 elever senere meldte avbud grunnet uforutsette hendelser.

Spørreskjema som datainnsamlingsmetode

Spørreskjema egner seg godt til å innhente informasjon fra en større gruppe mennesker (se for eksempel Robson & McCartan, 2016, s. 248). Siden jeg var interessert i elevenes opplevelse av undervisningsopplegget, og at det hadde vært tidkrevende å gjennomføre intervjuer med alle elevene, var det naturlig å benytte spørreskjema som datainnsamlingsmetode. Det har blitt gjort lignende studier om programmering i fysikkundervisning, der et spørreskjema har blitt utformet. Under utviklingen av eget spørreskjema tok jeg derfor utgangspunkt i et skjema utarbeidet av Nordby (2019), da hans studie kan sies å ha produsert valide resultater. Selv om flere av spørsmålene i spørreskjemaet er «lånt», var det viktig å vurdere om spørsmålene ville fungere med respondentene i denne studien, samt om de ville generere gode data (Oppenheim, 1992, s. 55). Både Nordby (2019) og jeg hadde respondenter som går i 2. klasse på videregående og har fysikk 1 som programfag. Dermed kan respondentene våre sies å være nokså like. Når det er sagt hadde det likevel vært fordelaktig å ha pilotert spørreundersøkelsen (Oppenheim, 1992, s. 55). Dette ble av praktiske årsaker ikke gjennomført.

Studiens spørreskjema (se Vedlegg B) består av både lukkede og åpne spørsmål. På de lukkede spørsmålene ble det brukt både intervall-skalaer og semantisk, bipolare skalaer i form av avkrysningsbokser (se for eksempel Oppenheim, 1992; Robson & McCartan, 2016, s. 311-314). I forsøk på å unngå vanlige feilkilder i forbindelse med faste svaralternativer, har jeg passet på at svaralternativene (1) stemmer overens med spørsmålet, (2) er gjensidig

utelukkende, (3) er realistiske, (4) er uttømmende og (5) er balanserte (Bjørndal, 2011, s. 106; Robson & McCartan, 2016, s. 259). For å sikre at spørsmålene var uttømmende, inkluderte jeg et åpent tekstfelt, der respondentene kunne skrive inn svar som ikke var nevnt i alternativene, eller skrive en utbroderende kommentar om alternativet de krysset av for. Tilsvarende ble kravet om balanserte svaralternativer oppfylt ved å gi like mange positive som negative svaralternativer (Bjørndal, 2011, s. 107). Åpne spørsmål ble benyttet der det var vanskelig å forutsi respondentenes svar. Dermed sto respondentene ganske fritt til å svare det de måtte ønske (se for eksempel Krosnick, 2018, s. 445; Oppenheim, 1992, s. 112). Når respondenter får såpass stor frihet, kan det være vanskelig å vite hvor lange svar det forventes at man avgir. For å gi respondentene et inntrykk av forventet svarlengde, ble det satt av en viss plass hvor de kunne skrive svaret sitt (Oppenheim, 1992, s. 112).

Bjørndal (2011) fremhever at respondentene må forstå spørsmålsformuleringen og begrepene som brukes i spørreskjema for å kunne avgi fornuftige svar. I tillegg bør spørreskjemaet oppleves som oversiktlig og simplistisk, slik at respondentene opplever det som enkelt å besvare. Spørsmålene på spørreskjemaet er relativt korte og inneholder et dagligdags språk. Dette kan ha bidratt til at elevene opplevde spørsmålene som mindre skremmende. Ved å unngå å stille spørsmål med dobbelt innhold, var det tydelig hva respondentene skulle svare på, og de slapp dermed å «gjette» seg frem til hva spørsmålet handlet om (se for eksempel Oppenheim, 1992, s. 128). Jeg har også forsøkt å stille så nøytrale, ikke-ledende spørsmål som mulig, for å ikke legge føringer på respondentenes svar (Oppenheim, 1992, s. 128). Spørreskjemaet ble også utformet slik at elevene skulle oppgi hva de syntes var mest interessant med undervisningsopplegget, i tillegg til hva de opplevde som utfordrende. På den måten fikk elevene mulighet til å gi en ærlig evaluering av undervisningsopplegget, noe som kan ha vært med på å ivareta studiens troverdighet.

Å benytte spørreundersøkelse som datainnsamlingsmetode byr på utfordringer som kan true undersøkelsens troverdighet. Robson og McCartan (2016) peker på utfordringen «low response rate». «Low response rate» går ut på at respondentene har lav grad av deltakelse og lav svarprosent. Mulige følger av dette kan være at den innsamlede datamengden er utilstrekkelig for å besvare forskningsspørsmålet, eller at responsen ikke er fra en representativ del av utvalget (s. 248). Spørreskjemaet ble delt ut på papir, i håp om at dette kunne stimulere til en høyere responsrate. For å sikre elevenes anonymitet, ble de bedt om å ikke oppgi navn på arkene, noe

Oppenheim (1992) mener kan gi en høyere responsrate. Ved bruk av spørreskjema er det også fare for at respondentene ikke tar undersøkelsen på alvor (Robson & McCartan, 2016, s. 248). Tegn på dette kan være «tullesvar», som er relativt enkelt å oppdage, mens andre useriøse svar kan være vanskeligere å avdekke.

Både egne observasjoner og de besvarte spørreskjemaene tyder på at elevene tok undersøkelsen på alvor. Selv om det finnes eksempler på at enkelte elever oppga mindre seriøse svar, er det store flertallet av besvarelsene seriøse. Det at spørreskjemaet ble utdelt på papir, og at elevene var nødt til å svare direkte på arket, kan ha ført til at de kjente på et større alvor enn om de hadde besvart undersøkelsen på PC. Det er imidlertid viktig å påpeke at det ikke er en garanti for at dette var tilfellet. Responsraten på spørreskjemaet var høy og det var få blanke svar. Dette kan tyde på at elevene opplevde spørsmålene som relevante og forståelige.

Spørreskjemaet ga også informasjon om elevenes opplevelse av undervisningsopplegget, noe som ikke ble registrert med ren observasjon. Eksempelvis svarte flere elever at det mest interessante med opplegget var programmeringsoppgaven 2d (se Vedlegg F). Dette var en oppgave de i timen uttrykte at de syntes var vanskelig og strevde med å løse. Basert på dette kan spørreskjemaet sies å ha avdekket ny og uventet informasjon som ga et mer helhetlig bilde av elevenes faktiske opplevelse av opplegget, fremfor deres tilsynelatende opplevelse.

Oppgaveark som datainnsamlingsmetode

I skolen vurderes ikke elevene etter hva de *føler* at de kan, men etter fremvist kompetanse. Basert på dette var det ønskelig å samle inn elevenes besvarelser på oppgavene, for å få innsikt i kompetansen de sitter på. Hovedhensikten med dette var at undervisningsopplegget også kan analyseres ut fra hva elevene fikk til. Dette kan ansees som et mer konkret mål på hva elevene får til enn hva de *føler* at de kan. Satt på spissen kan en elev *føle* at hen forstår Newtons lover og hvordan man *tegner fritt-legeme-diagram* (FLD), uten at vedkommende klarer å vise denne «indre» kompetansen. Siden elevene skal vurderes på en eller annen måte, er en slik situasjon mindre heldig. Idealet kan på mange måter sies å være at elevene *føler* at de forstår og kan fagstoff, men at de også kan vise dette i en vurderingssammenheng.

Oppgavearkene (se Vedlegg E, F og G) elevene fikk utdelt i timen er utformet etter *prinsipp 4* om LIST-oppgaver. For at oppgavearkene skulle virke overkommelige og mindre skremmende, repeterer de innledende oppgavene kjent fagstoff. Altså er det lagt opp til at samtlige elever kan

komme i gang med oppgavene. At de første oppgavene er lavterskel, og i stor grad «holder elevene i hånden», kan ansees som fordelaktig, da læreren ikke vil ha tid til å hjelpe alle elevene i gang når klassen er såpass stor. De videre deloppgavene tar for seg nytt fagstoff, der kompleksiteten i oppgaven og vanskelighetsgraden er ment å øke utover i oppgavesettet. Elevene arbeider seg gjennom oppgavearket til dels ved hjelp av programmering, og ved å skrive ned svar og resultater direkte på oppgavearket. Elevbesvarelsene ble samlet inn ved slutten av hver time og gjennomgått. Dermed fikk jeg konkrete data om hva elevene hadde kommet frem til og svart på de ulike oppgavene. Videre ble datamaterialet analysert mer systematisk, noe som beskrives nærmere i kapittel 5.3.

Observasjoner fra timen tyder på at elevene kom i gang med oppgaveark 1 «Bil med konstant akselerasjon» (se Vedlegg E) i økt 1 og oppgaveark 2 «Vertikalt kast med luftmotstand» (se Vedlegg F) i økt 2, uten særlig støtte fra lærer. Mens oppgaveark 1 synes å ha vært passe vanskelig, viste det seg at oppgaveark 2 var litt vanskeligere. Her var det oppgave 2d som utmerket seg som spesielt vanskelig. I likhet med deloppgave 1a, fikk elevene i deloppgave 2d utdelt delvis ferdiglaget programkode som de skulle fullføre for å besvare oppgaven. En sentral forskjell mellom programmeringsoppgavene var antallet kodelinjer elevene måtte skrive, og kompleksiteten i dem. I oppgave 1a var nødvendige beregninger gjort, slik at elevenes oppgave var å fremstille allerede eksisterende data. Oppgave 2d krevde større fysikkforståelse enn oppgave 1a, i tillegg til at det krevdes at elevene skrev flere kodelinjer på egen hånd.

Som nevnt tidligere var det mange elever som ikke kom seg forbi deloppgave 2d, og dermed ikke fikk arbeidet med deloppgavene som tok for seg det nye fagstoffet. Basert på dette kan man si at elevene hadde behov for mer støtte enn de fikk. På den ene siden hadde det vært hensiktsmessig å gi elevene programkode som i større grad var ferdiglaget, slik at arbeidsmengden deres ble redusert. På den annen side kunne dette resultere i at elevene ble tildelt for lite arbeid, og at oppgavene ble for enkle. Det er krevende å «oppfylle» Vygotskys prinsipp om *den proksimale utviklingssonen* der elevene får noe å bryne seg på uten at det blir for vanskelig, i tillegg til at de gis enkle nok oppgaver som ikke er for lette. Med bedre kjennskap til klassen, hadde det trolig vært enklere å tilpasse vanskelighetsnivået på oppgavearkene og undervisningen til elevgruppen.

Observasjon som datainnsamlingsmetode

I tillegg til oppgaveark, spørreskjema og intervjuer, har ustrukturert observasjon blitt benyttet som datainnsamlingsmetode. Ustrukturert observasjon gjorde det mulig å være åpen for å oppdage ting underveis, som man på forhånd ikke hadde sett for seg (Bjørndal, 2011, s. 66). Selv observerte jeg den pedagogiske situasjonen jeg selv tok del i, som er det Bjørndal (2011) omtaler som «observasjon av andre orden» (s. 32). Siden observasjonsoppgaven er krevende, var det både ønskelig og verdifullt å trekke inn en annen observatør som kunne konsentrere seg om å observere, og selv ikke tok del i den pedagogiske aktiviteten (Bjørndal, 2011, 46). Faglæreren fikk derfor rollen som en «første ordens observatør» (s. 32).

Når observasjon benyttes som metode, er det viktig å være klar over at det er begrenset hvor mye vi kan ta innover oss, at det er til dels tilfeldig hva vi får med oss, og at hva vi observerer avhenger av tidligere erfaringer (Bjørndal, 2011, s. 74; Robson og McCartan, 2016, s. 331). På den måten er ikke observasjoner en tro kopi av virkeligheten, men små glimt av virkeligheten, sett gjennom en personlig linse. Til tross for dette, anså jeg det som viktig å loggføre egne observasjoner umiddelbart etter endt undervisningsøkt. Ved å skrive logg fikk jeg satt av tid til å skape en dypere forståelse og et mer overordnet bilde av oppleggets hendelser på en systematisk måte (Bjørndal, 2011, s. 64). En annen fordel med loggen, var at jeg når som helst kunne hente den frem og få innblikk i hva som hadde hendt, og i hvilken rekkefølge det hadde hendt. Faglærer og jeg satte også av tid til å dele observasjoner, noe jeg opplevde var en hjelp i å danne meg et mer fullstendig bilde av klasseromssituasjonen. Dette beskrives av Bjørndal (2011) som en av fordelene ved å dele erfaringer og sammenligne logger (s. 65).

Det at vi ikke kan observere alt og at hva vi observerer avhenger av tidligere erfaringer (Bjørndal, 2011, s. 74; Robson og McCartan, 2016, s. 331), var noe jeg fikk erfare. I samtale med faglærer fremkom det at selv om vi hadde observert samme situasjon, var det ulike ting vi bet oss merke i. Et eksempel på dette er at faglæreren syntes økt 1 og 2 gikk «mye bedre enn forventet», mens det for meg virket som økt 2 gikk dårligere enn forventet, siden flere av elevene ikke kom seg forbi programmeringsoppgaven 2d. En av årsakene til at vi satt igjen med to ganske forskjellige inntrykk, kan være at vi trådte inn i situasjonen med ulike forventninger. Faglæreren kjenner elevene bedre enn det jeg gjorde, og visste dermed mer om hva de kunne fra før og hva som kunne forventes av dem. Ved å dele både like og ulike erfaringer, puslet vi sammen et mer helhetlig bilde av situasjonen enn det jeg kunne fått til på egen hånd.

Lydopptak som datainnsamlingsmetode

Det ble gjort lydopptak av elevdiskusjonene i økt 2 og 3, mens elevene arbeidet med oppgaver i klasserommet, og i forbindelse med gruppeintervjuene. Dette var for å fange øyeblikk som forsvinner like fort som de oppstår i løpet av en undervisningsøkt eller et intervju (Bjørndal, 2011, s. 75). Lydopptak er også en datainnsamlingsmetode som i stor grad speiler virkeligheten og kan sikre at informasjon ikke går tapt. Det er likevel viktig å påpeke at selv om lydopptak gjerne oppfattes som en gjengivelse av hendelsene som utspilte seg, er også denne observasjonsmetoden kun en versjon av virkeligheten, og ikke en tro kopi.

Elevsamtaler fra timen

Under undervisningsøkt 2 og 3 ble lydopptakeren håndtert av min medstudent Juni, som beveget seg mellom ulike grupper underveis i økten. Siden Juni ikke kunne bli stående i altfor lang tid hos hver gruppe, måtte hun hele tiden ta stilling til om hun skulle bevege seg videre til en annen gruppe eller bli stående. Disse avgjørelsene ble i hovedsak tatt ut fra skjønn. Hun beveget seg til en ny gruppe dersom samtalen i gruppen hun observerte dabbet av eller at gruppen sluttet å stille spørsmål. Basert på dette er det rimelig å anta at samtalen i gruppen Juni beveget seg bort fra, enten døde ut eller at det oppsto en ny samtale. Det er likevel ingen garanti for at gruppen ikke gjenopptok samtalen, og at nye øyeblikk som hadde vært interessante å fange opp, ikke ble dokumentert. Juni kunne også selv bestemme hvor ofte hun skrudde av og på lydopptakeren, og dermed lengden på opptakene. Dette resulterte i at det både ble gjort lengre lydopptak, der flere forskjellige elevsamtaler inngikk, og kortere opptak som kun inkluderte én elevsamtale. Total varighet av lydopptakene var 73 minutter, og ble gjort under undervisningsøkt 2 og 3, som til sammen utgjorde 135 minutter.

Lydopptakerens teknologiske begrensninger kan påvirke observasjonene. Bjørndal (2011) beskriver blant annet at lydopptakeren kan ha vansker med å sile ut bakgrunnsstøy, og identifisere hva som blir sagt når flere personer snakker samtidig og danner en lydgrøt (s. 78). Selv opplevde jeg at det til tider var utfordrende å skille mellom de ulike samtalene lydopptakeren fanget opp, samtaler som foregikk parallelt i et klasserom der elevene satt nokså nærme hverandre. Tilstedeværelsen av både selve lydopptakeren og dens operatør kan også ha påvirket observasjonene. Av etiske hensyn gikk Juni med lydopptakeren synlig, uten å skape unødvendig fokus rundt den. Dermed ble observasjonen gjort med det Bjørndal (2011)

omtaler som «høy grad av åpenhet» (s. 47). Elevene var også innforstått med at opptakene ville anonymiseres og ikke kunne brukes som vurderingsgrunnlag i faget, og dermed verken være til skade eller sjenanse for dem (Bjørndal, 2011, s. 80). Det at Juni under første møte med klassen presenterte seg, kan ha gjort at elevene fikk mer tillit til henne. Dette kan ha bidratt til at elevene i stor grad oppførte seg slik de pleier, noe som videre kan ha bidratt til å produsere valide resultater. Observasjoner fra undervisningsøkten tyder også på at elevene var komfortable med Junis nærvær, da de ikke nølte med å stille henne spørsmål.

Bruken av båndopptaker under økt 2 og 3 bidro til å fange situasjonene slik de faktisk oppsto. Her var det spesielt verdifullt å ha en person som Juni, som hadde som primæroppgave å operere båndopptakeren. Gjennom arbeidet hun gjorde, ble informasjon som ellers kunne blitt «oversett» eller «glemt», fanget opp og lagret. Dette gjorde at jeg kunne gå tilbake, spille av lydfiler, og høre hva og hvordan elevene diskuterte seg imellom når de arbeidet med oppgavene i timen (Braun & Clarke, 2022, s. 275). Jeg opplevde også at det å høre gjennom lydopptakene kunne trigge egne observasjoner, som ikke hadde blitt loggført. Lydopptakeren synes verken å ha påvirket eller forstyrret elevene nevneverdig underveis i timen eller under intervjuene. I utgangspunktet skulle Juni operere lydopptakeren uten å ta del i elevdiskusjonene. Underveis i økt 2 ble det nødvendig å endre arbeidsbeskrivelsen hennes, da antall elever som trengte hjelp var for stor til at jeg rakk bort til alle. Resultatet ble at Juni fikk besvare spørsmål fra elevene og hjelpe dem, i tillegg til å operere lydopptakeren.

Junis deltakelse i elevdiskusjonene kan på en måte ansees som uheldig, da dette kan ha påvirket elevene og de innsamlede dataene. På den annen side ville det ikke vært etisk forsvarlig overfor elevene å nekte henne å hjelpe dem. Dersom elevene ikke hadde hatt mulighet til å få hjelp av Juni, kunne dette vært til skade for elevenes forståelse i fysikkfaget. Dermed ville det vært en tydelig ulempe for elevene å delta i studien, noe som ikke er i tråd med de etiske retningslinjene som bør følges i en slik studie (se for eksempel Robson & McCartan, 2016, s. 212). At elevene kanskje kom til å trenge hjelp fra Juni, i tillegg til meg selv, var et scenario jeg hadde sett for meg på forhånd og orientert henne om. Dette i kombinasjon med at Juni er lektorstudent i fysikk og matematikk, med kompetanse innen programmering, gjorde at hun kunne hjelpe elevene på en hensiktsmessig måte. Videre ville det også vært uetisk å la elevene arbeide med oppgaver, spesielt programmeringsoppgaven, uten å til slutt gi dem forslag til hvordan oppgavene kunne løses. Basert på dette fikk faglæreren tilsendt løsningsforslag på samtlige oppgaver, inkludert

programmeringsoppgaver, som han kunne dele med elevene.

Det bør påpekes at elevene jobbet mot skjerm og pekte på ting som ikke tydelig kommer frem på lydopptaket. Når elever eksempelvis sier «kanskje det er denne som er feil» eller «vi flytter denne hit», vet vi ikke akkurat hva «denne» er. Lydopptakene gir altså ikke et fullstendig bilde av situasjonen de ble tatt opp i, men er heller en indikasjon på hvordan elevene arbeidet i timen. Basert på dette fikk jeg lite nytte av opptakene av elevsamtalene fra timen.

Gruppeintervjuer

Av praktiske årsaker ble de to gruppeintervjuene, bestående av to elever i hvert intervju, gjennomført i etterkant av økt 2, men før økt 3. Det ble gjort lydopptak under gruppeintervjuene for å sikre at informasjon ikke ble oversett eller gikk tapt (se for eksempel Postholm, 2010, s. 170; Robson & McCartan, 2016, s. 170). Dette gjorde at jeg kunne løsrive meg fra notatblokken, og kunne fokusere og være til stede under hele samtalen (Postholm, 2010, s.82 og s. 305; Robson & McCartan, 2016, s. 305). For å dokumentere intervjuene i sin helhet, ble det gjort et sammenhengende lydopptak av hvert intervju. Total varighet på de to lydopptakene var 97 minutter.

En intervjuguide (se Vedlegg C) ble utarbeidet i forkant av gruppeintervjuene. Under forarbeidet ble det tydelig hvilke momenter jeg ønsket informantens tanker og refleksjoner rundt. Spørsmålene i intervjuguiden retter seg hovedsakelig mot undersøkelsens forskningsspørsmål, hvor elevens opplevelse av undervisningsopplegget var i fokus. Intervjuguiden innledes med åpne spørsmål om elevens generelle opplevelse av fysikkfaget og deres erfaringer med numeriske metoder og programmering. Disse spørsmålene var ment å gi bakgrunnsinformasjon om elevenes tanker og holdninger i faget, og sette intervjuet inn i en større kontekst. De neste spørsmålene rettet seg mot elevenes opplevelse av undervisningsopplegget. Denne delen av intervjuguiden inkluderer konkrete spørsmål om hvordan elevene opplevde de ulike delene av opplegget, og hva de synes var interessant og hva som var vanskelig. Den siste delen av intervjuguiden retter fokus mot elevenes utbytte av undervisningsopplegget, samt hvorvidt de synes numeriske metoder og programmering bør være en del av fysikkfaget.

Intervjuguiden var ment som en støtte under intervjuet, og ikke en kokebokoppskrift som skulle følges til punkt og prikke. Under et intervju påvirkes informantene og intervjueren kontinuerlig

av hverandre (se for eksempel Kvale & Brinkmann, 2015, s. 52; Robson & McCartan, 2016, s. 170). Følgelig er det helt essensielt at intervjueren utnytter fleksibiliteten ved et intervju, og kan løsrive seg fra intervjuguiden, lytte, og stille oppfølgingsspørsmål (Postholm, 2010, s.82 og s. 305; Robson & McCartan, 2016, s. 305). Basert på dette var det ønskelig å gjennomføre semi-strukturerte gruppeintervjuer (se for eksempel Postholm, 2010, s. 290).

Å gjennomføre gruppeintervjuer kan ansees som fordelaktig, da informantene kan bygge videre på hverandres innspill, og at kritiske spørsmål kan føre til en mer produktiv samtale. Presset på hver enkelt informant kan også reduseres, siden det ofte ikke er behov for at alle informantene svarer på hvert spørsmål (Robson & McCartan, 2016, s. 299-300). På den annen side påvirkes informantene av hverandre og av intervjueren. Eksempelvis kan informantene skape en felles forståelse for temaet som diskuteres. Det er også en viss fare for at enkeltindivider dominerer intervjuet, og at motstridende ideer og nyanser ikke kommer frem (Kvale & Brinkmann, 2015, s. 114; Robson & McCartan, 2016, s. 299-305). Det er dermed en viss fare for at informantene holder tilbake informasjon de mistenker at andre kan være uenige i (se for eksempel Kvale & Brinkmann, 2015, s. 52)

Gruppesammensetning under intervjuer kan også påvirke resultatene og dermed studiens troverdighet (Kvale & Brinkmann, 2015, Robson & McCartan, 2016). Det var derfor viktig å sette sammen gode grupper, der elevene følte seg trygge og turte å ytre egne meninger. Faglæreren var behjelpelig med dette og kom med forslag til gruppeinndeling som i størst mulig grad oppfylte de nevnte kriteriene. *Intervju 1* hadde en varighet på 1 time og besto av to gutter, mens *intervju 2* varte i 37 minutter og besto av to jenter. I intervjuene fremkom det at elevene hadde ulike ambisjoner og interesse for fysikkfaget. Guttene uttrykte at de syntes fysikk var spennende og at de arbeidet mye med faget, med ønske om å forstå det best mulig. På den annen side uttrykte jentene at de syntes faget «ikke var det verste, men ikke det beste», og at de ikke var så opptatt av å skjønne alt de gikk gjennom i timen. Selv om det ikke var et krav om at informantene skulle representere en god spredning av ferdighetsnivå i faget, kan det virke som kravet likevel var relativt godt oppfylt.

Både mengden og kvaliteten på den innsamlede informasjonen avhenger av klimaet i intervjuet (Bjørndal, 2011, s. 98; Kvale & Brinkmann, 2015, s. 114). Intervju som kvalitativ forskning er altså svært kontekstavhengig, og må forstås utfra mikrososiologiske kontekster som miljø og situasjon. Det at intervjuene ble gjennomført på et grupperom, vegg i vegg med

fysikklasserommet, kan ha gjort at de opplevde miljøet som kjent og trygt. Intervjuet ble innledet med pizza og småprat. Praten gikk lett, og det virket som elevene følte seg trygge på situasjonen og meg som intervjuer. Informantene ble også betrygget i at uttalelsene deres kunne utelates fra transkripsjonen dersom de ønsket det. Dette, i kombinasjon med interessen jeg viste for informantenes uttalelser og refleksjoner, og den menneskelige relasjonen under intervjuet, kan ha bidratt til en mer åpen samtale (Bjørndal, 2011; Kvale & Brinkmann, 2015).

Spørsmålsstillingen under intervjuet kan ha blitt påvirket av intervjuerens egne oppfatninger (se for eksempel Robson & McCartan, 2016, s. 303; Tjora, 2018, s. 83). Ifølge Robson og McCartan (2016) er de fleste informanter velvillige i møte med intervjueren. Med dette i bakhodet prøvde jeg å unngå å stille ledende spørsmål, slik at det ikke ble lagt føringer på informantenes svar. Om et spørsmål et ledende avgjøres av faktorer som innholdet i spørsmålet som stilles, intervjuerens fremtoning og kroppsspråk (Kvale & Brinkmann, 2015, s. 47; Robson & McCartan, 2016, s. 288). Ved å stille ledende spørsmål kan man sjekke intervjusvarenes reliabilitet og verifisere intervjuerens fortolkninger, gjennom eksempelvis «member checking» (Kvale & Brinkmann, 2015, s. 201; Robson & McCartan, 2016, s. 172 og 288). På den måten kan det å stille ledende spørsmål styrke studiens troverdighet.

Det har også blitt tatt overordnede metodiske grep for å ivareta studiens troverdighet. Som forsker trer man inn i en studie med egne oppfatninger (se for eksempel Braun & Clarke, 2022; Robson & McCartan, 2016, s. 168-173). Dette kan føre til forventningsskjevhet. I forsøk på å redusere forventningsskjevhetens innflytelse på studien, har jeg vært bevisst egne oppfatninger, og vært åpen for alle slags oppdagelser, både dem som støtter og strider med egne holdninger (Postholm, 2010, s. 86; Robson & McCartan, 2016, s. 148-149). Dette, i kombinasjon med å presentere de ulike trinnene i forskningsprosessen, som rådata og kodeprosessen, gir leseren mulighet til selv å ta stilling til undersøkelsens troverdighet (Postholm, 2010, s. 173; Robert & McCartan, 2016, s. 172-173; Tjora, 2018, s. 80-81).

Intervjuguiden ble som planlagt ikke fulgt slavisk, men brukt som støtte under gruppeintervjuene. Dette var for å få til en så naturlig samtale som mulig, der informantene fikk mulighet til å snakke mer fritt og dra inn ulike momenter der det passet dem, noe som er med på å ivareta studiens troverdighet. Siden gode intervjuer innebærer at informantene får snakket ut (Kvale & Brinkmann, 2015, s. 272-273; Postholm, 2010, s. 287; Robson & McCartan, 2016, s. 305), anså jeg det som viktig å unngå å avbryte informantene så fort de driftet, da sidesprang

kan avdekke interessant informasjon. Intervjuguiden ble dermed i hovedsak brukt til å peile samtalen tilbake på rett spor dersom en informants anekdoter ble i overkant lange.

Guttene fra *intervju 1* (heretter kalt Felix og Hugo) hadde betraktelig mer på hjertet enn jentene fra *intervju 2* (heretter kalt Lisa og Ruby), noe som reflekteres i varigheten av intervjuene. Intervju 1 hadde en varighet på 1 time, mens intervju 2 varte i 37 minutter. I det første intervjuet deltok informantene aktivt og spilte på hverandres innspill. Her er et eksempel:

> **JXS:** Selv om dere er mange (i klassen), følte dere at dere fikk hjelp da dere ba om det?
> **Felix:** Jeg vil starte med å si at det er litt for mange elever i klassen, så det tar litt for lang tid fra vi rekker opp hånden til vi får hjelp. Men det er jo ikke noe du får gjort så mye med, men når du først kommer så synes jeg vi har fått god oppfølging... og litt mer spørrende hjelp som «hva tror du er riktig her» enn å liksom bare si hva som er galt. Prøve å forklare og få oss til å tenke selv.
> **Hugo:** Som Felix sa er vi 32 elever i klassen og vi er for mange for 1 lærer...

Som følge av informantenes aktive deltakelse, hendte det at de kom meg i forkjøpet, og besvarte spørsmål jeg ennå ikke hadde rukket å stille. Generelt var *intervju 1* lite preget av en spørsmål-svar-rytme, og opplevdes dermed mer som en samtale.

Til tross for at klimaet i de to intervjuene var nokså likt, hadde *intervju 2* et noe annerledes forløp. Her ser vi at intervjuet hadde en tydeligere spørsmål-svar-rytme, hvor informantene ga kortere svar uten å utbrodere noe særlig:

> **JXS:** Synes dere det er mye å gjøre i fysikkfaget?
> **Lisa:** Eh. Ja.
> **Ruby:** Med [navn på fysikklæreren] ja. Fordi etter hver time får vi oppgaver vi må gjøre og levere. Vi har mange innleveringer.
> **JXS:** Lærer dere noe av det da?
> **Ruby:** Ja
> **Lisa:** Jeg synes ja. Vi lærer mest av det, siden vi må gjøre oppgavene. At de liksom ikke bare blir anbefalt. For da tenker vi om vi skal gjøre det eller ikke, men nå må vi liksom gjøre det. Og da sitter det bedre i hjernen synes jeg.
> **JXS:** Så det er både bra og dårlig at dere får leksene?
> **Ruby:** Ja.
> **Lisa:** Ja, du får bra resultater ut av det til slutt.
> **JXS:** Er oppgavene dere får på slutten av timen oppgaver dere må gjøre hjemme?
> **Lisa:** Ja.

Utdragene viser at de to intervjuene utspilte seg forskjellig. Dette kan eksempelvis skyldes faglig ferdighetnivå, kjønnsforskjeller (Angell et al., 2019, s. 106-109), gruppesammensetning

og/eller personlighet. Det er viktig å understreke at leseren ikke presenteres for intervjuenes forløp for å bedømme om intervjuene var gode eller ikke. Hensikten er heller å vise at intervjuer kan forløpe ulikt, selv med likt utgangspunkt, noe som er en viktig faktor i kvalitativ forskning. Ved å få innsikt i hvordan intervjuene utspilte seg, kan leseren også bedre ta stilling til undersøkelsens troverdighet (Postholm, 2010, s. 173; Robert & McCartan, 2016, s. 172-173).

Transkripsjon av lydopptak

Både lydopptakene av elevsamtalene fra timen og intervjuene har blitt transkribert. Selv om det å transkribere hele intervjuer verken er tidseffektivt eller nødvendigvis ønskelig, er det viktig at forskeren er godt kjent med datamaterialet (se for eksempel Postholm, 2010, s. 469-471; Robson & McCartan, 2016, s. 305). Basert på sistnevnte har alle lydopptak gjennomgått en fullstendig transkripsjon (Braun & Clark, 2022), da dette ga meg muligheten til å se over intervjusvarene, samt vurdere egen spørsmålsstilling i analyseprosessen. For å ivareta informantenes personvern, er transkripsjonen gjort på bokmål. Videre har dialektord og formuleringer som kan avsløre informantens identitet blitt fjernet eller erstattet for å sikre informantens anonymitet (Kvale & Brinkmann, 2015, s. 97). Kvale og Brinkmann (2015) beskriver hvordan transkribert muntlig språk har en tendens til å virke usammenhengende og fordummende (s. 97). Følgelig har jeg enkelte steder i transkripsjonen skrevet om informantenes uttalelser til et mer skriftlig språk, men passet på å beholde essensen i uttalelsene (Kvale & Brinkmann, 2015, s. 213). Jeg har imidlertid beholdt pauser og ord som antyder usikkerhet eller tankesprang, slik at leseren får best mulig innsikt i informantenes tankeprosesser under intervjuet (Postholm, 2010, s. 469-471).

5.3 Analysearbeid

Undersøkelsens datamateriale har gjennomgått en refleksiv tematisk koding, med både en induktiv og deduktiv tilnærming (Braun & Clarke, 2022; Tjora, 2018). Analyseenheten har vært elevgruppen, selv om synspunktene deres er ulike på enkelte punkter. Jeg har likevel forsøkt å skille mellom hvilke elever som sa hva, slik at elevenes uttalelser kan presenteres på en ryddig og oversiktlig måte. I transkripsjonene av elevsamtalene kalles deltakerne «Elev 1-8». Her kan samme deltaker ha blitt tildelt ulikt navn underveis, siden det kunne være vanskelig å skille mellom dem på lydopptaket. Dette er ikke problematisk siden analyseenheten som nevnt er

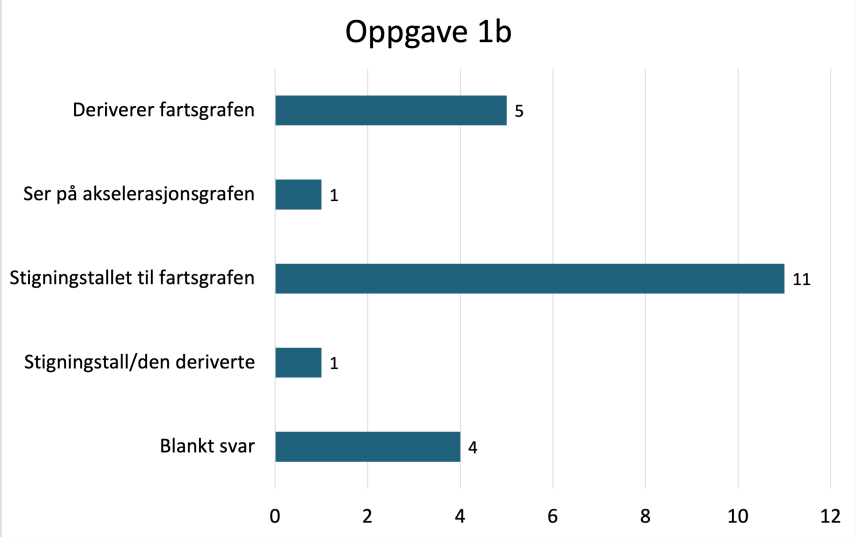
elevgruppen, og ikke enkelteleven. I transkripsjonen av gruppeintervjuene var det mer naturlig å skille mellom de ulike deltakerne, ved å tildele dem de fiktive navnene Felix, Hugo, Lisa og Ruby. Følgelig kan man ikke koble elever i intervju og elevsamtaler.

Analyse av elevbesvarelser på oppgaveark og spørreskjema

Av praktiske årsaker foregikk analysen av oppgavearkene og spørreskjemaet i Excel, som i essens likner bruk av et program for kvalitativ analyse som nVivo. Elevenes besvarelser ble gjennomgått én etter én, samt kodet tematisk og induktivt (se for eksempel Braun & Clarke, 2022; Robson & McCartan, 2016, s. 461, Tjora, 2018) . I kodearbeidet hentet jeg ut essensen av elevenes svar på hver deloppgave og tildelte svaret en passende, induktiv kode. De fleste av kodene ble opprettet tidlig i kodearbeidet, da det hyppig dukket opp elevbesvarelser med noe ulik essens enn i allerede eksisterende koder. Dersom essensen av en besvarelse passet inn i en eksisterende kode, var planen at besvarelsen skulle tildeles denne koden (Braun & Clarke, 2022, s. 77-78). Likevel ble ikke alle elevbesvarelser, som faglig sett beskrev det samme, nødvendigvis tildelt samme kode. Årsaken til dette var at jeg ikke visste om elevene så de samme faglige sammenhengene som meg. Denne måten å kode på kan sies å ha redusert risikoen for å overtolke elevenes svar, og produsere ugyldige resultater som fremviser funn som egentlig ikke var der.

Et eksempel på at flere elevsvar som faglig sett omhandler det samme, ikke har fått tildelt samme kode, er vist i Figur 5-2. Figuren viser at enkelte elever løste oppgaven ved å derivere fartsgrafen, mens andre så på stigningstallet til fartsgrafen. Rent faglig beskriver de to løsningsmetodene det samme, siden den deriverte av en funksjon per definisjon er stigningstallet til tangenten til grafen. Likevel kan jeg ikke med sikkerhet vite at elevene så denne sammenhengen. I prinsippet kan elevene både ha visst at akselerasjonen er den deriverte av farten og at akselerasjonen er stigningstallet til fartsgrafen, uten å nødvendigvis forstå sammenhengen mellom beskrivelsene. Ved å holde de to beskrivelsene adskilt, får leseren et innblikk i hva elevene faktisk svarte, fremfor forfatterens fortolkning av svarene, som kan avvike fra det elevene ønsket å formidle. I tillegg kan man tydeligere se hvilke elever som tar opp begge løsningsmetodene i svarene sine og ser sammenhengen mellom dem.

Det faktum at mange av elevene ikke fikk til oppgave 2e (se Vedlegg F), som ble gitt i økt 3, fikk betydning for analysen og kodingen av elevbesvarelsene. De kodede elevbesvarelsene



Figur 5-2: Oversikt over elevenes svar på oppgaven: «Vis at akselerasjonen er 1,7 m/s^2, og beskriv hvordan du kom frem til svaret». Tallet ved siden av hver søyle viser antall elever som benyttet den bestemte løsningsmetoden.

har fått tildelt merkelappene (1) «uten ferdiglaget programkode» og (2) «med ferdiglaget programkode». Her viser den første merkelappen til elevbesvarelsene fra økt 2, mens den andre merkelappen viser til elevbesvarelsene fra økt 3.

Kodingen av elevenes svar på spørreskjemaet ble i hovedsak gjort på tilsvarende måte som besvarelsene på oppgavearkene. Altså ble svarene på spørreskjemaet tilordnet egne koder, unntatt når de var tilnærmet identiske med eksisterende koder. Siden skjemaet (se Vedlegg B) også inneholdt spørsmål der deltakerne avga svar ved å velge ett av de predefinerte svaralternativene, ble disse besvarelsene behandlet ved å telle opp hvor mange ganger hver av dem var krysset av.

Analyse av elevsamtaler fra timen

Siden jeg under undervisningsøkten observerte at de forskjellige elevsamtalene artet seg ulikt, tenkte jeg at det å se nærmere på samtalemønstre kunne bidra til å besvare forskningsspørsmål 1. Kodingen av transkripsjonene ble gjort i programvaren nVivo som støtter kvalitative data. Programmet ga meg en oversikt over kodene som ble brukt, og hvor ofte hver av dem ble registrert. Jeg kodet i hovedsak transkripsjonene deduktivt (Braun & Clarke, 2022; Tjora, 2018) ut fra predefinerte koder om samtalestrukturer som Bungum et al. (2018) presenterer i sin studie. I deres studie omtales den første kategorien som *kumulativ*, og kjennetegnes ved at elevene bygger videre på hverandres innspill, uten å bringe inn nye syn eller nyanser.

Elevdiskusjoner kan også kategoriseres som *eksplorerende*, der deltakerne vurderer hverandres innspill på en kritisk måte. Her utfordres også påstander og forslag, for at gruppen sammen skal komme til en «felles løsning». Videre kan diskusjoner preges av *frittstående påstander*, der deltakerne har en tendens til å snakke forbi hverandre, og det å komme frem til en «felles forståelse» ikke ansees som like viktig som i *eksplorerende diskusjoner*. Den fjerde kategorien er *bekreftende snakk*, der det en elev sier repeteres av en annen.

Siden en diskusjon sjelden faller under bare én av de fire kategoriene, ble diskusjonene kodet etter den kategorien som var mest fremtredende, og kodene er dermed gjensidig utelukkende. Eksempelvis ble følgende utdrag kodet som en *bekreftende* diskusjon, selv om det også finnes aspekter av andre samtalemønstre, som *eksplorerende*:

> **JXS:** Synes dere det er mye å gjøre i fysikkfaget?
> **Elev 1:** Jo mindre dt, jo flere på en måte oppdateringer gjør den i utregningen. Så nå har den bare 1, 2, 3, 4, 5 utregninger.
> **Elev 2:** dt er nøyaktigheten da?
> **Elev 1:** Ja, på en måte. Den bestemmer nøyaktigheten på hvor mange utregninger som blir gjort da. Tror jeg. Så jo mindre tallet er, jo flere utregninger gjør den.
> **Elev 2:** Så når det er et mindre tall, så blir det mindre... det blir mindre punkter da på en måte. Så det blir vanskelig å gi en mer nøyaktig...
> **Elev 1:** Hvis tallet er kjempelavt så blir det nesten som at du regner alle, tror jeg.
> **Elev 2:** Det ser jo sånn ut.
> **Elev 1:** Hvis du endrer til 0.01. så blir det jo. . . .mer og mer nøyaktig da.

I utdraget ovenfor diskuterer elevene hvordan tidssteget dt påvirker ballens beregnede posisjon. Her ser vi at Elev 1 kommer med en forklaring, som Elev 2 stiller et oppfølgingsspørsmål til for å forstå, og deretter på mange måter godtar.

Underveis i kodearbeidet ble jeg oppmerksom på at flere av diskusjonene ikke passet inn i noen av hovedkategoriene. Felles for disse diskusjonene var at læreren deltok aktivt i samtalen, noe som gjerne innebar at elevene spurte om hjelp med oppgaven, eller at de ønsket bekreftelse på at resultatet var «rett». Basert på dette var det naturlig å opprette en ny kode «samtale med lærer» som hadde utspring fra datamaterialet (Braun & Clarke, 2022, s. 100). Følgelig inneholder det ferdigkodede datamaterialet forutbestemte, deduktive koder, i tillegg til induktive koder basert på dataene. I etterkant av kodingen, oppdaget jeg at de predefinerte kodene i liten grad bidro til å besvare studiens forskningsspørsmål. Av den grunn har jeg latt være å analysere

samtalestrukturer videre. Jeg fant imidlertid ut av at koden «samtale med lærer» kunne bidra til å besvare forskningsspørsmålene, og valgte derfor å fokusere på denne koden i det videre arbeidet.

Analyse av gruppeintervjuer

Kodingen av gruppeintervjuene foregikk også i nVivo, og har både deduktive og induktive aspekter (Tjora, 2018). Analysens deduktive aspekter innebærer at jeg anvendte relevant teori om numeriske metoder, programmering og situert læring, for å lage hovedkategorier som de induktive kodene kunne innlemmes i (se for eksempel Braun & Clarke, 2022, s. 6; Robson & McCartan, 2016). Teorien ble også brukt i utarbeidelsen av intervjuguiden, noe som kan ha påvirket spørsmålsstillingen og dermed informantenes svar. På den måten kan teorien sies å ha konstruert et rammeverk som de induktive kodene arrangeres etter (Braun & Clark, 2022, s. 267; Postholm, 2010, s. 68 og s. 469). Med utgangspunkt i intervjuguiden var det å forvente at informantenes svar kom til å omhandle minst én av de fire hovedkategoriene: (1) elevenes opplevelse av å arbeide med undervisningsopplegget, (2) elevenes opplevde utfordringer tilknyttet opplegget, (3) elevenes motivasjon, (4) elevenes tanker rundt kompetansemålet om programmering og numeriske metoder, og (5) elevenes forslag til forbedringer av undervisningsopplegget.

Totalt inneholder transkripsjonene 28 unike, induktive koder. Det videre kodearbeidet besto i å slå sammen de induktive kodene i *grupper*, som deretter ble plassert i en av *hovedkategoriene* (Braun & Clarke, 2022, s. 34-37, Tjora, 2018, s. 36- 54). I Tabell 5-1 (se neste side) finner leseren *noen* konkrete eksempler på hvordan utdrag fra transkripsjonen har blitt tolket og kodet, dertil hvilken gruppe og hvilket hovedtema de er plassert under. I kodingen av informantens uttalelser var også det teoretiske rammeverket til hjelp. Dermed er det en viss risiko for at rammeverket ubevisst har blitt påtvunget datamaterialet, og kan ha truet studiens troverdighet (Postholm, 2010, s. 170)

I den videre analysen av datamaterialet har kodenes innhold, og frekvensen de dukker opp med i transkripsjonen, påvirket hvilke koder det var naturlig å utforske videre. Årsaken til at bestemte koder dukker opp oftere enn andre, kan være at intervjuguiden inkluderte flere spørsmål om bestemte koders innhold (Braun & Clarke, 2022, s. 89-90). Altså kan jeg som intervjuer ha påvirket undersøkelsens resultater (Robson & McCartan, 2016, s. 290-291). Dermed er

Tabell 5-1: Oversikt over hvordan noen konkrete utdrag fra transkripsjonene av gruppeintervjuene er tolket og kodet, samt hvilken gruppe og hovedkategori de er plassert i.

Hovedkategori	Gruppe	Induktiv kode	Eksempel fra transkripsjonen
(1) Elevenes opplevelse av å arbeide med undervisningsopplegget	Lært nye løsningsteknikker	Nye løsningsteknikker	«...når vi lærer programmering så lærer vi jo en annen måte å løse en oppgave på da. Så sånn sett får vi jo bruk for det i fysikken»
(2) Elevenes opplevde utfordringer tilknyttet opplegget	Kodeforståelse	Sette seg inn i andres kode	«Selv om vi noen steder bare skulle bytte ut tall, så var programmet veldig langt, og da var det mye å sette seg inn i. I tillegg skulle du sette inn uttrykk for farten, høyden og tiden, og da ble det litt mye.»
(3) Elevenes motivasjon	Nyttig verktøy	Nyttig å kunne i fremtiden	«Programmering og teknologi gir jo flere muligheter. Så jo bedre du kan det, jo mindre problemer får du på høyskolen eller i fremtiden.»
(4) Elevenes tanker rundt kompetansemålet om programmering og numeriske metoder	Tidkrevende å oppfylle kompetansemålet	Trenger mer tid og opplæring i programmering	«Jeg synes ikke vi har hatt et så veldig stort fokus på det sånn egentlig. Og hvis det først skal være en del av fysikken så synes jeg vi burde brukt mer tid på det...»
(5) Elevenes forslag til forbedringer av undervisningsopplegget	Eierskap til programkoden	Ønsker å skrive hele programmet selv	«...jeg synes det hadde vært veldig lærerikt om vi hadde skrevet programmet selv, men det tar jo veldig mye tid. Det er jo veldig høy kompetanse, som jeg ikke tror alle får til.»

det ikke nødvendigvis slik at kodene som opptrer hyppigst i transkripsjonen gjenspeiler det informantene vektla som viktigst (Braun & Clarke, 2022, s. 102). Basert på dette er det begrenset hvilke valide slutninger som kan trekkes på bakgrunn av kodenes frekvens.

En forutsetning for at slutningene som trekkes på bakgrunn av det kodede datamaterialet skal være gyldige, er at kodingen er konsistent (se for eksempel Braun & Clarke, 2022; Robson & McCartan, 2016, s. 173). Det at transkripsjonene ble lest flere ganger, og at jeg underveis tilegnet meg mer kunnskap om tematisk koding, kan ha ført til en inkonsistent koding. Datamaterialet ble kodet to ganger, og gjennomgått i ulik rekkefølge (Braun & Clarke, 2022, s. 70). Andre gang jeg kodet datamaterialet oppdaget jeg momenter jeg hadde oversett den første gangen. Arbeidet i den andre koderunden innebar derfor å plassere de nyoppdagede momentene i passende, eksisterende koder, og opprette nye koder ved behov. Eksempelvis handlet et av momentene om hvordan det å få utdelt delvis ferdiglaget programkode gjorde det vanskeligere for elevene å drive feilsøk. I det ferdig kodede datamaterialet ble momentet tildelt koden «delvis ferdiglaget kode» og plassert under hovedkategori (2): Elevenes opplevde utfordringer tilknyttet opplegget. Gjennom slike oppdagelser ble det tydelig at det å kode kvalitativt datamateriale er en kompleks prosess (Braun & Clarke, 2022; Robson & McCartan, 2016, s. 471- 474). Selv om noen nye momenter fikk innpass i eksisterende koder og nye koder ble opprettet, ser det ut til at kodingen er konsistent.

Forskeren må tolke datamaterialet for å kunne kode det (Braun & Clarke, 2022). Dermed inneholder et kodet datamateriale alltid en viss grad av subjektivitet, og kan derfor ikke ansees som objektiv (se for eksempel Braun & Clark, 2022, s. 214; Tjora, 2018, s. 83; Postholm, 2010, s. 26). Kvale og Brinkmann (2015, s. 272-273) mener likevel at «objektivitet» er relevant i kvalitativ forskning, og kan innebære å la informanten snakke ut og forholde seg adekvat til fenomenet som undersøkes. «Objektivitet» kan også sees på som refleksivt, ved at forskeren reflekterer over hva en selv bidrar med i produksjonen av kunnskap. Forskerens subjektivitet kan dermed være uproblematisk, og heller ansees som en styrke dersom det kombineres med refleksiv tenkning (Braun & Clarke, 2022. s 12-13). For å ivareta størst mulig grad av objektivitet, har jeg transkribert intervjuene tilnærmet ordrett, og på den måten vært «lojal» mot informantene (Postholm, 2010, s. 127; Kvale & Brinkmann, 2015, s. 79).

Som forsker har jeg kunnet bestemme hvor dypt og kritisk informantenes uttalelser skulle tolkes (Kvale & Brinkmann, 2015, s. 52 og s. 79). For å redusere risikoen for å overanalysere

informantenes uttalelser, og tildele koder som ikke gjenspeiler det som ble formidlet i intervjuet, har jeg kodet delene av transkripsjonen som stakk seg ut, fremfor å gruble over hver setning og ulike måter den kan tolkes på (Braun & Clarke, 2022, s. 53, s. 61, s. 88-90; Postholm, 2010, s. 473).

6 RESULTATER

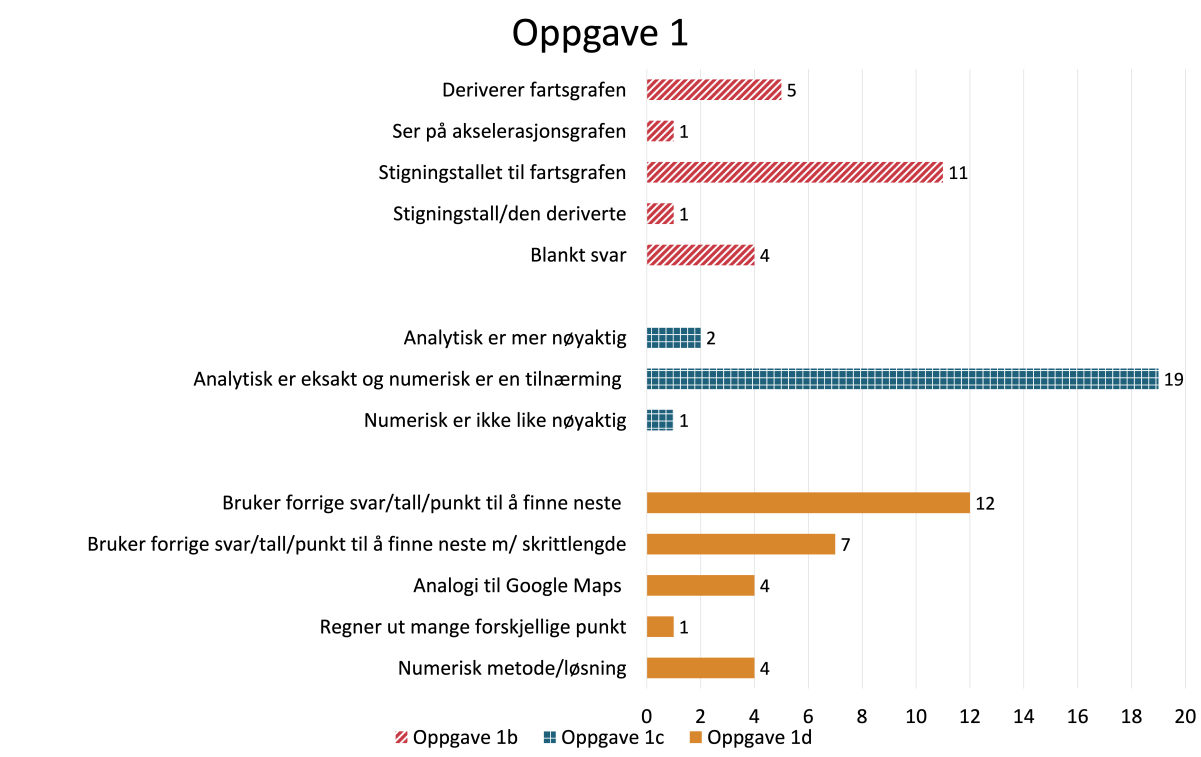
Kapittelet presenterer funnene fra analysearbeidet. I kapittel 6.1 presenteres resultatene fra de innleverte elevsvarene på oppgaveark 1 «Bil med konstant akselerasjon» (se Vedlegg E). På tilsvarende måte presenteres elevbesvarelsene på oppgaveark 2 «Vertikalt kast med luftmotstand» (se Vedlegg F) i kapittel 6.2. Resultatene på alle deloppgavene på oppgaveark 1 og 2, med unntak av programmeringsoppgavene 1a og 2d, fremvises grafisk og beskrives ytterligere i tekstformat. Kapittel 6.3 viser resultatene fra spørreundersøkelsen på både figur- og tekstformat, før fokuset rettes mot funnene fra elevsamtalene fra timen i kapittel 6.4 og gruppeintervjuene i kapittel 6.5. Jeg har valgt å inkludere relativt store mengder kvalitative data for å styrke studienes troverdighet, noe som også gjør oppgaven voluminøs. Undersøkelsens resultater vil underveis kommenteres kort, mens en mer fullstendig diskusjon kommer i kapittel 7.

6.1 Oppgaveark 1: «Bil med konstant akselerasjon»

I oppgave 1a skulle elevene utvide delvis ferdig programkode slik at de fikk frem bilens fart og akselerasjon som hver sin graf. Den ferdige koden skulle ikke leveres inn, men observasjoner fra timen tilsier at samtlige elever fikk til oppgaven, selv om de brukte varierende tid og fikk ulik grad av hjelp fra lærer.

Figur 6-1 (se neste side) viser resultatene fra den induktive kodingen av elevbesvarelsene på deloppgave 1b, 1c og 1d. Fra figuren ser vi at antall elevsvar er større i oppgave 1d enn 1b og 1c. Dette er fordi kodene her ikke er gjensidig utelukkende, noe som resulterte i at flere elevsvar passet inn i mer enn én kode. I oppgave 1b skulle elevene vise at bilens akselerasjon var 1,7 m/s^2 ved hjelp av fartsgrafen, og forklare hvordan de kom frem til svaret. Her ser vi at den mest brukte løsningsmetoden blant elevene var å se på fartsgrafen og regne ut stigningstallet. Den

nest mest brukt fremgangsmåten var å derivere fartsgrafen. Blant elevene som valgte denne fremgangsmåten, var det kun én elev som eksplisitt fant et matematisk uttrykk for farten, før hen deriverte det. Det er verdt å nevne at denne eleven kom frem til fartsuttrykket ved å putte inn den kjente akselerasjonen som stigningstall. De andre elevene som brukte samme fremgangsmåte oppga kun at akselerasjonen er den deriverte av farten, og har på den måten ikke kommet frem til en verdi for akselerasjonen, men heller kommet med en definisjon. Dette kan tyde på at elevene er godt kjent med relasjonen mellom farts- og akselerasjonsgraf, men ikke umiddelbart ser at akselerasjonen kan sees/hentes ut fra stigningstallet til fartsgrafen. En mer fullstendig relasjon mellom farts- og akselerasjonsgraf var det imidlertid én elev som formidlet. Vedkommende hadde en liknende forklaring som de 10 som svarte at de så på stigningstallet til fartsgraf, men la også til at dette ville vært det samme som å derivere fartsfunksjonen. Fra Figur 6-1 ser vi også at 1 elev benyttet akselerasjonsgrafen i sin løsning av oppgaven, og at 4 elever svarte blankt.



Figur 6-1: Oversikt over de 19 elevsvarene på deloppgave 1b, 1c og 1d på oppgavearket «Bil med konstant akselerasjon», og frekvensen de opptrer med.

I oppgave 1c skulle elevene forklare forskjellen mellom en analytisk og en numerisk løsning. Figur 6-1 viser at 19 elever beskrev forskjellen som løsningens grad av nøyaktighet, og presiserte at analytiske løsninger er eksakte mens numeriske løsninger er tilnærminger. De

resterende fem svarene omhandler det samme, men inneholder mindre presise formuleringer. Her er et eksempel på typiske elevsvar der forskjellen mellom analytiske og numeriske løsninger forklares, i tillegg til at det gis eksempler:

> **Analytisk:** Det eksakte svaret. 2/7
> **Numerisk:** En tilnærming til det eksakte svaret. 0,6667

Oppgave 1d omhandler Eulers metode, der elevene skulle forklare metoden med egne ord. Totalt 19 av elevene beskriver Eulers metode som at man bruker forrige punkt til å finne det neste, hvorav 7 av disse poengterer at det er en viss skrittlengde mellom det forrige og det nye punktet (se Figur 6-1). En annen elev beskriver Eulers metode som at man finner mange forskjellige punkter, men sier ingenting om hvordan man finner disse punktene. Videre oppgir fire elever at Eulers metode er en numerisk metode eller løsning, uten å spesifisere ytterligere. Et typisk elevsvar av denne typen er:

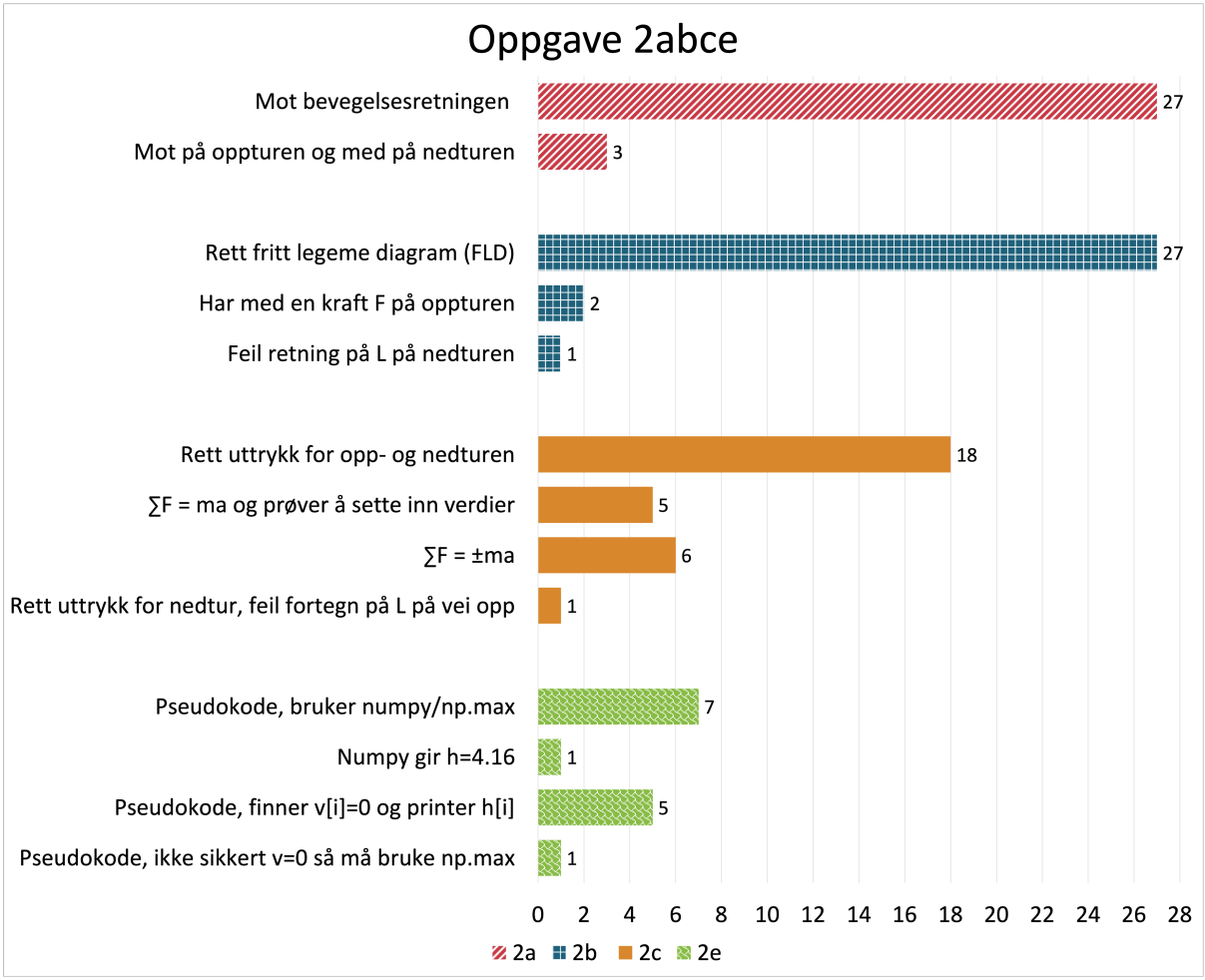
> Punktet du er i bestemmer neste punkt. Flyttes med et lite intervall fra
> det gamle punktet til det nye.

Vi ser også fra Figur 6-1 at fire av elevsvarene henviser til Google Maps som en analogi til Eulers metode.

6.2 Oppgaveark 2: «Vertikalt kast med luftmotstand»

På oppgave 2a svarte 27 av elevene rett, nemlig at luftmotstanden alltid virker *mot* ballens bevegelsesretning. Fra Figur 6-2 (se neste side) ser vi at 3 elever svarer at luftmotstanden virker *mot* ballens bevegelsesretning på oppturen og *med* ballens bevegelsesretning på nedturen. Dette kan tyde på at de forveksler luftmotstanden med tyngde. I oppgave 2b ble elevene bedt om å tegne et *fritt-legeme-diagram* (FLD) av ballen når den er på vei oppover og nedover. Resultatene viser at 27 av elevene tegnet korrekt FLD, avhengig av hva de definerte som positiv retning (se Figur 6-2). Et eksempel på en typisk elevbesvarelse er vist i Figur 6-3 (se side 75).

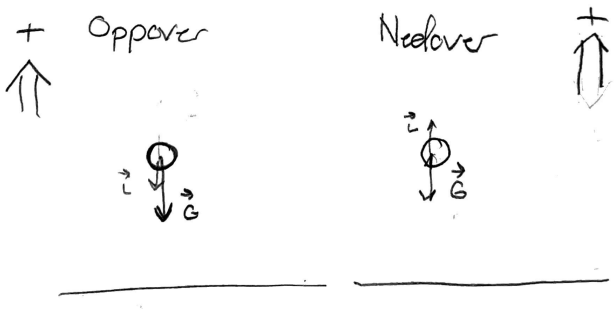
Videre ser vi fra Figur 6-2 at 2 av elevene skisserte nesten korrekt FLD, men at de tegnet inn en ekstra kraft F som virker oppover når ballen beveger seg oppover. En vanlig misoppfatning innen mekanikk er at gjenstander har en iboende kraft, som eksempelvis får ballen til å bevege seg oppover etter at den har blitt kastet (se for eksempel Angell et al., 2019, s. 136). Det er dermed forståelig at de to elevene ikke forstår at ballen kun har en oppoverrettet kraft F



Figur 6-2: Oversikt over de 30 elevsvarene på deloppgave 2a, 2b, 2c og 2e og frekvensen de opptrer med. Elevsvarene på de ulike oppgavene er fremvist med søyler med ulik farge og mønster.

idet den kastes oppover. Et annet elevsvar inkluderer også et nesten riktig FLD, med unntak av at luftmotstanden er tegnet inn med feil retning når ballen beveger seg nedover. Basert på resultatene i oppgave 2a, ville det vært rimelig å forvente at de 3 elevene som svarte at luftmotstanden virker *med* ballens bevegelsesretning på vei ned, ville tegnet inn luftmotstanden med feil retning når ballen beveger seg nedover. Dette er imidlertid ikke tilfelle, noe som er interessant.

I oppgave 2c skulle elevene skrive opp Newtons 2. lov for ballen når den er på vei oppover og nedover. Figur 6-2 viser at det totalt var det 18 elever som fikk til dette og svarte at $-L - G = ma$ når ballen er på vei oppover og at $L - G = ma$ når ballen er på vei nedover. Resultatene viser også at 1 elev hadde riktig uttrykk for når ballen beveget seg oppover, men hadde feil fortegn på luftmotstanden i uttrykket for når ballen var på vei ned. Dermed fikk hen at $-L - G = ma$ istedenfor $L - G = ma$. Videre oppga 11 elever svaret: $\Sigma F = ma$, hvorav 5



Figur 6-3: Eksempel på et typisk elevsvar på oppgave 2b fra oppgavearket «Vertikalt kast med luftmotstand», der elevene skulle tegne et *fritt-legeme-diagram* (FLD) for ballen på vei oppover og nedover.

elever forsøkte å sette inn verdier for m og a uten å lykkes.

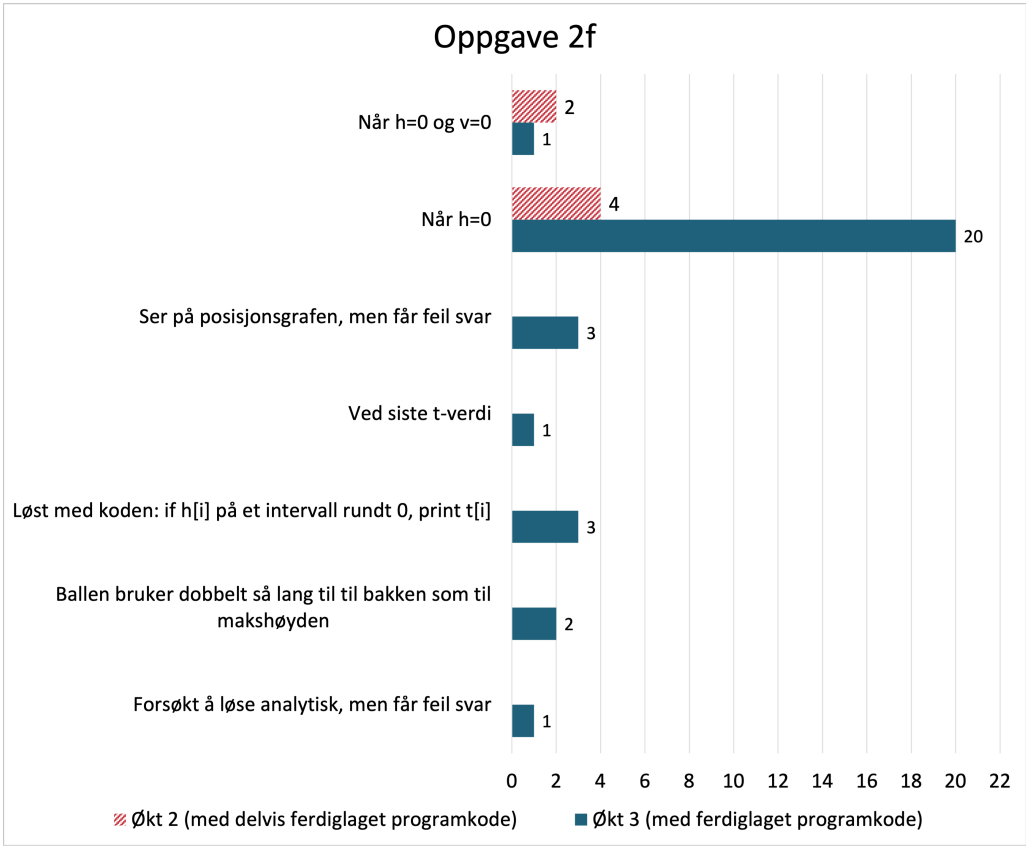
Oppgave 2d gikk ut på at elevene skulle utvide en delvis ferdiglaget kode ved å skrive noen kodelinjer slik at ballens posisjon, fart og akselerasjon ble fremstilt i hvert sin graf. For å tydeliggjøre overfor elevene hvor de skulle skrive kode, var kommentarer skrevet med store bokstaver i programkoden der de skulle utvide programmet. Denne oppgaven viste seg å være i vanskeligste laget for mange av elevene. Siden programmet skulle brukes i de neste deloppgavene, ble ikke elevene bedt om å levere inn programkoden, og det er derfor vanskelig å fastslå akkurat hvor mange elever som klarte å løse oppgaven. Likevel tyder observasjoner fra timen på at det var et fåtall elever som klarte å utvide programkoden, og dermed fikk plottet rett streknings-, farts- og akselerasjonsgraf. Av den grunn var det i hovedsak disse elevene som gikk videre til de neste deloppgavene i økt 2.

I oppgave 2e skulle elevene finne ballens maksimale høyde over bakken ved å skrive programkode. Figur 6-2 viser at flesteparten av elevsvarene inkluderer en form for pseudokode. 7 elever svarte at de vil bruke biblioteket «NumPy» og kommandoen «np.max», som finner elementet med størst verdi i en array. Videre ser vi at 5 andre elever ønsket å finne tidspunktet der $v[i] = 0$ og printe høyden $h[i]$ ved samme tidspunkt. En annen elev påpekte at siden vi regner numerisk, er det ikke sikkert at v er akkurat lik 0, og hen ønsket derfor å bruke «np.max» til å finne ballens makshøyde. Figuren viser også at 1 elev løste oppgaven ved å implementere NumPy-kommandoen «np.max» i programkoden sin, og har dermed et konkret tallsvar på makshøyden, nemlig $h = 4,16$ meter, noe som er rett svar.

I økt 2, der elevene arbeidet ut fra delvis ferdiglaget programkode, fikk kun et fåtall av elevene svart på de resterende deloppgavene 2f, 2g, 2h, 2i og 2j. Observasjoner fra timen tyder på at

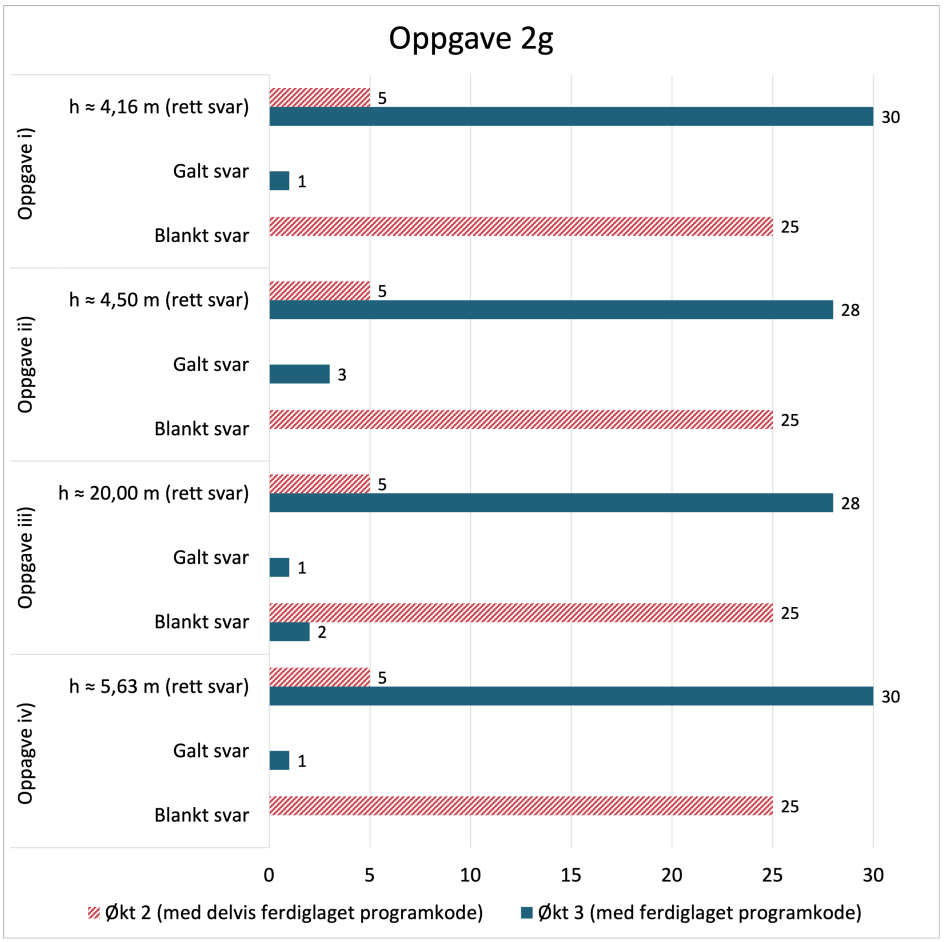
årsaken til dette skyldes at den foregående deloppgaven 2d var i vanskeligste laget, i tillegg til at tidsrammen var for liten. I økt 3 fikk elevene utdelt ferdiglaget programkode, som de skulle bruke til å besvare de samme deloppgavene. Ressursene elevene fikk utdelt for å besvare oppgave 2f-2j i de to øktene var altså ulike, noe som medfører at elevsvarene ikke er sammenlignbare. Leseren presenteres likevel for elevsvarene fra begge øktene, for å kunne danne seg et mer helhetlig bilde av undersøkelsen. For å lette lesingen, presenteres elevsvarene på de ulike deloppgavene i egne diagrammer, der hvert diagram inkluderer elevsvar fra økt 2 og 3, og er fargekodet etter hvilken arbeidsøkt svarene er hentet fra. Her viser de rødstripete søylene til elevsvarene fra økt 2 og de blå søylene til elevsvarene fra økt 3.

I oppgave 2f skulle elevene bestemme når ballen traff bakken, og beskrive hvordan de kom frem til svaret. De ulike elevsvarene er presentert i Figur 6-4 (se neste side). Med delvis ferdiglaget programkode (se de rødstripete søylene) svarte 4 av 6 elever at ballen traff bakken når høyden $h = 0$. De resterende 2 elevene svarte også at ballen treffer bakken når høyden $h = 0$, men la til at farten v også måtte være null. Med ferdiglaget programkode (se de blå søylene), svarte 20 elever at ballen traff bakken ved h=0, mens 1 elev svarte at ballen traff bakken når både høyden og farten er lik null. Sistnevnte svar er feil, men elevens tankegang er forståelig, da ballen har fart lik 0 rett etter at den treffer bakken, dersom ballen antas å ha null sprett. Videre ser vi at 3 elever svarte at de løste oppgaven ved å se på posisjonsgrafen, men kom frem til feil svar. Vi ser også at 1 elev mener at ballen treffer bakken ved siste t-verdi, noe som ikke er tilfellet da posisjonsgrafen avslører at ballens posisjon ved denne t–verdien er mindre enn 0, og dermed befinner seg under bakkenivå. Fra Figur 6-4 fremkommer det at 3 elever løste oppgaven ved å skrive programkode som tok hensyn til betingelsen om at ballen treffer bakken når høyden er innenfor et lite intervall rundt 0. Vi ser også at 2 av elevene glemmer å ta luftmotstanden i betraktning, og resonnerer seg frem til at det tar dobbelt så lang tid før ballen treffer bakken, som tiden det tar før ballen oppnår maksimal høyde. Det fremkommer også at 1 elev forsøker å løse problemet analytisk, og får feil svar.



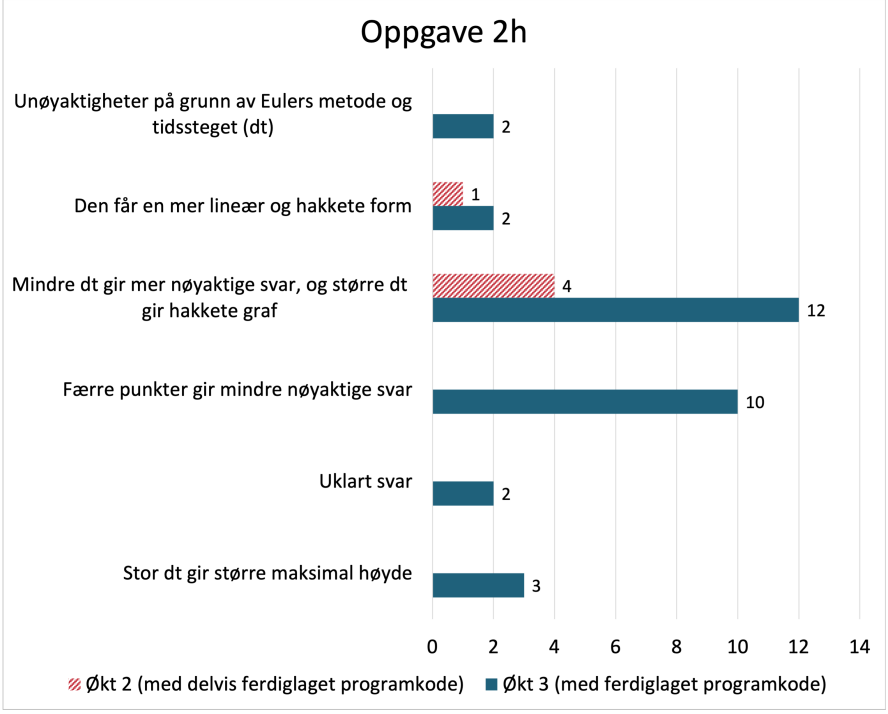
Figur 6-4: Oversikt over elevsvarene på deloppgave 2f, og frekvensen de opptrer med. De rødstripete søylene viser elevsvarene fra økt 2, mens de blå søylene viser elevsvarene fra økt 3. Her arbeidet elevene henholdsvis ut fra delvis ferdiglaget og helt ferdiglaget programkode.

I oppgave 2g fikk elevene i oppgave å endre lengden på tidssteget dt til bestemte verdier, og notere seg ballens maksimale høyde for de ulike verdiene av dt. Fra økt 2, der elevene arbeidet ut fra delvis ferdiglaget programkode, var det totalt 5 elever som besvarte alle deloppgavene. De rødstripete søylene i Figur 6-5 (se neste side) viser at alle disse 5 elevene fikk rett på samtlige deloppgaver, mens de resterende 25 elevene ga blanke svar. Svarene fra økt 3 (de blå søylene) fordeler seg annerledes enn svarene fra økt 2. I økt 3 oppga 30 av 31 elever rett svar på deloppgave i), mens sistemann oppga galt svar. På deloppgave ii) ser vi at det var 28 elever som svarte rett og 3 som svarte galt. På deloppgave iii) var det 28 elever som svarte rett, 1 elev som svarte feil og 2 elever som svarte blankt. Videre ser vi at 30 elever fikk rett på deloppgave iv), mens den siste eleven svarte feil.



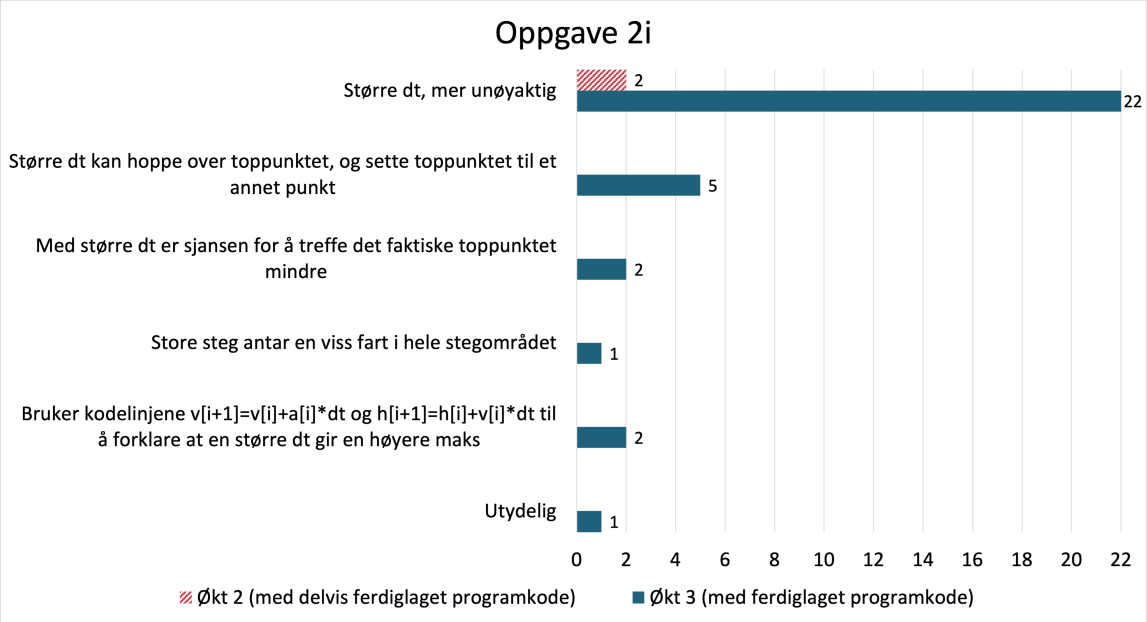
Figur 6-5: Oversikt over elevsvarene på deloppgave 2g, og frekvensen de opptrer med. De rødstripete søylene viser elevsvarene fra økt 2, mens de blå søylene viser elevsvarene fra økt 3.

Oppgave 2h gikk ut på at elevene skulle beskrive hva som skjer med posisjonsgrafen når lengden på tidssteget endres. Med delvis ferdiglaget programkode, svarte 1 elev at grafen fikk en mer lineær og hakkete form (se de rødstripete søylene i Figur 6-6). De 4 andre elevene svarte også at grafen ble hakkete, men la i tillegg til at dette skyldes en større dt, og beskriver videre at jo mindre dt er, desto mer nøyaktige svar får vi. Med ferdiglaget programkode (se de blå søylene i Figur 6-6), ser vi at henholdsvis 2 og 12 elever oppga at grafen blir mer hakkete, og at mindre dt gir mer nøyaktig svar og at større dt gir hakkete graf. Med andre ord virker det som om elevene, gjennom visuell feedback, ser hvordan presisjonen i de numeriske beregningene påvirker resultatet. I besvarelsene der elevene brukte ferdiglaget programkode, fremkommer det også at 2 elever svarer at endringen i posisjonsgrafen skyldes unøyaktigheter i Eulers metoder, som følge av lengden på dt. Videre viser Figur 6-6 at totalt 10 elever er inne på at lengden av dt påvirker hvor mange punkter som regnes ut. Vi ser også at 3 elever oppgir at ballens maksimale høyde blir større når dt øker. 2 elever svarte blankt.



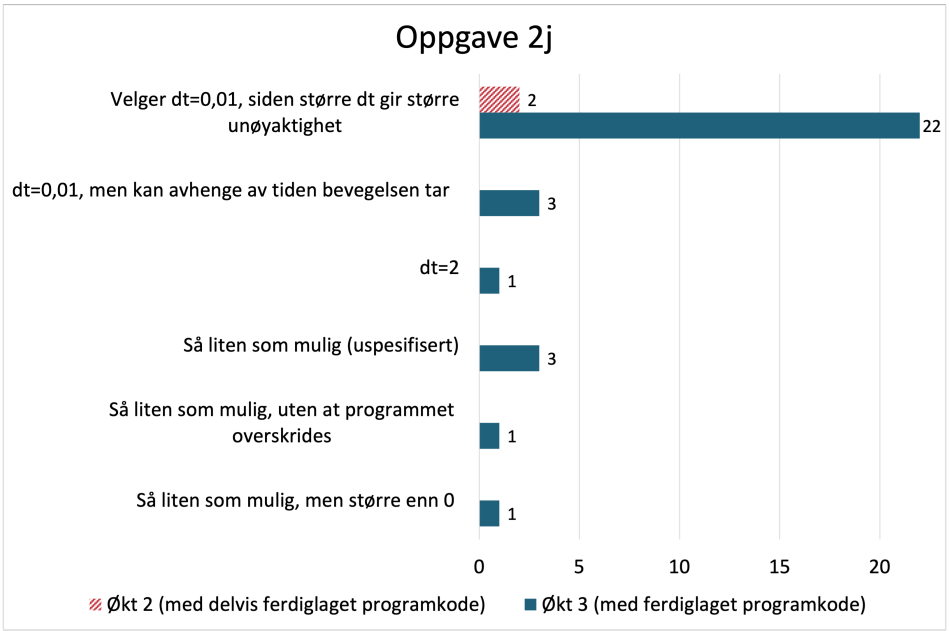
Figur 6-6: Oversikt over elevsvarene på deloppgave 2h, og frekvensen de opptrer med. De rødstripete søylene viser elevsvarene fra økt 2, mens de blå søylene viser elevsvarene fra økt 3.

Oppgave 2i omhandler mye av det samme som oppgave 2h. Her fikk elevene i oppgave å forklare hvorfor det å endre verdien til tidssteget dt endrer ballens beregnede maksimale høyde. Elevenes svar, og hvordan de fordeler seg, er fremstilt i Figur 6-7. Fra de rødstripete søylene i figuren ser vi at de 2 elevene som arbeidet utfra delvis ferdiglaget programkode, svarte at en større dt gir mindre nøyaktige svar. Med ferdiglaget programkode, ser vi at 22 elever oppga lignende svar (se de blå søylene). Vi ser også at 7 elever var inne på at større dt kan føre til at viktige punkter utelates. 5 av disse elevene svarer at vi risikerer å hoppe over det «faktiske» toppunktet, og at maksverdien dermed settes til det høyeste beregnede punktet, noe som ikke nødvendigvis er det «faktiske toppunktet». De 2 andre elevene svarer at sjansen for at datamaskinen beregner det faktiske toppunktet blir mindre, desto større dt blir. Videre beskriver én elev hvordan Eulers metode antar en viss, konstant fart i hele stegområdet, og at dette påvirker den beregnede maksimale høyden. Fra figur 6-7 fremkommer det også at 2 elever svarer at større dt gir en større maksimal høyde, og begrunner svaret med kodelinjen $v[i+1] = v[i] + a \cdot dt$ og $h[i+1] = h[i] + v[i] \cdot dt$. Fra figuren ser vi at antall elevsvar på oppgave 2i er høyere enn på de andre deloppgavene, som har 31 elevsvar. Årsaken til dette er at kodene som er brukt til å analysere svarene på oppgave 2i, ikke er gjensidig utelukkende. Elevsvar som passet inn i to ulike koder, ble dermed kodet to ganger, noe som har medført at antall svar på oppgave 2i fra økt 3 ikke gjenspeiler antall elever som besvarte oppgaven.



Figur 6-7: Oversikt over elevsvarene på deloppgave 2i, og frekvensen de opptrer med. De rødstripete søylene viser elevsvarene fra økt 2, mens de blå søylene viser elevsvarene fra økt 3.

I den siste deloppgaven 2j, skulle elevene avgjøre og begrunne hvilken verdi for dt de ønsker å bruke for å beregne ballens posisjon. Med delvis ferdiglaget programkode, svarer 2 elever at de ønsker å bruke $dt = 0,01$, siden en mindre dt gir en større nøyaktighet (se de rødstripete søylene i Figur 6-8). Fra elevsvarene fra økt 3 (se de blå søylene i Figur 6-8), der de fikk utdelt ferdiglaget programkode, ser vi at 22 elever svarer det samme. Her svarer også 3 elever at de ønsker å bruke $dt = 0,01$, men påpeker at lengden på tidssteget dt må stå i forhold til hvor lang tid bevegelsen vi studerer tar. Elevene begrunner dette med at desto lengre tid bevegelsen tar, desto større steglengde kan benyttes uten at de beregnede posisjonsdataene avviker nevneverdig fra de faktiske posisjonsdataene. Fra resultatene fremkommer det at 3 andre elever svarer at de vil bruke så liten dt som mulig, uten å spesifisere ytterligere. Videre skriver 1 elev at hen ønsker å bruke så liten dt som mulig, uten at programmet blir for tung og tidkrevende og krasjer. En annen elev svarer også at hen vil bruke en så liten dt som mulig, men at dt også må oppfylle kravet om å være større enn null. Figur 6-8 viser også at 1 elev svarer at hen ønsker å bruke $dt = 2$, uten å begrunne svaret noe videre.

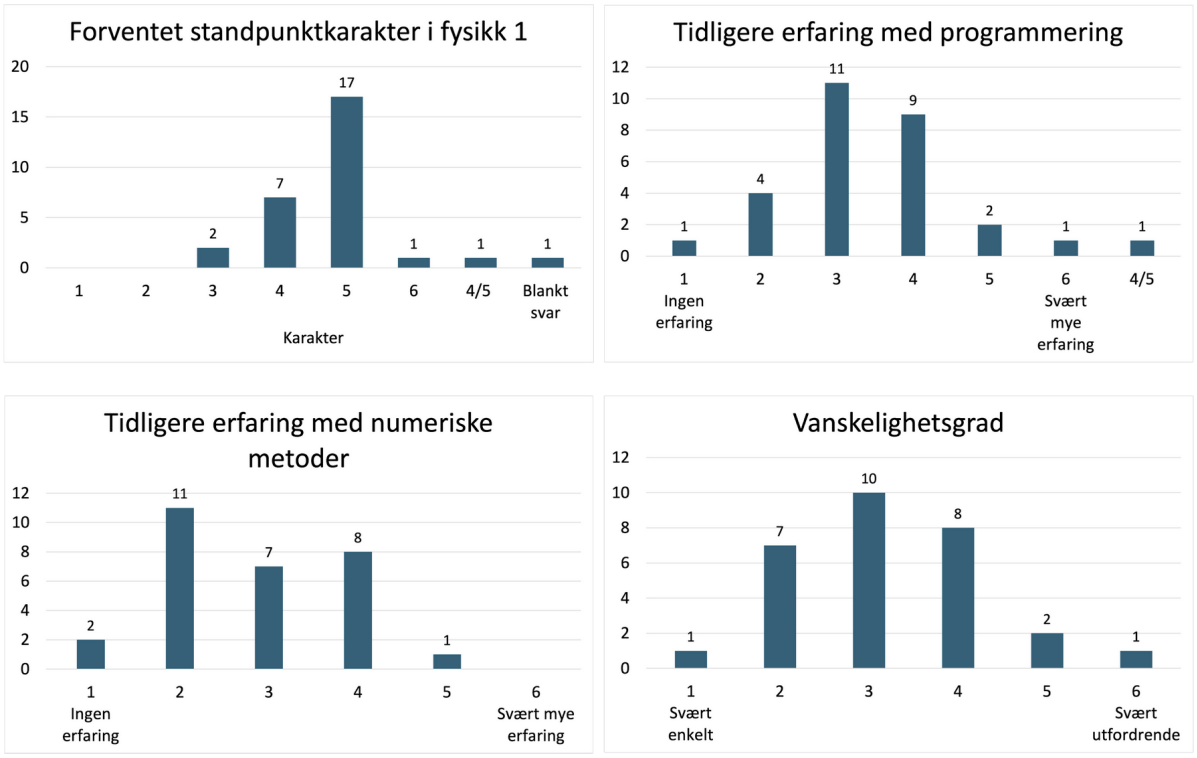


Figur 6-8: Oversikt over elevsvarene på deloppgave 2j, og frekvensen de opptrer med. De rødstripete søylene viser elevsvarene fra økt 2, mens de blå søylene viser elevsvarene fra økt 3.

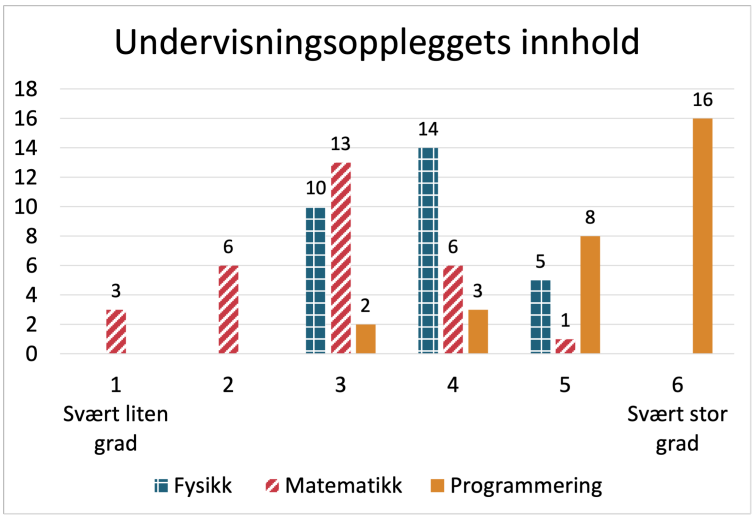
6.3 Resultater fra spørreskjema

De fire første spørsmålene på spørreskjemaet (se Vedlegg B) omhandler henholdsvis standpunktkarakteren elevene forventer å få i fysikk 1, hvor mye erfaring de har med programmering og numeriske metoder, og hvor utfordrende de opplevde undervisningsopplegget. Elevsvarene, og hvordan de fordeler seg, er vist i Figur 6-9 (se neste side). Her fremkommer det at klassen som helhet forventer høye standpunktkarakterer. Noen elever oppga at de hadde erfaring med både numeriske metoder og programmering i forkant av undervisningsopplegget. Typisk kommenterte elevene at de hadde erfaring med programmering både fra fysikken, hvor de brukte ferdiglagde programmer, og fra matematikken, der de har løst mattestykker i Python. Likevel ser vi at elevene oppgir at de hadde mer erfaring med programmering enn med numeriske metoder. Videre ser vi at flertallet av elevene opplevde undervisningsopplegget som middels vanskelig. Likevel er hele spekteret representert, hvorav 1 elev syntes opplegget var svært enkelt, mens 1 syntes det var svært utfordrende.

Videre ble elevene bedt om å angi i hvilken grad de opplevde at undervisningsopplegget omhandlet fysikk, matematikk og programmering. Elevene krysset av på en skala fra 1 til 6 på hvor mye de opplevde at opplegget omhandlet de ulike fagområdene. Dersom en elev krysset av ved tallet 1 i forbindelse med temaet fysikk, indikerer det at vedkommende opplevde at opplegget omhandlet fysikk i svært liten grad. På samme måte vil et kryss ved tallet 6 for tema fysikk, indikere at vedkommende opplevde at undervisningsopplegget i svært stor grad omhandlet fysikk. Elevenes svar, og hvordan svarene fordeler seg, er vist i Figur 6-10 (se neste side). Her ser vi at elevene generelt krysser av ved høyere tall for temaet programmering, enn for temaene matematikk og fysikk. Resultatene viser altså at klassen som helhet i hovedsak opplevde at undervisningsopplegget omhandlet programmering, men at timene også inneholdt mye fysikk og matematikk.

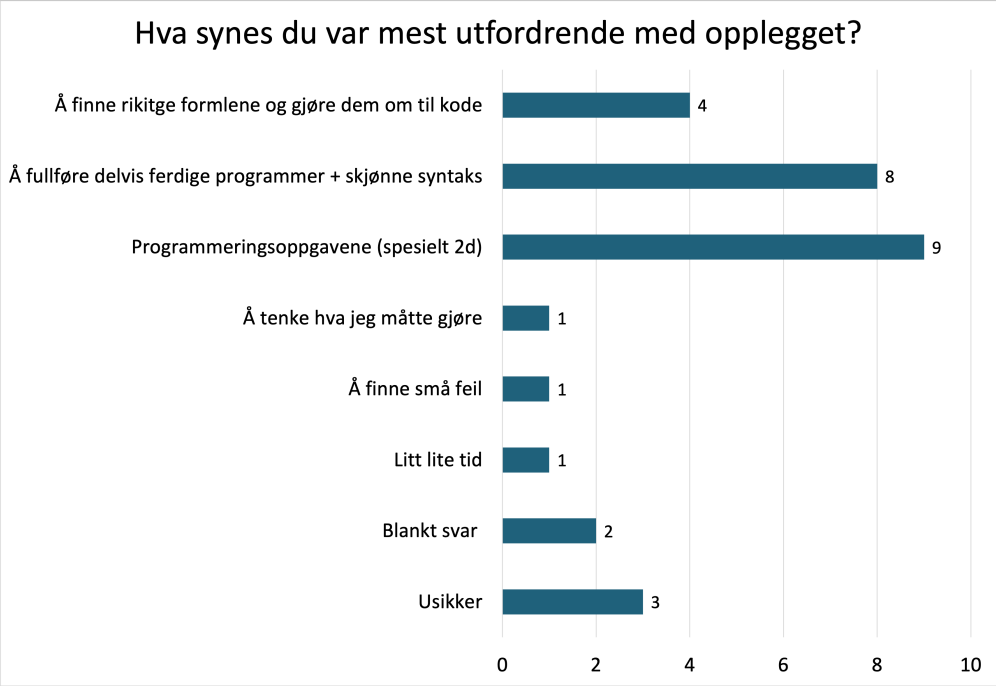


Figur 6-9: Resultatene fra 29 besvarte spørreskjemaer etter gjennomføring av økt 2. Her fremkommer det at elevene forventer høye standpunktkarakterer og at de generelt har mer erfaring med programmering enn med numeriske metoder. Flertallet av elevene oppgir også at de opplevde undervisningsopplegget som middels vanskelig, selv om hele spekteret er representert.



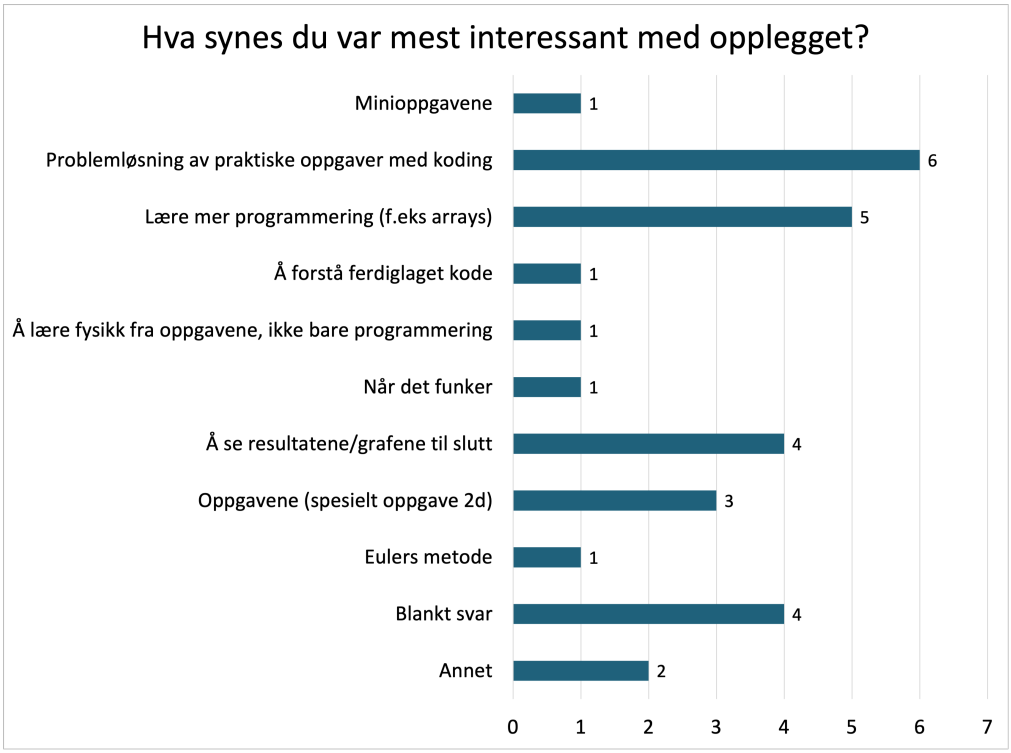
Figur 6-10: Resultatene fra 29 besvarte spørreskjemaer i etterkant av økt 2. Som helhet opplevde klassen at opplegget handlet mest om programmering, men også inneholdt mye fysikk og matematikk.

Spørreskjemaet inkluderte et åpent spørsmål om hva elevene opplevde som mest utfordrende med undervisningsopplegget. Svarene ble kodet induktivt, og plassert i ulike kategorier basert på innholdet i svarene deres. Figur 6-11 viser at 9 elever avga tekstsvar som indikerte at programmeringsoppgavene, og spesielt oppgave 2d, var noe av det mest utfordrende. Vi ser også at 14 av elevene opplevde det som vanskelig å finne de riktige formlene som skulle brukes og «gjøre» dem om til kode, og at det var vanskelig å finne feil og forstå syntaks, eller at det var vanskelig «å tenke hva de måtte gjøre». Én elev svarte at det mest utfordrende var at hen hadde for lite tid til å løse oppgavene. Videre ser vi at 3 elever svarer at de er usikre på hva som var mest utfordrende, mens 2 andre elever svarte blankt.



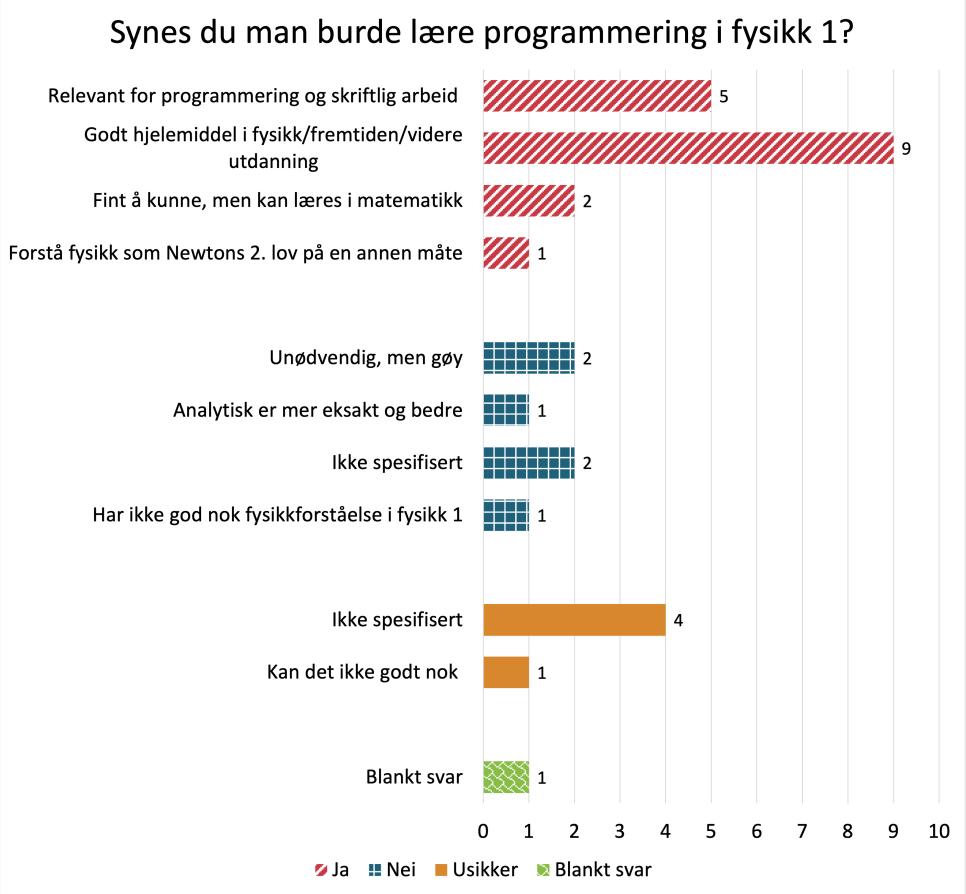
Figur 6-11: Oversikt over de 29 elevsvarene på spørsmålet: «Hva synes du var mest utfordrende med undervisningsopplegget?» og hvordan svarene fordeler seg.

Spørreskjemaet inkluderer også et åpent spørsmål om hva elevene syntes var det mest interessante med undervisningsopplegget. Figur 6-12 viser en kategorisering av svarene, hvor 12 elevesvar er kodet til at de syntes programmeringsdelen var mest interessant. Disse elevsvarene er å finne under «minioppgavene», «problemløsning av praktiske oppgaver med koding» og «lære mer programmering (f. eks. arrays)» i figuren. 6 elever vektla programforståelse, det å få programmer til å kjøre, og å kunne framstille resultater grafisk, som interessant. Videre ser vi at 3 elever syntes det å jobbe med «oppgavene, spesielt oppgave 2d», som i hovedsak var en programmeringsoppgave, var mest interessant. En annen elev oppga at hen syntes det var spennende å «lære fysikk fra oppgavene, ikke bare programmering». En tredje elev svarte at det mest interessante med opplegget var å lære Eulers metode. Figur 6-12 viser også at 4 svarer blankt, og at 2 av elevsvarene er kodet som «annet». Sistnevnte elevsvar inkluderer mindre seriøse svar, som eksempelvis at det mest interessante var å «høre hva jentene bak dem gjorde i helgen».



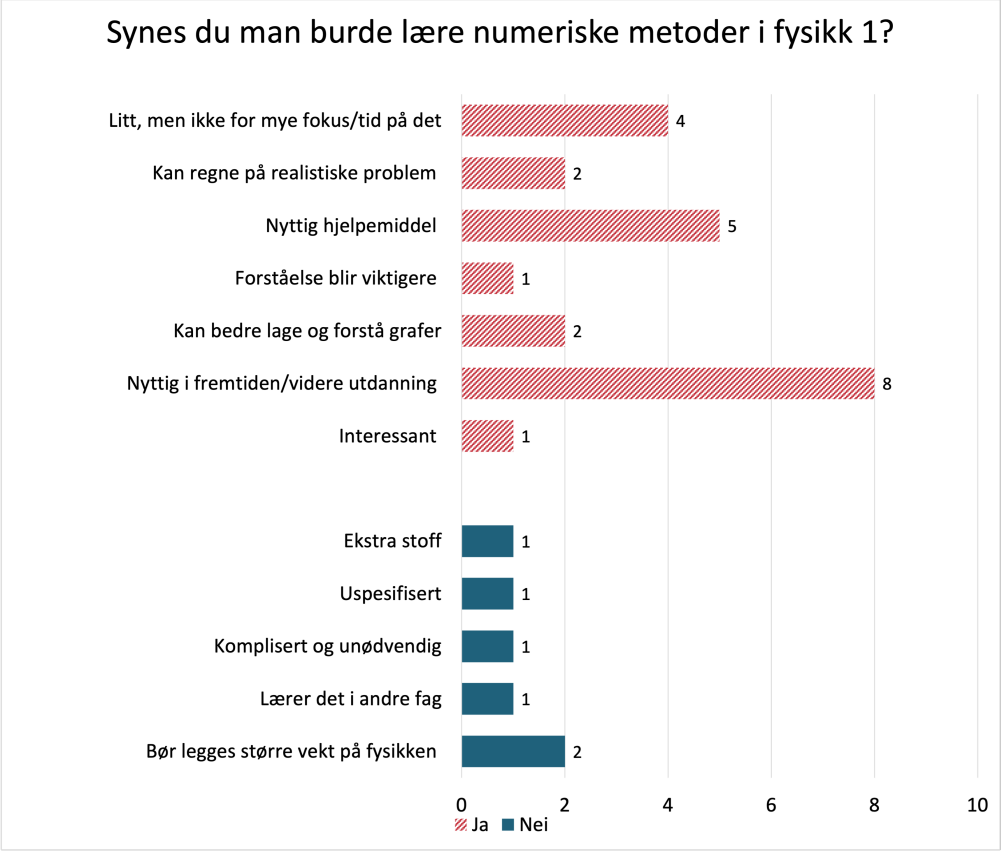
Figur 6-12: Oversikt over de 29 elevsvarene på spørsmålet: «Hva synes du var mest interessant med undervisningsopplegget?» og hvordan svarene fordeler seg.

Elevene fikk også muligheten til å uttale seg om de synes man burde lære programmering i fysikk 1. Figur 6-13 viser at 17 elever mente at programmering burde være en del av fysikkfaget. Svarene som inngår her beskriver programmering som et nyttig verktøy man får bruk for i fysikkfaget, i videre utdanning og/eller i fremtiden. På den annen side synes 6 elever at programmering ikke burde være en del av fysikkfaget. Elevene begrunnet standpunktet sitt ulikt. 2 elever svarte at det er unødvendig å lære programmering i fysikken, men at de likevel synes programmering er gøy. En annen elev svarte at siden analytiske løsninger er mer eksakte enn numeriske, synes hen at vi burde holde oss til å løse problemer analytisk, uten datamaskin. Videre ser vi at 2 av elevene ikke begrunner hvorfor de synes programmering ikke burde være en del av fysikkfaget. En annen elev forklarer standpunktet sitt med at hen ikke har god nok fysikkforståelse til at programmeringen er til hjelp. Fra Figur 6-13 ser vi også at 5 elever svarer at de er usikre på hva de mener, hvorav 1 elev oppga at hen ikke kan nok om programmering til å ta stilling til spørsmålet. Det er også 1 elev som svarte blankt på dette spørsmålet.



Figur 6-13: Oversikt over de 29 elevsvarene på spørsmålet: «Synes du man burde lære programmering i fysikk 1?», og hvordan svarene fordeler seg. Søylene med ulik farge og mønster indikerer om eleven synes man burde lære programmering i fysikk 1 eller ikke, samt begrunnelsen for standpunktet.

Gjennom et åpent spørsmål fikk jeg innsikt i elevenes refleksjoner rundt det å skulle lære numeriske metoder i fysikk 1. Fra Figur 6-14 fremkommer det at flertallet, 23 av 29 elever, synes man burde lære numeriske metoder i fysikkfaget. 15 av elevene underbygget svaret sitt med at numeriske metoder er et nyttig hjelpemiddel man får bruk for senere, hvorav 2 elever dro frem at det kan brukes til å regne på mer realistiske problemer. 2 andre elever pekte på at numeriske metoder gjør at de bedre kan lage og forstå grafer. Vi ser også at én elev svarte at man bør lære numeriske metoder i fysikken, siden hen synes det er interessant. Videre svarte en annen elev at fysikkforståelse blir viktigere når man skal lære numeriske metoder, noe hen fremhever som positivt. 6 elever var ikke like positive til at man burde lære numeriske metoder i fysikken. Disse elevsvarene dreier seg om at numeriske metoder blir «ekstra stoff» de må lære seg, at det er «komplisert og unødvendig», i tillegg til at det kan «læres i andre fag» og at det «bør legges større vekt på fysikken». Vi ser også at 1 elev ikke begrunnet standpunktet sitt. Det er imidlertid viktig å påpeke at elevene i hovedsak er positive til numeriske metoder som del av fysikkundervisningen, og ser verdien av dette.



Figur 6-14: Oversikt over de 29 elevsvarene på spørsmålet: «Synes du man burde lære numeriske metoder i fysikk 1?», og hvordan svarene fordeler seg. Søylene med ulik farge og mønster indikerer om elevene synes man burde lære numeriske metoder i fysikk 1 eller ikke, samt begrunnelsen for standpunktet.

6.4 Resultater fra elevsamtalene fra timen

Kodingen av elevsamtalene fra timen viser at hver samtale kunne tildeles en av de fire predefinerte kodene. Tabell 6-1 viser en oversikt over hvordan de 24 elevsamtalene fordelte seg mellom de ulike kodene, og frekvensen de dukket opp med. Kodene er gjensidig utelukkende, slik at hver elevsamtale dermed kun har fått tildelt én kode.

Fra kodingen fremkommer det at elevsamtalene, med og uten hjelp fra lærer, utspilte seg ulikt. Samtalene, der elevene fikk hjelp fra lærer, har blitt tildelt koden «samtale med lærer», mens elevsamtalene uten hjelp fra lærer fått tildelt kodene «kumulativ», «eksplorerende», «frittstående påstander» og «bekreftende diskusjon». Lydopptakene tyder på at elevene i større grad utdyper egne utsagn, og anvender sin fysikkforståelse i samtale med lærer. Et eksempel på en slik diskusjon er presentert i Sitat 6-1 (se neste side). Her fremkommer det at elevene trenger hjelp fra lærer til å vurdere om grafen de plottet i Python er rett. I sitatet begynner læreren med å spørre elevene om hva de tenker om formen på grafen, før de får hjelp til å beskrive ballens bevegelse ved ulike tidspunkter. Videre ser vi at elevene bruker sin fysikkforståelse til å avgjøre om plottet deres er rett, mer eller mindre på egen hånd.

Tabell 6-1 En oversikt over hvilken kode elevsamtalene fra timen ble tildelt, og frekvensen de opptrer med i transkripsjonen. Kodene i tabellen overlapper ikke, og er dermed gjensidig utelukkende.

Kode	Frekvens
Kumulativ	4
Eksplorerende	4
Frittstående påstander	1
Bekreftende diskusjon	5
Samtale med lærer	10

Sitat 6-1

Elev 2: Vi har fått frem graf, men er ikke helt sikre på om den er riktig.
Lærer: Hvordan synes dere at grafen (akselerasjonsgrafen) ser ut da?
Elev 3: Litt...ehh...
Elev 4: Det er jo...blir det ikke liksom når du kaster ballen opp og så stopper den litt og så fortsetter den?
Elev 3: Ja, skjønner.
Lærer: Men husker dere verdien for lille g?
Elev 3: 9.81.
Lærer: Mmm. Så når vi kaster ballen opp, så er den jo mye mindre enn 9.81. Høres det riktig ut?
Elev 3: Mmm...ja, nei, litt. Blir sånn minus eller... det spørs vel på den da.
Elev 4: Den blir jo mindre enn 9.81, fordi vi har minus luftmotstanden.
Lærer: Mmm, stemmer. Og så ser vi, hva er det som skjer her, som du sa istad. Akkurat her. På den flata. Hva tror du skjer med ballen da?
Elev 3: Den snur retning.
Lærer: Ja. Og så...
Elev 3: Øker jo akselerasjonen enda mer. Ja. Da må jo grafen være riktig.

I samtalene uten deltakelse fra lærer, observerte jeg at elevene i mindre grad anvendte fysikkforståelse da de skulle avgjøre om akselerasjonsgrafen var riktig. Her ble avgjørelsen tatt på grunnlag av om elevene syntes grafen så «rar» ut. Elevene anså gjerne grafen som «rar», dersom de ikke hadde sett en lignende akselerasjonsgraf fra tidligere fysikkundervisning. Her er et eksempel på hvordan en slik samtale utspilte seg i timen:

Sitat 6-2

Elev 5: Grafen (akselerasjonsgrafen), er den riktig?
Elev 6: Den ser jo veldig rar ut. Den ligner ikke på sånn den pleier å se ut. Men programmet kjører, så den er sikkert riktig.

Fra sitatet ser vi at den visuelle feedbacken grafen gir resulterer i at elevenes umiddelbare tanke er at grafen må være feil, siden den ser «veldig rar ut», Til tross for dette lander elevene på riktig konklusjon, nemlig at akselerasjonsgrafen de plottet er rett. Elevene begrunner imidlertid konklusjonen med at grafen «sikkert er riktig» siden programmet kjørte, og at de ikke fikk feilmeldinger. Elevene tar altså ikke i betraktning at programkoden deres *kunne* inneholdt logiske feil, som rent fysisk kunne vært ukorrekte, og dermed gitt en akselerasjonsgraf som ikke stemmer med situasjonen beskrevet i oppgaveteksten.

Lydopptakene ga altså inntrykk av at elevene i større grad anvender egen fysikkforståelse til å evaluere resultatene sine i samtalene der læreren deltok aktivt, enn i elev-elev-samtaler. Det er likevel viktig å påpeke at det finnes eksempler på at dette ikke alltid var tilfellet. Eksempelvis

diskuterte et elevpar hva uttrykket for Newtons 2. lov var når en ball er på vei oppover i et tyngdefelt der luftmotstanden virker. Elevene var uenige i hvorvidt ballen hadde en iboende kraft som fikk den til å gå oppover. Underveis i den *eksplorerende* samtalen brukte den ene eleven fysikkforståelsen sin til å oppklare en vanlig misoppfatning innen mekanikk, nemlig at gjenstander ikke har en iboende kraft (impetus) som får dem til å bevege seg. Her ser vi at elevene diskuterer fysikkinnholdet i programmeringsoppgaven:

Sitat 6-3

Elev 7: Tyngdekraft...og så har du luftmotstand.
Elev 8: Så hvis du kaster noe, etter du slipper den (ballen), så har du ikke dyttekraft.
Elev 7: Den har jo en fart, den tar den jo med seg, farten fra kastet.
Elev 8: Jaja, men den har ikke en dyttekraft.
Elev 7: Men den har jo farten, og den er jo større enn luftmotstanden og gravitasjonskraften. Den må jo komme fra et sted.
Elev 8: Jaja, men den har ikke en dyttekraft.
Elev 7: Ah det er jo helt... For du kaster den jo rett opp.
Elev 8: Ja, men etter at du slipper den har den ikke dyttekraft, bare mens du holder den.
Elev 7: Åja, så den (dyttekraften) skal ikke være med etter den har forlatt hånden...

Generelt viser lydopptakene at elevene ønsket å løse oppgavene på egen hånd, men også at de ønsket veiledning fra læreren.

6.5 Resultater fra gruppeintervjuer

Gruppeintervjuene ble kodet i fem hovedtemaer: (1) elevenes opplevelse av å arbeide med undervisningsopplegget, (2) elevenes opplevde utfordringer tilknyttet opplegget, (3) elevenes motivasjon (4) elevens tanker rundt kompetansemålet om programmering og numeriske metoder, og (5) forslag til forbedringer av undervisningsopplegget. I det følgende presenteres resultatene, samlet etter hovedtema og ikke etter kodene som ble brukt som verktøy i analysearbeidet. Som nevnt i kapittel 5.2, hadde informantene fra det første intervjuet betraktelig mer på hjertet enn informantene fra det andre intervjuet. Dette delkapittelet inneholder dermed mer fra intervju 1, gjort med Felix og Hugo, enn intervju 2, gjort med Lisa og Ruby.

Elevenes opplevelse av å arbeide med undervisningsopplegget

Det første hovedtemaet handler om hva informantene syntes de fikk ut av undervisningsopplegget. Informantene var ivrige på å fortelle at de fikk til minioppgavene, der de fikk repetert grunnleggende programmering. I intervju 2, beskriver Lisa og Ruby det slik:

> **Sitat 6-4** **Lisa:** Når jeg hadde matte i fjor så var det sånn «ahh programmering igjen», men nå (med minioppgavene) følte jeg at jeg fikk det til... så nå turte jeg liksom å prøve meg litt med programmeringen i fysikken.
> **Ruby:** Ja, når man får det (minioppgavene) til kan man bli litt mer motivert. Da har man mer lyst til å prøve.

Fra sitatet ser vi at informantene følte at de mestret minioppgavene og opplevde motivasjon, noe som førte til at de ønsket og turte å prøve på andre programmeringsoppgaver som ble gitt senere i undervisningsopplegget. Informantene fra intervju 1 var innom mye av det samme, men la også til at de syntes minioppgavene var en fin og oversiktlig måte å repetere grunnleggende programmering på. De la spesielt vekt på at de likte å få programmere selv, og ikke bare se læreren programmere på storskjerm. Her forteller informantene hvordan de opplevde å repetere grunnleggende programmering gjennom minioppgavene:

> **Sitat 6-5** **Felix:** Jeg synes de var bra,...hvertfall for å komme igang litt, at de var separert litt i starten. Istedenfor at vi fikk alt sammen samtidig, så jeg synes det var veldig hjelpsomt... fordi det hjalp meg med å huske litt hvordan det var igjen. Så...jeg synes det hjalp veldig at vi hadde de oppgavene først da.
> **Hugo:** Ja, jeg synes det var bra at vi startet med grunnleggende, litt lette oppgaver, før det ble mer utfordrende.

Videre viser resultatene at elevene satte pris på at minioppgavene var interaktive, der de i stor grad skulle skrive egen programkode ved hjelp av hint lagt inn som kommentarer. Felix beskriver det slik:

> **Sitat 6-6** **Felix:** Vanligvis føler jeg at det blir litt mer at vi skriver av og pugger programmene lærerne har laget, mer enn å forstå det og kunne bruke det selv. Så jeg likte metodene dine bedre enn hennes. Har også opplevd at lærere blir sure når vi ikke forstår det, så vi tør nesten ikke å spørre om hjelp med programmering. Du har vært mer forståelsesfull enn dem.

Informantene formidlet også at de likte at de innledende oppgavene på oppgaveark 2 (se Vedlegg F) repeterte kjent fysikk. Mer konkret beskriver Lisa at det var bra med

repetisjonsoppgavene, siden det ga dem tid til «å tenke over fysikken og hva de ulike tingene var». Felix er inne på noe av det samme:

Sitat 6-7 **Felix:** Jeg fikk jo repetert litt da, men jeg kunne det jo fra før. Men… det er ikke sikkert at jeg hadde husket det dersom vi ikke hadde fått en oppgave om det da.

I sitatet ovenfor fremkommer det at Felix har et litt annet syn på repetisjonsoppgavene enn Lisa. Selv om Felix beskriver at oppgavene hjalp ham med å repetere gammelt fysikkstoff, stiller han også spørsmålstegn ved hvor nødvendige oppgavene var. Videre forklarer han at læreren ga akkurat nok hjelp, og ikke ga svaret med en gang, men heller oppfordret ham til å tenke selv. Basert på dette mener Felix at det kanskje ikke var så nødvendig med de første repetisjonsoppgavene, da læreren kunne hjulpet ham med å huske det kjente fagstoffet når hen gikk rundt og hjalp. Han legger imidlertid til at en slik løsning kanskje ikke ville fungert i praksis, siden læreren ikke hadde rukket rundt til alle elevene.

Under intervjuet poengterte Hugo at han hadde fått god trening i å tolke grafer. Her er et eksempel på hvordan han uttalte seg om dette:

Sitat 6-8 **Hugo:** ...jeg fikk ut tre grafer, men de stemte ikke med virkeligheten. Etter 5 sekunder var ballen 120 meter over bakken, og den kom jo ikke ned igjen heller, noe den pleier å gjøre. Og da er det sånn «er dette riktig?». Når man tenker på virkeligheten, var det jo helt feil, siden det ikke er sånn som faktisk skjer. Først ble jeg glad for å ha fått tre grafer og fått til oppgaven, men så innså jeg at de måtte være feil siden det ikke stemte med virkeligheten.

Sitatet viser at Hugo i utgangspunktet trodde han hadde løst oppgaven rett, men etterhvert innså at grafene hans ikke stemte med virkeligheten. Altså evaluerte han løsningen sin, avgjorde at den var feil og identifiserte hvorfor løsningen var feil. Dette kan på mange måter vitne om at Hugo har en dypere fysikkforståelse. Det er imidlertid viktig å nevne at siden intervjuet ble gjennomført i etterkant av økt 2, men før økt 3, kom ikke alle elevene like langt som ham. Det var derfor flere elever som ikke fikk frem de tre grafene, og følgelig ikke fikk anledning til å tolke dem. Ruby beskriver det slik:

Sitat 6-9 **Ruby:** Vi byttet tall og la inn verdier til det vi trengte, men vi fikk fortsatt ikke til å plotte grafene.

Funnene fra intervjuene tyder på at animasjonen som ble vist i forbindelse med oppgaven «Vertikalt kast med luftmotstand» (se Vedlegg F), fikk mindre betydning enn forventet. Under

intervjuene ble informantene spurt om de følte at det hjalp å se animasjonen av oppgaven før de skulle arbeide med den, eller ikke. Her er et eksempel på hvordan informantene fra intervju 1 responderte på dette spørsmålet:

Sitat 6-10

> **Lisa:** Jeg synes det hjalp litt ja. Da tenker du liksom litt mens du ser den (animasjonen). Som «hvorfor går den (ballen) ned? Er det luftmotstand? Hva er det?». Du får et spørsmål i hodet du prøver å finne svar på. Så det var bra å få se den animasjonen ja.
> **Ruby:** Enig

Lisa og Ruby formidler at de opplevde det som nyttig å se animasjonen før de begynte å arbeide med oppgaven. I uttalelsen til Lisa legger jeg spesielt merke til at hun fikk «spørsmål i hodet» da hun så animasjonen, noe som ga henne et større overblikk over hva som skjedde i situasjonen, og tanker om hvorfor de ulike tingene skjedde. Et av målene med å vise animasjonen, var å sette i gang slike tankeprosesser. I intervju 1 var responsen noe annerledes. Her var informantene i større tvil om animasjonen hjalp noe nevneverdig:

Sitat 6-11

> **Hugo:** Jeg vet ikke hvor mye animasjonene hjalp, men du måtte tenke over grafen som programmet plottet. Dette stemmer ikke helt...når du kaster ballen opp, må den jo komme ned etter en viss tid. Så når den etter 5 sekunder er 120 meter over bakken, og ikke kommer ned igjen, så skjønner man at noe ikke stemmer. Men jeg vet ikke om det var animasjonen som gjorde at jeg tenkte det. Men jeg tror at animasjon kan være et godt hjelpemiddel, men følte ikke at jeg trengte det i den oppgaven. Vi har jo hatt et vertikalt kast før og vet litt hvordan det ser ut. Så vi kunne liksom tenke tilbake til det.
> **Felix:** Mmmm. Nei, jeg mener at det gjør det jo litt enklere når du får visualisert det da. Så...det var kanskje ikke den mest avanserte oppgaven («vertikalt kast med luftmotstand»), men prinsippet er jo bra da. At du får se det først og så vet du hvordan du skal programmere det etterpå. Så i denne oppgaven hjalp det vel egentlig kanskje ikke så mye, men hvis det hadde vært en mer avansert oppgave, så hadde det jo hjulpet meg da.

Sitatet viser at Felix og Hugo er enig i at det å se en animasjon kan hjelpe dem med å forstå situasjonen de skal programmere. Animasjoner kan gjøre det enklere å evaluere grafer og avgjøre om svarene deres gir mening. Felix og Hugo er imidlertid nokså tydelige på at animasjonen ikke var spesielt til hjelp i akkurat denne oppgaven, da de kjente godt til situasjonen fra før, og kunne trekke inn tidligere kunnskaper om situasjonen. De poengterer likevel at dersom oppgaven hadde handlet om en ukjent situasjon, så hadde det vært nyttig å se en tilsvarende animasjon i forkant av å skulle arbeide med oppgaver.

Resultatene tilsier at elevene forstår at programmering er en del av fysikken, og skjønner hvorfor. Basert på dette kan det sies at elevene har blitt kjent med, og forstår hvordan fysikere arbeider. Her er et eksempel på hvordan dette ble formidlet i intervjuet:

Sitat 6-12 **Felix:** Jeg skjønner jo hvorfor det er en del av fysikkfaget. Fordi det er jo veldig relevant opp mot at det ikke er alt man kan få svar på med å kun bruke penn og papir.

Det fremkommer at Felix forstår og er klar over at ikke alle problemer kan løses analytisk. I forlengelsen av dette forteller han at han forstår at det kan være lønnsomt å benytte programmering og numeriske metoder til å løse problemer som også kan løses med penn og papir. Basert på dette kan Felix sies å ha forstått programmeringens og numerikkens rolle i fysikken.

Elevutfordringer tilknyttet undervisningsopplegget

Under gruppeintervjuene beskrev informantene en rekke aspekter ved undervisningsopplegget som de opplevde som utfordrende. I det følgende tar jeg for meg de mest framtredende av disse.

Programmering: et nytt språk

Informantene i intervju 1 beskrev utfordringer som dukket opp i forbindelse med programmeringen. Her forklarer Hugo programmeringen som et nytt språk de må lære seg i tillegg til fysikken:

Sitat 6-13 **Hugo:** Jeg vil si at programmering er et eget språk, og ikke bare enda et kapittel i boka.

Elevene påpeker at programmering bringer med seg mange nye ord og begreper, og hvordan disse brukes hyppig, men sjelden forklares eksplisitt, slik at de forstår hva ordene og begrepene innebærer. Felix beskriver det slik: «Det er som å navigere i tåka. Du vet for eksempel at du skal opp på en fjelltopp, men tåka er for tjukk til at du ser den». Han konkretiserer også med et eksempel, der en lærer snakket om *pseudokode* og hvor nyttig det var, uten å gi en tydelig forklaring på hva det var. Dermed satt elevene igjen som spørsmålstegn, mens læreren la ut i det vide og brede om denne fantastiske nye tingen de ikke visste hva var. Elevene trekker også frem hvordan man er helt avhengig av å ha helt riktig syntaks, på samme måte som grammatikk, for at programmet skal kjøre. Under intervjuet ble dette formidlet slik:

Sitat 6-14

Hugo: Må være obs på det (syntaksen), noe som tar mye tid. I GeoGebra har vi lært at opphøyd er $\wedge$, men i Python er det $**$, og det har ikke lærerne fortalt oss.

Her fremkommer det at elevene er vant til å arbeide på ulike plattformer, med ulik syntaks. Vi ser eksempelvis at måten man «sier ifra til programmet» at man ønsker å kvadrere et tall, er forskjellig i GeoGebra og Python. Dersom elevene ikke gjøres oppmerksomme på slike syntaksmessige skift, kan de oppleve det å programmere i Python som unødvendig utfordrende.

Manglende kontinuitet i opplæringen i programmering

Informantene forteller også at de sjelden programmerer i fagene på skolen, og at det dermed ofte går lang tid mellom hver gang. Hugo beskriver hvordan «man glemmer mye av den grunnleggende programmeringen og det man lærte sist» fra gang til gang. Han forteller også at læreren sjelden har tid til å repetere grunnleggende programmering før hen går videre på nytt fagstoff. I likhet med Hugo, syntes informantene fra intervju 2 at opplæringen i programmering har vært oppstykket, og at det kunne gå lange perioder mellom hver gang de programmerte. Lisa og Ruby forklarte det slik:

Sitat 6-15

JXS: Dere sa at dere hadde erfaring med programmering, men at det var mest fra matten.
Lisa: Mmm.
JXS: Hvor ofte programmerer dere i matten da?
Lisa: Ikke mer enn én gang hvert semester.
Ruby: Vi hadde ikke så mye i fjor, men vi hadde litt på slutten av skoleåret. Men i år har vi hatt én time for hvert halvår. Men nå, etter en prøve, hadde vi plutselig en uke med programmering.
JXS: Når dere skal begynne å programmere igjen, husker dere hva dere lærte sist? Som at dere husker å ha hørt ordet «for-løkker», men husker ikke helt hva den gjør eller hvordan man skriver en.
Lisa: Ja, for eksempel når vi programmerer, så får vi beskjed om at det bare skal brukes. Så vi lærer det liksom aldri ordentlig. Vi vet ikke hva hver linje betyr. Vi får beskjed: «Bruk denne og denne og sett tall på det og kjør på».
JXS: Så dere føler ikke at dere lærer å skjønne kode?
Lisa: Nei. Vi må liksom lære oss alt på nytt hver gang, nesten.

Ifølge informantene blir det opp til elevene selv å repeterte den grunnleggende programmeringen. Basert på dette kan man på mange måter si at elevene tar to steg frem og ett tilbake for hver programmeringsøkt, som følge av manglende kontinuitet i opplæringen i programmering.

Arbeide ut fra delvis ferdiglaget programkode

Under intervjuene tok informantene også opp utfordringer tilknyttet det å få utdelt delvis ferdiglaget programkode. Her er et eksempel som illustrerer hvordan en elev mener dette kan gjøre det vanskeligere å «debugge» og oppdage hvor feilene ligger, samt kjøre koden fortløpende:

Sitat 6-16 **Hugo:** Når jeg programmerer selv, så deler jeg opp programmet og sjekker at hver del fungerer før jeg går videre. For eksempel å printe og se på feilmeldinger. Når vi får delvis ferdig kode er det vanskelig å oppdage hvor feilene ligger.

Fra sitatet ser vi at Hugo har en konstruktiv problemløsningsstrategi. Det ble også nevnt at det delvis ferdiglagde programmet opplevdes som langt, med mange kommentarer. Likevel tydeliggjorde informantene fra intervju 1 at de syntes det var nyttig og satte pris de mange kommentarene, siden det gjorde at de bedre forsto programkoden. På den annen side opplevde de at det ble i overkant mye tekst, og at det dermed kunne være vanskelig å finne frem i programkoden. Informantene i intervju 2 syntes også at det var mye tekst å forholde seg til, men hadde et annet syn på kommentarene i programkoden. Lisa forteller at hun ikke var så opptatt av å forstå den utdelte koden, og dermed unnlot å lese mange av kommentarene. Her beskriver hun at mengden kommentarer opplevdes som overveldende, mer enn et hjelpemiddel i arbeidet med oppgaven:

Sitat 6-17 **Lisa:** Jeg må ikke forstå hele koden. Det viktigste er at jeg kan bruke den...det var mye tekst etter hverandre. Da ender det opp med at man ikke leser alt.

Problemløsning

Å drive problemløsning var også noe informantene opplevde som utfordrende. I hovedsak syntes de at det kunne være vanskelig å stykke opp problemet i tilstrekkelig små biter, slik at de kunne begynne på oppgaven. Lisa beskriver det slik:

Sitat 6-18 **Lisa:** Med programmering vet jeg ikke hvordan jeg skal starte eller hva jeg skal skrive. Jeg forstår det ikke, så jeg vil heller løse det for hånd, selv om jeg vet det blir feil. Da føler jeg at jeg får til noe.

Her ser vi at Lisa synes det er vanskelig å komme ordentlig i gang med programmeringen, og dermed mister motivasjonen litt. Videre forteller Lisa at hun hadde følt at hun fikk til mer

dersom problemet ble løst for hånd, selv om hun vet at dette ikke ville gitt rett svar. Selv om Lisa ikke føler at hun mestrer å løse problemer med programmering, indikerer uttalelsen hennes at hun er innforstått med at ikke alle problemer kan løses analytisk, og at programmering er et nyttig og tidsbesparende problemløsningsverktøy.

Informantene trakk også frem at de opplevde at lærertettheten i timen var for lav. Hugo beskriver det slik:

Sitat 6-19

> **Hugo:** Som Felix sa er vi 32 elever i klassen og vi er for mange elever for én lærer. Og når vi programmerer skjer det så mye feil, og det er små feil du selv ikke oppdager, men som en lærer, som har programmert mye før og har kompetanse, lettere kan oppdage. Og når du ikke får hjelp tar det veldig lang tid å finne feilene selv. Og dere var jo to lærere (Juni og forfatteren) og dere merket jo at det var travelt. Dere rakk jo nesten ikke å puste. Alle ville jo ha hjelp. Og da synes jeg man burde ha flere lærere, selv om skolen sikkert ikke har råd til det.

Fra sitatet ser vi at Hugo opplevde at det kunne være vanskelig å oppdage småfeil på egenhånd, og at han dermed ble sittende og vente på hjelp fra lærer for å komme seg videre i arbeidet. Han poengterer også at andelen elever som ønsket hjelp, så ut til å overskride kapasiteten til de to lærerne som gikk rundt og hjalp. Det er imidlertid viktig å påpeke at lærertettheten i en slik time vanligvis er én lærer på omtrent 25-30 elever, og ikke to lærere på 31 elever som i dette tilfellet. Til tross for at lærertettheten var høyere enn normalt, opplevde informantene det som nødvendig med enda flere lærere som kan hjelpe under programmeringsøkter.

Elevenes motivasjon

Under intervjuene snakket informantene også om motivasjonen deres i løpet av undervisningsopplegget. De tok blant annet opp at de syntes det var motiverende å gjøre minioppgavene og repetisjonsoppgavene (se Sitat 6-4). Her var det motiverende at undervisningsoppleggets gode progresjon bidro til at de fikk til oppgavene, noe som ga dem mestringsfølelse. Hovedfokuset lå likevel på motivasjon knyttet til programmeringen. I Sitat 6-4 forteller Lisa om at mestringsfølelsen hun fikk da hun løste minioppgavene ga henne selvtillit til å tørre å prøve seg litt på andre programmeringsoppgaver senere i undervisningsopplegget. Ruby fortalte at hun syntes programmeringen ble forklart på en hverdagslig og forståelig måte, og at dette ga henne økt selvtillit. Hun eksemplifiserte dette med måten jeg hadde brukt hverdagslige analogier til å forklare hva arrays var, og hvorfor det ikke gikk å fylle en array

med flere elementer enn antall plasser i arrayen. Her ble et sett med 3 kleshengere, som bare kunne holde oppe ett plagg hver, brukt som analogi til en array med 3 plasser. Videre ble det forklart at dersom man prøver å henge et fjerde plagg på settet med kleshengere, faller alt ned, og at dette tilsvarer en feilmelding når vi programmerer.

Her er et annet eksempel på at elevene opplevde økt selvtillit på programmeringsfronten:

Sitat 6-20

> **Felix:** ...jeg synes de første oppgavene (mini-oppgavene), hvor vi fikk frisket opp litt, var veldig givende. Så jeg fikk litt mer selvtillit på programmeringen. Også følte jeg at jeg kunne bruke det (det de lærte i mini-oppgavene) videre til å lage et program litt selv. Så jeg føler at min kompetanse i programmering har økt etter disse to timene.
> **Hugo:** ...jeg føler jeg har lært mer på disse to timene enn de to forrige årene. Jeg synes jeg har lært mer forståelse, siden jeg fikk en utfordring. Det har jeg ikke pleid å få, siden vi bare pleier å kopiere det læreren skriver. Så utfordringen gjorde meg motivert. Og det likte jeg.

Fra sitatet ser vi at Felix og Hugo opplevde mestringsfølelse i arbeidet med minioppgavene, og at dette ga dem motivasjon til å tørre å prøve seg på andre programmeringsoppgaver. Generelt virker det som om elevene synes arbeidet med minioppgavene var givende og motiverende. Vi ser også at enkelte av elevene ble motivert av å få en utfordring, der de skulle skrive mye av programkoden selv.

Elevenes tanker rundt kompetansemålet om numeriske metoder og programmering

Basert på uttalelsene til Felix og Hugo under intervjuet, var det naturlig å spørre dem om deres tanker rundt det å lære numeriske metoder og programmering i fysikk 1. Det virket som begge forsto hvilken rolle de to temaene har i fysikkfaget, og dermed hvorfor de inngår i et av kompetansemålene i faget. Sitat 6-21 (se neste side) viser hvordan disse refleksjonene kom til syne under intervjuet. Her ser vi tydelig at informantene forstår hvorfor numeriske metoder og programmering er relevant i fysikken. Fra utdraget fremkommer det også noe interessant, nemlig at de opplever at det settes av og brukes for lite tid til opplæring i programmering i undervisningen. Hugo underbygger dette med å beskrive programmering som et «eget språk» som tar tid å lære. Av den grunn mener han at det bør settes av betraktelig mer tid til en slik «språkopplæring» enn «andre kapitler i boka». I forbindelse med dette beskriver Felix at han føler at han ikke mestrer programmeringen, og at dette «trekker ned kompetansen» hans i faget.

I likhet med Hugo, mener Felix at de burde få bedre opplæring i programmering og at det burde settes av mer tid til det når det skal være en del av fysikkfaget. Her forklarer Felix at han tror dette kan bidra til at flere elever føler at de mestrer programmeringen:

Sitat 6-21

> **Felix:** Man kan jo ikke få presise, analytiske svar på alle problemer. Så jeg skjønner jo hvorfor programmering er veldig relevant. Og for å være helt ærlig så blir jeg litt bitter over det, siden jeg føler at jeg ikke mestrer programmering spesielt mye. Eh... det kan jo ha noe med opplæringen vi har hatt i det (fra matematikken og fysikken). Så jeg skjønner jo hvorfor det (programmeringen) er det (del av fysikkfaget), men hvis jeg kunne valgt så hadde jeg valgt et uten det (programmering). Fordi jeg føler at det trekker ned kompetansen min i fysikk, fordi jeg ikke mestrer det godt. Og for at jeg da skal føle at jeg mestrer det, så synes jeg jo at vi burde hatt mer om det.... Hvis det først skal være en del av fysikken så synes jeg vi burde bruke mer tid på det sånn at alle får følt at de mestrer det da.
> **JXS:** Hugo, hva tenker du om det?
> **Hugo:** Jeg er litt enig med Felix. At når det står på kompetansemålet, så er det viktig at vi bruker nok tid på det. Jeg vil si at programmering er et eget språk, ikke bare enda et kapittel i boka. Og det (programmering) er noe annet. Så man bruker mer tid på det enn andre kapitler. Selv liker jeg programmering, og vil jobbe med det. Da er det viktig å se hvordan programmering brukes til andre ting enn ren IT, hvor man lager nettsider. Så jeg vil bruke programmering til å se på virkeligheten, sånn som vi har gjort med deg.

Hugos uttalelse tyder på at han anser programmering og numeriske metoder som en integrert del av fysikkfaget. Videre beskriver han at det kan hjelpe dem å forstå virkeligheten, på lik linje med likninger eller begreper. Kompetansemålet søker på mange måter å gi elevene kjennskap til programmering og numeriske metoder, slik at de blir i stand til å reflektere på en lignende måte som Hugo.

Elevenes forslag til forbedringer av undervisningsopplegget

Informantene ble også oppfordret til å komme med forslag de mente kunne forbedre undervisningsopplegget. Flere gode forslag til hvordan opplegget kunne forbedres ble lagt frem. I dette delkapittelet vil disse forslagene presenteres og bli gjort rede for.

I første omgang kommuniserte informantene at de likte minioppgavene (se Vedlegg D), men at de gjerne kunne inneholdt flere oppgaver. Her ønsket de både flere oppgaver om hvert av de gitte temaene, og oppgaver om andre temaer. Hugo beskriver det slik:

Sitat 6-22 **Hugo:** ...Det kunne jo vært flere oppgaver om hver ting (variabler osv) eller om flere temaer. Men det tar jo veldig lang tid å jobbe med, så det hadde jo ikke gått.

Dette begrunnes med at de hadde kunnet arbeide mer i dybden med hvert av temaene: variabler, if- og else-setninger, for-løkker, arrays, feilsøking og opprydding av feil i programkode, i tillegg til at de kunne blitt bedre kjent med nye temaer. Informantene legger også til at oppgavene om samme tema gjerne kunne hatt en stigende vanskelighetsgrad, slik at de både fikk repetert kjent fagstoff, men også fikk en utfordring. Imidlertid påpeker informantene at denne mer omfattende versjonen av minioppgavene hadde krevd en lengre arbeidsøkt, noe det trolig ikke hadde vært tid til. De forteller også at minioppgavene opplevdes som relativt enkle og overkommelige, men at vanskelighetsgraden på programmeringsoppgaven 2d var litt for høy. Her poengterer de at det burde vært en jevnere stigning i vanskelighetsgrad:

Sitat 6-23 **Hugo:** ...Jeg følte at oppgave d ble litt for vanskelig i forhold til den forrige oppgaven. Så det kunne kanskje vært litt mer gradvis, og ikke sånn lett og så vanskelig.
Felix: Ja, det er jeg enig i. Jeg syntes det var et stort sprang. Jeg følte at jeg mestret startoppgavene (mini-oppgavene) du ga oss først. Litt sånn «ja, det her får jeg til». Og så syntes jeg det ble litt vanskelig litt sånn brått...Så hvis det hadde vært en litt mer gradvis overgang hadde det kanskje vært lettere for oss å mestre det da.

Videre kommer informantene med forslag til hvordan dette kunne vært løst. De trekker eksempelvis frem at en mulig løsning hadde vært å legge inn flere kommentarer i den utdelte programkoden, som ga dem tydeligere hint om hva de skulle skrive. Informantene foreslo også at det å gå nøye gjennom lignende eksempler i forkant av oppgave 2d kunne bidratt til en jevnere vanskelighetsgrad. Her er et sitat som illustrerer hvordan dette ble diskutert i intervju 1:

Sitat 6-24 **Hugo:** ...Vi kunne godt ha sett på kodeeksempler på lignende oppgaver sammen, og gått gjennom dem sakte og detaljert
Felix: Ja, fordi jeg synes det blir lettere å følge med og skjønne når det blir gjennomgått mer detaljert...fordi da er det lettere å holde oversikt og sånt.

Informantene forteller også at behovet for de beskrivende kommentarene i koden hadde vært mindre dersom de hadde gjennomgått lignende eksempler i forkant av programmeringsoppgaven 2d. De påpeker at med et slikt forarbeid, hadde det kanskje bare

vært behov for kommentarer der de skulle endre eller skrive egne kodelinjer i den utdelte programkoden. Løsningen Felix og Hugo presenterer gir elevene bedre mulighet til å forstå programkode, noe de de begge ønsker og anser som viktig. Det er verdt å nevne at løsningen informantene presenterer krever mer tid enn jeg hadde til rådighet i denne studien. Dette ble for øvrig poengtert av Felix og Hugo i deres fremlegg av forslaget. Det er også viktig å påpeke at ikke alle elever har et like stort ønske om å forstå programkoden. I intervju 2 kommuniserte informantene at det ikke var så viktig for dem å forstå koden. For dem var det viktigste at de fikk rett svar.

En av informantene fremhever at programmering i stor grad handler om problemløsning. Her beskriver han at «det var litt dumt» at de fikk utdelt den ferdiglagde programkoden mot slutten av økt 2, da det gjorde det enklere å kopiere den ferdiglagde koden, enn å finne egne feil og bruke den utdelte koden til å rette opp i dem:

Sitat 6-25

> **Hugo:** ...jeg synes det hadde vært veldig lærerikt om vi hadde skrevet programmet selv. Men det tar jo veldig mye tid. Det er jo veldig høy kompetanse, som jeg ikke tror alle får til… men du må feile for å komme fremover og det tar lang tid å feile og komme fremover.

Sitatet viser at Hugo er innforstått med at det å skrive hele programkoden på egenhånd hadde vært tidkrevende, men at det hadde gjort at han lærte mer. Videre påpeker han at det å skrive alt selv krever høy kompetanse, både teknisk og forståelsesmessig. Med utgangspunkt i dette legger han til at han tviler på at alle i klassen hadde fått til en slik oppgave, da oppgaven for noen av elevene i klassen ville vært i overkant vanskelig. Likevel påpeker han at det å lære, noe han omtaler som å «komme fremover», krever at man feiler. Og at det er først da man lærer noe nytt, og tar ett skritt videre. Informantene fra intervju 2 hadde ikke samme ønske om å skrive programkoden selv. De er tydelige i sin tale om at de skulle ønske de fikk utdelt helt ferdig programkode, hvor de kun skulle bytte ut verdier på variablene. Ruby beskriver det slik:

Sitat 6-26

> **Ruby:** Jeg synes det hadde vært mer interessant dersom vi fikk ferdig kode, der vi bare skulle bytte ut tall og verdier. Men vi måtte skrive inn flere linjer selv, og det blir litt vanskelig. Så hvis alt hadde vært ferdiglaget for oss hadde det vært bedre.

Fra sitatet fremkommer det at Ruby syntes det var vanskelig å skrive flere kodelinjer på egen hånd. Det er dermed rimelig å anta at hun hadde opplevd det som enda mer utfordrende å skrive hele programkoden selv.

6.6 Oppsummering

Funnene fra de ulike datainnsamlingsmetodene: oppgaveark, spørreskjema, elevsamtaler fra timen og gruppeintervjuer, tyder på at oppgavene elevene ble tildelt holdt et passelig nivå og var engasjerende for dem. Likevel opplevde elevgruppen noen av oppgavene, spesielt programmeringsoppgaven 2d, som ekstra utfordrende. Flere av elevene oppga også at de syntes det mest interessante med undervisningsopplegget var å programmere, selv om det var nettopp dette mange opplevde som noe av det mest utfordrende. Generelt virker det som om elevene forstår at noen fysiske problemer må løses numerisk, siden ikke alle problemer kan løses analytisk. I forlengelsen av dette virker det også som elevene på mange måter godtar numeriske metoder og programmering som en integrert del av fysikkfaget.

7 DISKUSJON OG KONKLUSJON

Som del av denne studien har jeg utviklet et undervisningsopplegg. Gjennom elevbesvarelser, lydopptak fra timen og gruppeintervjuer har jeg undersøkt hvordan numeriske metoder og programmering kan inngå som en konstruktiv del av fysikk 1. Jeg vil i kapittel 7.1-7.3 diskutere resultatene og besvare oppgavens problemstilling. Her besvarer kapittel 7.1, 7.2 og 7.3 henholdsvis forskningsspørsmål 1, 2, og 3. I kapittel 7.4 diskuteres elevenes forslag til forbedring av undervisningsopplegget. Jeg presenterer i kapittel 7.5 potensielle implikasjoner av studiens funn og kommer med anbefalinger til fysikklærere, lærebok- og læreplanforfattere. Her kommer jeg også med forslag til videre forskning. I kapittel 7.6 belyses studiens styrker og svakheter, før jeg til slutt forsøker å trekke noen konklusjoner i kapittel 7.7.

7.1 Gjennomføring av opplegget: elevenes utbytte og utfordringer

I kapittel 6.3 fremkom det at klassen som helhet hadde erfaring med både numeriske metoder og programmering fra fysikken og matematikken. Dermed hadde elevene en viss kjennskap til metodene. Samtidig ser vi at intervjupersonene beskriver opplæringen de tidligere har fått i programmering som oppstykket, og at de dermed glemmer det grunnleggende fra gang til gang. I forlengelsen av dette beskrev deltakerne programmering som et nytt språk, med mange nye ord og begreper man måtte kunne «helt perfekt» for at programmet skal kjøre. Denne beskrivelsen er i tråd med det Sørby og Angell (2012) og Taub et al. (2015) skriver, men er også noe jeg tør å påstå at enhver som har programmert kan kjenne seg igjen i. Siden elevene ikke har fått arbeidet jevnt med programmeringen i skolefagene, og dermed må lære seg det grunnleggende gang på gang, er det rimelig å anta at de opplever liten progresjon. Dette, i kombinasjon med at informantene beskriver den programmeringsmessige progresjonen

i vanlige skolefag som jevnt høy, gir sannsynlig et gap mellom hva det forventes at de kan og kunnskapen de faktisk sitter på. På mange måter kan dette føre til at læringen foregår utenfor elevenes proksimale utviklingssone (Vygotsky, 1934/2001), slik at risikoen for at elevene opplever kognitiv overbelastning øker (Rutkowski & Saunders, 2019). Som resultat vil sannsynligheten for at elevene opplever motivasjon og mestring, være nokså lav (Ryan & Deci, 2017).

For å unngå at elevene opplever kognitiv overbelastning, tok jeg bevisste grep i utviklingen av undervisningsopplegget. Ett av grepene var å legge inn minioppgavene, der elevene fikk trening i grunnleggende programmering, som håndtering av variabler, for-løkker og arrays. Dette er i tråd med hvordan Rutkowski og Saunders (2019) mener at kognitiv overbelastning kan unngås. Informantenes uttalelser fra intervjuet indikerer at de opplevde minioppgavene som en nyttig repetisjon og passe vanskelige. Minioppgavene ga altså en god progresjon, og fikk dermed den effekten jeg hadde tenkt. Den gode progresjonen i oppgavesettene virker også å ha gitt elevene mestringsfølelse, noe som kan ha bidratt til at de opplevde programmering som overkommelig, og turte å prøve seg på andre programmeringsoppgaver som ble gitt senere (se Sitat 6-4 i kapittel 6.5). Dette stemmer godt overens med selvbestemmelsesteorien presentert av Ryan og Deci (2017). I tillegg underbygger disse funnene Vygotskys (1934/2001) teori om den proksimale utviklingssonen, der elevene tildeles «passe vanskelige» oppgaver. Det er viktig å påpeke at hva som ansees som «passe vanskelig» avhenger av enkelteleven, men også hvor i faglandskapet de befinner seg. Et eksempel er den delvis ferdiglagde programkoden som elevene opplevde ulikt. Dette betyr at mengden hjelp elevene trenger endrer seg etterhvert som elevenes proksimale utviklingssone utvides. Gjennom opplegg med jevn og god progresjon, kan denne utvidelsen skje kontinuerlig, der hver nye sone overlapper med den forrige, noe som medfører at risikoen for kognitiv overbelastning reduseres.

Med utgangspunkt i funnene fra oppgavearkene, virker det også som om elevene har opparbeidet seg kunnskap om numeriske metoder. Her fremkommer det at elevene kan beskrive Eulers metode med egne ord, men også er inneforstått med hvordan lengden på tidssteget dt påvirker den numeriske løsningen. Fra elevbesvarelsene er det også tydelig at de er i stand til å bestemme en passelig skrittlengde, og kan argumentere for det på en fysikkfaglig måte. Studiens resultater tyder altså på at undervisningsopplegget, hvor numerikk integreres som en naturlig del av fysikkfaget, har lyktes i å imøtekomme kompetansemålet om numeriske metoder

og programmering.

En forutsetning for at elevene skal ha utbytte av å arbeide med numeriske metoder og programmering, er at de tildeles komplekse nok problemer. Dette er problemer som det er hensiktsmessig å bruke både numeriske metoder og programmering til å løse. Dersom elevene gis mindre komplekse problemer, som kan løses med penn og papir, kan de oppleve det som unødvendig vanskelig og tidkrevende å lære en ny, og i deres øyne kanskje mer tungvint, løsningsmetode. Ett av formålene med kompetansemålet om numeriske metoder og programmering, er på mange måter å synliggjøre for elevene at det er et kraftfullt verktøy som er nyttig, og noen ganger nødvendig, for å løse fysiske problemer. Som nevnt av både Haraldsrud og Tellefsen (2018) og Malthe-Sørenssen et al. (2015) åpner verktøyet opp for at elevene kan arbeide med mer virkelighetsnære problemer. Disse problemene kan oppleves som mer interessante enn sterkt idealiserte problemer. Dette virker det som undervisningsopplegget har lykkes med, da problemene elevene ble tildelt gjenspeiler hverdagslige fenomener.

For at elevene skulle forstå hva numeriske metoder er og innebærer, ble de introdusert for Eulers metode gjennom kjente problemer. En slik tilnærming til det nye fagstoffet kan sies å være i tråd med Vygotskys (1934/2001) syn på læring. Det utformede undervisningsopplegget la opp til at elevene skulle gjøre flere påfølgende beregninger, med en viss steglengde for hånd, før de deretter ble bedt om å gjøre et urealistisk antall nye beregninger. Dermed ble elevene kjent med Eulers metode, før de skulle oversette den til programkode. Elevene fikk også erfare at numeriske metoder ikke nødvendigvis foregår på en datamaskin med programkode, men at det som regel er den mest effektive løsningsmetoden. At elevene så nytten og behovet for numeriske metoder og programmering i fysikken, kom tydelig frem i besvarelsene deres på spørreskjemaet (se kapittel 6.3), og det virker som de i stor grad godtar numerikk og programmering som en integrert del av fysikkfaget. Årsaken til dette kan være at elevene ble kjent med hvordan numeriske metoder og programmering anvendes i fysikken, og at de dermed har fått innsikt i hvordan fysikere arbeider. Dette samsvarer med Sitat 6-12 i kapittel 6.5.1, der Felix forteller at ikke alle problemer kan løses analytisk, og at numeriske metoder og programmering er nyttige verktøy som kan brukes til å løse problemer, også de som lar seg løse analytisk med penn og papir. Resultatene fra intervju 2, med Lisa og Ruby (se kapittel 6.5), tyder også på at elevene forsto prinsippet om Euler metode, selv om de strevde med å oversette metoden til programkode.

Generelt beskriver elevene utfordringer som er å finne som momenter innenfor digital problemløsning i taksonomien for «computational thinking» (CT), presentert av Weintrop et al. (2016) (se kapittel 2.1). Intervjuene og svarene på spørreskjemaet indikerer at mange av elevene opplevde programmeringsoppgaven 2d på oppgavearket «Vertikalt kast med luftmotstand» som utfordrende. At elevene måtte forholde seg til både det fysikkfaglige og det tekniske med programmeringen, kan ifølge Malthe-Sørensen et al. (2015) være problematisk. Årsaken til dette kan være at elever generelt sliter med å forstå og tolke grafiske representasjonsformer, og at digital visualisering dermed kan være mer til hinder enn støtte for elevens læring (Angell et al., 2019). Basert på elevsamtalene fra timen, intervjuer og besvarte spørreskjemaer, virker dette imidlertid ikke å ha vært tilfellet. Dataene tyder heller på at den digitale visualiseringen, i kombinasjon med elevenes kjennskap til den modellerte situasjonen, ga dem et større eierforhold til grafene, og ga dem verdifull trening i å tolke og forklare grafer. Det bør nevnes at observasjoer og lydopptak tyder på at elevene i større grad anvender egen fysikkforståelse til å evaluere resultatene sine i samtalene der læreren deltok aktivt, enn i elev-elev-samtaler (se kapittel 6.4). Dette kommer tydelig frem i Sitat 6-2, hvor elevene ikke brukte fysikkforståelse da de skulle avgjøre om akselerasjonsgrafen var riktig.

7.2 Elevenes læringsopplevelse av å arbeide med numeriske metoder og programmering

Datamaterialet viser at elevene begrunner deres opplevelse av undervisningsopplegget ut fra læringsopplevelse og læringsutbyttet, fremfor underholdningsverdi. Et eksempel på dette er at flere elever ytret at de opplevde minioppgavene som morsomme, og underbygget dette med de fikk nyttig repetisjon av kjent fagstoff, og trening i å programmere. Dette illustrerer at fysikkelever er faglig motiverte.

Arbeidsmåter og løsningsmetoder

Elevene oppga på spørreskjemaet at de hadde erfaring med numeriske metoder og programmering fra tidligere fysikk- og matematikkundervisning. Det fremkom også at de har arbeidet med lister som ikke har en forhåndsbestemt lengde. Elevene hadde ikke vært borti arrays og preallokering, hvor man på forhånd bestemmer antall elementer i arrayen. Arrays

og preallokering er mye anvendte løsningsmetoder på universitetsnivå og blant forskere, siden metoden krever relativt få operasjoner og er mer tidseffektiv. Å benytte arrays og preallokering i programmeringsproblemer kan gi elevene innsikt i fysikeres arbeidsmåter. Dermed kan man si at elevene bør få trening i å løse programmeringsproblemer med nettopp denne metoden, som er i tråd med det Malthe-Sørenssen et al. (2015) skriver om et mer autentisk fysikkfag.

I undervisningsopplegget ble elevene introdusert for NumPys system for arrays, hva det er og hvordan det fungerer. Forskjellen mellom arrays og lister er at man i arbeid med lister ikke må forholde seg til antall elementer i listen, og hvilken plass det nye elementet skal legges inn på, slik som man må med NumPy arrays. På den måten kan lister sier å være mer brukervennlige, og dermed være en løsningsmetode som passer godt for nybegynnere. Elevene opplevde det som utfordrende å bli introdusert for arrays som en ny løsningsmetode innen programmering. Følgelig kan elevene ha opplevd kognitiv overbelastning (Rutkowski & Saunders, 2019). En av årsakene til dette kan være at de hadde mer enn nok med å sette seg inn i det fysiske problemet som skulle løses, og dermed ikke hadde kapasitet til å samtidig implementere en ny løsningsmetode. Observasjoner fra timen tyder på at dette kan ha vært tilfellet for noen av elevene. Elevene var godt kjent med lister, elementenes indeksering, og hvordan ulike elementer kan hentes ut. Likevel forsto de ikke umiddelbart hvordan arrays fungerte, selv om det praktisk talt er det samme som lister. I arbeidet med minioppgavene, virket det som om elevene fikk til oppgavene om arrays uten å møte på særlig store problemer. Dette tyder på at elevene kunne håndtere og bruke arrays i én setting, men opplevde det som utfordrende å anvende det i en annen og mer kompleks setting. Dette funnet underbygger at læring er situert, som presentert av Brown et al. (1989).

Besvarelsene på oppgaveark og spørreskjema tyder også på at elevene forsto hva numeriske metoder er. De virket imidlertid å være mer opptatt av programmeringen og hvordan skrive programkode, enn de bakenforliggende prinsippene numeriske metoder er tuftet på, som de hadde relativt god kontroll på. Altså lå problemet med å simulere det fysiske problemet hovedsakelig i å oversette det de kunne om den numeriske metoden til programkode.

Animasjoner og modelleringsoppgaver

Studiens resultater viser at elevene hadde ulik grad av kjennskap til, og forståelse av, situasjonen «Vertikalt kast med luftmotstand», som skulle modelleres i oppgave 2d. Sitat 6-10 (se kapittel

6.5) indikerer at noen av informantene opplevde det som nyttig å se animasjonen, da det satte i gang tankeprosesser og gjorde at de bedre forsto situasjonen. Dette fremheves av Mork og Erlien (2017) som en av fordelene ved å inkludere animasjoner i fysikkundervisningen. Andre informanter påpekte at animasjonen var fin å se, men ikke tilførte ny informasjon om situasjonen, som de ikke visste fra før (se Sitat 6-11 i kapittel 6.5). Videre påpekte de at det trolig hadde vært mer nyttig å se animasjonen av en situasjon de hadde mindre kjennskap til, eller en situasjon som var vanskeligere å se for seg. Animasjonen hadde altså ikke like stor effekt som jeg hadde trodd (se kapittel 6.5). Likevel virker det som at animasjonen utelukkende hadde en positiv effekt, dersom den i det hele tatt hadde en effekt. Bruken av animasjonen sikret at samtlige elever hadde kjennskap til situasjonen, før den skulle modelleres ved hjelp av numeriske metoder og programmering, og oppfyller på den måten LIST-prinsippet om Lav Inngangsterskel og Stor Takhøyde (Wæge & Nosrati, 2018, s. 82-83).

Selv om effekten av animasjonen var mindre enn forventet, kan det være ønskelig å beholde den som del av undervisningsopplegget. Ved å beholde animasjonen gis elevene muligheten til å bli bedre kjent med *visuelle* representasjonsformer, og få trening i å se sammenheng mellom fysikkens ulike representasjonsformer, deriblant *matematisk-symbolsk*, og oversette mellom dem. Av Angell et al. (2019) beskrives denne treningen som nødvendig for at elevene skal kunne arbeide med realistiske problemer. Funnene fra besvarte spørreskjemaer og elevsamtaler fra timen indikerer at elevene opplevde at de fikk anledning til å trene på å oversette mellom fysikkens ulike representasjonsformer. Vi ser også at elevene behøver støtte fra lærer i å videreutvikle disse ferdighetene, slik at de ser en tydeligere sammenheng mellom ulike representasjonsformer som beskriver det samme fenomenet. I forbindelse med programmering og numeriske metoder, er dette viktige ferdigheter som elevene bør få trening i å bruke og videreutvikle. I gjennomføringen av undervisningsopplegget ble dette i hovedsak gjort i samtale med lærer, der elevene fikk støtte i og ble oppfordret til å vurdere om grafene de fikk frem virket rimelige, basert på deres kjennskap til det fysiske fenomenet. Læreren virker altså å ha bidratt til å gi elevene en god læringsopplevelse, og dermed tilrettelagt for motivasjon, mestring og læring (Ryan & Deci, 2017).

Undersøkelsens resultater viser også at elevene relativt enkelt fikk til å løse rene programmeringsoppgaver og fysikkoppgaver, men strevde med modelleringsoppgavene, der de måtte kombinere programmering og fysikk. Dette stemmer godt overens med det Sørby

og Angell (2012) skriver om at elevene sliter med å arbeide som den konseptuelle fysikeren og programmereren samtidig. Elevene oppga på både spørreskjema og i intervju at de hadde noe erfaring med programmering, men at denne i hovedsak innebar å se ferdiglaget kode eller skrive av programkode som læreren presenterte på storskjerm. Elevene opplever altså på mange måter at de har relativt lite kunnskap om programmering, tilhørende tankegang og hvordan man skriver egen programkode. Dette kan være en av årsakene til at de tekniske aspektene ved programmering virket å overskygge fysikkinnholdet i modelleringsoppgavene. Relativt store deler av undervisningstiden gikk med til å hjelpe elevene med å forstå programmering, noe som samsvarer med det Taub et al. (2015) skriver om at programmering ofte kan overskygge fysikkinnholdet i timen. Det er likevel viktig å påpeke at elevene opplevde at undervisningen inneholdt mye fysikk (se kapittel 6.3). Elevene arbeidet altså med både fysikk og programmering, noe som ga dem verdifull trening i å navigere mellom ulike representasjonsformer (se Sitat 6-3 i kapittel 6.4 og kapittel 6.5).

7.3 Dilemmaer relatert til undervisning om numeriske metoder og programmering

I arbeidet med å besvare studiens problemstilling, har det dukket opp en rekke nye dilemmaer. Her presenteres leseren for dilemmaene og en fortløpende diskusjon om dem.

Hvordan tolke kompetansemålet om numeriske metoder og programmering?

Et dilemma handler om hvordan kompetansemålet om numeriske metoder og programmering, som er utgangspunktet for denne studien, skal tolkes. Kompetansemålet inkluderer verbet «*å bruke*». Utdanningsdirektoratet (2021) definer «*å bruke*» slik:

> «*Å bruke vil si at vi gjør oss nytte av noe eller utfører en handling for å oppnå et mål. Å bruke henger nært sammen med å anvende, forstått som å gjøre bruk av, ta i bruk, for eksempel en metode eller et verktøy*»
>
> Utdanningsdirektoratet 2021

I Utdanningsdirektoratets definisjon står det ikke eksplisitt om elevene skal kunne skrive programkode selv eller ikke. Dermed er kompetansemålet fortsatt ganske åpent for tolkning.

Intervjuundersøkelsen som ble gjort med tre fysikklærere (se kapittel 2.1), viser at kompetansemålet kan tolkes på to vidt forskjellige måter (Skøien, 2022). På den ene siden kan det «*å bruke*» numeriske metoder og programmering tolkes som å trykke på en «kjøreknapp» som kjører et allerede ferdiglaget program, eller så kan det tolkes som at elevene skal skrive all programkode fra bunnen av. Altså synes det å være opp til faglærer hvor mye programmering elevene skal forstå, og hvor mye programkode de skal skrive.

Det er verdt å nevne at læreplanen ikke er mer uklar når det gjelder numeriske metoder og programmering, enn Newtons lover. Forskjellen er at kompetansemålet om numeriske metoder og programmering fortsatt er såpass nytt at det ikke har utviklet seg en felles praksis for hvordan det skal tolkes, slik som med kompetansemålet om Newtons lover. Beslutningen faglæreren tar, om hvor mye programkode elevene er forventet å skrive, får sannsynligvis betydning for hvor mye tid som bør settes av for å imøtekomme kompetansemålet, da det å forstå programmering godt og kunne skrive egen programkode tar betraktelig lengre tid enn å trykke på en «kjøreknapp». Det er også verdt å nevne at måten kompetansemålet om numeriske metoder og programmering tolkes på, sannsynligvis vil gi ringvirkninger i høyere utdanning. Dersom kompetansemålet tolkes dithen at eleven skal kunne skrive programkode på egen hånd, vil trolig pensum i eksempelvis introduksjonsfag til informasjonsteknologi måtte justeres. Det er dermed viktig at høyere utdanning vet hva elever kan når de blir studenter.

Kompetansemålet kan også sies å ha en skjult agenda, nemlig at elevene skal kunne forstå at numeriske metoder og programmering er en integrert del av fysikkfaget. For at elevene skal kunne få til dette bør de tildeles komplekse nok problemer. Med komplekse nok problemer menes problemer der det er hensiktsmessig å bruke både numeriske metoder og programmering.

Hvor mye programkode skal elevene skrive selv?

For at undervisningsopplegget skulle ha en jevn og kontinuerlig progresjon, fikk elevene i utdelt delvis ferdiglaget programkode når de skulle arbeide med programmeringsoppgavene. Tanken var at de på den måten kunne fokusere mer på forståelse og implementering av fysikken, fremfor å rydde opp i syntaksfeil. Likevel viser resultatene fra spørreskjemaet at noen elever syntes det mest utfordrende med undervisningsopplegget var «å fullføre delvis ferdiglaget programkode og å skjønne syntaks» (se Figur 6-11 i kapittel 6.3). Det å tilpasse og videreutvikle

programkode, og å drive feilsøk, beskrives av Sanne et al. (2016, s. 18) som viktige aspekter ved programmering. Funnet er interessant, siden den delvis ferdiglagede programkoden var ment å gjøre det enklere for elevene å løse programmeringsoppgavene. At det å tildele elevene delvis ferdiglaget programkode ikke fikk den effekten jeg hadde tenkt, kan ha sammenheng med informantenes uttalelser fra intervju 1. Her var informantene tydelige på at de gjerne ønsket å forstå programkoden, og helst ville skrive hele programmet selv. Ettersom elevene ønsket å forstå programkoden, er det rimelig å anta at de brukte mye tid og arbeidskapasitet på å forstå den utdelte programkoden. Dermed kan de ha blitt stående fast ved deler av programkoden og kommentarer de ikke forsto. Videre kan den delvis ferdiglagde programkoden også ha ført til at elevene ikke fikk eierforhold til koden, noe som kan ha bidratt til at de ikke var like interessert i å implementere kodelinjene som måtte til for å få programmet til å kjøre. På den annen side kan det å gi elevene delvis ferdiglaget programkode ansees som positivt, da det sannsynligvis gjør at det er mindre tidkrevende å løse oppgaven.

Hvilke og hvor mange løsningsmetoder innen programmering skal elevene lære?

Læreplanen gir ikke retning for hvilke eller hvor mange løsningsmetoder elevene skal lære innen programmering i fysikk 1. Dermed kan det diskuteres om elevene burde holde seg til å bruke lister eller om det er hensiktsmessig at de lærer flere løsningsmetoder innen programmering, som eksempelvis arrays. På den ene siden kan det være enklere for elevene å fokusere på fysikken i problemet, dersom lister anvendes som eneste løsningsmetode. Da vil problemets ukjente aspekter kun være fysikken, og ikke også en ny løsningsstrategi, der fokuset på indeksering er betraktelig større enn med lister. Å legge opp til at elevene i hovedsak arbeider med lister kan dermed bidra til at de bruker mer tid på å forstå fysikken i problemet, og dermed får en dypere fysikkforståelse. Basert på dette, kan det å tvinge arrayer på elevene medføre kognitiv overbelastning, der det at elevene ikke forstår den nye løsningsmetoden kan medføre at de heller ikke forstår fysikken i problemet. Dette taler altså for at elevene burde holde seg til å løse programmeringsproblemer med lister som løsningsmetode. Som resultat gis elevene større mulighet til å bli trygge på lister og hvordan de anvendes. Dette kan bidra til at det i fremtiden blir enklere for elevene å bytte fra å bruke lister til å bruke arrays.

På den annen side kan man argumentere for at elevene bør bli kjent med hvordan fysikere

arbeider, som beskrevet av Malthe-Sørenssen et al. (2015), og derfor behøver å lære å bruke arrays og preallokering som løsningsmetode. Et argument, er at elevene før eller siden vil komme til å arbeide med problemer som krever mange beregninger, slik at det er både hensiktsmessig og nødvendig å bruke arrays fremfor lister. Dermed kan det å «skåne» elevene fra arrays ved å la dem arbeide utelukkende med lister på videregående skole, for å unngå kognitiv overbelastning, sees på som en midlertidig løsning på et problem som uansett vil oppstå.

Hvor mye tid kan brukes på kompetansemålet om numeriske metoder og programmering?

Siden kompetansemålet om programmering og numeriske metoder kun er ett av mange kompetansemål i fysikk 1, er det begrenset hvor mye tid som kan settes av til dette. Basert på dette er det rimelig å anta at det å la elevene kun arbeide med lister, og ikke introdusere dem for arrays, i større grad sikrer at elevene lærer fysikk. Jeg ønsker likevel å presisere at det å lære elevene arrays, og tilhørende løsningsmetode, ikke nødvendigvis trenger å gå på bekostning av undervisningens fysikkinnhold. Dersom man har tilstrekkelig med tid, kan det å introdusere elevene for arrays gi dem bedre innblikk i fysikeres arbeidsmåter (Malthe-Sørenssen et al., 2015), og i at programmets kjøretid kan være av interesse og betydning. På den måten kan elevene få bedre kjennskap til hvor kraftig et programmeringsverktøy er, og hvilke funksjonaliteter det innehar, som omfatter mer enn det grunnleggende de lærer i fysikken. Dette kan også bidra ytterligere til at elevene ser på numeriske metoder og programmering som en integrert del av fysikken, samt et nyttig verktøy for problemløsning.

Når man skal avgjøre om elevene utelukkende skal benytte lister, arrays eller begge, er tid en viktig faktor. I et fullpakket fysikkfag er det trolig tidsaspektet som er mest avgjørende for om elevene introduseres for arrays eller ikke. Dette stemmer godt overens med intervju-undersøkelsen, der fysikklærerne beskrev det som utfordrende å finne tid til å gi elevene tilstrekkelig med programmeringsopplæring i et fullpakket fag med stram tidsramme. Med en stram tidsramme er det enkleste å holde seg til at elevene kun skal arbeide med lister. På den måten reduseres antall nye aspekter elevene presenteres for, og dermed risikoen for kognitiv overbelastning (Rutkowski & Saunders, 2019), noe som kan bidra til at flere forstår fysikkinnholdet i oppgaven. Dersom man dedikerer mer tid

til programmering i fysikken, kan det være en idé å lære elevene både lister og arrays, og deretter sammenligne de to løsningsmetodene. Hvor mye tid man har til rådighet er også avgjørende for om det å programmere i fysikken kan ansees som fysikkundervisning eller ren programmeringsundervisning. I forbindelse med dette kan det diskuteres hvorvidt læreplanen er godt nok gjennomtenkt med tanke på tidsbruk. Dersom det settes av for mye tid til å lære elevene nye løsningsmetoder innen programmering, samt det tekniske som følger med programmering, er det en risiko for at dette går på bekostning av undervisningens fysikkinnhold. Fysikk 1 er tross alt et fysikkfag, der elevene skal lære fysikk og ikke IT, som er et eget valgbart fag.

Hva er et passe antall elever per lærer?

Under økt 2 og 3 gikk både min medstudent Juni og jeg rundt og hjalp elevene mens de arbeidet med oppgaver. Siden det i en vanlig fysikklasse er én faglærer på opptil 30 elever, ble denne studien gjennomført med høyere lærertetthet enn det som er vanlig. Likevel viser resultatene fra intervjuet at elevene opplevde at det tok lang tid fra de rakk opp hånden og spurte om hjelp, til hjelpen faktisk kom. Faren ved at elevene i lengre perioder strevde med oppgavene på egenhånd, mens de venter på hjelp fra lærer, er at oppgaver som i utgangspunktet er passe vanskelig, kan oppleves som altfor vanskelige og kan føre til at elevene mister motivasjonen. Dersom dette er tilfellet, foregår ikke læringen lenger i elevenes proksimale utviklingssone (Vygotsky, 1934/2001), noe som i ytterste konsekvens kan resultere i at elevene opplever kognitiv overbelastning (Rutkowski & Saunders, 2019), og dermed ikke opplever faglig utvikling (Ryan & Deci, 2017). Som presentert i Sitat 6-19 (se kapittel 6.5), sa Hugo i sitt intervju: «*...dere merket jo at det var travelt. Dere rakk jo nesten ikke å puste. Alle ville jo ha hjelp*». Elevenes ønske og behov for hjelp, overskred dermed kapasiteten til de to lærerne som gikk rundt og hjalp. Dette tyder på at selv om lærertettheten var høyere enn i en vanlig fysikktime, opplevde elevene at de ikke fikk tilstrekkelig hjelp.

Resultatene fra elevsamtalene fra timen indikerer også at elevene i større grad utdyper egne utsagn og anvender fysikkforståelsen sin i samtale med lærer. Dette er ferdigheter elevene bør få støtte i å videreutvikle, da dette er en viktig del av fysikkfaget. På bakgrunn av dette kan en høyere lærertetthet sies å være gunstig, siden det i større grad sørger for at elevene får den hjelpen de trenger. Undervisning der elevene programmerer, skiller seg fra

«tradisjonell» fysikkundervisning på den måten at det er en rekke tekniske ting som kan gå galt. I «tradisjonell» undervisning kan man på mange måter si at elevene gjerne får til oppgavene dersom de forstår fysikken. Idet programmering blir en del av undervisningen holder det ikke lenger å forstå fysikken i problemet, men man må også få til programmeringen helt perfekt for å løse oppgaven, slik blant annet Sørby og Angell (2012) beskriver. Elevene har altså et behov for tett oppfølging fra lærer for å imøtekomme kompetansemålet, som Nordby (2019) også poengterer i sin studie. Dette taler for at undervisning som innebærer programmering bør ha høyere lærertetthet enn «tradisjonelle» fysikktimer. Dette kunne eksempelvis løses med delingstimer, slik som på kjemilab. En slik løsning krever imidlertid også midler som kan være utfordrende å få innvilget.

Hvordan forholde seg til at læring er situert?

Siden elevene oppga at de har erfaring med både numeriske metoder og programmering, ville man kunne tro at elevene hadde de nødvendige verktøyene til å løse programmeringsoppgavene de ble tildelt. Studiens funn indikerer at selv om elevene hadde kunnskap om hva numeriske metoder innebærer, og kunne grunnleggende programmering, slet de med å anvende kunnskapen for å løse oppgaver. Altså hadde flere av elevene problemer med å anvende kunnskap lært i én sammenheng i en annen kontekst. Dette underbygger læring som situert (Brown et al., 1989). I lys av dette vil ikke det å gi elevene mer opplæring i programmering nødvendigvis føre til at de blir bedre på å løse programmeringsoppgaver i fysikk. Siden læring er situert, holder det ikke at elevene lærer grunnleggende programmering. For å løse fysikkproblemer trenger elevene i tillegg opplæring i hvordan anvende det de kan om numeriske metoder og programmering. En slik tilnærming kan sies å øke sannsynligheten for at elevene forstår og kan løse programmeringsoppgavene, da de får øvelse i å arbeide som den konseptuelle fysikeren, programmereren og matematikeren på en og samme tid (Sørby & Angell, 2012).

7.4 Elevenes forslag til forbedring av opplegget

I kapittel 6.5 ble elevenes forslag til hvordan undervisningsopplegget kunne forbedres presentert. Hensikten med dette var å gi leseren innblikk i elevenes forslag, samt gi ideer til hvordan opplegget kan forbedres. Det er ikke meningen at alle forslagene elevene kommer

med skal eller bør implementeres i undervisningsopplegget. Likevel er elevenes forslag verdt å lytte til.

Under intervju 1 formidlet informantene at de synes den delvis ferdiglagde programkoden inneholdt «for mange» kommentarer. Disse kommentarene beskrev funksjonaliteten til de ulike kodelinjene. Informantene beskriver at de syntes kommentarene på mange måter var til hjelp, men at de skulle ønske at det var færre av dem. De foreslo også at vi kunne ha gått gjennom eksempler på programmeringsoppgaver som lignet på oppgavene de senere ville få utdelt, slik at det ble lettere å forstå tankegang og fremgangsmåte. Videre poengterte informantene at en slik gjennomgang kunne ha gjort at behovet for beskrivende kommentarer ble mindre, og at den utdelte programkoden dermed ble mer oversiktlig og leservennlig. De poengterte også at en detaljert gjennomgang av lignende eksempler trolig hadde gjort at de forsto programkoden bedre, noe de ønsket. Dette er i tråd med funnene til Nordby (2019), som omhandlet at elevene ønsker å forstå programkoden, og ikke bare kunne bruke den.

Med en grundig gjennomgang av eksempler, hadde elevene vært bedre rustet til å besvare programmeringsoppgavene de ble tildelt på egenhånd. Da kunne trolig noen av elevene ha løst oppgaven uten den delvis ferdiglagde programkode de fikk tildelt, slik både Hugo og Felix ytret at de ønsket. På den annen side var det flere av elevene som syntes det var utfordrende å besvare programmeringsoppgavene, selv med hjelp fra den delvis ferdiglagde koden. Basert på dette kan det være vanskelig å forsvare å gi klassen en oppgave som sannsynligvis hadde vært i heftigste laget for flere av elevene. Å tildele klassen en slik oppgave er også i strid med Vygotsky (1934/2001) sin teori og den «proksimale utviklingssonen» og LIST-prinsippet (Wæge & Nosrati, 2018), og bør av den grunn unngås.

7.5 Anbefalinger

Basert på dilemmaene nevnt i kapittel 7.3, vil jeg komme med noen anbefalinger for undervising om numeriske metoder og programmering. Anbefalingene er ikke bare for fysikklærerne i norske klasserom, men inkluderer også forslag til endringer som kan gjøres i den gjeldende læreplanen i fysikk 1. Jeg foreslår også interesseområder for videre forskning.

Anbefalinger til lærere, lærebok- og læreplanforfattere

Kompetansemålet om numeriske metoder og programmering kan for mange fysikklærere virke uklart. Selv om Utdanningsdirektoratet definerer hva «*å kunne bruke*» innebærer, foreslår jeg at verbets betydning spesifiseres ytterligere i læreplanen. Jeg mener vi bør benytte oss av den gylne muligheten kompetansemålet gir elevene i å bli kjent med fysikkens arbeidsmåter. Mitt forslag er at kompetansemålet bør tolkes dithen at elevene skal skrive *noe* programkode selv, fremfor å få tildelt ferdiglagde programmer. På den måten gjenspeiler fysikkfaget bedre hvordan fysikere faktisk arbeider, slik som Malthe-Sørenssen (2015) beskriver. Elevene vil altså arbeide ut fra delvis ferdiglaget programkode, som de må videreutvikle og tilpasse til det aktuelle problemet. På den ene siden er det viktig å ta i betraktning at elevene opplevde det som utfordrende å arbeide ut fra delvis ferdiglaget programkode. På den annen side muliggjør en slik tilnærming for at elevene kan velge mellom ulike programmer der de får utdelt *deler* av koden, som kan være alt fra noen få kodelinjer til nesten ferdige programmer. Kravet om tilpasset opplæring kan altså sies å oppfylles, da denne tilnærmingen tar hensyn til at ulike elever (Wæge & Nosrati, 2018), som Ruby og Hugo, befinner seg på forskjellig faglig nivå. Undervisningsoppleggets minioppgaver oppfyller også kravet om tilpasset opplæring. Jeg anbefaler derfor å tildele elevene denne type oppgaver. Minioppgavene gir elevene nødvendig repetisjon av grunnleggende programmering, og reduserer dermed risikoen for kognitiv overbelastnig. I tillegg gjør minioppgavene elevene bedre rustet til å anvende programmeringskunnskapen sin i ulike kontekster.

I de presenterte dilemmaene er tid en nøkkelfaktor. Hvilke og hvor mange løsningsmetoder innen programmering elevene skal lære, nødvendig lærertetthet, og hvordan forholde seg til at læring er situert, handler alle om tid. Settes det av for lite tid til kompetansemålets faginnhold, risikerer man at elevene opplever kognitiv overbelastning som følge av at læring er situert og/eller at de presenteres for større mengder informasjon enn det de klarer å prosessere. I et fullpakket fysikkfag er det vanskelig å få frigjort nok tid til at elevene får det faglige utbyttet spesifisert i kompetansemålet. Jeg foreslår derfor at læreplanen i fysikk 1 slankes, slik at det kan brukes lengre tid på hvert kompetansemål. Akkurat hvilke kompetansemål som er aktuelle kandidater til å tas ut av læreplanen, oppfordrer jeg Utdanningsdirektoratet til å se nærmere på og evaluere. I mine øyne bør imidlertid numeriske metoder og programmering forbli en del av fysikkfaget, da det gir et mer autentisk fag (Malthe-Sørenssen et al., 2015) ved å eksponere

elevene for fysikeres arbeidsmåter, og gir elevene mulighet til å bli kjent med dem og gå i dybden. På den måten kan målet om dybdelæring, som presenteres i læreplanen, oppfylles.

Forslag til videre forskning

Numeriske metoder og programmering i videregående skole er fortsatt ganske nytt, og det finnes derfor lite erfaring med å undervise om dette i skolen. Av den grunn trengs det å forskes mer på hva numeriske metoder og programmering kan innebære i fysikk, og hvordan det kan imøtekommes i klasserommet. Det utformede undervisningsopplegget fokuserer på hvordan numeriske metoder og programmering kan inngå som en naturlig del av undervisningen i mekanikk i fysikk 1. Det hadde derfor vært interessant å undersøke numeriske tilnærminger til andre tema som astrofysikk eller elektrisitetslære. Et annet interesseområde kan være å undersøke hvordan LIST-prinsippet, og dermed kravet om tilpasset opplæring, kan ivaretas når nye fysikkelever etterhvert har lært mer programmering. Tidsbruken i undervisningen virker også å være avgjørende for hvordan numerikk og programmering integreres i fysikkfaget, og det hadde dermed vært nyttig å studere dette nøyere. Går det an å effektivisere tiden som brukes på å undervise om numeriske metoder og programmering? For videre forskning kan det også være interessant å se på elevers overgang til realfag i høyere utdanning, og hvilke tilpasninger som må gjøres på universitetsnivå.

7.6 Studiens styrker og svakheter

Det har blitt brukt ulike strategier for å styrke studiens troverdighet. Her har ulike former for triangulering, som data- og metodetriangulering, spilt en sentral rolle. Datatrianguleringen gjorde at jeg kunne sjekke om dataene, innhentet med ulike metoder, fortalte samme historie. På den måten har ulike sett med data hatt innflytelse på min forståelse av andre datasett. Eksempelvis tydet observasjonene på at elevene syntes oppgave 2d var for vanskelig, mens spørreskjemaet indikerte at elevene opplevde oppgaven som noe av det mest interessante med undervisningsopplegget. På tilsvarende måte gjorde metodetrianguleringen det mulig å besvare studiens problemstilling fra både elev- og lærerperspektivet. Trianguleringen kan altså sees på som en av studiens styrker. En annen styrke er at arbeidet har en teoretisk forankring, noe som har vært viktig i utformingen av prinsippene for undervisningsopplegget. Prinsippene ble brukt som holdepunkter i utviklingen av opplegget, slik at det kunne gjennomføres i en reell og

konkret klasseromskontekst. Studiens utprøvingskomponentet innhentet verdifulle erfaringer om hvordan en fysikk 1-klasse responderer på et slikt undervisningsopplegg, noe det fortsatt er behov for på dette forskningsfeltet.

Studien har også begrensninger og svakheter. Som masterstudent er jeg nokså ny i forskerrollen. Dermed har jeg måttet tilegne meg en hel del kunnskap om litteratur på feltet, ulike forskningsmetoder og hva som er god forskningspraksis. Det kan derfor hende at jeg ikke har fått lest litteratur som kunne vært inkludert i denne studien. En annen svakhet ved studien, er at jeg har hatt rollen som både forsker og lærer, noe som kan ha skapt utfordringer i forbindelse med lojalitet. Som følge av dette kan elevene ha vært mer tilbakeholdne med å kritisere opplegget. Dette synes imidlertid ikke å ha vært et stort problem, siden elevene ga uttrykk for ting de ikke forsto. Dette, i kombinasjon med at elevene pratet fritt i intervjuene, kan derfor ansees som å styrke studiens troverdighet.

Det utviklede undervisningsopplegget bærer også preg av de praktiske begrensningene tilknyttet det å «låne» en fysikk 1-klasse fra en faglærer. Læreplanen i fysikk 1 er omfattende, og det var dermed begrenset hvor mange timer faglæreren kunne avse til utprøving av undervisningsopplegget, selv om opplegget søker å imøtekomme et av læreplanenes kompetansemål. Studiens kvalitative natur, og det faktum at det er brukt bekvemmelighetsutvalg, bringer med seg både fordeler og ulemper. En ulempe er at man ikke kan vite om utvalget er representativt. Det er likevel grunn til å tro at utvalget er typisk siden både kjønnsfordeling og ferdighetsnivået i utvalget virker å være ganske lik spredningen i en alminnelig fysikklasse. Studien kan likevel ansees som nyttig, da den viser hva som kan være tilfellet i en fysikk 1-klasse og hvordan både elever og lærer opplever undervisningsopplegget. På den måten kan studien ha overføringsverdi til andre fysikklasser og lignende situasjoner, altså være analytisk generaliserbar (se for eksempel Postholm, 2010). Overføringsverdien handler i hovedsak om at klassen er en nokså typisk fysikklasse. Siden undervisningsopplegget ble gjennomført med en ekstra lærer til stede, kan dette påvirke hvor stor overføringsverdi studiens resultater har. Studien indikerer også hvilket nivå programmeringen kan ligge på, uten at det går på bekostning av timens fysikkinnhold. Arbeidets erfaringer og resultater kan også benyttes som grunnlag for videre studier på hvordan programmering og numeriske metoder kan inngå som en konstruktiv del av fysikk 1.

7.7 Konklusjon

Numeriske metoder og programmering er blitt en viktig del av fysikk som vitenskap, men er fortsatt en relativt ny del av fysikk i skolen, og faginnholdet har lite tradisjon i norsk skole. Dette, i kombinasjon med at ordlyden i formuleringen av kompetansemålet om numeriske metoder og programmering er relativt vag i læreplanen, medfører at hva det faktisk innebærer er ganske åpent for tolkning. I denne studien er kompetansemålet tolket dithen at elevene skal modellere fysiske fenomener, med ikke-konstant akselerasjon, gjennom numeriske metoder, ved å videreutvikle delvis ferdiglagde programmer. Studiens resultater tyder på at numeriske metoder og programmering fint kan inngå som en konstruktiv del av fysikkfaget, men at dette bringer med seg en rekke dilemmaer i forbindelse med gjennomføringen av undervisningsopplegget.

Elevene synes å ha god forståelse for prinsippene bak numeriske metoder, men har større problemer med å oversette dette til programkode, selv om de har lært programmering i både fysikk- og matematikkfaget. Årsaken til dette er delvis fordi læring er situert, og at det ikke er enkelt å overføre kunnskap fra en kontekst til en annen, og/eller at informasjonsmengden elevene presenteres for er såpass stor at det kan føre til kognitiv overbelastning. I dilemmaene i forbindelse med gjennomføringen av undervisningsopplegget er tid et gjennomgående tema. Tidsaspektet virker altså å være en sentral faktor som påvirker hva og hvor mye elevene skal lære av programmering, men også hvordan programmeringen integreres i faget. Ved å sette av nok tid til å behandle kompetansemålets faginnhold, blir fysikkfaget mer representativt for hvordan man arbeider innen fysikk, både som forsker og ingeniør. Dette taler for at det er viktig å modernisere faget, slik at det gjenspeiler dagens fysikk.

REFERANSER

Andersson, T. & Shattuck, J. (2012). Design-Based Research: A Decade of Progress in Education Research?. *Educational Researcher, 41*(1), 16-25. https://doi.org/10.3102/0013189X11428813

Angell, C., Bungum, B., Henriksen, EK., Kolstø, S. D., Persson, J. & Renstrøm, R. (2019). *Fysikkdidaktikk* (2. utgave). Cappelen Damm

Bjørndal, C. R. P. (2011). *Det vurderende øyet: observasjon, vurdering og utvikling i undervisning og veiledning* (2. utg). Gyldendal akademisk.

Bowler, P. J. & Morus, I. R. (2020). *Making Modern Science: A Historical Survey* (2. utg.). The University of Chicago Press.

Braun, V. & Clarke, V. (2022). *Thematic analysis: A practical guide*. SAGE.

Brown, J. S., Collins, A., & Duguid, P. (1989). Situated Cognition and the Culture of Learning. *Educational Researcher, 18*(1), 32–42. https://doi.org/10.3102/0013189X018001032

Dahlquist, G. & Björck, Å. (2003). *Numerical Methods*. Dover

Dolin, J. (2002). *FYSIKFAGET I FORANDRING: Læring og undervisning i fysik i gymnasiet med fokus på dialogiske processer, autenticitet og kompetenceudvikling* [Doktorgradsavhandling]. Roskilde Universitet.

Erickson, T. (2006). Stealing from physics: modeling with mathematical functions in data-rich contexts. *Teaching Mathematics Applications, 25*(1), 23-32. https://doi.org/10.1093/teamat/hri025

Geyer, MA.& Kuske-Janßen, W. (2019). The "Math as Prerequisite" Illusion: Historical Considerations and Implications for Physics Teaching. I Pospiech, G., Michelini, M., Eylon, BS. (Red.), *Mathematics in Physics Education* (s. 37-52). Springer. https://doi.org/10.1007/978-3-030-04627-9_2

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

Haraldsrud, A. D. & Tellefsen, C. W. (2018). Programmering – for fysikkens skyld. *Fysikkens Verden*, (3), 70-75. https://www.mn.uio.no/ccse/om/aktuelt/i-media/programmering-for-fysikkens-skyld.pdf

Iversen, E. L. (2022). *Programmering i naturfag 1: Elevers forståelse, læringsmuligheter og motivasjon i møte med en programmeringsoppgave om halveringstid*[Masteroppgave, Universitetet i Oslo]. DUO vitenarkiv. http://urn.nb.no/URN:NBN:no-96470

Karam, Ricardo., Uhden, Olaf., Höttecke, Dietmar. (2019). Mathematical Representations in Physics Lessons. I Pospiech, G., Michelini, M., Eylon, BS. (Red.), *Mathematics in Physics Education* (s. 75-102). Springer. https://doi.org/10.1007/978-3-030-04627-9_4

Krosnick, J. A. (2018). Questionnaire Design. I Vannette, D & Krosnick, J. (red), The Palgrave Handbook of Survey Research (s. 439 - 455). Springer.

Kunnskapsdepartementet (2017). *Overordnet del – verdier og prinsipper for grunnopplæringen*. Fastsatt som forskrift ved kongelig resolusjon. Læreplanverket for Kunnskapsløftet 2020.

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

Landau, R. (2006), Computational Physics: A Better Model for Physics Education?. Computing in Science & Engineering, 8(5), 22-30. https://doi.org/10.1109/MCSE.2006.85

Lorentz, J., Roşca, D. (Red.). (2020). *Numerical Methods*. MDPI.

Malthe-Sørenssen, A., Hjorth-Jensen, M., Langtangen, H. P. & Mørken, K. (2015). Integrasjon av beregninger i fysikkundervisningen.*Uniped, 38*(4), 303-310. https://www.idunn.no/doi/pdf/10.18261/ISSN1893-8981-2015-04-06

Mork, S. & Erlien, W. (2017) *Språk, tekst og kommunikasjon i naturfag* (2. utg). Universitetsforlag.

National Research Council (NRC). (2012). *A framework for K-12 science education: Practices, crosscutting concepts, and core ideas*. Washington, D.C.: The National Academies Press.

NESH- Den nasjonale forskningsetiske komité for samfunnsvitenskap og humaniora (NESH). (2021, 16. desember). Forskningsetiske retningslinjer for samfunnsvitenskap og humaniora. hentet 20. februar 2023 fra https://www.forskningsetikk.no/retningslinjer/hum-sam/forskningsetiske-retningslinjer-for-samfunnsvitenskap-og-humaniora/

Nordby, S. T. (2019). *Programmering og algoritmisk tekning i fysikkundervisning* [Masteroppgave, Norges teknisk-naturvitenskapelige universitet.] NTNU Open. http://hdl.handle.net/11250/2610766

NTNU. (2019, 23. september). *Retningslinje for behandling av personopplysninger*. Hentet 20. februar 2023 fra https://i.ntnu.no/wiki/-/wiki/Norsk/Retningslinje+for+behandling+av+personopplysninger

Oppenheim (1992). *Questionnaire Design, Interviewing and Attitude Measurement* (ny utg.). Pinter.

Postholm, M. B. (2010). *Kvalitativ metode: en innføring med fokus på fenomenologi, etnografi og kasusstudier* (2. utg.). Universitetsforlaget.

Robson, C. & McCartan, K. (2016) *Real World Research* (4. Utg.). John Wiley & Sons Ltd.

Rutkowski, A-F. & Saunders, S. C. (2019). *Emotional and cognitive overload: the dark side of informational technology*. Routledge.

Ryan, R. M., & Deci, E. L. (2017). *Self-determination theory: Basic psychological needs in motivation, development, and wellness*. Guilford Press

Sanne, A., Berge, O., Bungum, B., Jørgensen, E. C., Kluge, A., Kristensen, T. E., Mørken, K. M., Svorkmo, A. & Voll, L. O. (2016). Teknologi og programmering for alle. En faggjennomgang med forslag til endringer i grunnopplæringen. https://www.udir.no/globalassets/filer/tall-og-forskning/forskningsrapporter/teknologi-og-programmering-for-alle.pdf

Sauer, T. (2012). *Numerical Analysis* (utg. 2). Pearson Education

Skøien, J. X. (2022). *Fysikklæreres refleksjoner rundt undervisning av numeriske metoder*. [Eksamen i RFEL 3100 Forskningsmetoder i matematikk- og realfagsdidaktikk]. Norges teknisk-naturvitenskapelige universitet.

Sørby, S. A. & Angell, C. (2012). Undergraduate student´s challenges with computational modelling in physics. *NorDiNa, 8*(3), 283-296.

Taub, R., Armoni, M., Bagno, E. & Ben-Ari, M. (2015). The effect of computer science on physics learning in a computational science environment. *Computers & Education, 87*, 10-23. https://doi.org/10.1016/j.compedu.2015.03.013

Tjora, A.H. (2018) *Viten skapt: Kvalitativ analyse og teoriutvikling*. Cappelen Damm akademisk.

Utdanningsdirektoratet (2006). *Læreplan i fysikk (FYS1-01)*. Fastsatt som forskrift. Læreplanverket for Kunnskapsløftet 2006. https://www.udir.no/kl06/fys1-01

Utdanningsdirektoratet (2020). *Læreplan i naturfag (NAT01-04)*. Fastsatt som forskrift. Læreplanverket for Kunnskapsløftet 2020. https://www.udir.no/lk20/fys01-02/kompetansemaal-og-vurdering/kv466

Utdanningsdirektoratet (2021). *Læreplan i fysikk (FYS01-02)*. Fastsatt som forskrift. Læreplanverket for kunnskapsløftet 2020. https://www.udir.no/lk20/fys01-02/kompetansemaal-og-vurdering/kv466

Utdanningsdirektoratet (2022, 31. mars). *Tilpasset opplæring*. Hentet 14. november 2022 fra https://www.udir.no/laring-og-trivsel/tilpasset-opplaring/

Utdanningsdirektoratet. (2019, 27. mars). *Algoritmisk tenkning*. Hentet 28. september 2022 fra https://www.udir.no/kvalitet-og-kompetanse/profesjonsfaglig-digital-kompetanse/algoritmisk-tenkning/

Vinnervik, P. & Bungum, B. (2022). Computational thinking as part of compulsory education: How is it represented in Swedish and Norwegian curricula?. *NorDiNa, 18*(3), 384-400.

Vygotsky, L. S. (1934/2001). *Tekning og tale* (Roster, M. T. & Bielenberg, T., Overs.). Gyldendal akademisk. (Opprinnelig utgitt 1934)

Waters, J. B. (2020). *Programmering og dybdelæring i fysikk: En kvalitativ studie av elevers arbeid med programmering i fysikk 1* [Masteroppgave, Universitetet i Oslo]. DUO Vitenarkiv. http://urn.nb.no/URN:NBN:no-82067

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

Wing, J. M. (2006). Computational thinking. *Commun ACM*, 49(3): 33–35. https://dl.acm.org/doi/pdf/10.1145/1118178.1118215

Wæge, K., & Nosrati, M. (2018). *Motivasjon i matematikk*. Oslo: Universitetsforlaget.

VEDLEGG

Vedlegg A: Samtykkeerlæring

For å ivareta anonymitet, er noe informasjon fjernet.

Vil du delta i forskningsprosjektet

«Numeriske metoder og programmering i fysikk 1»?

Dette er et spørsmål til deg om å delta i et forskningsprosjekt. Prosjektet går ut på å undersøke fysikkelevers opplevelse av undervisningsopplegg om programmering, numeriske metoder og modellering. I dette skrivet gir vi deg informasjon om målene for prosjektet og hva deltakelse vil innebære for deg.

Formål

Prosjektet omhandler fysikkundervisning på videregående nivå. Bakgrunnen for prosjektet er de nye kompetansemålene som kom med den nye læreplanen. Programmering fikk da en vesentlig større rolle i undervisningen, særlig i matematikk og fysikk. Formålet er å undersøke fysikkelevers opplevelse av undervisning der programmering og numeriske metoder brukes. Undersøkelsen vil danne grunnlaget for videre analyse og diskusjon til masterprosjektet mitt i fysikkdidaktikk.

Hvem er ansvarlig for forskningsprosjektet?

[fjernet], femteårsstudent ved Institutt For Fysikk, NTNU.

Hvorfor får du spørsmål om å delta?

Du får spørsmål om å delta fordi du har fysikk 1.

Hva innebærer det for deg å delta?

Du godtar å delta i et gruppeintervju og/eller at det gjøres lydopptak av diskusjoner i timen. Intervjuet er ment å vare i omtrent 1 time. Her vil du få spørsmål som om undervisningsoppleggene du har deltatt på: erfaring med programmering, hvordan du har opplevd opplegget, og hvordan dere tenkte da dere løste oppgaver. Jeg tar lydopptak og notater fra intervjuet, som lagres fram til prosjektets slutt, hvorpå datamaterialet vil bli anonymisert.

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 kun oppgaveansvarlig [fjernet] som vil ha tilgang til personopplysninger, via koblingsnøkkel.
- Personopplysningene dine vil bli erstattet med en kode som lagres på en navneliste adskilt fra øvrige data (koblingsnøkkel). Datamaterialet vil lagres ved hjelp av en ekstern lagringsløsning. I en eventuell publikasjon vil ingen utenforstående være i stand til å gjenkjenne deg. Dine personopplysninger vil ikke bli publisert.

Hva skjer med personopplysningene dine når forskningsprosjektet avsluttes?

Prosjektoppgaven er planlagt å avsluttes 01.06.2023. Ved slutten av prosjektet vil dataene bli anonymisert, det vil si at koblingsnøkkelen som gjør det mulig å kombinere dine personopplysninger og dine data vil bli slettet. Lydopptakene vil også bli slettet.

Hva gir oss rett til å behandle personopplysninger om deg?

Vi behandler opplysninger om deg basert på ditt samtykke. På oppdrag fra NTNU 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 personopplysninger som er registrert om deg,
- å få rettet personopplysninger om deg,
- få slettet personopplysninger om deg,
- å utlevert en kopi av dine personopplysninger (dataportabilitet), og
- å sende klage til personvernombudet eller 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:

- Student som gjennomfører prosjektet: [fjernet], tlf. [fjernet], e-post: [fjernet]
- Veileder: [fjernet], tlf. [fjernet], e-post: [fjernet]

- Personvernombud: [fjernet].

Hvis du har spørsmål knyttet til Personverntjenester sin vurdering av prosjektet, kan du ta kontakt med Personverntjenester på epost ([fjernet]) eller på telefon: [fjernet]. Med vennlig hilsen [fjernet] (student)

.....

Samtykkeerklæring

Jeg har mottatt og forstått informasjon om prosjektet programmering og numeriske metoder i fysikk"og har fått anledning til å stille spørsmål. Jeg samtykker til:

☐ å delta i grupperintevju

☐ at det gjøres lydopptak av diskusjoner i timen

.....

(Signert av prosjektdeltaker, dato)

Vedlegg B: Spørreskjema

Takk for at du vil svare på denne spørreundersøkelsen.

I de siste fysikktimene har du arbeidet med programmering og numeriske metoder. Nedenfor er noen spørsmål om din opplevelse av undervisningsopplegget og læringsutbyttet. Svarene dine er anonyme, og kan bli del av en masteroppgave ved NTNU.

1. Hvor utfordrende opplevde du undervisningen om numeriske metoder og programmering?

○1	○2	○3	○4	○5	○6
Svært enkel					Svært utfordrende

2. Hva synes du var mest utfordrende?

3. Hva synes du var mest interessant?

4. Hva synes du undervisningsopplegget har handlet mest om?

	○1	○2	○3	○4	○5	○6
Fysikk	Svært liten grad					Svært stor grad
Matematikk	Svært liten grad					Svært stor grad
Programmering	Svært liten grad					Svært stor grad

5. Synes du man burde lære numeriske metoder i fysikk? Begrunn svaret ditt.

6. Synes du man burde lære programmering i fysikk? Begrunn svaret ditt.

Vedlegg C: Intervjuguide

Takk for at du deltar på dette intervjuet om «Numeriske metoder og programmering i fysikk 1». Jeg ønsker å informere om at alt du sier under intervjuet vil bli anonymisert, og at jeg har taushetsplikt. Uttalelsene dine kan dermed ikke spores tilbake til deg. Intervjuet vil transkriberes, og anonymiserte utdrag fra transkripsjonen kan brukes i masteroppgaven min, og publiseres offentlig. Underskrift av samtykkeerklæring om bruk av lydopptak under intervjuet.

Bakgrunn for undervisningsopplegget

Ett av kompetansemålene i fysikk 1 er å «kunne bruke numeriske metoder og programmering til å modellere og utforske bevegelse i situasjoner der akselerasjonen ikke er konstant»

Elvenes opplevelse av fysikkfaget

- Hvordan synes dere det er å ha fysikk som valgfag?
  - Hva er det morsomste/mest interessante dere har lært i fysikk?
  - Hva har vært det mest utfordrende å lære?
  - Hva tenker dere om pensum og arbeidsmengden?
  - Arbeider dere med fysikk utenom fysikktimene?

Elevenes erfaring med programmering og numeriske metoder

- I kompetansemålet brukes begrepet «numeriske metoder». Har dere noen tanker om hva det er?
  - Kunne dere noe om numeriske metoder fra før?
- Har du noen erfaring med programmering?
- Hvor ofte bruker dere programmering i
  - fysikkfaget/ matematikken?
  - andre fag?

Elevenes opplevelse av undervisningsopplegget

- Hvordan opplevde dere mini-oppgavene som intro/oppfriskning av programmering?
  - Hva likte dere?
  - Hva kunne vært gjort annerledes?
- Hvordan opplevde dere undervisningen(oppgavene om vertikalt kast med luft motstand

og sprettballoppgaven)?
    - Hjalp det å se animajosnen før dere begynte på oppgavene?
    - Var det noe dere syntes var vanskelig/ interessant?
    - Var det noe du følte at du fikk til?
    - Lærte dere noe nytt i fysikk av å bruke programmering og numeriske metoder?
- Kan dere forklare hvordan dere begynte da dere løste oppgaven om
    - vertikalt kast med luftmotstand?
    - sprettball med luftmotstand?
- Hva opplevde dere at undervisningsopplegget handlet mest om? Begrunn svaret.
    - Fysikk/ Matematikk/ Programmering
- Hvordan var det å vurdere svaret deres ut fra grafer over ballens posisjon, fart og akselerasjon?
    - Hjalp det å se animasjonen?
- Hva synes dere om å få delvis ferdig kode?
    - Ville dere heller skrevet koden selv?
- Følte dere at dere fikk nok hjelp? Isåfall fra hvordan og fra hvem.
- Hvordan var det å bruke programemring på kjent fagstoff?
    - Hvordan tror dere det ville vært å lære nytt fagstoff ved hjelp av programmeirng på ligende måte?
        * Blir det for mye å lære på en gang?

Elevenes utbytte av undervisningen

- Lærte dere noe nytt i fysikk gjennom undervisnignsopplegget?
- Lærte dere noe nytt om numeriske metoder og programmering?

Numeriske metoder og programmering i fysikk

- Synes du man burde lære numeriske metoder i fysikk?
- Synes du man behøver å lære programmering i fysikk?
- Hva synes dere om å lære programmering i tillegg til annen fysikk?
    - Føler dere at lærerne deres klarer å hjelpe dere med programmeringen?
- Har det å jobbe med programemring påvirket hvrordan dere ser på fysikkfaget?
- Opplevde du det å programmere som motiverende?
- Hva tenker dere om at numeirske metoder og programmering er en del av læreplanen?

Vedlegg D: Minioppgaver

Minioppgavene er også tilgjengelig for nedlastning på nett fra https://tinyurl.com/laererressurser.

Frisk opp programmeringen

Nedenfor finner du små oppgaver om følgende:

- variabler
- if- og else-setninger
- for-løkker
- arrays
- print-kommandoen
- feilsøking og opprydding av feil i programkode

Oppgavene er ment som en oppfriskning av programmeringen dere arbeidet med i høst. Det er superlov å samarbeide og diskutere oppgavene underveis. For å kjøre kodecellene nedenfor, dvs. få datamaksinen til å utføre instruksene den blir gitt, må du trykke på kodecellen og deretter **shift** etterfulgt av **enter**.

Variabler

En variabel kan endre verdi i løpet av et program. Nedenfor ser du et eksempel der det er laget to variabler kalt **variabel_1** og **variabel_2** som henholdsvis har verdien 11 og 7. Vi kan finne differensen mellom de to variablene ved å lage en ny variabel kalt differanse som regner ut differensen mellom **variabel_1** og **variabel_2**.

```
# EKSEMPEL
# lager variabler og gir dem en verdier
variabel_1 = 11
variabel_2 = 7

# lager en variabel som regner ut differasen
differanse = variabel_1 - variabel_2

# printer differasen
print(differanse)
```

→ 4

Oppgave 1

- Lag to variabler og kall dem **tall_1** og **tall_2**.
- Gi variablene verdier (eksempelvis 2 og 7)
- Lag en ny variabel som heter **sum** som regner ut summen av **tall_1** og **tall_2**.
- Lag en ny variabel som heter **produkt** som regner ut produktet av **tall_1** og **tall_2**.
- Print ut summen og produktet av **tall_1** og **tall_2**.

```
# Lag variablene tall_1 og tall_2 nedenfor

# Lag variablene sum og produkt nedenfor

# print summen av tall_1 og tall_2 nedefor

# print produktet av tall_1 og tall_2 nedefor
```

If-setninger

If-setninger er en måte å instruere datamaskinen til å utføre bestemte kommendoer i ulike tilfeller. I koden nedenfor bes datamaskinen om å sjekke verdien til variabelen **nummer**. Dersom verdien av **nummer** er lik 9, printes det «Hurra!», og hvis verdien av **nummer** er større enn 9, printes det «Synd». Er verdien av **nummer** verken lik 9 eller større enn 9, printes det «Kjipern».

Siden nummer har verdien 90, printer programmet «Synd».Du kan gjerne endre på koden nedenfor og se hva som skjer.

```
# definerer variabelen nummer og gir den veriden 90
nummer = 90

if nummer == 9:       # sjekker om nummer er lik 9
    print("Hurra!")
elif nummer > 9:      # sjekker om nummer er storre enn 9
    print("Synd")
else:
    print("Kjipern") # printer Kjipern dersom nummer verken er 9
                     # eller storre enn 9
```

→ Synd

Oppgave 2

- Lag en variabel kalt **plassering** som beskriver hvilken plass du fikk i en klatrekonkurranse.
- Lag et program som sjekker verdien til **plassering** og printer ut forskjellige beskjeder.
- Dersom **plassering** er lik 1, skal programmet printe «Gratulerer med seieren!».
- Dersom **plassering** er lik 2, skal programmet printe «Du kom på andreplass!».
- Dersom **plassering** er lik 3, skal programmet printe «Wow, bronse».
- Dersom **plassering** er større enn 3, skal programmet printe «Det viktigste er å delta».

Din oppgave er å fullføre koden nedenfor slik at programmet gjør det punktene ovenfor beskriver.

```
# definer variabelen plassering og gi den en verdi mellom 1 og 15

    # sjekk om plassering er lik 1, og print riktig beskjed

    # sjekk om plassering er lik 2, og print riktig beskjed

    # sjekk om plassering er lik 3, og print riktig beskjed

    # print riktig beskjed dersom plassering verken er 1, 2 eller 3
```

For-løkker

Med en for-løkke kjører programmet en kodesnutt et bestemt antall ganger. Vi kan for eksempel lage et program som teller fra 0 til 4 ved hjelp av en for-løkke, slik som vist nedenfor. Her gir vi variabelen **tall** startverdien 0. Kommandoen **for i in range(5)** forteller programmet at kodesnutten totalt skal kjøres 5 ganger, hvor i holder tellingen på hvor mange ganger koden har kjørt. i er 0-indeksert, som betyr at i=0 første gang koden kjøres. For hver gang kodesnutten kjøres, øker verdien til **tall** med 1, og den nye verdien til **tall** printes.

```
# EKSEMPEL
# definerer variabelen tall og gir den startveriden 0
tall = 0

# ber datamaksinen iterere gjennomkoden nedenfor 5 ganger
for i in range(5):
    tall += 1       # legger til 1 til verdien til tall
    print(tall)     # printer den nye verdien til tall
```

→ 1

2

3

4

5

Oppgave 3

Lag et program som printer ut de første 5 tallene i 7-gangen.

- Definer variabelen **starttall**, og gi den verdien 0.
- Lag en for-løkke som for hver gang den kjører øker verdien til **starttall** med 7 og printer ut den nye veriden. Hvor mange ganger må løkken kjøre for å printe ut de 5 første tallene i 7-gangen?

```
# definerer variabelen starttall og gi den startverdien 0

# be datamaksinen iterere gjennom en kodesnutt slik at de 5 forste tallene
# i 7-gangen printes
      #legg til 7 til verdien til starttall
      #print den nye verdien til starttall
```

Arrays

En array er en type liste med et visst antall plasser, der verdier kan lagres. For å lage en array trenger vi å importere et bibliotek kalt numpy. Dette gjøres ved å skrive **import numpy as np**. Når numpy-biblioteket er importert kan vi lage «tomme» arrays med bare 0´er som **elementer**ved hjelp av kommendoen **np.zeros(x)**. Hvis vi vil ha en array med fem elementer der verdien 0 er lagret, skriver vi 5 istedenfor x.

I eksempelet nedenfor legges det inn verdier i en array. Her er det lagd en array, kalt **mine_tall** med fem **elementer** med verdi 0. Deretter legger vi inn verdier på de ulike plassene som har hvert sitt **element**, før vi printer den fylte arrayen. **Elementene** står på hver sin plass i arrayen, og er 0-indeksert, som betyr at det første **elementet** står på plass 0. Når vi vil lagre en ny verdi, eksempelvis 4, på plass 0 i array, bruker vi kommandoen **mine_tall[0]=4**. Da erstattes det som tidligere sto på plass 0 med 4, slik at elementet på plass 0 nå er 4.

```
# EKSEMPEL
#importerer numpy-biblioteket
```

```python
import numpy as np

# lager arrayen mine_tall som har fem elementer med verdi 0
mine_tall = np.zeros(5)

mine_tall[0] = 5 # erstatter verdien med indeks 0 i arrayen med verdien 5
mine_tall[1] = 4 # erstatter verdien med indeks 1 i arrayen med verdien 4
mine_tall[2] = 3 # erstatter verdien med indeks 2 i arrayen med verdien 3
mine_tall[3] = 2 # erstatter verdien med indeks 3 i arrayen med verdien 2
mine_tall[4] = 1 # erstatter verdien med indeks 4 i arrayen med verdien 1

print(mine_tall) # printer arrayen hvor de nye verdiene er lagret
```

→ [5. 4. 3. 2. 1.]

Oppgave 4

Lag en array med 10 elementer som alle har verdi 0, og kall den **en_gangen** . Fyll arrayen med nye verdier slik at elementene er de 10 første tallene i 1-gangen.

```python
# importerer numpy-biblioteket
import numpy as np

# lag arrayen en_gangen som har 10 elementer med verdi 0

        # endre verdien med indeks 0 til 1
        # endre verdien med indeks 1 til 2
        # endre verdien med indeks 2 til 3
        # endre verdien med indeks 3 til 4
        # endre verdien med indeks 4 til 5
        # endre verdien med indeks 5 til 6
        # endre verdien med indeks 6 til 7
        # endre verdien med indeks 7 til 8
        # endre verdien med indeks 8 til 9
        # endre verdien med indeks 9 til 10

# print arrayen en_gangen
```

Vi kan også endre elementene i en array mer effektivt ved hjelp av en for-løkke. Dermed kan samme array som i oppgaven over, lages slik:

```
# importerer numpy-biblioteket
import numpy as np

# lager arrayen en_gangen_igjen som har 10 elementer med verdi 0
en_gangen_igjen = np.zeros(10)

# lopper gjennom like mange ganger som det er elementer i arrayen
for i in range(len(en_gangen_igjen)):
    en_gangen_igjen[i] = i+1

print(en_gangen_igjen) # printer den nye arrayen
```

$\rightarrow [1.\,2.\,3.\,4.\,5.\,6.\,7.\,8.\,9.\,10.]$

Oppgave 5

Endre koden nedenfor slik at arrayen **fem_tall** ser ut som arrayen $[3.\,4.\,5.\,6.\,7.]$.

Hint: hva må stå etter likhetstegnet for at første element får verdien 3?

```
# importerer numpy-biblioteket
import numpy as np
# lag arrayen fem_tall som har 5 elementer med verdi 0
fem_tall = np.zeros(5)

for i in range(len(fem_tall)):
    fem_tall[i] = 0             # endre koden etter likhetstegnet


print(fem_tall) # printer den nye arrayen
```

$\rightarrow [0.\,0.\,0.\,0.\,0.]$

Feilsøking og opprydding av feil i programkode

Alle som programmerer gjør feil, og vi skal derfor trene på å forstå feilmeldinger og hvordan man rydder opp i feil.

NameError

Nedenfor ser du en vanlig feilmelding kalt **NameError**. Her sier datamaskinen ifra om at

variabelen **gullmedaljer** som jeg prøver å printe ikke finnes (dvs. at variabelen er ikke derfinert).

```
1 print(gullmedaljer)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-497a44bd9704> in <cell line: 1>()
----> 1 print(gullmedaljer)

NameError: name 'gullmedaljer' is not defined

SEARCH STACK OVERFLOW
```

→

Feilmeldingen løses ved å definere variablen før vi printer den, som vist nedenfor

```
1 gullmedaljer = 7
2 print(gullmedaljer)
```

→ 7

Oppgave 6

Endre koden under slik at feilmeldingen **NameError** løses.

```
1 pokaler = 8              # definerer variabelen pokaler
2 solvmedaljer = 3         # definerer variabelen solvmedaljer
3
4 print(pokaler)           # printer pokaler
5 print(solvmedaljer)      # printer solvmedaljer
6 print(bronsemedaljer)    # printer bronsemedaljer
```

→ 8

3

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-3-130a29565f7f> in <cell line: 7>()
      5 print(pokaler)           # printer pokaler
      6 print(solvmedaljer)      # printer solvmedaljer
----> 7 print(bronsemedaljer)    # printer bronsemedaljer

NameError: name 'bronsemedaljer' is not defined

SEARCH STACK OVERFLOW
```

IndexError

Vi får feilmeldingen **IndexError** når vi prøver å bruke et plassnummer, kalt **index** som ikke finnes. Nedenfor er et eksempel på dette.

```
import numpy as np

tre_tall = np.zeros(3) # lager en array med tre elementer med plassnummer 0, 1 og 2

for i in range(4): # kjører kodesnutten 3 ganger
    tre_tall[i]=1  # gir elementet på plassnummer 0, 1, 2 verdien 1

print(tre_tall) #printer den nye arrayen
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-4-e1cd086ab98b> in <cell line: 5>()
      4
      5 for i in range(4): # kjører kodesnutten 3 ganger
----> 6     tre_tall[i]=1  # gir elementet på plassnummer 0, 1, 2 verdien 1
      7
      8 print(tre_tall) #printer den nye arrayen

IndexError: index 3 is out of bounds for axis 0 with size 3

SEARCH STACK OVERFLOW
```

Problemet her er at arrayen **tre_tall** kun har tre elementer, og vi prøver å endre verdien på et fjerde, ikke-eksisterende element. I kodefeltet nedenfor er feilmeldingen løst ved å sørge for at for-løkka kjører **like mange ganger** som antall elementer i arrayen **tre_tall**.

```
import numpy as np

tre_tall = np.zeros(3) # lager en array med tre elementer
                       # med plassnummer 0, 1 og 2

for i in range(3): # itererer gjennom kodesnutten 3 ganger
    tre_tall[i]=1  # gir elementene med indeks 0, 1, 2 verdien 1

print(tre_tall) #printer den nye arrayen
```

→ [1. 1. 1.]

Oppgave 7

Endre koden under slik at feilmeldingen **IndexError** løses.

```
import numpy as np

ni_tall = np.zeros(9) # lager en array med 9 elementer
```

```
                        # med plassnummer 0 til 8

for i in range(12):     # itererer gjennom kodesnutten 12 ganger
    ni_tall[i]=9        # gir elementeene med indeks 0 til 11 verdien 9

print(ni_tall)          # printer den nye arrayen
```

Grubleoppgave

Endre koden under slik at feilmeldingen **IndexError** løses.

```
import numpy as np

fire_tall = np.zeros(4) # lager en array med ni elementer
                        # med plassnummer 0 til 8

for i in range(4):      # itererer gjennom kodesnutten 4 ganger
    fire_tall[i+1]= 1   # gir neste element, med plassnummer i+1, verdien 1

print(fire_tall)        #printer den nye arrayen
```

Vedlegg E: Oppgaveark 1 «Bil med konstant akselerasjon»

(a) Gå inn på linken: **https://tinyurl.com/Biloppgave**. Trykk på «fil», «Lagre en kopi i Drive» og gi filen et oversiktlig navn. Utvid programkoden slik at du får frem bilens fart og akselerasjon som hver sin graf.
Husk å gi figurene titler, og navn på aksene. Dere kan også endre fargene på grafene;)

(b) Bruk bilens fartsgraf til å vise at bilens akselerasjon er 1,7 m/s2. Hvordan gikk du frem?

(c) Hvordan vil du beskrive forskjellen mellom en analytisk og en numerisk løsning? Gi gjerne er eksempel.

(d) Forklar Eulers metode med egne ord.

Vedlegg F: Oppgaveark 2 «Vertikalt kast med luftmotstand»

En ball med masse $0,2$ kg kastes rett opp fra bakkenivå med startfart $v_0 = 10$ m/s. Luftmotstanden som virker på ballen er gitt ved

$$L = kv^2 \text{ der } k = 0,01$$

(a) Virker luftmostanden med eller mot ballens bevegelsesretning?

(b) Hvilke krefter som virker på ballen? Tegn en skisse over hvordan kreftene virker på ballen når den er på vei opp, og en annen skisse av hvordan kreftene virker på ballen når den er på vei ned.
Sett positiv retning oppover, og tydeliggjør fortegnet på farten til ballen.

(c) Sett opp Newtons 2. lov for ballens opp- og nedtur.

(d) Gå inn på linken: **https://tinyurl.com/Vertikalt-kast**. Trykk på «fil», «Lagre en kopi i Drive» og gi filen et passende navn. Utvid koden «vertikalt kast fra bakkenivå» slik at du får frem høyde over bakken, fart og akselerasjon som hver sin graf. Det er skrevet en kommentar i blokkbokstaver der dere skal utvide koden.

(e) Bestem ballens maksimale høyde. Beskriv hvordan du kom fram til svaret.

(f) Når treffer ballen bakken? Beskriv hvordan du kom fram til svaret.

(g) Nå skal vi endre lengden på tidssteget dt.

(i) Hva er ballens maksimale høyde når $dt = 0,01$?

(ii) Hva er ballens maksimale høyde når $dt = 0,1$?

(iii) Hva er ballens maksimale høyde når $dt = 2$?

(iv) Hva er ballens maksimale høyde når $dt = 0,4$?

(h) Beskriv hva som skjer med posisjonsgrafen når vi endrer lengden på tidssteget dt.

(i) Kan dere forklare hvorfor det å endre verdien til tidssteget dt endrer ballens beregnede maksimale høyde?

(j) Hvilken verdi av dt ville du brukt for å beregne ballens maksimale høyde? Begrunn svaret ditt.

Vedlegg G: Grubleoppgave «Sprettball med luftmotstand»

Når vi slipper en sprettball rett ned, spretter ballen lavere og lavere, før den ligger helt stille på bakken. Dette er fordi det virker luftmotstand på ballen. Luftmotstanden er gitt ved $L = kv^2$, og virker i motsatt retning av ballens bevegelse. Det også et energitap i det øyeblikket ballen treffer bakken. Ballens energitap kan forenkles til at farten ballen treffer bakken med, skifter fortegn og reduseres med faktoren e. Dvs. at dersom ballen treffer bakken med en fart v_0 , vil ballen ha en fart $v_1 = -v_0 \cdot e$ rett etter sprettet, på vei opp igjen.

Filen «Sprettball med luftmotstand» er en påbegynt programkode som skal plotte posisjonsgrafen til en sprettball som spretter. Her plottes tiden i sekunder på x-aksen og høyden over bakken på y-aksen.

(a) Gå inn på linken: **https://tinyurl.com/Sprettball**. Trykk på «fil», «Lagre en kopi i Drive» og gi filen et passende navn.

Lag en for-løkke som beregner ballens posisjon,fart og akselerasjon med Eulers metode, når det tas hensyn til luftmotstanden og energitapet i kontakt med bakken.

*NB: Det er skrevet en kommentar i **BLOKKBOKSTAVER** der dere skal utvide programkoden.*

(b) Endre koeffisienten for ballens sprettenhet til $e = 1,2$. Hva skjer med posisjonsgrafen til ballen? Forklar bevegelsen til ballen, og drøft om virkelighetens sprettballer oppfører seg sånn.

(c) Endre steglengen til $dt = 0,04$. Hva skjer med posisjonsgrafen? Bruk Eulers metode til å forklare hvorfor dette skjer.

Vedlegg H: Animasjoner

Nedenfor følger programkoden til animasjonene elevene ble vist i timen, som er laget i samarbeid med Trond Morten Thorseth, spesifikt for denne masteroppgaven. Programkoden er også tilgjengelig for nedlastning på nett fra https://tinyurl.com/laererressurser.

«Vertikalt kast med luftmotstand»

```
import numpy as np
import bpy

# definerer fysiske storrelser og konstanter
g = 9.81      # m/s^2
m = 0.2       # kg
k = 0.01      # koeffisienten for luftmotstand

# tidsintervaller
n = 1000          #antall tidspunkt vi beregner tilbakelagt strekning
dt = 0.01         # tidssteg, sekunder mellom hvert tidspunkt vi evaluerer

# tomme arrayer som fylles med verdier senere
h = np.zeros(n)       # array for strekningsdata
v = np.zeros(n)       # array for fartsdata
t = np.zeros(n)       # array for tidsdata
a = np.zeros(n)       # array for akselerasjonsdata

#initialbetingelser
v[0] = 10

#Bestemmer kreftene og regner ut akselerasjonen med N2 for hvert tidssteg
#Antar konstant fart og akselerasjon i tidssteget,
#og bruker i beregningen av ny fart og ny posisjon
#Her har vi valgt negativ retning nedover

for i in range(n-1):
    # Ballen beveger seg oppover
    if v[i] > 0:
        G = m * g
        L = k * v[i]**2
```

```
        F_sum = -G - L
        a[i] = F_sum/m
        v[i+1] = v[i] + a[i] * dt
        t[i+1] = t[i] + dt
        h[i+1] = h[i] + v[i]*dt

    # Ballen beveger seg nedover
    elif v[i] < 0:
        G = m * g
        L = k * v[i]**2
        F_sum = -G + L
        a[i] = F_sum/m
        v[i+1] = v[i] + a[i] * dt
        t[i+1] = t[i] + dt
        h[i+1] = h[i] + v[i]*dt

    # Hopper ut av for-lokken naar ballen nesten ikke spretter lenger
    elif h[i] < 0.01:
        break
************************************************************
# Bestemmer tidspunkt i animasjon
tid=np.arange(0,10,1/24)
# Interpolerer slik at det blir animert med riktig tid.
hoyde=np.interp(tid,t,h)

vx=0.0
#orange_material=bpy.data.materials['Orange']  # referanse til materiale.
ball=bpy.data.objects['Ball'] # henter ball som objekt
i=0
#for ti in t :
for ti in tid:
    bpy.context.scene.frame_set(i) # set frame til i
    # ball.location.x=0, og plasser klossen i punktet null vertikal fart
    ball.location.x=ti*vx # plasser klossen i punktet
    ball.location.z=hoyde[i] # plasser klossen i punktet
    #ball.location.z=h[i] # plasser klossen i punktet
    ball.keyframe_insert(data_path="location",index=-1) # lager en keyframe
    i=i+1

```

```
# En liten rutine som lager en liten prikk.
def Point_3d(p1,r,mat,name='Point_3d'):
    #create an uv sphere.
    rc=16
    seg=8
    bpy.ops.mesh.primitive_uv_sphere_add(segments=seg, ring_count=rc,
    radius=r, enter_editmode=False, align='WORLD',
    location=(p1[0], p1[1],p1[2]), scale=(1, 1, 1))
    pt_obj = bpy.context.object
    me = pt_obj.data
    me.materials.append(mat)
    pt_obj.name = name
    return pt_obj

# Legger inn prikker i stien.
i=0
for t in t_data:
     # plasser klossen i punktet
     o=Point_3d((t*vx,0,h[i]),0.01,orange_material,'pt')
    if i>2:
        o.hide_render=True
        o.hide_viewport=True
        o.keyframe_insert(data_path="hide_render",frame=i)
        o.keyframe_insert(data_path="hide_viewport",frame=i)
        o.hide_render=False
        o.hide_viewport=False
        o.keyframe_insert(data_path="hide_render",frame=i+1)
        o.keyframe_insert(data_path="hide_viewport",frame=i+1)
    i=i+1
```

Programkoden er laget i samarbeid med Trond Morten Thorseth spesifikt for denne masteroppgaven.

«Sprettball med luftmotstand»

```
1
2  import numpy as np
3  import bpy
4
5  # tidsintervaller
6  n = 1000          #antall tidspunkt vi beregner tilbakelagt strekning
7  dt = 0.01         # tidssteg, sekunder mellom hvert tidspunkt vi evaluerer
8
9  # tomme arrayer som fylles med verdier senere
10 h = np.zeros(n)       # array for strekningsdata
11 v = np.zeros(n)       # array for fartsdata
12 a = np.zeros(n)       # array for akselerasjonsdata
13 t = np.zeros(n)       # array for tidsdata
14
15 # Fysiske storrelser og konstanter
16 m = 10     # kg
17 g = 9.81   # m/s^2
18 k = 0.01   # koeffisienten for luftmotstand i N/(m/s)^2
19 e = 0.8    # koeffisient for ballens sprettenhet.
20           # 0 tilsvarer ingen sprett,
21           # mens 1 betyr at ballen ikke mister energi i sprettet
22
23 # Startverdier
24 v[0] = 0 # m/s
25 h[0] = 4 # m
26
27 # Beregner sprettballens neste posisjon,
28 # fart og akselerasjon fra ballens forrige
29 # posisjon, fart og akselerasjon
30 for i in range(n-1):
31     G = m*g
32     L = k*v[i]*abs(v[i])
33     F_sum = G + L
34     a[i] = F_sum/m
35     h[i+1] = h[i] - v[i] * dt
36     v[i+1] = v[i] + a[i] * dt
37     t[i+1] = t[i] + dt
```

```
38      if h[i+1] <= 0.01 and v[i+1]>=0: # Evaluerer ballen i sprettoyeblikket.
39          v[i+1] = -v[i+1] * e         # Skifter fortegn paa farten
40          # og reduseres med en faktor e
41          k = k*(-1)                   # Endrer retningen paa luftmotstanden
42  # ************************************************************
43  # Bestemmer tidspunkt i animasjon
44  tid=np.arange(0,10,1/24)
45  # Interpolerer slik at det blir animert med riktig tid.
46  hoyde=np.interp(tid,t,h)
47
48  vx=0.0
49  orange_material=bpy.data.materials['Orange']  # referanse til materiale.
50  ball=bpy.data.objects['Basketball'] # henter ball som objekt
51  i=0
52  for t in tid:
53  #for t in t_data:
54      bpy.context.scene.frame_set(i) # set frame til i
55      # ball.location.x=0 , og plasser klossen i punktet null vertikal fart
56      ball.location.x=t*vx           # plasser klossen i punktet
57      ball.location.z=hoyde[i]       # plasser klossen i punktet
58      #ball.location.z=h[i]          # plasser klossen i punktet
59      ball.keyframe_insert(data_path="location",index=-1) # lager en keyframe
60      i=i+1
61
62  # En liten rutine som lager en liten prikk.
63  def Point_3d(p1,r,mat,name='Point_3d'):
64      #create an uv sphere.
65      rc=16
66      seg=8
67      bpy.ops.mesh.primitive_uv_sphere_add(segments=seg, ring_count=rc,
68      radius=r, enter_editmode=False, align='WORLD',
69      location=(p1[0], p1[1],p1[2]), scale=(1, 1, 1))
70      pt_obj = bpy.context.object
71      me = pt_obj.data
72      me.materials.append(mat)
73      pt_obj.name = name
74      return pt_obj
75
76  # Legger inn prikker i stien.
```

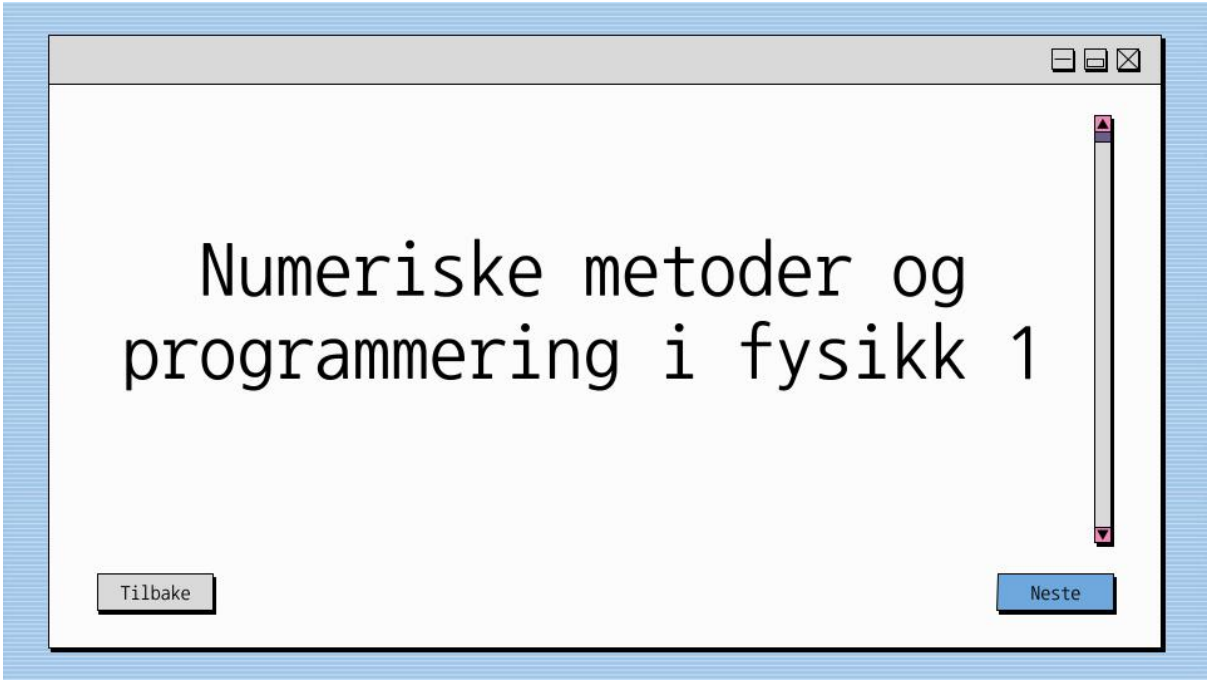
```
i=0
for t in tid:
    # plasser klossen i punktet
    o=Point_3d((t*vx,0,hoyde[i]),0.01,orange_material,'pt')

    if i>2:
        o.hide_render=True
        o.hide_viewport=True
        o.keyframe_insert(data_path="hide_render",frame=i)
        o.keyframe_insert(data_path="hide_viewport",frame=i)
        o.hide_render=False
        o.hide_viewport=False
        o.keyframe_insert(data_path="hide_render",frame=i+1)
        o.keyframe_insert(data_path="hide_viewport",frame=i+1)
    i=i+1
```

Programkoden er laget i samarbeid med Trond Morten Thorseth spesifikt for denne masteroppgaven.

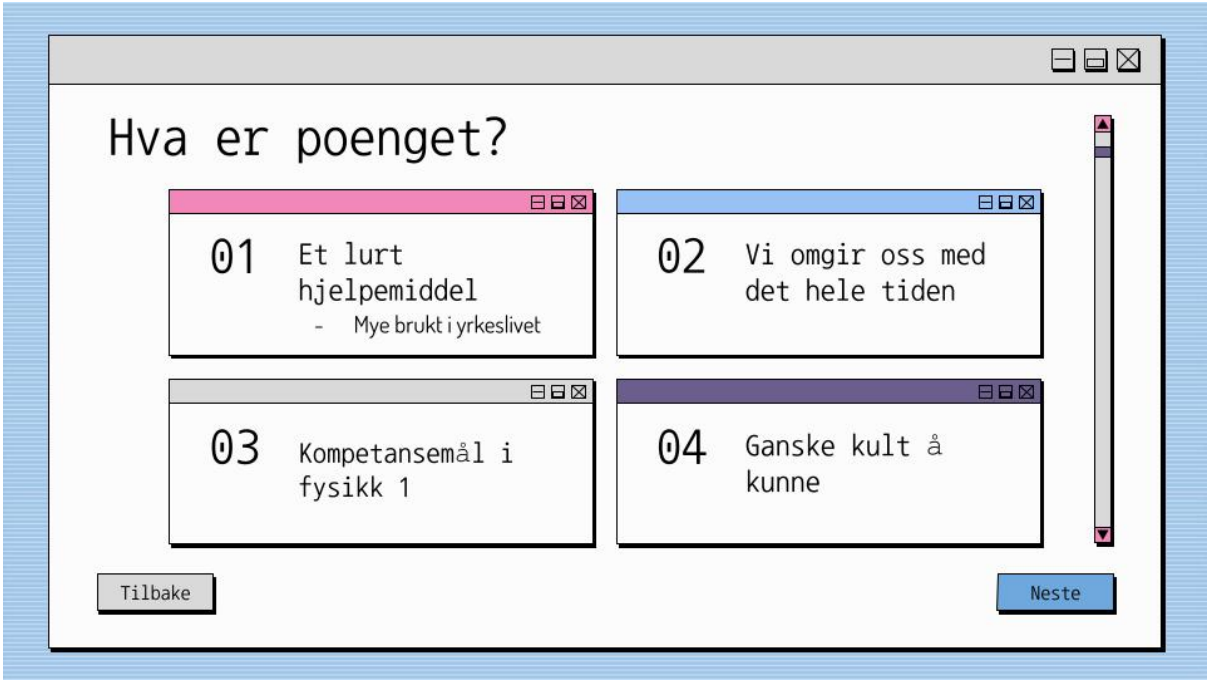
Vedlegg I: Lysbildepresentasjon

Lysbildepresentasjonen er også tilgjengelig for nedlastning på nett fra https://tinyurl.com/laererressurser.

Lysbilde I-1

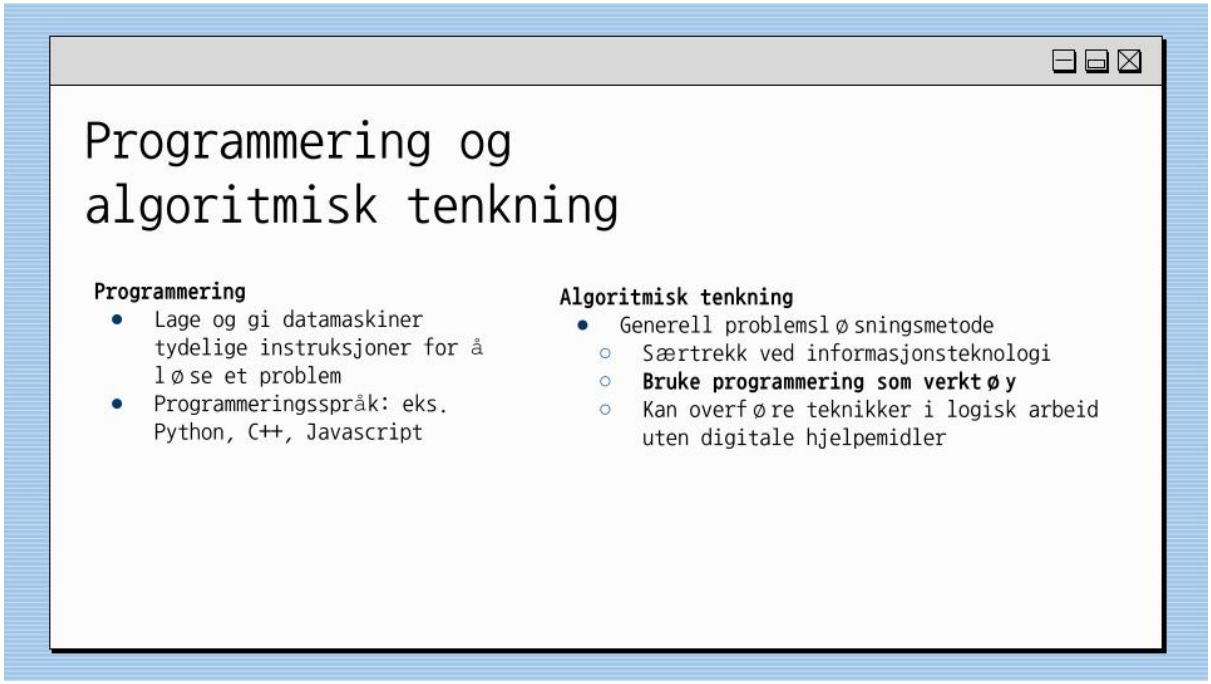


Lysbilde I-2



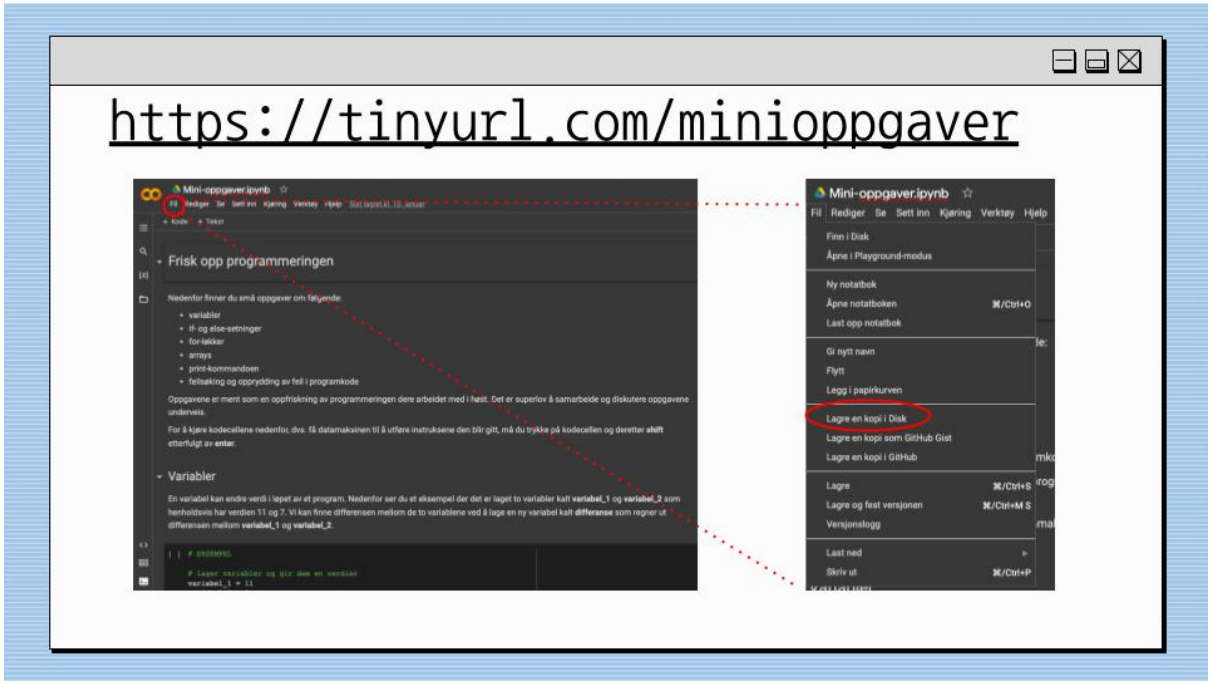
Lysbilde I-3

Lysbilde I-4



Lysbilde I-5

Lysbilde I-6



Lysbilde I-7

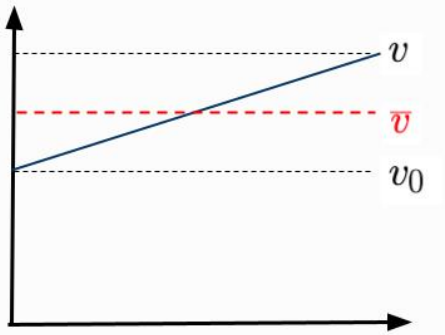
Repetisjon av bevegelseslikningene

En måte å beskrive rettlinjet bevegelse med konstant akselerasjon.

- Boks på friksjonsfritt skråplan
- Vertikalt kast uten luftmotstand

$$v = v_0 + at$$
$$s = \frac{1}{2}(v_0 + v)t$$
$$s = v_0 t + \frac{1}{2}at^2$$
$$v^2 = v_0^2 + 2as$$

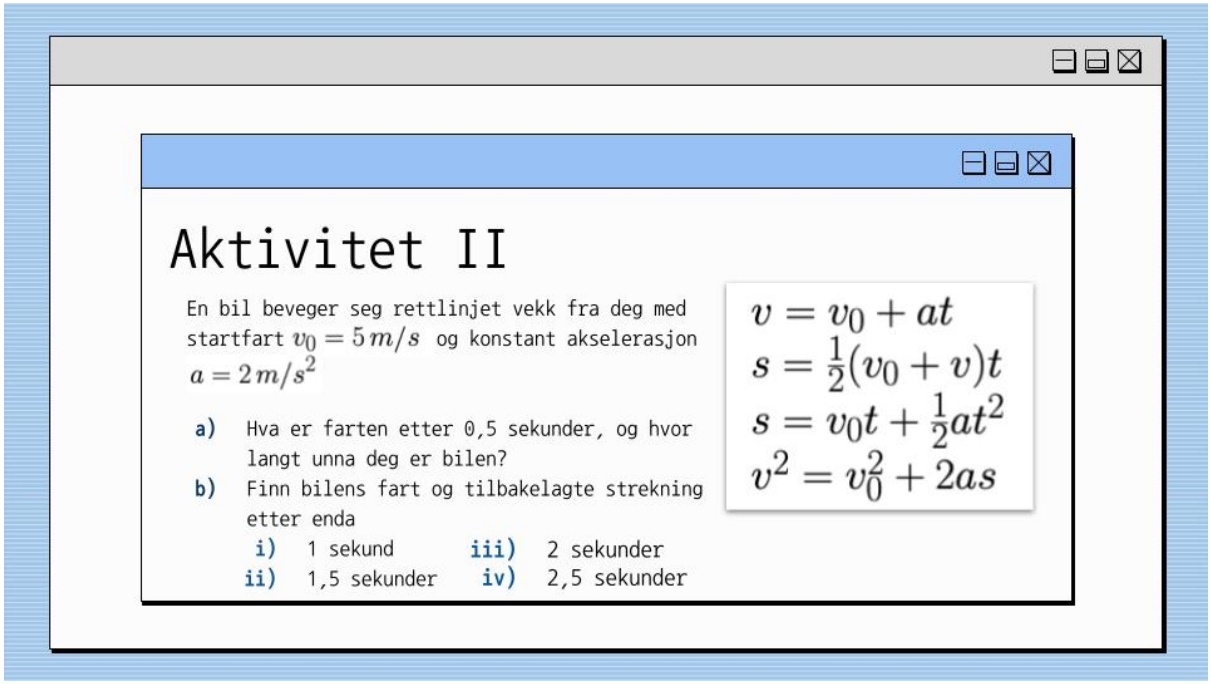
Lysbilde I-8



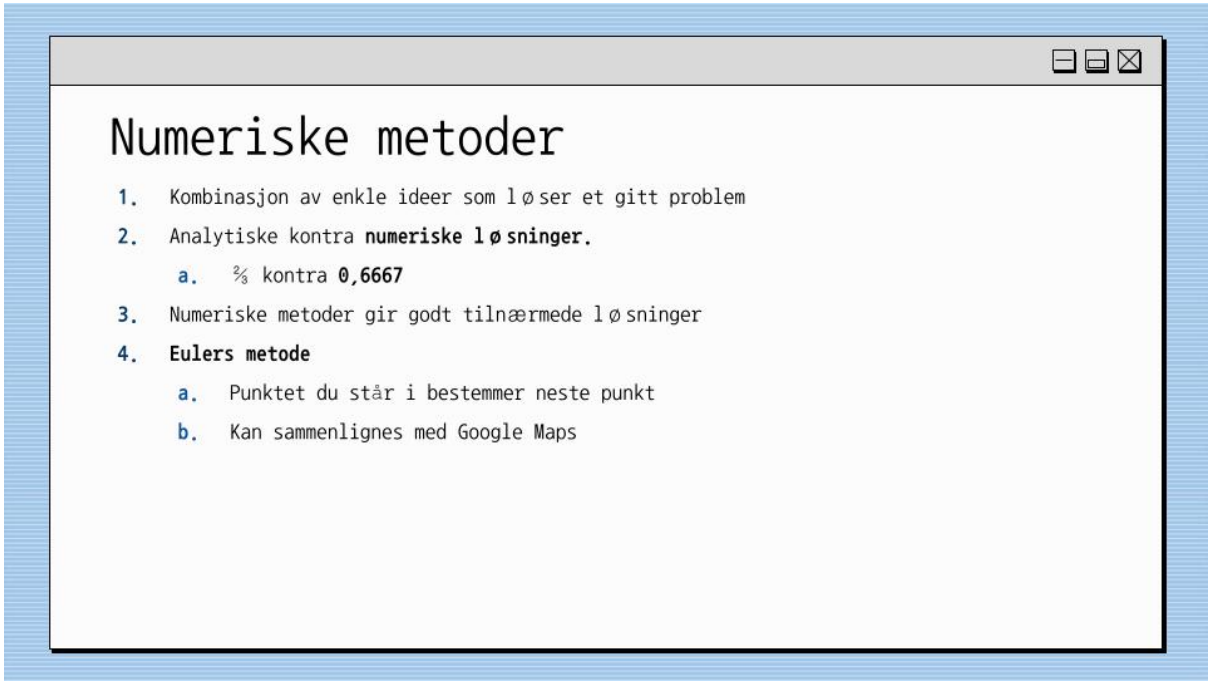
$$s = \frac{1}{2}(v_0 + v)t$$

$$\overline{v} = \frac{v+v_0}{2} = \frac{1}{2}(v + v_0)$$

Lysbilde I-9



Lysbilde I-10



Lysbilde I-11

Oppgave 1: Bil med konstant akselerasjon

a) Gå inn på linken: **https://tinyurl.com/Biloppgave**. Trykk på ≪fil≫, ≪Lagre en kopi i Drive≫ og gi filen et passende navn.
Utvid programkoden slik at bilens fart og akselerasjon fremstilles i hver sin graf.

b) Bruk bilens fartsgraf til å vise at bilens akselerasjon er 1,7 m/s^2.
Hvordan gikk du frem?

c) Hvordan vil du beskrive forskjellen mellom en analytisk og en numerisk løsning? Gi gjerne er eksempel.

d) Forklar Eulers metode med egne ord.

Lysbilde I-12

Oppgave 2: Vertikalt kast med luftmotstand

En ball med masse 0,2 kg kastes rett opp fra bakkenivå med startfart $v_0 = 10$ m/s. Luftmotstanden som virker på ballen er gitt ved

$$L = kv^2 \text{ der } k = 0,01$$

Hvordan ser ballens bevegelse ut?

- Animasjon i Blender

Lysbilde I-13

Oppgave 2: Vertikalt kast med luftmotstand

En ball med masse 0,2 kg kastes rett opp fra bakkenivå med startfart $v_0 = 10$ m/s. Luftmotstanden som virker på ballen er gitt ved

$$L = kv^2 \text{ der } k = 0,01$$

a) Virker luftmotstanden med eller mot ballens bevegelsesretning?

b) Hvilke krefter virker på ballen? Tegn en skisse over kreftene som virker på ballen når den er på vei opp, og en annen skisse av hvordan kreftene virker på ballen når den er på vei ned.

Sett positiv retning oppover, og tydeliggjør fortegnet på farten.

c) Sett opp Newtons 2. lov for ballen når den er på vei oppover og nedover.

Lysbilde I-14

Oppgave 2: Vertikalt kast med luftmotstand

En ball med masse 0,2 kg kastes rett opp fra bakkenivå med startfart $v_0 = 10$ m/s. Luftmotstanden som virker på ballen er gitt ved

$$L = kv^2 \text{ der } k = 0,01$$

d) Gå inn på linken: **https://tinyurl.com/Vertikalt-kast**. Trykk på ≪fil≫, ≪Lagre en kopi i Drive≫ og gi filen et passende navn.

Utvid koden slik at du får frem høyde over bakken, fart og akselerasjon som hver sin graf.

Det er skrevet en kommentar i blokkbokstaver der dere skal utvide koden.

Lysbilde I-15

Oppgave 2: Vertikalt kast med luftmotstand

En ball med masse 0,2 kg kastes rett opp fra bakkenivå med startfart $v_0 = 10$ m/s. Luftmotstanden som virker på ballen er gitt ved

$$L = kv^2 \text{ der } k = 0,01$$

e) Skriv programkode som finner ballens maksimale høyde. Beskriv hvordan du tenker når du løser oppgaven.

f) Når treffer ballen bakken? Beskriv hvordan du kom fram til svaret.

g) Nå skal vi endre lengden på tidssteget dt.

i) Hva er ballens maksimale høyde når dt=0,01?

ii) Hva er ballens maksimale høyde når dt=0,1?

iii) Hva er ballens maksimale høyde når dt=2,0?

iv) Hva er ballens maksimale høyde når dt=0,4?

Lysbilde I-16

Oppgave 2: Vertikalt kast med luftmotstand

En ball med masse 0,2 kg kastes rett opp fra bakkenivå med startfart $v_0 = 10$ m/s. Luftmotstanden som virker på ballen er gitt ved

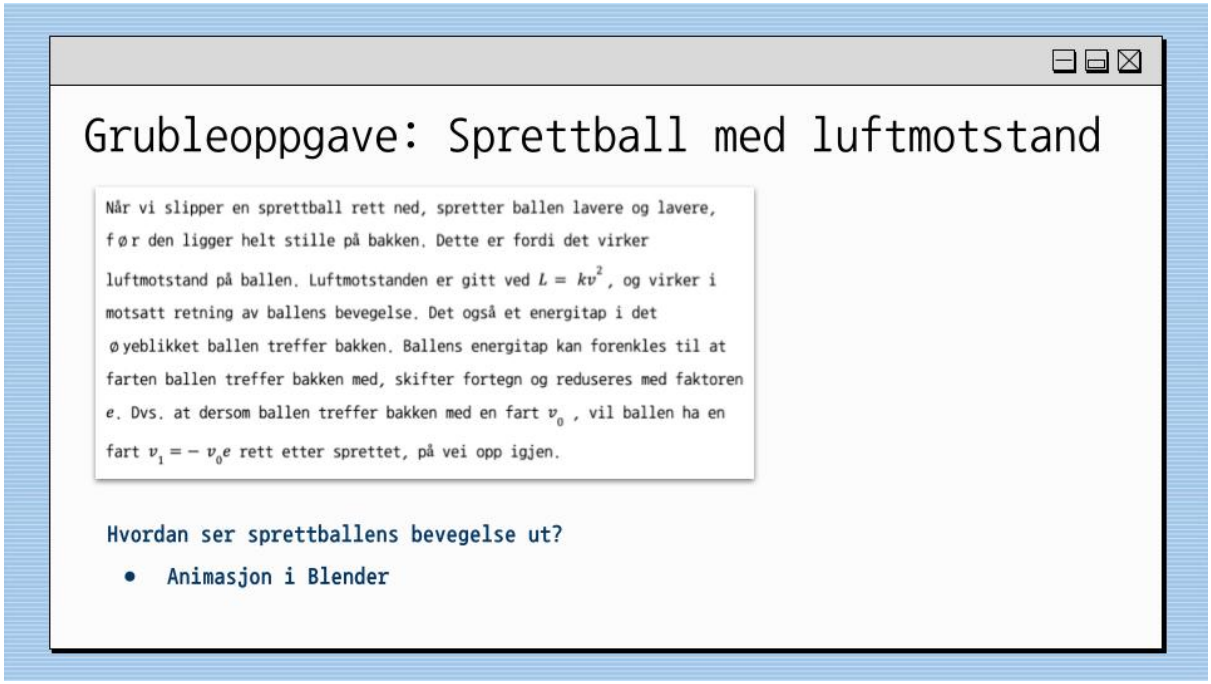
$$L = kv^2 \text{ der } k = 0,01$$

h) Beskriv hva som skjer med posisjonsgrafen når vi endrer lengden på tidssteget dt.

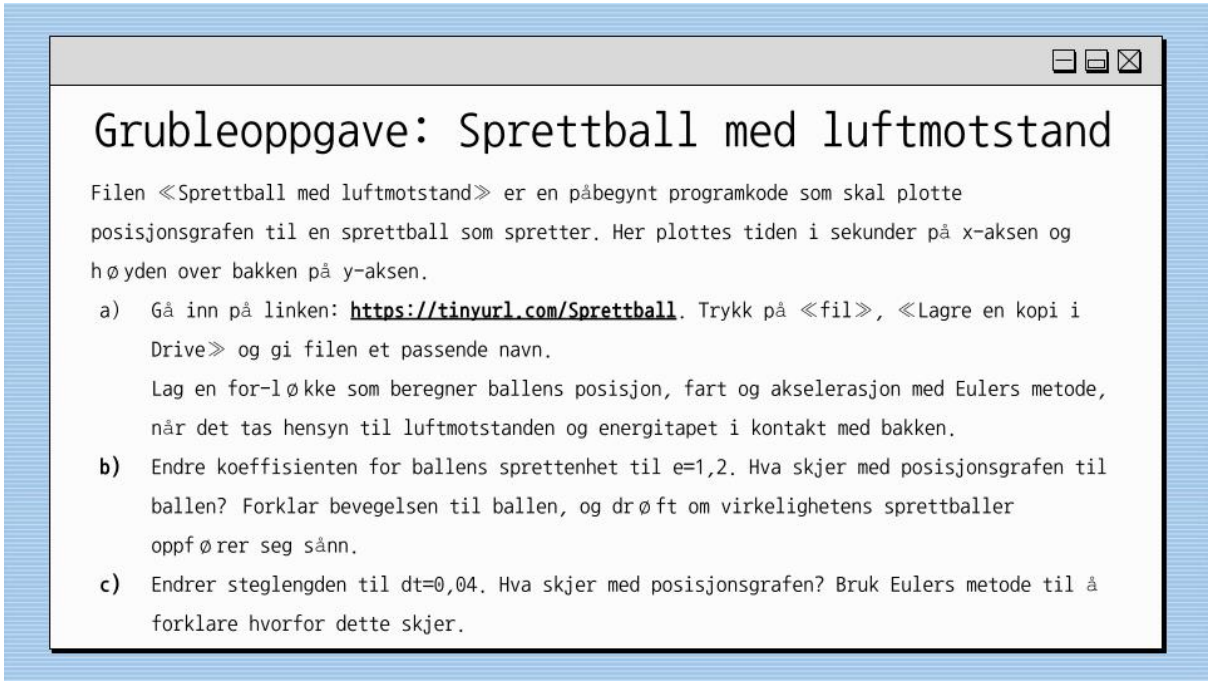
i) Forklare hvorfor det å endre verdien til tidssteget dt endrer ballens beregnede maksimale høyde.

j) Hvilken verdi av dt ville du brukt for å beregne ballens posisjon? Begrunn svaret ditt.

Lysbilde I-17



Lysbilde I-18



Vedlegg J: Utdelt programkode og løsningsforslag

Den utdelte programkoden og tilhørende løsningsforslag er også tilgjengelig for nedlastning på nett fra https://tinyurl.com/laererressurser.

Oppgaveark 1: «Bil med konstant akselerasjon»

```
import numpy as np
import matplotlib.pyplot as plt

# tidsintervaller
n = 250 # antall tidspunkt vi beregner tilbakelagt strekning
dt = 0.01 # tidssteg, sekunder mellom hvert tidspunkt vi evaluerer

# tomme arrayer som fylles med verdier senere
s = np.zeros(n)       # array for strekningsdata
v = np.zeros(n)       # array for fartsdata
t = np.zeros(n)       # array for tidsdata
a = np.zeros(n)       # array for akselerasjonsdata

#startbetingelser
v[0] = 5 # startfart i m/s
a[0] = 1.7 # startakselerasjon i m/s^2

for i in range(n-1):          # itererer gjennom kodesnutten n-1=999 ganger
    a[i] = a[0]               # setter akselereasjoen som konstant
    v[i+1] = v[i] + a[i]*dt   # finner farten etter en bestemt tid
                              # ved 1. bevegeleslikning
    t[i+1] = t[i] + dt        # regner ut hvor mye tid som har passert
    s[i+1] = s[i] + v[i]*dt   # regner ut ny posisjon,
                              # som er gammel posisjon pluss endring
    a[i+1] = a[0]             # setter akselereasjoen som konstant

#plotter posisjonsgraf
plt.figure(1)                 # lager en figur
plt.title("Posisjon")         # gir figuren en tittel
plt.xlabel("$t$(s)")          # gir x-aksen navn
plt.ylabel("$s$/m")           # gir y-aksen navn
plt.plot(t,s)                 # plotter tiden-strekningsgraf
```

```
33 plt.grid()                       # lager et rutenett
34
35 #plott fartsgraf
36 plt.figure(2)                    # lager en figur
37 plt.title("Fart")                # gir figuren en tittel
38 plt.xlabel("$t$/s")              # gir x-aksen navn
39 plt.ylabel("$v$(m/s)")           # gir y-aksen navn
40
41 plt.grid()                       # lager et rutenett
42
43 #plott akselerasjonsgraf
44 plt.figure(3)                    # lager en figur
45 plt.title("Akselerasjon")        # gir figuren en tittel
46 plt.xlabel("$t$/s")              # gir x-aksen navn
47 plt.ylabel("$a$(m/s^2)")         # gir y-aksen navn
48 plt.grid()                       # lager et rutenett
49
50 plt.show()                       # viser de tre figurene
```

Løsningsforslag: «Bil med konstant akselerasjon»

```
import numpy as np
import matplotlib.pyplot as plt

# tidsintervaller
n = 1000 # antall tidspunkt vi beregner tilbakelagt strekning
dt = 0.01 # tidssteg, sekunder mellom hvert tidspunkt vi evaluerer

# tomme arrayer som fylles med verdier senere
s = np.zeros(n)       # array for strekningsdata
v = np.zeros(n)       # array for fartsdata
t = np.zeros(n)       # array for tidsdata
a = np.zeros(n)       # array for akselerasjonsdata

#startbetingelser
v[0] = 5 # startfart i m/s
a[0] = 2 # startakselerasjon i m/s^2

for i in range(n-1):          # itererer gjennom kodesnutten n-1=999 ganger
    a[i] = a[0]               # setter akselereasjoen som konstant
    v[i+1] = v[i] + a[i]*dt   # finner farten etter en bestemt tid
                              # ved 1. bevegeleslikning
    t[i+1] = t[i] + dt        # regner ut hvor mye tid som har passert
    s[i+1] = s[i] + v[i]*dt   # regner ut ny posisjon,
                              # som er gammel posisjon pluss endring
    a[i+1] = a[0]             # setter akselereasjoen som konstant

#plotter posisjonsgraf
plt.figure(1)                 # lager en figur
plt.title("Posisjon")         # gir figuren en tittel
plt.xlabel("$t$/s")           # gir x-aksen navn
plt.ylabel("$s$/m")           # gir y-aksen navn
plt.plot(t,s)                 # plotter tiden-strekningsgraf
plt.grid()                    # lager et rutenett

#plotter fartsgraf
plt.figure(2)                 # lager en figur
plt.title("Fart")             # gir figuren en tittel
```

```
38 plt.xlabel("$t$(s)")           # gir x-aksen navn
39 plt.ylabel("$v$(m/s)")         # gir y-aksen navn
40 plt.plot(t,v, color = "r")     # plotter tid-farstgraf
41 plt.grid()                     # lager et rutenett
42
43 #plotter akselerasjonsgraf
44 plt.figure(3)                  # lager en figur
45 plt.title("Akselerasjon")      # gir figuren en tittel
46 plt.xlabel("$t$(s)")           # gir x-aksen navn
47 plt.ylabel("$a$(m/s^2)")       # gir y-aksen navn
48 plt.plot(t,a, color = "g")     # plotter tid-akselerasjonsgraf
49 plt.grid()                     # lager et rutenett
50
51 plt.show()                     # viser de tre figurene
```

Oppgaveark 2: «Vertikalt kast med luftmotstand»

```python
import numpy as np
import matplotlib.pyplot as plt

# HER SKAL DERE DEFINERE DE FYSISKE STORRELSENE
# OG KONSTATNENE DERE TRENGER I OPPGAVEN

# tidsintervaller
n = 1000          #antall tidspunkt vi beregner tilbakelagt strekning
dt = 0.01         # tidssteg, sekunder mellom hvert tidspunkt vi evaluerer

# tomme arrayer som fylles med verdier senere
h = np.zeros(n)       # array for strekningsdata
v = np.zeros(n)       # array for fartsdata
t = np.zeros(n)       # array for tidsdata
a = np.zeros(n)       # array for akselerasjonsdata

#initialbetingelser
v[0] = 10

#Bestemmer kreftene og regner ut akselerasjonen med N2 for hvert tidssteg
#Antar konstant fart og akselerasjon i tidssteget,
# og bruker i beregningen av ny fart og ny posisjon
#Her har vi valgt negativ retning nedover

for i in range(n-1):
    # Ballen beveger seg oppover
    if v[i] > 0:
        G = m * g
        L = k * v[i]**2
        F_sum = 0 # BYTT UT 0 MED ET UTTRYKK FOR SUMMEN AV KREFTER
                  # SOM VIRKER PAA BALLEN
        a[i] = F_sum/m
        # SKRIV KODE SOM REGNER UT FARTEN,
        # TIDEN OG HOYDEN VED NESTE TIDSSTEG MED EULERS METODE

    # Ballen beveger seg nedover
    elif v[i] < 0:
```

```
        G = m * g
        L = k * v[i]**2
        F_sum = 0 # BYTT UT 0 MED ET UTTRYKK FOR SUMMEN AV KREFTER
                  # SOM VIRKER PAA BALLEN
        a[i] = F_sum/m
        # SKRIV KODE SOM REGNER UT FARTEN,
        # TIDEN OG HOYDEN VED NESTE TIDSSTEG MED EULERS METODE

    # Hopper ut av for-lokken naar ballen nesten ikke spretter lenger
    elif h[i] < 0.01:
        break

#plotter posisjonsgraf
plt.figure(1)                           # lager en figur
plt.title("posisjon")                   # gir figuren en tittel
plt.xlabel("$t$(s)")                    # gir x-aksen navn
plt.ylabel("$h$(m)")                    # gir y-aksen navn
plt.xlim([0,2.5])                       # begrenser x-aksen
plt.ylim([-1,7])                        # begrenser y-aksen
plt.plot(t,h,label='k=%.2f'%k)          # plotter tid-strekningsgraf og
      # lager en merkelapp som viser verdien til k
plt.legend()                            # viser merkelappen for k-verdien
plt.grid()                              # lager et rutenett


#plotter fartsgraf
plt.figure(2)                           # lager en figur
plt.title("fart")                       # gir figuren en tittel
plt.xlabel("$t$(s)")                    # gir x-aksen navn
plt.ylabel("$v$/(m/s)")                 # gir y-aksen navn
plt.xlim([0,2.5])                       # begrenser x-aksen
plt.plot(t,v,label='k=%.2f'%k)          # plotter tid-fartsgraf og
      # lager en merkelapp som viser verdien til k
plt.legend()                            # viser merkelappen for k-verdien
plt.grid()                              # lager et rutenett

#plotter akselerasjonsgraf
plt.figure(3)                           # lager en figur
plt.title("akselerasjon")               # gir figuren en tittel
```

```
77 plt.xlabel("$t$(s)")                          # gir x-aksen navn
78 plt.ylabel("$a$/(m/s^2)")                     # gir y-aksen navn
79 plt.xlim([0,2.5])                             # begrenser x-aksen
80 plt.plot(t[:-1],a[:-1],label='k=%.2f'%k) # plotter tid-akselerasjonsgraf og
81       # lager en merkelapp som viser verdien til k
82 plt.legend()                                  # viser merkelappen for k-verdien
83 plt.grid()                                    # lager et rutenett
84
85 plt.show()                                    # viser figurene
```

Løsningsforslag: «Vertikalt kast med luftmotstand»

```
import numpy as np
import matplotlib.pyplot as plt

# definerer fysiske storrelser og konstanter
g = 9.81      # m/s^2
m = 0.2       # kg
k = 0.01      # koeffisienten for luftmotstand

# tidsintervaller
n = 1000          #antall tidspunkt vi beregner tilbakelagt strekning
dt = 0.01         # tidssteg, sekunder mellom hvert tidspunkt vi evaluerer

# tomme arrayer som fylles med verdier senere
h = np.zeros(n)       # array for strekningsdata
v = np.zeros(n)       # array for fartsdata
t = np.zeros(n)       # array for tidsdata
a = np.zeros(n)       # array for akselerasjonsdata

#initialbetingelser
v[0] = 10

#Bestemmer kreftene og regner ut akselerasjonen med N2 for hvert tidssteg
#Antar konstant fart og akselerasjon i tidssteget,
# og bruker i beregningen av ny fart og ny posisjon
#Her har vi valgt negativ retning nedover

for i in range(n-1):
    # Ballen beveger seg oppover
    if v[i] > 0:
        G = m * g
        L = k * v[i]**2
        F_sum = -G - L
        a[i] = F_sum/m
        v[i+1] = v[i] + a[i] * dt
        t[i+1] = t[i] + dt
        h[i+1] = h[i] + v[i]*dt

```

```
    # Ballen beveger seg nedover
    elif v[i] < 0:
        G = m * g
        L = k * v[i]**2
        F_sum = -G + L
        a[i] = F_sum/m
        v[i+1] = v[i] + a[i] * dt
        t[i+1] = t[i] + dt
        h[i+1] = h[i] + v[i]*dt

    # Hopper ut av for-lokken naar ballen nesten ikke spretter lenger
    elif h[i] < 0.01:
        break

#plotter posisjonsgraf
plt.figure(1)                           # lager en figur
plt.title("posisjon")                   # gir figuren en tittel
plt.xlabel("$t$(s)")                    # gir x-aksen navn
plt.ylabel("$h$(m)")                    # gir y-aksen navn
plt.xlim([0,2.5])                       # begrenser x-aksen
plt.ylim([-1,7])                        # begrenser y-aksen
plt.plot(t,h,label='k=%.2f'%k)          # plotter tid-strekningsgraf og
      # lager en merkelapp som viser verdien til k
plt.legend()                            # viser merkelappen for k-verdien
plt.grid()                              # lager et rutenett


#plotter fartsgraf
plt.figure(2)                           # lager en figur
plt.title("fart")                       # gir figuren en tittel
plt.xlabel("$t$(s)")                    # gir x-aksen navn
plt.ylabel("$v$/(m/s)")                 # gir y-aksen navn
plt.xlim([0,2.5])                       # begrenser x-aksen
plt.plot(t,v,label='k=%.2f'%k)          # plotter tid-fartsgraf og
                                        # lager en merkelapp som viser verdien til k
plt.legend()                            # viser merkelappen for k-verdien
plt.grid()                              # lager et rutenett

#plotter akselerasjonsgraf
```

```
77 plt.figure(3)                                # lager en figur
78 plt.title("akselerasjon")                    # gir figuren en tittel
79 plt.xlabel("$t$(s)")                         # gir x-aksen navn
80 plt.ylabel("$a$/(m/s^2)")                    # gir y-aksen navn
81 plt.xlim([0,2.5])                            # begrenser x-aksen
82 plt.plot(t[:-1],a[:-1],label='k=%.2f'%k) # plotter tid-akselerasjonsgraf og
83       # lager en merkelapp som viser verdien til k
84 plt.legend()                                 # viser merkelappen for k-verdien
85 plt.grid()                                   # lager et rutenett
86
87 plt.show()                                   # viser figurene
```

Grubleoppgave: «Sprettball med luftmotstand»

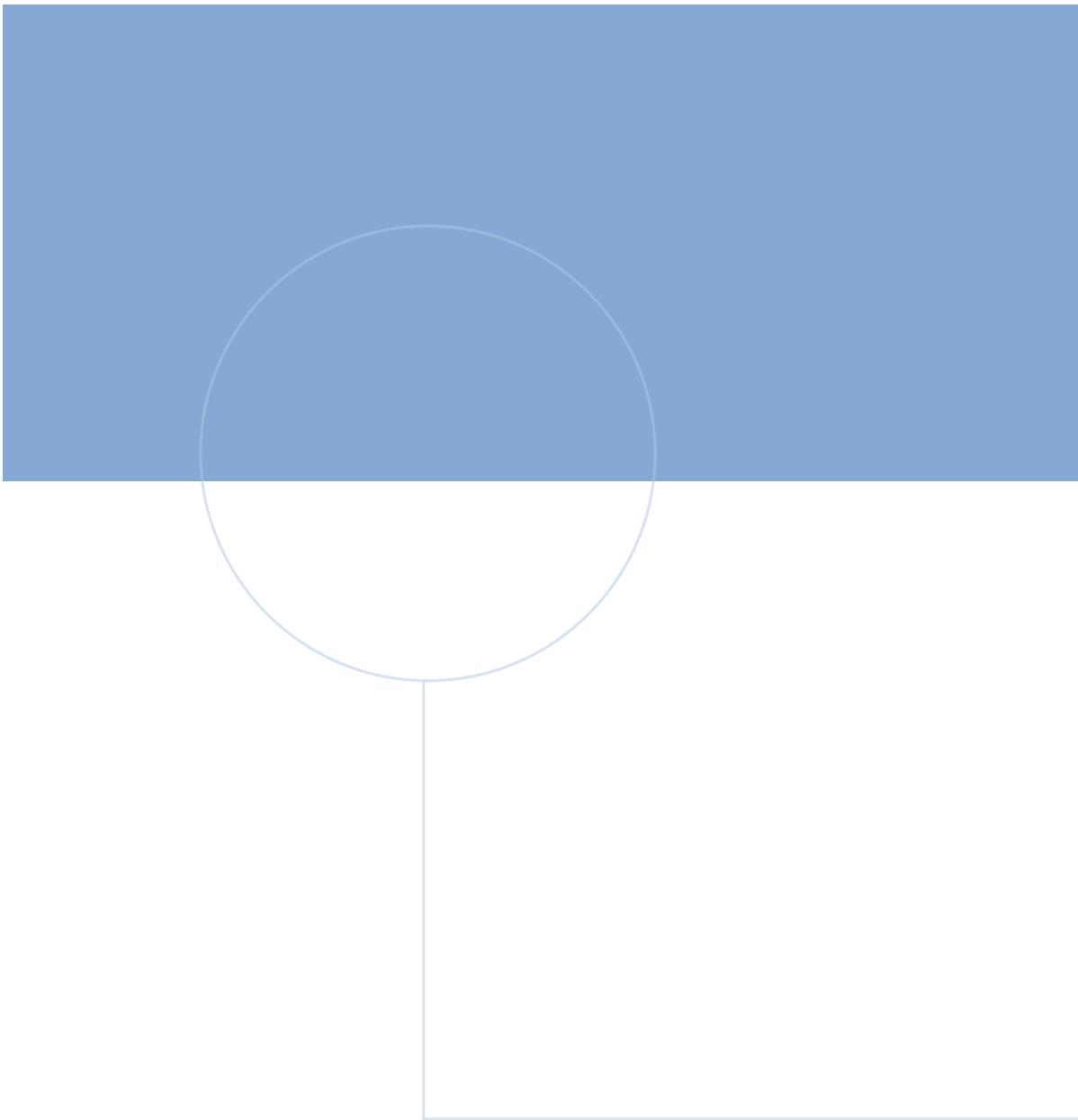
```
import numpy as np
import matplotlib.pyplot as plt

# tidsintervaller
n = 1000          #antall tidspunkt vi beregner tilbakelagt strekning
dt = 0.01         # tidssteg, sekunder mellom hvert tidspunkt vi evaluerer

# tomme arrayer som fylles med verdier senere
h = np.zeros(n)       # array for strekningsdata
v = np.zeros(n)       # array for fartsdata
a = np.zeros(n)       # array for akselerasjonsdata
t = np.zeros(n)       # array for tidsdata

# Fysiske storrelser og konstanter
m = 10     # kg
g = 9.81   # m/s^2
k = 0.01   # koeffisienten for luftmotstand i N/(m/s)^2
e = 0.8    # koeffisient for ballens sprettenhet.
           # 0 tilsvarer ingen sprett,
           # mens 1 betyr at ballen ikke mister energi i sprettet

# Startverdier
v[0] = 0 # m/s
h[0] = 4 # m

# LAG EN FOR-LOKKE SOM BEREGNER BALLENS NESTE POSISJON,
# FART OG AKSELERASJON FRA  BALLENS FORRIGE POSISJON, FART OG AKSELERAJSON
# HUSK AT BALLEN MISTER ENERGI IDET DEN TREFFER BAKKEN

#plotter posisjonsgraf
plt.figure(1)
plt.title("posisjon")
plt.xlabel("$t$(s)")
plt.ylabel("$h$(m)")
plt.plot(t,h)
plt.grid()

```

```
#plotter fartsgraf
plt.figure(2)
plt.title("fart")
plt.xlabel("$t$(s)")
plt.ylabel("$v$(m/s)")
plt.plot(t,v)
plt.grid()
plt.show()
```

Løsningsforslag: «Sprettball med luftmotstand»

```
import numpy as np
import matplotlib.pyplot as plt

# tidsintervaller
n = 1000          #antall tidspunkt vi beregner tilbakelagt strekning
dt = 0.01         # tidssteg, sekunder mellom hvert tidspunkt vi evaluerer

# tomme arrayer som fylles med verdier senere
h = np.zeros(n)       # array for strekningsdata
v = np.zeros(n)       # array for fartsdata
a = np.zeros(n)       # array for akselerasjonsdata
t = np.zeros(n)       # array for tidsdata

# Fysiske storrelser og konstanter
m = 10      # kg
g = 9.81    # m/s^2
k = 0.01    # koeffisienten for luftmotstand i N/(m/s)^2
e = 0.8     # koeffisient for ballens sprettenhet.
            # 0 tilsvarer ingen sprett,
            # mens 1 betyr at ballen ikke mister energi i sprettet

# Startverdier
v[0] = 0 # m/s
h[0] = 4 # m

# Beregner sprettballens neste posisjon,
# fart og akselerasjon fra ballens forrige
# posisjon, fart og akselerasjon
for i in range(n-1):
    G = m*g
    L = k*v[i]*abs(v[i])
    F_sum = G + L
    a[i] = F_sum/m
    h[i+1] = h[i] - v[i] * dt
    v[i+1] = v[i] + a[i] * dt
    t[i+1] = t[i] + dt
    if h[i+1] <= 0.01 and v[i+1]>=0: # Evaluerer ballen i sprettoyeblikket.
```

```
38          v[i+1] = -v[i+1] * e          # Skifter fortegn paa farten og
39                                        # reduseres med en faktor e
40          k = k*(-1)                    # Endrer retningen paa luftmotstanden
41
42  #plotter posisjonsgraf
43  plt.figure(1)
44  plt.title("posisjon")
45  plt.xlabel("$t$(s)")
46  plt.ylabel("$h$(m)")
47  plt.plot(t,h)
48  plt.grid()
49
50  #plotter fartsgraf
51  plt.figure(2)
52  plt.title("fart")
53  plt.xlabel("$t$(s)")
54  plt.ylabel("$v$(m/s)")
55  plt.plot(t,v)
56  plt.grid()
57  plt.show()
```



NTNU

Kunnskap for en bedre verden