

Bidragsformat (Presentasjon)

Numeriske metoder og fysikk-undervisning: Erfaringer fra en pilot ved UiB

Sivert Hagane¹, Augustin Alexander Winther¹, Johan Alme¹,
Vegard Gjerde¹ og Martino Marisaldi¹

¹ Institutt for fysikk og teknologi, Universitetet i Bergen, Norge

Sammendrag: Dette prosjektet har som mål å integrere numeriske metoder, databehandling og visualisering i alle fysikkemnene på bachelorprogrammet i fysikk ved hjelp av Python. Ved å bruke programmering jobber studentene mer i tråd med hvordan fysikere faktisk arbeider, både i videre studier og i arbeidslivet. Numeriske verktøy gir også muligheter for å visualisere komplekse fysikkonsepter, noe som kan hjelpe studentene å bygge forståelse for fysikken.

Høsten 2024 gjennomførte vi en pilot for dette prosjektet med kurset PHYS113 «Mekanikk II og Termodynamikk» ved Universitet i Bergen. I denne presentasjonen vil vi først og fremst fokusere på implementasjonen og resultatene fra piloten. I forarbeidet til piloten, ble det utviklet en del rettesnorer som lå til grunn for implementasjonen, for eksempel ønsket vi et studentdrevet prosjekt der programmeringskomponenten må være relevant for læringsutbyttet i fysikk.

Resultatene fra piloten er samlet inn via analyse av eksamensresultater og standardiserte tester i kurset, i tillegg til studentevalueringer. Et intervju med en fokusgruppe i etterkant av implementasjonen er også planlagt. Basert på dette er inntrykket positivt, og at piloten er et godt utgangspunkt for implementasjon i andre emner.

Nøkkelord:

Programmering, Fysikk, Studieprogram, Python, Numeriske metoder

1 Begrunnelse og relevans

1.1 Formålet med prosjektet

Målet med prosjektet er å inkludere numeriske øvelser, og/eller databehandling og visualisering i alle fysikkemnene i bachelorprogrammet i fysikk. Til dette er Python valgt som verktøy. Det er flere motivasjonsfaktorer for dette prosjektet. For det første muliggjør bruken av numeriske verktøy nye måter å visualisere fysikk-konsepter på, som kan være vanskelige for studentene å forstå. For det andre vil bruk av programmering få studentene til å i større grad “jobbe som fysikere” i kursene, dvs. de tilegner seg en metodikk som ligner det de vil møte senere i studiene og i jobbsammenheng. Det er også verdt å merke seg at under programevalueringen høsten 2024 understreket både industrirepresentanten og den eksterne fagfellen viktigheten av programmeringsferdigheter i sine evalueringsrapporter.

1.2 Motivasjon

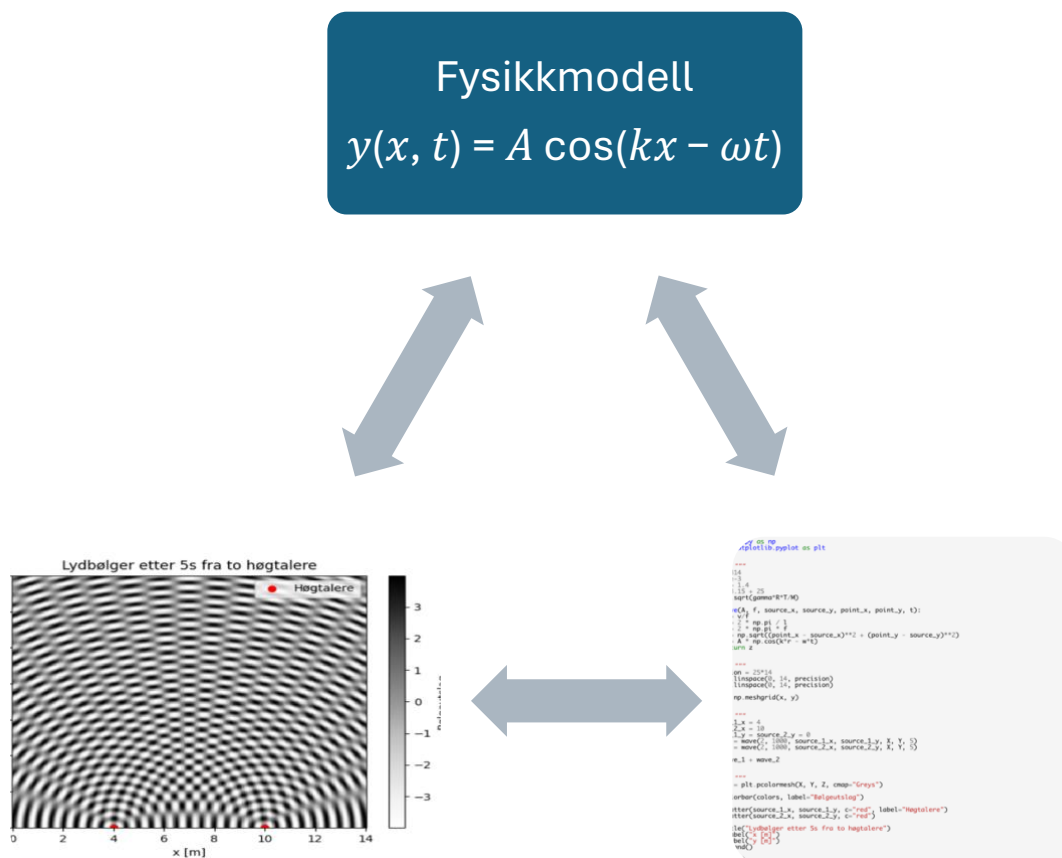
Frem til nå har studentene i bachelorprogrammet i fysikk ved UiB kun blitt eksponert for programmering i INF100 i første semester, PHYS114 i fjerde semester og PHYS116 i femte semester. Denne mangelen på kontinuitet i utviklingen av programmeringsferdigheter er svært tydelig i PHYS114, hvor det har vært observert studenter som sliter med oppgaver som krever svært grunnleggende programmeringsferdigheter. Ved å inkludere programmeringskomponenter i de andre fysikkemnene, har vi som mål å bygge bro over dette gapet, samt å gi studentene et solid grunnlag for masterprosjektene deres. Dette er viktig, da de fleste masterprosjektene ved vårt institutt krever noe programmeringskompetanse, enten det er dataanalyse og visualisering, simuleringsarbeid, utvikling av programvare og maskinvare eller statistisk analyse.

1.3 Tidligere forskning/Teori

Anders Malthe-Sørenssen et al. har gjennomført en liknende implementering av programmeringskompetanse ved fysikkutdanningen ved Universitetet i Oslo. En del av bakgrunnen for denne implementeringen var at beregninger er en viktig del av en fysikers jobb, både i academia og i industrien i dag, men man lærte nesten ingenting av dette i fysikkutdanningen (Malthe-Sørenssen et al., 2015).

For at studenten skal lære fysikk gjennom bruk av programmering krever det at studentene evner å bruke transduksjoner mellom ulike semiotiske system. Programmering av fysikk for å finne numeriske løsninger krever transduksjoner mellom ulike typer semiotiske systemer, der hvert system gir forskjellig meningspotensial, altså ulike muligheter til å forstå fysikken. Overganger mellom ulike semiotiske systemer vil

kreve at studenten definerer de ulike transduksjonene, noe som fremhever koblingen mellom de ulike systemene. I vårt tilfelle vil studentene ofte starte med å utforme en fysikkmodell bestående av matematiske symboler og må dermed overføre denne til et nytt semiotisk system, Python-kode. Deretter får studentene ut et visuelt resultat som kan tolkes og vi har en ny transduksjon og et nytt semiotisk system. Gjennom denne typen arbeid vil studentene kunne oppdage og utforske aspekter ved fysikken som det er vanskelig å gjøre kun med fysikkmodellen. Koden gir også rask feedback hvis man endrer verdier i koden, og det kan skape mange ulike læringsmuligheter gjennom overganger mellom de ulike semiotiske systemene. (Svensson et al., 2020).



Figur 1: Transduksjoner mellom ulike typer semiotiske systemer (fysikkmodell, kode og visuelt resultat) under programmering av fysikk.

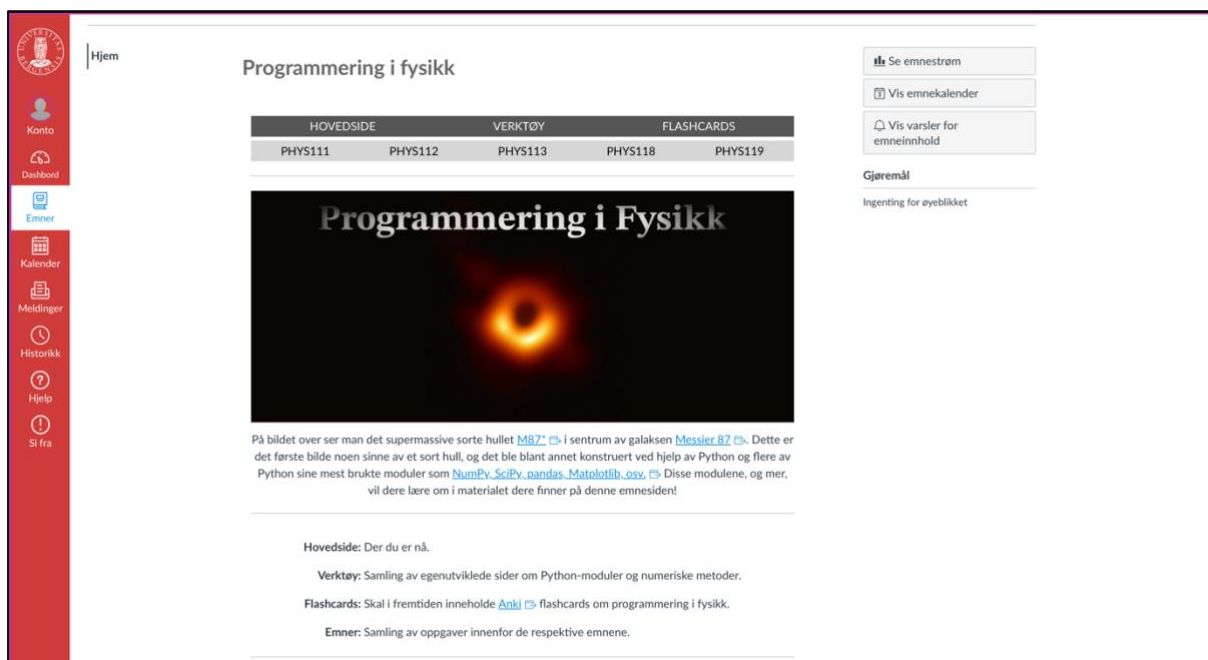
Studentene fikk i de fleste tilfeller tips, verktøy og/eller en «skjelettkode» som bestod av kode som visualiserer eller andre deler av koden som vi mente var nyttig for bedre fysikkforståelse, men som studentene ikke selv er forventet å kunne få til selv. Et eksempel på dette presenteres i Figur 5. Hvis studentene sitter fast eller trenger hjelp for å komme i gang er dette et eksempel på hvordan studentene får tilgang på hint og testkode de kan bruke når de løser oppgavene. Dette ble gjort ved å lage en mindre deloppgave av oppgavene. Dette kan kategoriseres som en «Hybrid approach» til programmering, hvor studentene får hjelp underveis i programmeringen eller deler av

koden. Dette blir gjort for å redusere den kognitive belastningen til studentene. Det er ofte at når programmering blir introdusert for å løse fysikkproblemer, fører det til kognitive overbelastningen hos studentene og dermed redusert læringsutbytte (Orban et al., 2018).

2 Metode

2.1 Introduksjon

Høsten 2024 gjennomførte vi en pilot hvor emnet PHYS113 ble valgt til å innføre programmering i først, for å lære og som inspirasjon for implementering i andre fag litt underveis. To studenter ble ansatt, og i tillegg til å lage oppgaver, eksempler og støttemateriale, utviklet de to studentene også et rammeverk bestående av en Mitt UiB-kursside¹, og et GIT repository for utvikling av materiale. Forsiden på Mitt UiB kurssiden er presentert i Figur 1.



Figur 2: Forsiden til Mitt UiB siden studentene på bachelorprogrammet i fysikk ved UiB har tilgang til. Her har de enkel tilgang til alle grunnkursene, samt verktøy og flashcards.

2.2 Prinsipper for utviklingsarbeidet

I parallell med dette utviklet vi i fellesskap noen rettesnorer for prosjektet:

- Prosjektet bør være studentdrevet, dvs. struktur, øvelser og eksempler bør lages av studenter i samarbeid med de ansvarlige for kursene. Dette har vist

¹ <https://mitt.uib.no/courses/50380>

seg å være en effektiv tilnærming i prosjektet “PAFYS: Prinsippbasert Aktiv Undervisning for Sterkere Fysikk- og Ingeniørstudenter” ved vårt institutt.

- Programmering bør introduseres enten som dataanalyse og visualisering og/eller som løsning av fysikkproblemer med numeriske metoder.
- Programmering skal øke læringsutbyttet i fysikk, og forbedring av programmeringsferdighetene bør være en viktig sideeffekt.
- En viss andel av fysikkøvelsene bør løses numerisk, og helst bør disse ikke være mulig å løse analytisk. Hvor stor andelen er bestemt av relevansen for kurset.
- Innføringen av programmeringen bør ikke øke arbeidsomfanget for det gitte emnet.
- Forelesningene må inkludere programmeringseksempler for å øke relevansen for studentene.
- Eksamen bør inkludere programmeringsrelaterte oppgaver med omfang på ca 5-10% for å sikre meningsskapende samsvar.

2.3 Oppgave og eksempeldesign i piloten

PHYS113 - Hovedside

HOVEDSIDE	VERKTØY	FLASHCARDS
PHYS111	PHYS112	PHYS113
PHYS118	PHYS119	

For hvert tema i emnet er det laget til én Python oppgave som det er tenkt at dere skal løse.


Forelesningskoder, oppgaver og løsningsforslag blir gradvis tilgjengelig utover semesteret, og noen av oppgavene vil fungere som obligatoriske oppgaver.

Oppgavene er laget med forbehold om at studenten har Python-kunnskap tilsvarende [INF100](#). Om du trenger litt oppfriskning/repetisjon i INF100, kan du ta en titt på [disse pensum-notatene](#).

Tema	Linker
Gravitasjon	Forelesningskode Oppgave Løsningsforslag
Periodisk bevegelse	Forelesningskode Oppgave Løsningsforslag
Mekaniske bølger	Forelesningskode Oppgave Løsningsforslag
Lyd og lydbølger	Forelesningskode Oppgave Løsningsforslag
Lagrange og Hamilton	Forelesningskode Oppgave Løsningsforslag
Temperatur og varme	Forelesningskode Oppgave Løsningsforslag
Termiske egenskaper	Oppgave Løsningsforslag
Termodynamikkens første lov	Oppgave Løsningsforslag
Termodynamikkens andre lov	Oppgave Løsningsforslag

Figur 3: Slik ser en emneside ut. Dette er fra PHYS113 Mekanikk II og Termodynamikk. Her ligger koder fra forelesningene som forleser bruker til eksempler, samt oppgaver til hver del av pensum og løsningsforslag til koden.

Piloten ble gjennomført med bakgrunn i disse prinsippene. For å sikre at ikke arbeidsomfanget økte ble tre obligatoriske øvelser med analytiske oppgaver endret til å bestå av programmeringsoppgaver rundt de samme temaene som tidligere. I tillegg ble eksempler på numerisk løsning gjennomgått i forelesningene, og 7% av eksamen ble knyttet til programmering. Eksamensoppgaven ble designet slik at studenten skulle vise at hen forstod en utlevert kode, samt å kunne knytte denne til et fysikkprinsipp i emnet. Eksamen for øvrig forble uendret.



Oppgave

Du er gitt samme CSV-fil som i eksempeloppgaven ovenfor.

Oppgavetekst: Estimer magnituden av banehastigheten til alle planetene ved å anta at de har sirkulær bane rundt Solen. Sammenlign så den estimerte banehastigheten med den faktiske, og print ut til terminalen absolutt-forskjellen i m/s , samt prosent, sammen med navnet og eksentrisiteten til planetenes baner. Eksempel på hvordan det kan se ut (med tullede tall):

```
Mercury : 33 m/s | 93 % | Eccentricity: 4432
Venus : 1 m/s | 3 % | Eccentricity: 123
Earth : 2454 m/s | 55 % | Eccentricity: 44
...
```

► Deloppgaver

► Print-ut fasit

Figur 4: Eksempel på hvordan en oppgave blir presentert for elevene på Mitt UiB siden.

Til sammen var 46 studenter med på piloten, hvor ca halvparten var fra bachelorprogrammet i fysikk, og resten var tilknyttet andre studieprogrammer ved fakultetet.

Figur 3 viser hvordan emneweb siden ser ut for studentene. Hovedtemaene som blir gjennomgått i kurset har alle minst et eksempel til bruk i forelesningen, siden det er viktig for studentenes følelse av relevans å møte programmering i alle læringssituasjoner. I tillegg er det oppgaver og løsningsforslag (som man kan velge å ikke publisere). Et eksempel på en oppgave er vist i Figur 4, der man ser at den er brutt ned i enklere deloppgaver. Figur 5 viser denne strukturen og viser også hvordan studentene kan få hint og hjelp underveis hvis de står fast.

▼ Deloppgaver

a) Lag 6 NumPy arrays: `name`, `pos_x`, `pos_y`, `vel_x`, `vel_y` og `eccen`. De skal inneholde verdiene til kolonnene med de samme navnene fra CSV-filen, hvor `eccen` representerer "eccentricity" kolonnen. Indeks 0 i hver array tilsvarer Solens egenskaper, indeks 1 i hver array tilsvarer Merkur sine egenskaper, osv.

▼ Hint

Her vil funksjonen `read_csv()` fra pandas modulen være veldig nyttig, sammen med DataFrame metoden `to_numpy()`.

▼ Testkode

Test koden din ved å legge til disse linjene nederst i filen før du kjører:

```
print('Tester name, pos_x, pos_y, vel_x, vel_y og eccen... ', end='')
import hashlib
hash_it = lambda x : hashlib.shake_256(str(x).encode("utf-8")).hexdigest(8)
assert hash_it(f"{name} {pos_x} {pos_y}"
               f" {vel_x} {vel_y} {eccen}") == "4964c0f59d39f993"
print('OK')
```

Om alt gikk fint kan du fjerne testkoden og gå videre.

b) Lag en funksjon `estimated_velocity(pos)` som tar inn en 1D NumPy array `pos` som inneholder x- og y-verdien til en posisjonsvektor. Funksjonen returnerer en skalar som er den estimerte hastigheten en masse ved posisjonen `pos` ville hatt om den gikk i sirkulær bane rundt Solen. Anta at gravitasjonskonstanten er $G = 6.674 \times 10^{-11} \text{ N m}^2 \text{ kg}^{-2}$

► Testkode

c) Ved hjelp av funksjonen du lagde i oppgave b), og arrayene du lagde i oppgave a), print ut absolutt hastighetsdifferansen mellom den estimerte hastigheten og den faktiske hastigheten til alle planetene, sammen med prosentdifferansen (typ. om en planet hadde hastighet på 100m/s og vi estimerte 110m/s eller 90m/s, så er prosentdifferansen 10%). Print også ut eksentrisiteten til banene til planetene.

► Hint

► Print-ut fasit

Figur 5. Hvis studentene sitter fast eller trenger hjelp for å komme i gang er dette et eksempel på hvordan studentene får tilgang på hint og testkode de kan bruke når de løser oppgavene. Dette ble gjort ved å lage mindre deloppgaver av oppgavene.

2.4 Målbare parametere av læringsutbyttet

For å måle effekten av innføringen av programmeringen i piloten har vi valgt å se på fire parametere:

1. Studentevalueringer. Disse kan gi oss en forståelse om hvordan studentene opplevde programmeringsoppgavene. Ble det en ekstra byrde som tok fokuset bort fra fysikken, eller opplevde de oppgavene som relevante og nyttige for læring?
2. Eksamensresultat. Er eksamensresultatene signifikant svakere enn tidligere år kan dette tyde på at fysikklæringen har blitt negativt påvirket av den nye komponenten.
3. Poengscore programmeringsoppgave eksamen. Klarer studentene knytte den numeriske løsningen til fysikkprinsippene?
4. Resultat på faktatest og konseptuell flervalgstest. Dette er tester som har vært gjennomført over flere år. Fordelen med disse testene er at de er, i motsetning til

eksamensoppgavene, uendret siden 2022. Dette vil gi et objektivt bedre svar på om programmeringen har påvirket læringsutbyttet til studentene i særlig grad.

Som nevnt over, er studentenes prestasjoner evaluert gjennom tre typer tester: en faktatest, en konseptuell flervalgstest og eksamen. Det er verdt å nevne at 2021 var kontrollåret for en intervensjonsstudie. Noen nye metoder ble innført i 2022 og 2023, og disse ble i stor grad videreført til 2024, så derfor er det mest relevant å se på resultatene fra 2024 i kontekst med de to foregående år.

- **Faktatest:** Faktatesten ble gjennomført på en digital plattform. Testen var utviklet av emneansvarlig og dekket SI-enheter, sentrale fysikkprinsipper og betingelser for bruk av disse prinsippene. Alle oppgavene hadde positive diskriminasjons-verdier, og testens pålitelighet var innenfor et ideelt område (Cronbachs alpha 0.81–0.85).
- **Konseptuell flervalgstest:** Den konseptuelle flervalgstesten ble også gjennomført på en digital plattform samme uke som faktatesten. Testen «Survey of Thermodynamic Processes and First and Second Laws» er benyttet, en veletablert og validert test innen termodynamikk.
- **Eksamen:** Eksamen følger en standardisert struktur der 70 % av poengene gis for tradisjonelle tekstbaserte fysikkoppgaver. I tillegg er det fem konseptuelle flervalgs-spørsmål (10 % av poengene) samt en skriftlig begrunnelse for valg av svar (20 %). For full uttelling måtte studentene argumentere ved hjelp av fysikkprinsipper og formelt korrekt resonnement. I 2024 ble en av de tradisjonelle tekstbaserte oppgavene endret til å også vise forståelse for en numerisk løsning.

I tillegg arbeides det med å sette ned fokusgrupper med studenter som har erfaring med piloten, samt med studenter som møter programmeringen for første gang i et annet emne. For å sikre diversitet ønsker vi snakke med studenter av ulike kjønn, fra ulike bakgrunner og forskjellige studieprogram. Arbeidet med å etablere fokusgruppen er pågående, og vi kan dermed ikke presentere resultater enda.

3 Resultater

Den årlige studentemneevalueringen i PHYS113 ble besvart av 15 av 46 studenter. Dette er en svarprosent på ca 33%, som er ganske vanlig i våre emner. Vi kan tolke dette dithen at omtrent 67% av studentene ikke hadde spesielle meninger om programmeringen spesielt og kurset generelt. Av de som svarte var det flere som var positive enn negative til hvordan programmeringen ble innført i emnet. De positive skriftlige tilbakemeldingene var også mer konstruktive i sin tilbakemelding, hvor vi kan ta frem to eksempler:

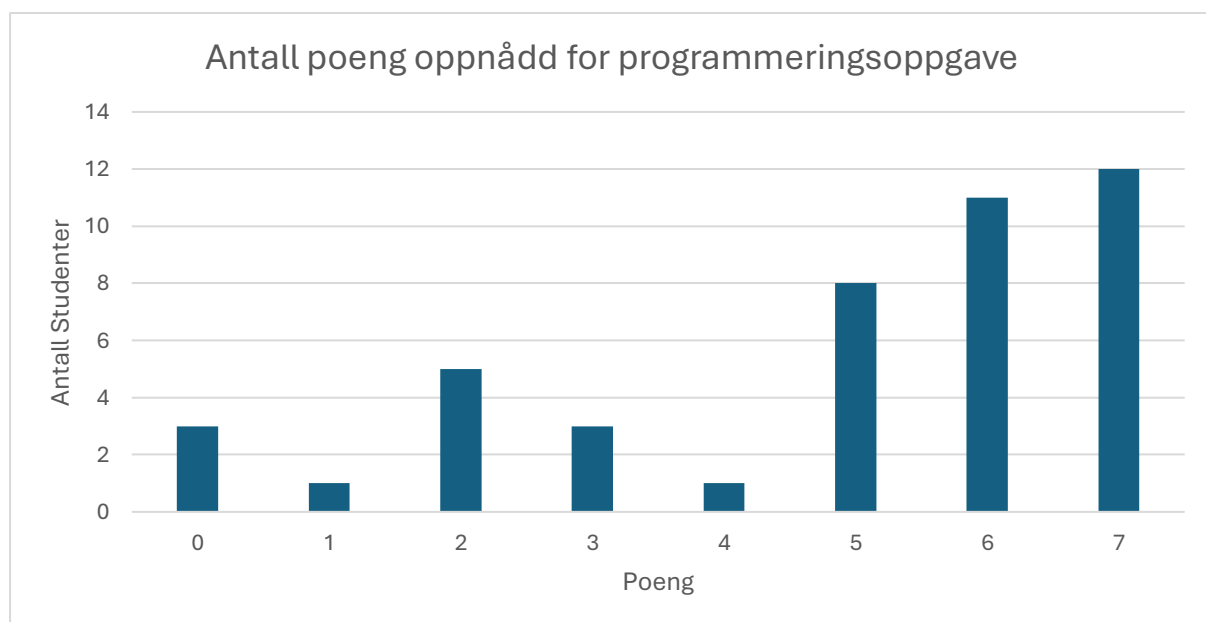
«Programmeringsoppgavene var også bra da jeg føler at de forbereder meg godt til videre studier»

«Det har vært spennende og læringsrikt å jobbe med programmeringsoppgaver kombinert med tradisjonelle oppgaver.»

Tabell 1. Eksamensresultater PHYS113 ved UiB siste 9 år

År	Antall studenter	A	B	C	D	E	F	Gjennomsnitts Karakter	Andel stryk	Andel A
2016	58	6	11	23	9	3	6	2.83	10%	10%
2017	73	10	14	21	12	9	7	2.77	10%	14%
2018	80	19	17	22	10	6	6	3.19	8%	24%
2019	79	7	19	22	19	5	7	2.78	9%	9%
2020	90	13	16	40	9	6	6	3.03	7%	14%
2021	49	4	6	13	11	9	6	2.33	12%	8%
2022	79	1	8	26	18	9	8	2.03	10%	1%
2023	50	6	13	10	7	10	4	2.72	8%	12%
2024	44	3	5	12	14	5	5	2.36	11%	7%

Eksamensresultatene er vist i Tabell 1, og vi ser at resultatene ikke skiller seg nevneverdig ut fra tidligere år. Gjennomsnittskarakteren, andel A besvarelser, og andel stryk samlet fra 2016 – 2023 er henholdsvis 2.7 ± 0.4 , $12\% \pm 7\%$ og $9\% \pm 2\%$. Resultatene for 2024 er altså innenfor et standardavvik fra de tidligere 8 årene.



Figur 6: Antall studenter som oppnådde ulike poengscore på programmeringsoppgaven på eksamen Høst 2024.

I Figur 6 vises scoren på programmeringsoppgaven, der 7 poeng er beste mulige resultat. 31 av 44 studenter som avla eksamen fikk 5 poeng eller bedre, og gjennomsnittlig poengsum er 4.9 poeng. Dette viser at dette ikke ble en snubleoppgave for studentene, og den har heller ikke hatt en signifikant påvirkning på strykprosenten i emnet. Om noe så har den dratt snittet opp – ikke ned.

Tabell 2: Resultater av faktatest, konseptuell flervalgstest og eksamen for årene 2022 – 2024. Tabellen viser midlere poengscore per år [M]. Resultatene er basert kun på studenter som har akseptert at dataene benyttes til forskning

	2022 [M]	2023 [M]	2024 [M]
Faktatest	17.0	20.0	17.5
Konseptuell flervalgstest	13.5	12.6	11.2
Eksamen	64.2	71.6	62.4

Notat. *** $p < .001$, ** $p < 0.01$, * $p < 0.05$, ^a $p < .10$

Tabell 3: Sammenlikning av resultater av faktatest, konseptuell flervalgstest og eksamen for 2024 vs 2022 og 2023 for samme utvalg av studenter som i tabell 2. Tabellen viser antall standardavvik [d] forskjell mellom 2024 og de andre årene. Er $d = 0.2$ så er det liten effekt, $d=0.5$ medium effekt og $d=0.8$ stor effekt. Et positivt tall betyr en forbedring

	2024 vs 2022 [d]	2024 vs 2023 [d]
Faktatest	0.15	-0.91**
Konseptuell flervalgstest	-0.42	-0.23
Eksamen	-0.12	-0.53*

Notat. *** $p < .001$, ** $p < 0.01$, * $p < 0.05$, ^a $p < .10$

Tabell 2 og Tabell 3 viser resultatene fra faktatesten, flervalgstesten og eksamen fra årene 2022 til 2024. Sammenlignet med 2023 er resultatene i 2024 generelt svakere, mens forskjellene fra 2022 er mindre markante.

Ved innføringen av programmering i PHYS113 var det en forventning om at dette potensielt kunne påvirke læringsutbyttet i fysikk, spesielt siden studentene i 2024 ikke hadde hatt nytte av en fullstendig integrasjon av programmering i de tidligere kursene. Likevel er effektene ikke dramatiske, og det er usikkert om nedgangen i 2024 skyldes programmeringen, naturlige svingninger, eller andre faktorer.

I faktatesten presterte studentene i 2024 signifikant svakere enn i 2023 ($d = -0.91$, $p < 0.01$), men det var ingen signifikant forskjell fra 2022 ($d = 0.15$). Den konseptuelle flervalgstesten viste en mindre nedgang fra både 2022 ($d = -0.42$) og 2023 ($d = -0.23$), men uten statistisk signifikans. Eksamen viste en signifikant reduksjon fra 2023 ($d = -0.53$, $p < 0.05$), men liten forskjell fra 2022 ($d = -0.12$).

Disse resultatene tyder på at studentenes prestasjoner er noe redusert fra de to foregående årene, men uten at dette entydig kan tilskrives programmeringsintegrasjonen. Videre analyser vil være nødvendig for å forstå årsakene til disse endringene.

4 Diskusjonsspørsmål

Hvordan kan vi sikre at programmering i fysikkundervisningen styrker fysikkforståelsen, og ikke bare blir en ekstra teknisk ferdighet studentene må lære?

Hva er de største utfordringene med å integrere programmering i fysikkfag, og hvordan kan vi overkomme dem?

Hvilke strategier kan vi bruke for å opprettholde og videreutvikle studentenes programmeringsferdigheter gjennom hele bachelorløpet?

Takk til

Universitetet i Bergen for finansiering av prosjektet via Pilotprosjekter for redesign av studieprogram, ref 2022/2735 og insentivmidler, ref 23/2414

Referanser

- Malthe-Sørenssen, A., Hjorth-Jensen, M., Langtangen, H. P., & Mørken, K. (2015). Integrasjon av beregninger i fysikkundervisningen. *Uniped*, 38(4), 303–310. <https://doi.org/10.18261/ISSN1893-8981-2015-04-06>
- Orban, C. M., Teeling-Smith, R. M., Smith, J. R. H., & Porter, C. D. (2018). A hybrid approach for using programming exercises in introductory physics. *American Journal of Physics*, 86(11), 831–838. <https://doi.org/10.1119/1.5058449>
- Svensson, K., Eriksson, U., & Pendrill, A.-M. (2020). Programming and its affordances for physics education: A social semiotic and variation theory approach to learning physics. *Physical Review Physics Education Research*, 16(1), 010127. <https://doi.org/10.1103/PhysRevPhysEducRes.16.010127>