Programmering – for fysikkens skyld

Programmering er på vei inn i skolen og det er mange som argumenterer både for og imot dette. I denne artikkelen vil vi vise hvordan programmering kan styrke fysikkfaget og hvordan det kan endre fysikkfaget.

Andreas D. Haraldsrud Valler vgs. og CCSE, Fysisk institutt, UiO
Cathrine W. Tellefsen CCSE, Fysisk institutt, UiO

Elever i videregående skole i dag har hver sin datamaskin med en regnekraft som tilsvarer en superdatamaskin for 10 år siden. Da bør vi tenke nøye gjennom bruken av den. Hva kan datamaskiner gjøre bedre enn mennesker? Hva kan mennesker gjøre bedre enn datamaskiner? Hvilken kompetanse trenger elevene for fremtiden?

Introduksjon

Grunnleggeren av Apple, Steve Jobs, har sagt: «Everybody in this country should learn how to program a computer, because it teaches you how to think.»

For å gi instrukser til en datamaskin må man analysere en problemstilling og bygge opp instruksjonene i en logisk rekkefølge – ikke ulikt hvordan vi jobber i både matematikk og fysikk. Det er gjerne tre hovedargumenter som blir brukt for matematikk i skolen;

1. matematikk er en viktig del av allmenndannelsen
2. du lærer en måte å tenke på som er svært kraftfull og derfor verdifull
3. du trenger matematikken i mange andre fag, som f.eks. fysikk.

Alle disse argumentene gjelder også for programmering. Og så kommer det inn et ekstra argument. I skolefysikken, og for så vidt også i begynneremner på universitetsnivå, blir ofte de fysikkfaglige problemstillingene bestemt av hva vi klarer å løse matematisk. Og da mener vi analytisk matematisk. Det fører igjen til et faglig innhold som er begrenset av elevenes matematiske ferdigheter eller av hva som kan løses matematisk. Med programmering forsvinner mange av disse begrensningene, og man kan arbeide med problemstillinger som er mer virkelighetsnære og interessante for elevene. I tillegg kan dette åpne for mer kreativitet og større fokus på drøfting, tolking og vurdering av fysiske systemer og modeller som beskriver disse systemene. Med dette perspektivet kan kompetanse i programmering åpne for at flere elever kan beherske og glede seg over fysikken.

Fysikkfaget i skolen

I skolen undervises fysikk som eget fag i 2. og 3. klasse på videregående skole (VG2 og VG3). På VG2-nivå heter faget Fysikk 1 og har et allmenndannende perspektiv. På VG3-nivå heter det Fysikk 2 og er studieforberedende [1]. Det er vesentlig flere elever som tar Fysikk 1 enn Fysikk 2, se tabell 1. Litt i underkant av 30 % av elevene som tar Fysikk 2, er jenter.

Det viser seg at jenter legger mer vekt på nytten av realfag enn det guttene gjør når de skal velge fag [2]. Når vi vet at Fysikk 2 ikke er et krav for å komme inn på noen studier i Norge, og vi samtidig vet at det er ansett som et vanskelig fag, kan dette forklare noe av den lave andelen av jenter i Fysikk 2. Hvordan kan i så fall introduksjon av programmering i fysikkfaget slå ut? Hvis programmeringen anses som en svært nyttig kompetanse, vil dette øke «nytteverdien» for jentene? Det kan hende at muligheten for å modellere og kreativt utforske mer virkelighetsnære problemstillinger vil være motiverende og åpne for at flere elevtyper vil lære fysikk.

I videregående opplæring lærer elevene differensiallikninger i matematikk R2 [3]. Det er ikke krav om R2 for å ta Fysikk 2. Det betyr at fysikkfaget ikke omfatter noen tema som fordrer løsing av differensiallikninger. Det er mange systemer i naturen som handler om endring og som derfor

Tabell 1. Antall elever i videregående skole som velger Fysikk 1 og Fysikk 2 (Utdanningsdirektoratet, 2018).

Skoleår	2015–2016	2016–2017	2017–2018
Fysikk 1	9 566	9 042	9 297
Fysikk 2	4 507	4 201	3 892

kan modelleres ved hjelp av differensiallikninger, men elevene må vente til de kommer på universitetet for å kunne modellere dette. Dersom elevene kan programmere, kan vi derimot modellere disse systemene allerede på videregående. Hvordan kan dette endre fagene og spesielt fysikkfaget i skolen?

I det følgende skal vi se på noen eksempler på hvordan programmering kan styrke skolefysikken. Vi vil vise hvordan dette har blitt gjort i programfaget programmering og modellering [4], og vi vil diskutere hvordan dette kan inngå som en del av framtidas fysikkfag. Vi har valgt å bruke programmeringsspråket Python, men implikasjonene for fysikkfaget er uavhengig av programmeringsspråk.

Programmering og modellering

Skoleåret 2017/2018 er det første året faget programmering og modellering gjennomføres. Forslaget til faget og læreplanen kom fra Per Husum og Andreas Haraldsrud ved Valler vgs., og det tilbys som forsøksfag de tre neste årene. Det er fire skoler som har fått tillatelse til å teste ut faget i 2017/2018: Valler vgs. i Akershus og Bamble, Skien og Porsgrunn vgs. i Telemark. Høsten 2018 starter antakelig ytterligere over 40 skoler til med faget. Faget er et tretimersfag som elevene tar i tillegg til fordypning i matematikk og naturvitenskapelige fag, enten i VG2 eller VG3.

Faget inneholder en introduksjon til grunnleggende programmering og en anvendelse av programmering på naturvitenskapelige og matematiske problemstillinger. Det er hele tida realfag som fysikk, kjemi, biologi og matematikk som står i sentrum. Fokuset ligger på problemløsning og modellering av fenomener i naturen, og elevene skal lære å forholde seg kritisk og utforskende til modellene de bruker og lager.

Faget er et tretimersfag, og den grunnleggende programmeringen tar mesteparten av det første halvåret. Vi kan knytte dette til fysikk på flere måter. Her følger to eksempler på dette:

Eksempel 1: Variabler, datatyper og syntaks i fritt fall

Alle elevene har lært bevegelsesformelen for fritt fall:

$$s = v_0 t + \frac{1}{2} g t^2,$$

```
g = 9.81 # tyngdeakselerasjon i m/s^2
v_0 = 0 # startfart i m/s
t = 3.0 # tid i s

# Beregner fallstrekningen
s = v_0*t + 0.5*g*t**2

print("Gjenstanden falt", round(s,1),
      "meter på", t, "sekunder.",
      "Startfarten var", v_0, "m/s.")
```

Figur 1. Program som beregner strekning for gjenstand i fritt fall.

der s er posisjon, v_0 er startfart, t er tid og g er tyngdeakselerasjonen. I figur 1 nedenfor ser du hvordan dette kan se ut i et pythonprogram. Alt som står etter # er kommentarer og brukes til å gjøre programmet mer lesbart og forståelig. Kommandoen round er brukt til slutt for å få et hensiktsmessig antall gjeldende siffer i svaret.

Programmet skriver ut denne linjen:

```
Gjenstanden falt 44.1 meter på 3.0
sekunder. Startfarten var 0 m/s.
```

Med dette programmet kan elevene lett utforske og endre på tid, startfart og tyngdeakselerasjon (se hva som skjer på månen!).

Eksempel 2: Lese fra fil og plotte solflekker

Når elevene skal lære å lese data fra fil og plotte kan vi velge motiverende eksempler fra fysikken. I figur 2 ser du hvordan vi kan hente reelle data om solflekksykluser fra internett og plotte disse.

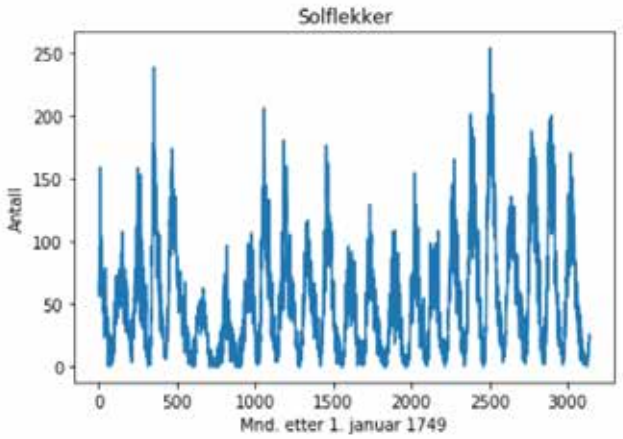
Eksemplene viser at vi med relativt enkel programmering kan undersøke empiriske data og fenomener på en litt annen måte enn før. Dessuten lærer elevene programmering ved å bruke naturvitenskap som eksempel, og det kan bidra til dybdelæring.

Det andre halvåret går med til en innføring i numerisk matematikk og en utvikling av naturvitenskapelige modeller ut fra denne matematikken. Numerisk derivasjon, integrasjon og likningsløsing blir gjennomgått og drøftet. Men selve kjernen i faget kan nok sies å være differensiallikningene. Elevene lærer hva slike likninger er, hva de betyr og hvordan vi kan bruke dem til å lage modeller av systemer som utvikler seg over tid. Fysikken

INPUT

```
# Importerer nødvendige funksjoner
from pylab import *

# Leser inn data fra filen sunspots.txt
data = loadtxt("sunspots.txt")

#Første kolonne er mnd. etter 1. januar 1749
aar = data[:,0]

# Andre kolonne er antall solflekker
solflekker = data[:,1]

plot(aar,solflekker)

title("Solflekker")
ylabel("Antall")
xlabel("Mnd. etter 1. januar 1749")
```

OUTPUT



Figur 2. Program som plotter antall solflekker ved ulike observasjoner.

blir dermed ikke lenger begrenset av at elevene (verken de med R1 eller R2) foreløpig ikke kan løse differensiallikninger analytisk. Det betyr for eksempel at vi kan undersøke avkjølingskurver og endre ulike parametere raskt og enkelt. Vi trenger heller ikke lenger å se bort fra luftmotstand og vi kan se på svingesystemer med demping og mye mer. Dessuten kan vi «leke» med modellene for å opparbeide en dypere forståelse for fysikken.

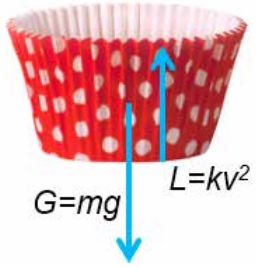
Vi viser her to eksempler på modellering ved differensiallikninger. Det første er et enkelt eksempel som elever som har tatt R2 i utgangspunktet kan løse analytisk, nemlig en gjenstand som faller med luftmotstand. Det andre eksempelet, kloss som henger i en fjær, kan ikke løses analytisk med reelle tall, og numeriske tilnærminger er her svært nyttig.

Eksempel 3: Fall med luftmotstand

«En kule faller fra det skjeve tårnet i Pisa. Startfarten er null. Hvor langt faller den på 1,2 sekunder? Se bort fra luftmotstand.»

Skolefysikken er full av denne typen oppgaver. Vi kan løse den ved hjelp av programmet i eksempel 1, men det er ikke alltid like realistisk å se bort fra luftmotstand. Vi kan derfor løse problemet også med luftmotstand og deretter sammenlikne med fritt fall. Elevene vil først måtte tegne opp et frilegemediagram slik de gjør i fysikk. I denne prosessen bør de vurdere hvilke krefter de skal ta hensyn til og hvilke modeller de skal bruke for disse kreftene. Disse modellene er utgangspunktet for differensiallikningen som skal løses. Vi bruker Newtons 2. lov, $\sum F = m \cdot a$, for å formulere disse likningene.

Figur 4. En muffinsform som faller, er påvirket av to krefter: tyngdekraften G og luftmotstanden L.



Først finner vi summen av kreftene. Her er $\sum F = G - L$ i vertikal retning hvis vi velger positiv retning nedover. Deretter må elevene velge en modell for kreftene. Modellen $G = mg$ er velkjent, men vi kan åpne for en diskusjon også rundt slike enkle modeller. Vi kan for eksempel diskutere hvordan modellen tar utgangspunkt i Newtons gravitasjonslov og deretter problematisere hvorvidt modellen stemmer godt dersom strekningen blir stor. Vi kan også diskutere om hvorvidt vi bør bruke modellen for luftmotstand ved lav fart, $L = kv$, eller for høy fart, $L = kv^2$. La oss her velge modellen for høy fart. Da får vi følgende sammenhenger:

$$\sum F = ma$$

$$G - L = ma$$

$$ma = mg - kv^2$$

$$a = g - \frac{kv^2}{m}.$$

Den siste likningen er en differensiallikning fordi a er den deriverte av farten, som også er den deriverte av strekningen: $a(t) = v'(t) = s''(t)$. Siden vi har å gjøre med den dobbelderiverte av farten, får vi en andreordens differensiallikning. Denne kan vi løse analytisk, men vi kan enklere (og kanskje mer intuitivt) også løse den numerisk. Da kan vi bruke Eulers metode.

Eulers metode tar utgangspunkt i at du kjenner den deriverte, men ikke selve funksjonen. Med initialbetingelser har vi en første funksjonsverdi – et punkt. Vi bestemmer den deriverte i punktet og følger tangenten et lite stykke, dt, og regner ut en ny funksjonsverdi. Så bestemmer vi den deriverte i dette nye punktet og følger tangenten et nytt lite stykke, dt:

$$v_{ny} = v_{gammel} + a \cdot dt.$$

Ved å lage en løkke i et program kan vi gjenta dette for svært mange små skritt. Det samme gjelder for strekningen, som igjen er den deriverte av farten.

$$s_{ny} = s_{gammel} + v \cdot dt.$$

Alternativt kan vi utlede metoden fra definisjonen av den deriverte, med utgangspunkt i at vi erstatter grenseverdien med en verdi dt som er liten nok:

$$f'(x) \approx \frac{f(x+dt) - f(x)}{dt}$$

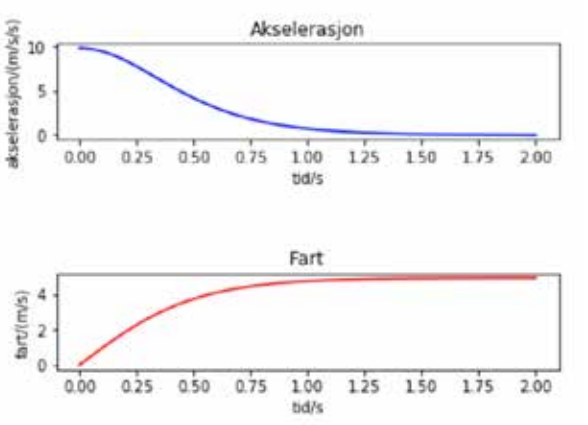
$$f'(x) \cdot dt = f(x+dt) - f(x)$$

$$f(x+dt) = f(x) + f'(x) \cdot dt$$

som vi kan kjenne igjen fra ovenfor. Hele programmet kan se ut som vist i figur 3 før plotting. Deretter kan vi velge hva vi vil plotte. Figur 5 viser plott av akselerasjon og fart.

Figur 3. Program som regner ut bevegelsen til fallende objekt med luftmotstand.

```
from pylab import *

# Fysiske størrelser
g = 9.81  # tyngdeakselerasjon i m/s^2
m = 0.5   # masse i kg
k = 0.2   # luftmotstandskoeffisient

# Tidsintervaller
N = 100000     # antall intervaller
tid = 2        # antall sekunder
dt = tid/N     # tidssteg

# Vektorer
a = zeros(N)  # akselerasjon i m/s^2
v = zeros(N)  # fart i m/s
t = zeros(N)  # tid i sekunder

#Initialbetingelser
v[0] = 0  #startfart

#Eulers metode
for i in range(N-1):
    a[i] = g - k*v[i]**2/m
    v[i+1] = v[i] + a[i]*dt
    t[i+1] = t[i] + dt
```



Figur 5. Plott av akselerasjon og fart i fall med luftmotstand.

Vi kan enkelt inkludere posisjon ved å bruke definisjonen av fart og Eulers metode på posisjonsvektoren:

```
s = zeros(N) #Lager posisjonsvektor
s[0] = 0 #Setter denne linjen inn som
initialbetingelse
s[i+1] = s[i] + v[i]*dt #legger denne
linjen inn i for-løkka
```

Med dette programmet kan elevene se hvordan ulike verdier for k og m påvirker bevegelsen. De kan for eksempel sammenlikne med eksperimenter og få en virkelighetsnær forståelse av hvordan naturvitenskapelig metode kobler teori, simulering og eksperiment i sin søken etter ny viten.

Eksempel 4: Kloss i fjær

Det klassiske eksemplet med en kloss som henger i ei fjær, passer fint for å introdusere fjærkrefter og harmoniske svingninger. Men en kloss henger ikke og dingler i fjæra til evig tid. Det er krefter som bremser bevegelsen. Dersom vi innfører

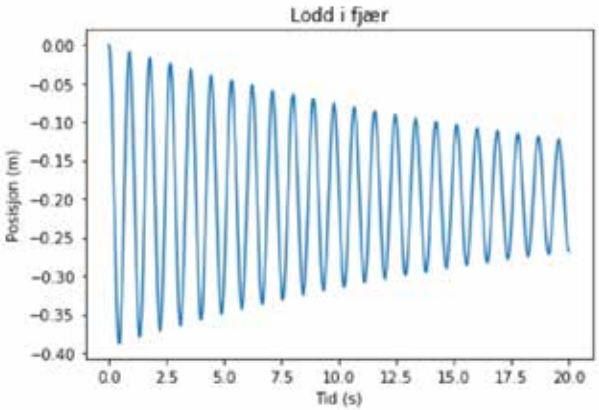
```
from pylab import *

# Konstanter og initalbetingelser
m = 1.0          # masse i kg
k_f = 50.0   # Fjærkonstanten i N/m
k_l = 0.2    # Luftmotstandskoeffisienten
g = 9.81         # Tyngdeakselerasjonen i m/s^2

# Tidssteg
tid = 20         # Tid i sekunder
N = 10000        # Antall iterasjoner
dt = tid/N       # Tidssteget

#  Vektorer
t = zeros(N)     # Tid
a = zeros(N)     # Akselerasjon
v = zeros(N)     # Hastighet
s = zeros(N)     # Posisjon

# Initialverdier
s[0] = 0         # Startposisjon i m
v[0] = 0         # Startfart i m/s


# Eulers metode
for i in range(N-1):
    a[i] = -k_f*s[i]/m - k_l*v[i]/m - g
    v[i+1] = v[i] + a[i]*dt
    s[i+1] = s[i] + v[i]*dt
    t[i+1] = t[i] + dt

plot(t,s)
title('Lodd i fjær')
xlabel('Tid (s)')
ylabel('Posisjon (m)')
```

Figur 6. Elevene klarer ikke å løse den matematiske difflikningen, men koden er relativt enkel å forstå. Løsningen av likningen er de fire linjene under #Eulers metode.

slike krefter, kan vi ikke lenger finne posisjonen til klossen analytisk. Men det lar seg lett gjøre numerisk, dersom elevene kan programmere.

Først må elevene drøfte hvordan systemet er satt sammen og hvilke krefter som virker. Dette gjør de ved å tegne systemet og tegne på kreftene. Dessuten må de diskutere hvilke kraftmodeller som kan brukes og hvilke fortegn disse modellene må ha. Dette er den samme framgangsmåten som brukes i fysikk, men som vi skal se, åpner programmeringen opp nye muligheter for drøfting og vurdering rundt denne prosessen.

Hvis vi velger origo der klossen er i likevektsposisjon og fjærkraften er 0, blir oppgava lettest å løse. Koordinatsystemet kan vi sette den «rette veien». Da har vi at fjærkraften er negativ når posisjonen over likevektspunktet er negativ (fjæra dytter loddet ned) og fjærkraften er positiv når posisjonen er negativ (fjæra drar loddet opp). Altså



Figur 7. Med programmering kan vi velge mer realistiske problemstillinger og drøfte systemer i naturen. Her er et plot av en dempet svingning.

har fjærkraften alltid motsatt fortegn av posisjonen, siden fjærkraften kan sies å være proporsjonal med posisjonen (Hookes lov). Kraftmodellen vår kan nå se slik ut:

$$\sum F = -F_{\text{fjær}} - G$$
$$\sum F = -kx - mg$$

Av dette kan vi få både en førsteordens og andreordens difflikning:

$$v'(t) = -\frac{ks(t)}{m} - g$$
$$s''(t) = -\frac{ks(t)}{m} - g.$$

Dette kan løses både analytisk og numerisk, og vi får fine, harmoniske svingninger. Men hva hvis vi innfører en kraft som demper svingningen? Antakelig er det vel realistisk å tenke at det som utgjør motstanden er en slags friksjon i fjæra, men la oss heller prøve med luftmotstand (vi kan eventuelt kalle den «dempekraften»).

La oss nå velge luftmotstand for lav hastighet, $L = kv$. Og la oss også kalle k-en noe annet, slik at den ikke forveksles med fjærstivheten. Vi kaller de to koeffisientene for k_f (for fjær) og k_l (for luft). Neste trinn blir for elevene å lage en programskisse for hvordan programmet kan løses. Dette er ofte en kreativ prosess der elevene trenes i algoritmisk tenkning og kommer med forslag til ulike løsningsmetoder. Koeffisientene setter de til en mer eller mindre tilfeldig verdi, men ofte opparbeider mange elever en intuisjon for hva de bør være for å få et realistisk resultat. Nå har vi følgende uttrykk for akselerasjonen:

$$a(t) = -\frac{k_f s(t)}{m} - \frac{k_l s(t)}{m} - g.$$

Modellen kan nå løses. Figur 6 viser utsnittet av et program som kan brukes for å finne hastigheten og posisjonen til klossen. Koden er nokså lesbar, også for de som ikke kan programmering. Programmet gir graf for posisjonen som vist i figur 6.

Når elevene har drøftet resultatene, kan de diskutere hvorvidt modellen var god, kanskje også med utgangspunkt i forsøk de gjør før eller i etterkant av simuleringen. De kan også systematisk variere koeffisientene for å finne ut hva de betyr for dynamikken i det fysiske systemet. Dette ligger det mye god fysikk i, og eleven opparbeider seg gradvis en «fysikkintuisjon» slik en fysiker oppnår etter mange år i faget. Forståelsen av at naturvitenskapen er et sett med teorier og modeller som hver har sine begrensninger og gyldighetsområder, kommer godt fram på denne måten. Dessuten kan det være motiverende å kunne eksperimentere med ulike modeller, for så å se resultatene nesten momentant. Da får elevene oppøvd en større grad av kreativitet og refleksjon som ikke alltid kommer til uttrykk i klassisk fysikk og matematikk.

Som vi har sett i de to eksemplene ovenfor, ligger fokuset på fysikken og tolkning av systemer i naturen ved hjelp av numerisk matematikk og programmering. Programmeringen brukes for å underbygge læring i fysikk, og til å se på mer virkelighetsnære og kanskje også mer motiverende problemstillinger. Ikke minst oppfordres elevene til å være kreative og utforske og lage egne modeller. Således utvikles både forståelse for og kompetanse i den naturvitenskapelige metoden.

Avslutning

Naturvitenskapelig programmering handler om mer enn det tekniske i å lage et dataprogram. Det handler om å bearbeide en problemstilling så den lar seg løse ved programmering, utvikle algoritmisk tenkning og bruke numeriske metoder for å studere fysikkens vidunderlige verden.

Mange av dagens eksempler i fysikk er begrenset til hva vi kan løse analytisk med matematikk. Med kompetanse i naturvitenskapelig programmering er mange av disse begrensningene borte. Hva gjør det med selve innholdet i fysikkfaget? Hvordan velger vi eksempler og faglige problemstillinger når elevene har hver sin datamaskin som kan gjøre 10^{12} beregninger per sekund? Vi har i denne artikkelen forsøkt å vise eksempler på hvordan selve faget kan endre seg med den digitale utviklingen, og hvorfor det er viktig at flere elevtyper tilegner seg denne kompetansen. Vi er i startfasen av en digital utvikling fra passive til aktive brukere. Vi må få med alle på denne utviklingen. Kunnskap er makt og denne kunnskapen kan ikke bare tilhøre noen få utvalgte.

Referanser

1. Utdanningsdirektoratet. «Læreplan i fysikk». Internettadresse: www.udir.no/kl06/FYS1-01 (2006).
2. M.V. Bøe. «Science choices in Norwegian upper secondary school: What matters?». *Science Education* 96(1), s. 1–20 (2012). doi: 10.1002/sce.20461.
3. Utdanningsdirektoratet. «Læreplan i matematikk for realfag». Internettadresse: www.udir.no/laring-og-trivsel/lareplanverket/finn-lareplan/lareplan/?kode=MAT3-01 (2006).
4. Utdanningsdirektoratet. «Forsøkslæreplan i programmering og modellering». Internettadresse: www.udir.no/kl06/PRM1-01 (2017).
5. S. Bocconi, A. Chioccariello og J. Earp. «The Nordic apporach to introducing Computational Thinking and programming in compulsory education». Internettadresse: www.itd.cnr.it/doc/CompuThinkNordic.pdf (2018).
6. CCSE. «Center for Computing in Science Education». Internettadresse: www.mn.uio.no/ccse/ (2016).
7. A.A. diSessa. «Computational Literacy and 'The Big Picture' Concerning Computers in Mathematics Education». *Mathematical Thinking and Learning* 20(1), s. 3–31 (2018). doi:10.1080/10986065.2018.1403544.
8. A. Malthe-Sørenssen, M. Hjorth-Jensen, H.P. Langtangen og K. Mørken. «Integrasjon av beregninger i fysikkundervisningen». *Uniped* 38(04), s. 303–310 (2015).
9. Utdanningsdirektoratet. «Fagfornyelsen». Internettadresse: www.udir.no/fagfornyelsen (2017).
10. Utdanningsdirektoratet. «Statistikkportalen». Internettadresse: www.udir.no/tall-og-forskning/statistikk/statistikk-videregaende-skole/ (2018).