Interaktiva föreläsningar i kurser inom beräkningsmekanik med Jupyter Notebooks

Jonas Lindemann, *LUNARC, Lunds Tekniska Högskola*

***Abstract* — Denna artikel beskriver hur idéer och koncept kring aktivt lärande har implementerats i kursen VSMN20 – Programutveckling för tekniska tillämpningar. Genom att använda Jupyter Notebooks har föreläsningarna gjorts interaktiva för studenterna. I föreläsningarna har vi också lagt in speciella interaktiva uppgifter för att engagera studenterna och på detta sätt få dem att öva och hämta fram de kunskaper som lärts ut.**

***Nyckelord* — Jupyter Notebooks, interaktiva föreläsningar, interaktiva uppgifter.**

I. Inledning

Avdelningen för byggnadsmekanik ger sedan många år tillbaka kursen VSMN20 – Programutveckling för tekniska tillämpningar [1]. I kursen får studenterna i projektform använda sina färdigheter inom byggnadsmekanik och speciellt finita element metoden för att implementera ett beräkningsprogram inom ett visst tillämpningsområde från beräkningsmodell, grafiskt gränssnitt till visualisering av beräkningsresultaten. Kursen består av ett antal föreläsningar och övningar under en läsperiod. Projektuppgiften är uppdelad i ett antal steg där varje steg är en inlämning. Sista inlämningen be-står av den kompletta programkoden samt en projektrapport som beskriver teori, användning och programmets uppbyggnad.

Kursen är mycket uppskattad då den uppmuntrar studenterna att själva utforska olika frågor kring utveckling av beräkningsprogramvaror som till exempel:

- Hur implementerar man beräkningsmodeller på ett sätt som gör att de kan an-vändas på ett effektivt sätt samt kan återanvändas i andra sammanhang.
- Hur skapar man ett grafiskt gränssnitt för en beräkningsmodell, så att en användare förstår hur man skall använda programmet och samtidigt hanterar eventuella fel på ett sätt så att användaren känner sig trygg att använda modellen.
- På vilka sätt kan resultaten från beräkningsmodellen visualiseras för att på ett effektivt sätt förmedla detta till användaren.

Föreläsningarna i kursen har traditionellt baserats på Powerpoint-bilder där kursmomenten gås igenom med exempel och bilder. Jag upplevde dock att det var svårt att under 2 timmar få studenterna att vara uppmärksamma och ta till sig föreläsningarna. Detta ledde till att jag började undersöka alternativ till de konventionella föreläsningarna som kunde involvera studenterna på ett mer aktivt sätt.

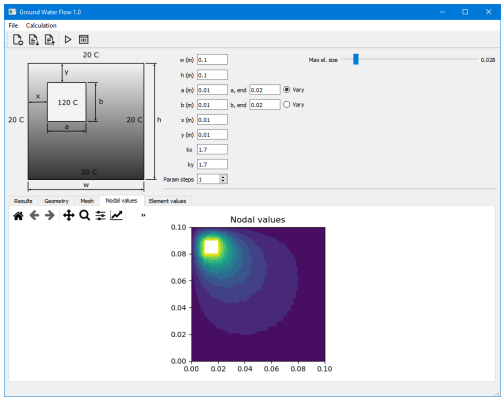

Fig. 1 - Exempel på program som utvecklas under kursens gång.

II. Pedagogiskt problem

Ett viktigt koncept i aktivt lärande är att ge studenterna möjligheter att öva/hämta fram på de kunskaperna som lärts ut (Retrieval), vilket stärker inlärningen [2]. Ett sätt att göra detta är att bryta av föreläsningarna med någon form av aktivitet som engagerar studenterna kring materialet [2].

Ett effektivt sätt att engagera studenterna i föreläsningar kring programmering är att använda live-coding [4]. Detta är ett pedagogiskt koncept där föreläsaren skriver kod direkt på en dator utan att använda någon form av färdig kod. Tanken med detta är att lära ut programmering som en process [5][6]. Metoden har visat sig mycket effektiv för att lära ut programmering på grundnivå [7]

För ett par år sedan blev jag introducerad till interaktiv/live undervisning genom Software Carpentry [8] initiativet. I deras kurser undervisas all programmering live där läraren genomför alla exempel på sin egen dator. Studenterna uppmanas också att följa med på sina egna datorer. För att alla skall hänga med används ett signalsystem med röda och gröna post-it lappar. Är man klar med uppgiften fäller man ner en grön post-it lapp. Behöver man mer tid eller hjälp fäller man ner en röd lapp. Konceptet är mycket intressant, men kräver också en hög grad av aktiva övningsledare, vilket ofta är svårt att implementera i det normala kursupplägget inom LU. Live-kodning kräver dessutom att alla studenter har samma utvecklingsmiljö som föreläsaren, vilket kräver en hel del merarbete för både studenter och föreläsare.

Hade det varit möjligt att hitta ett koncept som kombinerar live-kodning och på samma gång kunna erbjuda studenterna aktiviteter som engagerar dem under föreläsningen?

Centrat för tekniska och vetenskapliga beräkningar, LUNARC, vid Lunds universitet

Avdelningen för byggnadsmekanik vid Lunds Tekniska Högskola, Box 118, 221 00 Lund

jonas.lindemann@lunarc.lu.se

A. *Lösning*

Jupyter-Notebooks [9] är ett webbaserat verktyg i vilket man kombinerar text och kodexekvering i ett och samma dokument eller Notebook. Jupyter-Notebooks används också flitigt inom aktivt lärande [10]. Enligt A. Lorena et. Al [11] avsnitt 2.4.3 kan studenterna med hjälp av Jupyter-Notebooks fokusera mera på innehållet och koncepten än att fokusera på att föra anteckningar under föreläsningarna. Jupyter-Notebooks stödjer också en mängd metoder inom aktivt lärande. Verktyget kan användas på alla nivåer i utbildningen. För nya studenter som är i början av sin inlärning genom att ge stöd för hur de skall utföra en uppgift, men kan också användas för mer avancerade studenter för mer självständigt arbete.

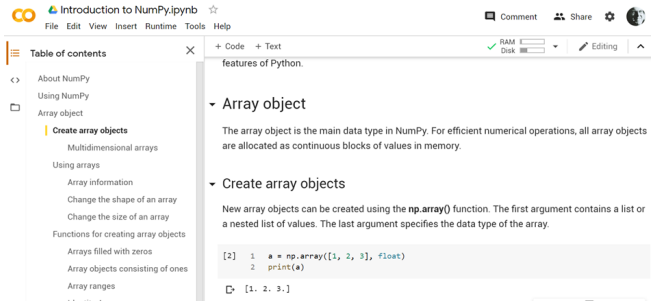

Fig. 2 - Exempel på Jypyter Notebook (Google Colab)

För att skapa interaktiva föreläsningar skulle Jupyter-Notebooks kunna erbjuda en enkel live-kodningsmiljö liknande den som används i Software Carpentry kurserna, fast med en mycket lägre instegströskel. Använder man till exempel en tjänst som Google Colaboratory [12] kan alla studenter på ett enkelt sätt ansluta till en interaktiv notebook genom en enkel webbaddress (URL) och på detta sätt kunna följa med på föreläsningen och utföra samma kodexempel som föreläsaren.

Under 2018 började jag att konvertera mina existerande Powerpoint-bilder till interaktiva notebooks. För de flesta föreläsningar går det utmärkt att konvertera, speciellt de som går igenom programmeringskoncept för det använda programmeringsspråket Python. Vi har också vidareutvecklat de programbiblioteket, CALFEM för Python [13], som används i kursen för att kunna göra diagram och figurer direkt i Notebooks. För vissa av momenten är det dock svårare att kombinera med Notebooks. Momenten som berör utveckling av grafiska gränssnitt i Python måste görs med live-kodning. Studenterna får dock ut dessa exempel så att de kan följa med på föreläsningen.

Interaktivitet i föreläsningen är ett första steg till att erbjuda ett mer aktivt lärande. Interaktiviteten kräver givetvis att studenterna själva har en vilja att följa med på föreläsningen. Vill man skapa tillfällen för interaktivitet måste man på något sätt bryta av föreläsningen och skapa naturliga tillfällen till interaktion. För att göra detta har jag gjort försök att lägga in interaktiva uppgifter i föreläsningen där studenterna prova att lösa ett enklare programmeringsproblem. Uppgiften består av en uppgiftsbeskrivning med ett efterföljande interaktivt område där de kan ange sin lösning i kod och också prova att köra koden och också se utdata från koden direkt. Här kan man be studenterna att beskriva sina lösningar innan man visar rätt svar. För att erbjuda studenterna ett facit används också en teknik inbyggd i Jyputer-Notebooks för att dölja ett interaktivt kodområde. I stället för att visa koden direkt visas en länk med ”Show code”. Klickar man på denna länk visas lösningen, som också kan köras. Detta erbjuder också en möjlighet att använda föreläsnings notebook:en självständigt.

I figurerna 3 och 4 visas hur detta koncept implementerat i en Jupyter Notebook.

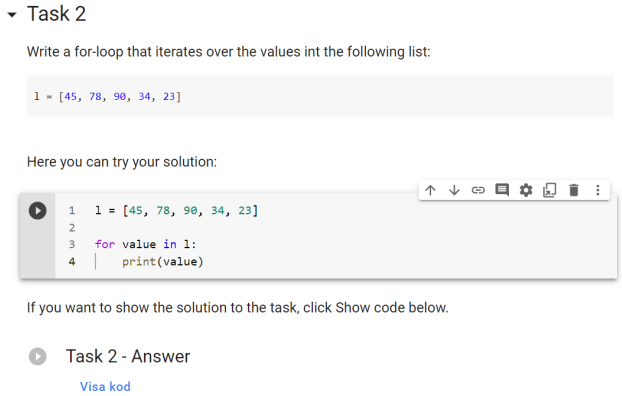

Fig. 3 - Exempel på interaktiv uppgift under föreläsningen.

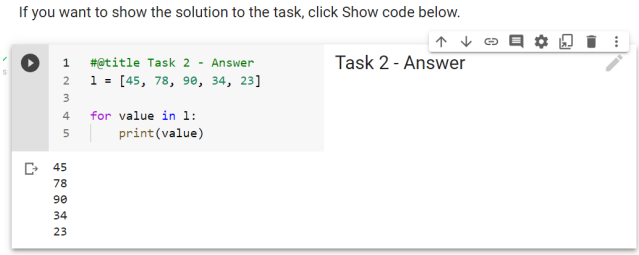

Fig. 4 - Dold lösning som visats.

När studenterna fått ett par minuter på sig att lösa uppgiften kan man ta en diskussion och reflektera kring om alla gjort uppgiften på samma sätt eller det finns andra alternativa lösningar. Föreläsaren kan sedan tillföra en diskussion kring fördelar och nackdelar olika lösningar på problemet.

En viktig aspekt är att uppgifterna är korta och berör ett eller ett få koncept, så det går att lösa uppgiften på några minuter.

III. Resultat och Reflektion

Den stora fördelen jag märkt med interaktivt föreläsningsmaterial är att studenterna är med aktiva och ställer fler frågor då de själva kan följa med och testa koncepten direkt. Detta behöver naturligtvis utvärderas djupare med hjälp av intervjuer och enkäter, vilket jag planerar att göra längre fram under våren. Personligt tycker jag att det är mycket roligare att föreläsa när jag direkt kan exekvera kod för att illustrera olika programmeringskoncept, till exempel genom att ändra en variabel och se vad effekten av ändring påverkar körningen av koden.

Jag provade att använda det utvecklade konceptet med interaktiva frågor under en föreläsning i Python-biblioteket NumPy med gott resultat. Även djupare analyser av detta kommer att göras när kursen kommer att ges under våren. Något jag observerat under användningen av Jupyter Notebooks är att studenterna är ofta mer uppmärksamma och ställer fler frågor under föreläsningen jämfört med konventionella föreläsningar med PowerPoint. Jag tror att anledningarna till detta är att det interaktiva materialet uppmunt-

rar till att prova frågeställningar och det är på så sätt också lättare att ställa frågor till föreläsaren som också kan exemplifiera med samma interaktiva material. Detta har även konstaterats av Paxton [4].

Genom att bädda in programmeringsuppgifter i Jupyter-Notebooks tror jag att interaktionen med studenterna kommer att öka. De blir också direkt konfronterade med de utlärda koncepten och får direkt möjlighet att själv prova och reflektera över resultaten från uppgiften. Även detta har visats i artikeln av Paxton [4].

Referenser

[1] VSMN20 - Programutveckling för tekniska tillämpningar. (2021). Retrieved from https://kurser.lth.se/kursplaner/21_22/VSMN20.html

[2] Yana Weinstein, C. R. (2018). Teaching the science of learning. Cognitive Research: Principles and Implications volume 3, p. Article number: 2.

[3] Dorothy Merritts, R. W. (2020). Interactive Lecture Demonstrations. Retrieved from https://serc.carleton.edu/introgeo/demonstrations/index.html

[4] John Paxton. 2002. Live programming as a lecture technique. Journal of Computing Sciences in Colleges 18, 2 (2002), 51–56.

[5] Jens Bennedsen and Michael E Caspersen. 2005. Revealing the programming process. In ACM SIGCSE Bulletin, Vol. 37. ACM, 186–190.

[6] Rex E Gantenbein. 1989. Programming as process: a “novel" approach to teaching programming. In ACM SIGCSE Bulletin, Vol. 21. ACM, 22–26.

[7] A Raj, J Patel, R Halverson and E Rosenfeld Halverson. 2018. Role of Live-coding in Learning Introductory Programming. In Proceedings of the 18th Koli Calling International Conference on Computing Education Research (Koli Calling '18). Association for Computing Machinery, New York, NY, USA, Article 13, 1–8. DOI:https://doi.org/10.1145/3279720.3279725

[8] Software Carpentry. (2020). Retrieved from Software Caprentry - Teaching basic lab skills for research computing: https://software-carpentry.org/

[9] Project Jupyter. (2020). Retrieved from Project Jupyter: https://jupyter.org/

[10] Tatman, R. (2018). Active learning with notebooks @Jupytercon. Retrieved from https://www.kaggle.com/rtatman/active-learning-with-notebooks-jupytercon/

[11] Lorena A. Barba, L. J. (2019). Teaching and Learning with Jupyter. Retrieved from https://jupyter4edu.github.io/jupyter-edu-book/

[12] Google Colaboratory. (2021). Retrieved from https://colab.research.google.com/

[13] CALFEM for Python. (2021). Retrieved from https://github.com/CALFEM/calfem-python

Exempel på föreläsning med uppgifter

https://bit.ly/python-intro-2021