

DELFT UNIVERSITY OF TECHNOLOGY

MASTERS THESIS

Flexible Authoring of Web-based, Open Answer Mathematics Exercises

Author:
Ruben KEULEMANS

Supervisor:
Marcus SPECHT

*A thesis submitted in fulfillment of the requirements
for the degree of Master of Science*

in the

Web Information Systems Group
Software Technology

June 21, 2021

Abstract

Ruben KEULEMANS

Flexible Authoring of Web-based, Open Answer Mathematics Exercises

Practice is central in mathematics skill acquisition. The practice process can be facilitated by flexible digital exercise systems, supporting personalized learning and providing students with parameterized, open answer exercises containing answer-specific feedback. However, current solutions for authoring these exercises lack efficiency and expressiveness, or rely on unpopular languages for content generation. To address these issues, this work demonstrates a flexible setup for authoring exercises in Jupyter Notebook cells using Python and the available libraries therein and evaluates this setup with expert users and authors of exercises.

Preface

This work is the result of my graduation project at the Web Information Systems Group for the Master Computer Science at the Delft University of Technology.

I considered this project as an opportunity to broaden my knowledge beyond computer science. After completing the first edition of the *Educational Technology* master specialization course in 2020, I was eager to learn more about education and learning. Following two months of exploring the educational domain, the thesis topic of mathematical exercise authoring was selected.

I am grateful to my supervisor Marcus Specht for providing me the freedom to investigate and explore, while providing feedback on scope and focus when necessary. Special thanks to Eric Bouwers from Grasple for his support. The constructive feedback and bi-weekly conversations were of great help in my learning process and improving this work.

Lastly, I want to thank everyone sharing their thoughts on this work throughout this project. These include the participants in the evaluation studies, Pim and Thijs from Grasple, members of the Centre for Education and Learning group of the Leiden-Delft-Erasmus collaboration, committee members and fellow students. An important lesson learned during this project is that continuously collecting suggestions from a broad range of people is central in shaping a solution.

*R. Keulemans
Delft, June 2021*

Contents

Abstract	i
Preface	ii
1 Introduction	1
1.1 Developments in Higher Education	1
1.2 Mathematics in STEM studies	3
1.3 Importance of practice in learning mathematics	5
1.4 Limitations of typical classroom practice	7
1.5 Initial research question	8
1.6 Methodology and approach	9
1.7 Contributions	10
1.8 Thesis structure	10
2 Related Work, Problem Statement & Authoring Requirements	12
2.1 GraspLe	12
2.2 Problem statement	18
2.3 Research question	19
2.4 Scope and context	19
2.5 Other solutions: MyOpenMath, RExams, WebWorX and MEGUA	20
2.6 Requirements and use cases	21
3 Setup for Flexible Exercise Authoring	24
3.1 Authoring setup context	24
3.2 A first exercise definition: One-off integer addition	26
3.3 Enabling repeated practice: Parameterized integer addition	27
3.4 High level of abstraction: SymPy code generation for matrix exercises	28
3.5 Guiding feedback	30
3.6 Architecture: Exercise player, JSON representation and answer evaluation	32
3.7 Complex exercises	35
4 Evaluation	40
4.1 Focus groups	40
4.2 First focus group feedback	41
4.3 Second focus group feedback	43
4.4 Setup usability evaluation	45
4.4.1 Exercise authoring tasks	45
4.4.2 Semi-structured interviews	46
4.5 Functional suitability	48

5 Conclusion & Future Work	52
5.1 Conclusion	52
5.2 Research limitations	53
5.3 Recommendations	53
5.4 Future work	53
5.5 Further directions	54
A Usability Evaluation Interview Questions	56
A.1 Understandability	56
A.2 Abstraction	56
A.3 Reusability	56
A.4 Learnability	57
A.5 Editing and collaboration	57
A.6 Miscellaneous	57
B Usability Evaluation Notebook	58
C Code Listings of Selected Exercises	64
Bibliography	69

Chapter 1

Introduction

This chapter begins with a high level perspective of the challenges and developments in the field of higher education relevant for the research in this work: the increasing demand of STEM (Science, Technology, Engineering, Mathematics) labour and the developments towards open and scalable education. Next, the central role of mathematics in STEM studies and corresponding issues for students are described. Informed by the importance of mathematics in STEM, the topic of learning mathematics is discussed, including the current and desired method of practice and instruction. Given these observations, an initial broad research question is formulated, to be further specified in Chapter 2. Finally, the methodology for answering this question is described and an overview of the conducted activities and contributions is provided.

1.1 Developments in Higher Education

Several important developments in Higher Education have an impact on the topics of this research. These developments include changes and requirements from the job market for increased qualifications on STEM subjects and up-scaling of education to accommodate this change by means of Massive Online Open Courses (MOOCs) and Open Educational Resources, including video-based learning materials and online learning opportunities.

STEM education as a key to innovation and engineering is becoming increasingly supported with digital tools. This work focuses on the potential of creating a setup for authoring parameterized exercises for online learning of STEM mathematics subjects in a flexible way.

Increasing demand of STEM labour STEM (Science, Technology, Engineering, Mathematics) occupations are primarily responsible for inventing and improving innovative technologies that increase global welfare and affect our daily lives. Consider the abundance of available food, global communication and information access offered by the internet and medical equipment for analyzing and curing the human body. These and many other offerings of modern society are owed to STEM practitioners. In the European Union, a growing demand for these STEM occupations is projected. A research report published in 2015 on past and projected STEM (Science, Technology, Engineering, Mathematics) occupations [6], requested by the European Parliament, states:

Employment of STEM professionals and associate professionals in the EU was approximately 12% higher in 2013 than it was in 2000.

Demand for STEM professionals and associate professionals is expected to grow by 8% between 2013 and 2025, whilst the average growth forecast for all occupations is 3%.

According to Shapiro et. al [32], the supply and demand of STEM graduates is a major concern:

The supply and demand of STEM graduates continues to be of major concern to industry leaders and policy makers due to the traditional importance of STEM graduates for technology-driven growth, and due to a major replacement need in the coming years as the senior STEM workforce gradually retires.

Open learning materials In 2001, Massachusetts Institute of Technology (MIT) started the OpenCourseWare project. OpenCourseWare is an online platform hosting the high quality learning materials created by MIT staff. This material is the exact same as offered to students enrolled at MIT. It includes recorded lectures, books written by instructors and assignments. The initiative enables any teacher or learner to take advantage of the supplied material, free of charge.

Publicly distributing learning materials created by universities, like MIT started in 2001, is nowadays becoming common practice at universities worldwide. This enables instructors to reference instead of recreate material and offers students choice in learning sources. However, adapting the material at hand is difficult, due to the container format (most often pdf) material is published in. Adaptation of material is important, as teachers want to shape material using their experience and subjects of interest, given a particular course and student population. Publicly sharing material is a major step in achieving open education, but the current container format of shared material limits the potential value.

MOOCs and scalable education Massive open online courses (MOOCs) are an approach to provide online, scalable education to anyone in possession of an internet-connected device, free of charge. MOOCs typically include short lecture videos, written explanations, quizzes that can be automatically assessed and a forum to discuss course content. The logging facilities offered by platforms hosting MOOCs, such as EdX or Coursera, allow teachers to track learner performance and identify difficulties. Due to the scalability of created content, automated answer assessment, data analytics and forums where learners help each other, there is a reduced need for expert human guidance, making learning efficient and accessible.

Educational videos With the advent of YouTube, anyone can become an instructor by uploading created content and everyone can become a learner by viewing this content. Social metadata like up/down vote ratio, comments and view count help learners to identify suitability and quality of the available content. Additionally, the algorithms implemented by YouTube aid in recommending new learning sources informed by the history of watched videos. Examples of YouTube channels containing informational videos of outstanding quality are:

- **3Blue1Brown**¹, having vivid visualizations providing intuition for many mathematical concepts;
- **Ben Eater**², providing instructional videos on how to build an 8-bit computer on breadboards from first principles (i.e. starting from atoms);
- **Khan Academy**³, providing contextualized explanations of mathematical theory.

Blogging Medium⁴ is an online open publishing platform, where users can write blog posts. Publishers and content ranges from hobbyists explaining how to build a go-kart⁵ to company representatives explaining the architecture of their open source self-driving car software⁶.

Educational material is often found on medium, including explanations of complex mathematical procedures like Principal Component Analysis⁷ and Gradient Boosting⁸. While Medium does not support mathematics notation, this is often circumvented by inserting images of the mathematical objects and formulas. The articles typically provide a contextualization of the discussed theoretical concepts and include detailed step-by-step visualizations, in contrast to abstract and dense information found on Wikipedia. Articles include links to other sources for additional background information.

A major limitation of Medium (and static text in general) for educational purposes, is that it is unclear whether the reader understood the information. The direction of information is mostly unidirectional, unless the reader explicitly takes action and performs additional Google searches. Testing the reader before moving on explaining further, dependent topics is not supported.

1.2 Mathematics in STEM studies

This section describes the relevance of mathematics in STEM studies and illustrates this by the mandatory mathematics courses in the bachelor studies at TU Delft. The shared learning goals of STEM mathematics are discussed next, including different perspectives on instruction for achieving these goals. The importance of mathematics is further stressed by considering the rise of artificial intelligence in STEM studies and the role of mathematics therein. Finally, two conclusions from educational research on STEM students are discussed. A body of research attributing STEM dropouts to insufficient prerequisite mathematical knowledge and a study concluding mathematics exercises to be the most stressful activity of freshmen students, further indicate the importance of mathematics and associated learning challenges.

Learning goals Mathematics is an integral part of STEM studies. To illustrate this, consider the fact that all 16 bachelor studies offered by TU Delft, at least include calculus (analysis), linear algebra and probability & statistics in their curriculum, except for Industrial Engineering and Architecture Engineering. The importance of mathematical thinking and reasoning is acknowledged by Bryan et al., listing it as an important skill in integrated STEM [5]. The following practices to achieve this skill as stated by the Common Core Standards Initiative⁹ are cited by Bryan et al.:

- Make sense of the problem and persevere in solving it;
- Explain the meaning of a problem and look for solution entry points;
- Reason abstractly and quantitatively;
- Decontextualize - create abstractions of a situation and represent it as symbols and manipulate;
- Construct viable arguments and critique the reasoning of others;
- Model with mathematics;
- Use appropriate tools strategically;
- Attend to precision;
- Look for and make use of structure;

- Look for and express regularity in repeated reasoning.

With respect to this, an important remark noted by Bryan et al. is that according to Devlin [12]: *Often times, school mathematics is about learning to think 'inside the box'; however mathematical thinking is about learning to think flexibly and 'outside the box'.*

The lack of learning to think flexibly has also been noted by Boaler [4]. Boaler compared the performance and student perceptions of two ways of teaching at two high schools in a 3-year case study, the open and the closed way. The closed way, at the time the predominant model according to Schoenfield [30], teaches textbook mathematical methods in an effective way by strictly adhering to the book content. As a result: *students developed an inert, procedural knowledge that was of limited use to them in anything other than textbook situations.* The open way shared some similarities with apprenticeship forms of learning, particularly because the students were introduced to new concepts and procedures only as part of authentic activities. This way of teaching put less emphasis on drilling procedures and more on relating to real world problems. As a result: *It seemed that the act of using mathematical procedures within authentic activities allowed the students to view the procedures as tools that they could use and adapt. The understandings and perceptions that resulted from these experiences seemed to lead to increased competence in transfer situations.* The observation that students taught using the open way were better able to use mathematics in transfer settings is attributed to three characteristics:

- A willingness and ability to perceive and interpret different situations and develop meaning from them [14] and in relation to them [19];
- a sufficient understanding of the procedures to allow appropriate procedures to be selected [35];
- and a mathematical confidence that enabled students to adapt and change procedures to fit new situations.

In conclusion, Boaler notes: *One important conclusion that I feel able to draw from this analysis is that a traditional textbook approach that emphasizes computation, rules, and procedures, at the expense of depth of understanding, is disadvantageous to students, primarily because it encourages learning that is inflexible, school-bound and of limited use.*

Increasing importance of machine learning in STEM, for which mathematics is required Machine learning techniques are becoming increasingly important in various STEM studies. The realization that a range of domain specific problems can be formulated as (supervised) learning optimization problems and the availability of sufficient data to learn from, opens many new opportunities. Mathematical theory and models form the fundamentals for these techniques, while application and experimentation certainly are important next to the theory. To be able to use and evaluate the available techniques for a problem at hand, a sufficient understanding of the underlying mathematical models is necessary. The theory required to understand these models depends on the used technique and ranges from linear algebra (required for k-nearest neighbour, linear regression, principal component analysis and support vector machines) to probability theory (for generative models) to (vector) calculus (for neural networks) to information theory (used in decision trees). This again signifies the importance of mathematics in STEM studies.

STEM dropouts attributed to insufficient mathematical knowledge Deeken et. al [10] observed that high dropout rates in STEM studies in various countries often originate from insufficient basic mathematical knowledge of incoming undergraduates, finding

evidence in multiple studies [7, 29, 15, 26, 23]. Addressing this issue is one of the main motivations for this work. By providing STEM students in higher education with more flexible and therefore also motivating ways of exercising mathematics in their application domain this knowledge deficiency can be complemented and learning improves.

Stress during mathematics exercises According to a recent study by Neumann et al. mathematical exercises in weekly assignments are experienced as the most stressful everyday activity of freshmen mathematics students [25]. This stress possibly originates from the changing nature of both the mathematical concepts (from procedures to proofs) and culture of learning (from supervised high school teaching to self regulated learning). To support this hypothesis, Neumann et al. cite the work of Martínez-Sierra et al. about emotions in linear algebra courses [21]. This work reports that solving mathematics problems is associated with multiple negative emotions ranging from disappointment and distress to fear. Particularly, failing to complete the (often mandatory) weekly homework assignment is perceived distressful as students related this to failing the exam.

1.3 Importance of practice in learning mathematics

Several learning theories stress the importance of practice in skill acquisition and competence development of procedural and conceptual mathematical knowledge. In the following, several aspects of practice and collected requirements from the educational literature on how good practice should be supported are discussed, including automaticity, deep processing of information, feedback and motivational factors.

Developing automaticity by repetition A metric for the skill level of procedural knowledge is fluency. Fluency is defined as *responding both accurately and quickly to a selected stimulus* [2]. According to Axtell et al., fluency can be increased by practice and sufficient fluency results in automation, defined as *the phenomenon that a skill can be performed with minimal awareness of its use*. The importance of practice to obtain automaticity is also stressed by Palmeri [27].

Sufficient automaticity of basic skills is a necessary condition for being able to solve more complex problems according to cognitive load theory [33]. The automated basic skills stored in long term memory allow the limited capacity of the working memory of the brain to be used to oversee the complexity of the problem, as opposed to being overloaded and constantly requiring information not present in working memory and losing track.

However, as noted by Lehtinen et al. [20], *the effects of instructional methods informed by cognitive load theory become less effective as a function of increasing expertise* [34], a phenomena known as the "expertise reversal effect". For learners with higher levels of expertise, a learning theory known as deliberate practice provides guidelines for improving learning. Lehtinen et al. studied the possible implications of deliberate practice on mathematics education. These include: (a) self-initiated practice of useful training, (b) specific, constructive and critical feedback, (c) practice at the edge of student competencies, (d) training on how to self-regulate and select more challenging activities in which they are likely to try and fail, (e) sustainable mathematics motivation regulation, enabling students to also be engaged in unenjoyable, but necessary, practice. However, the difficulties of implementing these are also considered: *Organizing deliberate practice in the classroom in such a way that students are continuously provided with tasks that are optimally challenging for each student, and pushing all students to work on or beyond the border of their current skills, may be too demanding for teachers in the regular*

classroom teaching situation without enabling learning environments such as well-planned digital systems.

Better retention of information by deep processing Processing information in a deep or meaningful way results in better retention as opposed to processing shallowly. This is studied by Craik & Lockhart, defining information processing depth as: *the meaningfulness extracted from the stimulus rather than in terms of the number of analysis performed upon it* [8]. Deeper analysis and processing of information leads to more persistent memory traces, resulting in improved learning and retention. An example of deep processing includes the semantic processing of information, for example by means of relating newly presented information to existing knowledge or by applying learnt knowledge. As Kirschner and Hendrick conclude in the chapter *How Deep is Your Processing* in the book *How Learning Happens* about this topic: *For most educational tasks, students will benefit from those strategies that specifically encourage them to extract the meaning of the to-be-remembered information. Thus if you want your students to learn well, and that means here that they not only know things, but they also have to do things with what they've learnt* [17].

Increasing motivation by experiencing success In the learning literature, the relation between motivation and skill acquisition is considered controversial. Kirschner and Hendrick state that motivation is no guarantee for learning [31]. There is no causal relationship indicating that motivation leads to better learning and performance. Neither is there a reciprocal relationship indicating that motivation leads to learning and learning in turn leads to motivation. The only proven relationship is that learning leads to motivation: *When we experience success, no matter how small that success is, it feeds our motivation to continue. For example, good maths performance has a significant positive effect on the intrinsic motivation of students for maths, but motivation for maths doesn't lead to better math performance* [13, 22].

Requirements from educational literature for steering the practice process Knowing that solving exercises is crucial for learning mathematics, one may wonder what the best-practices are regarding the instructional design of exercises. Several best-practices have been found in the scientific literature by Kirschner and Hendrick [31]. Some of these principles that are applicable to learning mathematics are identified:

- **Frequent feedback** *Several studies show firm evidence that innovations designed to strengthen the frequent feedback that students receive about their learning yield substantial learning gains* [17].
- **Interleaving** *Cognitive load can be reduced by interleaving content and problems as opposed to blocking. Blocking involves solving one type of problem at a time before the next (for example, "problem A" before "problem B" and so forth. The learning pattern formed looks like this: AAABBBCCC. Interleaving, in contrast, involves solving several related problems mixed up together. The learning pattern here looks like this: ABCBCAACB* [17].
- **Mastery learning** *Students should master one topic before learning new content. Mastery learning is an instructional approach where students are tested on material learned and if they get less than 90% in a test then they are given additional instruction on that material until they get over 90% or until they have "mastered" the content* [17].

- **Open answers** By requiring open answers, students are tested for answer generation instead of recall/recognition or guessing the correct answer, as is possibly the case when using multiple choice answers. When asked for answer generation, higher levels in Bloom's revised taxonomy of the cognitive domain [1] are covered.
- **Context or rationale** Finally, the discussed requirement for contextualization is identified by Boaler [4]. In case contextualization is infeasible, significant motivational improvement can be accomplished by providing a rationale for pure theoretical practice [16].

1.4 Limitations of typical classroom practice

The typical way of learning mathematics in a university setting is as follows. All students visit the same lecture, which consists of two 45-minute slots, in which they listen and take notes. After the lecture, students practice exercises from a textbook at home. This one-size-fits-all instruction does not account for different background knowledge levels and learning speed of students. As such, it does not provide the desired flexibility to conveniently implement the policies described above. Consider the paper or electronic books, being currently still the main source of exercise material in mathematics courses and the three principles discussed above. First, obtaining frequent feedback is inconvenient, because answers should be manually checked and are often not even included in the textbook. Furthermore, is inconvenient to get answer specific feedback by manually inspecting reference answers. Second, while interleaving can be applied in books, the interleaving is static and does not consider the prior knowledge level and current level of mastery. Finally, by using book exercises, measuring learning progress (including level of mastery) is an inconvenient, manual process. This possibly results in moving to new topics before sufficiently mastering other (dependent) topics, with inefficient learning as a consequence.

Computer-based exercises allow to overcome these issues because of their flexibility and automation. Feedback can be given immediately by automatically evaluating the given answer. Interleaving and mastery learning can be controlled by automatically tracking the learners performance. In addition, parameterization of exercises allows students to continue practicing until mastery is achieved.

Web-based approaches for learning mathematics A range of online platforms exists providing support for learning mathematics. To get an impression of the diversity of the solutions available, a selection is briefly discussed below.

- OpenStax provides complete, web-optimized, college level mathematics books¹⁰ without interactivity.
- Khan Academy¹¹ provides mathematics modules from kindergarten to college level including short videos, practice by both multiple choice and numerical-answer questions. Gamification elements are used to engage learners.
- Math is Fun¹² aims at explaining mathematics in a user friendly manner by means of providing worked out examples of easy, contextualized problem instances.
- Brilliant¹³ is a commercial platform for learning mathematics, next other topics. Major selling points of Brilliant are the interactive elements and animations helping the learner to develop a better understanding of the topic at hand.

- GraspLe¹⁴ is an online mathematics exercise platform used in higher education in The Netherlands. A detailed explanation of the available functionality for both content authors and students is provided in Chapter 2.
- SOWISO¹⁵ is a commercial platform allowing teachers to create mathematics learning material for high school and university students. It facilitates remote testing, learning analytics and specific answer feedback (e.g. the answer is correct, but can be further simplified).
- Slader¹⁶ is a platform providing detailed step-by-step answers of textbook exercises, typically provided by users. Answers are provided in a structured manner including used theory and intermediate steps. Social metadata is used to allow community rating of answers.
- StackExchange Mathematics¹⁷ allows learners to ask for help with respect to a mathematical problem. Fellow users can answer or up-vote answers of other users. In this way, the best answers propagate by collective rating. By openly sharing the problem statement and answers, the information can be reused by any learner.
- WolframAlpha¹⁸ can be used for solving user-defined mathematical problem statements and provides step-by-step explanations and visualizations of answer. While some functionality is freely accessible, step-by-step explanations are a paid feature. WolframAlpha includes an API for integrating the offered services in external applications.
- Jupyter Notebooks demonstrating mathematical concepts in books and course modules in relation to Python libraries implementing these. For example a series of notebooks covering the Linear Algebra course of Gilbert Strang from MIT OpenCourseWare¹⁹.

For some of these platforms, the available content is primarily human generated, while others rely on automated processing and generation of content. The utility for each of these tools for a learner highly depends on the educational setting, background knowledge and ability to self regulate and study independently. Some learners might use a selection of these to learn a new subject at their own pace, while others might use these when a college lecture falls short.

1.5 Initial research question

Informed by the current practice of learning mathematics and the limitations with respect to implementing the desired flexibility for steering the practice process, the following initial broad research question is formulated:

How can mathematics practice and the creation of exercises be supported in a flexible way?

This research question is further shaped in Chapter 2, after investigating the possibilities and limitations of related work.

1.6 Methodology and approach

To understand the context and the state of research an exploratory literature review and an analysis of related work is conducted first. Second, by talking to different stakeholders including software developers, mathematics course authors and teachers, needs are collected. Then, experimentation and prototyping informed by these needs is performed. Finally, focus groups and a usability study are used to collect suggestions for further refinements.

Exploratory literature review Prior to implementing a solution, sufficient scientific understanding of the educational background of practice in mathematics skill acquisition is necessary, because this affects the solution. To this end, an exploratory literature review is conducted, informed by initial sources provided by the thesis supervisor and snowball sampling from these sources.

Related work Existing online exercise solutions identified by various mathematics teachers participating in an open online workshop organized by GraspLe are briefly evaluated. Additional online search is performed to identify other solutions. Related work regarding the technologies composing the exercise format, authoring and player environments in these solutions are derived, including those of GraspLe.

Context analysis by conversational interviews Two conversational interviews with instructors at the TU Delft were conducted. Both stated the need for more expressiveness and efficiency when authoring exercises within GraspLe. The limitations of their current practices regarding instructional design and authoring of learning material are discussed, including suggestions for improvements.

Experimentation and prototyping Early in the thesis project, alongside problem and context analysis, small experiments with the available technologies are carried out to create minimal working prototypes. The final design is informed by the experiences of creating these prototypes.

Evaluation in focus groups About halfway throughout the project, the current solution is evaluated in two focus group meetings with instructors in Higher Education from different backgrounds. The goal of these meetings is to get feedback on the proposed solution and collect additional requirements.

Usability evaluation To get detailed feedback on the new flexible setup of authoring exercises and the Python classes facilitating in authoring exercises, a usability evaluation is performed. The evaluation procedure consists of having instructors perform four exercise authoring tasks, taking about 30 minutes, and a semi-structured interview about their experiences and suggestions afterwards, also taking about 30 minutes.

Functional suitability The software developed is reviewed in the light of ISO 25010: quality characteristics of a software product. ISO 25010 is part of the ISO 25000 series, known as *SQuaRE (System and Software Quality Requirements and Evaluation)*, having the goal of creating a framework for the evaluation of software product quality²⁰. This standard aids in systematically discussing the system aspects relevant to the research question and finding limitations. The quality characteristic functional suitability is discussed because of the primary focus of this project: extending exercise authoring possibilities.

Timeline A brief overview of the main activities is shown in Table 1.1. In the first months of this research project, considerable time is spent on topic orientation, refinement and learning about the educational domain, being mostly unknown to the author prior to this work.

1.7 Contributions

The main contributions as part of answering the research question by following the given methodology are as follows:

- Collection of requirements from authors and literature;
 - High level of abstraction: direct access to a scripting language and available libraries;
 - Contextualized, open answers exercises with direct and answer-specific feedback.
- Identification of shortcomings of existing solutions with respect to these requirements;
- Design and implementation of an experimental software system conforming to the identified requirements or allowing extension to support additional requirements, including:
 - An exercise editor setup using Jupyter Notebook and Python classes facilitating exercise authoring (definition, preview and export);
 - An exercise player responsible for web-based presentation of exercises, consisting of a formula editor to formulate answers and a feedback component;
 - An evaluation component responsible for evaluating user answers with respect to an exercise;
 - An exercise specification in JSON, allowing the exchange of exercises between components.
- Evaluation of the developed solution with authors;
- Three merged pull-requests to open-source projects used:
 - MathLive - *Add display = [] to example code to make code snippet valid* #796²¹
 - Arithmatex - *Update arithmatex.md* #1266²²
 - Phoenix LiveView - *Update documentation to recommend single root element in phoenix_live_component.ex* #1401²³, as a result of a bug identified in *LiveComponent appended instead of patched when parent changes* #1398²⁴

1.8 Thesis structure

This chapter covered the background and introduction to the topic of this work, including the initial problem statement and methodology. Chapter 2 covers related work with respect to digital mathematics exercises, limitations of these and use cases of exercises illustrating the need of flexibly authoring exercise content. Next, Chapter 3 demonstrates a new, flexible setup of authoring exercises. This setup is evaluated in Chapter 4, discussing the methodology and outcomes of the focus groups and usability evaluation in detail. Chapter 5 starts off with the conclusion, then discusses the limitations, future work and recommendations.

TABLE 1.1: Timeline of main activities

Month 2020 - 2021	Phase	Activity
November	Exploring education	<ul style="list-style-type: none"> • Exploratory literature review on initial topic: creating learning material in collaboration • Conversational interviews with two instructors and two managers
December	Context analysis and scoping	<ul style="list-style-type: none"> • Fix scope: authoring mathematics exercises • Further literature analysis (Chapter 1) • Experiments with available technologies
January	Detailed problem analysis, prototyping	<ul style="list-style-type: none"> • Presentation and discussion at Grasple • Literature review on importance of practice in learning mathematics (Chapter 1) • State-of-the-art analysis of technologies and approaches for creating math exercises (Chapter 2) • Experimenting with available technologies • Early prototyping of exercise format and player
February	Setup design and implementation	<ul style="list-style-type: none"> • Encapsulation of exercise abstractions in Python classes • Documentation and unit tests • Preparation of focus groups
March	Solution design and implementation, evaluation	<ul style="list-style-type: none"> • Further development of novel authoring setup (Chapter 3) • Presentation and discussion at Grasple • Focus groups (2x) (Chapter 4) • Preparation of usability evaluation
April	Evaluation	<ul style="list-style-type: none"> • Semi-structured interviews for usability evaluation (3 users) (Chapter 4) • Prepare Grasple player compatibility of exercises • Data analysis
May	Writing	<ul style="list-style-type: none"> • Minor adjustments to exercise player and format • Data analysis and writing
June	Wrap up	<ul style="list-style-type: none"> • Minor adjustments to exercise player and format • Writing
July	Defence	<ul style="list-style-type: none"> • July 2nd 11.00 - defence

Chapter 2

Related Work, Problem Statement & Authoring Requirements

This chapter describes the related work regarding online mathematics exercise solutions and the limitations of these. From these limitations, additional authoring requirements are identified and the corresponding problem statement addressed in this work is derived. Finally, multiple example exercise topics in need of the identified requirements are illustrated.

2.1 Grasple

Introduction Grasple is an online mathematics exercise platform used increasingly in courses containing mathematics in higher education in The Netherlands. It allows students to practice mathematics exercises online in their browser, providing guided feedback on erroneous answers toward the correct solution, based on answer characteristics. Authors can create learning modules containing slides with explanation and exercises using the provided online editing environment. Features of the exercises and authoring environment are further discussed next.

Exercise content Grasple allows mathematics instructors with no programming experience to author mathematics exercises using a WYSIWYG editor to edit exercise content. To author mathematical notation as part of the exercise content, a WYSIWYG formula editor is provided. Consider authoring the content for the following integer addition exercise.

Question What is $1 + 1$?

Creating this exercise within Grasple can be achieved by typing the sentence in the text field. Mathematics notation can be inserted at the current cursor position by clicking the formula editor icon. Clicking this icon reveals the formula editor and inserts a math field at the current cursor position, see Figure 2.1. This math field can be populated with mathematical notation by clicking the concepts in the formula editor GUI or by typing LaTeX code directly into the field.

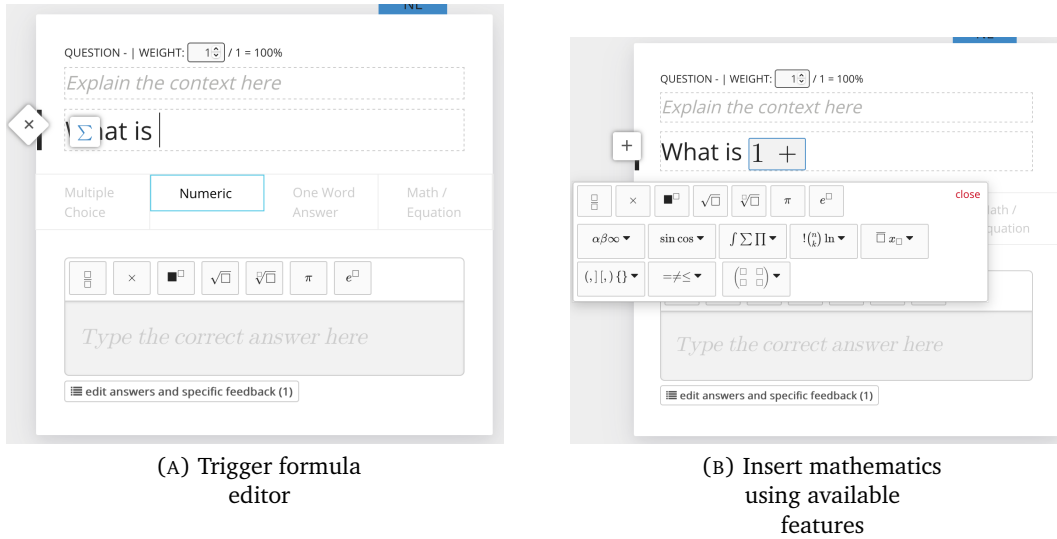


FIGURE 2.1: Insert mathematics notation field and populate with content

Next to using plain text and mathematics notation, styling text, inserting images, tables, lists and links are supported, see Figure 2.2.

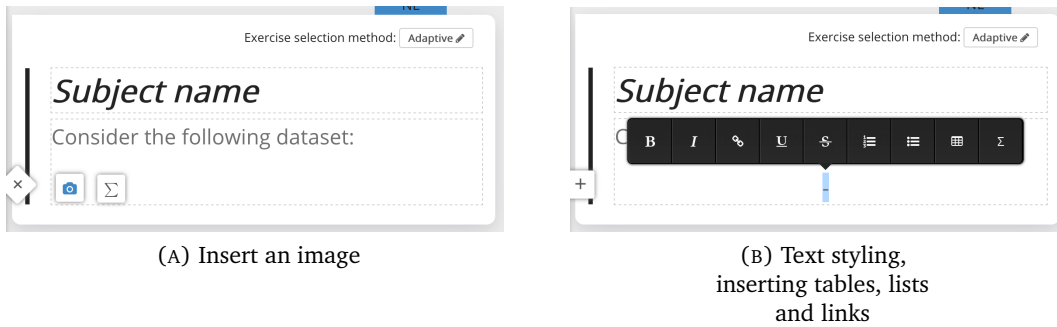


FIGURE 2.2: Additional exercise content editing features

Finally, 2-dimensional graphs can be added to illustrate problem statements formulated in exercises. For example, an exercise asking the learner to compute the intersection point of two lines is shown in Figure 2.3.

EN NL copy from NL

QUESTION - 48505 | WEIGHT: 1 / 1 = 100%

Explain the context here

Compute the intersection(s) of $2x + 3$ and $-2x - 2$

Edit graph remove graph

Line settings

$2x + 3$ formula ● label remove

$-2x - 2$ formula ● label remove

+ add line

Graph settings

x-axis $-10 < x < 10$ label

y-axis $-10 < y < 10$ label

hide legend

(A) Configuring a graph

QUESTION 1 SKIP

Compute the intersection(s) of $2x + 3$ and $-2x - 2$

$2x + 3$ $-2x - 2$

10.0
7.5
5.0
2.5
0.0
-2.5
-5.0
-7.5
-10.0

x

x y

x □ √ √ π e

Check my answer

(B) Exercise as presented to learner

FIGURE 2.3: Graph configuration and presentation

Exercise answers GraspLe currently supports four answer types, for each of which an example is shown in Figure 2.4:

- **Multiple Choice** In the supported multiple choice answer type, one answer out of the available options should be selected by the learner. The author can mark multiple answers as correct and provide specific feedback for each answer. Answer options can include mathematics, text and images or a combination of these.
- **Numeric** The numeric answer type allows for numerical answers, like 7.83, or statements that can be evaluated to numerical answers, like $\sqrt{2}$ or $e^{\frac{2}{3}}$. This type is different from the Math / Equation type, because it allows for a minor deviation between the student and reference answer.
- **One Word Answer** A one word answer is used to test whether a student correctly recalls the terminology of a certain concept. The answer can actually contain more than one word. Similar to the multiple choice question type, the author can

configure specific feedback for each answer. A matching accuracy can be configured to accept similar words as being correct.

- **Math / Equation** This answer type allows learners to type (complex, symbolic) mathematical answers. This answer type can be used to ask for exact analytical solutions.

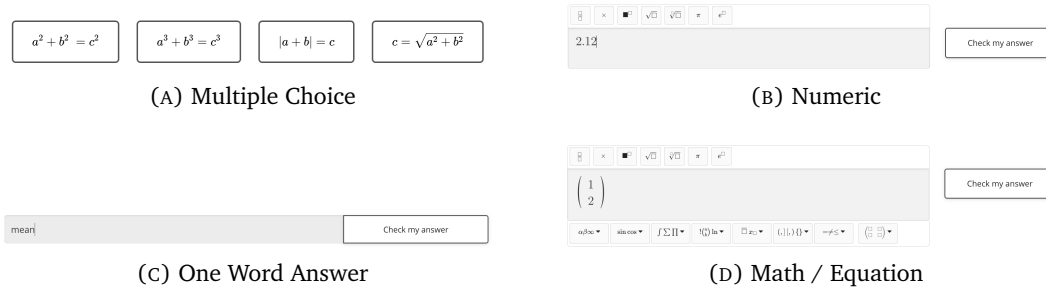


FIGURE 2.4: Answer types in Grasple

Both the Numeric and Math / Equation answer types also support answer-specific feedback. For the Numeric type, the answers to compare the user answer against are numeric values or expressions that evaluate to numeric values. The Math / Equation answer type is similar to the Numeric type, however, in addition, functions can be applied to the given student answer to conveniently provide specific feedback for a range of possible answers. For example, the feedback for any answer larger than 5 might state that this cannot be correct, given some argumentation about the exercise values and corresponding mathematical theory. The possible values for configuring answer specific feedback are listed in Figure 2.5. Answer specific feedback for Math / Equation answers are referred to as answer rules. Answer rules are important for students, as these can contain guiding hints given their answer, without revealing the solution.

Add answer rules and specific feedback for student.answer1 ✕ Close

Here you can specify answer rules for this specific sub-question.

- An answer rule is used to mark the answer as correct or incorrect (and optionally to provide specific feedback to the student).
- Answer rules are evaluated in order from top to bottom until a match is found. If no match is found, the answer is marked as incorrect and the default feedback is given.
- Save (changes to) answer rules by saving the exercise.
- [Read more about the many options in specifying answer rules here.](#)

Answer rules for marking and specific feedback

rule id: 55512 remove rule

how to use Algebraically an algebraic expression or matrix

student.answer1 is Algebraically Equivalent to $a - b$

add clause incorrect

specific feedback (optional) then and

Did you compute $a - b =$ instead?

FIGURE 2.5: Configuring answer rules

Parameterization To create multiple instances of an exercise, each having different values and answers, parameters can be used. Grasple allows configuration of parameters by a selected set of functions. The integer addition exercise can be parameterized using the configuration in Figure 2.6, using the Range function. The parameters can be inserted as part of the exercise content using the *parameters* dropdown menu in the formula editor. Parameters can also be used in answer rules to (a) express answers and (b) write answer feedback, see Figure 2.7. Parameterized exercises allow repeated practice of mathematical definitions or procedures at hand. Knowing that repetition is crucial in developing automaticity, using this feature greatly benefits students.

QUESTION - | WEIGHT: / 1 = 100%

Explain the context here

What is $a + b$?

Multiple Choice **Numeric** One Word Answer Math / Equation

$a + b$

edit answers and specific feedback (1)

Detailed solution

Describe a detailed solution here

Question at wrong answer

Provide feedback for a wrong answer here

+ Add Subquestion

Parameters | learn more about parameters and operators + new parameter

name	type	settings	options
...	a	Range min <input type="text" value="0"/> step <input type="text" value="1"/> max <input type="text" value="10"/>	<input type="radio"/> remove parameter <input type="button" value="+ add condition"/>
...	b	Range min <input type="text" value="0"/> step <input type="text" value="1"/> max <input type="text" value="10"/>	<input type="radio"/> remove parameter <input type="button" value="+ add condition"/>

FIGURE 2.6: Parameterized integer addition

Add extra answers and specific feedback Close

Here you can specify correct or incorrect answers for this specific sub-question.

- The specified answers are used to mark the student's answer as correct or incorrect (and optionally to provide specific feedback to the student).
- The specified answers are evaluated in order from top to bottom until a match is found. If no match is found, the answer is marked as incorrect and the default feedback is given.
- Save (changes to) these answer by saving the exercise.
- Read more about this here.

Answer	correct?	Specific Feedback	Threshold
$a + b$	<input checked="" type="checkbox"/>	Type your text	<input type="text" value="1"/> <input type="button" value=""/>
$a - b$	<input type="checkbox"/>	Did you compute $a - b = 0$ instead?	<input type="text" value="1"/> <input type="button" value=""/>

+ add additional answer

FIGURE 2.7: Answer specific feedback for parameterized integer addition exercise

Grasple functionality Grasple largely fulfills the requirements identified from educational literature for learning mathematics. The web based exercises provide students an accessible way to practice, with frequent, specific feedback on open answer exercises. Parameterized exercises in addition enable repeated practice of similar exercises, required to achieve automaticity. By automatically collecting responses, student performance can be evaluated and the learning path can be personalized, satisfying the requirements of mastery learning and interleaving. The rising amount of exercises available on Grasple provides the opportunity to select exercises that match the student level, allowing them to successfully complete exercises feeding their motivation to continue learning.

However, the current setup limits the efficiency and expressiveness of authors with respect to creating exercises, ultimately affecting the richness and diversity of exercises available for students. These limitations are discussed in detail next.

2.2 Problem statement

The available functionality within GraspLe allow authors with no programming knowledge to construct exercises. Since the number of available functions is limited, used functions in similar learning material are likely to be known by authors of this material, or can be understood easily. This allows authors to understand, edit and reuse learning material created by others and furthermore enables collaboration between authors. Two consequences of this design choice form the basis of this thesis.

- **Efficiency** The small set limits the efficiency of authoring exercises. For example, consider a matrix multiplication problem of two $n * n$ matrices containing random integers values between 0 and 10. In GraspLe, modeling this problem would require $2n^2$ parameters for the random integer values in both matrices. For two 3 by 3 matrices, the author must configure 18 parameters using the Range function with the exact same settings: generate a value between 0 and 10. One can imagine that using a scripting language, the same result can be achieved with less repetition by using the readily available (high-level) abstractions in libraries or by creating new functions.
- **Expressiveness** The small set of operations limits the expressiveness. Consider the same matrix multiplication problem described above. To optimally support practice of different problem instances, the dimensions of the matrices need to be parameterized. Unfortunately, this is not possible within GraspLe. The only possibility for achieving this, is crafting exercises with different matrix dimensions by hand, which is quite cumbersome and hardly matches the diversity of exercises that could have existed when parameterizing the dimensions. This potentially restricts the available learning material for students. Again, one can imagine that by using a scripting language, the desired exercise can be created, since it offers the flexibility to model this exercise.

Next to the lack of expressiveness with respect to instantiating mathematical objects, there is a lack of expressiveness with respect to possibilities for visualization. GraspLe only supports 2D plots of functions, whereas matrix visualizations by means of grid representations of images are desired for linear algebra. While inserting images of the desired visualizations created externally is possible, this does not allow for parameterization of visualizations.

The trade-off by providing access to a full scripting language instead of the small set of operators selected by GraspLe is increased efficiency and effectiveness for authoring selected exercises, at the cost of a smaller group being able to author these exercises, because knowledge of a scripting language is required. Since two interviewed authors indicated the discussed desire for more expressiveness and efficiency by means of authoring exercises using a scripting language, it is worth investigating the possibilities of facilitating this in a user-friendly way.

2.3 Research question

Given the design choice of GraspLe and identified trade-off, the following research question is formulated:

How can a user-friendly format and specification for a set of elementary linear algebra mathematics exercises be designed allowing for high level abstraction and parametrisation?

The scope of the linear algebra procedures to be covered in the exercises and the feasibility of automatically evaluating these is discussed next. Then, the context is described in terms of technology used by GraspLe.

2.4 Scope and context

This section specifies the scope of the selected linear algebra mathematics sub domain to be covered with exercises and the feasibility of automatically assessing the procedural knowledge within this domain. The context of this project is described next, consisting of the technology stack used by GraspLe and the standards for typesetting mathematical documents.

Scope Considering that the mathematics domain is very broad and deep, covering a wide range of concepts and operations, a sub-domain is selected to author exercises for: linear algebra. In particular, undergraduate linear algebra as typically found in first year computer science courses, like CSE1205 Linear Algebra. The exercise content should cover the concepts and procedures found in these courses, like:

CSE1205 Linear Algebra²⁵:

- *Be able to perform matrix operations (sum, scalar multiple, multiplication, transpose) and describe the law-like properties of matrix multiplication and apply these properties;*
- *Solve systems of linear equations;*
- *Determinants;*
- *Eigenvalues and eigenvectors;*
- *Diagonalization;*
- *Inner products and orthogonal sets.*

The outcomes of applying these skills these should be automatically evaluated and assessed. This knowledge mostly corresponds to the bottom four out of the six total layers of the revised edition of Bloom's taxonomy of the cognitive domain [1] being, from the bottom up; remember, understand, apply, analyze, evaluate and create. Knowledge in these layers within the linear algebra domain is suitable for automated assessment, which is typically harder for higher layers (e.g. think about automatically evaluating a written proof created using mostly natural language).

Exercises can be used at many places in the learning process. Prior to taking a course, these can facilitate checking whether sufficient prerequisite knowledge is present, or whether certain subjects should be revisited, in which these can facilitate as well. During a course, exercises can be used for self-study at home. Furthermore, exercises

can be embedded around or within lectures. Shortly before or at the start of a lecture, knowledge the lecture builds upon can be refreshed. During a lecture (break), exercises can be used to practice introduced material, helping the instructor to see whether the subject is sufficiently understood. Finally, shortly after a lecture, when the new material is (hopefully) still (partially) in memory, exercises can help strengthen the new connections.

Context The implemented solution uses (part of) the technology stack used by GraspLe. This allows to easily recreate part of the functionality offered by GraspLe by obtaining information about (the interplay between) used technologies (but no implementation), and thus prevents designing a complete solution from scratch.

A central technology used by GraspLe to generate and evaluate answers is SymPy, a computer algebra system written in Python. Authors are expected to be familiar with Python to be able to author exercises and use the functionality to generate mathematical objects in exercise content using SymPy.

Python is popular according to the TIOBE Index of April 2021, obtaining the third place after C and Java²⁶, having over 11% of the total ratings. Therefore Python is likely to be known by authors.

Next to SymPy, GraspLe uses LaTeX for mathematical expressions in exercises and answers. LaTeX is the de facto standard for mathematical typesetting in academic documents, likely to be known by authors having a technical academic background. Since not all mathematical concepts can be easily expressed directly in SymPy, knowledge of LaTeX is required to complement SymPy in providing exercise content. This will be further elaborated in Chapter 3.

2.5 Other solutions: MyOpenMath, RExams, WebWorX and MEGUA

Existing exercise solutions have been collaboratively investigated by the employees of GraspLe, instructors from Dutch universities and other interested individuals at an action lab at Open Education Global (OE Global)²⁷, a community encouraging open education. The goal of this action lab was to *formulate a common standard for an open format for online, interactive math exercises*²⁸ by learning from the existing solutions and experiences of exercise authors. Four open math format initiatives have been identified: MyOpenMath, RExams, WebWorX and MathDox. Due to the lack of documentation available about MathDox, this format is not discussed further. None of these formats uses Python to define exercises, instead the used languages range from PHP in MyOpenMath to Lisp in WebWorX to R in RExams. These languages are expected to be less likely to be known by (potential) authors.

Regarding the requirements for the format, six suggestions are stated by participants:

- **Version controllable** This allows authors to track changes between different versions of an exercise and revert these if needed.
- **Human readable** The format should have a textual representation that can be inspected and edited. This enables authors without an interface to inspect the exercises.
- **Easy to understand** The format should be easy to understand. This allows authors to understand and reuse material created by others.

- **Allow for an intuitive, visual editor based on the format** A visual editor allows authors with insufficient programming knowledge to author exercises.
- **Flexible and extensible** The format should enable all current features used in interactive mathematics exercises. Furthermore, it should be possible to extend the format with functionality not thought of now.
- **Declarative** The exercise definition should state how the exercise should look like, not the detailed procedures about how to obtain the content. This allows for an efficient definition of an exercise that is easy to understand.

From this list of requirements, the requirement of a *flexible and extensible* format is most closely related to the research question addressed in this work. The high level of abstraction should allow authors to be flexible and extensible in authoring exercises by using libraries for generating exercise content or creating custom functions.

MEGUA MEGUA is developed by Cruz et al. from the University of Aveiro in Portugal [9], it uses SageMath and LaTeX templates to author exercises. Multi-line LaTeX strings can be parameterized and functions can be defined for providing the substitutions and computing the answer. An example of an exercise asking the learner to compute the transpose of a matrix is stated in Listing 2.1.

CODE LISTING 2.1: Matrix transpose parameterized exercise in MEGUA

```

1  txt=r'''
2  \\\%SUMMARY Matrices; Types of Matrices About matrix types and transposition.
3  \\\%PROBLEM Type of a matrix Which is the order of the matrix given by:
4      $$ A=matrixA ? $$
5  Write down its transpose.
6  \\\%ANSWER Matrix $$ has $nr$ rows and $nc$ columns so its order is $nr \
   times nc $. Transposing a matrix consists on using rows from the first
   matrix as columns in the second matrix, for instance, column 1 in $$
   will be row 1 in the transpose of $$ The transpose is a matrix with $nr$
   columns and $nc$ rows, so its order is $nc \times nr$
7      $$ matrixATranspose $$
8  class E12X34_matrices_001(Exercise):
9      def make_random(s):
10         s.nr = ZZ.random_element(2,5)
11         s.nc = ZZ.random_element(2,5)
12         s.matrixA = ZZ.random_matrix(ZZ, s.nr, s.nc)
13     def solve(s):
14         s.matrixATranspose = s.someMatrix.transpose()
15 '''
16 meg.save(txt)

```

However, MEGUA does not support open answers that are evaluated automatically, therefore it neither supports answer rules.

2.6 Requirements and use cases

This section first summarizes the functional requirements for students and authors, and compares related work with respect to these requirements. Next, two use cases illustrate learning situations posing the identified requirements, one of which is the matrix multiplication operation discussed earlier.

Requirements The existing solutions are compared with respect to the authoring language and answering technology, requirements from literature, discussed problem statement and context in Table 2.1. The Final answer column in 2.1 states the maximum supported expressiveness in the answer options offered by the platform. This ranges from a WYSIWYG formula editor, to a raw expression string to selecting an alternative from a predefined list of answer options (multiple choice).

	Origin	Authoring language, Answer input		Requirements from literature			High level of abstraction	
		Language	Final answer	Parameterized	Open answer expression	Open answer specific feedback	Python	Libraries for contextualization
Grasple	2014	GUI	WYSIWYG	Yes	Yes	Yes	No	No
MEGUA	2014	Sage	fixed	Yes	No	No	Yes	Yes
R/Exams	2008	R	fixed	Yes	No	No	No	Limited
MyOpenMath	2005	PHP	WYSIWYG	Yes	No	No	No	No
WeBWork	1996	Perl	raw	Yes	Yes	No	No	No
textbook	-	LaTeX	raw	No	Yes	No	No	Yes

TABLE 2.1: Comparison of mathematical exercise solutions

Use cases The use cases described below illustrate the need for flexible authoring and access to high level abstractions. Exercises for a selection of use cases are elaborated in Chapter 3.

- **Matrix multiplication** Matrix multiplication is an operation used in linear algebra to transform matrices. Applications of this operation include transforming images in the computer graphics domain and computing forward passes of data through neural networks in the machine learning domain.

Learning matrix multiplication requires practice, being able to perform this operation automatically typically requires repeating the concept over time.

An expected mistake when performing matrix multiplication is making arithmetic errors when simplifying the final solution matrix. Furthermore, issues can occur on a conceptual level, because the definition is not properly applied (e.g. think about the corner case of multiplying two square matrices, tricking the inexperienced student to use multiplication similar to how matrix addition is defined, i.e. computing the product instead of the sum for each pair of values at the same index).

Thus, to support learning this operation, a diversity of exercises is required to cover all corner cases and enable repeated practice. Furthermore, answer specific feedback can guide students towards the correct answers based on their answer characteristics, without explicitly revealing the solution.

- **Matrix visualization** To illustrate the effects of applying operations on matrices used to model digital images, visualizations are desired. These visualizations show

how the mathematical concept of a matrix corresponds to the applied real world manifestation of an image. This provides the contextualization of learned theory. For example, consider these subjects:

- Matrix dimensions, corresponding to the width and height of an image
- Matrix indexing, corresponding to obtaining the intensity of a single pixel value in a gray scale image
- Matrix/vector similarity, corresponding to the concept of comparing two images (using absolute difference, euclidean distance or cosine similarity)
- Matrix convolution by applying an image processing kernel to an image matrix, corresponding to the concept of (pre)processing images, for example before training a machine learning classifier.

Chapter 3

Setup for Flexible Exercise Authoring

This chapter describes the developed authoring setup. After covering the system context by discussing the interactions between the actors and system components, the functionality for authoring exercises using the developed Python classes is discussed. Next, system components and their implementation details are discussed in detail. Finally the chapter demonstrates the flexibility of this new setup by describing a range of linear algebra exercises.

3.1 Authoring setup context

The context of the authoring setup is depicted in Figure 3.1, showing the main actors and their interactions with the systems in the setup. Authors interact with the exercise authoring environment to define exercises, through which they access the embedded exercise player to preview these. Learners interact with the player directly through their browser, without knowing about the authoring environment.

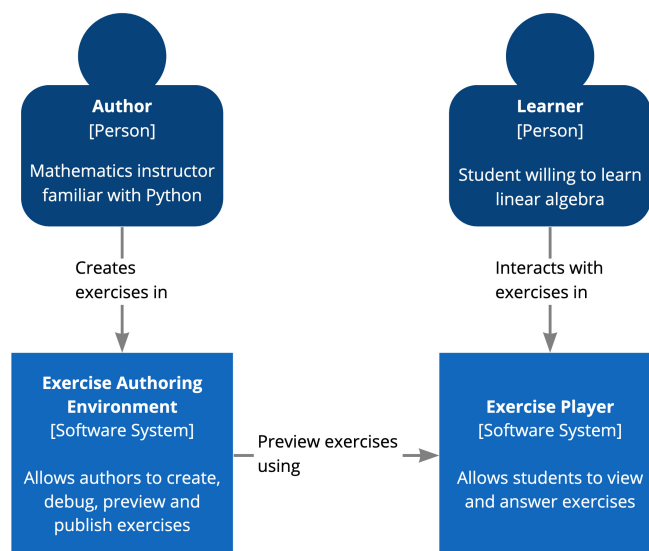


FIGURE 3.1: System context

In the developed setup, the authoring environment is a Jupyter Notebook. In Grasple, this is the web interface as described in Chapter 2. Within this Jupyter Notebook, exercises are defined at a cell level. Created exercises are previewed in iFrame, shown as the final result of executing the cell. An example of an exercise definition in a code

cell and corresponding iFrame cell output is shown in Figure 3.2. This iFrame contains the exercise player containing the exercise content (served from an external webserver). The exercise is shown as it will be shown to the learner. The author can interact with the exercise as the learner would, and verify the correctness both content- and interaction wise.

```
m = "What is $1 + 1$?"
e = Exercise(m)
e.add_answer(expression=2, correct=True, feedback="Indeed, $1 + 1 = 2$")
e.add_answer(expression=0, correct=False, feedback="Hmmm, did you compute $1 - 1 = 0$ instead?")
e.add_default_feedback("Please revisit the definition of natural numbers and the ($+$) operator")
e.play()
```

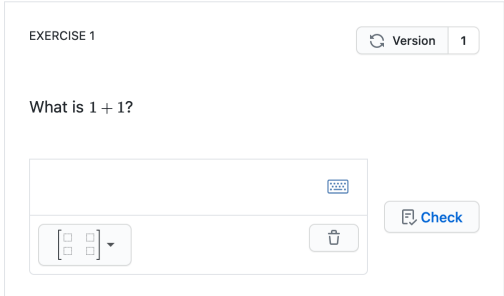


FIGURE 3.2: Example exercise instance in authoring environment (Jupyter Notebook): Exercise definition in code cell, corresponding exercise preview in exercise player embedded in iFrame

The player presents the exercise content and allows students to express answers in a WYSIWYG formula editor using either their physical keyboard or the virtual keyboard activated by clicking the keyboard symbol. Answer expressions are evaluated according to the exercise definition and the defined feedback is returned to the user. This process is shown in Figure 3.3.

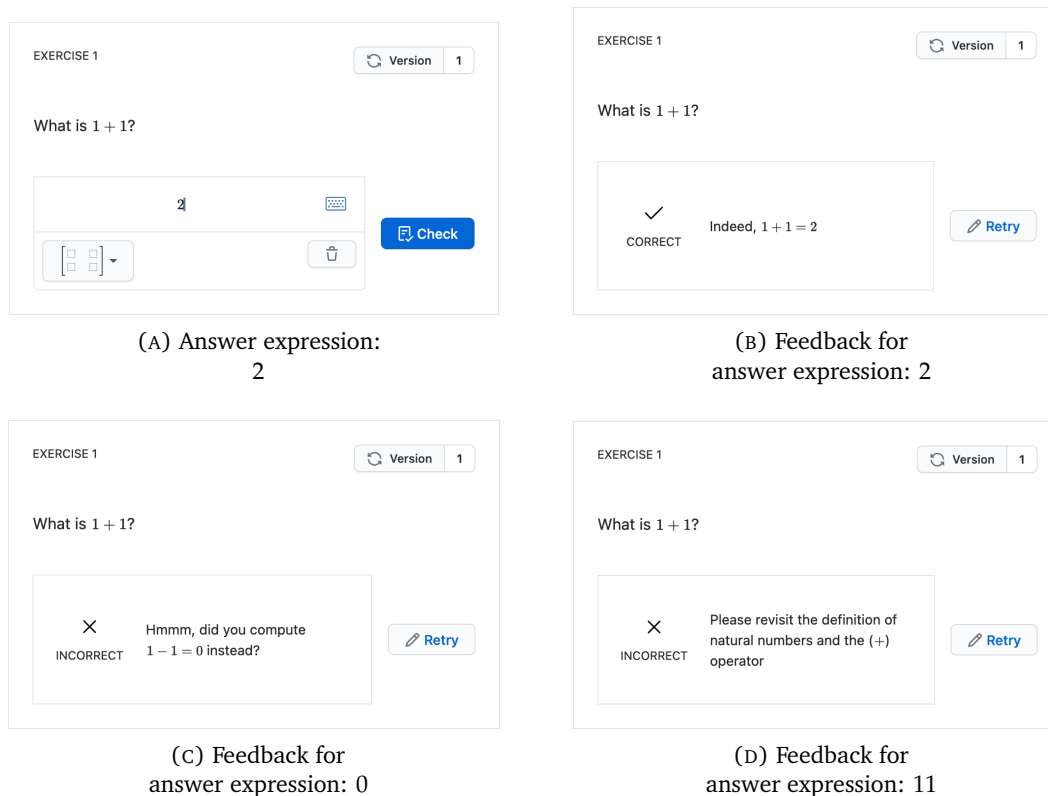


FIGURE 3.3: Exercise interaction: formulating an answer and receiving feedback

The exercise definition setup and player functionality are described next.

3.2 A first exercise definition: One-off integer addition

Consider the integer addition exercise described earlier. Let's create this exercise step by step using the developed setup. The content of this exercise is contained in a Markdown string, where mathematics notation can be included in an inline LaTeX expression wrapped around dollar signs, see Listing 3.1.

CODE LISTING 3.1: Integer addition exercise content markup

```
1 m = "What is  $\$1 + 1\$$ ?"
```

The exercise constructor accepts a Markdown string as an argument specifying the exercise content, shown in Listing 3.2.

CODE LISTING 3.2: Exercise object instantiation

```
1 e = Exercise(m)
```

Answers can be defined by calling the `add_answer` function on an exercise instance. This function takes three parameters, an answer expression being either an integer or a SymPy object, a boolean indicating whether the answer is correct and the feedback shown when this answer is provided by the user, see Listing 3.3.

CODE LISTING 3.3: Default feedback

```
1 e.add_answer(expression=2, correct=True, feedback="Indeed,  $\$1 + 1 = 2\$$ ")
```

```
2 e.add_answer(expression=0, correct=False, feedback="Hmmm, did you compute
    $1 - 1 = 0$ instead?")
```

Default feedback can be provided, shown in case of an incorrect answer not matching any of the specified answers, shown in Listing 3.4.

CODE LISTING 3.4: Answer definition

```
1 e.add_default_feedback("Please revisit the definition of natural numbers
    and the ($+$) operator")
```

With the exercise content and answer definitions in place, the exercise can be published and previewed using the play function, depicted in Listing 3.5. This reveals the exercise player as an iFrame as shown in Figure 3.2.

CODE LISTING 3.5: Publish and preview exercise

```
1 e.play()
```

That's it! This four lines of code define the exercise. Currently, this exercise is a one-off, meaning that students cannot practice the same concept again with different numbers. Fortunately, exercises can be parameterized to facilitate repeated practice, discussed next.

3.3 Enabling repeated practice: Parameterized integer addition

The integer addition exercise can be parameterized by replacing the static values in the sum by parameter names prefixed with the @ symbol, as listed in Listing 3.6.

CODE LISTING 3.6: Parameterized Markdown string

```
1 m = "What is $@a + @b$?"
```

To replace the parameters with integer values, a dictionary is created with parameters names as keys and as values the randomized integer values. To create a random integer value between 0 and 10, NumPy is used. This is shown in Listing 3.7.

CODE LISTING 3.7: Parameter declaration

```
1 params = {}
2 params["a"] = np.random.randint(10)
3 params["b"] = np.random.randint(10)
```

The markdown string and the parameter dictionary containing the replacements are passed as arguments to a new instance of a MarkdownBlock object. This object performs the replacements of the parameters with the string representations of the generated integers. This MarkdownBlock is then passed to the exercise constructor to provide the content for a one-off exercise instance, see Listing 3.8.

CODE LISTING 3.8: Parameter declaration

```
1 e = Exercise(MarkdownBlock(m, params))
```

Answers can be expressed in terms of parameters in the parameters dictionary and added to this dictionary to conveniently use these in answer feedback expressions, see Listing 3.9.

CODE LISTING 3.9: Parameter declaration

```

1 params["ans_correct"] = params["a"] + params["b"]
2 params["ans_incorrect"] = params["a"] - params["b"]
3 e.add_answer(expression=params["ans_correct"], correct=True, feedback=
    MarkdownBlock("That's right! $@a +
    @b = @ans_correct$")
4 e.add_answer(expression=params["ans_incorrect"], correct=False, feedback=
    MarkdownBlock("Hmm, did you compute
    $@a - @b = @ans_incorrect$ instead?"
    ))

```

Note that for this exercise, in case both $@a$ and $@b$ happen to be 0, the correct and incorrect answer are the same. Possibly this can be avoided by implementing additional logic to neglect instances for which this occurs, thereby being more sure the feedback corresponds to the procedure executed by the student. In the current implementation, another approach is chosen. The answers are compared against the user answer sequentially and the feedback for the first matching answer is returned. Assuming that in case the correct and incorrect answer are the same the student should be given the feedback for the correct answer, this answer should be declared first.

Each time the code cell containing the exercise definition is executed, new values are generated and a new one-off exercise instance is generated. To store different instances, a function returning an exercise instance can be supplied to the `write_multiple` function. This function calls the provided function for the specified amount of instances and stores the list of exercises under a given name. Generating multiple instances of the parameterized integer addition exercise is illustrated in Listing 3.10. The generator function can include additional checks on the generated parameter values and make a recursive call to generate a new instance in case the desired conditions are not met. By clicking the *Version* button in the player, learners can manually iterate through the generated instances.

CODE LISTING 3.10: Parameter declaration

```

1 def generator():
2     m = r"What is $@a + @b$?"
3
4     params = {}
5     params["a"] = np.random.randint(0, 10)
6     params["b"] = np.random.randint(10, 20)
7     params["ans"] = params["a"] + params["b"]
8
9     e = Exercise(MarkdownBlock(m, params))
10    e.add_answer(params["ans"], True, MarkdownBlock("Yes!", params))
11    e.add_default_feedback(MarkdownBlock("No!", params))
12
13    return e
14
15 Exercise.write_multiple(generator=generator, instances_count=100, id="
    int_param")
16 Exercise.play_by_id(id="int_param")

```

3.4 High level of abstraction: SymPy code generation for matrix exercises

Using vectors and matrices is at the core of linear algebra. SymPy supports many objects and operations for working with these. This allows authors to express exercise

content and answers using the abstractions provided by SymPy. For example, consider an exercise covering vector addition. Given the constructs explained so far, one could create a Markdown string for a one-off vector addition exercise as stated in Listing 3.11.

CODE LISTING 3.11: Naive vector addition exercise content

```

1 m = r"What is  $\begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 3 \\ 4 \end{bmatrix}$ "
2 # Note: the string is prefixed with 'r' to get a raw Python string,
3 # treating backslashes (\) as literal characters,
4 # this is required to maintain the proper representation of the LaTeX
   code

```

While this approach works and the values could even be parameterized to get multiple instances, this approach is limited. First, typing the LaTeX code for (large) vectors and matrices is rather inconvenient. Second, creating parameter values for each of the values is cumbersome (even if performed in a loop). Third, manually expressing the answers in terms of the parameters is again busy work. Finally, the length of the vectors are fixed, therefore, creating exercises with varying vector length requires specifying a new content string and adjusting the parameters. These limitations illustrate the need for a higher level of abstraction. Fortunately, SymPy provides these abstractions. The `Matrix` class allows to instantiate (randomized) matrices and vectors (a special type of matrix with a single column), as shown in Listing 3.12.

CODE LISTING 3.12: Random vector object using SymPy

```

1 # matrix with four rows, one column, containing values between 0 and 10
2 v1 = sp.randMatrix(r=4, c=1, min=0, max=10)

```

This vector can be further manipulated if needed. To use this vector as part of exercise content, the corresponding LaTeX expression should be obtained. SymPy provides the `latex()` function to produce this representation for a given object, stated in Listing 3.13.

CODE LISTING 3.13: LaTeX representation of SymPy object

```

1 sp.latex(v1)

```

For the statement in 3.13 the LaTeX string in 3.14 is returned.

CODE LISTING 3.14: LaTeX string returned from function call

```

1 '\\left[\\begin{matrix}1\\\\3\\\\7\\\\9\\end{matrix}\\right]'

```

This allows authors to use SymPy objects as parameter values as demonstrated in Listing 3.15, resulting in the exercise shown in Figure 3.4. The `MarkdownBlock` object is responsible for obtaining the LaTeX expression and replacing the parameter values with the LaTeX strings. The process of exercise transformation from templated Markdown string to exercise content shown to the learner is depicted in Figure 3.5.

CODE LISTING 3.15: Parameterized vector addition using SymPy objects

```

1 m = "What is  $@v1 + @v2$ ?"
2
3 params = {}
4 params["v1"] = sp.randMatrix(r=4, c=1, min=0, max=10)
5 params["v2"] = sp.randMatrix(r=4, c=1, min=0, max=10)
6 params["ans"] = params["v1"] + params["v2"]
7
8 e = Exercise(MarkdownBlock(m, params))
9 e.add_answer(params["ans"], True, "Correct!")
10
11 e.play()

```

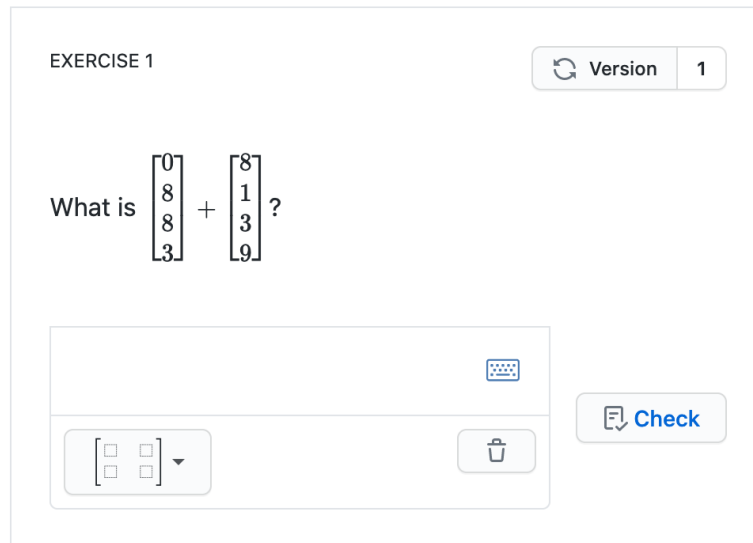


FIGURE 3.4: Parameterized vector addition exercise resulting from Listing 3.15

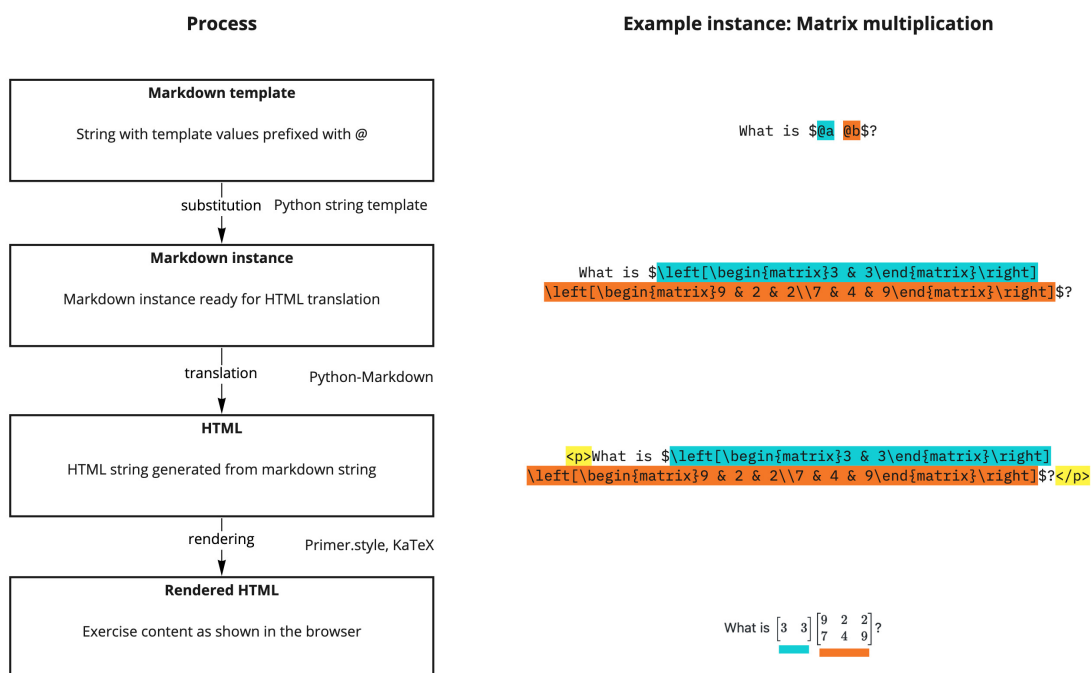


FIGURE 3.5: Exercise content transformation steps

Currently, the exercise contains no detailed feedback on a correct or incorrect answer. Next, the possibilities for facilitating this are discussed.

3.5 Guiding feedback

Consider the following detailed feedback for the vector addition exercise. In case of a correct answer, the complete solution is shown to the user. In case of an incorrect answer, a hint is given to the user revealing the definition of vector addition for two vectors of

the same size as in the problem instance. The feedback for each of these is shown in Figure 3.6.

EXERCISE 1 Version 20

What is $\begin{bmatrix} 9 \\ 7 \\ 1 \end{bmatrix} + \begin{bmatrix} 6 \\ 0 \\ 8 \end{bmatrix}$?

✓
CORRECT Yes, $\begin{bmatrix} 9 \\ 7 \\ 1 \end{bmatrix} + \begin{bmatrix} 6 \\ 0 \\ 8 \end{bmatrix} = \begin{bmatrix} (9+6) \\ (7+0) \\ (1+8) \end{bmatrix} = \begin{bmatrix} 15 \\ 7 \\ 9 \end{bmatrix}$!

Retry

(A) Feedback for correct answer

EXERCISE 1 Version 20

What is $\begin{bmatrix} 9 \\ 7 \\ 1 \end{bmatrix} + \begin{bmatrix} 6 \\ 0 \\ 8 \end{bmatrix}$?

✗
INCORRECT Remember the definition of matrix addition:
 $\begin{bmatrix} a_{1,1} \\ a_{2,1} \\ a_{3,1} \end{bmatrix} + \begin{bmatrix} b_{1,1} \\ b_{2,1} \\ b_{3,1} \end{bmatrix} = \begin{bmatrix} (a_{1,1} + b_{1,1}) \\ (a_{2,1} + b_{2,1}) \\ (a_{3,1} + b_{3,1}) \end{bmatrix}$

Retry

(B) Feedback for incorrect answer

FIGURE 3.6: Detailed feedback for correct and incorrect answer respectively

To generate this feedback, a function `explain_add` is created to obtain the unevaluated intermediate representation of the addition operation, as shown in the second to last vector at the correct answer feedback and the last vector at the incorrect answer feedback in Figure 3.6. This function relies on the feature of SymPy to keep expressions unevaluated²⁹, see Listing 3.16.

CODE LISTING 3.16: Function to obtain LaTeX of vector addition

```

1 from sympy import Symbol, latex, UnevaluatedExpr
2
3 u = lambda x : UnevaluatedExpr(x)
4
5 def explain_add(a, b):
6     # only matrices with the same dimensions can be added
7     assert(np.shape(a) == np.shape(b))
8     rows, columns = np.shape(a)
9     return sp.Matrix([[Symbol(f"({latex(u(a[i,j]))}) + {latex(u(b[i,j]))})")
                        for j in range(columns)] for
                        i in range(rows)])

```

This function can be used in the exercise definition as in Listing 3.17. It takes two matrices as arguments. Since a symbolic explanation is desired over revealing the full solution, two symbolic vectors of the same length are generated by the `symbolic_matrix` function in Listing 3.18.

CODE LISTING 3.17: Parameterized vector addition with detailed answer feedback

```

1 length = np.random.randint(1, 7)
2 v1 = sp.randMatrix(r=length, c=1, min=0, max=10)
3 v2 = sp.randMatrix(r=length, c=1, min=0, max=10)
4
5 s = "What is  $\$@v1 + @v2?$ "
6
7 params = {}
8 params["v1"] = v1
9 params["v2"] = v2
10 e = Exercise(MarkdownBlock(s, params))
11
12 params["v3"] = explain_add(v1, v2)
13 params["v4"] = v1 + v2
14 s1 = "Yes,  $\$@v1 + @v2 = @v3 = @v4!$ "
15 e.add_answer(v1 + v2, True, MarkdownBlock(s1, params))
16
17 a = symbolic_matrix("a", length, 1)
18 b = symbolic_matrix("b", length, 1)
19 ab = explain_add(a, b)
20 default_feedback = "Remember the definition of matrix addition:  $\$@a + @b$ 
                       =  $@ab$ "
21 e.add_default_feedback(MarkdownBlock(default_feedback, dict(a=a, b=b, ab=
                                       ab)))

```

CODE LISTING 3.18: Function to obtain the LaTeX representation of a symbolic matrix

```

1 def symbolic_matrix(character, rows, columns):
2     return sp.Matrix([[Symbol(f"{{{character}}}_{{{i+1}}, {j+1}}") for j
                        in range(columns)] for i in
                        range(rows)])

```

While SymPy provides the `MatrixSymbol` constructor to this end, it is zero indexed and therefore cannot be used for generating the mathematics notation being one indexed.

3.6 Architecture: Exercise player, JSON representation and answer evaluation

Player This paragraph describes the functionality and of the exercise player and interaction with other components.

The player allows students to view exercise content, express answers by means of the WYSIWYG formula editor and receive feedback with respect to the given answer. Furthermore students can request a new exercise instance with different values by using the *Version* button.

The player is implemented in Elixir using the Phoenix LiveView framework. After posting an exercise instance to a developed Python API (in FastAPI) for storing and retrieving exercises, by using the `play` function of an `Exercise` instance, the exercise can be requested to be served in the browser from the Phoenix app, by browsing to the url of the exercise (e.g. <https://www.mscthesis.nl/preview?id=bfcf5246-5064-429c-96ad-f61ba3e94e8c>). Using the `play` functionality, the exercise is loaded into the `iFrame` automatically, without explicitly requiring the user to copy the url in a new browser tab. Phoenix obtains the JSON file of the exercise from the API, processes the content and presents the exercise in the player to the user. It serves the raw HTML as part of the exercise definition contained in the JSON format discussed below and

calls a Phoenix hook³⁰ to use KaTeX to render the mathematics notation in the clients browser when the exercise content is updated. The sequential interaction between the components for publishing an exercise and loading it into the player is shown in Figure 3.7.

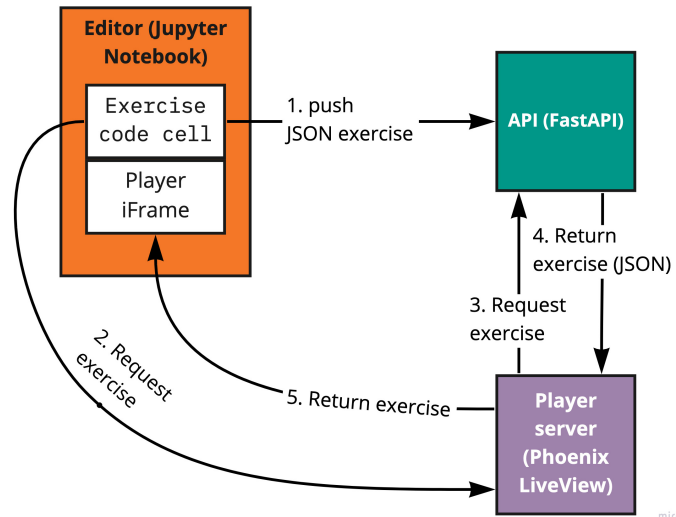
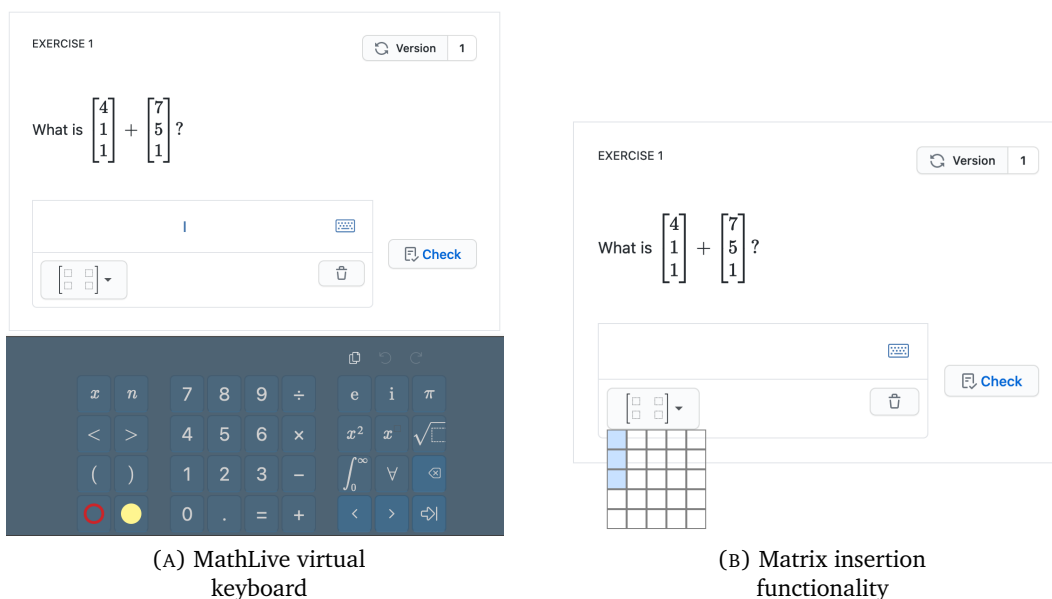


FIGURE 3.7: Component interaction of publishing and playing exercises

The MathLive³¹ formula editor is used to express answers. Answers can be expressed by using a physical keyboard and typical shortcuts for editing expressions (e.g. pressing \sim for power, $/$ for fractions), or by using the virtual keyboard triggered by pressing the keyboard icon with expression templates, shown in Figure 3.8. Since the virtual keyboard does not provide an option for inserting matrices, a feature for easily inserting matrix templates of specified dimensions is provided, depicted in Figure 3.8.



(A) MathLive virtual keyboard

(B) Matrix insertion functionality

FIGURE 3.8: Player answer editing functionality

The `pushEvent` and `pushEventToPhoenix` functions are used to interact between the Javascript state in the clients browser (e.g. the answer expression in the formula editor)

and the server. When the *Check* button is clicked, the user answer LaTeX expression is obtained from the client and send to a stateless remote cloud function together with the exercise JSON representation (containing the answers) for evaluation. This remote function returns the raw HTML feedback for the user answer, which is returned to the server, which in turn updates the client. The interaction between the components when answering an exercise is shown in Figure 3.9. By clicking the *Retry* button, a new answer can be formulated.

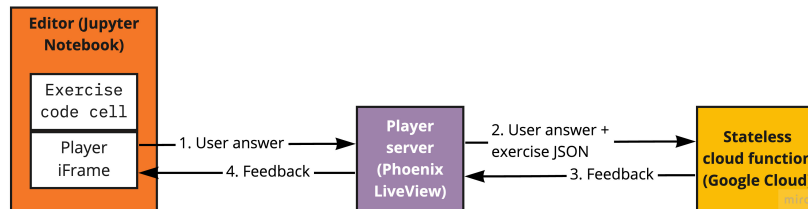


FIGURE 3.9: Component interaction for answer evaluation

Exercise format The created `Exercise` class allows the Python exercise object instance to be serialized in JSON. A serialized representation is necessary for storage and the exchange between components in the architecture. While the object could be naively serialized as a Python pickle file, the JSON representation has multiple advantages over a pickle. First of all, since JSON parsers are commonly available in many languages, thus the exercise can be exchanged between components using different languages. In the chosen setup, the player is implemented in Elixir while the authoring setup is developed in Python. The JSON format allows exchange of the exercise between these components, independent of the languages used in these components. Second, in case exercise functionality is altered or extended affecting the format, a parser can account for these changes and accept different versions, while this would be inconvenient using the pickled representation. Finally, the JSON format can easily be manually inspected because of the human readable textual representation, this is as easy for a binary object.

The JSON exercise format contains the following keys:

- **id** An unique identifier corresponding to this exercise;
- **html** The raw HTML content of the exercise;
- **default_feedback** The feedback shown to the user in case none of the specified answers matches the user answer.
- **answers** An array of answers, each containing the following keys:
 - **expression** A LaTeX answer expression;
 - **correct** A boolean indicating whether this answer is correct;
 - **feedback** The raw HTML feedback to be shown to the user.

The JSON representation of the vector addition exercise discussed earlier is listed in Listing 3.19. Parameterized exercises contain an JSON array of these exercise instances.

CODE LISTING 3.19: JSON representation of randomized vector addition exercise

```

1 {
2   "id": "955bf3e0-cd6b-4401-9e90-a04392a6c2c4",

```

```

3     "html": "<p>What is  $\left[ \begin{matrix} 7 \\ 7 \\ 5 \end{matrix} \right] + \left[ \begin{matrix} 6 \\ 1 \\ 8 \end{matrix} \right]$ ?</p>",
4     "default_feedback": "Incorrect, no specific feedback provided matching
5     your answer",
6     "answers": [
7         {
8             "expression": " $\left[ \begin{matrix} 13 \\ 8 \\ 13 \end{matrix} \right]$ ",
9             "correct": true,
10            "feedback": "<p>That's right!</p>"
11        }
12    ]
13 }%

```

The language independence of the JSON format is demonstrated by making the exercises compatible with the GraspLe player. A company representative and developer of GraspLe, created a working prototype for playing exercises in the specified JSON format in the GraspLe player.

Answer evaluation Answers received from the server are evaluated using a stateless cloud function written in Flask, running on Google Cloud. This function is run on Google Cloud for the automated sandboxing and configurable computational resources and time constraints. This is desired for handling undesired answer expressions, such as extreme expressions, like $10^{10^{10}}$, being impossibly large to evaluate, or invalid expressions not understood by SymPy. The function executes as follows. The function receives the JSON exercise object and the user LaTeX answer string from the server in a HTTP request. Next, the user answer is parsed to a SymPy object and simplified using the expression simplification functionality³². Then, this answer is sequentially compared against the parsed and simplified reference answers under the answers key in the JSON format. In case of the answer matches, the corresponding answer feedback is returned.

3.7 Complex exercises

This section demonstrates the flexibility of the designed format by means of describing some introductory linear algebra exercises, including the learning goals for the student and example exercise instances. The code snippets defining the exercises are added in Appendix C. Defined exercises rely on the developed Python classes and functionality already discussed in this chapter, therefore, only exercise specific code level details are discussed. The exercises start off with the basics of linear algebra and finish with complex operations on matrices.

Matrix definition: Rows, columns, total values Matrices can be used as a mathematical representation of data. For example, matrices can be used to model digital images. The width of the image in pixels corresponds to the amount of columns, while the height corresponds to the amount of rows. The total amount of values in a matrix is the amount of rows multiplied by the amount of columns.

Two exercise instances asking the user to determine the amount of columns in a matrix are shown in Figure 3.10. The first exercise provides a vanilla matrix representation, while the second provides a contextualized matrix as a random digital image. The latter relies on functionality created around the `imshow` Matplotlib function to plot matrices. Plotted matrices are stored as images, which can be referenced in exercise templates. By

using the Pymdownx.b64³³ Markdown extension, images are embedded in the HTML content of the exercise.

EXERCISE 1 Version 3

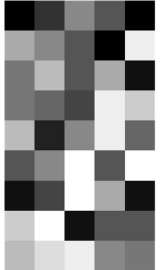
Consider the matrix A , how many columns does A have?

$$A = \begin{bmatrix} 1 & 1 & 4 \\ 3 & 3 & 12 \\ 7 & 12 & 1 \\ 14 & 15 & 14 \\ 8 & 10 & 4 \\ 6 & 5 & 7 \\ 11 & 3 & 14 \end{bmatrix}$$

3 Check

EXERCISE 1 Version 9

Consider the matrix randomart image below, how many columns does it contain?



5 Check

(A) Vanilla exercise
(B) Contextualized exercise

FIGURE 3.10: Exercise: Matrix dimensions - amount of columns

Similar exercises can be created asking students to compute the amount of rows and amount of total values in a matrix.

Matrix definition: Symbolic indexing Being able to determine the value at a given position in a matrix is a necessary skill for selecting specific information from matrices. Furthermore this is key for understanding other operations defined on matrices. In a grey scale image, the matrix values indicate the intensity of the pixel value (from white being 0 to black being 15). A first exercise on this topic could ask the learner to identify a matrix value by providing a symbolic reference (e.g. $a_{1,2}$) to the value. The exercise content can support the learner in understanding this concept. For example, the matrix to be indexed can contain axis names stating *row* and *column*. Additionally, a symbolic matrix of matching dimensions could be provided (possibly as part of an answer hint). Finally, in case the learner swaps the rows and columns, i.e. returns the value for $a_{j,i}$ instead of the value $a_{i,j}$ which is asked for, feedback could be provided indicating this error, if such value exists. Exercises demonstrating this are show in Figure 3.11.

(A) Vanilla exercise,
feedback for answer 10

(B) Contextualized
exercise, default
feedback on wrong
answer

FIGURE 3.11: Exercise: Matrix indexing

Another way to practice this concept is by asking students to create a matrix from given symbolic references, for which an exercise is visible in Figure 3.12. Creating the content for this exercise requires custom string formatting and LaTeX generation, both of which could possibly be further abstracted for in a future work.

FIGURE 3.12: Exercise: Construct matrix from description

Matrix operation: Distance score between digits With these matrix basics in place, students are ready to learn about operations that can be defined on matrices, allowing them to use these to solve interesting problems. Consider students being introduced to the topic of hand written digit classification by means of the kNN algorithm, using different distance measures to compare digits (matrices). Starting simple, students are tasked to compute the absolute difference matrix between two given binary digit matrices. This matrix is defined as the absolute difference between each two corresponding matrix entries at the same index. Next, this distance matrix is reduced to a single distance score

value between these two digits, being the sum of all elements in the distance matrix. An exercise asking learners to compute this distance score for two given digits is shown in Figure 3.13. This exercise combines the two steps, while these could also be separately tested.

EXERCISE 1 Version 5

Given the matrices

	1	2	3	4	5	6	7	8
1	0	0	0	1	1	0	0	0
2	0	0	1	1	1	0	0	0
3	0	0	1	0	1	1	0	0
4	0	0	0	1	1	0	0	0
5	0	0	0	1	1	0	0	0
6	0	0	1	0	1	0	0	0
7	0	0	1	0	1	0	0	0
8	0	0	0	1	1	0	0	0

	1	2	3	4	5	6	7	8
1	0	0	1	1	1	0	0	0
2	0	0	0	0	1	1	0	0
3	0	0	0	0	1	1	0	0
4	0	0	0	0	1	0	0	0
5	0	0	0	0	1	1	0	0
6	0	0	0	0	0	1	0	0
7	0	0	1	0	0	1	0	0
8	0	0	1	1	1	1	0	0

$A =$

	1	2	3	4	5	6	7	8
1	0	0	0	1	1	0	0	0
2	0	0	1	1	1	0	0	0
3	0	0	1	0	1	1	0	0
4	0	0	0	1	1	0	0	0
5	0	0	0	1	1	0	0	0
6	0	0	1	0	1	0	0	0
7	0	0	1	0	1	0	0	0
8	0	0	0	1	1	0	0	0

 $B =$

	1	2	3	4	5	6	7	8
1	0	0	1	1	1	0	0	0
2	0	0	0	0	1	1	0	0
3	0	0	0	0	1	1	0	0
4	0	0	0	0	1	0	0	0
5	0	0	0	0	1	1	0	0
6	0	0	0	0	0	1	0	0
7	0	0	1	0	0	1	0	0
8	0	0	1	1	1	1	0	0

Task

First, determine $D = |A - B|$, then compute $\sum_{i=1}^8 \sum_{j=1}^8 d_{i,j}$

✘

INCORRECT

Hint: $D =$

	1	2	3	4	5	6	7	8
1	0	0	1	0	0	0	0	0
2	0	0	1	1	0	0	0	0
3	0	0	1	0	0	0	0	0
4	0	0	0	1	0	0	0	0
5	0	0	0	1	0	1	0	0
6	0	0	1	0	1	1	0	0
7	0	0	0	0	1	1	0	0
8	0	0	1	0	0	1	0	0

[Retry](#)

FIGURE 3.13: Exercise: Compute distance score between two binary digit images

This exercise can be gradually extended by introducing more complexity. Full gray scale images can be provided, instead of the binary image, shown in Figure 3.14. This exercise asks learners to compute the difference score for the first row only to keep the exercise feasible. Furthermore other distance metrics could be used, such as Euclidean distance or cosine distance.

EXERCISE 1 Version 2

Given the matrices

	1	2	3	4	5	6	7	8
1	0	0	0	12	5	0	0	0
2	0	0	2	15	7	0	0	0
3	0	0	7	16	8	0	0	0
4	0	0	15	15	8	4	0	0
5	0	0	15	16	16	15	3	0
6	0	1	16	19	4	11	11	0
7	0	0	11	14	9	15	11	0
8	0	0	1	14	16	15	6	0

 $A =$

	1	2	3	4	5	6	7	8
1	0	1	11	16	16	4	0	0
2	0	7	16	8	14	11	0	0
3	0	0	0	10	16	6	0	0
4	0	0	0	15	16	6	0	0
5	0	0	0	0	8	16	2	0
6	0	1	5	0	0	14	9	0
7	0	4	16	10	11	16	6	0
8	0	1	15	16	16	10	0	0

 $B =$

Task

Defining D as $D = |A - B|$, compute $\sum_{j=1} d_{1,j}$

[□ □]

[□ □]

FIGURE 3.14: Exercise: Distance score for first row, between two gray scale digit images

Chapter 4

Evaluation

This chapter covers the evaluation of the developed setup explained in Chapter 3, by means of focus groups and a usability study consisting of authoring tasks and semi-structured interviews.

4.1 Focus groups

Two focus group meetings were organized to collect feedback from exercise authors.

The characteristics of focus groups are defined by Kreuger [18]:

1. A small group of people, who,
2. possess certain characteristics,
3. provide qualitative data,
4. in a focused discussion,
5. to help understand the topic of interest.

The key component making focus groups distinctive from other forms of feedback collection, such as interviews or surveys, is the conversational discussion in which participants can openly discuss and follow-up on suggestions of others.

The duration of the focus group is one hour. Zoom is used to facilitate the online meeting.

The outline of the activities during the focus group is as follows:

1. **Introduction.** Participants are asked to briefly introduce themselves, providing information about their professional activities and their affiliation with education (*5 minutes*).
2. **Presentation.** The topic of this thesis and the initial requirements originating from learning theory literature, authors and Grasple are presented. Next, the derived research questions are shown (*10 minutes*).
3. **Demo.** The features of the developed Python framework are shown by using the screen sharing functionality of Zoom. Exercises of increasing complexity are created, starting with a static exercise covering integer addition and finishing with operations on parameterized matrices. Participants are asked for questions/input while creating exercises. These questions are answered right away. In addition, participants were asked to write notes containing suggestions on a blank (i.e. no templates) Miro board (*15 minutes*).

4. **Discussion.** During the discussion, participants are asked to reply to suggestions of others. This results in an open discussion around additional requirements and limitations of the proposed solution (*25 minutes*).
5. **Closing.** Participants are made aware that the meeting is about to end. Any final comments can be stated and participants are acknowledged for their time and input (*5 minutes*).

4.2 First focus group feedback

Three external participants attended the first focus group meeting. Two of them did know each other prior to the meeting. Both are affiliated with the Electrical Engineering faculty at TU Delft. The other external member is a developer of learning and assessment material for Computer Science and Applied Mathematics from the University of Twente.

The result of the Miro board is depicted in Figure 4.1. Since the participants didn't put notes on the board by themselves, these are added by the group leader during the meeting.

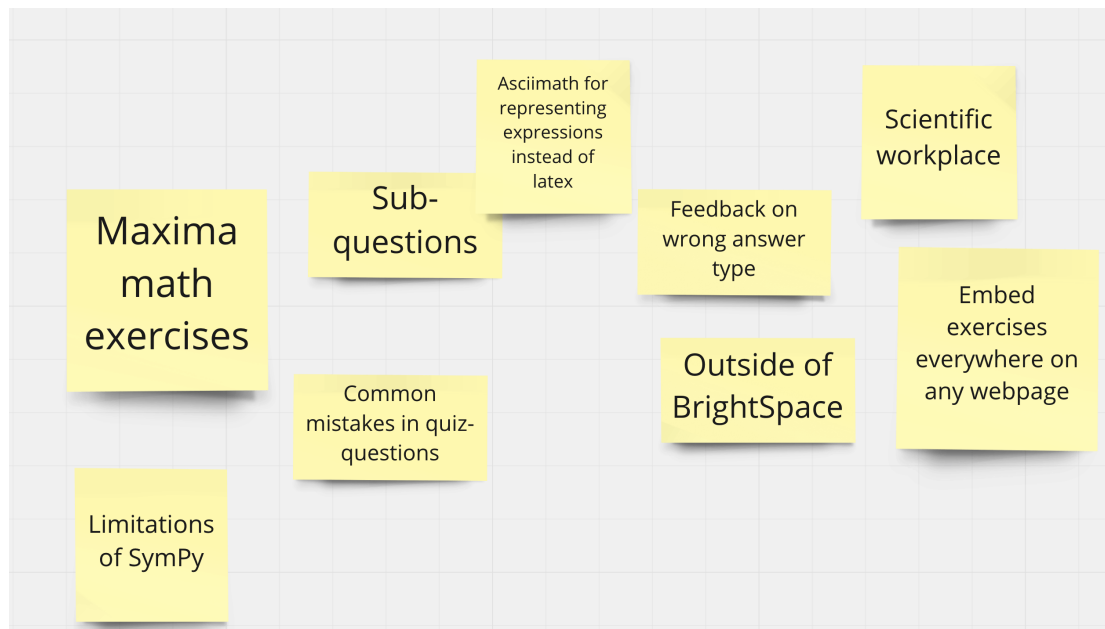


FIGURE 4.1: Miro board after first focus group

The topics listed in Figure 4.1 are elaborated below. The feedback is grouped in four categories: positive feedback, additional requirements, existing solutions and other comments. This grouping originates from reviewing the received feedback and identifying similar comments. For each category, a paraphrased quote from one of the participants is stated as an example for comments in the respective category. Considerations discussing selected feedback are written in *italic*.

Positive feedback This covers the positive feedback regarding the presented work.

"Digital exercise systems facilitate in flipping the classroom. We are implementing flipping the classroom for our courses to use the contact hours more effectively."

- **Efficiency** The high level of abstraction offered by providing direct access to SymPy objects and Markdown templates is recognized by one participant (a Grasple user) to be a more efficient way of authoring parameterized matrix exercises compared to current Grasple authoring options.
- **Flipping the classroom** Two participants acknowledged the need for digital exercise solutions. These facilitate in personalized learning by leveraging learning analytics for learners with different prior knowledge, leaving lecture time for discussing questions and instructor guided project support.

Additional Requirements This covers additional requirements discussed by participants.

"Can you support collecting common incorrect answers to allow instructors to provide answer-specific feedback?"

- **Sub-questions** Two participants recommended asking sub-questions, in case the initial question is too difficult to answer. This can guide students towards the correct answer without immediately showing the correct answer.
- **Feedback on incorrect answer type** In case a student provides a incorrect answer type, for example a scalar instead of a matrix for a matrix-addition problem, specific feedback should be given with respect to answer-aspects beyond symbolic equivalence with a reference answer.
- **Learning Analytics** Identifying typical mistakes can be facilitated by collecting user answers and grouping these. This gives teachers insight in common student misconceptions.
- **Tagging** A participant mentioned the need for tagging exercises to link to topics. This should support providing learning analytics on a per-topic basis.

Existing Solutions Existing solutions mentioned by participants are listed.

"Are you familiar with Maxima? SymPy has limitations as it comes to large matrix multiplications, and Jupyter Notebook is unstable from time to time because of kernel failures, look into Maxima instead."

- **Maxima** One of the participants recommended using Maxima over SymPy because of experience with limitations of SymPy as it comes to multiplications of matrices with large dimensions. *Maxima uses Lisp, which is unpopular compared to Python (0.36% versus 11.87% in the TIOBE index of May 2021³⁴ respectively). Furthermore more libraries supporting the authoring of exercises are available in Python.*
- **Scientific WorkPlace** A participant recommended investigating the possibilities of Scientific WorkPlace for WYSIWYG LaTeX editing and CAS functionality. *Scientific WorkPlace is commercial software and seems inappropriate for creating parameterized exercises due to the lack of programming possibilities.*

Other Comments This includes any comments outside of the other categories.

"Why are you using LaTeX to store answer expressions? In LaTeX, multiple notations exist for the same mathematical object (e.g. $matrix$, $pmatrix$, $bmatrix$), all of these need to be parsed and understood. Why not use AsciiMath instead for a uniform representation of such objects?"

- **AsciiMath** One participant advised to use AsciiMath to store and represent the answer expressions over the LaTeX representation currently used. The main concern of this participant is that in LaTeX, multiple syntactically different terms refer to the same semantic object, for example, a matrix can be one of: $matrix$, $pmatrix$, $bmatrix$, $Bmatrix$, $vmatrix$ or $Vmatrix$, all of which refer to a matrix object and should be properly recognized by the parser. *Indeed, by using LaTeX, which is designed for typesetting documents and not for formulating mathematical expressions serving as input for semantic recognition by a computer, all of the mentioned syntactic terms should be parsed. From a parser writer perspective, using AsciiMath to store expressions would indeed be a better option. However, SymPy does not support parsing AsciiMath expressions as input, therefore this suggestion cannot be implemented.*
- **Open Source** A strong opinion is voiced to create a fully open source solution, without any dependencies on commercial software systems. This allows for full control over all aspects of the system, including extension possibilities. For multiple-choice quiz questions generated with similar solutions, the instructors do not want to be restricted by the functionality offered by BrightSpace for presentation and learning analytics.

4.3 Second focus group feedback

Four external participants attended the second focus group meeting. Three members are instructors at TU Delft, two of these are from the teaching team of the Computer Science bachelor, one is an instructor teaching various mathematics courses. One member is a researcher at Utrecht University, studying and teaching mathematics.

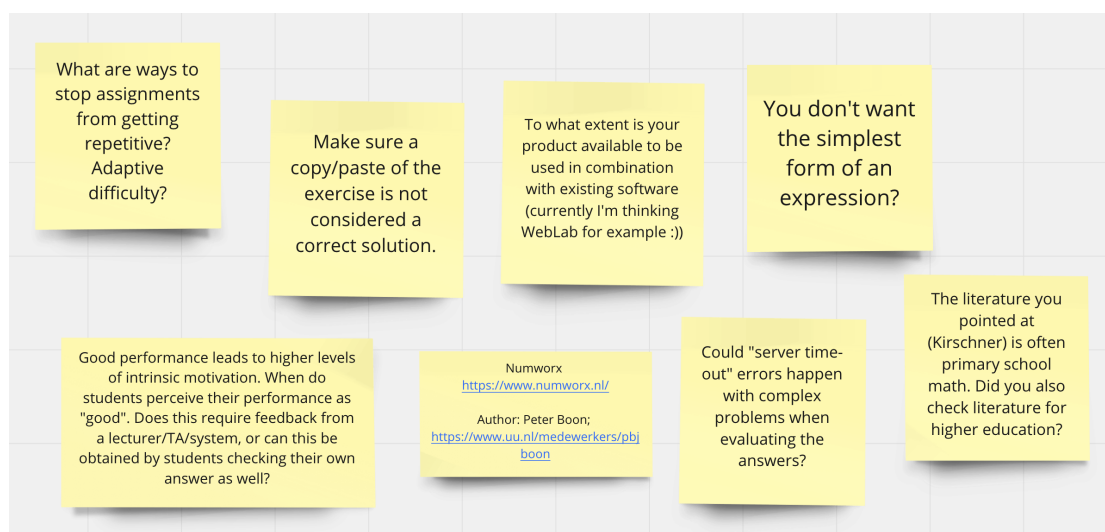


FIGURE 4.2: Miro board after second focus group

The topics in Figure 4.2 are elaborated below.

Positive Feedback This covers the positive feedback regarding the presented work.

While we provided you with a lot of comments and suggestions, it should be noted that the current setup is promising.

- **Current setup** A participant encouraged the current setup and demonstrated possibilities.

Additional Requirements This covers additional requirements discussed by participants.

Many platforms in our daily lives start adopting gamification elements, ranging from encouraging savings in banking apps to promoting physical activity using apps. Did you consider implementing some sort of gamification?

- **Answer-specific feedback** Two participants expressed their concerns regarding the possibility of copying the problem statement in the answer field. Another participant mentioned the need for an option to accept only the simplest form of an expression, that is, maximally analytically simplified.
- **Adaptive difficulty** One participant recommended to make the difficulty adaptive so similar assignments do not become too repetitive in a short practice session. Another participant recommended to use spaced repetition to avoid this problem.
- **Integration in existing software** Two participants expressed the desire for a fully open source solution, that can be easily integrated in the existing learning management system used by TU Delft, WebLab. Integration could include learning analytics and asking questions related to exercise instances.
- **Gamification** All participants agreed that gamification elements can potentially increase learner engagement. While gamification elements are found in various other platforms, these seem rarely available in instruction systems used in higher education.

Existing Solutions Existing solutions mentioned by participants are listed.

The problems you aim to address are already solved by Numworx and AlgebraKiT.

- **Numworx** According to a participant, the problems addressed by this work are already solved in Numworx, a commercial mathematics instruction platform.
- **AlgebraKiT** A participant recommended to investigate the possibilities of AlgebraKiT. *Both Numworx and AlgebraKiT are commercial solutions, while these solutions are not investigated thoroughly, these do not seem to support using external libraries for creating images, furthermore these do not support the flexibility offered by a general purpose programming language.*

Other Comments This includes any comments outside of the other categories.

Is it possible to configure the extent of automated answer simplification? For some exercises, simply copying the problem statement as the answer should not be considered a correct solution.

- **Evaluation timeout** A participant asked whether the answer evaluation will time-out on evaluating complex or extreme answers (such as: $10^{10^{10}}$). *This problem is, at the time of writing, mitigated by evaluating the student answer using a stateless cloud function.*
- **Answer simplification** A participant mentioned that configuring the extent to which an answer should be simplified by the student is impossible within GraspLe. Therefore, for some exercises, copying the problem statement in the formula editor as an answer is accepted correct. *At the time of writing, the implementation checks whether the simplified version of both the student and reference answer are the same, by using the `simplify` function. For applying the operations of matrices used in this work (e.g. matrix addition and multiplication), this works as desired, as these are not simplified. For other problems, configuration of the extent to which student solutions should be simplified is desired.*

4.4 Setup usability evaluation

To evaluate the usability of the developed Python classes for authoring exercises, with authors, three university staff members are asked to create four exercises based on the corresponding exercise task descriptions. After completing the tasks, which is estimated to take about 30 minutes, a 30 minute semi-structured interview is conducted to collect experiences and suggestions.

4.4.1 Exercise authoring tasks

While all authors attended the focus group meetings discussing the developed solution, the notebook introduced the exercise construction process from scratch, allowing authors to review the functionality prior to constructing new exercises. To make the authoring process as convenient as possible, the Jupyter Notebook environment required for authoring exercises is designed to be compatible with Binder³⁵. Binder allows the notebook to run in the browser, without requiring installing any dependencies locally. Binder creates a remote Docker container and installs all specified Python dependencies found in `requirements.txt` in a virtual environment within this container. This allows authors to start authoring exercises within the notebook by clicking the *Launch Binder* button in the `README.md` file shown on the public GitHub repository containing the notebook.

The plain task descriptions are listed below, the full notebook can be found in Appendix B. Tasks one to three are successfully completed by all participants.

For the third task, one participant asked learners to describe the position of a given value in a matrix, instead of providing the value for a given position. Given the open-ended task description, this formulation could have been expected, but wasn't anticipated for. The user succeeded in authoring the exercise under this interpretation by providing the answer in row-vector form.

The final task is only fully completed by one author. As for the other two authors, one did have issues with understanding the provided `explain_multiply` answer function, while the other missed the dollar signs around the mathematics parameters in an otherwise perfect exercise.

Task 1: Integer division Create an exercise asking learners to compute $3/3$. Provide answer-specific feedback in case learners compute $3 * 3$ instead. Add default feedback

(using `e.add_default_feedback(...)`) with a link pointing to a source of preference explaining (integer) division (hint: `[link](www.example.com)`). Feel free to embed your favorite meme or xkcd at a correct/incorrect answer (hint `![img](www.example.com/img)`).

Task 2: Parameterized vector addition Create an exercise asking learners to compute the sum of two vectors of random length (within reasonable limits), with random integer values. Note: if you prefer NumPy for working with matrices, you are in luck! NumPy objects can be passed to the SymPy matrix constructor, e.g. `sp.Matrix(np.arange(4))`.

Task 3: Matrix indexing Create an exercise asking learners to identify a value at randomized indices (but within bounds) in a 5 by 5 matrix. Please make sure all values are unique so there is only one correct answer.

Task 4: Matrix multiplication Create an exercise asking users to multiply two matrices. Provide a default answer explaining the procedure in case a wrong answer is supplied. You can use the `symbolic_matrix` and `explain_multiply` functions supplied in `helpers.py`.

4.4.2 Semi-structured interviews

The questions for the semi-structured interviews to collect experiences and suggestions are taken from an API usability study performed by Piccioni et al. [28] investigating the usability of an API written in Eiffel to perform database queries. This study investigates four characteristics of usability: understandability, abstraction, reusability and learnability. In addition to these characteristics, two other topics are added. *Editing & collaboration* asks participants about authoring using the given setup in comparison to a WYSIWYG editor and what collaboration with non-power users could look like. *Miscellaneous* asks participants for missing features and leaves room for any additional comments not covered by any of the earlier questions. The complete list of questions for each usability characteristic can be found in Appendix A. In a 30 minute Zoom interview one or more questions per characteristic are asked, depending on time of replying and topics already implicitly discussed in earlier questions. The comments collected are stated below and provided with additional context and reactions if necessary. In case a similar comment is provided by multiple participants, these comments are grouped and the number in front of the item indicates the amount of times this is mentioned. A discussion of the comments is written in *italic*.

Understandability

- (1) When mathematical LaTeX objects are not wrapped in dollar signs (e.g. by writing `@a + @b` instead of `$(a + b)$`), the rendered exercise content does not show as expected. For the author, it is unclear where this error originates from or how to debug this issue. *Automatic wrapping of SymPy objects when the generating LaTeX code is something not thought of at design time. In a future iteration, this is possible by setting the `mode` argument of the `latex` function of SymPy, with the desired delimiters.*

Abstraction

- (1) The `add_answer` function takes three arguments: the answer expression as a SymPy object, a boolean indicating whether the answer is correct and a feedback string. A participant asked whether this function supports overloading by supplying less arguments. *At the time of the usability study this is not the case, however, in a future iteration the feedback argument could be made optional.*
- (2) Two participants mentioned that it is unclear whether a string or a `MarkdownBlock` should be supplied to an `Exercise` constructor. While no explicit issues are encountered, both participants questioned the "right" way to use it. Neither of these participants inspected the documentation or implementation to find an answer.
- (1) For one participant, it is unclear why the answer is part of the `params` dictionary. This participant insisted on separating the parameters from the answers. *To prevent redefinition of exercise parameters used in the answer feedback, the answer is added to the exercise parameters.*

Reusability

- (3) All participants indicated that the amount of code required to solve the tasks matched their expectations.
- (3) The possibilities for evaluating progress while authoring an exercise is reported to be sufficient for the tasks at hand.
- (1) A participant indicated that multiple solutions exist to model and solve the task. In addition, it is noted that while some solutions may be more elegant than others in general perception, other solution differences may depend on personal code style and organization preference.

Learnability

- (3) Participants indicated a lack of knowledge with respect to the NumPy and SymPy dependencies. While the desired functions to solve the tasks were described to be intuitive and fast to find using Google, a lack of knowledge with respect to these was slightly underestimated.
- (1) To debug the output of mathematical operations by SymPy and NumPy, a participant used `print` statements. The participant indicated that this worked well.
- (1) One participant indicated that because of the increasing complexity of the tasks, performing later tasks was not easier.

Editing and Collaboration

- (2) Two participants suggested implementing a variable inspector to get a real time rendered preview of mathematical objects, similar to functionality found in MATLAB.
- (1) Collaboration by hosting the notebook on GitHub and using the GUI to request comments is mentioned by one participant. *While this sounds feasible, support for version control of notebooks is limited, including line-by-line comments on GitHub and diffing of notebooks allowing easy merging.*

Miscellaneous

- (1) To provide highly detailed feedback on the user answer, one participant requested support for providing feedback beyond symbolic equivalence answer matching. The participant immediately stated that this would possibly make the currently fairly straightforward programming model more complex. *This request is definitely something being thought of. The sketch for implementing this is supplying a custom Python evaluation function as a string to the stateless Google Cloud Function used for evaluation. This custom function should accept a LaTeX answer expression string as input, and return HTML as feedback. By running this function on Google Cloud, sandboxing is managed by Google and computational limits can be configured to manage timing issues (e.g. unanticipated infinite loops).*
- (1) For complex procedures in linear algebra, such as solving a system of equations, a participant suggested implementing the functionality to define multiple answer stages. For example, stage one is bringing the system in a certain form (e.g. echelon form), stage two is bringing the system in a subsequent next stage (e.g. reduced echelon form), stage three is performing an operation on the matrix in this form etc. *A suitable approach for creating exercises for complex procedures remains future work. The reusability of earlier created exercises and answer rules for less complex procedures composing complex procedures should be considered an important part of this.*
- (1) After using the functionality of providing answer specific feedback, a participant suggested revealing statistics of the given answer to the user, if sufficiently available. This participant noticed that during lectures, students pay careful attention to slides entitled *Common Exam Mistakes*. Therefore, seeing that your mistake is a typical mistake might help students paying extra attention to the error in their answer. *Your answer is incorrect, 60% of your fellow students made this mistake at least once when answering this exercise*, might be suggestion for information shown to the student. *Implementing this type of functionality is certainly possible, however it requires careful design. Considering that some students might follow a 'trail and error' approach compared to others being more conservative in using the check answer button, should the first answer attempt be compared to first attempts of others or to all attempts? Additionally, what are the effects on learning of constantly being compared to others, especially for the least performant students? These and other considerations come in play when designing this policy.*

4.5 Functional suitability

The developed software system is partially evaluated using ISO 25010: quality characteristics of a software product. Eight characteristics are distinguished: Functional Suitability, Performance Efficiency, Usability, Compatibility, Reliability, Security, Maintainability and Portability. One of these characteristics is discussed: Functional Suitability. Functional suitability is covered since it is most closely related to the research question relating to functional requirements: flexibility and access to a high level of abstraction. The descriptions for the listed characteristics originate from www.iso25000.com³⁶ and are stated in *italic*.

Functional Suitability *This characteristic represents the degree to which a product or system provides functions that meet stated and implied needs when used under specified conditions. This characteristic is composed of the following sub-characteristics:*

Functional completeness Degree to which the set of functions covers all the specified tasks and user objectives.

By allowing authors to access a scripting language (Python) and the available libraries therein, including SymPy and NumPy, users get access to high level abstractions to author the content of linear algebra exercises. This is demonstrated by authoring the matrix multiplication exercise defined in the Problem Statement in Chapter 2, using the provided abstractions. Furthermore, in case desired functionality is not directly available, authors can create helper functions for authoring the exercise at hand. Still, there are limitations with respect to the functional completeness, most of which originate from desired functionality not currently present in SymPy. These can be addressed by extending the functionality of SymPy.

- **Obtaining symbolic value references from SymPy matrix instances** Since SymPy is created for expressing intent with respect to computations, options for getting LaTeX representations of objects for creating explanations are sometimes limited. Consider creating an exercise asking a user to identify a value at a given position in a matrix:

<p>Given</p> $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad (4.1)$ <p>What is $a_{1,1}$?</p>
--

This is possible using the following code:

```

1     e = ""
2     Consider the matrix  $A$  below, what is the value at  $a_{1, 1}$ ?
3     <p align="center">
4      $A = @a$ 
5     </p>
6     ""
7
8     params = {}
9     params["a"] = sp.Matrix([[1, 2], [3, 4]])
10
11    e3 = Exercise(MarkdownBlock(e, params))

```

It is not possible to get the symbolic LaTeX representation $a_{1, 1}$ from the indexed matrix SymPy object. The expression `params["a"][0, 0]` obtains the value at that position. However, there is no way to get the corresponding reference as a LaTeX expression $a_{1, 1}$ out of this expression for creating parameterized exercise content or explanation. Obtaining a symbolic matrix for creating explanations requires custom functions, as demonstrated in Chapter 3.

- **Configuring requested answer simplification** For answer evaluation, currently only simplified expression equivalence is available: basic operations are evaluated (e.g. +, -, *, /), while complex operators are not (e.g. vector addition). Therefore, numeric answers, accepted within a certain margin, are not supported (e.g. the user answers 1.110 while the correct answer is 1.111). Additionally, requesting and strictly accepting only the simplest form of an answer expression (e.g. maximally simplifying fractions), is not supported. This limits the configuration of the strictness with which the answer should be given, resulting in less specific guidance toward the correct answer.

- **Highlighting errors in user answer matrices** It is not possible to provide a custom answer evaluation function (per exercise), providing highly specific answer feedback and error-highlighting. This restricts the possibilities of providing answer-specific feedback to the end user. For example, value specific errors within matrices cannot be easily highlighted. As such, consider the following exercise:

Compute:

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad (4.2)$$

Consider the user answer $\begin{bmatrix} 2 \\ 4 \\ 9 \end{bmatrix}$, containing a wrong value in the last row. By subtracting the user answer from the given answer, the non-zero entries in the resulting vector indicate the error position in the user answer. While this vector can be returned to the user for feedback, a more elegant solution is highlighting the error in the given answer. SymPy does not support markup for the latter.

- **Parsing unsupported symbolic answer expressions** Only LaTeX strings that can be parsed to SymPy objects are supported. For example, answering by symbolic indexing of a matrix is not possible. As such, the following exercise is not possible, since the LaTeX expression string $a_{2,1}$ cannot be translated to a SymPy object:

Given:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad (4.3)$$

What is the symbolic reference of the value 3 in this matrix?

Expected answer: $a_{2,1}$

Similarly, only LaTeX answer expressions are supported that can be produced by the MathLive formula editor, however, most likely this is not a limiting factor because of the broad range of MathLive compatible LaTeX.

Functional correctness *Degree to which a product or system provides the correct results with the needed degree of precision.*

- Since SymPy and MathLive are not formally compatible, it is possible that LaTeX produced by SymPy objects is not correctly shown in MathLive (which in turn partially relies on KaTeX for rendering LaTeX in the browser), because MathLive only supports a subset of LaTeX.
- Similarly, LaTeX expressions produced by MathLive possibly do not properly parse to the desired SymPy objects, since SymPy only supports object representations for a subset of LaTeX.

Functional appropriateness *Degree to which the functions facilitate the accomplishment of specified tasks and objectives.*

For creating and playing a set of elementary linear algebra exercises, the system setup is considered appropriate, because:

- End-users are able to complete the vast majority of exercise authoring tasks as requested in the usability study, within reasonable time.

-
- A Grasple user noticed the increased efficiency by direct access to the abstractions available in SymPy and NumPy.
 - The matrix multiplication exercise defined in the problem statement in Chapter 2 can be created (effectiveness), written in reasonable time (efficiency) by authors (a similar exercise is part of the usability evaluation) and can include explanations generated by high-level helper functions.

Chapter 5

Conclusion & Future Work

This final chapter begins with a concluding section with respect to the research question and performed evaluations. Next, research limitations are discussed. To enable future students conducting similar research to learn from issues encountered, a recommendations section is added. Finally, future work related to the research question is listed and suggestions for further directions are discussed.

5.1 Conclusion

Given the identified requirements from educational literature and authors, a new setup for authoring linear algebra exercises is developed. This setup is primarily inspired by the exercise string templating used in MEGUA and the open answer handling of GraspLe using SymPy. The Jupyter Notebook environment is used to author and preview exercises. Editing exercise content and specifying answers rely on developed Python classes. These classes in turn rely on the LaTeX input and output facilities of SymPy, and the Python Markdown library for generating the exercise content.

The setup allows authors to use Python as a scripting language for generating exercise content. This facilitates working at a high level of abstraction by means of using the available libraries for content generation or by creating custom functions. The flexibility of this setup is demonstrated by the contextualized linear algebra problems related to classifying hand written digits, discussed in Chapter 3. By means of visualized, parameterized, open answer exercises with guiding feedback based on answer characteristics, students get an in-depth understanding of the mathematical procedures required in applied problem solving.

The developed solution is evaluated in two focus groups to collect suggestions in a conversational way, and in a usability evaluation to collect feedback about authoring exercises.

The feedback from the focus group confirmed the identified functional requirements. Additional requirements provide possible directions for further extending this work.

The usability evaluation revealed some usability issues, most importantly, the lack of expected prior knowledge with respect to the NumPy library and lack of support for debugging rendered exercises in case parameters are not wrapped in the required dollar signs for rendering LaTeX. One of the three authors completed all authoring tasks perfectly, while two authors completed three out of the four tasks due to mentioned issues. In general, authors appreciated the flexibility of the designed solution for the tasks at hand. Authors reported that the amount of code required to solve the tasks matches their expectations and that the developed functionality maps to the domain concepts (exercise content, answers and feedback) as expected. In conclusion, the setup is considered usable for the four linear algebra exercise authoring tasks in the usability study for the three participating authors.

5.2 Research limitations

The performed research has multiple limitations.

First, the size of the evaluation groups are small and the amount of exercise authoring tasks in the usability evaluation is few. This limits the generalizability of the feedback and the linear algebra operations to be covered with exercises respectively. In case more participants would have joined the study, these limitations could be overcome.

Second, while the developed software and evaluation procedures are publicly available, the reproducibility of the feedback from the evaluations depends on the participants, being not explicitly mentioned.

Third, two of the seven participants are known to the author personally prior to conducting the evaluation studies. The existing relationship possibly limits the amount of critical feedback provided in order not to affect the relationship.

Finally, the formulation of the problem, collection of the requirements, implementation of the new setup, evaluation procedure and execution, analysis and reporting are all executed by a single person. Preferably, solution developers do not execute the evaluation of the implemented solution. Their prior knowledge and desire to deliver positive results can influence the outcome.

5.3 Recommendations

This section briefly discusses recommendations regarding the evaluation procedures. These recommendations inform future students conducting similar research.

Asking participants to use tooling in focus groups Properly guiding collective use of Miro is nontrivial. Participants should be educated on the possibilities of Miro and the desired features to be used should be discussed. To have everyone actively contributing, the requested contributions should start easy and effortless and only when everybody is aboard the complexity of the contributions can increase. Performing this guidance online, which is necessary due to COVID-19, complicates this task.

Observing author performance in usability evaluation To get detailed insight in the authoring process, logging intermediate versions of exercises while authoring could help understand the authoring process and identify problems.

5.4 Future work

Regarding the requirement stated in the research question, allowing exercises to be authored with a high level of abstraction and parameterization, three topics for future work are suggested.

Complexity of exercises Because of the high level abstractions available in SymPy, it is easy to generate an instance of a complex exercise. For example, generating a matrix with random values and asking the user to compute the QR or LU decomposition is straightforward by using the `QRDecomposition()` and `LUDecomposition()` methods of the `Matrix` class to define the answer. However, providing specific feedback or guiding the user through the steps within this procedure is not straightforward. Similarly, generating the LaTeX code for a complete worked out solution of the problem instance at hand is not facilitated by SymPy. SymPy, a computer algebra system, provides high level

abstractions for modeling problems, the procedure is typically not relevant for the user, as long as it is performant. In learning, understanding the steps in the procedure is key. Therefore, it might be worth investigating the possibilities for creating a library or even a domain specific language for the sake of instructing (and explaining) mathematical procedures.

Diversity of exercises Systematically authoring (linear algebra) exercises beyond the subset of exercises discussed in this work, for example by converting static book exercises to digital counterparts, should provide more insight in the power and limitations of the designed solution.

Measuring effectiveness and efficiency The effectiveness and efficiency gain of the developed solution can be evaluated by a comparative user study, asking authors to complete authoring tasks using different solutions. This would require carefully selecting suitable metrics for comparison.

5.5 Further directions

This includes broader topics thought of throughout project execution.

Improved editor support: real time preview, syntax highlighting and autocomplete Most Markdown editors have a preview screen to see a live preview of the rendered content while authoring. Similar functionality can possibly be implemented, making the authoring process more convenient. Additional functionality to improve the authoring experience includes syntax highlighting of the exercise template strings and automatic completion of defined parameters therein.

Extensions to the format and player: answer templates and sequential hints The exercise format and player can be extended with additional features supporting the learner. Answer templates guide the student towards the correct answer. Instead of showing the definition of a concept or procedure as feedback on a wrong answer, similar feedback can be used in an answer template. Within this template, students can easily replace the symbolic placeholder values with values in the problem instance. Alternatively, empty placeholder templates can be used instead. Examples for both cases of this functionality on the notion of cosine similarity between two vectors is shown in Figure 5.1.

EXERCISE 1

The cosine similarity between two vectors $a = [a_1 \ a_2 \ \dots \ a_n]$ and $b = [b_1 \ b_2 \ \dots \ b_n]$ is defined as:

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

Compute the cosine similarity between $a = [9 \ 9]$ and $b = [5 \ 4]$.

(A) Symbolic placeholder values

(B) Blank placeholder values

FIGURE 5.1: Cosine similarity answer template

Next to answer templates, sequential hints can be used. Instead of providing a single hint as default feedback on an incorrect answer, first a minor hint is shown. Only in case this hint is not sufficient to complete the exercise, more information is provided step-by-step.

Real-time collaborative exercise authoring Within Grasple, authors can collaborate by editing the same exercise at different points in time. Real time collaboration is possible by 'pair-programming' exercises, for example using the remote screen sharing and mouse/keyboard takeover functionality of Zoom. It could be interesting to compare authoring time and the exercise content (and implementation) between pair-programming couples and individual authors.

Student contributed exercises Since computer science students are likely to be familiar with Python and Markdown, it would be feasible to encourage students to create exercises for their peer population. This allows students to create rich artifacts of their learning process, being exercises demonstrating their understanding of the material at hand. In Bloom's revised taxonomy [1], this activity corresponds to the highest layer: create. Doing so benefits both the authoring student and fellow students, and vastly increases the amount of practice material available. To monitor the quality and utility of exercises, social feedback loops can be utilized. For multiple choice questions, this idea is implemented by Denny et al. [11] in a system called PeerWise. PeerWise is used in multiple university courses, including an introductory physics [3] and biology course [24] at the University of Edinburgh.

Appendix A

Usability Evaluation Interview Questions

This appendix includes the questions used in the semi-structured interviews for the usability evaluation. Sections A.1 to A.4 are after [28], the questions in Section A.5 are informed by interests of GraspLe and finally A.6 allows for any additional comments.

A.1 Understandability

1. Do you find that the API types map to the domain concepts in the way you expected?
2. Do you feel you had to keep track of information not represented by the API to solve the tasks?
3. Does the code required to solve the tasks match your expectations?

A.2 Abstraction

1. Do you find the API abstraction level appropriate for the tasks?
2. Did you need to adapt the API (inheriting from API classes, overriding default behaviours, providing non-API types) to meet your needs?
3. Do you feel you had to understand the underlying implementation to be able to use the API?

A.3 Reusability

1. Does the amount of code required for each tasks seem about right, too much, or too little for you?
2. How easy was it to evaluate you own progress (intermediate results) while solving tasks?
3. Do you feel you had to choose one way (out of many) to solve a task in the scenario?

A.4 Learnability

1. Once you performed the first two tasks, was it easier to perform the remaining tasks?
 2. Do you feel you had to learn many classes and dependencies to solve the tasks?
-

A.5 Editing and collaboration

1. Do you prefer this way of authoring over a WYSIWYG environment?
2. Do you see any merit in higher-order helper functions, not available in SymPy/NumPy you can now write? For example: step-by-step explanations, generation of specific matrices.
3. How would you see collaboration with non-power users?
4. Do you think this format allows for collaboration and reuse? Would you be able to understand/adjust an exercise created by someone else?

A.6 Miscellaneous

1. Are there features you missed while executing the tasks?
2. Anything else you want to comment on?

Appendix B

Usability Evaluation Notebook

This appendix includes the notebook used for the usability evaluation. The notebook contains an introduction to the exercise format by means of explanation and examples and four exercise authoring tasks.

Usability Evaluation Notebook

May 25, 2021

```
[1]: from sympy.matrices import Matrix
import sympy as sp
import numpy as np
from Exercise import Exercise, MarkdownBlock

from process_latex import process_sympy

try:
    from config import URL, TOKEN
except:
    None

# TODO: replace with supplied strings
Exercise.URL = ""
Exercise.TOKEN = ""
```

0.1 Introduction

In this notebook, you are about to create some (linear algebra) exercises using the developed Exercise Python library aiming to facilitate authoring parameterized mathematics exercises at a high level of abstraction (i.e. access to a scripting language and the libraries available in there, including as SymPy, NumPy and Matplotlib). Created exercises can be 'played' inline, using the web-based player developed as part of this project. Roughly speaking this project is new combination of existing approaches: MEGUA-like parameterized text, SymPy's CAS functionality and exercise-setup as used by GraspLe and SageMath for working with mathematical objects in notebooks.

The goal is to evaluate the usability of the developed library and the authoring setup (this notebook). Note that by no means you or your skills are being tested, it is by no means a problem if exercises are left uncompleted. Notes, comments and suggestions are very welcome, please write these either as code-comments or in the Markdown cells in the notebook. All feedback will be reported and reflected upon anonymously. Completing the notebook should take about 30 minutes, depending on setup time, prior knowledge about this project, familiarity with linear algebra and the supplied frameworks etc. Please download the notebook when done and send it by email. After completion, in a brief semi-structured interview, you can further elaborate upon your experiences.

To start creating exercises, please replace the URL and TOKEN in the block above with the strings supplied by email:

```
Exercise.URL = "<supplied_url_here>"
Exercise.TOKEN = "<supplied_token_here>"
```


Assumptions: - Familiarity with Python, Markdown, LaTeX - Familiarity with Jupyter-Notebook - Familiarity with the very basics of linear algebra

Recommendations: - Use Binder (www.mybinder.org) to edit this notebook, if you prefer local setup instead, see README.md. - Use Firefox, the iFrame exercise player embeddings do not work in Chrome or Safari due to global cross-origin policies set by these browsers. - Other browsers (Chrome, Safari) can still be used, however, playing exercises is only possible outside of the notebook by clicking the generated exercise links, which is rather inconvenient.

Notes: - Documentation can for the Python library can be found in the `html` directory. - Within Jupyter-Notebook, function documentation can be viewed by writing a `?` after the function, like so: `Exercise("What is $1 + 1$?").add_answer?` - Within exercises, only inline math notation is supported. - Preview-exercises are purged from the server from time to time, don't expect long-term, persistent availability of any played exercises. - Please skip an exercise in case completing it requires more than a few minutes.

Happy coding ;)

0.2 Exercise Basics

The most basic exercise contains a Markdown string with the exercise content and a single answer rule specifying the correct answer. Mathematics notation can be written inline in LaTeX between dollar signs.

```
[2]: # Create an exercise instance
e = Exercise("What is  $1 + 1$ ?")
# Add 2 as a correct answer
e.add_answer(2, True, "Correct!")
# Verify that the exercise is working correctly
e.play()
# Note: as of now, all basic arithmetic is simplified by sp.simplify(...),
# →there is not yet a way to control this behaviour;
# therefore writing  $1 + 1$  in the answer box is accepted correct
# Details on what is simplified: https://docs.sympy.org/latest/tutorial/simplification.html
```

<IPython.lib.display.IFrame at 0x7f6db198d850>

Published successfully, preview at:

<https://www.mscthesis.nl/preview?id=232910bc-4856-465f-a04e-7c191a18367a>

Let's imagine the typical student mistake for this exercise is computing $1 - 1 = 0$ instead. We add an answer rule to catch that error and provide the student with answer-specific feedback.

```
[3]: e.add_answer(0, False, "That's not right, did you compute  $1 - 1 = 0$  instead?
# →")
# Verify that the specific feedback is shown
e.play()
```

<IPython.lib.display.IFrame at 0x7f6de45c6a10>

Published successfully, preview at:

<https://www.mscthesis.nl/preview?id=9b044fa6-5830-4a38-b85c-4082fed9c92b>

0.2.1 Task 1

Create an exercise asking learners to compute $3/3$. Provide answer-specific feedback in case learners compute $3 * 3$ instead. Add default feedback (using `e.add_default_feedback(...)`) with a link pointing to a source of preference explaining (integer) division (hint: `[link](www.example.com)`). Feel free to embed your favorite meme or xkcd at a correct/incorrect answer (hint: `[img](www.example.com/img)`).

```
[4]: # Task 1 user code:
```

0.3 Templating Exercises

Exercises can be parameterized/templated (still looking for the correct terminology on this one), this allows for two things: 1. Randomization. By making part of the content random, multiple instances can be generated, allowing for repeated practice. 2. Abstraction. By utilizing the functionality of SymPy objects to be translated to LaTeX, authoring exercises remains efficient and effective.

The integer-exercise can be randomized as follows:

```
[4]: string = """
### Integer addition

Please compute  $@a + @b$ 
"""

params = {}
# avoid 0 + 0 instance, since 0 + 0 == 0 - 0, answer same in case our typical
# mistake is made
params["a"] = np.random.randint(0, 10)
params["b"] = np.random.randint(1, 10)
params["ans_correct"] = params["a"] + params["b"]
params["ans_incorrect"] = params["a"] - params["b"]

e = Exercise(MarkdownBlock(string, params))
e.add_answer(params["ans_correct"], True, "Correct!")
e.add_answer(params["ans_incorrect"], False, MarkdownBlock("Did you compute
 $@a - @b = @ans_incorrect$  instead?", params))

e.play()
```

<IPython.lib.display.IFrame at 0x7f6db13fa7d0>

Published successfully, preview at:

<https://www.msctheisis.nl/preview?id=c9f8db25-bdcc-4def-9443-36160609aa86>

Currently, only a single instance is generated played at a time. Support for multi-instance generation is planned.

0.3.1 Working with SymPy objects to represent mathematical objects

We can work with SymPy objects to represent mathematical objects, like vectors and matrices. An vector addition exercise can be created as follows:

```
[6]: string = "What is @$v_1 + @$v_2$?"

params["v_1"] = sp.Matrix([1, 2, 3])
params["v_2"] = sp.Matrix([4, 5, 6])
params["ans"] = params["v_1"] + params["v_2"]

e = Exercise(MarkdownBlock(string, params))
e.add_answer(params["ans"], True, "That's right!")

e.play()
```

<IPython.lib.display.IFrame at 0x7f6db13c1d10>

Published successfully, preview at:

<https://www.mscthesis.nl/preview?id=f547846d-30b5-4689-a899-15b7c381b974>

0.3.2 Task 2 Parameterized vector addition

Create an exercise asking learners to compute the sum of two vectors of random length (within reasonable limits), with random integer values. Note: if you prefer NumPy for working with matrices, you are in luck! NumPy objects can be passed to the SymPy matrix constructor, e.g. `sp.Matrix(np.arange(4))`.

```
[7]: # Task 2 user code:
```

0.3.3 Task 3 - Matrix indexing

Create an exercise asking learners to identify a value at randomized indices (but within bounds) in a 5 by 5 matrix. Please make sure all values are unique so there is only one correct answer.

```
[8]: # Task 3 user code:
```

0.3.4 Task 4 - Matrix multiplication

Create an exercise asking users to multiply two matrices. Provide a default answer explaining the procedure in case a wrong answer is supplied. You can use the `symbolic_matrix` and `explain_multiply` functions supplied in `helpers.py` as follows:

```
[7]: from helpers import symbolic_matrix, explain_multiply
a = symbolic_matrix("a", 2, 2)
b = symbolic_matrix("b", 2, 2)
display(explain_multiply(a, b))

a = sp.Matrix([1,2,3])
b = sp.Matrix(np.matrix([5,6,7]).reshape(-1))
display(explain_multiply(a, b))
```

$$\begin{bmatrix} a_{1,1} \cdot b_{1,1} + a_{1,2} \cdot b_{2,1} & a_{1,1} \cdot b_{1,2} + a_{1,2} \cdot b_{2,2} \\ a_{2,1} \cdot b_{1,1} + a_{2,2} \cdot b_{2,1} & a_{2,1} \cdot b_{1,2} + a_{2,2} \cdot b_{2,2} \end{bmatrix}$$

$$\begin{bmatrix} 1.5 & 1.6 & 1.7 \\ 2.5 & 2.6 & 2.7 \\ 3.5 & 3.6 & 3.7 \end{bmatrix}$$

[20]: `# Task 4 user code:`

Hooray! If you made it this far, you completed the notebook! Please add any additional comments below. Thank you for participating!

Write any additional comments here...

Appendix C

Code Listings of Selected Exercises

This appendix includes the code listings of exercises discussed in Chapter 3.

CODE LISTING C.1: Columns in matrix, vanilla

```

1 m = """
2 Consider the matrix  $A$ , how many columns does  $A$  have?
3 <p align="center">
4  $A = @a$ 
5 </p>
6 """
7
8 def generator():
9     params = {}
10    rows = np.random.randint(2, 10)
11    columns = np.random.randint(2, 10)
12    matrix = sp.randMatrix(rows, columns, min=0, max=15)
13    params["a"] = matrix
14
15    e = Exercise(MarkdownBlock(m, params))
16    e.add_answer(sp.simplify(columns), True, "Correct!")
17    if rows >= columns:
18        e.add_answer(sp.simplify(rows), False, "Nope, that's the amount
19                               of rows.")
20
21    return e
22
23 Exercise.write_multiple(generator, 100, "matrix_cols")

```

CODE LISTING C.2: Columns in matrix, visualized

```

1 m = """
2 Consider the matrix randomart image below, how many columns does it
3 contain?
4 <p align="center">
5 
6 </p>
7 """
8
9 def generator():
10    params = {}
11    rows = np.random.randint(2, 10)
12    columns = np.random.randint(2, 10)
13    matrix = sp.randMatrix(rows, columns, min=0, max=15)
14    params["a"] = matrix
15    matrix_to_image(np.array(matrix, dtype=float), "m", grid=False,
16                    values_in_cells=False,
17                    axis_indices=False, axis_titles=False)
18
19    e = Exercise(MarkdownBlock(m, params))
20    e.add_answer(sp.simplify(rows), True, "Correct!")

```

```

18     if columns >= rows:
19         e.add_answer(sp.simplify(columns), False, "Nope, that's the
                                         amount of columns.")
20     return e
21
22 Exercise.write_multiple(generator, 10, "cols_in_image")

```

CODE LISTING C.3: Matrix indexing, vanilla

```

1  m = """
2  Consider the matrix  $A$ , what is the value at  $a_{@i, @j}$ ?
3  <p align="center">
4   $A = @a$ 
5  </p>
6  """
7
8  def generator():
9      params = {}
10     rows = np.random.randint(2, 10)
11     columns = np.random.randint(2, 10)
12     matrix = sp.randMatrix(rows, columns)
13     params["a"] = matrix
14     i = np.random.randint(0, rows)
15     j = np.random.randint(0, columns)
16     params["i"] = i + 1
17     params["j"] = j + 1
18
19     e = Exercise(MarkdownBlock(m, params))
20     e.add_answer(sp.simplify(matrix[i,j]), True, "Correct!")
21     if i < columns and j < rows:
22         e.add_answer(sp.simplify(matrix[j,i]), False, "You seem to have
                                         swapped the row and column
                                         index.")
23
24     return e
25 Exercise.write_multiple(generator, 100, "matrix_indexing")

```

CODE LISTING C.4: Matrix indexing, visualized

```

1  m = """
2  Consider the matrix image  $A$  below, what is the value at  $a_{@i, @j}$ ?
3  <p align="center">
4  
5  </p>
6  """
7
8  def generator():
9      params = {}
10     rows = np.random.randint(2, 10)
11     columns = np.random.randint(2, 10)
12     matrix = sp.randMatrix(rows, columns, min=0, max=15)
13     params["a"] = matrix
14     i = np.random.randint(0, rows)
15     j = np.random.randint(0, columns)
16     params["i"] = i + 1
17     params["j"] = j + 1
18
19     matrix_to_image(np.array(matrix, dtype=float), "m", grid=True,
                     values_in_cells=True,
                     axis_indices=True, axis_titles=
                     True)
20
21     e = Exercise(MarkdownBlock(m, params))

```

```

22     e.add_answer(sp.simplify(matrix[i,j]), True, "Correct!")
23     if i < columns and j < rows:
24         e.add_answer(sp.simplify(matrix[j,i]), False, "You seem to have
                swapped the row and column
                index.")
25     params["d"] = symbolic_matrix("a", rows, columns)
26     e.add_default_feedback(MarkdownBlock("Remember how values are indexed
                : $0d$", params))
27     return e
28
29 Exercise.write_multiple(generator, 10, "matrix_indexing_image")

```

CODE LISTING C.5: Creating matrix from description

```

1  from helpers import *
2  import random
3
4  m = """
5  Place the following values at the right positions in a matrix:
6
7  <p align="center">
8      ${values}$
9  </p>
10 """
11 def generator():
12     params = {}
13     character = "a"
14     rows = np.random.randint(2,4)
15     columns = np.random.randint(1,3)
16     araw = [[[Symbol(f"{{{character}}}_{{{i+1}}, {j+1}}")], np.random.
                randint(0, 100)] for j in range(
                columns)] for i in range(rows)]
17     for r in araw:
18         random.shuffle(r)
19     random.shuffle(araw)
20     a = sp.Matrix(araw)
21
22     v = ""
23     ans = np.empty((rows, columns), dtype=int)
24     for i, row in enumerate(araw):
25         for j, col in enumerate(row):
26             v = v + f"{{latex(col[0])}} = {{col[1]}} , "
27             ans[i][j] = col[1]
28
29     # remove trailing space (latex wrapping should be tight e.g. $a,$
                instead of $a, $) and comma ($a$
                instead of $a,$)
30
31     v = v[:-2]
32     e = Exercise(MarkdownBlock(m.format(values=v), params))
33     e.add_answer(sp.Matrix(ans), True, "Indeed")
34     return e
35 Exercise.write_multiple(generator, 100, "values_in_matrix")

```

CODE LISTING C.6: Distance score between binary digits

```

1  m = """
2  Given the matrices
3
4  <div class="d-flex flex-1 flex-items-center">
5  $A = $ , $B = $ 
7  </div>

```

```

7
8 ##### Task
9 Frist, determine  $D = |A - B|$ , then compute  $\sum_{i=1}^{} \sum_{j=1}^{} d_{i,j}$ 
10
11
12 f1 = """
13 Correct!
14
15  $D =$  
16
17  $\sum D = @sum$ 
18 """
19
20 f2 = """
21 <div class="d-flex flex-1 flex-items-center">
22 Hint:  $D =$  
23 </div>
24 """
25 def generator():
26     # to vector: .reshape(-1, 1)
27     a = to_binary(nums[np.random.randint(1700)])
28     b = to_binary(nums[np.random.randint(1700)])
29     d = np.abs(a-b)
30     matrix_to_image(a, "a", grid=True, values_in_cells=True, axis_indices
31                     =True)
32     matrix_to_image(b, "b", grid=True, values_in_cells=True, axis_indices
33                     =True)
34     matrix_to_image(d, "d", grid=True, values_in_cells=True, axis_indices
35                     =True)
36
37     e = Exercise(m)
38
39     ans = int(np.sum(d))
40     e.add_answer(expression=ans, correct=True, feedback=MarkdownBlock(md=
41                     f1, params=dict(sum=ans)))
42     e.add_answer(expression=ans-1, correct=False, feedback="You are close
43                     , please check your answer")
44     e.add_answer(expression=ans+1, correct=False, feedback="You are close
45                     , please check your answer")
46     e.add_default_feedback(feedback=f2)
47     return e
48
49 Exercise.write_multiple(generator, 10, "digit_matrices")

```

CODE LISTING C.7: Distance score between gray scale digits

```

1 m = """
2 Given the matrices
3
4 <div class="d-flex flex-1 flex-items-center">
5  $A =$  ,  $B =$  
7 </div>
8 ##### Task
9 Defining  $D$  as  $D = |A - B|$ , compute  $\sum_{j=1}^{} d_{1,j}$ 
10
11
12 f1 = """
13 Correct!
14
15  $D =$  

```



```
16
17  $\sum D = @sum$ 
18 """
19
20 f2 = """
21 <div class="d-flex flex-1 flex-items-center">
22 Hint:  $D = \sum$  
23 </div>
24 """
25 def generator():
26     # to vector: .reshape(-1, 1)
27     a = nums[np.random.randint(1700)]
28     b = nums[np.random.randint(1700)]
29     d = np.abs(a-b)
30     matrix_to_image(a, "a", grid=True, values_in_cells=True, axis_indices
31                     =True)
32     matrix_to_image(b, "b", grid=True, values_in_cells=True, axis_indices
33                     =True)
34     matrix_to_image(d, "d", grid=True, values_in_cells=True, axis_indices
35                     =True)
36
37     e = Exercise(m)
38
39     ans = np.sum(d[0,:])
40     e.add_answer(expression=ans, correct=True, feedback=MarkdownBlock(md=
41                 f1, params=dict(sum=ans)))
42
43     e.add_default_feedback(feedback=f2)
44     return e
45
46 Exercise.write_multiple(generator, 5, "digit_vectors_grayscale")
```

Bibliography

- [1] Lorin W Anderson, Benjamin Samuel Bloom, et al. *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*. Longman, 2001.
- [2] Philip K Axtell et al. "Developing math automaticity using a classwide fluency building procedure for middle school students: A preliminary study". In: *Psychology in the Schools* 46.6 (2009), pp. 526–538.
- [3] Simon P Bates, Ross K Galloway, and Karon L McBride. "Student-generated content: Using PeerWise to enhance engagement and outcomes in introductory physics courses". In: *AIP Conference Proceedings*. Vol. 1413. 1. American Institute of Physics. 2012, pp. 123–126.
- [4] Jo Boaler. "Open and closed mathematics: Student experiences and understandings". In: *Journal for research in mathematics education* (1998), pp. 41–62.
- [5] Lynn A Bryan et al. "Integrated STEM education". In: *STEM road map: A framework for integrated STEM education* (2015), pp. 23–37.
- [6] M Caprile et al. "Encouraging STEM studies for the labour market". In: *Directorate General for Internal Policies, European Union* (2015).
- [7] W Cox. "On the expectations of the mathematical knowledge of first-year undergraduates". In: *International Journal of Mathematical Education in Science and Technology* 32.6 (2001), pp. 847–861.
- [8] Fergus IM Craik and Robert S Lockhart. "Levels of processing: A framework for memory research". In: *Journal of verbal learning and verbal behavior* 11.6 (1972), pp. 671–684.
- [9] Pedro Cruz, Paula Oliveira, Dina Seabra, et al. "Exercise templates with Sage". In: *Tbilisi Mathematical Journal* 5.2 (2012), pp. 37–44.
- [10] Christoph Deeken, Irene Neumann, and Aiso Heinze. "Mathematical Prerequisites for STEM Programs: What do University Instructors Expect from New STEM Undergraduates?" In: *International Journal of Research in Undergraduate Mathematics Education* 6.1 (2020), pp. 23–41.
- [11] Paul Denny, Andrew Luxton-Reilly, and John Hamer. "The PeerWise system of student contributed assessment questions". In: *Proceedings of the tenth conference on Australasian computing education-Volume 78*. Citeseer. 2008, pp. 69–74.
- [12] Keith Devlin. *The joy of sets: fundamentals of contemporary set theory*. Springer Science & Business Media, 2012.
- [13] Gabrielle Garon-Carrier et al. "Intrinsic motivation and achievement in mathematics in elementary school: A longitudinal investigation of their association". In: *Child development* 87.1 (2016), pp. 165–175.
- [14] James J Gibson. *The ecological approach to visual perception: classic edition*. Psychology Press, 2014.

- [15] Ulrich Heublein, Robert Schmelzer, and Dieter Sommer. “Die Entwicklung der Studienabbruchquote an den deutschen Hochschulen”. In: *HIS-Projektbericht, Hannover* (2008).
- [16] Hyungshim Jang. “Supporting students’ motivation, engagement, and learning during an uninteresting activity.” In: *Journal of Educational Psychology* 100.4 (2008), p. 798.
- [17] Paul Arthur Kirschner and Carl Hendrick. *How learning happens: Seminal works in educational psychology and what they mean in practice*. Routledge, 2020.
- [18] Richard A Krueger. *Focus groups: A practical guide for applied research*. Sage publications, 2014.
- [19] Jean Lave. “Situating learning in communities of practice.” In: (1991).
- [20] Erno Lehtinen et al. “Cultivating mathematical skills: From drill-and-practice to deliberate practice”. In: *ZDM* 49.4 (2017), pp. 625–636.
- [21] Gustavo Martínez-Sierra and María del Socorro García-Gonzalez. “Undergraduate mathematics students’ emotional experiences in Linear Algebra courses”. In: *Educational Studies in Mathematics* 91.1 (2016), pp. 87–106.
- [22] Andrew McConney et al. “Inquiry, engagement, and literacy in science: A retrospective, cross-national analysis using PISA 2006”. In: *Science Education* 98.6 (2014), pp. 963–980.
- [23] Nancy J McCormick and Marva S Lucas. “Exploring mathematics college readiness in the United States”. In: *Current Issues in Education* 14.1 (2011).
- [24] Heather A McQueen et al. “PeerWise provides significant academic benefits to biological science students across diverse learning tasks, but with minimal instructor intervention”. In: *Biochemistry and Molecular Biology Education* 42.5 (2014), pp. 371–381.
- [25] Irene Neumann, Colin Jeschke, and Aiso Heinze. “First Year Students’ Resilience to Cope with Mathematics Exercises in the University Mathematics Studies”. In: *Journal für Mathematik-Didaktik* (2020), pp. 1–27.
- [26] National Audit Office. *Staying the course: The retention of students in higher education*. Vol. 616. The Stationery Office, 2007.
- [27] Thomas J Palmeri. “Automaticity”. In: *Encyclopedia of cognitive science* (2006).
- [28] Marco Piccioni, Carlo A Furia, and Bertrand Meyer. “An empirical study of API usability”. In: *2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. IEEE. 2013, pp. 5–14.
- [29] Leanne J Rylands and Carmel Coady. “Performance of students with weak mathematics in first-year mathematics and science”. In: *International Journal of Mathematical Education in Science and Technology* 40.6 (2009), pp. 741–753.
- [30] Mark Schoenfield and Jeannette Rosenblatt. *Adventures with logic*. Fearon Teacher Aids, 1985.
- [31] Neil H Schwartz. *Kirschner, PA, & Hendrick, C. (2020). How learning happens: Seminal works in educational psychology and what they mean in practice*. Routledge. ISBN 9780367184575. 2020.
- [32] Hanne Shapiro, SF Østergård, and KF Hougaard. “Does the EU need more STEM graduates”. In: *Publications Office of the European Union: Luxembourg* (2015).
- [33] John Sweller. “Cognitive load theory”. In: *Psychology of learning and motivation*. Vol. 55. Elsevier, 2011, pp. 37–76.

- [34] John Sweller et al. “The expertise reversal effect”. In: (2003).
- [35] Alfred North Whitehead et al. *Aims of education*. Simon and Schuster, 1967.

References

- ¹https://www.youtube.com/channel/UCY0_jab_esuFRV4b17AJtAw
- ²<https://www.youtube.com/user/eaterbc>
- ³<https://www.youtube.com/user/khanacademy>
- ⁴<https://medium.com/>
- ⁵<https://medium.com/@adekunle.r.adepoju/how-to-build-a-techy-go-kart-part-1-everything-hurts-443916a2cae5>
- ⁶<https://medium.com/@chengyao.shen/decoding-comma-ai-openpilot-the-driving-model-a1ad3b4a3612>
- ⁷<https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c>
- ⁸<https://medium.com/analytics-vidhya/introduction-to-the-gradient-boosting-algorithm-c25c653f826b>
- ⁹<http://www.corestandards.org/Math/>
- ¹⁰<https://openstax.org/details/books/college-algebra>
- ¹¹<https://www.khanacademy.org/>
- ¹²<https://www.mathsisfun.com/>
- ¹³<https://brilliant.org/>
- ¹⁴<https://grasple.com/>
- ¹⁵<https://sowiso.nl/en/>
- ¹⁶<https://www.slader.com/>
- ¹⁷<https://math.stackexchange.com/>
- ¹⁸<https://www.wolframalpha.com>
- ¹⁹https://github.com/juanklopper/MIT_OCW_Linear_Algebra_18_06
- ²⁰<https://iso25000.com/index.php/en/iso-25000-standards>
- ²¹<https://github.com/arnog/mathlive/pull/796>
- ²²<https://github.com/facelessuser/pydown-extensions/pull/1266>

-
- ²³https://github.com/phoenixframework/phoenix_live_view/pull/1401
- ²⁴https://github.com/phoenixframework/phoenix_live_view/issues/1398
- ²⁵https://studiegids.tudelft.nl/a101_displayCourse.do?course_id=55101
- ²⁶<https://www.tiobe.com/tiobe-index/>
- ²⁷<https://docs.google.com/document/d/1Ti7MXtYOPGMAIFPnDc9PXr9-hyp0R2VTfxqPyVcfEs>
- ²⁸<https://connect.oeglobal.org/t/action-lab-standard-and-format-for-open-interactive-math-exercises/329>
- ²⁹<https://docs.sympy.org/latest/tutorial/manipulation.html>
- ³⁰https://hexdocs.pm/phoenix_live_view/js-interop.html
- ³¹<https://mathlive.io/>
- ³²<https://docs.sympy.org/latest/tutorial/simplification.html>
- ³³<https://facelessuser.github.io/pymdown-extensions/extensions/b64/>
- ³⁴<https://www.tiobe.com/tiobe-index/>
- ³⁵<https://mybinder.org/>
- ³⁶<https://iso25000.com/index.php/en/iso-25000-standards/iso-25010>