The major challenge for eLearning courses on undergraduate mathematics is the diversity of their audience and the varying demands posed by distinct interest groups.

Our proposal to deal with these challenges is the deployment of intelligent assistants that, given some initial knowledge concerning the audience, explore user behavior to build up a sophisticated model of the learner within the system.

Starting with models for learner and courses, we present a prototypical implementation of such a system within the virtual laboratory VIDEOEASEL developed at the TU Berlin.
The workplaces of scientists and engineers are in a phase of transition: numerical software and computer algebra systems remove the burden of routine calculations, but demand the ability to become familiar with new concepts and methods quickly. Traditional “learning on supply” might be able to provide some basic knowledge, but this learning model is rapidly showing its inadequacy in dealing with the rapid growth of knowledge in today’s sciences. Instead, learning and teaching methods have to be established that guide learners towards efficient self-controlled learning. New media and new technologies present a turning point in the educational system since they provide the basis to support the necessary changes (Jeschke & Keil-Slawik, 2004).

In our understanding, mathematics is the most attractive field for developing and deploying this new technology (Jeschke, Kohlhase, & Seiler, 2004; Natho, 2005): first of all, it is the key technology of the 21st century. Studies in engineering sciences, physics, computer science, and many other fields depend on a well funded mathematical education. Teaching mathematics, however, means that diverse backgrounds and varying interests of the audience have to be taken into account. On the positive side, mathematics already provides us with a very clear, precisely formulated language and well worked out internal structure. Therefore, in our opinion, it is well suited for applying methods of computer science.

In the next section, we introduce the concept of “virtual laboratories” as an option for interactive learning in engineering and theoretical studies. In this context, virtual laboratories act as the frame-work within which we explore our ideas on electronic courses, presented in the following section. As we want to optimize our courses to the learner, we introduce the learner model in the following, and then introduce our approach to optimize a course to a specific learner. Unlike earlier approaches, for example, the Felder-Silverman index (Felder & Silverman, 1988), we do not require an ad-hoc classification of learners or learning-styles, but rather model learners by random variables describing their background and learning success from which the system itself learns what suits the learner best.

The concrete implementation of this model into our virtual laboratory is discussed next, followed by the evaluation of the system and our experiences when using the system. We close with concluding remarks.
VIRTUAL LABORATORIES

Virtual Laboratories (Jeschke, Richter, & Seiler, 2005) use the metaphor of a “real,” scientific laboratory, thus providing a framework that emulates a scientific workplace for hands-on training, though implemented in the form of virtual spaces. Virtual laboratories enrich traditional mathematics typically taught in ”proofs” by providing means to access abstract objects and concepts in interactive experiments; they thus build a bridge between the theoretical fields and practical sciences by supplying experiments that run on computer-implemented algorithms either emulating real devices in idealized situations or representing theoretical concepts. Applications of virtual laboratories range from practical support for traditional lectures (e.g., demonstration), over homework assignments and practical training for students up to aiding researchers in experimentation and visualization. Let us stress again that we do not attempt to replace hands-on courses in undergraduate engineering courses; rather, we enrich these courses by experiments that would not be possible otherwise, be it for reasons of resource constraints, or be it because these experiments are conducted on abstract models rather than real devices. For that, they are best considered complementary to remote experiments, which also make experiments available anytime, anywhere, but in contrast to virtual labs are conducted on actual, physical set-ups (Jeschke, Richter, Scheel, Seiler & Thomsen, 2005).

The virtual laboratory VIDEOEASEL (Jeschke, Richter, & Seiler, 2005), developed at the TU Berlin, focuses on the field of statistical mechanics and statistical physics. The driving force for developing this laboratory was to demonstrate how macroscopic laws emerge from microscopic dynamics, and to explore the influence of microscopic physics on the overall-behavior of the system. As such, VIDEOEASEL naturally enables experiments with lattice gases, the Ising model (Ising, 1925; Metropolis, Rosenbluth, Teller, & Teller, 1953), predator-prey systems and phase-transitions (Onsager, 1944) in many-body systems, but is also applicable to present problems of analysis, probability theory, dynamic systems, or methods of image processing.

Figure. 1 shows a practical application of this laboratory as used in a course on mathematical physics: A specific simulation has been loaded into the system, here the Ising model of ferromagnetic material, and the simulation has been run for a short time; the black/yellow area to the left visualizes the result of this experiment: According to the Ising model, ferromagnetic materials consist of microscopic elementary magnets whose dynamics are
simulated within the laboratory. Elementary magnets whose north pole points towards the observer appear as black pixels, those pointing into the opposite direction are colored yellow. As can be seen from the figure, zones of constant magnetization appear after a short period of time—these are called “Weiss Zones” in physics. Similar to a real physical experiment, students are now free to adjust the parameters of the model, for example the temperature, and experiment how the model reacts on its change. Parameters of the model are visualized as sliders by the graphical user interface.

![Image of the Ising model simulated in VIDEOEASEL](image)

**Figure 1.** The Ising model simulated in VIDEOEASEL

Similar to a real laboratory, several measurements can be performed on the simulation by linking the measuring algorithm to the running simulation. In the example of the Ising model, these algorithms would measure the total magnetization, the Helmholtz Free Energy, the entropy, the inner energy, and so forth, of the probe. In Figure 1 a measurement device for the magnetization has been attached to the sample: The green plot on top of the window depicts the outcome of this measurement over time.

As noted, this is just one of the many possible experiments within the laboratory, which can be modified at running time to perform whatever simulation a student or instructor might wish. Virtual laboratories are not by themselves simulations, but rather provide the framework to carry out simulations easily and effectively.
The Ising model may act as an excellent example how we believe virtual laboratories are able to enrich existing lectures: The phenomenon of ferromagnetism can now be exploited in reality, a physical model of the Ising model can be studied alongside in the computer, and the results can be compared with the predictions of a physical theory taught in a lecture, for example by comparing the measured hysteresis loop of a real magnetic coil with that measured in a virtual laboratory (Jeschke, Richter, Seiler, & Thomsen, 2005). Measured data, data from the simulation and the result of a physical theory are then found in close, but—as often—not in total coincidence, and it is an important part of the education of engineers and physicists to understand the differences and the roles of experiment and model.

VIDEOEASEL implements microscopic dynamics by a cellular automaton (Toffoli & Margolus, 1987) that can be dynamically programmed at run-time, and makes its computational results available over a network interface implemented in CORBA (Scallan, n.d.). Students are not only invited to perform experiments that have been carefully crafted by university staff, but also to modify the physical model and thus the experiment itself and study the results. Furthermore, this interface can also be used to link the laboratory with external software such as computer algebra systems, namely to evaluate and visualize the measured phenomenon (Jeschke & Richter, 2006). Last but not least, the very same interface is used to exchange data with an assistant system we will discuss in the following.

As stated, VIDEOEASEL acts also a testbed used in our development of intelligent assistants, software modules that provide interactive course material that adapts to the learner, following some earlier ideas (Pangaro, 2001; Scott, 2001; Krauß & Körndle, 2005; Albert & Lukas, 1999; Albert & Schrepp, 1999). The eLearning module we are developing for this virtual laboratory could be exploited by other systems as well. The intelligent courses are dynamically loaded to a laboratory setup as external Java classes, extending the user interface of VIDEOEASEL by additional windows that present exercises and the evaluation of the learners’ input. More details on the inner workings of this system and its interaction with the virtual laboratory is found in section “Bayesian Learning” after having presented the conceptual model behind it.
COURSE MODEL

To build an effective and realistic model of a course, we propose the following three-level hierarchy of learning units (Figure 2):

A Course is the largest/coarsest unit we currently consider. It is the abstraction of a series of lectures held on a topic at a university. An example topic could be “linear algebra,” Figure 2. Courses are represented by directed graphs whose nodes are Knowledge Atoms of which each encodes one individual learning unit of a course. A learning unit in the field of mathematics could be a theorem, or a definition, or a motivation for a definition, a proof of a theorem, and so on. Typical examples for learning units of “linear algebra” are: The definition of the determinant, the theorem that a matrix is invertible if and only if its determinant is nonzero, and so on.

The edges in this graph are dependencies between learning units: A vertex is drawn from A to B if B is a precondition for A, that is, B must be taught in order to make A understandable. In the example, the definition of the determinant would be the precondition for the theorem about invertibility.

Dependencies themselves are classified into three groups: hard requirements that follow from the ontology of mathematics, as in the example
presented. Recommendations that are necessary for didactic purposes, though not imposed by the mathematical structure: for example, the node describing the determinant could recommend the Gaussian elimination algorithm as one suitable algorithm how to calculate it. And, finally, suggestions an interested student might want to follow, but which are neither required for didactic nor for mathematics-internal reasons. Historic side remarks how the definition of the determinant evolved over time may aid as an example here. Clearly, the network created in this way depends on the ontology chosen for the field. Specifically, the requirement subgraph encodes one possible ontology of linear algebra for the given network. A visualization of the combined recommendation and requirement subgraph is also called “HASSE Diagram” in educational sciences (Krauße & Körndle, 2005; Albert & Lukas 1999).

The learning atoms of a course are, for example, represented in the database of the “MUMIE” system developed at the TU Berlin (Dahlmann, Jeschke, Seiler, & Sinha, 2003; Jeschke, 2004; Mumie community, n.d.), though the dependency network is currently not yet encoded there.

Recommendations in the course network may also link to training units in the exercise network; in general, this is an n-to-m mapping as one knowledge atom might refer to more than one exercise, and one exercise might be useful for more than one knowledge atom.

This second layer in the proposed hierarchy is again a directed graph; however, its nodes are now representing exercises instead of learning units. One exercise defines learning material a student might want to use to repeat and train contents of the lecture. For example, one exercise in the course “linear algebra” would be to compute the determinant of a $4 \times 4$ matrix, or to solve a linear differential equation with constant coefficients. The edges in the exercise network encode dependencies of the exercise units: node A is linked to node B if B is an exercise for a subproblem that is required to solve A. Following these links, a student might be delegated to simpler subproblems of a harder assignment. Similar to this, the edges are annotated by the type of dependency, also including requirements for mathematically dependent subproblems, recommendations, and suggestions. Requirements might be satisfied by more than one node, that is, there are also cases where one out of several requirements is sufficient, see the middle layer in Figure 2 for an example. It will turn out that this freedom is actually a key requirement for efficient deployment of intelligent assistants.
Unlike the content area graph, the annotation is here, to an even larger degree, given by didactic reasoning and not necessarily by the mathematical ontology itself. Similar to the SCORM model (Dodds & Thropp, 2004), one exercise consists of one or several assets on the asset level hierarchy of the network, and here again form the nodes of a directed graph. An asset is one elementary operation that must be performed to solve an exercise, for example, an elementary row operation in the Gaussian elimination algorithm. Edges in the asset graph now define the reaction of the system on user input, for example, performing the wrong operation would redirect the user to an asset that demonstrates why the proposed solution would not work, and hints could be given by the system. At this level, the graph is a representation of a Storyboard (Jantke & Knauf, 2005); a concrete example will be given later.

Currently, the MUMIE eLearning framework (Mumie community) already contains a database containing knowledge atoms for a course on “linear algebra,” though the vertices (and thus their annotations) are missing. The intelligent assistants of VIDEOEASEL presented in the section “A Course in VIDEOEASEL” operate mostly on the asset level, though elements of the training area are already represented. Future developments will have to merge the two systems to couple them at the training level.

**LEARNER MODEL**

An intelligent assistant that operates on parts of this network needs to be equipped with a model for the learner that uses the eLearning software. For study purposes, we currently use a very simple learner/content model that, clearly, needs to be refined later.

A learner is part of a certain community, called the Audience. An audience is defined by a common language and notation, as well as a certain problem class. Examples for audiences would be “electrical engineers,” “physicists,” “mathematicians.” Note that even though the same course material, for example, “linear algebra” has to be taught to all three audiences, the notation, formulation, and exercises to be given will differ significantly. However, this does not necessarily impose that the exercise network will look completely different, or that exercises for one group will be unsuitable for the other. Specifically, the edges dictated by the ontology of the field are
likely to be independent of the audience, though it seems useful to define different representations of the same content and to formulate exercises differently.

Whenever a learner has successfully solved an exercise, the learner afterwards is completely aware of its contents. More precisely, the degree to which a learner has understood one asset can be encoded by a single number $x$, due to the atomicity of the asset, and the same learner, when coming back to the same asset later, will be able to solve this exercise to degree $x$ or better. Specifically, we currently ignore that students forget solutions, or hope that we can ignore this effect on the timescale on which assets within one exercise are typically solved. We furthermore ignore solutions that are obtained by pure luck or trial and error, or hope to be able to detect them by observing the usage pattern: namely, it follows from this simplification that if a student solves A to a better degree than B, but A depends on B, then it is likely that the solution obtained for A has been found by a trial-and-error approach and not by reflecting upon obtained knowledge.

Clearly, our learner model is a very crude simplification—but given that the quality of homework assignments are typically measured in “credit points” students can obtain, it is still a model that is often silently applied in education. Of course, the number “$x$” exactly represents the credit points.

**BAYESIAN LEARNING**

Adaptivity in eLearning systems is often achieved by first classifying users by an external questionnaire that is aimed at identifying the preferred learning style of the student, and then selecting the appropriate material within the system. A popular learning style model is, for example, the Felder-Silverman index (Felder & Silverman, 1988) that describes learners along the dimensions active/reflective, sensing/intuitive, visual/verbal, and sequential/global. Felder and Soloman (1999) developed a questionnaire that aims at identifying the preferred learning style of a student. Several works exist that discuss the validity and consistency of the questionnaire (Felder & Spurlin, 2005; Litzinger, Lee, & Wise, 2005; Zywno, 2003; Viola, Graf, Kinshuk, & Leo, 2006), but the outcome of the studies are controversial. We therefore propose a different approach to solve the adaptivity problem, namely to let the system decide online what is most
useful for a specific learner given the observed learning history directly. Despite that, the Felder Soloman questionnaire could also be used to define parts of the assets of the system—in other words, the proposed solution can be considered a super-set of existing assessment techniques.

As already mentioned in the section on our course model, the content and training material in the course should be designed in such a way that there is more than one possible resolution for the dependencies of a knowledge atom or a training unit. For example, an exercise on computing the determinant of a matrix requires knowledge of either Cramer’s rule, or the Gaussian algorithm, or the Laplace algorithm, and thus students should get hints on these dependencies. It is now up to the system to estimate which hint is likely to provide the best possible learning success for a student of a given audience. To this end, we present now a Bayesian decision system that aims at finding the optimal exercise for a given user.

The Bayesian decision system works on the following probabilistic model, where our “random events” encode the audience (see “Learner Model”) of a student, the learning history, and the learning success. The random event that a student is part of a specific audience is denoted by \( U \), and the event that a learner has successfully managed an evaluation, for example, an exam or an oral test, is called \( S \). It is not necessary to perform this test within the learning system itself as an “electronic” examination with all known deficiencies, but the knowledge on the performance of the students must be made available to the system, of course. Exercises or knowledge atoms are denoted by \( e \in E \), the vertex set of the content or exercise graph \( G(E,F) \), where \( F \) is the corresponding edge set of the graph. Furthermore, denote the random event that the learner has visited nodes \( e_1,...,e_k \) in this order by \( \Phi(1,...,k) \). That is, \( \Phi \) encodes the “learning path” of the student, very much like the history of a browser. Assume now that for a given node \( e \), the learning system reached a “decision point,” namely there are several possibilities to resolve the dependencies of \( e \). In the same formal language, this means that the learning system has to pick one node from the set \( F(e_k):=\{ e \in E \mid (e_k,e) \in F \} \), the set of vertices reachable as outgoing nodes from \( e_k \), and by that extend the learning path by one step. For brevity, we write \( \Phi=\Phi(1,...,k) \) for the unextended and \( \Phi_l=\Phi(1,...,k,l) \) for the learning path that has been extended by the node \( e_l \), the node picked by the system.
The optimization problem is now finding a node \( e_i \in F(e_k) \) such that the probability of passing the test successfully is maximal for the given audience \( U \), namely to maximize \( P(S \mid U \cap \Phi_i) \). Using Bayes’ formula (MacKay 2003), one finds

\[
\max \arg P(S \mid \Phi_i \cap U) = \max_i \arg \frac{P(\Phi_i \mid U \cap S)P(U \cap S)}{P(\Phi_i \mid U)P(U)}
\] (1)

The numerator contains now the probability of finding the extension \( \Phi_i \) of the path in the subgroup of successful learners of the audience \( U \) times the probability of being successful in \( U \), the denominator the similar probabilities for the full audience. All probabilities can be estimated by first running the system through an initial training phase where relative frequencies of all events are measured and the probabilities are estimated, and the system can keep updating the probabilities as students keep using it. By Laplace’s rule (MacKay, 2003), the estimation for the first term of the denominator would be, for example:

\[
P(\Phi_i \mid U \cap S) \approx \frac{|\Phi_i \cap U \cap S| + 1}{|U \cap S| + 1}
\]

where \( |\Phi_i \cap U \cap S| \) is the number of events the subpath \( \Phi_i \) has been found in the observation for successful \( U \) students, and \( |U \cap S| \) is the total number of successful students of audience \( U \). It is easy to find similar expressions for all other probabilities in eqn. (1) to see that finally

\[
\max \arg P(S \mid \Phi_i \cap U) = \max_i \arg \frac{|\Phi_i \cap U \cap S| + 1}{|\Phi_i \cap U| + 1}
\] (2)

that is, the best extension \( e_i \) of the path is that which provided the best ratio of students of the audience \( U \) passing the test so far, quite what one would have expected naively in first place. It thus remains an easy task for the learning system to identify the exercise paths picked by the user by querying a database, and update the counts appropriately as soon as a student fails or passes the final test for an exercise group.
We conclude this section with several remarks: First, to make the estimation (2) useful, numerator and denominator should be large and thus the sample size must be large. This imposes a restriction on the granularity of the audience since a finer granularity results in less members of each audience, reducing the sample size. Similarly, the number of paths to consider should be small enough to have useful sample-sizes. This could be realized by two mechanisms: first, one could define the learning goals small enough and by that limit the number of valid paths, that is, provide a lot of small evaluations within one course. Second, note that the number of possible paths grows very fast with the path length: by conditioning the expressions only by the last N steps taken by a student, the number of possible paths to take into account is also greatly reduced. This restriction of the path length has also the nice interpretation as modeling a system of finite memory, where “memory” quite nicely coincides with the memory of the average student.

A COURSE IN VIDEOEASEL

The virtual laboratory VIDEOEASEL (Jeschke, Richter, & Seiler, 2005), previously introduced, is user adaptive in several ways: first of all, more than one user interface is available; depending on the desired deployment, Java applets, stand-alone Java front-ends, Oorange (n.d.) plug-ins or Maple-plugins are available, Figure 3 (Jeschke & Richter, 2006). We shall not discuss the various front-ends further within this article. The laboratory also comes with a prototypical implementation of an exercise and tutoring system following the course and learner models presented in the three previous sections, Figure 4, and a Bayesian learning strategy optimizer.
Figure 3. Laboratory front-ends: Java front-end (top), orange interface communicating with Maple (bottom)

The Course Model consists of a database keeping elementary asset nodes in a textual representation (see the example in Appendix A). These asset nodes formulate the assignment given to the user, how to evaluate the presented solution and possible reactions to the solution, that is, they encode a storyboard by means of a graph. Furthermore, the asset nodes also encode the dependencies and requirements of an exercise and thus built up the training level of the exercise network. An asset node contains the following information:
Figure 4. A tutoring assistant in a laboratory on matrix convolution. On top of the current assignment to be performed; the middle window shows the configuration wizard for the convolution automaton, the target image is shown below. On top of the target image some feedback given by the assistant.

- A *name*, used to identify the node within a database describing the course.

- A *learning goal* or a learning unit that is provided by the node. The implied model here is that a user who has successfully solved one assignment is now aware of the learning unit defined by this node, thus, the learner model. Furthermore, other nodes can refer to learning material by means of this learning goal and thus define requirements in the sense of a course model.
The definition of a target audience for the given node: several copies of a node providing the same learning goal might exist, differing only in the complexity of the exercises or the formulation of the assignment depending on the course that is to be held.

One or several requirements, that is, learning units the user must be aware of to be able to understand the assignment. The learning system is able to resolve these dependencies itself by checking the user profile for nodes that have been solved successfully, and will automatically continue at one of the requirements in case of not all preconditions being met by the user. As presented in the section on the course model, the system distinguishes between three different types of dependencies. These links encode the dependency graph of exercises. If more than one node is able to resolve the requirement, the Bayesian system provides priorities as which resolution is likely to be the most successful one.

An explanation that is presented to the user in textual form and that defines the assignment that is to be solved.

One or several hints the system might optionally give on request.

One or several evaluators that check the answer the learner provides. It is important to note that an evaluator does not yet try to give a qualified estimation to which degree a learner has understood the learning goal; rather, it returns a string that encodes the returned answer in a compact way and that can be used by the asset node to come to a qualified decision. That is, evaluators are re-usable objects that can be used independently of the exercise. In the current implementation, these evaluators are java classes that are dynamically linked to the tutor program and the laboratory at run time and communicate over the interfaces of both programs to obtain information on the state of both systems.

One or several if-conditions that define the reaction of the system by matching the string returned by the evaluator. To cover several identical cases, the strings against which the evaluator results are matched may contain wild-cards, that is, they are “regular expressions.” These conditions implement the storyboard graph as in Figure 2. For the matrix convolution course shown in Figure 4, evaluators return for example a description of the properties of the convolution filter setup
by the student. It is up to the storyboard to decide whether the solution presented by the user is suitable to the learning target of the exercise unit.

If the user-provided answer matches one of the conditions, the node defines the next learning goal, and thus the next node the storyboard branches to, as well as a short explanation why the system came to this conclusion.

Finally, on a successful answer, the system provides a number of credits for the exercise defined by the node. Given this credit count, the system may decide to raise the complexity of future exercises. This number encodes the degree to which a user understood the assignment, and thus implements the credit system of the simple user model introduced.

The nodes of the storyboard are encoded in a straightforward text file that can be edited by hand as we haven’t developed our own editor for them yet. See Appendix A for an example what the code for an asset on convolution filters in image processing looks like.

The learner model of the tutoring system is implemented as a database which, indexed by the user, keeps information on the audience of the user, and the credits obtained by the user in the asset nodes so far; this information therefore encodes a User Profile of the learner. Given this information, the tutor program can decide which asset nodes the user has visited successfully already or should visit in future assignments, Figure 5. Furthermore, the audience defines which node out of several nodes for a given asset the tutoring program will pick from.
Figure 5. The Learner Model navigation in the exercise/asset network. The evaluator (left) picks a suitable target node for the solution presented by the user. Dependencies between nodes and the knowledge of the user can redirect the user’s attention to a more basic asset. The audience selects the suitable implementation of the node from a set of nodes all providing the same content (here blue and turquoise).

The Bayesian Estimator finally collects the learning paths of students and computes the success ratio for special test questions within a course. It does not use the credit points collected within regular assets, as doing so would drive the system towards optimizing the wrong goal, namely to maximize the credit points: It is hard to assign the credit points consistently throughout a course, and it is thus likely unavoidable to have learning paths that gain less and others that provide more credits even though the same learning goal is reached. A strategy that maximizes credits would thus pick the path with the optimal credit count, and not necessarily the path providing the best learning success. Estimation of the learning success and presentation of learning content should thus be separate.

It is furthermore important to note that our Bayesian optimizer only provides hints as to which material to pick next. It provides a list of useful exercises to follow, sorted by expected success ratio, but never attempts to patronize the user. We believe that it is important that the system remains predictable. Furthermore, the system should not enforce a particular learning strategy.
that might or might not be appropriate for an individual learner. A learning assistant must not act invasive, as it is otherwise likely that it won’t be accepted by its users—it then gets more annoying than helping. Furthermore, allowing students to select the material as they need it, not only allows them to decide for themselves what suits them best and thus allows them to interact with and reflect about the material, it also allows us to collect statistical data on unusual course elements that would be otherwise avoided by the system.

**EVALUATION**

The virtual laboratory introduced has been deployed, in the course “Mathematical Physics II” at the University of Technology, Berlin. Even though this course is the major target of the laboratory, it has also been deployed in a high-school/university bridge-course for girls and in several exploration courses for German high-school students. The mathematical physics course teaching thermodynamics and statistical physics targets graduate students of mathematics and physics; amongst others, the Ising model (Ising, 1925) for ferromagnetism is one of the prominent study objects here. Students have been asked to find the critical temperature, to measure the hysteresis loop and to explore the influence of boundary conditions on the physics of this model, as simulated by VIDEOEASEL. One of the homework assignments given to them was to plot the measured Helmholtz free energy and the magnetization of the model over the magnetic field, then to discuss the resulting plots and compare them to the material presented in the lecture. Figure 6 presents one of the plots obtained by a student, showing the Helmholtz free energy to the left, the magnetization to the right, with colors encoding different temperatures.

The majority of the student groups conjectured correctly a differential relation between magnetization and energy; several groups even derived this relation theoretically within their homework from the material taught in the lecture, even though a proof of this relation was not specifically asked for. An anonymous survey of the students subsequent to the course revealed that most of the students appreciated the experiments and found them very motivating and instructive for understanding the concepts of the lecture. They often found the experiments enlightening because they were able to see the mathematics “in action” and thus were able to see the impact of the
theoretical concepts to the experiments performed. A second interesting result was the offhand interest of many students in the mathematical background of the simulation itself, and the software architecture of the laboratory. It thus became apparent that our setup is also able to stimulate interest beyond the curriculum of a lecture.

![Figure 6. Plots of the Helmholtz free energy (left) and magnetization (right) of the Ising model, as obtained by a student using the discussed virtual laboratory in a homework assignment.](image)

The agent system has not gone through systematic testing yet, but we already employed the existing prototype in the bridge course for girls and observed their behavior; due to the smallness of the class a statistically meaningful evaluation of the performance, or the performance gain reached by using the agent system over a nonadaptive system unfortunately has not been possible. Most students had little or no problems using the system, but preferred to contact a human tutor in case of questions rather than asking the
system for hints or further suggestions, which is often available by a press of a button. Problems also appear in areas we cannot cover directly within the laboratory, as in how to use the graphical user interface in general, that is, how to use the mouse, how to move windows around and detect buttons obscured by other windows, and so forth. Once these initial barriers are managed, students often become comfortable with the system. We plan to make systematic tests on larger classes as soon as more learning material becomes available that focuses on larger audiences.

On the author side, we found that constructing courses and establishing the course network is burdensome work, especially since the only tool available to construct the material is an editor. It often happens that circular dependencies get into the system that must be avoided, and that requirements or recommendations remain unsatisfied. These logical “errors” could only, until very recently, be detected by extensive testing. We are now also providing an independent course test tool that can detect a major class of the mentioned errors in the course setup, and we plan to provide a graphical course creator for the laboratory courses to ease the job of the author.

**CONCLUSIONS**

We presented a virtual laboratory system providing experiments in statistical mechanics and related fields, thus allowing a hands-on approach to theoretical physics and mathematics. We also presented a conceptual framework for designing courses in an eLearning system and described the integration of this framework into the virtual laboratory introduced. By that, we are able to offer interactive courses in mathematics and physics that include experiments as an integral part of the eLearning system and thus clearly go beyond multiple-choice questionnaires often found in established systems. The presented system is highly modular and can be extended externally by the addition of further experiments and courses.

In addition, we presented a probabilistic approach on measuring and optimizing the learning success, based on a very simple learner model that avoids some conceptional problems of existing systems for user-adaptivity. Our tutoring system follows a storyboard that has to be prepared by the lecturer of an audience, and thus follows hard links between asset nodes given by the edges in the asset graph. Even though exercises can be adapted
by the system to a certain degree, means to escape this network are still limited. It should be noted that the user should be allowed to “escape from the storyboard” in future versions of our tutoring system because a tightly and statically linked asset network might possibly kill the user’s creativity. This problem also needs addressing in the design of courses: A course should rather evaluate the overall success of an experiment instead of “micro-managing” the student from one mouse-click to the next, that is, training units should be sufficiently large to allow students a certain amount of creativity on how to reach the goal and assets should be limited to giving specific help or providing introduction material.

Unfortunately, our work on Bayesian estimation has not been carefully tested on larger classes and thus our statistical material is not yet appropriate to make it efficient. On the other hand, traditional means to make systems adaptive to the learner, for example, by classifying the learner according to the Felder-Silverman Index (Felder & Silverman, 1988), have— at least to some studies—neither shown a large statistical significance (Viola 2006).

Our system is different in the sense that it does not assume or need an ad-hoc classification of learners, but rather gains this knowledge from the user in exactly the learning situation it tries to optimize.

An important aspect here is that the system must remain transparent and predictable. The decision made by the system, for example, why which asset node has been chosen, have to be made apparent to the learner, and the final say on a decision should be left to the learner. Otherwise, it is likely that students will not accept the feedback provided by the tutoring system since they are unable to comprehend the reasons on which the decision has been based.

References


APPENDIX A
EXAMPLE CODE FOR AN ASSET NODE

node = {
    name = "exercise3";
    provides = "ShiftByConvolution";
    requires = "DefinitionConvolution";
    recommends = "InvertibleOrNot";
    suggests = "CellularAutomaton";
    explanation = "Please modify the convolution operation such that it moves the image to the right.;"
    hint = "A convolution filter works like a weighted average over the neighborhood of one pixel. However, for shifting an image, how many neighbors does the new pixel value depend on?;"
    audience = "Basic";
}
using (evaluator = "CheckForMover") {
    if ("isMover\(1,0\)") {
        target = "exercise4";
        explanation = "Well done! Let's do the next exercise";
        credits = 10; }
    if ("isMover\(-1,0\)") {
        target = "exercise7";
        explanation = "The selected convolution filter moves pixels to the left, not to the right. Note that the convolution filter computes the new pixel value from the weighted sum over its neighbors. In case the left neighbor weight is set to +1, the new pixel value will be the value of the pixel to the left, thus moving this spot one to the left.;"
        credits = 3; }
    if ("isMover\(0,0\)") {
        target = "exercise3";
        explanation = "The selected filter did not move the image at all. Have you seriously tried to play a bit with the settings?;"
    }
    if ("isMover\(.\)") {
        target = "exercise7";
        explanation = "The selected filter moved the image to the left and up, not to the right. Let's try a simpler exercise.;";