Teaching the Foundations of Computational Science on the Undergraduate Level

C. Freundl, H. Köstler, and U. Rüde

Chair for System Simulation, University of Erlangen-Nürnberg, Germany
Christoph.Freundl@informatik.uni-erlangen.de
Harald.Koestler@informatik.uni-erlangen.de
Ulrich.Ruede@informatik.uni-erlangen.de

Abstract. This paper describes a new course introduced at University of Erlangen-Nürnberg to teach the foundations of computational science for undergraduates at the end of their second year. The course is a mandatory part of the curriculum in three different computer science-related degree programs. The paper discusses the motivation for introducing the course, the topics chosen for it, and the teaching scheme developed for this unique course.

1 Motivation

The University of Erlangen-Nürnberg offers programs in Computational Engineering (Bachelor, Master, PhD), Computer Science (Diplom Informatik), and Information and Communication Technology (Diplom). All three programs rely on a joint set of core course modules for the first two years of study. Beginning this year, a new required course was introduced for students in the fourth semester of these programs, entitled *Data Structures and Algorithms for Continuous Systems*.

The birth of this new course originates form a discussion within the department that many current computer applications deal with objects that are fundamentally of continuous nature. Example applications include the analysis and synthesis of images, processing of audio signals and movie sequences, and simulation. Arguably, these topics are gaining in importance for computer scientists and computational engineers. However, these topics were not covered in the curriculum of the first two years of any of these programs.

Continuous models have also become important to model extremely large data sets and are being used for data mining or web search engines. Many modern algorithms in this area depend on intricate techniques from numerical linear algebra and numerical analysis.

This, however, stands in contrast to the state to which our computer science curriculum had evolved over the past two decades. Topics, such as numerical linear algebra had essentially disappeared from the undergraduate curriculum and would only appear within optional courses for higher semesters, e.g. as part of taking a minor in mathematics.

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part II, LNCS 3992, pp. 185–192, 2006.

Of course, advanced classes on computer graphics, visualization, image processing, pattern recognition, signal processing, and simulation require these topics as a prerequisite, and therefore many of these courses included introductory material to cover their mathematical prerequisites. Clearly, this setup did not exploit that there is a shared theoretical basis. As a consequence, this had lead to a significant overlap in advanced level courses.

The new course *Data Structures and Algorithms for Continuous Systems* has been designed to address this problem and to make the overall curriculum more efficient by teaching the foundations in a unified form, with the goal of saving time and enable the higher level courses to progress more quickly to advanced material.

An early analysis of the desired topics for the course showed that a simple revitalization of a conventional numerical analysis course would not satisfy our requirements (see [1, 2, 3]). We have therefore decided to develop a completely new course based on the material extracted from existing advanced courses in pattern recognition, computer graphics, and simulation.

This approach was also chosen in the hope that it would be easier in this way to motivate students to study topics that many of them perceive as difficult and highly abstract. On the other hand, the designer of the course and its instructors face a challenge to balance the presentation of fundamental theory with the motivation through practical examples.

In summer 2005, the new course was given for the first time. Since no text books for such a unique course were available, the course has been taught based on a set of self-developed power point presentations which were made available to the students as a substitute for lecture notes and as a guideline to find the relevant material from the various textbook sources.

The remainder of this paper reports on our first experience with teaching this course. It is organized as follows. In section 2 we will outline the material taught in the lecture itself. Section 3 describes the design of the exercise classes, selected assignments are presented in section 4. A concept for instructing unexperienced tutors is shown in section 5. Finally, section 6 discusses evaluation results for this course.

2 Contents of the Lecture

The course *Data Structures and Algorithms for Continuous Systems*[1] consisted of a set of 26 two-hour lectures corresponding to the typical length of the summer semester in Germany.

The course began with an introduction to continuous data sets with examples such as audio or video signals, covering topics such as quantization, discretization, the curse of dimensionality, rounding, floating point arithmetic, stability, and the condition of a problem.

[1] Website of the course: `http://www10.informatik.uni-erlangen.de/Teaching/Courses/SS2005/AlgoIII/`

As a second major topic area, the course covered basic algorithms from numerical linear algebra, like the LU- and Cholesky factorization, QR decompositions, and also data structures for matrices, including sparse matrix formats. For iterative methods, it was decided to present only elementary relaxation methods, and neither Krylov space methods nor multigrid methods. This part of the course also included a review of norms for vectors and matrices.

Wherever possible, this material was motivated by applications as mentioned in Section 1, e.g. by emphasizing the geometric interpretation of rotations, reflections, and orthogonalization. When possible, the use of these methods in applications was explicitly worked out, e.g. the use of linear algebra for the coordinate transformations in computer graphics. Similarly, the lecture on least squares methods was given in the context of examples from pattern recognition and a more involved example was taken from the algebraic reconstruction technique within computed tomography, including a brief excursion to Tikhonov regularization. An introduction to vector space techniques for text retrieval and the page rank model of Google was also presented.

Generally speaking, the linear algebra part of the lecture was driven less by the mathematical analysis of the algorithms, but rather by their application. Nevertheless, the goal of the lecture was to present these techniques as a general toolbox for a wide variety of problems and to make the students aware of issues like the algorithmic complexity or numerical stability. Additionally, the course attempted to give guidelines for the practical implementation of the algorithms by discussing data structures and tools such as Matlab.

The next topic in the class revolved about data structures and algorithms for geometric objects, such as curves, surfaces, and volumes. Consequently, this part of the course started from polynomial approximation and interpolation, to introduce spline functions and Bézier curves. From there the lecture progressed to (tensor-product) surfaces, transfinite interpolation, and Bézier surfaces, and finally volume models, such as Octrees or constructive solid geometry, as used in computer aided geometric design. This part also included basic methods for numerical quadrature, motivated by computing volumes and weights of geometric objects.

A further topic covered in the course was optimization, and nonlinear equations. This included bisection, Newton's method, the Nelder and Mead simplex method, and steepest descent. Because of their practical importance, the course also covered some methods for discrete optimization, such as dynamic programming, backtracking (with heuristics), simulated annealing, and genetic algorithms.

Finite differences and finite elements were covered briefly in a two hour lecture each, and together with the discussion of the FFT algorithm, they concluded the course.

3 Design of the Exercise Classes

Typical for the German system, the course had the format of a lecture (given to the whole class of about 150 students) and was accompanied by tutorial sessions

(*exercises*) which were broken up to limit the number of students to at most twenty in each group. These exercises consisted of two parts:

- class room sessions where the mathematically oriented assignments were presented and discussed, and
- programming assignments.

Organizing the exercise sessions required a major effort. As another peculiarity of the German university system, the active participation in the exercises cannot be required of the students, since they only need to pass the final exam successfully. The success of the students is solely determined by their grade in the final exam which is formally independent of the lecture, and which — as another anachronism of the system — is given at earliest two months after the end of the semester. As an option the students can even choose to take this exam even at a later time, e.g. eight months after the end of the course. This ridiculous situation is one reason for the poor motivation of many students in such courses and the ineffective learning of students lacking a sufficient degree of self-discipline and self-motivation. As an example, more than a third of the participants of our course have chosen to defer the exam by more than half a year.

These constraints led us to develop a special strategy to motivate students: the exercises had to be sufficiently interesting and self-motivating as possible. On the other hand, with only limited teaching capacity and the high student numbers, the exercises had to be given in an efficient way.

3.1 Classroom Exercise Sessions

The exercise classes were given weekly as two-hours sessions and in groups of at most twenty students. The assignments were handed out and posted on the web in the week prior to the classes. Students were expected to prepare the assignments, but — as described above — were not formally required to do the assignments. Therefore, in the class itself the solution of the tasks was presented, but some time in each session was reserved for the students to work out the solutions on their own with the additional possibility of asking the exercise tutors for help. Depending on the difficulty of a task, the solution was then presented by either a student or the tutor on the whiteboard.

3.2 Programming Assignments

To additionally motivate students to do the work intensive programming assignments, they were cast in the form of a contest. To reduce the teaching load, students were grouped in teams of three. There were only four, but then somewhat larger programming assignments. Furthermore we set up a system based on the versioning software *Subversion* [4] which allowed the students to hand in their solutions electronically. Using Python [5] scripts, we provided mechanisms to evaluate the correctness of a solution automatically such that the students got immediate feedback on the quality of their solution. All the tasks required solving a numerical problem so the correctness of a solution could be determined

by measuring norms of residuals or errors. Also, we rewarded an efficient programming style by incorporating the runtime of a program into the score. The system provided a high score of the best solutions on the course's web site, thus promoting a competition among the strongest student teams. This did help to keep the contest interesting and has proven quite effective in giving especially the good students an additional motivation.

3.3 Results and Insights

The classroom exercise sessions were frequently and steadily attended by many students. This may also have been caused by the fact that we did not hand out or post master solutions. In all classes, at least some of the students showed high effort and enthusiasm in solving the given assignments. Nevertheless, as any of the more mathematically demanding classes, the course was found quite difficult by a majority of the students and many of them gave up early on finding their own solution.

The programming assignments showed different results. Although the level of participation was quite high in the beginning (34 groups with a total of about 75 students), only four groups (with nine students altogether) handed in solutions for all of the given tasks. Given the effort to create and present these assignments, this is of course quite disappointing. We believe that this unsatisfactory participation was caused by:

1. Since each assignment was quite work intensive, many students were discouraged from working on them.
2. The evaluation software initially had some technical problems, and was not always quick enough to provide feed back to the students, especially when the deadline was approaching.

Summarizing, the system used for the course was favoring the strong students who put into the course a large effort and who then commented favorably on the learning experience. The disadvantage was that many of the average and below average students were quickly getting frustrated and — not quite unexpectedly — essentially gave up on many of the assignments.

This could of course be corrected most easily by making participation in the exercises a requirement for passing the course, but the mentioned peculiarities of the German university system currently prohibit a more effective teaching setup. This anachronism is partly inherited from the traditional German university structure, and is additionally aggravated by the yet unfinished Bologna reform[2].

4 Selected Assignments

Over the whole semester the students had to do four programming assignments. For a somewhat more detailed exposition, we pick here the third one that was

[2] The Bologna Process aims at introducing a Europe-wide unification of the various national University systems. Besides introducing a Bachelor/Master degree structure it also requires the introduction of a standardized credit point system by year 2010.

motivated from medical image processing. The goal was to register two non-aligned images from the same patient e.g. in order to locate the position of a tumor in each image. This led us to use a variational approach for non-rigid image registration and thus mathematically to a system of nonlinear partial differential equation representing the deformation between the images [6]. The students were given parts of a Matlab program that performs the image registration for a simple test case. Their task was to implement the registration algorithm in C or C++ and to present some results for arbitrarily chosen images. The assessment in this case was on the one hand done by measuring the error of the registration algorithm and on the other hand every group had to hand in a video sequence showing the transformation from one image into the other. These sequences were then rated by the other students. The best ten groups got additional bonus points. Figure 1 shows the image sequence of a deformation process to register the first with the last image in the sequence.

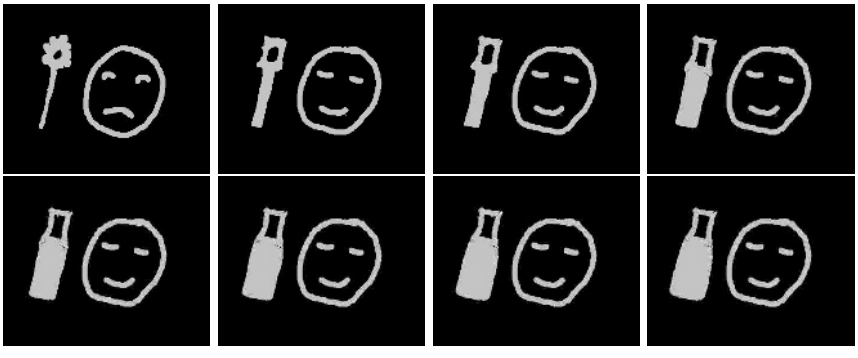

Fig. 1. Reference images (top) and template images (bottom) after the deformation

5 Supervised Teaching

Usually several student tutors from higher semesters are employed to teach some of the exercise classes. These students are supervised by regular university staff — e.g. an advanced graduate student or post doc. Within the context of the Elite Network of Bavaria [7] we have recently been awarded special funds to do this in a more systematic way and to introduce a scheme to *teach the teachers*.

The student tutors were participating in the elite program and thus they could earn credit for participating in our setup of *supervised teaching*. For this, they were required to prepare each assignment set and the corresponding exercise class fro presentation and discuss this before they gave the class with their supervisor. The supervisor additionally sat in the class itself and provided feedback to the instructor after each class. Typical discussions included e.g. how the instructor managed to explain the material to the audience or to what extent he or she could motivate the younger students to participate actively in the class.

Supervised teaching additionally included a one-day seminar given by the head of the Didaktikzentrum Nürnberg [8]. Here the focus lay again on general teaching skills, e.g. a discussion of several presentation techniques and types of leadership. Many practical tips and tricks for teaching were given, e.g. how to deal with students who try to disturb a lesson or how to motivate them.

In addition to to the individual supervison and the seminar, we had arranged a professional coaching of one of the exercise classes by a professional trainer. Here, one class session by the student tutor was videotaped and discussed in detail after class. The goal of this was to help young instructors with technical teaching skills like the right use of the blackboard, but also to systematically develop their teacher personality and to improve their interaction with the students.

The feedback from the first student participating as an instructor in "supervised teaching" was very positive. He felt that he himself had learned intensively since teaching required a high level of abstraction and a deep knowledge of the underlying material in order to be able to teach it well for the younger students. He felt that the responsibility to express material technically correctly and precisely presented a special challenge in itself, and he has found that he could already profit from these skills on several other occasions.

It remains to note that also the participants in the classes taught by student tutors were very positive, since this may have lead to better prepared exercise sessions overall.

6 Evaluation

The Faculty of Engineering Sciences of the University Erlangen-Nürnberg conducts a systematic evaluation of all courses through a web-based questionnaire of all enrolled students.

We cite a few student comments.

- "Mr Rüde tries hard to arouse interest in the fairly complex subject, and this with success. Particularly, I enjoyed the references to practical applications in image processing and the shown videos."
- "Absolutely interesting, I will probably choose this as a major subject. The lecturer is very capable, friendly, and stands relatively close to us students. Points out, what his subject is, what you can do with it, and which perspectives it offers."
- "Subject is way too difficult, very many topics. Better leave out a few issues and go more into details, give more examples or else comprehension will be a torture."

Altogether, the evaluation showed average ratings compared to similar classes. Since this class was given for the first time and was still in a prototype stage, we believe that this is a sign that we are on the right track to develop a good course. More in detail, the students have strongly acknowledged and commented positively on our effort as instructors both for the central lecture and for the exercise sessions. Some of their constructive suggestions, such as an encouragement to work more practical examples on the blackboard during in the central

lecture (as compared to using only power point slides) could even be realized in the ongoing semester and this was generally well received. However, the students commented in their majority quite critically on the topical composition of the course as too theoretical and too mathematical, and generally as too difficult. The disappointing turnout in the programming exercises together with this feedback will require a re-evaluation of the course goals and possibly an adaption of its contents.

Acknowledgments

The authors wish to thank Dr. Ch. Alberternst for the professional coaching of the supervised teaching students, Dipl.-Päd. J. A. Wendorff for giving the seminar of didactics and Q. Meyer for giving feedback on the supervised teaching.

References

1. Rüde, U.: Computational Engineering Programs at University Erlangen-Nuremberg. In: Proc. of the 2002 International Conference on Computational Science (ICCS2002), Part III. Volume 2331 of Lecture Notes in Computer Science., Amsterdam, The Netherlands, Springer (2002) 852–857
2. Bungartz, H.J.: Some remarks on CSE education in Germany. In: Proceedings of the 2004 International Conference on Computational Science: ICCS 2004. Volume 3039 of Lecture Notes in Computer Science., Heidelberg, Springer (2004) 1180–1187
3. Fabricius, U., Freundl, C., Köstler, H., Rüde, U.: High performance computing education for students in Computational Engineering. In Sunderam, V., Albada, G., Sloot, P., Dongarra, J., eds.: Computational Science - ICCS 2005. Volume 3515 of LNCS., Springer (2005) 27–35 ISBN-10 3-540-26043-9, ISBN-13 978-3-540-26043-1, ISSN 03-2-9743.
4. CollabNet: Subversion version control system (2000–2005) `http://subversion.tigris.org`.
5. Python Software Foundation: Python (2001–2005) `http://www.python.org`.
6. Modersitzki, J.: Numerical methods for image registration. Oxford University Press (2004)
7. Bavarian State Ministry of Sciences, Research and Arts: Elite Network of Bavaria web page (2005) `http://www.elitenetzwerk-bayern.de`.
8. DiZ: Zentrum für Hochschuldidaktik der bayerischen Fachhochschulen (2005) `http://www.diz-bayern.de`.