The challenges of developing computational physics: the case of South Africa

T Salagaram[1] and N Chetty[1,2]

[1] Department of Physics, University of Pretoria, Pretoria, 0001, South Africa
[2] National Institute for Theoretical Physics, Gauteng, 2000, South Africa

E-mail: trisha.salagaram@up.ac.za

Abstract. Most modern scientific research problems are complex and interdisciplinary in nature. It is impossible to study such problems in detail without the use of computation in addition to theory and experiment. Although it is widely agreed that students should be introduced to computational methods at the undergraduate level, it remains a challenge to do this in a full traditional undergraduate curriculum. In this paper, we report on a survey that we conducted of undergraduate physics curricula in South Africa to determine the content and the approach taken in the teaching of computational physics. We also considered the pedagogy of computational physics at the postgraduate and research levels at various South African universities, research facilities and institutions. We conclude that the state of computational physics training in South Africa, especially at the undergraduate teaching level, is generally weak and needs to be given more attention at all universities. Failure to do so will impact negatively on the countrys capacity to grow its endeavours generally in the field of computational sciences, with negative impacts on research, and in commerce and industry.

1. Introduction

Computational physics has been an important research tool in physics for over 40 years. It has enabled physicists to understand complex problems more completely compared to using theoretical and experimental methods alone. A generation ago, many postgraduate students working in computational physics research groups around the world would have been required to develop computational codes as part of their thesis work. This was during the infancy of computational physics, and this endeavour was confined only to a small number of specialised research groups around the world. Computational physics then was largely a research endeavour with very little formal training at the undergraduate level. Research students were simply expected to acquire the necessary computational skills as they proceeded with their research work. This resulted in highly skilled computational physics graduates, but these individuals were very few and far between.

This trend has changed significantly over recent times where more and more now do computational physics research groups rely on more sophisticated commercial software, or freeware computer programmes produced by more specialised groups for their research work. The advantage is that more researchers now have access to more efficient codes that incorporate more complex modelling of physical systems of interest. An ever-increasing body of researchers is now able to use such codes in their research endeavours, including experimental scientists. The downfall is that we are now producing students who are mostly schooled in utilising

large computational codes by applying these to physical systems of interest with very little fundamental knowledge and understanding of the intricate details of the underlying theoretical, algorithmic and programming aspects. The research focus now has shifted away from computational methods toward applications. It is reasonable to ask whether these graduates should be referred to as computational physicists or computational technicians.

It is our view that computational physics students should be involved in all aspects of the computational project: theoretical modelling, algorithmic design, numerical methods, programming development, applications to physical problems of interest, analysis of results, and the graphical presentation of the results which could involve animation and visualisation (see Table 1). In so doing, computational physics students learn useful marketable skills that are transferable to other disciplines involving the modelling of problems in engineering, chemistry, biology, ecology, finance, economics, and so on. These skills are absolutely vital in a developing country such as South Africa. This is important to enable commerce and industry to move significantly ahead in terms of developing in-house computational applications, for example in the South African mining industry, rather than to rely on commercial codes developed abroad which sometimes are of questionable relevance for local conditions.

Aside from preparing students for the workplace and being an important research tool, computational physics is also of great pedagogical value. Evidence suggests that allowing students to study complex problems using computation improves their analytical skills and enhances their understanding of the underlying physics[1]-[4]. Getting students to design physical models to explain and predict phenomena can also correct weaknesses in traditional approaches to teaching physics, which often involves using analytical techniques to solve simple problems.

The 2012 IUPAP Conference on Computational Physics held at Kobe, Japan, held a panel discussion on the importance of computational physics, and the question was asked whether computational physics should be viewed separately from theoretical and experimental physics. We believe that the methods of computational physics are best learnt by considering applications to real physical systems and therefore the endeavour of computational physics should not be viewed in isolation of theoretical and experimental physics.

Despite the benefits, it remains a challenge to include computational physics in the undergraduate physics curriculum. Although some physics departments at South African universities teach computational physics at the undergraduate level this trend is yet to fully develop.

There is enormous potential for computational physics to develop rapidly on the African continent. With the advent of cheaper computers, faster networking, access to foreign supercomputing centres and the availability of freeware software, computational physics will be able to thrive in Africa if a greater effort is made at developing this subject more fundamentally at the pedagogical level. This can be a strong basis for developing physics more generally in Africa where investing in expensive experimental infrastructure is always going to be a challenge for many years into the future.

We conducted an in-depth survey of the teaching programmes in all physics departments to draw our conclusions about the development of computational physics in South Africa. We were especially interested to learn whether there was any computational physics being taught at each of the South African universities and, if so, what is the mode, approach and content of the teaching of this subject at these institutions. We also used the survey to determine whether the undergraduate courses are equipping students with the widest range of computational skills, and to determine the local challenges of including computational physics in the undergraduate physics curriculum.

We also considered the pedagogy of computational physics at the postgraduate and research levels at various South African universities, and research facilities and institutions. Here,

Table 1. Different approaches to the teaching and practice of computational physics

Fundamental computational physics	**Algorithmic physics**	**Applied computational physics or Modelling**
The practitioner develops large production codes or short computational exercises *ab initio*	The practitioner uses high-level languages, e.g. MathLab, Mathematica, Easy Java Simulations, etc, to develop short computational exercises	The practitioner operates large production codes in black box-mode
The practitioner engages in all aspects of the computational project: theoretical and mathematical modelling, algorithmic development, programming, graphical visualisation of the data, analysis of the results	The practitioner develops good algorithmic understanding of the problem, but the detailed numerical methods are glossed over	The practitioner only has superficial understanding of the algorithmic and programming details since the focus is on the application and on the analysis of the results
$\Rightarrow$ Computational physicist	$\Rightarrow$ Algorithmic physicist	$\Rightarrow$ Computational technician or Modeller

we were especially keen to scrutinise those research programmes that were considered to be computationally intensive. How do these programmes ensure that the postgraduate training goes well beyond simply teaching students to utilise computational codes in producing the ubiquitous computational technician, to imparting the full range of skills that are absolutely essential for the complete development of the computational physicist?

The data for this survey was obtained from course information on physics department websites, telephone interviews of staff members at various physics departments, personal visits to physics departments and feedback from a questionnaire which was sent to university physics departments. The questionnaire was based on the details given in reference[5]. We also surveyed most of the major research institutions in South Africa to determine their approach to the training in computational physics at the postgraduate and research levels.

2. Undergraduate training in computational physics

There are 17 traditional and comprehensive universities in total in South Africa. Information on computational physics could not be obtained for one traditional and four comprehensive universities. We excluded universities of technology from our investigation since these universities primarily offer physics only as a service course to other diploma or degree courses. Our results show that computational physics is taught in some way or form at 8 universities at the undergraduate level.

The most comprehensive degree programme in computational physics is offered at the University of KwaZulu-Natal (UKZN). Students who enrol for a degree in computational physics take physics as a full major, computational physics as a half major and one other half major subject drawn from mathematics or computer science. At second year level students are given an introductory computational physics course with a focus on basic numerical methods using the Fortran programming language where they display their results using Gnuplot in an open source environment. The applications are mostly in classical mechanics where the problems quickly become fairly complex, for example chaotic systems. At the third year level there is a close mirroring of modules taught in the mainstream physics programme and the computational physics programme. In this way students cover the relevant theory in the mainstream physics

modules, and they are able to focus on algorithmic development and programming of problems in a wide range of topics including problems in quantum mechanics, statistical physics and solid state physics. By the end of the third year, the students acquire very advanced computational skills that enable them to embark on very complex projects at the postgraduate level, for example in astronomy. These skills have made these graduates highly employable in a wide range of industries, such as at the CSIR, Element Six, ISSI, etc., and many graduates have gone on to successfully pursue postgraduate studies at other universities and at other research institutions.

The University of Stellenbosh offers separate computational physics modules within its mainstream physics programme. Two computational physics modules are taught at second year level and one at third year level. The second year modules are designed to teach students the basics of computational methods in the context of solving classical mechanics and electrodynamics problems. Students use either Matlab or the Python programming language, and students may choose to work either in a Windows or a Linux environment. The third year module places a strong emphasis on the statistical skills necessary to do experimental analysis in the laboratory. Students write code either in C or Java within the Ubuntu Linux environment. A major computational project involves the students studying the Ising model.

The University of Pretoria teaches computational physics within the mainstream physics laboratory programme at second and third year levels. The course instructors have chosen to use symbolic programming in the teaching of this subject. In second year, students are introduced to symbolic programming using the Maple software package. They can work either in the Windows or Linux environments. In second and third year, students use the default solvers within the Maple environment to study various computational problems from classical and quantum mechanics, statistical physics and non-linear physics.

The University of Witwatersrand includes elements of computational physics in the laboratory courses from first to third year levels. Students are introduced to the use of computers in physics from data capture, processing and the presentation of results to the solution of mathematical models to fit data. At third year level, students have the option of choosing a computationally intensive project using Mathematica, Sigma Plot or Excel. Students are free to program in Python, C++ or Fortran, and may use either the Windows or Linux operating systems.

The University of the Western Cape includes computational physics within the second year laboratory course. The basics of computational physics are taught in the context of solving problems in classical mechanics. The computational problems are similar to the experiments which students conduct so that they can compare experimental data with numerical data. Students program in Python within the Windows environment. At third year level students are given a substantial computational physics project in each semester. They are taught to write scripts to solve problems using Octave, which is freeware, and they work within the Windows environment. The problems for these projects are taken from the mainstream theory courses taught in each semester thereby allowing students to focus on learning computational methods in the laboratory course.

The University of Cape Town includes elements of computational physics in the first year mainstream course, the second year laboratory course, and offers computational physics projects over the vacation period. In first year, students are shown how mathematical equations can be translated into computer code. This is based on the Matter and Interactions curriculum by Chabay and Sherwood[6, 7]. Using vPython simulations, students are able to visualize their dynamical results in real time. In second year, students learn to formulate algorithms and numerical methods by solving classical mechanics problems on the computer. Students have the option to use Matlab or they may program in Python or Java. There is no formal computational physics module taught at third-year level, however students are required to use computers to analyse their experimental results using techniques such as Fourier transforms where the students are required to write a short piece of code which reads in their experimental data and performs a

Fourier analysis. Undergraduate students may take an optional vacation computational project which is usually based on some aspect of research conducted within the department.

The North West University offers computational physics at both its Potchefstroom and Mafikeng campuses. The physics department at the Potchefstroom campus includes computational physics in the third year laboratory course. Two laboratory sessions have been allocated to teaching computational physics. In the first laboratory session students are taught the syntax of Fortran and the Interactive Data Language (or IDL, which is used for data analysis especially in astronomy). They can work either in the Windows or Linux environments. In the second laboratory session they write a program to solve a partial differential equation using the explicit and implicit numerical methods for numerical solutions of partial differential equations, and study the effects of various boundary conditions.

At the Mafikeng campus, computational physics is taught in the laboratory course at second and third year levels. In the second year laboratory course, there are several laboratory sessions allocated to the numerical and experimental study of a ball moving through a viscous medium. Students develop algorithms for the numerical solution of this problem and program in Fortran. The numerical results are compared to actual experimental results to test the accuracy and stability of the numerical methods used. At third year level, students are given mini computational projects on non-linear physics, stellar structure and the one dimensional Schrdinger equation. Students write their own codes in Fortran to solve these problems numerically.

At the University of Johannesburg the approach to teaching computational physics is to give students homework problems based on material covered in the theory courses which have to be solved numerically. These homework problems are designed to teach algorithmic development and programming in a systematic way, and students may use Matlab or Fortran for programming. They are required to work in the Linux environment.

Some of the challenges in not being able to properly teach computational physics at the undergraduate level include: it is difficult to include computational physics in the already full physics curriculum; finding the right balance between taking a fundamental approach to the teaching of computational physics which is time-consuming, and using ready-to-go packages in black box-mode is difficult; there is a lack of qualified lecturers and support staff with expertise in computational physics; there is not sufficient support from physics colleagues, i.e. the teaching of computational physics is still not recognised as being important by some senior colleagues and heads of departments; faculties do not appreciate the distinction between computer science and computational physics; there is not a proper understanding of what the teaching of computational physics entails; there are not a sufficient number of good computational physics text books appropriate at the undergraduate level; there are no dedicated computational facilities, or there is limited laboratory space for a computational physics local area network (LAN); there is a lack of staff to assist with LAN administration; South African physics students lack basic mathematical and programming skills that makes teaching undergraduate computational physics very challenging; many colleagues believe that Fortran is out of date and not worth using in the classroom.

2.1. Discussion

South African universities need to take a greater interest in developing computational physics at the undergraduate level. Computational physics combines the mathematical, theoretical, numerical and physical modelling of real systems in a unique manner that makes computational physics distinctly different from computer science. The wide range of problem solving skills that are developed with a training in computational physics make these graduates highly marketable and productive in endeavours that go well beyond physics, including in mainstream commerce and industry.

While Fortran has been traditionally used in the teaching of computational physics, we observe that Python is gaining in popularity in South Africa, with some departments using C, C++ and Java as their preferred programming languages. It should be noted, however, that many of the large production codes in physics totalling hundreds of thousands of lines of code are written in Fortran.

Time is needed to develop computational physics more completely at the undergraduate level. However, the undergraduate curriculum at many universities is generally very crowded with traditional topics in physics. It is very difficult to properly integrate computational physics into the current degree structure. There is much resistance to doing away with standard physics topics in lieu of introducing computational physics.

The most effective option is to introduce additional modules at the second and third year levels to teach computational physics fundamentally. This requires careful negotiation within the faculty, as this will impact on the number of credits that a student will be able to take in the co-major, and may impact also on the number of prerequisites and co-requisites. Not many South African university physics departments have been able to adopt this approach to date.

A lesser option is to integrate computational physics into physics laboratory programmes[8]. This requires that some experimental laboratory exercises be done away with, which requires careful negotiation amongst colleagues. Some physics departments have a dearth of good quality experimental equipment, and so introducing computational exercises as a substitute could be seen as a valuable improvement to the training of undergraduate students. With less time available for the development of computational physics, it is pertinent to ask what is realistically achievable?

The greatest flaw in the undergraduate teaching of computational physics at South African universities is to simply use ready-to-go packages in black box-mode (see Table 1). Here, students adopt a dropdown-drag-click approach and the entire exercise is often reduced to changing parameters and producing pictures. This may be instructive insofar as the application is concerned, for example in the understanding of a physical problem such as the changes of the magnetic field lines when a charge is accelerated in vacuum, but there is little gained in terms of developing real computational skills. This is the primary mode in which postgraduate training is taking place in South Africa in producing what we refer to as the computational technician, and this should be avoided as much as possible.

It is realistic within the South African university undergraduate system to introduce computational physics using higher level languages such as Easy Java Simulations or symbolic manipulation programmes such as Maple and Mathematica or interpreted languages such as Octave or Perl. This way, the students can solve complex computational problems applied to real physical systems without necessarily having to pay a lot of attention to detailed numerical matters such as determining the most stable algorithm for the solution of a set of coupled partial differential equations. We refer to this as algorithmic physics rather than computational physics (see Table 1). Maple, for example, has features that enable one to solve such numerical problems with ease. Still, the students have useful experiences in computation, especially in algorithmic development even if they are not called upon to develop large and complex programs. This mode of training is intermediate to taking a fundamental approach to computing which we have seen is not always possible given severe time constraints and teaching computational physics in black box-mode, which we believe should be avoided as much as possible.

3. Postgraduate training and research in computational physics

Eleven out of twelve universities train postgraduate students through computational physics research projects in various branches of physics. Very often students use large commercial or opensource production codes to conduct their research. At nine universities some postgraduate students write short pieces of code or some sections of their own codes or make some

modifications of production codes to carry out their research, by incrementally improving on mathematical models, algorithms, computational speed and data storage. However, there is little evidence that postgraduate students are routinely required to develop substantial sections of codes or to make substantial changes to existing production codes in their research. This is especially evident in the field of parallel computing. Many large production codes have been parallelised to run on massively parallel computers, and this aspect appears to be least understood or dissected in the training of postgraduate students in South Africa. Several physics departments have access to local computational clusters for high performance computing and many others make use of the supercomputers at the Centre for High Performance Computing (CHPC) to run large, complex, parallelisable calculations, for example in computational solid state physics, astrophysics, cosmology and particle physics. Postgraduate students working at two universities are taught the theory of parallel computing and are required to write message passing interface (MPI) codes to improve and manage their calculations on the cluster. CHPC runs specialised workshops annually for postgraduate students on the theory of parallel computing and how to parallelise and optimise codes using the CHPC platforms. But this endeavour is very far from developed in South Africa.

The African Institute for Mathematical Sciences (AIMS) has a strong focus on computational physics training. The students, who are drawn from various African countries, may be considered to be intermediate to the Honours and Masters levels. They have access to a state-of-the-art computational LAN working in an opensource environment and use the Python programming language. Depending on the lecturers who are selected each year, the students work on a variety of problems in applied mathematics, theoretical physics, computational biology and computational finance. The students are expected to develop their own computer codes during structured computational classes.

The Centre for Research in Computational and Applied Mathematics (CERECAM), at the University of Cape Town, provides a multidisciplinary research environment for postgraduates comprising theoretical mechanics, computational mechanics and high performance computing. Research programmes include projects of a fundamental nature such as numerical analysis and partial differential equations, and projects which are relevant to industry and other applications such as computational solid-, structural- and particulate-mechanics, computational electromagnetics, computational fluid dynamics and biomechanics. Computational work within research projects consists of the development and implementation of finite element (FE) approximations. Researchers use commercial software packages such as Abaqus FEA and ANSYS FLUENT CFD to implement FE solutions, and develop their own FE codes ab initio using Matlab, C++ or a C++ program library called deal.II.

The National Astronomy and Space Science Programme (NASSP) is hosted at the University of Cape Town and offers degree programmes in astrophysics and space science at the honours, masters and PhD levels. The program places strong emphasis on training in computational astronomy and astrophysics at the honours and masters levels. Students are taught to program in Fortran and Python. They also use commercial software such as Maple and Mathematica as well as open source astronomy and astrophysics software packages. Students typically write their own codes or extend existing in-house codes to perform parameter fitting, standard statistical and Bayesian statistical analysis of data and N-body simulations.

Astronomers are substantial users of computer resources, for reducing observational data, for robotically operating telescopes, for interfacing with hardware, and also for theoretical modelling and simulating astrophysical processes, especially in cosmology. Scientists at SALT have developed PySALT[9], which is written in Python, for reducing the SALT data. The software is opensource and therefore easily available for the entire international SALT community. The pathfinder KAT7 helped win South Africas bid to host the major share of the Square Kilometre Array (SKA) primarily because all aspects of the technical design, development and manufacture

of KAT7 were conducted very successfully and on schedule. In here, computation played a significant role in the design, and in the reduction and analysis of the astrophysical data. KAT7 science commissioners are routinely writing their own codes in Python, C and C++ to turn the raw radio data into meaningful quantitative astrophysical images of the sky. The SKA will require thousands of times more processing power than is currently available in the fastest supercomputers. The boundaries of data compression and on-site processing will be pushed by the SKA to limits that require a completely new approach to computing. This has spurred the interest by major hardware and software companies such as IBM; this has the potential to significantly improve South Africas capability in the field of computing.

The Nuclear Corporation of South Africa (NECSA) has developed its own software package to study the interaction between neutrons and matter within their research nuclear reactors. Their software package solves a partial differential equation, known as the neutron transport equation, using deterministic and stochastic techniques. This package is also being used by a research facility in the Netherlands with a similar nuclear reactor. Commercial codes are used to study other neutron transport problems involving criticality and shielding.

The medical physics research department of iThemba Labs studies the use of neutrons in the treatment of various diseases. Scientists there develop their own codes in Fortran to solve the neutron transport equation and they also use commercial software packages to calculate neutron dosages for treatment.

Mintek and the Advanced Mathematical Modelling and Simulation Group at the CSIR use commercial and opensource software packages for computational solid state research to predict the properties of new materials. Mintek also writes code using Java, C++ or C to study diffusion in alloys and to determine stable geometries of nanoparticles.

3.1. Discussion

Some of the large production codes, for example computational fluid dynamical codes or quantum mechanical electronic structure codes or astrophysical magneto-hydrodynamical codes total in excess of several hundred thousand lines of code. It is unrealistic and unproductive to expect any single research student to develop such production codes in its entirety starting from a clean slate. Today, it is more realistic to expect a postgraduate student to start with a working piece of code and to make some modifications to integrate new theoretical, mathematical or physical models and to test these models with new applications. Being able to accomplish this with ease is very central to the modern computational physics research endeavour. This requires that the student develops a level of competence in reading computer codes and making algorithmic and programmatic changes to these codes. These interventions are very much needed if we want our research students to move the boundaries of their disciplines significantly in new directions and to other disciplines. Using standard off-the-shelf codes with no recourse to making changes to the code ties the researcher to a restricted class of problems that would already have been conceived of by the original programme developers. There is little or no room for innovating in ones research discipline by only being able to execute code. For example, if a student develops a new model in the field of ecology, how does the student test this model in a comprehensive way if the student is unable to do a realistic simulation using the model? Writing the entire code ab initio is not necessarily a productive way to proceed. On the other hand, being able to modify an existing piece of similar code is a preferred way to progress if this is at all possible. Some commercial codes come with exorbitant licensing fees and, as a part of the licensing agreement, there is no access to the source code. This may be acceptable in an industrial setting, but is not helpful in an academic environment and this should be avoided whenever possible.

Developing paradigms for more complex scientific codes should be pursued more vigourously in our teaching of computational physics. This is relevant for (i) the pedagogy of computational physics at the postgraduate and research levels, and is (ii) an endeavour that is ripe for

research productivity in the field of computational physics education research, and finally, (iii) this is also a basis for developing new material for the teaching of computational physics at the undergraduate level. The idea is very simple: if research students today are not in a position to develop their production codes in its entirety from a clean slate and if they are primarily operating these production codes in black box-mode then, for pedagogical reasons, these students should be tasked with solving a series of self-contained computational exercises, each mimicking particular critical elements of the production code. There is a fair amount of creativity that is required to develop such computational exercises, and this in itself has the potential to become the subject of publishable research in computational physics educational journals. These exercises may also be utilised in teaching at the undergraduate level. This has the potential to stimulate a new and exciting research endeavour in computational physics.

4. Conclusions

The training in computational physics at the undergraduate level, and at the postgraduate and research levels at South African universities, and research facilities and institutions is generally weak and needs to be taken more seriously. South Africa is more in the mode of producing computational technicians rather than computational physicists. Failure to improve the culture of computational physics in South Africa will impact negatively on South Africas capacity to grow its endeavours generally in the field of computational sciences, with negative impacts on research, and in commerce and industry.

Acknowledgments

TS thanks the NRF and the department of Physics at the UP for funding. We thank the lecturers at all the physics departments and scientists at the research institutions who participated in the survey. TS thanks Norman Chonacky for assistance with the survey questions. NC acknowledges support from the National Institute for Theoretical Physics.

References

[1] Christian W O and Esquembre F (2007) Modeling physics with easy java simulations *The Physics Teacher* **45** 475
[2] Wieman C E, Perkins K K and Adams W K (2008) Oersted Medal Lecture 2007: Interactive simulations for teaching physics: what works, what doesn't and why Âm. J. Phys. **76** (4 and 5) 393
[3] Singh C (2008) Interactive learning tutorials on quantum mechanics *Am. J. Phys.* **76** (4 and 5) 400
[4] McKagan S B, Perkins K K, Dubson M, Mally C, Reid S, LeMaster R and Weimann C E (2008) Developing and researching PhET simulations for teaching quantum mechanics *Am. J. Phys.* **76** (4 and 5) 406
[5] Chonacky N and Winch D (2008) Integrating computation into the undergraduate curriculum *Am. J. Phys.* **76** (4 and 5) 327
[6] R Chabay and B Sherwood (2007) *Matter and Interactions* 2nd ed, Wiley, Hoboken, New Jersery. See also http://www.matterandinteractions.org
[7] Beichner R, Chabay R and Sherwood B (2010) Labs for the Matter and Interactions curriculum *Am. J. Phys.* **78** (5) 456
[8] Spencer R L (2005) Teaching computational physics as a laboratory sequence *Am. J. Phys.* **73** (2) 151
[9] Crawford SM, Still M, Schellart P, Balona L, Buckley D A H, Gulbis A A S, Kniazev A, Kotze M, Loaring N, Nordsieck KH, Pickering TE, Potter S, Romero Colmenero E, Vaisanen P, Wiliams T and Zietsman E (2010) PySALT: the SALT Science Pipeline, *SPIE Astronomical Instrumentation* 7737