GPUs in a COMPUTATIONAL Physics Course

Joan Adler, Gal Nissim and Ahmad Kiswani

Physics Department, Technion -IIT, Haifa, Israel, 32000

E-mail: phr76ja@tx.technion.ac.il

Abstract.

In an introductory Computational physics class of the type that many of us give, time constraints lead to hard choices on topics. Everyone likes to include their own research in such a class but an overview of many areas is paramount. Parallel programming algorithms using MPI is one important topic. Both the principle and the need to break the "fear barrier" of using a large machine with a queuing system via ssh must be sucessfully passed on. Due to the plateau in chip development and to power considerations future HPC hardware choices will include heavy use of GPUs. Thus the need to introduce these at the level of an introductory course has arisen. Just as for parallel coding, explanation of the benefits and simple examples to guide the hesitant first time user should be selected. Several student projects using GPUs that include how-to pages were proposed at the Technion. Two of the more successful ones were lattice Boltzmann and a Finite Element code, and we present these in detail.

1. Introduction

Projects are an important part of any Physics education but are especially important for Computational Physics education, [1, 2]. They help students approach realistic problems, learn to present and annotate code and provide material for use in general courses as well as the Computational Physics one. The course given at the Technion is based on several textbooks [3, 4] as well as own code and is described in [5]. There is a heavy emphasis on visualization, as well as gaining experience in a LINUX and parallel computer environment. Although there have been many changes in computer hardware over time, two of the largest have been the rise and (temporary?) demise of vector programming and the rise in parallel distributed memory architectures concurrent with the use of commodity hardware.

As the computational infrastructure continues to change with time, the most recent innovation in the supercomputer world is the increased adoption of GPUs (Graphical Processing Units) for computation. Thus these algorithms need a place even in an introductory course. Ideally, a senior undergraduate/early graduate course should provide a basis for many years as the student graduates and moves forward. This is especially true in Israel, where students may have a several year break between first and higher degrees due to national service. It is also important for students who take child rearing career breaks.

2. Computational physics course design

Our course has a few weeks of basic algorithms for numerical analysis, with examples taken from undergraduate physics, such as chaos and Schroedinger equation solution, followed by algorithms such as Metropolis simulation and multidimensional integration and Hoshen Kopelman for

percolation. Molecular dynamics/simulated annealing approches are discussed. Each week has sample codes with visualization, either in PGPLOT [6] (2D) or with the Technion AViz code [7] for 3D atomistic cases. It takes place in a purpose built classroom with LINUX boxes as well as communication infrastructure for laptops. Many of the fortran routines are adapted from either [3] or [4] with added PGPLOT graphics prepared as students projects from the 1990s.

Once some basic techniques are learnt and familiarity with a LINUX environment and ssh and the compile, link, run paradigm of remote use MPI and queue based submission on our local parallel computer are introduced. Each of the above mentioned examples is reinforced with compulsory individual homework, which the students are encouraged to submit on sourcecode html sites. There is sprited discussion on which language, are MATHEMATICA and MATLAB allowed, but compulsory exercises in c, fortran, and MATHEMATICA, as well as homework with free choice of code ensure the students are exposed to many possibilities. Except for the free choice questions, working code requiring adjustments is supplied to minimise startup pain.

2.1. Choices and tradeoffs

At some time, depending on the class composition and interests, the students vote on the final topics, that range from "big codes" such as QUANTUM ESPRESSO and LAMMPS to computational optics and lattice Boltzmann/finite element methods for fluid dynamics. These topics are presented with the aid of student projects. Unfortunately many of the older ones use X-graphics. This was the best option 20 years ago, and while still good to teach algorithms/physical phenomena are not so suitable for emulation in more modern systems where different options such as QT are more common. This vote is needed because since we only have 13 week semesters, emphasis on any specfic topic has to be at the expense of others. The final weeks need to give the most to the most students in each specific semester.

In order to update, combined with the hardware shift to GPUS, and the interest in python, some fresh codes were needed. Since GPUs are so suited to finite elements, and the current finite element/fluid had the most compilation issues the time had come to prepare new codes, and we describe these below.

During choice-of-code discussions, Technion students always bring up MATLAB, which is much loved locally, and has a site license. The class policy is that it is allowed for projects of students doing experimental research. (Likewise, science education students may use windows software.) Most of the rest of the course is based on public domain or academic-licensed code, to enable portability for the student's future, but for special cases minimal MATLAB/MATHEMATICA use occurs. In the 2016 reality, the changes to fortran compilers (specifically the discontinuation of some fortran 77 commands in gfortran), has meant that this is no longer completely reliable in the very long term. We recompiled our own f77 codes and and adapted our PGPLOT implementations, and an addedum to [5] is in preparation to cover this.

2.2. Fear barriers

Today, even some computer-aware students find it hard to move out of the comfort zone of personal windows, mac or ubuntu desktops and laptops. Since serious computational physics requires access and use of additional servers the first obstacle to overcome is that of compile/link/run to replace some local interface and to ssh to a remote machine. The next fear barrier of remote queue submission follows relatively easily, but introduction to MPI (or OpenMP) is more significant, but harder. Neither of these require a new language, but they do need some introducion to scripting commands which are fairly common to all langauges and for codes like MATHEMATICA/MATLAB. Parallel coding can be done easily for either many canned codes or for the case of trivial parallelization in a suite of simulations where each node

receives its own dataset/parameters and communication is minimal. Such situations are useful to build confidence.

Introduction of GPU codes is a new level of change of concept. Unless the GPU use is embedded in the code such as in the case of MATLAB, a GPU suited language has to be selected and rather more understanding of the algorithms required. Hence it is crucial to present examples that show how significant speed up can be.

3. Codes suited to GPUs

The GPU unit has much in common, from an algorithmic viewpoint with vector processors of yore. It can do simple operations very quickly. Thus one needs to choose problems where this is a large component of the total calculation load. However even for these, some effort is needed. It is not that different to teaching parallel coding in that simple well documeted examples give confidence and frameworks. However, there is not the easy way out of trivial parallelization to start students on the path to parallel coding that can be used to begin utilization of the GPUs.

Many excellent GPU codes exist for statistical physics models like the Ising model or molecular dynamics, but many require some sophistication to begin implementing. Within our course framework, these topics were already covered earlier with straightforward codes. Since the finite element codes were based on X-graphics, and this topic was covered at the end of the course only if voted for, the following example was prepared for this purpose.

3.1. Finite Element - water waves

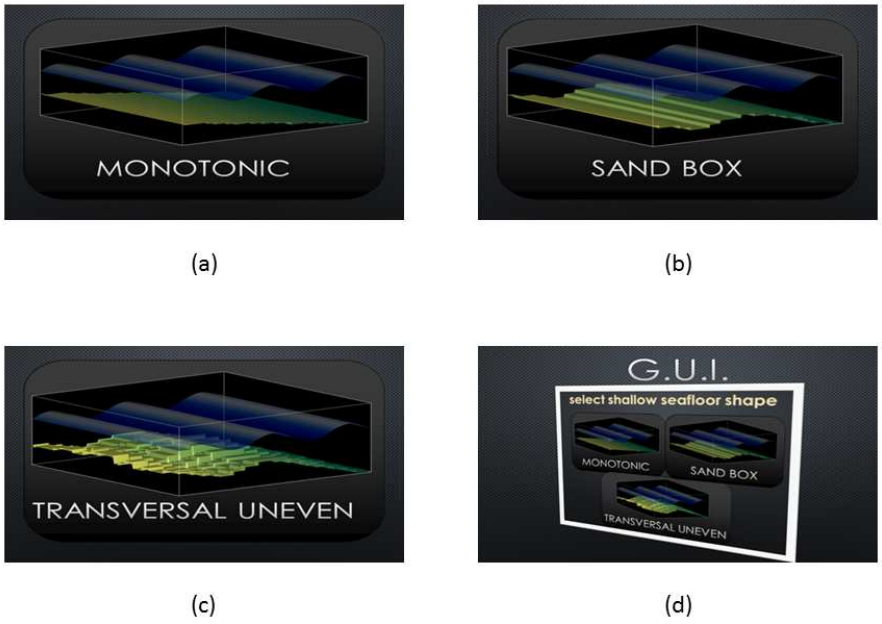

Figure 1. Substrate options (a) monotonic sloped bottom, (b) stepped (sandbox) option, and (c) uneven option. The GUI to select the substrate is shown in (d).

The Finite Element implementation here takes the easy route of using MATLAB; however it is most probable that some package would be needed for a Finite Element Method implementation so this is a good choice. Full code details and explanations are provided on the project website [8]. At the Technion, there is a site license for MATLAB use. The Finite Element model simulates water waves with a choice of sand substrates. The sand substrate determines the temporal

evolution of the surface waves, which are primarily excited by wind. There are possibilties to move between CPU only and CPU/GPU options. The substrate options are sloped (Figure 1(a)), stepped (Figure 1(b)) or uneven (Figure 1(c)). Theory, results, and downloadable files from the website run as downloaded with relatively recent versions of MATLAB for LINUX or windows, although the GPU options require 2015b and later. The instructions are very clear, with functions and GUIs for each function clearly described, including an explanation of how the GPU parameters can override the setparameter function. An image of the GUI to choose the bottom is given in Figure 1(d).

3.2. Lattice Boltzmann - fluid flow

Another very popular algorithm suited to GPUs is lattice Boltzman. It was previously taught with an earlier lattice Boltzmann Fortran90 code and a MATLAB graphics implementation [9] which provides an excellent introduction to the algorithm. An image from this code is shown in Figure 2 for fluid around a spherical object.

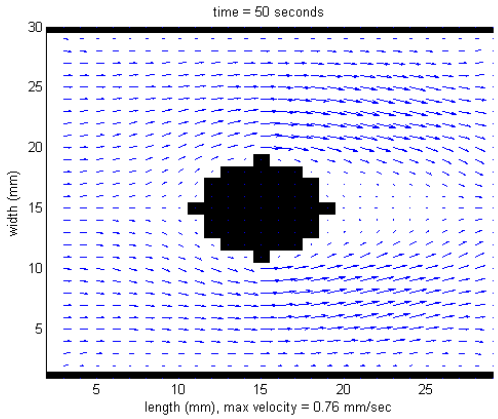

Figure 2. Fluid around a circular object.

However, in order to creaate a second GPU example a completely new implementation was prepared [10]. The website was prepared as a (downloadable) ipython notebook. with an explanation of the algorithm, source code, compilation instructions and instructions for compiler installation on a unix system. It is a beautiful and quite well documented project, entirely public domain, and we have run it under both CENTOS and UBUNTU, The speedup with the GPU is amazing. The code requires an Nvidia GPU with compute capability of at least 3.0; a version of CMake of at least 2.4 and OpenGL, GLUT and GLEW. The program uses OpenGL to draw a color map of the fluid velocity in the medium. Although all these work on our classroom server, this code does not have the wider system choice that the one described in the previous section does. This is more than balanced by the excellent speedup characteristics as presented in Table 1 for a Intel i73930k CPU and an Nvidia Geforce GTX 670 GPU. The reader should be warned that root privileges may be needed to install correct version of the OpenGL libraries.

4. Conclusions

Two websites that provide downloadable examples of GPU codes suited for an introductory Computational Physics class have been described, Both include a physical background, discussion of algorithm, downloadable codes with step-by-step instructions and sample results. The level is suited to senior undergraduate or beginning graduate students wihout prior experience in GPU coding.

Table 1. Speedup for the lattice Boltzman code.

Size	CPU[fps]	GPU[fps]	Speedup
300x300	125	1050	8.4
500x500	32	440	13.7
1000x1000	8	122	15.25
2000x2000	2	33	16.5

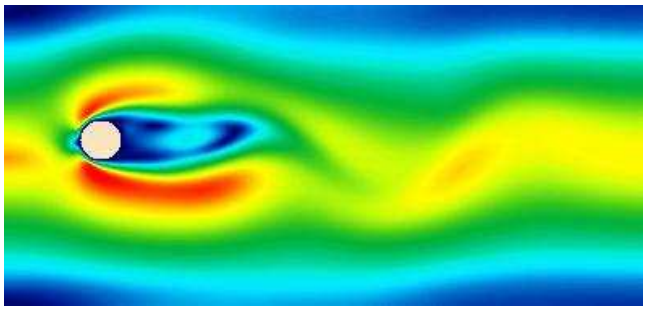

Figure 3. Flow around a single object.

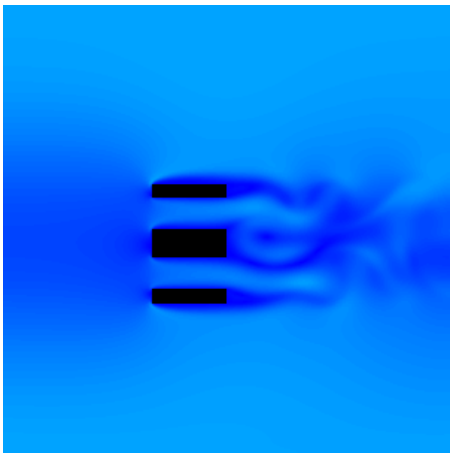

Figure 4. Flow around 3 objects.

Acknowledgments

A very big thank you to the Technion Physics Department staff, especially Israel ben Efraim, Avi Refaeli, Riva Kreizelman and Yan Katz for the renovations and installation of new equipment for the Computer classroom. Thanks to Adam Levi and Igal Rasin for help on selecting GPU hardware, and to research friends who lent me their GPU codes. The educational websites are part of the SimPhoNy project which is funded by FP7 under NMP-2013-1.4-1 call with grant agreement no: 604005.

[1] Mazvovsky D, Halioua G and Adler J 2012 *Physics Procedia* vol 34 p.1-5.
[2] Adler J, Artzi Y, ben Bashat L, Izraeli T Y, Kreif M, Lavi I, Leibenzon A, Levi A, Schlesinger I, Toledano E, Peretz U, Weisler Y and Alon Yagil 2014 *Journal of Physics: Conference Series* vol 510, p.012003.
[3] Koonin S C and Meredith D E 1998 "Computational Physics: Fortran version", 2nd edition, Westview Press.
[4] Gould H and Tobochnik J 1998 "An Introduction to Computer Simulation Methods: Application to Physical Systems, Parts I and II" Addison-Wesley.
[5] Adler J 2014 *Physics Procedia* vol 53, p. 2-6.
[6] http://www.astro.caltech.edu/∼tjp/pgplot
[7] http://phony1.technion.ac.il/∼aviz
[8] http://phelafel.technion.ac.il/∼gn/project
[9] http://phelafel.technion.ac.il/∼drorden/project
[10] http://phony1.technion.ac.il/∼ahmad