CASE STUDY

Financial computing literacy: 10 steps

Amogh Deshpande, Department of Mathematical Sciences, University of Liverpool, Liverpool, UK. Email: addeshpa@gmail.com

Abstract

It is often the case that in a financial engineering/mathematics master's curriculum, computer programming if taught, is before the start of an important course typically titled ‘Numerical analysis of financial derivatives’. Typically in this computer programming course, C++ is taught, and the material spans from basic constructs to an introduction to advanced programming such as design patterns. A major reason for its early introduction is that students would subsequently be able to use the skills to computationally solve problems occurring in numerics. We believe this curriculum strategy to be an element of ‘computer literacy’ which has been criticized in our context as one that discourages students with lower programming abilities who otherwise may have predisposed mathematical abilities. We aim to make fundamental financial computing inclusive via a series of 10 steps thereby leading to what we refer to as financial computing literacy.

Keywords: Computing literacy, financial computing, 10 Steps.

1. Introduction

MSc courses in Financial Engineering/Mathematics are popular amongst students for its job-market appeal. A crucial component of its curriculum is a course on numerics. It consists primarily of finite difference methods (FDM) for partial differential equations (PDE), implicit and explicit schemes, stability, consistency and convergence, and applications to option pricing via computer implementation. Typically, student backgrounds range from business, economics, engineering and mathematics. Enrolment is normally double digit. Unfortunately not all of these have even primary exposure to computer programming, like the students which we investigate for this case study. On the other hand, C++ being a favourite language of financial quants (or financial quantitative analysts), is taught in the programming class. The material therein spans from basic constructs to an introduction to advanced C++ programming like design patterns. The objective is that students would then be able to use the skills to computationally solve problems occurring in numerics. We believe this curriculum strategy to be a part of what Luehrmann (1981) calls “*computer literacy*". Luehrmann implies that the word “*literacy*" after the word “*computer*”, must also mean the ability to do computing, and not merely recognize, identify, or be aware of alleged facts about computing. Inherent in this definition, is a fact that there is something fundamental in computing that every student must be aware of. However, Harvey (1983) argues that the word “*literacy*" is unnecessarily broader in scope. Harvey in fact propositions that computers should be made available to students as a serious tool and more importantly in a pointed and embedded way, not as something they'll need later, which in our context happens if programming is taught preceding a numerics course. This may risk discouraging students who have weak programming skills but otherwise predisposed mathematical abilities. We believe that if standard computer programming is embedded within numerics, it can only enthuse students to tune their minds to learn and use advanced financial computing concepts later on. It also results in a deeper conceptual understanding of the involved numerics. Additionally, this embedded teaching pattern modulo few changes may also qualify for its early introduction into an undergraduate curriculum. In summary, the newness of this approach is its embeddedness within numerics and inclusivity to financial computing via simple steps through MATLAB. Expectedly, implementing this case study is to aid rather than substitute an important course on C++ programming.

To achieve this objective, we provide 10 steps that should lead to 'any slightly motivated' financial mathematics student interested in pursuing further advanced financial computing. These steps start from teacher or tutor-led lab demonstration (to teach students basics of MATLAB) and culminate with student self-driven projects. These steps must be followed in ascending order and may or may not contribute to final grade. With no prior assumed coding or any sort of programming background, the choice of the programming language becomes crucial. We choose MATLAB instead of C++, for the following reasons: availability, generality, engineering usefulness, stability, simplicity to use and learn, and hence importantly its inclusivity. This also has been cited as the reason in Öhström et al. (2005), and Higham (2004) and we concur with that.

We believe that there are three possible ways to introduce computing to students in a lab. These are:

1. Assign projects while you watch them do. Help them out if needed;
2. Have teaching assistants (TAs) who will join the instructor in helping in-need students in the lab session;
3. Provide stand and deliver demonstration and craft the lab tutorials in such a way that it makes them yearn to code.

The first method expects financial computing literacy while the second expects it from the TA's. Also, it is costly and restrictive. Hence we followed the third approach. In it lab instruction is carried out in a wireless internet enabled classroom with desktop and projector screen. We needed internet to access MATLAB from the university's software hub. We required wireless room because students were going to use their own laptops. Brown and Lippincott (2003) have appealed to the use of laptops in pedagogy, popularly termed as Bring Your Own Device (BYOD). This was not a constraint since all students usually own laptops or may get them on loan from the university library. Interestingly, my observation when I once took the lab in a wireless enabled class with desktop computers was that students found comfort in using their own laptops. We believe the primary reason being that this way they could sit with their colleagues and discuss the exercises. However we wish to point out that we envisage that the pedagogy developed here is as effective if implemented in a room with desktop computers, a computer with projector and a screen. In fact, we recommend mix use of desktop first and laptops later, since the first nine steps expect independent work, followed by the last one which encourages group activity in a classroom. We note here that laboratory teaching activity is also not just about designing the learning task but also the physical arrangement of the space and the roles of the teachers/students within it (Pretto, 2011).

The idea was to have a stand and deliver approach to lab teaching which had to be carried out. However, Ramsden (2003) argues that this instruction style encourages surface learning as the approach in his opinion is narrow and minimalist. Echoing Ramsden, Davies (2008) argues that lab instructors must ponder as to how to inculcate creativity and independence which could be fostered via designing lab questions that makes students understand the uncertainties and inaccuracies of computing experiments from the outset; which we believe is what our 10 steps achieve. In that spirit, the 10 steps that we designed are based on the pedagogy model of Hazel and Baillie (1998), and are obviously arranged based on increasing autonomy of these ten tasks. These activities are based on the classic book of Wilmott et al. (1995). The main reason behind the choice of this book is that, it has pseudo codes on financial computing using PDEs that can be made easily compatible with MATLAB or C++, and are also close in sprit to the classic Numerical recipes book, Press et al. (2007) that we will use at the end.

This laboratory activity resonates with the objective of the Quality Assurance Agency for Higher Education (QAA) and the UK Standard for Professional Engineering Competence (Engineering

Council, 2014) who state that students should have succinct exposure to hands-on laboratory work followed by project work to achieve satisfactory understanding (Davies, 2008). Given that such an objective needs to be covered in a short intensive period of a semester, on an average of three months that also includes teaching the mathematical theory involved, it is imperative to realize it as efficiently as we can. This can be achieved, we believe, by focussing only on the very basic programming tenets needed for implementing numerical pseudo codes as what these 10 steps do. Table 1 describes numbered exercises in conjunction with expected level of autonomy desired.

Table 1: Levels of autonomy for types of laboratory activity corresponding to exercises

Autonomy	Laboratory activity	Aims	Material	Method	Answer	Exercise
0	Demonstration	G	G	G	G	1, 2, 3
1	Test	G	G	G	O	4, 5
2	Structured Enquiry	G	G in w/p	O in w/p	O	6, 7, 8
3	Open Ended Enquiry	G	O	O	O	9
4	Project	O	O	O	O	10

Here G, O, G. in w/p, and O. in w/p stands for Given, Open, Given in whole or part, and Open in whole or part respectively. In the next section we describe a set of MATLAB based exercises. This set of 10 exercises forms our ten steps. In the table above we map the type of laboratory activity to these 'exercises' as the last column depicts.

2. The 10 steps

Exercise 1. Type a vector u with elements u = (1; 2; 3). Check if you can extract the first element i.e. 1, by typing u(0) in the command prompt. Comment. Now, extract its third element.

Exercise 2. Write a function that prompts the user to input two real valued numbers. It returns a sum and a product of these numbers.

Exercise 3. Now write a second function that has a real valued input argument. Call the first i.e. the earlier function in Exercise 2, in this second function. The product value returned by the first function is again multiplied by the new real value supplied to this second new function. Now return the new value of the computed product.

Exercise 4. Write a separate function to compute the Black-Scholes-Merton price. Use the standard command for the same. You can use MATLAB help or just Google to search the inbuilt command. Compare your answer with the one generated by this inbuilt MATLAB function.

Exercise 5. Note the grid drawn in Figure 1.

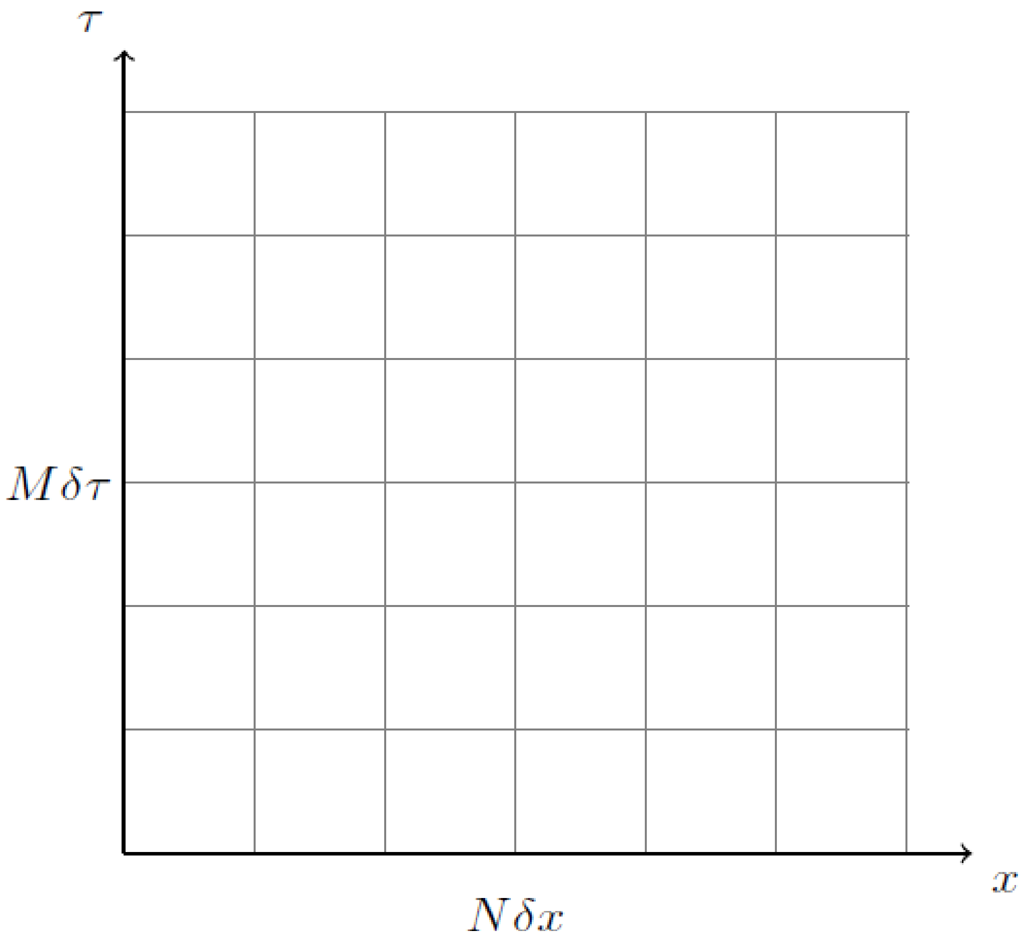

Figure 1: Discretized x – τ grid

In the already available MATLAB code (see Figure 2), replace FILL by appropriate statements that codes the grid defined as below. Run the code.

u=0, at x=0 and x=1 ∀ t belongs to [0,∞) ; u=1-4$(x-\frac{1}{2})^2$ for t=0 ∀ x ∈ [0,1].

```
function FILL = gridxtau(x,tau,M,N)
deltax=x/N;
deltatau=tau/FILL;
k=0;
% populate u at time 0 and all x i.e. u(.,1)
for n=0:1:FILL,
    k=k+1;
    u(k,FILL)=1-4*(n*FILL-1/2)^2;

end
% populate u at boundary points
for t=0:1:M,
    u(FILL,FILL)=0;
    u(FILL,t+1)=0;
end
```

Figure 2: Editable code provided to students

<u>Exercise 6.</u> In the below MATLAB code of this exercise, for Nminus<0 and Nplus>0 integers, substitute the word FILL so that the code, if typed on the command prompt in MATLAB, will run.

```
FILL
for n=Nminus:1:Nplus,
FILL
    oldu(k)=n;

end
```

Figure 3: Editable code to run in MATLAB

Hence, if you want to further code the following for loop pseudo code following the above MATLAB one, how will you do so?

```
for(n=Nminus+1; n<Nplus;++n)
b[n]=oldu[n]
```

Figure 4: Additional for loop pseudo code

<u>Exercise 7.</u> Use pseudo code Figure 8.5 (Wilmott et al., 1995) for the explicit FDM and compute output u at terminal time for Exercise 5 when the stability factor is in the range of [0,0.5] and otherwise. Comment.

<u>Exercise 8.</u> Use pseudo code Figure 8.11 (Wilmott et al., 1995) for the fully implicit FDM using SOR and compute output u at terminal time for Exercise 5 when the stability factor is in the range of [0,0.5] and otherwise. Comment.

Exercise 9. Use the codes developed in Exercise 7 or 8 to compute a Black Scholes Merton price for European Put.

Exercise 10. Use any pseudo codes in the book on Numerical recipes in C++ (Press et al., 2007) to compute related mathematical idea in MATLAB.

3. Reasoning

We discuss here the reasons behind exercises enumerated in the 10 steps above.

Exercise 1. This tests whether one can type an array of numbers in MATLAB. Also it tests whether the user understands that the first element of any array is accessed with counter 1 and not zero. That is, one should type u(1) in command prompt to get the answer which here is 1 and not u(0). This also helps the same user deal with a similar issue in C++ where the counter, unlike MATLAB, does start with 0. Since further FDM pseudo codes use vector operations instead of matrix, this exercise on vector serves well for later.

Exercise 2. This exercise prompts the user to first write a simple mathematical program in a function format akin to what "*Hello World"* is.

Exercise 3. Next, we ask the user to call a function inside a main function. This is what we will do in the FDM codes, where we write ancillary MATLAB functions that are called into the main function.

Remark A: Note that as per Table 1, Exercise 1, 2 and 3, being fundamental, are demonstration based. Though this pedagogy is primarily being 'stand and deliver', we expect pro-active participation from the students.

Exercise 4. This exercise tests whether students understood the demonstration held for Exercise 2.

Exercise 5. This exercise makes the student understand how to plot the grid (see Figure 1).

Remark B: Exercise 4 and 5 are test based laboratory type, as noted in table 1, since they are 'just' applications of what is covered in the theory class viz. Black Scholes pricing equation and demos of Exercises 1, 2 and 3.

Exercise 6. This exercise summarizes Exercise 1 on vector indices. The code is also useful while answering Exercises 7, 8 and 9. In this exercise they will learn to think as they write loops with arrays.

Remark C: One can heavily utilize MATLABs vectorization facilities instead of loop based vectors. Hence this may lead to eschewing 'for' loops. As argued by Higham (2004) who uses them in actual codes that he provides, the resultant codes are snappier, shorter and less daunting. We rightly believe this to be so. However we still introduce loops as these are often seen in the pseudo codes in Wilmott et al., (1995) and in the classic numerical recipes book of Press et al. (2007). Hence we introduce this exercise on 'for' loops to let the students become comfortable with loops.

Exercise 7. This is our first main FDM code. Students use the pseudo code given in Figure 8.5 of Wilmott et al. (1995). It defines a grid as in Exercise 5 and imports it in the main file that forms Figure 8.5 of the said book.

Exercise 8. This is our second main FDM code. The objective is similar to Exercise 7. They are expected to benefit with better understanding of the SOR method used in the fully implicit FDM.

Remark D: Exercise 6, 7, 8 all come under the pedagogy of what is known as laboratory based 'structured approach' in the sense that, as seen by the wordings of these questions, students are presented with a problem and are suggested some book/reference(s). The output is produced by an individual primarily in the classroom. Group work is encouraged only outside class. This is primarily to foster a deep approach to laboratory learning by coding in MATLAB solely via personal initiative.

Exercise 9. This is our third main FDM code. Student's use the codes developed in Exercise 8 to compute a Black Scholes Merton price for European Put.

Remark E: Exercise 9, expectedly, is a tricky one. Students are aware that analytic options pricing formulas are available in the book of Wilmott et al. (1995). They also have relevant FDM codes for both explicit (see Exercise 7) and fully implicit scheme (see Exercise 8). Using either of these to finally compute the Black-Scholes price is not that easy as they need to carefully synchronise codes in Exercise 7 or 8 with several analytic expressions related to options price. This ensures more decisions and experimental design considerations with students. Hence it is "*open ended enquiry*" as per Hazel and Baillie (1998).

Exercise 10. We next make the student do their first major independent project which uses any pseudo codes from the book in Numerical recipes.

Remark F: Exercise 10, is by nature an independent project. Group activity is encouraged for team building spirit. Students are expected to become comfortable with the Numerical Recipes book that is usually used in the quantitative industry.

4. Findings and conclusion

Being a small class, we found by observation that students were undoubtedly much more connected with numerics since they could visualize the numerical analysis concepts like stability and convergence using a computer. We believe that these ten steps could also be used to teach financial computing even to a typically large cohort of 2nd or 3rd year mathematics students who may only be exposed to smattering understanding of numerical analysis of PDEs. Effectiveness of these steps correlated strongly with the motivation level of the students. With no contributing credit, lack of motivation can be remedied by providing contributing credit to at least the 8th / 9th and 10th exercise. We aim to utilize this study to a large cohort of university undergraduates in financial mathematics and understand and report their response.

5. References

Brown, M. and Lippincott, J., 2003. Learning Spaces: More than meets the eye, Educause Quarterly.

Davies, C., 2008. Learning and teaching in laboratories: Engineering Subject Centre guide, *Higher Education Academy*. Available at: https://www.heacademy.ac.uk/system/files/learning-teaching-labs.pdf [Accessed 1 September 2017]

Engineering Council, 2014. UK Standard for Professional Engineering Competence. Available at: http://www.engc.org.uk/ukspec.aspx [Accessed 1 September 2017].

Harvey, B., 1983. Stop Saying "Computer Literacy"! *Classroom Computer News*, 3(6), pp. 56-57.

Hazel, E. and. Baillie, C., 1998. Gold Guide 4, Improving teaching and learning in laboratories, HERDSA publications.

Higham, D.J., 2004*. An introduction to financial option valuation, Mathematics, Stochastics and computation*: Cambridge University Press.

Luehrmann, A., 1981. Computer Literacy, What Should It Be? *The Mathematics Teacher*, 74(9), pp. 682-686.

Ohrstrom, L., Svensson, G., Larsson, S., Christie, M. and Niklasson, C., 2005. The pedagogical implications of using MATLAB in integrated chemistry and mathematics courses. *International Journal of Engineering Education*, 21(4), pp. 683-691.

Press, W.H.; Teukolsky, S. A.; Vetterling, W.T. and Flannery, B. P., 2007. *Numerical Recipes. The Art of Scientific Computing*, 3rd Edition: Cambridge University Press.

Pretto, G., 2011. Pedagogy and Learning Spaces in IT. *Proceedings Ascilite 2011 Hobart: Full Paper.*

Ramsden, P., 2003. *Learning to teach in higher education.* Routledge Falmer, 2nd Edition. London.

Wilmott, P., Howison, S. and Dewynne, J., 1995. The mathematics of financial derivatives: a student introduction, Cambridge University Press.