# Required Computer Science Education in BME Undergraduate Programs

**Prof. Robert A Linsenmeier, Northwestern University**

Robert Linsenmeier is a Professor Emeritus of Biomedical Engineering, Neurobiology, and Ophthalmology at Northwestern University. His interests are in the microenvironment of the mammalian retina and engineering education. His teaching is primarily in physiology for both biology and BME majors. He is a fellow if the Biomedical Engineering Society, American Institute of Medical and Biological Engineering, and Association for Research in Vision and Ophthalmology. He is the co-leader of CIRTL at Northwestern and Director of the Northwestern Center for Engineering Education Research.

©American Society for Engineering Education, 2020

# Required Computer Science Education in BME Undergraduate Programs


## Abstract

Data collected for the Biomedical Engineering Education Summit Meeting in 2019 showed that computer programming was required of undergraduates in biomedical engineering and bioengineering at more than 98% of the 57 accredited BME programs that responded to a survey. This is an increase over an earlier dataset from 2004, reflecting the increased need for engineers to be competent in programming. However, education in computer programming can take many forms, and there has been no attempt previously to understand whether there is any agreement across BME about what type of computing is important. While many universities offer elective courses in computer programming that can expand students' knowledge, the present project was done to determine the frequency of *requiring* different types of programming courses, and whether generalizations can be made about the state of undergraduate BME education in this respect. Required computer courses for BME students can be assigned to several categories: 1) MATLAB, 2) object-oriented programming (e.g. Python, C++), 3) programming of microcontrollers for on board sensing or control of electromechanical devices, 4) computer-aided design (CAD), sometimes used in conjunction with programs for additive manufacturing, 5) LabVIEW, and 6) numerical methods and simulations. University websites generally had adequate information about course requirements and sufficient course descriptions to categorize the computing courses, but additional internet resources were also used. The paper reports on computing requirements at all 118 currently accredited biomedical engineering and bioengineering programs. The most prevalent computing skill is MATLAB, which is required by at least 70% of BME programs. About half of that number require a course in an object-oriented programming language. CAD courses are required by 41% of BME programs, and modeling and simulation by 42%. Other categories of computer knowledge are less prevalent. Limitations of these data for understanding computing in BME programs are discussed.


## Introduction

Evidence from a comprehensive analysis of biomedical engineering and bioengineering (hereafter simply called BME) curricula showed that in 2004, 78% of accredited BME programs and 90% of unaccredited programs required a computer programming course for undergraduates. [1] A limited number of undergraduate programs were reassessed in 2014 and there was little change from 2004.[1] Data collected for the 2019 Biomedical Engineering Education Summit showed that all but one of the 57 programs responding to a survey required a computer programming course.[2] This is consistent with the ranking of computer programming by academic respondents to a survey about important skills and concepts prior to that summit meeting. Programming was ranked second, behind only statistics.[3] In responding to the same survey, 150 industry representatives (76% representing medical technology companies; 13% representing biotechnology companies and the rest diverse) listed "experience with relevant

software" eighth, although programming itself was not in the top nine (which were all that were listed).[3]

Education in computer programming can take many forms, and there has been no attempt previously to understand whether there is any agreement across BME about what type of computing is important or what is taught. While many universities offer elective courses in computer programming that can expand students' knowledge, the present project was done to determine the frequency of *requiring* different types of programming, and whether generalizations can be made about the state of undergraduate BME education in this respect. A number of specific questions were addressed here, some more completely than others, as explained in the results.

- To what extent is there a common set of computer skills that biomedical engineering undergraduates are expected to have?
- What is the distribution of the different types of computer programming courses across universities, in terms of the languages learned, computer skills for laboratory and design settings, and requirements for modeling and simulation?
- If there is a requirement for more than one computer language or course, what are the common individual courses and combinations of courses required?
- How frequently is the initial computer programming course dictated by a uniform engineering school requirement, and how often is it specific to the BME department?
- How many credit hours are devoted to computer programming courses, and when are courses taken during a student's program?
- How many programs have BME courses that focus on modeling and simulation of biomedical systems beyond programming, and how many credit hours do these represent?
- If a university wishes to cover many types of computing in its curriculum, are there good examples of how this might be done?

The terminology for categorizing computer languages is somewhat confusing. The term "high-level language" is often used for a programming language such as C, FORTRAN, or Python, in which programs are relatively independent of a particular type of computer. These languages "are considered high-level because they are closer to human languages and further from machine languages."[4] Some universities also call MATLAB a high-level language. Here we will refer to C and similar languages as "object oriented" languages. "***O**bject-**o**riented **p**rogramming* (**OOP**) refers to a type of computer programming (software design) in which programmers define the data type of a data structure, and also the types of operations (functions) that can be applied to the data structure. In this way, the data structure becomes an object that includes both data and functions. In addition, programmers can create relationships between one object and another. For example, objects can inherit characteristics from other objects."[5] These languages are compiled before execution in order to translate their operations into computer instructions. MATLAB and similar languages (e.g. Maple, Mathematica), are somewhat distinct, and are called "scripting languages," which are interpreted at the time of execution rather than compiled. MATLAB is designed for math and engineering, and has precompiled routines for many different functions, including matrix calculations and graphing,

so it is possible to use it more readily for complex analysis. However, students may be able to use MATLAB without understanding algorithmic thinking as deeply, which is at least one rationale for teaching object-oriented programming as well, or instead. Individuals trying to understand basic differences among languages may find an article at Stackify.com helpful.[6] There is a discussion on the internet about the distinction (and growing similarity) between compiled languages and scripting languages. According to the MATLAB website, MATLAB itself has elements that are written in C++, Java, and even Fortran. For our purposes we will distinguish between MATLAB and object-oriented languages.

**Methods**

Computer programming and skills courses at all 118 currently accredited BME programs were evaluated. The three accredited bioengineering programs at the University of California San Diego were analyzed separately, and the joint undergraduate program at the University of North Carolina Chapel Hill and North Carolina State University at Raleigh was analyzed as two separate programs. This project was restricted to counting only those courses that were required for all BME undergraduates at a university. In many (probably most) cases, additional programming, data analysis, image processing, or modeling courses are available as electives, and/or required in specific tracks or concentrations, but analysis of these elective courses was beyond the scope of the current project.

Curricula were studied primarily by using university websites. University undergraduate catalogs were the best source for finding requirements for BME programs and for course descriptions. The recommended placement of courses in the curriculum was often available from undergraduate catalogs, but department websites were also used for this purpose. In order to determine whether a course was required across the engineering school or was specific to a department, it was sometimes possible to identify common first year engineering curricula, but often necessary to look at the curriculum for each of a university's engineering programs.

The main focus was on courses in which learning a computer language was a principal topic. However, courses in which programming was used for implementing numerical methods, modeling, and simulations were also of interest. This required looking at course descriptions for many of the required BME courses. Such courses could have many different names, but in general, if computing, numerical methods, simulations, or modeling was not in the name of the course, it was it was only rarely that computer programming or an introduction to a particular application of computers was mentioned in the description.

For both of these categories, i.e. both computer programming courses and modeling courses, it was sometimes difficult to tell what language or computer package was taught. In these cases, various additional internet resources were used, including searching for course syllabi, finding student comments on reddit.com, and assessing posts on Coursehero (with materials generally uploaded by students) and similar sites. Unfortunately, the information available on sites like Coursehero without a subscription is limited, and even with a "Premier" subscription, an individual is limited to 30 course documents, so the author elected not to subscribe. Thus, there were some courses for which only limited information could be found.

## Results

### Fundamental Programming Languages

The required programming courses fall into several categories as shown in Figure 1. It was not possible to identify any computer programming course or a reference to any computer program in the curriculum at just two universities, and for another it was not possible to tell whether MATLAB or Object-oriented programming (OOP) was the required topic. At seven universities, there is no consistent requirement even within the BME department, sometimes with just two choices and others having a larger menu, with some students taking only MATLAB and others taking only OOP. However, as shown in Fig. 1, most programs have specific course requirements in OOP, MATLAB or both. Sixteen percent of the universities appeared to require only OOP, 50% appeared to require only MATLAB (actually one was Mathcad), and 20% required both OOP and MATLAB, often in the same course. This means that at present, at least 70% of BME programs require MATLAB and 36% require OOP. If anything, the percentage for MATLAB is likely to be an underestimate, because MATLAB may be used only in settings where it is embedded in another course but not listed in a course description or other resources that the author had access to. Still, it is clearly the most prevalent. It is likely that the percentage requiring OOP is more accurate, because these are usually taught in standalone courses that are easily identifiable. The most prevalent OOP language, when one was identified, was the C family (C and C++) (25 programs), with ten universities teaching Python, four teaching Java, one listed as "C or Java," and one listed as "Python and C++. No other object-oriented languages were mentioned.

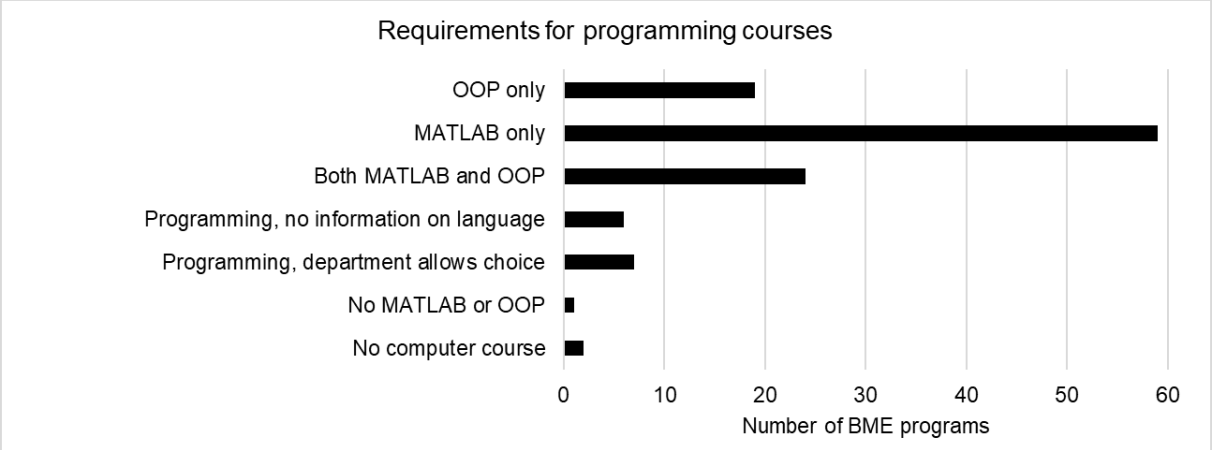


Figure 1: Number of programs requiring the computer programming skills indicated. Each of the 118 accredited programs is accounted for just once in this figure.

The distribution of credit hours for the programs represented by the top three bars of Figure 1 is shown in Figure 2. Each university is represented by only one number, which sums the credit hours required for MATLAB and OOP (but not the hours devoted to other types of computer use described in later sections of this paper). The distribution is broad, with the smaller number of hours often representing a standalone introduction to MATLAB, and the larger numbers representing cases where separate MATLAB and OOP courses are required. The caveat here is that for many of the three or four credit hour introductory courses involving MATLAB, the course is called "Introduction to Engineering," "Engineering Problem Solving" or something similar, so really the whole three or four credit hours cannot be attributed to learning computer programming. At the author's own university, the required MATLAB programming is taught in a first year course that also covers linear algebra, and learning MATLAB is probably responsible for the equivalent of one or two credit hours. It is not possible across universities to break out the part of the course due to MATLAB, however, so these introductory courses had to be counted in their entirety in Figure 2.

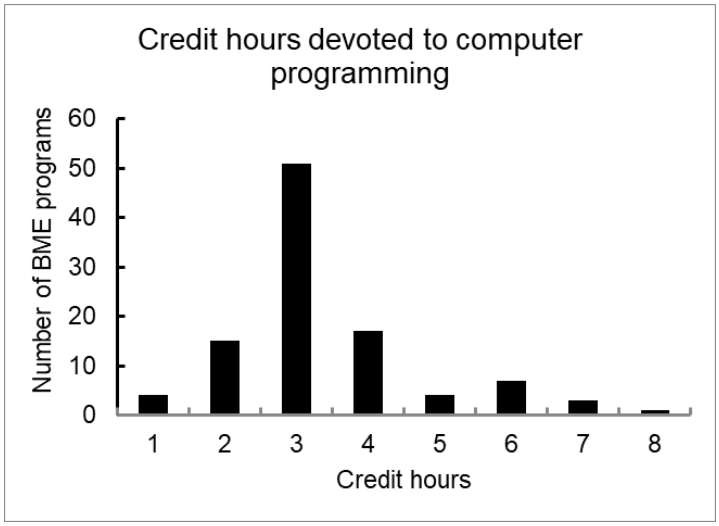


Figure 2. Required credit hours for computer programming courses

The position in the curriculum for these courses (again the top three bars in Fig. 1) was almost always the first or second year. Fifty-nine percent of the computer programming courses were in the first year, 33% were in the second year, and only 8% were later.

It was also possible to determine whether these courses were a uniform requirement across the engineering school, or were chosen by the BME department itself. Engineering school requirements accounted for 38.5% of the programming courses, with the balance being specific to the BME department. The engineering school requirements were often the result of a common first year curriculum for all engineers. At twelve universities, the common engineering requirement applied to all majors except electrical engineering and/or computer engineering (or very rarely another major). Electrical and computer engineering in these cases required OOP even if the rest of the engineering school did not, or required a different OOP language. Such cases were still considered to be university-required rather than department-required courses.

Finally, it should be noted that some universities have specific instruction in Microsoft Excel in the early courses, and mention that this is used for data analysis, graphing and curve fitting. Occasionally, there was a reference to the Visual Basic programming underlying Excel that can be done to automate Excel routines. Excel was not mentioned often enough that it seemed possible to give a reliable picture of its use, and the author expects that many university instructors assume that incoming students simply know how to use Excel. The author's personal experience, however, is that there is great diversity in students' knowledge of even the basic mathematical operations that are possible with spreadsheets.

**Other Computer Software**

Computer aided design, microcontrollers, and LabVIEW were the other topics that were referred to most commonly in required courses. There were also a few mentions of ImageJ and finite element software, but these were far less prevalent than the ones below.

**Computer aided design (CAD).** Courses that involve learning CAD software are required by 48 BME programs (41%), and at 16 of these, it is an engineering school requirement. All but six of these courses are in the first or second year, and many are standalone one or two credit hour computer graphics and/or rapid prototyping courses. (CAD was identified far more often than rapid prototyping and additive and subtractive manufacturing in required BME courses.) CAD is taught at 18 universities in three credit hour courses, also mainly in the first or second year, and in these cases it is usually in a course that includes aspects of design or other computer tools. Of the nine universities that mention a particular CAD program, eight use Solidworks, and one uses Autocad.

**Microcontrollers.** Programming of microcontrollers (e.g. Arduinos) for sensing, control, or actuation of devices is an identifiable requirement at only nine universities. At five of these, they are required in first year courses. The term "embedded computing" was very rare in BME required courses, and the term mechatronics, which mechanical engineers often use to refer to this topic, never appeared.

**LabVIEW.** Twelve universities mention LabVIEW as a programming tool in their required courses. It is always a departmental requirement. In seven of these, it is covered in first or second year courses.

**Courses in Numerical Methods, Modeling and Simulation**

The rationale for including a section on modeling and simulation is that modeling is an important component of BME, and this is a relatively advanced use of computing that many BME students experience. It is mainly an application of knowledge gained earlier about programming, although it may require new computer skills taught in the modeling course that build on basic programming. The attempt here was to include courses that taught numerical methods approaches, or created computer models from physical situations, or created simulations of biological processes. Such courses typically had names that included modeling, simulation or numerical methods and were typically junior or senior level courses. There was an attempt to *exclude* most signals and systems courses, which are highly prevalent in BME, or other courses

in which MATLAB might be used to do routine homework, but this is a somewhat fuzzy line. To illustrate, two descriptions of courses that were included are below:

"…development of computational models for understanding physiological systems; explores a variety of practical computational methods for biomedical problems. Includes an introduction to scientific computing, linear and non-linear models, and finite difference methods for ODEs and PDEs."

"…introduces computational techniques for solving BME problems and constructing models of biologic processes. Numerical techniques include … Applications include bioreactors, transport, pharmacokinetics, and biomechanics."

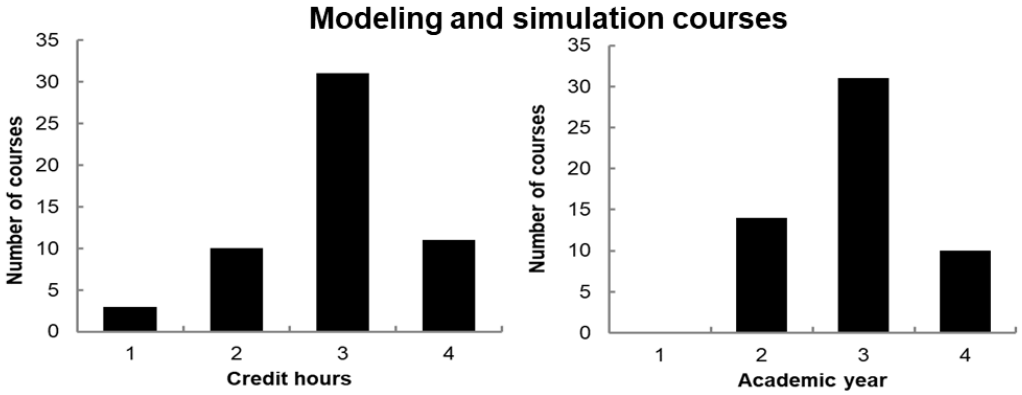


Figure 3. Credit hours and curricular placement of required modeling and simulation courses.

Fifty programs (42%) met the criterion for having at least one course included in this category. All were departmentally mandated, rather than engineering school requirements, and all but three courses in this category were taught by the BME department. As shown in Figure 3, these were generally taught in the third year or later, and were most often three credit hour courses.

**Who Teaches These Courses?**

First year courses, not only programming courses, but also courses that cover other computer topics discussed above, are required at 85 of the 118 accredited programs. Most frequently, the first year course or courses have general engineering course numbers (58%), so it is difficult to tell whether whether any BME faculty are involved in teaching them or not. The next largest group of universities have required first year courses within BME (28%). The BME courses usually cover either MATLAB or the tools of CAD or LabVIEW. Another sixteen percent of programs require courses in the computer science department, and a few (3.5%) require courses in mechanical engineering. (These values add to more than 100% because some universities require both a course in BME and a course in general engineering or computer science.)

The teaching distribution shifts substantially for required courses taken in the second year, with BME responsible for 66%, and the computer science department responsible for most of the

rest (26%). For junior and senior level courses, BME is responsible for almost all (92%), which by this point are largely the simulation and modeling courses.

It is uncommon for the BME department to be responsible for OOP courses, but this is the case at seven universities (Lawrence Technological University, Marquette University, and the Universities of Colorado at Denver, Maryland College Park, Memphis, Miami, and Texas at Austin).

**Examples**

As already seen, there is a great deal of variation across programs. Another way to look at this is to note how many courses, rather than credit hours, are required. For all programs in the dataset, 2 require no computing course, 34 require just one course, 42 require two courses, 35 require three, and six require four.

It is of interest to highlight a selection of universities that cover several of the types of computing in their requirements as examples of ways in which this might be done (Table 1). This group of universities all cover at least three of the types of computing discussed earlier. Better coverage of computing may mean that other topics get less coverage in the curriculum, but these universities seem to use the time for computing efficiently, with many of the courses covering not only computing, but other topics as well, as is often indicated by the course name. On average, the universities in Table 1 devote 8.3 credit hours to these topics. Generally the courses span the undergraduate years and start in the first year. Louisiana Tech University puts all the requirements in the freshman year in a set of courses whose syllabi are given in detail on their website, but two of these schools (Arizona State University and Marquette University) do not require any computing in the first year. It is also notable that these universities use credit hours flexibly, with some of the required courses using only one or two credit hours to cover topics that do not require full three or four hour courses.

**Discussion**

Different aspects of programming may each be most relevant for different types of careers, but BME programs generally do not know what careers their students will have, and with a crowded curriculum, choices have to be made about what type of computing course(s) to require. In some cases this decision is made for all programs by the engineering school, but more than 60% of universities give BME the responsibility of deciding on fundamental programming courses, and additional applications courses are at the discretion of the department.

The data may be useful in several ways. They show a diversity of approaches, with MATLAB, CAD, and modeling being the most prevalent courses. In general, they allow a program to judge whether it is in the mainstream of BME curricula. For instance, a department with a small requirement may wish to reflect on whether this is sufficient, and to be more cognizant of skills that other BME students may have. At the other end of the spectrum, a university with a substantial requirement relative to the average may be able to argue that its graduates are unusually well prepared for certain types of jobs.

**Table 1.** Selected universities that cover multiple types of computing in several required courses. The column D/U indicates whether the BME department (D) or the engineering school (U) mandates this course. Yr indicates when the course is usually taken, and Hr indicates the number of credit hours. The topics are M for MATLAB; OO for Object-oriented programming in Java (J), Python (P), C or C++; MC for microcontrollers, NM for numerical methods and CD for computer-aided design. An asterisk indicates that Excel is also covered. One of these universities also requires LabVIEW (U Cincinnati).

| | D/U | Yr | Hr | Topic | | | | | Course | Title |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | M | OO | MC | NM | CD | | |
| **Arizona State U** | D | 2 | 3 | | J | | | | CSE 110 | Intro to programming |
| | D | 2 | 1 | X | | | | | MAE215 | Intro programming in MATLAB |
| | D | 3 | 2 | | | | X | | BME301 | Numerical methods |
| | D | 3-4 | 3 | | | X | | | BME 370 | Microcomputer applications in BME |
| **George Mason U** | D | 1 | 3 | X | | | | | BME 101 | Intro to bioengineering |
| | U | 1 | 4 | | P | | | | CS 112 | Intro to computer programming |
| | D | 3 | 4 | | | | X | | BME 330 | computational methods in BME |
| **Indiana/Purdue U** | U | 1 | 2 | | C | | | | ENGR 19700 | Intro computer programming |
| **Indianapolis** | D | 1 | 1 | X | | | | | ENGR 29700 | Computer tools for engineering |
| | D | 3 | 3 | | | | X | | BME 33400 | Biomedical computing |
| **Louisiana Tech U** | U | 1 | 2 | | | | | X | ENGR 120 | Engineering problem solving I |
| | U | 1 | 2 | | | X | | | ENGR 121 | Engineering problem solving II |
| | U | 1 | 2 | X | | | | | ENGR 122 | Engineering problem solving III |
| **Marquette U** | D | 2 | 2 | X | C | | | X | BIEN 1120 | Inttro to computing for BME |
| | D | 2 | 2 | | | | X | | BIEN 2300 | Biomedical circuits and electronics |
| | D | 3 | 3 | | | | X | | BIEN 3200 | Computer applications in BME |
| **Rensselaer Poly.** | U | 1 | 1 | | | | | X | ENGR 1200 | Engineering graphics and CAD |
| **Institute** | D | 2 | 1 | X | | | | | CSCI 1190 | Beginning programming for engrs |
| | D | 3 | 4 | | | | X | | BMED 4200 | Modeling of Biomedical Systems |
| **U of Cincinnati** | U | 1 | 3 | X* | P | | | X | ENED 1100 | Foundations of engring design thinking I |
| | U | 1 | 3 | X | P | | | X | ENED 1120 | Foundations of engring design thinking II |
| | D | 1 | 2 | | | | | X | BME 1050C | Computer aided design |
| **U Colorado** | D | 1 | 2 | | | X | | X | BIOE 1010 | Bioeng. protoyping and design II |
| **Denver** | D | 1 | 2 | | | | | X | BIOE 1020 | Bioeng. protoyping and design I |
| | D | 2 | 2 | X* | C++ | | | | BIOE 2010 | Intro to programming for bioengineers |
| | D | 2 | 2 | | | | X | | BIOE 2020 | Intro computational methods for bioengrs |
| **U Louisville** | U | 1 | 2 | | P | | | | ENGR 110 | Engineering methods tools and practice 1 |
| | U | 1 | 1 | | | | | X | ENGR 151 | Engineering graphics technology |
| | D | 2 | 3 | | | | X | | BE 340 | Computational methodol. in bioeng'ring |
| **U Miami** | U | 1 | 3 | X | C++ | | | X | BME 111 | Introduction to engineering |
| | D | 2 | 3 | X | | | | | BME 211 | Intro to programming for BMEs |
| | D | 2-3 | 3 | | | | X | | BME 310 | Mathematical analysis in BME |
| **U Texas Austin** | D | 1 | 3 | * | | | | | BME 303L | Intro to BME Design |
| | D | 1 | 3 | | P/C++ | | | | BME 303 | Into to computing |
| | D | 3 | 3 | | | | X | | BME 313L | Intro to numerical methods in BME |
| **U Virginia** | U | 1 | 4 | X | | | | | ENGR 1624 | Intro to Engineering |
| | U | 1 | 3 | | P | | | | CS 1110 | Intro to programming |
| | D | 2 | 3 | | | | X | | BME 2315 | Computational biomed. engineering |
| | D | 3 | 3 | | | | X | | BME 3310 | Biomed. systems analysis and design |

The analysis done here was the first attempt to identify what computing knowledge is required by all the accredited BME programs. It is clear that MATLAB is the dominant computer language, but that object-oriented programming skills are not rare. The number of universities requiring other computer skills (CAD, microcontroller use, LabVIEW) was found to be small, and they are undoubtedly less common than the main computer languages. But the data collected about these programming skills is likely to be an underestimate of actual knowledge by BME students for several reasons. Course descriptions are brief and cannot capture all aspects of a course, so CAD could be left out. These also tend to be skills that can be taught in design courses on an "as-needed" basis. Also, for these topics, studying tracks or concentrations undoubtedly would show more courses taken by subsets of students. At the author's university, neither CAD nor microcontroller programming is included in any required course, and only a minority of BME students take the non-required formal courses that are available. Some students will learn some aspects of CAD as a hobby in Maker Spaces that have become prevalent. And, when these topics become relevant in senior design, a team may lean on the members who have learned the requisite skills in courses or another way, so that the team has a skill that each member does not.

Ideally, the computer skills that students learn should match those that they will need in their careers. While learning to program in any language undoubtedly prepares one to learn other languages more easily, it is still of interest to consider what students are likely to need upon graduation. However, BME students go in many directions, and even if they go to industry, there appears to be no single standard that all biomedical companies use. A few references may help universities sort out how to provide the most relevant education. First, a study done several years ago identified, using a Delphi Method, key areas of computer proficiency, which resulted in essentially a taxonomy and ranking of computer skills for engineers, only one of which was programming.[7, 8] BME departments could use this to evaluate where their students stand on the expected skills and what needs to be added. For engineers other than computer scientists, this study found that the most important skills, were, in descending order: 1) Microsoft Office tools, 2) Web search, 3) Data analysis skills, 4) Communication tools, 5) Process modeling and design, 6) Database fundamentals, and 7) Programming skills.[7] Second, in terms of the skills needed for the biomedical space, discussions at the Fourth Biomedical Engineering Education summit identified data science (including data collection, integration across datasets, visualization, storage, and analysis) as an important area for growth [3] and referred attendees to a consensus report from the National Academies.[9] The needs in data science for biotechnology may be well represented by an article about biomedical scientists at the University of California, San Francisco, who wanted to learn R and Python in order to analyze their data and engage in "Big Data" science [10] Finally, several websites give the relative popularity of different languages, and TIOBE, a Dutch company that concerns itself with software quality, has been tracking use of different languages for nearly two decades.[11] While the rankings in the monthly TIOBE Index are intended mainly for software developers, which will include few BMEs, it is still useful to note that the top four languages, in descending order, are Java, C, Python, and C++, the languages that were identified as being the main OOPs taught to BME students (Visual Basic is 5$^{th}$).[11] It should be a relief to academics that these languages have held relatively stable positions for years, with the exception of Python, which has grown considerably since 2018. In the TIOBE Index, MATLAB has dropped from 11$^{th}$ to 15$^{th}$ as a language for software developers in just the last year, but many websites attest to its continued

relevance for data analysis in engineering and other STEM fields. R is gaining in popularity, having gone from 14th to 11th since March of 2019.

Probably the way to further our understanding of the penetrance of computer skills apart from basic programming is through surveys of either instructors or students. However, the problem with surveys is that they never elicit data from the whole community. Only 57 of the 118 programs completed the course survey for the 2019 Biomedical Engineering Education Summit. Thus, while individuals from each institution could ensure accuracy of the information from that institution, it would not be a complete sample. Nevertheless, supplementing the present information with a survey that could obtain deeper information might be valuable.

**References**


[1] D. W. Gatchell, and R. A. Linsenmeier, "Similarities and Differences in Undergraduate Biomedical Engineering Curricula in the United States," in American Society for Engineering Education Annual Conference and Exposition, Indianapolis, IN, 2014. https://peer.asee.org/23015

[2] R. A. Linsenmeier, and A. Saterbak, "Fifty years of biomedical engineering undergraduate education," *Ann Biomed Eng*, https://doi.org/10.1007/s10439-020-02494-0 , 2020.

[3] J. A. White, D. P. Gaver, R. J. Butera, Jr., B. Choi, M. J. Dunlop, K. J. Grande-Allen, A. Grosberg, R. W. Hitchcock, A. Y. Huang-Saad, M. Kotche, A. M. Kyle, A. L. Lerner, J. H. Linehan, R. A. Linsenmeier, M. I. Miller, J. A. Papin, L. Setton, A. Sgro, M. L. Smith, M. Zaman, and A. P. Lee, "Core Competencies for Undergraduates in Bioengineering and Biomedical Engineering: Findings, Consequences, and Recommendations," *Ann Biomed Eng*, vol. 48, no. 3, pp. 905-912, Mar, 2020.

[4] V. Beal. "High-level language," Feb. 2, 2020; https://www.webopedia.com/TERM/H/high_level_language.html.

[5] V. Beal. "Object definition," Feb. 2, 2020; https://www.webopedia.com/TERM/O/object_oriented_programming_OOP.html.

[6] B. Putano. "A Look at 5 of the Most Popular Programming Languages of 2019," https://stackify.com/popular-programming-languages-2018/.

[7] D. Raubenheimer, E. Weibe, and C.-L. Ho, "Computational thinking: What should our students know and be able to do?," in American Society for Engineering Education Annual Conference and Exposition, Louisville, KY, 2010. https://peer.asee.org/15877

[8] E. Weibe, C.-L. Ho, D. Raubenheimer, L. Bullard, J. Joines, C. Miller, and G. Rouskas, "Computing across curricula: The view of industry leaders," in American Society for Engineering Education Annual Conference and Exposition, Austin TX, 2009. https://peer.asee.org/4918

[9] E. National Academies of Sciences, and Medicine, *Data Science for Undergraduates: Opportunities and Options*, The National Academies Press., Washington, DC., 2018.

[10] A. Deardorff, "Why do biomedical researchers learn to program? An exploratory investigation," *J Med Libr Assoc*, vol. 108, no. 1, pp. 29-35, Jan, 2020.

[11] TIOBE. "The TIOBE Index," https://www.tiobe.com/tiobe-index/.