
This is a textbook for a scientific computing course for graduate students and advanced undergraduates. It can also be used for self study. Why C? C is relatively old and well established; it’s not this week’s fashionable language. Compilers are freely available for every platform. C has been standardized. C gives the programmer a great deal of control, especially with respect to memory management. C is a small language. What are the drawbacks? Mainly the steep learning curve. The author tells us in the preface that his students generally have already had a one-semester introduction to C or some similar language. That would seem to be almost necessary, as the learning curve would otherwise be too steep.

The book is in two parts. The short first part consists of common background information, including material about file management, streams and the UNIX shell, pointers and arrays, and the use of makefiles.

The second part, which is the heart of the book, consists of the projects. The first are quite simple and address common needs such as memory allocation. The complexity builds from there. The last two projects produce finite element code for solving elliptic PDE on unstructured triangular meshes. Modularity and reuse of code are emphasized, but that does not mean that the projects have to be done strictly in order. The dependencies are clearly specified in a table. For example, to get to the two finite element projects one would first have to complete the projects on memory allocation, dynamic allocation of vectors and matrices, a data structure for sparse matrices, UMFPACK for solving sparse linear systems, two projects on numerical quadrature, and triangulation with the Triangle library. Another independent (except for memory allocation) sequence of projects progresses from Haar wavelets to image I/O to image analysis. There is a fun project that simulates the evolution of an asexually reproducing species. (Sexual reproduction would surely have been even more fun but would also have made the programming task much more difficult.) The evolution project relies on earlier projects on random number generation and linked lists. Other chapters include the Nelder–Mead simplex method, trusses, finite difference schemes for the heat equation, and the porous medium equation. This is quite a nice variety of topics.

The mathematics behind each project is discussed, but not in great technical detail. For example, the discussion of the Galerkin method in the finite element chapters does not mention Sobolev spaces. A lot of code is included in the book, but there are no complete programs. It is the student’s task to understand each code fragment and figure out how to put the pieces together to make working code.

The author maintains a book website, which provides additional resources.

The student who successfully works through this book will have learned a great deal about programming and will also have been exposed to several important methods for attacking scientific computing problems.

David S. Watkins
Washington State University

Copyright © Society for Industrial and Applied Mathematics