

AJP DRAFT Resource Letter CP-2: Computational Physics

Rubin H Landau*

Department of Physics, Oregon State University, Corvallis, Oregon 97331[†]

(Dated: November 8, 2007)

This Resource Letter provides a guide to print and electronic literature relevant to a computational physics course. The multidisciplinary aspect of computational physics and its relation to computational science is reflected in the sections on: Courses and Programs, Journals, Conferences and Organizations, Books, Languages and Environments, Tools, Parallel Computing, and Digital Libraries. ©2007 American Association of Physics Teachers.

PACS numbers:

Contents

I. Introduction	1
A. CP's Multidisciplinary Nature	1
B. The Contents of CP	2
II. Courses and Programs	2
III. Journals	4
IV. Conferences & Organizations	4
A. Conferences	4
B. Supercomputer Centers & Grids	5
C. Groups with CSE Education Focus	6
V. Books	6
A. CP Books	6
B. Applied Math & CSE Books	7
VI. Tools, Languages, Environments	8
A. Visualization	9
B. Fortran	10
C. C, C++ (OOP)	10
D. Java	10
E. Python	11
F. Matlab, Octave & Mathcad	11
G. Maple & Mathematica	11
H. Development Environments, Debuggers	12
I. Markup Languages	12
J. SQL: Database Languages	13
K. Concept Mapping	13
VII. Parallel Computing	13
VIII. Digital Libraries	14
A. Subroutine Libraries	14
B. General Digital Library	15
C. Digital Sky Libraries	15

I. INTRODUCTION

A. CP's Multidisciplinary Nature

In 1995, Paul DeVries assembled Resource Letter CP-1 in Computational Physics as an aide to those teaching a course in Computational Physics. [We refer to that letter here as (V A₄), which means that it is resource number 4 in §V A.] The number of resources now available has increased significantly, some of the original computational physics (CP) texts are now out of print, Java and Python are now popular programming languages, and the exponential growth in the power and pervasiveness of computers has made computation essential to all areas of science. In fact, although it appears that most physics classes now incorporate computation in some way, we would not call them “computational physics” classes if their use of computation is just to improve physics education. However, we would if they incorporate and analyze modern computational techniques as used in research to solve problems.

In this resource letter we adopt the view (Fig. 1 left) that CP is a multidisciplinary field of study that requires students to master physics along with computer science and applied mathematics. Teaching such a course creates new scholarly materials that fuse chunks of knowledge together (the central circle in Fig. 1 left). A CP course thus acts as a bridge connecting the disciplines and permits students to learn all three disciplines simultaneously in the process of solving concrete problems.

Of course, this inclusion into education of the computational tools and methods of modern research is happening in all the sciences, and accordingly CP is a sub-field of what is often called “Computational Science and Engineering (CSE)”. The similarities of techniques and methods throughout the computational sciences form a common educational base that permit its practitioners to solve realistic and practical problems in a broad range subjects, and to extend education to include problems that do not have analytic solutions. The effectiveness and power of computation in so many fields has led some people to add simulation to experiment and to theory as key avenues followed in searching for scientific truth (Fig. 1 right).

Recent advances in computational science affect this

*Director, Computational Physics for Undergraduates (CPUG) program; Electronic address: rubin@science.oregonstate.edu.

[†]URL: www.science.oregonstate.edu/~rubin

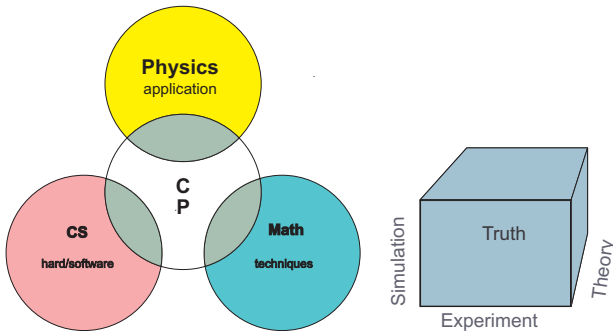


FIG. 1: *Left:* Computational Physics as the both the union of physics with computer science and applied mathematics, and as a bridge connecting the three traditional discipline [based on (II.9)]. *Right:* The addition of simulation as a path towards scientific truth.

resource letter in a number of ways. On the one hand, we have listed some CSE resources that we believe are of value in CP, even though they may not be derived from physics. [For example, resource number 9 in §II, which we designate as (II.9), indicates strong similarities between CP and CSE undergraduate degree programs.] You will note in these resources the inclusion of many tools in addition to traditional academic materials; this reflects the practical problem-solving paradigm at the heart of computational sciences, and the fact that these tools help define what computational physicists do and how they do it.

On the other hand, computation is now so integral to physics research that there are numerous papers and books combining computation with advanced or specialized topics. Consequently, we have had to leave off many excellent resources that are important for research, but which we do not view as general enough for a CP course. Accordingly, we would encourage another resource letter that collects resources which incorporate computation into the tradition physics education. This would include pioneering books like P. Visscher’s (*Fields and Electrodynamics: A Computer-Compatible Introduction* (Wiley, New York, 1988) for E&M, and introductory physics projects such as R. Chabay and B. Sherwood’s *Matter & Interactions*, (Wiley, New York, 2002), and W. Christian and M. Belloni’s *Physlet Physics* (Pearson/Prentice Hall, Upper Saddle Rive, 2004).

B. The Contents of CP

There is no standard curriculum for a course or degree in CP. Indeed, with only five CP degree programs in the country, the field is still very much under development (see §II for resources on various CSE programs and courses). A major challenge in a CP course, which is reflected in all the texts in §V A, is that much of the coverage needs to be of numerical methods, computational tools and computer languages; it is hard to get into any

significant physics until that is done. To illustrate, in Fig. 2 we present a *concept map* of CP showing the major disciplines at the top and the linking of subjects (§VI K). Although this detailed a map is complicated, it can be drawn at higher level of abstraction, or it can be used to outline different pathways for different learner levels. In any case, maps are useful when placing CP materials into a digital library, where they become *content maps* when the concepts are linked to information.

When reflecting upon what subjects might be appropriate for future CP education, it is helpful to look to the future of computation, where the funding agencies envision petascale computing by 2011. (A petaflop is 10^{15} floating point operations per second, which means that if Moore’s law were applied to our desktop computers, we would have to wait until 2037 for a petaflop.) It hard to imagine the type of problems that can be solved with petascale computing. Although we can always repeat our old simulations with higher resolution and with higher accuracy, petascale computing will be most appropriate for new problems that involve multi-time and multi-space scales, and, accordingly, multiphysics. They include: quantum chromodynamics of particle interactions; radiative, dynamic and nuclear physics of stars and the collision of stars; heterogeneous catalysis on both semiconductor and metal surfaces; intermittency, stratification, rotating turbulence in magnetic fluids; first principles simulations properties of matter under extreme conditions; molecular understanding of friction and lubrication; designing molecular electronic devices; generation and evolution of magnetic fields in planets and stars; modeling interaction of the Earth’s magnetic field with coronal mass ejections; formation and evolution of galaxies; burning plasmas and magnetic confinement techniques; formation of planetary nebula via low Mach-number flows.

II. COURSES AND PROGRAMS

There are less than 20 US undergraduate degree programs in all of the computational sciences, and somewhere around five degree programs in CP. Web sites and papers describing these program, and those for minors, options, concentrations, tracks, foci, *etc.*, as well as graduate programs are valuable CP resources.

1. **B.S. Degree in Computational Physics, SUNY, Buffalo**, a multi-department effort, undergrad-catalog.buffalo.edu/academicprograms/comphys.shtml.
2. **Computation in the Lawrence Physics Curriculum**, D. M. Cook, Proc. Int. Conf. on Computational Science, ICCS 2001, ed. Vasil N. Alexandrov et al. (Springer, Berlin, 2001) 1074–1083; Comput. Phys. **11**, (1997) 240–245; **11**, (1997) 331–335; (2006) www.lawrence.edu/dept/physics/cc11.

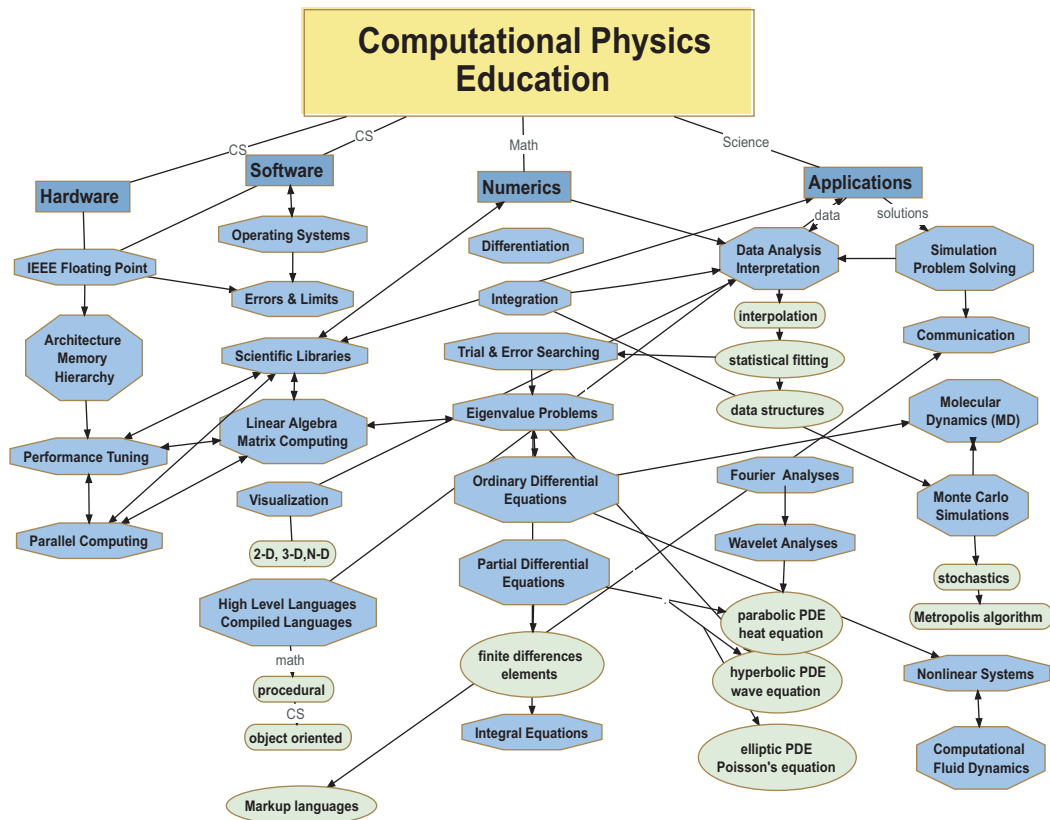


FIG. 2: A concept map of CP created with *VUE* (VIK.1). When the concepts are linked to information, the concept map becomes a content map. Simpler layers of maps and pathways can be formed for different learner levels. Information can also be placed in the links. [From (V A.13).]

3. **Computational Physics and the Undergraduate Curriculum**, H. Gould, *Computer Physics Communications* **127** (1), 6–10 (2000).
4. **Computational Physics: A Better Model for Physics Education?**, R. H. Landau, *Computing in Sci. & Engr.* **8**, 22–30 (2006). Survey results and relevant statistics; **Computational Physics for Undergraduates, the CPUG Degree Program at Oregon State Univ.**, R. H. Landau, *Computing in Sci. & Engr.* **6**, 68–75 (2004); Program description at www.physics.oregonstate.edu/CPUG.
5. **Computational Physics for Upper Level Courses**, **AAPT Topical Conference**, Davidson College, July 2007. There should be a number a papers appearing in the *Amer. J. of Phys.* (possibly September 2008) reporting on this conference. See too www.opensourcephysics.org/CPC.
6. **Computer Physics B.S., Illinois State Univ.**, possibly the first undergraduate CP program in the country, with CP classes starting in the 1970's, www.phy.ilstu.edu/programs/CompPhys.
7. **Computational Science Demands a New Paradigm**, D. E. Post and L. G. Votta, *Physics Today* **58** (1) 35–41 (2005).
8. **Teaching Computational Physics as a Laboratory Science**, R. L. Spencer, *Am. J. Phys.* **73**, 151–153 (2005).
9. **Elements of Computational Science Education**, O. Yaşa and R. Landau, *SIAM Rev.* **45**, 787–805 (2003), general program characteristics, contents and balance of subjects.
10. **Graduate Education in Computational Science and Engineering**, SIAM Working Group on CSE Education, *SIAM Rev.* **43**, 163–177 (2001), describes CSE contents and programs, www.siam.org/students/resources/report.php.
11. **Moderatorship in Physics and Computer Simulation**, an undergraduate degree program in CP at Trinity College, Dublin, www.tcd.ie/Physics/Courses/page39.php.
12. **New Perspective on Computational Science Education**, O. Yaşar, K. S. Rajasethupathy, R. E. Tuzun, R. A. McCoy, and J. Harkin, *Computing*

in *Sci. & Engr.* **5**, 74–79 (2000). Discusses SUNY Brockport’s Department of Computational Science.

13. **Ralph Regula School of Computational Science (RRSCS)**, Ohio’s statewide virtual school, contains some physics courses as well. Although in initial stage, may be model for other states, www.rrscs.org.
14. **Survey of Computational Science Education**, C. Swanson, a long-running, high-quality survey that lists programs and courses in all computational sciences, and contacts entries to ensure viability, www.krellinst.org/services/technology/CSE_survey.
15. **Undergraduate Computational Science and Engineering Education**, SIAM Working Group on CSE Undergraduate Education, (apparently not published), www.siam.org/about/pdf/CSE_Report.pdf.

III. JOURNALS

There are many computer science and IEEE journals dealing with aspects of computing important to CP, and we specify just some below. The main journals containing discussions of computational education developments appear to be *Computing in Science & Engineering*, *SIAM Review* and *American Journal of Physics*.

1. **American Journal of Physics**, devoted to instructional and cultural aspects of physical science, with increasing interest in CP, scitation.aip.org/ajp.
2. **Computer Physics Communications Program Journal**, started in 1969 as a pioneer in establishing importance of computer programs in the development of CP. Associated library now contains more than 2000 programs, www.sciencedirect.com/science/journal/00104655.
3. **Computing in Science & Engineering (CISE)**, joint publication of the IEEE Computer Society and the American Institute of Physics, merging (in 1999) the AIP’s *Computers in Physics* and the IEEE’s *Computing in Science and Engineering*. More a readable magazine than an archival journal; September/October 2006 issue devoted to *Computation in Physics Courses*, cise.aip.org.
4. **European Journal of Physics**, the European Physical Society’s physics education journal, www.iop.org/EJ/journal/0143--0807.
5. **HPCwire**, a weekly newsletter with plenty of advertisements, but also the latest news about high-performance computing trends, www.hpcwire.com.

6. **IEEE Computer Magazine**, covers all aspects of computer science, computer engineering, technology, and applications, www.computer.org/portal/site/computer/index.jsp.
7. **Journal of Computational Physics**, Elsevier, www.elsevier.com/wps/find/journaldescription.cws_home/622866/description#description
8. **Linux Journal**, a monthly magazine of the Linux community and a good place to learn about available tools, www.linuxjournal.com.
9. **Mathematics and Computers in Simulation**, *Transactions of IMACS*, Elsevier, www.elsevier.com/wps/find/journaldescription.cws_home/505615/description#description.
10. **SIAM (Society for Industrial and Applied Mathematics) Review**, has been a leader in computational science/mathematics, and particularly CSE education, epubs.siam.org/SIREV/sirev.toc.html.

IV. CONFERENCES & ORGANIZATIONS

Although the National Science Foundation and university faculties seem to have difficulties dealing with multidisciplinary fields, many CP practitioners find support in the resources and colleagues of the associated disciplines. In this section we indicate the websites of organizations that support computational science/physics. See too Courses and Programs in §II and Digital Libraries in §VIII.

A. Conferences

1. **Association for Computing Machinery**, supports advancing computing as a science and a profession; see too their digital library (VIII B.7), www.acm.org.
2. **APS Division of Computational Physics (DCOMP) Annual Meetings**, explores the use of computers in physics research and education. At times the division has stand-alone meetings, while at other times the meeting is joint with other APS meetings, or with international conferences on CP, units.aps.org/units/dcomp.
3. **Conferences on Computational Physics (CCP)**, continues America and European Physical Societies’ *Physics Computing* conferences: Boston 1989, Amsterdam 1990, San Jose 1991, Prague 1992, Albuquerque 1993, Lugano 1994, Pittsburgh 1995, Krakow 1996, Santa Cruz 1997, Granada 1998, Atlanta 1999, Brisbane 2000, Aachen 2001, San Diego 2002, Beijing 2003, Genova 2004, Los

- Angeles 2005, Gyeongju (2006), Brussels (2007), ccp2007.ulb.ac.be.
4. **European Physical Society Computation Group**, www.eps.org/directory/epsentity_view?uid=Computational+Physics+Group.
 5. **Gordon Research Conferences**, in addition to focusing on traditional research topics, also examine issues related to computation and education, www.grc.org.
 6. **IEEE Computer Society**, dedicated to advancing the theory, practice, and application of computer and information processing technology, www.computer.org/portal/site/ieeecs/index.jsp
 7. **IEEE Education Activities**, courses and more, www.ieee.org/web/education/home.
 8. **International Conferences on Computational Science (ICCS)**, a series of yearly conferences in nice places: Krakow 2008, Beijing 2007, Reading 2006, UK, Atlanta 2005, Krakow, 2004, Melbourne/St. Petersburg 2003, Amsterdam, 2002, San Francisco, 2001, www.iccs-meeting.org.
 9. **NSF's CISE Pathways to Revitalized Undergraduate Computing Education**, www.nsf.gov/div/index.jsp?org=CNS.
 10. **SCXY Education Program**, workshop at the SuperComputing conferences focusing on hands-on activities to engage teachers in applying computational science, grid computing and high performance computing resources, sc07.supercomputing.org/?pg=education.html.
 11. **SIAM Conferences on Computational Science & Engineering**, promotes computational science and engineering as an academic discipline, and simulation as mode of scientific discovery, www.siam.org/meetings/cse07.
 4. **Maui High Performance Computing Center**, affiliated with the Univ. of New Mexico, www.mhpcc.edu.
 5. **National Center for Atmospheric Research (Boulder)**, www.ucar.edu.
 6. **National Center for Supercomputer Applications at Univ. of Illinois**, www.ncsa.uiuc.edu.
 7. **National Energy Research Scientific Computing Center (NERSC)**, www.nersc.gov, Department of Energy.
 8. **Oak Ridge National Center for Computational Sciences**, nccs.gov.
 9. **Ohio Supercomputer Center**, a diverse organization with a strong emphasis on education (see too Ralph Regula School of Computational Science), www.osc.edu.
 10. **Open Science Grid (OSG)**, focused on meeting the computing and data management needs of researchers, especially collaborative science requiring high throughput computing, www.opensciencegrid.org.
 11. **Pittsburgh Supercomputing Center**, www.psc.edu.
 12. **San Diego Supercomputing Center**, www.sdsc.edu.
 13. **TerraGrid**, an open scientific discovery infrastructure combining leadership class resources at nine partner sites to create an integrated, persistent computational resource. Look here for links to the other grids, www.teragrid.org/about/: Open Science Grid (OSG), Special PRiority and Urgent Computing Environment (SPRUCE), Massive Pulsar Surveys using the Arecibo L-band Feed Array (ALFA), National Virtual Observatory (NVO), Linked Environments for Atmospheric Discovery (LEAD), Computational Chemistry Grid (GridChem), Computational Science and Engineering Online (CSE-Online), Network for Earthquake Engineering Simulation (NEES), Network for Computational Nanotechnology and nanoHUB, TeraGrid Geographic Information Science Gateway (GISolve), CIG Science Gateway for the Geodynamics Community, The Earth System Grid (ESG), National Biomedical Computation Resource (NBCR), Developing Social Informatics Data Grid (SIDGrid), Neutron Science TeraGrid Gateway (NSTG), Biology and Biomedicine Science Gateway, Open Life Sciences Gateway (OLSG), The Telescience Project, Grid Analysis Environment (GAE), TeraGrid Visualization Gateway.
 14. **Texas Advanced Computing Center**, www.tacc.utexas.edu.

B. Supercomputer Centers & Grids

Not only are the various supercomputer centers an excellent resource for state-of-the-art computational tools and research, but they also contain valuable resources classified as “Education, Outreach and Training”.

1. **Arctic Region Supercomputing Center**, www.arsc.edu.
2. **Cornell Theory Center**, www.tc.cornell.edu.
3. **Department of Energy Advanced Scientific Computing Advisory Committee**, www.sc.doe.gov/ascr/ASCAC/ASCAC.html.
14. **Texas Advanced Computing Center**, www.tacc.utexas.edu.

C. Groups with CSE Education Focus

1. **Engaging People in CyberInfrastructure (EPIC)**, a diverse team of educators with members throughout the country, www.eotepic.org.
2. **Krell Institute**, Emphasizes development of computational workforce and communicating science. See Computational Science Fellowship programs, Undergraduate Computational Engineering and Sciences (UCES) Award, Computational Science and Engineering Education Undergraduate Programs listing, www.krellinst.org.
3. **Ralph Regula School of Computational Science (RRSCS)** an Ohio statewide program at all levels of education in which participating colleges and universities confer degrees and certificates, www.rrscs.org.
4. **Shodor Education Foundation**, www.shodor.org; **Computational Science Education Reference Desk (CSERD)**, a pathway project of the National Science Digital Library (VIII B.6), www.shodor.org/refdesk; **National Computational Science Institute (NCSI)**, Provides on-the-road workshops, www.computationalscience.net.
3. **A First Course in Computational Physics**, P. L. DeVries (Wiley, New York, 1994). Good coverage of the basic subjects with careful pedagogy. Microsoft Fortran codes on 3" disk. Appears to be out of print. (E)
4. **Resource Letter CP-1: Computational Physics**, P. L. DeVries, *Am. J. Phys.* **64**, 364–367 (1996). (E)
5. **Numerical Methods for Physics**, 2nd ed., A. L. Garcia (Prentice Hall, Upper Saddle River, 2000). Concise, yet covers a good number of computational and physics topics. Matlab, Fortran, and C++ codes available from Web site, www.algarcia.org/nummeth/nummeth.html. (I)
6. **Computation in Modern Physics**, 3rd ed., W. R. Gibbs (World Scientific, Singapore, 2006). First half of text deals with an assortment of computational techniques, while second half applies them to quantum-mechanical problems in nuclear physics. (I–A).
7. **Computational Physics**, 2nd ed., N. Giordano and H. Nakanishi, (Prentice Hall, Saddle River, 2005). An excellent text with ample explanations and interesting applications. Presents only pseudocodes, which makes the text language independent, but with higher start up costs. No CD, but author's web site provides some True basic and Fortran codes, www.physics.purdue.edu/~hisao/book. (I)

V. BOOKS

The books listed here tend to be of a general nature appropriate for course use. We do not list the large number of books dealing with computation in specific areas, except for some of the Applied Math & CSE books. Some of these texts provide annotated bibliographies indicating specialized books and articles for individual topics. There are some very good book that we leave out because their language is just too obsolete, although we do include some out-of-print books that are still available.

A. CP Books

1. **Computation and Problem Solving in Undergraduate Physics**, D. M. Cook. The self-published product of a project that incorporated various forms of computation throughout an undergraduate curriculum. Alternate versions of the materials available using languages and tools specified by reader. www.lawrence.edu/dept/physics/ccli.
2. **Introduction to Computational Physics**, M. L. DeJong (Addison-Wesley, Reading, 1991). More elementary than most other texts, with detailed pseudocodes and Basic programs. Appears to be out of print. (E)
8. **An Introduction to Computer Simulation Methods**, 3rd ed. H. Gould, J. Tobochnik and W. Christian (Addison-Wesley, Reading, 2006). A new edition of a well-tested and excellent text, now with Java and the Open Source Physics library (which has strong object-orientation and some overhead), but with a continued emphasis on statistical physics. (I)
9. **Computational Methods in Physics, Chemistry and Biology**, P. Harrison, (Wiley, New York, 2001). An unusual, concise book that tries to inspire students by introducing many CP topics at an elementary level without much in the way of background. C codes in text. (E)
10. **Computer Simulation Using Particles**, R. W. Hockney, and J. W. Eastwood (Adam Hilger, Bristol, 1988). A lot can be simulated as the motion of particles. (A)
11. **Introductory Computational Physics**, A. Klein and A. Godunov (Cambridge Univ. Press, Cambridge, 2006). Careful treatment of a few basic topics, not many applications. Useful preparation for those using CERN's ROOT framework. Programs mainly in C++, with calls to Fortran, using Linux and free software packages. (I)

12. **Computational Physics**, S. E. Koonin (Benjamin, Menlo Park, 1986); **Fortran Edition**, S. E. Koonin and D. Meredith (Westview Press/Perseus Books, Cambridge, 1998). Probably the first book on CP, serious, but a challenge to use with undergraduates. Basic and Fortran codes. (I–A)
 13. **A Survey of Computational Physics**, *Introductory Computational Science*, R. H. Landau, M. J. Paez, and C. C. Bordenaiu (Princeton Univ. Press, Princeton, 2008). A fuller range of topics than (14), with a computational science and practical viewpoint. Procedural Java codes in the text, Fortran and C codes and tutorials on the CD, and video lectures on the Web, physics.oregonstate.edu/~rubin. (I)
 14. **Computational Physics, Problem Solving with Computers**, 2nd Edition, R. H. Landau, M. J. Paez, and C. C. Bordenaiu (Wiley-VCH, Berlin, 2007). Java version of 1997 book. Extended survey in (13). (I)
 15. **Computational Techniques in Physics**, P. K. MacKeown and D. J. Newman (Adam Hilger, Bristol, 1987). Some unusual (and thoughtful) applications, although not a broad spectrum of computational subjects. No programs in text, but a CD was provided by publisher. (I)
 16. **An Introduction to Computational Physics**, 2nd ed., T. Pang (Cambridge Univ. Press, Cambridge 2006). First half of text deals with computational tools, second half with applications, including unusual ones on genetic programming and renormalization group. Java programs in text, with Fortran and C programs on author's Web page, www.physics.unlv.edu/~pang/cp.html. (I)
 17. **Theoretical Physics on the Personal Computer**, 2nd ed., E. W. Schmid, G. Spitz, and W. Löschi (Springer-Verlag, Berlin, 2000). The 2nd ed. of one of the earliest CP texts contains great deal of physics and insight for this small a book, although on limited topics. Fortran, CD. (I–A)
 18. **A Physicist's Guide to Mathematica**, P. Tam (Elsevier, Oxford, 1997). Essentially a CP course using Mathematica. (I)
 19. **Computational Physics**, J. M. Thijssen (Cambridge Univ. Press, Cambridge, 1999). A graduate/research level text delving into the theoretical underpinnings of CP, with emphasis on condensed matter. Programming implications discussed, but no programs printed in text, no accompanying CD and only few programs on web, www.tn.tudelft.nl/tn/People/Staff/Thijssen/comphybook.html. (A)
 20. **Computational Physics, An Introduction**, 2nd ed., F. J. Vesely (Plenum Press, New York, 2001). Mainly computational techniques, with a few serious applications, but not much discussion. No codes. (E–A)
 21. **Introduction to Computer Simulation**, M. M. Wolfson and G. J. Perl (Oxford Univ. Press, Oxford, 1999). Small number of topics. (E–I)
- ### B. Applied Math & CSE Books
1. **Numerical Methods that Work**, F. S. Acton (Math. Assoc. of Amer., DC, 2007) is a 1970 classic reissued by the MAA. (E–I).
 2. **The Illustrated Wavelet Transform Handbook**, P. S. Addison (Institute of Physics Publishing, Bristol, 2002). (I)
 3. **Computational Methods for Applied Science & Engineering**, M. G. Ancona (Rinton Press, Princeton, 2002), uses Mathcad with digital text available. (E–I)
 4. **Engineering Programming, C, MATLAB, and Java**, M. Austin and D. Chancogne (Wiley, New York, 1999). A set of four tutorials. (E)
 5. **The DFT, An Owner's Manual**, W. L. Briggs and V. E. Henson (SIAM, Philadelphia, 1995). Great name for serious study. (A)
 6. **Numerical Methods for Engineers with Software and Programming Applications**, 5th ed., S. C. Chapra (McGraw-Hill, New York, 2006). Matlab and Excel programs. An impressively broad view with excellent pedagogy. (E–I)
 7. **Computational Science Education Project**, dated, but a high-quality, early exploration into the use of the Web for computational education, csep1.phy.ornl.gov/csep.html. (E)
 8. **Projects in Scientific Computing**, R. E. Crandall (Springer-Verlag, New York, 2000). Excellent selection of topics, with focus on algorithms. (I)
 9. **The Fast Fourier Transform for Experimentalists**, D. Donnelly and B. Rust *Computing in Sci. & Engr.* **7**, 80–88 (2005), one of a series of articles. (I)
 10. **An Introduction to High Performance Scientific Computing**, L. D. Fosdick, E. R. Jesup, C. J. C. Schauble, and G. Domik (MIT Press, Cambridge, 1996). An exceptionally complete, yet readable, survey of computer science basics and tools of computational science, with a few applications. (I)

11. **Fundamentals of Wavelets**, J. C. Goswami and A. K. Chan (Wiley, New York, 1999). (I)
12. **Scientific Computing**, 2nd ed., M. T. Heath (McGraw-Hill, New York, 2002). A practical guide to numerical computing. No codes, but useful book site, <http://www.cse.uiuc.edu/heath/scicomp>. (E-I)
13. **Introduction to Scientific Computation and Programming**, D. T. Kaplan (Thompson, Brooks/Cole, Belmont, 2004). Matlab code fragments in text. Fundamentals of computer science for scientists, with very nice examples. (E)
14. **A First Course in Scientific Computing, Symbolic, Graphic, and Numerical Modeling Using Maple, Java, Mathematica, and Fortran 90**, R. H. Landau (Princeton Univ. Press, Princeton, 2005). Also L^AT_EX and alternate versions of text in Mathematica and Fortran 90 on the CD along with notebooks, workbooks and codes. (E)
15. **Numerical Methods for Mathematics, Science and Engineering**, 2nd ed., J. Mathews (Prentice Hall, Englewood Cliffs, 1992). (I)
16. **Numerical Recipes**, 3rd ed. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling (Cambridge Univ. Press, Cambridge, 2007). A ground-breaking classic from a scientist's point of view, with erudite writing no less. All codes from previous versions in all computer languages are included in this new edition. (I-A)
17. **High Performance Computing**, 2nd ed., C. Severance and K. Dowd (O'Reilly, Sebastopol, 1998). Contains useful, down-to-earth discussions of how architecture influences performance and how to benchmark and tune codes. (A)
18. **Computing for Scientists and Engineers**, W. J. Thompson (Wiley, New York, 1992). One of the earlier books that introduced us to the field, and though thin, contains much insight about numerical methods. (E)
19. **Computational Methods in Physics and Engineering**, 2nd ed., S. S. M. Wong and S. S. Wong (World Scientific, Singapore, 2003). Full discussions of computational techniques from a physicist's point of view, but little in the way of applications. Fortran codes on CD and Web, www.worldscibooks.com/physics/3365.html. (I)
20. **A First Course in Computational Physics and Object-Oriented Programming with C++**, D. Yevick (Cambridge Univ. Press, Cambridge, 2005). Basically a C++ programming course with a few examples. (I)
21. **Introduction to Scientific Programming, Computational Problem Solving Using Maple and C**, J. L. Zachary (Springer/Telos, New York, 1996). Early effort to teach CSE to lower-division students. (E)

VI. TOOLS, LANGUAGES, ENVIRONMENTS

Everyone seems to agree that visualization is important in understanding physics concepts and understanding the output of computations. Accordingly, we place Visualization first in this section. We note, however, that the modern role of visualization in CP includes includes three scenarios, with the first two probably new to many readers: 1) concurrent visualization, with intermediate processing such as data reduction, feature extraction, statistical aggregates, and analysis; 2) on-the-fly interactive visualization using parallel software techniques or hardware accelerators; 3) post-simulation visualization reading data.

In contrast to visualization, not everyone agrees whether compiled/interpreted languages or *Problem Solving Environments* such as **Maple** and **Mathematica** are best for a CP course. Indeed, disagreement may be even more pronounced when the advocates of a compiled language discuss *which* compiled language should be advocated. In practise, most CP texts adopt a compiled/interpreted language, while most texts that incorporate computation into traditional physics disciplines adopt a problem solving environment. This probably reflects the view that compiled languages show best how an algorithm is implemented and what level of precision is being demanded, while being in a form that encourages exploration; all items of importance in a CP course.

In our view, **Java's** attention to precision, useful error messages, and object-orientation make it good for scientific computing, while its universality, free compilers, and use in the commercial sector, make it popular with students. However, it is not as efficient or as well supported for high-performance computing (HPC) and parallel processing as is Fortran and C, the latter two having highly-developed compilers and many more scientific subroutine libraries available. **Fortran**, in turn, is still the dominant language for HPC, with Fortran 90/95 being a surprisingly nice, modern, and effective language. But alas, it is hardly taught by any CS departments, and compilers can be expensive. **C**, and its object-oriented brother **C++**, are good for HPC, have good free compilers and libraries available, but may be too flexible with precision and memory access for beginners to learn good scientific programming practices. **Python**, a new programming language that has garnered a small, but devoted following, is free, user friendly, good for beginning programming and has a nice 3-D graphics library. However, it is still developing, is not meant for HPC, and does not have any scientific subroutine libraries or parallel versions yet available. In any case, we recommend

that students become familiar with at least *two* compiled languages (preferably one being Fortran), and that they keep their focus on how clearly the algorithm is implemented and not on the minimum number of statements that is needed to do it.

As we use the term **problem solving environment** in this letter, we refer to **Maple** and **Mathematica**, and their less-popular cousins such as **Derive** (now obsolete) and **Macysma/Maxima**. All are capable of symbolic manipulations. In our experience [which includes a 1:1 translation of the Maple version of our “First” book (VB.14) into Mathematica], these environments have much more in common than they do differences, and once you know one, learning the other is just a question of grammar.

Another group of high-level languages that are sometimes also called problem-solving environments, include **Matlab** (especially popular with engineering departments), its free cousin **GNU Octave**, and **Mathcad**. These are essentially command line interfaces for running numerical calculations that employ build-in libraries for linear algebra, mathematical methods, visualization, signal processing, and such. These environments are powerful and work well, but do separate the user more from the algorithms than compiled languages (a separation, daresay, some users may prefer).

For pedagogical purposes in CP classes, we recommend that students use compile and execute commands directly from a shell. However, when projects get complicated, such as with multiple data sets or multiple computers, then an **integrated development environment (IDE)**, also known as a **studio** or a **computational framework**, is often used by the power players. An IDE provides an integrated, tightly-knit set of components that may include: code editor, compiler, a build tool, a version control system, a debugger, a class browser, an object inspector and a class hierarchy diagram. Examples include **Cactus**, **Kdevelop**, **Eclipse**, and **ROOT**.

A. Visualization

1. **Gnuplot** is a classic. It is a free, nearly universal, command-line driven, interactive data and function plotting utility. Although Grace produces higher, publication-quality 2-D figures by default, Gnuplot is powerful and flexible, and the standard for 3-D (surface) plots. Also for Windows, which Grace is not, www.gnuplot.info. (E)
2. **PtPlot**, a 2-D data plotter and histogram tool implemented in Java, part of UCB’s Ptolomy IDE (VIH.8). Highly recommended for graphical output from within Java programs, ptolemy.eecs.berkeley.edu/java/ptplot. (E)
3. **Grace**, a free WYSIWYG 2-D plotting tool for the X Window System (Unix/Linux), or Windows with Cygwin. This descendant of *ACE/gr* and

Xmgr is our choice for publication quality, 2-D graphics, but it cannot create 3-D (surface) plots, plasma-gate.weizmann.ac.il/Grace. (E)

4. **IDL, The Data Visualization & Analysis Platform**, a powerful, but costly commercial off-the-shelf (COTS) package, www.itervis.com/idl.
5. **Open DX**, This open source, Linux/Unix software package is based on IBM’s *Data Explorer* and provides industrial-strength visualization comparable to the COTS (and expensive) *AVS*. Although standard 2-D plotting is possible, the real value of DX is for the next step beyond 3-D surface plots, namely, interactive 3-D rendering and volume visualization (slicing and dicing) of N-D datasets. Uses graphical programs (networks) that resemble flowcharts, and can run under windows with Cygwin, www.opendx.org. (I)
6. **OpenDX: Paths to Visualization**, D. Thompson, J. Braun, R. Ford, (Visualization and Imagery Solutions, Missoula, 2001). (I)
7. **Advanced Visual Systems (AVS5)**, a COTS product with similar properties as OpenDX, but maybe more so, www.avs.com/avs5.html. (I)
8. **Maple and Mathematica** contain outstanding visualization tools for mathematical functions with analytic forms, but we do not find them as convenient as the tools listed here for visualizing large, numerical data sets. See §VIG. (E–I)
9. **NCAR Graphics**, an open source time-tested UNIX package of Fortran and C utilities for drawing contours, maps, vectors, streamlines, weather maps, surfaces, histograms, X-Y plots, annotations, *etc.*, ngwww.ucar.edu. (A)
10. **NCSA Visualization Suite** is free and includes **Easy-Viz** a simple tool designed for educators to visualize both 2-D and 3-D scalar data, and **NCSA Databridge**, an application that organizes data into common file formats for visualization, education.ncsa.uiuc.edu/products/dvs.html.
11. **OpenGL**, a vendor-neutral low-level environment for developing portable, interactive 2-D and 3-D graphics applications. Essentially the assembler language of computer graphics, www.opengl.org. (A)
12. **Visualization ToolKit (VTK)**, an open source C++ class library for 3-D graphics, image processing and visualization, with interpreted interface layers for Tcl/Tk, Java, and Python, public.kitware.com/VTK. (I)
13. **ParaView, Parallel Visualization Application**, open-source multi-platform visualization of large data sets by use of parallel computers, www.paraview.org/New. (A)

14. **VisIt**, a free interactive parallel visualization and graphical analysis tool, developed by DOE labs for viewing scientific data on Unix and PC platforms, www.llnl.gov/visit. (I)
15. **VolView**, a COTS powerful volume visualization tool appropriate for research and medical application, www.kitware.com/products/volview.html. (A)

B. Fortran

1. **Fortran 90/95 for Engineers and Scientists**, 2nd ed., S. J. **Chapman** (McGraw-Hill, New York, 2004). Practical and effective. (E-I)
2. **Fortran 90 and Computational Science**, The Computational Science Education Project, dated, but a high-quality, early exploration into the use of the Web for education, csep1.phy.ornl.gov/pl/pl.html. (E)
3. **Univ. of Liverpool Fortran 90 Course Notes**, A. C. **Marshall**, the place we look for good and full explanations, www.liv.ac.uk/HPC/F90page.html. (I)
Essential Fortran 90 and 95, L. **Meissner** (The Fortran Company, Tucson, 1997). (E-I)
Fortran 95/2003, M. **Metcalf**, J. **Reid** and M. **Cohen** (Oxford, New York, 2004). Clear and straightforward. (E-I)
4. **Fortran 90 Programming**, T. M. R. **Ellis**, I. R. **Phillips** and T. M. **Lahey** (Addison Wesley, Wokingham, 1994). Both Fortran 90 and 77 with academic discussions. (E-I)

C. C, C++ (OOP)

1. **C for Scientists and Engineers**, R. **Johnsonbaugh** and M. **Kalin** (Prentice Hall, Upper Saddle River, 1997). Meant to be used as a text, with programs on Web page, condor.depaul.edu/mkalin/cse. (E-I)
2. **A Book on C**, 4th ed., A. **Kelley** and I. **Pohl** (Addison-Wesley, Upper Saddle River, 1997). A useful tutorial that includes information on converting to Java and C++ from C. Electronic materials available from author, www.soe.ucsc.edu/pohl/abc4.html. (A-I)
3. **The C Programming Language**, 2nd ed., B. **Kernighan** and D. **Ritchie** (Prentice-Hall, Englewood Cliff, 1988). The classical way to learn C as presented by its creators. (E-I)

4. **Accelerated C++**, *Practical Programming by Example*, A. **Koenig** and B. E. **Moo** (Addison-Wesley, Upper Saddle River 2000). Exceptional discussion of practical object oriented programming, with electronic materials online at www.acceleratedcpp.com. (E-I)
5. **Practical C Programming**, 3rd ed., S. **Oualline**, (O'Reilly, 1997, Sebastopol). Also **Practical C++ Programming**, 2nd ed.. Short and to the point. (E-I).
6. **The C++ Programming Language**, 3rd ed., B. **Stroustrup** (Addison-Wesley, Upper Saddle River, 1997). Written by creator of C++, with electronic materials at his site, www.research.att.com/~bs/3rd.html. (I-A)

D. Java

1. **Java for Engineers and Scientists**, G. J. **Bronson** (Thompson, Brooks/Cole, Toronto, 2003). Some readers may appreciate the well-worded explanations. (E-I)
2. **Understanding Object-Oriented Programming with Java**, T. **Budd** (Addison Wesley, Reading, 1998). The place to go for a deep understanding of object-oriented programming with Java. (I)
3. **Teach Yourself Java in 21 Days**, 5th ed., R. **Cadenhead** and L. **Lemay**. A popular tutorial, now updated to Java 6. (Samms/Pearson, Indianapolis, 2007).
4. **Java for Engineers and Scientists**, 2nd ed., S. J. **Chapman** (Pearson/Prentice Hall, Upper Saddle River, 2004). Discusses Java in the language of scientists, and teaches good scientific programming practise. My favorite Java book. (E-I)
5. **Introductory Java for Scientists and Engineers**, R. **Davies** (Addison-Wesley, Harlow, 1999). A good introduction to Java for someone who already knows C or Fortran. (E)
6. **Java, How to Program**, 7th ed., H. M. **Deitel** and P. J. **Deitel** (Prentice Hall, Upper Saddle River, 2007). Some colleagues swear by this book, we find it an overload. (I-A)
7. **Introduction to Programming Using Java, Version 4.0**, D. **Eck** (2002). Although free, a serious and excellent textbook, math.hws.edu/javanotes. (I)
8. **Easy Java Simulations**, F. **Esquembre**, a software tool designed for the creation of discrete computer simulations with Java, www.um.es/fem/Ejs/Ejs_en. (E-I)

9. **Java in a Nutshell, Desktop Quick Reference**, 5th ed., D. Flanagan (O'Reilly, Sebastopol, 2005). Useful and concise, but not for beginners. (I-A)
10. **NMath Product Suite (CenterSpace Software)**, building blocks for various applications callable from any .NET language (one that produces programs that execute within the Microsoft .NET framework, *e.g.*, C#, Visual Basic .NET, J# and C++/CLI), www.centerspace.net. (A)
11. **Sun Java Developer's site**, the official version of Java, java.sun.com. See too the documentation site java.sun.com/docs and tutorials at java.sun.com/developer/onlineTraining. (I-A)

E. Python

1. **Learn to Program Using Python**, A. Gauld (Addison-Wesley, Reading, 2000); also online. Another Web book that got got turned into paper, www.freenetpages.co.uk/hp/alan.gauld. (E-I)
2. **Python/Vpython**, official website for python, www.python.org.
3. **Python First: Introduction to Computing with Python**, A. Radenski. A commercial digital book/course with text, slides, labs and quizzes, studypack.com/comp/course/view.php?id=232. (E)
4. **An Introduction to Python - The Python Tutorial**, G. van Rossum and F. L. Drake Jr. (Network Theory Ltd, Bristol, 2006); also online. The official Python tutorial, www.network-theory.co.uk/python/intro. (E-I)
5. **Vpython**, a free package including Python, an interactive development environment, a 3-D graphics package, and a module for array processing, www.vpython.org.

F. Matlab, Octave & Mathcad

1. **Matlab Programming for Engineers**, 3rd ed., S. J. Chapman (Thompson, Toronto, 2005). MATLAB as a technical programming language. Has password-protected online site for instructors. (E)
2. **Mastering MATLAB 7**, D. C. Hanselman and B. Littlefield (Prentice Hall, Upper Saddle River, 2005). M files on line. (E-I)
3. **Matlab Guide**, 2nd ed. D. J. Higham and N. J. Higham, (SIAM, Philadelphia, 2005). A good reference for all levels, with electronic media on the Web, www.siam.org/books/ot75. (E-I)

4. **Numerical Computing with MATLAB**, C. Moler, (SIAM, Philadelphia, 2004); also available on line, www.mathworks.com. The author is the original creator of MatLab. (E-I)
5. **GNU Octave**, an open source, free high-level language that is mostly compatible with Matlab, and shares many of the same features, www.gnu.org/software/octave/.
6. **Mathcad**, an environment for creating, documenting, and sharing engineering calculations. Like Matlab, has various libraries and packages (packs). www.ptc.com/products/mathcad/mathcad14/promo.htm.
7. **MATLAB Tutorial**, free online tutorial, www.mathworks.com/academia/student_center/tutorials/launchpad.html

G. Maple & Mathematica

1. **Mathematica in Theoretical Physics**, 2nd ed., G. Baumann (Springer-Verlag, Berlin, 2005). (I)
2. **Maxima** computer algebra system is a descendant of **Macsyma** and is now available as opensource, maxima.sourceforge.net.
3. **Essential Maple 7, An Introduction for Scientific Programmers**, R. M. Corless (Springer, New York, 2002). Emphasis on Maple programming. No apparent update to recent Maple. (E-I)
4. **Mathematica for Scientists and Engineers**, R. Gass (Prentice-Hall, Upper Saddle River, 1998). Contains many physics applications. Mathematica 3.0 notebooks on CD; appears out of print. (I)
5. **Introduction to Maple, Third Ed.**, A. Heck (Springer, New York, 2003). My favorite Maple book. Maple 8 in text, Maple 11 worksheets on Web, remote.science.uva.nl/~heck/Maplebook/. (E-I)
6. **Mathematical Methods Using Mathematica, or Students of Physics and Related Fields** H. Hasani, (Springer, New York, 2004). A supplementary text for use in a math methods course. (E)
7. **Maple Documentation, Maplesoft**. Both printed manuals and free electronic version available, www.maplesoft.com/documentation_center.
8. **Maple, A Comprehensive Introduction**, R. Nicolaides, and N. Walkington (Cambridge Univ. Press, Cambridge, 1996). Only Maple V, but a nice introduction. (E)
9. **Mathematica Navigator**, H. Ruskeepaa (Elsevier, Oxford, 2004). An 844 page complete treatment. (E-I)
10. **Scientific Computing with Mathematica**, H. Ruskeepaa, (Birkhäuser, Boston, 2001).

11. **Physics with Maple**, F. Y. Wang (Wiley-VCH, Berlin, 2006). Emphasis on the physics more than Maple. (I)
12. **The Mathematica Book**, 5th ed. S. Wolfram, (Wolfram Media, Champaign, 2003). At 1488 pages this should be complete. (E-I)
13. **Mathematica for Physicists**, 2nd ed., R. L. Zimmerman and F. I. Olness, (Addison-Wesley, San Francisco, 2003). For physicists, with many applications. Mathematica 4 and 5 notebooks on Web, darkwing.uoregon.edu/~phys600. (I-A)

H. Development Environments, Debuggers

An hour to design, a day to write, and a week to debug. Debugging complicated scientific programs can be the most time-consuming part of computational physics. Good programming practices and systematic code development can help, as well as these tools.

1. **Cactus Problem Solving Environment**, an open development platform, used over many years by a large international collaboration of physicists and computational scientists, www.cactuscode.org.
2. **Condor**, a project aimed at supporting high throughput computing on large collections of distributed computing resources, www.cs.wisc.edu/condor.
3. **Eclipse**, an open source development platform runs in Windows, MacOS or Linux, www.eclipse.org.
4. **GNU Debugger (gdb)**, see what is going on inside a program while it executes or crashes, sourceware.org/gdb.
5. **GuardSoft Relative Debugger**, new debugging technology of *relative debugging*, guardsoft.com.
6. **KDevelop**, the development environment for KDE, a powerful free graphical desktop environment for Linux and Unix workstations, www.kdevelop.org.
7. **TotalView Debugger**, a debugger for the multi-core age, www.totalviewtech.com.
8. **Ptolomy**, a Java-based software framework with a graphical user interface used to assemble and control the interaction of concurrent simulation components. Contains the excellent Java plotting program, **PtPlot**, ptolemy.berkeley.edu/java/ptplot. (I)
9. **ROOT**, an object oriented framework for large scale data analysis. ROOT is a C++ replacement of the popular PAW program developed at CERN, root.cern.ch.

10. **Your Printer**, probably the most common debugging tool; include write statements as you write the program and then comment them out for production.

I. Markup Languages

A markup language is used to write the source code for a document that then gets compiled into presentation form. The communication and publishing of scientific documents, as well as their storage in and retrieval from digital libraries often makes use of markup languages. For example, most Web documents use the **Hyper Text Markup Language (HTML)**, and if you push the menu buttons in your browser to *View/Page Source*, then you will see the “marks” (commands) interspersed with the text.

1. **Docbook**, *The Definitive Guide*, N. Walsh and L. Muellner (O’Reilly, Sebastopol, 2003). The official documentation for the DocBook document type definition, available online at www.docbook.org/tdg/en/html/docbook.html.
2. **Hermes**, a semantic XML, MathML, Unicode e-publishing tool for \LaTeX authored scientific articles, hermes.roua.org.
3. **LaTeX: A Document Preparation System**, 2nd ed., L. Lamport (Addison-Wesley, Upper Saddle River, 1994). My first and favorite for simplicity and conciseness. (E)
4. **LaTeXML: A LaTeX to XML Converter**, dlmf.nist.gov/LaTeXML/LaTeXML.xhtml.
5. **LaTeX Companion**, 2nd ed., F. Mittelbach, M. Goossens, J. Braams, D. Carlisle and C. Rowley (Addison-Wesley, Upper Saddle River, 2004). (E)
6. **Guide to \LaTeX** , 4th ed., H. Kopka and P. W. Daly (Addison-Wesley, Upper Saddle River, 2004). Not as friendly as Lamport, but has most everything needed. (E-I)
7. **MathML**, a product of the W3C Math working group, provides a low-level foundation for the inclusion of mathematical expressions in digital documents, www.w3.org/Math/, with white paper at www.w3.org/Math/XSL.
8. **Future Scientific Digital Documents with MathML, XML, and SVG**, R. Landau, D. Vediner, P. Wattanakasiwich, and K. Kyle, *Computing in Sci. & Engr.* **4**, 77–86 (2002). (E)
9. **Vector Markup Language (VML)**, an XML application defining format for encoding vector information, www.w3.org/TR/NOTE-VML.html.

10. **W3C Home**, the home of the Work Wide Web Consortium (WWW = “the Web”), www.w3.org.
11. **W3C Math Home**, the home of the W3C’s views on how Math should be incorporated into Web documents, www.w3.org/Math.
12. **XML, Extensible Markup Language**, a Web markup language that separates contents of text from presentation, www.w3.org/XML/.
13. **XHTML, The Extensible HyperText Markup Language**, a reformulation of HTML 4 in XML 1.0, www.w3.org/TR/xhtml1.
14. **XML FAQ**, P. Flynn, frequently-asked questions about the extensible markup language, www.ucc.ie/xml.

J. SQL: Database Languages

Present and future computation is focusing more on data-intensive computing with distributed databases, which means that we should know something about dealing with databases. A database management system is a collection of programs for storing, modifying and extracting information from a database. The standard query language is Structured Query Language, SQL.

1. **Database Management Systems**, 3rd ed., R. Ramakrishnan and J. Gehrke (McGraw-Hill, New York, 2003), pages.cs.wisc.edu/~dbbook, a popular textbook (E–I).
2. **A First Course in Database Systems**, 3rd ed., J. Ullman and J. Widom (Prentice-Hall, Upper Saddle River, 2007). (E)
3. **SQL.org**, SQL information, including MySQL, PostgreSQL, Oracle, mSQL and Microsoft SQL, www.sql.org.

K. Concept Mapping

Fig. 2 shows a map of the abstract concept space containing the CP knowledge base, with the concepts as nodes and with arrows indicating links among the concepts. When the concepts are connected to chunks of knowledge, a content map is formed. Such maps are created with the tools below:

1. **VUE, The Visual Understanding Environment**, Tufts Univ. Academic Technology, vue.uit.tufts.edu.
2. **AAAS Strands Maps**, traces paths followed by learners, www.project2061.org/publications/rs1/online/GUIDE/CH2/RSTRAND0.PDF

3. **CITI Computing and Information Technology Interactive Digital Educational Library**, visualizes contents of digital library, www.citidel.org.

VII. PARALLEL COMPUTING

Parallel computing is the dominant form of high performance computing (HPC). Even now, multicore CPUs are appearing on desktops and laptops, while computational scientists are planning for the use of 100’s of 1000’s of CPUs with closely integrated communication and arithmetic units. Nevertheless, parallel computing is hard to implement for many scientific problems and now requires lower-level coding than used for serial programs. Most parallel programs utilize distributed memory computers with a message-passing interface known as MPI. A bright spot for the future of parallel computing is *Partitioned Global Address Space (PGAS) Languages*, UPC for C and CAF for FORTRAN, and OpenMP (Open Multi-Processing), which should place less of a burden on the programmer.

1. **Argo Beowulf Cluster: MPI Commands and Examples**, Academic Computing & Communications Center, Univ. of Illinois at Chicago, www.uic.edu/depts/acc/hardware/argo/mpl_routines.html. (I–A)
2. **Parallel Computing Works!**, G. Fox, R. Williams and P. Messina (Morgan Kaufmann, San Diego, 1994). (A)
3. **Web Pages for MPI and MPE**, Mathematics and Computer Science Division, Argonne National Laboratory, www-unix.mcs.anl.gov/mpl/www. (I–A)
4. **Parallel Programming in C with MPI and OpenMP**, M. J. Quinn (McGraw Hill, New York, 2004). The CS theory behind parallel programming, yet practical with many scientific examples. MPI codes in text. (I)
5. **How to Build a Beowulf**, T. Sterling, J. Salmon, D. Becker, and D. Savarese (MIT Press, Cambridge, 1999). (I)
6. **The OpenMP Application Program Interface**, a specification for parallel programming, www.openmp.org.
7. **Parallel Programming with OpenMP**, Ohio Supercomputing Center, Science & Technology Support Group, www.osc.edu/supercomputing/training/openmp/big/fs1d.001.html
8. **Partitioned Global Address Space (PGAS) Languages**, *UPC, CAF OpenMP, UPC, CAF*.
9. **Parallel Programming with MPI**, P. S. Pacheco, (1997), Morgan Kaufmann, San Diego.

10. **PVM: Parallel Virtual Machine A User's Guide and Tutorial for Networked Parallel Computing**, A. Geist, A. A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam (1994), Oak Ridge National Laboratory, Oak Ridge.

VIII. DIGITAL LIBRARIES

A major change in computing since the previous CP Resource Letter is the increased importance of networks, distributed data sets, CPU architectures designed for numerically intensive computing, and parallel processing. The high speed networks permit us to get quite easily to distributed data sets and digital libraries, while the sophisticated architectures make the use of high-performance subroutine libraries crucial.

A. Subroutine Libraries

Although physicists often like to write their own software, we recommend the use good software libraries as being faster and more reliable than the simple algorithms found in textbooks. These libraries typically contain the BLAS (Basic Linear Algebra Subprograms), LAPACK (Linear Algebra Package), random number generators, Fourier transforms, convolutions, correlation functions, sorting, searching, interpolation, and quadrature. Note, many of these libraries contain versions for specific architectures and parallel computers, for example, SCALAPACK for the latter. Note, there are also extensive software libraries on specialized subjects such as Finite Elements, Finite Differences, Signal Processing, Fourier Transforms, Wavelet Transforms, Integral Equations, and many other of the subjects you may find in a CP textbook; we suggest that you search the Web for what you need.

1. **CMLIB**, a free library of some 750 high-quality Fortran 77 subroutines for numerical computation. Maintained by the National Institute of Standards and Technology, who seems to have serviced it last in 1988, gams.nist.gov/serve.cgi/Package/CMLIB. (I)
2. **Computer Physics Communications Program Library**, started in 1969 as a pioneer in establishing the importance of programs in the development of CP. It now contains more than 2000 programs, with papers describing the programs published in the *Computer Physics Communications Journal*, cpc.cs.qub.ac.uk/cpc. (A)
3. **FFTW, Fastest Fourier Transform in the West**, a C subroutine library for computing the discrete Fourier transform (DFT) in one or more dimensions for both real and complex data, www.fftw.org. (A)

4. **NIST Guide to Available Math Software (GAMS)**, a cross-index and virtual repository of mathematical and statistical software components for use in computational science and engineering, gams.nist.gov. (I)
5. **GMP, the GNU Multiple Precision Arithmetic Library**, a free library for arbitrary precision arithmetic, gmplib.org. (I)
6. **Gnu Project**, a leader in providing free software, including subroutine libraries and computational tools for Unix/Linux. The Gnu scientific library (GSL) for C and C++ is free, extensive, and open source, www.gnu.org/software/gsl; manuals at www.gnu.org/manual/manual.html. (E)
7. **IBM Engineering and Scientific Software Library (ESSL)** a complete, yet proprietary library provided by IBM that is optimized for IBM computers. Parallel ESSL (PESSL) supports 32-bit and 64-bit Fortran, C and C++ distributed-memory parallel-processing subroutines for AIX and Linux. IBM's **MASS (Mathematical Acceleration SubSystem)** provides scalar and vector libraries of tuned mathematical intrinsic functions, www-03.ibm.com/systems/p/software/essl.html. (I)
8. **IMSL**, a commercial set of high-quality and high performance mathematical and statistical functions written in C, C++, Java, and Fortran, www.vni.com/products/ims1. (A)
9. **Jama, A Java matrix package**, part of the Java Numerics project at the National Institute of Science and Technology, is a matrix library for Java (see too IMSL and NAG), math.nist.gov/javanumerics/jama; math.nist.gov/javanumerics. (I)
10. **Java Matrix Package, Jampack** A sibling matrix package to Jama, developed at NIST and the Univ. of Maryland, ftp://math.nist.gov/pub/Jampack/Jampack/AboutJampack.html. (I)
11. **LAPACK, Linear Algebra Package**, an efficient, portable free library of Fortran 77 subroutines for common problems in linear algebra; a successor to LINPACK and EISPACK, www.netlib.org/lapack. (I)
12. **LAPACK Users' Guide**, E. Anderson, , Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen 3rd ed., (SIAM, Philadelphia, 2000). (I)
13. **Netlib** contains a variety of free libraries, including ones for vector and matrix algebra, sparse matrix algebra, fast Fourier transforms, and the famous BLAS and LAPACK, www.netlib.org. (I)

14. **Numerical Algorithms Group (NAG)**, originally from UK labs, now a commercial set of high performance mathematical and statistical functions for C, Excel, Java, Labview, Maple, MATLAB, R, and Visual Basic, www.nag.co.uk; similar package, **Aspen HSL (formerly Harwell Subroutine Library)**, www.aspentech.com/HSL. (A)
 15. **Open Source Physics**, a collection of curricular material for physics computation and physics education at all levels, www.opensourcephysics.org. (I)
 16. **Open Source Physics User's Guide with Examples**, W. Christian (Addison-Wesley/Pearson, Boston, 2006).
 17. **PORT (Portable, Outstanding, Reliable, and Tested) Mathematical Subroutine Library**, a free library of a wide range of Fortran 77 routines. Originated from Bell Labs research, www.bell-labs.com/project/PORT. (I)
 18. **ScaLAPACK (Scalable LAPACK)**. Get program running on single processor with LAPACK, and then switch to ScaLAPACK for parallel processing, netlib2.cs.utk.edu/scalapack. (A)
6. **National Science Digital Library (NSDL)**, nsdl.org. The nation's online library for education and research in science, technology, engineering and mathematics. CP content still in need of growth and organization. Direct use of NSDL is less effective than using its pathways:
 - (a) **CSERD, Computational Science Pathway**, provided by Shodor Education Foundation, www.shodor.org/refdesk. (E)
 - (b) **ComPADRE Physics and Astronomy Pathway**, a collaboration of AAPT, APS, AIP/SPS and AAS, www.compadre.org/portal. (E)
 - (c) **Materials Science Pathway**, provided by MatDL at Kent State Univ.. Contains simulations tools of interest, matdl.org/repository. (A)
 7. **The Association for Computing Machinery Digital Library**, contains journals, magazines, transactions, proceedings, newsletters, publications by affiliated organizations, special interest groups, and ACM oral history interviews, portal.acm.org/dl.cfm.

B. General Digital Library

1. **American Physical Society Journals**, just journals, but for more see Compadre portal (6b), publish.aps.org.
2. **Project Gutenberg**, first producer of free electronic books (ebooks), www.gutenberg.org/wiki/Main_Page.
3. **IEEE Computer Society Digital Library**, access to 25 society magazines and transactions, and more than 1,700 selected conference proceedings, www.computer.org/portal/site/csdl/index.jsp.
4. **lanl.arXiv.org**, open access to e-prints in Physics, Mathematics, Computer Science, Quantitative Biology and Statistics. CP at xxx.lanl.gov/list/physics.comp-ph/recent.
5. **Merlot**, Multimedia Educational Resource for Online Teaching, www.merlot.org/merlot.

C. Digital Sky Libraries

1. **AstroGrid**, a UK government funded, open source, working virtual observatory. www2.astrogrid.org. (I)
2. **Digitized Sky Survey** contains set of all-sky photographic surveys conducted with the Palomar and UK Schmidt telescopes, archive.stsci.edu/dss. (A)
3. **Sloan Digital Sky Survey**, to eventually include detailed optical images covering more than a quarter of the sky, and a three-dimensional map of about a million galaxies and quasars, www.sdss.org. (I)
4. **US National Virtual Observatory**, us-vo.org, provides access to data and computing resources. (A)