VISUALIZATION

Andrew N. Hrymak and Patricia Monger
McMaster University
Hamilton, Ontario L8S 4L7 Canada

Abstract

Visualization is the use of computer graphics for scientific and engineering data analysis. This chapter highlights the current directions for scientific visualization software on personal computer and workstation platforms.

Introduction

Scientific visualization does not require a computer, as demonstrated by many examples in Edward Tufte's milestone book, *The Visual Display of Quantitative Information*. However, since the late 1980s, the term has come to be associated with the use of computer graphics to represent relationships in scientific data, especially for systems with very large quantities of information.

The use of computers as visualization tools began with techniques such as contour plotting, line graphics, and 2-D image display. Two factors combined to give this field much more prominence: an increase in the amount of data (due to computer simulation, image capture, etc.) which has overwhelmed the user's ability to process the information, and technological advances in computers that have made much more sophisticated data representation possible and affordable (Schmitz, 1994).

Early scientific graphics applications used *de facto* standard line graphics software libraries, almost all written in Fortran. The most widespread of these was the PLOT subroutine library written by Calcomp to drive their pen plotters. This same library was then adapted to support other hardcopy devices, e.g. electrostatic plotters, dot matrix printers, and graphics terminal devices. The first graphics terminal to come into widespread use was the Tektronix 4010. Almost all subsequent line graphics display systems supported Tektronix protocols, and even on modern systems, e.g. X window terminals, one finds client software that accepts Tektronix graphical escape sequences.

2-D data were visualized using contour or vector field representations, which relied on the line graphics subroutine libraries. Image display routines were tailored to the specific display hardware, so that use of this technique required writing interfaces to the graphics screen or hardcopy device drivers.

As recently as 15 years ago, the use of color hardcopy in data analysis was rare, and tech-

niques such as animation virtually unknown. This began to change in the mid 1980s, when researchers gained access to higher performance computers and supercomputers, with concurrent improvements in data gathering instrumentation. This presented the researchers with the challenge of finding means to interpret these huge quantities of data. This challenge was brought to public attention with the 1987 report by the National Science Foundation, entitled *Visualization in Scientific Computing*. At about the same time, computer hardware supporting megapixel bitmapped displays in 8-bit color became widely available. Higher-end graphics workstations with specialized graphics hardware, while still largely out of reach of individual researchers, began to appear as departmental or group resources.

Software development accelerated with the advent of graphics standards and vendor-supplied libraries for simplifying the use of more sophisticated graphics techniques. The early standardization efforts developed on three fronts: attempts to define a standard plot file format that hardware vendors could then support for their display technologies, graphics systems standards to provide an interface layer between the hardware drivers and the application code at the level of graphics "primitives," and higher-level libraries that use the primitives and produce standard plot files. CGM, the computer graphics metafile, is the standard file format surviving from these efforts. GKS, Graphical Kernel System, remained as the 2-D graphics primitives system. No clear winner emerged from the attempts to standardize the higher level libraries, but examples of systems that developed from these efforts are NCAR, DISSPLA, and PLOT10.

As graphics technology developed, the need for standardization became more apparent. The development of laser printers and color hardcopy devices paralleled the spread of PostScript. The advent of bitmapped terminals brought the X Windowing System, and as 3-D graphics became more feasible, the standardization impetus that created GKS led to the effort to define PHIGS. At the same time, developers of the 2-D high level libraries extended these systems to provide support for 3-D data representation (e.g. hidden or ruled-surface plots, 3-D line plots), as well as animation.

The acceptance of standardized systems in 3-D display was hampered by the fact that the vendors provided their own libraries that were in general more efficient than, and at least as easy to use as, the standard libraries. For higher end visualization applications, the fact that only a very small number of vendors had hardware to support those applications meant that portability was not a driving consideration. The dramatic drop in the cost of the hardware also made software developers (especially those producing freeware) less hesitant to target the software to a particular hardware platform. This is less of a problem than appears at first glance, because of the trend for hardware vendors to license their software for other platforms, and because of the standardization of windowing systems on workstations. Most modern visualization systems ultimately rely on X window protocols, which are portable.

The next set of improvements in visualization built upon the development of windowing systems and graphical user interfaces to create data exploration systems that attempt to simplify the visualization process. These systems are founded on two ideas; that the focus of the visualization process should be on the flow of data through the stages of the process, and that the visualization program itself can be constructed using a visual programming model. The researcher selects the appropriate visualization modules, drags the module icons onto a workspace area, selects the data inputs, links the modules together in the appropriate order, accesses

the data, and waits for the process to display. An example of a visual programming system, Explorer from Silicon Graphics Inc., is shown in Figure 1.

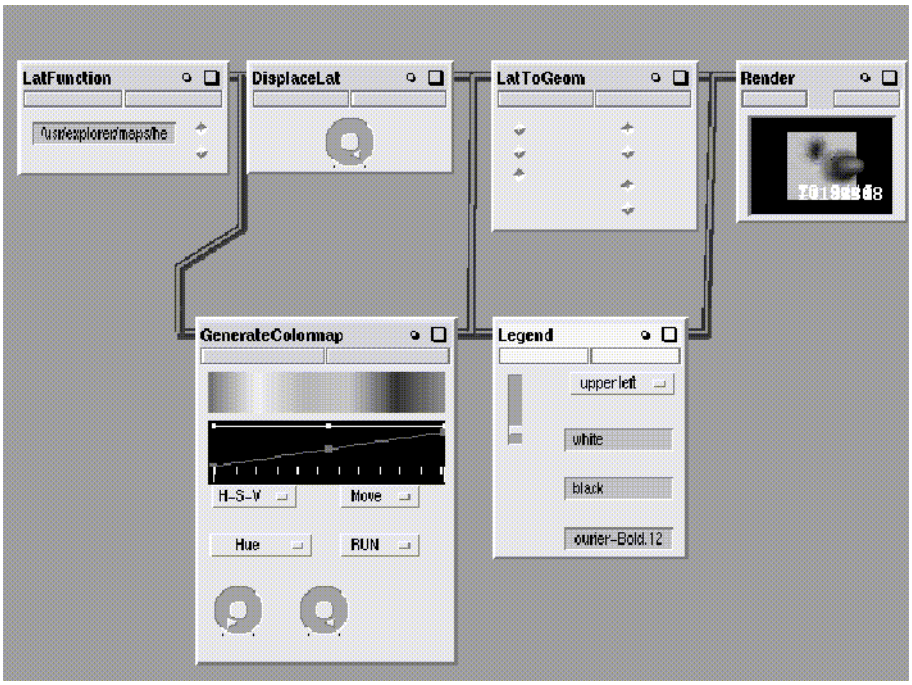

Figure 1. Screen image building blocks from the Explorer package (Silicon Graphics Inc.).

Of course, more traditional scripting languages for visualization also exist, e.g. PV-Wave (Precision Visuals, Inc.). Other visualization packages are customized for the needs of a particular class of problems, for example, Tecplot™ (Amtec Engineering, Inc.) or Sterling SSV™ (Sterling Federal Systems, Inc.) which have capabilities for visualizing data from finite difference and finite element calculations in computational fluid dynamics. A key feature within advanced visualization packages are cutting tools which allow the user to cut sections away from the object, model or graph to see inside the outer surface. A variant on arbitrary cutting planes, are isosurface plots of variables such as temperature, streamlines or stress. For an interesting example of visualization of a complex flow see Göde and Cuénod (1990). Some packages, such as AVS (Advanced Visual Systems), Explorer (Silicon Graphics) and the Visualization Data Explorer (IBM), are general purpose packages which provide visualization "environments" to allow animation and visualization of machinery, as well as derived data from computer simulations. The visualization software packages cited above are not an exhaustive list and are given as examples from classes of visualization tools. Interested readers should consult hardware and software vendors for more information.

Most recently, developers have been experimenting with ways to display data of more than 3 dimensions, using non-visual cues such as sound, and more flexible and intuitive ways

of interacting with the data, e.g. with stereo viewers or virtual reality systems wherein the researcher uses a head-mounted display or other motion-sensitive detector to "move" the data. The goal of visualization software development is to integrate it with the data generating process, so that the researcher can explore the data interactively and use that information to modify the numerical computations.

Script language visualization systems increasingly incorporate numerical functions. Similarly, numerical and symbolic manipulation programs have increasingly sophisticated graphics capabilities.

Graphics systems on personal computers have not traditionally been tailored to scientific visualization, because these machines did not have the CPU power for complex calculations or complex graphics. Numerical and statistical systems with 3D graphics support, as well as image display and animation programs now run on personal microcomputer platforms. A number of the more specialized visualization packages for the workstation platform have versions, sometimes for limited problem classes, for personal computers.

The term scientific visualization has evolved to refer exclusively to more complex scientific graphics: animation and/or exploration of data in 3 or more dimensions. The discussion of visualization in this chapter is therefore also focussed on higher-end visualization requirements.

The Visualization Process

With all the impressive increase in hardware and software capabilities, it must be acknowledged that the techniques of data visualization have become more difficult for the user. In the past, most researchers wrote their own programs to call the subroutine libraries for line graphics. Putting together a program for interactive manipulation of 3-D fluid flows is quite another matter! Furthermore, software developers also find it difficult to create a generic program that will do this kind of visualization for arbitrary data types, and of course programs tailored to the users' specific data requirements are costly. The result is that the user may be faced a vast array of generic tools for gridding, color map manipulation, geometrical mapping, etc., but no clear idea of how the data fits into this toolset.

Two things aid the user in getting started with visualization. One is that the most common visualization tasks have been encapsulated into modules, for example: 3D plotting and animation in a projection calculation module, accumulation of single frames with playback into a movie module, or object rotation and zooming within a rendering module. The second helpful development is the explosion of freeware, user contributed modules, and user newsgroups for all visualization systems.

The visualization process is well represented by the dataflow programming model. The result at each stage depends strictly on the data input, not on the results of a previous execution, so that the process can be envisioned as data traveling in a pipeline through various processing steps.

The basic steps are:

- Normalization: Scale the data to the values expected by the display sys-

tem (e.g. 0-255 for 8 bit display devices). This step also includes any scaling required to better analyze trends in the data (e.g. converting to logarithmic values, filtering).

- Mapping: Define the display image according to the normalized data and the computational grid - regrid from computational space to display space.
- Rendering: Add color, lighting, shading, projection information, display the image and store it in display memory.
- Interaction and/or animation using the rendered images.

The rendering step is often very computationally intensive, involving calculating projection information, calculating and removing occluded surfaces, and calculating the effect of light sources on the object's appearance. Despite the complexity of this step, it is of less concern for the user because it does not involve the scientific data, but rather the computer graphics image. Most visualization systems provide modules to handle rendering as a "black box" from the user's standpoint.

The mapping step is the point at which the computer graphics is added to portray the scientific data. The user will find this the most difficult step, because it requires an understanding of both the science and the graphics. First, one must decide what graphical representation best conveys the information about the data. For example, is a 3-D velocity field best rendered as a stream of particles in a 3-D coordinate space, as vectors on a plane, or as streamlines? What color scheme is best to display data values? Once these issues have been decided, the user must determine how to use the visualization system to convert the data into the desired display representation. The 3-D velocity field, for example, may not be defined over a regular 3-D matrix, but regridding will be required in order that the data be converted to graphics primitives for the renderer. Consider the set of mapping steps for the visualization of the energy conservation equation for a heat transfer problem. The heat flow is evaluated over a regular grid, and then output as a one-dimensional array, or lattice. This is then converted to a geometrical representation as expected by the renderer. A colormap is also attached when the geometrical representation is generated, as is a text legend for the color bar.

A crucial requirement in the visualization process is appropriate hardware. The rendering and mapping steps can both require significant CPU power and memory. Animation is very demanding of disk space. Higher end graphics workstations, supporting display and interaction of 3D solids, are distinguished from general purpose workstations by having options such as 24 or more bitplanes (memory specific to the storage of display frames), extra RAM for storing distance information (often called a z-buffer), and specialized ASICs for encoding graphical primitive instructions in hardware. Output of color images or animation will also require the appropriate hardware to interface to the computer (color printers, video output, CD-ROM, etc.).

Visualization Resources

In compiling a list of useful guides in visualization, one must cite in particular the US National Center for Supercomputer Applications. Modules for visualization software systems on PCs, Macintoshes, and workstations are all available here, as are programs developed in-house

and made available to the research community. More information as well as the tools themselves are available by anonymous ftp from *ftp.ncsa.uiuc.edu*.

Usenet groups for computer graphics, visualization, and specific visualization software also exist. Users of Mosaic or Netscape and the World Wide Web can make use of the CUI World Wide Web Catalog. A keyword search of the catalog for "visualization" yields a number of entries. In particular the user should consult the NASA Ames annotation bibliography of scientific visualization URLs for visualization examples and references to freeware.

The usenet group *comp.graphics* publishes a biweekly resource listing for computer graphics systems that includes visualization software. This resource listing is also available at usenet archive sites, for example *rtfm.mit.edu*, and includes information on public domain and commercial visualization software for all hardware platforms.

Finally, as more of the technical tasks in visualization get absorbed into automated procedures, what the researcher will really need to know is how to make an effective visual presentation; e.g. what colors to use, how much information can be conveyed without cluttering the visualization, etc. For insight into that process, we would strongly recommend the books by Edward Tufte mentioned previously, *The Visual Display of Quantitative Information*, and the second book, *Envisioning Information*. These books make extensive use of examples of good and bad visualizations, which, though static, provide valuable lessons that can be extended to animations as well.

The collection of papers edited by Cunningham and Hubbold (1992) provides an interesting cross-section of experiences in education with various visualization tools. While there are many exciting possibilities with multimedia and even hypermedia, certain common problems recur in the use of visualization in education. The general availability of reasonable hardware with adequate graphics performance cannot be taken for granted, especially in developing countries. More importantly, the plethora of specialized graphics platforms means that the learning module must be portable for any kind of extended life in the fast changing world of computer hardware. The time needed to prepare an instructional module is considerable and there is a lack of authoring tools that are flexible and allow portability of the module. The instructional module must be accessible by students who are at different levels of ability with the scientific subject at hand so that there is help for the novice and depth for the student who wishes to try more difficult cases. The use of visualization in education is definitely limited in the traditional lecture mode, but is particularly well suited for self-study and enrichment exercises.

An Example of Visualization Techniques - Flow in a Driven Cavity

As an illustration of a simple method of getting useful graphical information for a problem of interest in chemical engineering, we take the program *cavpit*, part of the MINPACK-2 test problem collection from the Army High Performance Computing Research Center and Argonne National Laboratory (Brett et al., 1992). Program *cavpit* solves the steady-state Navier-Stokes equation for the 2-D driven cavity flow problem. The modified program uses MATLAB to display the streamline contours, velocity contours, and velocity vectors at several values of the Reynolds number (Kus, 1994).

The program makes use of the MATLAB sockets toolkit written by Barry Smith. The tool-

kit uses Internet protocols to establish socket-based communication between an application program and a MATLAB session. The toolkit provides an object library of routines for passing a matrix (the streamline values) to MATLAB, and a MEX function that waits for the matrix and then returns it to the MATLAB macro. The macro then calculates the vorticity contours and the velocity vectors from the streamlines, and uses the MATLAB graphics functions to plot these quantities. The macro for calculating and plotting the data is given in Figure 2. Plots of vorticity, streamlines and velocity vectors for Reynolds number 400 are shown in Figure 3 (a - c).

```
function cav(portnumber)
%
%  Waits at socket portnumber 5005 until a matrix arrives
%
%  This file is for the cavity program
%
%

if ( nargin == 0 ) portnumber = 5001; end;
for ii=0:10000
  m = receive(portnumber);
  if ( sum(size(m)) ~= 0 )
    nvar = size(m,1)
    n = nvar-1
    nx = sqrt(n)
    ny = n/nx
    for j = 1: nx
      for i = 1:ny
        psi(i+1,j+1) = m((i-1)*nx+j,1) ;
      end
    end
    for j = 1:nx+2
      psi(1,j) = 0.0 ;
      psi(ny+2,j) = 0.0 ;
    end
    for i = 1:ny+2
      psi(i,1) = 0.0 ;
      psi(i,nx+2) = 0.0 ;
    end

    xc = m(nvar);
    tc = num2str(xc)

%     Streamlines plots

    V = [ -.11 -.1 -.08 -.06 -.04 -.02 -.01 ...
        -1.0e-5 1.0e-6 1.0e-5 1.0e-4 1.0e-3 2.0e-3 ] ;

    figure(1);
```

```
   contour(psi,V);
   title([ 'Streamline Contours for Flow in a Driven Cavity for Reynolds Number R  =  ',tc]);

%     Equivorticity plots

   hx = 1/(nx+1); hy = 1/(ny+1);
   for j = 1:nx-1
    for i = 1:ny-1
      v(i+1,j+1) = -(psi(i+2,j+1)-2*psi(i+1,j+1)+psi(i,j+1))/(hx*hx) ...
                -(psi(i+1,j+2)-2*psi(i+1,j+1)+psi(i+1,j))/(hy*hy) ;
    end
   end
   V = [ -5.0 -4.0 -3.0 -2.0 -1.0 0.0 1.0 2.0 3.0 4.0 5.0 ] ;

   figure(2);
   contour(v,V);
   title([ 'Vorticity Contours for Flow in a Driven Cavity for Reynolds Number R  =  ',tc]);
   for j = 1:nx
    x(j) = j*hx;
    y(j) = j*hy;
    for i = 1:ny
      vx(i,j) = (psi(i,j+1)-psi(i,j))/hy ;
      vy(i,j) = -(psi(i+1,j)-psi(i,j))/hx ;
    end
   end
%
%     transpose the x and y matrices to take account of matlab's coordinate
%     system conventions for the quiver function
    vx = vx';
    vy = vy';

    figure(3);
    quiver(x,y,vx,vy,4);
    title([ 'Velocity Vector for Flow in a Driven Cavity for Reynolds Number R  =  ',tc]);
  else
   break;
  end;
end;
```

Figure 2. Matlab code for generating graphics for driven cavity problem.

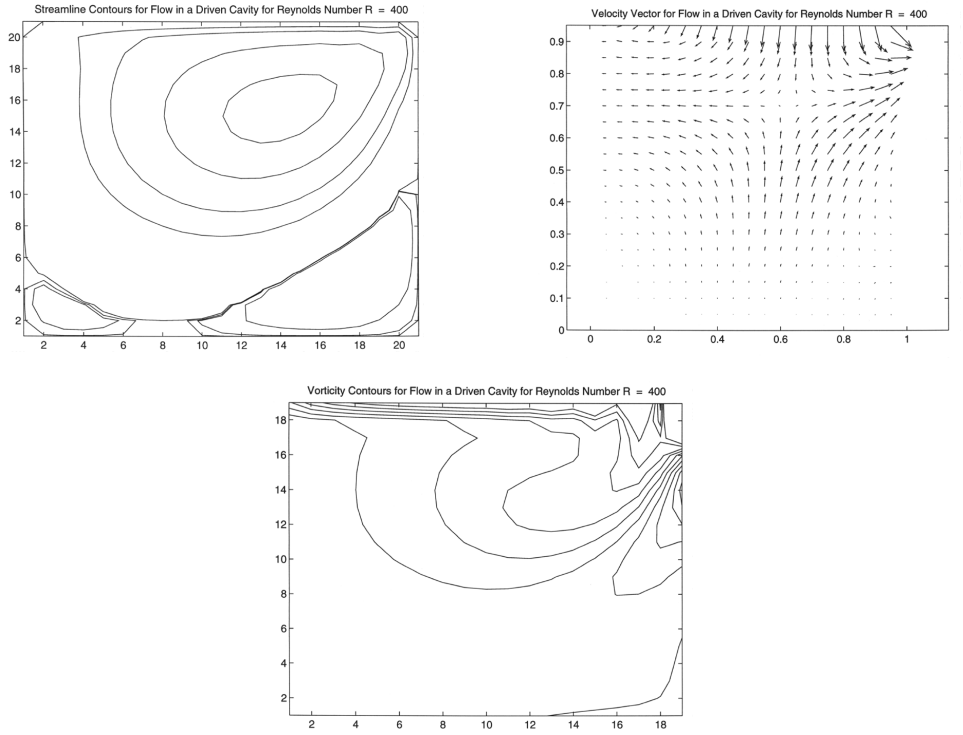

Figure 3. Images of streamlines, velocities and vorticity for driven cavity problem.

MATLAB is then an example of a numerical computation program that has incorporated graphical capabilities into its language. The actual plotting is done with a few simple commands; *contour* for the streamline and vorticity contours, and *quiver* to generate the velocity vector plots, plus commands for labeling the axes and putting titles on the graphs. Simple 3-D plots are also done by invoking only a few commands; for regridding the domain over which the function is defined if needed, and for display the data or function as a scatter plot, wire frame plot, or surface plot.

CACHE Activities

Elements of visualization are seen in a number of CACHE Corp. products for use in undergraduate education. An effective use of visualization is seen in applications where there is a large amount of data generated at one time or over a number of simulations which must be compared. The level of interaction between the user and the software is important to effectively use graphics to aid the student in understanding a concept.

Interactive computer modules for chemical engineering instruction by Fogler and co-

workers (1993) is an example of interactive use of graphics to aid the student to explore the use of simulation models. Collections of modules exist for material and energy balances, fluids and transport, separations and reaction kinetics.

Visualization and graphics can also be used to provide a simulation of an industrial process experiment. Students can work with realistic models of existing industrial processes to develop strategies for operation and optimization through the use of dynamic simulation. The Purdue-Industry Simulation Modules (Squires et al.,1991) provide modules, developed through the support and collaboration of industrial sponsors, for hydrodesulfurization, catalytic reforming, emulsion polymerization, methyl acetate production and process heat transfer.

The Chemical Reactor Design Tool (see p. 121) and the Transport Module (see p. 129) both make extensive use of graphics. The data entry is done in X-windows, which provides a screen-oriented input. The power of these educational tools, however, is in the graphical output. Indeed, the programs are designed so that the graphs are the only output the user studies; a text file is created only for trouble shooting and special purposes. Line plots, 2D contour plots, and 3D perspective views can be chosen for the solution variables as well as derivatives, such as the difference between the catalyst temperature and fluid temperature, the diffusion term, or in the case of the Navier-Stokes equation, the vorticity.

Thermodynamic Visualization

West (1992) argues that a visual thinking approach to problem solving is particularly useful in fields where large amounts of information have underlying physical relationships. An example of such a discipline is thermodynamics. Professor Kenneth Jolls and co-workers at Iowa State University have developed visualization tools tailored to the needs of thermodynamics problems, especially those involving equations of state. Thermodynamic concepts are difficult for the average student to assimilate on the first pass and learning is made more difficult with the complex mathematical formulae which are necessary to describe vapor-liquid or reaction equilibria. Jolls et al. (1991) effectively show how thermodynamic information can be captured in graphical images in a package called *Equations of State*, which is available and used by a number of universities. Modern computing environments allow one to tackle more complex multicomponent equilibrium problems using color and shading techniques to capture added dimensionality which is lost in wireframe drawings. As the number of species goes beyond three, a decision must be made on what specific features need to be rendered since it becomes counter-productive to superimpose too much information on a diagram. Binary and ternary systems, common to problems in the undergraduate thermodynamics course, are especially amenable to visualization.

Thermodynamic visualization can be used as a tool for studying more advanced phenomena such as phase stability (Jolls and Butterbaugh, 1992) or process unit modeling (Jolls et al., 1994).

Conclusions

Visualization using computer graphics systems has undergone a remarkable change in the last thirty years. The student can use a number of packages which have many graphics modules already included to facilitate the analysis of data. Advanced visualization packages allow the use of more complex data representations on both personal computer and workstation platforms. A major need, for educational uses, remains authoring systems that provide flexible use of graphics while not burying the underlying scientific concepts that are being addressed. For research uses, visualization is a necessary component for large data set analysis and is addressed by a number of software vendors on various workstation platforms. The frontiers of visualization have pushed beyond what can be expressed as a simple two-dimensional monochrome line images to include interaction modes for color, sound, motion and touch. As data sets become larger, choices need to be made about the manner in which to represent (or interact with) the data which means that users cannot avoid learning about the opportunities, as well as the limitations, of the visualization process.

References

Brett M. Averick, Richard G. Carter, Jorge J. More, and Guo-Liang Xue, *The MINPACK-2 Test Problem Collection*, Argonne National Laboratory, June 1992 (preprint MCS-P153-0692)

Cunningham, S. and R.J. Hubbold (Eds.)., *Interactive Learning Through Visualization*, Springer-Verlag, New York, 1992.

Fogler, H.S., and S.M. Montgomery (1993). Interactive Computer Modules for Chemical Engineering Instruction. *CACHE News*. Fall, 1-5.

Göde, E. and R. Cuénod (1990). Numerical Simulation of Flow in a Hydraulic Turbine. *Chemical Engineering Progress*, **86**(8), 35-41.

Jolls, K.R., M.C. Schmitz, and D.C. Coy (1991). Seeing is believing: a new look at an old subject. *The Chemical Engineer*. 30 May, 42-16.

Jolls, K.R. and J.L. Butterbaugh (1992). Confirming Thermodynamic Stability. *Chemical Engineering Education*, **26**(3) 124-129.

Jolls, K.R., M. Nelson, and D. Lumba (1994) Teaching Staged-Process Design Through Interactive Computer Graphics, **28**(2) 110-115.

Kus, F., Personal Communication, McMaster University, 1994.

Schmitz, B. (1994). Engineering Visualization: Not Just Pretty Pictures. *Computer-Aided Engineering*, March, 44-50.

Squires, R.G. et al (1991). Purdue-Industry Computer Simulation Modules: The Amoco Resid Hydrotreater Process. *Chemical Engineering Education*, Spring, 98-101.

Tufte, Edward R., *The Visual Display of Quantitative Information*, Graphics Press, Cheshire, Connecticut, 1983

West, T. (1992). A Return to Visual Thinking. *Computer Graphics World*, November, 116-115.

2001

Brice Carnahan
University of Michigan
Ann Arbor, MI 49109

Abstract

1996 is a special year in the history of computing. It marks the 50th anniversary of the public announcement of the ENIAC, the first general-purpose electronic digital computer, the 25th anniversary of the first commercial microprocessor, the Intel 4004, and the 15th anniversary of the first IBM PC, whose basic structure and open architecture set the standard for subsequent personal computer development.

Most technological developments of consequence pass through the familiar S curve of slow initial growth, then a period of rapid acceleration, and finally another slow-growth phase of important, but marginal, improvement. For the digital computer, the slow growth period lasted about 15 years. The acceleration phase began with the introduction of the transistor in the late 1950s, received additional thrust with each new transforming technology (time-sharing, integrated circuits, real-time minicomputers, networking, the microprocessor, interactive graphical operating systems and programming environments, supercomputers and parallel machines, high speed communications, the Internet, the World Wide Web, etc.), and continues unabated to this day.

We describe past developments and current trends in the areas of computing hardware, communication, and software and make a few predictions about the state of computing in general and in chemical engineering education in the year 2001.

Introduction

Only “yesterday” those two most prominent dates of 20th century fiction, Orwell's *1984* and Clark's *2001*, seemed far into the future. Today, 1984 (do you remember your IBM PC AT or Macintosh 128?) is long past, and Big Brother is (almost) nowhere in sight.

2001 looms over the horizon, a mere half-decade away. And Clark's HAL, malevolent and almost infallible, also seems nowhere in sight. But, of course, he won't be “born” until 1997, so there is still time! One certainty is that HAL will look nothing like Kubric's 1969 vision of him. No astronaut in full space suit will float among HAL's innards. More likely, all of HAL's parallel processing capability will be encompassed in many microprocessors, housed in a “box” of quite modest size.

Neither Orwell nor Clark should be faulted for his vision. Predictions about the future of technology have not been particularly accurate. Verne was right about moon travel, and in his 1863 novel *Paris in the Twentieth Century* correctly imagined the automobile, electric lights,

and the fax machine, but was wrong about many others. Edward Tenner, in his recent *Why Things Bite Back*, argues convincingly that most predictions about technological developments turn out to be either off the mark or just dead wrong.

Two of my favorite predictions, both from Vannevar Bush (inventor of the analog computer and World War II production czar), show how long-term right and short-term wrong the *same* person can be about the *same* subject:

> " ... The MEMEX will be for individual use, about the size of a desk with display and keyboard that will allow quick reference to private records, journal articles and newspapers, and perform calculations ..."
>
> in *As We May Think* (1945)
>
> " ... Will we soon have a personal machine for our own use? Unfortunately not!"
>
> in *MEMEX Revisited* (1967)

Incidentally, Bush, a consultant for IBM, is reputed to have told the IBM Board in 1955 that no more than 100 IBM 650s should be built, since they could do all the computing that the World needed done!

Even short-term prediction is fraught with peril. Few would have predicted in 1993 that a hyperlinked World Wide Web would transform computing from a machine-centered view to a network-centered one almost overnight. Despite such revolutionary surprises, technology instills a sense of unfolding promise, making prediction almost impossible to resist. Hence my title, *2001* (granted, I'm not looking all *that* far into the future!).

Hardware

Hennesey and Patterson, in their superb 1995 text *Computer Architecture* (a must-read for anyone interested in low-level computer structure and functionality), briefly outline the history of computer hardware and architectural development, noting that a 1995 desktop PC (costing say $4,000 in 1995 dollars) had more main and disk memory, and better performance, than a million-dollar (1965 dollars) 1965 main-frame.

Processors and Memory

During the twenty five years following ENIAC, actual computer "performance" (Hennesey and Patterson), a rough measure of ability to process a comprehensive mixed set of test programs, improved by about 25% per year for main-frames, year in and year out. That rate improved to between 25% and 30% during the 1970s, primarily because of the introduction of minicomputers into the hardware mix. Since 1980, following widespread production of microprocessors, the overall performance growth rate has climbed to roughly 35% per year. The Intel CISC (complex instruction set computer) microprocessor family, used in perhaps 85 percent of the approximately 75 million machines that will be sold worldwide in 1996, is primarily responsible for this acceleration in average performance.

RISC (reduced instruction set computer) processors were first introduced in 1984. Since then, overall performance of computers incorporating this class of processors has improved at about a 50% annual rate. Although used in only 10 to 15 percent of computers in the current

market, RISC processors play a significant role in engineering computing, since they are used primarily in high-performance engineering workstations and parallel machines (also in Power Macintoshes).

Figure 1 (data from URL *infopad.eecs.Berkeley.edu/CIC*) shows that raw peak processor capability (rather than observed average performance for assembled computers), estimated roughly as proportional to the product of processor transistor count and clock speed, has doubled about every eighteen months over a 20 year period for the Intel family of CISC microprocessors. The Power PC and DEC Alpha RISC microprocessor families show even shorter peak performance doubling times of 12 to 14 months.

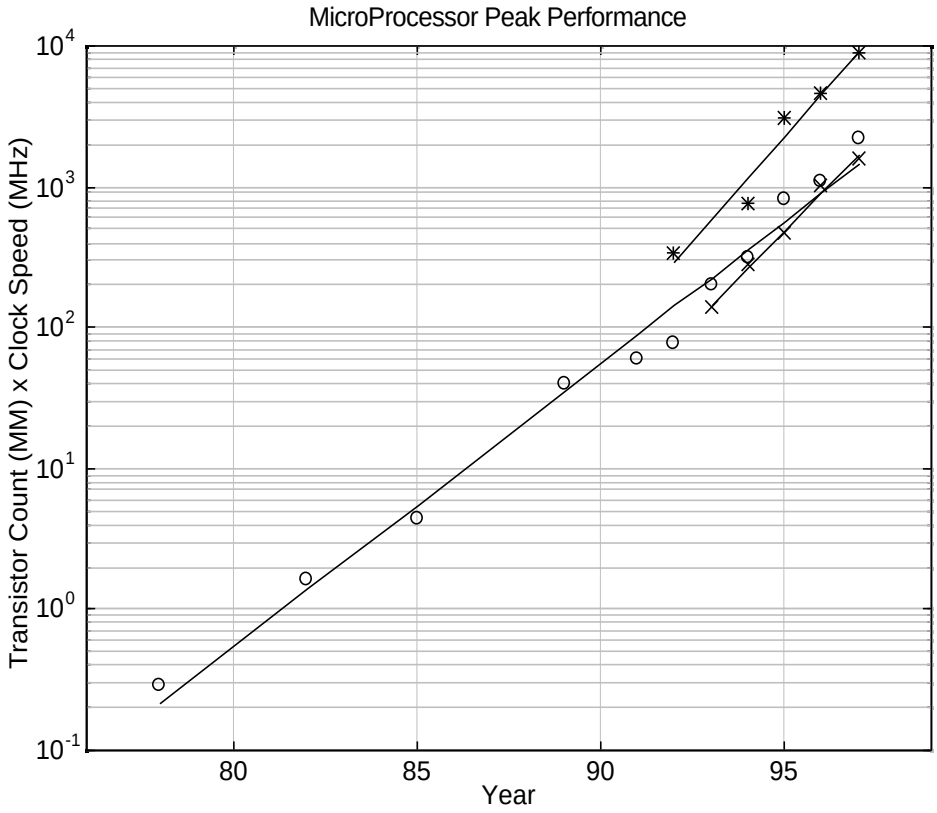

Figure 1. Peak performance estimated from the product of transistor count and clock speed for Intel (o), Power PC (x) and DEC Alpha (*) microprocessors.

Three current trends in chip and magnetic disk technologies are (Hennesey and Patterson):

1. Transistor density for integrated circuit logic (microprocessors) is increasing at about 50% per year; growth rates in transistor count per chip are in the range 60% - 80% per year.

2. Dynamic (main memory) RAM bit density is increasing at about 60% per year, resulting both from higher transistor densities and design improvements requiring fewer transistors per memory cell. The result is RAM of much larger capacity and much lower cost than was available even two or three years ago. 16 Mb chips are now standard, and 32 Mb and 64 Mb chips are widely available. [NEC has recently reported development of an experimental 4 Gbit RAM chip (256 times the capacity of the current standard chip), with production planned for around 2005.]

3. Prior to 1990, magnetic disk densities increased at about 25% per year. Since 1990, the rate has accelerated significantly to about 50% per year.

What about costs? Here, the impact of the technical developments alluded to in the preceding paragraphs, improved manufacturing performance (yield of good chips), and economies of scale resulting from volume production and commoditization of microcomputers, have led to incredible reductions in the cost/performance ratio.

Figure 2 shows a retail price curve for 16 Mb DRAM (dynamic RAM) chips between 1993 and 1996. The cost reduction by a factor of 8 to 10 over the 3 to 4 year life cycle of this DRAM chip is typical of that for prior standard chip sizes as well. With the compounding of cost reductions for standard chips, the cost per megabyte of DRAM has dropped incredibly from over $17,500 in 1977 to about $5 in 1996 (both in 1996 dollars). Thus real RAM costs have been reduced by a factor of 3500 in about 20 years!

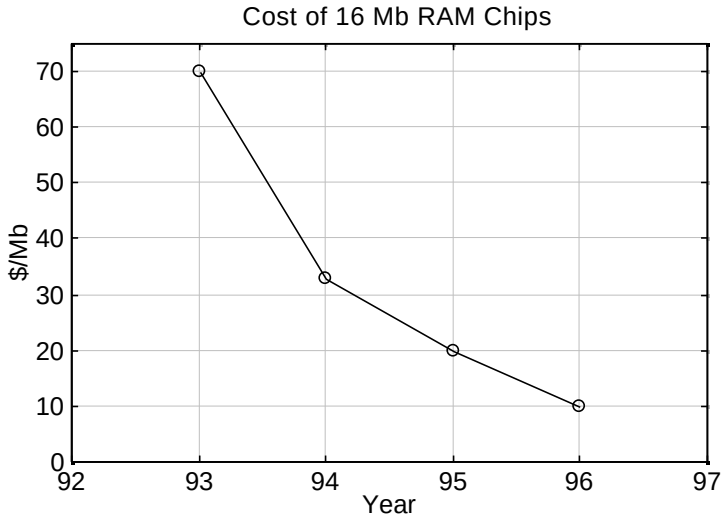

Figure 2. Cost of 16 Mb RAM chips - 8 required for 16 MB (data from Hennesey and Patterson).

What are the prospects for continuation of the performance trends for processor and memory chips described above? Very good, at least until 2001 and probably for several years beyond. Processor chips with a billion transistors and memory chips with many billions of

transistors are certain to appear before the end of the first decade of the next millennium.

However, physical limitations of current photolithographic processes will force major changes in chip production technology before then. As shown in Fig. 3, the minimum feature size (e.g., width of “wire” traces) of chip-making photolithographic processes on silicon has fallen from 12 microns in 1970 to 3.5 microns in 1980 to 0.8 microns in 1990 to 0.3 microns in 1996; at least two new 0.25 micron fabrication facilities will be in full production by mid-1997. The apparent limit of this technology, 0.365 microns, the wavelength of the ultraviolet light currently used for photoresist exposure, will in fact be reduced by more than half, to 0.1 microns, with lasers emitting 0.248 micron or 0.193 micron ultraviolet light and clever use of phase-shifted interference lithography. However, barring some even more clever (and unlikely) improvement within the next very few years, the minimum feature size of photolithographic technology is probably about 0.1 microns, likely to be in common use during the first half of the next decade (Stix).

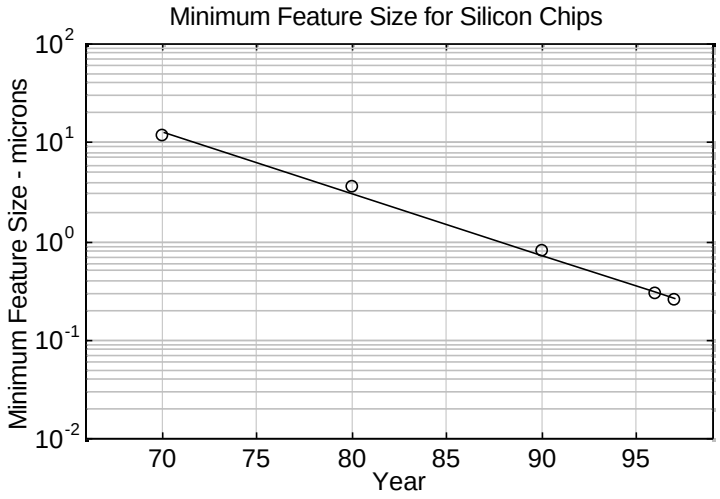

Figure 3. Minimum feature size for silicon chips.

Thus new technologies using photolithography with much shorter wavelengths or, more likely, completely new technologies not involving the interplay between light and photoreactive polymer resists, will be required. The new technologies may involve very-difficult-to-focus x-ray or rather slow electron beam processing (used by NEC in development of the experimental 4 Gbit DRAM chip mentioned earlier).

IBM researchers have reported creation of microstructures on silicon as small as 30 nanometers, and many university and corporate researchers worldwide are working on development of the ultimate smallest silicon component, the single-electron transistor, for which the minimum feature size is about 7 nanometers. Thus, for transistor-based micro-electronic silicon chips there *will* be an ultimate maximum transistor density and minimum feature size, whatever chip fabrication technology supplants photolithography. If the trend toward smaller

feature size illustrated in Figure 3 were to continue unabated (improbable), that ultimate limit, two orders of magnitude smaller than 0.1 micron, would be reached sometime around 2025.

From the numbers cited in the previous paragraphs, it appears likely that the rapid increases in transistor density (hence computer performance) we have seen in past decades will continue for at least one more decade. Whether the pace continues for the decade after that will depend as much on the development of cost-effective manufacturing techniques as on theoretical possibilities. The sheer cost of a new “fab” (chip fabrication plant) may, in fact, determine when Moore's Law (transistor density doubles every eighteen months), first postulated in the mid-1960’s by a founder of Intel, finally breaks down. The capital investment in a typical fab has climbed from about $1 million in 1970 to about $1 billion in 1994 to about $2.5 billion for a new Texas Instrument 0.25 micron fab scheduled to become operational in 1997.

Given that gross revenues for the largest chip maker, Intel, will be approximately $25 billion in 1996 (and growing at a compounded rate of about 35% per year) it seems doubtful that more than a very few companies worldwide will have the resources to create fabs based on entirely new technologies that may well cost $10 billion each by 2005 or so when the current photolithographic process will have run its course. $10 billion is not small change for any company; it is about 10% of the current *combined* market capitalization of the Big Three US automakers.

Parallel Processing

This is not to say that computational power of typical, affordable computers will at some point fail to grow at the exponential rate we are accustomed to. What cannot be accomplished with speedier individual processors will be made possible by processing in parallel on more than one.

The supercomputing and parallel processing paradigm is currently going through perilous times, as several once-prominent companies have either gone bankrupt or been taken over by larger ones whose major revenues come from more conventional computer and/or workstation sales. One current trend that seems certain to continue involves use of off-the-shelf mass-produced (typically RISC) processors, rather than extremely fast but very expensive custom-designed processors, such as have traditionally been used in the fastest supercomputers with pipelined vector architectures. At least one experimental parallel machine with over 9000 processors (from Intel, using Pentium II CISC processors) has already performed useful calculations at speeds exceeding 1 Teraflop (trillions of floating-point operations per second), almost three orders of magnitude faster than the fastest current custom-built supercomputer processor made by Fujitsu. By 2001 a 10-teraflop machine (to be built by IBM for the Energy Department) will be in operation at the Los Alamos or Livermore National Laboratories.

Currently, two general architectures predominate, one called SYM (symmetric multiprocessing), involving a shared memory accessible (either directly or indirectly) by all processors, and the other called MIMD (multiple instruction, multiple data) involving networked processors, each having its own “private” memory, inaccessible to the other processors.

Each of the architectures has its strengths and weaknesses. Programming of SYM machines is simpler than for MIMD machines (though explicit synchronization is required to

avoid data access "races" by contending processors). For machines with a small number of processors (for example, just a few for desktop machines), the SYM architecture is likely to dominate. Unfortunately, as the number of processors grows, the bus structures for the shared memory SYM machines can become extremely complicated, so scaleup to massive parallelism (beyond say a few hundred processing nodes) is difficult technically and very expensive.

On the other hand, the networked individual processor MIMD model typically requires message-passing software such as PVM (Parallel Virtual Machine) or MPI (Message Passing Interface) to explicitly request transfer of information from one processor's memory to the memory of another processor that needs it. This complicates software development. The individual processors on these parallel machines behave like individual workstations with each node, normally running its own copy of the (Unix) operating system, attached to a local area network (albeit a very, very fast one, called a *switch*). Because of the architecture, MIMD machines are inherently scalable to quite large numbers (many thousands) of processors.

Parallel programs for tightly-switched MIMD machines can often be run as easily (though not as quickly) on networks of workstations. My guess is that this model will be the predominant one for engineering computing by 2001. Software to ease the programming problem is coming to market, and there will be a vast resource of unused compute cycles on very high performance networked workstations already in place in most engineering-oriented businesses (and universities), just waiting to be exploited for solution of computationally complex problems.

Communications

Without doubt, communications is the "computing" area experiencing the greatest acceleration in growth and greatest decrease in price/performance ratio. Daniel Atkins, Dean of the School of Information and Library Sciences and Professor of Computer Engineering at Michigan, estimates (personal communication) that the price/performance ratio for communication bandwidth is currently decreasing at a rate ten times that for computing hardware.

Tables 1 and 2 show the astounding impact of the capacity of storage media and network bandwidth on the storage and transmission of typical documents (here for uncompressed text only). Note also that the recently released single-side/single-layer DVD (Digital Versatile Disk) has about eight times (4.7 GBytes) the capacity of the conventional CD ROM listed in the table. If two layers per side (read by adjusting the focus of the laser) and both sides are used, then a single DVD, physically the size of a CD, can store a total of 18.8 GBytes, equivalent in capacity to 32 current CD ROMs. So all the text (but not the graphics!) in all the documents in the Library of Congress could be stored in fewer than 1000 such DVDs. (Don't expect to see this anytime soon! The Library of Congress is currently creating a digital library by scanning at high-resolution about 1 million new and recent documents per year; the library will be available to K12 and university libraries as an on-line resource. Digitizing all 20 million documents in the Library would be a massive and expensive undertaking, and is unlikely to be funded.)

The fastest local area network data-transmission rates, about 5 Mbps (millions of bits per second) in 1980, will certainly be greater than 1 Gbps by 2001 (1 Gbps Ethernet equipment will be available from manufacturers sometime in 1997). Milti-gigabit rates for high-performance

switched networks, such as those used in massively parallel MIMD machines, will also be common by 2001.

Table 1: Sizes of Typical Stored Documents (Lucky)

Document	Bytes	Floppy Disks	CD ROMs
Page	2,400	0	1
Report	72,000	0	1
Book	720,000	1	1
Dictionary	60,000,000	43	1
Encyclopedia	130,000,000	93	1
Local Library	70,000,000,000	50,000	108
College Library	700,000,000,000	500,000	700
Library of Congress	18,000,000,000,000	12,900,000	26,000

Table 2: Transmission Times for Typical Documents (Lucky)

Document	Modem 28,000 bps	T1 (1.5Mbps)	Fiber Optic (1.7 Gbps)
Page	0.68 sec	0.013 sec	0.0000113 sec
Report	20.5 sec	0.38 sec	0.000339 sec
Book	3.42 min	3.84 sec	0.0034 sec
Dictionary	4.76 hr	5.3 min	0.28 sec
Encyclopedia	10.32 hr	11.6 hr	0.61 sec
Local Library	232 days	4.32 days	5.49 min
College Library	6.34 yr	43.2 days	0.92 hr
Library of Congress	163 yr	3.04 years	23.5 hr

In late 1996, fast backbone Internet lines operate at about 45 Mbps. By the end of 1997, the facilities of a yet faster Internet, called vBNS (very high speed backbone network service) or Internet II, will connect about 100 research Universities and the national supercomputing sites with optical backbone connections operating at 155 Mbps. No doubt Internet II will, in turn, be replaced by Internet III, having a substantially greater bandwidth. Gigabit Internet

backbone service will almost certainly be available for at least some users by 2001.

Most current modems attached to commercial cable television networks support Internet transmissions at T1 rates (1.5 Mbps), some 50 times typical telephone modem rates (28.8 kbps). Potentially, however, broadband cable modems with optical cable lines are capable of very much higher data rates, in the 10 to 40 Mbps range, at least when downloading (transfers from the Net to local PC); uploading (PC to Net) is normally slower, but still several times faster than for current telephone modem communication. Unfortunately, only about 10% of current commercial cable systems support two-way communication at all (so that PC to Net transmissions require a separate telephone modem), a situation that will probably be remedied by 2001 as companies upgrade their equipment and replace copper lines with optical ones.

Wireless services are now pervasive, and relatively inexpensive. Wireless local area networks are proliferating, and represent a cost-effective alternative to copper or optical cabling as a way of "wiring" a building (though the data rates cannot compete with those for high-bandwidth optical fiber). Several different sets of globe-girdling LEO (low-earth-orbit) satellite systems are about to be put into place by consortiums of computing and communication companies. They will provide high speed wireless services from virtually any place on the earth's surface. For example, 28 Mbps wireless Internet connections are planned for the $9 billion Microsoft/McCaw/Boeing Teldesic system of 288 satellites, scheduled to be fully operational in 2001.

The Web Changes Everything

Local (LAN) and wide area (WAN) networks have been with us for about twenty five years, starting with the Ethernet LAN, developed at XEROX PARC in the early 1970s, and the wide-area ARPANet, for which the first machine to machine transmission (from the University of Southern California to the University of California at Berkeley) took place in September 1969. Prior to the development of these technologies, most remote computing was performed at dumb terminals connected by telephone modem to mainframes operating in time-sharing mode.

Local area networks (LANs) made possible the high-speed sharing of resources such as mass storage units and printers. With the development of Unix in the late 1960s and early 1970s, remote computational resources on the network could also be tapped by multiple users directly from their own networked machines.

ARPANet allowed for remote messaging (EMail) and file transfer, and the sharing of ideas, data, research results and computational resources among users of machines that were widely separated geographically. A key breakthrough came with the adoption of the Internet transmission protocol (TCP/IP) and Internet domain addressing scheme in 1974. TCP/IP created a universally recognized standard for transmission of information "packets" over the network, and the addressing standard supported creation of world-wide directories, allowing unique identification of every networked machine and the speedy delivery of information packets to their intended target machines.

With support from the US Defense Department, ARPANet grew at a steady but unspectacular pace during the 1970s from 24 sites in 1971 to about 200 in 1981. In 1986 ARPANet

was replaced by a less "military," more academic, network called NSFNet. One of the principal functions of NSFNet was to provide high bandwidth access between University campuses and five National Supercomputing sites (reduced to two in 1996). NSFNet was in turn replaced by the loosely (some believe chaotically) managed, rather amorphous Internet (a network of networks) in the late 1980s. As of late 1996, the world-wide Internet ties together more than 100,000 individual networks with more than 50 million attached (either directly or through Internet service providers) individual machines.

The incredible recent growth in Internet infrastructure and use can be attributed in large part to the development of the World Wide Web (*www* or simply *the Web*), which allows for easy access to *hyperlinked* documents stored on any accessible machine located anywhere on the Internet. Hypertext, which permits the nonlinear linking (essentially without restriction) of one piece of information with another (presumably related) one, was first suggested by Vannevar Bush (see earlier) in 1945, and was promoted by researchers at XEROX PARC in the early 1980's as a way of referencing and retrieving related information.

Tim Berners-Lee, an English physicist working at CERN in Zurich, is credited with developing three key Web concepts in the late 1980s and early 1990s:

A standard language for encoding hypertext documents.

A standard Internet protocol for linking and transmitting hypertext documents.

An addressing system for locating documents.

The language HTML (hypertext markup language), protocol HTTP (hypertext transfer protocol), and addressing system URL (universal resource locator) form the underpinnings of the current Web. One of the key decisions made by Berners-Lee was that a central directory (which would allow for deleting and updating links as hyperlinked documents were removed or modified) was infeasible for a "World-Wide" information web, i.e., that "dangling" hyperlinks to a removed document would simply have to be tolerated for the greater good of allowing independent actions by individual Web document owners.

In 1990, Berners-Lee developed a browser/editor at CERN for creating, accessing and displaying hypertext documents (principally research papers and data at CERN) using his HTML, HTTP, and URL concepts. However, it was not until 1993 when the Mosaic browser was created at NSCA (University of Illinois), that the potential of the Web became apparent (at least to some). With the formation of Netscape, Inc. and the release of the Netscape browser in 1994, the Web "took off".

The principal resources that can be hyperlinked on the web go far beyond mere text, indeed can be almost anything (text, graphics, sound, animation, video, virtual reality images, signals from an instrument on a remote experiment, software, ...); essentially any information that can be digitized can, or soon will be, Web-linkable.

It is, I think, fair to say that the Web changes everything. If, as McLuhan suggested in 1967, the medium is the message, then here the message is the Web. Some (myself included) think the Web may be as revolutionary an invention affecting the way we communicate and access and disseminate information as was the printing press.

The Web is the first truly new medium since television; it is, in fact, a multi-medium be-

cause the range of resources it can access potentially includes essentially all of the other mass media (newspapers, movies, radio, television). But in addition to providing a vehicle for "push" technology or "webcasting," in which the Net delivers information to the user (much as broadcasters do), the Web (1) allows an individual user to *interact* with resources and other users on the Web, and (2) allows an individual user to *create* resources, in essence to become a "publisher," without formal approval by an oversight authority. The Web can be an extraordinarily liberating medium for individual creativity (much that is worthless gets created in the process as well). The Web (or whatever supplants it) is clearly a multi-medium destined to have enormous impact on information dissemination, education, and commerce.

The Web's growth during the past three years is simply phenomenal. Virtually every corporation, academic institution, government agency, and uncounted individuals now have a presence on the Web, at least in the form of a hyperlinked home page.

Commercial Web activity is growing apace. It is hard to believe that the first commercial advertisement on the Web (for a lawyer's services) (1) raised a furor among Web users as a violation of Web etiquette, and (2) happened as recently as April 19, 1994. Contrast this with an estimate by Forrester Research that commercial Web transactions will exceed $10 billion in 1999, with computer hardware and software sales comprising a large piece of the commercial pie.

The recent flurry of multi-billion dollar buyouts and takeovers involving television, telephone, wireless, cable, computing and on-line Net-access companies attests to the fluidity of the situation and to the uncertainty about which technologies or combinations of them will eventually win in the "interactive" educational, entertainment, and commercial marketplace. My guess is that by 2001 commercial and educational interactivity will be centered on the Web and computer (with commercial cable systems providing Net connectivity for a substantial fraction of home users), rather than on interactive cable and the television set. News and entertainment currently delivered by mass media will stay in the domains of newspapers, radio, movies, and television, but the Web will play a role in these areas too (on-line magazines, webcast stock market reports and breaking news, etc.).

Perhaps the Web's greatest impact has been on the nature of "computing" itself. The center of gravity for computing activity is shifting perceptibly from individual computers and workstations to the network. Major corporations such as IBM and Microsoft have reoriented their missions to focus on delivery of "Net-centric" services and "Net-aware" application software. The distinction between what's done locally and what's done remotely will blur substantially in the coming years. Many corporations have already taken advantage of the net-centric Web model by creating internal internets called *intranets*, mini-Internets that use standard Web software such as browsers for accessing and updating corporate data and information; firewalls (software to isolate the internal intranet from the Internet) protect corporate information from the outside world.

Because the Web is so new and has grown so rapidly, it has many rough edges and problems. Service can be slow; breakdowns do occur (and will surely continue to occur). However, the Net's distributed nature and redundancy of communication paths virtually insures that only parts of it will be affected by a particular failure; a catastrophic Net breakdown comparable to a nation-wide power outage seems unlikely. On balance, the commercial carriers have done a

remarkable job in installing new lines and equipment in response to rising demand, and will undoubtedly continue to do so, as a wholly new optical fiber and wireless communication infrastructure takes shape over the next decade.

Two issues that need to be addressed before the Web can reach its full potential in the commercial arena are: (1) security of Web-based financial transactions, and (2) copyright protection for Web-accessible intellectual property. The problems in these areas are challenging both technically and legally, and too involved to discuss here (and I don't know much about them!).

Substantial work has been done on financial security issues, mostly involving some form of encryption, and the Web is already being used fairly heavily for on-line sales and financial transactions, with apparently little fraud or error (but the occasional horror story shows that problems exist). US copyright laws are under active discussion in the Congress, and an International Copyright Convention of 160 nations, meeting during 1996 and 1997, will attempt to reach global agreement on the handling of intellectual property, particularly computer-based intellectual property. Real copyright protection is more likely to come from the bottom up (in the form of technical safeguards on the Web) than from the top down (international treaties or US copyright law). Because these issues are so important to the future of the Web and to the use of Web-based information for commerce and education, I believe that some reasonable and effective solutions for current security and copyright problems are likely to be in place by 2001.

Software

The most important software for engineering students, faculty, and professionals can be broadly classified by general function: (1) system software, (2) network services, (3) programming language translators/compilers, (4) general productivity tools, (5) problem, task, or discipline-oriented applications, and (6) multi-media instructional aids.

System software includes the base operating system for the computer and a myriad of programs for providing communication and access to system resources. The first operating system, for controlling the processing of batches of programs on punched cards on the IBM 704, was released in 1957. Subsequent systems, for time-shared access to mainframes, appeared in the mid-1960s. With each new class of hardware (minicomputers, personal computers, workstations, etc.), new and usually more sophisticated operating systems appeared and then slowly evolved over the years.

Currently, the most important operating systems are Unix, used on engineering workstations with RISC processors, Windows95/DOS and Windows NT for personal computers with Intel and compatible processors, and the Macintosh OS for Macintosh and clones with Power PC processors. All now have graphical user interfaces that are descendants of windowed interfaces developed at XEROX PARC in the early 1970s and popularized with release of the Macintosh in 1984; the interfaces appear remarkably alike to the user, making switching from one machine to another much less onerous than in earlier times (but still not without pain!).

New features such as support for multiple users (Unix), preemptive multi-tasking (time-sharing of the local processor among several running programs), threads (parallel minitasks within a single application), and network services (e.g., remote file transfer, electronic mail,

Internet and Web access) have over the years been incorporated into or made available to operating systems.

Future versions of the predominant operating systems will almost certainly be much more "network-aware" than current ones, i.e., it is likely that the user interface (probably with a three-dimensional appearance) will eventually make little distinction between what is local to the user's workstation and what is on the network or Web.

The desktop metaphor with its emphasis on files and folders has been dominant since the introduction of the Macintosh in 1984, but is giving way to new metaphors, in particular that of the compound document as a container for objects - textual, graphical, sound, video, computational. Group interfaces for conferencing and for working collaboratively over the network are under development, and likely to be very important in both the academic and industrial workplace by 2001. Virtual reality will almost certainly play a role as a 3-dimensional imaging tool for navigating in the net-centered operating systems of the future. New multi-sensory tools involving gesture, eye and body motion, voice, smell (maybe not!) will eventually supplement the keyboard and mouse for interacting with those future operating systems.

Stephanopolous and Han (see p. 239) have described programming language developments in lucid detail, so I won't elaboratye on their observations here, except to note that there are strong trends toward development of modular, structured, and more easily maintained and reusable object-oriented programs (the latter allow for hierarchical decomposition, and encapsulate both data structure and algorithm). My guess is that by 2001 most chemical engineering students will be trained to use high-level and visually-oriented computational and "programming" tools rather than traditional procedure-oriented languages such as FORTRAN and C.

In general, software improvements come much more slowly than those for hardware described earlier. This results from the fact that much code is still "handcrafted," that many popular applications were originally written years ago with unstructured code, that subsequent "improvements" have been made in ad hoc fashion, and that programs are often quite large (many are enormous), hence difficult to validate.

Nevertheless, there is hope from a new computing discipline, software engineering, from object-oriented programming technology, from the trend toward open systems, cross-platform (different kinds of computers) compatibility, emerging international standards for communications and hardware interfacing, and, in some cases, cooperation among software developers and hardware manufacturers.

One of the most exciting recent software developments is the emergence of Sun Microsystems' Java, an object-oriented language for creating programs that reside on network servers. When a networked user requests a service or a calculation or a display provided by a Java program (small ones are called applets), the Java code is downloaded to the user's workstation from the server, and either immediately interpreted locally by a *Java virtual machine* (JVM) translator or compiled into machine code for the workstation and executed. JVMs are available for essentially all personal computers and workstations and are being incorporated into Web browsers such as Netscape and Microsoft Explorer and other application programs.

This means that a Java program automatically has cross-platform compatibility (Nirvana for a programmer), since it can be run without modification on any computer, such as a Unix

workstation, Macintosh, or "Wintel" (Windows operating system and Intel processor) machine with a JVM or Java-compliant browser. The user need not store the program locally, and the Java programmer can change the program without concern for the machine it will eventually be run on. Java, or some successor to it, is central to the notion of Net-centric computing.

Sun and a few other hardware/software vendors view Java as the tool for creating network-centered operating systems to replace conventional workstation operating systems such as Windows and the Macintosh OS, treating individual machines as network computers. My guess is that this will not happen, and that conventional operating systems will instead become much more Java-compliant by incorporating JVMs and Java compilers as core components.

Many other new software developments are also Web oriented, especially for creating and improving search engines for locating and cataloging URLs of interest on the Web, and for creating intelligent Web agents (small programs that move about the Web searching data bases and filtering information to carry out specific tasks of interest to the user). Current Web search engines are quite primitive, and identify URLs of interest based on text matching only. Engines available in 2001 will likely allow for searching of text in context and searching of non-textual information, in particular, images (e.g., find pictures containing sunsets) and sounds (e.g., find soundbites with references to global warming).

Implications for Chemical Engineering Education

How do (and will) all these intense (and accelerating) computing activities affect academia in general and engineering education in particular? In many respects, the impact of computing on education and academic life has paralleled that in the world outside the academy. Substantial basic research that feeds the computer revolution is performed by academics and their students. The University of the present looks quite different from the University of even a decade ago. Virtually every desk supports a networked desktop computer, the University library is "on-line," and every dorm room is (or soon will be) connected to the rest of the electronic world. The computer has brought with it systemic changes in the ways the University conducts its business and research and interacts with its students, graduates, faculty, and staff.

The impact of the computer *in* the classroom has, to date, been much less dramatic than in other areas of the academy. An 1896 still photo of an engineering classroom, professor lecturing with chalk in hand, would look remarkably similar to most engineering classrooms in 1996. Will that paradigm last for yet another century? Not likely.

Despite its apparent lack of impact to date in most classrooms, the computer has certainly affected what we teach and how students learn, as is evident from the many earlier papers in this monograph. Most chemical process design [Biegler, Seader and Seider (p. 153), Grossmann (p. 171), Grossmann and Morari (p. 185)] and many control (Arkun and Garcia, p. 193), and separations processes (Taylor, p. 139) courses have already been transformed into computationally centered ones. Most undergraduate laboratories have also undergone major upgrading, and the computer is now an integral part of experimental systems, both for data acquisition and data processing, as described by Mellichamp and Joseph (p. 203).

Interactive multi-media instructional tools, such as those described by Fogler and Montgomery (p. 57) and Fogler (p. 103), allow faculty to create remedial and supplemental tools for

self-paced learning of course material that are particularly effective for students whose learning styles differ from the linear, analytical learning and teaching styles of most faculty. Numerical mathematical tools, described by Shacham, Cutlip, and Brauner (p. 73), allow students to solve nonlinear model equations and to carry out simulations that could not easily have been included in core engineering science coursework in thermodynamics (p. 85), transport processes (Finlayson and Hrymak, p. 125), and reaction engineering (Fogler, 103), as recently as a few years ago.

Nevertheless, the overall impact of the computer in our undergraduate curricula, particularly in our core chemical engineering science courses, has been less than might have been expected, given the computing resources now available on most engineering campuses. As pointed out by Kantor and Edgar (p. 9), most of our standard textbooks do not yet have a significant computer-orientation. I foresee the standard textbooks of the next century as much different from those of the past. The future text will not be structured as the linear sequence of static information found in our current texts, but will contain hyperlinked navigational aids, animation and sound, tools for manipulating data, and programs for carrying out simulations with user-supplied inputs. By definition, they will require the computer for their use, and may even be completely Web based, with small charges assessed for each access. (No doubt students will still want to print many screen images as they appear, so even these “electronic” texts are likely be accompanied by hard copy materials).

Most classrooms are not yet equipped with ready-to-use networked computing and projection facilities needed for doing good in-class demonstrations. Such equipment represents a significant investment by the university, but it will be essential for engineering classroom instruction in the next decade. Classroom visualization (Hrymak and Monger, p. 253) of the unfolding dynamic solution of a fluid mechanics problem is worth much more than a thousand faculty words.

Creation of good computer-oriented instructional materials and stimulating in-class demonstrations (even with the best computing and display equipment in place) is challenging, time-consuming, and expensive. Unfortunately, such efforts by faculty often receive little recognition by department chairs and deans, particularly at research universities, so we may continue to see fairly slow progress in fully integrating computing work into our core engineering science courses.

On the other hand, the Web could revolutionize the way we manage course-related activities. By 2001, virtually every course will have a home page on a Web site, with links to class schedules, problem assignments and solutions, interactive messaging, and student records. Driven by faculty interest and commitment, some courses will be strongly Web-centered, with facilities for student collaboration and group interaction, in-class note taking (using laptops with wireless communication?), interactive multi-media tools that integrate materials from local machines, CD ROM and the network, on-line examinations, and access to a wide array of data base resources and numerical and discipline-specific software; classical lecture presentations in these courses will diminish in importance. Some departments may have full-time Webmasters whose principal duties involve assisting faculty in bringing the curriculum to the Web.

The Virtual University?

Viewed more broadly, the new computing technologies, and particularly the Web, promise to transform the way Universities keep in touch with their alumni and deliver continuing education to professionals. As the bandwidth of Internet trunk lines rises dramatically in the next decade, videoconferencing, on-line interaction and collaboration, and Web access to digital data bases and libraries will make cost-effective delivery of distance learning a reality.

Some Internet-based experimental distance learning experiments are already underway. One, a joint venture of the University of Michigan, the State of Michigan and the auto industry called the "Virtual Automotive College," is being directed by James Duderstadt, a former Dean of Engineering and now President Emeritus of the University. The virtual college will deliver its first all-electronic courses to engineers in the automobile industry over the Internet during the Winter term of 1997.

In theory, a "virtual university" could be created using the new computing and communication tools and instructional technology with the Internet as its infrastructure. The driving force behind such a university would be delivery of instruction to large numbers of students and professionals in their homes or workplaces at much lower cost than with traditional campus-based courses. However, unless students have access to substantial interaction, group collaboration, and the opportunity to form learning communities, I'm skeptical of the near-term success of such a university, at least for young (18 to 24 year old) students. A college education involves much more than passive learning in near isolation. As suggested by Brown and Duguid, social experience, groups joined, scholars worked with, friendships made, prestige and marketability of degrees earned, and a sense of place play powerful roles in the College experience. It seems unlikely that a virtual university will ever be viewed with nostalgia as *alma mater*.

Summary

I have attempted to give an overview of past developments and current trends in digital computing and communications (in particular of networks, the Internet, and the Web), and to make some predictions about what the computing world will look like half a decade hence, in 2001. Many of the facts and figures are unattributed; for the most part, they come from extensive notes taken and articles (from newspapers, trade magazines, technical journals) clipped over many, many years.

The conjectures about the future are mostly my own; fortunately I can't be proved wrong until 2001, at which point the whole exercise can be done again!

References

Brown, J.S., and P. Duguid (1995). Universities in the Digital Age. Position paper, XEROX PARC, Palo Alto.

Hennesey, J.L., and D.A. Patterson (1995). *Computer Architecture: A Quantitative Approach.* Morgan Kaufmann, San Francisco.

Stix, G. (1995). Toward "Point One." *Scientific American,* (2), 90-95.

Lucky, J. (1989). *Silicon Dreams*. St. Martins Press.

AUTHOR INDEX

SUBJECT INDEX

N

O

P

Q

R

S

T

U

V

W

X

Y

Z