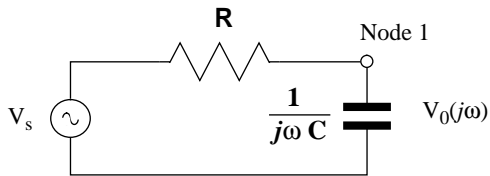


above but now to the phasor circuit (i.e., to the Fourier transform of the circuit and of the corresponding differential equation).



We obtain the algebraic equation

$$\frac{V_s - V_0(j\omega)}{R} = j\omega C V_0(j\omega)$$

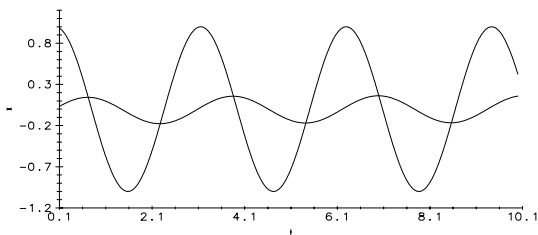
from which we obtain that

$$V_0(j\omega) = \frac{1}{1 + j\omega RC} V_s$$

Therefore, back in the time-domain

$$v_0(t) = \frac{A}{\sqrt{1 + (\omega RC)^2}} \cos(\omega t + \Phi)$$

Now, plotting the corresponding sinusoids at a given frequency ω for different values of R we obtain curves like these.



From these curves we see that the smaller the resistance, the larger the amplitude of the steady state response becomes. The correlation follows from the fact that the smaller the value of R , the larger the amplitude of the term

$$\frac{v_s(t)}{RC}$$

becomes in the ODE for $v_0(t)$. As expected from the analysis of the response to the pulse train, the faster the circuit can respond to changes in the input.

In the language of electrical engineering, we say that as the bandwidth of a communication channel increases, the data rate increases (the bandwidth increases as resistance drops in the circuit). This is an important principle illustrated with a first order system, and could easily be discussed in a class on ODEs, especially with the aid of computers.

Editors Note: good ODE solvers should have square and triangular pulses and pulse trains as pre-defined functions. Felzer's article shows why these functions are useful. □

Buying a Computer Lab

David Lerner

University of Kansas

Lawrence, KS 66045

ILI, ODEs, Mathematica Notebooks:

Last summer the Mathematics Department at the University of Kansas was awarded an ILI grant to establish a computer laboratory for our introductory ODE course. Our intention is to base the lab on a collection of Mathematica notebooks which we are writing. The existence of a suitable version of Mathematica is the only obvious constraint on the hardware. This article attempts to lay out the features we deemed most important and the reasons for our choice.

Our situation may not generalize to other institutions. Perhaps the most important

thing to remember when dealing with vendors is that everything is negotiable, and there is (almost) always another source or alternative product.

The Shopping List

User interface: We need a “point and click” graphical user interface (GUI). For one thing, Mathematica notebooks require this; for another, we do not want to explain anything about any operating system in class. A further consideration is that most of our faculty are accustomed to GUIs, since they have either Macs or PCs (with Windows) on their desks. We want to make it as easy as possible for them to use the lab.

Money: Finances are an issue to the extent that we must meet certain minimal requirements. The grant of \$100,000 (half from NSF and half from KU) must provide at least 17 workstations. This will allow classes of 30 or more students. We need floppy drives, keyboards, mice, and good color monitors as well as 100 or more MB of free space on each machine’s hard disk. To minimize swapping and paging, we’d like to have about 8MB of memory available after loading Mathematica.

Software: There should be one copy of Mathematica on each machine. It would, of course, be possible to run Mathematica off a server, but this would cause trouble on the occasions when everyone in the lab wants to do the same thing at the same time. (Mathematica has an educational grant program under which the software is sold for roughly half the usual, discounted educational price. The grant program is intended for this sort of project.)

Other uses were considered, as well. In our case, there is an exploding demand for computing resources in the upper level un-

dergraduate courses (PDEs, statistics, modeling, scientific computing, numerical analysis, dynamical systems, etc.). Thus we need things like C and FORTRAN compilers, simple text editors, and access to software like **Matlab**, public domain solvers, and the like. The availability of calculus and pre-calculus courseware is of little interest to us since the graphing calculator is our weapon of choice in these lower division courses.

Hardware:

- A file server and tape drive are necessary for printing, for collecting and distributing notebooks, and to provide access-protected, back-up storage of students’ files.
- Laser printer(s) must be capable of handling 10 pages/minute of post-script output. (Dot matrix printers in graphics mode are too slow, particularly in a lab setting.)
- Cards, transceivers or cables are required to hook things together and to the department network.

Speed: This is (probably) of less importance than the GUI and software in an introductory course, but it becomes significant as the problems get bigger (in PDEs or numerical analysis, for example). All other things being equal, we want as much raw computing power as we can get.

Simplicity: It must be relatively easy to get everything up and running in the first place. More important, it must be easy to maintain the system, upgrade the software, add or change printers, and so on. In this context, what constitutes “easy” depends a great deal on local expertise.

Choices: Our initial intention was to use the Mac IIci as a workstation. Most of our faculty use Macintoshes of some sort, so local

“dissemination” of our project would have been relatively easy; although the Mac II is no longer a “state of the art” machine, it would have served quite well for both the ODE course and for some of the upper level courses. But between November 1991, when our proposal was submitted, and the following October, when we finally put things out for bid, Apple came out with an improved machine called the Quadra, and Sun produced a new machine called the Classic.

If we were going to buy Macs, it seemed reasonable to get the newest model; the Quadra 700 would have suited us. Somewhat surprisingly, the deciding issue (in Apple versus anything else) turned out to be price. In spite of many meetings and assurances that they wanted our business, Apple couldn't or wouldn't sell us the Quodras at a price we could afford. We could have gone ahead with fewer machines, but we were not impressed by the capabilities of the Quadra. It seemed to us to be just an incremental improvement over the Mac II with its 5 year old technology. [For example, the Mathematica scripts we tested ran from 2 to 5 times faster in Windows 3.1 on a 25Mhz I486 machine than on a Quadra 700.]

We knew, of course, that we could get plenty of MSDOS machines for our money, but we had some concerns about [1] networking and security (password protection, vulnerability to viruses), [2] the stability of the combination of Mathematica and Windows together, and [3] the lack of a good memory management system. It is only fair to point out that these concerns should largely disappear with the introduction of Microsoft's new OS, Windows NT, assuming the product performs as advertised.

We were pleasantly surprised to discover that we could afford the Sun Classics *with* 15

inch color monitors. These are significantly faster than either Macintoshes or MSDOS machines. They come with 16MB of RAM and 207MB hard disks. The file server will be a (Sun) SPARCstation 10/30 with 32MB of RAM, a 400MB internal disk, and a 1.3GB external disk. The Solaris 2.1 operating system comes with x-windows support.

Third party developers routinely port their applications to the Sun platforms, and there is a lot of excellent public domain software. Moreover, Suns are the “standard” workstations for scientific computing, and to make a long story short, this is what we're getting.

Caveat: This is surely not the best choice for everyone. Indeed, many of our colleagues were (at first) frankly suspicious of our intentions. Behind the slick looking screen, the Suns are still Unix platforms. Many people claim that prior exposure to Unix is a sufficient condition for purchasing a Macintosh! In our case, we are fortunate to have enough local Unix expertise that setting up and maintaining the network is not a major concern.

The faculty's initial resistance disappeared when people realized that Mathematica's notebook interface is, for all practical purposes, independent of the platform. This means that notebooks can be created and tested on other machines, so that no one has to learn any Unix. There was also general support for the notion of providing students with “laboratory equipment” of the same quality that they will encounter in their future careers.

Finally, I should mention that low-end workstation technology is still improving rapidly; if this article were written a year from now, our choices might be altogether different. In particular, things to watch for include

the next generation of Intel chips, Windows NT, and PC versions of NeXTStep and Solaris which take advantage of the new hardware.

(Editors Note: Lerner's article is the first in a series on setting up an ODE computer lab. Stay tuned for further information.) □

The Savvy Solver II

Larry Shampine

Southern Methodist University

Dallas, TX 75275

Ed. note: Second order linear ODEs can have solutions that decay and others that become unbounded. This divergence of solutions causes problems that require an understanding of how solvers function. Shampine's example shows one difficulty and how the savvy solver resolves it.

Airy Functions: Several important functions are defined as solutions of Airy's equation, $y'' = xy$. $Ai(x)$ is the solution with initial values given below, and $Bi(x)$ is an independent solution:

$$Ai(0) = \frac{3^{-2/3}}{\Gamma(2/3)} \approx 0.355028$$

$$Ai'(0) = \frac{-3^{-1/3}}{\Gamma(1/3)} \approx 0.258819$$

$$Bi(0) = Ai(0) \sqrt{3}$$

$$Bi'(0) = -Ai'(0) \sqrt{3}$$

Airy's equation is so simple that a natural

way to compute $Ai(x)$ is to integrate the initial value problem. I did this with "top-of-the-line" codes based on the most popular kinds of discrete variable methods, namely Runge-Kutta, Adams, and backward differentiation formulas. In each case the code was asked to produce about five correct digits at $x=1$ and $x=10$. The results were:

	x=1	x=10
Runga Kutta	.135300	3742.20
Adams	.135298	3140.81
back solve	.135293	548.399
$Ai(x)$.135292	$\sim 10^{-10}$

Bad results from good software: The results at $x=1$ are what you might expect, but those at $x=10$ are terrible! The numerical solution plotted (Figure 1) was obtained using the variable step size Runge-Kutta code in the package MDEP (see the review in the last issue of this newsletter). It is just as bad as the other solutions. Towards the end of the interval it is negative and its magnitude is increasing rapidly. This despite the fact that $Ai(x)$ is positive for $x \geq 0$ and decays rapidly to zero. It looks like the codes all failed, but they *didn't* – they did just what they were supposed to do! The savvy solver knows that what codes do is different from what you want them to do, so let's talk about the difference.

Forgetting the Past: Discrete variable methods for the numerical solution of

$$y' = F(x, y), a \leq x \leq b, y(a) = A$$

start with the given value $y_0 = A$ and then produce an approximation $y_1 = y(x_1)$ at a point $x_1 > a$. The process is repeated, successively producing approximations y_j on a