02E15I

51-PWD-1625

WPI

Educational Software

Project Submitted to:

Professor Paul Davis

Worcester Polytechnic Institute

Kevin Smith [illegible]

July 30, 2002

Paul Davis

Professor Paul Davis, Advisor

Abstract

This project reviewed DELab, a piece of software written in conjunction with a differential equations textbook. DELab was assessed using an industry standard set of human-computer interface usability guidelines. Though the course of the evaluation it was discovered that DELab failed to satisfactorily meet expectations set forth by the surrogate user's interpretation of these guidelines. The recommendations that were generated included restructuring the user interface and improving the accuracy of the help text and instructions

Executive Summary

The Textbook Differential Equations Modeling With MatLab (P. Davis) and the software learning aid DELab are part of a system to be used in the teaching of differential equations. This system is not being fully utilized by teachers or students. To better understand the difficulties in DELab the system was evaluated from a design perspective based on user oriented design principles of the human computer interface. The result is recommendations for design improvements that will enhance the user interface.

The evaluation involved performing several surrogate user tests using the first MatLab exercise in chapters 1, 3, and 7 of Davis's differential equations text. The industry accepted method of comparison against guidelines was utilized while performing the evaluation. Extensive research in learning theory, teaching theory and user interface design and evaluation provided the necessary background.

Upon conclusion of this research and evaluation a list of recommendations was made concerning the areas of reworking the instructions and help systems, user interface redesign, and developing an educational purpose for the software.

The software DELab did not meet the usability guidelines selected for use in this project. Several shortcomings in the program were identified including instruction errors, creation of multiple windows to the point of confusing the user, and the absence of a menu-naming scheme. Through the course of evaluating DElab there were repeated errors in the instructions making the operation of DElab difficult or impossible. The interaction language of the menu system is not sufficient to allow the user to navigate the software and accomplish tasks.

To improve DELab several different remedies could be implemented. These would involve [1] rewriting the helpfile so that it includes accurate menu selections, function names, and more descriptive instructions; [2] redesigning the GUI so that objects display in a single window; [3] and, creating and putting into practice a uniform naming scheme for menu selection and object actions.

Within the process of evaluating DELab against the user interface usability guidelines DELab's role as an aid to education came into question. Considering all the above DELab is not devoid of purpose. With a little bit of modification it could be useful as part of a project based learning experience or as a tool to provide immediate feedback to an independently working Diffy Q's student.

Table of contents

Introduction

The Textbook Differential Equations Modeling With MatLab (P. Davis) and the software learning aid "DELab" are part of a system to be used in the teaching of differential equations. DElab was developed with and operates as an application within MatLab. MatLab is a high-level math utility software that is capable of creating user defined applications.

Operating from the author's experience that this system is not being well received and therefore fully utilized, the software DElab/text system was evaluated from a design perspective based on "user oriented design principles" of the human computer interface. The result is recommendations for design improvements that will enhance the user interface. Extensive research in learning theory, teaching theory and user interface design and evaluation provided the necessary background.

The evaluation involved performing several surrogate user tests using the first MatLab exercise in chapters 1, 3, and 7 of Davis's differential equations text. The industry accepted method of comparison against guidelines was utilized while performing the evaluation. Extensive research in learning theory, teaching theory and user interface design and evaluation provided the necessary background.

Upon conclusion of this research and evaluation a list of recommendations was made concerning the areas of reworking the instructions and help systems, user interface redesign, and developing an educational purpose for the software.

The software DELab did not meet the usability guidelines selected for use in this project. Several shortcomings in the program were identified including instruction errors, creation of multiple windows to the point of confusing the user, and the absence of a menu-naming scheme. Through the course of evaluating DElab there were repeated errors in the instructions making the operation of DElab difficult or impossible. The interaction language of the menu system is not sufficient to allow the user to navigate the software and accomplish tasks.

To improve DELab several different remedies could be implemented. These would involve [1] rewriting the helpfile so that it includes accurate menu selections, function names, and more descriptive instructions; [2] redesigning the GUI so that objects display in a single window; [3] and, creating and putting into practice a uniform naming scheme for menu selection and object actions.

Within the process of evaluating DELab against the user interface usability guidelines DELabs role as an aid to education came into question. Considering all the above DELab is not void of purpose. With a little bit of modification it could be useful as part of a project based learning experience or as a tool to provide immediate feedback to an independently working Diffy Q's student.

I would like to thank Zak Wheeling and Joshua West for their contribution to this project.

1 Literature review

1.1 Teaching Styles and Methods:

"Teaching in the college classroom is becoming increasingly complicated. It is no longer sufficient for the college professor to be competent in a field of specialty and to "profess" a substantial base of knowledge to a classroom full of willing students. Today's effective college teachers must be prepared not only to share in-depth knowledge of their discipline but also to know something about college students and how they learn" (Bonwell & Sutherland 3). To help improve DElab, it will be important for us to touch upon certain aspects of the alternative styles of teaching.

Due to the number of learning institutions using the traditional teaching style of lecturing as opposed to any alternative style, we assume it would be more effective for us to try to aid the prevalent method of teaching than to try to change the method all together. . In the traditional style, the prevalent method is "… lecture that is so common in college classrooms (primarily teacher-talk)" while the students passively listen (Sutherland & Bonwell 4). The students are then sent to interact with the material on their own.

"Students are simply more likely to internalize, understand, and remember material learned through [the] active…learning process" (Sutherland & Bonwell 3). "Speaking [as] a psychologist who studies learning, I have to say that the first variable of human learning is time on task" (Andserson 3). Davis describes active learning by saying, "students learn best by doing, writing, discussing, or taking action, because active learning situations provide opportunities for students to test out what they have learned and how thoroughly they understand it" (Davis 181). Therefore most of the alternative styles incorporate exposure to information and interaction through application. The student is made more responsible for acquiring the information and the teachers' time is spent facilitating the student's interaction with the material.

Traditionally, lecturing has been used to get information to students. "To teach something new, the instructor must first present the information. This may take a number of forms… the instructor will perform the skills so that students can imitate them" (Alessi and Trollip 60). The LSU Teaching and Learning Center has identified the following problems with this style:

1. Students' attention to what the instructor is saying decreases as the lecture proceeds
2. It is based on the assumptions that all students need the same information presented orally, presented at the same pace, without dialogue with the presenter, and in an impersonal way.
3. Other obstacles: preoccupation with other classes or events earlier in day; emotional moods that block learning and cognitive processing (e.g. frustration with lack of understanding in course); disinterest by students who go to sleep or turn on recorders; feelings of isolation and alienation that contributes to feelings that no one cares about the student or their academic progress; and entertaining lectures that misrepresent the complexity of material being presented (Alessi and Trollip 60).

A traditional aid of the lecturer is that of a tutor. With a tutor the student is able to sort out what went on in a lecture, or get help with specific questions that the lecturer did not have time to answer in the lecture time allotted. Anderson has described the benefits of the private tutor:

> Changing curriculum, lowering class size, and improving teacher quality will do little as long [as] there is less effective time spent learning. In contrast to the general lack of effective educational interventions, it is well known to the intelligent tutoring community that there is one intervention, which can produce enormous achievement gains with time on task fixed. This is private tutoring. In contrast to typical efforts to reduce class size, if we could reduce class size to one the student can get much more effective instruction. If a private tutor is extended to monitoring homework, that time can also be spent with maximum efficiency. The effectiveness of private tutoring does not contradict the importance of time on task but rather reinforces it. What private tutors do is manage the microstructure of learning time to assure it is effectively spent. In contrast to other proposals that requiring [sic] changing societal attitudes in America, personalized instruction is in keeping with the American belief in the uniqueness of the individual.

> Unfortunately, a private human tutor is rather too expensive for the average American and is no conceivable in a public education system (Anderson 3).

Although a personal tutor is effective through micromanaging time on task, they are impractical due to cost.

An alternative to lecturing is active learning. Active learning is a broad term that encompasses any type of learning where there is interaction between the teacher and the student, the student and another student, or the student with the material (Teaching with Style). Some of the more popular alternatives are:

1. Computer-Based Learning (CBL): "The computer is fundamentally different. When properly used as an interactive device, it demands intense intellectual participation rather than passivity on the part of the user. This alters the instructional context, opening the door to levels of effectiveness not obtainable through didactic presentation alone"(Arons 1051). Software and assignments needed for a course are available through a networked computer. This provides a virtual lab experience. The main advantage of this is that students can work at their own pace. They do not feel stressed due to information overload.
2. Group learning: Tasks are assigned which cause small groups of students to interact with the material in a collaborative or cooperative manner. The main advantage is that the student can receive help from his/her peers, and interacts with the material at a social level. "Faculty who use this technique report that students use time wisely, learn effectively from one another, and can concentrate above the din of everyone talking at once. This change-of-pace activity also recaptures students' attention" (Davis 131).

3. Studio Approach: Jack M. Wilson of RPI states that the studio approach consists of interactive work stations for hands on experience, group discussions to talk about and get a better understanding of the material, and teacher's acting as a mentor's guiding the students as opposed to just lecturing. Students acquire information at their own pace, and interaction with the professor is reserved for questions and clarifications. The main advantages of the studio approach are the personal attention from the professor, and the lack of frustration due to information overload (Educational Technology…).

Studies have also found problems with the alternative approaches. The following is a list of these problems from a teacher's perspective:

1. One cannot cover as much content in one class.
2. Active learning requires too much time in preparation for the class.
3. It seems impossible to use active learning approaches in large classes.
4. Materials and resources are lacking.

(Menges & Svinichi 4)

In the book Tools for Teaching, Davis states:

> Researchers in cognitive and instructional psychology, learning, and motivation have proposed various hypotheses, models, and theories about learning, intellectual development, and information processing. This research is reshaping how we think about learning. Above all, learning is an active, constructive process [active learning] that is contextual: new knowledge is acquired in relation to previous knowledge; information becomes meaningful when it is presented in some type of framework, [schema] In addition, the acquisition and application of knowledge benefit from social interaction, [Myers-Briggs]
>
> Questions about student learning can be grouped into four categories:
>
> 1. How do students select, acquire, and construct knowledge?
> 2. How do students integrate and maintain knowledge?
> 3. How do students retrieve knowledge when they have to use it?
> 4. How do students develop effective learning skills?
>
> (Davis 177)

Active learning, schema, and Myers-Briggs, mentioned above, will be defined later in the paper.

1.2 Learning Styles:

In order to make tutorial software effective we must first understand the current thinking on how knowledge is acquired, and how knowledge is converted into understanding, so as to meet the individual's learning needs. Students can be categorized into learning styles by how they absorb and process information. The software or any instructional method has to be flexible enough that individuals of widely varying learning styles can use it effectively. "(R)esearch provides growing evidence that learning is about making connections-whether the connections are established by firing synapses in the brain, the "ah ha" experience of seeing the connection between two formerly isolated concepts, or satisfaction of seeing the connection between an abstraction and a "hands-on" concrete application" (Cross 5).

Current trends in teaching try to meet the students learning needs by presenting the material in the manner that the student relates to best. These different methods of receiving information are called modalities.

> The acronym VARK stands for Visual, Aural, Read/write, and Kinesthetic sensory modalities that are used for learning information. Fleming and Mills suggested four categories that seemed to reflect the experiences of their students. Although there is some overlap between categories, for purposes of our discussion, they are defined as follows (Fleming).
>
> **Visual (V):**
> This preference includes the depiction of information in charts, graphs, flow charts, and all the symbolic arrows, circles, hierarchies and other devices that instructors use to represent what could have been presented in words.
>
> **Aural (A):**
> This perceptual mode describes a preference for information that is "heard." Students with this modality report that they learn best from lectures, tutorials, tapes, and talking to other students.
>
> **Read/write (R):**
> This preference is for information displayed as words. Not surprisingly, many academics have a strong preference for this modality.

Kinesthetic (K):
By definition, this modality refers to the "perceptual preference related to the use of experience and practice (simulated or real)." Although such an experience may invoke other modalities, the key is that the student is connected to reality, "either through experience, example, practice or simulation" (Fleming & Mills 140-141).

Simply put modalities are the best manner in which an individual is able to take in information, the dominant sensory input device. Multi-modal is presenting the information in a combination of all the modalities listed above. This is similar to multimedia where the information is presented with words both written and spoken, animations, and sounds. Being able to develop a multi-modal piece of software will make conveying information to a wide variety of students more effective.

Once a student is comfortable in receiving the information, we must then understand the many ways in which it is processed. In the most basic sense we all do a similar type of preliminary processing. We try to relate new information to our current framework of knowledge by figuring out how this new information is similar to what we already know and understand. "Schema" is an internal representation of the world, an organization of concepts and actions that can be revised by new information about the world. The schema relates new information to past information so we have a basis to work with. "The schema is a working structure, changing and growing throughout life. Each new event, filtered by perception into the schema, is organized and connected to the existing structure to create meaning" (Cross 8).

> What students can learn depends... on what they already know. It is easier to learn something where we already have some background than it is to learn something completely new and unfamiliar. For example, advanced courses in a subject are often easier to teach and to learn than introductory courses. Cognitive theory would explain this paradox by observing that if the schema is very sparse with respect to a particular subject, connections are hard to find and to make, whereas if the schema already has a dense network of vocabulary, terms, and concepts, it is easier to make the connections that constitute learning (Cross 9).

A good professor or tutor will work off some type of assumed base knowledge and relate new information to it, so that a student is not totally lost. The idea of software as a tutor is the same. The software should be programmed with some level of background knowledge so that the cyber tutor can also relate any of the fundamental information. John R Anderson has been working in the field of computerized tutors for twenty years advancing the understanding of what a computerized tutor should be. In "Intelligent Tutoring and High School Mathematics," Anderson states

> We have been working with a style of tutor, which we think is particularly well designed to deliver individual computer-based instruction in the mathematics classroom. Our research on tutoring systems began with the completion of the ACT theory of cognition. That theory proposes that human problem solving is enabled by a set of production rules. The theory describes how these rules are learned and how they are executed in the process of solving a problem. The theory makes strong claims about how problem-solving skills like those in mathematics are learned. The theory can be turned into a set of strong prescriptions for instruction. In 1984 it seemed that the obvious vehicle for delivering these prescriptions was the intelligent tutoring paradigm, which had been evolving in artificial intelligence as a way of getting computers to interact with students much as private human tutors interact with students. Merging our cognitive models with the intelligent tutoring methodology had the promise of providing a demanding test of our cognitive theory and making substantial educational contributions.
>
> Over the eight years that we have been working on the topic we have developed a highly articulate approach to tutoring, which we call *model-tracing tutoring*. The basic premise of the tutoring approach is to develop a cognitive model of how the student should solve problems and use this model to interpret the student's problem-solving behavior and to guide the student through the curriculum. This cognitive model is represented as a set of production rules. This cognitive model represents an "ideal" that we want the student to achieve. It should be capable of producing any acceptable solution path for a problem. We supplement this model with some of the bugs that students are observed to make. We use this cognitive model to interpret the student's problem solving behavior. When the student makes errors we can interpret these errors and provide appropriate feedback. When the student asks for help we can propose an appropriate path of solution. The key to the model-tracing methodology is the ability to interact with the students at this step-by-step grain size and interpret their behavior in terms of cognitive rules.
>
> We have had some success with this methodology and have followed up that success with some research trying to identify what determines its success. It seems that there are three key factors: (1) Most important is the creation of a

successful cognitive model and communication of that model to the student. The tutor, if well designed, facilitates the communication of the model but we have gotten partial success communicating theses models off-line with just verbal instruction. (2) Critical to minimizing learning time is to have some means of protecting the student from the potentially devastating cost of errors. In untutored environments students can spend hours on problems, which can be done in a few minutes with a few well chosen pointers. (3) Critical to a successful growth of knowledge in a course is the ability to monitor the students' acquisition of individual rules in the tutor and only promote students when they have mastered these rules. The cognitive model provides a psychologically viable analysis of the skill into individual components. The process of following the growth of rule knowledge over problems we can call knowledge-tracing to contract it with model-tracing which is following the students' use of rules within individual problems. Our knowledge-tracing capacity enable individualized learning.

By 1987 we had completed three computer-based tutors—one for proof skills in geometry, one for symbol-manipulation skills in algebra, and one for beginning coding skills in the computer programming language LISP. We found that these tutors could accelerate the rate of skill acquisition by as much as a factor of three. This result, which has been many times replicated, remains our major finding: Our tutors dramatically accelerate the learning of a curriculum by optimizing the learning process through individualization of instruction.
(Anderson 3-4)

The Architecture of Cognitive Theory (ACT) is an important way to model in software the way the mind works and would make a good guide toward our goal for DElab.

Now that we have a basic idea of how the processing begins in general, we need to look more specifically at the complex processing that is done at the individual level. To do this we are going to look into two generally accepted theories of how an individual processes information.

The first theory is actually two separate theories but for the purposes of our research we are going to group them together. They are the Perry / Belenky theories of learning. Both Perry and Belenky studied groups of people and drew conclusions about how each group learned. The difference between them is that Perry studied all or mostly males and Belenky studied females. Perry found that males tend to want information presented in a yes or no answer fashion (Dualism) where as teachers understand the subject in many different shades of gray

(Multiplisum). These two different viewpoints tend to impede the learning process because neither the student nor the teacher can understand where the other is and what they need to make a connection. Belenky found that women tend to categorize knowledge by what authority we place on it i.e. received knowledge (as told to you by someone), subjective knowledge (learned by one's self), and procedural knowledge(obtained through a repetitive process). It's not just about the knowledge but what one thinks about the knowledge that gives it value and therefore it is retained (Davis 178).

The other way in which we are going to explore the way people process information at a higher level, is to break them up into personality types. Once a person is categorized we will then be able to see how the personality types influence the way in which they learn. The Myers-Briggs type indicator is widely used to discern a student's personality type. There are four axies of this type indicator. For each type there is a specific way that people learn best. They are as follows:

Extroversion or Introversion
1. An extravert learns by interacting with the material presented externally.
2. An introvert learns by passively observing the material and reacting with the material cognitively.

Sensing or Intuition
1. Sensing is learning by using the five senses.
2. Intuitive people seek out patterns and relationships among the facts they have gathered.

Thinking or Feeling
1. Thinking refers to choosing to decide things impersonally on analysis, logic, and principle.
2. Feeling refers to the focus on human values, and the need to make decisions or arrive at judgments.

Judging or Perceiving
1. Judging people are decisive, planful and self regimented.
2. Perceptive people are curious, adaptable, and spontaneous.

(Ditiberio & Hammer 6)

By understanding all the learning types, we may be able to design a tutorial system, which incorporates many of these learning types that will accommodate a wider range of students' learning needs. Vincent relates Gardner's theory of multiple intelligences: Multiple Intelligences are a combination of both modalities and Myers-Briggs.

> The seven intelligences are Linguistic, Logical-Mathematical, Visual-Spatial, Musical, Body-Kinesthetic, Interpersonal, and Intrapersonal.
>
> *Linguistic:*
> Linguistic is the capacity to communicate effectively in writing or orally. Linguistic people like to read, write, and tell stories. They are good at memorizing names, places, dates, and other detailed information.
>
> *Strategies for teaching Linguistics learners*:
> Linguistics learn best by saying, hearing, and seeing words. Have them debate important issues, write instructions for others to follow, explain how to work a problem, and solve problems with a partner. Tools for learning include computers, games, multimedia, books, tape recordings, and lecture.
>
> *Logical-Mathematical:*
> Logical-Mathematical learners have the capacity to work with numbers and engage in higher order thinking. They like to do experiments, solve puzzles/problems, work with numbers, ask cosmic questions, and explore patterns and relationships. They are good at math, reasoning, logic, and problem solving.
>
> *Strategies for teaching Logical-Mathematical learners:*
> Logical-Mathematical students learn best by categorizing, classifying, and working with abstract patterns/relationships. Have them compare and contrast situations, find patterns in problems, create outlines, create time sequence charts, and show cause and effect relations. Let them play logic games and perform investigations of complex cases. They need to learn and form concepts before they can deal with details.
>
> *Visual-Spatial:*
> Visual-Spatial learners think in terms of physical space. They have the capacity to learn through graphic images. They like to draw, build design and create things, daydream, look at pictures/slides, watch movies, and play with machines. They are good at imagining things, sensing changes, mazes/puzzles, and reading maps and charts.
>
> *Strategies for teaching Visual-Spatial learners:*
> Visual-Spatial students learn best by visualizing, dreaming, using the mind's eye, and working with colors/pictures. Have them make visual diagrams and flow charts of facts, create graphs of information, do "mind mapping" as a note taking process, and imagine and draw what they think about a situation. Tools which are useful are models, graphics, charts, photographs, drawings, 3-D modeling, video, television, multimedia, text with pictures/charts/graphics.

Musical:
Musical students show sensitivity to rhythm and sound. They have the capacity to think and express in musical forms. They like to sing, hum tunes, listen to music, play an instrument, and respond to music. They are good at picking up sounds, remembering melodies, noticing pitches/rhythms, and keeping time. They love music but are also sensitive to other sounds in their environment.

Strategies for teaching Musical learners:
Musical students learn best by rhythm, melody and music. They learn better with music in the background. Have them learn through songs and jingles (turn lessons into lyrics), use music to reduce stress, create songs/rap to learn information, and watch films/presentations using multimedia. Tools include musical instruments, radio, stereo, CD-ROM, and multimedia.

Body-Kinesthetic:
Body-Kinesthetic students have the capacity to use body movement in learning and expression. They like to move around, make things, touch, talk, and use body language. They are good at physical activities (sports/dance/acting), and crafts.

Strategies for teaching the Body-Kinesthetic learner:
These people learn best by touching, moving, interacting with space, and processing knowledge through bodily sensations. Have them learn by performing the job, acting out a situation, demonstrating to others, and simulating real situations. Tools include equipment and real objects.

Interpersonal:
Interpersonal learners have the ability to understand and interact with other people. They like to have lots of friends, talk to people, and join groups. They are good at understanding people, leading others, organizing, communicating, manipulating, and mediating conflicts.

Strategies for teaching Interpersonal learners:
These people learn best by sharing, comparing, relating, cooperating, and interviewing. Have them role play, conduct interviews/seminars, solve problems in a group, learn through playing games, lead discussions, describe everything they do to solve a problem, and interact with the instructor. Tools include audio conferencing, video conferencing, time and attention of instructor, writing, and e-mail.

Intrapersonal:
Intrapersonal learners have the ability to understand themselves - their interests and goals. They like to work alone and pursue their own interests. They are good at understanding self, focusing inward on feelings/dreams, following instincts, pursuing interests/goals, and being original. They have wisdom, intuition, and motivation, as well as strong wills, confidence and opinions.

Strategies for teaching Intrapersonal learners:

> These learners learn best by working alone, individualized projects, self-paced instruction, and having their own space. Have them keep a journal, reflect on their learning, use guided imagery to solve problems, and write about what they have learned. Tools include books, creative materials, diaries, privacy and time.
> Gardner maintains that all individuals are born with potential in all these areas. The intelligences develop differently depending on the cultural and personal contacts of individuals as they grow. These intelligences rarely operate independently of each other, but are used together and complement each other. Linguistic and mathematical intelligences have traditionally been those that most teachers address. Teachers need to work with as many of the intelligences as they can, and nurture the strengths and weaknesses of students to provide a better learning environment.
> When planning an activity or lesson, teachers might ask these questions: How can I include spoken or written word? (Linguistic); How can I bring in numbers, calculations, or critical thinking? (Logical-Mathematical); How can I use visual aids, visualization, or color visual organizers? (Visual-Spatial); How can I bring in music or environment sounds? (Musical); How can I involve the whole body or hands-on experiences? (Body-Kinesthetic); How can I engage students in cooperative learning or large-group simulation? (Interpersonal); and How can I evoke personal feelings or memories or give students choices? (Intrapersonal)
>
> (Vincent 38-39)

Where each individual student may best receive and process information in these varieties of different manners, tutorial software should attempt to meet the individual learner in as many of these varieties as it is possible to implement.

1.3 Current Trends in Educational Tutorial Software

According to Geisert and Futrell, software learning aids should interact with the student sensing the individual student's needs and responding with some form of constructive feedback (Geisert & Futrell 143). The trends in educational software currently used, all work together to build towards this idea. In order to understand where the current trends in educational software are going, we start at a basic level and build up to something more complex in terms of a tutoring system. Comparing the system to these current trends will serve to give ideas on where the system we are examining could be improved

The most basic educational computer implementation used is the drill."[A] Drill…is perhaps the least sophisticated mode intellectually. Nevertheless, [an] efficient and well-planned drill, presented on an individual basis with immediate feedback reinforcing correct responses and correcting mistakes, is a powerful instructional device. It is important in helping the student build bases," where bases refers to the building blocks of Schema (Arons 1051). The author also goes on to explain that in higher education the drill can be used to build vocabulary, perform exercises, and to practice operations that should become routine.

A simple tutorial builds on the idea of a drill. "Tutorials are programs that…take the role of the instructor by presenting information and guiding the learner in initial acquisition. Drills and games typically engage…the student to practice for fluency and retention. Tests almost always [are used in] assessing the level of learning" (Alessi and Trollip 63). Within the tutorial we will also examine simple simulations. "In a simulation, the computer does not just present predetermined situations. The strength of a simulation is the fact that a computer responds to a student's input; that is, the computer's responses depend on the choices the students make…" (Geisert and Futrell 94).

The next step is the idea of problem modeling. Arons describes a problem modeling process he calls "Socratic dialog." "The term 'Socratic dialog' is used here in its classical sense of using a series of questions to lead a student through lines of reasoning to insights and conclusions"(Arons 1052). His method of getting the student to interact with the material is by posing questions and requiring oral responses. "Most questions require response in words, symbols, or numbers chosen by the student. The computer recognizes a verbal response by searching for specified combinations of key words or phrases. Graphic displays are used throughout: graphs are formed and interpreted; objects move across the screen; flashlight bulbs light with various degrees of brightness; the student uses the built-in pointer to point to places in diagrams in answer to some of the questions or to indicate how to construct appropriate graphs" (Arons 1052). This is an appropriate use of many modalities (multiple intelligences) to engage the user.

Arons' software moves the student from being exposed to information to understanding the material in an incremental process. "A correct answer allows the student to continue in the main sequence; an incorrect answer is dealt with in a remedial sequence. Dialogs usually end with a test, a number of questions being chosen randomly form a large bank of questions, and the student is given an assessment of his performance" (Arons 1052). This assessment of performance provides feedback so that the students know what they understand and what they do not understand, as well as provides and opportunity for reflection. Oblinger et al. stressed the importance of reflection in the learning process "student learning increases proportional to the amount of interaction...[whether it be] student to student, student to faculty, or student to information" (Obelinger). Arons compares Socratic dialog to Anderson's ITS when he says: "From the user's point of view these dialogs are, in some ways, similar to expert systems, which can, for example, lead a medical student through symptoms and test results to diagnosis of an illness. In this case, however, the intelligence of the expert teachers who design the dialog is reflected in the sequence of questions rather than by any intelligence built into the program"(Arons 1052). Socratic dialog models the problem solving process through questions that both lead the student to understanding by relating the new information to what the software assumes the student already knows, and at the same time, checks for that understanding by analyzing the responses to the questions. "The student is led to articulate the physical conclusion without formulas" (Arons 1053). The responses direct the software forward or down a remedial sequence.

At the same time, Arons compares his model to ITS, he offers the following caveats:

> To write an effective instructional dialog, the author or authors must have a clear perception of the basic reasoning processes that will be cultivated in the dialog. The logical structure of the subject matter must be thoroughly thought through at the particular level of instruction. Motivation and plausibility must be carefully built in. The learner's prior experience must be evoked as much as possible. Much writing of dialogs is undertaken without this sort of preparation, and the result is usually ex cathedra presentation, as in most textbooks, rather than a genuine Socratic dialog.
>
> The authors must have knowledge of student difficulties both with the subject matter and with the types of abstract reasoning entailed. Such knowledge is not acquired simply through goodwill, lecturing, or conjecture, but only through personal dialogs with students and through the careful examination of responses on well-designed test questions that probe for lines of reasoning and for understanding rather than for calculational procedures or end results (Arons 1054).

By using problem structures that a student understands, the software introduces new solution methods that are new concepts. This idea relates the software back to the schema where the software is making connections to ideas the student already understands.

Network Based Monitored Tutorial Software monitors the time a student uses the software, and then reports back to the professor. "Personal computers, or terminals connected to central computers, are being used to monitor the progress of students taking self-paced courses; to administer tests that, at the end of each unit, are used to ascertain whether a student is ready to proceed; and to provide tutorial help and drill, relieving instructors of this repetitive activity, and freeing them for more sophisticated and less routine work with their students"(Aarons 1052). Students may need some form of motivation in order to do something on their own, and if a student doesn't use the software it cannot help the student learn. "[H]ighly motivated learners learn better than poorly motivated ones and not much…is likely to occur if one is not intellectually engaged with the task" (Software Goes to School 10). So, if a student is being graded on their performance with the software then that will encourage the student to use the software.

As the complexity of the tutorial software increases it will be important that it can "think" on its own. Using a cognitive model, programmers have been able to get software to accomplish this (Anderson 3).

The highest level of complexity of tutorial software created so far is a mix of all the previous practices and ideas. An Intelligent Tutoring System is able to "think" on it's own. By working with and testing the student, the system can actually learn the students learning style, and convey information to best suit them.

"A special advantage of computer-based model systems is that they permit the linking of multiple representations…for example, the verbal, numerical, pictorial, conceptual, and graphical" (Software Goes to School 120). Arons' comparison of his simpler-to-implement software to the more complex and harder-to-implement ITS gives us a more realizable model to attempt to emulate in an improved version of DElab.

1.4 Supplemental Instruction

A potential way to aid DElab in achieving its goal would be to generate additional instructions to guide the user to understanding the material. These instructions could be modeled after a course supplement for the Continuous Time Signal Analysis (EE2311) taught at WPI. This course supplement, called a Discovery Project, (See appendix C) is a tutorial on paper. Within the paper are many of the features and functions described in the three sections above. It does this by: clearly listing its objectives, relating information to an assumed knowledge to build a big picture, providing an alternate view of information covered in the lecture, relating the high level understanding to real world scenarios, explaining what will be covered and how it will be applied, as well as step by step instructions of how to accomplish the fundamental objectives of the assignment.

These instructions include brief software tutorials and screen shots to aid the student in becoming familiar with the software being used. It encourages students to explore certain aspects of software to promote additional understanding. Throughout the assignment the student is required to respond in writing to questions that necessitates understanding of concepts and cause reflection.

The student is required to design solutions based on what they have learned up to that point. The Discovery Project are similar to the real world where there are many ways to resolve a problem and any way that meets the design criteria is correct. The effectiveness of the Discovery Project lies in its ability to tie the information just acquired to the student's schema and structure the new material into understanding by reusing the information. The steps necessary to emulate their solutions in software follow these design questions.

The Discovery Project points out potential pitfalls the student may fall into. Once the student has been instructed to a solution on the software the Discovery Project points out what is significant about the solution as well as what the flaws are and the whys for both. At this point there is an explanation of the results including its meaning. These are put in, to ensure the student is not totally lost in the learning process. If they are incorrectly applying the steps, these explanations give the students a chance to look at their answers and ask themselves if this is correct.

The Discovery Project continues increasing the complexity incrementally while always relating the new information back to what you already know either from the Discovery Project itself or from generalized knowledge of the world around us. The Discovery Project steps through the process and points out complexities and pitfalls while posing questions that construct knowledge by expanding the newly formed understanding.

The professor of that course, Richard Vaz, informed us that the author of the Discovery Project Nicholas Arcolano used http://fie.engrng.pitt.edu/fie98/ as a source. Mr. Arcolano was not available to comment on the discovery projects. The site contained papers on the subjects of supplemental instruction and using MatLab as an instructional aid. Other papers available discussed how to implement active learning into courseware.

The term supplemental instruction in the two papers refers to structured supplemental help available to the students outside of regular class hours. These help sessions were guided by a knowledgeable undergraduate who had successfully completed the course. The "[l]eaders are trained to design active learning experiences around the content of the course." "In general, sessions focus on actively involving attendees in an in-depth learning activates of course content. Activities give learners the opportunity to see an alternative presentation of course materials as well as hone learner critical thinking and problem solving skills." "...[T]he goals of [the sessions are] a) making connections between what the student already knows and the new knowledge in order to provide a synthesized knowledge domain, b) applying there new knowledge to applicable domain problems." The students were also involved in "...a questioning activity where the leader asked students various questions witch require both surface level and in-depth knowledge processing. The most important thing about these activities is that the learners are active – this in contrast to the typically passive student lecture experience" (Marra, Litzinger 1,2).

> The design of [supplemental instruction] is based upon a view of how humans develop intellectually from the educational philosophers Piaget and Inhelder. Piaget and Inhelder distinguish between students who can reason at an abstract level verses a concrete level. Many students in college (especially freshman and sophomores) still reason at a concrete level, which means they have difficulty

> processing the unfamiliar information they often receive in lectures and textbooks. They are not typically capable of linking this new information with prior knowledge they have in the domain, and questions that they are capable of asking, tend to be detail-oriented and at a superficial level. [Supplemental instruction] can help concrete thinkers move to an abstract level by providing additional instruction that helps to directly link new knowledge to learners' prior experiences. This additional instruction is critical for concrete thinkers (Marra, Litzinger 2).

A unique aspect of the Discovery Project is that they are not simply a reiteration of material covered in class, but a vehicle for moving the student through what they learned in the lecture and through material that has not been covered in lecture. The teacher spends time ensuring the student understands the fundamentals and leaves it to the Discovery Project to do the rest. Certainly there is help available in the form of teaching assistants (TA) but when students come to the TA they are expected to be able to articulate a question. Just being able to do that means the students have some grasp of the material. To help the student in understanding the TA can place a few well-chosen words to realign the students' thoughts and allow them to continue on these projects of discovery. It is this level of student interaction which promotes learning.

1.5 Computer Implementation

Software design and implementation is key for having a successful program. "The success of an application is largely determined by how easily the application can be learned and used" (Larson 2). Generally, there are two basic components that either impede or encourage usage. If the software is easy to install the student is more likely to use it. Even if the software is easy to install, a non-intuitive interface will impede the students' ability to learn.

"User interfaces are important because end users need them to interact with computer applications" (Larson 22). Ease of use, starting from the installation and setup to the user interface, will keep things simple and reliable, and make the software more likely to be used. "Very simply stated, if the application is not easy to use, some users become frustrated and abandon the application" (Larson 2).

Installation practices vary by ease and affordability. From one end of the spectrum, for ease and relative affordability, WinZip can be used as a self-extractor. WinZip by just changing some settings can go from a compression agent to a self-extractor. The software costs about $30 (www.winzip.com). A negative aspect of this is that there are a few manual steps the user has to input them self.

In the middle is Wise. "Wise Solutions offers products that meet or exceed the demands of system administrators and software developers, delivering superior software packaging and installation solutions. Whether a simple setup tool or an advanced process-driven repackaging solution is needed, Wise provides the perfect combination of power, ease of use, return on investment, versatility, and customer satisfaction" (www.wise.com).

On the high side of the spectrum is a program called Install Shield. This does everything for you, finds all the separate folders for all the extensions, and even finds the base program on its own (www.installshield.com). The negative aspect of this product is that it costs about $2000 for one license.

"The user interface is one of the most important components of any computer system, the success of a computer system is often dependent on how easily the user can learn and use the user interface" (Larson viii). "The user interfaces to many powerful tools are cryptic at best.

The manuals are often written for advanced users and can be difficult for beginning engineers" (Tilburry & Messner 1). The interface's main purpose is to act as an interpreter between the user and the software. A "user friendly" interface is important for tutorial software to be effective. The interface should present information to the user in a form that is easy to understand. Input areas for the user should be clearly presented and define, making the interface more intuitive. If the user can easily understand and work with the interface then the user (student) will be able to learn and understand the objectives of the tutorial.

1.6 Human computer user interface

If a piece of software cannot be utilized by the person for whom it is intended, then it is as if that software was not developed at all. According to Maolich and Nielsen "(a)ny system designed for people to use should be easy to learn and remember, effective, and pleasant to use" (Molich 338). This software evaluation will be focused on DElab which is a computer application meant for education purposes in a Differential Equations class. "The design of software for learners must be guided by educational theory"(Soloway 1). "Much evidence exists about structures applicable to human memory and thought. It not only comes from personal experience introspection-based examples of how people are dependent on structure in their daily lives; it is also confirmed by a the pertinent literature and psychology in cognitive science" (Treu 1994, 59).

1.7 Human computer interface design (HCI)

The purpose of a human-computer interface is to convey information from the user to the computer and from the computer to the user. User interface design is an attempt to standardize these interfaces around the skills and expectations of the user. "Since the human mind has such constraints, HCI researchers should realize that the computer interface must be structured accordingly, that is, to present information that is not overwhelming and that it must be organized in a manner conducive to being followed. In other words, the logical next step is to enable and encourage the human and the computer to become suitably interdependent" (Treu 1994).

True defines Human-computer interaction (HCI) as "the combination of physical, logical, conceptual, and language-based actions between a human user and a computer, toward achieving some purpose" (Treu 1994).

There are two types of User Interfaces (UI), command line entry and graphical user interface (GUI). Command line entry is a prompt at which the user must enter one of a limited number of cryptic commands in the correct syntax for the software to initiate an action (Treu 1994). GUI is based on concepts embodied in icons that the user points to and clicks on an icon which stands for an action the user would like to perform.

There are as many different rules or standards for a good user interface as there are authors on the subject. Molich and Nielsen provide a list of nine that are consistent with most lists. Their list of usability considerations in a good user interface are: simple and natural dialogue, speak the user's language, minimize the user's memory load, be consistent, provide feedback, provide clearly marked exits, provide shortcuts, provide good error messages, and error prevention (Molich, 339). These are presented here as a good summery of UI standards, an expanded list of considerations (46) from Treu's "User Interface Evaluation" is given in the appendix A.

A good user UI design will be associative, have limited conceptual distance, and provide structure of object actions. To be associative the U.I. will make clear the available tasks that can be executed by the user. Conceptual distance is the difference both in the users mind and geographically between what the user expects and what is actually present in function and form. Keeping the conceptual distance between what the user expects to occur from activating an object action, i.e. clicking a button or filling in a text field, and what really occurs to a minimum does this. In an associative UI, one that keeps the geographic conceptual distance to a minimum, the icons or text fields that made up the object actions are located in groups or chunks so that associated tasks and concepts are kept together.

1.7.1 User Interface Evaluation

Types of evaluation are: heuristic, comparison against guidelines, cognitive walkthrough, and usability testing. Heuristic evaluation is done by an expert in HCI and puts all of the users knowledge and experience into determining if the U.I. is adequate for the intended user called the target user, to use the software. Comparison against guidelines is when the observer acts as a surrogate user to evaluate software using design guidelines that represent various features that should be present in software (Treu 1994). Cognitive walkthrough is "Using the interface developers to 'walk through' the interface in a model-specific, task oriented manner; given an understanding of the target user's goals and knowledge, the evaluators compare these against the interface's behavior; discrepancies between what the interface actually does and what the user would expect are recorded" (Treu 1994). The walk through refers to using a mock up of the U.I.

to represent what the user would see and describing the next action taken for those evaluating the software. Usability testing is using empirical tests in real would situations and recording any difficulties encountered by the user.

The user interface should be tested for its ability to allow an inexperienced user to use the software effectively. During evaluation the software should be checked for different user commands that accomplish the same task or multiple uses of the same keyword or icon to accomplish different tasks (Treu 1994, 36). The software will be evaluated on its ability to recover from user error and redirect the user to the correct way to accomplish what a task (Treu 1994, 24).

The guidelines to be used in the evaluation will be selected from Treu's text on UI evaluation. His textbooks are based on the works of many experts in the field as well as his own works (Shneiderman). Treu's text's on user interface design is based on his work from such frequently cited papers (Treu 1992) He is also frequently quoted by other authors (Blandford)

1.8 Summary of literature review

A software learning aid should be effective if it meets a majority of the following criteria and is also utilized in a manner that promotes interaction with and reflection on the material presented. This is a high level list of the criteria:

- Ease of installation and use: how easily the user can acquire, install and use the software at a proficient level.
- Intuitive user interface: that promotes the user's comfort and productivity while using the software.
- Multi-modal/multi-media: application presents the same information using different representations: pictures, text, sound and video.
- Online help: needs to be easy to access and navigate as well as provide helpful information to address the user's needs.
- Model problem solving: In order to promote learning, a learning aid should be able to walk the student through the necessary steps to do a problem while also pointing out the thought process that will lead them from one step to the next.
- Active modeling: to a learning aid's ability to engage the user by allowing the user to manipulate different aspects of what is being modeled and view the results.
- Model cognitive process: the software is aware of the thought processes needed to learn and actively models the user in the learning process to remeditate when necessary.

Gardner's theory of multiple intelligences on page17 in the section on learning styles can be viewed as a synthesis between Myers Briggs and modalities. This simplification combined with the premise that a good learning system should attempt to meet as many of the student's learning needs for as many of their multiple intelligences as possible further defines a large portion of the criteria. The rest of the criteria can be fleshed out by adding to this the understanding gained from Anderson and Arons in addition to that of the Discovery Project's integrated with the sensibilities discussed in current trends for educational software, computer implementation and user interface.

The learning environment needs to be interactive and also have multiple stimuli utilizing auditory, visual, read/write, and kinesthetic/tactile senses. In order for the student to acquire knowledge the information first needs to be presented in a manner that suits the learner (multi-model). For an auditory learner the information should be presented as auditory stimuli, it should speak and respond when spoken to. For visual learners the information needs to be in the form of visual cues as well as words, video, charts, and pictures. Visual learners require the information in a visual-spatial form, with the information presented in a visually stimulating manner through colors and pictures that change in a spatially interactive manner. Read/Write learners interact with the material through reading and writing. For kinesthetic learners who receive information by touching it in a real or simulation manner the learning environment needs to be of a form that provides this for them. Because all learners learn though some combination of these learning modalities there is a responsibility to provide a learning environment that has all four types and as many of the subsets as is practical.

Next, the reason for receiving the information has to be significant for the information to become knowledge. Time on task and amount of interaction with the material is the result of placing a value on the information. There needs to a reason to spend time on task, similar to a dialog which prompts visualizing or response and relates new information to previously existing knowledge or understanding

To both motivate learners and provide a context for learning there needs to be stimulus in the form of real situations, active learning, in order to connect the material to the bigger picture, ones schema.

To integrate and maintain knowledge, the knowledge needs to be practiced. This can be accomplished through drills or execution of ACT production rules. What is really important is continued interaction with the material without time lost to frustration so your time on tasks is effective. It is in this phase that the leaner integrates the information and understands the concepts at a high level.

Of the criteria, the ones that should be most important are the ease of use and intuitive user interface. If a piece of software were to meet all but these two criteria it would not be an effective learning aid because it would not be used. If a piece of software is difficult to use the user will become frustrated and stop. For this reason the software should be evaluated for it's ease of use and user interface.

The exposure to information and concepts in a manner that will allow the concepts to be understood became the framework for a portion of the questions used to evaluate DElab. This focus exhibits itself in the form of questions on the user interface design.

2 Methodology

The purpose of this chapter is to describe the methods that were used to evaluate DELab. This evaluation was based on testing the following hypothesis:

> The user interface of DELab contains software design features that shift the learner's focus to learning the software rather than learning the mathematical concepts associated with differential equations

DELab is an application developed within a high-level language called MatLab. MatLab provides a user interface that meets most of the HCI requirements leaving the programmer to ensure that the remaining needs are met.

2.1 Selected method of evaluation

Software evaluation is a method for determining if a piece of software meets the needs it is meant to fulfill. User interface evaluation is a method to determine if the software has an appropriate display of information from the computer to the user and an effective manner for the user to submit requests (object actions) to the computer. The following is a list of widely accepted methods for evaluation of the human computer interface that are described in the HCI section of the literature review on pg 29.

a. Heuristic evaluation
b. Comparison against guidelines
c. Cognitive walkthrough
d. Usability testing

A guidelines approach to U.I. evaluation was selected based on its ease of implementation, efficient use of time, and its effectiveness (Jeffries). The criteria draw on established practices by asking two questions about the software and target user: Is the information displayed sensibly? Does it fit the intended purpose (Treu 1994)?

The guidelines selected are a narrow subset of those listed in Appendix A and they are selected based on representing an object action, a selection of a menu, item task or field, in a familiar and recognizable manner. The focus is on visualizable objects and specifically the conceptual distance between the interface structures. The conceptual distance is the user's ability to recognize associations between objects including object actions and their associated meaning and the actions that are taken by the computer. This also pertains to the locations of the object actions and whether they are continuously visible when they would be of interest.

These interface structures, i.e. the DELab windows, should represent knowledge in the patterns of the human mind as well as the computer-based implementation. To do this the software design needs to take advantage of what the user is able to conceptualize and visualize in the interface. This evaluation determined if in any step in a transaction sequence, the data required will be available for the user on the display. It was seen if DELab displays data to users in a directly usable form and does not make users convert displayed data. Effective displayed design must provide all necessary data in proper sequence to complete a task. To work within limited display sizes designers should organize the displays into meaningful groupings. To this end, the evaluation paid close attention to DELab's use of menus and windows. The emphasis was on whether or not the spatial layout of the interface to provides a pattern of displayed objects in a manner similar to the spatial orientation of the user's mind.

2.2 Evaluation of DElab

The evaluation of DE lab was performed using the following guidelines taken from appendix A.

- Is DELab accessible? Does it enable the user to interact via the interface?
- Is DELab associative? Does is providing explicit and implicit association links among objects?
- Is DELab clear and concise, in choice and meaning of vocabulary and other objects used in interaction?
- Is DELab facilitative, making it easy to use, learn, get things done?
- Is DELab learnable, in its use of mnemonics, logical actions, etc?
- Is DELab non-overwhelming in its interaction language, display layout, functions, state transitions, etc?
- Is DELab visualizable and conceivable, does it reinforce conceptualization, imagination, and viewing of objects?
- Does DElab have continuous visible representation of the objects of interest?
- Does DELab require minimal user knowledge to accomplish the indicated tasks?

The response to each of these questions included the following clarifications.

- Descriptive, i.e., how good or bad is it?
- Diagnostic, i.e., what is wrong (or right) with it?
- Prescriptive (or predictive), i.e. how could it be improved (or degraded)?

The evaluation was performed on the first task were the user is asked to use DELab in Chapters 1, 3, and 7 of Davis Differential Equations text . The variety of chapters achieves a cross section of the typical tasks that a user will perform. The first task of each section was selected because if the user has difficulty in the beginning then chances are that the user will give up out of frustration. The evaluation consisted of the tester using DElab as a surrogate user and answering the guidelines questions when each of the three boxes is completed.

When the tests of the three MatLab boxes were completed the data was analyzed for problems with the user interface and this information was summarized. The summery was used in generating recommendations, based on good user interface design, of how DELab could be improved.

3 Summary of Evaluation Results

The results of the evaluation are available in Appendix C in their raw form. The following is a summary of those results organized by type of error.

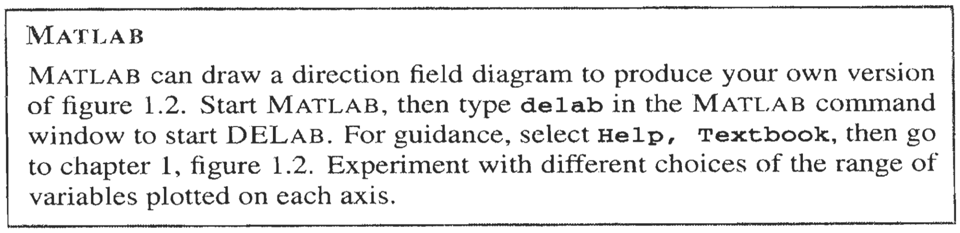
3.1 Overview

The first MatLab exercise in chapters 1, 3, and 7 of Davis's Differential Equations text were evaluated using the selected guidelines described on page 37. As seen below the user is guided by a set of instructions located in DELab's help file. DElab's instructions can successfully lead the user through the necessary steps up to a point, then this stream of successful execution reaches a point of entanglement. As the data illustrates the point of entanglement occurs when the instructions fail and the user interface does not allow the user to continue using DELab. In other words the user does not understand what to do next.

The issues noted below fall into the categories of instruction errors, user interface problems, programming issues, and a fundamental issue about the purpose of DELab.

3.2 Instruction errors

Through the course of evaluating DElab there were repeated errors in the instructions making the operation of DElab difficult or impossible.



MATLAB

MATLAB can draw a direction field diagram to produce your own version of figure 1.2. Start MATLAB, then type `delab` in the MATLAB command window to start DELAB. For guidance, select `Help, Textbook`, then go to chapter 1, figure 1.2. Experiment with different choices of the range of variables plotted on each axis.

fig 1: Chapter 1 MatLab box

In the first MatLab box of chapter 1 (fig 1) the instructions direct the user to start DElab and go to the help file for further instructions (fig 2). These help file instructions lack accuracy. There is an error where a selection listed in the help file as "select rock model" was not present in the menu selections. The closest menu item was "rock with air resistance". Selecting "rock with air resistance" resulted in an error message being displayed and the phase plane graph to display without any data represented on it.

To start DELab, type "delab" in the Matlab command window.
From the DELab window menu bar, select Equation/1st Order.

From the first-order equation window menu bar, select Equation/From list/Rock model. Then select Graphical tools/Direction field.

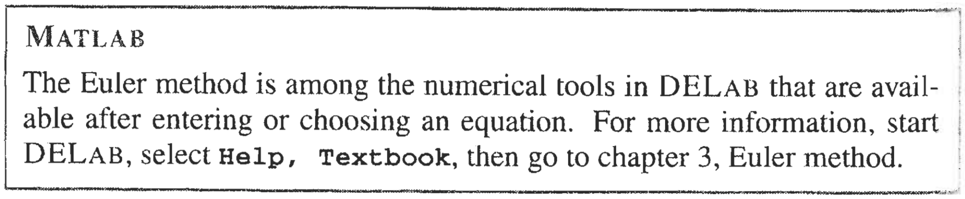
Click the Plot button in the Direction field window.

Select an initial value by editing the value in the Equation window or by clicking on the direction field diagram; click the radio button in the Direction field window for the desired option. If the initial value is being chosen on the figure, click Plot, then position the cursor over the figure.

Note that from this window, the user can adjust the end time, set the maximum and minimum values of v that are plotted, and choose to plot on a new figure or on a existing figure.
Other features can be controlled from the Option item in the menu bar.

Fig 2: Chapter 1 Help File Instructions

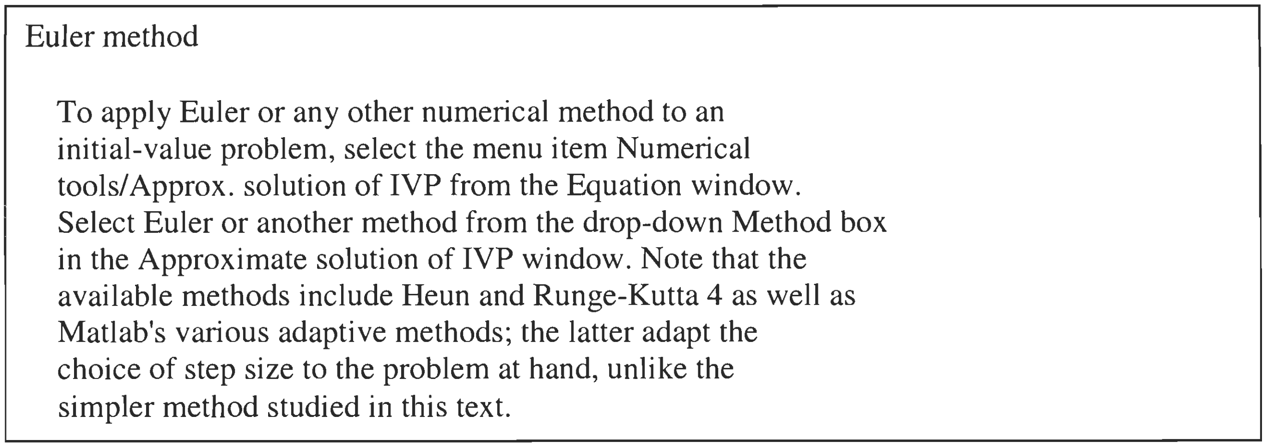
In the second example (see figure “Ch 3’s Matlab box” and Ch 3 help file instructions) there are similar issues. The instructions, located in the help file, for the first activity in chapter 3 are missing the first step which should read “from the DElab window select the *Equation* menu, from the *Equation* menu select *1st order system.*” Instead it reads “To apply Euler or any other numerical method to an initial-value problem, select the menu item Numerical tools/Approx. solution of IVP from the Equation window.” The instruction “To apply Euler or any other numerical method to an initial-value problem, select the menu item Numerical tools/Approx.” is not executable from the DElab window.


MATLAB

The Euler method is among the numerical tools in DELAB that are available after entering or choosing an equation. For more information, start DELAB, select **Help, Textbook**, then go to chapter 3, Euler method.

Fig 3: Chapter 3 MatLab box

In the help file for the first Matlab box in chapter 3 the second instruction ("Select Euler or another method from the drop-down Method box in the Approximate solution of IVP window") is unclear because the object actions are not presented in the order that they need to be executed. The order should read "in the Approximate solution of IVP window select the Method box which will display a drop down menu. From that menue select Euler or another method."

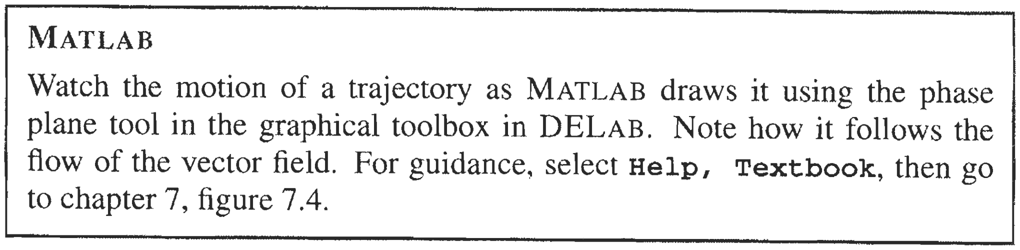

Euler method

To apply Euler or any other numerical method to an initial-value problem, select the menu item Numerical tools/Approx. solution of IVP from the Equation window. Select Euler or another method from the drop-down Method box in the Approximate solution of IVP window. Note that the available methods include Heun and Runge-Kutta 4 as well as Matlab's various adaptive methods; the latter adapt the choice of step size to the problem at hand, unlike the simpler method studied in this text.

Fig 4: Chapter 3 Help File Instructions

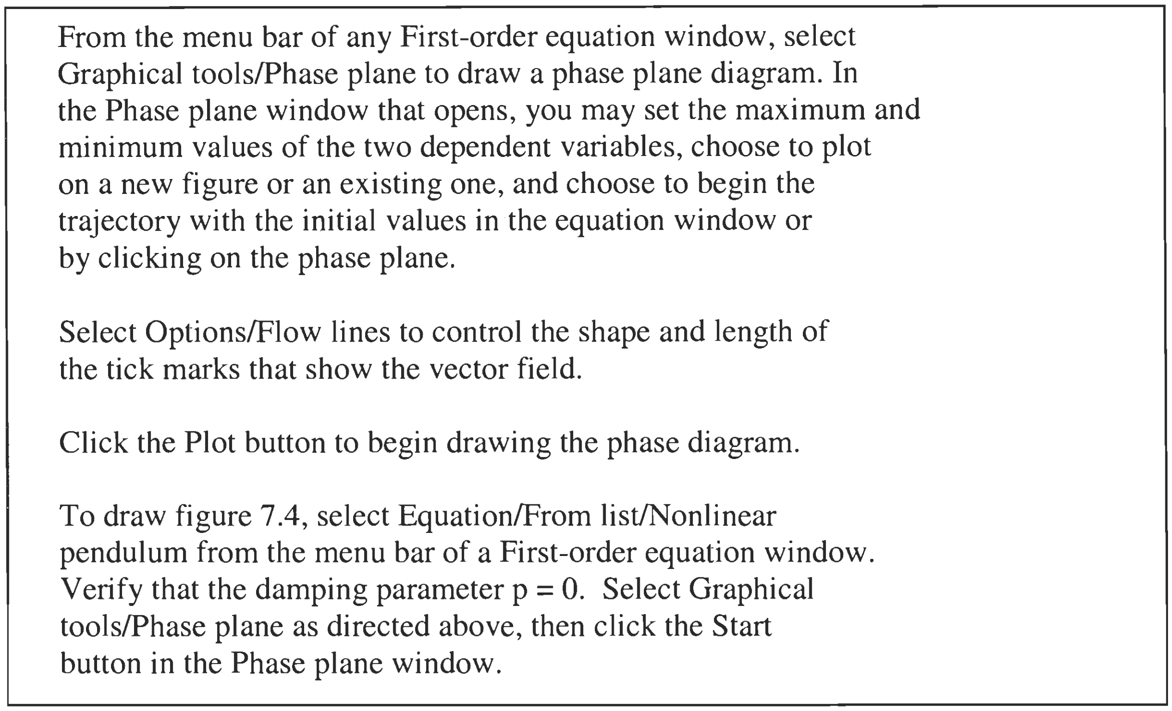
As seen in the above two examples, often instruction steps have to be interpreted to accomplish tasks because the steps are not described using the same vocabulary that is used in the menu selection. The users is reduced to constructing the next step based on what the users see before them.

The following figures support an example of the user attempting to construct the likely next step.


MATLAB

Watch the motion of a trajectory as MATLAB draws it using the phase plane tool in the graphical toolbox in DELAB. Note how it follows the flow of the vector field. For guidance, select `Help, Textbook`, then go to chapter 7, figure 7.4.

Figure 5: Chapter 7 MatLab box



From the menu bar of any First-order equation window, select Graphical tools/Phase plane to draw a phase plane diagram. In the Phase plane window that opens, you may set the maximum and minimum values of the two dependent variables, choose to plot on a new figure or an existing one, and choose to begin the trajectory with the initial values in the equation window or by clicking on the phase plane.

Select Options/Flow lines to control the shape and length of the tick marks that show the vector field.

Click the Plot button to begin drawing the phase diagram.

To draw figure 7.4, select Equation/From list/Nonlinear pendulum from the menu bar of a First-order equation window. Verify that the damping parameter p = 0. Select Graphical tools/Phase plane as directed above, then click the Start button in the Phase plane window.

Fig 6: Chapter 7 Help File Instructions

The following is a block quote from the evaluation results in appendix C.

There were two difficulties in the instructions for the first MatLab box in chapter 7 which prevented the user from completing the task. The initial error is in the first instruction where the user is asked to "select Graphical tools/Phase plane to draw a phase plane diagram." "Phase plane" is not an option of the "Graphical tools" menu. This error was recovered from by opening each of the three options and seeing if the next line of instructions was executable within that window. It was surmised that the correct menu selection was "direction field."

The second error occurred three object actions later when the user was instructed to "select Equation/From list/Nonlinear pendulum from the menu bar of a First-order equation window." "Nonlinear pendulum" was not an option from the "Equation/From list" menu of the "First-order equation" window. This was an non-recoverable error because it was not easily determined which of the menu items should have been selected.

The help file is the only set of instructions available to the user to operate DElab. Again the errors in these instructions make the operation of DElab difficult or impossible.

Furthermore, as seen in the above examples the instructions do not go into great detail about the expected responses to the users' input. No indication is given as to what the user will see when the instruction is executed.

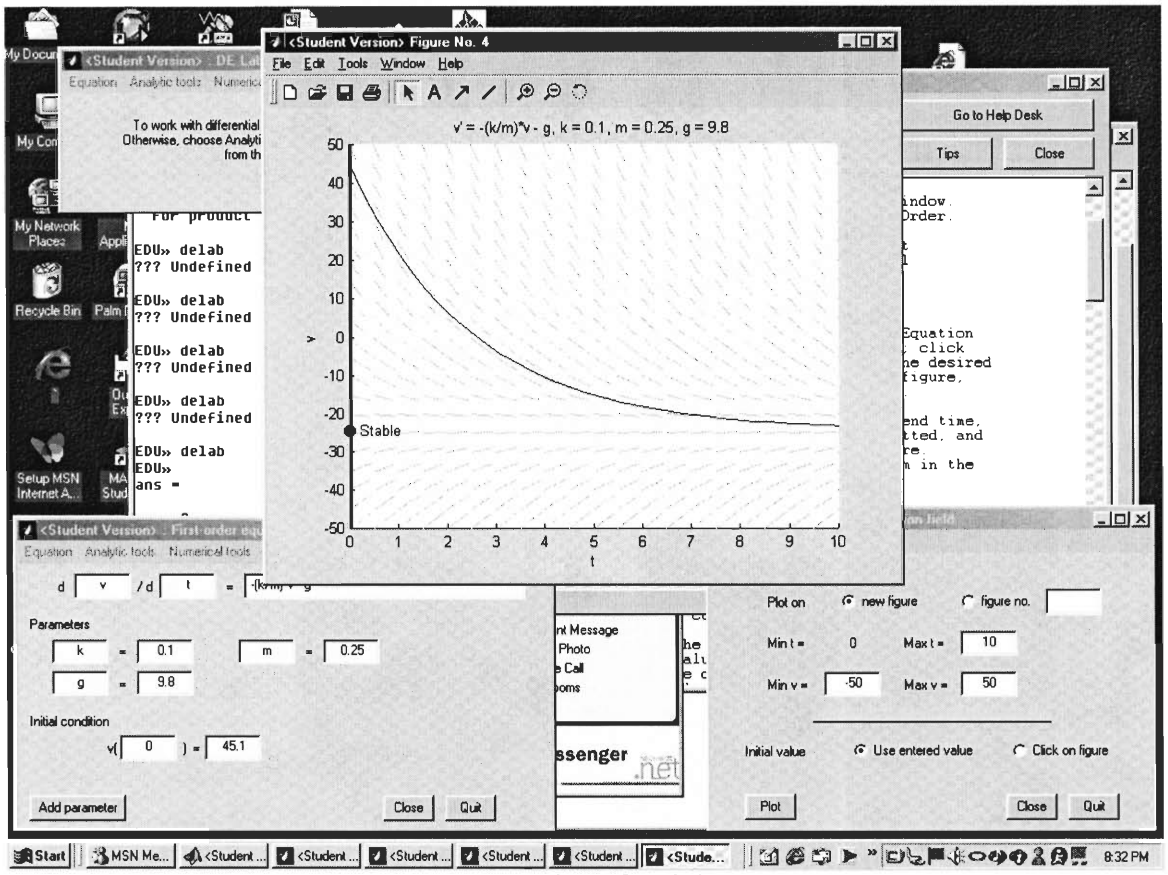
In general, the instructions are not incremental and explicit. When an action object from one window causes a new window to appear, the purpose of the new window is not always clear. The user must rely on the instructions in the help file to interpret what he sees before him. If the help file is ambiguous or in error, the remainder of the user interface is not sufficiently self discriptive to allow the user to proceed unguided. The instructions do not even indicate that a new window should appear.

3.3 User interface problems

The user interface is the means with which the users convert their desire to carry out a task into an instruction the computer can execute. If the user interface is appropriately designed then the user can accomplish the desired outcomes without additional aid.

In DELab "Object actions" (menu items, input fields, or selections i.e. buttons) are obscured by overlapping windows which make them difficult to utilize and their "conceptual distance" (True 1994) is great. This distracts the user from the task at hand and increases the difficulty of using DElab. As seen in the screen shot below there are as many as five or six windows open to complete a single task. Because of this, objects from one window are obscured by other windows. Therefore DElab does not have a continuous visual representation of the objects of interest. A user interface that continuously displayed the objects of interest would rectify this issue.

The interaction language of the menu system is not sufficient to allow the user to navigate the software and accomplish tasks. DElab is not clear and consistent with the textbook or the discipline of differential equations. Along with the misdirection of the instructions noted above there is also the added difficulty of the labels in menu selections not necessarily being named what the user would expect based on the conventions used in the textbook. Specifically, in order to achieve a graph similar to the direction field in the text, the selection necessary in the first Matlab box of chapter 1 would be "Nolan Ryan". This represents a large gap between what the user is expecting and what the interface is requiring. The text refers to this type of diagram as a direction field (slopes of v vs t) predicted by the model $dv/dt = -g$. The lack of standardized or uniform function names in the equations menu of the first order equation window did not make the interface associative enough to overcome the failing within the help file. This represents a large conceptual distance that the user must bridge.



<Student Version> Figure No. 4
File Edit Tools Window Help
v' = -(k/m)*v - g, k = 0.1, m = 0.25, g = 9.8
Stable
v
t
<Student Version> : DE Lab
Equation Analytic tools Numerical tools
To work with differential
Otherwise, choose Analyti
from th
Go to Help Desk
Tips
Close
<Student Version> : First-order equ
Add parameter
Close
Quit
Plot
Start
MSN Me...
<Student ...
8:32 PM

```
For product
EDU» delab
??? Undefined

EDU» delab
??? Undefined

EDU» delab
??? Undefined

EDU» delab
??? Undefined

EDU» delab
EDU»
ans =
```

d v / d t = -(k/m)*v - g

Parameters

k = 0.1 m = 0.25

g = 9.8

Initial condition

v(0) = 45.1

Plot on (● new figure) (figure no.)

Min t = 0 Max t = 10

Min v = -50 Max v = 50

Initial value (● Use entered value) (Click on figure)

Fig 7: screen shot of DElab

DElab also lacks clarity in the menu toolbar. The difficulty with the menu is that the menu items on the toolbar all have the same names in different windows. In these different windows these same named menus will have entirely different sub menu functions. This leads to confusion when trying operate DElab. Because of lack of standardized menu functions DElab can be overwhelming (see fig 7).

The need to switch between windows during operation of a task makes DElab difficult to get used to. The menu system/multiple window layout does not directly guide the user to accomplish a task also when the figure window is opened it covers up the help window which has the instructions in it.

3.4 Programming issue

Also noted were problems with DElab's programming.

When the "Nolan Ryan" selection was executed an error was reported. This error read "Error using = = > eval. Function 'eval' not defined for variables of class 'double'", the software appears to recover and provided a graph but no solution is present. This error message is worded for the programmer or a sophisticated user but not for the intended user of DElab. This represents a large gap between what the user is conceptualizing and what the visual interface is requiring.

3.5 Fundamental issues

Because the instructions are often in error and the user interface is not intuitive enough to allow the user to operate DElab without them, the users' focus is on operating DElab not on learning the concepts of differential equations. For this reason DElab does not create or cement a concept or stimulate imagination. The use of DELab becomes a series of disjointed tasks.

The user interface is not intuitive; it is not clear to the user how to take advantage of the full functionality of this software. Neither the text nor the help file defined for the user what the purpose of DElab is or how is to be utilized.

4 Recommendations and Conclusions

Of importance to improving the user interface and usability of DElab is assuring that the instructions, which presently are located in the help file, are accurate and correct. As seen in the previous section errors in instructions were a major obstacle in using DELab. Vocabulary used in the instructions should be the same as the menu selections in the lab as well as the text. In those places were the vocabulary of the menu instructions differed from the instruction and the text the conceptual distance the users needed to overcome prevented the use of DELab. It should be a goal of the user interface to wean the user off the instructions and get them to use DELab independently. To do this the menu selections need to be self-descriptive and consistent with the users' ability, understanding, schema, and vocabulary.

Furthermore, the instructions need to be sequential and explicit. They should guide the user step by step and explain each step so that the user understands what needs to be done and why these actions are taken. The instructions should include indications of the expected outcome, i.e. new window will open.

Because the instructions are an integral part in using DElab they should be located separately from the help-file. Locating the instructions in a window, file or part of the interface separate from the help file will clarify to the user its purpose and leave the help file for just that - help. Because the instructions are an integral part of using DELab they should be a integral part of the user interface.

Help files should contain descriptions of each field and menu selection as well as descriptions of how to accomplish tasks. The descriptions of tasks should be categorized by types of calculations i.e. first order differential equations.

To improve the usability of DElab the user interface should be restructured. DElab's user interface should guide users through tasks without requiring the use of the help file. Windows should overlap. If multiple windows are to be present for different tasks then the toolbars should have different menu selections. Presently they all say the same thing: "Student version." A better solution would be to integrate all of DElab functions into one window. A user interface structure that uses frames may allow this to be done.

The software redesign should provide a continuous representation of visual objects of interest (Shneiderman 205). Specifically any buttons, text fields, or other object actions that the user needs or is instructed to use should be plainly visible on the screen and not covered up by another window.

At a higher-level DELab should have a clear purpose, this purpose being described in the text or instructions. In this evaluator's opinion DELab in its present form is a graphing calculator for differential equations. A good use for this, in a manner that might aid in understanding the practical uses and purpose of differential equations, would be to incorporate DELab as part of a project-based learning exercise i.e. discovery projects. Another purpose for DELab would be for students to check the results of calculations done by hand. Hand done calculations are the best way for a student to familiarize themselves with the process of doing differential equations (see section on drills and tutorials on pg. 20). Having a method of checking the results of one's work would provide the much-needed feedback to the student in a timely fashion that would aid the learning process. A combination of these two uses for DELab would provide a purpose for DELab and some benefit to the student.

Active-X is a technology that could be used to solve many of the problems existing in DELab including the multiple windows issue and the instructions within the help file. Active-X is a set of controls that could be used to access and run subsets of the Matlab code through a visual basic environment and display them on a Internet Explorer window thereby allowing the use of html features such as frames to be used on the user interface.

In conclusion the software DELab did not meet the usability guidelines selected for use in this project. DELab was compared against a subset of Treu's user interface usability guidelines. In each instance DELab failed to satisfactorily meet expectations set forth by the surrogate user's interpretation of the chosen guidelines. Several shortcomings in the program were identified including instruction errors, creation of multiple windows to the point of confusing the user, and the absence of a menu-naming scheme. To improve DELab several different remedies could be implemented. These would involve [1] rewriting the helpfile so that it includes accurate menu selections, function names, and more descriptive instructions; [2] redesigning the GUI so that objects display in a single window; [3] and, creating and putting into practice a uniform naming scheme for menu selection and object actions. Considering all the above DELab is not void of purpose. With a little bit of modification it could be useful as part of a project based learning experience or as a tool to provide immediate feedback to an independently working Diffy Q's student.

Cited Material

Alessi, and Stanley R. Trollip. Computer-Based Instruction. Englewood Cliffs, New Jersey: Prentice-Hall Inc., 1985

Arons, A. B. "Computer-Based Instructional Dialogs in Science Courses." Science 8 June 1984, Volume 224, Number 4653: 1051-1056.

Bernard, Russell H. Research Methods in Anthropology. New York: AltaMira Press, 2002.

Blandford A. E., Duke D.J. "integrating user and computer systems concerns in the design of interactive systems ." Int. J. Hum.-Comput. ST 46 (5): 653-679 May 1997

Davis, Barbara Gross. Tools for Teaching. San Francisco: Jossey-Bass Publishers, 1993.

Ditiberio, and Allen L. Hammer. Introduction to Type in College. Palo Alto, California: Consulting Psychologists Press, 1993.

Frasson, Gauthier, and G.I. McCalla, eds. Intelligent Tutoring Systems. Germany: Soringer-Verlag Berlin Heidelberg, 1992.

Geisert, and Mynga K. Futrell. Teachers, Computers and Curriculum. Needham Heights, Mass.: A Simon & Schuster Company, 1995.

Larson, James A. Interactive Software: Tools for Building Interactive user interfaces. Englewood Cliffs, New Jersey: Prentice-Hall, 1992.

Marra, and Thomas A. Litzinger. A Model for Implementing "Supplemental Instruction" in Engineering. University Park, Pennsylvania: Pennsylvania State University, 1996.

Mobasseri, Bijan G. Develoment of an Experimental Component for a Senior Communication Systems Course: A Web-assisted Approach. Villanova, Pennsylvania: Villanova University, 1997.

Myers, and Mary H. McCaulley. A Guide to the Development and Use of the Myers-Briggs a Type Indicator. Palo Alto, California: Consulting Psychologists Press, 1985.

Educational technology what's new and how you can use it /
Videotape. Diana Oblinger, Don Jones. Starlink Austin, Tex.

Perkins, Schwartz, West, and Martha Stone Wiske, eds. Software Goes to School. New York: Oxford University Press, 1995.

Sutherland, and Charles C. Bonwell, eds. Using Active Learning in College Classes: A Range of Options for Faculty. San Francisco: Jossey-Bass Publishers, 1996.

Shneiderman, B., Designing the User Interface-Strategies for Effective Human Computer Interaction, 2nd ed., reading MA, Addison-Wesley, 1992

Tilbury, and William Menssner. Development and Integration of Web-based Software Tutorials for an Undergraduate Curriculum: Control Tutorials for Matlab. Michigan: University of Michigan, 1997.

Treu, Siegfried. User Interface Design: A Structured Approach. New York: Plenum Press, 1994.
Treu. "Interface Structures: conceptual, logical, and physical patterns applicable to human-computer interaction." Int. J. Man-Mach. Stud. **37** 1989:565-593

Teaching with style [enhancing learning by understanding teaching and learning styles]. Videotape. Workshop Leader. Tony Grasha. Old Dominion University Academic Television Services, 1996.

Vincent, and Dianne Ross. "Personalize Training: determine learning styles, personality types and multiple intelligences online". The Learning Organization 2001 Volume 8, Number 1: 36-43.

Webster, and Kay C. Dee. Supplemental Instruction Benefits Students in an Introductory Engineering Course. Troy, New York: Renesselaer Polytechnic Institute, 1996.

Non-Cited Material

Anderson, John R. The Adaptive Character of Thought. Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1990.

Anderson, John R. Rules of the Mind. Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1993.

Anderson, John R. The Atom Components of Thought. Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1998.

Belenky, Mary Feild. Women's Ways of Knowing. United States of America: Basic Books, 1986.

Bourner, Tom. Teaching Methods for learning outcomes. Education and Training. Volume 39, Number 9, 1997. 344-348.

Briggs, and Peter B. Myers. Gifts Differing. Palo Alto, California: Consulting Psychologists Press, 1980.

Goos, and J. Hartmanis, eds. Computer Assisted Learning. Germany: Springer-Verlag Berlin Heidelberg, 1992.

Ellis, Allan B. The Use and Misuse of Computers in education. New York: McGraw-Hill, 1974.

Johnson, and Roger T. Johnson. Circles of Learning. United States of America: Edwards Brothers Inc., 1988.

Mckeough, Lupart, and Anthony Marini, eds. Teaching for Transfer. Mahwah, New Jersey: Lawrence Erlbaum Associates, 1995.

Summers, and Richard P. Vlosky. Technology in the classroom; the LSU College of Agriculture Faculty Perspective. Campus-Wide Information. Volume 18, Number 2, 2001. 79-84.

Treu, Siegfried. User Interface Evaluation: A Structured Approach. New York: Plenum Press, 1994.

Appendix A Treu’s list of guidlines

Appendix A

	Adjective: desirable characteristic	Qualified object or intended purpose
P1	Accessible	To enable user interaction via the interface
P2	Accurate, correct	With regard to values of objects output
P3	Adaptable	Interface is changeable
P4	Adaptive	Interface adapts to user
P5	Aesthetically pleasing	In color, appearance, etc.
P6	Associative	Providing explicit and implicit association links among objects
P7	Clear, concise	In choice and meaning of vocabulary and other objects used in interaction
P8	Comfortable	Making user feel at ease (see also P34, P39)
P9	Compatible	With specified user factors
P10	Complete, comprehensive	In functional capabilities, etc.
P11	Congenial	Agreeable, suitable, etc.
P12	Consistent	In commands, spatial location, etc.
P13	Context-providing	In surrounding layers of support for the user
P14	Continuous	In interactive behavior, and output of changing objects
P15	Controllable	Providing user override, etc.
P16	Correctable	When errors occur
P17	Error-preventing or – minimizing	To avoid the need to correct errors (P16)
P18	Expressive	To enable clear, concise communication
P19	Extendible	To add other functions, commands, etc.
P20	Facilitative	Making it easy to use, learn, get things done
P21	Fail-safe	When user commits mistakes, serious consequences are precluded
P22	Feedback-providing	Acknowledging user input, and informing the user of current state
P23	Flexible	In interface layout, devices, scrolling, etc.
P24	Forgiving	After the user makes mistakes or takes other undesirable actions
P25	Helpful	With error messages, explanations, feedback (P22), tutorials, etc.
P26	Learnable	With respect to user capabilities and limitations
P27	Meaningful	Use of mnemonics, logical actions, etc.
P28	Memory-supporting	To supplement or obviate user memory
P29	Network-oriented	To give the user a network perspective
P30	Non-overwhelming	In display density, action choices
P31	Organized or structured	In interaction language, display layout, functions, state transitions, etc.
P32	Powerful	To support more complex tasks
P33	Precise, exact	In relative positioning of objects, etc.
P34	Recoverable	From undesirable actions and states
P35	Responsive in content	With respect to information in system response
P36	Responsive in time	With respect to system time taken to respond
P37	Reliable	In advice, performance, etc.
P38	Simple or simplified	To avoid necessary complexity
P39	Stress-reducing or –minimizing	With regard to user stress
P40	Unambiguous	Similar to P7
P41	Un-constraining	To encourage or enable learning
P42	Uniform	In available functions, etc. (see also P12)
P43	Understandable	In phraseology, expectations, etc.
P44	Usable	Toward achieving desired goals
P45	Versatile	Offering different input devices, etc.
P46	Visualizable, conceivable	Reinforcing conceptualization, imagination, and viewing of objects

Appendix B Evaluation data

The following is the raw data collected from the surrogate user when evaluating DELab against the selected usability guidlines.

Test 1

Is DELab accessible? Does it enable the user to interact via the interface

DELab is accessible, with a few difficulties. Because of the large number of steps necessary to accomplish the first task and a slight error in the instructions where a selection is listed in the help file as "select rock model" and the closest selection was a "rock with air resistance". This difference once overcame, resulted in a graph that was not similar to the text and thereby not explained at that point during the text. A remedy to this would be to correct the instruction in the help file so that a selection would be made that more closely corresponded with page 4 in the text.

Is DELab associative? Does is providing explicit and implicit association links among objects

There are both explicit and implicit association links among objects. These links are obscured by overlapping windows which make them difficult to utilize, there conceptual distance is great. Non-overlapping windows would help to alleviate this issue

Is DELab clear and concise, In choice and meaning of vocabulary and other objects used in interaction?

DELab is not clear, along with the misdirection of the instructions noted above there is also the added difficulty of the labels in menu selections not necessarily been named what the user would expect based on the conventions used in text, specifically in order to achieve a graph similar to the direction field in the text the selection necessary would be "Nolan Ryan". Also of note is that when the "Noalan Ryan" selection was executed an error was reported. This error read "Error using = = > eval. Function 'eval' not defined for variables of class 'double'", the software appears to recover and proved a graph but no solution is present. This error message is worded for the programmer or a sophisticated user but not for the intended user of DELab. This presents a large gap between what the user is conceptualizing and what the visual interface is requiring. More conventional naming based on the text or on what the user would be able to conceptualize it as would help alleviate this difficulty.

Is DELabs facilitative, making it easy to use, learn, get things done?

DELab is facilitative in that the help file instructs the user through the steps necessary to achieve an output. This seems to be the limitation of its facilitative ness as the help files do not give explanations of selections or data entry fields, also the use of the user interface is not intuitive, it is not clear to the user how to take advantage of the full functionality of this software. The instructions that are presently within the help file should probably be located somewhere else possibly under a menu selection of instructions as they are to be utilized in normal operation. The help file needs the addition of descriptions of each field and menu selection as well as descriptions of how to do specific tasks so that a user could look up a type of calculation and find the instructions on how to accomplish that type of calculation.

Is DELabs learnable, in it's use of mnemonics, logical actions, etc?

DELab it is learnable through extensive use of the interface and supplemental instruction from someone familiar with the desired outcomes of the software i.e. demonstration of differential equations. Neither the text nor the help file defined for the user with the purpose of DELab is or how is to be utilized. The first 20 steps have a logical progression to them but the steps that follow require extensive decoding by the user to accomplish the task.

Is DELabs non-overwhelming, in it's interaction language, display layout, functions, state transitions, etc?

DELab is somewhat overwhelming in its interaction language and display layout, functions and state transitions. Above is described some of the interaction language difficulties. The display layout becomes visually distracting and conceptually difficult when each successive step produces a new window.

Is DELabs visualizable and conceivable, does it reinforce conceptualization, imagination, and viewing of objects?

With the above mentioned difficulties it is hard to describe DELaba as visualizable and conceivable because the resulting graph had no real information on its. For this reason DELab did not create or cement a concept, stimulate imagination and performing the steps became a series of random tasks.

Does it have continuous visible representation of the objects of interest?

As seen in the screen shot below there are as many as five for six windows open to complete this single task because of this input from one window are obscured by other windows. Therefore DELab does not have a continuous visual representation of the objects of interest.

Does it require minimal user knowledge to accomplish the indicated tasks?

Each object action is listed in the help file this greatly reduces the amount of user knowledge needed to complete the tasks but considering the difficulties above it also places a great demand that the steps be accurate.

Test 2

These guidelines to be applied with the above understanding and the following three refinements

Descriptive, i.e., how good or bad is it?

Diagnostic, i.e., what is wrong (or write) with it?

Prescriptive (or predictive), i.e. how could it be improved (or degraded)?

Is DELab accessible? Does it enable the user to interact via the interface?

For DELab to be accessible the user would have to be able to accomplish the tasks assigned in the Matlab boxes in the text. This however was not found to be the case. The launching of DELab first required the user to set the path as in the first example. It is not apparent whether this is a difficulty associated with DELab or with the computer with which the test was run, this will have to be checked latter.

Once DELab was successfully launched only five steps (object actions) were accomplished before another difficulty was encountered. The instructions, located in the help file, for the first activity in chapter 3 are missing the first step of "from the DELab window select the *Equation* menu, from the *Equation* menu select *1st order system*." Instead it reads "To apply Euler or any other numerical method to an initial-value problem, select the menu item Numerical tools/Approx. solution of IVP from the Equation window." This instruction is not executable from the DELab window. The next instruction ("Select Euler or another method from the drop-down Method box in the Approximate solution of IVP window.") would be clearer, although maybe not grammatically correct, if the "object actions" were presented in the order with which they needed to be executed. To make DELab accessible the instructions should be checked for errors and corrected. In general the instructions need to be incremental and explicit

.

Is DELab associative? Does is providing explicit and implicit association links among objects?

When an action object from one window causes a new window to appear the purpose of the new window is not always clear. The user must rely on the instructions in the help file to interpret what he sees before him. If the help file is: ambiguous, un-clear, or in error the remainder of the user interface is not sufficiently explicitly or implicitly associative to allow the user to proceed un-guided. The instructions do not even indicate that a new window should appear.

Is DELab clear and concise, In choice and meaning of vocabulary and other objects used in interaction?

DELab lacks clarity both in the instructions in the help file and in the menu toolbar. Some of the difficulties in the help file were discussed above. The difficulty with the menu at is that the menu items on the toolbar will have the same names in different windows. In these different windows these same named menus will have entirely different sub menu functions. This leads to confusion when trying operate DELab. To alleviate some of this confusion either the toolbars should have differently named menus or all of DELab functionality should be integrated to one window.

Is DELabs facilitative, making it easy to use, learn, get things done?

At the initial level of getting started in using DELab is not very facilitative, the multiple windows with the same menu issues and ailing instructions mentioned before make it difficult for the user to get started and accomplish tasks. Once these issues are overcome DELab appears to be a useful tool in generating representations' of data created from differential equation. If the user had some reason to take advantage of this tool he would certainly overcome the difficulties presented. If the user needed to model a design problem that involves a differential equation then he would find DELab useful.

Is DELabs learnable, in it's use of mnemonics, logical actions, etc?

There are two levels with which this question can be asked, one is at the level of the software user interface and the other is that level related to the differential equations. The software user interface would include the tool bar menu items. If the tool bar menu items and sub menu items were the same as within the text, this would promote ease of the user interface. At the differential equations level this would include any button or field descriptions within the window itself. The text field and text field descriptions require repeated use to become acquainted with them. Potentially the user could become accustom to these features.

In regards to logical actions necessary to navigate DELab and utilize its functions the user has to become acquainted over time with the interface but eventually becomes accustom to this interface.

Is DELabs non-overwhelming, in it's interaction language, display layout, functions, state transitions, etc?

DELab is overwhelming when the instructions are incorrect. With out an appropriate guide in the form of help file the interaction language of the menu system is not sufficient to allow the user to navigate the software and accomplish tasks. The display layout of multiple overlapping windows requires the user to understand the function of each window and what object actions need to take place within that window. The functions within the user interface i.e. sub menu items need to be used under the direction of instructions while the user is becoming accustomed to them. If these things are present then DELab it is not overwhelming. In there absence the user is easily overwhelmed.

Is DELabs visualizable and conceivable, does it reinforce conceptualization, imagination, and viewing of objects?

In the user interface portion of DELab it cannot be said that is visualizable or conceivable. This is because the menu system/multiple window layout does not directly guide the user to accomplish a task also when the figure window is opened it covers up the help window which has the instructions in it. A menu system/window layout which could guide the user to or through a task would make DELab more visualizes will and conceivable.

That aside once the user has access to the features and functions of DElab and knows how to use them DELab does provide graphical representations' of differential equations. This aspect of DELab has merit.

Does DELab have continuous visible representation of the objects of interest?

With the exception of the instructions in the help window, a window that is needed for future tasks is not covered up by a new window. Under most circumstances DELab does have a continual visual representation of objects of interest. The exception to this is if more than one graph is generated, the new graph is placed over the old graph. It is up to the user to discover this on his own. A more structured set of instructions that would highlight these features would aid the user

Does DELab require minimal user knowledge to accomplish the indicated tasks?

Because the instructions in the help file are limited and sometimes in error DELab requires the user to know a considerable amount about MatLab and DELab in order to accomplish the indicated tasks. A more complete instruction tool to guide the user through the steps necessary to both implement and understand what they're implementing in DELab would decrease the amount of user knowledge needed.

<u>Test 3</u>

Is DELab accessible? Does it enable the user to interact via the interface?

There were two difficulties in the instructions for the first MatLab box in chapter 7 which prevented the user from completing the task. In the first instruction the user is asked to "select Graphical tools/Phase plane to draw a phase plane diagram." "Phase plane" is not an option "Graphical tools" menu. By opening each of the three options and seeing if the next line of instructions was executable within that window it was surmised that the correct window was "direction field." Three object actions later the instruction was "select Equation/From list/Nonlinear pendulum from the menu bar of a First-order equation window." "Nonlinear pendulum" was not an option from the "Equation/From list" menu of the "First-order equation" window. This was an un-recoverable error because it was not easily determined which of the menu items should have been selected. The help file is the only set of instructions available to operate DELab. When there is an error in these instructions makes the operation of DELab difficult or impossible. Accurate instructions would help. A better solution would be if the instructions were not within the help file and the help file contained descriptions of each of the menu items available.

Is DELab associative? Does is providing explicit and implicit association links among objects?

With regards to user operated features within the direction field window the instructions within the help file were fairly explicit. The lack of a standardized or uniform function names in the equations menu of the first order equation window did not make the interface associative enough to overcome the failing within the help file. The uniform or standardize set of options in the equation field of the first order equation window would help to overcome this issue.

Is DELab clear and concise, In choice and meaning of vocabulary and other objects used in interaction?

Lack of a uniform or standardized list of menu options in the equation menu for the first order equation window was compounded by the error in the help file and prevented DELab from being clear and concise.

Is DELab facilitative, making it easy to use, learn, get things done?

Because the task was not able to be completed it cannot be said that DELab is facilitative or easy to use.

Is DELabs learnable, in it's use of mnemonics, logical actions, etc?

The need to switch between Windows during Operation of a task makes DELab difficult to get used to but one can learn to use this interface

Is DELabs non-overwhelming, in it's interaction language, display layout, functions, state transitions, etc?

Because of lack of standardized menu functions DELab can be overwhelming in its interaction language. The multiple window design also lends itself to certain amount of confusion.

Is DELabs visualizable and conceivable, does it reinforce conceptualization, imagination, and viewing of objects?

This question was not answered due to the inability to complete the task.

Does DELab have continuous visible representation of the objects of interest?

Because the figure window covers up the help file which is still being utilized DELab does not have a continuously visible representation of objects of interest. A window layout that would prevent this from happening would be advisable.

Does DELab require minimal user knowledge to accomplish the indicated tasks?

Because of the error is in the instructions within the help file cannot be said that DELab requires a minimum user knowledge to accomplish the indicated tasks. Accurate instructions and separate instruction and help files would aid in this.

Appendix C Discovery Projects

Worcester Polytechnic Institute Department of Electrical and Computer Engineering
EE 2311: Continuous-Time Signal and System Analysis

Discovery Project I — Fun with Analog Filters

This project will give you the opportunity to explore in greater detail an area of electrical and computer engineering that is absolutely vital to communications technology—analog filtering. Filters are a necessary component of an endless number of products—everything from the most basic analog telephones to the most advanced digital communication systems. There is very little in the world of electrical engineering that would be possible without the concepts of analog filtering we will be examining in this project.

Also, this project will serve as an introduction to a very useful software tool—Matlab. Like many computer applications, Matlab has the ability to do things more accurately in a few seconds than we could do in weeks by hand. It is a powerful and widely accepted application in today's electrical engineering industry, and you will find as you learn to use Matlab that its usefulness and versatility is limited only by your own imagination. Also, we will be learning how to use Simulink, a Matlab module for creating and analyzing system-level block diagrams. Finally, we will be learning how to use Electronics Workbench (a program with which you should already be familiar) to perform circuit analysis in the frequency domain. Used correctly, knowledge of these kinds of computer applications will give you the power to do far more in electrical and computer engineering than can be done without them.

So, our objectives for this project will be:

- To review transfer functions of first-order low-pass and high-pass analog filters.
- To introduce basic operations in Simulink for the use of analyzing filters.
- To examine the decibel unit of measurement and the benefits it provides in plotting transfer functions.
- To introduce the Bode plot and discuss its characteristics.
- To explore basic commands for mathematical operations and data plotting in Matlab.
- To perform frequency-domain analysis of circuits using Electronics Workbench.
- To introduce methods for connecting first-order low-pass and high-pass filters in stages to create a band-pass filter.
- To develop skills in presenting and explaining your work, and in interpreting your results.

As you work through the project, keep in mind the main focus is to gain an understanding that goes above and beyond what you could learn in lectures alone. Throughout the project, you will not only be asked to graph and print the computer solutions to the exercises, but you must also provide your analysis and explanations of this material. At all times, be sure to *write complete sentences that convey your understanding of the material.* Feel free to sketch diagrams and provide specific numerical values, whenever they are requested or they help to illustrate your ideas. Remember—useful though they are, computers are still limited to only the *manipulation* of data—the *interpretation* is up to us.

Also, you are expected to take the time to present your work and results in a thorough, professional manner. Your work will be judged not only on completeness and correctness, but also on quality of presentation. Figures and graphs should be captioned, and any data should be clearly labeled. When appropriate, explain what you are doing and why, preferably in complete sentences. Arrange your work in a logical and highly legible manner. Handwritten reports are fine as long as you can produce a neat and attractive result; if not, you should probably use word processing software.

First-Order Low-Pass and High-Pass Filters

You may recall from EE 2011 that capacitors and inductors are *dynamic* circuit elements, that is, they react to *changes* in current or voltage. This interesting quality allows us two ways to mathematically model the behavior of these elements. In terms of *time*, we can model the response of capacitors and inductors with *differential* *equations*. However, we can also make the journey from the *time domain* into another mathematical realm known as the *frequency domain*—and it is here that we can express the operation of dynamic elements in terms of their *impedance* (*Z*), or their response to different frequencies. Recall that

$$Z_{capacitor} = \frac{1}{j\omega C} \qquad \text{[Equation 1.1]}$$

and

$$Z_{inductor} = j\omega L\,. \qquad \text{[Equation 1.2]}$$

If you have not discovered so already, you will find as you delve deeper into the world of signals that the frequency domain is an invaluable resource. It is often able to provide us with mathematical descriptions of signal behavior that are easier to work with and far more understandable and intuitive than anything we could hope to find in the time domain. Think of it as a taking the "scenic route" to travel from input to output. It may seem like a roundabout way to get there, but it's definitely a more pleasant ride—and often times, it is the fastest route, as well!

Recall that impedance in series and in parallel can be analyzed with the same formulas as resistance. Consider a circuit such as the one below:

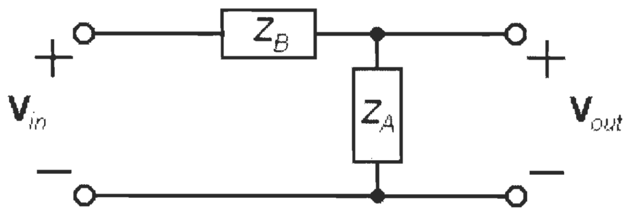


Figure 1: A voltage divider.

The output voltage (as a *phasor*, remember) is simply

$$V_{out} = V_{in}\left(\frac{Z_A}{Z_A + Z_B}\right). \qquad \text{[Equation 2]}$$

You may recognize this circuit as a *voltage divider*—the total input voltage is divided up between the two elements, based on the ratio of their individual impedance to the total impedance. In this example, we happen to be interested in the voltage across the element with impedance Z_A as our output.

The quantity in the parentheses, as you know, is called the *transfer function* of this system, and it tells us how the circuit reacts to different frequencies. Since it is complex, it can be expressed as having both *magnitude* and *phase*. Thus, we can look at either the *magnitude response* (by taking the complex magnitude of the transfer function) or the *phase response* of this system and determine how it affects the magnitudes and phases of sinusoids at different frequencies.

When graphing the magnitude response, sometimes we use a semi-logarithmic scale—a graph with one logarithmic and one linear axis. One reason for this is because we tend to be interested in an extremely wide range of frequencies. Also, the way in which frequency naturally behaves lends itself more to a logarithmic scale than a linear one. Perhaps the best example of this is in the world of music. For instance, at one point on a piano, you can play the note 'A' at a frequency of 440 Hz (880π rad/sec). If you go up by one *octave*, you can play 'A' again at a higher frequency. Now, although it may sound like octaves are evenly spaced in frequency, this is not true—every octave represents a *doubling* in frequency. Thus, at one octave higher, the frequency of 'A' has doubled to 880 Hz (1760π rad/sec), and at two octaves higher, the frequency of 'A' has doubled again to 1760 Hz (3520π rad/sec). What may sound to us like a *linear* scale of frequency is actually a *logarithmic* scale.

As a consequence, we will want to *compress* the intervals between higher frequencies, relative to the intervals between lower frequencies, so that we can examine the magnitude response of these ranges equally. For our transfer functions, we will want to see the behavior of the points on a magnitude response between 1 and 10 rad/sec with about the *same level of detail* as the points between 10^5 and 10^6 rad/sec. If we were to use a linear scale, we would be greatly shrinking the first region and greatly expanding the second—even though the two intervals are of an equal level of interest to us.

Problem 1: What are the transfer functions of the two first-order filters below, in terms of R and C? Show all your work. Make a quick sketch of the magnitude response of each filter on a semi-logarithmic scale for $R = 10$ kΩ and $C = 1$ nF. What kind of filters are they?

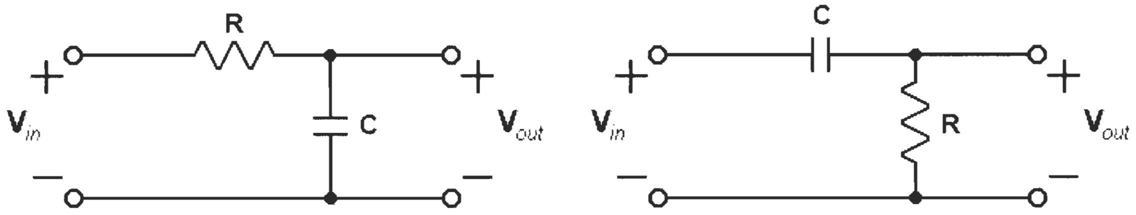


Figure 2: Filters for Problem 3.

You should note on your plots the point where the magnitude response really begins to drop off quickly. This is called the *cutoff frequency*, or ω_c. Coincidentally, we find that

$$\omega_c = \frac{1}{RC}. \qquad \text{[Equation 3]}$$

By substituting this equality into your transfer functions, it is possible to get a transfer function in terms of only ω_c. As you might imagine, this can be quite a handy way to specify the design of a filter (rather than worrying about the specific values for *R* and *C*).

Low-Pass Filtering in Simulink

Let's take a look at what these low-pass and high-pass filters actually *do* to signals in the time domain, using Simulink. Let's say that we have a signal consisting of the single musical tone—we'll use our old friend, the note 'A'. Thus, our signal will consist of a single sinusoid at a frequency of 440 Hz (880π rad/sec). However, our note has some other unwanted signal added to it—another sinusoid with one-tenth the amplitude of our note, at a frequency of 10 kHz (20000π rad/sec). Regardless of where this other signal came from, it's something we don't want. In communications, it is common practice to refer to *any* other signal that is unwanted and unintentionally added to our information signal as *noise*. This is one of the main purposes of a filter—if it works well, it can allow us to get rid of most of this unwanted higher-frequency noise, while changing our lower-frequency signal as little as possible.

It is important to remember in this example, as well as others in this project, that the filters with which we are dealing are *linear, time-invariant* (abbreviated *LTI*) systems. One of the major benefits of linearity in a system is that it allows us to examine all of the components of an input signal *separately*. For this reason, we can observe how our LTI filter affects the music signal and the noise signal individually, and from this information, know how it will affect the *sum* of the two signals.

Open up Simulink by typing `simulink` at the Matlab command prompt. You should see a window containing a tree of Simulink block directories. You can double-click on any of these directories to expand their contents—try the one marked "Simulink" (this is where most of the blocks you will need are located). The tree should expand a sub-selection of libraries, with names like "Continuous" or "Sinks". You can then double-click on any of these, to expand a selection of components. (You can tell the difference between "libraries" and "components" by their icons; one looks like a little block diagram, while the other looks like a small grey cube.) Now, you will want to open a new workspace. You can do this by clicking on the page icon on the toolbar. Drag and drop whatever components you want into the workspace by clicking on their names, and dragging them to the workspace.

There is also a special EE 2311 Blockset, which can be accessed from the "WPI ECE EE 2311" directory, by opening the "EE 2311 Blockset" library. Alternately, you can simply type `ee2311` at the command prompt. By double-clicking on any block itself, you can change the different values associated with that block, and you can also get more information on any block by clicking on the help button. Try it! You will need the following blocks for this exercise (feel free to rearrange the blocks into a configuration that makes sense for this problem):

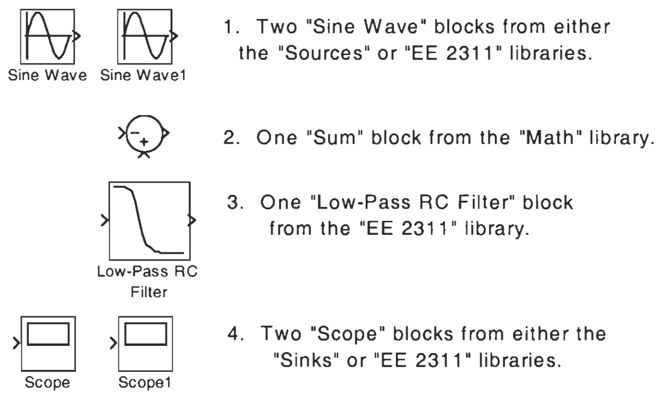

Sine Wave
Sine Wave1
1. Two "Sine Wave" blocks from either the "Sources" or "EE 2311" libraries.
2. One "Sum" block from the "Math" library.
Low-Pass RC Filter
3. One "Low-Pass RC Filter" block from the "EE 2311" library.
Scope
Scope1
4. Two "Scope" blocks from either the "Sinks" or "EE 2311" libraries.

Figure 3: Simulink blocks for simulation of low-pass filtering.

We must connect the blocks in order for the simulation to work. Click on the arrowheads on the right side of the sine wave blocks and drag them over to the inputs of the sum block. One sine wave will be our signal, the other will be noise, and the sum block will add them together. Also, connect the output of the sum to the input of the filter, and, connect the filter output to the input of the scope. Finally, we need to tap another line for the second scope off the line between the sum block and the filter. You can do this by clicking with the *right* mouse button on the line, and dragging it over to the second scope.

You have the option of changing any of the names on the blocks—just click on the title and alter the text as you wish. No two blocks may have identical labels. With different block labels and connected input and output lines, your model should look something like Figure 4.

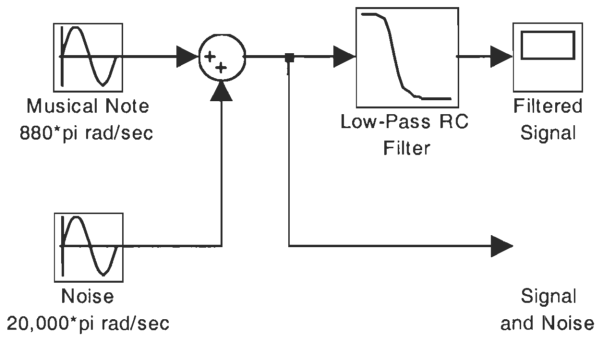


Figure 4: Simulation of a low-pass filtering application.

The next step is setting all of the values for the model—just double-click on each block to set its respective values. The musical note is a sine with amplitude 1 and frequency 880π (enter `880*pi`). The noise is a sine with amplitude 0.1 and frequency 20000π. Both sine waves have a phase shift of zero, and a sample time of zero (this means that Simulink will treat them as continuous). Although it does not need to be changed, take a look at the options for the sum block—it can add or subtract any number of inputs. The last block that needs to be altered is the filter—we have to find appropriate values for R and C.

Problem 2: Recall the role of the cutoff frequency, ω_c, in the first-order filters of Problem 1; the choice of ω_c depends on the application. For the scenario described above, choose a value of ω_c (there is no single "correct" answer!) and explain how you chose it. Then, find some practical values for R and C that would result in your chosen ω_c. Also, using Equation 3, derive the transfer function of a low-pass filter in terms of ω_c (this will be useful later, as well). Then, find the transfer function for the filter with your value of ω_c.

Once you have found values for R and C, you can enter them into the filter block. The final step is to set the parameters of the simulation. Go to "Parameters..." under the menu bar heading "Simulation", or hit <Ctrl-E>. You should see a dialog box of simulation options—look at the "Simulation time" section. You can leave the start time at zero, but since we only want to see about *two periods* of our signal, change the stop time to something more appropriate. The only other item in this menu you may want to look at is the "Refine factor". Simulink plots data points at discrete intervals, but it can interpolate a number of data points, determined by this factor, into the output to produce smoother graphs. You will want this to be fairly high—try a value of 40 for this example. If this value is *not* high enough, then it may be difficult to spot precise things on your scope (such as the high-frequency noise, in this example). However, if your simulation runs too slow, or your graphs seem to be malfunctioning, then lower this number.

We're ready to begin the simulation. Double-click on each scope to bring up the scope view window, and click on the start button on the toolbar to begin. You should see data points being plotted in the scope window. When the simulation is finished, you can click on the binoculars icon on the scope window toolbar to automatically resize the axes.

Problem 3: Estimate the amplitude of the output waveform from the graph (use the zoom function if you need to). From your calculations (using the transfer function), what should the amplitude of the *musical note* be at the output? What should the amplitude of the *noise* be at the output? From your observations and your knowledge thus far of analog filtering, what could we do to improve the filter performance by further reducing the magnitude of the *noise* at the output?

A Practical Filtering Example

Bobby Baritone and Susie Soprano are the hottest singing duo on the market—and you are fortunate enough to be an engineer in their recording studio where they are working on their latest single, titled "D and A". It's a catchy little tune consisting of Bobby singing a D at 247 Hz (494π rad/sec) and Susie singing an A at 1760 Hz (3520π rad/sec). The only problem is, due to an international cotton swab shortage, Bobby has a lot of waxy buildup in his ears today, and the producer can't get him to stop singing louder than Susie. However, as the studio engineer, you've noticed that the amplitude of his voice signal is always *5 times* the amplitude of Susie's (that is, if Susie's microphone outputs 1.0 V, then Bobby's outputs 5.0 V). You offer to build a first-order filter that will attenuate Bobby and Susie's voices differently, so that they appear to be of equal volume on the recording.

Problem 4: Given that your filter must reduce the amplitude of Bobby's voice more than Susie's, what type of first-order filter is called for? What value of ω_c will cause the filter to attenuate Bobby's voice five times as much as Susie's? Hint: create an equality that relates Bobby's output to Susie's (in terms of your filter's unknown transfer function) and then solve for ω_c. After you have found a value for ω_c, show that your filter will attenuate Bobby's voice and Susie's voice to the same level.

Bode Plotting

We have already discussed how our perception of the *frequency* of signals is logarithmic, rather than linear. It is also, in many cases, much more convenient to examine of the *amplitude* of signals on a logarithmic scale. Before we consider this, however, we need to discuss a very important unit of measure you may have already encountered (but may not be as familiar with as you might think)—the *decibel* (abbreviated "dB").

If you have had any experience with decibels before, it may have been in the context of measuring the "loudness" of sounds. What we perceive as "loudness" is a function of two values—the power of the sound signal (usually in watts), as well as the spatial area over which it hits our eardrum (usually in square meters). The problem with representing sound on a linear scale is that our ear can hear over a fantastic range—from sounds as quiet as 10^{-13} watt/m^2, up to and beyond our "threshold of pain" at about 1 watt/m^2. Furthermore, the steps in "loudness" that we can perceive represent logarithmic—not linear—increases in sound power: a raging party at 90 dB may only "sound" two or three times as loud as a crowded restaurant at 80 dB, but it is actually 10 times greater in sound power level.

So, just as with frequency, at some point it was considered a good idea to compare the power of sound signals using a logarithmic scale. The original unit of measure was known as the *Bel*, but this was soon found to be too large for practical purposes, so the standard was changed to tenth-Bels, or *decibels*.

It is important to keep in mind that decibels are a *relative* measure, not an *absolute* one—essentially, we are asking, "how much greater, in a logarithmic scale, is A than B?" Any value to be measured in decibels must first be divided by some *reference value* to which we are comparing it. For example, in order to find the *power* of a signal in decibels, we can use

$$P_{dB} = 10\log_{10}\left(\frac{P}{P_{ref}}\right), \qquad \text{[Equation 4.1]}$$

where the reference power must be agreed upon ahead of time—for example, it is often 1 milliwatt for audio applications. Another common use of decibels is to *compare* two signals. For example, the *gain* of a system is a comparison of the system output and input; gain is a dimensionless ratio that can be expressed in decibels. To find power gain, you could use Equation 4.1. However, we usually know the gain of a system in terms of input and output amplitudes. But since the power of a signal is defined as its magnitude squared, the gain of a system in decibels can found from the input and output magnitudes as:

$$A_{dB} = 10\log_{10}\left(\frac{V_{out}^{2}}{V_{in}^{2}}\right) = 20\log_{10}\left(\frac{V_{out}}{V_{in}}\right). \qquad \text{[Equation 4.2]}$$

We will be using decibels primarily to measure and plot the magnitude response of a transfer function, using the magnitude version of decibel measure in Equation 4.2. Note that a unity gain system has a gain, in decibels, of 20 $\log_{10}$ 1, or 0 dB; therefore, any gain less than unity will appear on our magnitude response plots as *negative* decibels, and any gain greater than unity will appear as *positive* decibels.

Problem 5: If a filter attenuates a sinusoidal input at some frequency f_0 by 6 dB (the input is amplified by -6 dB), how much greater is the magnitude of the output compared to the input? If the gain of the filter at f_0 instead attenuates the input to *one-thousandth* of its original *power*, what is the *power gain* of the filter in decibels at that frequency? How much greater is the *magnitude* of the output in relation to the input magnitude? Finally, recall Bobby and Susie from Problem 4. How much louder, in dB, was Bobby than Susie? (Recall that "loudness" is a measure of sound *power*.)

Armed with this knowledge of decibel measure, we can now plot our transfer functions on axes that are logarithmic in both magnitude and frequency. Such graphs are called *Bode plots* (pronounced \bō- dē\), and we will find as we begin to work with them that there are several other benefits to this method of graphing, beyond the ones we have already discussed.

Plotting in Matlab

This section is intended to be a "crash course" in the basics of Matlab and its data-plotting capabilities. It is *by no means* a thorough introduction. For extra help on Matlab procedures, you should definitely explore the tutorials and help pages on the EE 2311 website. Also, always remember that you can type `helpwin` at the Matlab command prompt at any time to call up the Matlab Help Window, which will give you information on *any* Matlab command. As you go through this section, please feel free to try all of the commands in Matlab, to better illustrate to yourself how they actually function.

The most efficient way to do any problem in Matlab that is more than a few calculations is to use an "m-file". An m-file is a script of Matlab commands that are intended to be executed together and in succession. This allows us to change any parameters and fix any errors without having to re-enter all the code for the entire problem. You can enter the Matlab Editor by typing `edit` at the command prompt. Commands can be entered into the editor exactly as if they were typed in the Matlab

Command window. Also, comments can be added on any line using the percent sign. Any text on a line following the symbol '`%`' will be ignored by the Matlab compiler.

An m-file must be saved in a directory that is part of the Matlab path before it can be run. You can save at any time by selecting "Save As..." under the menu bar heading "File". Save the file wherever you want—as long as you remember the directory. It is necessary then to go to the Matlab Path Browser (type `editpath` at the prompt, and the Matlab Path Browser should appear), and add to the path the directory in which you chose to save your program. Just select "Add to Path..." under the menu bar heading "Path" and enter the directory. *This is important*—your program will not run if Matlab cannot find it due to an incorrect path. To run your program, you can either select "Run" from the "Tools" menu from within the Matlab Editor, or you can simply go back to the command window and type the name of your program, and hit <Enter>. Printing your m-file, as well as any graphs you might create, is simple—just go to the "File" heading on the menu bar of the window whose contents you wish to print, and select "Print...".

Basic operations in Matlab are fairly straightforward. Operators such as the addition sign, the caret for exponential expressions ('^'), and parentheses all perform their usual functions. Keep in mind when performing calculations, however, that Matlab stores *all* of its values in matrices (Matlab is actually short for Matrix Laboratory). Even a scalar value is considered by Matlab to be a 1-by-1 matrix. Although it is this method of data storage and calculation that gives Matlab its power, it is easy to run into problems. For instance, the operator '`/`' is used for *matrix division* (remember linear algebra?), whereas the operator '`./`' (notice the period before the slash) is used for *array* or *element-by-element division*, which is generally what we want to do. Thus, when working with arrays and scalars in the same expression, it is better to be safe than sorry, and use the '`./`' operator. However, when working only with scalar values, it is fine to simply use the '`/`' operator. It is advised that you look up these operators in the Matlab Help Window for more information.

Perhaps the most important operator is the data storage symbol '='. This operator allows us to enter commands such as:

```
apples = 3+4;
```

which will store 7, the result of the calculation, as the variable `apples`. Note that Matlab commands *may* be punctuated with a semicolon, however, this only serves to suppress the output of the calculations. Omitting the semicolon only results in Matlab producing the output, which would appear as:

```
apples =
     7
```

Matlab also has hundreds of pre-programmed functions, including standard functions such as `sin()`, `cos()`, and so on. Some other important functions, which we will use for this project, include `log10()`, which takes the base-10 logarithm of the quantity in parentheses, as well as `abs()` and `angle()`, which calculate the complex magnitude and phase of the quantity in parentheses, respectively.

We have already touched on how Matlab requires arrays and matrices of numbers to perform operations. As you can imagine, then, creating arrays is an extremely important part of Matlab programming. We will examine three methods for creating a data array—two linear and one logarithmic.

The first way to create a data array is by typing a command such as:

```
arrayname = 0:0.1:1
```

where the first number is the initial array value, the middle number is the increment value, and the last number is the final array value. This command, for example, will create an 11-element array where the values count by 0.1 from 0 to 1, and store the result as `arrayname`. Matlab should illustrate that this has been accomplished by its output (since the semicolon was omitted).

Another way to create an array is to use the function `linspace()`. It would be called by typing something resembling:

```
arrayname = linspace(0,10,20);
```

where the first parameter is the initial array value, the second parameter is the final array value, and the third parameter is the desired number of data points. A third way to create an array is using the function `logspace()`, which works similarly to `linspace()`. It is called using a command such as:

```
arrayname = logspace(0,6,100);
```

causing Matlab to create a *logarithmically spaced* array of 100 data points. Note that the range will be from 10^0 to 10^6, not from 0 to 6. The numbers in the first two arguments may be non-integers, to produce any range desired.

Finally, we can plot arrays against one another, using the `plot()` function. Although there are many optional commands involved (allowing you the greatest ability to customize your data presentation as you see fit), the most basic way to plot data is by typing:

```
plot(x_array,y_array);
```

where `x_array` and `y_array` are arrays consisting of the same number of elements. A new window (called a "figure window") should appear with your graph on it. There are several other ways to plot, including `semilogx()`, which plots using a linear x-axis and a logarithmic y-axis, as well as `semilogy()`, and `loglog()`. Also, feel free to look up syntax for the function `subplot()`, which allows you to put several different graphs on a single figure. Finally, although Matlab will automatically create the first figure window for you, you can add another one yourself with the `figure()` command, where the argument is the integer number that will be assigned to the new figure. This is helpful, as it prevents Matlab from plotting a subsequent graph on top of (and thus erasing) a previous one.

Before you continue, try the entering and running as an m-file the following example, which plots the magnitude response of a first-order low-pass filter on a decibel scale. Be sure that you completely understand all of the commands used, and be sure to look up any commands or syntax of which you are unsure in the Matlab Help Window or in any of the EE 2311 Matlab resource materials.

```
     % LPF.m - Plots Bode magnitude response of a low-pass filter.

     omega = logspace(0,6,100);
     R = 1000;
     C = 10^-6;
     omega_cutoff = 1/(R*C);
     transfer_function = 1./(1+(j*omega./omega_cutoff));
     magnitude = abs(transfer_function);
     magnitude_in_dB = 20*log10(magnitude);
     phase = angle(transfer_function);
     figure(1);
     semilogx(omega,magnitude_in_dB);
     figure(2);
     semilogx(omega,phase);
```

Note the element-by-element

If you fully understood this example, you should be ready to do some graphing of your own in Matlab. Of course, now would be a good chance to discuss some of the other interesting characteristics of the Bode plot. The magnitude graph (the first figure) produced by your program should look something like the figure below.

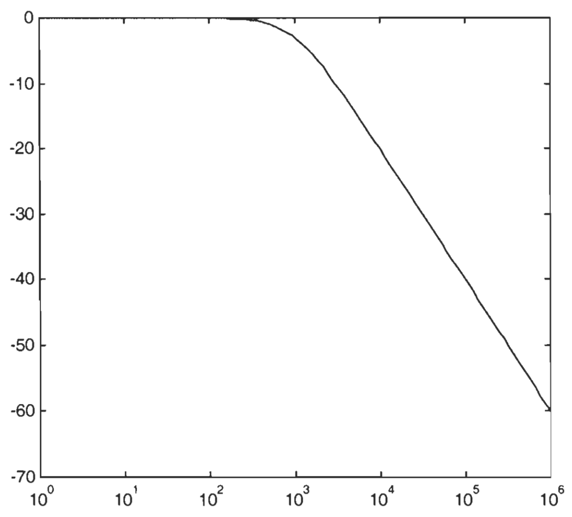


Figure 5: Matlab plot of Bode magnitude response of low-pass filter.

As you can see, the Bode plot of our low-pass filter appears for the most part to be simply two straight lines, with the exception of the region directly around the cutoff frequency (in mathematical lingo, the function "approaches two intersecting linear asymptotes"). This alternate visualization provides us with several benefits. First of all, the Bode plot makes it very easy for us to see that there is little attenuation in the pass-band—gain appears to be close to unity almost entirely up to ω_c. Secondly, we can easily identify ω_c, as the point where the two lines, if they were to continue straight instead of curving, would meet. Observe that the actual magnitude response at this point is not unity, but is instead approximately –3 dB. For this reason, ω_c is sometimes referred to as the *3-dB point*. As we have already seen, ω_c is one of the more important specifications of a filter.

Another visual advantage is that we can easily see how the gain drops after ω_c. In fact, on a decibel scale, the gain after the pass-band decreases linearly, at a rate of –20 dB per *decade* (the interval over which frequency increases by a factor of 10). Moreover, this convenient measure increases predictably with filter order—so a "second-order" low-pass filter would decrease by –40 dB per decade (although this will change when we start using different types of analog filters). At any rate, we can sketch these plots quickly and easily by hand—we know to approximate the magnitude response before the cutoff as a horizontal line, and after the cutoff as a line with a slope of –20 dB per decade.

Bode plots do have some disadvantages, however. Although they can make the cutoff frequency appear obvious, it is difficult to discern the transition-band from the pass-band and stop-band. Furthermore, Bode plots can make it difficult to see other important characteristics of the transfer function, such as distortion in the pass-band or stop-band. Finally, Bode plots cannot show a filter's performance at DC—of course, this is because the logarithm of zero is undefined.

Problem 6: Sketch by hand the Bode magnitude plot of the high-pass filter below, for $R = 100$ kΩ and $C = 10$ nF. You shouldn't need to plot a lot of data points—remember, since it's a first-order filter, it should be just two straight lines on a decibel scale. (Also remember the earlier comment that Bode plots don't show DC, so where your graph starts depends instead on ω_c and the range you've chosen.) Using Matlab, plot the magnitude response, Bode magnitude, and phase response. Print your results. From the transfer function, at what frequency the gain is –3 dB? Calculate the attenuation at $\omega = 10^2$ rad/sec and at $\omega = 10^5$ rad/sec (in dB). Do these values correspond with your plot?

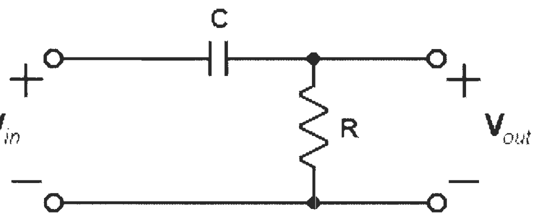


Figure 6: High-pass filter for Problem ***6***

Building a Band-Pass Filter

We have already discussed to some extent low-pass and high-pass filters, as well as a few of their many applications. However, there are many more types of filters that serve a great many different purposes in the world of communications, and one of the most important is the *band-pass filter*. A band-pass filter, as you probably already know, is a filter that *passes*, or outputs with little attenuation, a certain range or *band* of frequencies. We deal with these filters every day—if you've ever played with the equalizer on a stereo or listened to a radio station, you've had experience with band-pass filters.

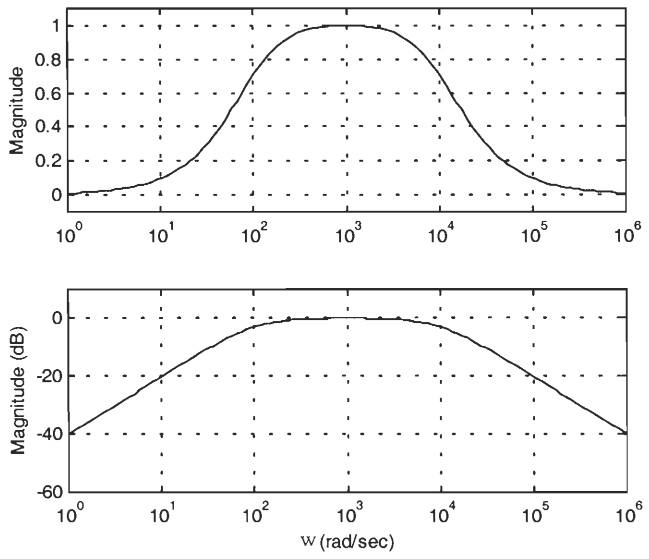


Figure 7: Magnitude response and Bode magnitude plot of band-pass filter.

How do we design a band-pass filter? Well, if we can block all frequencies *below* a certain point using a high-pass filter, and we can block all frequencies *above* a certain point using a low-pass filter, does it not seem possible that we could pass a finite band of frequencies by combining these two filters? From a systems level, the band-pass filter would look something like the diagram below.

$$\begin{tikzcd} {} \arrow[r, "V_{in}"] & \boxed{\text{High-Pass Filter}} \arrow[r, "V_1"] & \boxed{\text{Low-Pass Filter}} \arrow[r, "V_{out}"] & {} \end{tikzcd}$$

Figure 8: Block diagram of a band-pass filter.

Now, we know that the transfer function of each block individually is simply the ratio of the phasor output to the phasor input. That means that the complete transfer function for the band-pass filter should be

$$H_{BPF}(\omega) = \frac{V_{out}}{V_{in}} = \frac{V_1}{V_{in}} \cdot \frac{V_{out}}{V_1} = H_{HPF}(\omega) \cdot H_{LPF}(\omega). \qquad \text{[Equation 5]}$$

In theory, we should be able to obtain the transfer function of our band-pass filter simply by multiplying the transfer functions of the two filters from which it is composed. This raises another important point about the benefits of Bode plotting. Equation 5 shows that the first-order frequency responses must be *multiplied* when represented as the *linear* function *H*(ω). However, as soon as we convert to a *logarithmic* scale (such as the Bode plot), we can simply *add* the magnitude and phase responses to get the total response.

Before we continue, however, let us discuss how we can teach our "old dog" from EE 2011, *Electronics Workbench*, a few "new tricks"—namely, frequency-domain analysis. You should already be comfortable designing circuits in EWB, such as the filter circuit below. Notice that we have added an *AC voltage source* and a *ground*, both of which will be necessary to perform the frequency analysis.

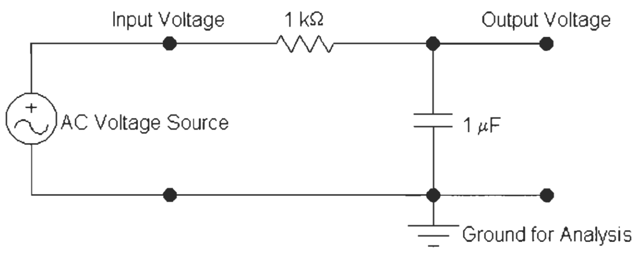


Figure 9: Low-pass filter in Electronics Workbench

The actual analysis is rather simple. First of all, select the voltage source and bring up its properties dialog box. This can be done by double-clicking on the component, or also by selecting "Component Properties..." under the "Circuit" menu bar heading. Once the dialog box has appeared, click on the "Analysis Setup" tab, and make sure that the "Use in AC Frequency Analysis" box is checked. Next, go to the node you wish to consider the output. Call up its property dialog box by double-clicking once again, and click on the "Node" tab. You should see a "Node ID" field, with a number in it. Take note of this number—we'll need it when we go to perform the analysis.

We're now ready to begin. Go to the menu bar, and select "AC Frequency..." under the "Analysis" heading. The dialog box that appears has several options—feel free to set them as you see fit. Take note of the fact that the frequencies are in *Hertz*, not radians per second; you will need to adjust your desired range accordingly. You will want to perform a decade sweep, and you will probably want your vertical scale to be in either linear or decibel units. Finally, you should see a field containing a list of nodes—don't forget to add the one you wish to analyze as your output.

Once you are ready, click on "Simulate". You should be presented with a lovely matching pair of magnitude and phase responses. There are several useful buttons on the graph window toolbar—please take a few moments to play with some of them. Once again, make sure that you realize that whatever values you calculate from this graph will be in units of Hertz, so be sure to convert to radians per second to find figures such as ω_c.

We should now be comfortable enough to begin analyzing band-pass filters. Let us examine the circuit we had proposed earlier—a high-pass filter and a low-pass filter connected together to make a band-pass filter.

Problem 7: Consider the two filters in Figure 10.1, where R_H = 159 kΩ, C_H = 1 nF, R_L = 20 Ω, and C_L = 8 nF. What is the value of ω_c for each filter? Sketch by hand the linear-approximation Bode magnitude response for both filters. If we were to *add* these two magnitude responses together, what band of frequencies would pass? What would be the approximate gain of the magnitude response in the pass-band?

Figure 10.2 shows the band-pass circuit obtained by combining the two filters. Using Electronics Workbench, plot the magnitude, Bode magnitude, and phase response of the combined filter circuit. (Be sure to remember to add an AC voltage source and a ground for analysis.) Print out your circuit and the graphs you have obtained. Does the pass-band appear to contain the same range of frequencies as you had calculated? Remember—the scale is in Hertz, not rad/sec. What is the gain in the pass-band (standard, and in dB)? Is this value close to what you expected? If not, how would you explain your result?

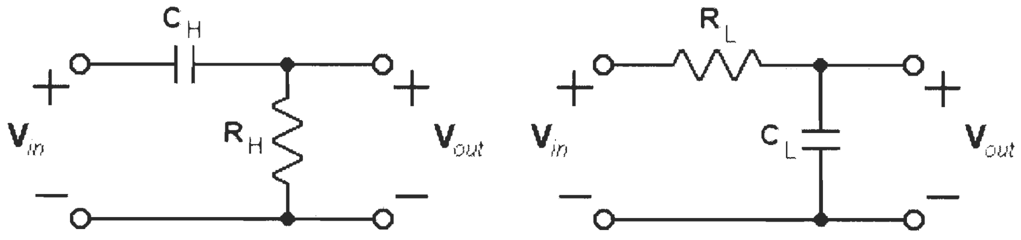


Figure 10.1: High-pass and low-pass filters for Problem **7**.

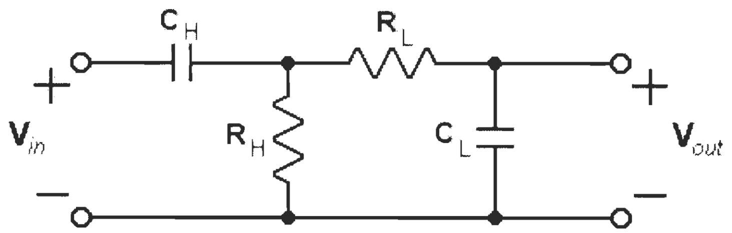


Figure 10.2: Combined band-pass filter for Problem **7**.

The results you obtained for Problem **7**—especially for the gain in the pass-band—should tell you that this approach to creating a band-pass filter has a flaw. The reason that a problem exists is that the low-pass and high-pass sections are attached to one other *directly*. This allows the second stage of the circuit to *load down* the first—an effect you might have seen a little in EE 2011. Because the two stages are not isolated, we must take into account the voltage and current divisions that occur when any elements are combined in series or in parallel—this means that we *can't* simply multiply the two first-order transfer functions, or add their Bode plots. However, there is one simple solution to this problem—using a *buffer* to isolate the two filters. Consider this next problem.

Problem 8: **We have now added an [ideal] operational amplifier to our circuit. What is the gain of the op-amp section of the circuit, V_B / V_A? Using Electronics Workbench, plot the magnitude, Bode magnitude, and phase response of the combined filter circuit. Be sure to remember to add an AC voltage source and a ground for analysis. Print out your circuit and the graphs you have obtained. What is the range of frequencies in the pass-band? What is the approximate gain in the pass-band? In a few sentences, compare your results here to your results from Problem 7, to explain how this circuit behaves differently from the last.**

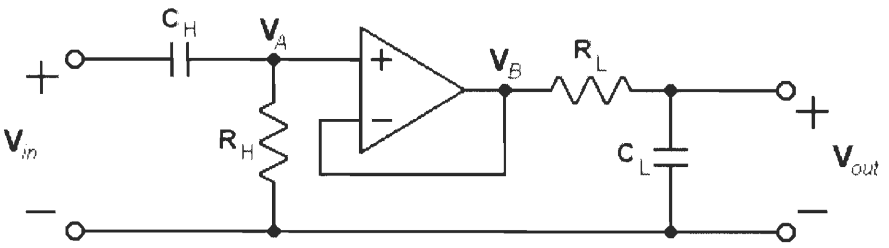


Figure 11: Band-pass filter for Problem 8.

You should be able to see rather easily that the buffer has an effect on the gain in the pass-band. This problem illustrates an important distinction—the difference between *block level* and *system level* design. In most applications, filters are just individual blocks within an entire system of other components, such as embedded processors, analog-to-digital converters, or transmitters and receivers. Often times, it is far less difficult to get the individual parts of this application to work, than it is to get these many different parts to *work together* on the "system level" of design. A circuit can be a lot like a PC in that respect—everything may make sense as individual components, and sometimes it is tough to figure out why they don't just work when you hook them together.

Thanks for Playing, We Have Some Lovely Parting Gifts for You

Well, you've nearly made it through Discovery Project I. That wasn't so bad, was it? (Actually, don't answer that.) Just one more problem to go…

Problem 9: Imagine that among the many things you have been putting off in order to complete this assignment, one of the most important is a long-overdue letter to your favorite high school English teacher to whom you promised to write when you got to college. Since you had promised him/her that you'd share all of the joys and sorrows of studying engineering, you decide to write a letter in which you will teach him/her some of what you've learned about in Discovery Project I.

Write a letter in which you convey to your teacher a basic understanding of filters. Keep in mind that, although very bright, he/she probably doesn't completely understand what "frequency" is, has never encountered resistors and capacitors, and hasn't studied math much beyond high school algebra and geometry. Also keep in mind that they will be far more interested in the "big picture" of what filters are, why they are useful, and what you now know about them, than in the gory details of what you actually did in this project.

Written by Nicholas Arcolano—21 July 1999.
Revised 20 December 1999.
Revised 25 October 2000.
Revised 8 March 2001.
Revised 29 October 2001. (AD)

Discovery Project II — Signals From Scratch

You have already traveled a fair way down the long and winding road of continuous-time signal analysis. You have learned the basics of how we measure signals, how we tell them apart, and how we can classify them. Also, you have learned about special signals, such as the impulse, that can make life a little easier in the time and frequency domains. You have even stared into the gaping maw of convolution, and witnessed what evil lies therein.

But now, you ask yourself, "If it takes me an hour to convolve a rectangular pulse with an exponential, how in the world will I ever be able to do anything *more* complicated??" One of the most important problems we have to solve in the world of communications is the age-old "input-output problem"—that is, how do we find the *output* of a given system, knowing the *input*? So far, however, the only way we have discovered to accomplish this task is through the rigors of convolution, performed in our own familiar time domain. But looking ahead, you think, "*Convolution is such a pain...isn't there a better way?*"

And there is! It's called *Fourier theory*, and if you haven't heard of it already, it is a beautiful and elegant way to analyze signals and systems. To get to the root of this method, however, we need to know: what is it that *makes up* a signal? Just as one cannot understand chemistry without knowledge of the atom, in order to comprehend signal analysis, it is necessary to know how signals are built from basic elements. That is what this second project will examine—how do we actually *build* signals?

So, our objectives for this project will be:

- To present the basics of signal construction by examining the similarities between *signals* and *vectors*.
- To discover how mathematical operation of a *vector dot product* can be expanded into a "dot product" for signals.
- To define the quality known as *orthogonality*, for both vectors and signals.

- To explore a measure of signal similarity called *correlation*.

- To examine the use of an *orthonormal basis*—a set that can serve as the "atomic building blocks" of vectors or signals—for vector or signal construction.

- To utilize an orthonormal basis set to construct a signal, using Matlab.

Part 1: Vectors and Signals, Signals and Vectors

Dot Products and Vector Components

What is a vector? Mathematically, it is an abstract representation of a quantity with multiple dimensions. To represent this quantity, we may use an arrow (graphically), or an ordered set of values (numerically). Unlike scalar quantities such as temperature, vector quantities—things like displacement or force—must be represented in two or three (or possibly even more) dimensions. An important question, then, is to consider what makes a "vector" representation of a quantity so useful to us? One advantage is that we can break down these vectors into their *components*—their basic dimensional elements. By doing this, we can take two seemingly unrelated vectors (such as two forces pulling the same object in two unrelated directions), and combine them. We accomplish this by breaking each vector down into its set of components; effectively, we re-create the two different vectors as sums of differently scaled versions of the *same standard set* of vectors. Then, with the two vectors expressed in terms of the same kinds of components, we add up these separate parts of each vector that are related. If you have ever added two vectors in a physics class, then you have had experience with this operation before.

First, though, we have to define a vector operation that will be useful to us later—the *vector dot product* (also known as an *inner* or *scalar* product). The dot product operation is performed on a pair of vectors, and is denoted by a pair of angled brackets: <**a**, **b**> is read as "the dot product of vectors **a** and **b**". (Note that vectors in this document will be denoted by boldface type, and that |**v**| will represent the length or *magnitude* of some vector **v**.) For any two vectors **a** and **b** with an angle of θ radians between them,

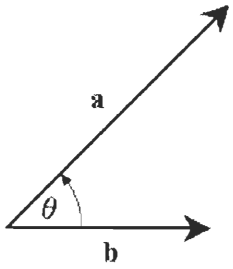


Figure 12: Finding the dot product of two vectors.

$$\langle \mathbf{a}, \mathbf{b} \rangle = |\mathbf{a}| \cdot |\mathbf{b}| \cdot \cos(\theta), \qquad \text{[Equation 6.3]}$$

and when **a** = **b** (the dot product of a vector with itself),

$$\langle \mathbf{a}, \mathbf{a} \rangle = |\mathbf{a}| \cdot |\mathbf{a}| \cdot \cos(\theta) = |\mathbf{a}|^2 \cdot \cos(0) = |\mathbf{a}|^2. \qquad \text{[Equation 6.4]}$$

Problem 10: What is significant about the dot product when two vectors are pointing at right angles to each other? What about when they are both pointing in the same direction? What does a dot product of $-|\mathbf{a}|^2$ tell us? Provide support for your arguments. **[5 pts.]**

So, how do we use the dot product, along with our knowledge of trigonometry, to find the components of a vector? Figure 13 shows a two-dimensional vector, denoted by **v**. As you may remember from physics, it is often the convention to break a two-dimensional vector into the sum of scaled versions of a standard set of two *basis vectors*, denoted by **i** and **j**. These two familiar vectors have some special characteristics that make them a very good foundation to form components. In fact, we refer to the vectors **i** and **j** as the "basis set" for the space of two-dimensional vectors. We will discuss the concept of a basis set in greater detail later; for now, let's review how we derive the components of a vector.

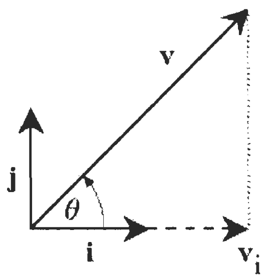


Figure 13: A vector and one of its components.

We should know already that given our "basis set" all we need to completely make up **v** is a certain scalar multiple of the vector **i** that we can add to a certain scalar multiple of the vector **j**. So, all we have to do to break up vector **v** into components is find out how much we need of each basis vector, correct? First, let's try to find out how much of the vector **v** lies along the path of **i**. This will give us $\mathbf{v}_i$—known as the component in the **i**-direction of vector **v**, or the *projection* of vector **v** along vector **i**. Performing some simple trigonometry on the vectors shown in Figure 13 gives us

$$\cos(\theta) = \frac{|\mathbf{v_i}|}{|\mathbf{v}|}, \qquad \text{[Equation 7.1]}$$

and thus

$$|\mathbf{v_i}| = |\mathbf{v}| \cdot \cos(\theta). \qquad \text{[Equation 7.2]}$$

Now, part of the benefits of using **i** and **j** as a basis set is that they are *unit vectors*. Since the length of each vector is unity, it makes it easier to express the component in the **i**-direction of **v** in terms of **i**. For instance, if we want to express $\mathbf{v}_i$ in terms of **i**, then $\mathbf{v}_i$ is simply $|\mathbf{v}_i|\cdot\mathbf{i}$, since $|\mathbf{i}| = 1$. We say that the basis vectors **i** and **j** have been *normalized*—but we will discuss this as well later on in the project.

We can use the unit length property of vector **i** to express Equation 7.2 in terms of the dot product. Since

$$|\mathbf{v_i}| = |\mathbf{v}| \cdot \cos(\theta) = |\mathbf{v}| \cdot |\mathbf{i}| \cdot \cos(\theta), \qquad \text{[Equation 8.1]}$$

then

$$|\mathbf{v_i}| = \langle \mathbf{v},\mathbf{i} \rangle . \qquad \text{[Equation 8.2]}$$

However, what would be the case if the length of vector **i** were *not* unity? Figure 14 shows vector **v** again, but this time the vector we are using for part of the basis set is *not* of unit length (vector **w**). As before, we want to find out how much of the basis vector (this time, vector **w**) is used to make up **v**. However, we will ultimately want to express this relationship in terms of some constant *c*, which represents by how much we would scale the basis vector **w** to make it equivalent to the component of vector **v** along **w**.

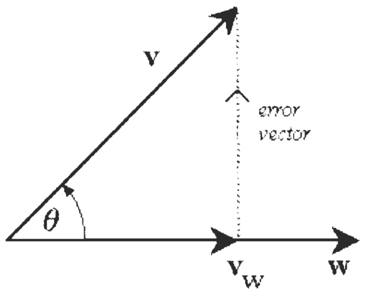


Figure 14: Finding the component of a vector along another vector.

Now, stop and look at Figure 14 and think for a moment. We know that there is some amount of vector **w**, some scaled version of itself, that can contribute to making vector **v**—this has been drawn as $\mathbf{v_w}$. But in fact, we don't *have* to use the vector $\mathbf{v_w}$ as it is drawn above—we could use *any* amount of vector **w** in constructing vector **v**. No matter how long we choose to make vector $\mathbf{v_w}$, there would always be some *other* vector that we would have to add to $\mathbf{v_w}$ to totally make up **v**. We can imagine this other vector—the leftover part of **v** that cannot be represented by $\mathbf{v_w}$ alone—as the vector that we would add to the tip of $\mathbf{v_w}$, to get us back to the tip of **v**. This difference between $\mathbf{v_w}$ and **v** is the *error vector*, represented above by the dotted line.

So if we could potentially use *any* amount of **w** to build **v**, then what's the point? Well, by making the error vector as small as possible, we are creating the *best* $\mathbf{v_w}$—the perfect amount of **w** to try to make up **v**. Think of it as using $\mathbf{v_w}$ to *approximate* **v**; the best possible approximation will occur when the error vector is the smallest. From the figure, it may appear that the perfect length of vector $\mathbf{v_w}$ occurs when the error vector is *perpendicular* to our component (as it is drawn). And indeed, this is the case.

Perhaps it would help at this point to make a note about *orthogonality.* As was stated previously, the unit vectors **i** and **j** have a few different properties that make them an excellent basis for two-dimensional vector construction. One such property, which we have mentioned, is that they have been normalized. Another is that they are *orthogonal.* The mathematical definition of orthogonality is that if the dot product of two vectors equals zero, then the two vectors are orthogonal. Thus, if

$$\langle \mathbf{v},\mathbf{w} \rangle = |\mathbf{v}| \cdot |\mathbf{w}| \cdot \cos(\theta) = 0, \qquad \text{[Equation 9]}$$

then vectors **v** and **w** are orthogonal. From Equation 9, you can see that orthogonality must occur when $\cos(\theta) = 0$, or when $\theta = 90°$. In other words: *perpendicular vectors are orthogonal.*

Why is this a good quality for a basis set to have? Well, since **i** and **j** are perpendicular (or orthogonal), the component of each vector in the direction of the other is zero, and thus *neither vector can be made up of any part of the other*. And since we're trying to build vectors from elemental parts, you can hopefully see the sense in requiring that these elemental parts are *not at all* made up of each other, as that would get kind of confusing.

Problem 11: Figure 15(a) shows a vector **v**, whose magnitude $|\mathbf{v}| = 6.20$. Also shown are the unit vectors **a** and **b**, with $\theta_a = 97°$ and $\theta_b = 25°$, which we will use a basis set for constructing **v**. Is it possible for us to express **v** as a sum of scaled versions of **a** and **b**? (Think carefully.) Find the projection of vector **v** in the direction of each vector. Will these projections add up to form **v**? Perform this same set of operations on **v** again, this time using the basis set shown in Figure 15(b), with $\theta_i = 70°$ and $\theta_j = 20°$. Compare and contrast our two sets of basis vectors: how are they the same, and how are they different? **[10 pts.]**

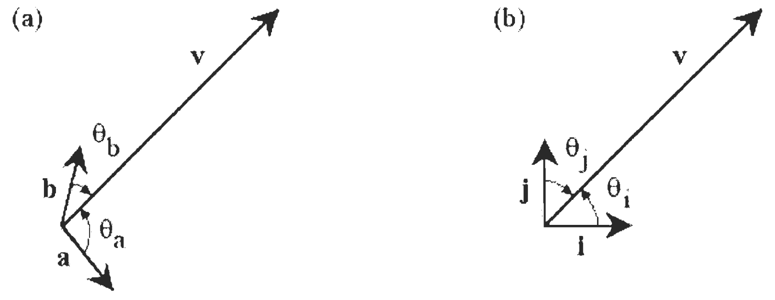


Figure 15: Vectors and basis sets for Problem ***11****.*

Now, where were we? Oh, yes—we were attempting to find a general rule, using the dot product, to find the component of one vector along any other vector—even if it is not of unit length. To do this, however, remember the argument we made that the smallest error in a component occurs when the component is perpendicular with the error vector (the difference between the vector and its component). Looking back at Figure 14 with our newfound understanding of orthogonality, it should make intuitive sense that this is the case. Since $\mathbf{v_w}$ and the error vector are orthogonal, there *cannot* be any amount of $\mathbf{v_w}$ in the error vector, or vice versa. Moreover, if $\mathbf{v_w}$ were any smaller or larger, there would have to be some component of $\mathbf{v_w}$ in the error vector—and thus, the error vector would *not* be as small as possible.

Knowing this, how do we modify our existing vector equations to account for the non-unit length of vector **w**? Well, since our ideal component along **w** forms a right triangle with the error vector and **v**, we can use simple trigonometry again to show that

$$|\mathbf{v_w}| = |\mathbf{v}| \cdot \cos(\theta). \qquad \text{[Equation 10.1]}$$

However, what we really want to find is that constant *c*, that shows us by how much we scale vector **w** to make vector $\mathbf{v_w}$. Substituting into Equation 10.1, we find that

$$c \cdot |\mathbf{w}| = |\mathbf{v}| \cdot \cos(\theta), \qquad \text{[Equation 10.2]}$$

and multiplying both sides by $|\mathbf{w}|$,

$$c \cdot |\mathbf{w}|^2 = |\mathbf{v}| \cdot |\mathbf{w}| \cdot \cos(\theta). \qquad \text{[Equation 10.3]}$$

So finally, substituting into Equation 10.3 according to the dot product formulas found in Equation 6.3 and Equation 6.4,

$$c = \frac{\langle \mathbf{v}, \mathbf{w} \rangle}{\langle \mathbf{w}, \mathbf{w} \rangle}. \qquad \text{[Equation 10.4]}$$

We've gone through a lot of work to get to this formula but you can see that it represents the constant c in a very understandable way. The dot product in the numerator is basically the same simple trigonometry operation we performed in Equation 10.1—the projection of one vector on another. The dot product in the denominator is responsible for normalizing c, in the case when our basis vector is not of unit length. This equality will work when attempting to find by how much you would scale *any* vector **w** to make it the perfect component—the best approximation—of any other vector **v**. Now, we are finally ready to delve into the world of signals, armed with our vector knowledge. We will find that there are countless similarities between what we have learned in the realm of vector construction, and the practice of building signals from their components.

The Dot Product: Strong Enough for a Vector, But pH Balanced for a Signal

Up to this point, it may seem to you as though we've gone a little out of our way in our discussion of signal analysis—you've been led through a winding discussion of vector mathematics, a great deal of which you probably already know. However, have faith—this journey has been truly worthwhile, because there is an amazing parallel between what we have done so far with vectors, and what we are about to do with signals. As Lathi powerfully states it, "the analogy is so strong that the term '*analogy*' understates the reality. Signals are not just *like* vectors. Signals *are* vectors!"

What does this mean? Just as we can break down *vectors* into a sum of components, given some arbitrary basis set, we can break down *signals* into a sum of components, according to a set of basis signals. But wait, there's more! Just as we were able to define a *vector dot product*, and use it to find vector components, we can determine an analogous *signal dot product*. And perhaps, we can use it for the same noble purpose—breaking down signals into their components.

For any two real signals $a(t)$ and $b(t)$, the dot product of the signals over the interval $[t_1, t_2]$ is

$$\langle a(t), b(t) \rangle = \int_{t_1}^{t_2} a(t) \cdot b(t)\,dt. \qquad \text{[Equation 11]}$$

Like its vector counterpart, the dot product in Equation 11 is a way of measuring of the "relatedness" of two signals. As a result—just like the vector dot product—we can make assumptions about the relationship between the two signals $a(t)$ and $b(t)$, given the sign of their dot product. Although we will discuss this more in the section on *correlation*, a positive dot product means that two signals have a lot in common—they are related in a way very similar to two vectors pointing in the same direction. Likewise, a negative dot product means that the signals are related in a *negative* way, much like vectors pointing in opposing directions.

Problem 12: What does the dot product of a signal with itself tell us about that signal? How does this correspond to what we have learned about vectors? **[8 pts.]**

Perhaps the most interesting point is that we can also draw assumptions about two signals whose dot product is zero. Just as in the world of vectors, the two signals whose dot product is zero are *orthogonal*—and thus, like vectors, are completely unrelated and cannot be formed from one another. Hence, two real signals $a(t)$ and $b(t)$are orthogonal over the interval $[t_1, t_2]$ if

$$\int_{t_1}^{t_2} a(t)\cdot b(t)dt = 0. \qquad \text{[Equation 12]}$$

We will examine signal orthogonality further in the next section. But wait, the vector-signal analogy does not stop there, either! Just as with vectors, we can use the signal dot product to find the component functions of signals. Let's assume we have two signals, $f(t)$ and $g(t)$. Now, let's consider $g(t)$ to be a *basis function*, which we can use to construct other signals. As you may imagine, in constructing the signal $f(t)$ from components, we will need a complete *set* of basis functions may include $g(t)$ (as well as *many* others). Take a look back at Figure 14, and remember how we constructed the component of vector **v**. It would follow from our vector-signal analogy that for our signals $f(t)$ and $g(t)$, there exists some constant c such that

$$f(t) = c\cdot g(t) + \mathit{errorsignal}. \qquad \text{[Equation 13]}$$

Once again, we can think of this situation as using $g(t)$ to *approximate* the signal $f(t)$ over some interval. And, just as we did with the vectors, we want to be able to find a value for c that minimizes the error in our approximation. However, since we are working with signals, instead of measuring the error in terms of magnitude, it is far more appropriate to quantify our error in terms of *signal energy* (a measurement with which you should be familiar). Fortunately, it is the case—Lathi derives this result well in Section 3.1-2—that as with vectors, our minimal error occurs when

$$c = \frac{\langle f(t), g(t)\rangle}{\langle g(t), g(t)\rangle}. \qquad \text{[Equation 14.1]}$$

And thus, the exact value of c, over the interval $[t_1, t_2]$, is given by

$$c = \frac{\int_{t_1}^{t_2} f(t)\cdot g(t)\,dt}{\int_{t_1}^{t_2} g^2(t)\,dt} = \frac{1}{E_g}\int_{t_1}^{t_2} f(t)\cdot g(t)dt, \qquad \text{[Equation 14.2]}$$

where E_g is the energy of the basis signal $g(t)$. Now, let's see if we can put these equations to work obtaining the components of a signal.

Problem 13: The figure below shows a signal $f(t)$, along with three basis functions, $g_1(t)$, $g_2(t)$, and $g_3(t)$, represented over the interval $[0,T]$, where $A = 4$ and $T = 4$. Take the dot product of $f(t)$ with each basis function—what do your results tell us about the relationship of $f(t)$ with each function? Which of the three would be the best basis function to use in beginning to build $f(t)$? Explain. Using this "most desirable" function, find the value of c, the amount by which the basis function should be scaled to make it a component of $f(t)$. Sketch or graph the error signal, $e(t)$. Can this error be reduced using only the one basis function you chose? **[12 pts.]**

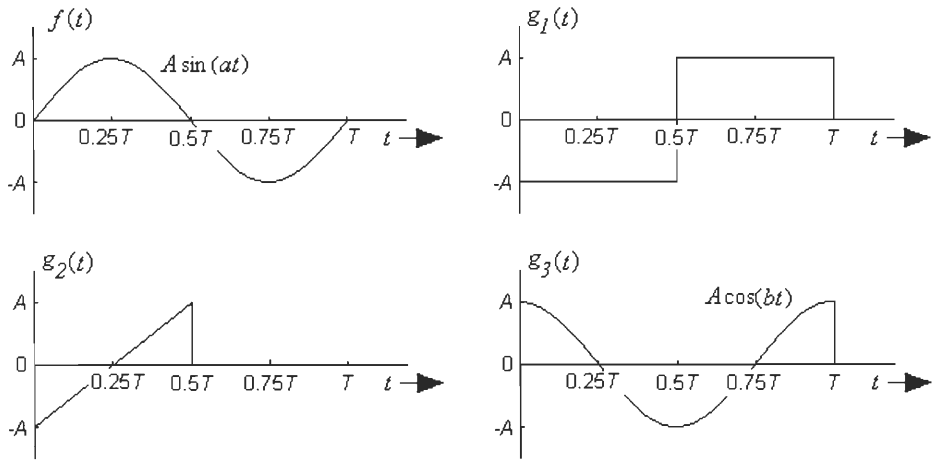


Figure 16: Signal and basis functions for Problem 13.

Signal Correlation: Lathi's "Best Friends, Worst Enemies, and Complete Strangers"

We have now approached the point where we can begin to use what we've learned about signal construction to perform some very useful operations. One of these valuable calculations is *correlation*—a measure of the *similarity* between signals. And as you may guess, the computation of two signals' correlation is based on our new friend the signal dot product.

Let's begin by taking a look back at vectors. We have already stated the dot product can be seen as a measure of the "relatedness" of two vectors. However, this measure is not absolute, since the size of the dot product depends not only on the *angle* between the two vectors, but also their individual *magnitudes*. And, if we define "similarity" in a vector sense to mean simply *direction*, then our measure of correlation should be *independent* of the magnitudes of the vectors involved. So, to compensate for magnitude, we just divide by the lengths of the vectors—*normalizing* the dot product, so that we can examine only directional similarity. The similarity of direction c_n between two vectors **a** and **b**, with an angle of θ between them, would then be

$$c_n = \frac{\langle \mathbf{a}, \mathbf{b} \rangle}{|\mathbf{a}| \cdot |\mathbf{b}|} = \frac{|\mathbf{a}| \cdot |\mathbf{b}| \cdot \cos(\theta)}{|\mathbf{a}| \cdot |\mathbf{b}|} = \cos(\theta). \qquad \text{[Equation 15]}$$

So, to measure how *correlated* two vectors are, all we need is the cosine of the angle between them. Consequently, c_n—which is called the *correlation coefficient*—always ranges from –1 to 1, regardless of the size of the vectors measured. A correlation of $c_n = 1$ corresponds to two vectors that are as similar as possible—thus, they must be positively scaled versions of one another (since the angle between them is zero). Likewise, a correlation of $c_n = -1$ corresponds to two vectors that are *oppositely* similar (since they have an angle of 180° between them). Finally, vectors with a correlation of zero are, as before, *orthogonal*—they have nothing to do with one another.

How does this relate to signals? Just as with vectors, we should be able to "divide out" the sizes of the two signals from the signal dot product, leaving us with a measure of pure similarity. To accomplish this, we have to compensate for our measure of signal size—that is, we have to normalize the two signals to *unit energy*. So, if we consider the entire range of $(-\infty,\infty)$ then the correlation between and two real signals $a(t)$ and $b(t)$ is

$$c_n = \frac{1}{\sqrt{E_a \cdot E_b}} \int_{-\infty}^{\infty} a(t) \cdot b(t) \, dt \, , \qquad \text{[Equation 16]}$$

where E_a and E_b are the respective signal energies of $a(t)$ and $b(t)$. Observe that as before, c_n is independent of the energies of our signals, ranging always from –1 to 1.

So how do we interpret signal correlation? Well, if the correlation between two signals is $c_n = 1$, then as with vectors, the signals are simply scaled versions of one another. Barring signal energy, the two signals are as similar in shape as they could possibly be. And, as with vectors, a correlation of $c_n = -1$ signifies two signals which are *also* identical in shape, with the important exception that one is the negative of the other. Finally, two signals that have no correlation ($c_n = 0$) are *orthogonal*. Orthogonal signals cannot be components of one another. Here is another way to think about it: if we were to use a signal $g(t)$ to approximate another signal $f(t)$, and $f(t)$ and $g(t)$ were orthogonal, then the minimum error that we could achieve is if we used *none* of $g(t)$. Adding or subtracting even the least little bit of $g(t)$ to or from $f(t)$ would only serve to *increase* the error in our approximation.

Problem 14: The figure below shows a function $x(t)$, along with five other functions: $y_1(t)$, $y_2(t)$, $y_3(t)$, $y_4(t)$, and $y_5(t)$. For $A = 5$, $b = 3$, and $T = 6$, state whether $x(t)$ is positively correlated with, negatively correlated with, or orthogonal to each of the other five functions. (It may not be necessary to perform the dot product integral in *every* case; you are free to combine mathematical and graphical arguments.) **[15 pts.]**

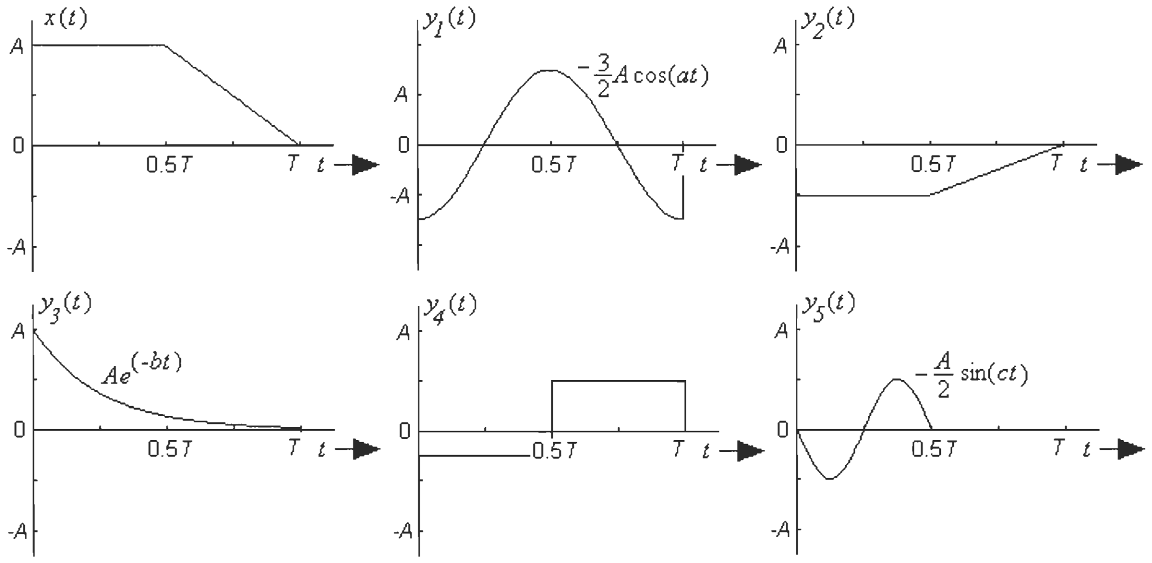


Figure 17: Signals for Problem 14.

As Lathi phrases it in Section 3.2, the three different instances of signal correlation where $c_n = 1$, -1, or 0 are like the relationships between "best friends, worst enemies and total strangers". To clarify: positively correlated signals are related in the way of a friendship, whereas negatively correlated signals have an association akin to that of enemies—very similar in essence, except in opposing ways. Orthogonal signals, however, are truly like "complete strangers". They possess a quality that would compare to being, in the words of Lathi, "of different species or from different planets".

On a side note, correlation itself is actually much more than just another mathematical avenue for us to travel on our route to signal construction. Correlation calculations are used in a number of applications—for instance, in signal detection applications. One example of this, with which you are probably familiar, is *radar* (*ra*dio *d*etection *a*nd *r*anging). In radar, a pulse shape is transmitted via radio waves into the open air, where any object that we would want to detect may be located. Although the receiver usually reads only noise from the environment, if an object *does* enter the detection space, then the signal is reflected back. There is at least one major problem with this approach, though: how does the receiver identify the reflection of the transmitted pulse, amongst all of the noise? One solution is to use correlation—we can quantify the extent to which the signal we received resembles the signal we sent. Then, we can make a decision about whether or not an object has been detected, based on this correlation value. The thing that makes correlation particularly well suited for this purpose is that *it doesn't care about the size of the pulses*. Thus, we can evaluate how closely the *shape* of the reflected pulse matches the transmitted pulse, even though it has been heavily attenuated by propagation through the air.

Part 2: The Building Blocks of Signals

How Do I Get My Very Own Orthonormal Basis Set?

Before we can answer that question, we need to consider: what *is* an orthonormal basis set? Most obviously, it is a *set* that we may use as a *basis*—that is, a collection of vectors (for constructing vectors) or functions (for constructing signals) that we can use as basic building elements. What truly defines a basis set, however, is that there are enough elements in the set that when they are scaled properly and added together, they can perfectly build *any* vector or signal. An *orthonormal* basis, though, is a little more than just another basis. Perhaps the best way to attack this subject is by deconstructing and analyzing the term itself.

So, let's begin—the first part of the term "orthonormal basis" is *ortho-*, as in *orthogonal*. What does this part tell us? It tells us that in order to break down accurately vectors or signals into their components with a single dot product calculation per component, the components need to be orthogonal. This is a lesson we learned back in Problem 11—if our components are even slightly made up of one another, then it becomes far more difficult to discern how much of each element we need. This is because we have to consider the relationship—the *correlation*—between our basis vectors or functions. Of course, we know for orthogonal elements that this relationship *does not exist*; they are, as we said before, "strangers". Thus, we can examine the component of a vector or signal along each basis element *individually*.

The next part of the term to consider is *-normal*. We discovered very early on in this discussion the convenience of having our basis elements *normalized*. This does not even necessarily mean that the basis vectors have to be of *unit* length, just that your basis vectors are all at least the *same* length. Another way to think about it is: if you have a two-dimensional displacement vector, it would be ridiculous to break it down into a component measured in feet, and another measured in meters. The two values would have no meaning with respect to each other, without performing some kind of a unit conversion—such a basis would be inconvenient. In signal construction terms, this means that normalization allows us to *compare the size of the component functions with each other*, without having to perform extra calculations.

Finally, as we said before, our orthonormal basis set must truly be a *basis* for its intended vector or signal space. No matter how many vectors or functions the basis set includes, there has to be enough such that we can completely build *any* vector or signal that could exist in the space. Consequently, for two-dimensional or three-dimensional vectors, we need two or three basis vectors in the set, respectively. (Of course, all of the basis vectors must be *linearly independent*—they cannot be scaled versions of one another, or we wouldn't really have different vectors.) For signals, however, this usually means that we need an *infinite number* of orthogonal basis functions.

The necessity for an infinite basis set makes the other two characteristics—orthogonality and normalization—even more important. Just imagine if our infinite set of functions was not orthogonal. We would have to take into account the correlation between infinite pairs of signals—this would be impossible! Or, imagine if our basis set was not normalized—we would have to keep track of the infinite series of coefficients that told us how the signal energy of each basis function differed from every other. That sounds a little difficult as well, does it not?

Problem 15: Consider the three qualifications a set must possess to be considered an orthonormal basis. In terms of its usefulness in representing signals as their components, which would you consider to be the *most* important qualification, and why? Which would you consider to be the *least* important, and why? **[12 pts.]**

So, let's recap—an orthonormal basis is a very useful set of elements that possesses three main characteristics. First, all of the elements need to be *orthogonal*, so we can calculate the projections of our signal or vector on each basis element individually. Second, all of the elements should be *normalized*, so that we don't have to concern ourselves with a different size factor for each element, and we can easily compare the components to one another. Finally, the set should form a complete *basis*, from which we can construct any vector or signal we could imagine within the given space.

Fun with Walsh Functions

One well-known orthonormal basis is the set of *Walsh functions*. As a basis, these functions can be used to construct any continuous signal within the finite range of time over which they are defined. The function values are actually obtained from the rows of a special matrix known as the *Hadamard matrix*. If it interests you, feel free to further investigate the origins of this fascinating collection of functions.

Walsh functions have a number of unique qualities. First of all, the Walsh functions have the same characteristics that are possessed by any orthonormal basis set for signals; namely, there are an infinite number of them, they are all mutually orthogonal, and they are all normalized to the same signal energy. One of the more unusual characteristics of the set of Walsh functions is that they take on only two amplitude values (usually 1 and –1). This means that they can be generated rather inexpensively and efficiently using basic logic circuits, such as inverters—and more importantly, it is possible to implement the multiplication of a Walsh function by a signal, simply by inverting the signal over the correct intervals. The following problem examines some more of the Walsh functions' characteristics.

Problem 16: The figure below shows the first eight elements in the infinite set of Walsh functions. First, notice the Walsh function $w_1(t)$—if we use these Walsh functions to create a signal, what "frequency" does $w_1(t)$ represent? From looking at the graphs, can you derive a general rule for how the Walsh functions have been ordered in terms of symmetry and sign reversals? Which Walsh functions are *even* (about the center of the interval, $t = 0.5$)? Which are odd? If the Walsh functions are to be a complete orthonormal basis set, can you explain why it is necessary to have *both* even and odd types of components? **[8 pts.]**

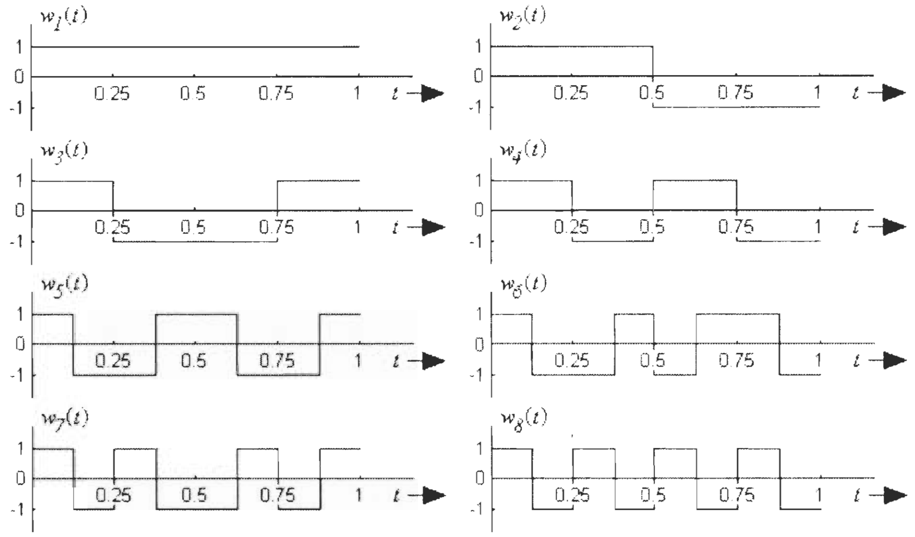


Figure 18: First eight Walsh functions over the interval [0,1].

Problem 17: Consider the continuous-time signal

$$x(t) = \cos^3\left(\frac{t}{2}\right) - \sin^3(t) \qquad \text{[Equation 17]}$$

over the interval $[0,2\pi]$. With respect to the center of the interval, is this signal even, odd, or neither? Of the first eight Walsh functions (shown in Figure 18), which ones form nonzero components of $x(t)$?

What kind of difficulties will we encounter if we try to represent $x(t)$ accurately with a *limited number* of Walsh function components? How might a different basis allow us to improve upon these difficulties?

Using the set of Walsh functions as an orthonormal basis, write a Matlab m-file that can find the components of the signal $x(t)$, and reconstruct it over the interval $[0, 2\pi]$. To accomplish this feat, you are going to need to review everything you have learned so far about what an orthonormal basis set is, and how to find out how much of each basis function is needed. When approximating how much of each basis function is needed, remember that it needs to be normalized by the energy of the basis function over the range of the signal. Also, since an infinite number of components are needed to reproduce $x(t)$ *exactly*, the actual reproduction will be an approximation using a finite number of terms. Thus, your m-file should allow you to specify *any* number of Walsh functions as an input. **BEFORE YOU START**, be sure to carefully review the project—come up with a sensible algorithm, *in words*, for accomplishing this task, and write this algorithm out *before* you begin to work in Matlab.

Some helpful hints: Matlab can simulate continuous integration by performing a numerical analysis of trapezoidal integration, using the function `trapz` (look it up in Matlab help for instructions). Also, to generate the Walsh functions, you can use the Matlab function `walsh(t,n)`. For the function arguments, `t` is an array of the range of points over which the Walsh function is represented (identical to the kind of array you must create to graph any function), and `n` is the index of the Walsh function. (Note: this function was created specifically for this project, so if you are using Matlab at home, you'll need to download it off the course web page.)

Also, don't forget to use the element-by-element multiplication (`.*`) and power (`.^`) operators, when multiplying two arrays or raising an array to a scalar power, respectively.

To document your results: plot $x(t)$, along with its reconstruction using the first n Walsh functions, for $n = 3$, 7, 15 and 31. Print each graph—be sure they are titled, and the axes and curves are appropriately labeled (you can write labels in, or use the appropriate Matlab commands). Also, be sure to provide the original algorithm you developed before you started coding, and be sure to print out your m-file code and clearly state what each part of your program does (you will want to use the comment operator (`%`), to put remarks in your m-file). It is your responsibility to do a good job documenting your algorithm, code, and graphs, since it is the only way to prove that the work you submit is indeed your own. **[20 pts.]**

Written by Nicholas Arcolano — 21 July 1999.
Revised 21 December 1999; 14 November 2000; 28 March 2001; 13 November 2001 (AD)

Discovery Project III — The Wide World of Modulation

One characteristic of the study of continuous-time communications that you have probably already noticed by now is that there are a great many different ways to classify signals and systems. This practice is very useful to us—depending on the situation, each classification allows us to think about the signal or system in a different way, often making difficult problems much easier to tackle. This project is based on yet another way of classifying signals—by whether they are *baseband* or *bandpass*.

Although we will go into the specifics of these two types of signals later in the project, it is important to understand that *most of the signals we create would normally be baseband.* Thus, whenever we need to produce a bandpass signal, it is generally the case that we must *begin* with a baseband signal, and then alter its frequency content. In essence, what we are doing is introducing higher frequencies into the spectrum that contain the same information—in short, transmitting the *same data* with *different frequencies.* This is accomplished by way of a process called *modulation.*

Even more exciting is the fact that the process of modulation opens the door to a vast new world of techniques for transmitting and receiving signals. In this project, we will be examining an application of modulation with which you should be very familiar. It is called *amplitude modulation* (AM), and it is used primarily for the transmission of radio signals. So, we will be learning exactly how AM radio signals are created, and how they are detected in AM radio receivers—just like one you might have in your home right now. Sound interesting?

So, our objectives for this project will be:

- To specify the difference between baseband and bandpass signals, and understand the need for bandpass signals to transmit and receive information.
- To introduce the general concept of modulation.
- To examine the generation of an AM signal in both the time and frequency domains.
- To briefly discuss the concept of using frequency-division multiplexing to simultaneously transmit different AM signals.
- To discuss the process of demodulation to reproduce a received baseband signal.
- To describe the operation of an *envelope detector* for the reproduction of baseband information from a bandpass signal.

- To investigate the demodulation of AM signals using a *product detector*.
- To perform an example of the generation, transmission, and product detection of an AM signal, using Simulink.

Part 1: Baseband, Bandpass, and Modulation

What's the Difference?

A *baseband* signal has a magnitude spectrum consisting primarily of frequencies at and around DC ($\omega = 0$), as you can see from the figure below. As was mentioned earlier, most of the signals we produce are baseband. When you speak into a microphone, (producing a voltage waveform representing your voice), you create a baseband signal consisting of frequencies somewhere in the range of 300 Hz to 3 kHz (these are the frequencies that your voice is naturally able to produce). Note that a baseband signal does not necessarily have to contain values *exactly* down to and including DC—a more general definition of baseband simply refers to *any* signal has not yet undergone a form of modulation.

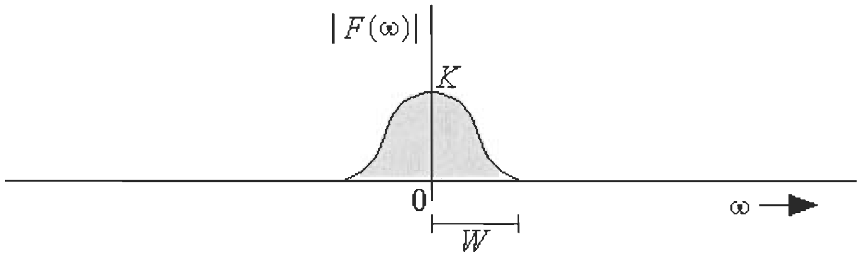


Figure 1: Magnitude spectrum of a baseband signal.

A *bandpass* signal, however, consists primarily of frequencies that are, appropriately, within some band. This range of frequencies is usually located much higher than DC. For instance, when you tune your radio to 107.3 MHz (WAAF—as they say, "the only station that really rocks"), you are receiving a bandpass signal consisting solely of frequencies at and around 107.3 MHz. Below is the magnitude spectrum of a bandpass signal. As you can see, since bandpass signals we receive are real, the magnitude spectrum is still symmetrical.

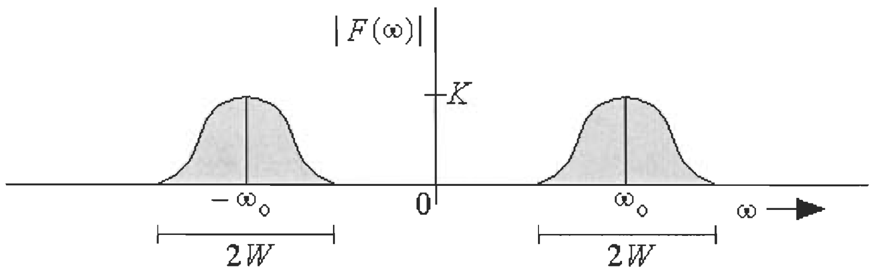


Figure 2: Magnitude spectrum of a bandpass signal.

So, Why Do We Need Bandpass Signals?

Regardless of whether a signal is baseband or bandpass, its general purpose is always to contain *information*, of some sort. Usually, we want to transmit this information from one point to another. Often times, it is little trouble to transmit baseband signals over short distances on closed channels, such as the case of transmitting a voice signal over a direct line from a microphone to a speaker.

However, a great deal of the signals we transmit, such as radio, television, or cellular telephone, use the open *air* as a channel—otherwise, they would hardly be so convenient. This means that to send and receive these signals, we need the use of an *antenna*. This is the first justification for why we sometimes need bandpass signals—antenna reception is a problem for baseband transmission. The reason is that in order to receive a signal transmitted through the air, the general "rule of thumb" is that you need an antenna that is *at least as long as one-tenth the wavelength of the signal.*

Now, you may remember from the study of physics that for any oscillating wave,

$$\upsilon = f\lambda \qquad \text{[Equation 1]}$$

where v is the velocity of the wave, f is the frequency in Hertz, and λ is the wavelength. Also recall that for electromagnetic waves, like radio, the velocity is equal to c (the speed of light, $3 \cdot 10^8$ meters per second). The problem below should illustrate why this poses such a dilemma for the reception of baseband signals.

Problem 1: An electromagnetic baseband signal, consisting of standard voice frequencies (300 Hz to 3 kHz) has been transmitted directly into the air. What is the wavelength of the highest possible frequency in this range? What is the wavelength of the lowest? How long of an antenna (in meters) would be necessary to receive all of the frequencies being transmitted? How long of an antenna is needed to listen to WBMX (98.5 FM) on an FM radio receiver?

As you can see, the transmission of baseband signals over open-air channels is completely impractical—modulation is a necessity. Furthermore, even the *closed* channels (such as a microphone cord) over which we wish to transmit can be considered "bandpass channels". This simply means that they have their *lowest attenuation* over some bandwidth, and attenuation is more for frequencies above and below this band. For instance, standard telephone lines have minimum attenuation (or maximum gain) at around 1000 Hz. Attenuation increases as you go to higher or lower frequencies from this point. Fortunately, it works out that a standard telephone channel is suitable for baseband telephone transmission, so no modulation is necessary. However, there are many signals that must undergo modulation before they can be transmitted over certain closed channels.

Finally, it is sometimes the case that we can use modulation to create bandpass signals simply because we *want* the information to be located at certain frequencies. By doing so, we can distinguish different sources of information from one another. A good example of this is modem communication. In the case of a modem, the information being transmitted from one end of the connection is located in a different band of frequencies than the information that is being transmitted from the other end of the connection. This way, two modems can both transmit and receive *simultaneously*. This idea of separating information in the frequency domain that is transmitted simultaneously in the time domain is the foundation of *frequency-division multiplexing* (FDM), which we will discuss later.

Methods of Modulation

The actual process of modulation occurs when any baseband signal is used to alter, or *modulate*, a high-frequency sinusoid in such a way that the information of the baseband signal remains intact within the properties of our newly altered high-frequency signal. As a result, we are left with a bandpass signal that contains the same information content of its baseband counterpart, and can be transmitted. The baseband message signal is called the *modulating signal*, since it is used to change the high-frequency sinusoid. The high-frequency sinusoid that "carries" the baseband information is called, appropriately, the *carrier*. After the baseband signal has modulated the carrier, and the two have been combined, the new information signal is called the *modulated signal*.

Although there are many different ways to create a bandpass signal through modulation, most methods fall into one of two main categories—linear modulation, and non-linear or angle modulation. The difference between these methods depends on what it is about the carrier that the modulating signal changes to encode information. *Amplitude modulation* (AM) is the premier method of linear modulation. Although we will be exploring this process in greater depth in the next section, you should realize for now that in amplitude modulation, the baseband message signal encodes information into the carrier by changing the carrier's *amplitude*. The two main types of angle modulation, which are well beyond the scope of this project, are *phase modulation* (PM) and *frequency modulation* (FM). As you may or may not have guessed, in these modulation schemes, the message is encoded into the carrier by altering (respectively) either the *phase* or the *frequency* of the carrier sinusoid.

Part 2: Amplitude Modulation

Make-Your-Own Amplitude-Modulated Signal

Before we can fully understand how modulation works, we have to explore exactly how it affects a signal in both the time and frequency domains. The simplest and most interesting way to accomplish this is through an examination of amplitude modulation. As was stated before, an amplitude-modulated signal is created when the *modulating signal* (our baseband message signal) is used to modulate the *amplitude* of a high-frequency carrier. This is simply done by multiplying the modulating signal and the carrier. So, for any modulating signal *m*(*t*), the bandpass modulated signal is represented by

$$s(t) = m(t) \cdot A_c \cos(\omega_c t) \qquad \text{[Equation 2]}$$

where A_c is the amplitude of the carrier sinusoid before modulation, and ω_c is the *carrier frequency*, in radians per second. The signal with which we end up, *s*(*t*), is a bandpass signal—it contains all of the information that was encoded in *m*(*t*), but that information is now represented using higher frequencies. But what exactly has happened in the frequency domain? Let's take a look:

Problem 2: You've been quietly working for years on planet Earth, in the business of extraterrestrial espionage. However, the men in black have been snooping around quite a bit lately, and it's probably time to give the mother ship a call.

The signal you must send, in order to notify them to come rescue you, is the sinc pulse shown in Figure 3. (Don't be concerned that it ranges over (-∞, ∞) in time—aliens are allowed to send non-causal, infinite signals if they want to, since the math is easier in this case.) What is the mathematical expression for this sinc pulse in the time domain? What is the Fourier transform of this signal? Graph the magnitude spectrum of the pulse.

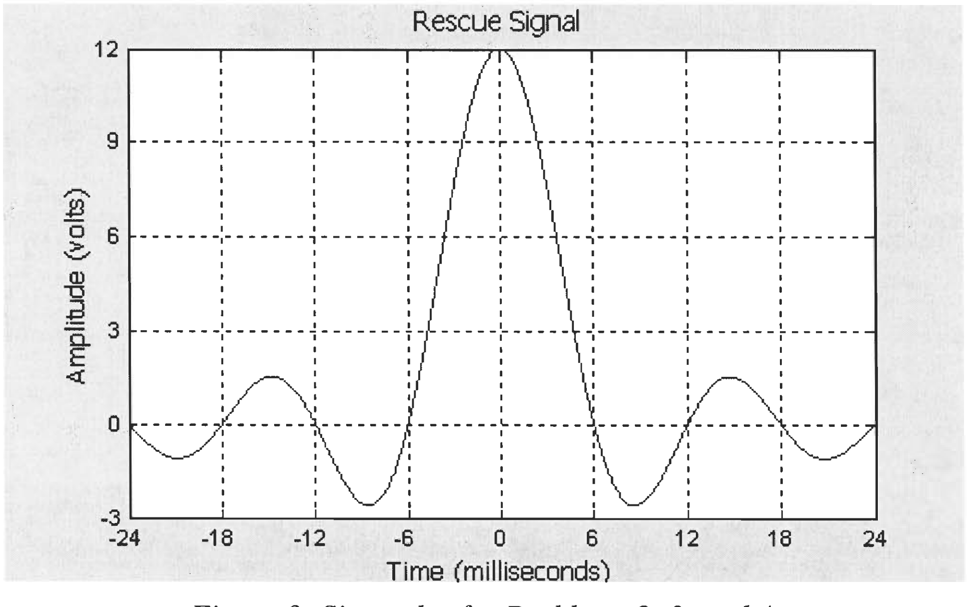


Figure 3: Sinc pulse for Problems 2, 3, and 4.

Another measurement that we use to classify signals is *bandwidth*—quite simply, the width (in Hertz or radians/second) that a spectrum takes up in the frequency domain on the positive frequency axis (refer to Lathi, page 212). In other words, it is the difference between the highest and the lowest frequency in the signal. Thus, for a (band-limited) baseband signal, it would be the span from DC to the highest frequency component in the signal. Once a signal is modulated (to become a bandpass signal), the bandwidth is then the width of the bandpass spectrum.

Problem 3: What is the bandwidth of your sinc pulse? Before you transmit your sinc pulse, you have to make it into a bandpass signal. Your trusty Alien Spy Handbook suggests a frequency of 1.2 GHz ($\omega_c = 2.4 \cdot 10^9 \pi$ rad/sec). If $A_c = 9$, what is the mathematical expression for this carrier wave and its transform?

Problem 4: You have your signal, you have your carrier—you're finally ready to "phone home". What is the mathematical expression for the Fourier transform of the final AM signal? Remember, the AM signal is the *product* of the modulating signal (the sinc pulse) and the carrier. Thus, in frequency, you will need to *convolve* the transforms of these two components. You may need to review in your text the process of convolution with an impulse. Finally, sketch the magnitude spectrum of the AM signal. What is its bandwidth now?

Below is a basic overview of the operations in both the time domain and frequency domain needed to generate an AM signal. The modulating waveform in Figure 4(a) is an arbitrarily chosen signal, and is assumed to have the magnitude spectrum shown in Figure 5(a). Note, once again, that since we are multiplying the modulating signal and the carrier in the time domain, we must convolve their transforms in the frequency domain.

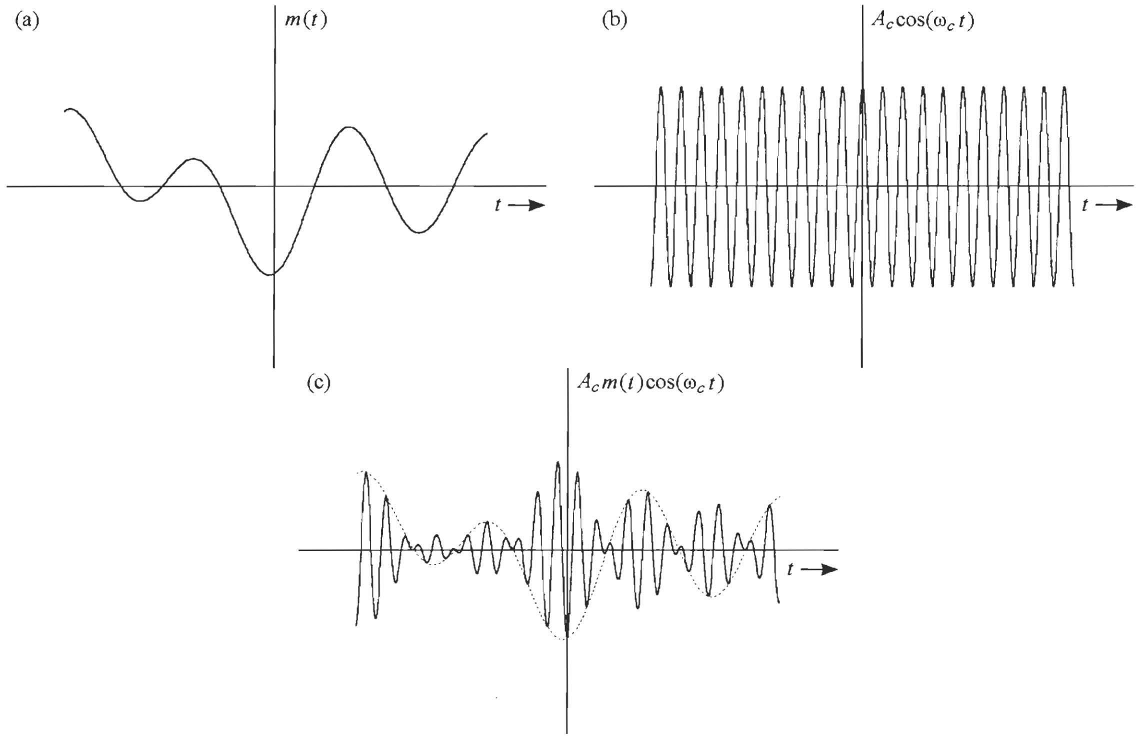


Figure 4: Time-domain graphs of modulating signal (a), carrier (b), and final AM signal (c).

The graph in Figure 5(c) shows the magnitude spectrum of the finished AM signal, ready to transmit. Notice that the magnitudes in the spectrum are now *one-half* what they were originally (times any gain A_c from the carrier). Also notice that while the baseband version of the signal had a bandwidth of W, its bandpass alter ego now has a bandwidth of $2W$.

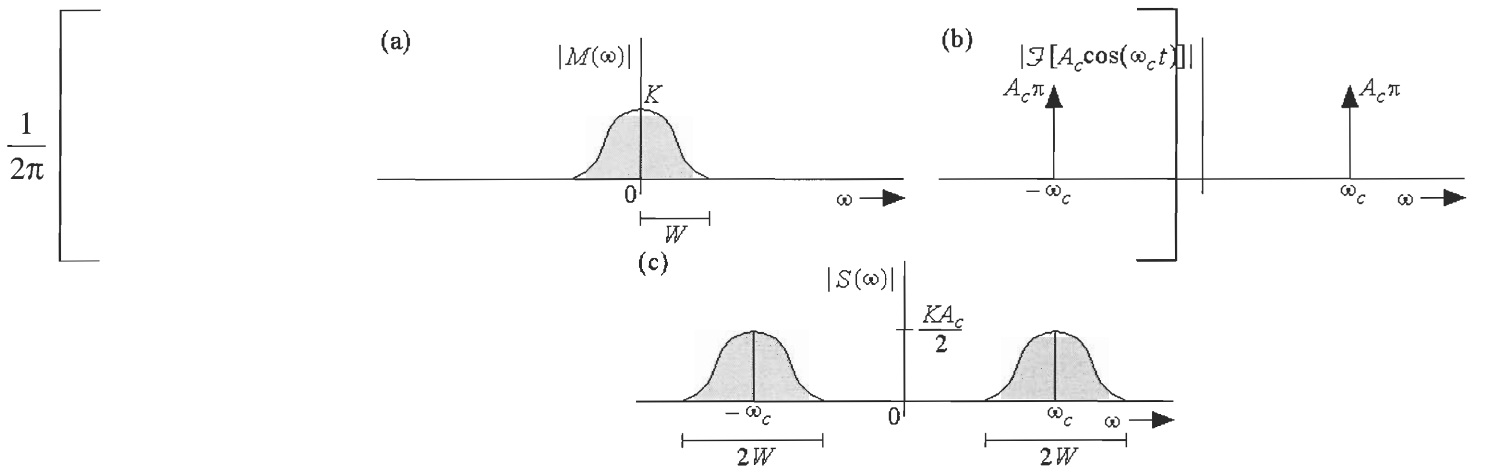


Figure 5: Magnitude spectra of modulating signal (a), carrier (b), and final AM signal (c).

Why is this difference in bandwidth important? Bandwidth is basically the amount of "space" that the signal takes up in frequency—and often in engineering, this space is very valuable. For instance, we have already stated that most channels only act desirably over a rather limited range of frequencies. This is certainly one reason why would want the bandwidth of a signal to be a small as is tolerable. However, a much more important reason for minimizing bandwidth is that we simply want to pack as much information into a range of frequencies as is possible. Probably the best way to understand this is through a brief discussion of *frequency-division* multiplexing.

Multiplexing—What's the Frequency, Kenneth?

Consider the following series of operations. First, we'll use the band-limited modulated signal we created in the previous section—we'll call it $S_1(t)$. Now, let's create another AM signal. We'll start with different baseband information, with a different (arbitrary) Fourier transform, and we'll use it to modulate a *different* carrier sinusoid. What would happen if this second bandpass signal were added to the first, so that we were essentially transmitting the two bandpass signals *at the same time*? Well—if the second carrier was different enough from the first, such that the magnitude spectra of the two modulated signals did not overlap, it is possible that the composite signal could look like the figure below.

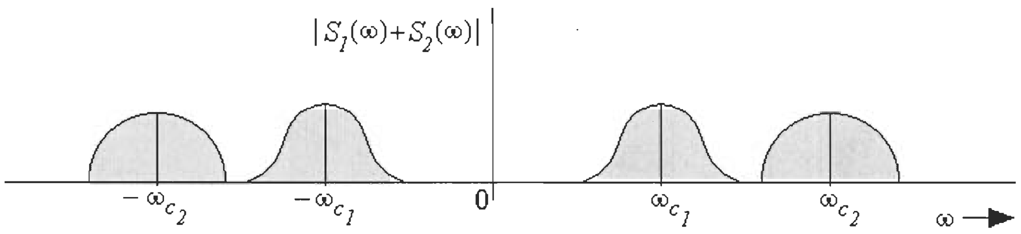


Figure 6: Frequency-division multiplexing of two signals.

What do you think this information might look like in the time domain? Truthfully, being the sum of two arbitrary bandpass signals—well, it could look like anything, but regardless it's probably a real mess. Does this mean that our information is lost or destroyed? Of course not! While it is simultaneous in time, it is being *divided in frequency*, or in other words, the information is *frequency-division multiplexed.* All we might need, for instance, is a bandpass filter to block out the other frequencies, and we can single out any one bandpass signal that we want.

On a grander scale, this is how AM radio (and FM radio) is organized—all of the stations to which we can "tune in" are all being broadcast *simultaneously*. Unfiltered, it is so much gibberish to any receiver. However, by moving that AM radio dial, you are essentially changing the station to which you are choosing to listen, and the radio is filtering out the rest. Consequently, this is one of the major reasons why minimizing bandwidth of any bandpass signal is important—the smaller we can make each signal in frequency, the more signals we can pack into any one range of frequencies.

Problem 5: Do a little research (such as going to a radio dial) to find out the minimum difference between FM carrier frequencies. (In other words, how much bandwidth is allowed for an FM radio signal?) Given that FM transmitters are only allowed to operate between the frequencies of 88.5MHz to 108.0MHz, how many FM stations can transmit simultaneously? How does this compare to AM radio? From your personal experience with the sound quality of AM and FM radio, comment on the relationship between sound quality and bandwidth of a radio signal?

On an even grander scale still, FDM is how the FCC (Federal Communications Commission) organizes *all* of the modes of wireless communication. AM radio occupies one range of frequencies, while FM radio is on another. Television signals are present on different frequencies still, as are cellular phone signals, satellite communications, and so on. All of these signals are bombarding you everywhere at every moment—they are pure incomprehensible noise in the time domain, but carefully divided in frequency to allow you the reception of any AM radio, FM radio, or television station that you wish.

Demodulation

So—we have discovered how we can generate an AM signal through multiplication with a high-frequency carrier. At this point, let's pretend that we've even been lucky enough to create and transmit an AM signal without too much trouble. Actually, let's even be so bold as to pretend that the signal we get back when we receive the signal is reasonably close to the one we sent. As you can imagine, all this hard work would be for naught unless we took the final step—detecting the original baseband information that we have encoded into the modulated carrier wave. This process is known as *demodulation*—the shifting of a bandpass signal back to baseband, and hopefully insuring that the information contained therein remains intact.

In the case of amplitude-modulated signals, there exist a couple of methods by which this can be accomplished. All of these methods involve passing the incoming bandpass signal through some sort of a circuit or process, known as a *detector*, that changes its frequency content from bandpass to baseband in some way. Detectors can generally be categorized into two classes—*coherent* and *non-coherent*. First, we will briefly discuss an extremely common circuit used for non-coherent detection (you can find one in any AM radio receiver)—the *envelope detector*.

<u>The Envelope Detector</u>

Let's return for a moment to the AM signal displayed in Figure 4(c). At first glance, it appears that the AM signal shown is *very* different from the baseband signal with which we started, because it is composed with an entirely different band of frequencies. However, we have learned that our information is still contained *somewhere* within the signal. Now, look at the dotted waveform in Figure 4(c). Notice that it resembles the original baseband signal in 4(a), and also that it follows along the peak values of the AM signal. It may be that if we could capture only the peak values of the modulated signal, and smooth them out into a curve, we could reconstruct our signal that way. And in fact, the envelope detector does exactly that—it uses analog circuitry to approximate the *envelope* (or the trace of the peak values) of the signal.

The basic operation of the envelope detector circuit depends on a diode, a capacitor, and a resistor, arranged as shown in the circuit below. In case you are unaware, a diode is a nonlinear circuit element that acts, basically, like a one-way valve for current. Whether or not the diode conducts depends on whether the voltage across it is positive or negative.

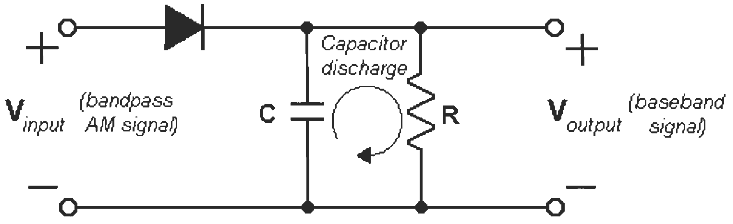


Figure 7: Envelope detection circuit.

The envelope detector works like this: when the diode is conducting, it allows the capacitor to charge (quickly) up to the peak value of the AM signal. When the diode stops conducting, the capacitor then discharges through the resistor, according to their RC time constant. The result is a waveform very much like the one shown below.

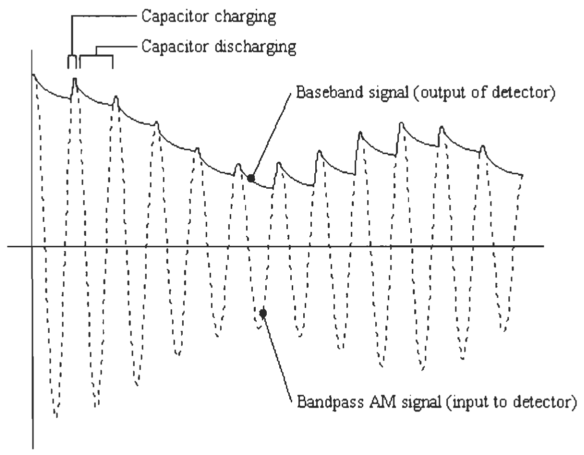

Capacitor charging
Capacitor discharging
Baseband signal (output of detector)
Bandpass AM signal (input to detector)

Figure 8: AM signal and envelope-detected reproduction.

Envelope detection has several important advantages, which is why it is so widely used. First of all, it is a non-coherent detection scheme—this means that you do not need to know very much about the carrier, such as its precise phase or frequency, in order to detect the signal. Also, due to the fairly small and simple circuit that drives it, it is very inexpensive to implement. However, the envelope detector does have some major disadvantages, which we will examine.

Problem 6: Refer once again to Figure 4(c). Sketch the resulting waveform if we were to put this AM signal through an envelope detector. You should see that even if we ignore the natural ripple that is introduced by the envelope detector, the reproduction of the baseband signal in this case is not really accurate. Why is this so? What is it about the original baseband signal that causes this improper reconstruction to occur? What, if anything, could we do to fix this error? Finally, consider now the ripple that is introduced by the detector (you can see the rippling phenomenon very clearly in Figure 8). What familiar device might we use to lessen this effect?

The Product Detector

We have already discovered that envelope detection is a useful and economical demodulation scheme, although it does not work for every AM signal. However, there *are* detectors that can properly reproduce baseband information from AM signals exactly like the one in Figure 4(c). One such device is the *product detector*—a detection scheme that uses frequency-domain operations to reconstruct the baseband spectrum.

The basic concept behind the product detector is simple. How do we re-create our baseband signal? Well, if we shifted the spectrum *up* to ω_c at the transmitter, all we have to do is find some way to shift the spectrum back *down* to DC at the receiver. Incidentally, the shifting up in frequency of a spectrum is known (fittingly) as *up-conversion*, while the shifting down in frequency of a spectrum is know as—you guessed it—*down-conversion*. But anyway, how can we accomplish this? Well, fortunately, we are able to re-create the signal at baseband by using it to modulate the *same* high-frequency carrier *again*. The figures below show the effect of this operation.

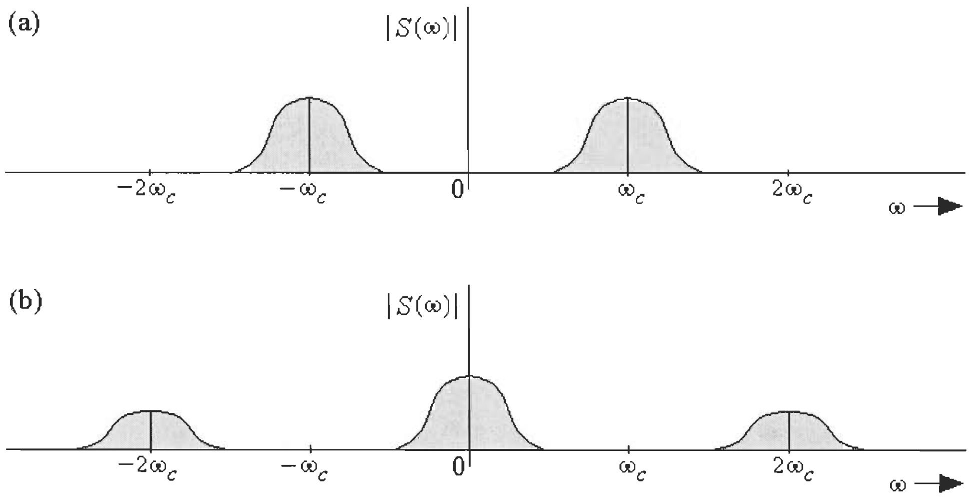


Figure 9: Magnitude spectra of AM signal before (a) and after (b) second multiplication with carrier.

The spectrum with which we end up, in Figure 9(b), is consistent with what we have discovered so far about modulation. That is, when the bandpass signal modulates the second carrier, its spectrum *splits in half, into two identical spectra, that shift by* ω_c *and* $-\omega_c$. Looking at the pair of curves in Figure 9(a), imagine one copy of the pair shifting up to center around ω_c, and the other copy of the pair shifting down to center around $-\omega_c$. As you can see in Figure 9(b), the half of the original AM spectrum that shifts *up* to ω_c

places a copy of the baseband spectrum at DC. Likewise, the half of the original AM spectrum that shifts *down* to $-\omega_c$ also places a copy of the spectrum at DC. The two identical copies of the baseband spectrum at DC then add together! Ultimately, we end with copies of our original baseband spectrum at $-2\omega_c$, at $2\omega_c$—and most importantly, at DC.

Problem 7: Why is the signal represented by the spectrum in Figure 9(b) not *yet* an accurate re-creation of the baseband signal? What still needs to be done? Assuming that all processes are ideal, how does the final baseband spectrum compare with the original one?

The major disadvantage of product detection is that it is a *coherent* detection scheme. That means that in order for it to function properly, a product detector must know the *exact* frequency and phase of the carrier that was originally modulated to create the AM signal. Unfortunately, this information is usually rather difficult to determine—it can be done, of course, but for something as simple as AM radio reception, it tends to cost far more than would be economically efficient.

Product Detection Using Simulink

Now that we have been comfortably introduced into the world of AM signals, let's try to perform a simulation of the modulation and demodulation of an AM signal using Simulink. In fact, we will use our new friend—the product detector—to retrieve our baseband information in our simulation.

In case you have forgotten, you can open Simulink simply by typing `simulink` at the Matlab command prompt. For this simulation, we will use a triangle wave as the modulating signal—you can find a triangle pulse generator in the EE 2311 library (type `ee2311` at the prompt).

Problem 8: We are going to modulate the triangle wave $m(t)$, pictured below. What is the mathematical expression for $M(\omega)$, the Fourier transform of $m(t)$? Sketch the magnitude spectrum $|M(\omega)|$, marking important values. What is the *absolute bandwidth*, the full range of non-zero frequencies for the spectrum of the signal $m(t)$? What is the *first-null bandwidth*, the range of frequencies up to the *first zero crossing* of the spectrum of the signal $m(t)$?

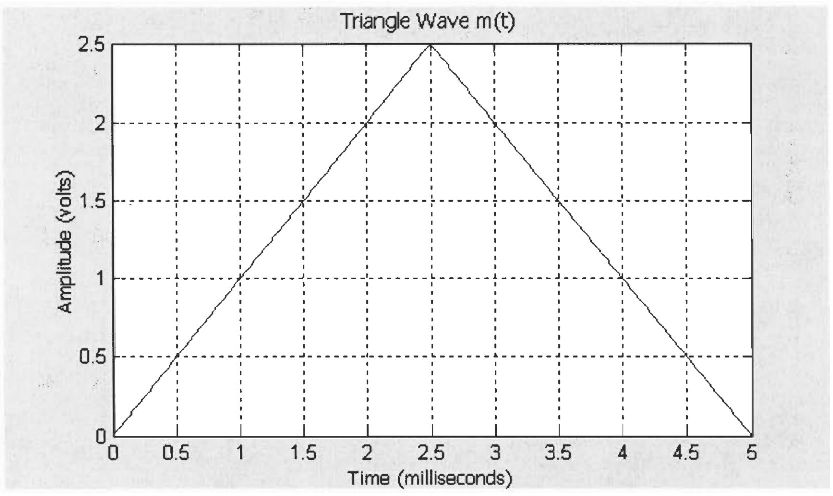


Figure 10: Modulating signal for Problem 8 and Problem 9.

In order to perform modulation, you will need, in addition to the Triangular block, a Sine Wave block (don't forget to enter a phase shift so that it generates a *cosine*) and a Product block (from the Math library). You will also need some Scope blocks from the Sinks library, as well as whatever components are necessary to implement a product detector.

Once you modulate a signal, it would be typical to transmit it. However, simulating transmission over a channel can be rather complicated and messy. For this exercise, we are simply going to demodulate the signal and examine with what amount of accuracy our original pulse can be re-created.

Problem 9: Using Simulink, modulate a carrier cosine with unit amplitude and carrier frequency of $\omega_c = 24000\pi$ rad/sec (12 kHz) with the triangular pulse from above. Generate and print oscilloscope outputs of the modulating and modulated signals.

Next, develop a product detection scheme to demodulate and reconstruct the triangular pulse from the modulated signal. If you choose to utilize a filter in your design, you can use the Analog Filter Design block from the EE 2311 library. Feel free to experiment with the different filter types, but to be reasonable, nothing higher than a fifth-order filter should really be necessary. Also, experiment with different cutoff frequencies. Print out a couple of different output pulses, including with each a description of your product detection scheme (carrier frequency, filter bandwidths, etc.) as well as your observations and explanations of the detector's performance. Sketch magnitude spectra whenever appropriate.

Congratulations Are In Order

So, you have nearly finished the last of your Discovery Projects. What did you think? Perhaps you found them to be an unbearable ordeal—or perhaps they weren't all that bad after all. (Perhaps, you even *learned* something.) Regardless, one of the most important parts of any educational experience—good or bad—is the way that you reflect on it. Was it a valuable endeavor? Why or why not? What could be better? The answers to questions like these affect how your educational experiences will shape you as you learn throughout your lifetime.

In this final exercise, we ask you to reflect a little on your EE 2311 experience. In case this concerns you: for a question like this, you will be graded on how you *present* and *support* your opinions, and not what your opinions *are*. (So, don't be afraid to give honest feedback!) You will be graded on such things as your ability to identify specific lessons learned, and also how you interpret new material in terms what you've already learned. As you think about your experience with this course, try to make critical value judgments about education experiences, and provide supporting evidence.

Problem 10: Write a short essay (between one and two pages) about what you perceive to be the advantages and disadvantages of having used these Discovery Projects as part of your EE 2311 coursework. Be sure to provide specific examples to support your opinions. As you begin to structure your thoughts about this essay, here are a few related questions that you might want to consider to help get the ideas flowing:

- What other components (such as a laboratory) typically make up a course curriculum, and how would you rate the value of these, as compared to the Discovery Projects?
- What do you consider to be your personal "learning style", and how is this style either aided or obstructed by the Discovery Projects?

- What is your stance on the use of (and accordingly, the necessity for education in) computer applications in the field of engineering?
- What do you think is the importance of learning "on your own", and do the Discovery Projects help at all your ability to do this?

Written by Nicholas Arcolano — June 30, 1999.
Revised December 21, 1999; 28 November 2000; 4 December 2001 (AD).