An Active Engagement Pedagogy for Introductory Solid Mechanics

Jaspal Singh Sandhu

S.B., Mechanical Engineering (1999)

Massachusetts Institute of Technology

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1, 2002

Author ....
Department of Mechanical Engineering
February 1, 2002

Certified by ....
Mary C. Boyce
Distinguished Alumnae Professor of Mechanical Engineering
Thesis Supervisor

Accepted by ....
Ain A. Sonin
Chairman, Department Committee on Graduate Students

An Active Engagement Pedagogy for Introductory Solid Mechanics

by

Jaspal Singh Sandhu

Submitted to the Department of Mechanical Engineering on
February 1, 2002, in partial fulfillment of the requirements for the
degree of Master of Science

Abstract

Introductory Solid Mechanics has historically been taught using the traditional methods of blackboard instruction. In the Mechanical Engineering Department, we have undertaken an initiative to comprehensively transform the pedagogy of 2.001 *(Mechanics and Materials I*), an undergraduate course in Solid Mechanics. This transformation represents a radical shift in the teaching paradigm, one which is best described as an active engagement model. Through discovery-based and cooperative learning, it is hoped that students will develop conceptual understanding of the course material, that students will become comfortable working in teams, that student retention of course material will improve, that students will be able to engage in independent learning, and that student satisfaction will improve. There are several components to this new pedagogy: physical desktop experiments, Web-enabled learning modules, a portable computing initiative, a new classroom, and a change in the lecture format. This thesis will describe all of these, but will focus on the development of the Web modules, the synthesis of these elements in developing the new pedagogy, and preliminary assessment of the project. The thesis is dually intended as a presentation of original research and as a working document for others who may wish to undertake a project of similar scope.

Thesis Supervisor: Mary C. Boyce
Title: Distinguished Alumnae Professor of Mechanical Engineering

Acknowledgements

First I would like to thank Mary Boyce, who has served both as my advisor and mentor. Working with her was a great opportunity and a true pleasure; she taught me many important lessons regarding teaching and research. Thanks also to Sanjay Sarma, for his constant encouragement and sage advice, with regards to the project as well as plans for the future.

This research could not have been possible without the generosity and commitment to education of the MIT-Microsoft iCampus initiative, the School of Engineering, and the Department of Mechanical Engineering. Additionally, the following donors have helped build the infrastructure for the new pedagogy: the Hewlett Packard Foundation, Dr. B.J. and Chunghi Park, and the Singapore-MIT Alliance.

Thanks to Eberhard Bamberg for his support and mentorship, particularly for allowing me to experiment and learn new things. I also extend my gratitude to those who have been involved in the iCampus project and/or 2.001 during my time here: Una Sheehan, Pierce Hayward, Jung-Wuk Hong, Sebastian Heersink, Ebenezer Woode, Vince Costanzo, Rebecca Brown, Farid Ganji, James Williams, and Klaus-Jürgen Bathe.

Thanks to the rearhousers - emby, big worm, and yogadeemus - who have made it a joy to go home at the end of every day. In teaching me the meaning of things like f-stop, shock controller, and downward dog, they have brought balance to my often work-intensive life.

Thanks to my family - my parents and my brother - who have always supported me with their love and understanding. It is they who have imparted in me the importance of education.

And thanks to Arundhati Singh for her paitience, for her caring, and for being a constant in my life. May your thesis use less cellulose than this one.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The rapid advancements in information technology in recent years enable the development of new teaching methods in both distance-based and on-campus learning. We in the Department of Mechanical Engineering undertook a project to comprehensively transform the pedagogical structure of an introductory Solid Mechanics course: 2.001 (*Mechanics and Materials I*).

The new pedagogy is based on an active engagement model, one that emphasizes discovery-based and cooperative learning. The four major desired outcomes of this new pedagogy are: 1. improved conceptual understanding of the course material, 2. an introduction to teamwork, 3. improved student retention across courses in the curriculum, and 4. an improved ability for students to engage in independent learning. While pursuing these goals, it is important that we maintain or improve student satisfaction, both with the course and the department. There are several components to this new pedagogy: physical desktop experiments, Web-enabled learning modules, a portable computing initiative, a new classroom, and a change in the lecture format.

This thesis will describe all of the components of the new pedagogy, focusing primarily on the development of the Web modules, the synthesis of these elements in developing the new pedagogy, and preliminary assessment of the project. This document is dually intended as a presentation of original research and as a working document for others who may wish to undertake a project of similar scope. As such, practical details may be included insomuch as they do not detract from the primary purpose of the thesis.

2.001 (formerly 2.01) has been an introductory Mechanical Engineering course for over 50 years now [52]. This course provides an introduction to statics and the mechanics of deformable solids [50]. The main emphasis is on the basic principles of equilibrium, geometric compatibility, and material behavior. These principles are covered in a progressive manner: systems of rigid bodies are first introduced; then one-dimensional linear elastic deformable elements are presented; eventually, the class progresses to the analysis of three-dimensional linear thermoelastic materials and structures. The current prerequisites are 8.01 (*Physics I*) and 18.02 (*Multivariable Calculus*), and the corequisite is 18.03 (*Differential Equations*). With few exceptions, the course content has remained the same, and the main methods of teaching this course have been the blackboard and the textbook. Given the introductory nature of the course, and the fact that this course has been taught with a reasonable amount of success over the department's history, 2.001 represented a good specimen on which to test the new pedagogy.

In terms of course history, Solid Mechanics has been taught using very few experimental techniques. The material has changed slightly over the time that it has been taught, but the general topics have remained nearly unchanged through the history of the course [16]. Just after World War II, the Department of Mechanical Engineering was focused primarily on applied mechanics. There were separate courses for Statics and Kinematics; and two courses each on Dynamics and on Strength of Materials. Many junior- and senior-level courses provided the opportunity for more thorough exploration of these topics. The Mechanical Engineering faculty made a conscious decision at that time to consolidate

Statics and Strength of Materials. The result was one course on Solid Mechanics and another focusing on Mechanical Behavior of Materials; these are the present-day courses 2.001 and 2.002, respectively. In 1959, after five years of work by seven faculty members, a book named *Introduction to Solid Mechanics* (commonly referred to as Crandall, Dahl, and Lardner) was published [17]. Many engineering programs across the nation immediately adopted the use of this book [28]. This revolutionary text is still used in 2.001 today. Though many of the more competitive engineering programs have adopted the concept of consolidation of Statics and Strength of Materials, many engineering schools still offer separate courses for each of the elements of Applied Mechanics. Testament to this is the fact that Beer and Johnston's texts (*Statics* and *Dynamics*) are still used so prevalently [6].

The course has primarily been taught using traditional lecture methods. Even before this course was started in the 1950's - when the Department of Mechanical Engineering had the largest enrollment of any department at the Institute - all engineering students were required to take classes in applied mechanics. This included civil engineers, electrical engineers, and aeronautical engineers.[1] This was also a time when many more faculty were involved in teaching the course. Early on, as major engineering fields became more specialized, electrical engineers were no longer required to have such strong footing in Applied Mechanics. Civil engineers and aeronautical engineers still needed a basis in this area, but - starting in the 1970's - these departments took on the responsibility of teaching these subjects to their own undergraduates. Today most students enrolled in 2.001 are from the Department of Mechanical Engineering. As the course fulfills a requirement for

[1] A policy that there be no more than 30 students in a class once resulted in 22 sections for Solid Mechanics.

Ocean Engineering, Nuclear Engineering, and Mechanical Engineering minors, there are typically a few students from other departments.

In terms of other courses at MIT, 2.002 (*Mechanics and Materials II*), the follow-up course to 2.001, has used some different forms of physical experimentation through its history including “desktop experiments”, physical demonstrations, laboratory experimentation, and design exercises. The first “desktop experiment kit” correlated to the text written by the course instructors, Profs. Frank A. McClintock and Ali M. Argon; the text indicated problems that could be verified through experimentation [53]. More involved experiments were developed in the form of a shoebox of take-home experiments by Boyce, et al. (1995). The more formal laboratory component of the course has been in place since at least the 1970’s and has taken many forms. It has primarily been demonstration material, but in recent years has evolved to provide more hands-on involvement with the various phenomena; this has proven to be successful in imparting learning. For Mechanical Engineering fluid mechanics courses, Prof. Ascher H. Shapiro developed a set of videos of fluids experiments [92]. The professional creation of these videos - sponsored by the National Science Foundation (NSF) - made for an illustrative classroom aid. These videos were at first shown in class, were later displayed outside of class time and made optional, and finally were removed from the curriculum.

In terms of the Web, Prof. David Wallace experimented with Web-based lectures in the graduate level course 2.744 (*Product Design*) [104]. The Mechanical Engineering Hypermedia Project (MEHP) was another early attempt to bring Mechanical Engineering courses online, culminating in MIT's first Fluid Mechanics Hypercourse CD-ROM [70]. This idea, however, was not permanently integrated into the undergraduate curriculum.[2]

Aside from these few examples and upper-division lab classes, much of the Mechanical Engineering curriculum was - and is - based on lectures and textbook problems. Despite its root in “chalk and talk” lectures, the Massachusetts Institute of Technology (MIT) has recently become much more active in using technology to develop engineering education. These recent attempts collectively have the potential to create the most radical shift in education at MIT since the Institute’s founding in 1861.

1.2 Motivations

A 1995 report from the Board of Engineering Education identified many problems with undergraduate engineering education [20]. Though finding U.S. engineering programs to be the strongest in the world, the report criticized teaching styles, and the focus and flexibility of the typical undergraduate curriculum. Teaching styles were deemed to be highly ineffective for the masses. A “boot camp” mentality left many students with a weak understanding of fundamental engineering principles. This was evidenced by a nationwide meager 65% retention rate in engineering programs. In terms of content, the curricula at various universities often sacrificed the process of design for analysis methods. Many students graduated from college without a solid understanding of engineering design principles, leaving them ill-prepared for the real world. The relatively slow change to external stimuli, such as changes in industry and computing, left the Board to conclude that engineering curricula were excessively rigid.

[2] Almost all of the core undergraduate classes in the Department of Mechanical Engineering make use the Web, but the Web is not linked directly to the classroom, nor is its use particularly innovative; most of these classes use websites as a means of posting homework assignments, solutions, labs, and announcements.

In addition to these general criticisms of engineering education, problems specific to MIT’s Department of Mechanical Engineering program have also been identified. The original project proposal, drafted by six Mechanical Engineering faculty, stated that MIT’s most important internal asset has always been the quality of its people [10]. Despite this, the medium for transmission of knowledge from faculty to students has only changed marginally since the inception of the Institute. The need for MIT to utilize information technology tools in its educational process was motivated by several factors: competition from new universities, tuition-free universities and Web-based learning.

New universities, such as Olin College and Rowan University, are currently developing engineering programs from the ground up. Olin College, in Needham, Massachusetts, is the first new engineering college in the U.S. in nearly 40 years [105][69]. Rowan University, formerly Glassboro State College (NJ), is in the process of building up its young school of engineering and is quickly improving its national academic reputation [3]. Such programs are unhindered by the presence of existing curricula and infrastructure; the advantages to this include the ability to easily leverage technology and form programs that will appeal to today’s students.

Other top universities, such as Stanford and Duke, offer students reduced or waived tuition based on academic performance [100][58]. The Cooper Union, in New York City, has always operated as a tuition-free university and, though small, provides an excellent engineering education [14]. MIT’s policy is to offer financial aid only based on need [65]. In order to compete with universities which offer such financial incentives to highly qualified students, MIT must look for ways to distinguish its education: it must provide the best possible education to this talented pool of individuals.

The Internet age has also spawned websites specializing in education. Extending the successful model of video-based distance learning, made famous by Stanford University in the 1970’s, Web-based education offers students many benefits including an opportunity to review material at their own pace, inexpensive tuition, and flexible scheduling [25]. Though this will likely not pose a threat to MIT or to other top universities in the near future, MIT must be proactive in developing and advertising the advantages of a residence-based education. Key among these advantages is personal interaction with top faculty and gifted peers. Such personal interaction cannot be - at least at this point in our electronic evolution - adequately reproduced through digital media.

Aside from these external factors, there are a great many internal forces driving educational change at MIT. MIT’s internal mission to provide a world-class education to the best and brightest students in the world is key to its success in the future. Content development occurs frequently in courses at MIT, but the means of delivering the content has changed very little over the past 100 years. The advances in electronic communication in the last part of the 20th century necessitate that MIT - if it is to remain a world-class teaching institution - reevaluate its methods for information delivery.

1.3 Project Description

In October 1999, MIT and Microsoft announced an alliance to “conduct research and create new technologies that [would] improve information technology-enabled teaching models and educational tools for university education” [62]. Initially, the project would focus on "using information technology to enhance multimedia learning opportunities, designing a program that optimizes student learning in distance learning environments, and developing an educational curriculum that uses information technology to enhance collaborative learning" [11]. The purpose of this five-year effort - named “iCampus” - was to encourage cooperative research among students, researchers and faculty at MIT; and members of Microsoft Research. In an effort to achieve the most global impact, MIT and Microsoft agreed to produce material that would adhere to open standards, and to openly disseminate results and source code.

Within this framework, the Department of Mechanical Engineering presented a proposal to reform its undergraduate curriculum [10]. Three basic instructional paradigms were identified for development and evaluation over the course of this project: Scientific Discovery, Socratic, and Just-in-Time. The goal of Scientific Discovery is to allow students to learn through exploration. Through team-oriented student exploration, students observe physical phenomena. Then, with the guidance of an instructor, the students construct an analytical model of the phenomenon. The discrepancies between model and experiment are then discussed and students understand the limitations of their model and the reasons for these limitations. Such an approach is expected to create confidence in engineering principles, and to build a sense of how to design experiments and interpret results. This approach is the synonymous with the active engagement pedagogy mentioned earlier. The Socratic method is intended to allow students to engage in Web-based independent learning. These Web sessions are followed by small recitation-style mentoring sessions; this model is similar to the one used for 6.001 (*Structure and Interpretation of Computer Programs*) and 6.034 (*Artificial Intelligence*) in the MIT Department of Electrical Engineering and Computer Science (EECS). The Just-in-Time paradigm is mainly intended for project-centered courses. Students learn about topics as necessitated by the project. Such learning simulates a real engineering environment. The team environment and obvious objective give students a clear motivation to learn.

The traditional large-lecture format of *Mechanics and Materials I*, a first-year Mechanical Engineering course, provided an opportunity to examine the potential effectiveness of one of these paradigms: Scientific Discovery. The other methods would be tested on other classes at a later date.

Chapter 2

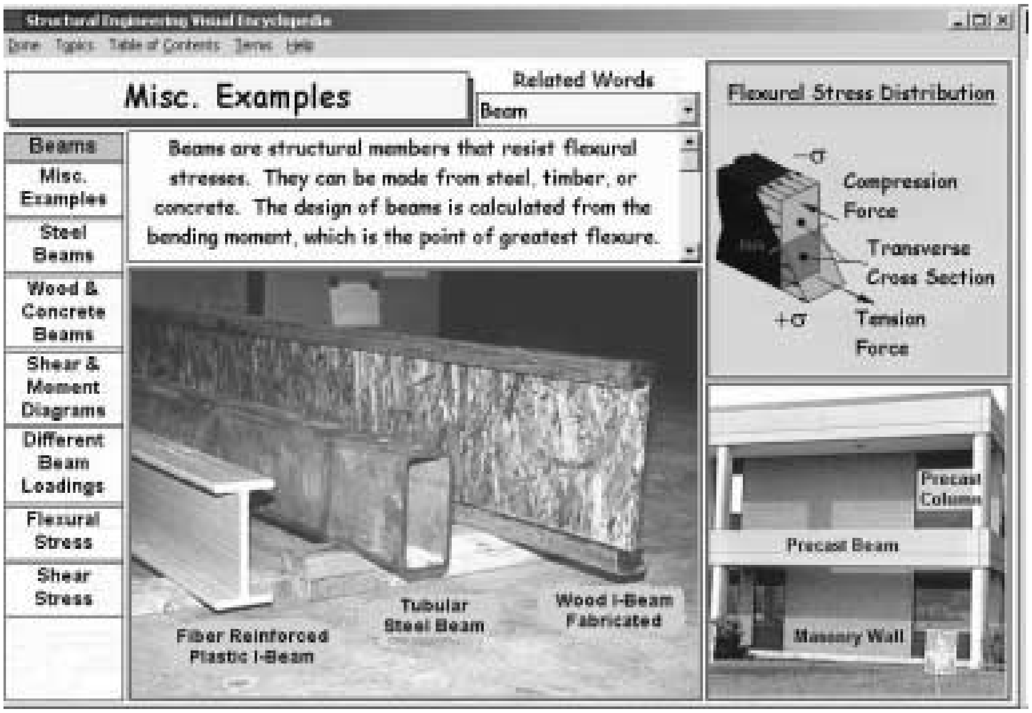
Prior Work

2.1 Novel Approaches to Teaching Mechanics

Solid Mechanics is a course that is taught across many disciplines. Courses that cover a substantial portion of 2.001 topics are taught within eight departments at MIT: Civil Engineering, Materials Science, Architecture, Physics, Earth Atmospheric and Planetary Sciences (EAPS), Ocean Engineering, Aeronautics and Astronautics, and Mechanical Engineering [50]. Additionally, the course 2.001 is an explicit requirement for students from Mechanical Engineering (including minors), Ocean Engineering, and Nuclear Engineering (Nuclear Energy Option).

Many of the experiments on - and advances in - mechanics education at the undergraduate level have focused on the use of computers in the curriculum. The work that has been done in recent years in this domain is presented here.

SEVE (Structural Engineering Visual Encyclopedia) is an educational software package, designed specifically for civil engineering students [30][83]. Developed by Prof. Robert Henry of the University of New Hampshire (UNH), it is distributed on CD-ROM. The purpose of this software address the omission of reading and interpretation of engineering construction drawings from most civil engineering curricula. The program provides access to common structural engineering terminology. The multimedia presentation of this information - text-based explanations, sketches, photographs, solid models, and animations - creates an engaging and interesting interface for students to learn (Figure 2.1).



Beams are structural members that resist flexural stresses. They can be made from steel, timber, or concrete. The design of beams is calculated from the bending moment, which is the point of greatest flexure.

Figure 2.1: SEVE (Structural Engineering Visual Encyclopedia), developed by Robert Henry of the University of New Hampshire [30].

The primary focus of this program is on structural engineering concepts and engineering terminology through the use of real world examples and applications. Three-dimensional models provide students with the ability to interact with and fully understand the applications of concepts. It is important to note that this single program has found applications across the Civil Engineering curriculum at UNH. SEVE is used in six classes at UNH: *Introduction to Civil Engineering Applications*, *Classical Structural Analysis*, *Reinforced Concrete Design*, *Timber Design*, *Matrix Structural Analysis*, and *Structural Design in Steel*. In 1998, SEVE received national recognition in the form of the Premiere Award for Excellence in Engineering Education Courseware, sponsored by NEEDS (National Engineering Education Delivery System) and John Wiley & Sons [68].

MDSolids - Mechanics of Deformable Solids - another software package that won the 1998 Premiere Award is an example of successful mechanics courseware [56][73].[3] Developed by Prof. Timothy Philpot at Murray State University (KY), MDSolids was originally distributed as a downloadable executable (.exe) program. It discretized Mechanics and Materials into 10 major sub-topics, and allowed students to explore each one (Figure 2.2). Within each topic, students had the opportunity to determine the best way to solve problems of mechanics. For example, in the Beam Bending module, students had an opportunity to receive instant feedback, in the form of shear and moment diagrams, on a beam problem they constructed.

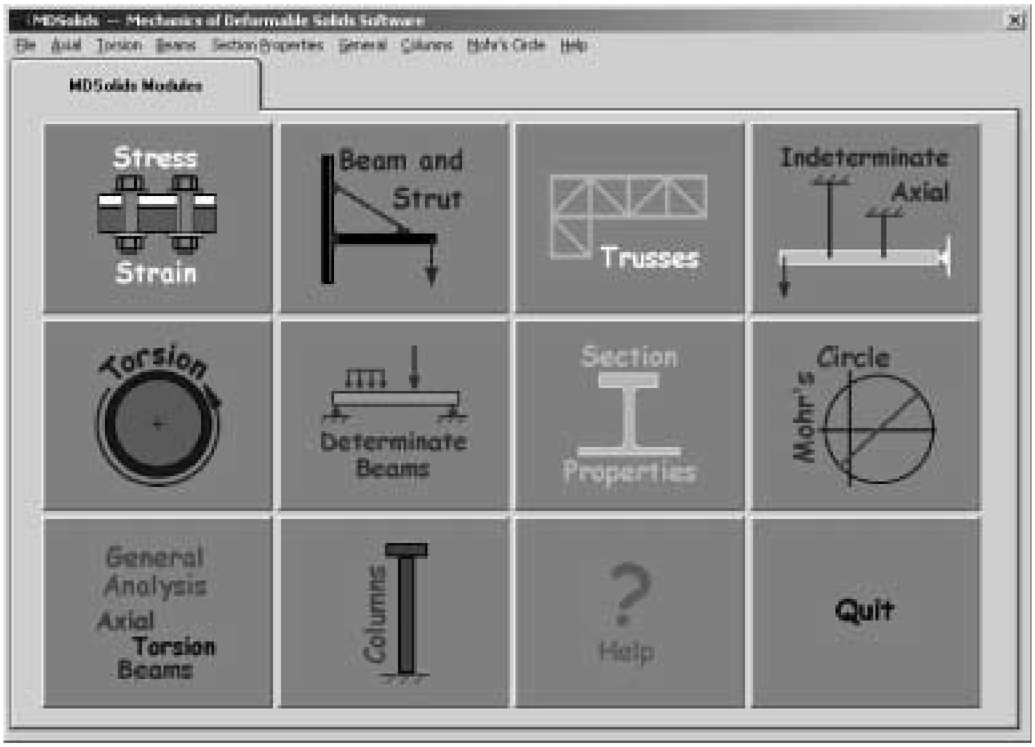

MDSolids — Mechanics of Deformable Solids Software
File Axial Torsion Beams Section Properties General Columns Mohr's Circle Help
MDSolids Modules
Stress Strain
Beam and Strut
Trusses
Indeterminate Axial
Torsion
Determinate Beams
Section Properties
Mohr's Circle
General Analysis Axial Torsion Beams
Columns
? Help
Quit

Figure 2.2: MDSolids, an educational package developed by Timothy Philpot at Murray State University [74].

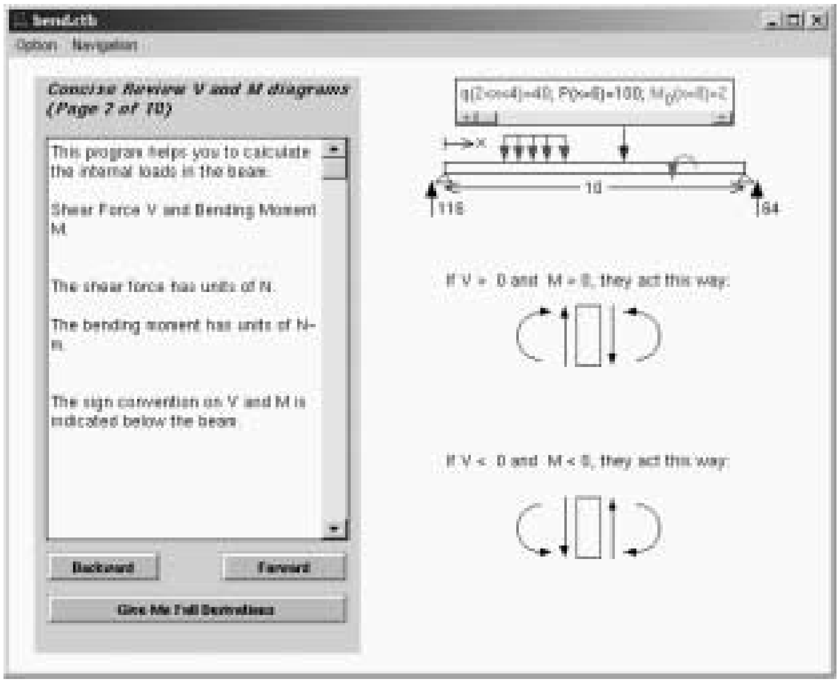
[3] http://www.mdsolids.com

The theory behind the approach of MDSolids was that students would be most interested in understanding the specific homework problems assigned by their professors, and that they would use the software if it helped them with their immediate course needs. Special attention was paid to creating an easy-to-use interface and a common visual language among topics, to allow students ease of navigation. Since 1998, the program has been adopted by several schools since its creation, including the University of Alberta and the University of South Florida [54][55]. It is now included on CD-ROM from Wiley and Sons with the purchase of Roy R. Craig Jr.'s *Mechanics of Materials*. The latest revision of MDSolids (MDSolids 2.0) includes modules that incorporate three-dimensional renderings and animations. Adding to the appeal of this program is that it is available for $25 to students as shareware, and it is free to universities and colleges who incorporate it as part of their curriculum. The ease of use and low cost make this an attractive package for instructor of Solid Mechanics courses.

StressAlyzer, a package developed by Prof. Paul Steif at Carnegie Mellon University, is similar to MDSolids [97].[4] StressAlyzer is intended as a teaching supplement to courses in Mechanics of Materials; the package is segmented into 6 topics: axial loading, torsional loading, shear force and bending moment diagrams, load and stress calculations, beam deflections, and stress transformations.

This is not a comprehensive list of the topics covered in most Solid Mechanics classes, but instead engages the most difficult topics encountered by students in such a course.

[4] http://www.me.cmu.edu/stressalyzer/



Concise Review V and M diagrams (Page 7 of 10)

This program helps you to calculate the internal loads in the beam.

Shear Force V and Bending Moment M.

The shear force has units of N.

The bending moment has units of N-m.

The sign convention on V and M is indicated below the beam.

If V > 0 and M > 0, they act this way:

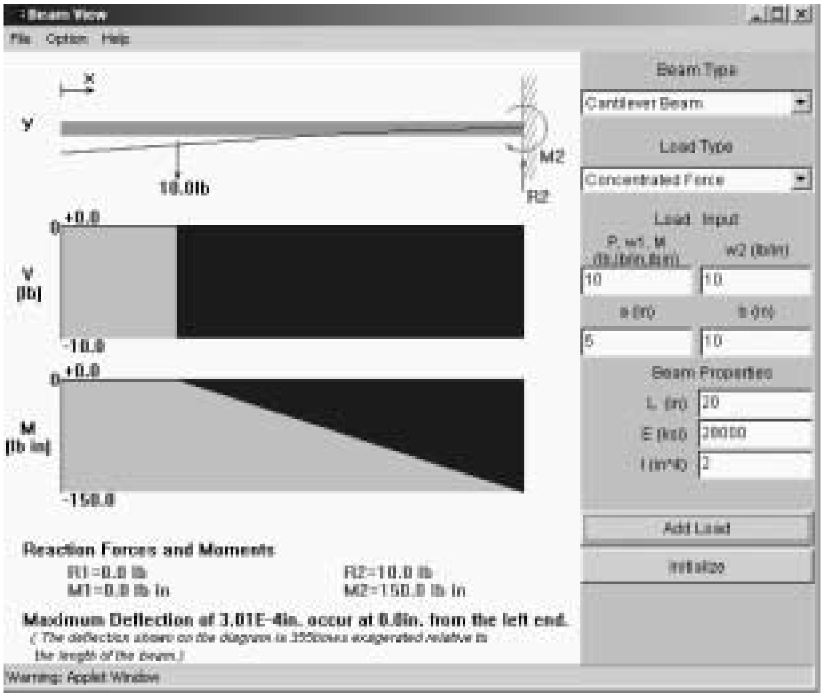
If V < 0 and M < 0, they act this way:

Figure 2.3: StressAlyzer, mechanics courseware authored by Paul Steif of Carnegie Mellon University [96].

The basic format of StressAlyzer is that of a self-testing study guide (Figure 2.3). Concepts are explained to students and they are periodically asked questions. The student answers take the form of text input. They are provided with instant feedback including reasons for incorrect answers as appropriate. The benefit of such a system is that it fully engages the student and provides real-time feedback. Additionally the system allows students to submit homework electronically and has a common interface across all modules. One major drawback is that a specific pace is set for all students. Students who fully understand concepts will still have to engage the simple questions in using StressAlyzer. Also, notable is the recurrence of the two-dimensional engineering schematic, without real examples or applications of concepts.

The Aerospace and Ocean Engineering (AOE) and Civil Engineering Departments at Virginia Polytechnic Institute and State University (Virginia Tech) have developed a suite of Java applets for use in engineering courses [39][84].[5] "Java Applets for Engineering Education", a project headed up by Profs. William Devenport, Rakesh Kapania, Kamal Rojiani, and Kusum Singh, aims to utilize the universality of Java applets while addressing the need for easily-accessible learning tools (Figure 2.4). Java, a language developed by Sun Microsystems, can run across many platforms [40]. A Java applet is a "small application" that can run from within a Web browser. The applet is restricted from accessing the local computers file system so that users can download the applet without security concerns. The major benefits of applets are the ability to run in Web browsers, the ability to run on multiple platforms, and the fact that they have almost as much power as stand-alone applications.

[5] http://www.engapplets.vt.edu



Cantilever Beam

Concentrated Force

P, w1, M (lb,lb/in,lb in): 10
w2 (lb/in): 10
a (in): 5
b (in): 10

L (in): 20
E (ksi): 30000
I (in^4): 2

Reaction Forces and Moments
R1=0.0 lb
M1=0.0 lb in
R2=10.0 lb
M2=150.0 lb in

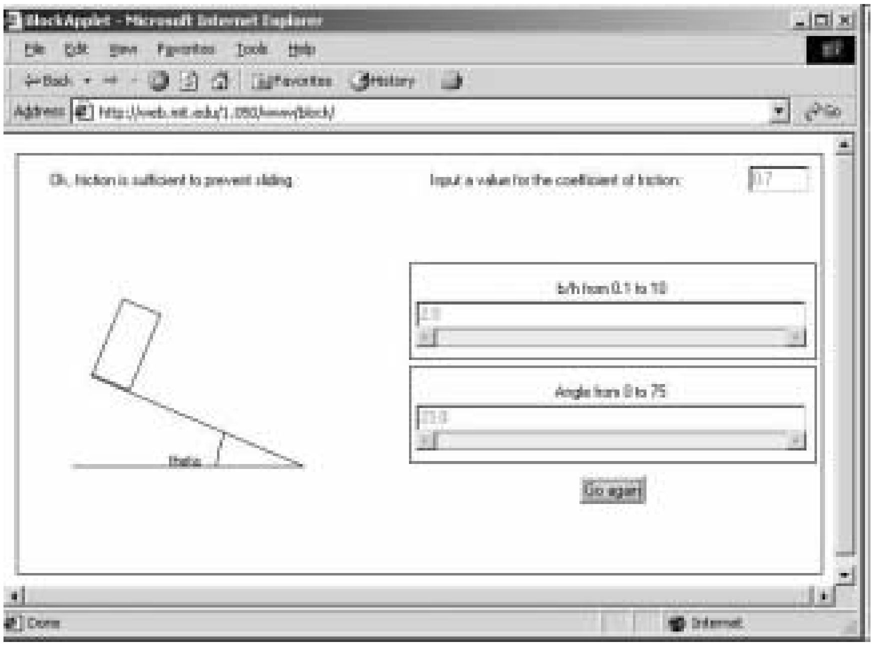
Maximum Deflection of 3.01E-4in. occur at 0.0in. from the left end.
(The deflection shown on the diagram is 3550times exagerated relative to the length of the beam.)

Warning: Applet Window

Figure 2.4: Beam bending applet from Java Applets for Engineering Education, a joint project of the Aerospace and Ocean Engineering and Civil Engineering Departments at Virginia Tech [39].

The applets from Virginia Tech cover the following major engineering topics: Statics, Dynamics, and Fluid Dynamics. The major analog to Mechanics are the applets created for Statics: Mohr’s Circle, Beam Analysis, and Resultant Vectors. The Mohr’s Circle applets show Mohr’s circles for different inputs in two and three dimensions. The Beam Analysis applet takes several inputs and displays displacement, shear force, and moment diagrams for the beam. This is the most extensive of the Statics applets. The Resultant Vectors applet simply displays a resultant vector based on x and y force inputs. Much of the focus appears to be on topics other than statics. A recurring concern is the potential utility of some of these applets, since some cover trivial topics, and others are computationally intensive. It is also unclear how these applets are used in the engineering curricula in various departments at Virginia Tech.

A similar project in the MIT Department of Civil Engineering gave birth to several applets for use in mechanics education [21].[6] This project, headed up by Prof. Louis Bucciarelli, developed a set of applets to be used for the Civil Engineering course 1.050 (*Engineering Mechanics for Structures*). There were six applets in total, covering the following topics: truss structures, frame structures, friction (sliding block), equilibrium (roller over a bump), vectors, and steel sections (Figure 2.5). The greatest strength of these applets was their range of application; two were design or analysis tools, two others were simulations, and one was a game.


BlockApplet - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Favorites History
Address http://web.mit.edu/1.050/www/block/
Go
theta
Done
Internet

Ok, friction is sufficient to prevent sliding

Input a value for the coefficient of friction: 0.7

b/h from 0.1 to 10: 2.0

Angle from 0 to 75: 23.0

Go again

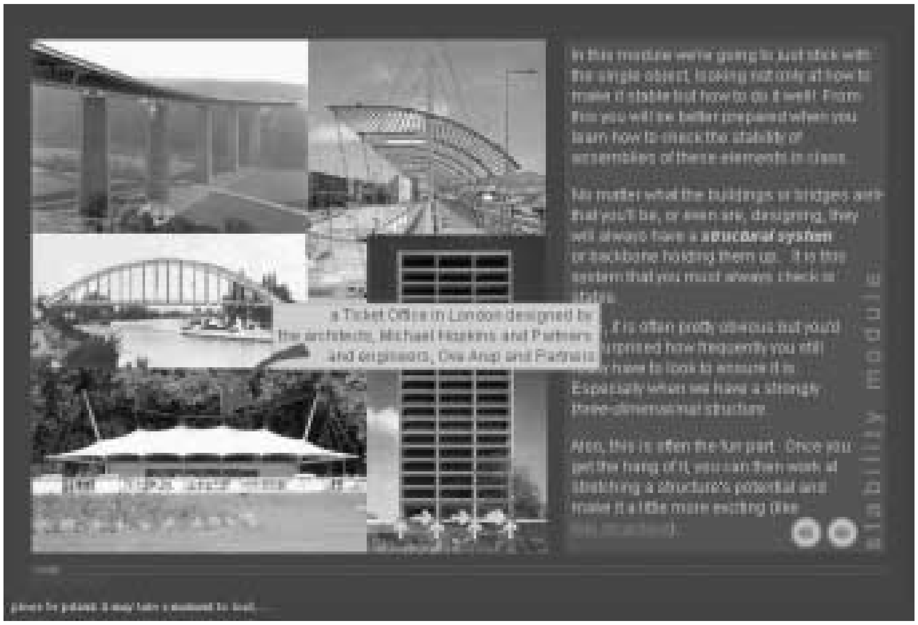
Figure 2.5: Friction applet from 1.050 (*Engineering Mechanics for Structures*), an MIT Civil Engineering course [21].

Dr. Emma Shepherdson, also from the MIT Department of Civil Engineering, investigated active learning environments in her doctoral work [67][93].[7] This yielded a Macro-

[6] http://web.mit.edu/1.050/www/

[7] http://moment.mit.edu

media Shockwave-enabled, Web-based learning module on basic statics (Figure 2.6). The students interacted with the module and were able to get different feedback for different responses. The design of the module involved development of specific interactive exercises to address specific key concepts. For example, students examining stability would be presented with multiple choice questions regarding an earlier beam problem. As another example, students who were studying mechanical assemblies would be presented with a simulation combining previous concepts. They would actively break down an assembly into its key components and check each element for independent stability, thus determining if the overall structure was stable and how many levels of redundancy existed.



In this module we're going to just stick with the single object, looking not only at how to make it stable but how to do it well! From this you will be better prepared when you learn how to check the stability of assemblies of these elements in class.

No matter what the buildings or bridges are that you'll be, or even are, designing, they will always have a **structural system** or backbone holding them up. It is this system that you must always check is stable.

, it is often pretty obvious but you'd urprised how frequently you still y have to look to ensure it is. Especially when we have a strongly three-dimensional structure.

Also, this is often the fun part. Once you get the hang of it, you can then work at stretching a structure's potential and make it a little more exciting (like this structure).

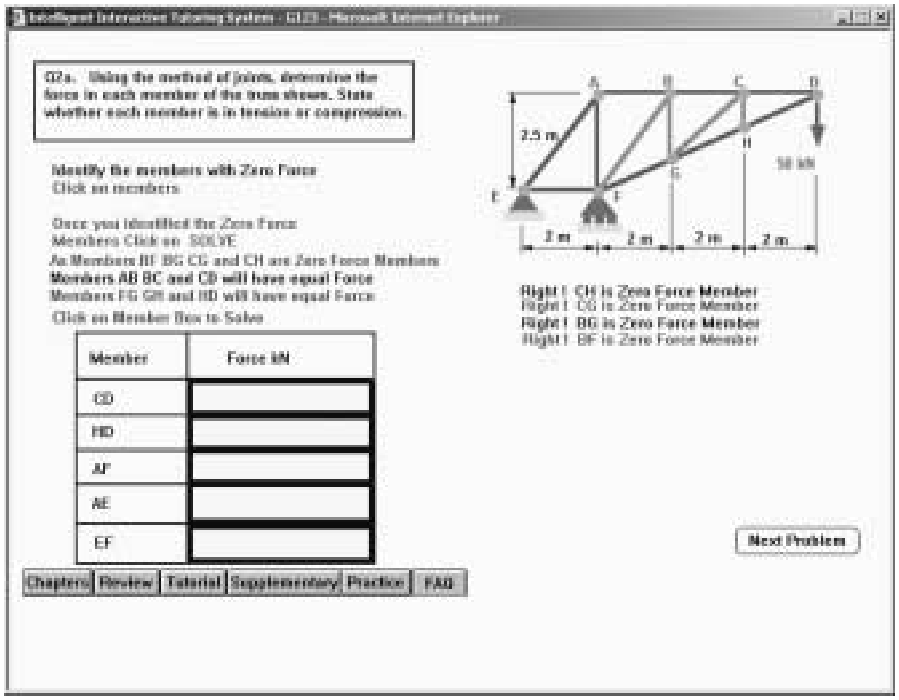
a Ticket Office in London designed by the architects, Michael Hopkins and Partners and engineers, Ove Arup and Partners

please be patient, it may take a moment to load...

Figure 2.6: Emma Shepherdson's Web-based learning module on basic statics from the MIT Department of Civil and Environmental Engineering [67].

The major strength of this approach was the interactivity which it offered the student. The module could identify different misconceptions, acting as a somewhat intelligent tutor. Potential drawbacks include the overhead time involved in developing such a tool, the skill necessary to modify such a tool, and the dependence on proprietary technology (Shockwave).

Faculty at Nanyang Technical University (NTU) in Singapore investigated the potential of a system entitled Intelligent Interactive Tutoring System [95]. The system was developed for use in the engineering mechanics course taken by first-year students; this course typically has an enrollment of 1,600 students. The authors opted for developing their own software out of a determination that commercially-available software did not fulfill the need for an intelligent system. The key objective in designing this system was to develop an intelligent and interactive tutor, one that would provide appropriate guidance as opposed to a "fancy electronic page-turner." Soh and Gupta also point out that the system was designed to be portable, user-friendly, and self-paced.



Q2a. Using the method of joints, determine the force in each member of the truss shown. State whether each member is in tension or compression.

Identify the members with Zero Force
Click on members

Once you identified the Zero Force
Members Click on SOLVE
As Members BF BG CG and CH are Zero Force Members
Members AB BC and CD will have equal Force
Members FG GH and HD will have equal Force
Click on Member Box to Solve

Member	Force kN
CD	
HD	
AF	
AE	
EF	

Right ! CH is Zero Force Member
Right ! CG is Zero Force Member
Right ! BG is Zero Force Member
Right ! BF is Zero Force Member

Figure 2.7: Intelligent Interactive Tutoring System from Nanyang Technological University (NTU), Singapore [36].

The system was segmented into 8 major chapters, including equilibrium of rigid bodies, truss analysis, and forces in beams (Figure 2.7).[8] Each chapter included a Review, Tutorial, Supplementary, Practice, and Frequently Asked Questions (FAQ) module. The Review was intended to be a concise treatment of topics of the chapter. The Tutorial guided students through problem solving step-by-step. The Supplementary problems were more advanced problems that the students were expected to do on their own, though they could seek assistance if needed. The Practice section contained more difficult, optional problems. Since tutorial help was offered, Soh and Gupta expected that serious students would be encouraged to attempt them. Finally, the FAQ section answered questions commonly asked by students of the topics in the chapter. The system was developed using Authorware 4.0 (1997), a tool which enables the creation of executable courseware. The program was scaled from an initial trial of 200 students to the entire 1600 student population over the period of a year. The initial results were positive, with both students and instructors responding favorably.

The intelligence of this system is a benefit, though the term “intelligence” may be misleading. In this context, it implies that the authors have developed a pre-determined set of responses to specific inputs. This is a feature available in some of the other courseware presented here. The other design parameters mentioned - ease of use, portability, and self-paced - may seem obvious, but not all systems implemented these parameters. These were all important aspects of the development of the Web-based modules used in 2.001.

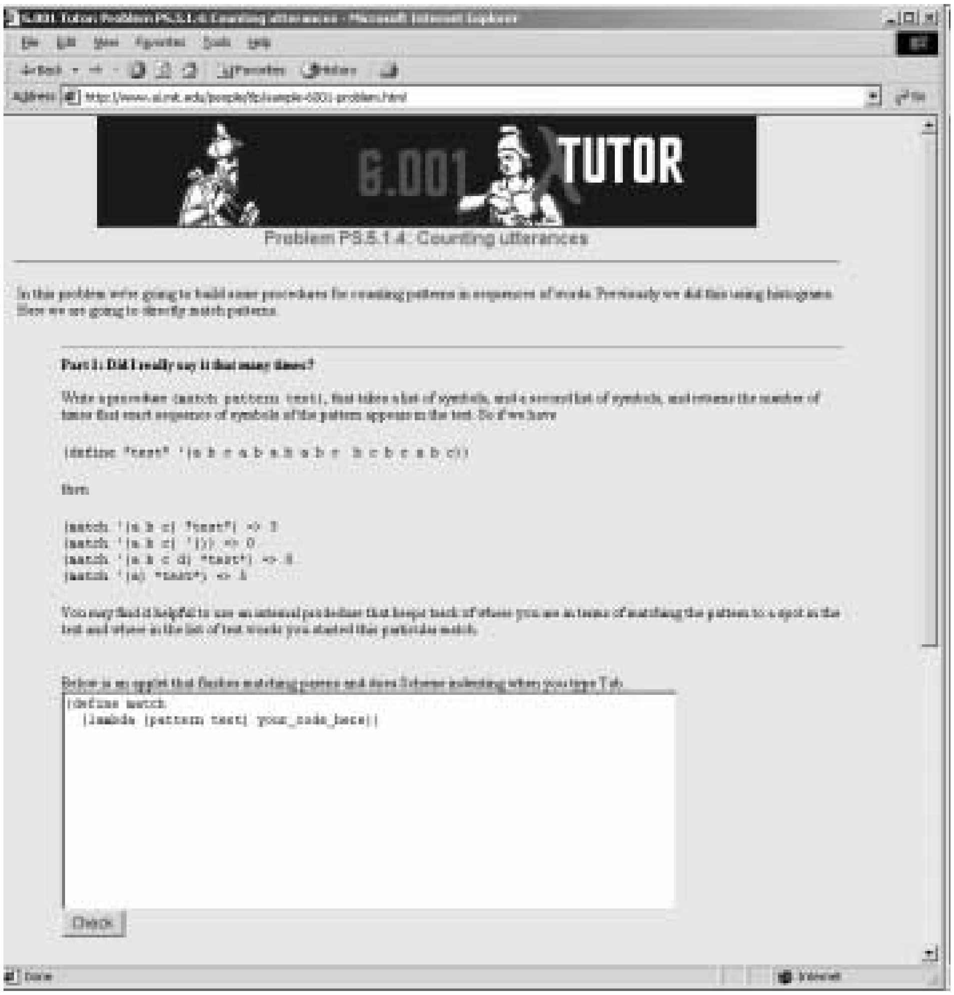
2.2 Development of Educational Technology at MIT

MIT is currently in the midst of a radical, large-scale overhaul of its educational system.

[8] http://cse.ntu.edu.sg/iits/

In April 2001, MIT announced that it would make the materials from almost all of its courses freely available on the World Wide Web for non-commercial use [64]. This initiative, called MIT OpenCourseWare (OCW), is intended to lead other universities into a new era of knowledge dissemination. The OCW initiative instantly received international recognition and was lauded by many in the academic community [26][98][107]. This announcement was typical of MIT's new focus and perspective on higher education in the Internet age.

The current educational change is fueled largely by the Microsoft-MIT iCampus Initiative and the Alex and Brit d'Arbeloff Fund for Excellence in MIT Education; these groups are enabling departments across the Institute to engage in projects to enhance education through the innovative use of information technology. While this is by no means a comprehensive treatment of the current educational research at MIT, it does represent a sampling of the projects which have been implemented to a significant degree. Many of these projects have transcended the traditional barriers between departments and have resulted in many inter-departmental, collaborative efforts. This environment of change was critical in facilitating the execution of this project, and in providing a community and resources for development of these new educational concepts for 2.001.


6.001 Tutor: Problem PS.5.1.4: Counting utterances - Microsoft Internet Explorer
File
Edit
View
Favorites
Tools
Help
Back
Favorites
History
Address
http://www.ai.mit.edu/people/tlp/sample-6001-problem.html
Go
6.001
TUTOR
Done
Internet

Problem PS.5.1.4: Counting utterances

In this problem we're going to build some procedures for counting patterns in sequences of words. Previously we did this using histograms. Here we are going to directly match patterns.

Part 1: Did I really say it that many times?

Write a procedure (match pattern text), that takes a list of symbols, and a second list of symbols, and returns the number of times that exact sequence of symbols of the pattern appears in the text. So if we have

```
(define *text* '(a b c a b a b a b c  b c b c a b c))
```

then:

```
(match '(a b c) *text*) => 3
(match '(a b c) '()) => 0
(match '(a b c d) *text*) => 0
(match '(a) *text*) => 5
```

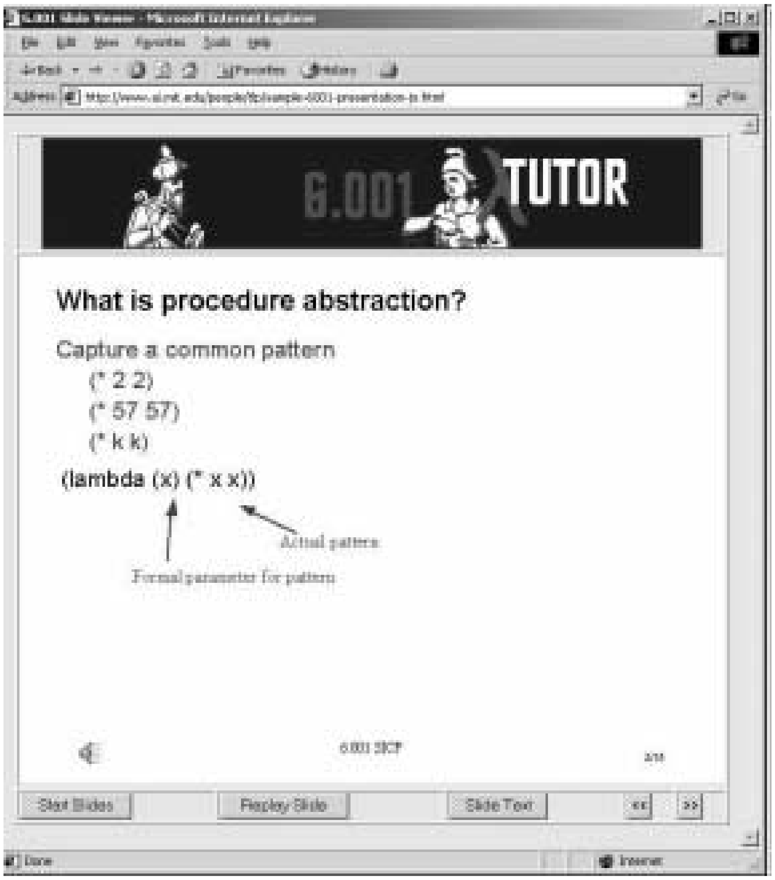
You may find it helpful to use an internal procedure that keeps track of where you are in terms of matching the pattern to a spot in the text and where in the list of text words you started this particular match.

Below is an applet that flashes matching parens and does Scheme indenting when you type Tab.

```
(define match
  (lambda (pattern text) your_code_here))
```

Figure 2.8: Sample homework problem given to students in 6.001 (*Structure and Interpretation of Computer Programs*) [1].

Prof. Tomas Lozano-Perez, of the MIT Department of Electrical Engineering and Computer Science (EECS) has been engaged in using the Web to enable teaching at the undergraduate level [99]. 6.001 (*Structure and Interpretation of Computer Programs*) and 6.034 (*Artificial Intelligence*) were both taught using online homework assignments and lectures beginning in 2000 [1][2]. The traditional lecture has been eliminated. Instead of attending a physical lecture, students review online lectures at their own convenience (Figure 2.8). Then, students meet in small, faculty-led tutorial sessions that, because of their intimacy, allow students the ability to freely ask questions. This class-style follows the Socratic paradigm discussed in Chapter 1. Homework assignments are also completed online (Figure 2.9).



What is procedure abstraction?

Capture a common pattern

```
    (* 2 2)
    (* 57 57)
    (* k k)
(lambda (x) (* x x))
```

Figure 2.9: Slide from a Web-based lecture for 6.001. This is accompanied by an audio or text-based narration, depending on the student's preference [1].

Both courses make extensive use of Scheme, a dialect of Lisp developed at MIT [91]. Because of the use of a common programming language for the two classes, developing an infrastructure for the homework assignments was a simplified process. Students were often asked to submit fragments of code through the Web interface; this code was compiled and run against evaluation code. Given the deterministic nature of computer programs, the Scheme compiler performed much of the "intelligent" work, evaluating student answers. Students were then presented with real-time information on their success. This system was beneficial in that it allowed students the freedom to review material at their own pace, allowed students to interact with faculty in a more intimate setting, and provided real-time feedback on homework problems, allowing students to better understand their own misconceptions. Based on the initial success of these two classes, in fall 2001, this program was expanded to use in another undergraduate course, 6.004 (*Computation Structures*).

The MIT Department of Physics, in conjunction with the MIT Center for Advanced Educational Services (CAES), developed an online tutoring system for 8.01 (*Introductory Classical Mechanics*) that was first used during fall 2000 [76]. The system, named PIVoT (Physics Interactive Video Tutor), and based on the lectures of well-known Prof. Walter Lewin, was used at 3 schools: MIT, Rensselaer Polytechnic Institute (RPI), and Wellesley College (Figure 2.10).

Figure 2.10: PIVoT, the Physics Interactive Video Tutor, used by the MIT Department of Physics [76].

PIVoT provides a comprehensive online resource for physics students. The system offers: digital video clips in which Prof. Lewin explains difficult concepts, demonstrations of physics principles, step-by-step solutions to example problems, 35 of Prof. Lewin's lectures in streaming video, an online textbook, physics simulations, and practice problems. An additional feature is a "Personal Tutor," an intelligent agent that provides individualized help based on individual use of the site. Since this initiative started several terms ago, there has been useful feedback regarding the effectiveness of PIVoT.[9] Specifically, PIVoT has been shown to be useful in developing conceptual understanding, primarily for students with a previous physics background; for students with a weaker preparation in physics, PIVoT users did not show any appreciable gain in conceptual understanding. PIVoT did not seem to affect exam or course grades - the assessors of the project speculate that this may be because exams test both conceptual understanding and algorithmic problem-solving capabilities.

[9] Evaluation methods took the form of an in-depth study at RPI and surveys at all three schools. At RPI, students were divided into two groups, one that used PIVoT and one that did not. Both groups were given conceptual diagnostic exams at the beginning and end of the semester in order to determine gains in conceptual understanding. RPI also collected data on grade performance, site usage, and student participation.

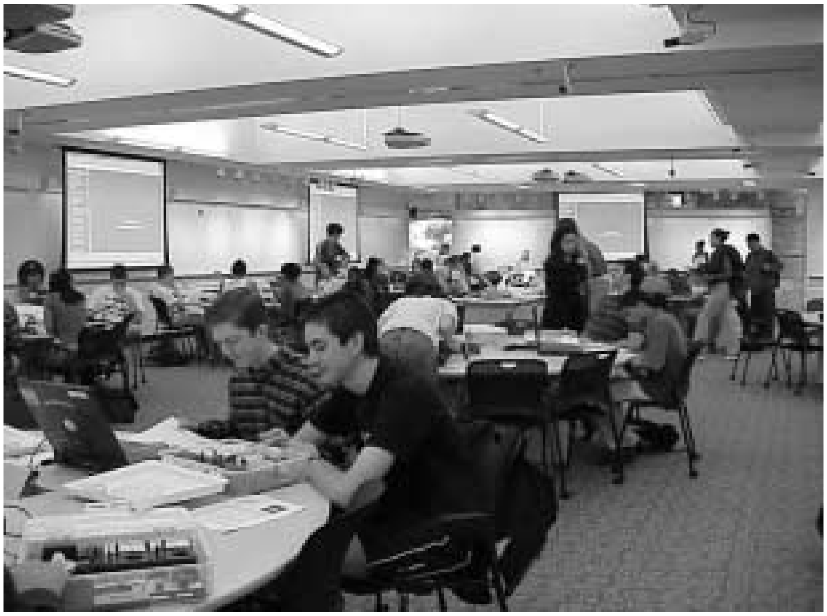


Figure 2.11: An 8.02T (*Electricity and Magnetism*) lecture. Students work in teams on problems and experiments, using laptops to aid in visualization, data acquisition, and data analysis [35].

Prof. John W. Belcher, also of the Department of Physics, is leading an effort called Technology Enabled Active Learning, or TEAL [35]. The focus of this project is introductory physics classes, which typically have large enrollments and no laboratory component. Since physics is by nature an experimental science, Belcher developed a new course, called 8.02T (*Electricity and Magnetism*), that would enable students to participate in hands-on experiments. Following the Studio Physics format of Prof. Jack Wilson at Rensselaer Polytechnic Institute in 1994, the TEAL format combines lecture, recitation, and hands-on laboratory experiments into one classroom experience (Figure 2.12, Figure 2.13). In this sense, there are similarities between 8.02T and the new form of 2.001. Animations and simulations are incorporated into course materials to help students visualize and understand difficult concepts [75].

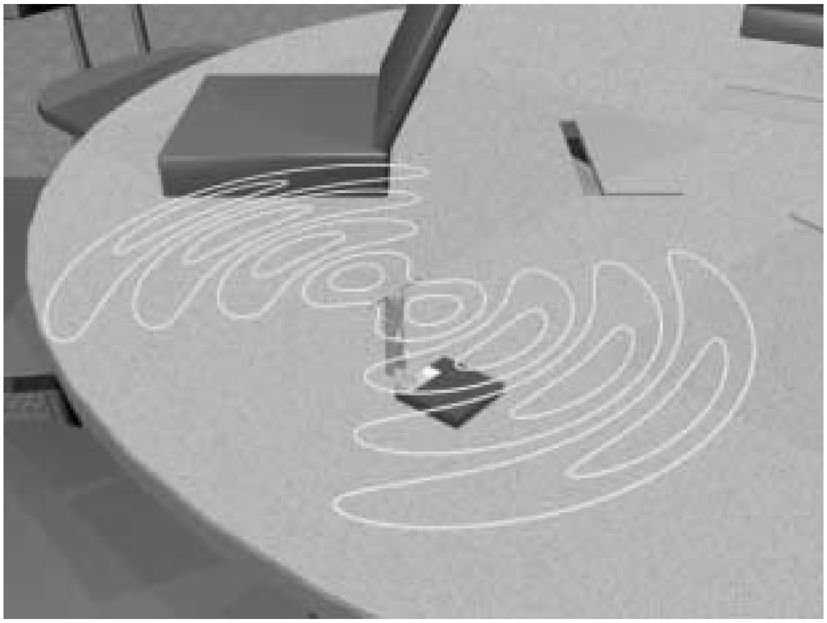


Figure 2.12: A simulation from 8.02T. Students use laptops to acquire and interpret data from a physical experiment [35].

A new classroom, designed for the different components of the TEAL format, was built. This classroom consists of 11 round tables that seat 9 students each. A centrally-located station allows the instructor to present material onto eight projection screens located around the perimeter of the room. Numerous whiteboards allow for impromptu discussions and presentations by both staff and students. On each table are three laptops, provided for the students to work in teams of three on experiments and problems assigned in class. Details of the room configuration are critical to the learning process, and much of the room design was based on the work of others in this domain [7].[10] Homework assignments are submitted electronically using WebAssign, an external homework delivery system [106].

[10] The design of the physical space was based primarily on the work of Prof. Robert J. Beichner, Director of the Physics Education R & D Group at North Carolina State University.

The TEAL format was first introduced in fall 2001, and it has been very successful according to both students and faculty [13]. Students have had positive feedback about the hands-on components and attendance has been much higher than in the past. Plans are in place to expand the project next year, by including more students in the class. It is eventually hoped that all introductory physics classes will be taught using the TEAL format.

Another example of technology-enabled education at MIT is the Personal Response System (PRS) used by the Department of Aeronautics and Astronautics (Aero/Astro) [22]. Students are supplied with a pocket-size wireless transmitter. This transmitter can then be used in the classroom to communicate with the instructor. This system presupposes that all questions asked of students can be cast in multiple-choice form. The PRS allows students to use their transmitters to respond to the posed question; the signals are received by a portable receiver that is connected to a central display. The display can then be used to view histograms of student responses. Students - in addition to being able to submit answers - can attach a confidence level (high, medium, low) to the answer sent. The instructor has the option of tracking specific students or leaving the system in anonymous mode, which prevents the instructor from determining how specific students responded. In known mode, the instructor can write a file containing student responses to disk for later review.

This system was first used in Unified Engineering, an Aero/Astro sophomore-level course block that consists of 4 courses taken over 2 semesters [72]. The success in these courses has led the Department of Aeronautics and Astronautics to adopt the system in other classes as well. There are a great many advantages to this system over the traditional methods of encouraging student participation during lecture. Such advantages are not unique to the PRS; they are prevalent in similar methods of allowing anonymous student participation [22]. The instructor - especially during large lectures - may have a difficult time determining whether he or she is proceeding too quickly or too slowly with the material. By breaking up the lecture with mini-quizzes, the instructor can more precisely gauge student perception. Since many students are often afraid to ask questions, this provides a clear basis for what does and what does not need to be reviewed. These tools allow the lecturer to better manage class time and assess student needs in real-time.

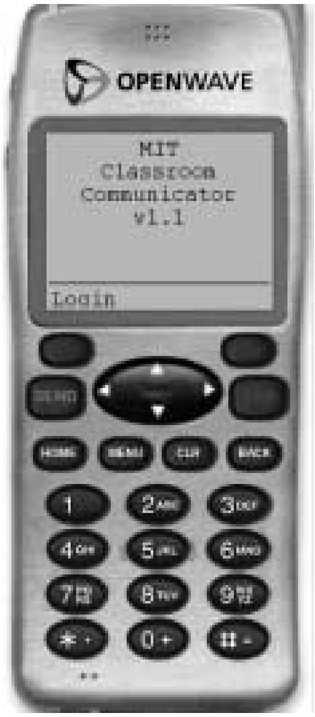


Figure 2.13: The Classroom Communicator, a system that allows courses to utilize Web-enabled cell phones for student-instructor interaction [22].

A related system is the Classroom Communicator, developed by Eric Brittain, an MIT doctoral candidate in the Department of Electrical Engineering and Computer Science [22]. The system is based on the use of Web-enabled cell phones (Figure 2.13). Though similar in concept to the PRS, the Classroom Communicator also allows students to ask specific questions of the instructor during lecture. The obvious drawbacks are the cost of these phones (though they do not need to be in active service to work with the system) and the cumbersome process of entering in questions on a numeric keypad. Another potential disadvantage to these systems is that, if not used carefully, students may not be encouraged to speak up in class. Consequences of this are that students will not gain confidence in defending their ideas or speaking in public.

The iLab project is based on the theory that students are better motivated and can learn more effectively if they have the opportunity to conduct experiments [33][63]. The significant expense and space considerations associated with a course laboratory prevent many engineering courses from having a lab component. iLab, led by Prof. Jesús A. del Alamo, is creating remote Web-accessible laboratories that will provide a new framework for science and engineering courses. Remote laboratories allow for much more efficient use of laboratory equipment and give students the opportunity to conduct experiments from the comfort of an Internet browser. They additionally provide the capability for conducting sophisticated experiments located at various company sites, increasing the range of possible laboratory experiences.

The iLab concept was first implemented for a Microelectronics WebLab. Students were able to acquire data in real-time on the latest CMOS (Complementary Metal Oxide Semiconductor) hardware, which was located at Compaq's Alpha Development Group center in Shrewsbury, MA. This WebLab was used in three courses: 6.012 (*Microelectronics Devices and Circuits*), 6.720J/3.43J (*Integrated Microelectronics Devices*), and SMA5104 (*Fundamentals of Semiconductor Device Physics*). Based on the success of the Microelectronics WebLab, the iLab group developed an Instrumentalized Flagpole WebLab which has been used in three Civil Engineering courses. Work is also in progress on WebLabs for Chemical Engineering (Heat Exchanger) and Aeronautics and Astronautics (Mechanical Structures).

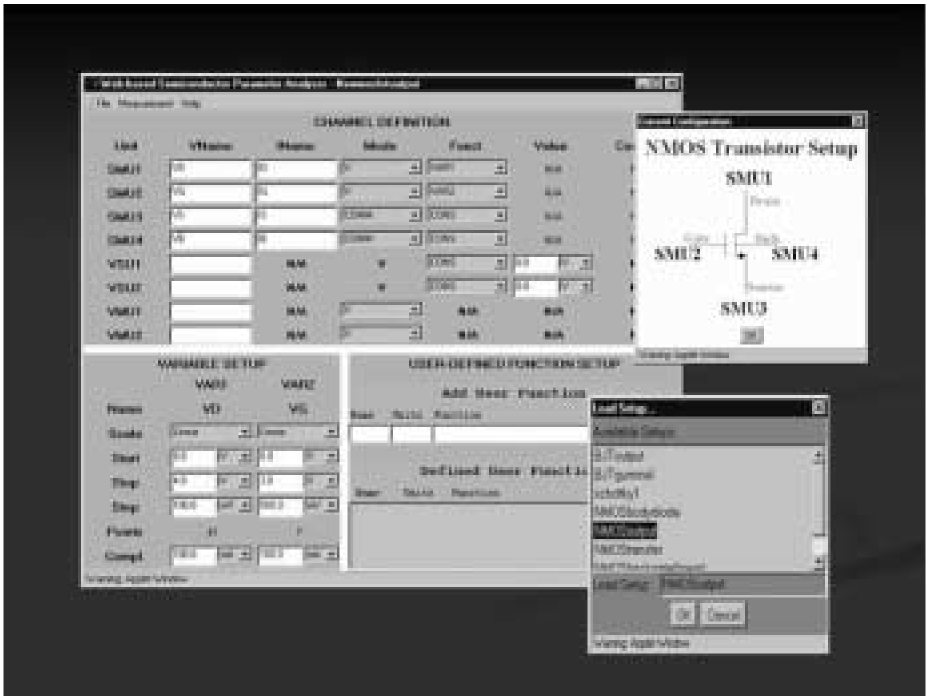


Figure 2.14: Web-based control console used in Microelectronics WebLab [33].

The Department of Architecture StudioMIT project seeks to create “a new approach to educational technology approach - one that emphasizes creative learning communities” [34]. Dean William J. Mitchell (School of Architecture and Planning), Prof. Susan Yee (Department of Architecture), and Lili Cheng (Microsoft Research Social Computing Group) have outlined a strategy that builds upon the familiar and successful studio approach to education in architecture. The studio approach, a problem-oriented environment, offers students the full set of necessary resources, a support community, and the ability to participate in collaboration, discussion, and criticism. StudioMIT is about carefully building upon this infrastructure, rather than seeking to replace it.

The main objective of StudioMIT is to create a Web-based community for students, staff, and faculty of MIT's studio-centered professional degree programs; however, it seeks to be much more than online delivery of content. The heart of the system is an open-ended collection of continually updated, searchable databases. All members of the Web community have individual workspaces which provide them with personalized access to resources, and an ability to represent themselves to other members of the studio community. The environment also provides an electronic means for collaboration and discussion, as also occurs in the physical studio environment. During fall 2001, this project was in its initial phase; StudioMIT was tested in a select set of pilot classes by students, faculty, and teaching assistants. This project is facilitated by a portable computing initiative that has provided students with laptop computers. These computers provide constant access to the StudioMIT environment, which provides capabilities absent from the traditional studio environment (Figure 2.15).

Figure 2.15: A StudioMIT student using her laptop at a design site [34].

These projects are representative of an environment that has not been seen at MIT before. There is widespread support for the creative use of technology in education; departments across MIT are beginning to investigate how educational technology and curriculum reform may improve education. This atmosphere has been critical in fostering the development and support of the 2.001 experiment.

Chapter 3

Pedagogy

3.1 Teaching Paradigm

As was mentioned previously, 2.001 has traditionally been taught in the "chalk and talk" pedagogy. Lectures and recitations, with little student-faculty interaction, even less student peer-interaction, and no experimental component, comprised the class. The new teaching paradigm, in stark contrast with the traditional lecture format, incorporates components of faculty-facilitated learning, hands-on experiments, group discussion, Web-enabled exploration (using laptop computers), and peer learning.

Prior to fall 2001, a 2.001 student would have the following weekly schedule:

- three 1-hour lectures
- one 2-hour faculty-led recitation section
- optional TA-led tutorial

Prior to fall 2000, this was no experimental component to the class. Beginning in fall 2000, though, students participated in hands-on desktop experiments during the 2-hour recitation sections. This enabled faculty to interact with students on a more intimate level, exposed students to elements of teamwork, and allowed for peer learning to take place. Beginning in fall 2001, the course took on an entirely new format.

A 2.001 student's schedule would now consist of:

- two 2-hour lectures
- one 1-hour faculty-led recitation section
- optional TA-led tutorial

The shift to longer lectures was necessary because of the way that class time would be utilized. The lecture would now consist of lecture, desktop exploration, laptop use, and peer interaction. The recitation sections were designated as opportunities to review lecture topics and to solve problems. By bringing both the desktop experiments and the use of the laptops into the classroom, it was deemed necessary to lengthen the lectures. Most lectures at MIT are 50 or 80 minutes; the overhead time associated with the physical experiments and the use of the laptops required 2 hours [88].

This class timing will change again in spring 2002 to the following format:

- one 2 hour lecture
- one 1 1/2 hour lecture
- one 1 1/2 hour faculty-led recitation section
- optional TA-led tutorial

The major differences between fall 2001 and spring 2002 are that some of the desktop experiments will be used in the recitation sections and that the “pure lecture” sessions will be conducted during the shorter lecture period. The timing of the course is constantly evolving as assessment provides information on how to better integrate the course elements.

The decision to use both physical experiments and computer-based simulations was a conscious one. Students have extensive experience and comfort with computers, and learning using computers. We are striving to take advantage of this comfort level by developing Web-based learning modules which they can actively engage. These modules have been created for each of the major course topics and serve multiple roles: an interactive learning tool, a basis for class discussion, a study aid, an in-class faculty-student communicator. In this manner, these modules enhance student communication, retention, and enthusiasm. The laptop computers allow them to engage the Web-based learning modules both in class and at their own convenience.

Students need more physical interaction due to the reduction of such experiences in life; such interaction enables students to develop a "feel" for phenomena. Desktop experiments that demonstrate particular phenomena represent the physical interaction of the new course. The experiments enable the students to physically explore a phenomenon prior to seeing the underlying theory and equations. These experiments serve to enhance deep understanding of the material, independent curiosity, communication, and enthusiasm.

The integration of the laptops into the curriculum was a much more delicate procedure than the desktop experiments [88]. There were three major considerations in performing this integration: timing, flow, and activity. Timing has already been discussed - it was necessary to increase lecture time to allow for the use of the laptops. It was imperative that the use of the laptops fit seamlessly into the flow of the class; the use of the laptops could not disrupt the organization of the course content. Finally, the use of the laptops had to be significant; they could not be used simply for the sake of bringing computing into the classroom, but instead would need to add to the educational experience. As such, the laptops were used during only three lectures during fall 2001. These three lectures all utilized Java simulations to enable student design or analysis of some physical system. The shift to longer lectures did not alter the amount of student-faculty interaction; however, it did make a change - when performing physical or electronic experiments - in the form of interaction. The entire teaching staff was involved with these experiments and were able to answer questions and explain concepts to individuals or small groups.

The use of technology in education is not a new topic; it dates back at least as far as S.L. Pressey's early papers [23]. Pressey - of Ohio State University - exhibited a test scoring and teaching device at the December 1924 meeting of the American Psychological Association; this device, and the other "automated teaching" devices that would follow for the next 30 years never made it into the mainstream of higher education [94]. The successful use of technology in education is yet a young and undeveloped science. There is a lack of prominent studies addressing the "intertwining of new technology and conceptual learning"; most studies tend either to indicate that conceptual learning is not taking place, or that new technology could help conceptual understanding [89]. Additionally, most of the research that has been done in this area involves grades K-8.

The primary failure in implementing educational technology is an integration process that ignores the pedagogical goals of educational reform; new technology approaches are often propelled by high-tech interests, rather than educationalist concerns [81]. This type of approach often misses the point - technology is only helpful if applied judiciously and may not be the appropriate solution in all educational situations. MIT's own President Charles Vest states the importance of maintaining the focus on learning: "We should not just use these technologies because they are there, or only because we see some potential revenues associated with them" [103].

Dr. Diana Laurillard, of the Open University in the United Kingdom, is at the forefront of research in how technology affects understanding [89]. She points out that simulations can be especially useful for conceptual understanding; however, the design and implementation of the simulations is critical to their success [45]. Specifically, the quality of understanding will be based on the level of a student's interaction with the simulation and with other students. In the active engagement pedagogy of 2.001, students are encouraged not only to interact with the systems (desktop explorations and Web modules), but they are also encouraged to work with one another; one expected result of this approach is an intuitive understanding of the concepts underlying the exercises.

Extensive research has shown that active engagement is an important aspect of student cognition [48][80]. Prof. Eric Mazur, of the Harvard University Physics Department, has had great success in implementing a student engagement procedure, whereby students must respond to questions posed by the lecturer [51]. The MIT Aero/Astro PRS approach, discussed in Chapter 2, follows the Mazur model. Students who are actively involved with the course will develop a better understanding of the material. This is why active engagement is such a major thrust of the new pedagogy.

Cooperative learning is equally established as a method for improving student understanding. Cooperative learning can be regarded as a specific form of collaborative learning; researchers have developed a taxonomy that categorizes cooperative learning as collaboration among students [61]. Such cooperative learning is of great benefit because the act of reciprocal teaching benefits students of varying strengths and weaknesses [77]. Students who assume the role of "tutor" are forced to develop a thorough understanding of the concept, while the students who pose questions learn from the "tutors." Research in this area indicates that small groups (2-6) are the most effective, since little organization is required, and since there are no issues of leadership or apprehension [37]. The applications of small group strategies are many, from formal engagements, such as experiments, to informal scenarios, where the instructor asks students to work in pairs to discuss a certain problem [49]. 2.001 makes use of both these strategies, integrating both student engagement and cooperative learning into the course.

John Dewey presented the first definitive argument for experiential learning in American education in 1938, stating that “education in order to accomplish its ends both for the individual learner and for society must be based on experience” [18]. The modern model of experiential learning most widely regarded is that presented by David Kolb [41][43]. Kolb presents four basic learning styles, each characterized by one of two perception types and one of two processing types (Figure 3.1).

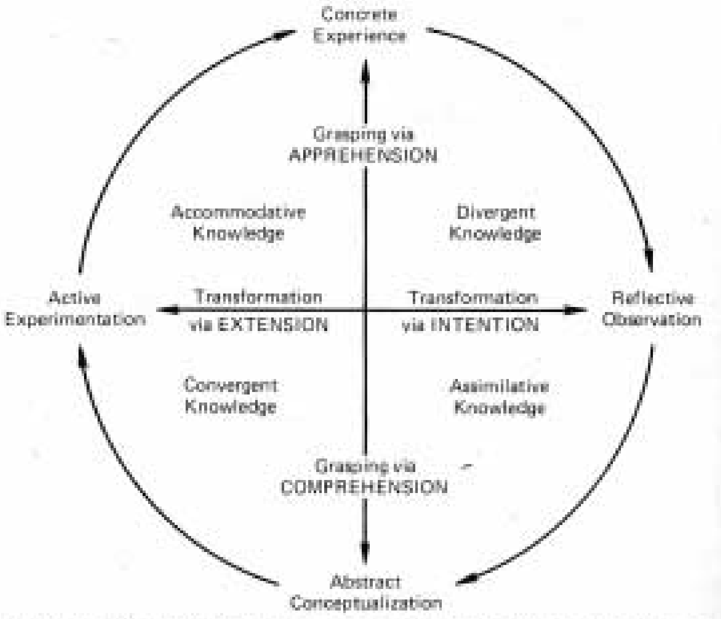


Figure 3.1 Structural Dimensions Underlying the Process of Experiential Learning and the Resulting Basic Knowledge Forms

Figure 3.1: Kolb’s learning styles: structural dimensions underlying the process of experiential learning and the resulting basic knowledge forms. Each adjacent perception/processing pair results in a different learning style, as indicated by the quadrants above.

Brief descriptions of the four learning types identified by Kolb follow [42]:

- Diverging. The dominant learning styles of this group are Concrete Experience and Reflective Observation. They are best at viewing concrete situations from many different points of view. These people tend to specialize in the arts. In formal learning situations, they prefer group work, listen with an open mind, and prefer personal feedback.
- Assimilating. Abstract Conceptualization and Reflective Observation are the dominant abilities of people belonging to this group. Members of this group tend to be best at understanding a wide range of information and putting it into a concise form. People with this style tend to place greater importance on the logical consistency of a theory than its practical applications. These people like to have time to think things through in formal learning scenarios.
- Converging. Abstract Conceptualization and Active Experimentation characterize the strengths of this style. These people are best at finding practical uses for ideas and theories. In formal learning situations, members of this group prefer to experiment with new ideas and practical applications.
- Accommodating. The dominant learning styles of this group are Concrete Experience and Active Experimentation. People with this learning style tend to learn from hands-on experience. They enjoy involving themselves in new and challenging experiences. In formal learning situations, they prefer group work, field work, and different approaches to completing a project.

In *Experiential Learning*, Kolb points out a correlation between undergraduate major and learning style; students of particular learning styles tend towards specific careers [41]. Specifically, Kolb conducted a study at a “well-known technological university”, that he generically refers to as “TECH”, in developing this theory (Figure 3.2). Interestingly, TECH is actually MIT; Kolb conducted this study on graduating MIT seniors in 1973 while at the Sloan School of Management [44]. In his experiments, he focused on the Department of Mechanical Engineering and found that most students belonged to the accommodative style of learning (Figure 3.3). It was obvious though that the mechanical engineers were generally very close to the border of the accommodator and converger quadrants. Kolb attributed this to the “general abstract bias of the university as a whole”, a description which many still consider to be accurate today.

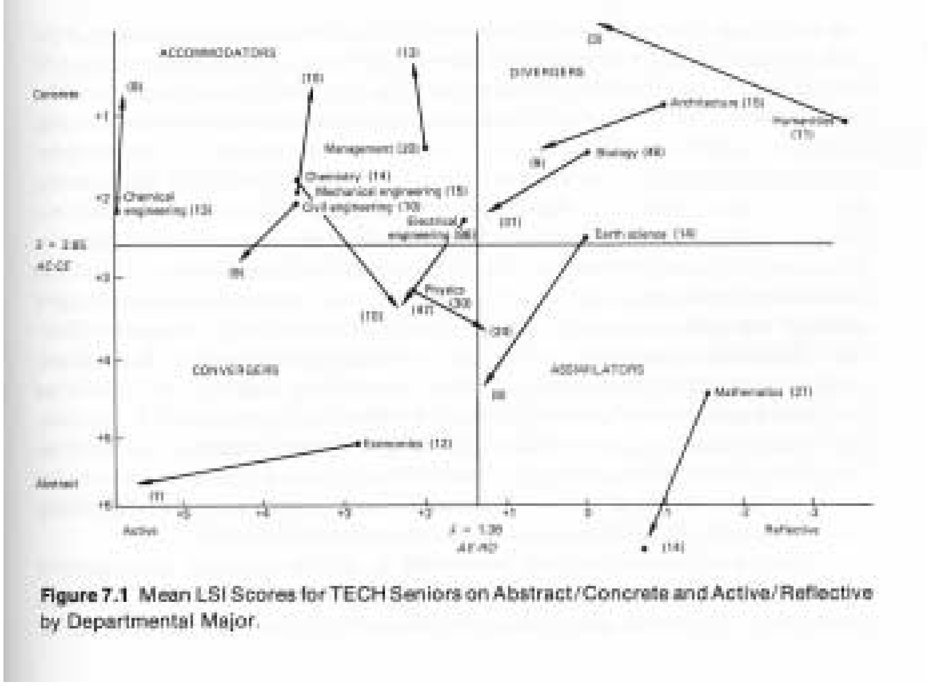


Figure 3.2: The learning style distribution of 342 graduating MIT seniors in 1973, by departmental major, based on the Learning Style Inventory [41].

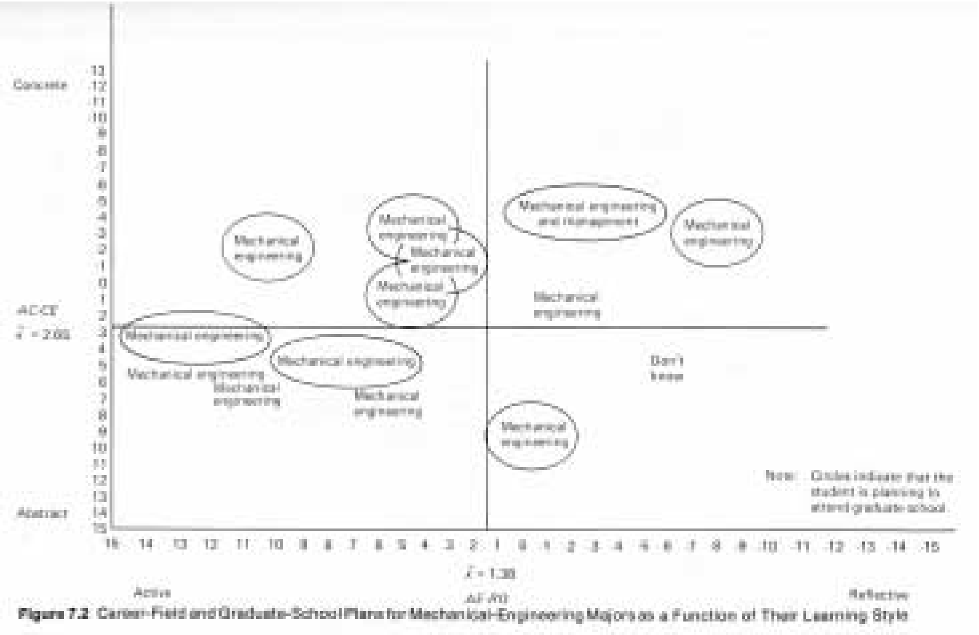


Figure 3.3: Career and graduate-school plans of graduating Mechanical Engineering MIT seniors in 1973, as a function of learning style [41].

How does this information about the learning styles of mechanical engineers at MIT relate to the active engagement pedagogy? Accommodative and convergent styles of learning both rely on active experimentation as a procedure of informational transformation. The discovery-based learning aspect of the active engagement pedagogy caters to these students by providing active experimentation as one means for learning. To imply, however, that all mechanical engineers will either be accommodators or convergers belies the actual situation. In fact, there are students who belong to each of the learning styles (Figure 3.3). The cooperative learning aspect of the new pedagogy attempts to provide an alternate resource for these students. While physical experimentation may not be the strength of these students, physical intuition is an important aspect of good mechanical engineers, so group learning allows students of different learning styles to benefit from the strengths of their peers.

Another important topic to address is the nature of the students that attend college in the present-day. Media and communication have changed the expectations that today's students have of higher education. Today's students, whether they be classified as Generation X, Generation Y, the Games Generation, or otherwise, represent a population that grew up watching MTV, playing video games, and instant-messaging [79][85]. This group, more technology-savvy than any other segment of the population, feels a need to be engaged and entertained. Having been bombarded with information their whole lives, these "media connoisseurs" are less patient than most; if something does not keep them interested, they simply move on to something else [82]. Not only do they expect to be entertained, many expect college to accommodate individual learning styles [85]. It is yet unclear whether this technological experience will produce students who learn differently from their predecessors [57]. Though the details of this generation's cognition can certainly be debated, there is little doubt that the expectations of college students have changed dramatically in recent years. This obviously necessitates a change in the way we teach and indicates that the time is ripe for educational change.

This active engagement pedagogy is different from prior work in several key respects. Most prior work in mechanics education has focused either on physical experiments or on the use of computers in the curriculum, often in the form of some electronic tutor. The 2.001 active engagement pedagogy seeks to integrate both physical experimentation and computer simulation into the curriculum; the realization is that computing is not a solution in and of itself, but rather is a tool that should be used appropriately. Laptops have not been used expressly for the purpose of reforming mechanics education. The 2.001 project not only uses laptops, but fully leverages the capabilities of mobile, wireless computing. Finally, the new pedagogical development is a holistic approach; rather than attempting to create new experiments or courseware for the class, this project is redesigning all aspects of the class in order to produce components that complement one another.

3.2 Interactive Media: Web Modules

A major component of the new teaching methodology is a set of interactive, Web-based learning modules [32].[11] These modules, which are described in more detail in Chapters 5 and 6, are intended to be accessed by the students both in-class and at-home. They provide a means for independent exploration, as well as team-based experimentation and design. The Web modules utilize Java applets, Active Server Pages, and SQL Server to present dynamic and interactive content that cover the spectrum of topics in the course. Each Web module is specific to a certain topic in Solid Mechanics, such as Beam Bending or Truss

[11] http://icampus1.mit.edu

Structures.

These modules are intended as an integral part of 2.001; they are not intended as a replacement for any portion of the course. The modules provide capabilities, such as self-testing and computer-based analysis tools that no other portion of the course can provide. This is a critical part of the new teaching paradigm, that we maintain physical experiments to develop a “feel” for mechanical processes, and that we do not take away from student-faculty interaction.

3.3 Laptop Initiative

Another critical component of the new pedagogy was ready access to the online materials developed for this course, both inside and outside the classroom. To enable this access, it was necessary to provide the students with laptop computers. Many other colleges and universities are experimenting - or have experimented - with mobile and wireless computing in undergraduate education; examples include the University of Kentucky, Cornell University, and the University of Akron [27][15][101].

MIT Information Systems (IS) itself embarked on an educational technology experiment in fall 2001 [66]. In this experiment, called the Student Laptop Project, four academic departments at MIT used laptop computers in selected courses. In three of these departments, students were issued a laptop for use during the term: Architecture StudioMIT, course 1.00 (*Introduction to Computers and Engineering Problem Solving*, in Civil Engineering), and 2.001. In the Physics TEAL project, the laptop is a part of the laboratory equipment and is used for the class 8.02T (*Electricity and Magnetism*).

This experiment had two major outcomes: to determine the pedagogical and learning benefits of wireless, mobile computing in education and to determine the support requirements and resources needed to sustain such an environment. These four educational projects were selected expressly because they proposed to use laptops in a different manner. As such, IS could best evaluate the educational benefits of the laptops.

The equipment for this experiment was acquired through a grant from the Hewlett-Packard Corporation, and was supplemented by additional resources from MIT’s Council on Educational Technology.

To understand the motivation for this project, it is necessary to chronicle the recent history of computing at MIT. In May 1983, MIT established a five-year program to explore new uses of computing in the MIT curriculum [5]. At the time, the MIT faculty was concerned that the administration was not being proactive in integrating new computational technology into the undergraduate curriculum. This program, called Project Athena, was born from this concern. The goal of Project Athena was to investigate diverse uses of computing and to develop a long-term plan for the integration of computing into the curriculum. Project Athena and the Student Laptop Project, though they dealt with different computing technology, fundamentally had very similar objectives in their respective charters. In 1988, Project Athena was granted a three-year extension to the original five-year program, and, on June 30, 1991, Project Athena finally came to an end. In 1991, the Athena system was taken over by IS and was adopted as MIT's official academic computing infrastructure; it was extended beyond the educational domain, into the research and administrative activities of the Institute.

Athena is a system of hundreds of end-user workstations distributed around campus that provide round-the-clock capability to do work and communicate with others. The system is comprised of a large number of networked machines including workstations, printers, and servers. Each workstation is connected to MITnet, the campus-wide computer network, allowing access to central resources.

Athena has been a very successful undertaking. In addition to representing the first large-scale application of distributed computing into education, Project Athena developed the X Windows System, commonly used on UNIX systems today, provided email and newsgroup access to students prior to the popularization of the Internet, and spawned Zephyr, the first instant messaging service [109]. In 1999, IS reported that 96% of undergraduates, and 94% of graduate students, had Athena accounts [60]. They additionally stated that, on a typical day, 6000 Athena users accessed their accounts. Athena is still an integral part of the undergraduate experience at MIT; however, as the overwhelming majority of students now own personal computers that they can readily connect to the MITnet, Athena is not as essential as it was in the past.

The Student Laptop Project also provided students access to the MITnet, though students were also able to access the network wirelessly. In July 2001, IS concluded the installation of 209 wireless access points, covering over 200 locations in 39 buildings [108]. Wireless access to MITnet is currently available in most classroom and library spaces as well as many public spaces across campus. Additional access points are being deployed in some areas currently undergoing renovation. The wireless access was important to the 2.001 project as it enabled students to easily connect to the network in the classroom and across campus.

The many obvious similarities between Project Athena and the Student Laptop Project suggest that the impact of the latter is one which - if successful - could likely have resounding effects through the educational community. It is clear that Project Athena made much greater leaps than can be expected of the Student Laptop Project; however, the laptop project may yield a fundamentally new paradigm for educational computing, to be adopted at other institutions.

3.4 Desktop Experiments

Farid Ganji, Pierce Hayward, Prof. Mary C. Boyce, Prof. L. Mahadevan, and Prof. Emanuel Sachs have all been involved in the creation of physical desktop experiments for 2.001 over the past two years. There are currently nine desktop experiments: Free Body Diagram, Static Friction, Capstan Effect, Effective Spring Constants, Truss Structures, Biaxial Loading, Beam Strain, Beam Deflections, and Sandwich Beams (Appendix A) [87].

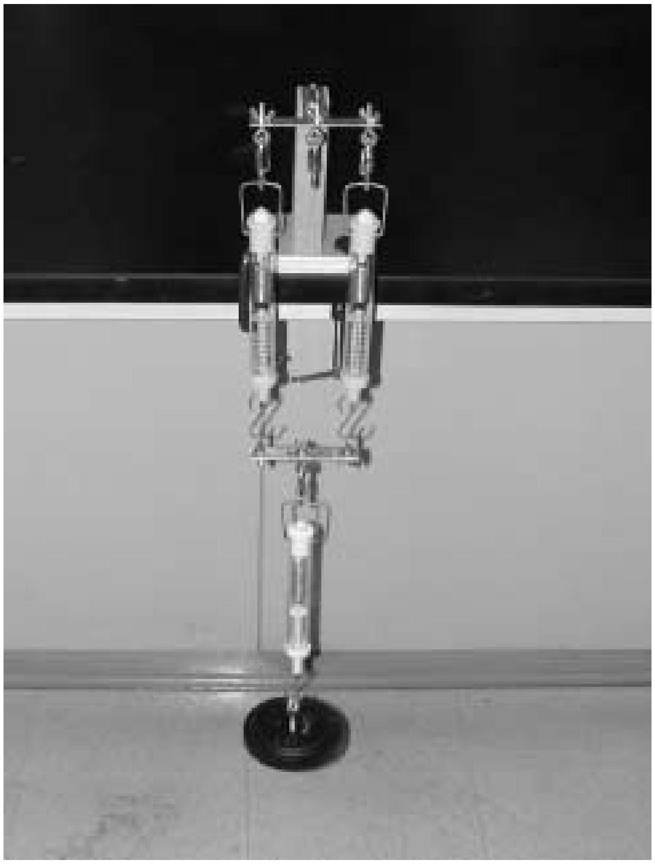


Figure 3.4: Effective Spring Constants desktop experiment.

These desktop experiments each correlate to a specific Web module or to a topic within a Web module. For the Effective Spring Constants experiment, students use spring scales and weights to determine system deflections and compare those results with their experimental results (Figure 3.4). For the Beam Deflections experiment, students measure beam deflections under different loading conditions to determine the relation among beam thickness, beam length, and beam stiffness (Figure 3.5).

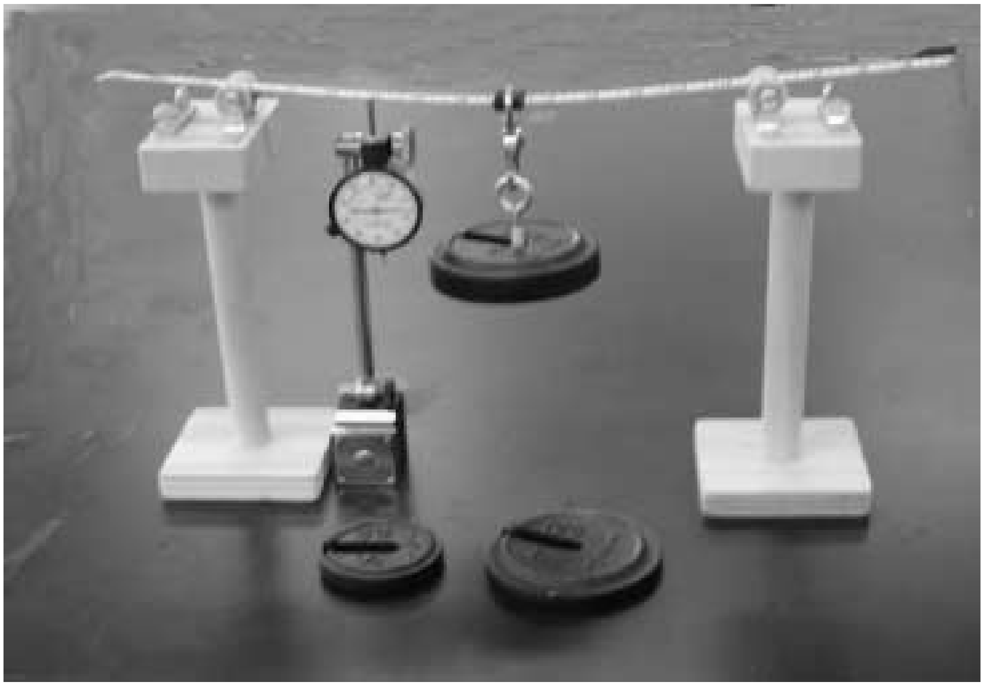


Figure 3.5: Beam Deflections desktop experiment.

Some of the experiments have been used during the last three academic terms, and - based on student and instructor feedback - the desktop experiments provide an excellent means for learning new concepts. Students have remained interested and engaged. By working in teams they have not only learned to work together, but they have also learned to use each other as a part of the learning process (Figure 3.6).

Figure 3.6: Student team working together on Capstan desktop experiment (spring 2001).

These experiments have been designed to allow students to participate in the process of Scientific Discovery. Students observe physical phenomena through experimentation.

They then analyze the mechanisms and make the connection between the real process and theory. These desktop experiments are a major component of the active engagement pedagogy, as they incorporate both cooperative and discovery-based learning.

3.5 New Classroom

A new classroom designed to align with the objectives of the new pedagogy was constructed during summer 2001. This project was jointly funded by a donation from MIT alumnus Dr. B.J. Park, his wife Chunghi Park, and the Singapore-MIT Alliance (SMA). This project was responsible for the complete renovation of two classrooms in the Mechanical Engineering buildings: 3-270 and 3-370. This course, 2.001, is currently taught in 3-370.

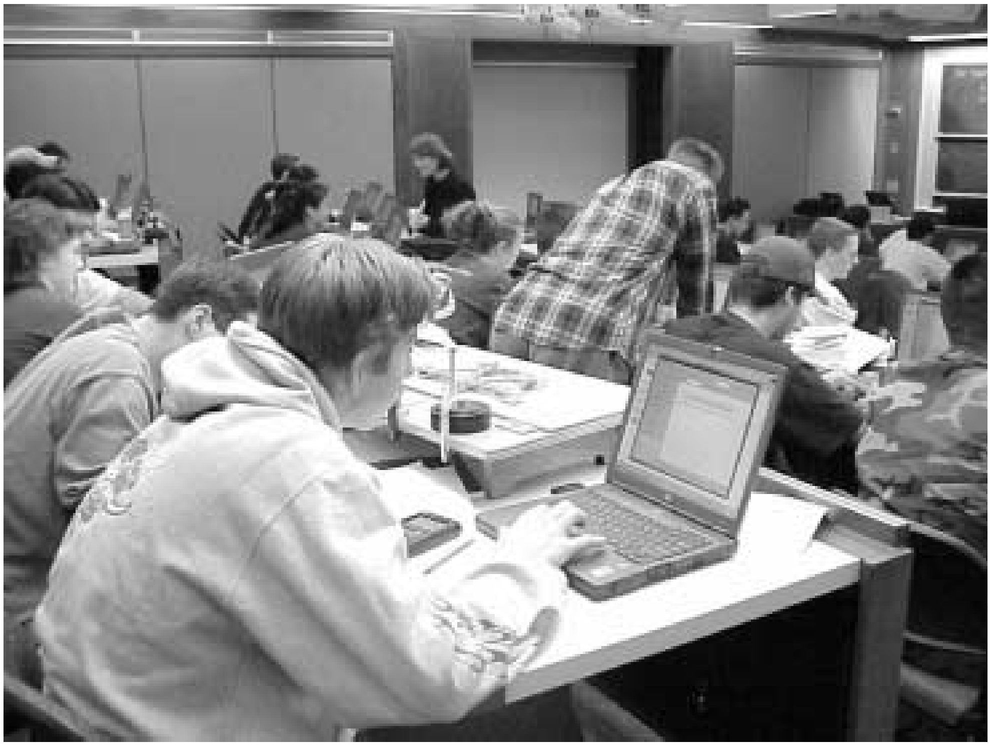


Figure 3.7: Students participating in a design exercise in the Park Room for Innovative Education (3-370).

Dubbed the Park Room for Innovative Education, 3-370 is significantly different from its previous state (Figure 3.7). The old classroom seated almost 100 students and was comprised of rows of small chairs with built-in desks [87]. It had no audio-visual capabilities, was acoustically poor, and offered uncomfortable, cramped seating (Figure 3.8). The new classroom features, in contrast, a multitude of technological capabilities, and a comfortable, well-lit environment.

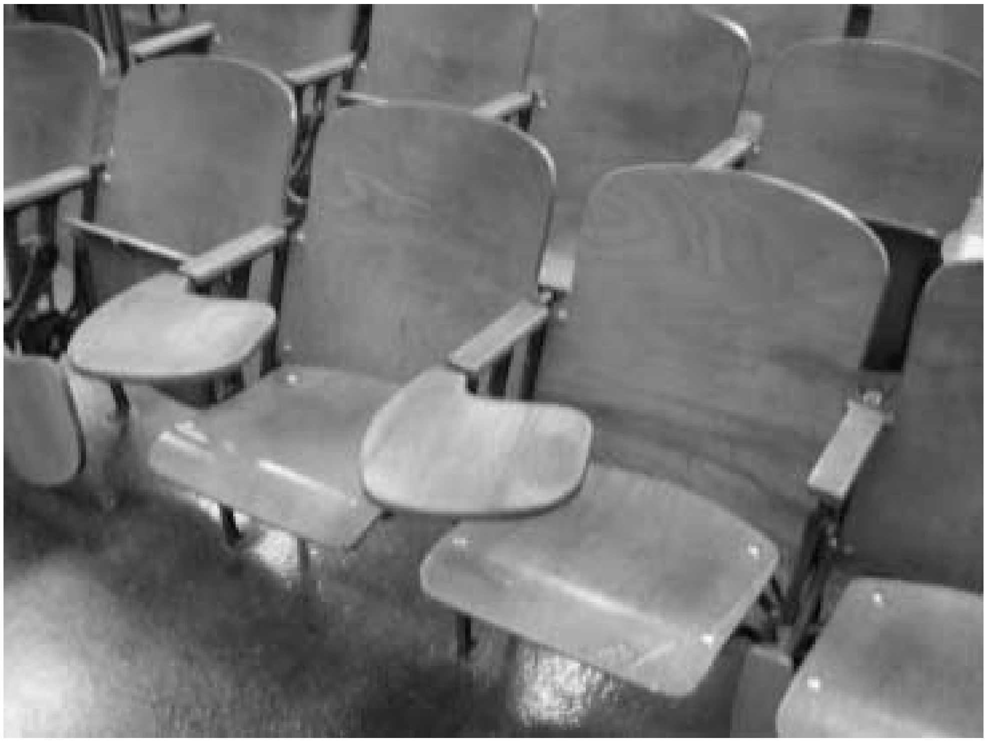


Figure 3.8: The seating style in the old classroom consisted of rows of wooden chairs with built-in desks.

The new design features modular tables which can be configured to form tiered rows (lecture-style arrangement) or individual tables that seat students in groups of four (lab-style arrangement); these tables were designed specifically for the active engagement pedagogy by Prof. Emanuel Sachs (Figure 3.9, Figure 3.10). The reduction in seating capacity (60 students) allows students to sit comfortably, to more easily engage in cooperative learning, and to be easily approached by instructors. The utility of this flexible design manifests itself during classes that are used for both lecturing and desktop experiments. Multiple network drops and power outlets are included at each table. Students can access the MITnet using ethernet or, alternatively, can access the MITnet Wireless Network via the wireless access point installed in the rear of the classroom.

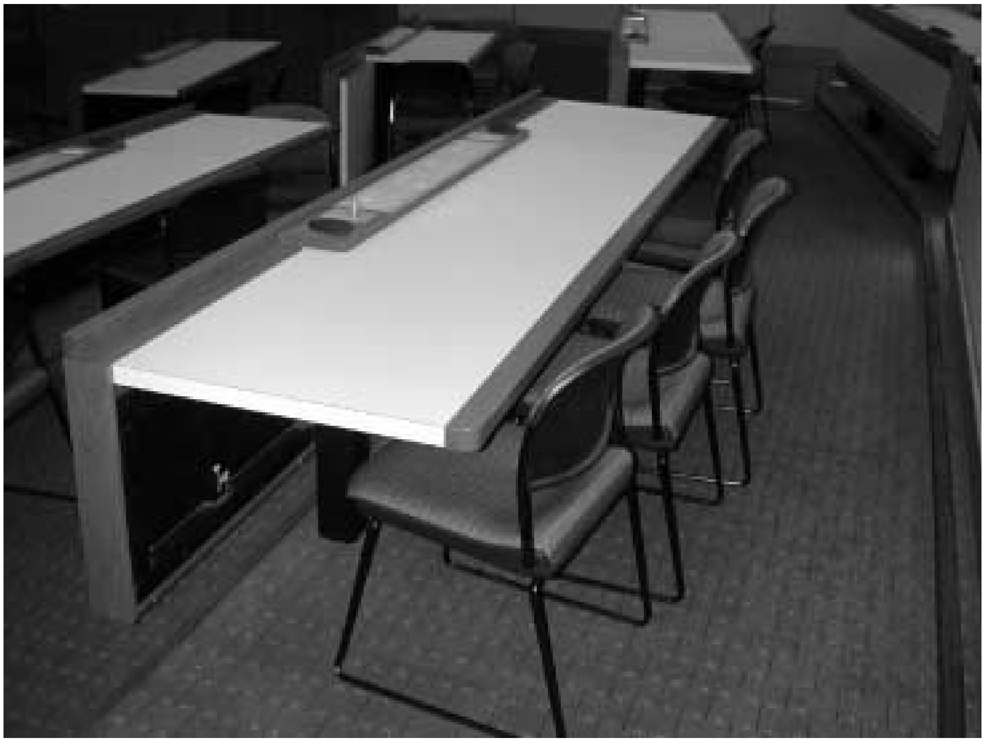


Figure 3.9: Modular desks in 3-370. Shown in lecture-style configuration.

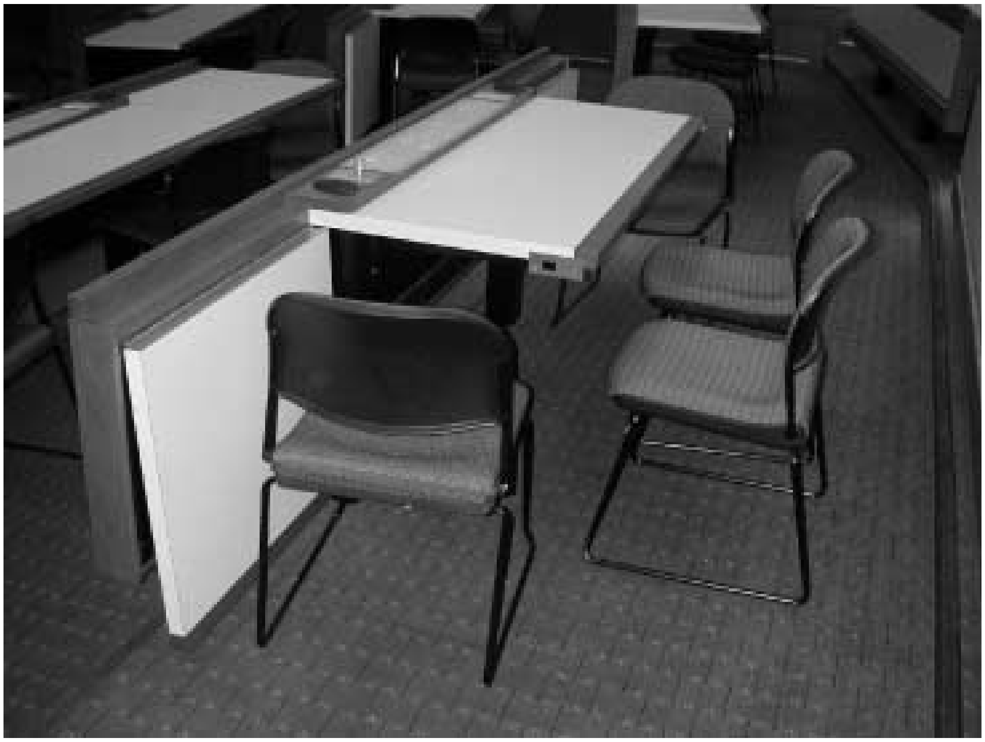


Figure 3.10: Modular desks in 3-370. Shown in lab-style configuration.

A major technological feature of the classroom is a projection system that quickly toggles between a video-based projector, DVD/VHS input, a computer console, and external device (such as a personal laptop). The seamless integration and ease-of-use of these technologies has brought an end to the painful procedures of acquiring equipment for a specific class and subsequently struggling with the technology.

Chapter 4

Equipment

4.1 Overview

This section is a partial deviation from the primary focus of the thesis, but is sufficiently important to the reader to be included as a section of the body, rather than as an appendix. Included herein, are details of the equipment used to develop, deliver, and maintain the Web modules. Specifications are given for: the server; the software used to create, run, and maintain the content; and the system requirements of the end-user.

4.2 Server-side Requirements

The site was hosted on MicronPC NetFRAME NF3400 Server. Relevant technical details are given below:

- Dual 600MHz Intel Pentium III processors
- 512 MB 100MHz SDRAM-2 DIMMs
- 90 GB disk space (five 18 GB hard drives)
- 20/40 GB SCSI DDS 4 tape backup unit

The server ran Microsoft Internet Information Services (IIS) on a Windows 2000 platform. IIS supported Active Server Pages (ASP), which enabled display of dynamic and database-driven content. Microsoft SQL Server 2000 was utilized as the database backend; it replaced Microsoft Access, which was found during testing to be insufficient given multiple user loading. Development tools varied, but several were selected to enable multi-user modification of files. Navigation images and logos were created using Macromedia Fireworks. Microsoft Visio was used for most other image creation, though Pro/ Engineer and Adobe Photoshop were used in applications where Visio was insufficient. Adobe FrameMaker and WebWorks Publisher were used for creation of extensive textbook-style content, such as analytical derivations and case studies. Macromedia Dreamweaver was used for HTML and ASP creation and modification; and for exchanging files with the server. Different development environments were used for Java programming, since these all supported the same file types.

These requirements are not all essential to a project such as this one, they are only included as an inventory of the tools used. The minimum requirements are: a server configured to generate dynamic content; a database; and an agreement on a set of programs for site development.

4.3 User Requirements

Specific requirements were set for users to be able to access all course material available on the website. In terms of hardware, a display resolution of 1024x768 or higher was required. While many Web designers target 800x600 as the low-end of the resolution scale, it was impractical (particularly with respect to simulations) to fit all necessary material in that space. This was not a significant concern, however, since the students were all provided with laptops at the higher resolution as a part of the portable computing initiative. Internet access, through either MITnet or an Internet Service Provider (ISP), was also required, though the possibility of distributing the material on a CD-ROM has also been discussed. At present time, since so much of the site is database-dependent, distributing the site as courseware on a CD-ROM is not a short-term reality. A sound card, and speaker or headset was another requirement that would allow students to engage in the audio portion of the online lectures. As will also be noted later, a text-based narration was also provided as an alternative to students. Finally, though much of the site was navigable using keyboard shortcuts, some portions, such as the simulations, necessitated mouse input.

In terms of software, there were only 2 requirements, a Web browser and the Java Run-time Environment (JRE). Because of the site's reliance on Cascading Style Sheets (CSS) for formatting, Microsoft Internet Explorer or Netscape Communicator (both version 4.0 or higher) were recommended. In order to run the simulations, a recent version of the JRE was also required (in the form of the Java 1.3.0 plug-in). All applets on the site checked for the existence of the software and downloaded newer files as necessary.

The laptops donated by Hewlett-Packard, and used by all students in 2.001 during fall 2001, met all of these specifications. The model (HP Omnibook 6000) specifications are given below [47]:

- 256 MB RAM
- 10 GB disk space
- Lucent Wireless LAN cards (802.11b)
- 650 MHz Pentium III
- CD-RW drive
- Software: Netscape, Internet Explorer, MIT Utilities and TSM, Pro/Engineer

It should also be noted that students were able to access the MITnet Wireless Network from most non-residential sectors of campus. Additionally, built-in Ethernet cards and pre-configured DHCP (Dynamic Host Configuration Protocol) settings allowed access from any MIT Ethernet port on campus, including residential rooms.

Chapter 5

Systematic Design

5.1 Overview

The design, development, and synthesis of the Web modules and the new pedagogy has been a systematic process. Beginning in fall 2000, the intention was to stage the different portions of development. Discretizing the design process has long been recognized as important in both software development and product design [19][71]. There was no reason that the development of these Web modules or the redesign of the course pedagogy should tend away from such a process. Though not commonly discussed, the systematic design of a course is important in designing a successful course, and being able to assess it effectively [4]. An iterative process is the key to successful curriculum design because "for each cycle to benefit from its predecessor, there must be a constructive link forward and into the next development" [24]. The following phases were planned and executed: 1. conceptual development; 2. initial development of the Web modules; 3. limited deployment of the modules; and 4. large-scale implementation. The final step, full integration, will be performed over the course of the next two semesters. Assessment, both formative and summative, is an ongoing process that runs in parallel to many of these development stages.

5.2 Conceptual Development

At the onset of the project it was necessary to define the goals of the reform initiative. Four main goals were identified. First, students should have a better conceptual understanding of the course material. Secondly, they should begin to understand the dynamics of teamwork, and should become comfortable speaking in the language of engineers. Thirdly, students should have greater retention, implying a deeper understanding of the course material. Finally, students should be able to engage in independent learning. While team projects and team learning are definitely important, students should also be self-sufficient. While implementing these goals, it was also important to maintain, or to improve, student satisfaction with the course and with the academic department.

Upon defining the goals of the project, it was imperative to determine what actions would be needed to attain these goals. Web modules covering the main topics of the course, and designed to enhance conceptual understanding, would help students in developing a solid understanding of mechanics. Physical, desktop experiments would allow the students to learn about teamwork. These experiments would also serve to shape and strengthen their physical understanding of the material. Redesign of the 2.001 classroom would allow the students to more easily interact with each other. Supplying the students with laptops would enable them to interact with one another and to use the Web modules. Further, by integrating technology into the curriculum, and by adding "life" to the course material, through experiments and simulations, it was hoped that student satisfaction would improve.

Periodically reviewing these objectives and the plans for meeting these objectives during the development process has kept the project on track.

5.3 Initial Development

The initial plan was to have five Web modules. These Web modules would each consist of five module components: an interactive simulation tool, analytical derivations, real world examples, a dynamic quiz, and a reference section. By creating a basic infrastructure for each module, development time of subsequent modules would be minimized.

Upon the initial creation of the Bending module, faculty and other peers offered feedback. Additionally, MIT Information Systems (IS) assessed our Web modules. They presented very useful feedback, especially in terms of site usability. Poor user interfaces can actually be a hindrance to the learning process, so this was very important [46]. Later, in January 2001, IS helped organize focus groups to provide formative assessment of the Web modules (Appendix B). These focus groups, comprised of former 2.001 students to identify issues with our site. Elements of site content, appearance, and ease-of-use were identified as needing improvement [102].

This feedback was all used to refine the Bending Web module. First, site navigation was revamped, making it more intuitive, more graphically appealing, and more powerful. The user interface of the beam bending simulation was also modified, making it more intuitive. Based on results of the focus groups, the need for students to have a focused task became apparent; this led to the development of a design problem that would be used with the simulation. From discussions with Prof. Tomas Lozano-Perez of the MIT Department of Electrical Engineering and Computer Science (EECS), a sixth component - a system of online lectures - was added to the modules, following the model of 6.001 (*Structure and Interpretation of Computer Programs*) [1]. The quiz was also modified to include an in-class quiz, using the Mazur model [51].

Implementation of these changes occurred before the spring 2001 class reviewed beam bending. The next step was to use this module during class to determine its effectiveness and to determine if there existed any unforeseen issues in using the Web modules in class.

5.4 Limited Deployment

During spring 2001, the Bending module was used during one lecture and during the three recitation sections. For this, it was necessary to conduct the classes in a classroom equipped with computers and large enough to handle 50 students (this was prior to the renovation of room 3-370). During all classes, students worked in pairs, out of a desire to encourage cooperative learning. The plan for the lecture was to introduce students to the website, allow them to explore the site, and use the site to review topics in beam bending. During recitation, students were expected to – after a short presentation of the problem by the faculty – work in teams, using the simulation to solve the design problem, the design of a diving board.

This experience yielded several key lessons. First, it is absolutely imperative with a tool like this to have a clearly defined plan of what needs to be done. Without a task, many of the students wandered to other sites or simply lost interest; however, while doing the design problem, the students all –without exception – were deeply engaged in designing the diving board. Technical difficulties with the server and with the server software also became apparent. The solutions (obtaining additional site licenses and upgrading from Access to SQL Server) were easily obtained, but the problems would have remained unidentified without this deployment. Overall the student – and faculty – feedback was very positive. The students were excited about the results of the in-class quiz being displayed at the front of the classroom, because they could begin to understand not only the common misconceptions that they had, but also those of their peers. The anonymity of the quiz also made it easier to participate in class. Students responded very positively to both the simulation and design problem.

5.5 Large-scale Integration

All issues identified during the in-class use of the module were resolved. By the beginning of fall 2001, ten Web modules were planned, each with seven components. The addition of a case study, the seventh module component, would provide students the opportunity to learn about mechanical design. The new classroom was used, all students were furnished with laptops, and class time was reorganized per the new teaching paradigm. Three modules were deployed in their entirety: Equilibrium, Truss Structures, and Beam Bending. Additionally, simulations from the Multiaxial Stress-Strain module were used. By deploying a subset of the modules, their effectiveness could fully be assessed. Additionally, the proposed shift from lecture-recitation to the more integrated format could be assessed. The specific details of the assessment and the results can be found in Chapter 8.

Chapter 6

Web Modules Overview

6.1 Site Organization

From the beginning of the project it was imperative that there be a logical structure to the Web-based learning tools. This led to the discretization of the course material into distinct topics; each topic would correspond to a specific Web module.

The reasons for this approach were twofold. First, this would provide a logical means for the student to navigate the site. Second, it would provides a systematic means for developing the course; the Web-based portions of the course could be developed and assessed more easily in this manner. Moreover, the same structure for all the modules would streamline the process of creating Web content. After the initial module (Bending) was completed, there was a very direct sense of what work needed to be done in order to create more modules. The infrastructure - in terms of module components, scripts, content development procedures - were all already in place. The time spent in creating subsequent modules was much less than in creating the initial one.

The only danger in segmenting the course into discrete topics was the possibility of certain topics "slipping through the cracks." This was especially of concern given the inter-related nature of the course material in Solid Mechanics. This was acknowledged as an open issue and a conscious effort was made to avoid this.

6.2 Modules

This course will consist of ten Web-based modules in total: Equilibrium, Friction, Axial Force-Deformation, Truss Structures, Multiaxial Stress-Strain, Linear Thermo-Elasticity, Bending, Buckling, Torsion, Energy Methods, and Materials Selection. Four of these modules – Equilibrium, Friction, Truss Structures, and Bending – have been completed and three were used in teaching the course in fall 2001. A brief description of each of these sections follows:

- Equilibrium. This section introduces the basic concepts of mechanics and the concept of static equilibrium. Forces and moments are discussed and vectors are reviewed. Internal forces are also covered.
- Friction. Concepts of static and dynamic friction are reviewed with applications to mechanical systems. Static equilibrium remains a focus of this section. The Capstan effect is also introduced.
- Axial Force-Deformation. This is the first introduction to deformable bodies. Specific discussion includes the concepts of stress, strain, and elastic material behavior.
- Truss Structures. This section covers the analysis of idealized trusses, and focuses on planar structures. Truss design, internal forces, system displacements, matrix analysis, and the conceptual-level behavior of trusses are discussed.
- Multiaxial Stress-Strain. This section covers multiaxial stress and multiaxial strain. Transformation equations, Mohr's circle, and pressurized cylinders are key topics.
- Linear Thermo-Elasticity. This section presents stress-strain-temperature relations in the elastic regime. In particular, it covers problems where the student is required to postulate boundary conditions, such as zero-strains or zero-stresses.
- Bending. Strain and stress distributions in beams, the moment-curvature relationship, beam deflections, the second moment of inertia, the singularity function, and the design of beam cross-sections are all discussed. The concept of superposition is also introduced.
- Buckling. Students are introduced to the concept of columnar buckling. Buckling analysis is used in the design of axial members subjected to compressive loads.
- Torsion. Stress, strain, and deformation in twisted shafts are covered. The advantages of hollow shafts over solid shafts are also discussed.
- Energy Methods. Energy methods (Castigliano's theorem) are applied to a variety of problems. Specific emphasis is on the simplification of complex problems - such as trusses - through the use of energy methods.
- Materials Selection. This section builds on the fundamentals of this class and explores the process of materials selection in mechanical design. This topic provides an appropriate transition to the follow-up course, 2.002 (*Mechanics and Materials II*).

6.3 Module Components

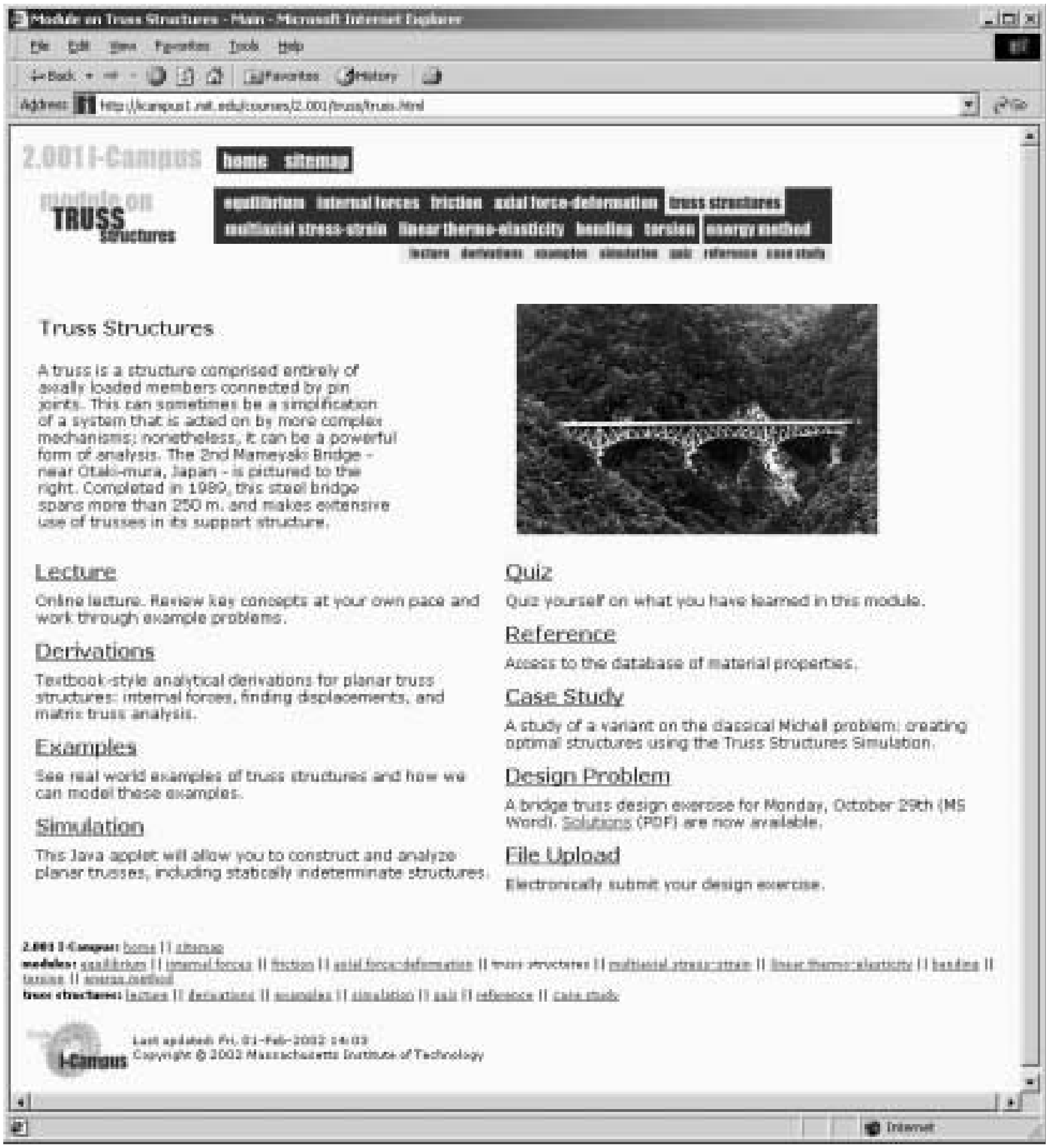
Each module is made up of the same eight components: a simulation, a set of online lectures, textbook style derivations and explanations, real world examples, dynamic quizzes, a design exercise, a reference section, and a case-study. These components are all described in Chapter 7.

Chapter 7

Web Module Components

7.1 Overview

As stated, this course will consist of ten Web-based modules in total: Equilibrium, Friction, Axial Force-Deformation, Truss Structures, Multiaxial Stress-Strain, Linear Thermo-Elasticity, Bending, Buckling, Torsion, Energy Methods, and Materials Selection. Each module is made up of the same eight components: a simulation, a set of online lectures, textbook style derivations and explanations, real world examples, dynamic quizzes, a design exercise, a reference section, and a case-study. The Truss Structures module will be used as a thread for describing the module components (Figure 7.1). Other modules will be mentioned as appropriate. The following sections describe the components of each of the modules in detail [86][87]. The instructor interface to the course website will also be discussed.


Module on Truss Structures - Main - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Favorites History
Address http://icampus1.mit.edu/courses/2.001/truss/truss.html
2.001 I-Campus
home sitemap
module on TRUSS Structures
equilibrium internal forces friction axial force-deformation truss structures
multiaxial stress-strain linear thermo-elasticity bending torsion energy method
lecture derivations examples simulation quiz reference case study
Internet

Truss Structures

A truss is a structure comprised entirely of axially loaded members connected by pin joints. This can sometimes be a simplification of a system that is acted on by more complex mechanisms; nonetheless, it can be a powerful form of analysis. The 2nd Mameyaki Bridge - near Otaki-mura, Japan - is pictured to the right. Completed in 1989, this steel bridge spans more than 250 m. and makes extensive use of trusses in its support structure.

Lecture

Online lecture. Review key concepts at your own pace and work through example problems.

Derivations

Textbook-style analytical derivations for planar truss structures: internal forces, finding displacements, and matrix truss analysis.

Examples

See real world examples of truss structures and how we can model these examples.

Simulation

This Java applet will allow you to construct and analyze planar trusses, including statically indeterminate structures.

Quiz

Quiz yourself on what you have learned in this module.

Reference

Access to the database of material properties.

Case Study

A study of a variant on the classical Michell problem: creating optimal structures using the Truss Structures Simulation.

Design Problem

A bridge truss design exercise for Monday, October 29th (MS Word). Solutions (PDF) are now available.

File Upload

Electronically submit your design exercise.

2.001 I-Campus: home || sitemap
modules: equilibrium || internal forces || friction || axial force-deformation || truss structures || multiaxial stress-strain || linear thermo-elasticity || bending || torsion || energy method
truss structures: lecture || derivations || examples || simulation || quiz || reference || case study

Last updated: Fri, 01-Feb-2002 14:03
Copyright © 2002 Massachusetts Institute of Technology

Figure 7.1: Truss Structures Web module main page, providing access to all module components.

7.2 Simulation

Properly designed simulations can be very useful in developing students' conceptual understanding of scientific topics [45] and this is the intent, in this case, of the Truss Structures simulation, called Torsus (Appendix C).

This investigative tool allows students to construct and analyze planar truss structures subject to various constraints (Figure 7.2). This tool is specifically intended to help student develop a conceptual understanding of large structures. Prior to the use of this simulation, students will have been asked to analyze simple trusses with 2-5 members. This simulation supplants the tedium involved in analyzing more complex structures, and allows the student to focus on a higher level understanding of the behavior of trusses.

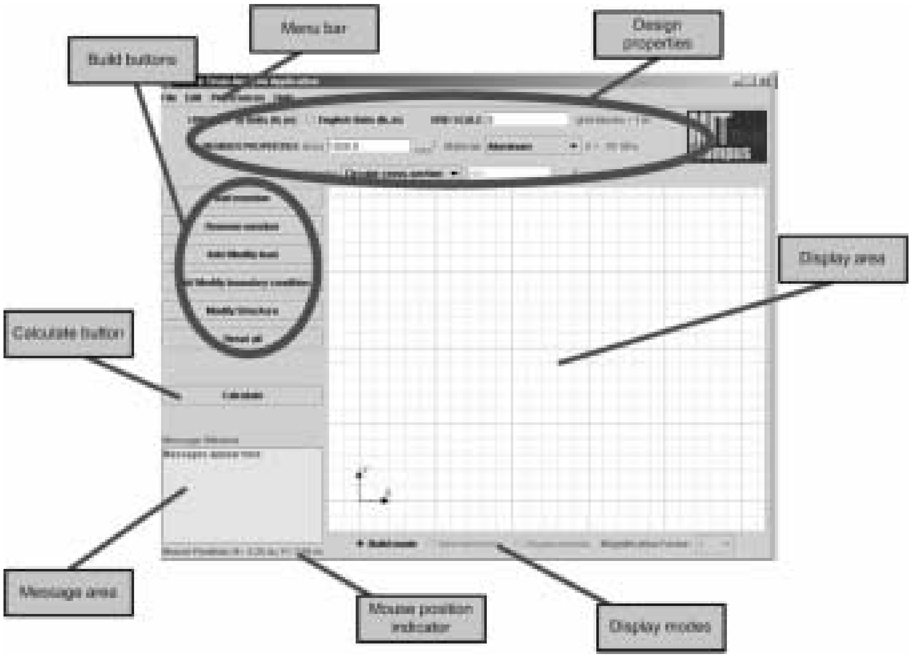
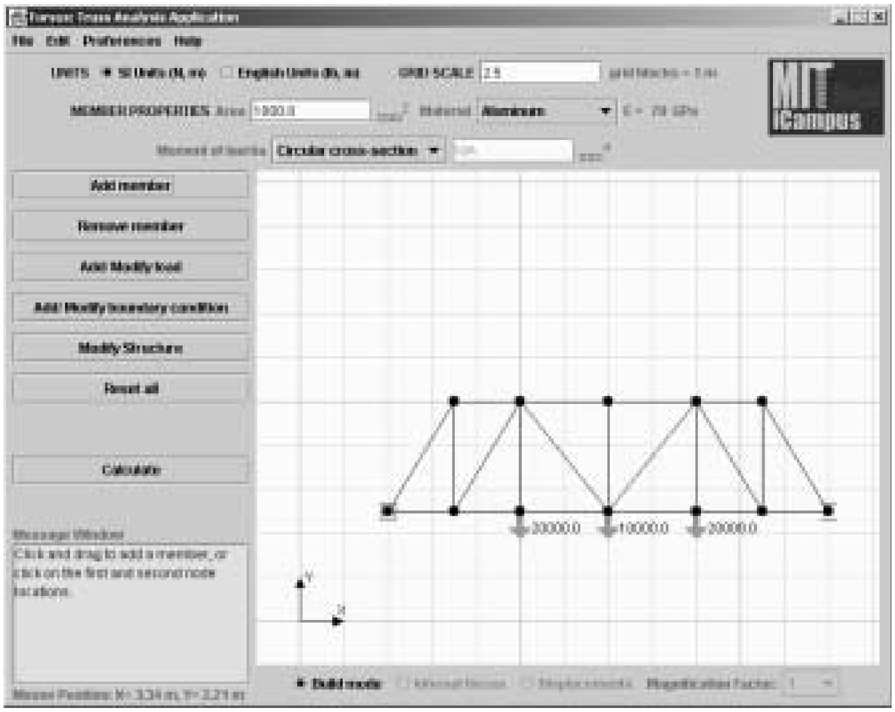


Figure 7.2: Torsus, Truss Structures simulation, program layout.

The user graphically creates a truss structure, applies loads and boundary conditions, and runs the computation engine of the program (Figure 7.3, Figure 7.4). The simulation uses matrix truss analysis to produce an analysis of the truss structure, so users can choose to analyze statically indeterminate systems. The user is then able to view graphical representations of the internal forces and system deformation (Figure 7.5, Figure 7.6). There is additionally a text-based output of relevant information, such as internal forces, nodal displacements, buckling analysis, and structural mass.



UNITS ● SI Units (N, m) □ English Units (lb, in) GRID SCALE 2.5 grid blocks = 1 m

MEMBER PROPERTIES Area 1000.0 mm² Material Aluminum E = 70 GPa

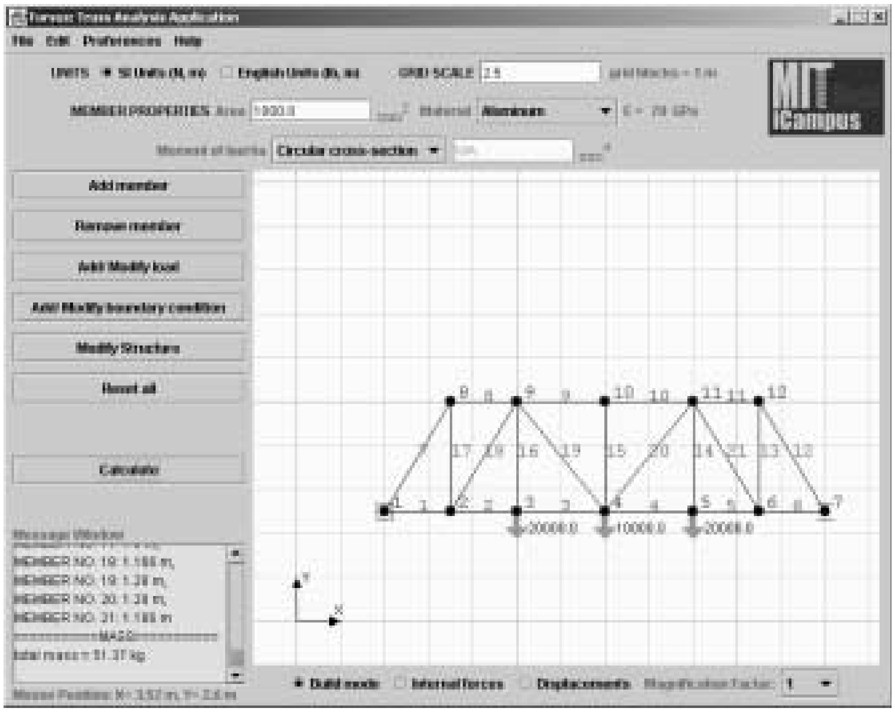
Moment of Inertia Circular cross-section mm⁴

Click and drag to add a member, or click on the first and second node locations.

Mouse Position: X= 3.34 m, Y= 2.21 m

● Build mode ○ Internal forces ○ Displacements Magnification Factor: 1

Figure 7.3: A truss construction in Torsus.



UNITS ● SI Units (N, m) □ English Units (lb, in) GRID SCALE 2.5 grid blocks = 1 m

MEMBER PROPERTIES Area 1000.0 mm^2 Material Aluminum E = 70 GPa

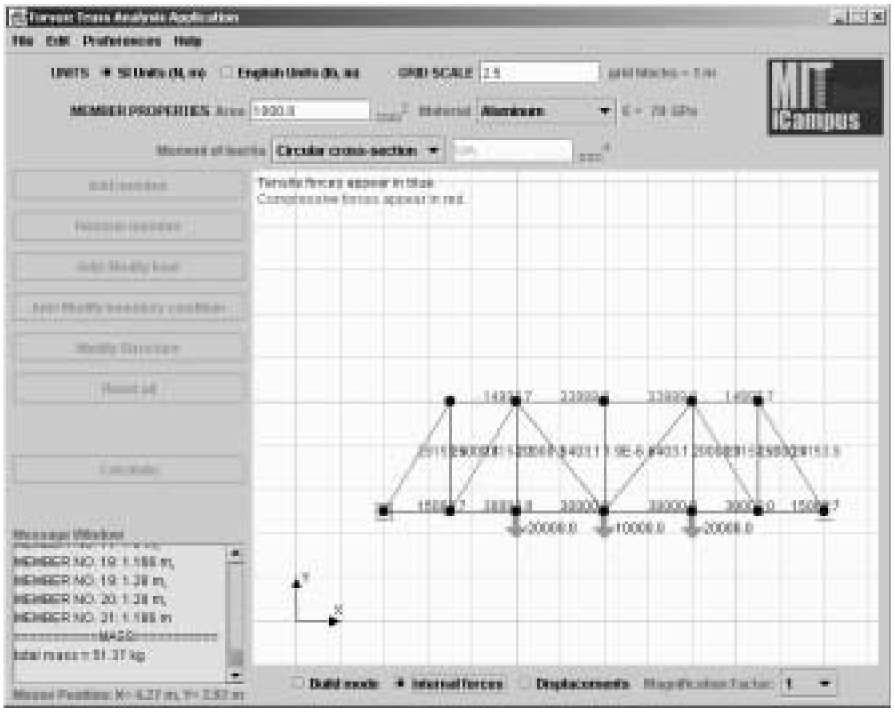
Moment of Inertia Circular cross-section

MEMBER NO. 19: 1.166 m,
MEMBER NO. 19: 1.28 m,
MEMBER NO. 20: 1.28 m,
MEMBER NO. 21: 1.166 m
-------------MASS-------------
total mass = 51.37 kg

Mouse Position: X= 3.57 m, Y= 2.6 m

Magnification Factor: 1

Figure 7.4: Upon computation, members and nodes are assigned numeric values that correspond to the text-based output in Torsus.



UNITS ● SI Units (N, m) □ English Units (lb, in) GRID SCALE 2.5 grid blocks = 1 m

MEMBER PROPERTIES Area 1000.0 mm^2 Material Aluminum E = 70 GPa

Moment of Inertia Circular cross-section

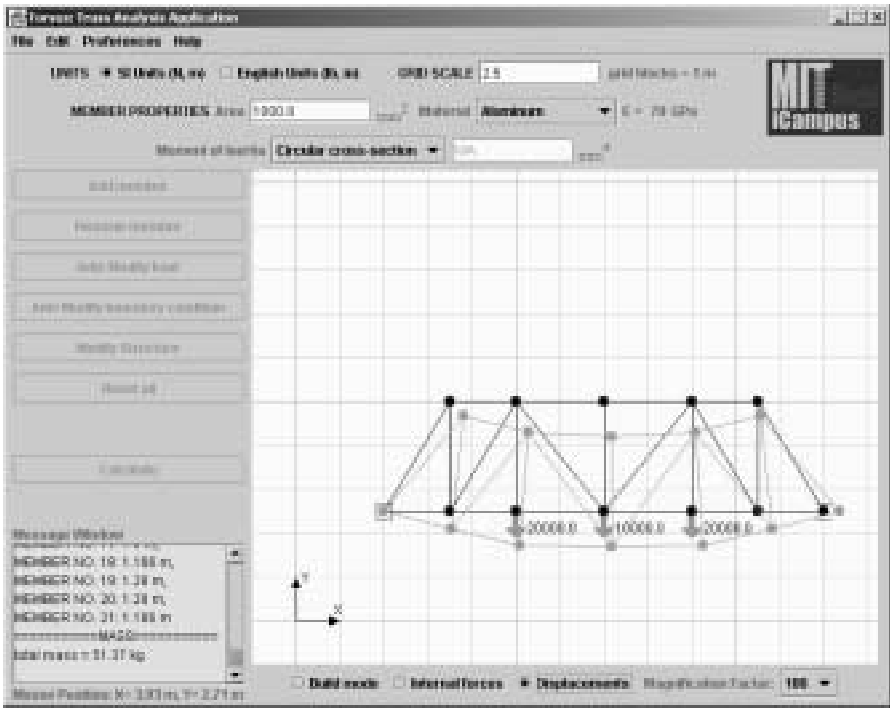
Tensile forces appear in blue.
Compressive forces appear in red.

MEMBER NO. 19: 1.166 m,
MEMBER NO. 19: 1.28 m,
MEMBER NO. 20: 1.28 m,
MEMBER NO. 21: 1.166 m
-------------MASS-------------
total mass = 51.37 kg

Mouse Position: X= 4.27 m, Y= 2.92 m

Magnification Factor: 1

Figure 7.5: Internal forces in the truss structure.



UNITS ● SI Units (N, m) □ English Units (lb, in) GRID SCALE 2.5 grid blocks = 1 m

MEMBER PROPERTIES Area 1000.0 mm^2 Material Aluminum E = 70 GPa

Moment of Inertia Circular cross-section mm^4

Message Window

MEMBER NO. 18: 1.186 m,
MEMBER NO. 19: 1.28 m,
MEMBER NO. 20: 1.28 m,
MEMBER NO. 21: 1.186 m
=============MASS=============
total mass = 51.37 kg

□ Build mode □ Internal forces ● Displacements Magnification Factor: 100

Figure 7.6: System displacements magnified by a factor of 100.

While students are able to modify the material and cross-sectional area of the structural members, they are not able to specify different cross-sections or materials for members individually. This was a conscious decision. In developing a simulation such as this, the focus should be on the creation of a learning tool, not necessarily an engineering tool. These two may or may not be the same, depending on the topic and the class, but in this case they were not entirely the same. The key was finding a balance between confining students to a variable set of parameters and allowing them enough freedom to partake in meaningful exploration. Certainly it is desirable to give good students the chance to be curious, but not at the expense of confusing the other students.

When students gain a better understanding of a specific topic, they are asked to use this simulation as a tool in a tackling a design problem presented to them. For the Truss Structures simulation, this involved the design of a pedestrian bridge.

The simulations are all currently based on Java technology, which was chosen for several reasons [12]. First, Java applets can call from a powerful suite of features, enabling an applet to be almost as powerful as a standalone application. Secondly, applets run in Web browsers by nature, which makes their integration into the Web module quite easy. Students can automatically have access to the latest version of the program by accessing the course site. This is a proven concept: Java applets have already been used successfully in mechanics education [84]. Consistency across platforms is achieved by relying on the 1.30 version of the Java plug-in, a tool that, if necessary, downloads and installs the correct version of the Java Runtime environment.

Torsus was developed in the summer of 2001 and used during fall 2001, for portions of 2 homework assignments, and for an in-class design exercise (Figure 7.7, Figure 7.8). The open-ended quality of this simulation, and others such as the Beam Bending simulation, allows future instructors to develop different design problems for the same tools. Framing the in-class design exercise as a contest provided solid motivation for the students. As they would in a game, students became fully engrossed with the design task. Some students were even witnessed using instant messaging (IM) capabilities to discuss the design problem with other teams, quite the opposite of the expected effect of exposure to IM in the classroom.

Figure 7.7: Students working together, using Torsus during an in-class design exercise (fall 2001).

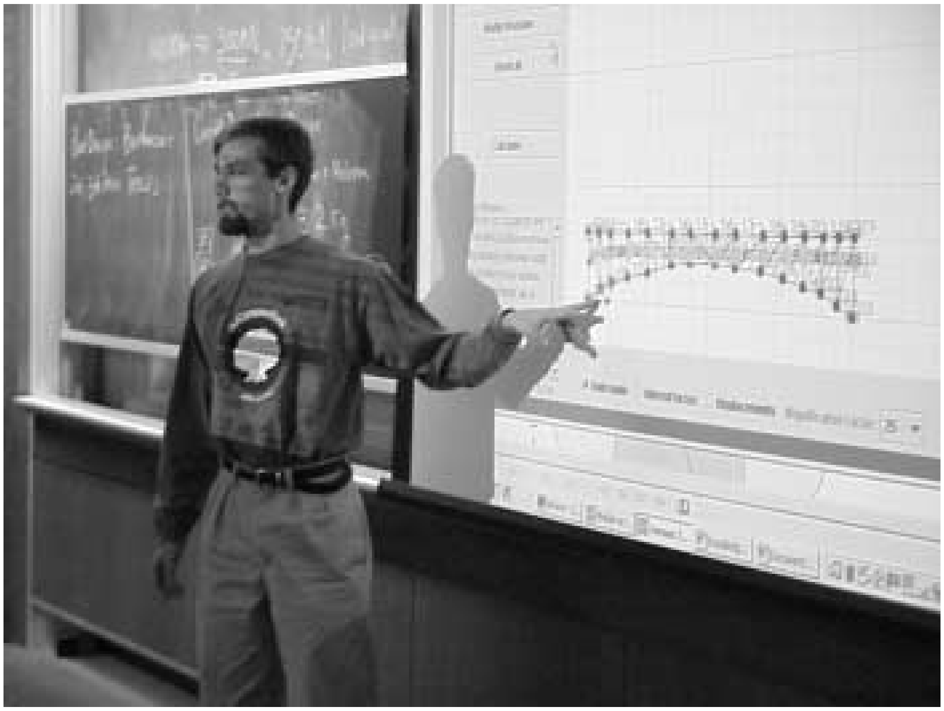


Figure 7.8: A student presents his team's best truss design to the class using his laptop, and the classroom's built-in projection equipment (fall 2001).

The computation engine of the program made extensive use of JAMA (Java Matrix), a matrix manipulation package developed by NIST and MathWorks [38]. Students generally provided positive feedback about the simulation; however, there were three major complaints. Students requested an undo function, were unable to copy the text-based data, and wanted the ability to save truss structures. The first of these - the undo function - was implemented directly into the applet using object serialization. The copy and save capabilities would not be possible from within the confines of the standard Java applet; access to the local computer’s resources (system clipboard and file system) was restricted by the nature of the applet. Two choices existed for resolving these issues: using signed applets or running Torsus as a standalone application. The latter was chosen and the new application will be used during the 3-unit IAP[12] course 2.973 (*Mechanics and Design*) [32]. The code can run as both an applet and as an application, but the save/load/copy functionality is specific to the application version (object serialization was again utilized, here to create and load *.trs extension files).

The extensibility of this code also enabled the rapid creation of a simulation for the Equilibrium module. The Equilibrium simulation used the same computation engine as Torsus, and a similar GUI. It enabled students to create structures and to determine reaction forces at the joints; though the engine was capable of analyzing statically indeterminate structures, the program restricted students to analyzing determinate structures since they had not yet been exposed to deformable bodies or geometric compatibility.

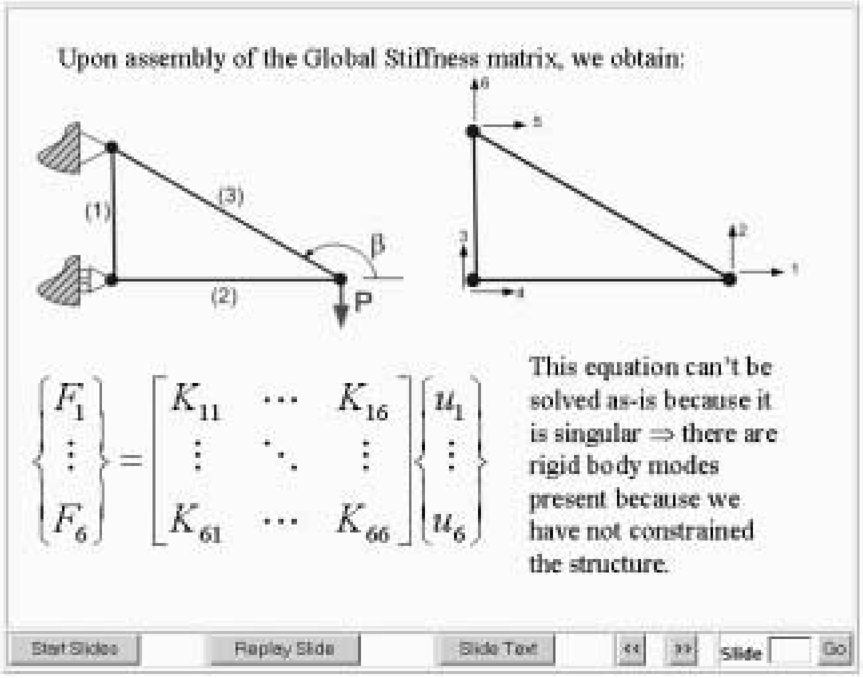
7.3 Online Lectures

Online lectures are primarily intended as a supplemental resource, allowing the students to

[12] Independent Activities Period (IAP) is a special 4-week term at MIT that runs from the first week of January until the end of the month. It was started in 1970.

review material at their own convenience. The Web-based lectures are important because they afford students of different learning styles the same ability to learn the material [104]. Our general approach is based on the online lecture model of Lozano-Perez [1][2]. The lectures are a narrated series of images (Figure 7.9); a custom tool[13] allows the conversion a Microsoft PowerPoint file to this Web-specific format [78]. Text of the narration is also offered for students who are hearing-impaired or simply prefer to skip the audio portion. The online lectures present the material covered in traditional class lectures. This format was chosen over a video recorded format for several reasons. First, video downloads tend to be slow, and streaming video requires a separate, dedicated video server, which can be expensive and difficult to maintain. Secondly, viewing the notes in this clear format is more value-added than watching a video of an instructor writing on a chalkboard, and also enables a certain level of interactivity with the lecture. Lastly, despite the initial time investment, this format enables a quick and reliable system for updating lectures as time progresses. Changing a portion of a video lecture for a future class is not only time-consuming; it may necessitate that the entire lecture be redone.

[13] http://www.mit.edu/~jsandhu/ppt.html



Upon assembly of the Global Stiffness matrix, we obtain:

$$\begin{Bmatrix} F_1 \\ \vdots \\ F_6 \end{Bmatrix} = \begin{bmatrix} K_{11} & \cdots & K_{16} \\ \vdots & \ddots & \vdots \\ K_{61} & \cdots & K_{66} \end{bmatrix} \begin{Bmatrix} u_1 \\ \vdots \\ u_6 \end{Bmatrix}$$

This equation can't be solved as-is because it is singular ⇒ there are rigid body modes present because we have not constrained the structure.

Figure 7.9: Narrated PowerPoint lecture on Matrix Structural Analysis.

In the future, the lectures will take the form of narrated Flash[14] animations. This will enable the online lectures to leverage the full power of information technology, through the inclusion of animations and movie clips. This will necessitate more time in creating and maintaining lectures, but the integration of multimedia elements is expected to outweigh this increased commitment.

7.4 Textbook-Style Derivations and Explanations

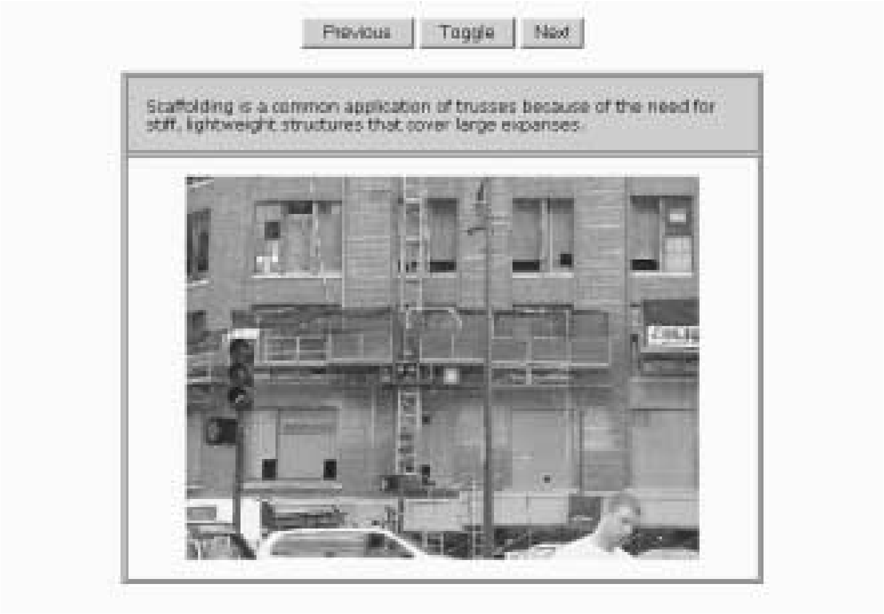
The derivations section contains a much more detailed presentation of the material and goes beyond what is covered in lecture (Appendix D). These pages serve to analytically examine the principles and equations underlying a specific topic, similar to a textbook. This system, however, has many advantages over a traditional textbook. First, the use of hyperlinks allows us to create a set of core documents that contain only essential material. Detailed derivations are generally provided through hyperlinks - as are the simulation, ref-

[14] Flash is a Macromedia format that plays in most Web browsers.

erence material, real world examples, and the case study - allowing the students to easily navigate and quickly find relevant material. Given that engineering education builds heavily on previously learned material, providing such navigation aid makes the learning process considerably easier. Additionally, the Web allows for the inclusion of many multimedia objects, such as animations and movie clips that can illustrate points more clearly than any description.

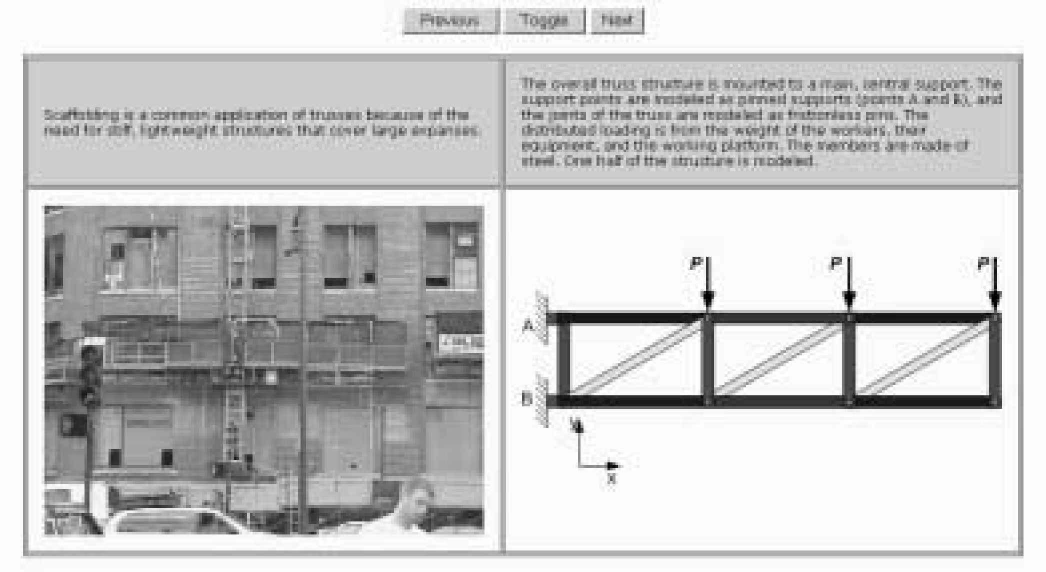
7.5 Real World Examples

The real world examples section seeks to strengthen a major weakness that is often encountered in an engineering class: the inability of students to take a real problem, make reasonable abstractions and simplifications, and solve the problem with the tools acquired. The modeling required is an area where students tend to be lacking, because they are generally used to the schematics provided in textbook problems. We have therefore created an ASP (Active Server Pages) utility that allows users to scroll through a library of photographs of real world examples (Appendix E). Each of these photographs is tied to a description of the situation (Figure 7.10). Students can then view a schematic drawing of the real world system (Figure 7.11), along with a description of the schematic model and the assumptions made in creating it. Students are challenged to determine a mechanical model for solving a problem and can then review the model. In appropriate cases, links are included to other portions of the site, such as the case study, where the examples are studies to completion. The real world examples section gains importance when students are later presented with a design problem.



Scaffolding is a common application of trusses because of the need for stiff, lightweight structures that cover large expanses.

Figure 7.10: Real world example showing building scaffolding.



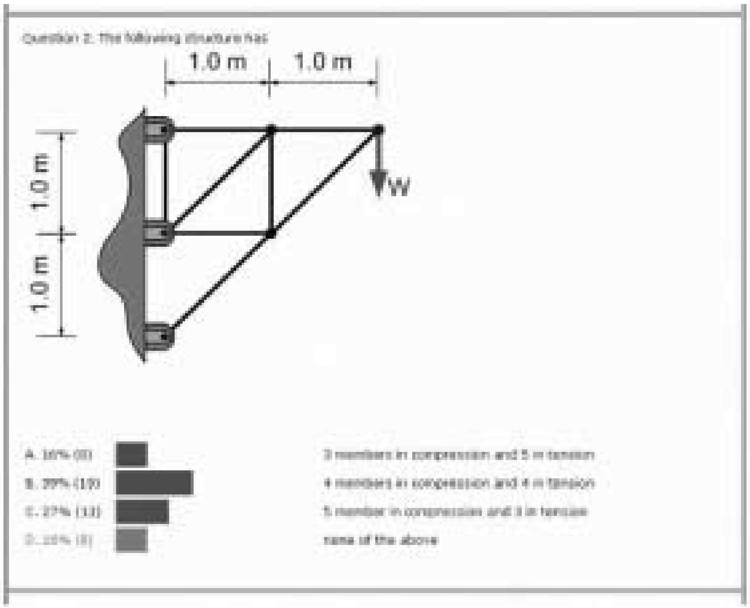
Scaffolding is a common application of trusses because of the need for stiff, lightweight structures that cover large expanses.

The overall truss structure is mounted to a main, central support. The support points are modeled as pinned supports (points A and B), and the joints of the truss are modeled as frictionless pins. The distributed loading is from the weight of the workers, their equipment, and the working platform. The members are made of steel. One half of the structure is modeled.

Figure 7.11: The same page also showing the schematic model of the scaffold-

7.6 Dynamic Quizzes

The online quizzes are tools for assessment, both for the students and the faculty. The quiz section actually consists of two quizzes, an in-class quiz and an at-home quiz. The former is administered during class, and the latter is one that the students can use on their own. Both quizzes make use of ASP, which allows them to access a database and produce dynamic content. Following the student engagement model of Mazur [51], the in-class quiz consists of short conceptual-type questions that the instructor poses to the class. The students use their computers to respond to the questions. The instructor then displays a screen showing the distribution of answers (Figure 7.12). This allows the instructor to identify common misconceptions and the student to gauge his or her understanding of the material. A tool has also been built that allows the instructor to ask a question "on-the-fly." This tool enables the instructor to pose a general type of question (true/false, yes/no, A/B/C/D) and to electronically receive responses from the students.



Question 2: The following structure has

A. 16% (6) 3 members in compression and 5 in tension

B. 39% (15) 4 members in compression and 4 in tension

C. 27% (11) 5 member in compression and 3 in tension

D. 16% (6) none of the above

Figure 7.12: In-class quiz results (correct answer shown in green, incorrect answers in red).

The at-home quiz randomly draws from a large set questions (Appendix F). The questions are subdivided into 20% textbook-style problems, and 80% short, analytical and conceptual questions. The textbook-style questions are similar to those found in traditional mechanics texts: students are provided with a drawing of a system and other crucial information (e.g. loads and dimensions), and are asked to answer questions about the system. The analytical questions probe knowledge that is based more strongly in the mathematics of the topic (e.g. "What is the bending stiffness of a steel plate 1 m wide and 5 cm thick?"). The conceptual questions seek to test fundamental knowledge about a topic (e.g. "Which of the following parameters affect the stiffness of a truss element?"). Five questions are asked per instance of the quiz, giving students ample opportunity to test their knowledge of a topic. Submitting the quiz calls another ASP page; this page displays the correct and incorrect answers, giving instant and detailed explanations about the answers to the questions that the student answered incorrectly.

7.7 Design Problem

A design problem that pertains to the module topic is presented to the students. For most of the students, this is their first exposure to design; this section acts both to introduce the concepts of design and to create another learning opportunity for the current topic. For example, in Truss Structures, the problem asks students to design a pedestrian bridge given specific weight and dimensional constraints (Appendix G). They are expected to make additional assumptions regarding the problem. They are additionally expected to use the Truss Structures simulation to aid in their design. The fact that there is more than one answer to the problem forces students to examine all aspects of mechanics and to draw from their own experiences. By introducing design at such an early stage of the curriculum, it is hoped that the students will be much more comfortable with - and proficient in - design. It is also important to note that new design exercises may be created over time;

the open-ended nature of the simulations allows for instructors to develop new design exercises to meet their exact teaching desires.

7.8 Reference Section

The target audience for the reference section is the group of students who have already gained proficiency in the topic. After completing a module, students may at some point in the future need to revisit the subject. This reference section gives those students quick access to key information that presupposes knowledge of the topic. A typical reference section contains a topic overview, material properties, and solutions to common problems.

7.9 Case Study

The case study is an involved analysis of an actual engineering problem. This section differs from the real world examples in that it introduces a problem from its conception and leads the student through the entire design process. Not only does this offer the student a deeper understanding of the real-world applications of a topic, but it also allows the student to experience the process of mechanical design. The problem presented in the Truss Structures module is a variation on the Michell truss (Appendix H) (Figure 7.13). The case studies are all presented in the same, systematic manner. First, design requirements are gathered based on the basic problem statement. Upon determination of the design parameters, a schematic model is developed. This model is used to determine the underlying mechanics of the problem. Using this information, and the simulation, the design is optimized. This optimization process gives students a better understanding of the procedures involved in mechanical design, and the use of the simulation allows students to actively participate in the design process. Finally, students are presented with an example of an actual design. In the case of the Truss Structures module, the case study presents solutions to the original Michell problems (Figure 7.14); these solutions are typically used to benchmark topological optimization programs. This actual design enables students to understand the motivations for design and to appreciate the design process.

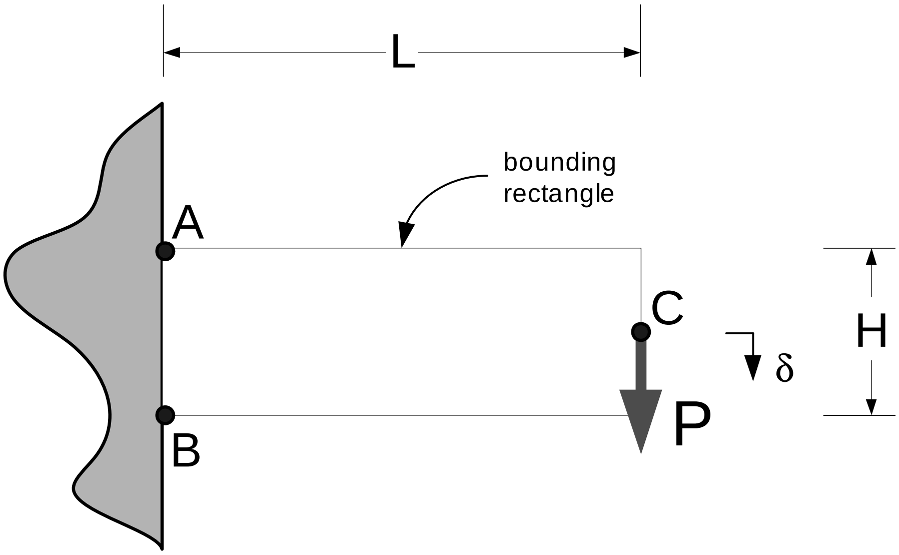
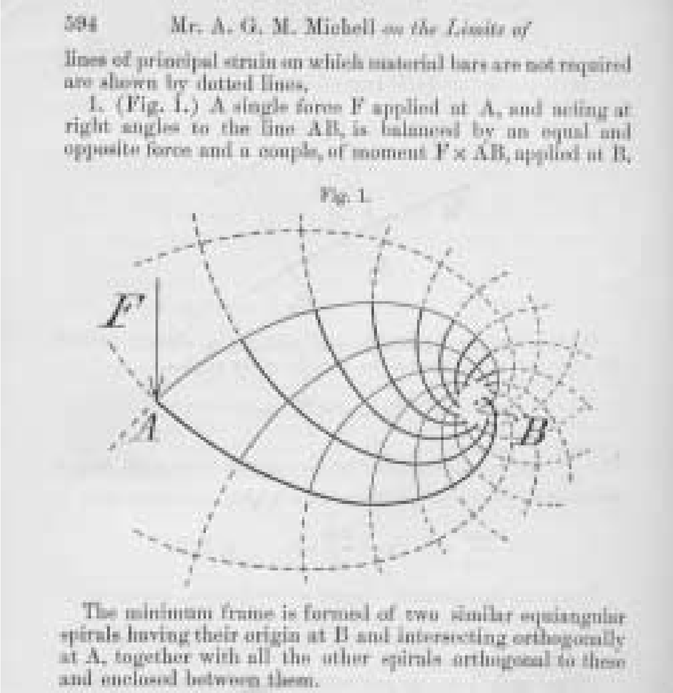


Figure 7.13: Truss Structures case study: a variation on the Michell problem.



594 Mr. A. G. M. Michell *on the Limits of*

lines of principal strain on which material bars are not required are shown by dotted lines.

1. (*Fig.* 1.) A single force F applied at A, and acting at right angles to the line AB, is balanced by an equal and opposite force and a couple, of moment F × AB, applied at B.

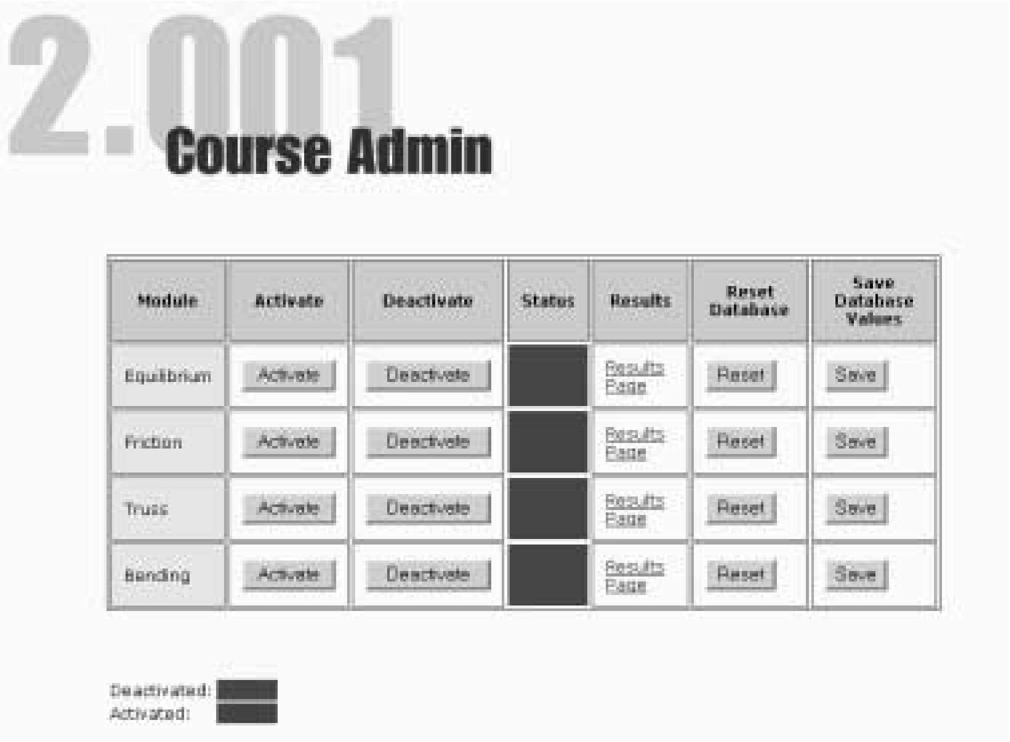
Fig. 1.

The minimum frame is formed of two similar equiangular spirals having their origin at B and intersecting orthogonally at A, together with all the other spirals orthogonal to these and enclosed between them.

Figure 7.14: Solution to one of Michell’s original problems from Truss Structures case study [59].

7.10 Instructor Interface

The instructor is provided with an interface for facilitating the classroom experience. The main goal in creating this interface was to create a quick and intuitive means for the instructor to perform necessary actions. The instructor has access to a restricted portion of the site. From this section, the instructor can control access to the in-class quizzes, view student feedback, and review at-home quiz questions by module. For the in-class quizzes, the instructor can activate or deactivate the quiz, reset the response database, display quiz results, and store the quiz results to a file for later review (Figure 7.15).



2.001 Course Admin

Module	Activate	Deactivate	Status	Results	Reset Database	Save Database Values
Equilibrium	Activate	Deactivate	■	Results Page	Reset	Save
Friction	Activate	Deactivate	■	Results Page	Reset	Save
Truss	Activate	Deactivate	■	Results Page	Reset	Save
Bending	Activate	Deactivate	■	Results Page	Reset	Save

Deactivated: ■
Activated: ■

Figure 7.15: Instructor interface for in-class quizzes.

Chapter 8

Assessment

8.1 Assessment Procedures

Assessment is an important, and often neglected, part of the development of new pedagogies. Even when assessment is conducted, it is often of poor quality [9][31]. The ideal method for assessing any pedagogical initiative is to teach half of the class with the new methodology, teach the other half with the previous methodology, and then evaluate their performance. David Wallace and others have successfully used this approach in the evaluation of computer-based tools for Mechanical Engineering education [104][29]. This was not a practical option for 2.001, however, because of external limitations. Specifically, as the first course in the Mechanical Engineering major, 2.001 serves as an introduction to the department, and has a significant impression on the students who take the course. Teaching the class in such an experimental fashion might risk a negative impact on student retention in the department. Additionally, the resources needed to teach such a large class using two different methodologies - even for a portion of the course - were not realistically attainable.

The next best approach was to obtain data to allow inter-semester comparison of results; the effectiveness of each paradigm could be measured on different groups of students taught using different approaches. This assessment procedure had multiple components: surveys, focus groups, student performance in terms of grades, in-class observation, and server logs. The use of multiple indicators was needed to provide a more assertive basis for drawing conclusions. Hsu indicates the lack of in-depth assessment for educational initiatives, and praises the strengths of an assessment methodology that combines “surveys with focus groups and classroom observations” [31]. These multiple indicators were used for both formative and summative assessment of the project.

Formative assessment is the process of obtaining information to enhance teaching and learning, while summative assessment is the process of gathering information for gauging the relative success of a teaching methodology [8]. Formative assessment is an ongoing process in which we continuously seek to improve the teaching paradigm. Summative assessment is a process of evaluation which necessitates a continuous gathering of data. While some summative assessment can be made regarding this initiative to date, it is imperative to not prematurely make judgement, either positively or negatively, on its success. Statistics at the early stages of curricular development can be suggestive, but should not be used to make a definitive evaluation of new methods [4].

At this stage in the project - recall that this is a multiyear initiative - a simple, binary judgement on the success of this initiative is not appropriate. Instead, it is better to provide a detailed assessment of the successes and shortcomings of the project to date. This section also provides the basis for future work, both in terms of further developing the pedagogy and continuing assessment.

8.2 Course Structure

During fall 2000 and spring 2001, desktop experiments were used in recitation, but the course was primarily taught in the traditional manner. The new teaching format was tested out during one week in spring 2001, when the Bending Web module was used in-class. During fall 2001, the course was taught using a mixture of lecture, desktop experiments, in-class Web-enabled exploration, and peer discussion, all using the new 2 hour lecture format. Approximately 1/2 of the sessions were of the pure lecture-style, though cooperative learning and demonstrations were included in these lectures (Figure 8.1). Though this mixture of new and old was due to resource limitation, it provided data that could be compared to previous as well as future classes.

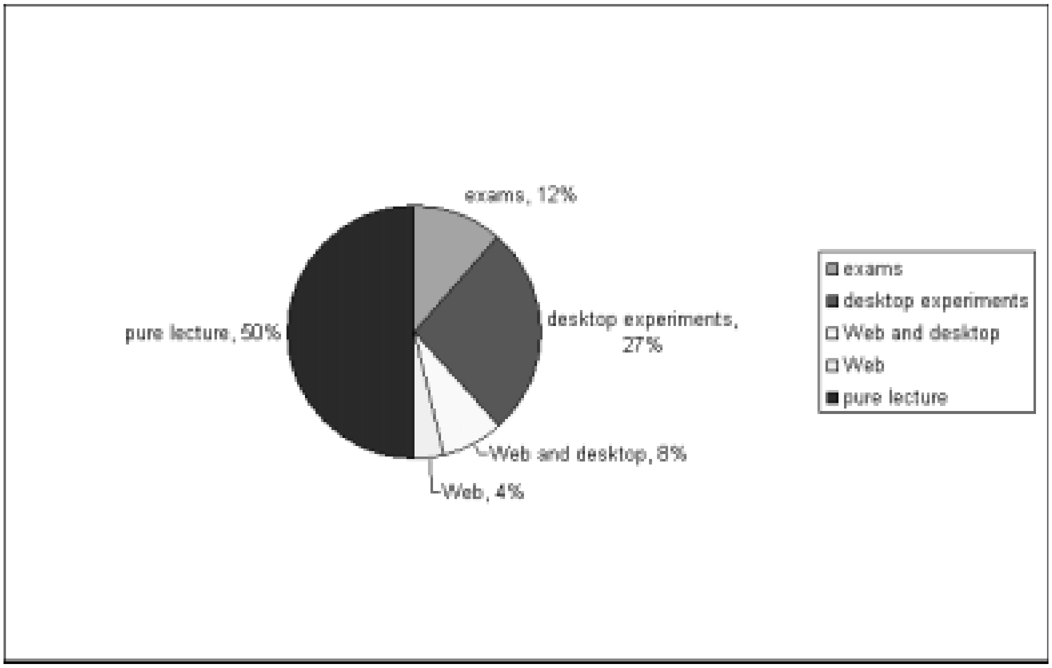


Figure 8.1: Breakdown of 2.001 lectures from fall 2001.

8.3 Exams

Part of this assessment procedure took the form of evaluating student performance on a particular set of questions. Specifically, in spring 2001 and fall 2001, 25% of the final examination was comprised of a set of short-answer conceptual questions. Though these questions changed from one semester to the next, they were of similar difficulty, as determined by the teaching staff that developed the questions, and probed the same topics. This procedure also provides capacity for future evaluation in a similar manner.

There was no significant difference in the scores of the students who took the exam in spring 2001 and fall 2001. During the spring term, students scored an average of 76.8% on these questions, with a standard deviation of 14.9% (Figure 8.2). During the fall term, the average was 76.6%, with a standard deviation of 11.5% (Figure 8.3). These data do not provide any conclusion at this point, though they do provide for a future basis of comparison, when greater portions of the course will be taught using the active engagement pedagogy. Specifically, since the desktop experiments and one of the Web modules were used in spring 2001, there may not be a noticeable difference in these scores until significant progress has been made in implementing the new teaching methodology. At this stage, these data only suggest that the new teaching methodologies do not have a negative impact on conceptual understanding.

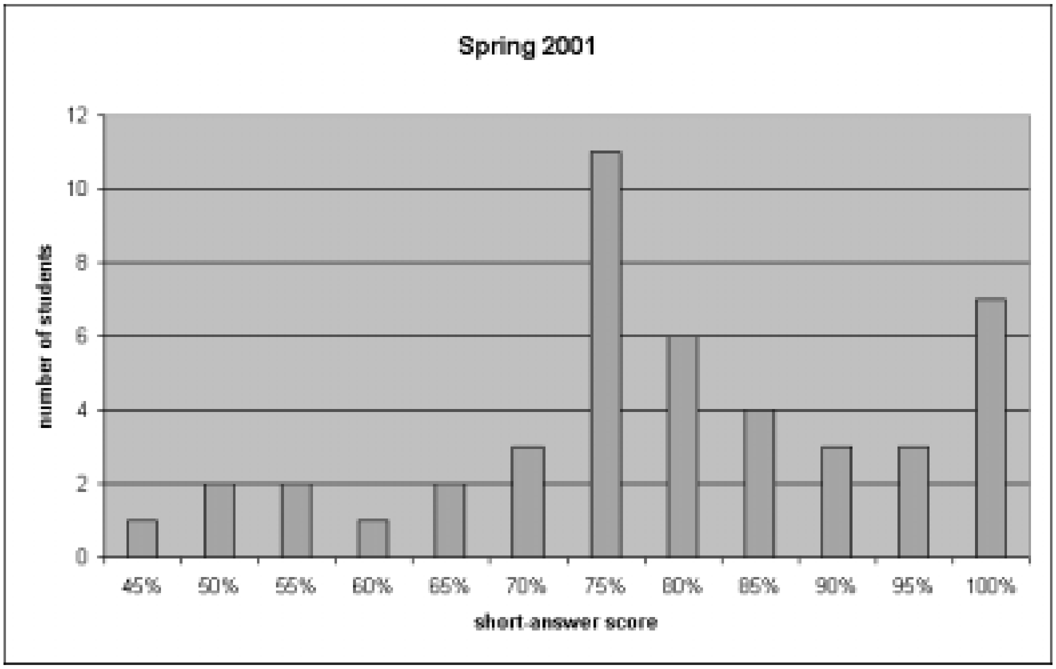


Figure 8.2: Distribution of scores on short-answer portion of spring 2001 final examination.

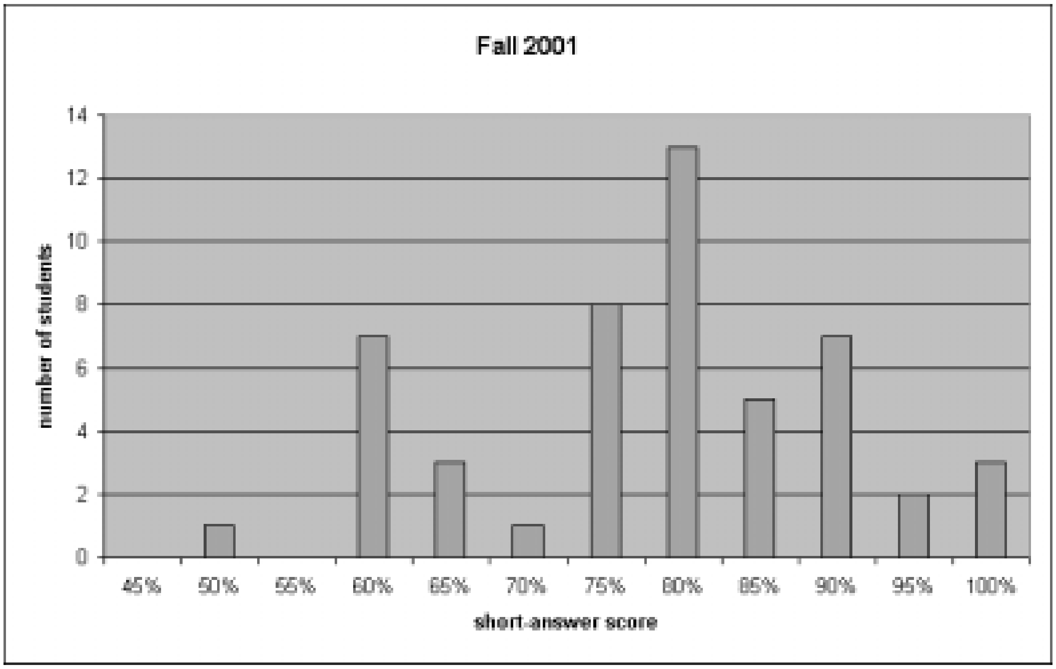


Figure 8.3: Distribution of scores on short-answer portion of fall 2001 final examination.

8.4 Surveys

Surveys have taken two basic forms in 2.001: Pi Tau Sigma evaluations and self-developed surveys. Pi Tau Sigma, a Mechanical Engineering honor society, has students administer standardized surveys for all classes in the department; these surveys are typically administered twice during the semester, once in the middle and once at the end. In addition to these generalized comments, it was necessary to get feedback about the desktop experiments, the laptop computers, the Web modules, and the new teaching initiative. These surveys provided summative data, as well as comments for formative evaluation. These project-administered surveys are listed in Table 8.1.

Academic Term	Survey Description
Fall 2000	End-of-term general survey
Spring 2001	Midterm general survey
Spring 2001	Bending Web module survey
Fall 2001	Midterm general survey
Fall 2001	Truss Web module survey
Fall 2001	End-of-term general survey

Table 8.1: Table of survey descriptions.

With respect to the desktop experiments, we asked students to rate their utility in helping learn the material. From this we found that students had a more positive opinion of the experiments in fall 2000 and spring 2001 than in fall 2001 (Table 8.2). This may have been due to the integration of the experiments into the 2 hour lecture; prior to fall 2001, the experiments were conducted during recitation. An important point to note, however, is that these results from the fall 2001 survey contradict results from the same survey, which indicate that most students wanted to do more experiments in lecture (Table 8.3). Only 7% of students wanted the experiments to be a less essential part of the class. Students during spring 2001 had a significantly different opinion on this issue, despite the experiments being used in the same manner as the prior term.

	Fall 2000	Spring 2001	Fall 2001
Number of Responses	24	33	46
Average	3.56	3.18	2.90

Table 8.2: Response to a survey question that asked students to rate the desktop experiments in helping them learn the material (scale 1-5, 1 = poor, 3 = average, 5 = outstanding).

	Fall 2000	Spring 2001	Fall 2001
Standard Deviation	1.12	0.88	1.17

Table 8.2: Response to a survey question that asked students to rate the desktop experiments in helping them learn the material (scale 1-5, 1 = poor, 3 = average, 5 = outstanding).

	Fall 2000	Spring 2001	Fall 2001
Number of Responses	24	33	44
More	71%	21%	61%
Less	8%	42%	7%
Same	21%	36%	32%

Table 8.3: Response to a survey question that asked students whether the number of experiments should be increased, decreased, or kept the same.

Students in fall 2000 and spring 2001 suggested that the use of computers in addition to these physical experiments would be helpful; one student stated “it'd be a good idea to have a little computer work on some of the analysis” while another said the “an incorporation of higher technology, computers,” would be beneficial. Many students indicated that the desktop experiments enabled them to better understand the course material: “[the experiments] provided a lot of insight into what the theory and equations were trying to convey.”

Another survey question asked students to rate their overall satisfaction with the class. Notable was a decline in fall 2001 from previous terms; however, the satisfaction level at the end of the term showed a slight improvement, perhaps due to the increased integration of Web-based course materials (Table 8.4).

	Fall 2000	Spring 2001	Fall 2001 (midterm)	Fall 2001 (end-of-term)
Number of Responses	24	33	41	35
Average	3.56	3.61	3.12	3.34
Standard Deviation	0.80	0.86	1.05	1.06

Table 8.4: Response to a survey question asking students to rate their overall satisfaction with the class (scale 1-5, 1 = poor, 3 = average, 5 = outstanding).

Students were also asked to rate the aspects of the class that best helped them learn. Their responses - even after the preliminary implementation of the new teaching paradigm - indicated a strong preference for homework, lecture, recitation instruction, and office hours (Table 8.5). Two students indicated at the end of the term that the iCampus website was the most helpful part of the class. These results are likely tied to the fact that students are primarily graded on exams: student performance on these exams is most helped by homework and activities that pertain specifically to homework, such as recitations and office hours. The grading methodology is the motivating factor for student assessment of what best helps them learn. Even though desktop experiments and the Web modules may help them develop conceptual and high-level understanding, they do not provide a direct connection to algorithmic problem-solving, so students may feel that these aspects of the class are not helping them learn.

	Spring 2001	Fall 2001 (midterm)	Fall 2001 (end-of-term)
Number of responses	54	46	35

Table 8.5: Response to a survey question asking students to indicate the aspect of the course that best helped them learn.

	Spring 2001	Fall 2001 (midterm)	Fall 2001 (end-of-term)
Lecture	20%	17%	17%
Recitation Instruction	26%	24%	29%
Desktop Experiments	7%	2%	0%
Homework	28%	33%	26%
Office Hours	19%	11%	17%
Textbook	N/A	N/A	6%
iCampus Website	N/A	0%	6%
Other	0%	13%	0%

Table 8.5: Response to a survey question asking students to indicate the aspect of the course that best helped them learn.

Other questions probed the use and opinion of the iCampus website. Most students indicated that the website was useful in helping them learn, a promising result (Figure 8.4, Figure 8.5).

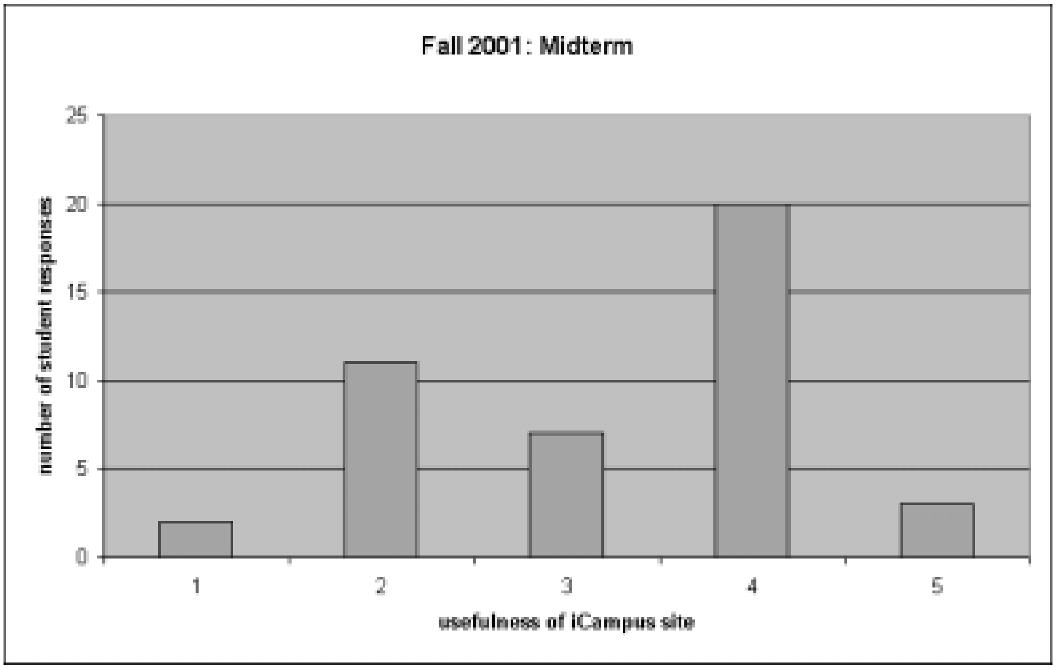


Figure 8.4: Distribution representing student response to a survey question asking them to rate the usefulness of the iCampus website: midterm, fall 2001 (scale 1-5, 1 = poor, 3 = average, 5 = outstanding).

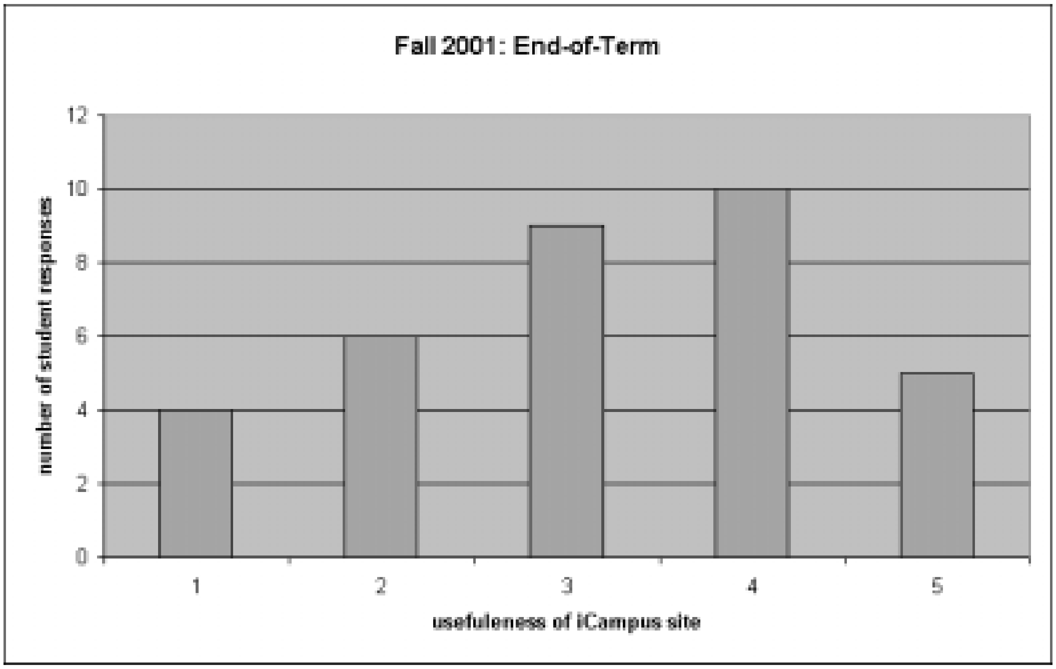


Figure 8.5: Distribution representing student response to a survey question asking them to rate the usefulness of the iCampus website: end-of-term, fall 2001 (scale 1-5, 1 = poor, 3 = average, 5 = outstanding).

To gauge what aspects of the site were most useful, we asked students to rank the top 3 portions of the site in terms of usefulness. The choices were each of the core module components: simulation, quiz, electronic textbook, real world examples, online lectures, case study, and reference section. The implication of these results is that the students find the simulation the most useful portion of the site by far (Table 8.6, Table 8.7). The electronic textbook and online lectures are also considered important; to a lesser degree some students find the quiz and real world examples helpful. The reference section and case study do not appear to be highly valued by any substantial number of students. One issue that this reveals is the need to motivate students to use these other parts of the site; through this motivation, we will be better able to gauge the effect that these components have on student learning.

	Fall 2001 (Truss module)	**Fall 2001 (end-of-term)**
Simulation	54%	66%
Quiz	4%	0%
Electronic Textbook	17%	6%
Examples	17%	6%
Lecture	4%	16%
Case Study	0%	6%
Reference	4%	0%

Table 8.6: Response to a survey question asking students to rank the top 3 portions of the iCampus site in terms of usefulness. Results given are proportion of top (#1) ranking.

	Fall 2001 (Truss module)	**Fall 2001 (end-of-term)**
Simulation	31%	36%
Quiz	8%	12%
Electronic Textbook	20%	17%
Examples	22%	12%
Lecture	14%	11%
Case Study	2%	6%
Reference	5%	7%

Table 8.7: Response to a survey question asking students to rank the top 3 portions of the iCampus site in terms of usefulness. Results given are proportion of top 3 (#1-3) rankings.

During the fall 2000 survey, 100% of the respondents said they would use Web tutorials, and 83% said they would like lectures to be put into some online format. They indicated a desire to see a variety of features on such a website, but the common themes were: interactivity, animation, visual representation, and real applications. These comments helped us revise and add to our Web modules for the following 2 terms.

A strong majority of the fall 2001 class thought that the iCampus site was a positive addition to the class (Table 8.8). The indications from comments received from students opposed to the idea generally centered around the problems arising from the fact that the Web modules were not yet fully developed.

	Fall 2001 (Truss module)	**Fall 2001 (end-of-term)**
Number of responses	24	35
Yes	96%	80%
No	4%	20%

Table 8.8: Student response to a survey question asking whether they thought the iCampus site was a positive addition to the course.

Negative student sentiments regarding the new teaching paradigm indicated issues with the 2 hour lecture being too long, the iCampus site needing more content, and the site needing to be publicized better. These were all a result of the partial development of the Web modules. Over the next two terms, as the Web modules become fully developed and integrated into the class, there will be few - if any - 2 hour lecture sessions and the iCampus site will have sufficient content to satisfy the needs of the students. As for the issue of publicizing the website, this was a lesson learned in the first implementation of the new pedagogy and we plan to make the role of the website clear in future terms.

8.5 Focus Groups

We held two student focus group sessions with MIT Information Systems (IS) in order to assess the new pedagogy.[15] One was held during the middle of the fall 2001 term, and the other was held at the end of the same term. IS was interested in determining how the students used the laptops, while our interest was in finding out how students felt about the new teaching format. Very little quantitative data was gathered from these focus groups; the intent was to get a more qualitative sense of student feelings towards the course.

[15] We also held student focus groups with MIT IS during January 2000 for purposes of formative evaluation of the bending Web module during the development phase, as mentioned in Chapter 5.

During the first set of focus groups, prior to use of the laptops, we gauged student expectations. 21 of the 22 students who attended already had computers, and 3 of them had other laptops, but this group generally responded positively to the laptop initiative, indicating the mobile capability was expected to be useful, and the wireless access would be key. They also acknowledged that very few of them were comfortable using Athena. Many students did not think that the laptops would be appropriate for the course material given their understanding of Solid Mechanics.

The second set of focus groups concentrated on student perception of the course, particularly what needed to be improved. Students expressed feelings that the 2 hour lectures were too long, that more in-class labs were needed, that in-class laptop use needed to be increased, that there needed to be more site content, and that the site needed to be better integrated into the course. These concerns matched the student response on the surveys and were primarily a result of the partial development and integration of the Web modules, as mentioned previously.

8.6 Server Logs

The logs recorded by the server used for hosting the new course website also provided valuable information, specifically about the use of the Web modules. From this data - over 300,000 log entries- we were able to extract information that indicates how the students used these modules. Access to the 2.001 site was restricted to students, teaching staff, developers, and a handful of colleagues with an interest in engineering education. In analyzing the logs, we analyzed only student access to the site.

First, we measured visits to the main site page; since this served as a portal to all other parts of the site, it would be a very good indicator of how often students accessed the site.[16] The weekly data indicated that the peak traffic was during use of the Truss Module, during weeks 7-9 (Figure 8.6). This usage can be explained by the heavy integration of this module into the class; students used the site for homework, an in-class design exercise, and an at-home team design project, as well as at their own convenience. Usage of the equilibrium module (released in week 5) was minimal in contrast; further analysis showed that the equilibrium module received only 62 hits over the course of the semester. Though this was made available to the students prior to their first exam, it was not integrated into any portion of the class. The second most active periods were during weeks 11 and 12, corresponding to the use of the Multiaxial Stress-Strain simulations and the Bending module, respectively. These were more closely tied to other portions of the class than the Equilibrium module, but not so much as the Truss Structures module. Though students did not appear to have used the Web modules for studying for exams during the semester, a spike in usage prior to finals week suggests that some portion of the class did use it to study for the final examination.

[16] By bookmarking they could circumvent this page, but this was not a common occurrence.

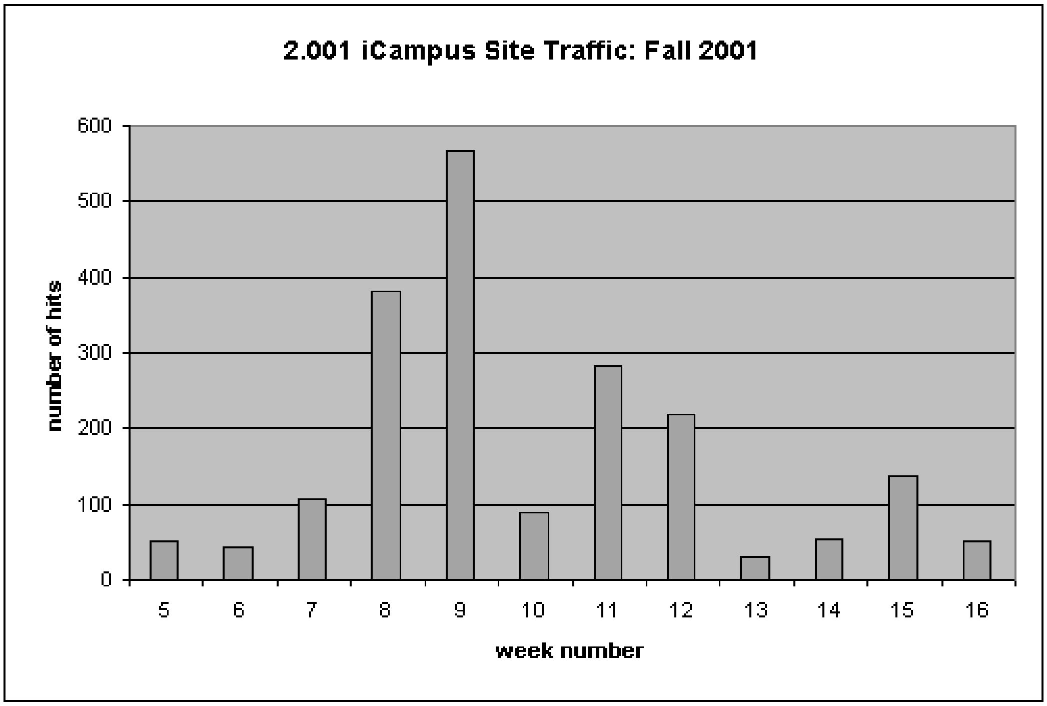


Figure 8.6: Weekly student visits to the main iCampus website page. Week 1 marks the first week of the fall 2001 academic term, and week 16 represents final examination week.

Based on the heavy traffic sparked by the Truss Structures Web module, a closer examination of the use of this module was undertaken. Specific analysis entailed the proportion of traffic on each of the Truss module components (Figure 8.7). Over the course of the term, we found that students spent the most time using the simulation. Derivations and examples were also used substantially, while lectures saw less traffic. According to these data, the quiz, reference, and case study were used sparingly. This generally matches with our survey results. As the project is still in its early stages, this does not imply that we should halt development of the lesser used components. We need to find a way to motivate students to use these components; assessment resulting from such use in future terms will provide more substantial data, which can in turn be used to make decisions regarding the development of these components.

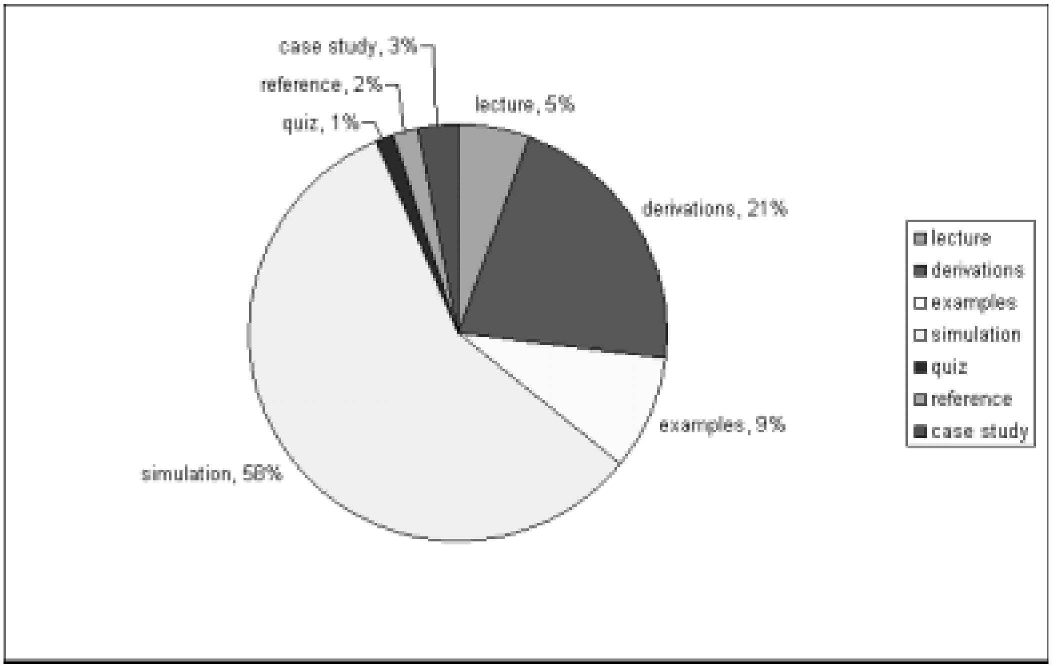


Figure 8.7: Chart indicating student usage of each of the Truss Structures Web module components.

We also examined the use of each of these Truss module components over the course of the semester (Figure 8.8). This analysis indicates that peak use of the module components was during the week surrounding the in-class truss design exercise, as expected, since the module was incorporated into homework, the in-class exercise, and an at-home design project. All indications suggest that students did not use the module to study for any of the exams.

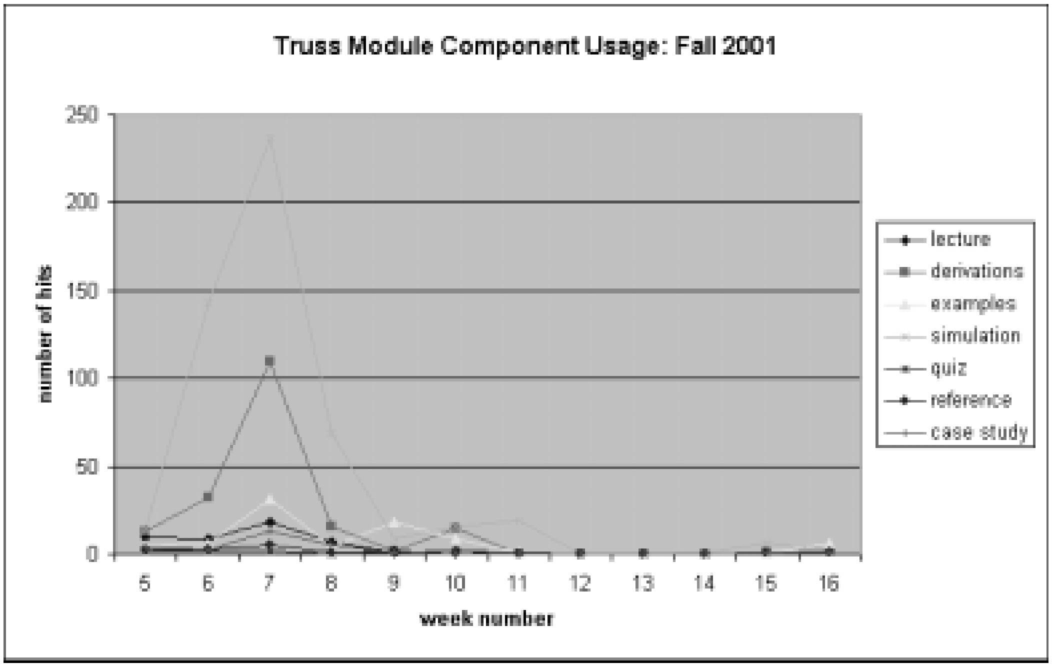


Figure 8.8: Chart indicating student usage of each of the Truss Structures Web module components over the course of the fall 2001 academic term.

8.7 Future Assessment

Formative assessment will continue to tailor the project to the educational needs of students and summative forms of assessment will allow the department to decide on the impact, sustainability, and appropriateness of the new teaching paradigm. It is important to realize that assessment is a significant and ongoing process; as this is a multiyear initiative, assessment will require continued time and attention.

The assessment methods that are in place must be applied to future classes in order to provide further comparative data. Short-answer exam questions, project-developed surveys, Pi Tau Sigma evaluations, focus groups, instructor feedback, and server data must all be used for this purpose, and should take similar form to the processes used during the last 1 1/2 years. Additionally, a multiple-choice quiz covering the major topics in Solid

Mechanics will be developed for *2.007, Design and Manufacturing I*, a follow-up course to 2.001; this quiz, which will be administered in-class, will give us a good sense of student retention of learned material. This quiz will be given at the beginning of the 2.007 course each time it is offered in order to provide comparative data. Together with the other indicators, this will provide a legitimate means for evaluating the effectiveness of the new pedagogy.

Chapter 9

Recommendations and Future Work

Assessment gives us the opportunity to examine the current state of this initiative. High-level findings indicate that the new pedagogy works well when everything is in place, but not otherwise. Classes which integrated desktop experiments and/or laptop exercises were very successful, but full 2 hour lectures were not. Evidence for the potential of this pedagogy is given by the use of the Truss Structures Web module. Given the development time needed to create the tools for the new paradigm, we currently find ourselves in a period of transition. As a result, there will inevitably be issues with operating between two modes of teaching, specifically with long lectures and little, in-class laptop use. The question is how we address the issues identified and improve the effectiveness of the active engagement pedagogy. Following are a set of the most important recommendations for implementation of the active engagement pedagogy in 2.001, for the spring 2002 academic term and beyond. These recommendations are based wholly on the formal project assessment presented in the previous chapter.

The site needs to be better publicized. Introducing the module components to the students at the beginning of the term - as was done with the Truss Simulation - would expose students to the wealth of information on the site. Many students commented that were unaware of the information available on the site until it was too late. Also, at the beginning of the academic term, it will be necessary to explain, in a straightforward manner, the purpose of the website, desktop experiments, and laptops.

Laptop use needs to be increased. This laptop use must be valuable to the learning experience, though; this is why students used the laptops in-class primarily for simulations. This use will obviously increase as we are continuously in a process of development, but use of the laptops for simulations will only result in 5-6 in-class laptop lectures during spring 2002. More laptop use in the classroom will allow us to leverage the power of these tools (lack of in-class use was a direct complaint of many students). With the laptops present in the classroom, we can begin to leverage the faculty-student "communication" tool, in the Mazur style. The single biggest student complaint was the length of the lectures where there was no experiment or laptop use; increased laptop use will help ameliorate this student concern. A proposed solution is having them use the laptops for desktop experiments. They can record data, analyze results, and answer questions on their laptops, since there is already a tool in place for electronic file submission.

Use of the Web modules in homework assignments was successful and must be continued. As with every other aspect of the pedagogy, the most important aspect of doing this is ensuring that it helps students learn the material. The team truss design project was highly successful; students indicated, as did the course TA, that this was a good learning experience based on what they had done and seen in class. Some students indicated that this was the single best learning experience in the course.

Enough time must be allowed for in-class experiments. This has proved to be difficult to gauge, but experiments that ran past the end of lecture frustrated students, interrupted the flow of the class, and may have hindered learning. Students complained that the focus was on measurement and results rather than understanding. Having attempted to integrate these experiments into the 2 hour lecture, we will have a much better sense of how long students need to complete these experiments.

Finally, and most importantly, project assessment must continue. The procedures for doing so have already been outlined, but the concept is repeated here because of its significance. Assessment will allow us to consistently gauge the effectiveness of the most current procedures, and will allow us to modify the pedagogy accordingly in real-time. Additionally, this assessment process will help us determine the overall effectiveness of the new teaching paradigm, and to make a decision as to whether this teaching methodology should be sustained in the long-term, and whether it should be applied to other courses in the undergraduate curriculum.

Regarding future development of the Web modules, the work primarily entails content creation. The key infrastructural work is complete and - with the exception of the creation of simulations - minimal software expertise is required for the content development. Further, instructor use of the site requires even less technical know-how; however, long-term maintenance of the site does require an individual with broad technical knowledge. In a sustained use of these Web modules, it will be necessary to have an individual dedicated to the technical aspects of the site, perhaps an individual who also maintains modules for other courses as well; the qualifications and time needed for this position cannot reasonably be expected to be fulfilled by the course teaching assistant. Another alternative to investigate in the future is the migration of these Web modules to an IS-supported machine; by doing so, many of the technical issues would no longer be the concern of the Mechanical Engineering Department.

Another important issue is whether to share the online portions of the course with other universities. Aside from the pragmatic issues of doing so, and the details of what form the sharing would take, this is essential to the constant growth and development of the project. It is acknowledged that the goal of this initiative is not to create a static product; the key is to develop a framework within which the course content can continually evolve. Limiting the Web modules to the MIT realm will hinder this evolution by providing an artificial environment which may result in biased feedback. Sharing this work with other schools will serve the dual purposes of providing useful feedback and helping MIT and the Department of Mechanical Engineering obtain recognition for their progressive efforts in engineering education.

This project has great potential to avoid the fate of past educational initiatives, which have typically faded very quickly. The main difference between these past initiatives and the current one is the complete redevelopment of the course structure. Rather than being an appended “gimmick”, the new teaching paradigm is highly integrated into the course material; the components of the new pedagogy define the class. Not only does this provide for a good chance of long-term survival, but it also serves as a model for meaningful curricular reform.

Appendix A

Desktop Experiments

Following are the handouts used for the desktop experiments in 2.001 between fall 2000 and fall 2001. The development of these handouts - and the experiments - was primarily the work of Farid Ganji, Pierce Hayward, Prof. Mary C. Boyce, Prof. L. Mahadevan, and Prof. Emanuel Sachs. Images have been included as appropriate to fully illustrate the individual experiments.

A.1 Equilibrium Desktop Exploration

Massachusetts Institute of Technology
Department of Mechanical Engineering
2.001 Mechanics and Materials I
Desktop Exploration
Forces, Moments, and Free-Body Diagrams

Objective

The purpose of this exercise is to review free body diagrams and the determination of reaction forces and moments.

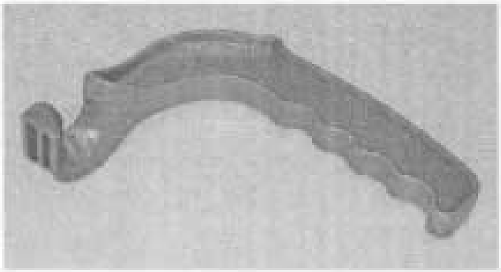
Apparatus

2-liter bottle of soda

"Handl-it" bottle handle

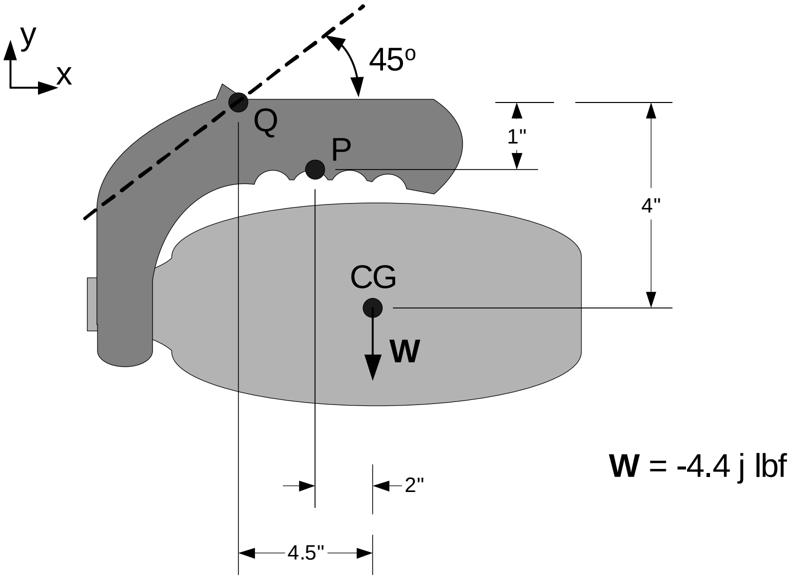
Procedure

Snap the handle onto the bottle and answer the questions on the following pages.



Questions

1. Consider the bottle-handle assembly in the horizontal position shown below. On the diagram, sketch the resultant forces that are applied to the assembly by your hand. Show them in the proper direction. Assume that the resultant forces occur at points P and Q labeled below, and that the line of action of the force at Q is as shown.

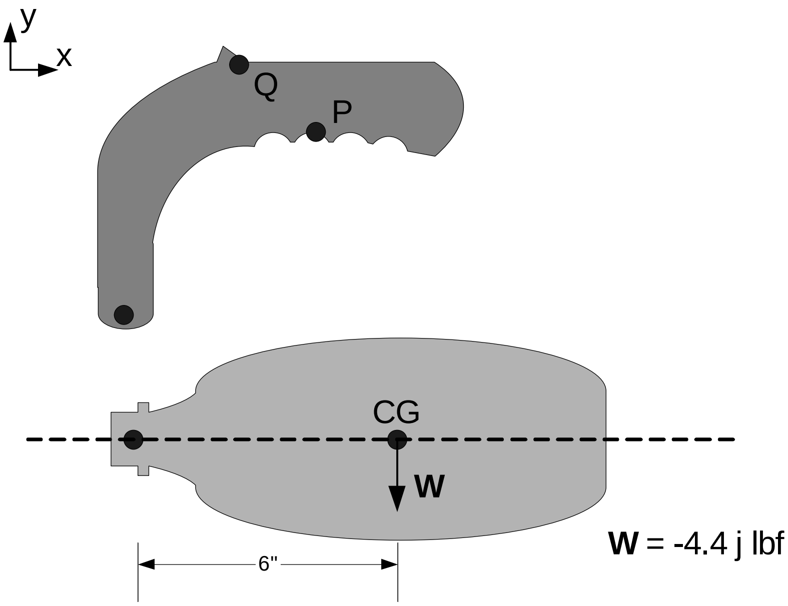


Pick up the assembly and hold the bottle in the horizontal position. Check your diagram based upon what you feel in your hand as you hold the bottle.

Check your diagram by making sure that you will be able to satisfy the equilibrium equations for the assembly. In other words, with the directions you have shown for the reaction forces, will you be able to satisfy $\Sigma F_x = 0$, $\Sigma F_y = 0$, and $\Sigma M_o = 0$?

Find the resultant forces at P and Q.

2. Below are incomplete free body diagrams of the handle and bottle in the horizontal position.

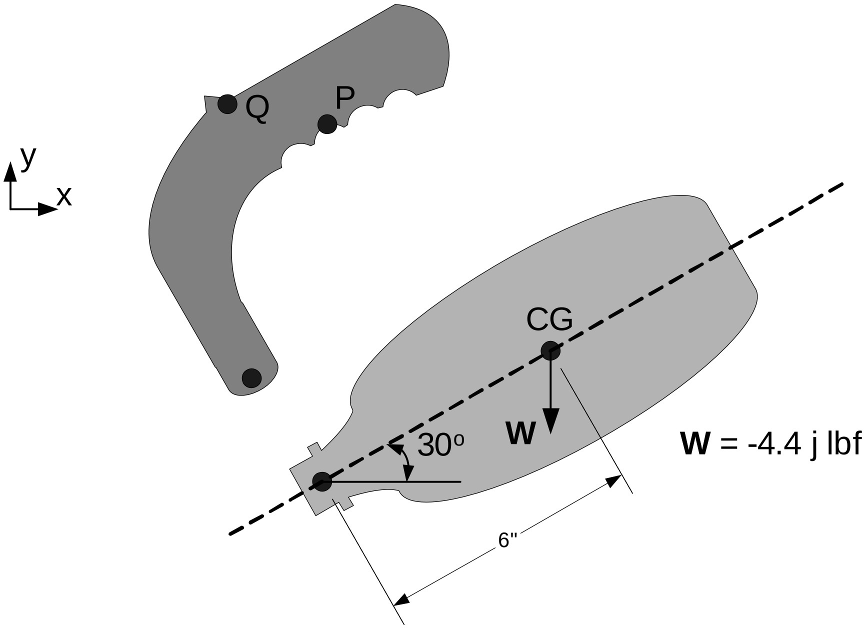


Sketch the resultant forces and moments (if they exist) on the handle and bottle. Show them in the proper orientation.

Check your diagrams by looking at the bodies as you hold the assembly in the horizontal position, and by making sure that you will be able to satisfy the equilibrium equations for each free body.

Find the reaction forces and moments on the bottle due to the handle.

3. Now consider the bottle in a "pouring" position. Below are incomplete free body diagrams of the bottle and handle, inclined at 30 degrees from horizontal. Sketch the resultant forces and moments (if they exist) on the handle and bottle. Show them in the proper orientation.

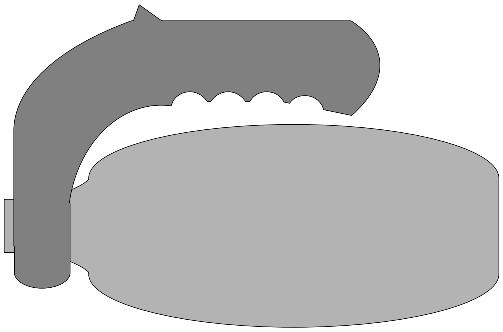


Check your diagrams by looking at the assembly while holding it in the pouring position and by making sure that you will be able to satisfy the equilibrium equations for each free body.

Find the reaction forces and moments on the bottle due to the handle.

4. Compare your results from questions 2 & 3. What do you notice about the reaction forces and moments in the two positions? In your own words, explain why this occurs.

5. If you have time, answer this question. Consider the assembly in the horizontal position once more. Without doing any calculations, can you devise a means of locating the center of gravity of the assembly in the plane shown below? Explain your method, using a free body diagram to justify your approach.



A.2 Friction Desktop Exploration

Massachusetts Institute of Technology

Department of Mechanical Engineering

2.001 Mechanics and Materials I

Desktop Exploration

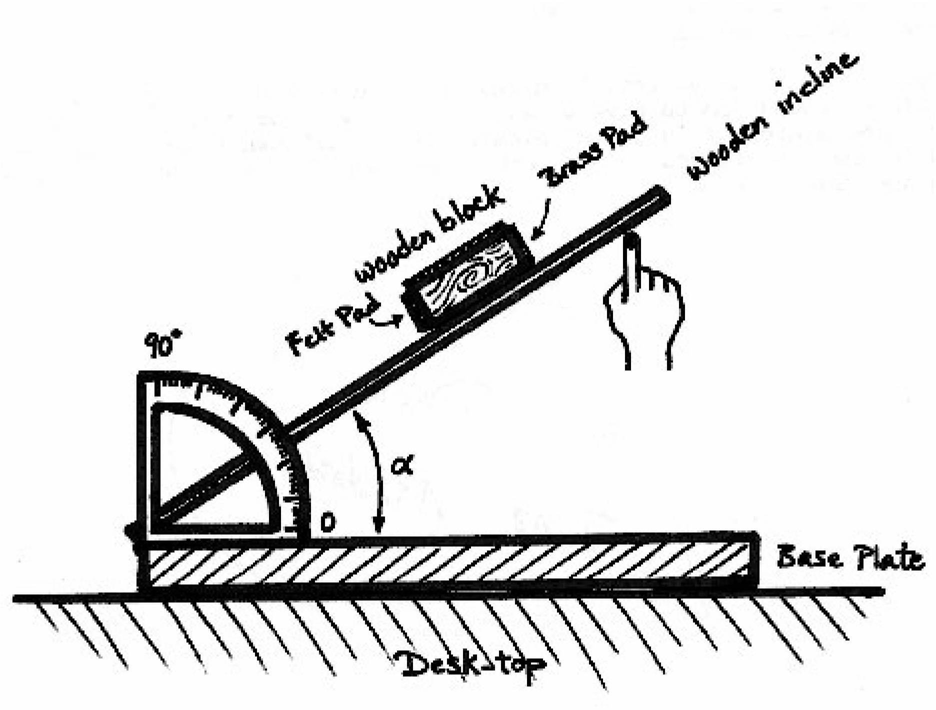
Friction

Experimental Setup

Purpose: Explore the properties of dry or Coulomb friction.

Assignment: Write a report based on your observations. The report should clearly address all of the posed questions.

The experimental setup is shown below. It consists of a rectangular wooden block and an inclined wooden plane of which the inclination angle can be changed manually in a continuous manner and measured.



Two sides of the block with different areas are covered with felt pads. Their respective, opposing sides are covered with brass sheets. The remaining two opposing sides are bare wood and only have different surface finish. The weight of the block may be referred to as W .

Preliminary Questions

P (a). What parameters do you expect the dry friction force between two solid bodies to depend on? Write your answer in the form of $F_f = F_f\ (p_1, p_2, \ldots , p_m)$.

P (b). Does the dry friction force need relative motion between the bodies in contact? Explain.

P (c). What quantities do you think affect the coefficient of friction, f?

Observation 1 *(Limiting value concept, static coefficient of friction)*

1 (a). Place the block on the horizontal surface at rest so that the larger felt side lies on the surface. Draw a Free Body Diagram. What is the magnitude of the friction force at this point (your answer should be in terms of W) ?

1 (b). Incline the surface slowly to an angle α so that the block is still at rest. Draw a Free Body Diagram. What is the magnitude of the friction force now (your answer should be in terms of W) ?

1 (c). Where at the interface, does the resultant surface normal reaction act? Explain.

1(d). Slowly increase the angle until the block is just about to slide down. Record the angle at which the block just begins to slide. Repeat at least 5 times and record the readings. Draw a Free Body Diagram. Calculate the static coefficient of friction, f.

Observation 2 *(Material dependence)*

Repeat Observation 1(d), but this time put the large brass side in contact with the surface.

2 . What is the static coefficient of friction in this case? Compare with that of the felt.

Observation 3 *(Area dependence)*

Repeat Observation 2, but put the smaller brass side in contact with the surface.

3 . What is the static coefficient of friction in this case? Compare with that of Observation 2.

A.3 Capstan Desktop Exploration

Massachusetts Institute of Technology

Department of Mechanical Engineering

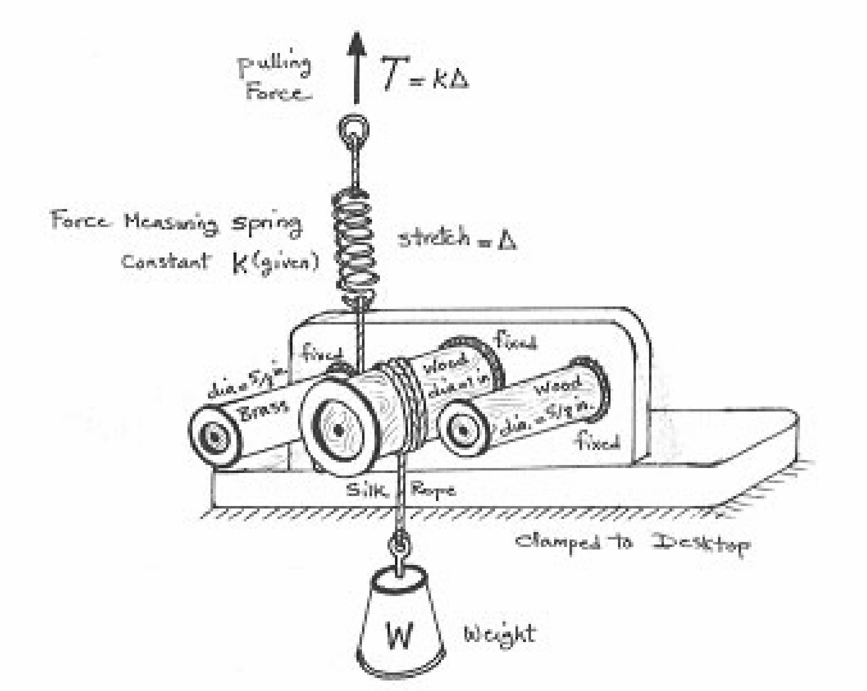
2.001 Mechanics and Materials I

Desktop Exploration

The "Capstan Effect"

Experimental Setup

The experimental unit as shown in the figure below consists of three posts supported by a wooden stand. The middle post is a wooden rod of 1 inch diameter. The other two posts have a diameter of 5/8 inch; one is brass-covered and the other is bare wood. The stand must be tightly fixed to the desk by c-clamps before conducting the experiment.



A rope with a seat for a weight W at one end and a spring with spring constant (K in lb/in; two springs are provided as described later); at its other end is also supplied. Weights are supplied in the form of steel washers. The rope will be sequentially wrapped around each post when conducting the experiments as described later below. The spring acts as the load cell: the spring extension Δ will be measured with the ruler (supplied) and the force will be obtained as $T = K\Delta$.

When conducting the experiments, care must be taken such that the rope turns do not overlap each other. Also each wrap should be adjacent (no space) to the prior wrap since the axial wrapping pitch is the rope thickness.

The following data is also needed:

$W_{hs} = W_{(hook + seat)} = 3.0$ oz

$W_{washers} = 3$oz (note that the weight of each washer is written on the washers)

Spring Stiffness:

Compliant Spring: $K_1 = .22$ lb/in for use when 1 lb < T < 3.0 lb.

Stiff Spring: $K_2 = 4.5$ lb/in for use when 5 lb < T < 30 lb.

Preliminary Question

What quantities do you expect to affect the pulling force? Write your answer in the form of $T = f(q_1, q_2, q_3,\ldots)$.

Observations

1. Place a washer weight (W_{washer} = 3 oz; $W_{total} = W_{hs} + W_{washer}$) on the rope. Wrap the rope with the weight one turn ($\theta = 2\pi$ radians) around the larger diameter wooden post so that the opposing end of the rope is pointing upward. For ease of measurement, keep the upper portion of the rope between the bar and the spring as short as possible (about 2 in). Now, try to raise the weight by pulling the handle, hooked to the spring. At the onset of motion of the weight, measure and write down the spring stretch, Δ, with the ruler provided. (You will need to work with a partner on this).

Repeat the above measurement for 2, 3, 3.5 and more turns until you are almost unable to get the rope to slip around the post. Each time record your measurements.

Calculate the spring force, T, for each θ and tabulate the data, including the force $T = T_0 = W$ corresponding to $\theta = 0$.

2. Repeat the steps of observation 1 for the lower diameter wooden post.

3. Repeat the steps of observation 1 for the brass-covered post.

4. Repeat the steps of observation 1 (use large diameter wood post), but now use a larger weight.

Analyses

I. Using your data from observation 1 (large diameter wood post), plot T vs. θ. Is the relationship linear? How would you describe the relationship?

II. Using your tabulated results for the wooden posts of different diameter, i.e. observations 1 and 2, plot T-versus-θ for the two cases on one plot and compare. What is your conclusion?

III. Using your data from observations 1 and 4, plot T vs. θ for each case on the same plot. Discuss.

IV. Using your data from observations 2 and 3, plot T vs. θ for the two cases on the same plot. Discuss.

V. Using your data from observations 1, 2, 3 and 4 plot ln T vs. θ (where ln is natural logarithm). What do you observe? What does this indicate about the relationship between T and θ?

A.4 Spring Constants Desktop Exploration

Massachusetts Institute of Technology

Department of Mechanical Engineering

2.001 Mechanics and Materials I

Desktop Exploration

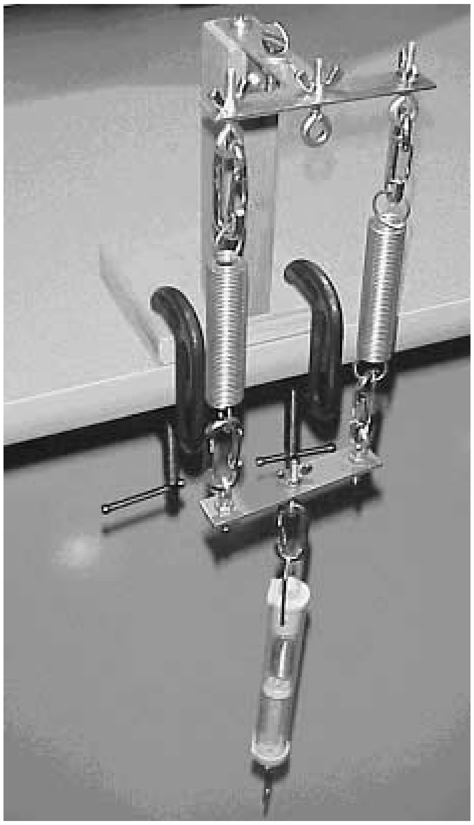
Force-Deformation: Spring Constants

Objective

Explore the relationship between force and deformation for various spring assemblies.

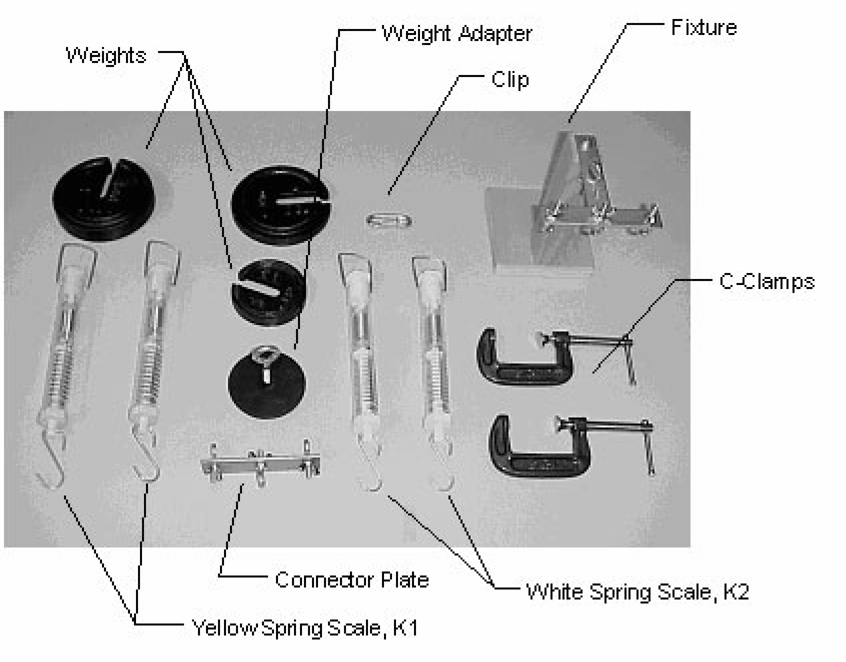
Experimental Setup

The experimental unit is shown in the Figure below. The unit consists of a fixture with three hooks; various spring assemblies will be attached to these hooks and then loaded by dead weights.



Supplies

- Fixture, C-clamps, connector plate, weight adapter, clips
- Springs
  - Spring 1:Calibrated Force Measuring Spring Scale (white)
    - K1 = 590 N/m (= 3.36 lb/in)
  - Spring 2:Calibrated Force Measuring Spring Scale (yellow)
    - K2 = 977 N/m (= 5.58 lb/in)
- Weights:0.5 lb (= 2.23 N), 1.0 lb (= 4.45 N), 2.0 lb (= 8.9 N)
- Ruler



Preliminary Question

What is the relationship between force and displacement of a linear spring system?

Hint for the experiment

In order to account for the fact, that all of the components used in this experiment are subject to gravity, and therefore cause displacements in the spring scales, even though no weights are applied, it is recommended to do a "zero" before applying the weights. This zeroing is achieved by turning the adjustment screw at the top of each spring scale until the indicator rests at the zero line (before applying the weights!).

Observation 1

Verify the spring constant of the yellow spring scale by successively loading it with weights and recording the resulting displacement. Is the yellow spring scale linear?

Weight [lb]	Displacement Δx	Stiffness K3
0		
0.5		
1		
2		
3		

Observation 2

Repeat the experiment described in observation 1 with the white spring scale. Is the white spring scale linear?

Weight [lb]	Displacement Δx	Stiffness K4
0		
0.5		
1		
2		
3		

Observation 3

Springs in series. Put two white spring scales in series (one after another). Load it successively with weight and record the total change in length as well as the change in length of the individual springs. What is the spring constant K5 of the system and how does it compare to a single spring scale? Draw a free body diagram, isolate the individual springs, and set up a relation between the applied load and the system (= total) displacement.

Weight [lb]	System length	Length Δx	Length $\Delta x1$	Length $\Delta x2$	Stiffness K5
0					
0.5					
1					
2					
3					

Observation 4

Springs in parallel. Using the clips provided, attach two white spring scales to the outer loops of the fixture and attach the connector plate at the bottom, again using the outer loops. Attach the weight adapter to the center of the connector plate and increase successively the loading while recording the system displacement as well as the displacements of the individual springs. What is the spring constant K6 of this system and how does is compare to a single spring scale? Draw a free body diagram and set up a relation between the applied load and the system (= total) displacement.

Weight [lb]	System length	Length Δx	Length $\Delta x1$	Length $\Delta x2$	Stiffness K6
0					

Weight [lb]	System length	Length Δx	Length Δx1	Length Δx2	Stiffness K6
0.5					
1					
2					
3					

Observation 5 *(optional, if there is enough time left)*

Springs in parallel and series. Remove the weight adapter from the setup of observation 4 and replace it with a yellow spring scale. Attach the weight adapter to the bottom of the yellow spring scale, apply the loading and record the changes in total length as well as the change in length of the individual springs. What is the spring constant K7 of the system and how does is compare to the individual spring scales? Draw a free body diagram and set up a relation between the applied load and the system (= total) displacement.

Weight [lb]	System length	Length Δx	Length Δx1	Length Δx2	Length Δx3	Stiffness K7
0						
0.5						
1						
2						
3						

A.5 Truss Analysis Desktop Exploration

Massachusetts Institute of Technology

Department of Mechanical Engineering

2.001 Mechanics and Materials I

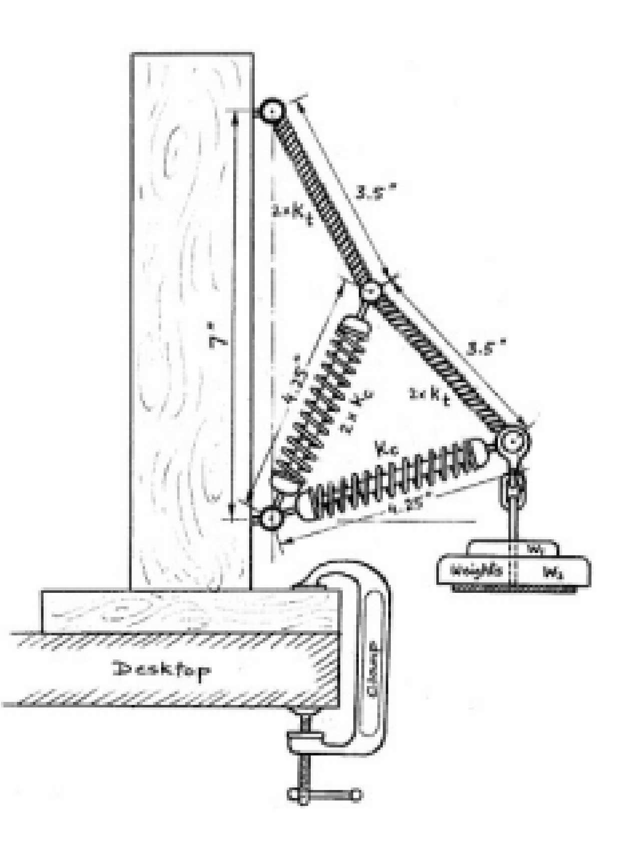
Desktop Exploration

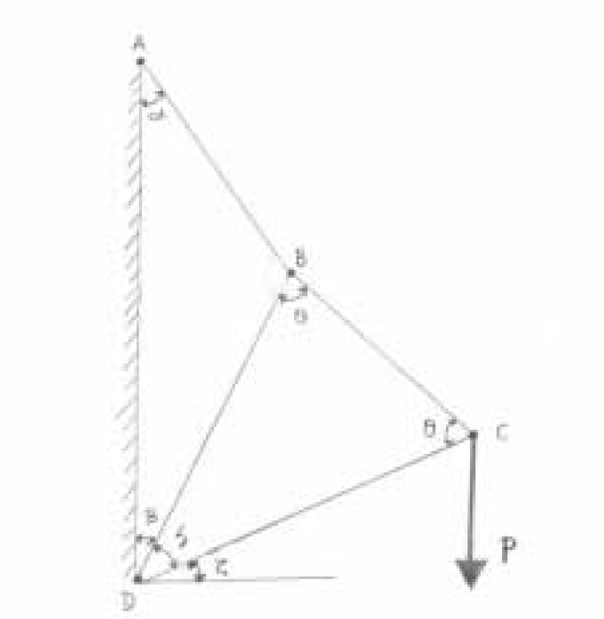
Truss Analysis

Experimental Setup

A truss structure has been constructed as shown in the figure below. Coil springs constitute the truss members. The springs have the spring constants listed below.

- K_c = 8lb/in
- K_t = see attached plot





Note the following:

α = 28.19101434º

β = 22.89512844º

δ = 48.63147834º

ζ = 18.47339322º

θ = 65.68426083º

Step 1

Assume that the truss elements are actually single steel rods of diameter 5 mm (.2 inches). The Young’s modulus of steel 200 GPa (30,000 ksi).

- draw the free body diagrams.
- determine the force in each truss member.
- determine the axial deformation of each member.
- which members are in tension? compression?
- derive an expression for the displacement of pin B using compatibility

Comment

Notice that the deflection would be too small to measure in this setup. This is why springs will now be used as an alternative in order to directly visualize the deformations in the truss members and the deflection of the truss structure. You will see, however, that this has additional consequences.

Step 2

Place weights of 1, 1.5, 2, 2.5 and 3 pounds in the cradle. In each loading situation:

- measure the vertical deflection;
- for the weights of 1lb, 2lb, and 3lb, measure the lengths of the members and the angles between them.

- which members are in tension? compression?
- remove the weights; what happens to the structure? is the structure elastic?
- now, plot the load versus deflection. Should it be a straight line? Is it a straight line? If not, why not?

Step 3

Now, let's explore this a little further.

Check if the geometric compatibility equations you derived in step 1 are satisfied for the 2lb configuration. Quantify the discrepancy between the vertical deflection calculation assuming small displacements vs. that actually measured.

Step 4

Using Castigliano's theorem, determine an expression for the vertical displacement of point C. Under what conditions does this hold?

A.6 Biaxial Loading Desktop Exploration

Massachusetts Institute of Technology

Department of Mechanical Engineering

2.001 Mechanics and Materials I

Desktop Exploration

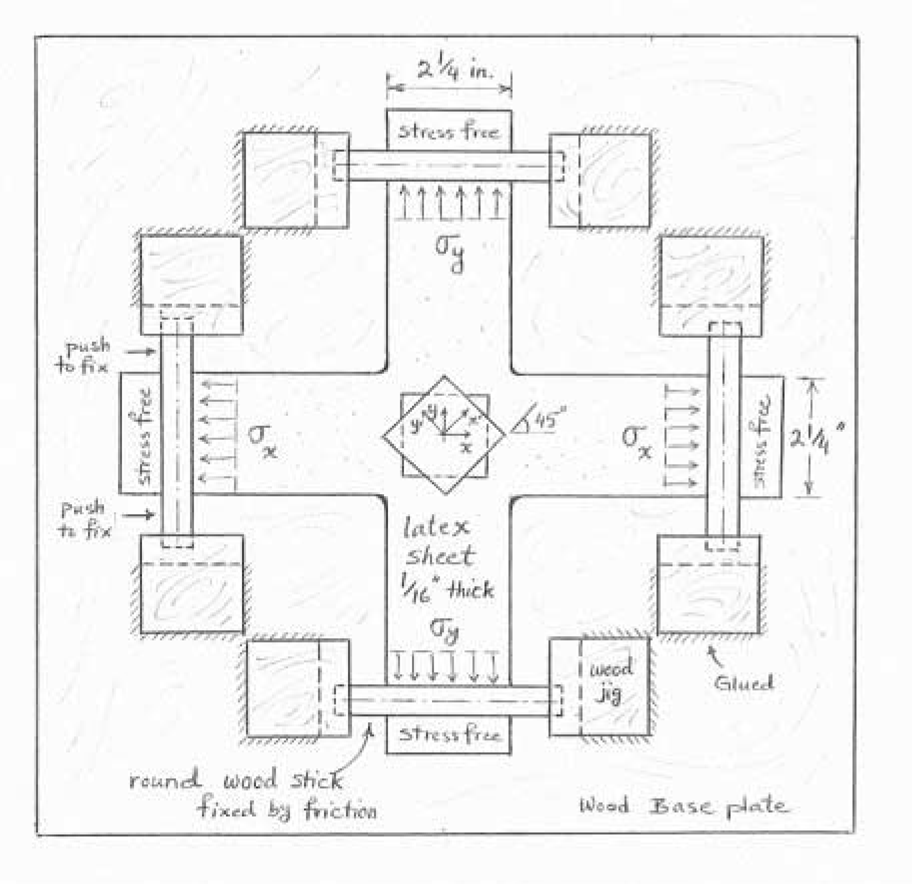
Biaxial Loading

This module qualitatively and quantitatively explores biaxial loading and deformation of materials.

Experimental Setup

In order to easily observe the strain in the material, we load a latex sheet in our experiment. The sheet has been cut in the shape of a plus sign as shown in the picture below, and can be loaded by hand. Loads can be easily locked in the wedge clamps.

The sheet has a rectangle and a diamond with 90-degree internal angles inscribed in the middle. We load the sheet and observe the change in shape of these figures.



Experimental equipment: biaxial stretching device, calipers, protractor.

Material: cruciform specimen of latex sheet with inscribed rectangle and diamond.

Laboratory Tasks

1. Measure the dimensions of the rectangle and diamond in an unloaded state. In particular, measure the angles. They should all be 90 degrees.

2. Load the latex sheet a small amount in one direction. Lock in the load.

3. Measure the new dimensions of the rectangle. In particular, measure the new internal

angles of the rectangle and the diamond. Have they changed?

Questions

a) What are the axial strains as measured in the rectangle?

b) What is the Poisson’s Ratio for this material?

c) Is there a shear strain in the rectangle? How much?

d) Is there a shear strain in the diamond? How much?

e) Using transformation equations (or Mohr's Circle), predict the diamond state of strain from that of the rectangle. How does your prediction for the diamond shear strain compare to your measurement?

4. Now load the strip in the second direction while keeping the loading in first direction in place. Use a small load.

5. Measure the new dimensions of the rectangle and the diamond. In particular, measure the new internal angles of the rectangle and the diamond.

Questions

a) What are the axial strains as measured in the rectangle?

b) What is the shear strain in the rectangle?

c) What is the shear strain in the diamond?

d) Using transformation equations (or Mohr's Circle), predict the diamond state of strain from that of the rectangle. How does your prediction for the diamond shear strain compare to your measurement?

6. Increase the load in the second direction until the diamond has once again 90-degree angles (although it is larger!).

Questions

Repeat the questions from step 5.

A.7 Beam Strain Desktop Exploration

Massachusetts Institute of Technology

Department of Mechanical Engineering

2.001 Mechanics and Materials I

Desktop Exploration

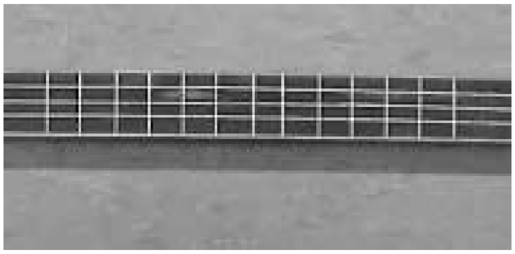
Beam Bending: Strain Distribution

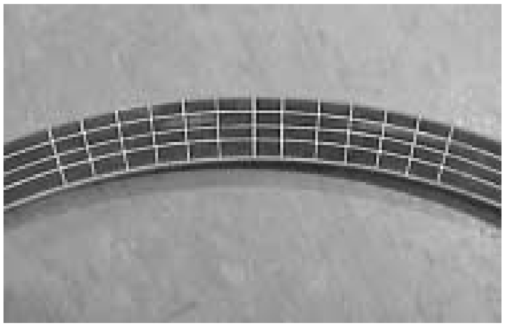
Objective

Explore strain distribution on beams subjected to bending.

Supplies

- 3-foot long 1"x1" polyurethane beam with rectangular grid
- measuring tape
- paper with various semi-circles
- clamps





Observation 1 *(Pure Bending)*

1. Take the beam, measure the initial length between two specific grid marks (use a spacing of about 6-10 inches).

2. Take the beam, bend it into the radius of curvature R_1 = 24 inches. Measure the length between the grid marks chosen in step 1 at various positions through the beam (i.e. at top surface, at center, at bottom).

3. Determine the strain distribution through the thickness of the beam.

4. Repeat steps 2 and 3 for R_2 = 18 inches and R_3 = 12 inches.

5. Determine an expression for the axial strain through the thickness ($\varepsilon_{xx}(y)$) as a function of radius of curvature.

Observation 2 *(Cantilevered Beam with End Displacement)*

1. Take the beam, measure the initial distance between several pairs of grid marks along the top surface of the length of the beam.

2. Clamp the beam in a cantilevered manner. Subject the beam to an end displacement of 3 inches.

3. Measure the strain distribution in the beam (take measurements at several points along the length and through the thickness).

4. Plot the bending moment diagram.

A.8 Beam Deflection Desktop Exploration

Massachusetts Institute of Technology

Department of Mechanical Engineering

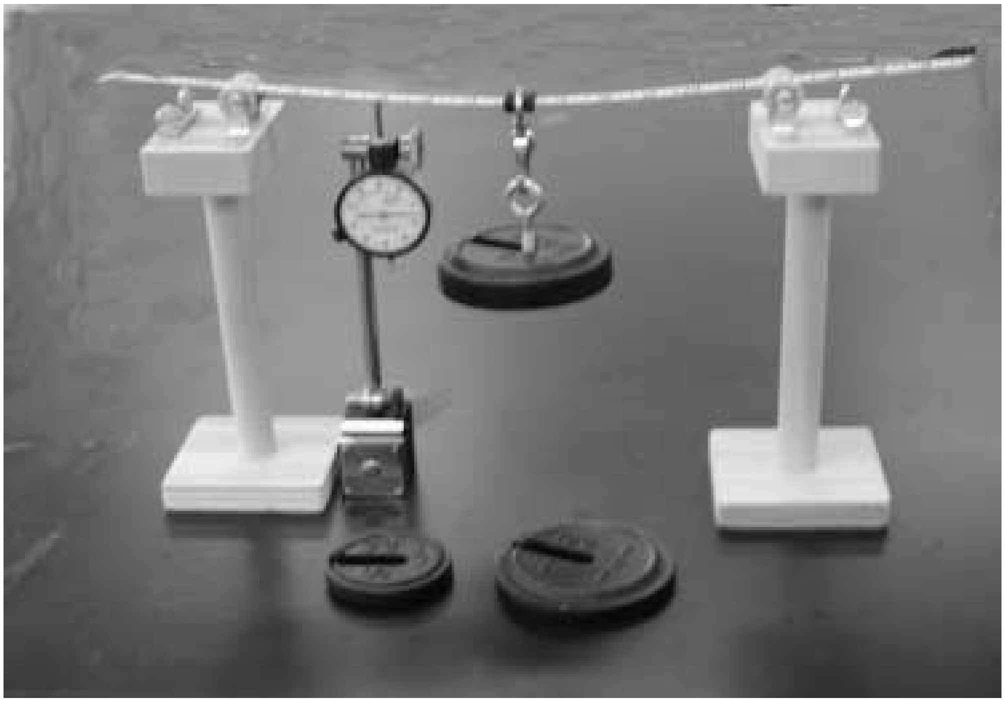
2.001 Mechanics and Materials I

Desktop Exploration

Beam Bending: Deflections

Experiment

A simply supported beam will be subjected to a central load, P (3-point bending). The lateral displacement, δ, at the center of the beam will be measured. The beam stiffness will be assessed in terms of the slope of the P-δ curve. You will explore the dependence of stiffness on various parameters including length, thickness, and load.



Supplies

- 3 20-inch long strips of spring steel t_1 = .025 inch (=0.635mm), t_2 = .05 inch (= 1.27mm), t_3 = .062 inch (= 1.575mm)
- Bending apparatus
- Weights
- Ruler

Observation 1 *(Force-Displacement Behavior)*

- Place the 20 inch long, .05 inch (1.27mm) thick, .5 inch (12.7mm) wide strip of spring steel on the rollers with the rollers placed 10 inches (254mm) apart.
- Place the hook at the center point of the beam.
- Place the 1lb (4.44N) weight on the hook, measure the deflection.
- Place the 1.5lb (6.66N) on the hook, measure the deflection.
- Plot P vs. δ, determine the stiffness for this loading condition.

*the platform weight is 3oz (0.12N).

Observation 2 *(Dependence on length)*

Using the .05 inch (1.27mm) thickness beam:

1) Place the rollers 10 inches (254mm) apart.

a) Place 1lb (4.44N) in center, measure the deflection.

2) Place the rollers 15 inches (381mm) apart.

a) Place 1lb (4.44N) in center, measure the deflection.

3) Determine the stiffness for these two cases. How does stiffness scale with length?

Observation 3

For the case of L = 12 inches (304.8mm) (rollers placed 12 inches apart). Measure the stiffness (using 1.5lb (6.66N) weight) for spring steel with:

- t_1 = .05 inches (1.27mm)
- t_2 = .062 inches (1.575mm)

How does stiffness scale with thickness?

Appendix B

MIT Information Systems Usability Report

Usability Report
MIT MechE I-Campus Project - Beam Bending Module
January 2001

The purpose of the project

This project's goal is to move away from the traditional large lecture format for first year Mechanical Engineering subjects towards an active learning model via simulations and modules. This methodology brings the laboratory into the classroom, using the "observe, study, experiment" model. Each class would become a session involving laboratory, mini-lecture, Web-based learning modules, and discussion.

The module topics are:

- Friction
- Capstan
- Truss
- Torsion
- Bending

The goal of the usability testing of the project

Usability testing helps insure that the final product is easy to use and effectively helps the users attain their goals which in this case would be the learning of mechanical engineering principles and techniques. Planning for the usability testing began with initial meeting between Jean Foster, a Usability Engineer with the MIT IS sponsored Usability Team and the two of the project developers, Ebbe Bamberg and Jaspal Sandhu. Jean recommended that testing be done on each module as work progressed. The first module ready for testing was the module on the Beam Bending topic. This report describes the tests, presents the results and suggests way to improve the usability of the module.

Description of the Tests

On November 7th the Usability Team conducted a heuristic review of the 2.001 Web site with the developers present. We also asked Debby Levinson from the Web Communication Services (WCS) group to have her team and designers from the Publishing Bureau review the site for it's graphical look. Debby delivered her report at the meeting on the 7th. The developers took their own notes and came away with several problems to fix and changes to make in the graphical design of the site.

Jean Foster then developed a usability testing plan for the Beam Bending module (see http://web.mit.edu/is/usability/MechE/) which would involve student test participants using a thinking aloud protocol. In this test method a test participant is given test tasks or problems and then and then is asked to think out loud while they are solving the tasks. In this case the participants were presented with a sample assignment scenario to solve (see. Attachment A) The participants are observed by one or more usability team members who take notes on what they observe and the tests are video taped for later review.

The tests were conducted during IAP 2001 using students who had just completed course 2.001 the previous (Fall 2000) semester. It was important that the students had already been through the classroom lectures on the beam-bending topic. Jaspal Sandhu recruited the student testers and planned the test schedule. Seven students were recruited for 4 tests with 3 of the tests scheduled for students to work in pairs. Two of the students did not show up but in both cases they were scheduled to test with another student and so we went ahead and tested with one. Two pilot tests were conducted with test participants who were not students.

The tests took place in the Usability Team's testing laboratory in building N42. The lab consists of a test room (which also is used for integration testing) and an adjoining room which contains two monitors, one connected to a video camera aimed at the test participant, and another which is hooked up to the workstation that the test participant is testing on. This monitor displays the test participant's screen as she sees it so that the observers can follow exactly what the participant is doing in the application.

The student test participants tested on a PC laptop in the Integration Lab. The laptop was running Windows 2000. Some test participants used Netscape 4.76, and others used IE5.5. One pilot tester used Netscape under Linux.

Test results

Summary

The students were able to solve the problem in the test scenario. Most test participants seemed to like the beam bending applet, several using the adjective "cool". Generally they seemed to understand the layout of the 2.001 site with some exceptions which are noted in the details below.

Note: The recommendations below are only recommendations and are not necessarily the only possible solution for the problems found.

About the site in general:

One user said he liked the graphical bend in the word "bending".

Several users commented on the length of time it took for the applets to load. "Oh, it's a big applet or the network is slow".

Finding 1: Confusion about top and bottom navigation bars

Observation:

Several of the users had to scroll back and forth comparing the navigation bars at the top and bottom before they were sure that they were duplicate information.

Recommendation:

Use the same image map navigation bar for the bottom of the page as for the top.

Finding 2: Expectation of "where am I" component in Navigation bar

Observation:

Participants expected that the pull-down section of the top navigation bar would reflect where they are in the site.

Recommendation:

Highlight the word of the section that the user is in. E.g. if they are on the examples page examples should somehow be highlighted in the list under bending

Finding 3: Users were confused about usage of the terms simulation/applet

Observation:

The participants were looking for the word applet (from the scenario) and didn't realize that the simulation was the applet

Recommendation:

Be consistent. Either use the term simulation applet or just applet or just simulation.

Finding 4: There is no way to search for particular topics within the module.

Observation:

Participants went from one section of the module to another looking for particular explanations or topics. One participant said he wished there were a search feature.

Recommendations:

Add a search feature to each module and to the site on the home page.

2.001 Home Page

Finding 1: The word disclaimer is misleading.

Observation:

At least one test participant started reading the disclaimer and stopped when he saw that it was really a recommendation for screen resolution. He never continued to read the Java version information.

Recommendation:

The Java version information should be labeled as a Warning and should come first.

MAC users should be warned that there is no JRE 1.3 available (yet?) for MAC.

The screen resolution recommendation should be labeled as Recommendation and should come after the warning.

Real Life Examples

Finding 1: Photos too large.

Observation:

Users complained that the photos were too big. They had to scroll extensively. Users also had to scroll to get to the links under the picture.

Recommendation:

Make the photos smaller.

Move the links below the photos closer to the photos or place above the photos.

Finding 2: Not all concepts were shown in the Examples?

Observation:

One user commented that there was no example for the problem they were given.

Recommendation:

If this is accurate, more examples should be created.

Beam Bending Applet

Users seemed to feel that the applet saved them time.

One student used equations link and was glad she didn't have to calculate it herself.

Finding 1: Help menu problems

Observation:

Some users didn't notice the help menu.

Recommendation:

Make the Help menu more noticeable, larger, bolder?

Observation:

Users opened Help but didn't find what they were looking for. Some were looking for an explanation of the icons. Some went for instructions for how to use the tool.

Recommendation:

Add more info to the Help.

Chunk the information; use headings; embolden keywords; provide numbered steps about how to use the tool.

Finding 2: Users had trouble figuring out how to use a material that wasn't listed in the materials pull-down menu

Observation:

Some users didn't notice the custom option in the materials pull-down menu when they were searching for a material that wasn't listed.

Recommendation:

Put Custom first on the list and separate it slightly from the rest of the list.

Finding 3: Confusion about simulation/applet wording

Observation:

Some users were confused when told to use the applet but only saw a link for simulation.

Recommendation:

Be consistent in naming the tool. Simulation is more generic. Do they need to know it is an applet?

Finding 4: Calculator needed to figure weight

Observation:

Several users expected a calculator to be included in the tool.

Recommendation:

Provide a calculator or instruct users to use a system calculator in the Help instructions.

Finding 5: Users expected graph units would change when they changed the unit of measurement for the beam

Observation:

Users commented that the graphs didn't recalculate properly when the unit of measure on bottom left was changed from feet to inches.

Recommendation:

Don't know if this is possible to fix?

Finding 6: Graph doesn't appear to update

Observation:

Several users thought that the tool was buggy when they made a change to either the material, boundary conditions, or measurements and the graph seemed to stay the same. (In retrospect it is possible that the graph numbers were changing but not the graphics, but users didn't notice the numbers changing.

Recommendation:

Have the graph "flash" when updating.

Finding 7: icon definitions weren't found

Observation:

Many users never noticed the rollover explanations of the icons.

Recommendation:

Adjust the timing of the rollovers so that they show up immediately.

Give the rollovers a bright color background (such as yellow or red) so they will be more noticeable.

Finding 8: Users didn't notice the unit change option immediately

Observation:

Most users didn't notice the unit change options until they had filled in the materials, measurements, and loads sections.

Recommendation:

Move this option into the Dimension section.

Finding 9: confusion about using the calculate button

Observation

Users did not always push the Calculate button when it was needed and seemed to be waiting for updates to happen automatically.

Recommendation:

Label the button, e.g. "Press after changing load specifications".

Finding 10: Users weren't sure how to use Loads section

Observation:

Users seemed to be unsure of what to do in the Loads section. They kept getting "out of bounds" error messages. Lots of guessing.

Recommendation:

Include specific instructions and description of how this works in Help.

Finding 11: dimensions failed to recalculate when unit system was changed

Observation:

Users expected that when unit system was changed, for example from feet to inches, that the dimensions would change automatically from 1 foot to 12 inches.

Recommendation:

Make the dimensions recalculate automatically when the unit system is changed.

Finding 12: dimensions failed to recalculate when unit system was changed

Observation:

Users expected that when unit system was changed, for example from feet to inches, that the dimensions would change automatically from 1 foot to 12 inches.

Recommendation:

Make the dimensions recalculate automatically when the unit system is changed.

Finding 13: Users weren't sure whether to single click or double click in the tool

Observation:

Users sometimes double clicked

Recommendation:

Explain that single clicks are required.

Finding 14: users looked for Young's modulus for the materials

Observation:

Users looked for Young's modulus for the materials listed.

Recommendation:

Provide Young's modulus for materials.

Finding 15: most users did not use comparison feature

Observation:

Some users didn't try to use this feature and others tried but stopped when it didn't seem to be working.

Recommendation:

Put the word compare somewhere between beam 1 and beam 2.

Finding 16: Controls for beam 2 weren't obvious

Observation:

Some users didn't understand that F2 and Q2 corresponded to Beam 2.

Recommendation:

Color-code these labels. Make F1 and Q1 blue and F2 and Q2 red.

Finding 17: Users expected the tool to include a way to calculate weight

Observation:

Users commented that it would be nice if the was an easy way to calculate the weight within the tool

Recommendation:

Include this feature.

Finding 18: Users were unsure of the meaning of some of the terms

Observation:

Some users didn't know what lbf stood for.

Recommendation:

Include a glossary as part of the reference materials.

Finding 19: HEIGHT fill-in needs zero in front of DECIMAL

Observation:

Once user repeatedly tried to enter a number < 1 in the height fill-in but it kept failing until he put a zero in front of the decimal. Only saw this once.

Recommendation:

Fix so it isn't necessary to put a zero in front of the decimal.

Finding 20: BEAM 2 covered beam 1 in chart

Observation:

Users sometimes couldn't tell if Beam 1 was still showing up in the graph because the lines were completely covered by the red Beam 2 lines.

Recommendation:

Make Beam 1 lines a bit wider so they will show up behind the Beam 2 lines.

Miscellaneous

Finding 1: There was no references for standards

Observation:

Users were unsure of what would be a reasonable weight to use for load; width of bridge

Recommendation:

Include reference information for standards (if such exist).

Accessibility Problems

Add ALT attributes to navigation image map hotspots on all pages. See WAI Techniques document: http://www.w3.org/TR/WCAG10-HTML-TECHS/#client-side-text-equivs

Use Java Swing accessibility features for applets. E.g., allow for keyboard navigation in applets. See the IBM Java checklist: http://www-3.ibm.com/able/javakeyboard.html

Examples: use ALT attributes or LONGDESC element to describe the pictures.

Possible Bugs

Custom dialog box didn't always refresh.

When entering height in the dialog box, it kept failing until he put a zero in front of the decimal point. There was no error message or explanation.

Beam 2 graph didn't clear (in one test).

Graph didn't recalculate when the boundary conditions were changed.

Possible problem with Linux: "Scrolling over large graphics caused a different Web page to display." (jlittell)

Using IE 5.5 under Windows 98, I couldn't get the examples to load. The Javascript seemed to hang. (jfoster)

In the Examples sections, the schematic button seemed buggy. Users had to push the button twice for it to respond.

IE 5.5 crashed once.

Applet couldn't be used on a laptop with a screen resolution of 800 X 600. Calculate button was out of sight.

User comments

On first seeing the applet: "Wow, there's a lot of stuff here. I have no idea what to do. I'd better read the instructions."

"I wish there was a way to express how to apply the load …to indicate Q1 on the load graph."

"It would be nice if they showed the same input (Young's modulus?) (so you would) know if you were comparing the same units."

"It would be nice to see X coordinate numbers to see where the bending moment is crossing the zero in the Y direction."

"I like the force line down from the force arrow."

I assumed you were applying the same load to both."

Inertia table: "It would be helpful if the words at the top of the table stayed in sight when you scroll down".

When user got an error "Surface load - left location cannot exceed right location" and commented, " I'm not sure what that means."

I wish the Q load diagram were clearer."

"It would be handy to have conversion factors on the site."

"Oh, you can try two different beams to compare. That's pretty cool!"

"I don't like that it doesn't show the weight of the beam"

"I wish it would calculate automatically when I hot enter." (Instead of having to push the Calculate button.)

In Help: " I don't want to read it but it doesn't look like there is much info there."

"I would like there to be fewer words and more pictures (on the site)."

Appendix C

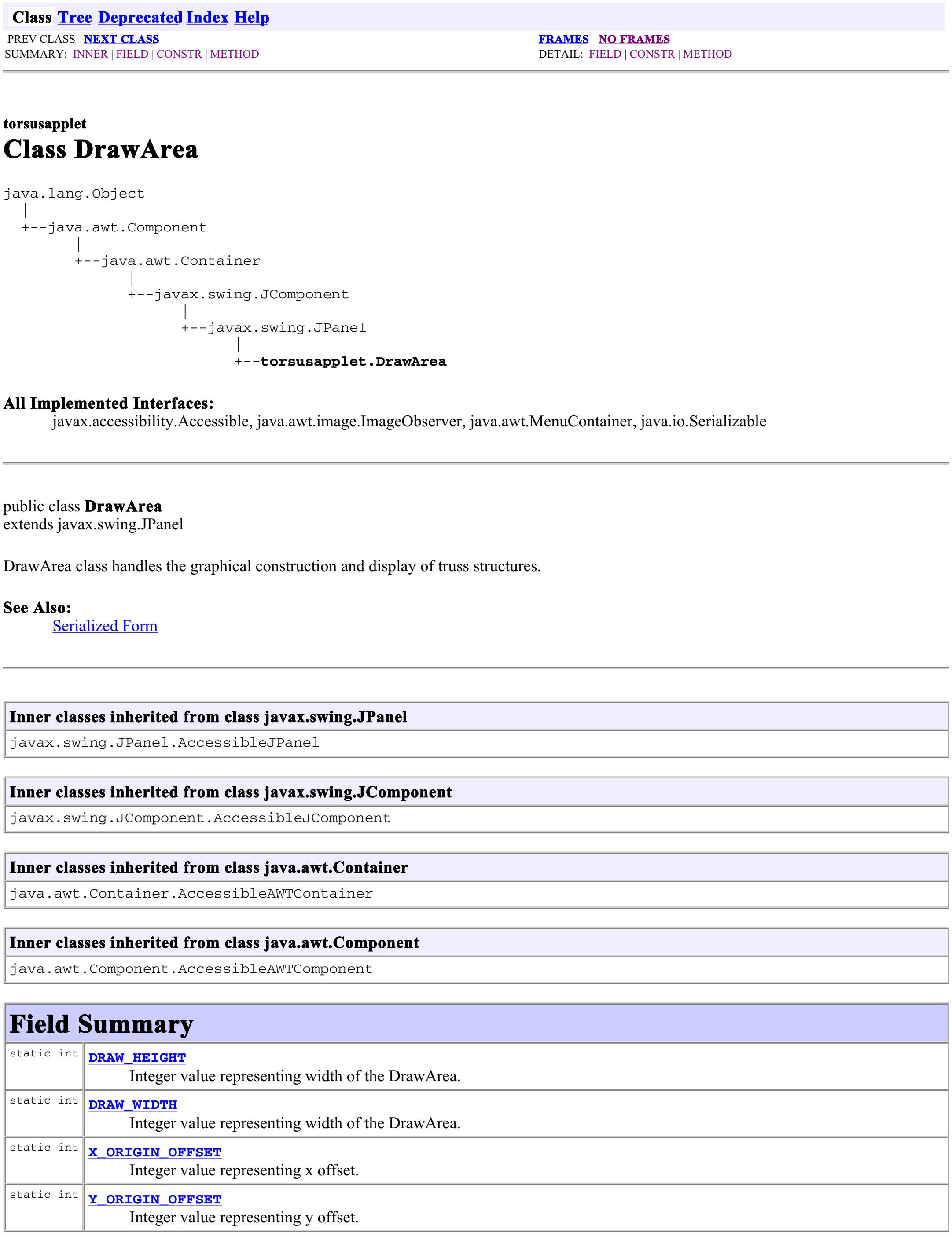
Torsus Technical Details

C.1 Revision History

Revision	Date	Notes
1.0	16 Aug 01	Core functionality: allows basic construction and analysis of planar truss structures.
1.0a	14 Oct 01	Calculates and prints mass of truss structure.
1.0b	18 Oct 01	Text-based modification of truss-structure allows more precise construction of structures.
1.0c	22 Oct 01	Member stresses are calculated and printed. Drawing area is larger. Scale is given in terms of grid blocks, more intuitive than pixels; changing the grid spacing (small, medium, large) updates the scale factor automatically. Mouse position indicator gives x and y distance from (0,0) in inches or meters (not pixels). Modifying the structure using the dialog (right-clicking) is now in terms of inches or meters, as opposed to pixels.
1.0d	25 Oct 01	Major grid lines (every 5 grid blocks) are indicated by a darker color for ease in drawing. The origin (0,0) has been shifted (up and right) to allow user to reference it more effectively. The modify structure function allows user to right click on a node once to bring up the text-based dialog; the left click functionality remains the same. The output in the message window has been reformatted such that it is easier to read.
1.0e	28 Oct 01	Buckling calculation (based on solid circular cross-section) included in printed output.
1.0f	2 Nov 01	Unstable determinate structures identified by limiting values of matrix elements of inverse of global stiffness matrix; limitations also set on scale and area values. Maximum stress printed. MIT MechE iCampus logo included.

Revision	Date	Notes
1.0g	21 Nov 01	Ability to undo last change to structure (TrussStructure, Member, Node all implement Serializable; ObjectCloner class performs deep copy operation). Now catches unstable determinate structures by looking for non-invertible matrices (since JAMA package does not throw an exception when inverting non-invertible matrices, program determines if rank = dimension of matrix). Unnecessary code removed; code commented for purposes of javadoc.
1.1	07 Jan 02	Ability to run as application or applet from same code; to run application, simply double-click jar file (should work by default on Windows machines; if not, it may be necessary to set the program associated with JAR extension manually). Length of truss members printed (for purposes of truss construction in 2.973). Application capabilities (do not work in applet version): quit, save truss (*.trs), load truss (*.trs), copy and paste (by highlighting Message Window), copy data function (under edit menu). Also (in both applet and application), program now allows choice of solid circular and custom moment of inertia for buckling analysis.
1.1a	18 Jan 02	Minor changes to structure modification input, system output in terms of significant figures (more precise). Buckling analysis also computes minimum moment of inertia necessary to prevent buckling in individual members.

C.2 Javadoc



Class **Tree** **Deprecated** **Index** **Help**

PREV CLASS **NEXT CLASS** | **FRAMES** **NO FRAMES**
SUMMARY: INNER | FIELD | CONSTR | METHOD | DETAIL: FIELD | CONSTR | METHOD

torsusapplet

Class DrawArea

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--javax.swing.JComponent
                    |
                    +--javax.swing.JPanel
                          |
                          +--torsusapplet.DrawArea
```

All Implemented Interfaces:
javax.accessibility.Accessible, java.awt.image.ImageObserver, java.awt.MenuContainer, java.io.Serializable

public class **DrawArea**
extends javax.swing.JPanel

DrawArea class handles the graphical construction and display of truss structures.

See Also:
Serialized Form

Inner classes inherited from class javax.swing.JPanel
javax.swing.JPanel.AccessibleJPanel

Inner classes inherited from class javax.swing.JComponent
javax.swing.JComponent.AccessibleJComponent

Inner classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Inner classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent

Field Summary	
static int	**DRAW_HEIGHT** Integer value representing width of the DrawArea.
static int	**DRAW_WIDTH** Integer value representing width of the DrawArea.
static int	**X_ORIGIN_OFFSET** Integer value representing x offset.
static int	**Y_ORIGIN_OFFSET** Integer value representing y offset.

Fields inherited from class javax.swing.JComponent
`accessibleContext, listenerList, TOOL_TIP_TEXT_KEY, ui, UNDEFINED_CONDITION, WHEN_ANCESTOR_OF_FOCUSED_COMPONENT, WHEN_FOCUSED, WHEN_IN_FOCUSED_WINDOW`

Fields inherited from class java.awt.Component
`BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT`

Fields inherited from interface java.awt.image.ImageObserver
`ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH`

Constructor Summary

`DrawArea``(TorsusView v)` Constructor takes view as an argument to enable communication in both directions between classes.

Method Summary

`void`	**`addMember`**`(Node m, Node n)` Calls the method of the same name on the local TrussStructure.
`void`	**`displayNodeMemberNumbers`**`(boolean b)` If boolean b is true, the numbering scheme of the Nodes and Members is displayed.
`void`	**`drawLine`**`(java.awt.Point a, java.awt.Point b)` Draws a line between Points a and b.
`void`	**`drawSnap`**`(Node n, boolean b)` Depending on value of boolean b, this method will draw a "snap square" around Node n.
`void`	**`drawXLoad`**`(Node n, java.awt.Graphics graphics)` Draws the arrow and force corresponding to the of x-component of the load at Node n.
`void`	**`drawYLoad`**`(Node n, java.awt.Graphics graphics)` Draws the arrow and force corresponding to the of y-component of the load at Node n.
`TrussStructure`	**`getTrussStructure`**`()` Returns most current copy of local TrussStructure.
`void`	**`highlightMember`**`(Member m, boolean b)` Depending on value of boolean b, this method will highlight Member m.
`void`	**`highlightModifyNode`**`(Node n, boolean b)` Depending on value of boolean b, this method will highlight Node n.
`void`	**`modifyStructure`**`(Node n, java.awt.Point p)` Calls the method of the same name on the local TrussStructure.
`void`	**`paintComponent`**`(java.awt.Graphics g)` This method overrides the paintComponent() method of JPanel.
`void`	**`paintDisplacedNodes`**`(TrussStructure t)` Causes DrawArea to display only the displaced structure.
`void`	**`paintInternalForces`**`(TrussStructure t)` Causes DrawArea to display only the internal forces of the structure.
`void`	**`paintStructureOnly`**`()` Causes DrawArea to display only the structure (build mode).
`void`	**`refresh`**`()` Calls repaint() on self.
`void`	**`removeMember`**`(Node m, Node n)` Calls the method of the same name on the local TrussStructure.
`void`	**`resetScreen`**`()` Resets local TrussStructure and reverts to build mode.
`void`	**`resetTempLine`**`()`

	Erases any remnant of the "sticky" line used in graphical addition Members.
void	**setBC**(Node n, int mode) Calls the method of the same name on the local TrussStructure.
void	**setBGBlack**() Changes the color scheme of the DrawArea to correspond to a black background (~15 different items are assigned colors).
void	**setBGWhite**() Changes the color scheme of the DrawArea to correspond to a white background (~15 different items are assigned colors).
void	**setGrid**(boolean b, int n) If boolean b is true, grid is turned on at a spacing of n pixels.
void	**setLoad**(Node n, double fx, double fy) Calls the method of the same name on the local TrussStructure.
void	**setSnapDistance**(int n) Calls the method of the same name on the local TrussStructure.
void	**setTrussStructure**(TrussStructure t) Sets the current TrussStructure according to argument.
void	**undo**() Reverts local TrussStructure to form before last build operation.

Methods inherited from class javax.swing.JPanel
getAccessibleContext, getUIClassID, paramString, updateUI

Methods inherited from class javax.swing.JComponent
addAncestorListener, addNotify, addPropertyChangeListener, addPropertyChangeListener, addVetoableChangeListener, computeVisibleRect, contains, createToolTip, disable, enable, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getActionMap, getAlignmentX, getAlignmentY, getAutoscrolls, getBorder, getBounds, getClientProperty, getComponentGraphics, getConditionForKeyStroke, getDebugGraphicsOptions, getGraphics, getHeight, getInputMap, getInputMap, getInputVerifier, getInsets, getInsets, getListeners, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPreferredSize, getRegisteredKeyStrokes, getRootPane, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getVerifyInputWhenFocusTarget, getVisibleRect, getWidth, getX, getY, grabFocus, hasFocus, hide, isDoubleBuffered, isFocusCycleRoot, isFocusTraversable, isLightweightComponent, isManagingFocus, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isOptimizedDrawingEnabled, isPaintingTile, isPreferredSizeSet, isRequestFocusEnabled, isValidateRoot, paint, paintBorder, paintChildren, paintImmediately, paintImmediately, print, printAll, printBorder, printChildren, printComponent, processComponentKeyEvent, processFocusEvent, processKeyBinding, processKeyEvent, processMouseMotionEvent, putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify, removePropertyChangeListener, removePropertyChangeListener, removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, resetKeyboardActions, reshape, revalidate, scrollRectToVisible, setActionMap, setAlignmentX, setAlignmentY, setAutoscrolls, setBackground, setBorder, setDebugGraphicsOptions, setDoubleBuffered, setEnabled, setFont, setForeground, setInputMap, setInputVerifier, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPreferredSize, setRequestFocusEnabled, setToolTipText, setUI, setVerifyInputWhenFocusTarget, setVisible, unregisterKeyboardAction, update

Methods inherited from class java.awt.Container
add, add, add, add, add, addContainerListener, addImpl, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getLayout, insets, invalidate, isAncestorOf, layout, list, list, locate, minimumSize, paintComponents, preferredSize, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, setLayout, validate, validateTree

Methods inherited from class java.awt.Component
action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, bounds, checkImage, checkImage, coalesceEvents, contains, createImage, createImage, disableEvents, dispatchEvent, enable, enableEvents, enableInputMethods, getBackground, getBounds, getColorModel, getComponentOrientation,

getCursor, getDropTarget, getFont, getFontMetrics, getForeground, getGraphicsConfiguration, getInputContext, getInputMethodRequests, getLocale, getLocation, getLocationOnScreen, getName, getParent, getPeer, getSize, getToolkit, getTreeLock, gotFocus, handleEvent, imageUpdate, inside, isDisplayable, isEnabled, isLightweight, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, processComponentEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processMouseEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, repaint, repaint, repaint, resize, resize, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setLocale, setLocation, setLocation, setName, setSize, setSize, show, show, size, toString, transferFocus

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Field Detail

DRAW_WIDTH

`public static final int DRAW_WIDTH`

Integer value representing width of the DrawArea. Set to 570 pixels.

DRAW_HEIGHT

`public static final int DRAW_HEIGHT`

Integer value representing width of the DrawArea. Set to 450 pixels.

X_ORIGIN_OFFSET

`public static final int X_ORIGIN_OFFSET`

Integer value representing x offset. Used in determining origin on DrawArea with respect to coordinate (0,0) of DrawArea.

Y_ORIGIN_OFFSET

`public static final int Y_ORIGIN_OFFSET`

Integer value representing y offset. Used in determining origin on DrawArea with respect to coordinate (0,0) of DrawArea. Note: this is in the standard, coordinate geometry meaning of y, not the Java coordinate system.

Constructor Detail

DrawArea

`public DrawArea(TorsusView v)`

Constructor takes view as an argument to enable communication in both directions between classes.

Method Detail

paintComponent

```
public void paintComponent(java.awt.Graphics g)
```

This method overrides the paintComponent() method of JPanel. Based on the current settings, it will display the appropriate information in the DrawArea.

Overrides:

`paintComponent` in class `javax.swing.JComponent`

---

getTrussStructure

```
public TrussStructure getTrussStructure()
```

Returns most current copy of local TrussStructure.

---

setTrussStructure

```
public void setTrussStructure(TrussStructure t)
```

Sets the current TrussStructure according to argument. Used to load a truss from a file.

---

refresh

```
public void refresh()
```

Calls repaint() on self.

---

paintStructureOnly

```
public void paintStructureOnly()
```

Causes DrawArea to display only the structure (build mode).

---

paintDisplacedNodes

```
public void paintDisplacedNodes(TrussStructure t)
```

Causes DrawArea to display only the displaced structure.

---

paintInternalForces

```
public void paintInternalForces(TrussStructure t)
```

Causes DrawArea to display only the internal forces of the structure.

drawLine

```
public void drawLine(java.awt.Point a,
                     java.awt.Point b)
```

Draws a line between Points a and b.

drawSnap

```
public void drawSnap(Node n,
                     boolean b)
```

Depending on value of boolean b, this method will draw a "snap square" around Node n. Used in informing user that the mouse cursor is within snap distance of Node n.

highlightMember

```
public void highlightMember(Member m,
                            boolean b)
```

Depending on value of boolean b, this method will highlight Member m. Used in informing user that the mouse cursor is over Member m (for graphical removal of Members).

highlightModifyNode

```
public void highlightModifyNode(Node n,
                                boolean b)
```

Depending on value of boolean b, this method will highlight Node n. Used in informing user that the mouse cursor is over Node n (for graphical modification of Node position).

modifyStructure

```
public void modifyStructure(Node n,
                            java.awt.Point p)
```

Calls the method of the same name on the local TrussStructure. Also updates a different local TrussStructure which is used in undo operation.

addMember

```
public void addMember(Node m,
                      Node n)
```

Calls the method of the same name on the local TrussStructure. Also updates a different local TrussStructure which is used in undo operation.

removeMember

```
public void removeMember(Node m,
                         Node n)
```

Calls the method of the same name on the local TrussStructure. Also updates a different local TrussStructure which is used in undo operation.

resetScreen

```
public void resetScreen()
```

Resets local TrussStructure and reverts to build mode. Also updates a different local TrussStructure which is used in undo operation.

resetTempLine

```
public void resetTempLine()
```

Erases any remnant of the "sticky" line used in graphical addition Members.

setBC

```
public void setBC(Node n,
                  int mode)
```

Calls the method of the same name on the local TrussStructure. Also updates a different local TrussStructure which is used in undo operation.

setBGBlack

```
public void setBGBlack()
```

Changes the color scheme of the DrawArea to correspond to a black background (~15 different items are assigned colors).

setBGWhite

```
public void setBGWhite()
```

Changes the color scheme of the DrawArea to correspond to a white background (~15 different items are assigned colors). This is the default configuration.

setGrid

```
public void setGrid(boolean b,
                    int n)
```

If boolean b is true, grid is turned on at a spacing of n pixels. If boolean b is false, grid is turned off.

---

setSnapDistance

```
public void setSnapDistance(int n)
```

Calls the method of the same name on the local TrussStructure.

---

setLoad

```
public void setLoad(Node n,
                    double fx,
                    double fy)
```

Calls the method of the same name on the local TrussStructure. Also updates a different local TrussStructure which is used in undo operation.

---

displayNodeMemberNumbers

```
public void displayNodeMemberNumbers(boolean b)
```

If boolean b is true, the numbering scheme of the Nodes and Members is displayed. If boolean b is false, the numbering scheme is not displayed.

---

drawXLoad

```
public void drawXLoad(Node n,
                      java.awt.Graphics graphics)
```

Draws the arrow and force corresponding to the of x-component of the load at Node n.

---

drawYLoad

```
public void drawYLoad(Node n,
                      java.awt.Graphics graphics)
```

Draws the arrow and force corresponding to the of y-component of the load at Node n.

---

undo

```
public void undo()
```

Reverts local TrussStructure to form before last build operation.

Class Tree Deprecated Index Help

PREV CLASS **NEXT CLASS** **FRAMES** **NO FRAMES**

SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

torsusapplet

Class ExtensionFileFilter

```
java.lang.Object
  |
  +--javax.swing.filechooser.FileFilter
        |
        +--torsusapplet.ExtensionFileFilter
```

public class **ExtensionFileFilter**
extends javax.swing.filechooser.FileFilter

A FileFilter that lets you specify which file extensions will be displayed. Also includes a static getFileName method that users can call to pop up a JFileChooser for a set of file extensions.

Adapted from Sun SwingSet demo. Taken from Core Web Programming from Prentice Hall and Sun Microsystems Press, http://www.corewebprogramming.com/. © 2001 Marty Hall and Larry Brown; may be freely used or adapted.

Field Summary

static int	**LOAD**
static int	**SAVE**

Constructor Summary

ExtensionFileFilter()
ExtensionFileFilter(boolean allowDirectories)

Method Summary

boolean	**accept**(java.io.File file)
void	**addExtension**(java.lang.String extension, boolean caseInsensitive)
java.lang.String	**getDescription**()
static java.lang.String	**getFileName**(java.lang.String initialDirectory, java.lang.String description, java.lang.String extension)
static java.lang.String	**getFileName**(java.lang.String initialDirectory, java.lang.String description, java.lang.String[] extensions)
static java.lang.String	**getFileName**(java.lang.String initialDirectory, java.lang.String description, java.lang.String[] extensions, int mode) Pops up a JFileChooser that lists files with the specified extensions.

static java.lang.String	**getFileName**(java.lang.String initialDirectory, java.lang.String description, java.lang.String extension, int mode)
void	**setDescription**(java.lang.String description)

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

LOAD

```
public static final int LOAD
```

SAVE

```
public static final int SAVE
```

Constructor Detail

ExtensionFileFilter

```
public ExtensionFileFilter(boolean allowDirectories)
```

ExtensionFileFilter

```
public ExtensionFileFilter()
```

Method Detail

getFileName

```
public static java.lang.String getFileName(java.lang.String initialDirectory,
                                           java.lang.String description,
                                           java.lang.String extension)
```

---

getFileName

```
public static java.lang.String getFileName(java.lang.String initialDirectory,
                                           java.lang.String description,
                                           java.lang.String extension,
                                           int mode)
```

---

getFileNam

```
public static java.lang.String getFileName(java.lang.String initialDirectory,
                                           java.lang.String description,
                                           java.lang.String[] extensions)
```

---

getFileName

```
public static java.lang.String getFileName(java.lang.String initialDirectory,
                                           java.lang.String description,
                                           java.lang.String[] extensions,
                                           int mode)
```

Pops up a JFileChooser that lists files with the specified extensions. If the mode is SAVE, then the dialog will have a Save button; otherwise, the dialog will have an Open button. Returns a String corresponding to the file's pathname, or null if Cancel was selected.

---

addExtension

```
public void addExtension(java.lang.String extension,
                         boolean caseInsensitive)
```

---

accept

```
public boolean accept(java.io.File file)
```

Overrides:
`accept` in class `javax.swing.filechooser.FileFilter`

---

setDescription

```
public void setDescription(java.lang.String description)
```

---

getDescription

```
public java.lang.String getDescription()
```

Overrides:
`getDescription` in class `javax.swing.filechooser.FileFilter`

Class **Tree** **Deprecated** **Index** **Help**

PREV CLASS **NEXT CLASS** **FRAMES** **NO FRAMES**

SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

torsusapplet

Class MainFrame

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--java.awt.Window
                    |
                    +--java.awt.Frame
                          |
                          +--torsusapplet.MainFrame
```

All Implemented Interfaces:

javax.accessibility.Accessible, java.applet.AppletContext, java.applet.AppletStub, java.awt.image.ImageObserver, java.awt.MenuContainer, java.lang.Runnable, java.io.Serializable

public class **MainFrame**
extends java.awt.Frame
implements java.lang.Runnable, java.applet.AppletStub, java.applet.AppletContext

See Also:

Serialized Form

Inner classes inherited from class java.awt.Frame
java.awt.Frame.AccessibleAWTFrame

Inner classes inherited from class java.awt.Window
java.awt.Window.AccessibleAWTWindow

Inner classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Inner classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent

Fields inherited from class java.awt.Frame
CROSSHAIR_CURSOR, DEFAULT_CURSOR, E_RESIZE_CURSOR, HAND_CURSOR, ICONIFIED, MOVE_CURSOR, N_RESIZE_CURSOR, NE_RESIZE_CURSOR, NORMAL, NW_RESIZE_CURSOR, S_RESIZE_CURSOR, SE_RESIZE_CURSOR, SW_RESIZE_CURSOR, TEXT_CURSOR, W_RESIZE_CURSOR, WAIT_CURSOR

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary
MainFrame(java.applet.Applet applet, int width, int height)
MainFrame(java.applet.Applet applet, java.lang.String[] args)
MainFrame(java.applet.Applet applet, java.lang.String[] args, int width, int height)

Method Summary	
void	**appletResize**(int width, int height)
java.applet.Applet	**getApplet**(java.lang.String name)
java.applet.AppletContext	**getAppletContext**()
java.util.Enumeration	**getApplets**()
java.applet.AudioClip	**getAudioClip**(java.net.URL url)
java.net.URL	**getCodeBase**()
java.net.URL	**getDocumentBase**()
java.awt.Image	**getImage**(java.net.URL url)
java.lang.String	**getParameter**(java.lang.String name)
boolean	**handleEvent**(java.awt.Event evt)
boolean	**isActive**()
void	**run**()
void	**showDocument**(java.net.URL url)
void	**showDocument**(java.net.URL url, java.lang.String target)
void	**showStatus**(java.lang.String status)

Methods inherited from class java.awt.Frame
addNotify, finalize, getAccessibleContext, getCursorType, getFrames, getIconImage, getMenuBar, getState, getTitle, isResizable, paramString, remove, removeNotify, setCursor, setIconImage, setMenuBar, setResizable, setState, setTitle

Methods inherited from class java.awt.Window
addWindowListener, applyResourceBundle, applyResourceBundle, dispose, getFocusOwner, getGraphicsConfiguration, getInputContext, getListeners, getLocale, getOwnedWindows, getOwner, getToolkit, getWarningString, hide, isShowing, pack, postEvent, processEvent, processWindowEvent, removeWindowListener, setCursor, show, toBack, toFront

Methods inherited from class java.awt.Container
`add, add, add, add, add, addContainerListener, addImpl, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getInsets, getLayout, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, processContainerEvent, remove, remove, removeAll, removeContainerListener, setFont, setLayout, update, validate, validateTree`

Methods inherited from class java.awt.Component
`action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addPropertyChangeListener, addPropertyChangeListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentOrientation, getCursor, getDropTarget, getFont, getFontMetrics, getForeground, getGraphics, getHeight, getInputMethodRequests, getLocation, getLocation, getLocationOnScreen, getName, getParent, getPeer, getSize, getSize, getTreeLock, getWidth, getX, getY, gotFocus, hasFocus, imageUpdate, inside, isDisplayable, isDoubleBuffered, isEnabled, isFocusTraversable, isLightweight, isOpaque, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processKeyEvent, processMouseEvent, processMouseMotionEvent, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, reshape, resize, resize, setBackground, setBounds, setBounds, setComponentOrientation, setDropTarget, setEnabled, setForeground, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, size, toString, transferFocus`

Methods inherited from class java.lang.Object
`clone, equals, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Methods inherited from interface java.awt.MenuContainer
`getFont, postEvent`

Constructor Detail

MainFrame

```
public MainFrame(java.applet.Applet applet,
                 java.lang.String[] args,
                 int width,
                 int height)
```

MainFrame

```
public MainFrame(java.applet.Applet applet,
                 java.lang.String[] args)
```

MainFrame

```
public MainFrame(java.applet.Applet applet,
                 int width,
                 int height)
```

Method Detail

handleEvent

```
public boolean handleEvent(java.awt.Event evt)
```

Overrides:
`handleEvent` in class `java.awt.Component`

---

run

```
public void run()
```

Specified by:
`run` in interface `java.lang.Runnable`

---

isActive

```
public boolean isActive()
```

Specified by:
`isActive` in interface `java.applet.AppletStub`

---

getDocumentBase

```
public java.net.URL getDocumentBase()
```

Specified by:
`getDocumentBase` in interface `java.applet.AppletStub`

---

getCodeBase

```
public java.net.URL getCodeBase()
```

Specified by:
`getCodeBase` in interface `java.applet.AppletStub`

---

getParameter

```
public java.lang.String getParameter(java.lang.String name)
```

Specified by:
`getParameter` in interface `java.applet.AppletStub`

---

appletResiz

```
public void appletResize(int width,
                         int height)
```

Specified by:
`appletResize` in interface `java.applet.AppletStub`

---

getAppletContext

```
public java.applet.AppletContext getAppletContext()
```

Specified by:
`getAppletContext` in interface `java.applet.AppletStub`

---

getAudioClip

```
public java.applet.AudioClip getAudioClip(java.net.URL url)
```

Specified by:
`getAudioClip` in interface `java.applet.AppletContext`

---

getImage

```
public java.awt.Image getImage(java.net.URL url)
```

Specified by:
`getImage` in interface `java.applet.AppletContext`

---

getApplet

```
public java.applet.Applet getApplet(java.lang.String name)
```

Specified by:
`getApplet` in interface `java.applet.AppletContext`

---

getApplets

```
public java.util.Enumeration getApplets()
```

Specified by:
`getApplets` in interface `java.applet.AppletContext`

---

showDocument

```
public void showDocument(java.net.URL url)
```

Specified by:
`showDocument` in interface `java.applet.AppletContext`

showDocument

```
public void showDocument(java.net.URL url,
                         java.lang.String target)
```

Specified by:
`showDocument` in interface `java.applet.AppletContext`

showStatus

```
public void showStatus(java.lang.String status)
```

Specified by:
`showStatus` in interface `java.applet.AppletContext`

Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS **FRAMES NO FRAMES**

SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class **Tree** **Deprecated** **Index** **Help**
PREV CLASS **NEXT CLASS** **FRAMES** **NO FRAMES**
SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

torsusapplet

Class Member

```
java.lang.Object
  |
  +--torsusapplet.Member
```

All Implemented Interfaces:
java.io.Serializable

---

public class **Member**
extends java.lang.Object
implements java.io.Serializable

Stores data about a Member in terms of two constituent Nodes, stiffness constant (AE/L), local stiffness matrix, internal force.

See Also:
Serialized Form

Constructor Summary

Member(Node m, Node n)
Constructor takes two Nodes as arguments.

Method Summary

double	**getAngle**() Returns the angle of the member to the horizontal in radians.
Node	**getFirst**() Returns first node.
double [][]	**getFullLocalStiffnessMatrix**() Returns a matrix (double [][]) representation of the 4x4 local stiffness matrix.
int	**getID**() Returns ID number.
double	**getInternalForce**() Calculates and returns internal force based on stiffness K and deformed length, which is calculated by obtaining the x and y displacements for each of the constituent Nodes.
double	**getLength**() Returns the length of the Member in pixels.
double [][]	**getLocalStiffnessMatrix**() Returns a matrix (double [][]) representation of the upper left (elements 11,12,21,22) of the local stiffness matrix.
Node	**getSecond**() Returns second node.
boolean	**isEqual**(Member m) Determines if two Members are comprised of the same two Nodes, using the isEqual() method of the Node class.
void	**setFirst**(Node n) Sets first node.

void	`setID(int n)` Sets ID number.
void	`setInternalForce(double d)` Sets internal force.
void	`setK(double d)` Sets element stiffness (AE/L).
void	`setSecond(Node n)` Sets second node.

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

Member

```
public Member(Node m,
              Node n)
```

Constructor takes two Nodes as arguments. ID must be set later as there is no constructor that takes the ID as an argument.

Method Detail

getFirst

```
public Node getFirst()
```

Returns first node.

---

getSecond

```
public Node getSecond()
```

Returns second node.

---

setFirst

```
public void setFirst(Node n)
```

Sets first node.

---

setSecond

```
public void setSecond(Node n)
```

Sets second node.

---

getID

```
public int getID()
```

Returns ID number.

---

setID

```
public void setID(int n)
```

Sets ID number.

---

setK

```
public void setK(double d)
```

Sets element stiffness (AE/L).

---

setInternalForce

```
public void setInternalForce(double d)
```

Sets internal force. Note that a negative force is assumed to be compressive, and that a positive force is assumed to be tensile.

---

getInternalForce

```
public double getInternalForce()
```

Calculates and returns internal force based on stiffness K and deformed length, which is calculated by obtaining the x and y displacements for each of the constituent Nodes.

---

getLocalStiffnessMatrix

```
public double[][] getLocalStiffnessMatrix()
```

Returns a matrix (double [][]) representation of the upper left (elements 11,12,21,22) of the local stiffness matrix. This information is sufficient to construct the entire 4x4 matrix for any element stiffness matrix.

---

getFullLocalStiffnessMatrix

```
public double[][] getFullLocalStiffnessMatrix()
```

Returns a matrix (double [][]) representation of the 4x4 local stiffness matrix. It uses the getLocalStiffnessMatrix() method to construct the entire matrix.

---

getAngle

```
public double getAngle()
```

Returns the angle of the member to the horizontal in radians.

getLength

```
public double getLength()
```

Returns the length of the Member in pixels.

isEqual

```
public boolean isEqual(Member m)
```

Determines if two Members are comprised of the same two Nodes, using the isEqual() method of the Node class. This is only a geometric check and does not identify differences in stiffness constants, stiffness matrices, or ID numbers. (Note: this is distinct from the equals() method, which yields different results.)

Class **Tree** **Deprecated** **Index** **Help**

PREV CLASS **NEXT CLASS** **FRAMES** **NO FRAMES**

SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class **Tree** **Deprecated** **Index** **Help**

PREV CLASS **NEXT CLASS** **FRAMES** **NO FRAMES**

SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

torsusapplet

Class Node

```
java.lang.Object
  |
  +--torsusapplet.Node
```

All Implemented Interfaces:

java.io.Serializable

public class **Node**
extends java.lang.Object
implements java.io.Serializable

Node object stores the coordinates, boundary condition state, and loading condition of a particular node - or joint - in the truss structure.

See Also:

Serialized Form

Field Summary

`static int`	**`FIX_X`** Constraint in the x-direction.
`static int`	**`FIX_XY`** Constraints in the x- and y-directions.
`static int`	**`FIX_Y`** Constraint in the y-direction.
`static int`	**`NO_BC`** No boundary condition imposed.

Constructor Summary

`Node``(int x, int y)` Constructor takes x and y coordinates; ID number assigned to be 0.
`Node``(int x, int y, int id)` Constructor takes x and y coordinates, and an integer ID number.

Method Summary

`void`	**`addMember`**`(Member m)` Add a Member to the Node's Vector of Members.
`double`	**`distanceToNode`**`(Node n)` Calculates distance (in pixels) between two Nodes.
`int`	**`getBC`**`()` Return an int representing the boundary condition of the Node.
`double`	**`getFx`**`()` Return the force in the x-direction.
`double`	

	getFy() Return the force in the y-direction.
int	**getID**() Returns ID number.
java.util.Vector	**getMembers**() Returns a Vector containing all Members associated with this Node (can be used to eliminate "floating nodes", Nodes that are not associated with any Members (since Member can be directly deleted by user input but Nodes cannot)
int	**getNumberOfLoads**() Returns number of loads (0-2) that are not applied in constrained directions.
java.awt.Point	**getPoint**() Returns Point representation of Node.
int	**getX**() Returns x coordinate.
double	**getXDispl**() Returns the displacement of the Node in the x-direction.
int	**getXDOF**() Returns integer representing the x degree of freedom in global sense.
int	**getY**() Returns y coordinate.
double	**getYDispl**() Returns the displacement of the Node in the y-direction.
int	**getYDOF**() Returns integer representing the y degree of freedom in global sense.
boolean	**isEqual**(Node n) Determines if two Nodes have the same x and y coordinates.
boolean	**nearNode**(Node n, int e) Determines if two nodes are within some distance (in pixels) epsilon of one another.
void	**removeMember**(Member m) Remove a Member from the Node's Vector of Members.
void	**setBC**(int condition) Set the boundary condition of this Node.
void	**setFx**(double force) Set the force in the x-direction.
void	**setFy**(double force) Set the force in the y-direction.
void	**setID**(int n) Sets ID number.
void	**setMembers**(java.util.Vector v) Set Vector of Members to be the argument.
void	**setX**(int n) Sets x coordinate.
void	**setXDispl**(double d) Sets the x displacement.
void	**setXDOF**(int n) Sets the integer representing the x degree of freedom in global sense.
void	**setY**(int n) Sets y coordinate
void	**setYDispl**(double d) Sets the y displacement.
void	**setYDOF**(int n) Sets the integer representing the y degree of freedom in global sense.

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

NO_BC

```
public static final int NO_BC
```

No boundary condition imposed.

FIX_X

```
public static final int FIX_X
```

Constraint in the x-direction.

FIX_Y

```
public static final int FIX_Y
```

Constraint in the y-direction.

FIX_XY

```
public static final int FIX_XY
```

Constraints in the x- and y-directions.

Constructor Detail

Node

```
public Node(int x,
            int y,
            int id)
```

Constructor takes x and y coordinates, and an integer ID number.

Node

```
public Node(int x,
            int y)
```

Constructor takes x and y coordinates; ID number assigned to be 0.

Method Detail

getX

```
public int getX()
```

Returns x coordinate.

---

getY

```
public int getY()
```

Returns y coordinate.

---

setX

```
public void setX(int n)
```

Sets x coordinate.

---

setY

```
public void setY(int n)
```

Sets y coordinate

---

getID

```
public int getID()
```

Returns ID number.

---

setID

```
public void setID(int n)
```

Sets ID number.

---

getXDOF

```
public int getXDOF()
```

Returns integer representing the x degree of freedom in global sense.

---

getYDOF

```
public int getYDOF()
```

Returns integer representing the y degree of freedom in global sense.

getXDispl

```
public double getXDispl()
```

Returns the displacement of the Node in the x-direction.

getYDispl

```
public double getYDispl()
```

Returns the displacement of the Node in the y-direction.

getNumberOfLoads

```
public int getNumberOfLoads()
```

Returns number of loads (0-2) that are not applied in constrained directions.

setXDOF

```
public void setXDOF(int n)
```

Sets the integer representing the x degree of freedom in global sense.

setYDOF

```
public void setYDOF(int n)
```

Sets the integer representing the y degree of freedom in global sense.

setXDispl

```
public void setXDispl(double d)
```

Sets the x displacement.

setYDisp

```
public void setYDispl(double d)
```

Sets the y displacement.

---

getMembers

```
public java.util.Vector getMembers()
```

Returns a Vector containing all Members associated with this Node (can be used to eliminate "floating nodes", Nodes that are not associated with any Members (since Member can be directly deleted by user input but Nodes cannot)

---

setMembers

```
public void setMembers(java.util.Vector v)
```

Set Vector of Members to be the argument.

---

getPoint

```
public java.awt.Point getPoint()
```

Returns Point representation of Node.

---

addMember

```
public void addMember(Member m)
```

Add a Member to the Node's Vector of Members.

---

removeMember

```
public void removeMember(Member m)
```

Remove a Member from the Node's Vector of Members.

---

getFx

```
public double getFx()
```

Return the force in the x-direction.

---

getFy

```
public double getFy()
```

Return the force in the y-direction.

---

setFx

```
public void setFx(double force)
```

Set the force in the x-direction.

---

setFy

```
public void setFy(double force)
```

Set the force in the y-direction.

---

getBC

```
public int getBC()
```

Return an int representing the boundary condition of the Node.

---

setBC

```
public void setBC(int condition)
```

Set the boundary condition of this Node.

---

isEqual

```
public boolean isEqual(Node n)
```

Determines if two Nodes have the same x and y coordinates. (Note: this is distinct from the equals() method, which yields different results.)

---

nearNode

```
public boolean nearNode(Node n,
                        int e)
```

Determines if two nodes are within some distance (in pixels) epsilon of one another.

---

distanceToNode

```
public double distanceToNode(Node n)
```

Calculates distance (in pixels) between two Nodes.

Class **Tree** **Deprecated** **Index** **Help**

PREV CLASS **NEXT CLASS** **FRAMES** **NO FRAMES**

SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

torsusapplet

Class ObjectCloner

```
java.lang.Object
  |
  +--torsusapplet.ObjectCloner
```

public class **ObjectCloner**
extends java.lang.Object

Static method deepCopy is used for making a deep copy of an object. This class exists for no other reason than the use of this method. TrussStructure is copied using this method for purposes of undo procedure. This utilizes serialization, ObjectOutputStream,ByteArrayOutputStream, ByteArrayInputStream, and ObjectInputStream to circumvent problems involved with cloning and shallow copying. Note that a hand coded deep copy based on clone() - though it may be more cumbersome to code - will run much more quickly if creating copies of very complex data structures.

Method Summary

`static java.lang.Object`	`deepCopy(java.lang.Object oldObj)` Returns a deep copy of an object.

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Method Detail

deepCopy

```
public static java.lang.Object deepCopy(java.lang.Object oldObj)
                                 throws java.lang.Exception
```

Returns a deep copy of an object. Note that - since this method returns an Object - it will be necessary to perform the proper casting.

Class Tree Deprecated Index Help
PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

torsusapplet

Class OverconstrainedStructureException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--torsusapplet.OverconstrainedStructureException
```

All Implemented Interfaces:
java.io.Serializable

---

public class **OverconstrainedStructureException**
extends java.lang.Exception

This class extends the functionality of Exception and is used to allow the controller to determine sources of computation error in "model" of model-view-controller paradigm.

See Also:
Serialized Form

Constructor Summary

`OverconstrainedStructureException()`

Methods inherited from class java.lang.Throwable

`fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString`

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

OverconstrainedStructureException

```
public OverconstrainedStructureException()
```

Class Tree Deprecated Index Help
PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

torsusapplet

Class TorsusController

```
java.lang.Object
  |
  +--torsusapplet.TorsusController
```

All Implemented Interfaces:

java.awt.event.ActionListener, java.util.EventListener

public class **TorsusController**
extends java.lang.Object
implements java.awt.event.ActionListener

This class performs handles all communication between TorsusView and TorsusModel. It additionally is used to handle ActionEvents passed to it from TorsusView. It is the "controller" in the model-view-controller paradigm.

Field Summary

static int	**FAR_SNAP_SPACING** Integer representing "large" snap spacing in pixels.
static int	**LARGE_GRID_SPACING** Integer representing "large" grid spacing in pixels.
static int	**MEDIUM_GRID_SPACING** Integer representing "medium" grid spacing in pixels.
static int	**MEDIUM_SNAP_SPACING** Integer representing "medium" snap spacing in pixels.
static int	**NEAR_SNAP_SPACING** Integer representing "small" snap spacing in pixels.
static int	**SMALL_GRID_SPACING** Integer representing "small" grid spacing in pixels.

Method Summary

void	**actionPerformed**(java.awt.event.ActionEvent e) ActionEvent fired in TorsusView is passed to TorsusController.
double	**formatNumber**(double d, int n) This method takes a double as an argument and formats it to n positions after the decimal point (accounts for scientific notation as well).
java.lang.String	**getDensityValue**(java.lang.String mtl, java.lang.String units) Based on material name and unit type, this method returns a String representation of the numerical value of the material density, in appropriate units.
java.lang.String	**getEValue**(java.lang.String mtl, java.lang.String units) Based on material name and unit type, this method returns a String representation of the numerical value of the material Young's modulus, in appropriate units.
java.lang.String	**getUnitsType**() Queries local TorsusView object for current unit system and returns a String representation of this.
void	

	paintDisplacedNodes(TrussStructure t) Calls the method of the same name on the local TorsusView object.
void	**paintInternalForces**(TrussStructure t) Calls the method of the same name on the local TorsusView object.
void	**resetAddBCNode**() Resets the local copy of the Node which represents the latest boundary condition change (used by own actionPerformed() method).
void	**resetAddLoadNode**() Resets the local copy of the Node which represents the latest load change (used by own actionPerformed() method).
void	**setAddBCNode**(Node n) Sets the local copy of the Node which represents the latest boundary condition change (used by own actionPerformed() method).
void	**setAddLoadNode**(Node n) Sets the local copy of the Node which represents the latest load change (used by own actionPerformed() method).
void	**setEnabledViewRadioButtons**(boolean b) Calls the method of the same name on the local TorsusView object.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

SMALL_GRID_SPACING

```
public static final int SMALL_GRID_SPACING
```

Integer representing "small" grid spacing in pixels. Set to 10.

MEDIUM_GRID_SPACING

```
public static final int MEDIUM_GRID_SPACING
```

Integer representing "medium" grid spacing in pixels. Set to 20.

LARGE_GRID_SPACING

```
public static final int LARGE_GRID_SPACING
```

Integer representing "large" grid spacing in pixels. Set to 40.

NEAR_SNAP_SPACING

```
public static final int NEAR_SNAP_SPACING
```

Integer representing "small" snap spacing in pixels. Snap spacing is the distance at which one Node will graphically snap to another. This distance also determines the graphical tolerance for the user in modifying Node properties. Set to 10.

MEDIUM_SNAP_SPACING

```
public static final int MEDIUM_SNAP_SPACING
```

Integer representing "medium" snap spacing in pixels. Snap spacing is the distance at which one Node will graphically snap to another. This distance also determines the graphical tolerance for the user in modifying Node properties. Set to 20.

FAR_SNAP_SPACING

```
public static final int FAR_SNAP_SPACING
```

Integer representing "large" snap spacing in pixels. Snap spacing is the distance at which one Node will graphically snap to another. This distance also determines the graphical tolerance for the user in modifying Node properties. Set to 50.

Method Detail

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

ActionEvent fired in TorsusView is passed to TorsusController. This method parses source of ActionEvent and delegates appropriate actions to TorsusModel and TorsusView.
Specified by:
`actionPerformed` in interface `java.awt.event.ActionListener`

---

setAddBCNode

```
public void setAddBCNode(Node n)
```

Sets the local copy of the Node which represents the latest boundary condition change (used by own actionPerformed() method).

---

resetAddBCNode

```
public void resetAddBCNode()
```

Resets the local copy of the Node which represents the latest boundary condition change (used by own actionPerformed() method).

---

paintDisplacedNodes

```
public void paintDisplacedNodes(TrussStructure t)
```

Calls the method of the same name on the local TorsusView object.

---

paintInternalForces

```
public void paintInternalForces(TrussStructure t)
```

Calls the method of the same name on the local TorsusView object.

---

setAddLoadNode

```
public void setAddLoadNode(Node n)
```

Sets the local copy of the Node which represents the latest load change (used by own actionPerformed() method).

---

resetAddLoadNode

```
public void resetAddLoadNode()
```

Resets the local copy of the Node which represents the latest load change (used by own actionPerformed() method).

---

setEnabledViewRadioButtons

```
public void setEnabledViewRadioButtons(boolean b)
```

Calls the method of the same name on the local TorsusView object.

---

formatNumber

```
public double formatNumber(double d,
                           int n)
```

This method takes a double as an argument and formats it to n positions after the decimal point (accounts for scientific notation as well).

---

getUnitsType

```
public java.lang.String getUnitsType()
```

Queries local TorsusView object for current unit system and returns a String representation of this.

---

getEValue

```
public java.lang.String getEValue(java.lang.String mtl,
                                  java.lang.String units)
```

Based on material name and unit type, this method returns a String representation of the numerical value of the material Young's modulus, in appropriate units.

---

getDensityValue

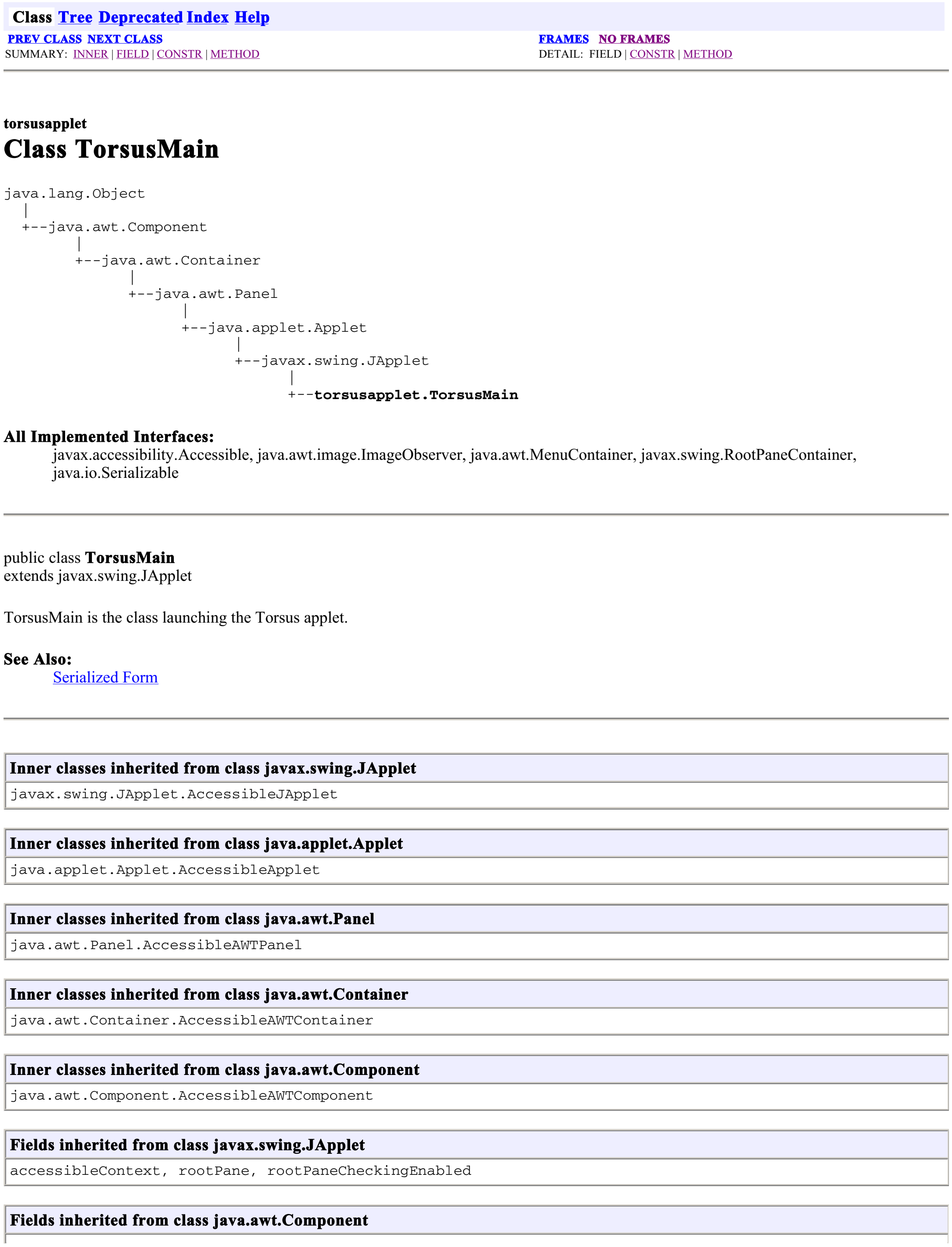
```
public java.lang.String getDensityValue(java.lang.String mtl,
                                        java.lang.String units)
```

Based on material name and unit type, this method returns a String representation of the numerical value of the material density, in appropriate units.

Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD



Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS FRAMES NO FRAMES

SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

torsusapplet

Class TorsusMain

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--java.awt.Panel
                    |
                    +--java.applet.Applet
                          |
                          +--javax.swing.JApplet
                                |
                                +--torsusapplet.TorsusMain
```

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.image.ImageObserver, java.awt.MenuContainer, javax.swing.RootPaneContainer, java.io.Serializable

public class **TorsusMain**
extends javax.swing.JApplet

TorsusMain is the class launching the Torsus applet.

See Also:

Serialized Form

Inner classes inherited from class javax.swing.JApplet
javax.swing.JApplet.AccessibleJApplet

Inner classes inherited from class java.applet.Applet
java.applet.Applet.AccessibleApplet

Inner classes inherited from class java.awt.Panel
java.awt.Panel.AccessibleAWTPanel

Inner classes inherited from class java.awt.Container
java.awt.Container.AccessibleAWTContainer

Inner classes inherited from class java.awt.Component
java.awt.Component.AccessibleAWTComponent

Fields inherited from class javax.swing.JApplet
accessibleContext, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Component

`BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT`

Fields inherited from interface java.awt.image.ImageObserver

`ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH`

Constructor Summary

`TorsusMain``()`
Allows public static void main() to perform same operations as init().

Method Summary

`void`	**`init`**`()` Initializes the applet by calling mainMethod().
`static void`	**`main`**`(java.lang.String[] args)` Program main method.

Methods inherited from class javax.swing.JApplet

`addImpl, createRootPane, getAccessibleContext, getContentPane, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isRootPaneCheckingEnabled, paramString, processKeyEvent, remove, setContentPane, setGlassPane, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, update`

Methods inherited from class java.applet.Applet

`destroy, getAppletContext, getAppletInfo, getAudioClip, getAudioClip, getCodeBase, getDocumentBase, getImage, getImage, getLocale, getParameter, getParameterInfo, isActive, newAudioClip, play, play, resize, resize, setStub, showStatus, start, stop`

Methods inherited from class java.awt.Panel

`addNotify`

Methods inherited from class java.awt.Container

`add, add, add, add, add, addContainerListener, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getInsets, getLayout, getListeners, getMaximumSize, getPreferredSize, insets, invalidate, isAncestorOf, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, processContainerEvent, processEvent, remove, removeAll, removeContainerListener, removeNotify, setFont, validate, validateTree`

Methods inherited from class java.awt.Component

`action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addPropertyChangeListener, addPropertyChangeListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentOrientation, getCursor, getDropTarget, getFont, getFontMetrics, getForeground, getGraphics, getGraphicsConfiguration, getHeight, getInputContext, getInputMethodRequests, getLocation, getLocation, getLocationOnScreen, getName, getParent, getPeer, getSize, getSize, getToolkit, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, hide, imageUpdate, inside, isDisplayable, isDoubleBuffered, isEnabled, isFocusTraversable, isLightweight, isOpaque, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processMouseEvent, processMouseMotionEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, reshape, setBackground, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setEnabled, setForeground, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, show, size, toString, transferFocus`

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

TorsusMain

```
public TorsusMain()
```

Allows public static void main() to perform same operations as init().

Method Detail

init

```
public void init()
```

Initializes the applet by calling mainMethod().
Overrides:
`init` in class `java.applet.Applet`

---

main

```
public static void main(java.lang.String[] args)
```

Program main method. Allows program to be run as an application.

Class Tree Deprecatec Index Help

PREV CLASS NEXT CLASS **FRAMES NO FRAMES**

SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class Tree Deprecated Index Help
PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

torsusapplet

Class TorsusModel

```
java.lang.Object
  |
  +--torsusapplet.TorsusModel
```

public class **TorsusModel**
extends java.lang.Object

This class performs all computation related to the behavior of the TrussStructure under specified loading and boundary conditions. It is the "model" in the model-view-controller paradigm.

Constructor Summary

`TorsusModel()`
Initializes an empty TorsusModel object.

Method Summary

double	`calculateTrussMass(java.util.Vector v, double a, double rho, double scale)` Computes the mass of the TrussStructure based on Vector of Members, member cross-sectional area, material density, and user-defined scale (units of the last three must be consistent for a correct output).
void	`compute(double a, double e, double rho, double scale)` This method performs the heavy computation.
double	`formatNumber(double d, int n)` This method takes a double as an argument and formats it to n positions after the decimal point (accounts for scientific notation as well).
TrussStructure	`getTrussStructure()` Returns the local TrussStructure.
void	`paintDisplacedNodes()` Calls the TrussController method of the same name.
void	`paintInternalForces()` Calls the TrussController method of the same name.
java.lang.String	`printOutput(java.lang.String momentOfInertiaType, double momentOfInertiaValue)` Generates an output of the computation to be used in the Message Window.
TrussStructure	`resetIDNumbers(TrussStructure t)` Resets ID numbers for all Nodes and Members in local TrussStructure.
void	`setController(TorsusController c)` Sets the controller to allow the Model to communicate with the controller.
TrussStructure	`setIDNumbers(TrussStructure t)` Generates ID numbers for all Nodes and Members in local TrussStructure.
void	`setTrussStructure(TrussStructure t)` Sets the local TrussStructure.

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

TorsusModel

```
public TorsusModel()
```

Initializes an empty TorsusModel object.

Method Detail

setController

```
public void setController(TorsusController c)
```

Sets the controller to allow the Model to communicate with the controller.

---

setTrussStructure

```
public void setTrussStructure(TrussStructure t)
```

Sets the local TrussStructure.

---

getTrussStructure

```
public TrussStructure getTrussStructure()
```

Returns the local TrussStructure.

---

compute

```
public void compute(double a,
                    double e,
                    double rho,
                    double scale)
             throws java.lang.Exception
```

This method performs the heavy computation. Utilizing matrix truss analysis (and the Jama, or Java Matrix, package published jointly by NIST and MathWorks) it computes the displacements of the Nodes of the TrussStructure. It throws Exceptions in the following cases: improper scale value (out of range or invalid number), improper area value, overconstrained structure, underloaded structure, and unstable structure. Note that a statically indeterminate structure is not a barrier to computation since we are using the displacement method of solving the system.

---

paintDisplacedNodes

```
public void paintDisplacedNodes()
```

Calls the TrussController method of the same name.

paintInternalForces

```
public void paintInternalForces()
```

Calls the TrussController method of the same name.

setIDNumbers

```
public TrussStructure setIDNumbers(TrussStructure t)
```

Generates ID numbers for all Nodes and Members in local TrussStructure.

resetIDNumbers

```
public TrussStructure resetIDNumbers(TrussStructure t)
```

Resets ID numbers for all Nodes and Members in local TrussStructure.

calculateTrussMass

```
public double calculateTrussMass(java.util.Vector v,
                                 double a,
                                 double rho,
                                 double scale)
```

Computes the mass of the TrussStructure based on Vector of Members, member cross-sectional area, material density, and user-defined scale (units of the last three must be consistent for a correct output).

printOutput

```
public java.lang.String printOutput(java.lang.String momentOfInertiaType,
                                    double momentOfInertiaValue)
```

Generates an output of the computation to be used in the Message Window. Based on Node numbers and Member numbers - which are displayed graphically, this method displays member forces, member stresses, node displacements, members which are in danger of buckling (assuming solid, circular cross-sections), maximum magnitude of stress in structure, and structure mass. Takes a String representing moment of inertia type and a double representing value of moment of inertia for buckling analysis.

formatNumber

```
public double formatNumber(double d,
                           int n)
```

This method takes a double as an argument and formats it to n positions after the decimal point (accounts for scientific notation as well).

Class **Tree** **Deprecated** **Index** **Help**

PREV CLASS **NEXT CLASS** **FRAMES** **NO FRAMES**

SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class Tree Deprecated Index Help
PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

torsusapplet

Class TorsusView

```
java.lang.Object
  |
  +--java.awt.Component
        |
        +--java.awt.Container
              |
              +--java.awt.Panel
                    |
                    +--java.applet.Applet
                          |
                          +--javax.swing.JApplet
                                |
                                +--torsusapplet.TorsusView
```

All Implemented Interfaces:

javax.accessibility.Accessible, java.awt.event.ActionListener, java.awt.datatransfer.ClipboardOwner, java.util.EventListener, java.awt.image.ImageObserver, java.awt.event.KeyListener, java.awt.MenuContainer, java.awt.event.MouseListener, java.awt.event.MouseMotionListener, javax.swing.RootPaneContainer, java.io.Serializable

---

public class **TorsusView**
extends javax.swing.JApplet
implements java.awt.event.ActionListener, java.awt.datatransfer.ClipboardOwner, java.awt.event.KeyListener, java.awt.event.MouseListener, java.awt.event.MouseMotionListener

Responsible for displaying program GUI and delegating actions to TorsusController. Public static final int and public static final String values are declared for defining draw modes and for assigning values to specific actions. Note that this class (in addition to TorsusMain) extends JApplet. In the previous, application version of this program, TorsusView extended JFrame.

See Also:

Serialized Form

Inner classes inherited from class javax.swing.JApplet
`javax.swing.JApplet.AccessibleJApplet`

Inner classes inherited from class java.applet.Applet
`java.applet.Applet.AccessibleApplet`

Inner classes inherited from class java.awt.Panel
`java.awt.Panel.AccessibleAWTPanel`

Inner classes inherited from class java.awt.Container
`java.awt.Container.AccessibleAWTContainer`

Inner classes inherited from class java.awt.Component
`java.awt.Component.AccessibleAWTComponent`

Field Summary

static int	BC_ADDER
static int	BC_REMOVER
static int	I_ABOUT_MENU_ITEM
static int	I_ADD_BC
static int	I_ADD_BC_CANCEL
static int	I_ADD_BC_X
static int	I_ADD_BC_XY
static int	I_ADD_BC_Y
static int	I_ADD_LOAD
static int	I_ADD_LOAD_CANCEL
static int	I_ADD_LOAD_OK
static int	I_ADD_MEMBER
static int	I_BLACK_BACKGROUND
static int	I_BUILD_MODE
static int	I_CALCULATE
static int	I_COPY_DATA
static int	I_DISP_MODE
static int	I_ENGLISH_UNITS
static int	I_FAR_SNAP
static int	I_FILE_LOAD
static int	I_FILE_SAVE
static int	I_FORCE_MODE
static int	I_INSTRUCTIONS_MENU_ITEM
static int	I_LARGE_GRID
static int	I_MAG_CHANGE

static int	I_MATERIAL_CHANGE
static int	I_MEDIUM_GRID
static int	I_MEDIUM_SNAP
static int	I_MOD_NODE_TEXT_CANCEL
static int	I_MOD_NODE_TEXT_OK
static int	I_MOD_STRUCTURE
static int	I_MOMENT_CHANGE
static int	I_NEAR_SNAP
static int	I_NO_GRID
static int	I_OK_ABOUT_DIALOG
static int	I_QUIT
static int	I_REMOVE_BC
static int	I_REMOVE_LOAD
static int	I_REMOVE_MEMBER
static int	I_RESET
static int	I_RESET_DEFAULT_PREFS
static int	I_SI_UNITS
static int	I_SMALL_GRID
static int	I_STICKY_ADD_MEMBER_MODE
static int	I_UNDO
static int	I_WHITE_BACKGROUND
static int	LOAD_ADDER
static int	LOAD_REMOVER
static int	MEMBER_ADDER
static int	MEMBER_REMOVER
static int	MOD_STRUCTURE

static int	**NULL_MODE**
static java.lang.String	**S_ABOUT_MENU_ITEM**
static java.lang.String	**S_ADD_BC**
static java.lang.String	**S_ADD_BC_CANCEL**
static java.lang.String	**S_ADD_BC_X**
static java.lang.String	**S_ADD_BC_XY**
static java.lang.String	**S_ADD_BC_Y**
static java.lang.String	**S_ADD_LOAD**
static java.lang.String	**S_ADD_LOAD_CANCEL**
static java.lang.String	**S_ADD_LOAD_OK**
static java.lang.String	**S_ADD_MEMBER**
static java.lang.String	**S_BLACK_BACKGROUND**
static java.lang.String	**S_BUILD_MODE**
static java.lang.String	**S_CALCULATE**
static java.lang.String	**S_COPY_DATA**
static java.lang.String	**S_DISP_MODE**
static java.lang.String	**S_ENGLISH_UNITS**
static java.lang.String	**S_FAR_SNAP**
static java.lang.String	**S_FILE_LOAD**
static java.lang.String	**S_FILE_SAVE**
static java.lang.String	**S_FORCE_MODE**
static java.lang.String	**S_INSTRUCTIONS_MENU_ITEM**
static java.lang.String	**S_LARGE_GRID**
static java.lang.String	**S_MAG_CHANGE**
static java.lang.String	**S_MATERIAL_CHANGE**
static java.lang.String	**S_MEDIUM_GRID**

static java.lang.String	**S_MEDIUM_SNAP**
static java.lang.String	**S_MOD_NODE_TEXT_CANCEL**
static java.lang.String	**S_MOD_NODE_TEXT_OK**
static java.lang.String	**S_MOD_STRUCTURE**
static java.lang.String	**S_MOMENT_CHANGE**
static java.lang.String	**S_NEAR_SNAP**
static java.lang.String	**S_NO_GRID**
static java.lang.String	**S_OK_ABOUT_DIALOG**
static java.lang.String	**S_QUIT**
static java.lang.String	**S_REMOVE_BC**
static java.lang.String	**S_REMOVE_LOAD**
static java.lang.String	**S_REMOVE_MEMBER**
static java.lang.String	**S_RESET**
static java.lang.String	**S_RESET_DEFAULT_PREFS**
static java.lang.String	**S_SI_UNITS**
static java.lang.String	**S_SMALL_GRID**
static java.lang.String	**S_STICKY_ADD_MEMBER_MODE**
static java.lang.String	**S_UNDO**
static java.lang.String	**S_WHITE_BACKGROUND**

Fields inherited from class javax.swing.JApplet
accessibleContext, rootPane, rootPaneCheckingEnabled

Fields inherited from class java.awt.Component
BOTTOM_ALIGNMENT, CENTER_ALIGNMENT, LEFT_ALIGNMENT, RIGHT_ALIGNMENT, TOP_ALIGNMENT

Fields inherited from interface java.awt.image.ImageObserver
ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

TorsusView(java.lang.String s, javax.swing.JApplet applet)
Sets the title to to the value of String s and uses the applet argument to set the URL target for the instructions page (in the help menu).

Method Summary

void	**actionPerformed**(java.awt.event.ActionEvent e) This method catches ActionEvents and delegates them to the actionPerformed() method of the local TorsusController.
void	**closeAddBCDialog**() Closes the dialog corresponding to adding/modifying boundary conditions.
void	**closeAddLoadDialog**() Closes the dialog corresponding to adding/modifying loads.
void	**closeModStructureDialog**() Closes the dialog corresponding to modifying the structure.
void	**copyData**() Copies message area contents to System clipboard.
void	**displayNodeMemberNumbers**(boolean b) Calls method of same name on local copy of DrawArea.
double	**formatNumber**(double d, int n) This method takes a double as an argument and formats it to n positions after the decimal point (accounts for scientific notation as well).
double	**getAreaTextField**() Returns a double representation of the current user-defined area.
TorsusController	**getController**() Returns the local TorsusController.
double	**getEValueLabel**() Returns a double representation of the Young's modulus corresponding to current material.
java.lang.String	**getFilePath**() Returns path of file corresponding to JFileChooser.
double	**getFxLoadFromDialog**() Returns a double representation of the current user-defined force in the x-direction (from the dialog corresponding to adding/modifying loads).
double	**getFyLoadFromDialog**() Returns a double representation of the current user-defined force in the y-direction (from the dialog corresponding to adding/modifying loads).
int	**getGridMenuItemSpacing**() Returns an integer representation of the current grid spacing based on the state of the grid menu items and on the public static final spacing values defined as in TorsusController.
int	**getMagFactor**() Returns a integer representation of the current magnification factor (for displaying the displaced structure).
java.lang.String	**getMaterialType**() Returns a String representation of the current material type.
java.lang.String	**getMessage**() Returns the contents of the Message Window.
Node	**getModStructureNode**() Returns the Node currently selected to be modified.
double	**getMomentInertiaTextField**() Returns a double representation of the current user-defined moment of inertia.
java.lang.String	**getMomentInertiaType**() Returns a String representation of the current user-defined moment of inertia type.
java.awt.Point	**getRealTimeMousePoint**() Returns Point, adjusted to normal coordinate geometry (not Java coordinates), representing mouse position on DrawArea in pixels.
double	**getScaleTextField**() Returns a double representation of the current user-defined scale in pixels.
TrussStructure	**getTrussStructure**()

	Returns the TrussStructure belonging to the DrawArea.
java.lang.String	**getUnitsType**() Returns a String representation of the current unit system.
double	**getXModStructureDialog**() Returns a double representation of the current user-defined x coordinate from the dialog corresponding to modifying the structure.
double	**getYModStructureDialog**() Returns a double representation of the current user-defined y coordinate from the dialog corresponding to modifying the structure.
void	**keyPressed**(java.awt.event.KeyEvent e) Empty method.
void	**keyReleased**(java.awt.event.KeyEvent e) Handles the key released events.
void	**keyTyped**(java.awt.event.KeyEvent e) Empty method.
void	**launchInstructions**() Launches the instructions page in a separate browser window.
void	**lostOwnership**(java.awt.datatransfer.Clipboard clip, java.awt.datatransfer.Transferable tr) This method does nothing in this implementation.
void	**mouseClicked**(java.awt.event.MouseEvent e) Empty method.
void	**mouseDragged**(java.awt.event.MouseEvent e) Handles user mouse dragging.
void	**mouseEntered**(java.awt.event.MouseEvent e) Empty method.
void	**mouseExited**(java.awt.event.MouseEvent e) Empty method.
void	**mouseMoved**(java.awt.event.MouseEvent e) Handles user mouse motion.
void	**mousePressed**(java.awt.event.MouseEvent e) Handles mousePressed MouseEvents.
void	**mouseReleased**(java.awt.event.MouseEvent e) Handles mouseReleased MouseEvents.
void	**paintDisplacedNodes**(TrussStructure t) Calls the method of the same name on the local DrawArea object.
void	**paintInternalForces**(TrussStructure t) Calls the method of the same name on the local DrawArea object.
void	**paintStructureOnly**() Calls the method of the same name on the local DrawArea object.
void	**refresh**() Calls the method of the same name on the local DrawArea object.
void	**repositionNodeModStructure**(java.awt.Point p) Calls modifyStructure() method on local DrawArea object based on previously selected Node.
void	**resetAddLoadDialog**() Resets the values in the dialog corresponding to adding/modifying loads.
void	**resetClassNodeVariables**() This method resets all (private) class Node variables.
void	**resetDefaultPrefs**() Resets the default preferences: sticky mode off, medium grid spacing, medium snap distance, white background.
void	**resetMessage**() Sets the Messgae Window TextArea text to "Messages appear here.".
void	**resetModStructureDialog**()

	Resets the values in the dialog corresponding to modifying the structure.
void	**resetModStructureNode**() Resets the private class variable representing the Node to be modified in the "modify structure" procedure.
void	**resetScreen**() Calls the resetSceen() method on the local DrawArea object and resets the GUI components accordingly.
void	**setAddLoadDialogUnitsLabel**(java.lang.String s) Sets the units label on the dialog corresponding to the addition of loads to String s.
void	**setAreaTextField**(java.lang.String s) Sets the area text field to String s.
void	**setAreaUnitsLabel**(java.lang.String s) Sets the units label associated with the area text field to be the String s.
void	**setBC**(Node n, int mode) Calls the method of the same name on the local DrawArea object and updates the GUI accordingly.
void	**setBGBlack**() Calls the method of the same name on the local DrawArea object and updates the GUI accordingly.
void	**setBGWhite**() Calls the method of the same name on the local DrawArea object and updates the GUI accordingly.
void	**setController**(TorsusController c) Sets the local TorsusController to the argument.
void	**setDrawMode**(int mode) Sets the current draw mode.
void	**setEnabledBuildButtons**(boolean b) Enables (or disables) buttons used in construction of structure.
void	**setEnabledUndoMenuItem**(boolean b) Enables (or disables) the menu item corresponding to the undo operation.
void	**setEnabledViewRadioButtons**(boolean b) Enables (or disables) radio buttons corresponding to display of internal forces, and of displacements.
void	**setEUnitsLabel**(java.lang.String s) Sets the units label associated with the material drop-down to be the String s.
void	**setEValueLabel**(java.lang.String s) Sets the E label associated with the material drop-down to be the String s.
void	**setFarSnapMenuItem**() Sets the selection states for snap distance menu items to indicate that the current snap distance is "far".
void	**setFxLoadFromDialog**(java.lang.String s) Sets the force in the x-direction in the dialog corresponding to adding/modifying loads.
void	**setFyLoadFromDialog**(java.lang.String s) Sets the force in the y-direction in the dialog corresponding to adding/modifying loads.
void	**setGrid**(boolean b, int n) Calls method of same name on local copy of DrawArea.
void	**setLargeGridMenuItem**() Sets the selection states for grid menu items to indicate that the current grid spacing is "large".
void	**setLoad**(Node n, double fx, double fy) Calls the method of the same name on the local DrawArea object and updates the GUI accordingly.
void	**setMediumGridMenuItem**() Sets the selection states for grid menu items to indicate that the current grid spacing is "medium".
void	**setMediumSnapMenuItem**() Sets the selection states for snap distance menu items to indicate that the current snap distance is "medium".
void	**setMessage**(java.lang.String s) Sets the Message Window TextArea text to be String s.
void	**setModStructureDialogUnits**(java.lang.String s) Sets the units label on the dialog corresponding to structure modification to String s.
void	**setMomentInertiaTextField**(java.lang.String s)

	Sets the moment of inertia text field to String s.
void	**setMomentInertiaUnitsLabel**(java.lang.String s) Sets the units label associated with the moment of inertia text field to be the String s.
void	**setNearSnapMenuItem**() Sets the selection states for snap distance menu items to indicate that the current snap distance is "near".
void	**setNoGridMenuItem**() Sets the selection states for grid menu items to indicate that the grid is off.
void	**setRealTimeMouseLabel**(double x, double y) Sets the JLabel corresponding to the current mouse position according to the input, which is assumed to be in pixels.
void	**setScaleTextField**(double d) Converts d from pixels to gridBlocks (based on the grid spacing), converts this double to a String, and displays the String in the scale text field.
void	**setScaleUnitsLabel**(java.lang.String s) Sets the units label associated with the scale text field to be the String s.
void	**setSmallGridMenuItem**() Sets the selection states for grid menu items to indicate that the current grid spacing is "small".
void	**setSnapDistance**(int n) Calls the method of the same name on the local DrawArea object.
void	**setStickyMode**() This method sets the sticky mode based on the menu item state.
void	**setStickyMode**(boolean b) This method takes a boolean b as an argument sets the the sticky mode according to b and updates the menu item accordingly.
void	**setTrussStructure**(TrussStructure t) Sets the TrussStructure belonging to the DrawArea according to the argument.
void	**setURL**(java.net.URL u) Sets the local URL to be u (used for setting the target for the instructions page in the help menu).
void	**setXModStructureDialog**(java.lang.String s) Sets x coordinate in the dialog corresponding to modifying the structure to String s.
void	**setYModStructureDialog**(java.lang.String s) Sets y coordinate in the dialog corresponding to modifying the structure to String s.
void	**showAboutDialog**(boolean b) Shows (or hides) the dialog corresponding to program "about" information.
int	**showOpenDialog**() Launches the JFileChooser corresponding to "open" function.
int	**showSaveDialog**() Launches the JFileChooser corresponding to "save" function.
void	**undo**() Calls the undo() method on the local DrawArea object and performs all necessary operations on UI objects.
void	**updateMomentInertiaTextField**() Updates the enabled state and the contents of the textfield corresponding to the custom moment of inertia.

Methods inherited from class javax.swing.JApplet
addImpl, createRootPane, getAccessibleContext, getContentPane, getGlassPane, getJMenuBar, getLayeredPane, getRootPane, isRootPaneCheckingEnabled, paramString, processKeyEvent, remove, setContentPane, setGlassPane, setJMenuBar, setLayeredPane, setLayout, setRootPane, setRootPaneCheckingEnabled, update

Methods inherited from class java.applet.Applet
destroy, getAppletContext, getAppletInfo, getAudioClip, getAudioClip, getCodeBase, getDocumentBase, getImage, getImage, getLocale, getParameter, getParameterInfo, init, isActive, newAudioClip, play, play, resize, resize, setStub, showStatus, start, stop

Methods inherited from class java.awt.Panel
`addNotify`

Methods inherited from class java.awt.Container
`add, add, add, add, add, addContainerListener, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getInsets, getLayout, getListeners, getMaximumSize, getMinimumSize, getPreferredSize, insets, invalidate, isAncestorOf, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, processContainerEvent, processEvent, remove, removeAll, removeContainerListener, removeNotify, setFont, validate, validateTree`

Methods inherited from class java.awt.Component
`action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addPropertyChangeListener, addPropertyChangeListener, bounds, checkImage, checkImage, coalesceEvents, contains, contains, createImage, createImage, disable, disableEvents, dispatchEvent, enable, enable, enableEvents, enableInputMethods, firePropertyChange, getBackground, getBounds, getBounds, getColorModel, getComponentOrientation, getCursor, getDropTarget, getFont, getFontMetrics, getForeground, getGraphics, getGraphicsConfiguration, getHeight, getInputContext, getInputMethodRequests, getLocation, getLocation, getLocationOnScreen, getName, getParent, getPeer, getSize, getSize, getToolkit, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, hide, imageUpdate, inside, isDisplayable, isDoubleBuffered, isEnabled, isFocusTraversable, isLightweight, isOpaque, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, printAll, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processMouseEvent, processMouseMotionEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, reshape, setBackground, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setEnabled, setForeground, setLocale, setLocation, setLocation, setName, setSize, setSize, setVisible, show, show, size, toString, transferFocus`

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Field Detail

NULL_MODE

```
public static final int NULL_MODE
```

MEMBER_ADDER

```
public static final int MEMBER_ADDER
```

MEMBER_REMOVER

```
public static final int MEMBER_REMOVER
```

BC_ADDER

```
public static final int BC_ADDER
```

BC_REMOVER

```
public static final int BC_REMOVER
```

LOAD_ADDER

```
public static final int LOAD_ADDER
```

LOAD_REMOVER

```
public static final int LOAD_REMOVER
```

MOD_STRUCTURE

```
public static final int MOD_STRUCTURE
```

S_ADD_MEMBER

```
public static final java.lang.String S_ADD_MEMBER
```

I_ADD_MEMBER

```
public static final int I_ADD_MEMBER
```

S_REMOVE_MEMBER

```
public static final java.lang.String S_REMOVE_MEMBER
```

I_REMOVE_MEMBER

```
public static final int I_REMOVE_MEMBER
```

S_ADD_LOAD

```
public static final java.lang.String S_ADD_LOAD
```

I_ADD_LOAI

```
public static final int I_ADD_LOAD
```

S_REMOVE_LOAD

```
public static final java.lang.String S_REMOVE_LOAD
```

I_REMOVE_LOAD

```
public static final int I_REMOVE_LOAD
```

S_ADD_BC

```
public static final java.lang.String S_ADD_BC
```

I_ADD_BC

```
public static final int I_ADD_BC
```

S_REMOVE_BC

```
public static final java.lang.String S_REMOVE_BC
```

I_REMOVE_BC

```
public static final int I_REMOVE_BC
```

S_RESET

```
public static final java.lang.String S_RESET
```

I_RESET

```
public static final int I_RESET
```

S_QUIT

```
public static final java.lang.String S_QUIT
```

I_QUIT

```
public static final int I_QUIT
```

S_MOD_STRUCTURE

```
public static final java.lang.String S_MOD_STRUCTURE
```

I_MOD_STRUCTURE

```
public static final int I_MOD_STRUCTURE
```

S_ADD_BC_X

```
public static final java.lang.String S_ADD_BC_X
```

I_ADD_BC_X

```
public static final int I_ADD_BC_X
```

S_ADD_BC_Y

```
public static final java.lang.String S_ADD_BC_Y
```

I_ADD_BC_Y

```
public static final int I_ADD_BC_Y
```

S_ADD_BC_XY

```
public static final java.lang.String S_ADD_BC_XY
```

I_ADD_BC_XY

```
public static final int I_ADD_BC_XY
```

S_ADD_BC_CANCEL

```
public static final java.lang.String S_ADD_BC_CANCEL
```

I_ADD_BC_CANCEL

```
public static final int I_ADD_BC_CANCEL
```

S_ADD_LOAD_OK

```
public static final java.lang.String S_ADD_LOAD_OK
```

I_ADD_LOAD_OK

```
public static final int I_ADD_LOAD_OK
```

S_ADD_LOAD_CANCEL

```
public static final java.lang.String S_ADD_LOAD_CANCEL
```

I_ADD_LOAD_CANCEL

```
public static final int I_ADD_LOAD_CANCEL
```

S_SI_UNITS

```
public static final java.lang.String S_SI_UNITS
```

I_SI_UNITS

```
public static final int I_SI_UNITS
```

S_ENGLISH_UNITS

```
public static final java.lang.String S_ENGLISH_UNITS
```

I_ENGLISH_UNITS

```
public static final int I_ENGLISH_UNITS
```

S_MATERIAL_CHANG

```
public static final java.lang.String S_MATERIAL_CHANGE
```

I_MATERIAL_CHANGE

```
public static final int I_MATERIAL_CHANGE
```

S_BUILD_MODE

```
public static final java.lang.String S_BUILD_MODE
```

I_BUILD_MODE

```
public static final int I_BUILD_MODE
```

S_FORCE_MODE

```
public static final java.lang.String S_FORCE_MODE
```

I_FORCE_MODE

```
public static final int I_FORCE_MODE
```

S_DISP_MODE

```
public static final java.lang.String S_DISP_MODE
```

I_DISP_MODE

```
public static final int I_DISP_MODE
```

S_CALCULATE

```
public static final java.lang.String S_CALCULATE
```

I_CALCULATE

```
public static final int I_CALCULATE
```

S_MAG_CHANGE

`public static final java.lang.String` **`S_MAG_CHANGE`**

I_MAG_CHANGE

`public static final int` **`I_MAG_CHANGE`**

S_NO_GRID

`public static final java.lang.String` **`S_NO_GRID`**

I_NO_GRID

`public static final int` **`I_NO_GRID`**

S_SMALL_GRID

`public static final java.lang.String` **`S_SMALL_GRID`**

I_SMALL_GRID

`public static final int` **`I_SMALL_GRID`**

S_MEDIUM_GRID

`public static final java.lang.String` **`S_MEDIUM_GRID`**

I_MEDIUM_GRID

`public static final int` **`I_MEDIUM_GRID`**

S_LARGE_GRID

`public static final java.lang.String` **`S_LARGE_GRID`**

I_LARGE_GRID

`public static final int` **`I_LARGE_GRID`**

S_NEAR_SNAP

```
public static final java.lang.String S_NEAR_SNAP
```

I_NEAR_SNAP

```
public static final int I_NEAR_SNAP
```

S_MEDIUM_SNAP

```
public static final java.lang.String S_MEDIUM_SNAP
```

I_MEDIUM_SNAP

```
public static final int I_MEDIUM_SNAP
```

S_FAR_SNAP

```
public static final java.lang.String S_FAR_SNAP
```

I_FAR_SNAP

```
public static final int I_FAR_SNAP
```

S_STICKY_ADD_MEMBER_MODE

```
public static final java.lang.String S_STICKY_ADD_MEMBER_MODE
```

I_STICKY_ADD_MEMBER_MODE

```
public static final int I_STICKY_ADD_MEMBER_MODE
```

S_ABOUT_MENU_ITEM

```
public static final java.lang.String S_ABOUT_MENU_ITEM
```

I_ABOUT_MENU_ITEM

```
public static final int I_ABOUT_MENU_ITEM
```

S_OK_ABOUT_DIALOG

```
public static final java.lang.String S_OK_ABOUT_DIALOG
```

I_OK_ABOUT_DIALOG

```
public static final int I_OK_ABOUT_DIALOG
```

S_INSTRUCTIONS_MENU_ITEM

```
public static final java.lang.String S_INSTRUCTIONS_MENU_ITEM
```

I_INSTRUCTIONS_MENU_ITEM

```
public static final int I_INSTRUCTIONS_MENU_ITEM
```

S_RESET_DEFAULT_PREFS

```
public static final java.lang.String S_RESET_DEFAULT_PREFS
```

I_RESET_DEFAULT_PREFS

```
public static final int I_RESET_DEFAULT_PREFS
```

S_WHITE_BACKGROUND

```
public static final java.lang.String S_WHITE_BACKGROUND
```

I_WHITE_BACKGROUND

```
public static final int I_WHITE_BACKGROUND
```

S_BLACK_BACKGROUND

```
public static final java.lang.String S_BLACK_BACKGROUND
```

I_BLACK_BACKGROUND

```
public static final int I_BLACK_BACKGROUND
```

S_MOD_NODE_TEXT_OK

```
public static final java.lang.String S_MOD_NODE_TEXT_OK
```

I_MOD_NODE_TEXT_OK

```
public static final int I_MOD_NODE_TEXT_OK
```

S_MOD_NODE_TEXT_CANCEL

```
public static final java.lang.String S_MOD_NODE_TEXT_CANCEL
```

I_MOD_NODE_TEXT_CANCEL

```
public static final int I_MOD_NODE_TEXT_CANCEL
```

S_UNDO

```
public static final java.lang.String S_UNDO
```

I_UNDO

```
public static final int I_UNDO
```

S_FILE_SAVE

```
public static final java.lang.String S_FILE_SAVE
```

I_FILE_SAVE

```
public static final int I_FILE_SAVE
```

S_FILE_LOAD

```
public static final java.lang.String S_FILE_LOAD
```

I_FILE_LOAD

```
public static final int I_FILE_LOAD
```

S_MOMENT_CHANGE

```
public static final java.lang.String S_MOMENT_CHANGE
```

I_MOMENT_CHANGE

```
public static final int I_MOMENT_CHANGE
```

S_COPY_DATA

```
public static final java.lang.String S_COPY_DATA
```

I_COPY_DATA

```
public static final int I_COPY_DATA
```

Constructor Detail

TorsusView

```
public TorsusView(java.lang.String s,
                  javax.swing.JApplet applet)
```

Sets the title to to the value of String s and uses the applet argument to set the URL target for the instructions page (in the help menu).

Method Detail

setController

```
public void setController(TorsusController c)
```

Sets the local TorsusController to the argument.

getController

```
public TorsusController getController()
```

Returns the local TorsusController.

getTrussStructure

```
public TrussStructure getTrussStructure()
```

Returns the TrussStructure belonging to the DrawArea.

---

setTrussStructure

```
public void setTrussStructure(TrussStructure t)
```

Sets the TrussStructure belonging to the DrawArea according to the argument.

---

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

This method catches ActionEvents and delegates them to the actionPerformed() method of the local TorsusController.
Specified by:
`actionPerformed` in interface `java.awt.event.ActionListener`

---

mouseMoved

```
public void mouseMoved(java.awt.event.MouseEvent e)
```

Handles user mouse motion.
Specified by:
`mouseMoved` in interface `java.awt.event.MouseMotionListener`

---

mouseDragged

```
public void mouseDragged(java.awt.event.MouseEvent e)
```

Handles user mouse dragging.
Specified by:
`mouseDragged` in interface `java.awt.event.MouseMotionListener`

---

getRealTimeMousePoint

```
public java.awt.Point getRealTimeMousePoint()
```

Returns Point, adjusted to normal coordinate geometry (not Java coordinates), representing mouse position on DrawArea in pixels.

---

setRealTimeMouseLabel

```
public void setRealTimeMouseLabel(double x,
                                  double y)
```

Sets the JLabel corresponding to the current mouse position according to the input, which is assumed to be in pixels. This method handles all scale conversion for purposes of current mouse position display.

---

setEnabledViewRadioButtons

```
public void setEnabledViewRadioButtons(boolean b)
```

Enables (or disables) radio buttons corresponding to display of internal forces, and of displacements.

---

setEnabledBuildButtons

```
public void setEnabledBuildButtons(boolean b)
```

Enables (or disables) buttons used in construction of structure. It is necessary to disable these buttons when in one of the view modes other than build mode.

---

setAddLoadDialogUnitsLabel

```
public void setAddLoadDialogUnitsLabel(java.lang.String s)
```

Sets the units label on the dialog corresponding to the addition of loads to String s.

---

setModStructureDialogUnits

```
public void setModStructureDialogUnits(java.lang.String s)
```

Sets the units label on the dialog corresponding to structure modification to String s.

---

formatNumber

```
public double formatNumber(double d,
                           int n)
```

This method takes a double as an argument and formats it to n positions after the decimal point (accounts for scientific notation as well).

---

setGrid

```
public void setGrid(boolean b,
                    int n)
```

Calls method of same name on local copy of DrawArea.

---

displayNodeMemberNumbe

```
public void displayNodeMemberNumbers(boolean b)
```

Calls method of same name on local copy of DrawArea.

setStickyMode

```
public void setStickyMode()
```

This method sets the sticky mode based on the menu item state.

setStickyMode

```
public void setStickyMode(boolean b)
```

This method takes a boolean b as an argument sets the the sticky mode according to b and updates the menu item accordingly.

setURL

```
public void setURL(java.net.URL u)
```

Sets the local URL to be u (used for setting the target for the instructions page in the help menu).

getMessage

```
public java.lang.String getMessage()
```

Returns the contents of the Message Window.

setMessage

```
public void setMessage(java.lang.String s)
```

Sets the Message Window TextArea text to be String s.

undo

```
public void undo()
```

Calls the undo() method on the local DrawArea object and performs all necessary operations on UI objects.

updateMomentInertiaTextField

```
public void updateMomentInertiaTextField()
```

Updates the enabled state and the contents of the textfield corresponding to the custom moment of inertia.

setEnabledUndoMenuItem

```
public void setEnabledUndoMenuItem(boolean b)
```

Enables (or disables) the menu item corresponding to the undo operation.

resetMessage

```
public void resetMessage()
```

Sets the Messgae Window TextArea text to "Messages appear here.".

resetDefaultPrefs

```
public void resetDefaultPrefs()
```

Resets the default preferences: sticky mode off, medium grid spacing, medium snap distance, white background.

closeAddBCDialog

```
public void closeAddBCDialog()
```

Closes the dialog corresponding to adding/modifying boundary conditions.

closeAddLoadDialog

```
public void closeAddLoadDialog()
```

Closes the dialog corresponding to adding/modifying loads.

closeModStructureDialog

```
public void closeModStructureDialog()
```

Closes the dialog corresponding to modifying the structure.

copyData

```
public void copyData()
```

Copies message area contents to System clipboard.

resetAddLoadDialog

```
public void resetAddLoadDialog()
```

Resets the values in the dialog corresponding to adding/modifying loads.

resetModStructureDialog

```
public void resetModStructureDialog()
```

Resets the values in the dialog corresponding to modifying the structure.

resetModStructureNode

```
public void resetModStructureNode()
```

Resets the private class variable representing the Node to be modified in the "modify structure" procedure.

showAboutDialog

```
public void showAboutDialog(boolean b)
```

Shows (or hides) the dialog corresponding to program "about" information.

showSaveDialog

```
public int showSaveDialog()
```

Launches the JFileChooser corresponding to "save" function. Returns according to showSaveDialog() function in JFileChooser class

showOpenDialog

```
public int showOpenDialog()
```

Launches the JFileChooser corresponding to "open" function. Returns according to showSaveDialog() function in JFileChooser class

getFilePath

```
public java.lang.String getFilePath()
```

Returns path of file corresponding to JFileChooser.

launchInstructions

```
public void launchInstructions()
```

Launches the instructions page in a separate browser window.

---

lostOwnership

```
public void lostOwnership(java.awt.datatransfer.Clipboard clip,
                          java.awt.datatransfer.Transferable tr)
```

This method does nothing in this implementation. It is included as a requirement for implementing the ClipboardOwner interface.
Specified by:
`lostOwnership` in interface `java.awt.datatransfer.ClipboardOwner`

---

setBC

```
public void setBC(Node n,
                  int mode)
```

Calls the method of the same name on the local DrawArea object and updates the GUI accordingly.

---

setBGWhite

```
public void setBGWhite()
```

Calls the method of the same name on the local DrawArea object and updates the GUI accordingly.

---

setBGBlack

```
public void setBGBlack()
```

Calls the method of the same name on the local DrawArea object and updates the GUI accordingly.

---

getEValueLabel

```
public double getEValueLabel()
                      throws java.lang.NumberFormatException
```

Returns a double representation of the Young's modulus corresponding to current material.

---

getScaleTextField

```
public double getScaleTextField()
                         throws java.lang.NumberFormatException
```

Returns a double representation of the current user-defined scale in pixels. The user specifies the scale in grid blocks, so this method performs all necessary conversion. Note that all modifications to allow the user to specify grid blocks (as opposed to pixels) have been made within the getScaleTextField and setScaleTextField methods of this class.

---

getMomentInertiaTextField

```
public double getMomentInertiaTextField()
                                 throws java.lang.NumberFormatException
```

Returns a double representation of the current user-defined moment of inertia.

---

getMomentInertiaType

```
public java.lang.String getMomentInertiaType()
```

Returns a String representation of the current user-defined moment of inertia type.

---

setMomentInertiaTextField

```
public void setMomentInertiaTextField(java.lang.String s)
```

Sets the moment of inertia text field to String s.

---

getAreaTextField

```
public double getAreaTextField()
                        throws java.lang.NumberFormatException
```

Returns a double representation of the current user-defined area.

---

setAreaTextField

```
public void setAreaTextField(java.lang.String s)
```

Sets the area text field to String s.

---

getFxLoadFromDialog

```
public double getFxLoadFromDialog()
                           throws java.lang.NumberFormatException
```

Returns a double representation of the current user-defined force in the x-direction (from the dialog corresponding to adding/modifying loads).

---

getFyLoadFromDialo

```
public double getFyLoadFromDialog()
                           throws java.lang.NumberFormatException
```

Returns a double representation of the current user-defined force in the y-direction (from the dialog corresponding to adding/modifying loads).

---

repositionNodeModStructure

```
public void repositionNodeModStructure(java.awt.Point p)
```

Calls modifyStructure() method on local DrawArea object based on previously selected Node. Makes appropriate adjustments to the GUI.

---

getModStructureNode

```
public Node getModStructureNode()
```

Returns the Node currently selected to be modified.

---

getXModStructureDialog

```
public double getXModStructureDialog()
                              throws java.lang.NumberFormatException
```

Returns a double representation of the current user-defined x coordinate from the dialog corresponding to modifying the structure.

---

getYModStructureDialog

```
public double getYModStructureDialog()
                              throws java.lang.NumberFormatException
```

Returns a double representation of the current user-defined y coordinate from the dialog corresponding to modifying the structure.

---

setFxLoadFromDialog

```
public void setFxLoadFromDialog(java.lang.String s)
```

Sets the force in the x-direction in the dialog corresponding to adding/modifying loads.

---

setFyLoadFromDialog

```
public void setFyLoadFromDialog(java.lang.String s)
```

Sets the force in the y-direction in the dialog corresponding to adding/modifying loads.

---

setXModStructureDialog

```
public void setXModStructureDialog(java.lang.String s)
```

Sets x coordinate in the dialog corresponding to modifying the structure to String s.

---

setYModStructureDialog

```
public void setYModStructureDialog(java.lang.String s)
```

Sets y coordinate in the dialog corresponding to modifying the structure to String s.

---

setLoad

```
public void setLoad(Node n,
                    double fx,
                    double fy)
```

Calls the method of the same name on the local DrawArea object and updates the GUI accordingly.

---

setEValueLabel

```
public void setEValueLabel(java.lang.String s)
```

Sets the E label associated with the material drop-down to be the String s.

---

setScaleTextField

```
public void setScaleTextField(double d)
```

Converts d from pixels to gridBlocks (based on the grid spacing), converts this double to a String, and displays the String in the scale text field. Note that all modifications to allow the user to specify grid blocks (as opposed to pixels) have been made within the getScaleTextField and setScaleTextField methods of this class.

---

setEUnitsLabel

```
public void setEUnitsLabel(java.lang.String s)
```

Sets the units label associated with the material drop-down to be the String s.

---

setScaleUnitsLabel

```
public void setScaleUnitsLabel(java.lang.String s)
```

Sets the units label associated with the scale text field to be the String s.

---

setAreaUnitsLabel

```
public void setAreaUnitsLabel(java.lang.String s)
```

Sets the units label associated with the area text field to be the String s.

---

setMomentInertiaUnitsLabel

```
public void setMomentInertiaUnitsLabel(java.lang.String s)
```

Sets the units label associated with the moment of inertia text field to be the String s.

---

setSnapDistance

```
public void setSnapDistance(int n)
```

Calls the method of the same name on the local DrawArea object.

---

setNoGridMenuItem

```
public void setNoGridMenuItem()
```

Sets the selection states for grid menu items to indicate that the grid is off.

---

setSmallGridMenuItem

```
public void setSmallGridMenuItem()
```

Sets the selection states for grid menu items to indicate that the current grid spacing is "small".

---

setMediumGridMenuItem

```
public void setMediumGridMenuItem()
```

Sets the selection states for grid menu items to indicate that the current grid spacing is "medium".

---

setLargeGridMenuItem

```
public void setLargeGridMenuItem()
```

Sets the selection states for grid menu items to indicate that the current grid spacing is "large".

---

getGridMenuItemSpacin

```
public int getGridMenuItemSpacing()
```

Returns an integer representation of the current grid spacing based on the state of the grid menu items and on the public static final spacing values defined as in TorsusController.

---

setNearSnapMenuItem

```
public void setNearSnapMenuItem()
```

Sets the selection states for snap distance menu items to indicate that the current snap distance is "near".

---

setMediumSnapMenuItem

```
public void setMediumSnapMenuItem()
```

Sets the selection states for snap distance menu items to indicate that the current snap distance is "medium".

---

setFarSnapMenuItem

```
public void setFarSnapMenuItem()
```

Sets the selection states for snap distance menu items to indicate that the current snap distance is "far".

---

getMagFactor

```
public int getMagFactor()
                 throws java.lang.NumberFormatException
```

Returns a integer representation of the current magnification factor (for displaying the displaced structure).

---

getMaterialType

```
public java.lang.String getMaterialType()
```

Returns a String representation of the current material type.

---

getUnitsType

```
public java.lang.String getUnitsType()
```

Returns a String representation of the current unit system.

---

resetScreen

public void **resetScreen**()

Calls the resetSceen() method on the local DrawArea object and resets the GUI components accordingly.

---

refresh

public void **refresh**()

Calls the method of the same name on the local DrawArea object.

---

paintStructureOnly

public void **paintStructureOnly**()

Calls the method of the same name on the local DrawArea object.

---

paintDisplacedNodes

public void **paintDisplacedNodes**(TrussStructure t)

Calls the method of the same name on the local DrawArea object.

---

paintInternalForces

public void **paintInternalForces**(TrussStructure t)

Calls the method of the same name on the local DrawArea object.

---

setDrawMode

public void **setDrawMode**(int mode)

Sets the current draw mode.

---

mousePressed

public void **mousePressed**(java.awt.event.MouseEvent e)

Handles mousePressed MouseEvents.
Specified by:
mousePressed in interface java.awt.event.MouseListener

---

mouseRelease

```
public void mouseReleased(java.awt.event.MouseEvent e)
```

Handles mouseReleased MouseEvents.

Specified by:
`mouseReleased` in interface `java.awt.event.MouseListener`

---

mouseEntered

```
public void mouseEntered(java.awt.event.MouseEvent e)
```

Empty method. Included since this class implements the MouseListener interface.

Specified by:
`mouseEntered` in interface `java.awt.event.MouseListener`

---

mouseExited

```
public void mouseExited(java.awt.event.MouseEvent e)
```

Empty method. Included since this class implements the MouseListener interface.

Specified by:
`mouseExited` in interface `java.awt.event.MouseListener`

---

mouseClicked

```
public void mouseClicked(java.awt.event.MouseEvent e)
```

Empty method. Included since this class implements the MouseListener interface.

Specified by:
`mouseClicked` in interface `java.awt.event.MouseListener`

---

resetClassNodeVariables

```
public void resetClassNodeVariables()
```

This method resets all (private) class Node variables. It is used because operations can be aborted part-way through the procedure.

---

keyTyped

```
public void keyTyped(java.awt.event.KeyEvent e)
```

Empty method. Included since this class implements the KeyListener interface.

Specified by:
`keyTyped` in interface `java.awt.event.KeyListener`

---

keyPressed

```
public void keyPressed(java.awt.event.KeyEvent e)
```

Empty method. Included since this class implements the KeyListener interface.
Specified by:
`keyPressed` in interface `java.awt.event.KeyListener`

---

keyReleased

```
public void keyReleased(java.awt.event.KeyEvent e)
```

Handles the key released events.
Specified by:
`keyReleased` in interface `java.awt.event.KeyListener`

Class Tree Deprecated Index Help

PREV CLASS NEXT CLASS **FRAMES NO FRAMES**
SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Class **Tree** **Deprecated** **Index** **Help**

PREV CLASS **NEXT CLASS** **FRAMES** **NO FRAMES**

SUMMARY: INNER | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

torsusapplet

Class TrussStructure

```
java.lang.Object
  |
  +--torsusapplet.TrussStructure
```

All Implemented Interfaces:

java.io.Serializable

public class **TrussStructure**
extends java.lang.Object
implements java.io.Serializable

Class that stores all information about a particular structure in terms of Nodes and Vectors (stores a Vector each).

See Also:

Serialized Form

Constructor Summary

`TrussStructure``()` Initializes Vector of Nodes and Members to be new, empty Vectors.
`TrussStructure``(java.util.Vector nodes, java.util.Vector members)` Initializes Vector of Nodes and Members to arguments passed to constructor.

Method Summary

`void`	**`addMember`**`(Member m)` Adds a member to the Vector of Members, if it does not already exist.
`void`	**`changeLoad`**`(int force_x, int force_y, int nodeID)` Updates the loads corresponding to a Node based on ID number.
`boolean`	**`existsMember`**`(Member m)` Determines whether Member already belongs to TrussStructure.
`double`	**`getArea`**`()` Returns the cross-sectional area of the structure.
`java.util.Vector`	**`getMembers`**`()` Returns internal Vector of Members.
`java.util.Vector`	**`getNodes`**`()` Returns internal Vector of Nodes.
`double`	**`getScale`**`()` Returns scale (user-defined).
`int`	**`getSnapDistance`**`()` Returns snap distance, which is the distance (in pixels) at which one Node will graphically snap to another.
`double`	**`getTrussMass`**`()` Returns the mass of the structure.
`double`	**`getYoungsMod`**`()`

	Returns Young's modulus (E).
boolean	**interval**(double x, double a, double b) Determines whether x falls in the interval [x,y] where x = min(a,b) and y = max(a,b).
void	**modifyStructure**(Node n, java.awt.Point p) Changes the position of Node n to Point p.
Member	**nearMember**(java.awt.Point p, int e) Determines whether a Point p is within a distance e (epsilon) of any of the Members (used primarily for the graphical removal of Members).
Node	**nearNode**(java.awt.Point p, int e) Determines whether a Point p is within a distance e (epsilon) of any of the Nodes (for graphical change of boundary conditions and loads).
double	**pointDistance**(java.awt.Point a, java.awt.Point b) Measures the distance between any two Points a and b.
void	**removeMember**(Member m) Removes a member from the Vector of Members.
void	**setArea**(double d) Sets the cross-sectional area of the structure.
void	**setBC**(Node n, int bc_type) Updates the boundary conditions corresponding to a Node belonging to the structure.
void	**setLoad**(Node n, double fx, double fy) Updates the loads corresponding to a Node based on Node class isEqual() method.
void	**setMembers**(java.util.Vector v) Sets internal Vector of Members.
void	**setNodes**(java.util.Vector v) Sets internal Vector of Nodes.
void	**setScale**(double d) Sets scale (user-defined).
void	**setSnapDistance**(int n) Sets snap distance, which is the distance (in pixels) at which one Node will graphically snap to another.
void	**setTrussMass**(double d) Set the mass of the structure.
void	**setYoungsMod**(double d) Sets Young's modulus (E).

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

TrussStructure

```
public TrussStructure()
```

Initializes Vector of Nodes and Members to be new, empty Vectors.

TrussStructure

```
public TrussStructure(java.util.Vector nodes,
                      java.util.Vector members)
```

Initializes Vector of Nodes and Members to arguments passed to constructor.

Method Detail

addMember

```
public void addMember(Member m)
```

Adds a member to the Vector of Members, if it does not already exist. Updates the Vector of Nodes accordingly. Note that this method does not allow the user to create multiple Members between Nodes in this implementation of this method.

---

removeMember

```
public void removeMember(Member m)
```

Removes a member from the Vector of Members. Updates the Vector of Nodes accordingly. This method also handles the removal of "floating nodes", Nodes which - upon removal of a Member - no longer belong to any Member.

---

changeLoad

```
public void changeLoad(int force_x,
                       int force_y,
                       int nodeID)
```

Updates the loads corresponding to a Node based on ID number.

---

setBC

```
public void setBC(Node n,
                  int bc_type)
```

Updates the boundary conditions corresponding to a Node belonging to the structure.

---

setLoad

```
public void setLoad(Node n,
                    double fx,
                    double fy)
```

Updates the loads corresponding to a Node based on Node class isEqual() method.

---

getNodes

```
public java.util.Vector getNodes()
```

Returns internal Vector of Nodes.

---

setNodes

```
public void setNodes(java.util.Vector v)
```

Sets internal Vector of Nodes.

setMembers

```
public void setMembers(java.util.Vector v)
```

Sets internal Vector of Members.

getMembers

```
public java.util.Vector getMembers()
```

Returns internal Vector of Members.

existsMember

```
public boolean existsMember(Member m)
```

Determines whether Member already belongs to TrussStructure.

nearMember

```
public Member nearMember(java.awt.Point p,
                         int e)
```

Determines whether a Point p is within a distance e (epsilon) of any of the Members (used primarily for the graphical removal of Members). If such a Member exists, it is returned by the method. Otherwise, the method returns null.

pointDistance

```
public double pointDistance(java.awt.Point a,
                            java.awt.Point b)
```

Measures the distance between any two Points a and b. Called by the nearMember() method.

interval

```
public boolean interval(double x,
                        double a,
                        double b)
```

Determines whether x falls in the interval [x,y] where x = min(a,b) and y = max(a,b). This method is called by the nearMember() method for Member removal. A buffer of 5 pixels is hardcoded in this method to allow for ease of Member removal from user perspective.

---

modifyStructure

```
public void modifyStructure(Node n,
                            java.awt.Point p)
```

Changes the position of Node n to Point p. Makes all appropriate adjustments to the Members associated with this Node.

---

nearNode

```
public Node nearNode(java.awt.Point p,
                     int e)
```

Determines whether a Point p is within a distance e (epsilon) of any of the Nodes (for graphical change of boundary conditions and loads). If such a Node exists, the method returns the nearest Node within distance e (NOT the first Node determined to be within distance e) of Point p. If not, the method returns null.

---

getTrussMass

```
public double getTrussMass()
```

Returns the mass of the structure.

---

setTrussMass

```
public void setTrussMass(double d)
```

Set the mass of the structure.

---

getArea

```
public double getArea()
```

Returns the cross-sectional area of the structure. The program only allows users to specify one cross-sectional area per structure.

---

setArea

```
public void setArea(double d)
```

Sets the cross-sectional area of the structure. The program only allows users to specify one cross-sectional area per structure.

---

getYoungsMod

```
public double getYoungsMod()
```

Returns Young's modulus (E).

---

setYoungsMod

```
public void setYoungsMod(double d)
```

Sets Young's modulus (E).

---

getScale

```
public double getScale()
```

Returns scale (user-defined).

---

setScale

```
public void setScale(double d)
```

Sets scale (user-defined).

---

getSnapDistance

```
public int getSnapDistance()
```

Returns snap distance, which is the distance (in pixels) at which one Node will graphically snap to another. This snapDistance also determines the graphical tolerance for the user in modifying Node properties.

---

setSnapDistance

```
public void setSnapDistance(int n)
```

Sets snap distance, which is the distance (in pixels) at which one Node will graphically snap to another. This snapDistance also determines the graphical tolerance for the user in modifying Node properties.

Class **Tree** **Deprecated** **Index** **Help**

PREV CLASS NEXT CLASS   FRAMES NO FRAMES

SUMMARY: INNER | FIELD | CONSTR | METHOD   DETAIL: FIELD | CONSTR | METHOD

torsusapplet

Class UnstableStructureException

```
java.lang.Object
  |
  +--java.lang.Throwable
        |
        +--java.lang.Exception
              |
              +--torsusapplet.UnstableStructureException
```

All Implemented Interfaces:
java.io.Serializable

public class **UnstableStructureException**
extends java.lang.Exception

This class extends the functionality of Exception and is used to allow the controller to determine sources of computation error in "model" of model-view-controller paradigm.

See Also:
Serialized Form

Constructor Summary
`UnstableStructureException()`

Methods inherited from class java.lang.Throwable
`fillInStackTrace, getLocalizedMessage, getMessage, printStackTrace, printStackTrace, printStackTrace, toString`

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait`

Constructor Detail

UnstableStructureException

```
public UnstableStructureException()
```

Appendix D

Textbook-Style Derivations: Truss Structures

Module 5

TRUSS STRUCTURES

5.1 Introduction

A truss is a structure constructed of slender members, joined at the ends of these members by "frictionless" pins, and loaded only at the joints. Truss structures carry loads via axial forces (tensile and compressive) in the members; the members do not resist shear or moment loading. Each truss member is a two-force member (refer to Two-Force Members for more details). Truss structures provide for good stiffness and strength over large expanses at a minimum weight. Trusses, for example, are used in the construction of large crane arms (Figure 5.1), where building a solid arm would be impractical because of its weight.

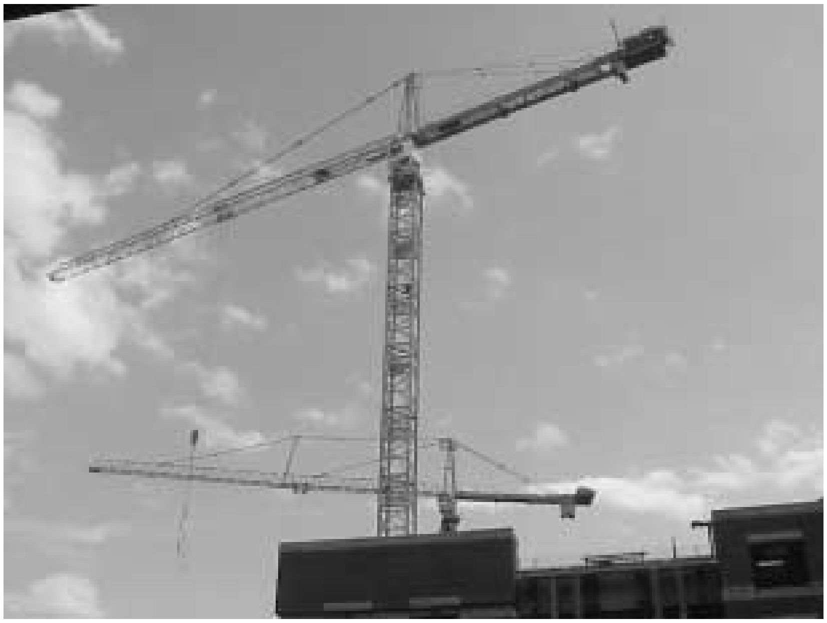


Figure 5.1 Construction cranes near Haymarket in Boston make use of truss structures.

Other common examples of trusses are in roof supports, aircraft wings, radio towers, and bridges. Some common roof and bridge truss designs are shown in Figure 5.2 and Figure 5.3.

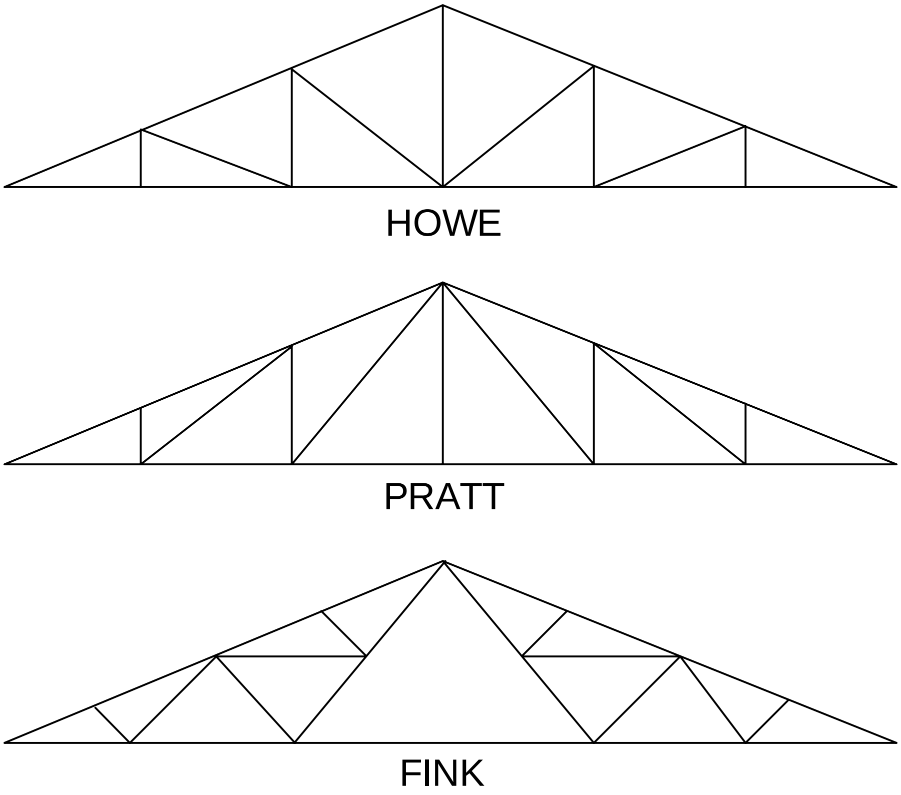


Figure 5.2 Common roof truss designs. [Taken from Laursen, Harold I. *Structural Analysis*. McGraw Hill, New York: 1978.]

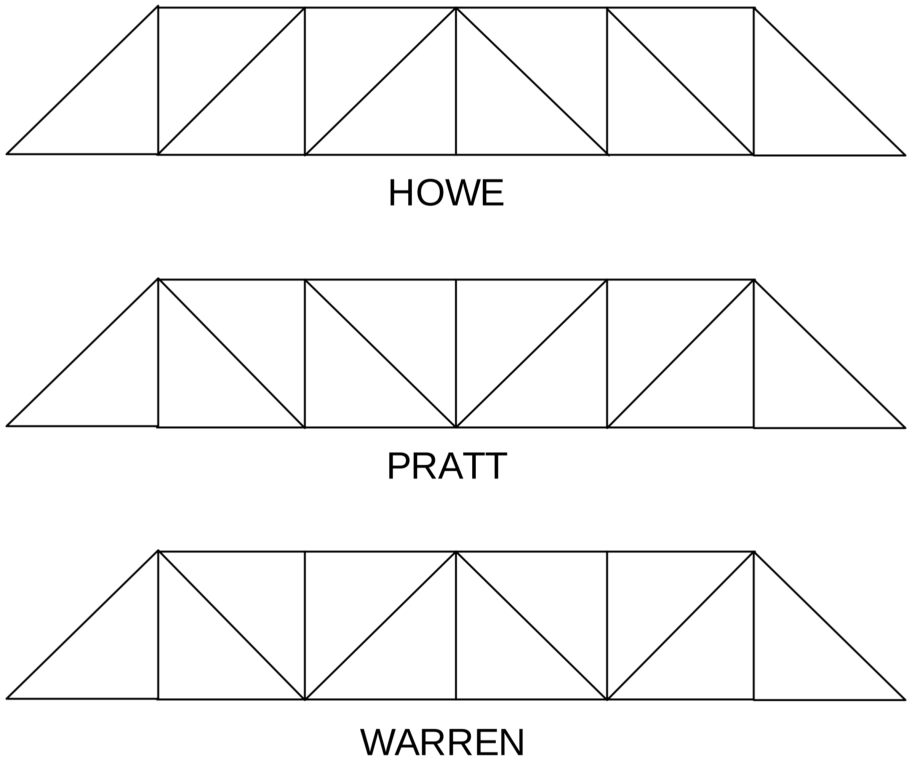


Figure 5.3 Common bridge truss designs. [Taken from Laursen, Harold I. *Structural Analysis.* McGraw Hill, New York: 1978.]

Although the members of a truss structure do not resist shear forces and bending moments, the structure itself is designed to carry these loads in a manner analogous to a beam. For example, when a bridge truss structure is subject to a distributed load (for example, cars on a bridge, as shown in Figure 5.4), the moment at any section of a truss is carried by compressive loads of the top members and tensile loading of the bottom members; the shear loading is carried by the cross-members. A detailed example of this will be given later (How a Truss Structure Carries Shear and Moment Loads).

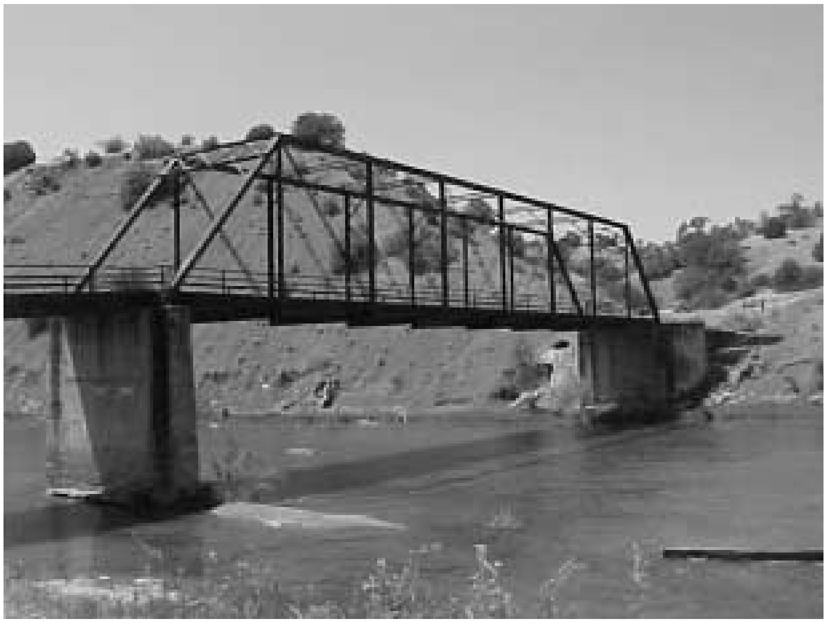


Figure 5.4 A truss bridge in Glenn County, CA. [Taken from *Truss Bridges of Northern California*: http://community-2.webtv.net/DAROSAMAR-CIELII/TrussBridgesOf, October 2001.]

Though the origin of truss structures is unclear, they appear to have been used as early as 2500 B.C. for primitive lake dwellings (Brittanica.com). The Chinese are also known to have constructed wooden truss bridges during this period (Palmes). Timber trusses appeared extensively in Greek architecture and were used mainly for roofing purposes. Famous Italian architect Andrea Palladio, in *I quattro libri dell'architettura* (1570; *Four Books on Architecture*), presented written plans for timber trusses, stating that the diagonal members “support the whole work.”; however, a solid theoretical understanding of the behavior of trusses was not developed until the 19th century. During the Industrial Revolution, builders introduced iron trusses, and eventually, iron was replaced by steel. Today trusses are made from a variety of materials, including plastics and composites.

5.2 Plane Truss: Definitions and Assumptions

The trusses presented in this module are plane trusses, structures where the forces and deformations only occur in one plane. The methods used, however, can be generalized to the three-dimensional case.

We make several assumptions in analyzing a plane truss structure.

- First, the members are only connected to one another at their ends.
- Second, the connection points must be frictionless. This prevents moments from being applied at the joints, and helps ensure the condition of only axial-loading of members.
- Third, the applied loads all occur at the joints.

Note that these three conditions enforce each truss member to act as a two-force member. As a result, truss members will only experience axial loading.

In the analyses presented here, structural weight is neglected. Though weight is generally insignificant for smaller structures, a larger structure can be impacted by the weight of its constituent members.

Here we introduce the "degree of freedom" (d.o.f.). The number of degrees of freedom of a joint is defined as the number of independent motions possible and thus the number of independent equations necessary to define its position at any given time. For the two-dimensional case, a "free" joint can translate in the x- and y- directions and has two degrees of freedom. Similarly, a joint constrained in the x-direction has one degree of freedom, which is translation in the y-direction; a joint constrained in the y-direction has one degree of freedom, translation in the x-direction; and a fixed joint has zero degrees of freedom.

Truss structures must be stable. An unstable truss structure is one which can experience rigid body motion (i.e. - motion without the deformation of individual members). In other words, the structure (or part of the structure) can be moved without resistance from the structural elements. Such a “structure” is undesirable for obvious reasons. Figure 5.5 shows an unstable truss. It is shown in two unique states; note that the members remain of the same length in both states.

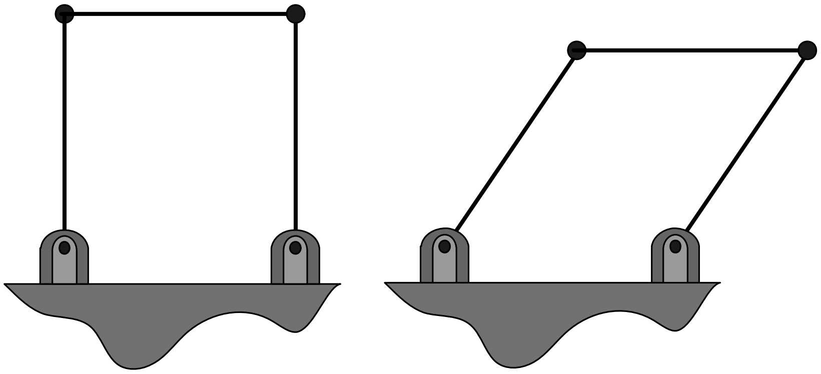


Figure 5.5 An unstable structure is shown in two unique states. Note that no member in either state has deformed.

A triangular truss (or a truss structure constructed of triangular elements) is stable. For that reason, it is a very common repeating entity in truss construction. An example of the use of triangular elements in truss construction is illustrated in the photograph of a billboard support structure (Figure 5.6) and was also evident in the roof and bridge designs of Figure 5.2 and Figure 5.3.

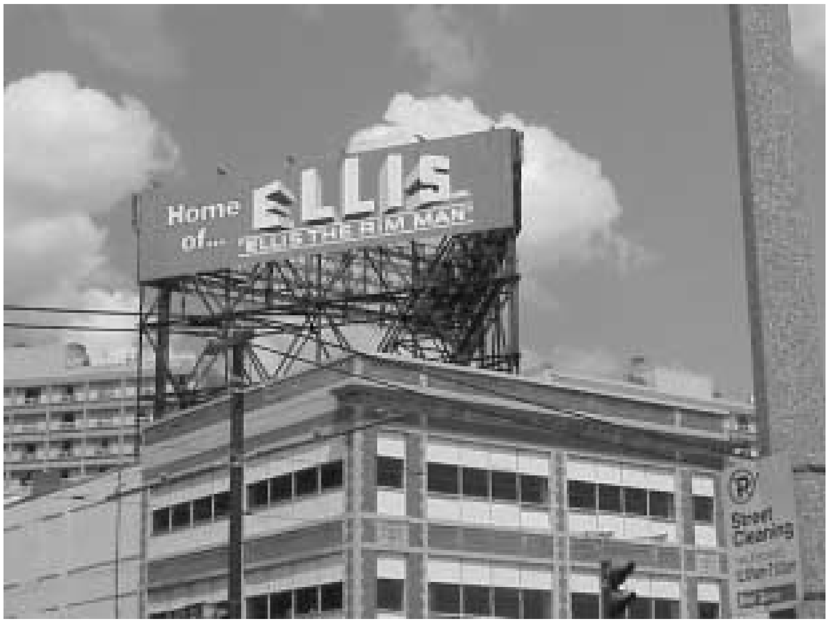


Figure 5.6 This billboard on Commonwealth Ave. utilizes triangular truss elements.

Next, we introduce the topic of static determinacy.

A statically determinate system is one where the equations of static equilibrium are sufficient to solve for the system forces. Conversely, a statically indeterminate system is one where the equations of static equilibrium are insufficient to solve for the system forces. In other words, if the number of unknown forces exceeds the number of independent equilibrium equations, the structure is statically indeterminate. It is necessary to incorporate elements of the system deflections in order to determine the forces in the case of statical indeterminacy.

To determine the instability of a system and to determine whether the system is statically determinate, one can carefully compare aspects of the structure geometry to its constraints (Strang).

Let us define the following quantities

- N is the total number of joints in the system
- r is the number of reactions in the system
- m is the number of members in the system
- n = 2N - r

If m < n, the system is unstable. If m is greater than or equal to n, there is no guarantee that the system is stable. If m = n, we know that the system is statically determinate. If m > n, the system is statically indeterminate. These rules are described in the table below.

TABLE 5.1 Stability and static determinacy of different structures. [Taken from Strang, Gilbert. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, Wellesley, MA: 1986.]

Condition	**Stability**	**Static Determinacy**
m < n	unstable	n/a
m = n	unknown	statically determinate
m > n	unknown	statically indeterminate

There are sound reasons for why this procedure works. Let us consider the quantity m + r. This quantity represents the total number of unknown forces in the system: r is the total number of reaction forces and m represents the total number of unknown member forces. We can compare this number of unknown forces to the number of independent equilibrium equations in order to determine whether or not the system is statically determinate. The number of independent equilibrium equations is exactly equal to 2 times the number of joints, or 2N. This is because there are 2 equilibrium equations at each joint ($\Sigma F_x = 0$, $\Sigma F_y = 0$). If m + r is equal to 2N, the system will be statically determinate.

We have

$$m + r = 2N \qquad \text{(A.1)}$$

which simplifies to

$$m = 2N - r = n \qquad \text{(A.2)}$$

In other words, if m = n, the system is statically determinate. Similarly, we expect that if the number of independent equilibrium equations is less than the number of unknown forces in the structure (m > n), we have insufficient knowledge to solve for the system forces. This implies a case of a statically indeterminate structure. Finally, if the number of independent equilibrium equations is greater than the number of unknown forces (m < n), we have a problem which may not have a solution; the result is an unstable structure.

Let us look at a few examples to demonstrate this procedure.

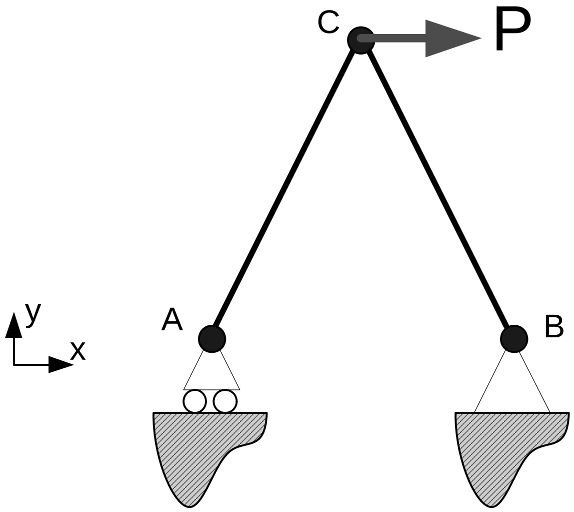


Figure 5.7 A 2 member truss subjected to a load P in the x-direction.

In the example shown in Figure 5.7, we obtain the following values:

TABLE 5.2 Values of quantities for determining static determinacy, example from Figure 5.7.

Quantity	**Value**
N (number of joints)	3
r (number of reactions)	3
m (number of members)	2
n (=2N-r)	3

Since m < n, this system is unstable. This system is completely free to displace in the x-direction.

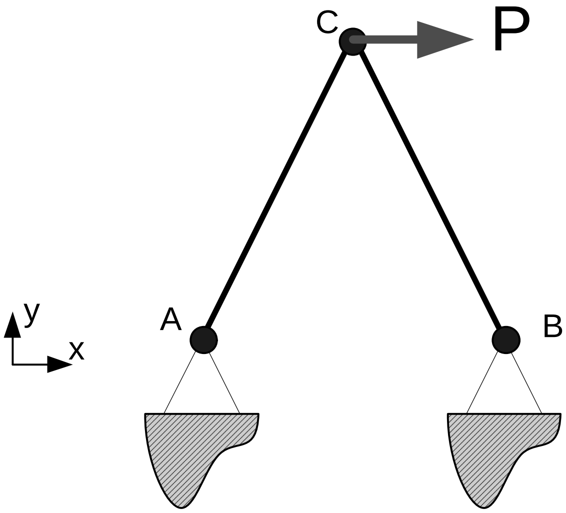


Figure 5.8 Another 2 member truss subjected to a load P in the x-direction.

In the example shown in Figure 5.8, we obtain the following values:

TABLE 5.3 Values of quantities for determining static determinacy, example from Figure 5.8.

Quantity	Value
N (number of joints)	3
r (number of reactions)	4
m (number of members)	2
n (=2N-r)	2

Since m = n, this system is statically determinate.

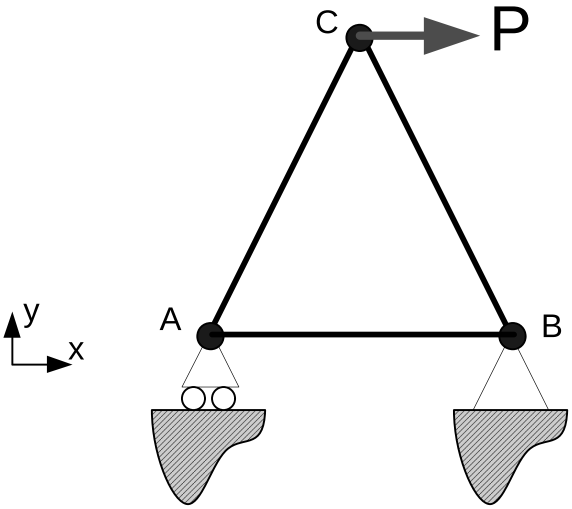


Figure 5.9 A 3 member truss subjected to a load P in the x-direction.

In the example shown in Figure 5.9, we obtain the following values:

TABLE 5.4 Values of quantities for determining static determinacy, example from Figure 5.9.

Quantity	Value
N (number of joints)	3
r (number of reactions)	3
m (number of members)	3
n (=2N-r)	3

Since m = n, this system is also statically determinate.

In conclusion, it is essential to note, that though many idealizations are made in truss analysis, real structures behave closely to the theoretical models. This knowledge allows us to proceed in analyzing truss structures with great confidence, provided that we can correctly identify them.

5.3 Internal Forces

The procedure for solving for internal forces in truss systems is a basic one; however, the procedure presented in this section will only solve for forces for statically determinate systems. The entire procedure is based on the concept of force balances. Simply put, by balancing forces at the joints, the forces acting at all joints and in all members in the system can be found. Some texts refer to this method as the “Method of Joints”. A simple example of finding internal forces follows (Figure 5.10). The structure shown is comprised of two members, identical in material and geometry.

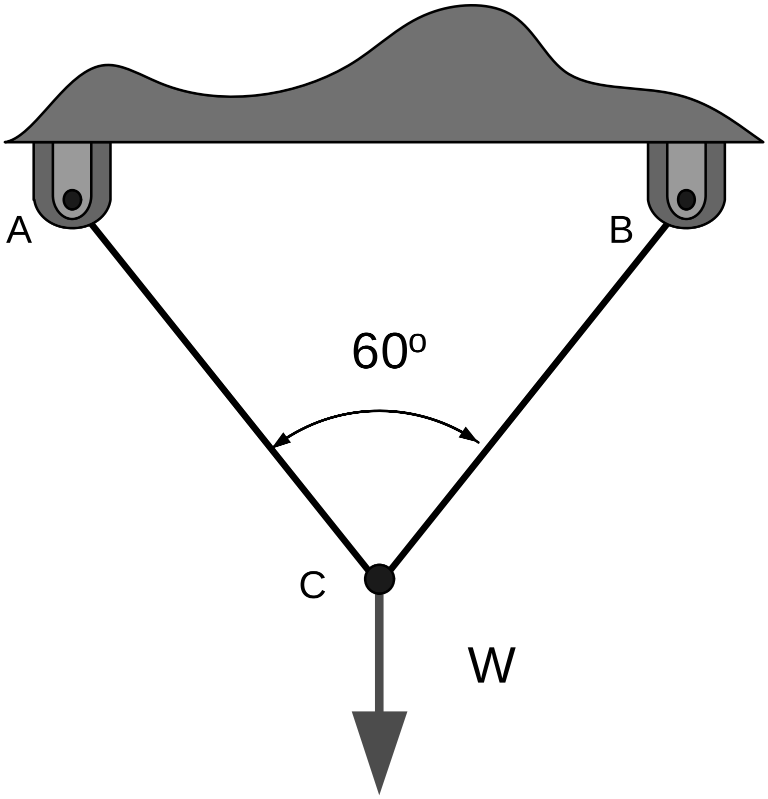


Figure 5.10 This symmetric, 2-member truss is subjected to a downward load of magnitude W.

To determine the forces acting in the members, we analyze joint C (Figure 5.11).

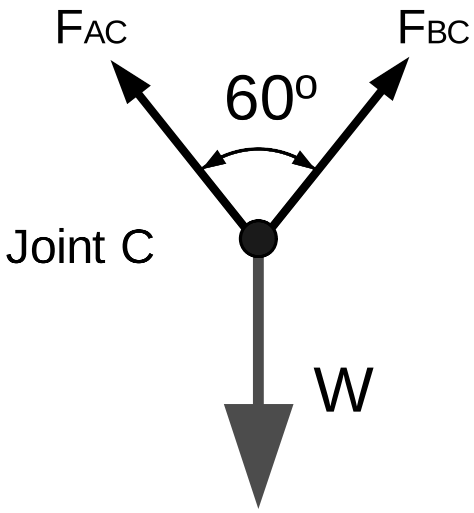


Figure 5.11 Balancing the forces at Joint C yields the forces in elements AC and BC.

Balancing forces in the x and y directions yields the forces acting in members AC and BC.

$$\Sigma F_x = 0 \tag{5.1}$$

$$-F_{AC}\cos 60° + F_{BC}\cos 60° = 0 \tag{5.2}$$

$$F_{AC} = F_{BC} \tag{5.3}$$

$$\Sigma F_Y = 0 \tag{5.4}$$

$$F_{AC}\cos 30° + F_{BC}\cos 30° = W \tag{5.5}$$

We find that

$$F_{AC} = F_{BC} = \frac{W}{\sqrt{3}} \tag{5.6}$$

This implies that the forces in members F_{AC} and F_{BC} are tensile. The previous figure (Figure 5.11) displays the forces with which the joint acts upon the members. The resulting force within the member will be equal in magnitude and opposite in direction. The direction in which the forces are drawn implies a tensile force in the two members see (Figure 5.12).

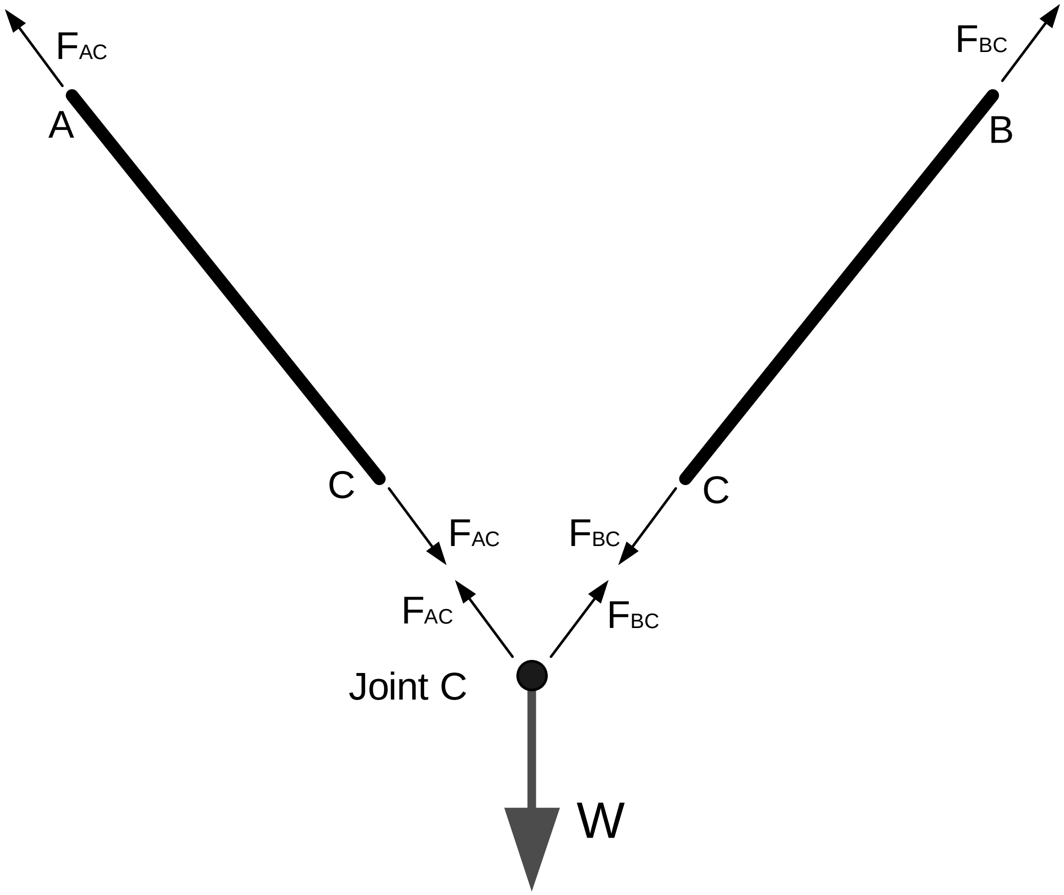


Figure 5.12 The forces acting on Joint C and the members AC and BC are shown above.

Now, let's examine a more complex example. The structure shown below is constructed of aluminum members with identical cross sections, but varying lengths (Figure 5.13). The structure is subjected to an end load W.

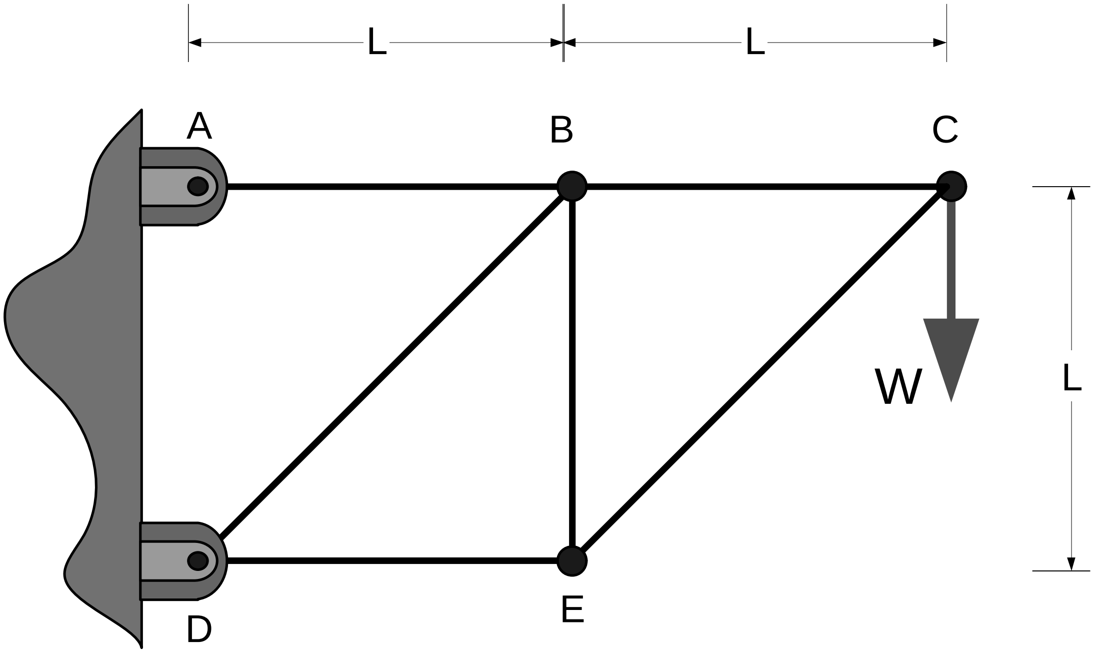


Figure 5.13 This 6-member planar truss is subjected to a downward load of magnitude W at Joint C.

We apply the same procedure to determine the internal forces in the six elements. We balance forces beginning at the joints, beginning with joint C (Figure 5.14).

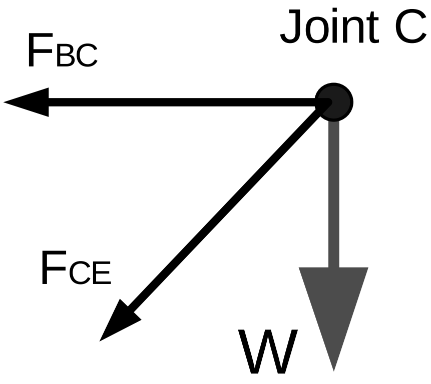


Figure 5.14 Balancing the forces at Joint C yields the forces in elements BC and CE.

Balancing forces in the x and y directions yields the forces acting in members BC and CE.

$$\Sigma F_x = 0 \tag{5.7}$$

$$-F_{BC} - F_{CE}\cos 45° = 0 \tag{5.8}$$

$$\Sigma F_y = 0 \tag{5.9}$$

$$F_{CE}\cos 45° + W = 0 \tag{5.10}$$

We find that

$$F_{CE} = -\sqrt{2}W \tag{5.11}$$

$$F_{BC} = W \tag{5.12}$$

Next we examine joint E (Figure 5.15).

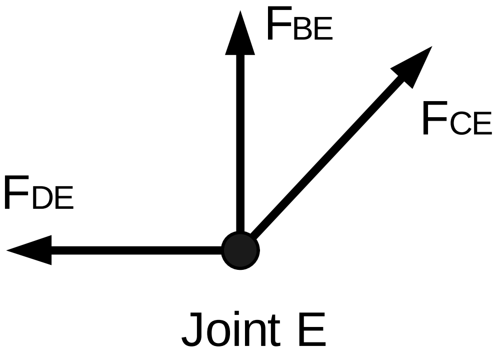


Figure 5.15 Balancing the forces at Joint E yields the forces in elements DE and BE.

Balancing forces in the x and y directions yields the forces acting in members DE and BE.

$$\Sigma F_x = 0 \tag{5.13}$$

$$-F_{DE} + F_{CE}\cos 45° = 0 \tag{5.14}$$

$$\Sigma F_y = 0 \tag{5.15}$$

$$F_{BE} + F_{CE} cos 45° = 0 \tag{5.16}$$

We know that $F_{CE} = -\sqrt{2}W$ from (5.11), so

$$F_{DE} = -W \tag{5.17}$$

$$F_{BE} = W \tag{5.18}$$

Finally, we analyze joint B (Figure 5.16).

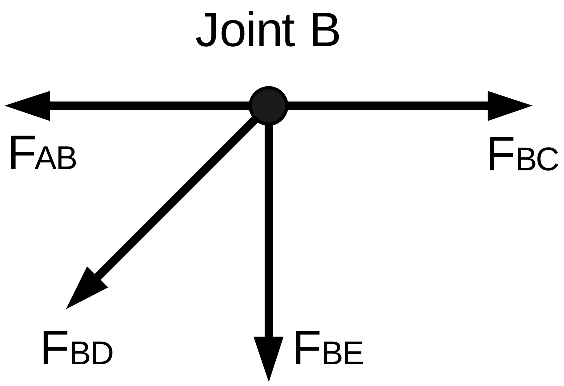


Figure 5.16 Balancing the forces at Joint B yields the forces in elements AB and BD.

Balancing forces in the x and y directions yields the forces acting in members AB and BD.

$$\Sigma F_x = 0 \tag{5.19}$$

$$-F_{AB} + -F_{BD} cos 45° + F_{BC} = 0 \tag{5.20}$$

$$\Sigma F_y = 0 \tag{5.21}$$

$$-F_{BD} cos 45° - F_{BE} = 0 \tag{5.22}$$

We find that

$$F_{AB} = -2W \tag{5.23}$$

$$F_{BD} = -\sqrt{2}W \tag{5.24}$$

The resulting internal forces are displayed in the figure below (Figure 5.17). Compressive forces are displayed in red while tensile forces are displayed in blue.

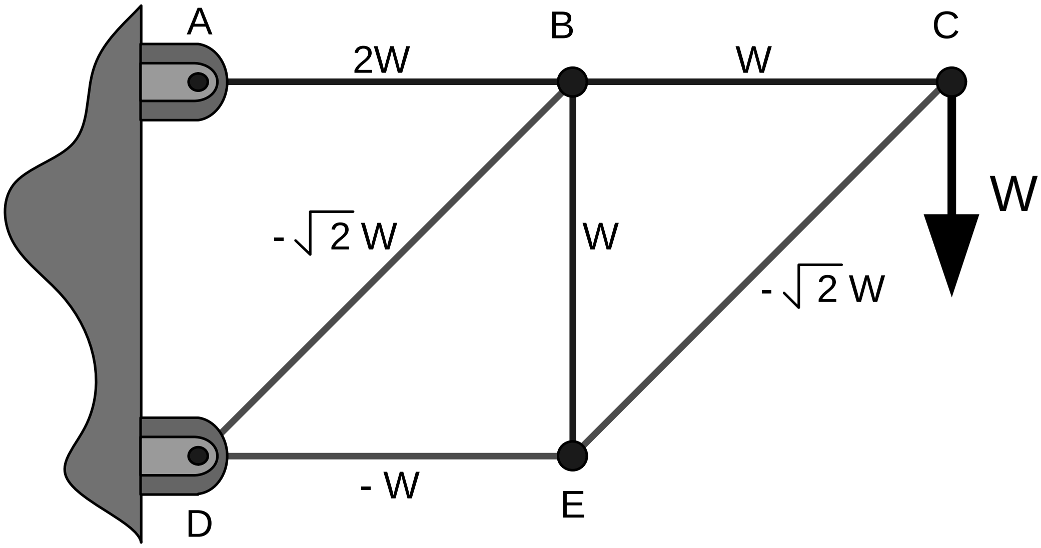


Figure 5.17 The internal forces are displayed above. Compressive forces are displayed in red (negative); tensile forces are displayed in blue (positive).

Note also that once we have solved for the forces in the elements of the truss, we can determine the deformation of the structure. In the next section, we will build on our knowledge to this point.

5.4 Finding Displacements

We know from our study of axial force-deformation that the axial deformation of an axially-loaded bar is given by

$$\delta = \frac{PL}{EA} \tag{5.25}$$

where P represents the axial load, L represents the undeformed length of the bar, E represents the Young’s modulus, and A represents the undeformed cross-sectional area of the bar. Recall that the stress in an axially-loaded bar is given by $\sigma = \frac{P}{A}$, that the strain in the bar is given by $\varepsilon = \frac{\delta}{L}$, and that these are related by $\sigma = E\varepsilon$, where E, the Young's modulus is a property of the material (refer to Material Properties Reference). In general, $\varepsilon \ll 1$ (typically < 0.2%), so we can assume δ to be small relative to L. A schematic indicating these quantities is given in Figure 5.18.

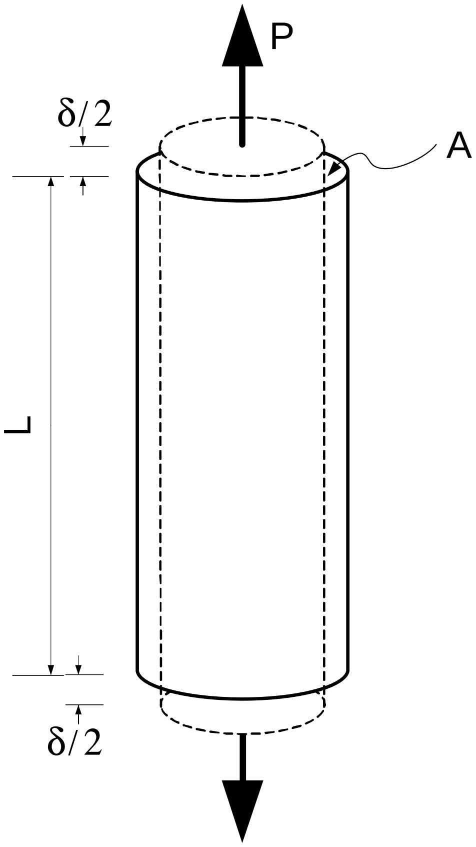


Figure 5.18 A schematic indicating the behavior of an axially-loaded bar. Note that the displacements are exaggerated for purposes of illustration.

Note that in the previous section we did not introduce specific materials or cross-sectional areas; this is because the internal forces of a truss are independent of material and area (within the elastic regime of the material).

Individual truss members are only subjected to axial forces. For the statically determinate structure, we have solved for all forces in all members (issues regarding solutions to statically indeterminate structures will be discussed later). The length, Young's modulus, and cross-sectional areas are also known. Therefore, we can use the force-deformation of Equation (5.25) to determine the axial deformation of each member. The final step in determining the deformed shape of the structure is to use a condition known as compatibility. Compatibility requires that the structure deforms such that it remains together - or compatible - during loading and deformation.

Regardless of what sort of deformation occurs, several things must remain true. Joints will only displace in active degrees of freedom. This means that fixed joints will remain fixed. It also means that joints constrained in the x-direction will only translate in the y-direction (if at all). Similarly, joints constrained in the y-direction will only translate in the x-direction. Finally members that were pinned together prior to deformation will remain pinned after deformation. They will not become disconnected. Using this knowledge, the problem of determining the displacement of the truss system, such that it remains compatible with the deformation of all members, reduces to a problem of geometry. Using an example from the previous section, we will determine the truss deformation.

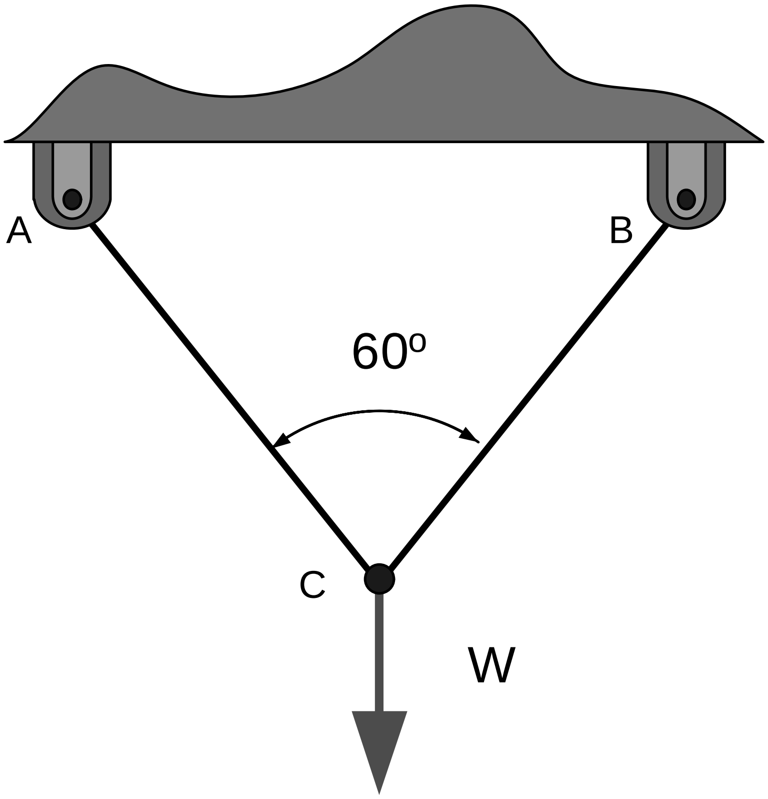


Figure 5.19 This symmetric, aluminum, 2-member truss is subjected to a downward load of magnitude 10,000 N. The members are of length 0.5 m and area 500 sq. mm.

Let us assume that the length L of each of the aluminum members is 0.5m, that the cross-sectional area A of each member is 500 mm^2, that the load W has a magnitude of 10,000 N. We know from the previous section (5.6) that the force in each of the members is $\frac{W}{\sqrt{3}}$. We find that the deflection of each of the members is

$$\delta_{AC} = \delta_{BC} = \delta = \frac{PL}{EA} = \frac{\left(\frac{W}{\sqrt{3}}\right)(0.5)}{(70\times 10^{9})(500\times 10^{-6})} = (8.25\times 10^{-9})W = 8.25\times 10^{-5}m \tag{5.26}$$

We know several characteristics of the deformed structure:

- members AC and BC are each pinned to ground at joints A and B

- each member will elongate by δ
- members AC and BC are pinned to each other at joint C

Compiling this information yields a picture as shown in Figure 5.20. The undeformed structure is represented in green. Each red arc represents the line traced by rotating the elongated element about its upper fixed joint. Note that the deformations have been grossly exaggerated for the purposes of illustration. The point where the elements meet is shown in black. At this point, all conditions of compatibility are met. The members are still pinned at joints A and B. The members are each elongated by δ. The members are still pinned together at joint C. Although joint C has 2 degrees of freedom (x and y translation), it will only displace in the vertical degree of freedom due to the symmetry of the material, geometry, and loading of the problem.

[Note that if A_{AC} and A_{BC} were different, this would not be true. In the case where $A_{AC} < A_{BC}$, for example, the member displacements would be given by

$$\delta_{AC} = \frac{PL}{EA_1} > \delta_{BC} = \frac{PL}{EA_2} \tag{5.27}$$

As a result, the displacement of C would not be strictly vertical.]

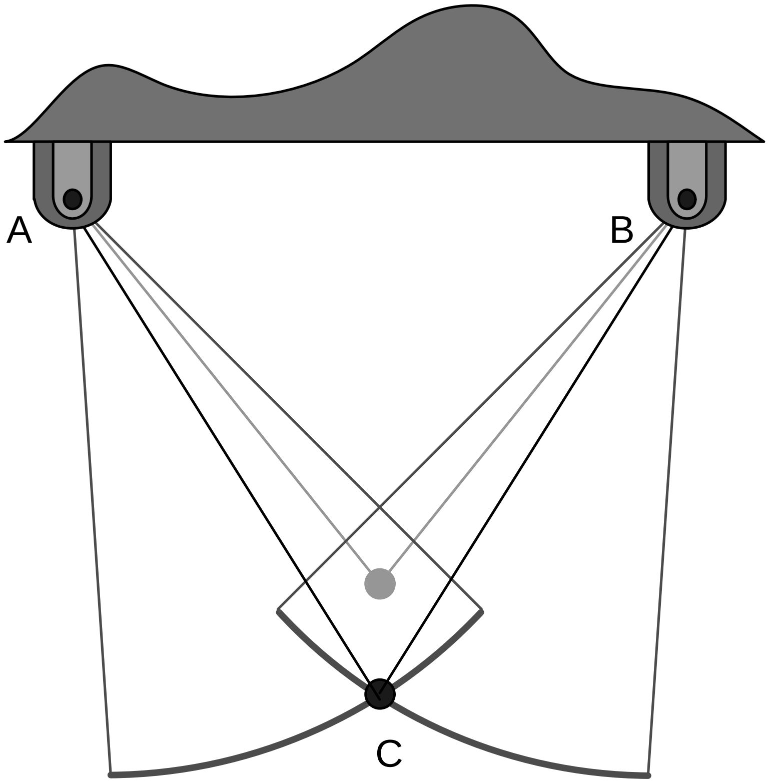


Figure 5.20 Exaggerated deformations are shown for purposes of illustration. The green elements represent the undeformed structure. The red elements represent the range of motion of the deformed elements (with out the constraint of compatibility). The black elements represent the forced compatibility constraint. This is the deformation mode of the structure - the joint deflects vertically.

Using knowledge of the geometry, we can find the vertical deflection of joint C.

We know from (5.26) that

$$\delta_{AC} = \delta_{BC} = 8.25x10^{-5}m \tag{5.28}$$

Since the undeformed length of each of the members is the same, and the deflection of each of the members is the same, we can represent the deformed structure with an isosceles triangle, as shown.

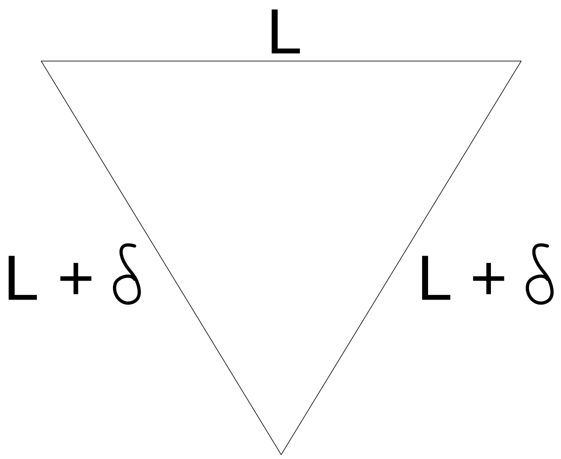


Figure 5.21 This isosceles triangle represents the deformed shape of the aluminum 2-member truss.

We can split the triangle in half by drawing a vertical line from the bottom vertex to the midpoint of the edge with length L.

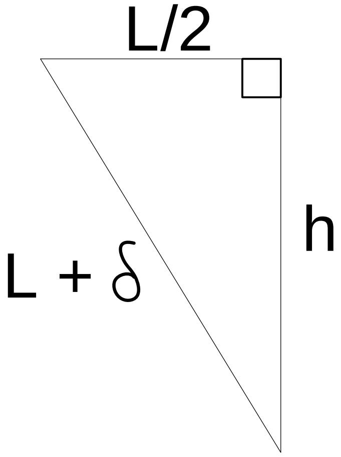


Figure 5.22 This right triangle provides enough information to find the height of the isosceles triangle. This is enough information to find the vertical displacement of the joint.

This will enable us to use the Pythagorean theorem to find the height, h, of the triangle.

$$(h)^2 + \left(\frac{L}{2}\right)^2 = (L + \delta)^2 \tag{5.29}$$

We know that L = 0.5 m and that $\delta = 8.25x10^{-5}m$. This gives us

$$h = 0.4331080m \tag{5.30}$$

The height, u, of the undeformed structure, was

$$u = 0.4330127m \tag{5.31}$$

The difference between these yields the vertical deformation, d, of Joint C:

$$d = 9.53x10^{-5}m \tag{5.32}$$

This is the general procedure we will use in determining the deformation of a system:

- use equilibrium to find internal forces
- use force-deformation to find member deformations
- apply compatibility and use geometric relations to determine system deformation

Now let’s look at a more complex example. Let’s examine the same structure, subjected to a horizontal load W, as shown. Note that the material and geometry in this case are still symmetric, but the loading is not.

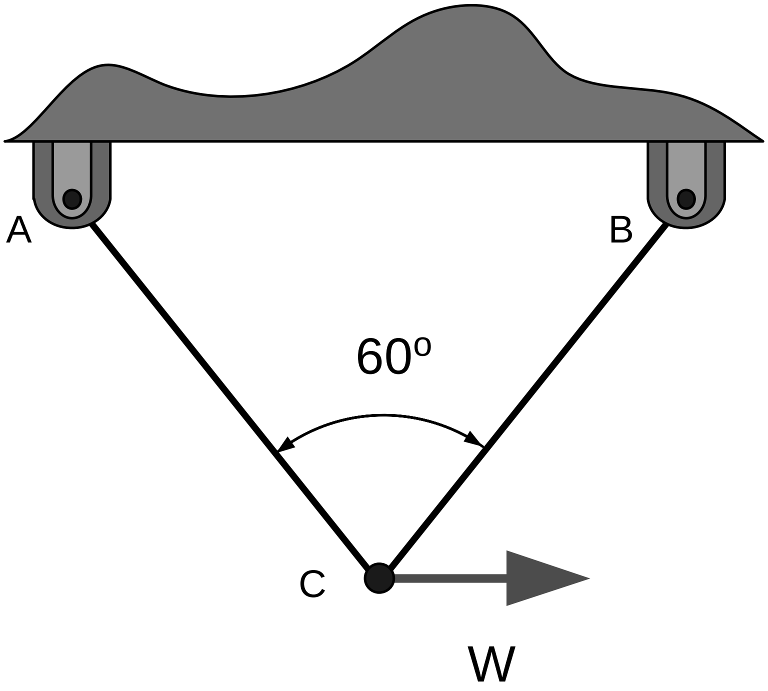


Figure 5.23 This is the same structure as in the previous example; however, it is now subjected to a horizontal load of magnitude W.

Isolating joint C allows us to apply the equations of equilibrium to the problem (Figure 5.24).

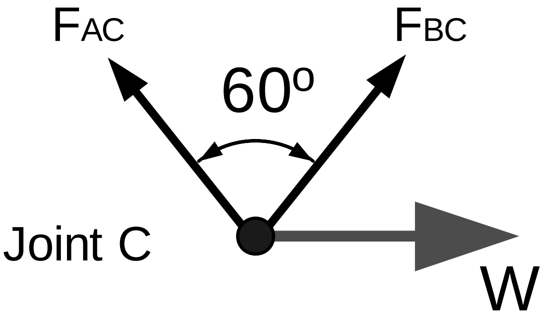


Figure 5.24 A free-body diagram of joint C.

Balancing forces in the x and y directions yields the forces acting in members AC and BC.

$$\Sigma F_x = 0 \tag{5.33}$$

$$-F_{AC} \times \cos 60° + F_{BC} \cos 60° + W = 0 \tag{5.34}$$

$$\Sigma F_y = 0 \tag{5.35}$$

$$F_{AC} \times \cos 30° + F_{BC} \cos 30° = 0 \tag{5.36}$$

$$F_{AC} = -F_{BC} \tag{5.37}$$

From (5.34) and (5.37), we find

$$F_{AC} = -F_{BC} = W \tag{5.38}$$

We find, in other words, that member AC is subjected to a tensile load of magnitude W and that member BC is subjected to a compressive load of magnitude W. We again identify a few characteristics of the deformed structure:

- the members will still be pinned at joints A and B in the deformed state
- member AC will elongate by

$$\delta_{AC} = \frac{PL}{EA} = \frac{(10,000)(0.5)}{(70 \times 10^9)(500 \times 10^{-6})} = 1.43 \times 10^{-4} m \tag{5.39}$$

- member BC will shorten by

$$\delta_{BC} = 1.43 \times 10^{-4} m \tag{5.40}$$

- the members will still be pinned together at joint C

Using this information, we can again construct a diagram that will help us determine the displacement of joint C (Figure 5.25). The result of this diagram is a triangle with known side lengths. The problem of determining the position of C is a geometric one, resolved using the law of cosines.

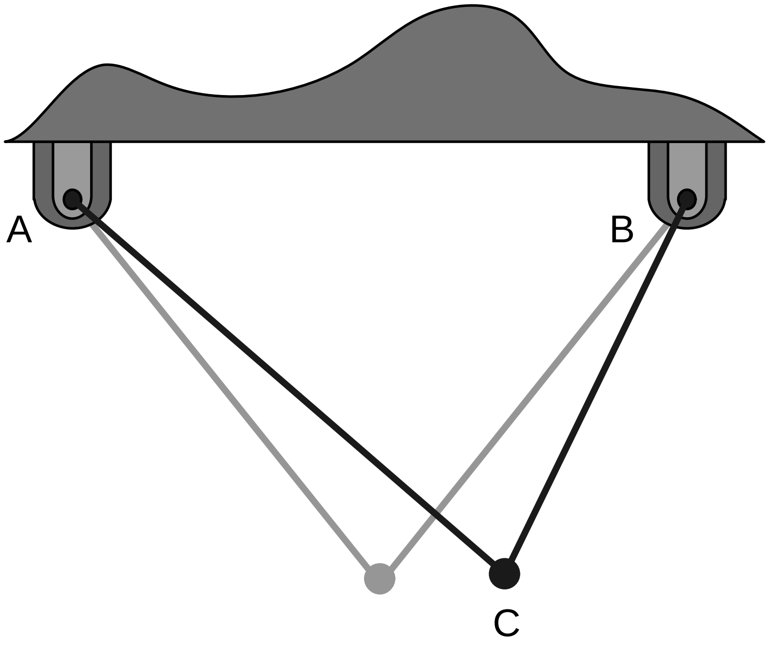


Figure 5.25 Exaggerated deformations are shown for purposes of illustration. The result of this deformation is a triangle where the lengths of the sides are given by the distance between the 2 supports, and the 2 deformed member lengths.

Given the diagram as shown in Figure 5.26, we first find the angle θ

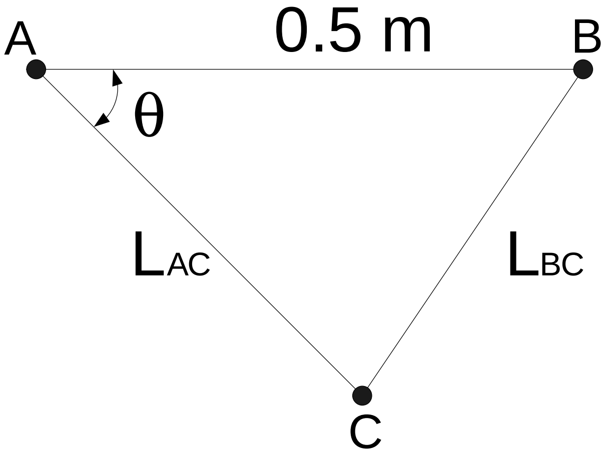


Figure 5.26 Idealized deformation of truss structure.

Using the law of cosines, we have

$$L_{BC}^{2} = L_{AC}^{2} + (0.5)^{2} - 2(L_{AC})(0.5)\cos\theta \tag{5.41}$$

$$(0.5 - 1.43 \times 10^{-4})^{2} = (0.5 + 1.43 \times 10^{-4})^{2} + (0.5)^{2} - 2(0.5 + 1.43 \times 10^{-4})(0.5)\cos\theta \tag{5.42}$$

Simplifying, we find

$$\theta = 59.97162° \tag{5.43}$$

We now construct a right triangle as shown in Figure 5.27.

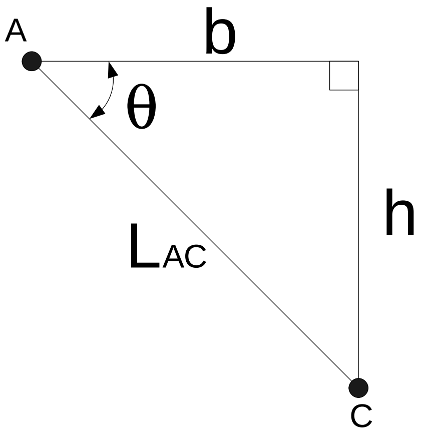


Figure 5.27 Idealized deformation of truss structure.

Using basic trigonometric relations, we find

$$b = L_{AC}\cos\theta = 0.250286m \tag{5.44}$$

$$h = L_{AC}\sin\theta = 0.43301262m \tag{5.45}$$

We know that the values of b and h were - in the undeformed state - 0.25 m and 0.4330127019 m, respectively (this is given by the geometry of the isosceles triangle representing the undeformed truss structure).

This implies that joint C displaces an amount $d_x = 2.86 \times 10^{-4} m$ in the x-direction and $d_y = 7.80 \times 10^{-8} m$ in the y-direction.

The method for finding system displacements is – as is finding internal forces – a systematic one. The procedure is to use the forces to determine the deformed member lengths, determine characteristics of the deformed structure, and solve the problem. Note that relying on the law of cosines can quickly lead to increased complexity, especially when dealing with multi-member truss structures. An alternative approach - matrix truss analysis - provides us with a methodology for determining system displacements, that is decidedly simpler for multi-member truss structures.

5.5 Matrix Truss Analysis

Matrix truss analysis is a matrix-based approach for solving truss system unknowns. This can necessitate seemingly inordinate amounts of work for simple problems that we could solve using the methods presented earlier; however, matrix truss analysis can be very useful for more complex problems, and we can use computers to aid in our computation. Matrix truss analysis will take full advantage of the fact that deformations and displacements are small. Matrix truss analysis also enables us to solve for statically indeterminate problems directly.

Recall the force-deformation behavior of a uniaxially-loaded bar:

$$F = \frac{AE}{L}\delta \tag{5.46}$$

The quantity $\frac{AE}{L}$ represents the axial stiffness of the member (in both tension and compression).

We can define a stiffness matrix for the member in a more general framework by considering its possible degrees of freedom. We will refer to this as the local stiffness matrix (it will later be differentiated from the global stiffness matrix).

Let us first consider a bar oriented along the x-axis, as shown in Figure 5.28.

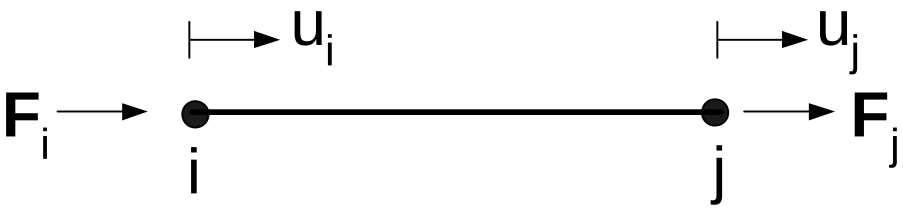


Figure 5.28 This member has two d.o.f., one x translation at each node.

This bar has two degrees of freedom, the x translation of each of its two nodes, i and j. Now, let us set $u_j = 0$ and require that $u_i \neq 0$, as shown in Figure 5.29.

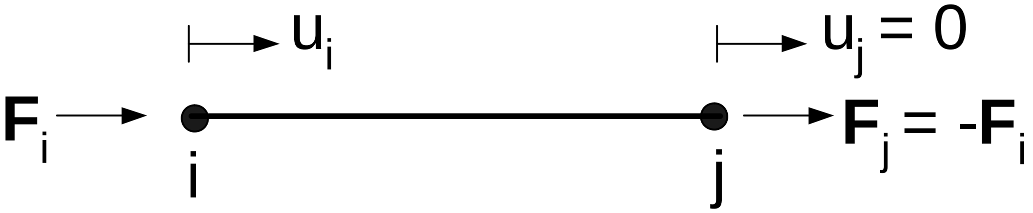


Figure 5.29 Member subjected to a positive displacement at node i while fixing the displacement of node j to be zero.

From (5.46), we find

$$F_i = \frac{AE}{L}u_i \tag{5.47}$$

$$F_j = -\frac{AE}{L}u_i \tag{5.48}$$

Now, let us set $u_i = 0$ and require that $u_j \neq 0$, as shown in Figure 5.30.

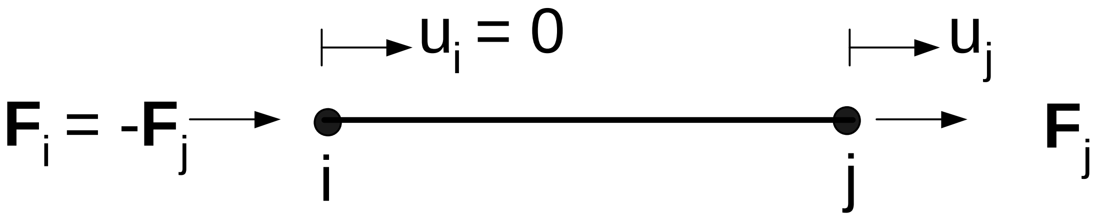


Figure 5.30 Member subjected to a positive displacement at node j while fixing the displacement of node i to be zero.

From (5.46), we find

$$F_j = \frac{AE}{L}u_j \tag{5.49}$$

$$F_i = -\frac{AE}{L}u_j \tag{5.50}$$

Combining (5.47), (5.48), (5.49), and (5.50) into matrix form, we have

$$\begin{bmatrix} F_i \\ F_j \end{bmatrix} = \begin{bmatrix} \frac{AE}{L} & -\frac{AE}{L} \\ -\frac{AE}{L} & \frac{AE}{L} \end{bmatrix} \begin{bmatrix} u_i \\ u_j \end{bmatrix} \tag{5.51}$$

or, more generally

$$\{F\} = [k]\{u\} \tag{5.52}$$

where

$$\{F\} = \begin{bmatrix} F_i \\ F_j \end{bmatrix} \tag{5.53}$$

and

$$\{u\} = \begin{bmatrix} u_i \\ u_j \end{bmatrix} \tag{5.54}$$

are vectors and where

$$[k] = \begin{bmatrix} \frac{AE}{L} & -\frac{AE}{L} \\ -\frac{AE}{L} & \frac{AE}{L} \end{bmatrix} \tag{5.55}$$

is the stiffness matrix. We can write this more generally as

$$F_m = \sum_{n=1}^{numberDOF} k_{mn}u_n \tag{5.56}$$

Physically, the matrix element k_{mn} is the force in the m d.o.f. due to a unit displacement in the n d.o.f. when all other d.o.f. are constrained to be zero. Specifically, for the example shown in Figure 5.28, we have

$$\{F\} = [k]\{u\} \tag{5.57}$$

or, more specifically

$$\begin{bmatrix} F_i \\ F_j \end{bmatrix} = \begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} \begin{bmatrix} u_i \\ u_j \end{bmatrix} = \begin{bmatrix} \frac{AE}{L} & -\frac{AE}{L} \\ -\frac{AE}{L} & \frac{AE}{L} \end{bmatrix} \begin{bmatrix} u_i \\ u_j \end{bmatrix} \tag{5.58}$$

This is the same as

$$F_i = \sum_{i=1}^{2} k_{ij}u_j \tag{5.59}$$

or, in terms of the linear equations,

$$F_1 = k_{11}u_1 + k_{12}u_2 \tag{5.60}$$

$$F_2 = k_{21}u_1 + k_{22}u_2 \tag{5.61}$$

If we consider the specific case where $u_1 = 0$ and $u_2 = u*$, we have

$$F_1 = k_{12}u^* = (-\frac{AE}{L})u^* \tag{5.62}$$

$$F_2 = k_{22}u^* = (\frac{AE}{L})u^* \tag{5.63}$$

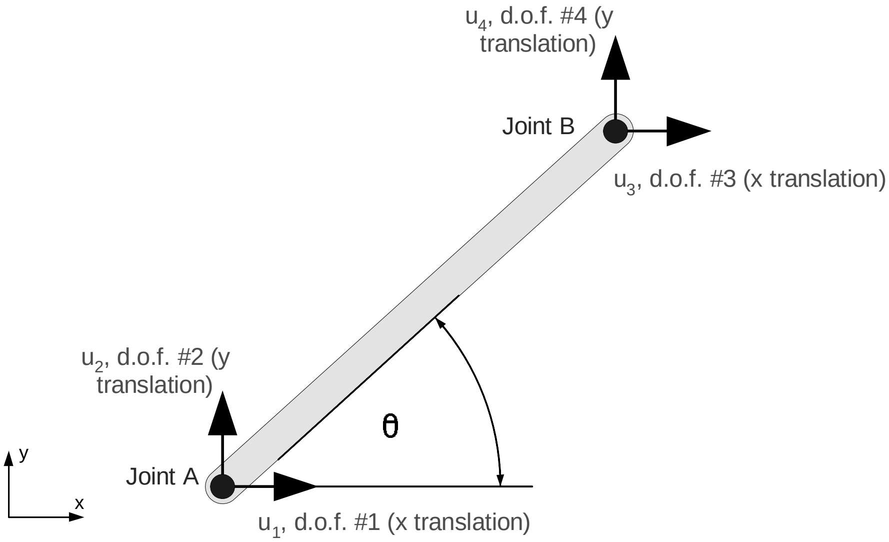
If we instead consider the case where $u_1 = u_2 = u^*$, we have

$$F_1 = k_{11}u^* + k_{12}u^* = (\frac{AE}{L} - \frac{AE}{L})u^* = 0 \tag{5.64}$$

$$F_2 = k_{12}u^* + k_{22}u^* = (-\frac{AE}{L} + \frac{AE}{L})u^* = 0 \tag{5.65}$$

Note that F_1 and F_2 are both equal to zero in this case because the bar undergoes rigid body displacement and no deformation.

Now, let us consider the example of a general planar (two-dimensional) truss element, inclined at some arbitrary angle θ (Figure 5.31). This truss element has four degrees of freedom, two degrees of freedom (x and y translation) at each joint.



u4, d.o.f. #4 (y translation)
Joint B
u3, d.o.f. #3 (x translation)
u2, d.o.f. #2 (y translation)
θ
y
x
Joint A
u1, d.o.f. #1 (x translation)

Figure 5.31 This general plane truss member has 4 d.o.f., 2 (x and y translations) at each node. It is inclined at an arbitrary angle θ .

We will show that forces at the joints corresponding to each degree of freedom can be related to the degrees of freedom (u_1, u_2, u_3, u_4 as depicted in Figure 5.31) as follows:

$$\begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} = \begin{bmatrix} k_q \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{5.66}$$

where

$$[k_q] = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \\ k_{31} & k_{32} & k_{33} & k_{34} \\ k_{41} & k_{42} & k_{43} & k_{44} \end{bmatrix} \tag{5.67}$$

is the stiffness matrix of the member.

Physically, a matrix element k_{ij} is defined as the force along degree of freedom i due to a virtual unit displacement along degree of freedom j. Consider the case of k_{11}. This represents the force along degree of freedom 1 due to a unit displacement along degree of freedom 1 when all other degrees of freedom are constrained. As an example, in matrix form, we have for a unit displacement in degree of freedom #1:

$$\begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \\ k_{31} & k_{32} & k_{33} & k_{34} \\ k_{41} & k_{42} & k_{43} & k_{44} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.68}$$

$$\begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} = \begin{bmatrix} k_{11} \\ k_{21} \\ k_{31} \\ k_{41} \end{bmatrix} \tag{5.69}$$

The forces required in the member to achieve this unit displacement in the 1-direction are $F_1 = k_{11}$, $F_2 = k_{21}$, $F_3 = k_{31}$, $F_4 = k_{41}$. In general, k_{ij} is the force in the i d.o.f. given a unit displacement in the j d.o.f. when all other d.o.f. are set to zero. A unit displacement along degree of freedom 1 will lead to a compression of the member as shown (Figure 5.32).

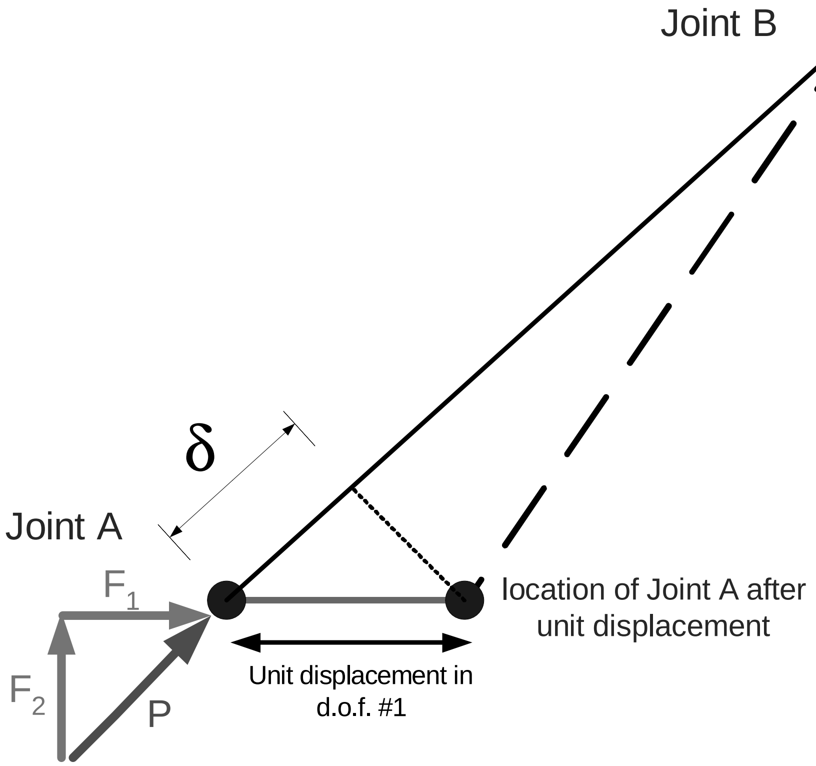

Joint B
δ
Joint A
F_1
F_2
P
location of Joint A after unit displacement
Unit displacement in d.o.f. #1

Figure 5.32 General member AB is subjected to a unit displacement in d.o.f. #1. This results in a compressive force of magnitude P within the member. The x and y components of P are F_1 and F_2, respectively.

Under this loading, the compression, δ, of the member is given by $cos(\theta)$ if we assume the deformation and displacements are small (refer to Assumption of Small Displacements). To induce a displacement of this magnitude, we return to our definition of the deformation of an axially-loaded member:

$$\delta = \frac{PL}{EA} \tag{5.70}$$

We know that P is the force that must be applied along the axis of the member to induce a deformation δ. This yields

$$P = \frac{AE}{L}cos(\theta) \tag{5.71}$$

The component of this axial force that acts in degree of freedom 1 is given from geometry by

$$F_1 = Pcos(\theta) \tag{5.72}$$

Combining (5.71) and (5.72) gives us that

$$F_1 = k_{11} = \frac{AE}{L}cos(\theta)cos(\theta) = \frac{AE}{L}(cos(\theta))^2 \tag{5.73}$$

If we find the y component of force P, we will find the value of F_2, which is k_{21}.

$$F_2 = k_{21} = \frac{AE}{L}cos(\theta)sin(\theta) \tag{5.74}$$

For degrees of freedom 3 and 4 (in the case of a unit displacement in the x-direction), we find:

$$F_3 = k_{31} = -F_1 = -\frac{AE}{L}(cos(\theta))^2 \tag{5.75}$$

$$F_4 = k_{41} = -F_2 = -\frac{AE}{L}cos(\theta)sin(\theta) \tag{5.76}$$

since the forces ar Joint B must be equal and opposite to those at Joint A.

Next, to find k_{12}, we virtually apply a unit displacement in degree of freedom #2, as shown in Figure 5.33.

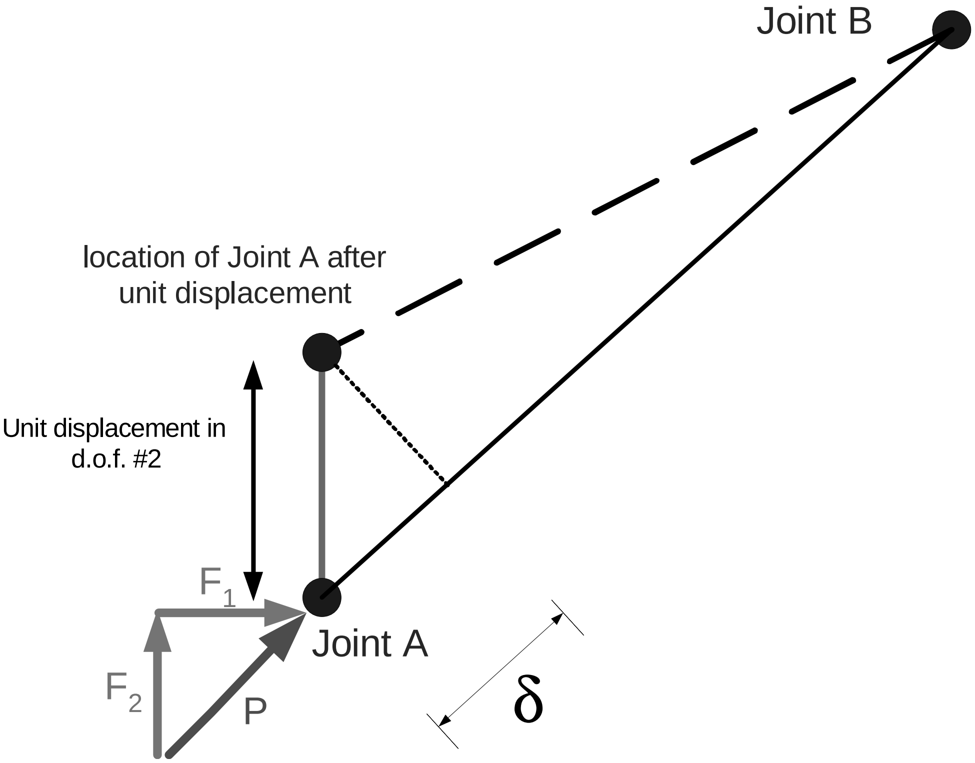


Figure 5.33 General member AB is subjected to a unit displacement in d.o.f. #2. This results in a compressive force of magnitude P within the member. The x and y components of P are F_1 and F_2, respectively.

P is again the force that must be applied along the axis of the member to induce a deflection δ. It now has a value of

$$P = \frac{AE}{L}\sin(\theta) \tag{5.77}$$

The component of this axial force that acts in degree of freedom 1 is given from geometry by

$$F_1 = P\cos(\theta) \tag{5.78}$$

Combining (5.77) and (5.78) gives us

$$F_1 = k_{12} = \frac{AE}{L}\sin(\theta)\cos(\theta) \tag{5.79}$$

We use a similar procedure to find the other terms in the local stiffness matrix. We note that, similar to the case of k$_{12}$ and k$_{21}$, that $k_{ij} = k_{ji}$ for every value of i and j. This implies, by the definition of a symmetric matrix, that the local stiffness matrix is symmetric. The diagonal components are also all positive. This makes sense because we expect that a positive force in a particular degree of freedom will induce a positive displacement in the same degree of freedom; likewise, a negative force in a degree of freedom will induced a negative displacement in the same degree of freedom.

We obtain the following local stiffness matrix

$$[k_q] = \frac{AE}{L}\begin{bmatrix} CC & CS & -CC & -CS \\ CS & SS & -CS & -SS \\ -CC & -CS & CC & CS \\ -CS & -SS & CS & SS \end{bmatrix} \tag{5.80}$$

where $C = cos(\theta)$ and $S = sin(\theta)$.

Another important observation to make is that only three different terms are calculated in determining the local stiffness matrix: CC, CS, and SS. This knowledge will save us time now and computing power later.

The next step is to obtain the local stiffness matrices of all the members of the structure. Upon completion of this step, we assemble the previously mentioned global stiffness matrix. This matrix combines the effects of the stiffnesses of each of the local elements and enforces structural compatibility. In order to construct this global stiffness matrix, we identify global d.o.f. and match local d.o.f. to global ones. Upon doing so, we add the stiffness components from the local stiffness matrices to the appropriate corresponding element in the global stiffness matrix. Now, we will look at a specific example to illustrate the procedure for constructing the global stiffness matrix.

Consider the three-member truss shown in Figure 5.34.

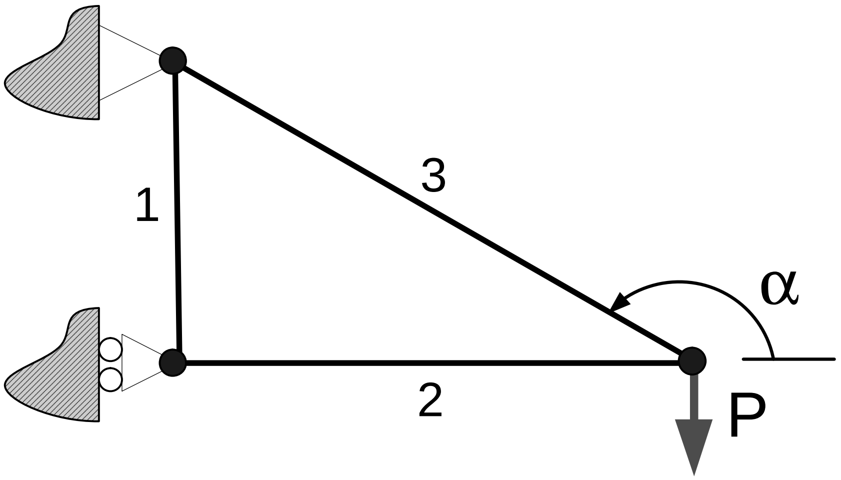


Figure 5.34 Consider this three-member truss subjected to an end load of magnitude P.

For this problem, assume that the areas and materials of the three members are identical. In other words, $A_1 = A_2 = A_3 = A$ and $E_1 = E_2 = E_3 = E$. We first compute the stiffness matrix for each element i.

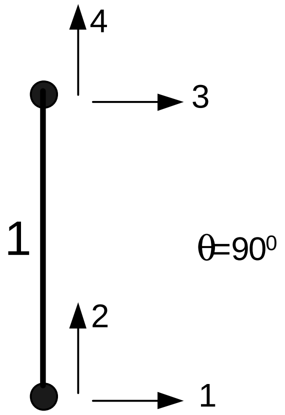


Figure 5.35 Local degrees of freedom for Member 1.

For, element 1, as shown in Figure 5.35, $\theta = 90°$, so we obtain

$$k_1 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{AE}{L_1} & 0 & -\frac{AE}{L_1} \\ 0 & 0 & 0 & 0 \\ 0 & -\frac{AE}{L_1} & 0 & \frac{AE}{L_1} \end{bmatrix} \tag{5.81}$$

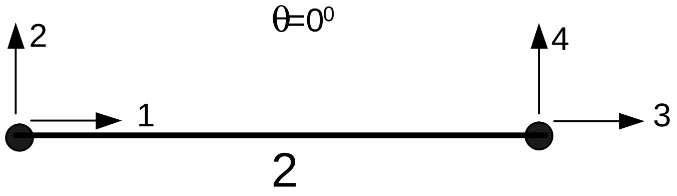


Figure 5.36 Local degrees of freedom for Member 2.

For, element 1, as shown in Figure 5.35, $\theta = 0°$, so we obtain

$$k_2 = \begin{bmatrix} \frac{AE}{L_2} & 0 & -\frac{AE}{L_2} & 0 \\ -\frac{AE}{L_2} & 0 & \frac{AE}{L_2} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{5.82}$$

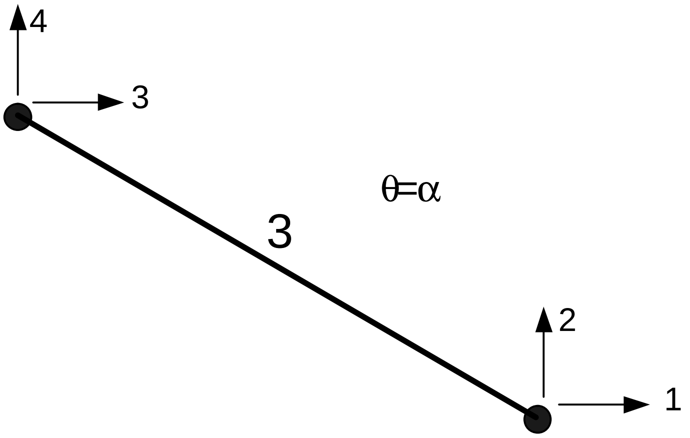


Figure 5.37 Local degrees of freedom for Member 3.

For, element 1, as shown in Figure 5.35, $\theta = \alpha$, so we obtain

$$k_3 = \frac{AE}{L_3}\begin{bmatrix} CC & CS & -CC & -CS \\ CS & SS & -CS & -SS \\ -CC & -CS & CC & CS \\ -CS & -SS & CS & SS \end{bmatrix} \tag{5.83}$$

where $C = cos(\theta)$ and $S = sin(\theta)$.

We now identify common degrees of freedom between elements; this enforces the compatibility of the structure. We identify global degrees of freedom and match local degrees of freedom to global ones. Figure 5.38 and Figure 5.39 indicate the relation of global and local degrees of freedom as we have constructed them.

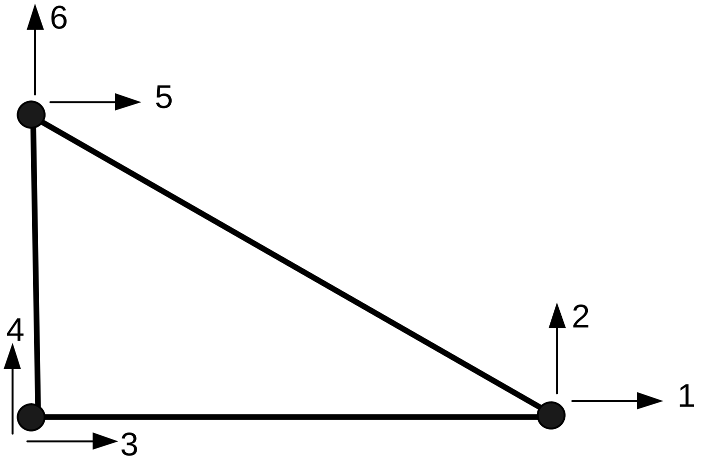


Figure 5.38 Global degrees of freedom for the three-member truss problem.

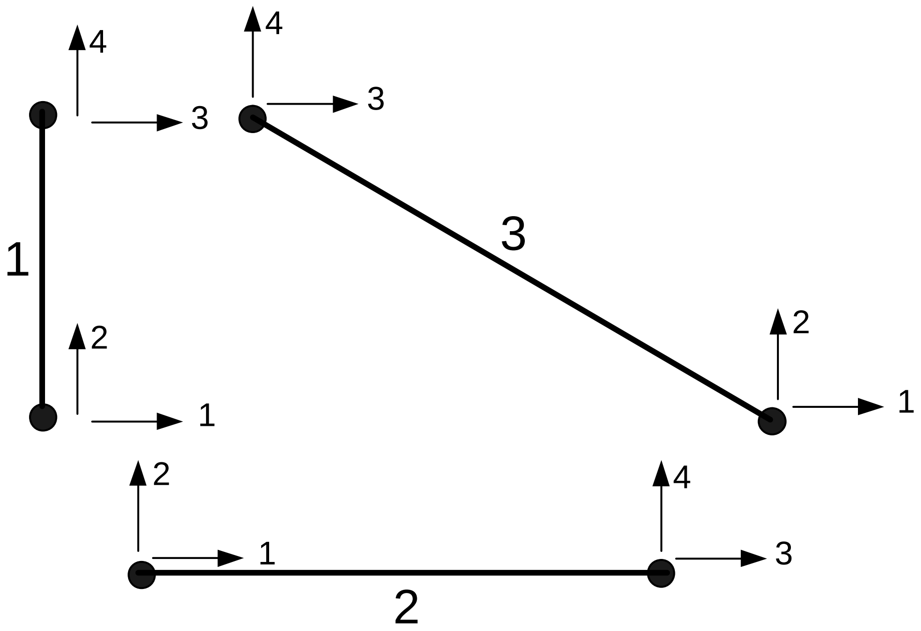


Figure 5.39 Local degrees of freedom for the three-member truss problem.

We now construct a table that indicates the relation among global and local degrees of freedom (Table 5.5).

TABLE 5.5 List of which global d.o.f. are associated with which local d.o.f.

Global degree of freedom (d.o.f.)	Corresponding local d.o.f.
#1	Member 2, #3 Member 3, #1
#2	Member 2, #4 Member 3, #2
#3	Member 1, #1 Member 2, #1
#4	Member 1, #2 Member 2, #2
#5	Member 1, #3 Member 3, #3
#6	Member 1, #4 Member 3, #4

We now assemble the global stiffness matrix K_G, where each element K_{ij} has a contribution from the members that share those degrees of freedom. We find that

$$[K_G] = \begin{bmatrix} K_{11} & K_{12} & K_{13} & K_{14} & K_{15} & K_{16} \\ K_{21} & K_{22} & K_{23} & K_{24} & K_{25} & K_{26} \\ K_{31} & K_{32} & K_{33} & K_{34} & K_{35} & K_{36} \\ K_{41} & K_{42} & K_{43} & K_{44} & K_{45} & K_{46} \\ K_{51} & K_{52} & K_{53} & K_{54} & K_{55} & K_{56} \\ K_{61} & K_{62} & K_{63} & K_{64} & K_{65} & K_{66} \end{bmatrix}^G \tag{5.84}$$

$$[K_G] = \begin{bmatrix} k_{11}^{(3)} + k_{33}^{(2)} & k_{34}^{(2)} + k_{12}^{(3)} & k_{31}^{(2)} & k_{32}^{(2)} & k_{13}^{(3)} & k_{14}^{(3)} \\ k_{21}^{(3)} + k_{43}^{(2)} & k_{22}^{(3)} + k_{44}^{(2)} & k_{41}^{(2)} & k_{42}^{(2)} & k_{23}^{(3)} & k_{24}^{(3)} \\ k_{13}^{(2)} & k_{14}^{(2)} & k_{11}^{(1)} + k_{11}^{(2)} & k_{12}^{(2)} + k_{12}^{(1)} & k_{13}^{(1)} & k_{14}^{(1)} \\ k_{23}^{(2)} & k_{24}^{(2)} & k_{21}^{(2)} + k_{21}^{(1)} & k_{22}^{(1)} + k_{22}^{(2)} & k_{23}^{(1)} & k_{24}^{(1)} \\ k_{31}^{(3)} & k_{32}^{(3)} & k_{31}^{(1)} & k_{32}^{(1)} & k_{33}^{(1)} + k_{33}^{(3)} & k_{34}^{(1)} + k_{34}^{(3)} \\ k_{41}^{(3)} & k_{42}^{(3)} & k_{41}^{(1)} & k_{42}^{(1)} & k_{43}^{(1)} + k_{43}^{(3)} & k_{44}^{(1)} + k_{44}^{(3)} \end{bmatrix} \tag{5.85}$$

Upon creating this global stiffness matrix, we can relate the system forces to the system displacements in the following manner:

$$\{F\} = [K_G]\{u\} \tag{5.86}$$

where {F} represents the force vector (representing x and y forces at each of the joints), $[K_G]$ is the global stiffness matrix, and {u} is the displacement vector (representing x and y displacements at each of the joints).

{F} will be of the following form

$$\{F\} = \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ \dots \\ F_i \end{bmatrix} \tag{5.87}$$

and {u} will be of the form

$$\{u\} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \dots \\ u_i \end{bmatrix} \tag{5.88}$$

We must apply the problem boundary conditions by assigning known values to these two vectors. We know that the displacement of any fixed degree of freedom will be 0. We also know that - unless an external force is applied - the force in the direction of any unconstrained degree of freedom will be 0. In the case where an external load is applied in the direction of an unconstrained degree of freedom, that force will be equal to the applied load. It is important to note that if a force is applied in the direction of a fixed degree of freedom, it will not affect the structure. Note that we can solve for both forces and displacements using this method.

To illustrate this entire process of matrix structural analysis, we will look at a simple example that we have already examined using other methods.

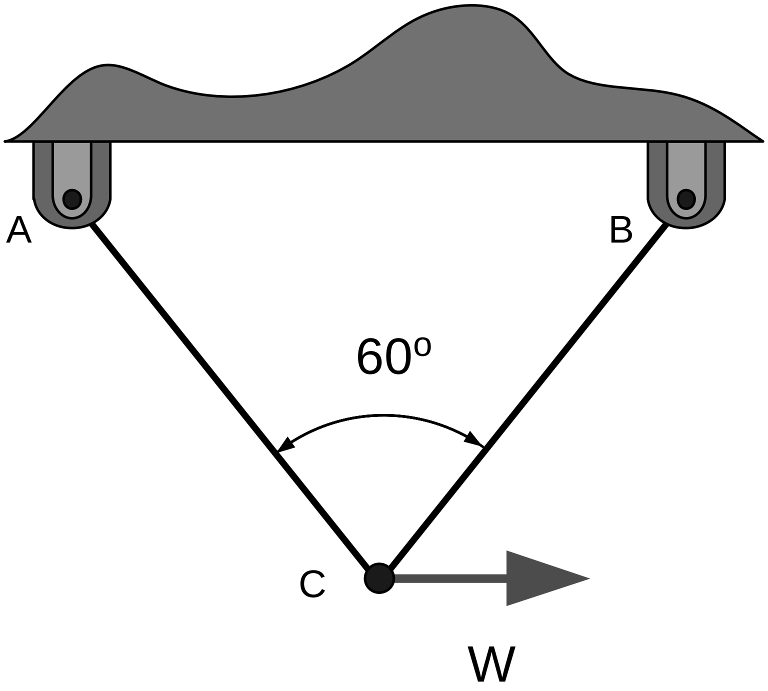


Figure 5.40 We revisit this example that we solved earlier. Now we will use a different approach: matrix structural analysis.

We will again consider the members to be of aluminum construction, of length 0.5 m, the cross-sectional areas of the members to be 500 sq. mm., and the load W to have magnitude 10,000 N.

We will first define our local d.o.f.

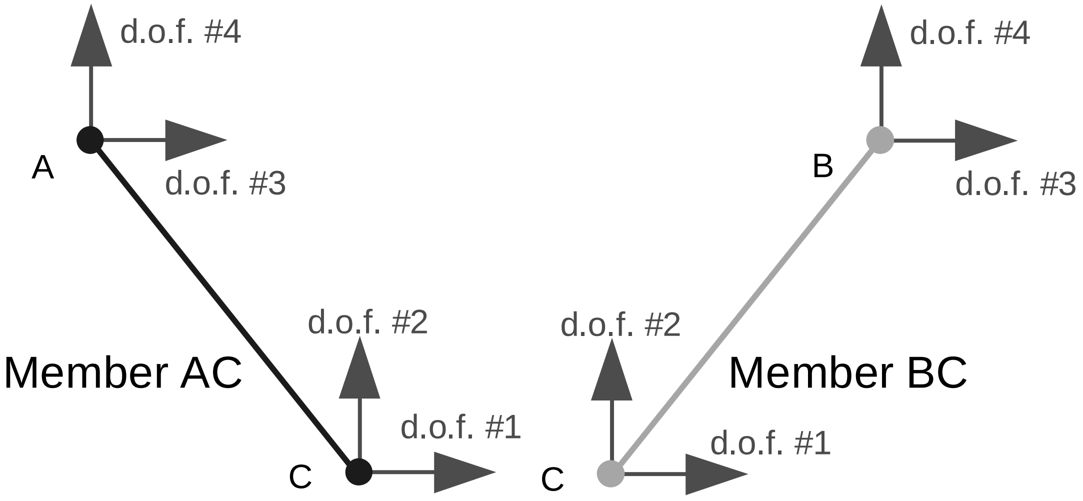


Figure 5.41 Local d.o.f. definitions for members AC and BC.

Next we will define our global d.o.f.

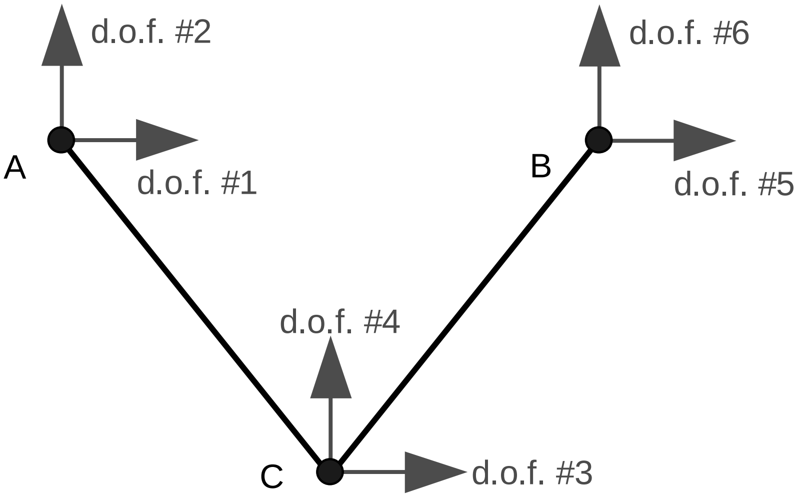


Figure 5.42 Global d.o.f. definition for symmetric two-member truss. This assignment is arbitrary and will not affect our results as long as we are consistent.

Now we will construct the local stiffness matrices for each of the elements. We begin with member AC. The angle θ at which this member is positioned from the horizontal is $120°$. The length, area, and Young's modulus are also known. Using (5.80) we have:

$$k_{AC} = \frac{AE}{L}\begin{bmatrix} CC & CS & -CC & -CS \\ CS & SS & -CS & -SS \\ -CC & -CS & CC & CS \\ -CS & -SS & CS & SS \end{bmatrix} \tag{5.89}$$

$$CC = (\cos(120°))^2 = \frac{1}{4} \tag{5.90}$$

$$CS = \cos(120°)\sin(120°) = -\frac{\sqrt{3}}{4} \tag{5.91}$$

$$SS = (\sin(120°))^2 = \frac{3}{4} \tag{5.92}$$

Now we construct the local stiffness matrix for BC. The angle θ at which this member is positioned from the horizontal is $60°$. Again using we (5.80) have:

$$k_{BC} = \frac{AE}{L}\begin{bmatrix} CC & CS & -CC & -CS \\ CS & SS & -CS & -SS \\ -CC & -CS & CC & CS \\ -CS & -SS & CS & SS \end{bmatrix} \tag{5.93}$$

$$CC = (\cos(60°))^2 = \frac{1}{4} \tag{5.94}$$

$$CS = \cos(60°)\sin(60°) = \frac{\sqrt{3}}{4} \tag{5.95}$$

$$SS = (\sin(60°))^2 = \frac{3}{4} \tag{5.96}$$

Having created the local stiffness matrices, it is time to assemble the global stiffness matrix. We begin by creating a list of which local d.o.f. correspond to which global d.o.f. (Table 5.6).

TABLE 5.6 List of which global d.o.f. are associated with which local d.o.f.

Global degree of freedom (d.o.f.)	**Corresponding local d.o.f.**
#1	Member AC, #3
#2	Member AC #4
#3	Member AC, #1; Member BC, #1
#4	Member AC, #2; Member BC, #2
#5	Member BC #3
#6	Member BC #4

Now we assemble the global matrix by adding local stiffness matrix components to the appropriate global components.

$$[K_G] = \begin{bmatrix} K_{11} & K_{12} & K_{13} & K_{14} & K_{15} & K_{16} \\ K_{21} & K_{22} & K_{23} & K_{24} & K_{25} & K_{26} \\ K_{31} & K_{32} & K_{33} & K_{34} & K_{35} & K_{36} \\ K_{41} & K_{42} & K_{43} & K_{44} & K_{45} & K_{46} \\ K_{51} & K_{52} & K_{53} & K_{54} & K_{55} & K_{56} \\ K_{61} & K_{62} & K_{63} & K_{64} & K_{65} & K_{66} \end{bmatrix}^G = \begin{bmatrix} k_{33} & k_{34} & k_{31} & k_{32} & 0 & 0 \\ k_{43} & k_{44} & k_{41} & k_{42} & 0 & 0 \\ k_{13} & k_{14} & k_{11} & k_{12} & 0 & 0 \\ k_{23} & k_{24} & k_{21} & k_{22} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^{AC} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & k_{11} & k_{12} & k_{13} & k_{14} \\ 0 & 0 & k_{21} & k_{22} & k_{23} & k_{24} \\ 0 & 0 & k_{31} & k_{32} & k_{33} & k_{34} \\ 0 & 0 & k_{41} & k_{42} & k_{43} & k_{44} \end{bmatrix}^{BC} \tag{5.97}$$

Note that the local stiffness matrix of member AC does not contribute to the global stiffness matrix in global degrees of freedom 5 or 6. This is because member AC has no effect on (nor is it affected by) these d.o.f.; it is not connected to the joint where these d.o.f. act.

Now, in order to solve for the forces and displacements, we solve the following equation

$$\{F\} = [K_G]\{u\} \tag{5.98}$$

where [K_G] is given by

$$[K_G] = \frac{AE}{L}\begin{bmatrix} \frac{1}{4} & -\frac{\sqrt{3}}{4} & -\frac{1}{4} & \frac{\sqrt{3}}{4} & 0 & 0 \\ -\frac{\sqrt{3}}{4} & \frac{3}{4} & \frac{\sqrt{3}}{4} & -\frac{3}{4} & 0 & 0 \\ -\frac{1}{4} & \frac{\sqrt{3}}{4} & \frac{1}{2} & 0 & -\frac{1}{4} & -\frac{\sqrt{3}}{4} \\ \frac{\sqrt{3}}{4} & -\frac{3}{4} & 0 & \frac{3}{2} & -\frac{\sqrt{3}}{4} & -\frac{3}{4} \\ 0 & 0 & -\frac{1}{4} & -\frac{\sqrt{3}}{4} & \frac{1}{4} & \frac{\sqrt{3}}{4} \\ 0 & 0 & -\frac{\sqrt{3}}{4} & -\frac{3}{4} & \frac{\sqrt{3}}{4} & \frac{3}{4} \end{bmatrix} \tag{5.99}$$

In order to solve (5.98), we must also account for boundary conditions. We know that the displacements at the fixed joints will be 0; we also know that the loads at joint C will be 0 in the vertical direction (d.o.f. #4) and 10,000 N in the horizontal direction (d.o.f. #4). This information is given below (Table 5.7).

TABLE 5.7 Boundary conditions and known forces.

Known Quantities	Value
F_3	10,000 N
F_4	0 N
u_1	0 m
u_2	0 m
u_5	0 m
u_6	0 m

This information provides us means to solve this problem.

[Note that this stiffness matrix is based only on the geometry of the truss structure; it is not based on the loading conditions. This K_G could thus be used to solve the problem shown Figure 5.19 (where the applied load is in the y-direction), as well as this problem.]

So we have

$$\begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \end{bmatrix} = \frac{AE}{L}\begin{bmatrix} \frac{1}{4} & -\frac{\sqrt{3}}{4} & -\frac{1}{4} & \frac{\sqrt{3}}{4} & 0 & 0 \\ -\frac{\sqrt{3}}{4} & \frac{3}{4} & \frac{\sqrt{3}}{4} & -\frac{3}{4} & 0 & 0 \\ -\frac{1}{4} & \frac{\sqrt{3}}{4} & \frac{1}{2} & 0 & -\frac{1}{4} & -\frac{\sqrt{3}}{4} \\ \frac{\sqrt{3}}{4} & -\frac{3}{4} & 0 & \frac{3}{2} & -\frac{\sqrt{3}}{4} & -\frac{3}{4} \\ 0 & 0 & -\frac{1}{4} & -\frac{\sqrt{3}}{4} & \frac{1}{4} & \frac{\sqrt{3}}{4} \\ 0 & 0 & -\frac{\sqrt{3}}{4} & -\frac{3}{4} & \frac{\sqrt{3}}{4} & \frac{3}{4} \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_5 \end{bmatrix} \tag{5.100}$$

which simplifies to

$$\begin{bmatrix} F_1 \\ F_2 \\ 10,000 \\ 0 \\ F_5 \\ F_6 \end{bmatrix} = \frac{AE}{L}\begin{bmatrix} \frac{1}{4} & -\frac{\sqrt{3}}{4} & -\frac{1}{4} & \frac{\sqrt{3}}{4} & 0 & 0 \\ -\frac{\sqrt{3}}{4} & \frac{3}{4} & \frac{\sqrt{3}}{4} & -\frac{3}{4} & 0 & 0 \\ -\frac{1}{4} & \frac{\sqrt{3}}{4} & \frac{1}{2} & 0 & -\frac{1}{4} & -\frac{\sqrt{3}}{4} \\ \frac{\sqrt{3}}{4} & -\frac{3}{4} & 0 & \frac{3}{2} & -\frac{\sqrt{3}}{4} & -\frac{3}{4} \\ 0 & 0 & -\frac{1}{4} & -\frac{\sqrt{3}}{4} & \frac{1}{4} & \frac{\sqrt{3}}{4} \\ 0 & 0 & -\frac{\sqrt{3}}{4} & -\frac{3}{4} & \frac{\sqrt{3}}{4} & \frac{3}{4} \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ u_3 \\ u_4 \\ 0 \\ 0 \end{bmatrix} \tag{5.101}$$

Note that we can leave the $\frac{AE}{L}$ term outside of the matrix, since A, E and L are the same for all members in the structure. Solving this equation will yield the reaction forces and the displacements. Since u_3 and u_4 are the only desired unknowns, (5.101) reduces to a simple pair of linear equations:

$$\begin{bmatrix} 10,000 \\ 0 \end{bmatrix} = \frac{AE}{L}\begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{3}{2} \end{bmatrix}\begin{bmatrix} u_3 \\ u_4 \end{bmatrix} \tag{5.102}$$

If we need to find the other 4 unknowns (F_1, F_2, F_5, F_6), we can plug the values for u_3 and u_4 back into equation (5.101). Then, multiplying [K_G] by {u} will yield the complete force vector; this will provide us with information about the reaction forces at the two pinned supports.

Specifically we find that $u_3 = 2.86 \times 10^{-4} m$ and $u_4 = 0.0m$ which matches up with our earlier non-matrix solution (Figure 5.23). Note that u_3 is much smaller than the member length ($u_3 << L$, or 0.000286 m << 0.5 m). It is important to realize that - in solving (5.101) - we did not need to solve for the member forces. In matrix truss analysis, most of the work is performed in initially setting up the stiffness matrix. After that, parameters can easily be varied in analyzing different solutions. In this manner matrix truss analysis provides a systematic and successful methodology for complex truss analysis.

The Truss Structures Simulation uses matrix truss analysis as the basis for its computation; as such, it enables the calculation of solutions to complex truss problems (go to Truss Structures Simulation).

5.6 How a Truss Structure Carries Shear and Moment Loads

Now that we have developed the means to analyze truss structures, we can examine the bridge truss presented in Figure 5.43 in order to understand how a truss structure carries shear and moment loads.

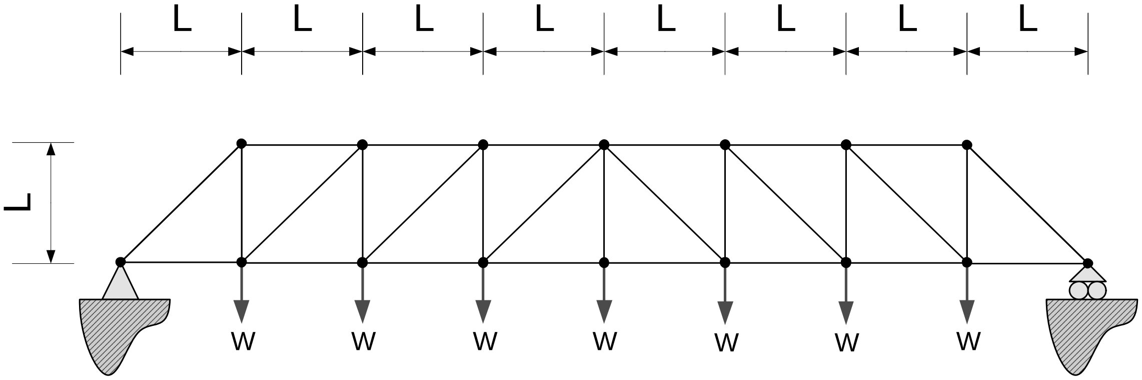


Figure 5.43 A 29-member bridge truss, subjected to distributed loading acting at the joints. This truss is an example of a Howe truss design. The left edge is pinned and the right edge is simply supported.

We first determine that it is a statically determinate structure, as illustrated in Table 5.8.

TABLE 5.8 Values of quantities for determining static determinacy for 29-member bridge truss example.

Quantity	Value
N (number of joints)	16
r (number of reactions)	3
m (number of members)	29
n (=2N-r)	29

Since the structure is comprised solely of triangular sub-elements, we can conclude that the structure is also stable. Using the Truss Structures simulation, we obtain the load distribution within the structure: Figure 5.44 illustrates the distribution of compressive and tensile loads.

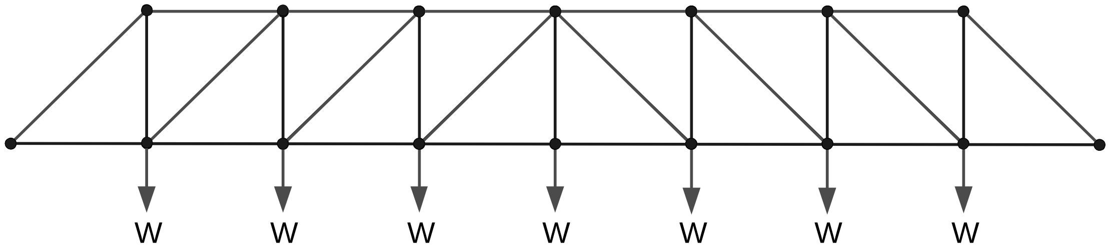


Figure 5.44 Loads in the 29-member bridge truss. Tensile loads are shown in blue and compressive loads are shown in red.

As can be seen, the top and angled members bear the compressive loads; the bottom and vertical members bear the tensile loads. This is analogous to the behavior of a beam - pinned on one edge and simply supported on the other edge - subjected to a distributed load. Such a beam, as shown in Figure 5.45, will have a stress distribution through the height of the beam, such that compressive stresses will act in the upper half of the beam and tensile stresses will act in the lower half of the beam.

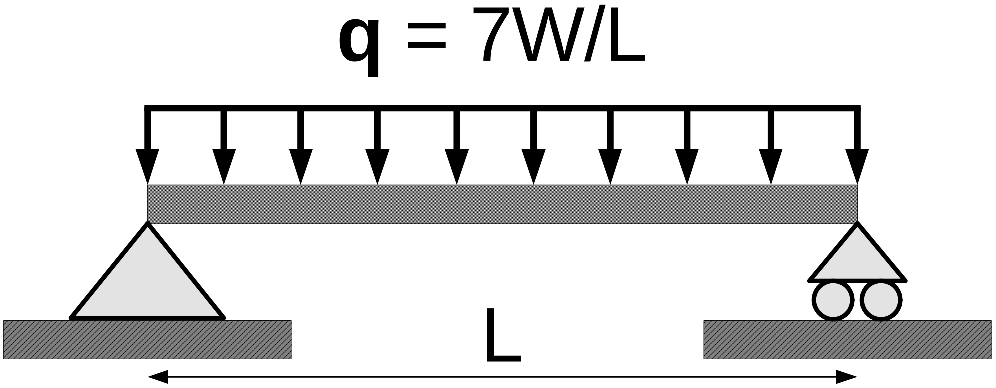


Figure 5.45 A beam, pinned on the left edge and simply supported on the right, subjected to a distributed load q.

To better understand how truss structures can act in this manner, we will determine the forces and moments acting within the beam.

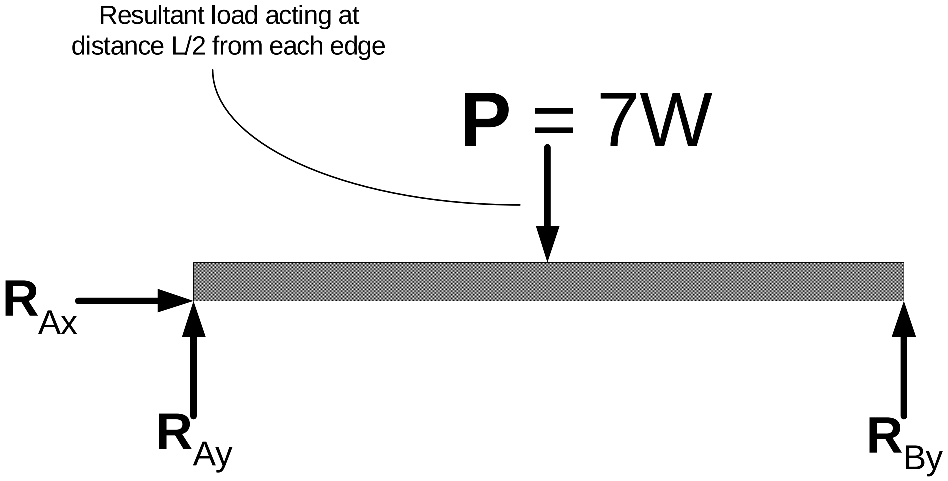


Figure 5.46 Free-body diagram of the beam - pinned on one edge and simply supported on the other edge - subjected to a distributed load q.

Drawing a free-body diagram (FBD), as shown in Figure 5.46, yields the system reaction forces:

$$R_{Ax} = 0 \tag{5.103}$$

$$R_{Ay} = R_{By} = \frac{7W}{2} \tag{5.104}$$

Taking a cut of the beam at some location x allows us to determine the shear and moment acting within beam. This is illustrated in Figure 5.47.

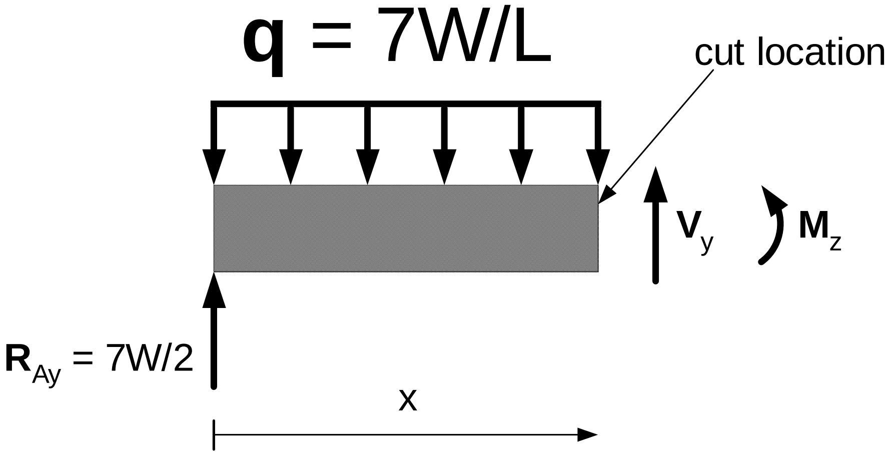


Figure 5.47 "Cut" of the beam - pinned on one edge and simply supported on the other edge - subjected to a distributed load q.

Applying the equations of equilibrium, we find

$$\Sigma F_y = 0 \tag{5.105}$$

$$\frac{7w}{2} - \frac{7W}{L}x + V_y = 0 \tag{5.106}$$

$$Vy = \frac{7W}{L}x - \frac{7w}{2} \tag{5.107}$$

$$\Sigma M_{cut} = 0 \tag{5.108}$$

$$\frac{7w}{2}x - \frac{7Wx}{L}\left(\frac{x}{2}\right) + M_z = 0 \tag{5.109}$$

$$M_z = -\frac{7Wx^2}{2L} - \frac{7w}{2}x \tag{5.110}$$

We can perform the same type of analysis on the 29-member truss bridge. We will first draw a free-body diagram of the entire system, using this to determine reaction forces, as shown in Figure 5.48.

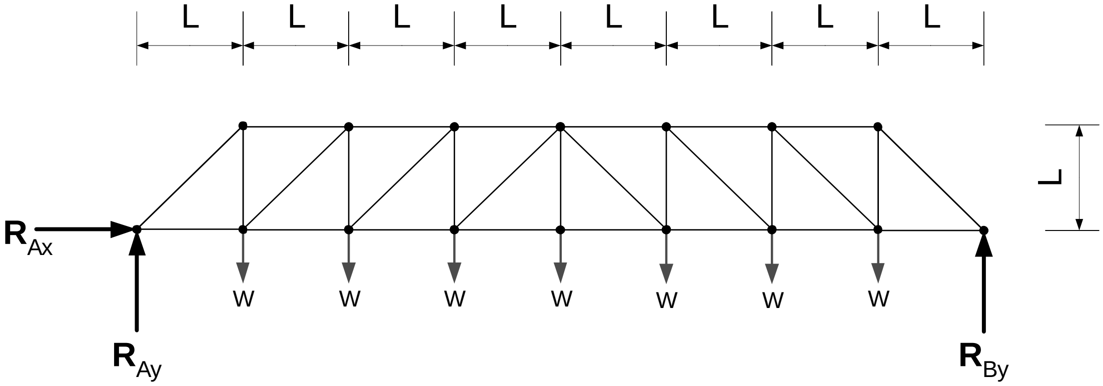


Figure 5.48 Free-body diagram of the 29-member truss bridge.

We find that

$$R_{Ax} = 0 \tag{5.111}$$

$$_{Ay} = R_{By} = \frac{7W}{2} \tag{5.112}$$

Taking a "cut" of the system, as shown in Figure 5.49, we can find the net shear and moment that are acting through the structure at that position.

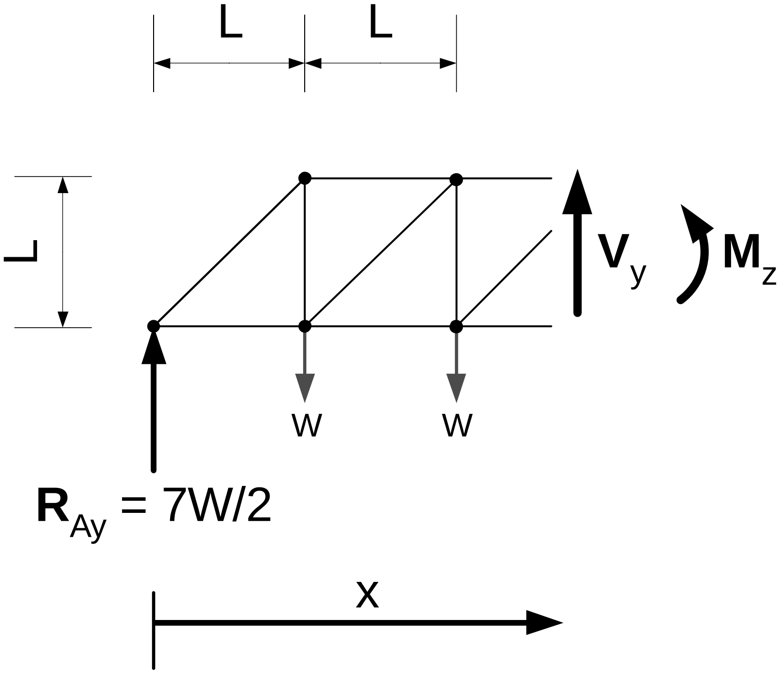


Figure 5.49 "Cut" of the 29-member truss bridge.

Summing forces and moments, we find that

$$V_y = -\frac{3W}{2} \tag{5.113}$$

$$M_z = \frac{7W}{2}(2L) - WL = 6WL \tag{5.114}$$

By analyzing the forces at work at the cut location, as in Figure 5.50, we can understand how the forces in the members act to bear these shear and moment loads.

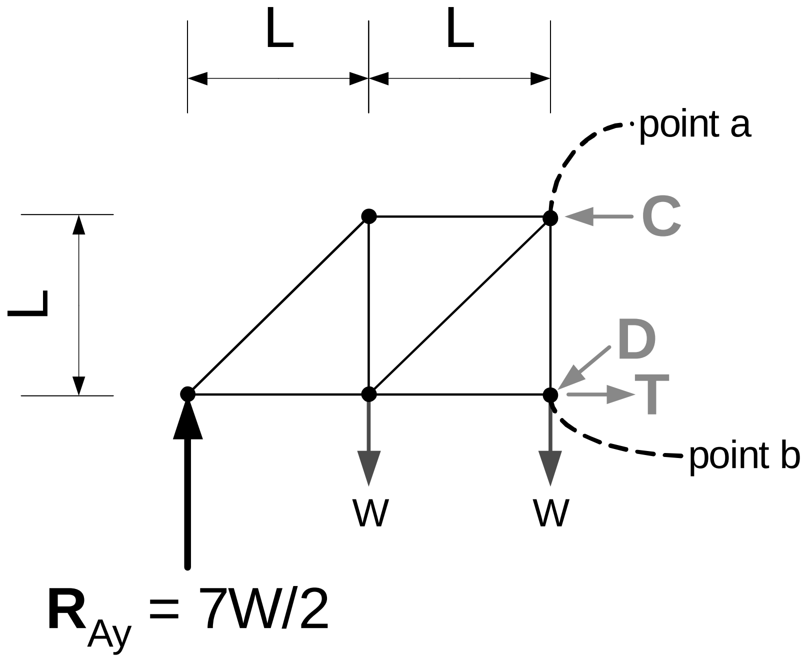


Figure 5.50 "Cut" of the 29-member truss bridge. Member forces are given at the cut location.

Applying the equations of equilibrium, we find

$$\Sigma F_y = 0 \tag{5.115}$$

$$\frac{7w}{2} - W - W - D\sin 45° = 0 \tag{5.116}$$

$$D = \frac{3W\sqrt{2}}{2} \tag{5.117}$$

$$\Sigma F_x = 0 \tag{5.118}$$

$$-C + D\cos 45° + T = 0 \tag{5.119}$$

$$\Sigma M_b = 0 \tag{5.120}$$

$$CL = 6WL \tag{5.121}$$

$$C = 6W \tag{5.122}$$

Combining (5.117), (5.119), and (5.122), we find

$$T = \frac{15W}{2} \tag{5.123}$$

Returning to Figure 5.50, we see that the vertical component of force D, $D\cos 45°$, must bear the shear loading. This is because C and T do not have components in the vertical direction.

$$V_y = -D\cos 45° = -\left(\frac{3W\sqrt{2}}{2}\right)\left(\frac{\sqrt{2}}{2}\right) = -\frac{3W}{2} \tag{5.124}$$

which matches exactly with the result found in (5.113).

We also see that the top, bottom, and diagonal members bear the moment loading. Taking the moment about any point along the section cut will yield the net section moment:

$$\Sigma M_b = CL = 6WL \tag{5.125}$$

$$\Sigma M_a = TL - D\cos 45° L = \frac{15W}{2}L - \frac{3W}{2}L = 6WL \tag{5.126}$$

These results match up exactly with our results from (5.114).

In this manner, truss structures - whose members cannot individually resist shear or moment loading - can bear shear and bending loads as entire systems.

5.7 References

[1] *Britannica.com*: http://www.britannica.com, July 2001.

[2] Cook, Nathan H. *Mechanics and Materials for Design*. McGraw-Hill, New York: 1984.

[3] Cook, Robert D., David S. Malkus, and Michael E. Plesha. Concepts and Applications of Finite Element Analysis. John Wiley and Sons, New York: 1989.

[4] Crandall, Stephen H., Norman C. Dahl, and Thomas J. Lardner. *An Introduction to the Mechanics of Solids*. McGraw Hill, New York: 1978.

[5] Dildine, James P. *Heron's Calculation of Triangle Area*: http://www.mste.uiuc.edu/dildine/heron/triarea.html, August 2001.

[6] Laursen, Harold I. *Structural Analysis*. McGraw Hill, New York: 1978.

[7] Palmes, J.C. A *History of Architecture*. Charles Scribner's Sons, New York: 1975.

[8] Strang, Gilbert. *Introduction to Applied Mathematics*. Wellesley-Cambridge Press, Wellesley, MA: 1986.

[9] *Truss Bridges of Northern California*: http://community-2.webtv.net/DAROSAMARCIELII/TrussBridgesof/ , October 2001.

Supplementary Information

A.1 Two-Force Members

A two-force member has only two points at which forces are applied. Take the general body displayed in Figure A.1.

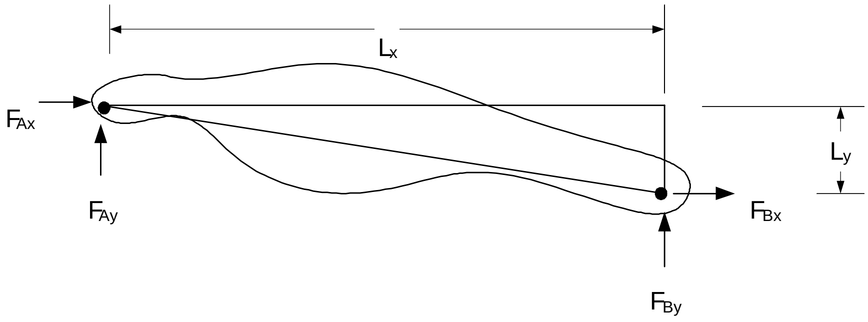


Figure A.1 General body subjected to loading at two points, A and B.

Summing forces in the x- and y- directions, we have

$$\Sigma F_x = 0 \tag{A.3}$$

$$F_{Ax} = -F_{Bx} \tag{A.4}$$

$$\Sigma F_y = 0 \tag{A.5}$$

$$F_{Ay} = -F_{By} \tag{A.6}$$

Summing moments about point A,

$$\Sigma M_A = 0 \tag{A.7}$$

$$F_{Bx}L_y + F_{By}L_x = 0 \tag{A.8}$$

$$F_{Bx} = -\frac{L_x}{L_y}F_{By} \tag{A.9}$$

$$F_{By} = -\frac{L_y}{L_x}F_{Bx} \tag{A.10}$$

Thus, moment equilibrium shows that the line of action of F_B must pass through A as shown in Figure A.2.

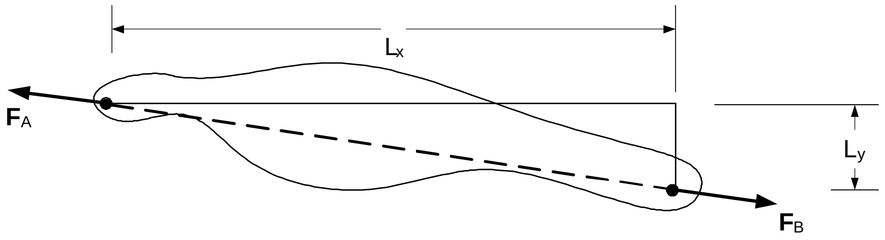


Figure A.2 General body as a two-force member.

This yields the definition of a two-force member: a member that is acted on by two forces $\mathbf{F_A}$ and $\mathbf{F_B}$, that are colinear, and equal and opposite in direction.

A.2 Assumption of Small Displacements

In determining the relation between a unit displacement in one degree of freedom and the force resulting from such a displacement in another degree of freedom, we made an assumption that δ was small in relation to L (Figure A.3).

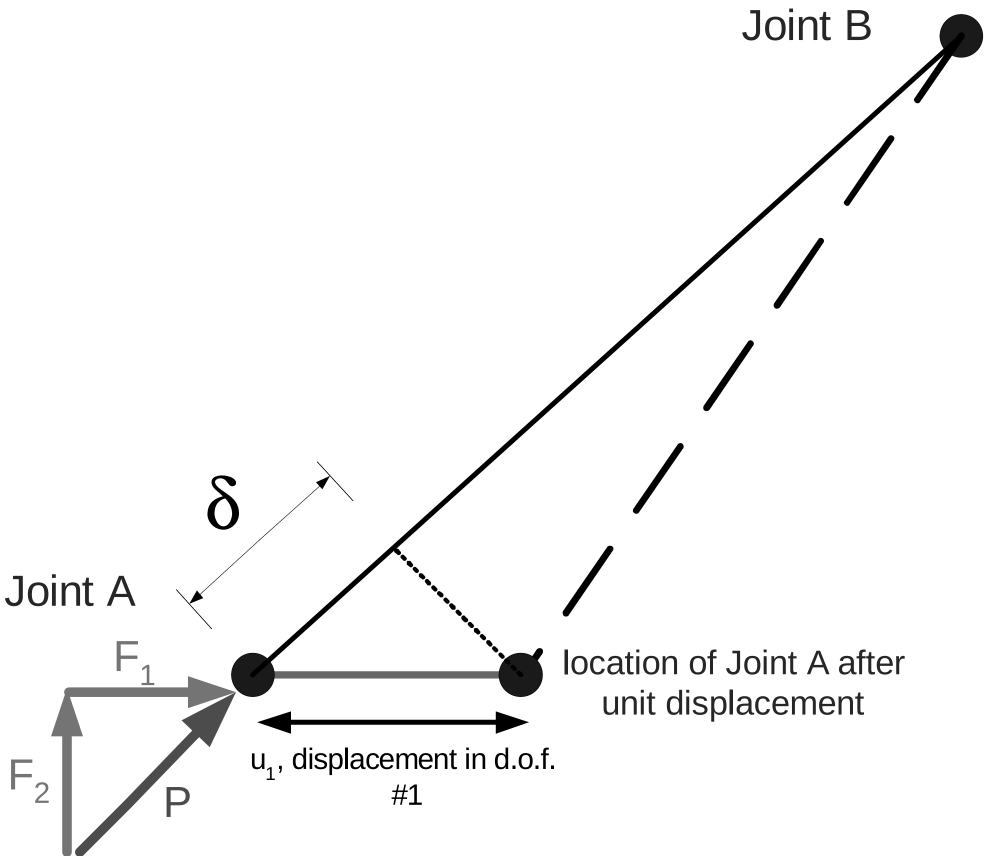

Joint B
δ
Joint A
F_1
F_2
P
location of Joint A after unit displacement
u_1, displacement in d.o.f. #1

Figure A.3 General member AB is subjected to a unit displacement in d.o.f. #1. this results in a compressive force of magnitude P within the member. The x and y components of P are F_1 and F_2, respectively.

The approximation was in the calculation of δ, which we stated was equal to $u_1 cos\theta$. The difference between the actual δ and the assumed value of δ is illustrated in Figure A.4.

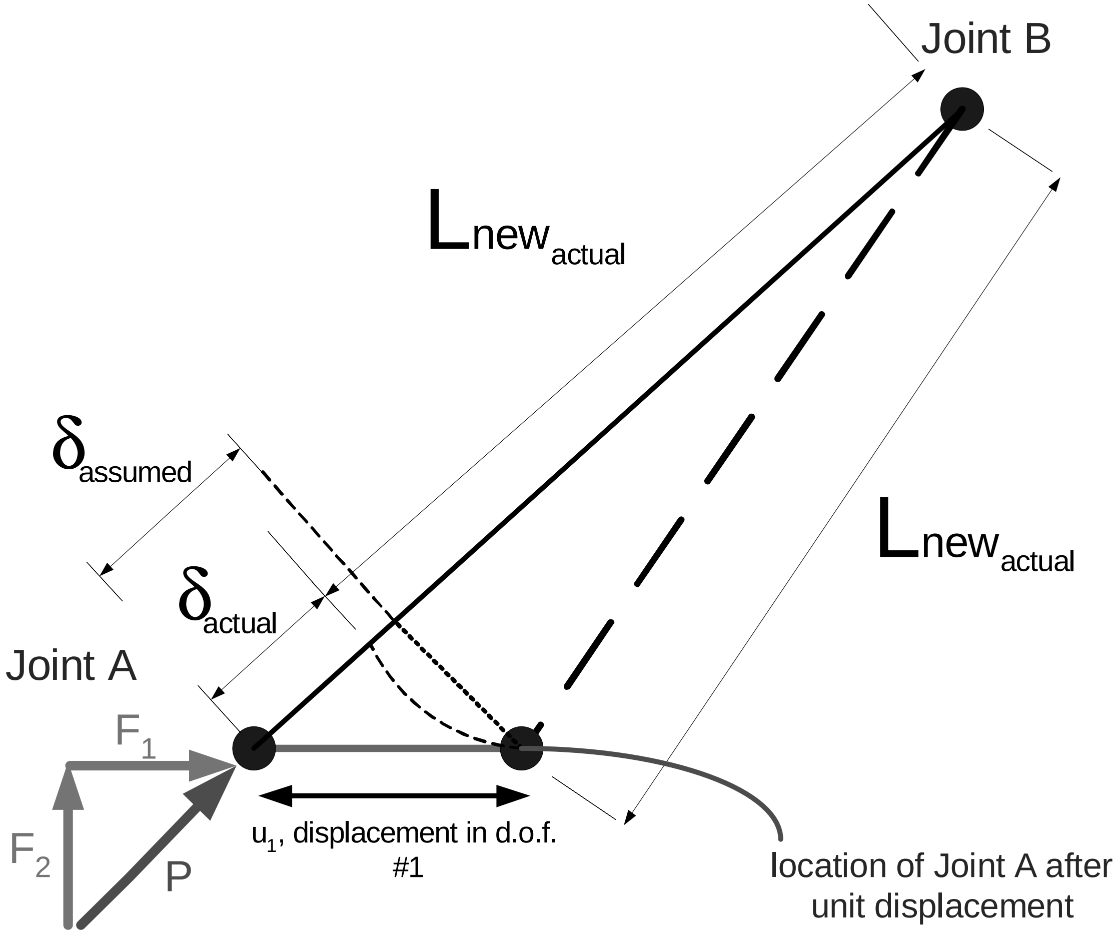


Figure A.4 $u_1 cos\theta$ is actually an approximation of the value of δ.

As can be seen, $\delta_{assumed}$, the assumed value of δ, is an upper bound on δ_{actual}, the actual value of δ. Joint A traces an arc - not a straight line - when member AB is rotated about Joint B. For small angles, however, we can approximate this arc by a straight line, as indicated. Examining the value of the assumed and actual displacements given different values of u_1 will allows us to determine the extent of validity of this approximation We have

$$\delta_{assumed} = u_1 cos\theta \tag{A.11}$$

and, from geometry

$$L_{newactual} = \sqrt{(u_1 sin\theta)^2 + (L - u_1 cos\theta)^2} \tag{A.12}$$

which gives

$$\delta_{actual} = \left|L - L_{newactual}\right| = L - \sqrt{(u_1 sin\theta)^2 + (L - u_1 cos\theta)^2} \tag{A.13}$$

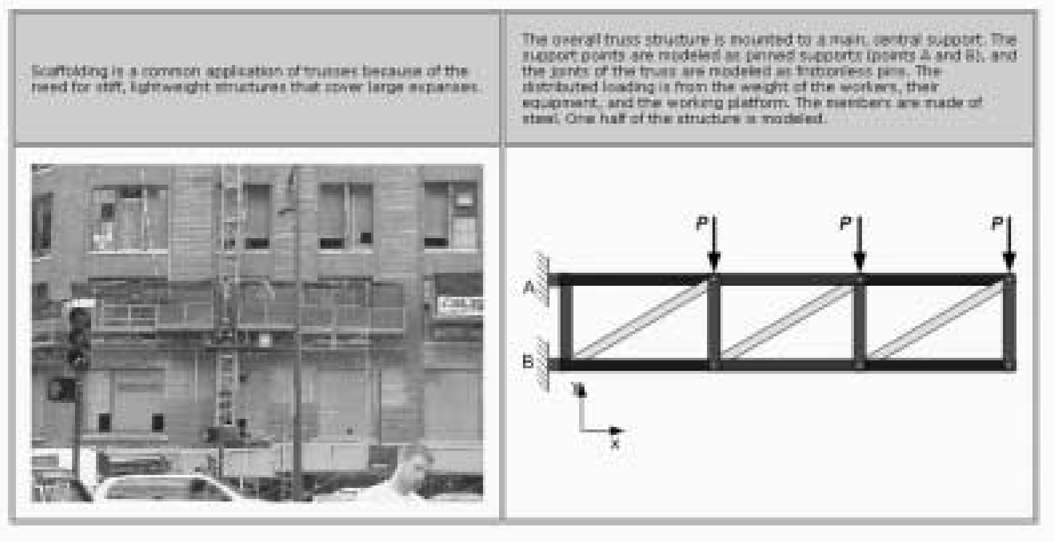
We will arbitrarily set $\theta = 45°$ so that we can easily compare the relation of $\delta_{assumed}$ to δ_{actual}. The results, as shown in Table A.1, indicate that our approximation is valid for small displacements.

TABLE A.1 Assumed and actual values of δ for different displacements u_1.

u_1	$\delta_{assumed}$	δ_{actual}	Percentage difference
0.005L	0.00353L	0.00353L	0.18%
0.01L	0.00707L	0.00705L	0.36%
0.025L	0.0177L	0.0175L	0.91%
0.05L	0.0354L	0.0347L	1.87%
0.1L	0.0707L	0.0680L	3.95%

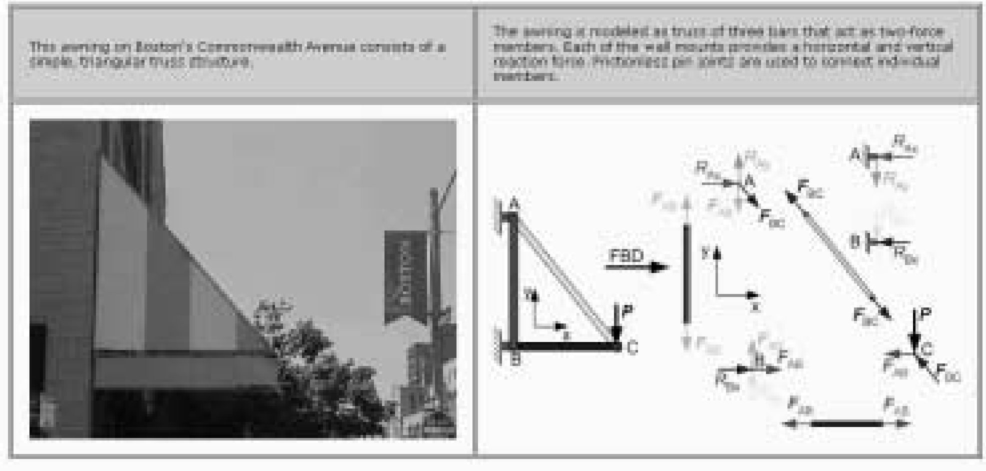
Appendix E

Real World Examples: Truss Structures



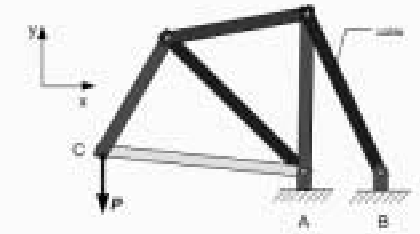
Scaffolding is a common application of trusses because of the need for stiff, lightweight structures that cover large expanses.

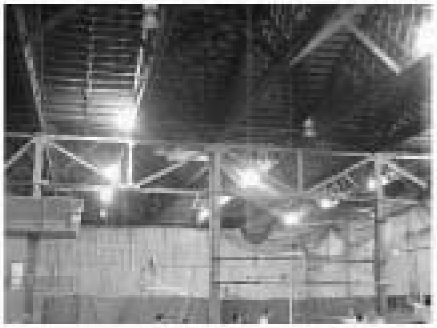
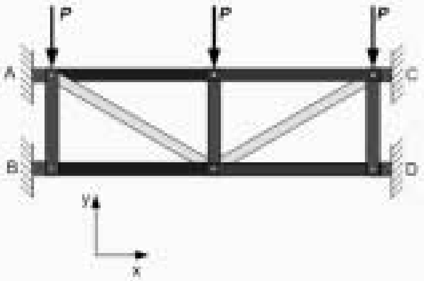
The overall truss structure is mounted to a main, central support. The support points are modeled as pinned supports (points A and B), and the joints of the truss are modeled as frictionless pins. The distributed loading is from the weight of the workers, their equipment, and the working platform. The members are made of steel. One half of the structure is modeled.

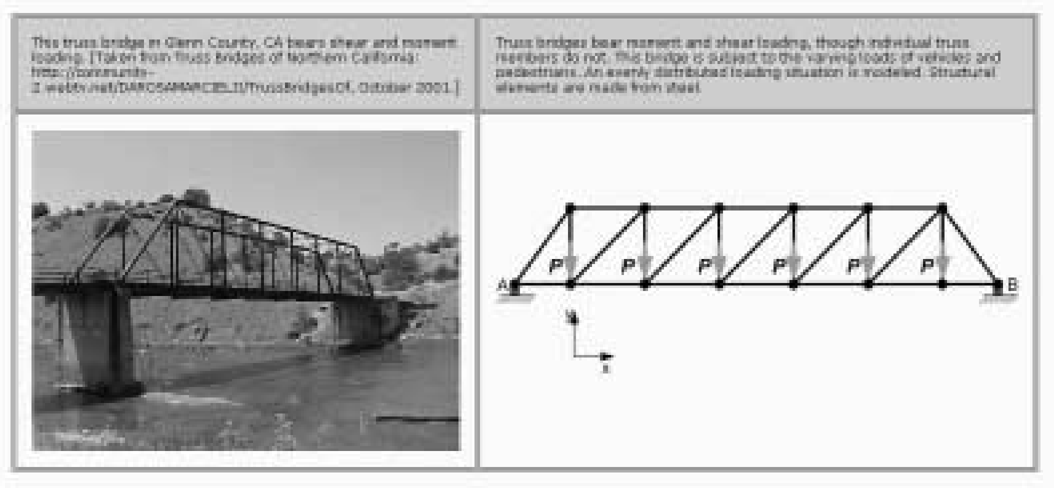


This awning on Boston's Commonwealth Avenue consists of a simple, triangular truss structure.

The awning is modeled as truss of three bars that act as two-force members. Each of the wall mounts provides a horizontal and vertical reaction force. Frictionless pin joints are used to connect individual members.

Construction equipment is a common application of truss structures.	Strictly speaking, this structure is not a truss. However, its loading condition is mostly in the axial direction and the structure can therefore be idealized as a truss. The upper part of the structure is connected to the main support with one pinned joint (Point A). A cable is attached at the right side of the truss (Point B).
	

This truss structure provides the support for the roof in Dupont.	The overall truss structure that connects the main supports can be modeled as a truss segment as shown. The main supports are modeled as pinned supports (points A, B, C, and D), the bolted joints of the truss as frictionless pins. The load from the roof is modeled as three concentrated loads. The structural members of the truss are made from steel I-Beams.
	



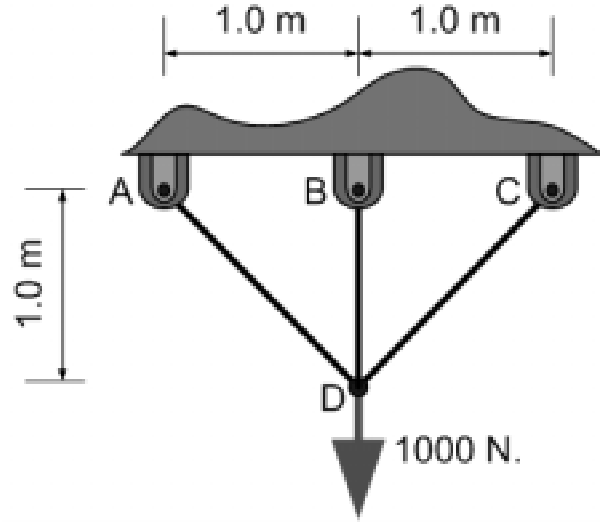
This truss bridge in Glenn County, CA bears shear and moment loading. [Taken from Truss Bridges of Northern California: http://community-2.webtv.net/DARCSAMARCELLI/TrussBridgesOf, October 2001.]	Truss bridges bear moment and shear loading, though individual truss members do not. This bridge is subject to the varying loads of vehicles and pedestrians. An evenly distributed loading situation is modeled. Structural elements are made from steel.

Appendix F

Online Quiz: Truss Structures

F.1 Book-style questions

1. The three-member planar truss shown is constructed of 10mmx10mm square aluminum bars.



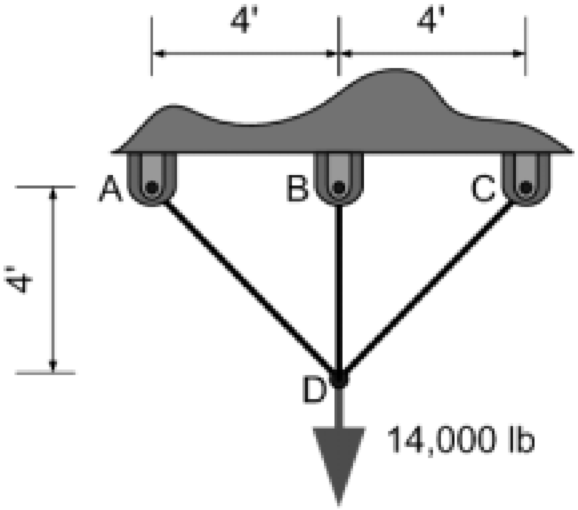
a . The internal force acting in member AD is
i. 293 N
ii. 424 N
iii. 707 N
iv. 1414 N

b . The internal force acting in member BD is
i. 424 N
ii. 586 N
iii. 1000 N
iv. 1414 N

c . The internal force acting in member CD is
i. 293 N
ii. 424 N
iii. 707 N

iv. 1414 N

2. The three-member planar truss shown is constructed of steel tubing. Each tube has a cross sectional area of 0.5 sq. in.

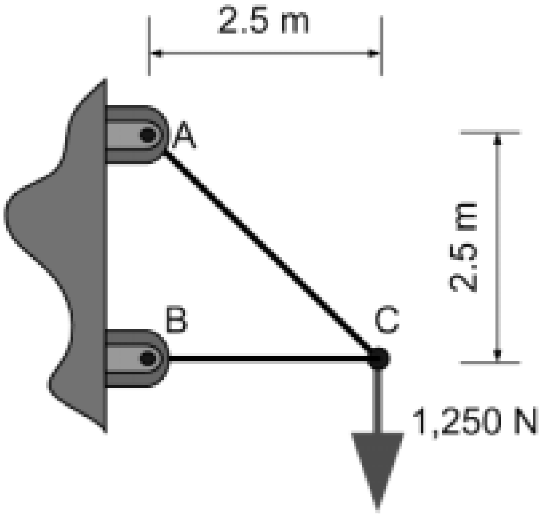


a . The internal force acting in member AD is
i. 2200 lb
ii. 4100 lb
iii. 8400 lb
iv. 9800 lb

b . The internal force acting in member BD is
i. 2600 lb
ii. 4400 lb
iii. 8200 lb
iv. 14000 lb

c . The internal force acting in member CD is
i. less than that in AD and more than that in BD
ii. more than that in AD and less than that in BD
iii. equal to that in AD and more than that in BD
iv. equal to that in AD and less than that in BD

3. The following planar truss is constructed of solid steel rods, each 1/2” in diameter. Member BC can be assumed to be perfectly horizontal.

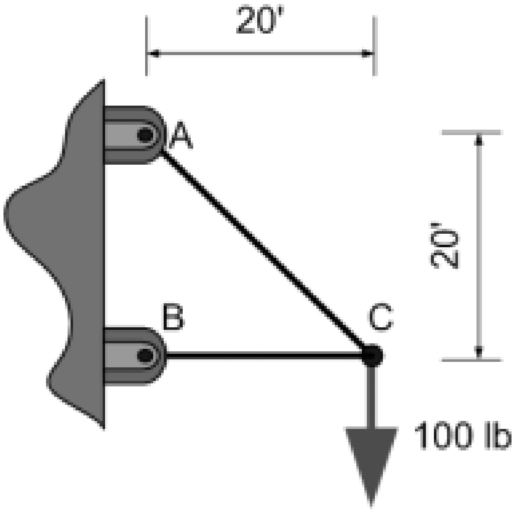


a . The internal force acting in member AC is
i. 442 N
ii. 884 N
iii. 1768 N
iv. 3536 N

b . The internal force acting in member BC is
i. 625 N
ii. 875 N
iii. 1250 N
iv. 1875 N

c . Member BC is in
i. tension
ii. compression
iii. the state of tension/compression cannot be determined with the given information

4. Suppose that the following planar truss is constructed of solid steel rods, each 1/2” in diameter. Member BC can be assumed to be perfectly horizontal.



a . The internal force acting in member AC is

i. 50 lb

ii. 71 lb

iii. 100 lb

iv. 141 lb

b . What is the horizontal deflection of node C?

i. 0.0000024”

ii. 0.000042”

iii. 0.00028”

iv. 0.0036”

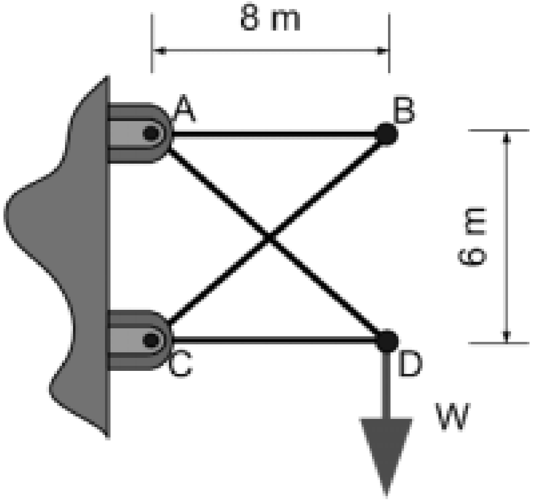
c . Now suppose that the deflection found in b is the maximum deflection allowed. How would changing the material to a HDPE (high density polyethylene) affect the weight of the structure?

i. the HDPE structure would be lighter

ii. the HDPE structure would be heavier

iii. both structures would weigh about the same (within 5%)

5. Consider the following planar truss. The elements are constructed of aluminum tubing, with an OD of 2” and an ID of 1.8”.



a . The internal force acting in member AD is

i. 0.6 W

ii. 0.8 W

iii. W

iv. 1.67 W

b . What is the horizontal deflection of node B?

i. 0.0000018 m

ii. 0.0000024 m

iii. 0.0000036 m

iv. none of the above

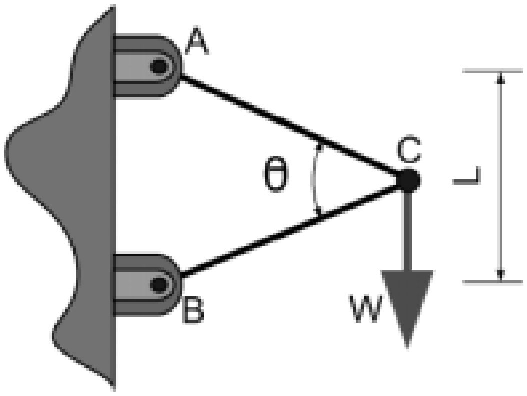
c . Which of the following is true about the structure?

i. it is statically indeterminate

ii. it has zero-force members

iii. it cannot be solved using matrix truss analysis

6. Consider the following planar truss subjected to a load W. The length between the supports is fixed at L. The angle theta can be varied between 30 and 150 degrees. The goal is to minimize the maximum member force in the structure.

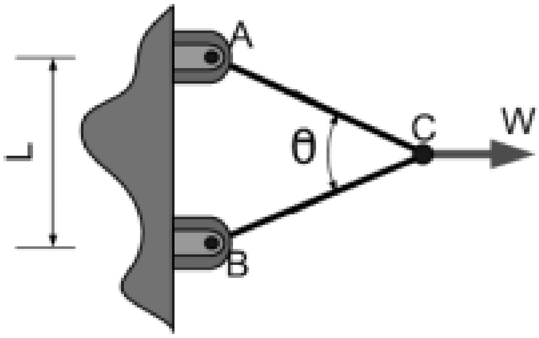


a . What angle theta produces the minimum force in AC?
i. 30 degrees
ii. 45 degrees
iii. 90 degrees
iv. 120 degrees
v. 150 degrees

b . What angle theta produces the minimum force in BC?
i. 30 degrees
ii. 45 degrees
iii. 90 degrees
iv. 120 degrees
v. 150 degrees

c . Now suppose the goal is to make the lightest structure. Assume that the maximum allowable stress (compression or tension) is S. What angle theta produces the lightest structure?
i. 30 degrees
ii. 45 degrees
iii. 90 degrees
iv. 120 degrees
v. 150 degrees

7. Consider the following planar truss subjected to a load W. The length between the supports is fixed at L. The angle theta can be varied between 30 and 150 degrees. The goal is to minimize the maximum member force in the structure.



a . What angle theta produces the minimum force in AC?

i. 30 degrees

ii. 45 degrees

iii. 90 degrees

iv. 120 degrees

v. 150 degrees

b . What angle theta produces the minimum force in BC?

i. 30 degrees

ii. 45 degrees

iii. 90 degrees

iv. 120 degrees

v. 150 degrees

c . Now suppose the goal is to make the lightest structure. Assume that the maximum allowable stress (compression or tension) is S. What angle theta produces the lightest structure?

i. 30 degrees

ii. 45 degrees

iii. 90 degrees

iv. 120 degrees

v. 150 degrees

F.2 Conceptual Questions

1. Which of the following is NOT an assumption made in treating a system as a truss structure?

i. Forces must balance

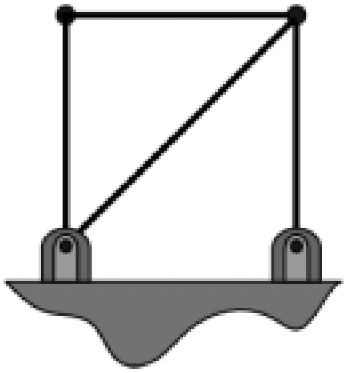
ii. Joints are frictionless

iii. Members are made of the same material
iv. Members are slender
v. All of these assumptions are made

2. Why is diamond an impractical material for a truss structure?

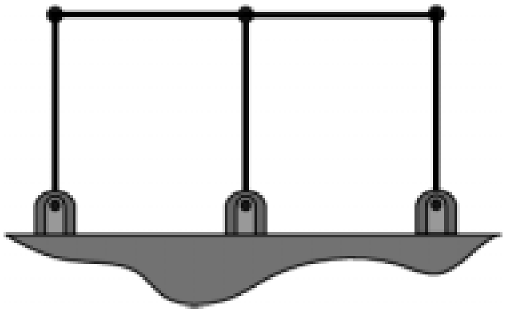
i. It is expensive
ii. It cannot readily be formed into truss members
iii. It would be too heavy
iv. a. and b.
v. a., b., and c.

3. Is the following a stable truss (consider any applied loads to occur in the plane of the truss)?



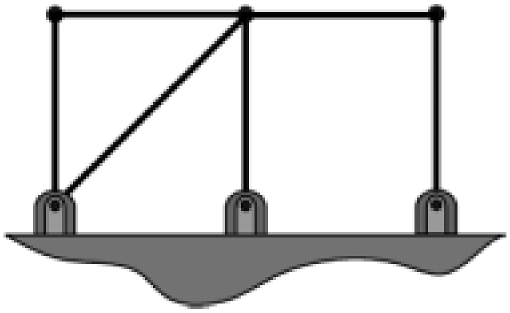
i. Yes
ii. No

4. Is the following a stable truss (consider any applied loads to occur in the plane of the truss)?



i. Yes
ii. No

5. Is the following a stable truss (consider any applied loads to occur in the plane of the truss)?



i. Yes
ii. No

6. If subjected to a specified tensile load, which of the following will deform LEAST?

i. A 48" long steel pipe with OD = 2", ID = 1.75"
ii. A solid 24" long aluminum cylindrical bar with radius of 1/2"
iii. A solid 36" long steel bar with a square 2"x2" cross-section
iv. A solid 40" long aluminum bar with a rectangular 8"x1/4" cross-section

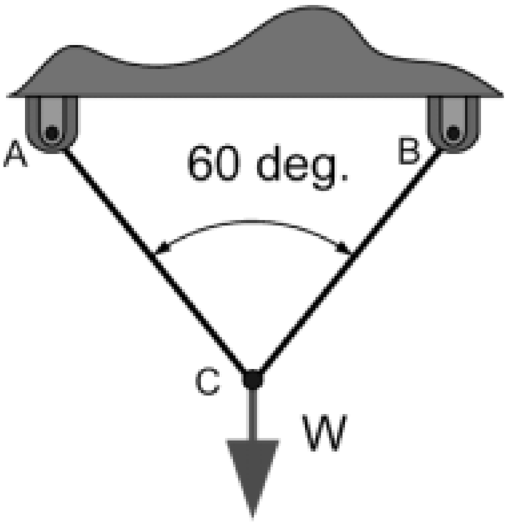
7. True or false, certain trusses have members that do not behave as two-force members.

i. True
ii. False

8. True or false, a truss joint can be fixed so that members attached at that joint cannot rotate.

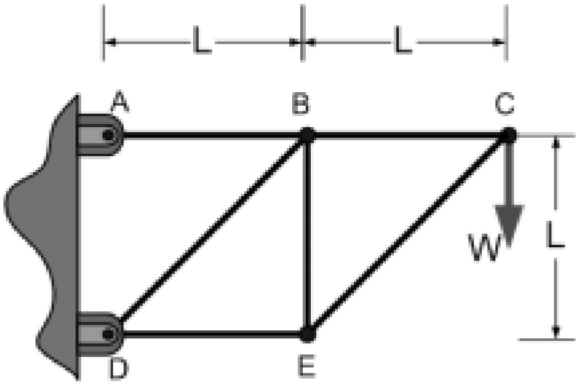
i. True
ii. False

9. The following structure has



i. One member in tension and one in compression
ii. Two members in tension
iii. Two members in compression
iv. none of the above

10. The following structure has



i. One member in compression

ii. Two members in compression
iii. Three members in compression

11. The axial load-deformation behavior of an individual truss element is based on its

i. Material
ii. Length
iii. Cross-sectional shape
iv. All of the above
v. a. and b. only

12. The stiffness of an individual truss element is given by

i. E/AL
ii. EA/L
iii. L/EA
iv. LE/A

13. Truss systems where the internal forces can be found using only the method of joints

i. are statically indeterminate
ii. are statically determinate
iii. cannot be solved using matrix truss analysis
iv. a. and c.
v. b. and c.

14. True or false, truss matrix analysis can be used to solve statically indeterminate systems.

i. True
ii. False

15. If subjected to a specified tensile load, which of the following will deflect MOST?

i. A solid 2.0 m long steel rod with a radius of 0.5 cm
ii. A solid 2.0 m long steel bar with a square 2 cm x 2 cm cross-section
iii. A solid 2.0 m long aluminum bar with a square 2 cm x 2 cm cross-section

iv. A solid 2.0 m long aluminum rod with a radius of 2 cm

16. In the planar case, an individual truss member can have as many as ___ degrees of freedom.

i. 1
ii. 2
iii. 3
iv. 4

17. Which of the following is true about the stiffness matrix for an individual member of a truss?

i. It is symmetric
ii. The values of the matrix elements depend on the orientation of the member
iii. The values of the matrix elements are independent of material
iv. a. and b.
v. a., b., and c.

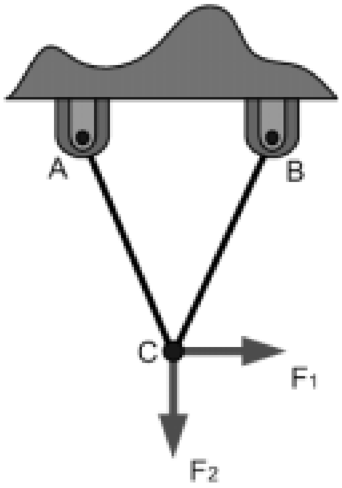
18. Which of the following is true about a truss member with 0 degrees of freedom?

i. It will translate
ii. It will rotate
iii. It will not deform
iv. a. and c.
v. b. and c.

19. A singular global stiffness matrix implies

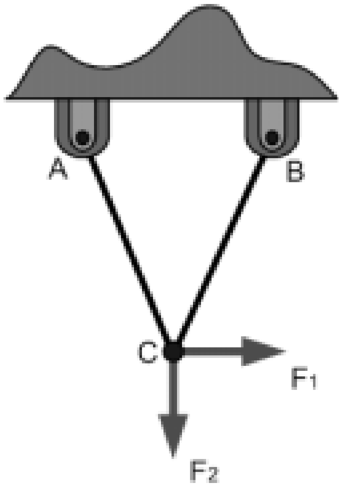
i. The structure is overconstrained
ii. The structure is underconstrained
iii. Rigid deformation modes are not possible
iv. None of the above

20. Consider the following planar truss subjected to two loads. The length of elements AC and BC are the same. Is member AC in tension or compression?



i. tension
ii. compression
iii. cannot tell given this information

21. Consider the following planar truss subjected to two loads. The length of elements AC and BC are the same. Is member BC in tension or compression?



i. tension
ii. compression
iii. cannot tell given this information

Appendix G

Design Exercise: Truss Support Structure

2.001: Mechanics of Materials I
Fall 2001
Department of Mechanical Engineering
Cambridge, MA 02139

Design Project: Truss Support Structure

October 29, 2001

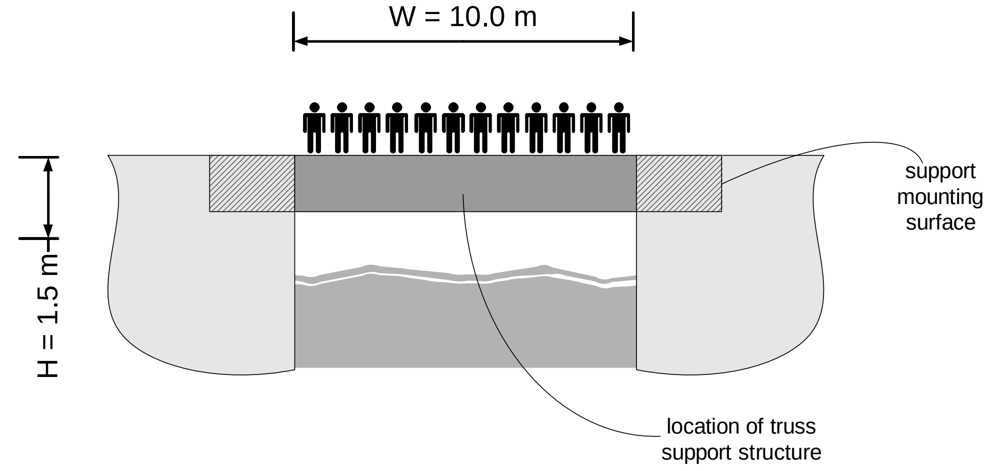
SYNOPSIS

The goal of this project is to design a truss bridge to span a moat surrounding a castle. The design goal is to provide the lowest weight structure that meets the design requirements.

DESIGN

Your job is to – in teams of 2 students – design a lightweight, two-dimensional truss structure that will meet the following design requirements. A schematic outlining the design constraints is shown in the figure below.

.



A government agency in Germany is planning to reconstruct a pedestrian bridge across a castle's moat. They have contacted you to provide the most lightweight design. The proposed method for supporting this walkway is by means of a truss structure running the length of the bridge on both sides (therefore, you may assume the bridge loading is equally shared by the truss structure on each side and just model one truss structure as a two-dimensional truss). The distance across this moat is 10.0 m, and the mounting points for supports are given in the figure above. The structure must fit within the 1.5 m x 10.0 m rectangle illustrated in the figure.

The agency has told you that they expect there to be – at a maximum – 200 people on the bridge at a time. The mass of the concrete walkway (supported by the truss structure) is 9600 kg (4.0 m wide, 12 cm thick, density of concrete = 2000 kg/m^3). The concrete walkway must be pinned to the truss structure at a minimum of four locations.

In modeling the boundary conditions you should expect there to be pinned joints on one side of the bridge, and expansion joints (constrained in the y-direction, free to slide in the x-direction) on the opposite side.

There are design constraints relating to the stiffness of the structure and to the strength of the structure:

a) Stiffness constraint: The vertical deflection of the bridge cannot exceed 0.025 m (2.5 cm) at any of the top joints (those connected to the walkway).

b) Strength constraint: You must ensure that the structure does not fail under the loading conditions. Failure is defined in this case as a situation where the stress in any truss member exceeds 65% of the yield stress of the material.[1]

c) Buckling constraint: You must also ensure that the structural members do not undergo buckling. Buckling – assuming solid circular cross-sections – is defined as a situation where the compressive stress in a column exceeds:

$$\sigma_{buckle} = \frac{(\pi^2)E}{16\left(\frac{L}{R}\right)^2}$$

where E is the Young's modulus, L is the member length, and R is the radius of the circular cross-section.[2]

Approved materials for this project are shown in the table below, along with relevant material properties. You should make reasonable assumptions about the problem in formulating your design. You may (and should) use the Truss Structures simulation in determining your recommended design.

Material	**Young's Modulus (GPa)**	**Density (kg/m^3)**	**Yield Stress (MPa)**
Aluminum	70	2700	270-480
Cast Iron	100	7000	120-300
Magnesium Alloy	45	1800	150-230
Steel	200	7850	280-1600
Titanium	110	4500	400

[1] Recall, stress in an axially loaded member is defined as the axial load divided by the cross-sectional area. Yield stress is a material-specific property; when the axial stress level reaches the yield stress, the material will begin to deform in an inelastic fashion.

[2] The quantity L/R is called the slenderness ratio. Note that the critical buckling stress decreases with the square of the length. The derivation of buckling conditions will be covered later in the course, but you should still consider this in your design.

DELIVERABLES

Each team of students must submit a handwritten design report. Each team should develop two functional designs, which have distinct geometries. Essential elements of this report include:

I. Assumptions (what are the loading conditions? how was the system modeled?)

II. Recommendations

- a. Design details (cross-sectional area, material, truss geometry, mass, etc.)
- b. Conformity to specifications
  - i. Maximum deflection
  - ii. Maximum stress

III. Discussion

- a. Advantages and disadvantages of each design
- b. How does each truss bear shear and moment loading?

Appendix H

Case Study: Truss Structures

Module 5

CASE STUDY: MICHELL TRUSS

v.1 Background

In 1904, Anthony George Maldon Michell, a mechanical and hydraulic engineer from Melbourne, Australia, published in *Philosophical Magazine* an article entitled *The Limits of Economy of Material in Frame-Structures* (Michell, Niedenfuhr). In this article, Michell introduced - and presented analytical solutions to - a set of problems of minimizing the weight of structures subject to specific loading conditions. The structures that resulted from his analysis are today called *Michell structures*. The most commonly referenced of these structures describes a force applied at a point A, a specified distance from another point B, acting orthogonally to the line AB (Figure v.1).

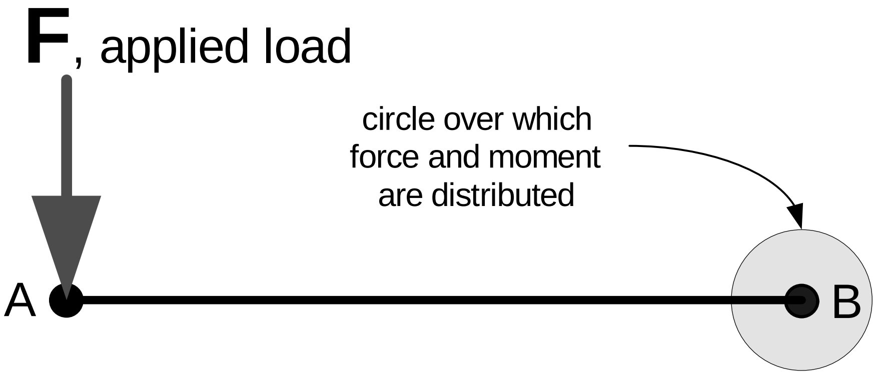
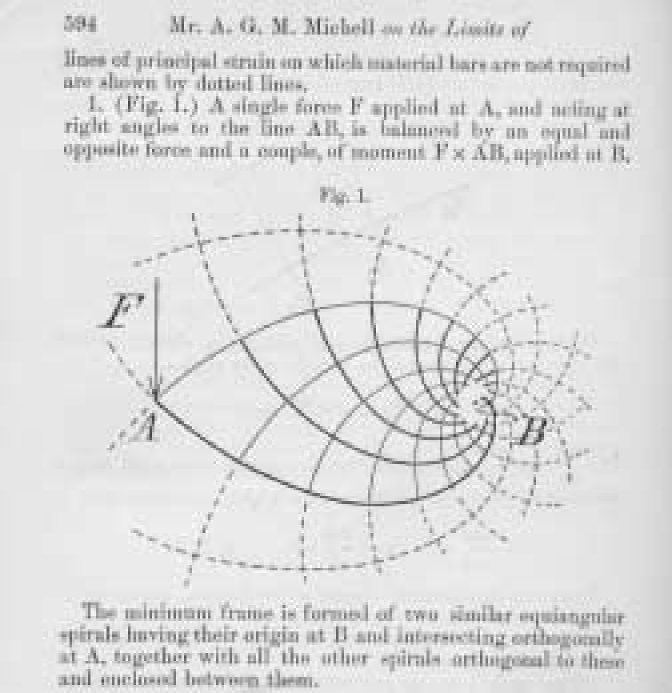


Figure v.1 Formulation of the common Michell structure problem. A force F is applied at point A, a specified distance from another point B, acting orthogonally to the line AB.

If the force and moment at B are distributed over a small circle, there is a closed-form analytical solution, as shown in Figure v.2.



594 Mr. A. G. M. Michell *on the Limits of*

lines of principal strain on which material bars are not required are shown by dotted lines.

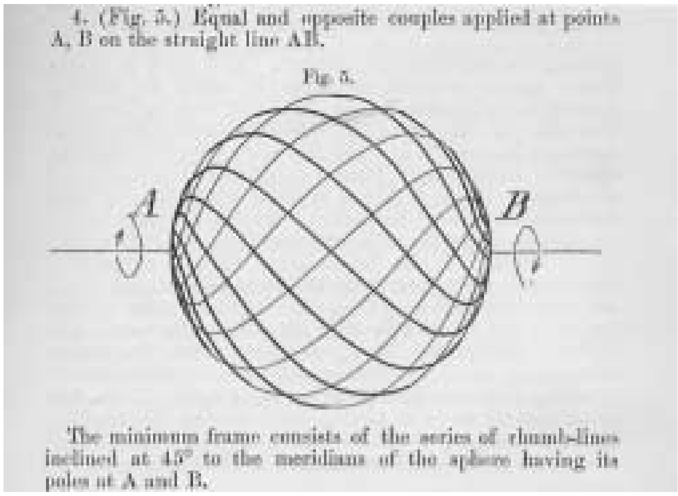
1. (Fig. 1.) A single force F applied at A, and acting at right angles to the line AB, is balanced by an equal and opposite force and a couple, of moment F × AB, applied at B.

Fig. 1.

The minimum frame is formed of two similar equiangular spirals having their origin at B and intersecting orthogonally at A, together with all the other spirals orthogonal to these and enclosed between them.

Figure v.2 "A single force F applied at A, and acting at right angles to the line AB, is balanced by an equal and opposite force and a couple, of moment F x AB, applied at B." [This figure is taken from Michell, A.G.M. *Limits of Economy of Material in Frame-Structures. Philosophical Magazine 6 (8), 589-597 (1904).*]

Michell also presented solutions for several other problems, in the context of economy of material, including the centrally loaded, simply supported beam, and the structure needed to resist equal and opposite couples along a straight line (Figure v.3).



4. (Fig. 5.) Equal and opposite couples applied at points A, B on the straight line AB.

The minimum frame consists of the series of rhumb-lines inclined at 45° to the meridians of the sphere having its poles at A and B.

Figure v.3 "Equal and opposite couples applied at points A, B on the straight line AB." [This figure is taken from Michell, A.G.M. *Limits of Economy of Material in Frame-Structures. Philosophical Magazine 6 (8), 589-597 (1904).*]

Michell’s solutions to these problems have been used to benchmark the performance of design optimization programs. They have also provided the basis for the study of structural optimization. Rozvany (1972) and others used Michell’s findings to explore analytical solutions to other structural optimality problems (Cox 1963, Hemp 1975, Dewhurst 2001). More recently, structural optimization has advanced further on modern computation. Recent research has examined several possibilities such as the idea of a “soft variation of material density between the limit states ’empty’ and ’solid’" (Mlejnek, 2000). Instead of treating the structure as a problem of assigning regions to be filled with material or to be empty, Bendsoe presented the idea of replacing these empty regions with less dense material. This more pragmatic approach has found applications in the actual design of least-weight linear-elastic structures and the creation of advanced materials. Other research has investigated the possibility of incorporating non-structural constraints (such as aesthetic constraints or the need to be compatible with other parts of a structure) into design optimization programs (Kim, 1998). We will examine a variation on the problems presented by Michell in 1904 and ascertain the effectiveness of various truss structures in providing reasonable solutions to the problem.

v.2 Problem Statement

This variation on the Michell structure problem centers around the creation of a truss structure that is attached to a wall within a confined area. Referring to Figure v.4, the structure is subjected to a load, P, a specified distance away from the wall and must meet a minimum level of structural stiffness. The structural stiffness is defined as $k = \frac{P}{\delta}$, where P is the applied load, and δ is the vertical displacement at the point of load application. The entire structure must fit within a bounding rectangle. The goal in creating these structures is to minimize the overall weight while meeting the required stiffness. The problem is illustrated below (Figure v.4).

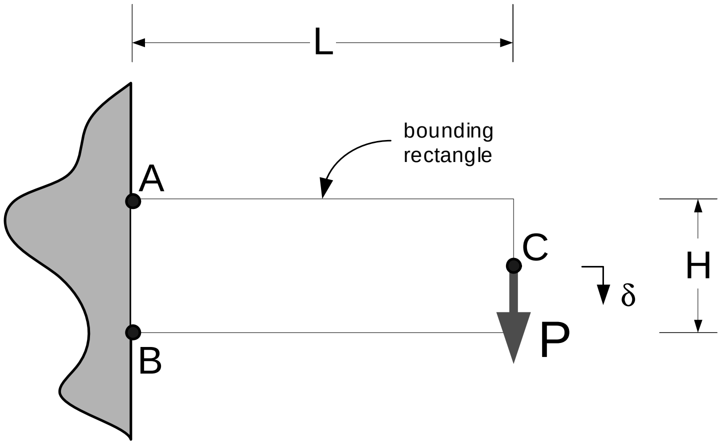


Figure v.4 General Michell truss problem illustration.

Though dimensions are not particularly critical to the concept of the problem, we will consider specific dimensions to provide a concrete problem for discussion. The structure constraints are as follows:

- The overall length of the structure, L, is fixed. Here we fix L = 3.0 m.
- The overall height of the structure is fixed. Here we fix H = 1.0 m.
- The structure is required to have a vertical deflection, δ, at point C less than 0.02 m ($\delta < 0.02$ m) when subjected to a vertical load P = 100 kN. Therefore, the required structural stiffness $k = \frac{P}{\delta} > 5000$ kN/m.

The figure below shows the problem constraints (Figure v.5).

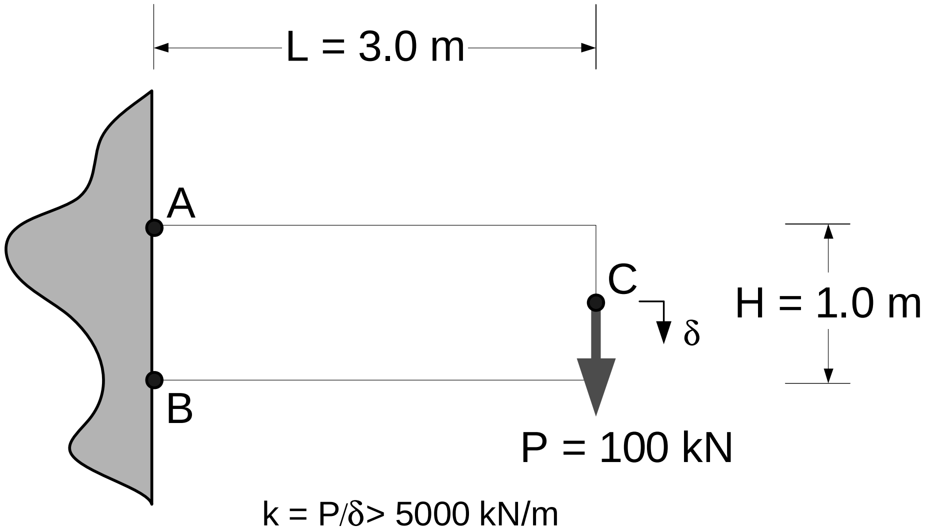


Figure v.5 Michell truss problem with specific parameters. The concept of this problem is not dependent on specific dimensions and loads, but a concrete construction will give us a more solid basis for discussion.

The task before us is to define the structure that will meet the design requirements while minimizing weight. The difficulty is in the number of parameters that may be varied: material, geometry, and boundary conditions. With no prior knowledge of the problem, we might reasonably expect our solution to have only 2 members, or to have thousands. Below, we first choose a material and then move on to design of the truss geometry.

v.3 Material Selection

Material selection is a critical choice in designing a structure. For the given problem, we will consider all truss members to be made from the same material. Material affects both the stiffness characteristics of the truss, as well as the weight. There are numerous materials that could be selected. For the current problem, we would like to select a material that will meet the stiffness requirement and also provide a lightweight structure.

If we examine the case of an individual member, we see that the member axial stiffness is given by

$$k = \frac{P}{\delta} = \frac{AE}{L} \qquad \text{(v.1)}$$

where A is the member cross-sectional area, E is the Young's modulus of the member material, and L is the member length.

We wish to meet a given member stiffness (call it k*) while minimizing the bar mass. The mass of the bar is given by

$$m = \rho \times V = \rho(A)(L) \qquad \text{(v.2)}$$

where ρ is the material density.

We combine equations (v.1) and (v.2); A is common to both equations, so we can use these equations to eliminate it. The result is an equation relating m, k*, and the material properties:

$$\mathrm{m} = \mathrm{k}^{*} L^2\left(\frac{\rho}{E}\right) \qquad \text{(v.3)}$$

Since we wish to minimize m for a particular stiffness k*, and since L is a constant for an individual bar, we need to minimize the quantity $\frac{\rho}{E}$, or, alternatively, maximize the quantity $\frac{E}{\rho}$. We will do so by selecting the material that maximizes the quantity $\frac{E}{\rho}$.

A material selection chart is shown in Figure v.6. This is a plot of E vs. ρ on a log-log scale. Ideally, a material with a high E and a low ρ (i.e. near the upper left corner of the plot) is desired, but does not exist.

In order to understand how to read the chart, we first take the quantity $\frac{E}{\rho}$ to be a constant.

$$\frac{E}{\rho} = C \tag{v.4}$$

Taking the logarithm of both sides,

$$log\frac{E}{\rho} = log C \tag{v.5}$$

or

$$log E - log p = log C \tag{v.6}$$

We can rewrite this as,

$$log E = log p + log C \tag{v.7}$$

Note that this is an equation of a line, of the form

$$y = mx + b \tag{v.8}$$

since we are plotting logE as the y-coordinate, and logρ as the x-coordinate (m = 1 in this case). This implies that lines of slope 1 will represent materials with identical values of $\frac{E}{\rho}$. The greater the y-intercept of a particular line (logC), the greater value of $\frac{E}{\rho}$ it will represent.

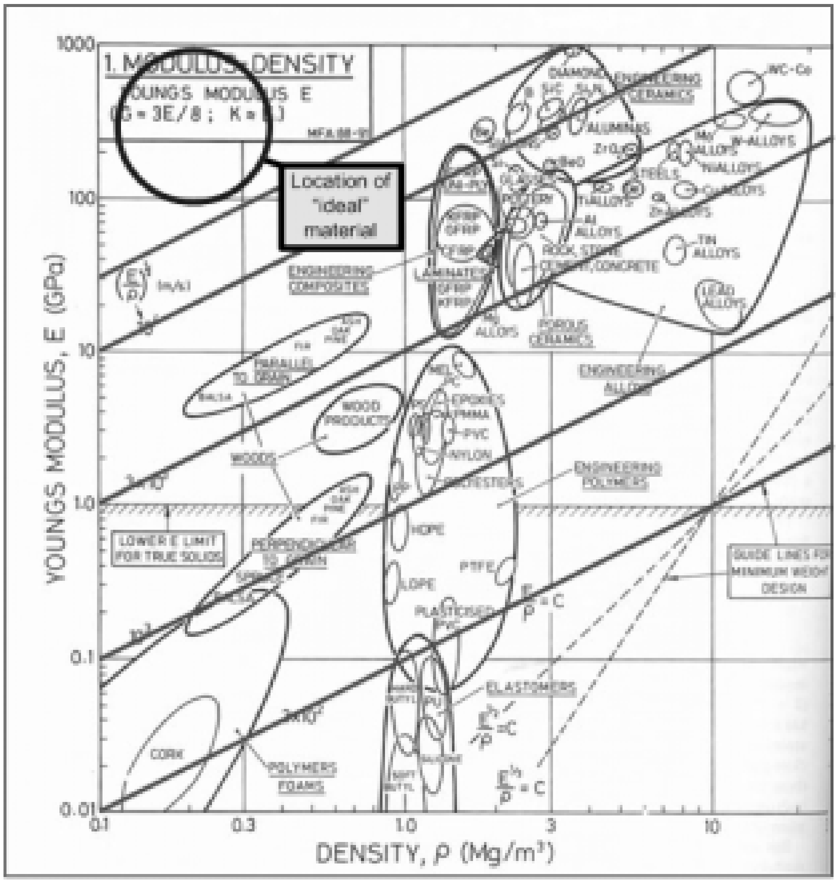


Figure v.6 This material selection chart plots Young's modulus (E) vs. density (ρ) on a log-log scale. The ideal material for our purpose is a material near the upper left of the plot, as indicated in blue. Lines representing constant E/ρ are shown in red. [This chart is taken from Ashby, M.F. *Materials Selection in Mechanical Design*. Butterworth-Heineman, Oxford: 1992.]

As is clear from the Figure v.6, we wish to select materials that are closest to the top left corner of the plot. Using this method, diamond and ceramics appear to be good candidates; however, we must recall that we are designing a truss structure. Diamond is expensive, and it cannot be formed into truss members (slender members that can be pinned at the ends). We can reasonably expect many of our truss members to undergo tensile loading; while ceramics display desirable stiffness characteristics, they fail under low tensile loads; even though we are not considering strength in out selection criteria now, we should stay away from low tensile strength materials. Considering the final application of our material, and the usability of the material we might select steel, aluminum alloys, CFRP (Carbon-Fiber Reinforced Polymer), or wood (parallel to the grain). Properties of these materials are given in Table v.1.

TABLE v.1 Typical properties of materials with high values of E/ρ. PVC, which has a much lower value of E/ρ, is given as a reference point.

Material	**E (GPa)**	**ρ (kg/m^3)**	**(E/ρ)x10^{-9} N-m/kg**
Steel	190-210	7850	0.0242-0.0268
Aluminum (6061-T6)	70	2700	0.0259
CFRP	50	1500	0.0333
Douglas fir (parallel to grain)	13	550	0.0236
PVC (high impact)	2.4	1380	0.00174

For the problem at hand we will choose an aluminum alloy as our material. We will then attempt to minimize the overall mass of the truss through the geometric design of the truss, which shall meet the required truss stiffness k = 5000 kN/m.

v.4 Design of Truss Geometry: Two-Member Truss

As a first attempt at a minimum weight truss design, we will begin with a simple, symmetric two-member truss (Figure v.7). We expect that the best orientation of the members for a symmetric two-member truss is with the ends fixed at points A and B, so we will examine this case first.

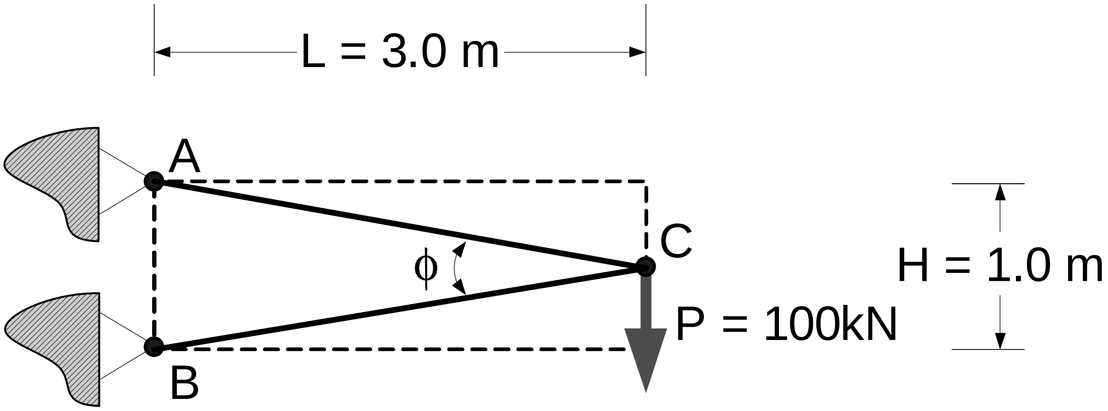


Figure v.7 Two-member truss configuration with ends pinned at points A and B.

We will analyze this truss using matrix analysis. Though this may seem tedious at this point, it will soon become clear that this method will simplify the process of analysis. First, it is necessary to identify local and global degrees of freedom. In constructing the global stiffness matrix, we will only consider active, or unconstrained degrees of freedom (Figure v.8).

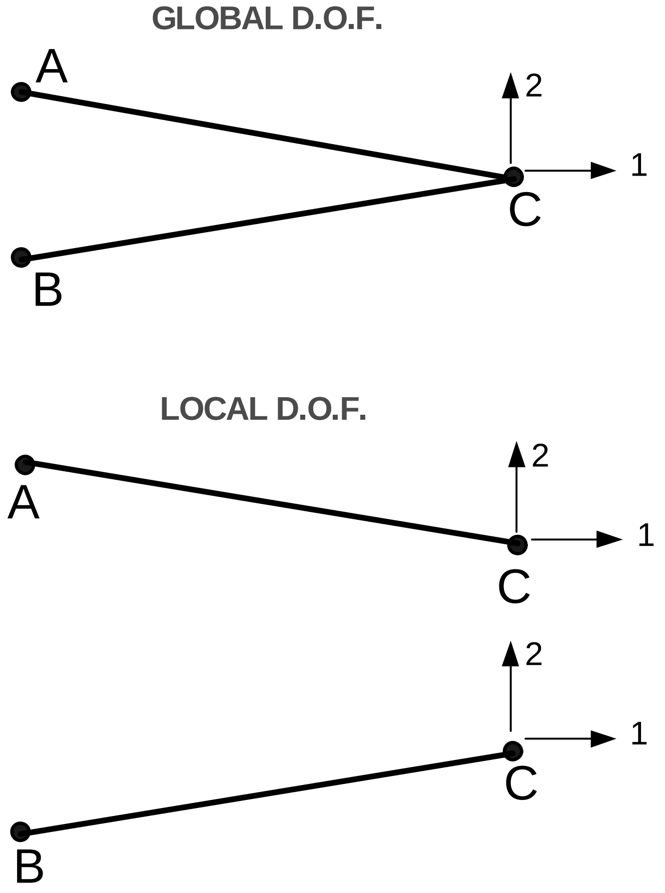


Figure v.8 Global and local degree of freedom (d.o.f.) assignments for two-member symmetric truss (active d.o.f. only).

We construct a table indicating the relation of global degrees of freedom to local degrees of freedom (Table v.2).

TABLE v.2 Relation of global d.o.f. to local d.o.f. for the symmetric two-member truss.

Global degree of freedom (d.o.f.)	Corresponding local d.o.f.
#1	Member AC, #1 Member BC, #1
#2	Member AC, #2 Member BC, #2

We use this information to construct the global stiffness matrix.

$$[K_G] = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix}^G = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix}^{AC} + \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix}^{BC} \tag{v.9}$$

$$[K_G] = \left(\frac{AE}{L}\right)_{AC} \begin{bmatrix} C_{AC}C_{AC} & C_{AC}S_{AC} \\ C_{AC}S_{AC} & S_{AC}S_{AC} \end{bmatrix}^{AC} + \left(\frac{AE}{L}\right)_{BC} \begin{bmatrix} C_{BC}C_{BC} & C_{BC}S_{BC} \\ C_{BC}S_{BC} & S_{BC}S_{BC} \end{bmatrix}^{BC} \tag{v.10}$$

where the quantities C_{AC}, S_{AC}, C_{BC}, and S_{BC} are found from the angle that the member makes with the x-axis:

$$C_{AC} = \cos\left(180° - \frac{\phi}{2}\right) \tag{v.11}$$

$$S_{AC} = \sin\left(180° - \frac{\phi}{2}\right) \tag{v.12}$$

$$C_{BC} = \cos\left(\frac{\phi}{2}\right) \tag{v.13}$$

$$S_{BC} = \sin\left(\frac{\phi}{2}\right) \tag{v.14}$$

L is given from geometry by

$$L = \frac{3.0}{\cos\left(\frac{\phi}{2}\right)} \tag{v.15}$$

E is known to be 70 GPa, since all truss members will be made from aluminum. For now, we will assume the area of all members to be the same.

The resulting global stiffness matrix is:

$$[K_G] = \frac{AE}{L} \begin{bmatrix} C_{AC}^2 + C_{BC}^2 & C_{AC}S_{AC} + C_{BC}S_{BC} \\ C_{AC}S_{AC} + C_{BC}S_{BC} & S_{AC}^2 + S_{BC}^2 \end{bmatrix} \tag{v.16}$$

We then use this in combination with our loading conditions to construct a relation among the applied loads and the displacements at joint C.

We know that $\{F\} = [K_G]\{u\}$, so we have

$$\begin{bmatrix} 0 \\ 100,000 \end{bmatrix} = \frac{AE}{L}\begin{bmatrix} C_{AC}^{2} + C_{BC}^{2} & C_{AC}S_{AC} + C_{BC}S_{BC} \\ C_{AC}S_{AC} + C_{BC}S_{BC} & S_{AC}^{2} + S_{BC}^{2} \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{v.17}$$

For the case where the truss is pinned at points A and B (Figure v.7), we have

$$\begin{bmatrix} 0 \\ 100,000 \end{bmatrix} = \frac{A(70x10^{9})}{3.0414}\begin{bmatrix} 1.9459 & 0 \\ 0 & 0.05406 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{v.18}$$

Given the problem constraints, we know that u_2 (the vertical deflection at point C), must be no greater than 0.02 m. From equation (v.18), we find that the member area must be 40.19 cm^2.

Note that we did not account for truss weight in our analysis. The mass of this truss design is only 66.0 kg, which is much smaller in magnitude than the end load. This justifies neglecting weight in the analysis. Should we encounter trusses that are heavier than this one, weight may become a contributor to the system displacement; however, since we are searching for the optimal (lightest) truss, trusses heavier than this one would not be considered. Therefore, we can safely neglect weight in our future analyses. Currently, this two-member symmetric truss is our optimal design.

Now, we will remain with the symmetric two-member truss design and examine the effect of varying the angle ϕ on the result, as shown in Figure v.9.

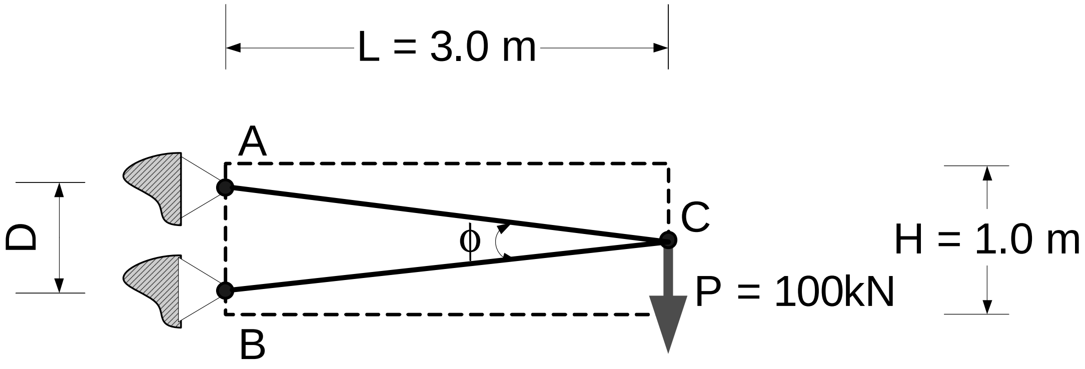


Figure v.9 Two-member truss with variable angle ϕ .

We assume the area to be the same for all members, and vary ϕ between 0^o and its maximum value of 18.92^o. Using (v.17), a spreadsheet program is used to organize the procedure of calculation. Selected results are given in Table v.3. A chart graphically indicating the results is given in Figure v.9.

TABLE v.3 Two-member symmetric Michell trusses, varying configurations.

Angle ϕ	Spacing, D, between 2 supports (m)	Area of truss member (cm^2)	Truss mass (kg)
18.92^o	1.0	40.19	66.0
15^o	0.79	63.43	103.6
10^o	0.52	141.6	230.2
5^o	0.26	563.7	914.0
2^o	0.10	3518.2	5700.3

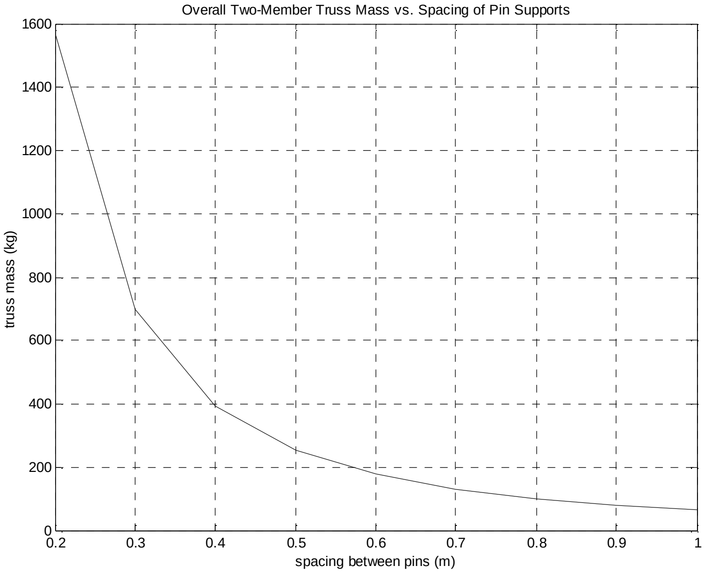


Figure v.10 Chart plotting angle between members vs. truss mass for a 2-member symmetric truss designs (19 points). 18.92^{o} is the maximum spacing allowed given the problem constraints.

An obvious pattern emerges: the mass decreases as the spacing increases. This leads us to conclude that the best orientation for a symmetric two-member truss is with the pins at the outer edges of the confined space along the wall (at points A and B).

We can also vary the area of the two members in an attempt to make the structure lighter. Since we have already determined the optimal loading situation to be the one where the members are pinned at the extreme edges of the confined space, we will vary the areas of the 2 members in this case in an attempt to lighten the structure. We will arbitrarily pick a cross-sectional area for the top member, determine the minimum required area for the bottom member in order to meet the stiffness requirement, and calculate the mass of the structure.

We will use the following general form of equation (v.17) in order to do so.

$$\begin{bmatrix}0\\100,000\end{bmatrix} = \frac{E}{L}\begin{bmatrix} A_{AC}C_{AC}^{2} + A_{BC}C_{BC}^{2} & A_{AC}C_{AC}S_{AC} + A_{BC}C_{BC}S_{BC} \\ A_{AC}C_{AC}S_{AC} + A_{BC}C_{BC}S_{BC} & A_{AC}S_{AC}^{2} + A_{BC}S_{BC}^{2} \end{bmatrix}\begin{bmatrix}u_1\\u_2\end{bmatrix} \tag{v.19}$$

A spreadsheet program is again used to simplify the process of computation. We will vary the area of the upper member between 40 cm^2 (the value for the optimal symmetric two-member truss) and 80 cm^2 (the value at which the top member by itself will weigh 66 kg). The following results are obtained

TABLE v.4 Two-member Michell truss, members with different cross-sectional areas

Area of upper member (cm^2)	Area of lower member (cm^2)	Truss mass (kg)
80.0	26.9	87.8
75.0	27.5	84.2
70.0	28.2	80.6
65.0	29.1	77.3
60.0	30.2	74.1
55.0	31.7	71.2
50.0	33.6	68.7
45.0	36.4	66.8
40.0	40.4	66.0

Again, we see that the symmetric two-member truss is the optimal design. Varying the area actually has negative effects on the overall mass in this case.

In examining two-member trusses, we have only looked at members of the same length. Now, we will examine the two-member case where the members are of different lengths, as shown in Figure v.11.

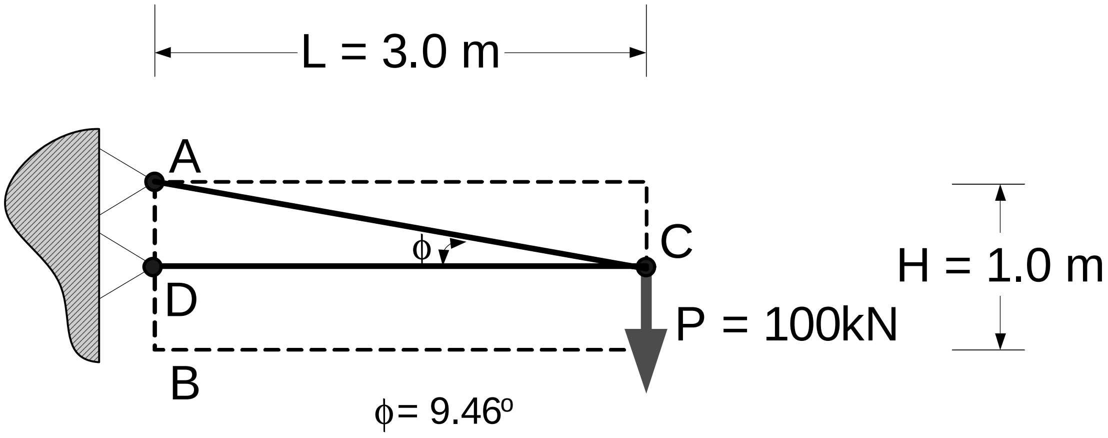


Figure v.11 An asymmetric two-member truss design.

We set up the matrix relation for this structure by considering only the active degrees of freedom.

$$\begin{bmatrix} 0 \\ 100,000 \end{bmatrix} = E\begin{bmatrix} \frac{A_{AC}C_{AC}^2}{L_{AC}} + \frac{A_{CD}C_{CD}^2}{L_{CD}} & \frac{A_{AC}C_{AC}S_{AC}}{L_{AC}} + \frac{A_{CD}C_{CD}S_{CD}}{L_{CD}} \\ \frac{A_{AC}C_{AC}S_{AC}}{L_{AC}} + \frac{A_{CD}C_{CD}S_{CD}}{L_{CD}} & \frac{A_{AC}S_{AC}^2}{L_{AC}} + \frac{A_{CD}S_{CD}^2}{L_{CD}} \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \qquad \text{(v.20)}$$

If we assume the member area to be the same for the top and bottom members, we find the minimum member area to be 157.5 cm^2. The resulting truss mass is 256.9 kg. This is not an improvement over the symmetric two-member design.

We can now consider the effect of different cross-sectional areas on this design. Using equation Figure v.20 and the spreadsheet program, we fix the cross-sectional area of one of the members and interpolate to find the area of the other member that will allow the design to meet the stiffness requirement. The results are given in Table v.5.

TABLE v.5 Effect of differing cross-sectional areas on truss mass for case of two-member asymmetric design.

Area of member AC (cm^2)	Area of member CD (cm^2)	Truss Mass (kg)
100.0	393.2	400.6
120.0	233.7	287.8
140.0	181.1	261.7
157.5	157.5	256.9
180.0	139.4	260.7
200.0	129.0	268.7
220.0	121.6	279.1
240.0	116.0	291.0

As was the case with the symmetric two-member design, we see that the ideal assignment of member areas is to set both areas to be the same.

To simplify our future analyses, we will fix the cross-sectional area and vary only other parameters. As we proceed with cases of multi-member trusses, we will create trusses with the following characteristics:

- all members will be identical in cross-section and material
- the structure will be symmetric about C
- the structure will be pinned at least at points A and B

These assumptions should help narrow the search for a more optimal solution than the symmetric two-member truss.

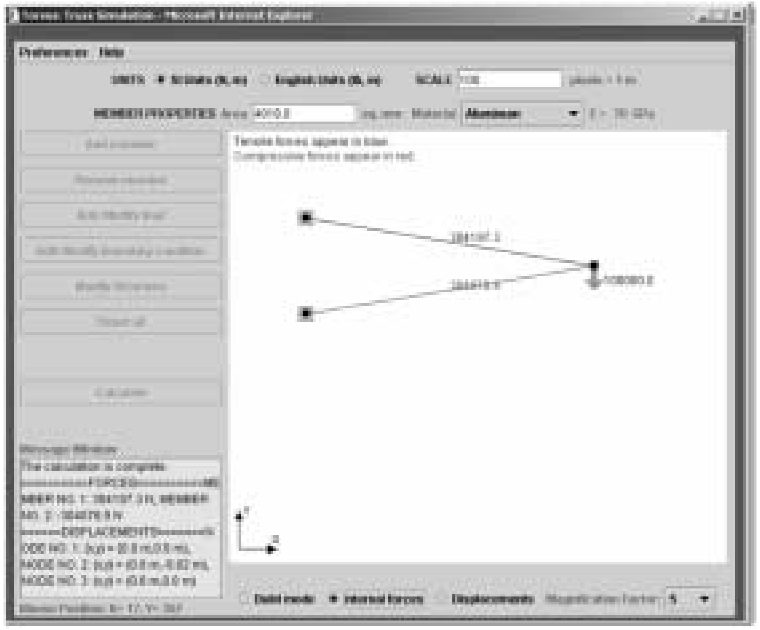
v.5 Design of Truss Geometry: Multi-member Trusses

Now we will examine several designs of multi-member truss structures that meet the requirements set above (see Problem Statement). All designs will be evaluated using the Truss Structures simulation tool.

Before we begin, we will first analyze the solution of the two-member symmetric truss using the simulation. We perform the following steps:

- choose SI units and set the scale to 100 pixels / m
- choose aluminum as a material
- set the cross-sectional area to be 1000 mm^2, or 10 cm^2
- draw the two members
- apply boundary conditions (fixed) at points A and B
- apply the loading conditions at point C (F_y = -100000)

Next, after we calculate, we determine that the vertical displacement of joint C is 0.0804 m, which is greater than allowed by the stiffness requirement. We then perform an iterative process of adjusting the member area until the vertical deflection of joint C is as close as possible (but no greater than) 0.02 m, as set by our stiffness requirement. The final solution, with a cross-sectional area of 4018 mm^2 (40.18 cm^2), is shown in Figure v.12 and Figure v.13. Note that this matches almost exactly with the solution found in the previous section (equation (v.18)). From the simulation, we find the mass to be 66.0 kg, as before.



UNITS ● SI Units (N, m) ○ English Units (lb, in) SCALE 100 pixels = 1 m

MEMBER PROPERTIES Area 4010.0 sq. mm Material Aluminum E = 70 GPa

Tensile forces appear in blue.
Compressive forces appear in red.

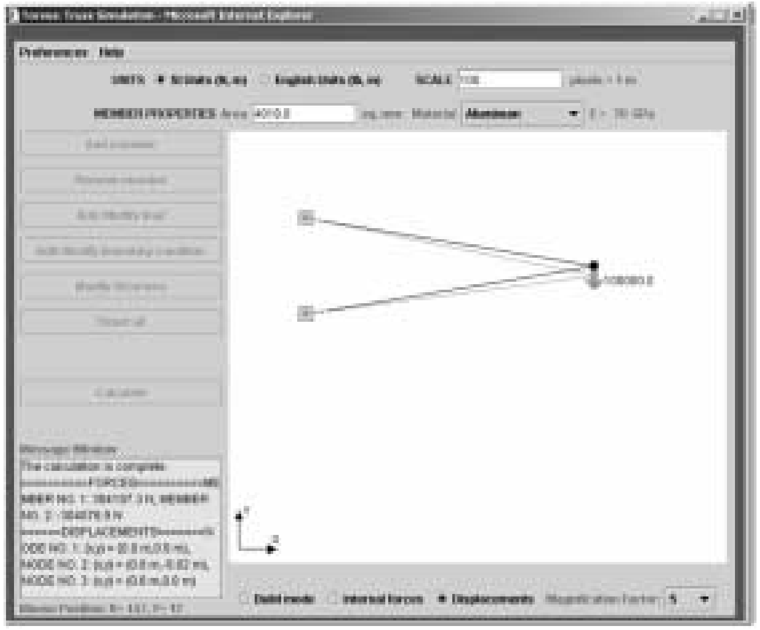
Message Window:

```
The calculation is complete.
**********FORCES**********ME
MBER NO. 1: 384197.3 N, MEMBER
NO. 2: -384078.9 N
******DISPLACEMENTS******N
ODE NO. 1: (u,v) = (0.0 m,0.0 m),
NODE NO. 2: (u,v) = (0.0 m,-0.02 m),
NODE NO. 3: (u,v) = (0.0 m,0.0 m)
```

Mouse Position: X= 17, Y= 357

○ Build mode ● Internal forces ○ Displacements Magnification Factor 5

Figure v.12 Two-member symmetric truss construction in Truss Structures simulation: internal forces shown.


Preferences Help
UNITS ● SI Units (N, m) ○ English Units (lb, in) SCALE 100 pixels = 1 m
MEMBER PROPERTIES Area 4010.0 sq. mm Material Aluminum E = 70 GPa
Calculate
100000.0
○ Build mode ○ Internal forces ● Displacements Magnification Factor 5

Message Window
The calculation is complete.
==========FORCES==========MEMBER NO. 1: 164107.3 N, MEMBER NO. 2: -164076.9 N
======DISPLACEMENTS======NODE NO. 1: (x,y) = (0.0 m,0.0 m),
NODE NO. 2: (x,y) = (0.0 m,-0.02 m),
NODE NO. 3: (x,y) = (0.0 m,0.0 m)

Figure v.13 Two-member symmetric truss construction in Truss Structures simulation: displacements shown.

Now we will begin the analysis of multi-member truss structures. A three-member design is proposed in Figure v.14.

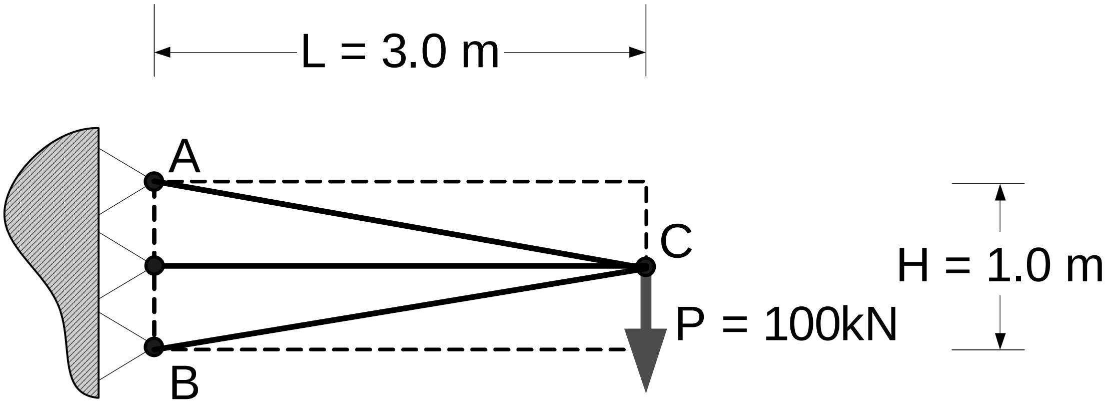
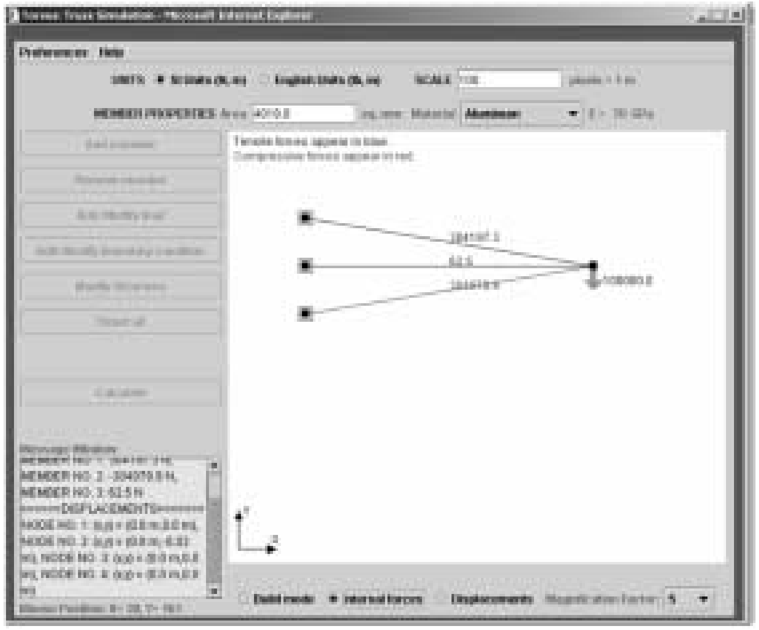


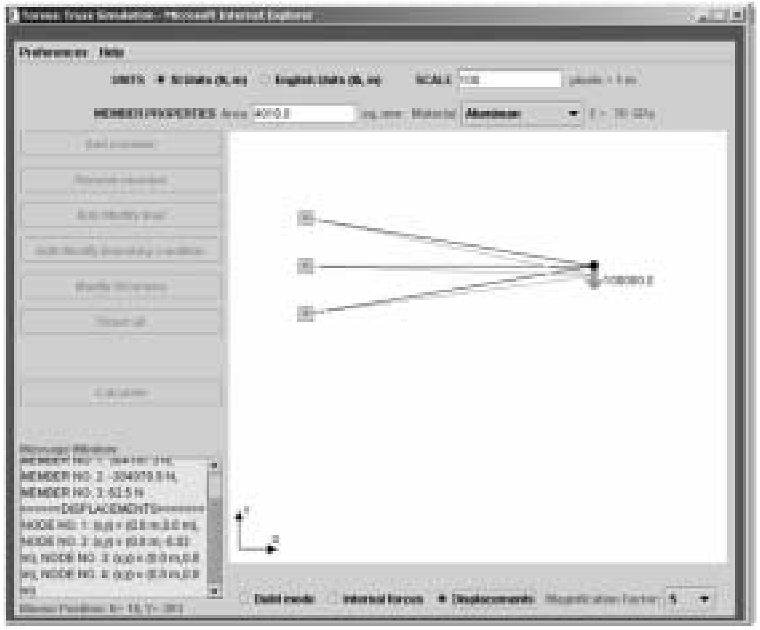
Figure v.14 A three-member truss proposed as a problem solution.

We do not expect this design to be any more effective than the two-member design, since the addition of the central member will be ineffective in bearing a load perpendicular to its main axis. We examine this design using the simulation and verify this result: the central member effectively acts as a zero-force member (the force in this member is much smaller than the forces in the other members). The three-member design necessitates a cross-sectional area of 40.18 cm^2, which results in an overall truss mass of 98.5 kg. The results are shown in Figure v.15 and Figure v.16.


Preferences Help
UNITS: ● SI Units (N, m) ○ English Units (lb, in) SCALE
MEMBER PROPERTIES Area 4018.0 sq. mm Material Aluminum E = 70 GPa
Tensile forces appear in blue.
Compressive forces appear in red.
62.5
Calculate
○ Build mode ● Internal forces ○ Displacements Magnification Factor 5

MEMBER NO. 2: -384079.8 N,
MEMBER NO. 3: 62.5 N
======DISPLACEMENTS======

Figure v.15 Three-member symmetric truss construction in Truss Structures simulation: internal forces shown.



Preferences Help

UNITS ● SI Units (N, m) ○ English Units (lb, in) SCALE 100 pixels = 1 m

MEMBER PROPERTIES Area 4010.0 sq. mm Material Aluminum E = 70 GPa

Message Window:

```
MEMBER NO. 1: [illegible]
MEMBER NO. 2: -304079.5 N,
MEMBER NO. 3: 62.5 N
======DISPLACEMENTS======
NODE NO. 1: (u,v) = (0.0 m,0.0 m),
NODE NO. 2: (u,v) = (0.0 m,-6.83
m), NODE NO. 3: (u,v) = (0.0 m,0.0
m), NODE NO. 4: (u,v) = (0.0 m,0.0
m)
```

○ Build mode ○ Internal forces ● Displacements Magnification Factor: 5

Figure v.16 Three-member symmetric truss construction in Truss Structures simulation: displacements shown.

A second multi-member truss design is shown below; this one is comprised of 5 members (Figure v.17).

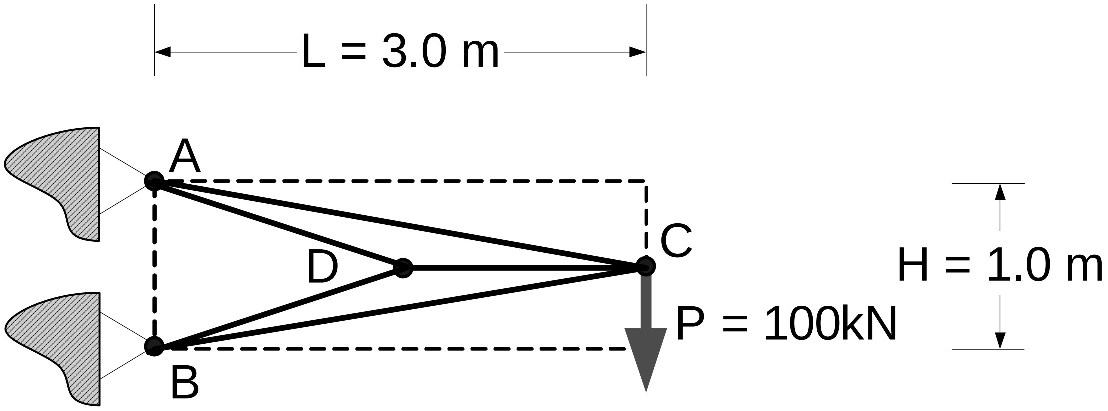
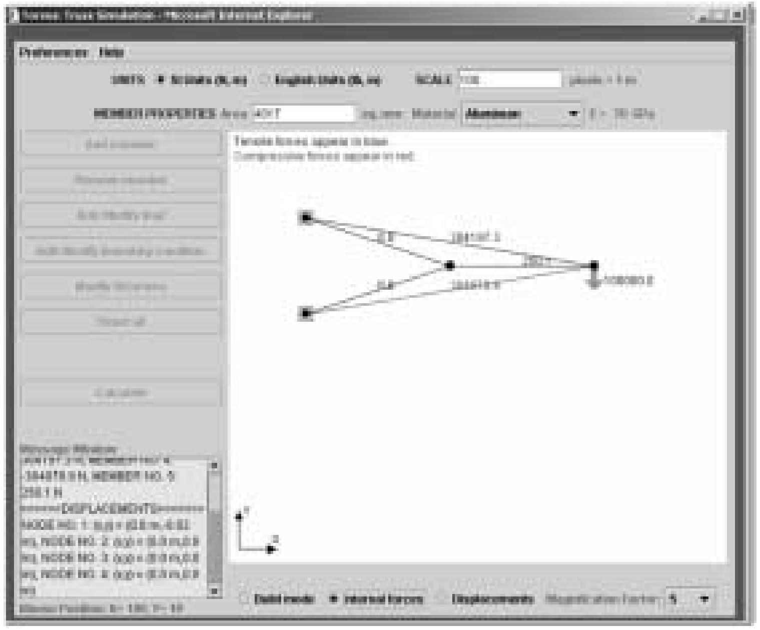


Figure v.17 A five-member truss proposed as a problem solution.

The truss is symmetric about the horizontal line running through point C. The central node, D, is taken to lie a distance 1.5 m from the wall.

We expect the upper- and lowermost members to carry most of the load; we do not expect the addition of the central members to help structurally. Using the Truss Structures simulation to verify that our intuitions are correct, we find that this does not provide a better solution than the two-member truss. The results for the five-member truss are shown below (Figure v.18 and Figure v.19).



```
-384078.9 N, MEMBER NO. 5
258.1 N
=======DISPLACEMENTS=======
```

Figure v.18 Five member truss construction in Truss Structures simulation: internal forces shown.

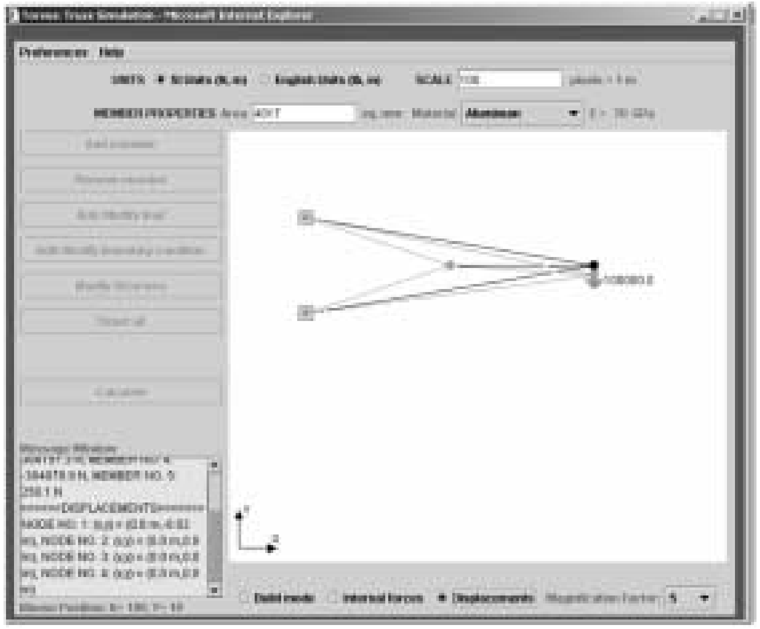


Figure v.19 Five member truss construction in Truss Structures simulation: displacements shown.

Each member requires a cross-sectional area of 40.17 cm^2 to meet the specified structural stiffness (k = 5000 kN/m), which results in an overall mass of 116.5 kg. This mass is greater than the mass of the symmetric two-member truss. Therefore, we will continue our search for a more optimal solution.

v.6 Design of Truss Geometry: More Complex Multi-Member Trusses

Before we continue, we will apply some reasoning to our design. We know that truss members best resist loads that have a major component acting along the member axis. This implies that a desirable orientation of the truss members connected to the end load is as close as possible to the vertical orientation. Using this assumption, and based on our constraints of symmetry and homogeneity of material, we will proceed.

Next, we will consider an seven-member truss, as shown below (Figure v.20).

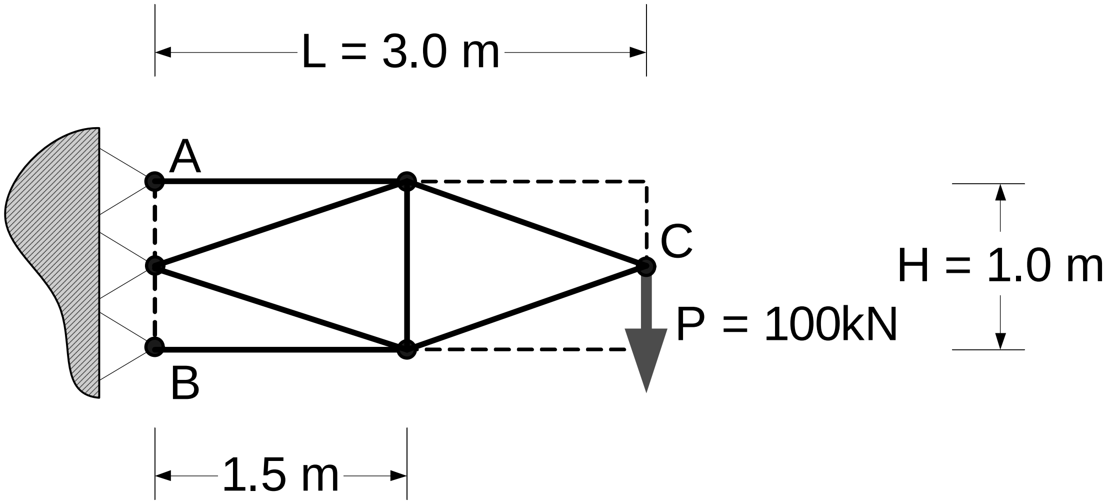
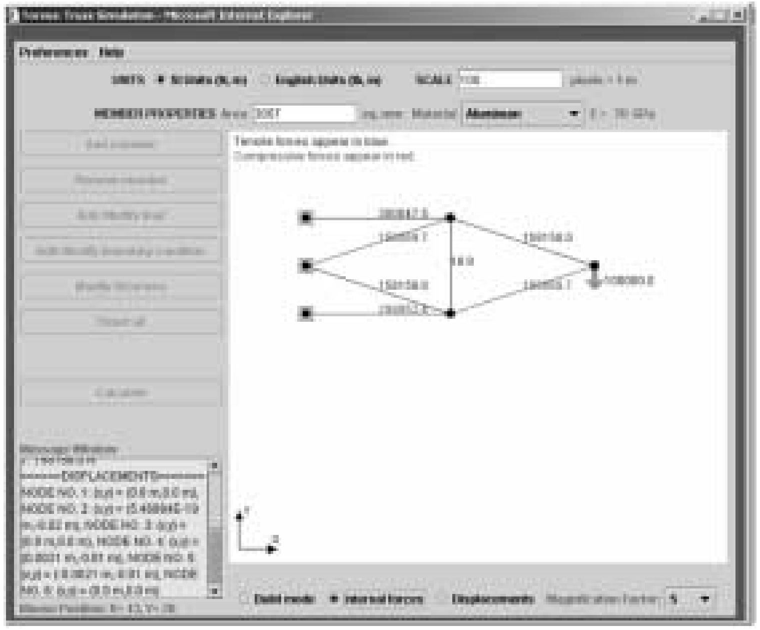


Figure v.20 Seven-member truss construction in Truss Structures simulation.

Using the simulation, we find that the minimum cross-section necessary to meet the stiffness bound is given by 30.57 cm^2 (Figure v.21, Figure v.22). The result is an overall mass of 85.2 kg. We note that the vertical member is not supporting any substantial load; in other words, it is essentially acting as a zero-force member. We will now analyze the same design, without the inclusion of the vertical member.



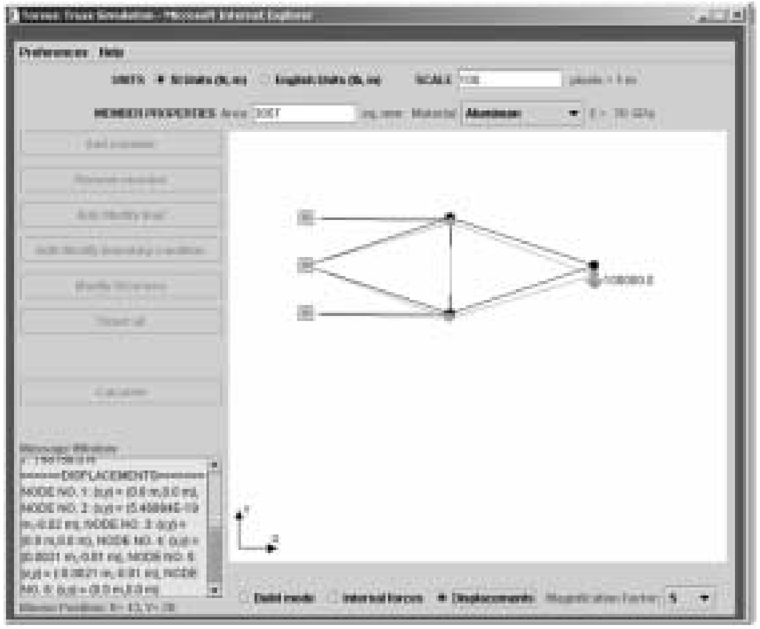
Stress: Truss Simulation - Microsoft Internet Explorer
Preferences Help
UNITS ● SI Units (N, m) ○ English Units (lb, in) SCALE 100 pixels = 1 m
MEMBER PROPERTIES Area 3087 sq. mm Material Aluminum E = 70 GPa
Add member
Remove member
Add/Modify load
Add/Modify boundary condition
Modify Structure
Clear all
Calculate
Tensile forces appear in blue.
Compressive forces appear in red.
280847.5
166656.7
150158.0
0.0
150158.0
166655.7
280852.5
-100000.0
Y
X
○ Build mode ● internal forces ○ Displacements Magnification Factor 5

Message Window

y: 150158.0 N
======DISPLACEMENTS======
NODE NO. 1: (u,v) = (0.0 m,0.0 m), NODE NO. 2: (u,v) = (5.40884E-19 m,-0.02 m), NODE NO. 3: (u,v) = (0.0 m,0.0 m), NODE NO. 4: (u,v) = (0.0021 m,-0.01 m), NODE NO. 5: (u,v) = (-0.0021 m,-0.01 m), NODE NO. 6: (u,v) = (0.0 m,0.0 m)

Mouse Position: X= 13, Y= 28

Figure v.21 Seven-member truss construction in Truss Structures simulation: internal forces shown.



Stress: Truss Simulation - Microsoft Internet Explorer
Preferences Help
UNITS ● SI Units (N, m) ○ English Units (lb, in) SCALE 100 pixels = 1 m
MEMBER PROPERTIES Area 3087 sq. mm Material Aluminum E = 70 GPa
Add member
Remove member
Add/Modify load
Add/Modify boundary condition
Modify Structure
Clear all
Calculate
-100000.0
Y
X
○ Build mode ○ internal forces ● Displacements Magnification Factor 5

Message Window

y: 150158.0 N
======DISPLACEMENTS======
NODE NO. 1: (u,v) = (0.0 m,0.0 m), NODE NO. 2: (u,v) = (5.40884E-19 m,-0.02 m), NODE NO. 3: (u,v) = (0.0 m,0.0 m), NODE NO. 4: (u,v) = (0.0021 m,-0.01 m), NODE NO. 5: (u,v) = (-0.0021 m,-0.01 m), NODE NO. 6: (u,v) = (0.0 m,0.0 m)

Mouse Position: X= 13, Y= 28

Figure v.22 Seven-member truss construction in Truss Structures simulation: displacements shown.

The modified design has six members, as shown in Figure v.23.

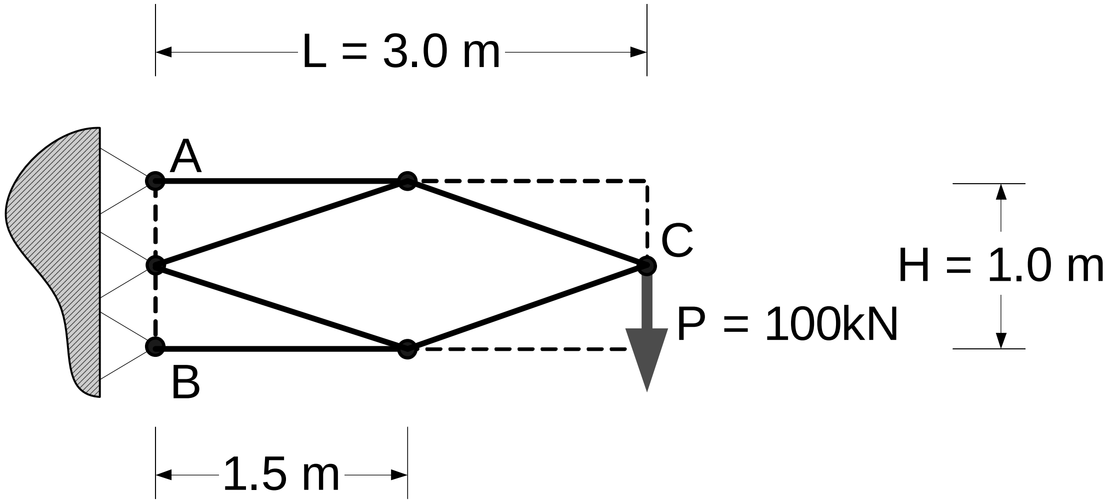
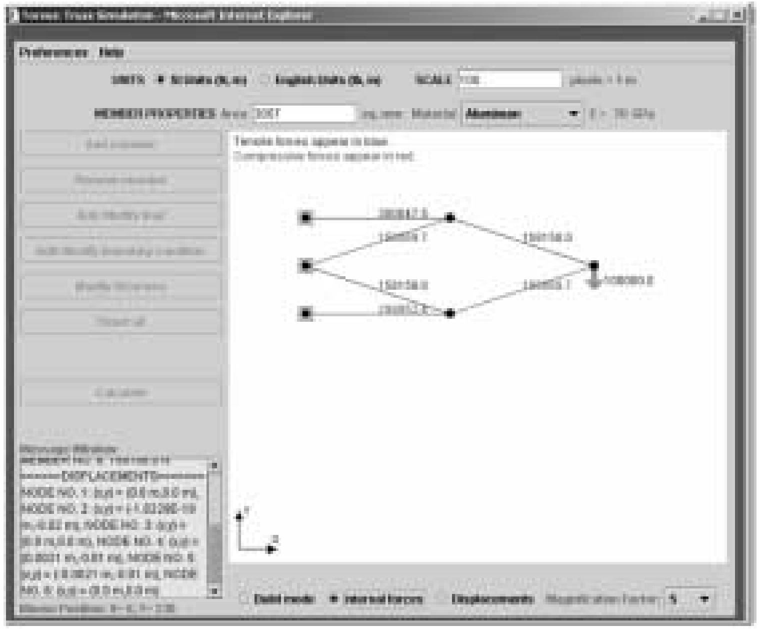


Figure v.23 Six-member truss design.

Using the simulation, we find that the minimum cross-section necessary to meet the stiffness bound is again - as in the case of the eight-member design - 30.57 cm^2 (Figure v.24 and Figure v.25). The result is an overall mass of 77.0 kg. Since this is greater than our current optimal value of 66.0 kg, this configuration will not be of interest.



UNITS ● SI Units (N, m) ○ English Units (lb, in) SCALE 100 pixels = 1 m

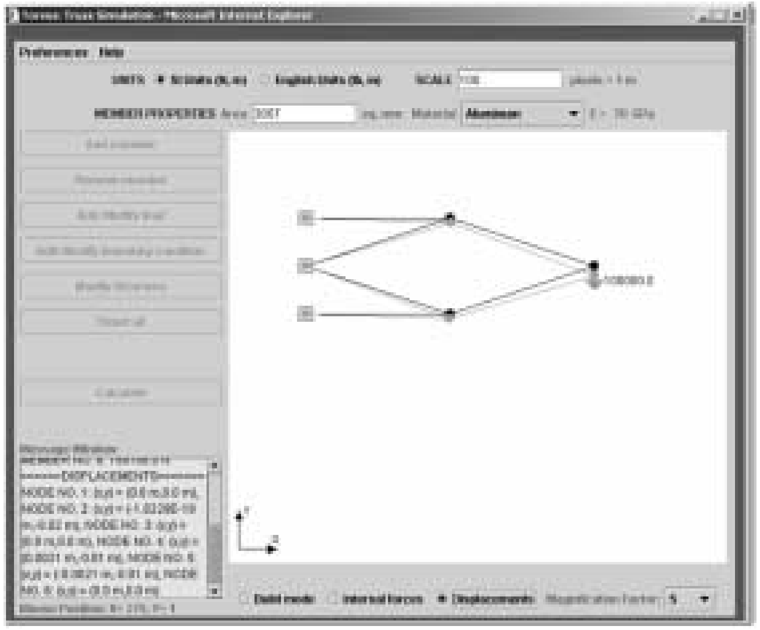
MEMBER PROPERTIES Area 3357 sq. mm Material Aluminum E = 70 GPa

Tensile forces appear in blue.
Compressive forces appear in red.

Message Window
MEMBER NO. 6: 150158.0
======DISPLACEMENTS======
NODE NO. 1: (x,y) = (0.0 m,0.0 m), NODE NO. 2: (x,y) = (-1.0228E-19 m,-0.02 m), NODE NO. 3: (x,y) = (0.0 m,0.0 m), NODE NO. 4: (x,y) = (0.0031 m,-0.01 m), NODE NO. 5: (x,y) = (-0.0021 m,-0.01 m), NODE NO. 6: (x,y) = (0.0 m,0.0 m)
Mouse Position: X= 0, Y= 230

○ Build mode ● Internal forces ○ Displacements Magnification Factor 5

Figure v.24 Six-member truss construction in Truss Structures simulation: internal forces shown.



UNITS ● SI Units (N, m) ○ English Units (lb, in) SCALE 100 pixels = 1 m

MEMBER PROPERTIES Area 3357 sq. mm Material Aluminum E = 70 GPa

Message Window
MEMBER NO. 6: 150158.0
======DISPLACEMENTS======
NODE NO. 1: (x,y) = (0.0 m,0.0 m), NODE NO. 2: (x,y) = (-1.0228E-19 m,-0.02 m), NODE NO. 3: (x,y) = (0.0 m,0.0 m), NODE NO. 4: (x,y) = (0.0031 m,-0.01 m), NODE NO. 5: (x,y) = (-0.0021 m,-0.01 m), NODE NO. 6: (x,y) = (0.0 m,0.0 m)
Mouse Position: X= 215, Y= 1

○ Build mode ○ Internal forces ● Displacements Magnification Factor 5

Figure v.25 Six-member truss construction in Truss Structures simulation: displacements shown.

Next, we will consider a similar configuration, which has eight members. This design differs from the previous design in that the only supports along the wall are located at points A and B, as shown below (Figure v.26).

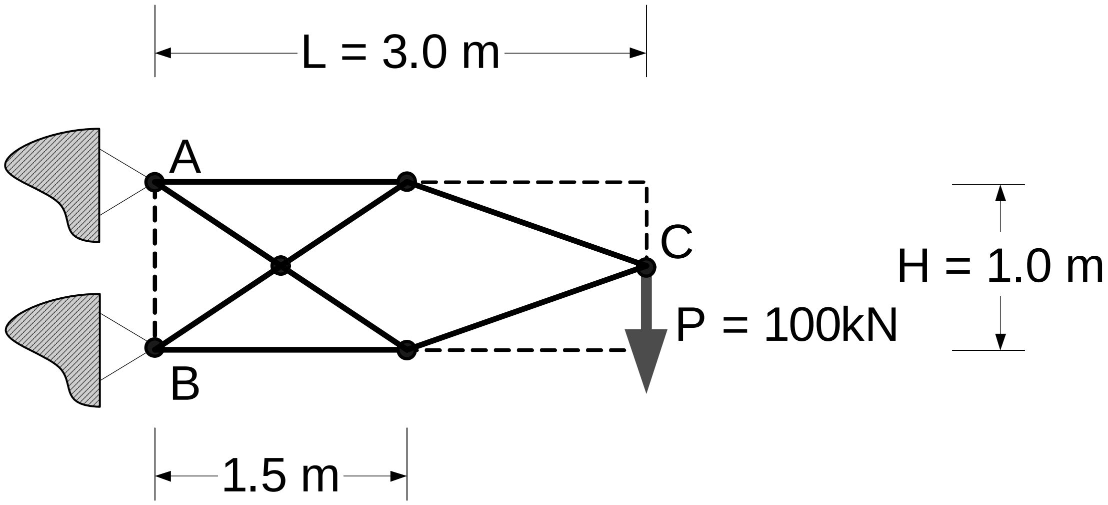
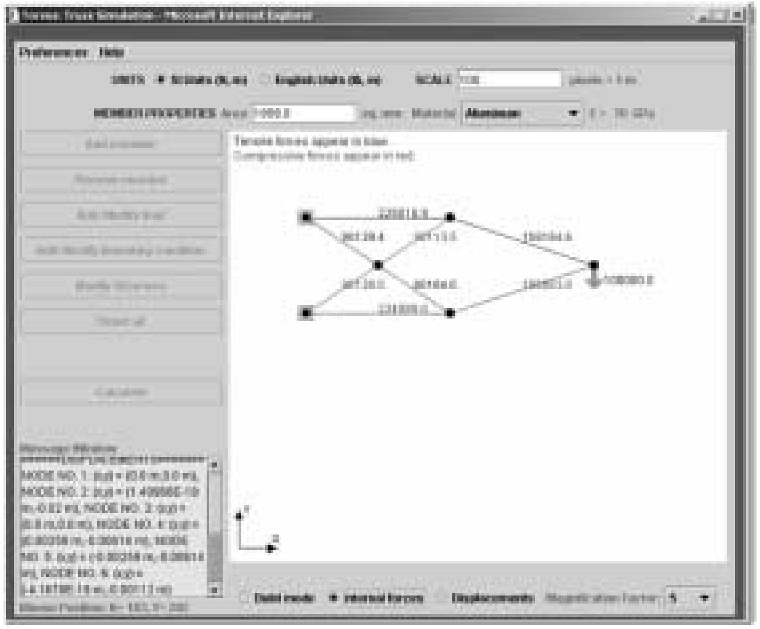


Figure v.26 Eight-member truss design.

We again examine this structure using the simulation. We find that the minimum area required to meet our stiffness bound is 18.58 cm^2. This results in a truss mass of 49.00 kg. The simulation results are shown below (Figure v.27, Figure v.28).



Tensile forces appear in blue.
Compressive forces appear in red.

Figure v.27 Eight-member truss construction in Truss Structures simulation: internal forces shown.

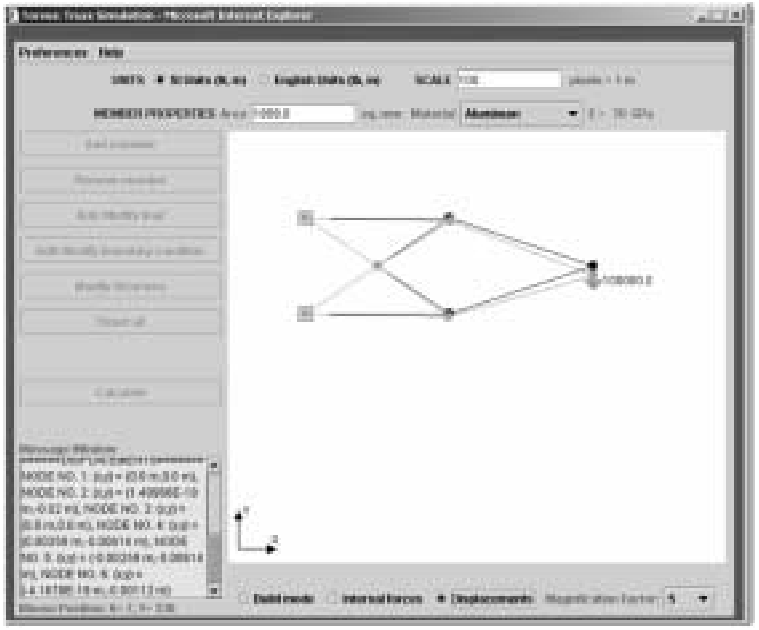


Figure v.28 Eight-member truss construction in Truss Structures simulation: displacements shown.

Next we will examine a variation on our previous design. Specifically, we will attempt to create a structure using the same geometric design that will allow us to position the members joined at the externally applied load P in a more vertical orientation. A fourteen-member truss given this design is shown below (Figure v.29).

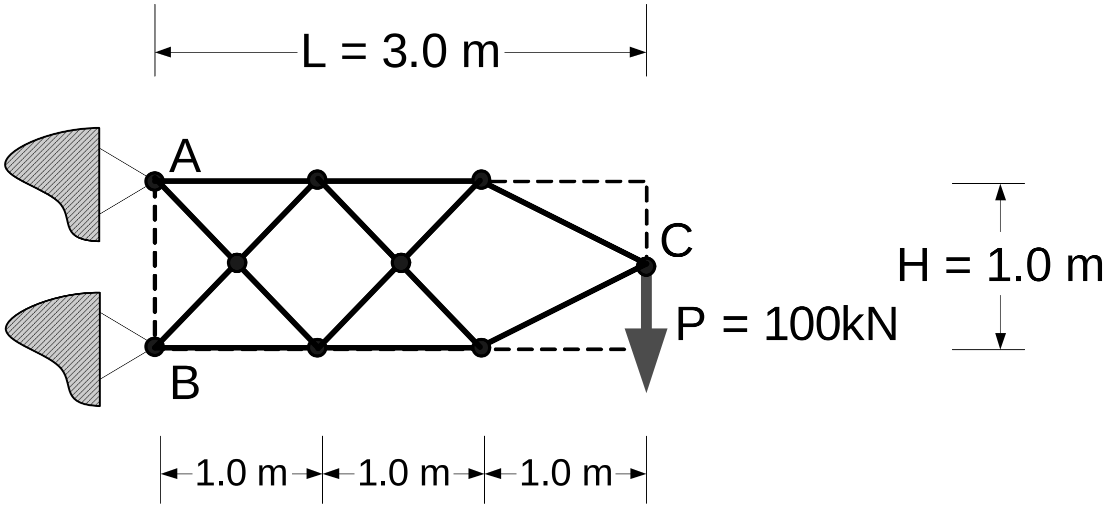
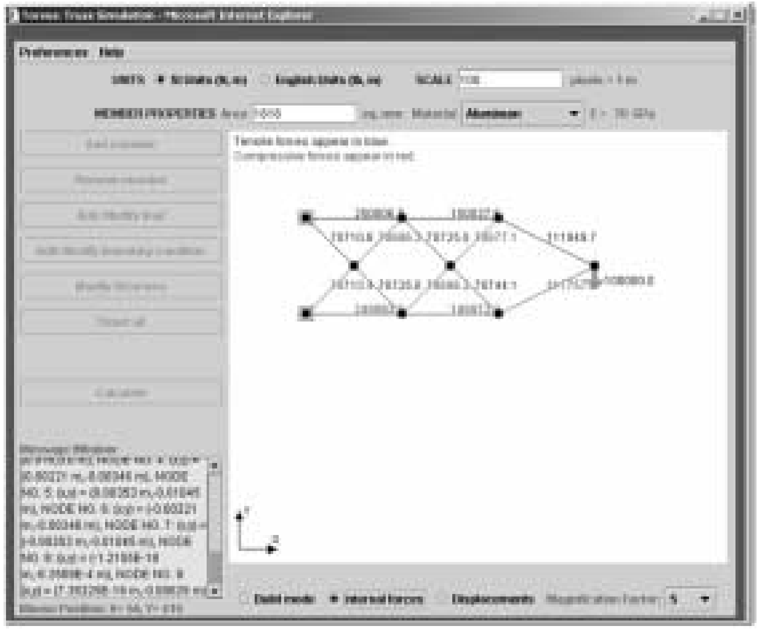


Figure v.29 A fourteen-member truss design.

This design, which was evaluated using the simulation, requires a cross-sectional area of 16.16 cm^2. The result of this is a truss that weighs 51.89 kg.



Preferences Help

UNITS ● SI Units (N, m) ○ English Units (lb, in) SCALE 100 pixels = 1 m

MEMBER PROPERTIES Area 1.0/10 sq. mm Material Aluminum E = 70 GPa

Tensile forces appear in blue.
Compressive forces appear in red.

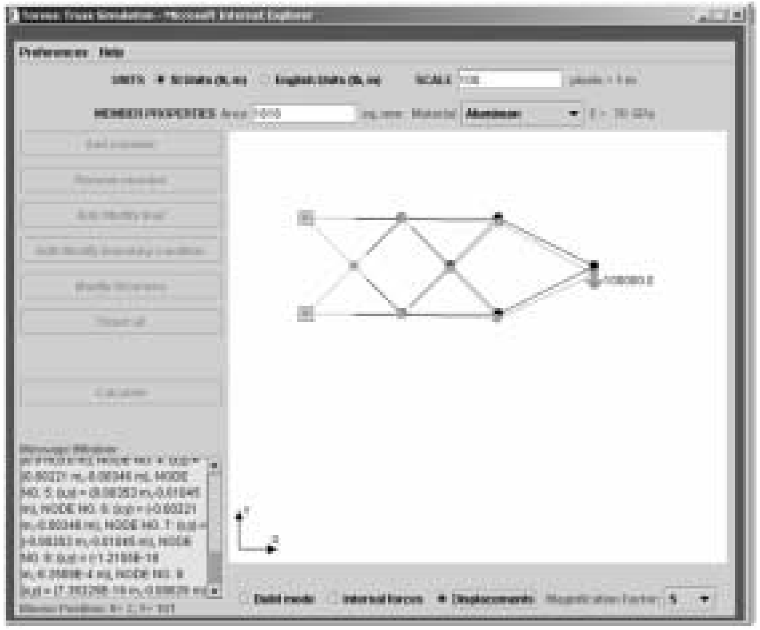
Message Window:

(0.00221 m,-0.00346 m), NODE NO. 5: (u,v) = (0.00283 m,-0.01045 m), NODE NO. 6: (u,v) = (-0.00221 m,-0.00346 m), NODE NO. 7: (u,v) = (-0.00283 m,-0.01045 m), NODE NO. 8: (u,v) = (-1.2195E-18 m,-6.2569E-4 m), NODE NO. 9: (u,v) = (7.3932E-19 m,-0.00829 m)

Mouse Position: X= 56, Y= 410

○ Build mode ● Internal forces ○ Displacements Magnification Factor: 5

Figure v.30 Fourteen-member truss construction in Truss Structures simulation: internal forces shown.



Preferences Help

UNITS ● SI Units (N, m) ○ English Units (lb, in) SCALE 100 pixels = 1 m

MEMBER PROPERTIES Area 1.0/10 sq. mm Material Aluminum E = 70 GPa

Message Window:

(0.00221 m,-0.00346 m), NODE NO. 5: (u,v) = (0.00283 m,-0.01045 m), NODE NO. 6: (u,v) = (-0.00221 m,-0.00346 m), NODE NO. 7: (u,v) = (-0.00283 m,-0.01045 m), NODE NO. 8: (u,v) = (-1.2195E-18 m,-6.2569E-4 m), NODE NO. 9: (u,v) = (7.3932E-19 m,-0.00829 m)

Mouse Position: X= 2, Y= 101

○ Build mode ○ Internal forces ● Displacements Magnification Factor: 5

Figure v.31 Fourteen-member truss construction in Truss Structures simulation: displacements shown.

Based on this information, we expect that our second 6 member truss (Figure v.26) is the best basic design. We will now tweak the parameters to see if we can further optimize this result.

As shown the figure shown below, we will attempt to vary the lengths of the members (by changing the value of d) and determine the structure for this application.

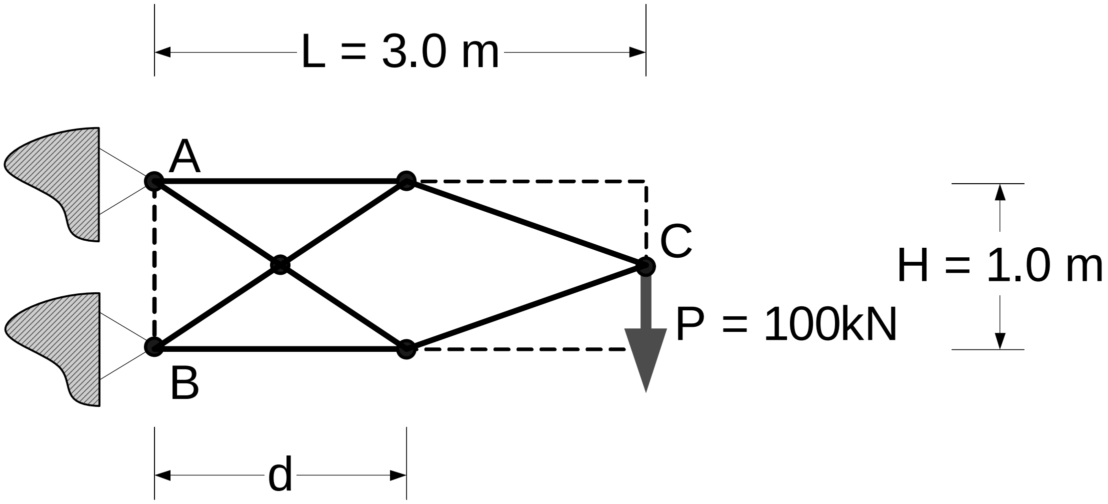


Figure v.32 Methodology for optimizing the best eight-member truss solution.

Again, using the simulation, we track the value of d against the minimum cross-sectional member area necessary to meet our compliance requirement. Note that the cross-sectional area and the geometry affect the overall truss mass, since - as d varies - the overall length of the members also changes. The results are given in table format below.

TABLE v.6 Value of d vs. minimum cross-sectional area and truss mass.

value of d (m)	cross-sectional area (mm^2)	truss mass (kg)
0.2	3565	78.24
0.4	3144	70.03
0.6	2785	63.43
0.8	2488	58.26
1.0	2240	54.14
1.2	2054	51.36
1.4	1912	49.53
1.5	1858	49.00
1.6	1815	48.74
1.8	1759	49.00
2.0	1741	50.34
2.2	1758	52.78
2.4	1806	56.38
2.6	1883	61.27
2.8	1985	67.66

It is clear, when the results are displayed graphically, that there is an optimal minimum mass obtained between d = 1.5 and d = 1.8.

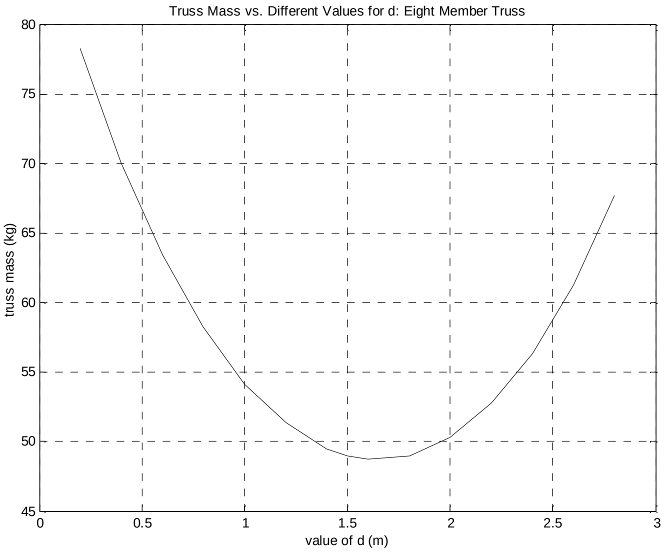


Figure v.33 Chart plotting truss mass vs. different values of d for the 8-member truss.

The exact value is not critical since the range of masses in this domain is less than 500 g. We will thus propose our best solution is of a geometry where d = 1.6 m, as shown below.

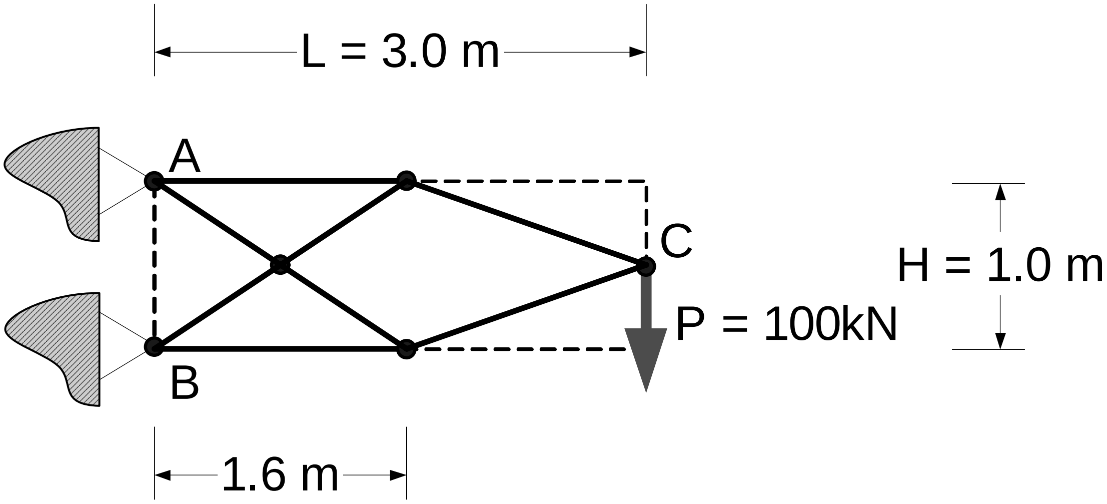


Figure v.34 Our optimal design for this version of the Michell truss problem.

v.7 Conclusions

This problem is an excellent example of the iterative process of design. We examined several designs, used our findings to limit our design parameters, examined several different designs that met our revised design parameters, selected the best candidate, and fine-tuned it. The proposed solution is not intended to be the best possible solution, but it is meant to be a very good one. The applications of such a problem should be evident.

Trusses are used in applications where it is necessary to create a structure that can span a large area, at a minimum to weight and, in most cases, cost. Scaffolding is an excellent example of an application where low cost, low weight, and quick assembly are essential components of a good design. A loading arm used by NASA is an example of a design where low weight is the most critical design parameter. Material cost is less of a barrier since the weight can adversely affect other costs, such as fuel.

A major key to design is understanding the application of the design, and abstracting design requirements from this application. A solid understanding the design process can prove to be an invaluable asset in mechanical engineering.

v.8 References

[1] Ashby, M.F. *Materials Selection in Mechanical Design*. Butterworth-Heineman, Oxford: 1992.

[2] Cox, H.L. *The Design of Structures of Least Weight*. Pergamon Press, Oxford: 1965.

[3] Dewhurst, Peter. *Analytical Solutions and Numerical Procedures for Minimum-Weight Michell Structures*. Journal of the Mechanics and Physics of Solids 49, 445-467 (2001).

[4] Hemp, W.S. *Optimum Structures*. Clarendon Press, Oxford: 1973.

[5] Kim, H., O.M. Querin, G.P. Steven, Y.M. Xie. *Development of an Intelligent Cavity Creation (ICC) Algorithm for Evolutionary Structural Optimisation*. SORGA Structual Optimisation Notes Number 3. January 1998.

[6] Michell, A.G.M. *Limits of Economy of Materials in Frame-Structures*. Philisophical Magazine 6 (8), 589-597 (1904).

[7] Mlejnek, H.P. *Optimal Material Distribution - History, Foundation, Benefits*. Computer Aided Topology Optimization Seminar: University of Bayreuth, Germany, April 2000.

[8] Niedenfuhr, F.W., and J.R.M. Radok. *The Collected Mathematical Works of J.H. and A.G.M. Michell*. P. Noordhoff Ltd. Groningen, Netherlands: 1964.

[9] Rozvany, G.I.N. *Grillages of Maximum Strength and Maximum Stiffness*. International Journal of Mechanical Sciences 14 (10), 651-666 (October 1972).

References

[1] *6.001 - Structure and Interpretation of Computer Programs*. MIT Department of Electrical Engineering and Computer Science, course 6.001. Sep. 2001 <http://sicp.ai.mit.edu/>.

[2] *6.034 - Artificial Intelligence Homepage*. MIT Department of Electrical Engineering and Computer Science, course 6.034. Sep. 2001 <http://www.ai.mit.edu/courses/6.034/>.

[3] *About Rowan*. Rowan University. Jan. 2002 <http://www.rowan.edu>.

[4] Ambrose, Susan A., and Cristina H. Amon. *Systematic Design of a First-Year Mechanical Engineering Course at Carnegie Mellon University*. Journal of Engineering Education, Vol. 86, No. 2 (1997): 173-181.

[5] *Athena History*. MIT Academic Computing. Dec. 2001 <http://web.mit.edu/acs/athena.html>.

[6] Beer, Ferdinand P., and E. Russell Johnston, Jr. *Mechanics for Engineers, Statics and Dynamics*. New York: McGraw Hill, 1962.

[7] Belcher, John W. Personal interview. 19 Dec. 2001.

[8] Bell, Beverly, and Brownen Cowie. *Formative Assessment and Science Education*. Dordrecht, The Netherlands: Kluwer Academic Publishers, 2001.

[9] Blackburn, Kelly. *Faculty Engagement and Resource Allocation*. UROP Final Report to Lori Breslow, MIT Teaching and Learning Laboratory, Summer 2001.

[10] Boyce, Mary C., Woodie Flowers, Seth Lloyd, Sanjay Sarma, Sunny Siu, David Wallace. *A New Era in Undergraduate Engineering Education.* Project proposal, Dec. 1999.

[11] *Bringing Cyberspace into the College Classroom*. Microsoft PressPass. Dec. 2001 <http://www.microsoft.com/PressPass/features/1999/10-05mit.asp>.

[12] *Center for Applied Research in Digital Government Information Systems.* Columbia University. Mar. 2001 <http://www.cs.columbia.edu/digigov/>.

[13] Chen, Vincent. *Teal Project a Success; May Expand Next Year*. The Tech (MIT student newspaper), 26 Oct. 2001, Vol. 121, No. 54.

[14] *The Cooper Union: Admissions Main Page*. The Cooper Union. Nov. 2001 <http://www.cooper.edu/administration/admissions/Welcome.html>.

[15] *Cornell News: Evaluating wireless networking on campus*. Cornell University press release. May 2001 <http://www.news.cornell.edu/releases/April01/wirelesseval.ws.html>.

[16] Crandall, Stephen H. Personal interview. 21 Sep. 2001.

[17] Crandall, Stephen H., N.C.Dahl, and T.J. Lardner. *An Introduction to the Mechanics of Solids*. 2nd ed. New York: McGraw Hill, 1978.

[18] Dewey, John. *Experience and Education*. New York: Macmillan, 1948.

[19] Eliëns, Anton. *Principles of Object-Oriented Software Development*. Harlow, England: Pearson Education, 2000.

[20] *Engineering Education: Designing an Adaptive System, a report from the Board of Engineering Education*. Washington, D.C.: National Academy Press, 1995.

[21] *Engineering Mechanics for Structures*. MIT Department of Civil and Environmental Engineering, course 1.050. Oct. 2001 <http://web.mit.edu/1.050/www/>.

[22] *Educational Technologies Group*. MIT Educational Technologies Group. Sep. 2001 <http://web.mit.edu/etg/www/>.

[23] Galanter, Eugene. *Automatic Teaching: the State of the Art*. New York: John Wiley & Sons, 1959.

[24] George, Judith, and John Cowan. *A Handbook of Techniques for Formative Evaluation*. London: Kogan-Page, 1999.

[25] Gibbons, J.F., W.R. Kinchloe, and K.S. Down. *Tutored Videotape Instruction: A New Use of Electronics Media in Education*. Science, Vol. 195, No. 4283 (1977): 1139-1146.

[26] Goldberg, Carey. *Auditing Classes at M.I.T., on the Web and Free*. The New York Times, 4 Apr. 2001.

[27] Griffioen, James, W. Brent Seales, and James. E. Lumpp, Jr. *Teaching in Real-time Wireless Classrooms*. Journal of Engineering Education, Vol. 88, No. 4 (1999): 397-402.

[28] Griffith, Peter. Personal interview. 10 Sep. 2001.

[29] Hailey, Christine E., and David E. Hailey. *Evaluation of Instructional Design of Computer-Based Teaching Modules for a Manufacturing Process Laboratory*. Journal of Engineering Education, Vol. 89., No. 3 (2000): 345-352.

[30] Henry, Robert M. SEVE, *Structural Engineering Visual Encyclopedia, version 1.0*. CD-ROM. John Wiley and Sons: 2000.

[31] Hsu, Jovan. *Student Engagement Literature Review*. UROP Final Report to Lori Breslow, MIT Teaching and Learning Laboratory, Summer 2001.

[32] *iCampus Home*. MIT Department of Mechanical Engineering, iCampus Home. Jan. 2002 <http://icampus1.mit.edu>.

[33] *iCampus Project: iLab*. MIT iCampus Alliance. Dec. 2001 <http://www.swiss.ai.mit.edu/projects/icampus/projects/ilab.html>.

[34] *iCampus Project: StudioMIT*. MIT iCampus Alliance. Dec. 2001 <http://www.swiss.ai.mit.edu/projects/icampus/projects/studiomit.html>.

[35] *iCampus Project: TEAL*. MIT iCampus Alliance. Dec. 2001 <http://www.swiss.ai.mit.edu/projects/icampus/projects/teal.html>.

[36] *IITS Homepage*. Nanyng Technological University School of Structural and Civil Engineering. Jan. 2002 <http://cse.ntu.edu.sg/iits/>.

[37] Jaques, David. *Learning in Groups, a Handbook for Improving Group Work*. 3rd ed. London: Kogan Page, 2000.

[38] *JAMA: Java Matrix Package*. National Institute of Standards and Technology (NIST). Jul. 2001 <http://math.nist.gov/javanumerics/jama/>.

[39] *Java Applets for Engineering Education*. Virginia Tech. Sep. 2001 <http://www.engapplets.vt.edu/>.

[40] *java.sun.com*. Sun Microsystems. Sep. 2001 <http://java.sun.com>.

[41] Kolb, David A. *Experiential Learning: Experience as the Source of Learning and Development*. Englewood Cliffs, NJ: Prentice-Hall, 1984.

[42] Kolb, David A., Boyatzis, Richard E., and Charalampos Mainemelis. *Experiential learning theory: Previous research and new directions*. *Perspectives on thinking, learning and cognitive styles*. Eds. R. Sternberg and L. Zhang. Mahwah, NJ: Lawrence Erlbaum Associates, 1999. Jan. 2002 <http://www.learningfromexperience.com/Research_Library/research_library.html>.

[43] Kolb, David A. *On Management and the Learning Process*. Working paper (Sloan School of Management), 652-73. Cambridge, MA: Massachusetts Institute of Technology, Alfred P. Sloan School of Management, 1973.

[44] Kolb, David A., and Marshall B. Goldman. *Toward a Typology of Learning Styles and Environments: an investigation of the impact of learning styles and discipline demands on the academic performance, social adaptations and career choices of MIT seniors*. Working paper (Sloan School of Management), 688-73. Cambridge, MA: Massachusetts Institute of Technology, Alfred P. Sloan School of Management, 1973.

[45] Laurillard, D. *Learning Through Collaborative Computer Simulations*. British Journal of Educational Technology, Vol. 23, No. 3 (1992): 164-171.

[46] Laurillard, Diana. *Rethinking University Teaching: A Framework for the Effective Use of Educational Technology*. London: Routledge, 1993.

[47] Long, Phillip D. *Student Laptop Project*. Presentation to MIT course 2.001. 10 Sep 01.

[48] Lowman, Joseph. *Mastering the Techniques of Teaching*. San Francisco: Jossey-Bass, Inc., 1984.

[49] MacGregor, Jean, James L. Cooper, Karl A. Smith, and Pamela Robinson. *Strategies foe Energizing Large Classes: From Small Groups to Learning Communities*. San Francisco: Jossey-Bass Publishers, 2000.

[50] *Massachusetts Institute of Technology Bulletin*. Vol. 137, No. 1. Cambridge, MA: Massachusetts Institute of Technology, September 2001.

[51] *Mazur Group*. Harvard University. Jun. 2001 <http://mazur-www.harvard.edu/education/educationmenu.html>.

[52] McClintock, Frank A. Personal interview. 19 Sep. 2001.

[53] McClintock, Frank A., and Ali S. Argon. *Mechanical Behavior of Materials*. Reading, MA: Addison Wesley, 1966.

[54] *MDSolids: Educational Software for Mechanics of Materials*. University of Alberta MDSolids homepage. Oct. 2001 <http://www.mece.ualberta.ca/Tutorials/mdsolids/mdsolids.htm>.

[55] *MDSolids General Information*. University of South Florida, Mechanics of Materials Laboratory. Oct. 2001 <http://www.eng.usf.edu/~garbin/mdsolids.htm>.

[56] *MDSolids.com*. MDSolids homepage. Sep. 2001 <http://www.mdsolids.com>.

[57] Menges, Robert. *Shortcomings of Research on Evaluating and Improving Teaching in Higher Education*. *Evaluating Teaching in Higher Education: A Vision for the Future*. Ed. Katherine E. Ryan. San Francisco: Jossey-Bass Publishers, 2000.

[58] *Merit Scholarships Lure Best Students*. Duke University. Chronicle (Duke student newspaper), 10 Sep. 1999. Dec. 2001 <http://www.duke.edu/web/abduke/archive/Chronicle09-10-99MeritScholarships.htm>.

[59] Michell, A.G.M. *Limits of Economy of Materials in Frame-Structures*. Philisophical Magazine, Vol. 6, No. 8 (1904): 589-597.

[60] Mihalik, Aaron D. *Project Athena*. MIT Tech archives. The Tech, Vol. 119, No. 9, 13 Apr. 1999. Dec. 2001 <http://the-tech.mit.edu/V119/N19/history_of_athe.19f.html>.

[61] Millis, Barbara J., and Philip G. Cottell, Jr. *Cooperative Learning for Higher Education Faculty*. Phoenix: Oryx Press, 1998.

[62] *MIT and Microsoft announce long-term collaboration for innovation in higher educatio*n. MIT, press release. Aug. 2001 <http://web.mit.edu/newsoffice/nr/1999/microsoft.html>.

[63] *MIT I-Campus I-Lab*. MIT, iLab homepage. Dec. 2001 <http://i-lab.mit.edu>.

[64] *MIT: OpenCourseWare*. MIT OpenCourseWare homeapge. Jun. 2001 <http://web.mit.edu/ocw>.

[65] *MIT Student Financial Aid*. MIT. Sep. 2001 <http://web.mit.edu/afs/athena.mit.edu/org/f/finaid/>.

[66] *MIT: Student Laptop Project*. MIT Information Systems. Nov. 2001 <http://web.mit.edu/slp/>.

[67] *moment - an interactive learning environment*. MIT Department of Civil and Environmental Engineering. Jun. 2001 <http://moment.mit.edu>.

[68] *NEEDS-Engineering*. National Engineering Education Delivery System (NEEDS). Sep. 2001 <http://www.needs.org/premiere>.

[69] *New Engineering School to Reflect NSF Philosophy*. IEEE Spectrum. Sep. 2001 <http://www.spectrum.ieee.org/INST/aug97/school.html>.

[70] *Nish Bio*. Nishikant Sonwalker personal homepage. Jul. 2001 <http://web.mit.edu/nish/www/home.html>.

[71] Otto, Kevin N., and Krisitin L. Wood. *Product Design: Techniques in Reverse Engineering and New Product Development*. Upper Saddle River, NJ: Prentice Hall, 2000.

[72] *Personal Response System*. MIT Department of Aeronautics and Astronautics, Unified Engineering. Aug. 2001 <http://web.mit.edu/16.unified/www/FALL/prs.html>.

[73] Philpot, Timothy A. *Bridging the Gap Between Mechanics of Materials Lectures and Homework with MDSolids*. The Technology Interface, Vol. 2, No. 3 (1998). Jun. 2001 <http://et.nmsu.edu/~etti/spring98/spring98.html>.

[74] Philpot, Timothy A. *MDSolids*. Vers. 1.5. Jun. 2001 <http://www.mdsolids.com/>/

[75] *Physics 8.02 - Electricity & Magnetism*. MIT Department of Physics. Dec. 2001 <http://web.mit.edu/8.02t/www/>.

[76] *PIVoT: The Physics Interactive Video Tutor*. MIT Center for Advanced Educational Services. Jan. 2002 <http://caes.mit.edu/research/pivot/>.

[77] Ploetzner, Rolf, Pierre Dillenbourg, Michael Preier, and David Traum. *Learning by Explaining to Oneself and to Others*. *Collaborative Learning.* Ed. Pierre Dillenbourg. Amsterdam: Pergamon, 1999.

[78] *PowerPoint Web Converter*. Jaspal Sandhu personal page. Jan. 2002 <http://www.mit.edu/~jsandhu/ppt.html>.

[79] Prensky, Marc. *Digital Game-based Learning*. New York: McGrawHill, 2001.

[80] Prichard, Keith W., and R. McLaren Sawyer. *Handbook of College Teaching: Theory and Applications.* Westport, CT: Greenwood Press, 1994.

[81] Recker, Mimi. *Appropriate Use of Educational Technologies: A Layered Approach*. Educational Technology Review, No. 7 (1997). Nov. 2001 <http://it.usu.edu/~mimi/papers/edtech.html>.

[82] Ritchie, Karen. *Marketing to Generation X*. New York: Lexington Books, 1995.

[83] *Robert M. Henry Homepage*. Structural Engineering Visual Encyclopedia (SEVE) homepage. Oct. 2001 <http://pubpages.unh.edu/~rmh1/seve/index-seve.html>.

[84] Rojiani, Kamal B., Yong Y. Kim, and Rakesh K. Kapania. *Web-Based Java Applets for Teaching Engineering Mechanics*. Proceedings from ASEE 2000 Annual Conference, Session 2620, 18-21 Jun. 2000, St. Louis, Mo.

[85] Sacks, Peter. *Generation X Goes to College*. Chicago: Open Court, 1996.

[86] Sandhu, Jaspal, Eberhard Bamberg, and Mary C. Boyce, *Development of Interactive Web-Based Modules in Redesigning an Introductory Solid Mechanics Course*, Proceedings from 2001 International Conference on Technology in Teaching and Learning in Higher Education, 27-29 Jun. 2001, Samos Island, Greece.

[87] Sandhu, Jaspal, Eberhard Bamberg, and Mary C. Boyce. *Integration of Information Technology into an Introductory Solid Mechanics Course*, 2001 ASEE/SEFI/TUB International Colloquium, 15-18 Sep. 2001, Berlin, Germany (conference postponed to 2002, proceedings not published due to tragic events of September 2001).

[88] Sandhu, Jaspal, and Sanjay E. Sarma. *Discovery-based and Cooperative Learning in Solid Mechanics*. Presentation to the MIT-Microsoft Project iCampus Joint Steering Committee. 3 Dec. 2001.

[89] Sarathy, Rahul. *UROP Summary for Conceptual Learning and Higher Education*. UROP Final Report to Lori Breslow, MIT Teaching and Learning Laboratory, Summer 2001.

[90] Sarma, Sanjay E., Eberhard Bamberg, and Jaspal Sandhu. *Teaching Mechanical Engineering in 2.008, the Case of 2.001*, presentation to the MIT Council on Educational Technology. 12 Dec. 2001.

[91] *The Scheme Programming Language*. MIT Scheme Programming Language homepage. Oct. 2001 <http://www.swiss.ai.mit.edu/projects/scheme, October 2001>.

[92] Shapiro, Ascher H. *Fluid Dynamics: Video Course Manual*. Cambridge, MA: MIT Center for Advanced Engineering Study, 1984.

[93] Shepherdson, Emma F. *Teaching Concepts Utilizing Active Learning Computer Environments*. Doctoral dissertation. MIT, Feb. 2001.

[94] Siegel, Mark E. *Instructional Infrastructure Planning: Innovation Theory, Systems Theory, and Computer Modeling*. University of North Carolina. Dec. 2001 <http://horizon.unc.edu/projects/monograph/CD/Change_Innovation/Siegel.asp>.

[95] Soh, C.K., and Ashok Gupta. *Intelligent Interactive Tutoring System for Engineering Mechanics*. Journal of Professional Issues in Engineering Education and Practice, Vol. 126, No. 4 (2000): 166-173.

[96] Steif, Paul S. *StressAlyzer, version 2.0*. Downloadable EXE, 2001.

[97] *StressAlyzer Homepage*. StressAlyzer homepage. Oct. 2001 <http://www.me.cmu.edu/stressalyzer/>.

[98] Thalang, Ping na. *Digitizing Management: Real World Example of Taking by Giving; MIT Earns Praise for Free Content Online*. Bangkok Post, 2 May 2001.

[99] *Tomas Lozano-Perez*. Tomas Lozano-Perez personal homepage. Dec. 2001 <http://www.ai.mit.edu/lab/faculty-survey/viewsurvey.cgi?Filename=tlp>.

[100] *Tuition-free MIT*. Philip Greenspun. Jul. 2001 <http://philip.greenspun.com/school/tuition-free-mit.html>.

[101] *UA Laptop Initiative*. University of Akron. Dec. 2001 <http://www.uakron.edu/laptop/index.html>.

[102] *Usability Report: MIT MechE I-Campus Project – Beam Bending Module*. MIT Information Systems, Jan. 2001.

[103] Vest, Charles M. *Disturbing the Educational Universe: Universities in the Digital Age - Dinosaurs or Prometheans?* Report of the President, Academic Year 2001-2002. Cambridge, MA: Massachusetts Institute of Technology, 2001.

[104] Wallace, D.R., and P. Mutooni. *A Comparative Evaluation of World Wide Web-*

Based and Classroom Teaching. Journal of Engineering Education, Vol. 86., No. 3 (1997): 211-219.

[105] *Welcome to Olin College*. Olin College. Sep. 2001 <http://www.olin.edu>.

[106] *Welcome to WebAssign*. WebAssign homepage. Dec. 2001 <http://www.webassign.com>.

[107] Whitworth, Damian. *Scientists Open Books to World on Internet*. The Times (London), 6 Apr. 2001.

[108] *Wireless*. MIT Information Systems Wireless LAN Deployment. Oct. 2001 <http://web.mit.edu/is/np/projects/wireless/>.

[109] Yue, Tao. *Learning to Embrace Athena*. MIT Tech archives. The Tech, Vol. 121, No. 37, 28 Aug. 2001. Dec. 2001 <http://www-tech.mit.edu/V121/N37/col37taoyu.37c.html>.