

PROCEEDINGS
1991 ANNUAL
CONFERENCE

JUNE 16-19, 1991
UNIVERSITY OF
NEW ORLEANS



VOLUME TWO

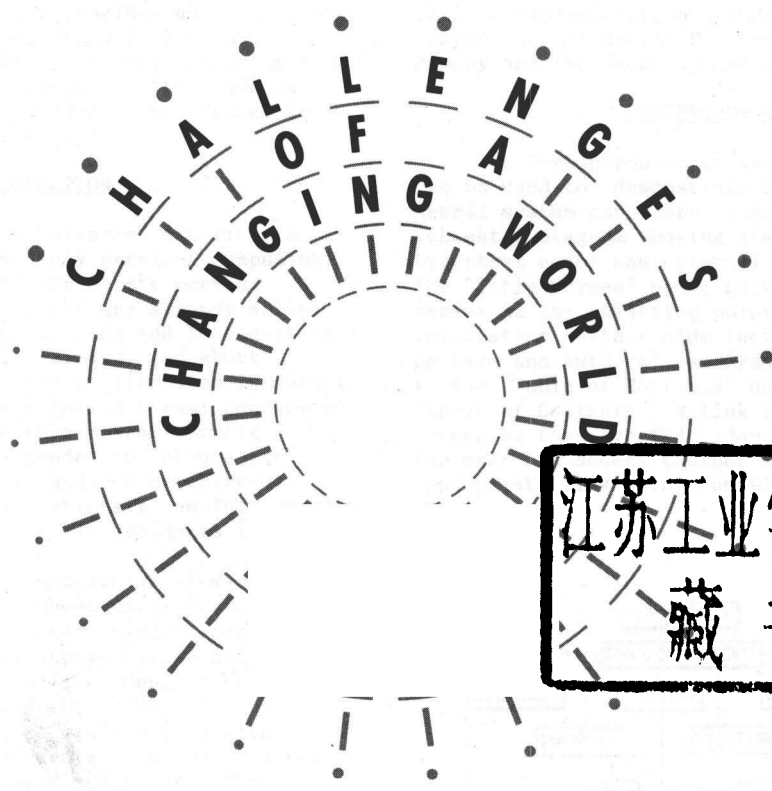
AMERICAN SOCIETY
FOR ENGINEERING
EDUCATION



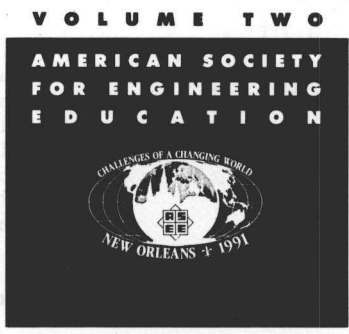
Ans 75 4.33
T13
W5.91(2)

PROCEEDINGS
1991 ANNUAL
CONFERENCE

JUNE 16-19, 1991
UNIVERSITY OF
NEW ORLEANS



江苏工业学院图书馆
藏书章



Educational Artificial Neural Networks: An Integration of Hypertext and Simulation Software

Siripong Malasri
Christian Brothers College

ABSTRACT

Several artificial neural network architectures have been developed for educational purposes, integrating hypertext with simulation programs. Such an approach provides an interactive textbook with hands-on experience. This paper discusses the implementation techniques using the Perceptron Module as an example. Other modules implemented include the Bidirectional Associate Memory and the Backpropagation.

INTRODUCTION

"Artificial Neural Networks" describes a dynamic new form of massively parallel computing based on an analogy with vertebrate nervous systems [1]. Neural networks are already making vital contributions to computing and to cognitive science [2]. Workshops, seminars, and short courses on the subject have proliferated in recent years. Many schools are offering formal courses on the subject. We believe that neural network simulation packages are needed to enhance the learning process through hands-on experience. These packages, to be educationally useful, should be affordable and designed with students in mind.

Hypertext provides a nonlinear viewing of information [3]. It was conceptually introduced in 1945 [4], and has been widely used in recent years to develop on-line documentation and tutoring materials. There is evidence that hypertext can provide a deeper level of understanding of materials studied [5]. With a hypertext system, the students have more control of the materials to be studied. Several PC-based hypertext authoring tools have emerged in the past few years. They significantly facilitate the development of hypertext documents.

Integrating hypertext with simulation programs provides a new instructional tool with which students will have access to an interactive textbook with hands-on experience. It offers opportunities that do not exist in traditional textbooks and software packages. A set of such integrated software has been developed for several neural network models. We have developed these packages primarily for a workshop under the 1990-1991 Chautauqua Program. They will also be used later with graduate students in the Neurocomputing

course at Memphis State University and the Knowledge Engineering course at Christian Brothers University. The first neural network model implemented using this integrated approach is the Perceptron. It is used in this paper to illustrate various implementation techniques. Other models completed include the Bidirectional Associative Memory and the Backpropagation.

PERCEPTRON MODEL

The Perceptron model is easy to implement and can be used to demonstrate how an artificial neural system can learn. Figure 1 shows a schematic diagram showing the relationships among hypertext nodes and external simulation programs. The "Title Screen" node, known as the root node, serves as the starting point of the package. The information on this node includes the title of the package and authors' names/addresses. It is linked to the "Table of Contents" node via a link named "Table of Contents". A link is identified on the screen by the inverted video of a word or phrase. The user can access another node by selecting an appropriate link using function keys or a mouse.

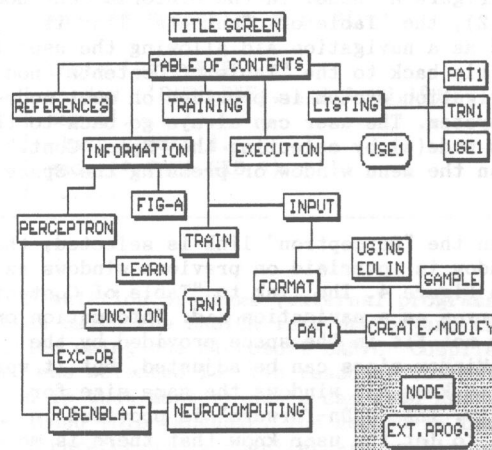


Figure 1. Schematic Diagram of ANNI.

The "Table of Contents" node, as shown in Figure 2, serves as a navigation system allowing the user to access each part directly. Five links appear on this node. The "Reference" link leads to a list of references. The "Information" link provides information about the system, as well as about the Perceptron model. The "Training Mode" link allows the user to train a network. The "Execution Mode" runs a trained network. The "Listing" link displays the listing of simulation programs.

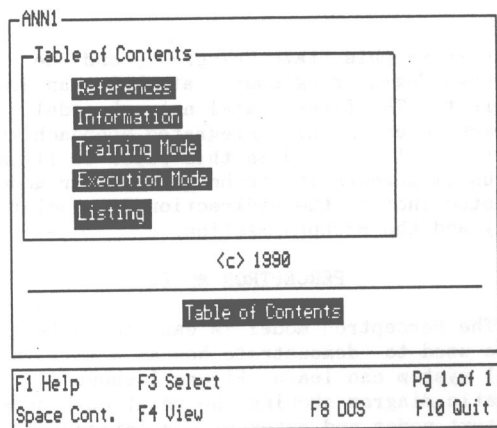


Figure 2. "Table of Contents" Node.

When the "Information" link is activated, the "Information" node is displayed with more links -- "Perceptron", "Figure A", and "Table of Contents", as shown in Figure 3. A node can contain textual information, such as the "Information" node, or graphics information, such as the "Figure A" node. In the "Information" node (Figure 2), the "Table of Contents" link is provided as a navigation aid allowing the user to go directly back to the "Table of Contents" node. This navigation system is provided on most nodes in the system. The user can always go back to the previous node(s) by selecting the "Space Cont." option on the menu window or pressing the Space Bar.

When the "Perceptron" link is selected, the next window is overlaid on previous windows as shown in Figure 4. The link to "Table of Contents" again serves as a navigation aid. Information on a node may not fit in the space provided by the window. Window sizes can be adjusted, but it was decided to keep all windows the same size for uniformity. The "PgDn-forward" is provided in Figure 4 to let the user know that there is more information on this node and that the function key "PgDn" can be used to move forward to the next screen of the current node. "PgUp" is also used on other screens when appropriate to let the user move backward to the previous screen.

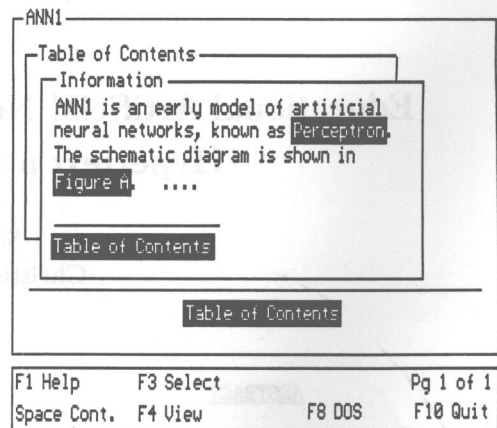


Figure 3. "Information" Node.

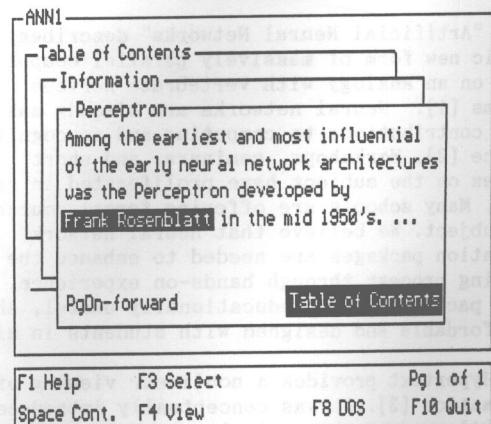


Figure 4. "Perceptron" Node.

Some nodes are directly linked to external programs. There are three external programs in ANN1, namely PAT1, TRN1, and USE1. PAT1 allows the user to create or modify the training pattern data file. TRN1 trains a network using the patterns previously created by PAT1. USE1 executes a trained network. The user can observe how the network adapts itself to a new training pattern. Weights are adjusted and displayed on the screen. Two modes are available, the Slow Mode and the Fast Mode. The former allows the user to view each training cycle as long as he or she wishes, while the latter runs through cycles continuously. Figure 5 shows a sample of the screen layout during a training session.

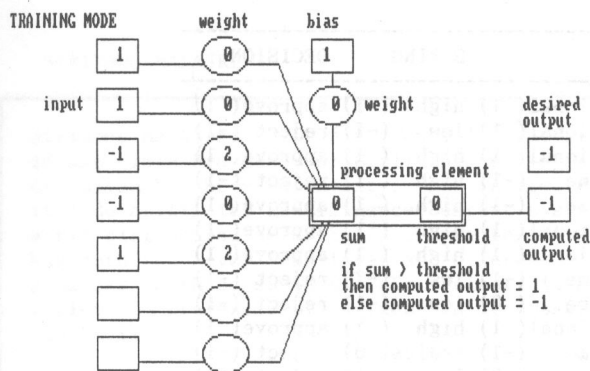


Figure 5. Training of a Network.

IMPLEMENTATION

KnowledgePro [6] was selected for this implementation for several reasons. The most important factor is the no-fee delivery of applications. This allows us to deliver the application to an unlimited number of students and other interested faculty at a very low cost (the cost of diskette and mailing). The MS-DOS hardware is another major factor, since there are a large number of installations in many engineering schools. KnowledgePro also supports external programs, which is crucial for this application.

Some PC-based packages have advantages over KnowledgePro in other areas. For example, GUIDE [7] is a very flexible tool in creating hypertext documents. The development time of a large complex hypertext system would be much less using the GUIDE system. The delivery cost of GUIDE applications is, however, much more than KnowledgePro. Hyperties [8] provides an excellent built-in navigation system. It is also very easy to create nodes and links with Hyperties. The version that we use does not, however, provide the external program interface capability, which is critical to this application. There are also problems with its graphics interface. From the three packages available to us, KnowledgePro turns out to be our best choice in terms of both cost and features.

Table 1 shows a partial listing of the source code for the hypertext system. Line numbers are provided for reference purposes in this paper and are not part of the actual source code. Messages can be displayed using the `say('messages')` command, as shown on line 1. A hypertext link can be created by enclosing the desired word(s) in `#m...#m`, as shown in line 3. Lines 1 through 3 at the beginning of the source code construct the root node. Other nodes are created using the `topic` command as shown in line 4. A window can be opened using the `window` command (line 5) and the `close window` command (line 10). Line 25 contains

the `picture` command. It is used to display a PCX graphics file (in this case figal.pcx). A PCX graphics file can be created by a graphics editor, such as Paintbrush [9]. An external program can be executed using the `run` command, as shown on line 29 where the `restore` command displays the hypertext screen after finishing the external program execution.

Table 1. Partial Listing of the Hypertext Source Code.

Line	Source Code Listing
1	<code>say(' title screen')</code>
2	<code>.....</code>
3	<code>#mTable of Contents#m ').</code>
4	<code>topic 'Table of Contents'.</code>
5	<code>window ('Table of Contents',,,,,,).</code>
6	<code>say (' #mReferences#m</code>
7	<code>#mInformation#m</code>
8	<code>.....</code>
9	<code>#mListing#m').</code>
10	<code>close_window().</code>
11	<code>end. (* Table of Contents *)</code>
12	<code>.....</code>
13	<code>topic 'Information'.</code>
14	<code>window ('Information',,,,,,).</code>
15	<code>say (' ANN1 is an early model of artificial</code>
16	<code>neural networks, known as #mPerceptron#m.</code>
17	<code>The schematic diagram is shown in</code>
18	<code>#mFigure A#m.</code>
19	<code>.....</code>
20	<code>#mTable of Contents#m').</code>
21	<code>close_window().</code>
22	<code>end. (* Information *)</code>
23	<code>.....</code>
24	<code>topic 'Figure A'.</code>
25	<code>picture (figal).</code>
26	<code>end. (* Figure A *)</code>
27	<code>.....</code>
28	<code>topic 'Execution Mode'.</code>
29	<code>run('usel.exe',restore).</code>
30	<code>end. (* Execution Mode *)</code>
31	<code>.....</code>
32	<code>topic 'PAT1.EXE'.</code>
33	<code>window ('PAT1.EXE',,,,1,1,80,19).</code>
34	<code>say ('.....').</code>
35	<code>close_window().</code>
36	<code>end. (* PAT1.EXE *)</code>
37	<code>.....</code>

Simulation programs (external programs) were written using the BASICA interpreter and then compiled using the Microsoft BASIC Compiler [10]. BASIC was selected for two reasons. The main reason is that most students can read and understand this language. The second is that both an interpreter and a compiler are available, which facilitates the development process. The program can be tested and modified using the interpretive mode, and the final version compiled for speed. The algorithm used to train a network can be found elsewhere [11].



Table 2. Previous Case Data and Input Data

No	AMOUNT	AGE	FAMILY	JOB	SAVING	DECISION
1	above (1)	above (1)	yes(1)	professional(1)	high (1)	approve(1)
2	above (1)	above (1)	no (-1)	professional(1)	low (-1)	reject (-1)
3	average(0)	above (1)	yes(1)	professional(1)	high (1)	approve(1)
4	above (1)	above (1)	no (-1)	part time (-1)	high (1)	reject (-1)
5	below (-1)	above (1)	yes(1)	part time (-1)	high (1)	approve(1)
6	below (-1)	above (1)	yes(1)	professional(1)	high (1)	approve(1)
7	above (1)	average(0)	yes(1)	professional(1)	high (1)	approve(1)
8	below (-1)	above (1)	yes(1)	part time (-1)	low (-1)	reject (-1)
9	average(0)	below (-1)	no (-1)	full time (0)	low (-1)	reject (-1)
10	average(0)	average(0)	yes(1)	professional(1)	high (1)	approve(1)
11	average(0)	below (-1)	yes(1)	part time (-1)	medium(0)	reject (-1)
12	above (1)	average(0)	no (-1)	full time (0)	low (-1)	reject (-1)
13	average(0)	above (1)	yes(1)	full time (0)	high (1)	approve(1)
14	average(0)	average(0)	yes(1)	professional(1)	medium(0)	approve(1)
15	below (-1)	average(0)	yes(1)	professional(1)	high (1)	approve(1)

EDLIN line editor from DOS has been used to create and modify the training pair data file. The user can also use other wordprocessing software to create the data file by using the DOS option on the menu window to temporarily leave the KnowledgePro environment. The data file must, however, be in the ASCII format, a standard text file.

LOAN ADVISOR

In evaluating a loan application, a bank officer takes into account several factors. She considers the amount of the loan (AMOUNT), the age of the applicant (AGE), his family status (FAMILY), his job (JOB), and the amount of money in his savings account (SAVING). The outcome of the decision (DECISION) could be either "Approve" or "Reject". Data on previous applications are available from bank records. The data was reduced by grouping them into ranges. Since ANN1 was written to be a general program that can be used to handle a variety of problem domains, its input data is in the form of 1, 0, and -1. The data on previous cases were summarized in Table 2 with assigned numerical values. These previous cases are used to train a network. Once the network is trained, it can be used to evaluate new loan applications.

CONCLUSION

ANN1 represents a new wave of educational software that integrates hypertext with simulation programs. At the time of writing, we have not yet tested the program with students to obtain their responses. We have, however, informally tested a hypertext version of the Graduate Engineering Catalog at Christian Brothers University. All students indicated that they preferred the hypertext version over the printed version. Several of their comments, however, highlighted weak points of the hypertext document

such as hardware dependency and lack of portability. Many indicated that they would like to be able to browse through the hypertext document in a linear fashion, as they would be able to do with a printed book. We think that this feature may be necessary for a transition period during the in the next few years until students are familiar with nonlinear viewing of information. This certainly is not a surprise if we consider how many years we have been trained to view information in a linear sequence. It will probably take sometime to develop skills in reading hypertext documents.

REFERENCES

- [1] Simpson, P. K., Artificial Neural Systems, Pergamon Press, New York (1990).
- [2] Maren, A. et al, Handbook of Neural Computing Applications, Academic Press, San Diego (1990).
- [3] Shneiderman B., and Kearsley, G., Hypertext Hands-on!, Addison-Wesley (1989).
- [4] Bush, V., "As we may think", Atlantic Monthly, 176(1), 101- 108 (1945).
- [5] Beeman, W. O., Anderson, K. T., Bader, G., Larkin, J., McClard, A. P., McQuillan, P. J., and Shields, M., "Hypertext and pluralism: From lineal to nonlinear thinking", Proceedings of the Hypertext'87, ACM, New York (1987).
- [6] Knowledge Garden, Inc., KnowledgePro Version 1.43 (1989).
- [7] Owl International, Inc., GUIDE Version 3.0 (1990).
- [8] Cognetics Corp., Hyperties Version 2.35j (1988).
- [9] Microsoft, Paintbrush Version 2.0 (1987).
- [10] Microsoft, BASIC Compiler Version 2.0 (1985)
- [11] Soucek, B. and Soucek, M., Neural and Massively Parallel Computers, John Wiley & Sons, New York (1988).



SIRIPONG MALASRI

Siripong Malasri, Ph.D. from Texas A&M University, is Associate Professor of Civil Engineering at Christian Brothers University. His interests include artificial intelligence, expert systems, artificial neural networks, and hypertext. He has authored several articles and offered seminars on these subjects. Dr. Malasri is a registered professional engineer in Tennessee and Florida.

STANLEY P. FRANKLIN

Stanley P. Franklin is a professor of mathematical sciences and a member of the Institute for Intelligent Systems at Memphis State University. His research interests include artificial neural networks and mechanisms of mind. In particular, he's concerned with neural network computability, with training neural networks via parallel genetic algorithms, with neural network applications to medical diagnosis, and with enculturation of artificial agents. His B.S. is from Memphis State, his M.A. and Ph.D. from UCLA, all in mathematics.



Engineering Systems Evaluation Using Classical and Bond Graphic Methods, and Computer Simulations

C.T. Lin

California State University, Northridge

Abstract

Bond graphs are a fairly new approach for evaluating the dynamic response of a system. This paper will attempt to show how this new approach lends itself to the study of dynamic systems and how this method can be utilized in courses ranging from system simulation to process control. Issues on computer laboratory assignments tailored to the course study will also be addressed. The two examples, one involving an electro-mechanical motor system and the second involving a hydro-mechanical actuator, will demonstrate the versatility of the bond graph method when the system model consists of various energy domains. Optimization of the system performance through simulation, using an available software program called ENPORT, will also be discussed.

Bond Graph Introduction

A bond graph is an alternative method used to model physical systems. This method was originated by Professor H. Paynter¹ and further developed by Professors Karnopp and Rosenberg² and others³. The development of the bond graph techniques, some 30 years ago, have enabled simple and complex dynamic systems to be accurately modeled and solved with ease and with a better understanding of the system. These bond graphs have produced a convenient yet systematic way to model the system so as to be able to predict its dynamic behavior. The bond graphs are particularly suited to the development of models of dynamic systems in which energy occurs in several forms.

Bond graphs relate the power transferring among sub-systems where power is the product of an effort variable (e.g. force) and a flow variable (e.g. velocity). Most physical and engineering systems can be represented directly in terms of these component energy characteristics and their power interactions. The interactions of these two types of power variables are related by the cause and effect relationship. Bond graphs are also quite versatile in so far as this graphical approach can lend itself to modular additions. Once the primary system is defined, an expansion of other energy sub-systems can be added with relative ease. The same method used to solve the examples in this paper can be further expanded to more complex systems.

At first glance, the bond graph, shown in

Figure 1, looks like a graphical representation of chemical bonds in a molecule. The elements of the bond graph, where L_e = electrical inductance, R_e = electrical resistance, C_b = shaft compliance, which is equal to $1/k_b$, R_b = shaft/bearing resistance, I_c = rotor inertia and R_c = mechanical resistance, appear to be assembled similar to the atoms of a chemical bond. This type of graphical representation allows the user to relate the effect of an element, or component, on the interaction of the system.

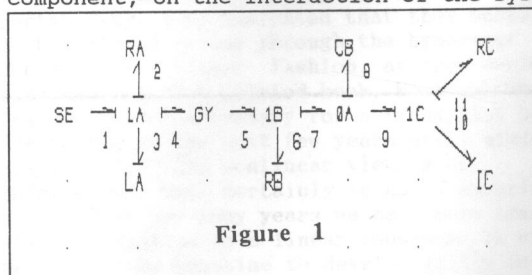


Figure 1

Addition of storage elements, such as, inertances, compliances, or resistive elements to an existing system could require extensive mathematical manipulation if pursued using classical methods, but this is not so with bond graphs. Additions of more storage elements can be added without affecting the base structure of the original bond graph, in essence, the added elements merely bond to the existing structure just as many molecules can absorb atoms. The system can then be reduced into the state equations just as easily as the base system was.

This paper will show only the basic bond graph elements required to solve two simple engineering system examples. A complete reference to all of the bond graph elements and procedures of modeling a system can be found in References 2 & 3 which are books tailored to a more complete introduction of bond graph theory. The next section will elaborate on the potential advantages that bond graphs have for both the undergraduate and postgraduate engineering program.

Curriculum Issues

As engineering technology continues to expand and the increasing concern that industry must develop systems that are



guaranteed to work, without modifications, prior to build and testing, in order to stay competitive, further reinforces the need to develop educational programs tailored toward system dynamic analysis and simulation.

A more thorough understanding of engineering fields other than the student's specialty discipline is now required in order for the student to maintain a competitive edge. The introduction of bond graphs and their unified approach to energy systems will help develop this competitive edge. Bond graphs introduce a new technique where the evaluation of effort and flow principals are the same regardless of which energy regime is involved. This new method reduces the chance for errors in developing the system response regardless of how many various energy domains are used.

There are presently two courses offered at California State University, Northridge, that utilize bond graphs. The under-graduate course is identified as "System Dynamics: Modeling and Analysis" and is tailored at developing the student's ability to solve for the response of engineering systems using classical and bond graph methods. The course uses "Introduction to Physical System Dynamics" as the required text. The course introduces the bond graph approach for most of the standard energy domains (mechanical and electrical) having linear and non-linear elements and requires a few computer projects to acquaint students with the modeling procedures for the given engineering systems and the ENPORT software which is specifically developed for the bond graph method.

This course attempts to give the student a firm understanding on what engineering systems are and how the dynamic response of the system can be developed. This course provides the background for further courses in state feedback control, which require it as the prerequisite, and mechanical vibrations and simulation.

The post graduate course, "Simulation of Dynamic Systems", starts with an overview of bond graph theory with linear systems, and then proceeds to non-linear systems involving mechanical, hydraulic, acoustic, electronic, thermal and transducer systems. Application of bond graphs to feedback control systems and mechanical vibrations are also introduced. The second half of the course involves introduction of bond graphs and other modeling methods that can be used to derive the transient response of systems containing artificial intelligence, or other complex systems, such as, radiation energy interaction and rocket propulsion. A project involving the development and simulation of a system, chosen by the student, is required.

The graduate course provides an understanding of the multiple approaches available to solve for the response of a system. It further develops an understanding of how any energy system can be evaluated and optimized prior to the physical model being built.

Bond graphs and the ENPORT⁵ software program have been used to evaluate and simulate dynamic responses on projects involving robotics, solar energy vehicles and

rocket propulsion systems.

Presently most computer modeling and simulation using the ENPORT software is done on P.C.s dedicated to the Mechanical Engineering department. This imposes a limitation since all other departmental courses utilize the same computers. The ENPORT software version available on the time-sharing main frame can reduce the amount of students on each computer terminal, but other limitations do exist. ENPORT is developed around the Tektronix graphics software. This software interfacing is used for the graphical display of bond graphs. Without having Tektronix work stations or their equivalence available, a graphical terminal emulator on the P.C.s is required to run ENPORT. This imposes some extra work on the students by requiring learning the emulator besides learning ENPORT. This dilemma also exists with getting access to and maintaining the software. In fact, the graphical outputs cannot be accomplished when using the main-frame computer system. This is due to the incompatibility of the text terminals with the ENPORT main-frame software version. Also producing a hard copy of the color coded energy levels as they appear on the bond graph, without having graphical plotters available, can only be accomplished by taking a "photograph" of the terminal screen.

There are plans including floor space in the new Engineering Additions to develop a system dynamics lab dedicated to developing and evaluating engineering systems from the individual research projects to the group under-graduate projects. This future expansion of the existing system dynamics and modeling laboratory will help alleviate this dilemma.

The rest of this paper will be dedicated to the development and evaluation of two simple engineering systems.

Examples

The two examples developed here will show the versatility of bond graphs. The first example involves both the electrical and rotational energy domains, while the second example involves the linear mechanical and hydraulic energy domains.

1. Electro-mechanical Example

The bond graph shown in Figure 1 represents the system model shown in Figure 2. The state equations, for this electro-mechanical system, using bond graphs, will now be defined.

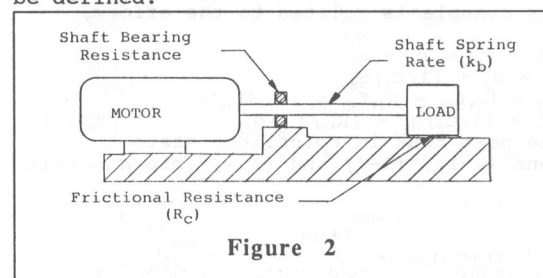


Figure 2

There are only two types of state

variables defined in the bond graph. The first type, for an inertia element, is a generalized momentum "p" which is related to its effort "e" by $dp/dt = e$; while the second type, for a capacitor or a compliance element is a generalized displacement "q" which is related to its flow "f" by $dq/dt = f$. Thus, the state variables from Figure 1 can be identified as:

- (1) p_a (or p_3) that has an effort e_3 (armature inductance voltage drop) is related to the energy storage element L_a (the inductance).
- (2) q_b (or q_8 , shaft angular twist) that has a flow f_8 (shaft angular velocity) is related to the storage element C_b (or C_8 , shaft compliance), and
- (3) p_c (or p_{10}) that has an effort e_{10} (torque exerted on the load) is related to the storage element J_c (or J_{10} , mass moment of inertia of the load).

Note that the subscripts a, b, c are used only to distinguish the sections of the system ("a" for the motor, "b" for the shaft, and "c" for the load), and the integer subscripts are assigned to the elements by arbitrarily numbering the bonds.

Using the bond graph rules defined in Reference 2, the state equation for " p_3 " results:

$dp_3/dt = e_3 = e_1 - e_2 - e_4$
 $e_1 = S_e = V_a$, which is the motor terminal voltage as an input source to the system.

$e_2 = R_2 f_2 = R_2 p_3 / I_3$, where R_2 is the terminal resistance of the motor.

$e_4 = M f_5 = M f_6 = M e_6 / R_6 = (M/R_6)(e_5 - e_7)$, where "M" is the conversion parameter, called "modulus" and R_6 is the bearing friction.

Since, $e_5 = M f_4 = M f_3 = M p_3 / I_3$,

and $e_7 = e_8 = q_8 / C_8$

Thus, $e_4 = (M^2 / I_3 R_6) p_3 - (M/R_6 C_8) q_8$

By substitution,

$$dp_3/dt = V_a - ((R_2/I_3) + (M^2/I_3 R_6)) p_3 + (M/R_6 C_8) q_8 \quad (3-1)$$

The next state variable, " q_8 ", is associated with the shaft compliance. The flow variable, " f_8 ", is required here since we are trying to find the angular velocity difference of the elastic shaft, from location "b", near the motor, to location "c", near the load. The state equation for " q_8 " results:

$$dq_8/dt = f_8 = f_7 - f_9$$

$$f_7 = f_6 = e_6 / R_6 = (e_5 - e_7) / R_6 = (M/I_3 R_6) p_3 - (1/R_6 C_8) q_8$$

$$f_9 = f_{10} = (1/I_{10}) p_{10}$$

$$dq_8/dt = (M/I_3 R_6) p_3 - (1/R_6 C_8) q_8 - (1/I_{10}) p_{10} \quad (3-2)$$

The last state variable, p_{10} , of concern for this example is related to the effort, " e_{10} ":

$$dp_{10}/dt = e_{10} = e_9 - e_{11}$$

$$e_9 = e_8 = (1/C_8) q_8$$

$$e_{11} = R_{11} f_{11} = R_{11} f_{10} = (R_{11}/I_{10}) p_{10}$$

$$dp_{10}/dt = (1/C_8) q_8 - (R_{11}/I_{10}) p_{10} \quad (3-3)$$

The next task is to collect state equations (3-1), (3-2) and (3-3) into a matrix format:

$$d/dt \begin{bmatrix} p_3 \\ q_8 \\ p_{10} \end{bmatrix} = \begin{bmatrix} -(R_2/I_3 + M^2/I_3 R_6) & M/R_6 C_8 & 0 \\ M/I_3 R_6 & -(1/R_6 C_8) & -1/I_{10} \\ 0 & 1/C_8 & -R_{11}/I_{10} \end{bmatrix} \begin{bmatrix} p_3 \\ q_8 \\ p_{10} \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} V_a \quad (3-4)$$

Notice that only simple mathematics and algebra were used to generate the state equations shown above. The three state equations can be easily implemented into existing computer algorithms. This is one of the advantages with using bond graph methods.

The electrical schematic and the free body diagram (FBD) that represent the motor assembly of Figure 2 are shown in Figure 3. The classical derivation of the schematic and the FBD will result in the electrical domain containing one differential equation and the mechanical domain containing two differential equations. Since these three equations are not in the state equation format, further mathematical manipulation will be required to develop the state equations.

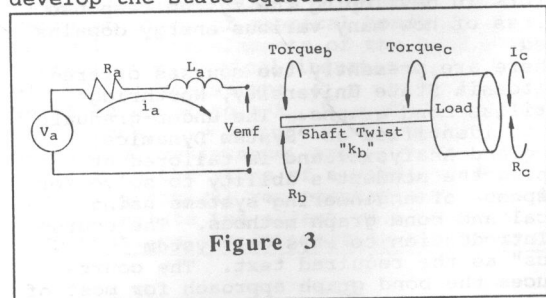


Figure 3

The three differential equations are:

$$V_a = R_a i_a + L_a di_a/dt + V_{emf} \quad (3-5)$$

$$T_b = w_b R_b + k_b (\Theta_{bc}) \quad (3-6)$$

where $\Theta_{bc} = \int (w_b - w_c) dt$, and R_b = bearing friction.

$$T_c = w_c R_c + I_c dw_c/dt \quad (3-7)$$

The back electromotive force " V_{emf} " is equal to:

$$V_{emf} = k_b w_b \quad (3-8)$$

where " k_b " is the constant which relates the motor shaft speed " w_b " to the " V_{emf} ". The " k_b " shown here, is identical to the "M" in the bond graph derivation.

By combining and manipulating the mechanical equations (3-5), (3-6) and (3-7) the following matrix can be developed: which is identical to Equation (3-4).

$$d/dt \begin{bmatrix} p_3 \\ q_8 \\ p_{10} \end{bmatrix} = \begin{bmatrix} -(R_2/I_3 + M^2/I_3 R_6) & M/R_6 C_8 & 0 \\ M/I_3 R_6 & -(1/R_6 C_8) & -1/I_{10} \\ 0 & 1/C_8 & -R_{11}/I_{10} \end{bmatrix} \begin{bmatrix} p_3 \\ q_8 \\ p_{10} \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} V_a \quad (3-9)$$

2. Hydro-mechanical Transducer Example

In the transducer shown in Figure 4, we attempt to show a relationship between the classical and the bond graph methods when hydraulic and mechanical energy domains are involved. This hydro-mechanical system uses the inlet chamber pressure to oppose the mechanical force. This pressure to force relationship is used to maintain a constant piston location. Friction on the mechanical side keeps the piston from developing a resonance by requiring a moderate "effort differential" between the mechanical force and the chamber pressure prior to piston movement. The different energy domains are related by the hydraulic pressure (force) to the mechanical force.

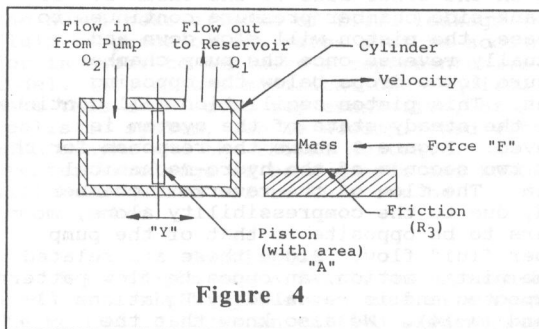


Figure 4

Figure 5 shows the complete bond graph for the hydro-mechanical system of Figure 4. There are three sources identified. They represent the hydraulic pump and reservoir as flow-source " S_{f7} " and " S_{f11} " and the mechanical force represented as " S_e ".

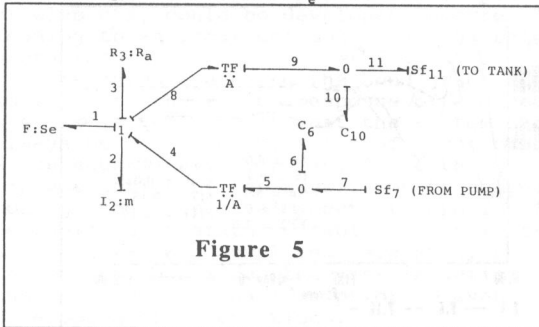


Figure 5

Solving the bond graph using the same process as shown in the motor example will result in the following state equations:

$$\frac{dp_2}{dt} = -S_e + Aq_6/C_6 - R_3p_2/I_2 - Aq_{10}/C_{10} \quad (3-10)$$

$$\frac{dq_6}{dt} = S_{f7} - Ap_2/I_2 \quad (3-11)$$

$$\frac{dq_{10}}{dt} = Ap_2/I_2 - S_{f11} \quad (3-12)$$

The solution using the classical method results directly from Reference 4. Accounting for the compressibility of the fluid and the relationship between the chamber pressure and volume when the piston is in motion will result in the three equations below:

$$Q_2 = A \cdot \text{vel} + (\text{Vol}/B) \cdot (dp/dt) \quad (3-13)$$

$$Q_1 = A \cdot \text{vel} - (\text{Vol}/B) \cdot (dp/dt) \quad (3-14)$$

where A = piston area, $\text{vel} = dy/dt$ = piston velocity = p_2/I_2 , vol = chamber volume, B = Bulk modulus of elasticity, dp/dt = time rate of change in chamber pressure, Q_2 = in flow rate = S_{f7} , Q_1 = out flow rate = S_{f11} , dq_6/dt = inflow rate change due to fluid compressibility, and dq_{10}/dt = out flow rate change due to fluid compressibility

$$\text{and; } m \cdot d^2y/dt^2 + R_b \cdot \text{vel} = -F_1 + A(P_2 - P_1) \quad (3-15)$$

where, $m \cdot d^2y/dt^2$ = inertia force = dp_2/dt , R_b = friction = R_3 , F_1 = applied force = S_e , P_2 = pump pressure = q_6/C_6 , and P_1 = tank side pressure = q_{10}/C_{10} .

It is clear that the last set of three equations is identical to the set of state equations (3-10), (3-11) and (3-12) obtained from the bond graph approach. Now, these state equations derived from the above two examples can be used to determine the effect

any element has on the complete system.

Computer Simulation

In this section we will show how these bond graphs can be utilized to directly derive system responses without any required state equation derivations. The principal objective of the ENPORT³ software is to require the user to input only the energy elements involved (mechanical, electrical, hydraulic, etc...), the conversion parameters, if any, between the different energy domains and the coefficients required for each of the elements used. Although the examples used here do not involve non-linear elements, the ENPORT software does allow definition of non-linear elements.

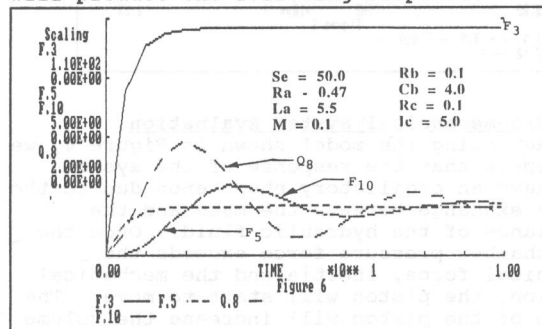
Once the above three requirements have been accomplished, assuming the model does not have any energy storage elements in differential causality, which is currently not supported by ENPORT, the software will evaluate the model for transient and steady-state responses.

Two versions of this program are available. The full version runs on a main-frame computer, such as VAX. The simplified version can be run on a P.C. computer having "DOS" residence software. The full version ENPORT has the benefit of operating with many of the basic source inputs (unit step, ramp, pulse, etc...) and a comprehensive library of system-defined mathematical functions and will accept user-defined non-linear elements. The P.C. version is somewhat limited and is only useful when evaluating linear systems that are not too complex and that only require a constant source. All of the ENPORT runs shown here were accomplished on the P.C. version of ENPORT.

Figure 1 shows an ENPORT program display of the motor-load bond graph model complete with the cause and effect element relationships which are indicated by the short straight strokes. This plot was produced on a dot matrix printer by activating the DOS "GRAPHICS.COM" prior to entering ENPORT. A specific software driver is required to get a plot of the bond graph from a laser printer.

1. Motor-Load System Evaluation

Using the set of element parameters, shown in Figure 6, for a small servo motor with a mass at the end, as shown in Figure 2, will produce the following response.

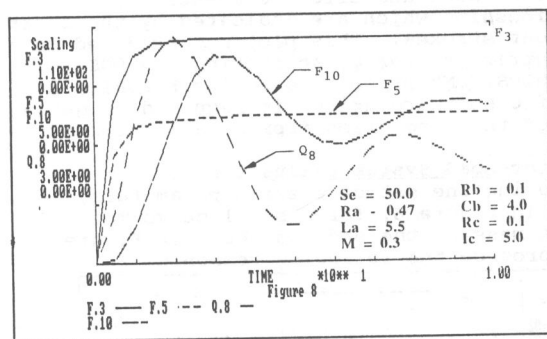
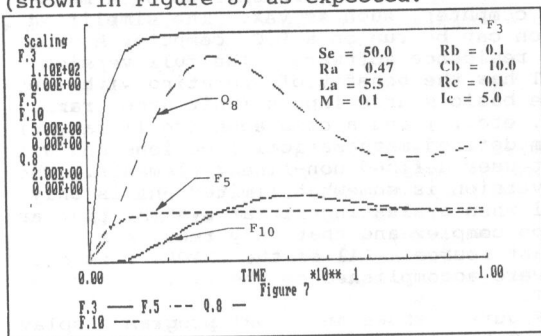


The four curves plotted represent the motor current, F.3, the shaft velocity at the motor end, F.5, the shaft twist, Q.8, and the

velocity of the load, F.10. The response of this system with these defined parameters indicate an underdamped response in the mechanical regime.

Decreasing the shaft stiffness will increase the settling time (refer to Figure 7). A greater amount of twist will be required to move the load since the shaft spring rate is less. The shaft speed oscillations will continue until an equilibrium speed is achieved.

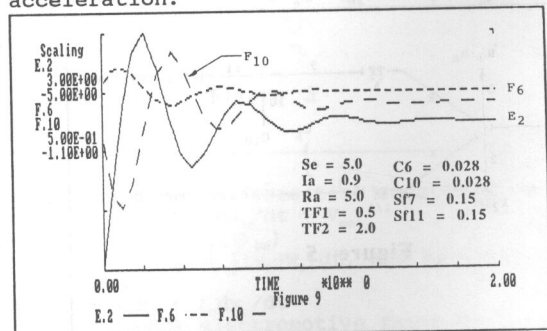
A change in the gyrator modulus "M", from the .1 defined in the previous ENPORT runs, should not change the frequency of the load but should affect the amplitude of the load speed since the back electromotive force is equal to the shaft speed times the gyrator modulus, and is related to the motor current and the shaft torque. Therefore, increasing the gyrator modulus should increase the shaft speed and the shaft twist according to Equation (3-8). Figure 8 is an ENPORT run that increases the gyrator modulus to 0.3. The result does show an increase in the motor shaft speed "F.5" and in the load speed "F.10" (shown in Figure 8) as expected.



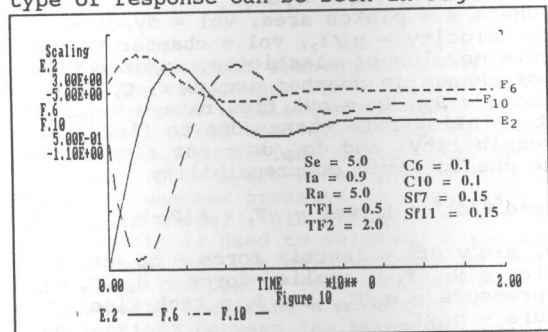
2. Hydro-mechanical System Evaluation

Reviewing the model shown in Figure 4, we can expect that the response of the system will have an oscillatory phenomenon due to the energy exchange between the mass and the compliance of the hydraulic fluid. Once the pump chamber pressure force exceeds the mechanical force, inertia and the mechanical friction, the piston will start to move. The motion of the piston will increase the volume on the pump chamber side which will temporarily reduce the pressure in the pump chamber and build up the elastic restoring

force on the other side of the chamber. If the tank-side chamber pressure continues to increase, the piston will slow down and eventually reverse once the pump chamber pressure force drops below the opposing forces. This piston oscillation will continue until the steady-state of the system is achieved. Figure 9 shows the response for the first two seconds of the hydro-mechanical system. The flow of the reservoir chamber fluid, due to the compressibility alone, appears to be opposite to that of the pump chamber fluid flow. Since these are related by the piston motion, an opposite flow pattern is expected and is revealed by Equations (3-13) and (3-14). We also know that the mechanical mass peak acceleration occurs at the largest fluid velocity change which agrees with the "E.2" curve shown in Figure 9. The effort of the mechanical mass is $E.2 = \text{force} = \text{mass} * \text{acceleration}$. Since the mass is constant, E.2 indirectly represents the mass acceleration.



If the hydraulic fluid was replaced with a fluid that was more compressible, we can visualize that the frequency of oscillation of the mechanical mass would decrease from the previous response. This would be due to the ability of the fluid to absorb energy. And since the pump is defined as a constant flow pump, the pump chamber fluid would take longer to absorb the energy required to overcome the mechanical force, but once the pressure exceeded the opposing force, the piston motion would move for a longer duration. In fact, the piston would be displaced farther. This type of response can be seen in Figure 10.



The objective of this section was to show the benefits of using ENPORT and bond graphs to develop, evaluate and optimize a system model. In most realistic hydraulic actuators

the non-linear effects of fluid leakage and fluid compressibility rates must be accounted for in order to evaluate a representative model. All of the modifications required to make this model more representative, of a realistic system, can be accomplished with bond graphs and the ENPORT software. A greater knowledge of bond graphs and ENPORT will be required before this can be accomplished.

Conclusion

Bond graphs are a fairly new approach used for evaluating the response of a system. The method utilizes a unified element/bond approach for showing the power flow and the energy interactions between the elements in the system.

For engineering students, not well versed in the development of the conventional electrical schematics and free body diagrams, some of the relationships between components, or elements, could be developed incorrectly leading to an incorrect model, and therefore, wrong results.

An advantage using the bond graph methods to evaluate the system response of engineering systems is the benefit that the system can always be described by a set of first order state equations, which facilitate the implementation of the algorithms to computer coding, thus the development of ENPORT. The recognition of state variables directly from the inertial and compliance elements, by inspection, has aided the user in defining the number of state equations prior to even developing state equations.

System responses due to variations in some of the element parameters can readily be evaluated using ENPORT. Results due to element parameter changes were easily made to the two example models which further supports the simplicity of generating and evaluating systems using the bond graph method and the ENPORT software package. There are not many software programs that may facilitate system modeling and easily develop its dynamic response with such simplistic user interaction. Most available software, we're familiar with, requires either the state equations, transfer functions, or a complete model to develop the response of the system, which often times are not easily derived.

The ENPORT software package was not without its own problems. Difficulties encountered in plotting graphical displays and the inability to evaluate models that have energy storage elements in differential causality left this package far from complete. An approach that can be used to allow evaluation of a system with an element that has differential causality, is to modify the bond graph by adding parasitic elements, such as a very stiff spring for a rigid body or an extremely small resistance for frictionless surfaces in a manner that will turn all the storage elements including the added parasitic element into integral causality and that will still represent the model. These parasitic elements can be defined as having minimal effect on the system model.

The simple examples shown above indicate the versatility and simplicity of developing

bond graphs that can be used to solve for the response of a system. Engineering projects, such as senior design projects and graduate projects, involved with understanding the energy interactions of the system can use bond graph theory to quickly derive the system equations. A simple version of a complex system model could be developed initially representing the project and expanded utilizing the modular addition approach until a model representing the complete system is developed. Once the system is developed, the ENPORT software program could be utilized to optimize the element parameters and the system response. This would alleviate many of the redundant steps commonly occurring in university projects.

References

1. H.M. Paynter. Analysis and Design of Engineering Systems, Cambridge, MA: MIT Press 1961.
2. D.C. Karnopp and R.C. Rosenberg. Introduction to Physical System Dynamics, McGraw-Hill, 1983.
3. J.U. Thoma. Introduction to Bond Graphs and their Applications, Pergamon International, 1975.
4. C.W. deSilva. Control Sensors and Actuators, Prentice-Hall, 1989.
5. ROSENCODE Associates, Inc. Enport-7.0, Lansing, MI: 1987



CHI-TSEN LIN

Dr. Chi-Tsen Lin, Associate Professor of Mechanical Engineering at California State University, Northridge, is a consultant on robotics and automation to industry and the Naval Air Engineering Center. He has 10 years experience in engineering and teaching capacities, including positions with the U.S. Army Material Command, Bosch Corporation, Germany and the University of Colorado, Denver where he was an Assistant Professor from 1983-1987.

Dr. Lin earned his Ph.D. degree in Engineering from the University of California at Davis, with emphasis on Controls, Mechanical Design and Data Processing. He has been continuously teaching these subjects at universities. He has also demonstrated a continuing effort in research and development of technology through funded university and Naval projects. Most of these projects involve design and control of autonomous robotic vehicles, robotic arms and walking machines. He is a member of ASME, IEEE, RI/SME, SAE and ASEE.

ROBERT (BOB) P. GIBSON

Robert (Bob) P. Gibson is a Design Engineer in the Advanced Combustion Devices Design Group at Rocketdyne Division, Rockwell International Corporation. Since 1987 he has been involved in redesigning fluid heat exchangers, cryogenic fuel injectors, thrust nozzles and the structural chambers used on the Space Shuttle Main Engines. From 1985 - 1987 he worked at Bendix, Oceanics Division as a Quality Engineer and from 1979 - 1985 he worked as a Research Engineer at Cyprus Industrial Minerals Company, Research Division, a subsidiary of Amoco Oil Corporation, developing and evaluating polymer composites for both the rubber and plastic industries. Bob received his B.S. Degree in Mechanical Engineering from California State University Northridge in 1985 and his M.S. Degree in Mechanical Engineering from California State University, Northridge in 1991.



Interactive Approach to Numerical Experiments on Personal Computers: Case Study for Engineering Students

F. Finaish

University of Missouri-Rolla

Summary

The paper presents an approach and a case study on the use of personal computers in performing numerical experiments designed to complement and enhance classroom instruction and/or physical laboratory learning experience. The approach allows for performing numerical experiments that focus on the dependence of the problem solution on a wide parameter range. As an example, a simplified numerical model that simulates the translational and rotational dynamic response of a rigid wing model mounted on elastic supports and subjected to a steady flow is presented.

The article may serve as a reference on development of numerical experiments suitable for engineering educational purposes, and/or provide a guide to design physical experimental arrangements. It is hoped that the study will help introduce the use of the numerical experiment approach in engineering education and inspire further efforts to enhance this concept.

Nomenclature

A = Dimensionless aerodynamic constant
 $= \rho g S / 2 k_z$
 a = Elastic axis location
 C_z = Dimensionless translational damping
 $= 2\pi c_z / \sqrt{m k_z}$
 C_α = Dimensionless rotational damping
 $= (2\pi c_\alpha \sqrt{m / k_z}) / I$
 c_l = Wing lift coefficient
 c_z = Translational damping coefficient
 c_α = Rotational damping coefficient
 g = Acceleration due to gravity
 I = Wing mass moment of inertia

K_z = Dimensionless translational stiffness
 $= k_z / (m / t_c^2)$
 k_z = Translational spring stiffness
 K_α = Dimensionless rotational stiffness
 $= m k_\alpha / I k_z$
 k_α = Rotational spring stiffness
 L = Wing lift
 l_c = Characteristic length
 M = Dimensionless inertia = $agm^2 / I k_z$
 m = Wing mass
 S = Wing area
 t = Time
 t_c = Characteristic time
 T = Dimensionless time = t / t_c
 w_c = Characteristic velocity
 ω = Wing angular wing velocity, $\omega = d\alpha / dt$
 U_∞ = Free stream velocity
 U = Dimensionless velocity in x-direction
 $= U_\infty / w_c$
 v = Wing vertical velocity, $v = dz / dt$
 V = Dimensionless velocity in z-direction
 $= v / w_c$
 z = Vertical amplitude
 Z = Dimensionless wing vertical amplitude
 $= z / l_c$
 α = True angle of attack
 α_0 = Static angle of attack
 $\Delta\alpha$ = Induced angle of attack
 Ω = Dimensionless wing angular velocity
 $= \omega / (1 / t_c)$
 ρ = Air density

Introduction

Computers became one of the primary tools employed by industry to solve engineering problems in all disciplines. Almost every phase of design in major companies is either on the computer or heavily dependent upon computers. These advancements are forcing a restructuring of engineering courses and demand changes in college education. One step towards the direction of these changes is the use of personal computers as an educational tool. In engineering, such computers



are heavily used for text processing, program editing, and communicating with larger computers. However, the use of these machines in engineering education remains limited, and not enough scientific computation is done on them. With the advances in computer technology, these machines become adequate for modeling engineering problems including, in particular, illustrative examples for use in instruction. Just as a physical laboratory equipment is valuable as a component of undergraduate laboratory course, so should computations originally done for research purposes be reworked to provide practice in numerical experimentation¹. Computer programs can be developed and installed in a personal computer laboratory where students can perform numerical experiments. Specific assignments can be made for students, not only to experiment with different value of parameters in the programs, but also to modify the program in order to change the output or expand them in other ways.

The main objective of this article is to present an approach that provides an electronic learning environment, which enables students to study a wide variety of configurations for a given problem. To accomplish this task, a user interactive program is developed. This program allows the user to manipulate a given set of input parameters, compute output results, and describe "graphically" the relationships between selected output variables. To demonstrate this approach a case study that focuses on the dynamic response of a rigid wing model in steady freestreams is selected. Our aim is to develop a numerical experimental setup that allows for investigating the dependence of the translational and rotational wing motions on the problem parameter space. In the subsequent sections we discuss the governing equations, the numerical method employed to solve the problem, and the program organization and setup. Some results which show the vertical and angular wing response for several configurations are presented.

Governing Equations:

Figure 1 presents a schematic of the wing model. The wing is supported by translational and rotational springs of spring constants k_z and k_α respectively, which are related to the stiffness of a wing on an airplane. The cross section of the wing is considered to be a thin symmetric

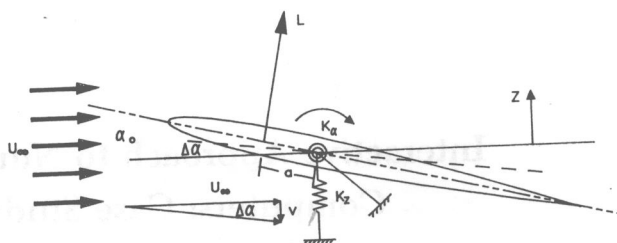


Fig. 1 A wing model mounted on translational and rotational elastic supports.

airfoil. In the absence of flow, the airfoil makes an angle α_0 with the horizontal and its center of mass sets at a height $z = 0$. Assume that the wing is supported so it can only rotate or move vertically. When the wing is subjected to a flow velocity U_∞ , the wing angle of attack can be written as

$$\alpha = \alpha_0 - \Delta\alpha = \alpha_0 - \tan^{-1}[v/U_\infty] \quad (1)$$

where v is the vertical flow velocity due to vertical wing motion. The wing lift normal to the flow direction is given by

$$L = 1/2\rho(U_\infty^2 + v^2)Sc_l, \quad c_l = 2\pi\alpha \quad (2)$$

The assumption here is that the airfoil is thin symmetric² with a large aspect ratio and wing angle of attack is below the static stall angle of attack. In this study, we assume that $c_l = 2\pi\alpha$ holds for the range of α between -12 and 12 degrees. Beyond this range we assumed the wing stalled and hence c_l is zero. The airfoil moment coefficient is assumed to be zero. With these assumptions, the equations of the vertical and angular motions can be written as

$$m d^2 z / dt^2 = -c_z dz / dt - k_z z + 1/2\rho c_l S U_\infty \sqrt{U_\infty^2 + (dz/dt)^2} \quad (3)$$

$$I d^2 \alpha / dt^2 = -c_\alpha d\alpha / dt - k_\alpha \alpha + 1/2\rho c_l S U_\infty \sqrt{U_\infty^2 + (dz/dt)^2} \quad (4)$$

To write the governing equations in a dimensionless form, we select characteristic quantities for length, time, and velocity. The period of free oscillation of the system is taken as a

reference time $t_c = 2\pi\sqrt{m/k_z}$; the deformation of spring due to the wing weight as the reference length $l_c = mg/k_z$. From this, the ratio of l_c/t_c is chosen as a characteristic velocity

$$w_c = g/(2\pi\sqrt{m/k_z}).$$

The second order governing equations (3) and (4) can be rewritten as four dimensionless first order ordinary differential equations.

$$DZ/DT = V \quad (5)$$

$$DV/DT = -(2\pi)^2 Z - C_z V + AC_l U \sqrt{U^2 + V^2} \quad (6)$$

$$D\alpha/DT = \Omega \quad (7)$$

$$D\Omega/DT = -(2\pi)^2 K_\alpha \alpha - C_\alpha \Omega + AMC_l U \sqrt{U^2 + V^2} \quad (8)$$

where A and M are the dimensionless aerodynamics and inertia parameters, C_z and C_α are the translational and rotational damping parameters and K_α is the rotational stiffness parameter.

As can be seen, the number of parameters are reduced to five dimensionless parameters. Each set of values of these parameters represent a large number of combination of the dimensional parameters. For more comprehensive model on the dynamic response of a rigid wing model mounted on elastic supports or on the response of elastic wings rigidly supported, the reader may refer to the study by Strganac, et al⁴.

Numerical Method and Programming Approach

To integrate the governing differential equations, the fourth order Runge-Kutta integration scheme is employed. Details on the use of this scheme can be found, for example, in the book by Press et al.⁵, where details on the method and a routine, written in C, are given. As presented in the previous section, the second order governing equations can be rewritten as four first order equations which led to equations (5-8). The Runge-Kutta method requires the evaluation of the functions on the R.H.S. of equations (6) and (8) each time step. These functions represent the force and the moment acting on the

wing respectively. For this, the functions Force (), and Moment () are written to compute the force and moment on the wing each time step. To integrate the equations of motion the function Runge-Kutta () is developed.

All computer routines were written in C, employing the Microsoft Quick C compiler. For further details on this

compiler and on programming with Quick C, the reader may refer to the book by the Waite Group's³. The language was selected because it gives direct access to most capabilities of a machine-language instruction set. It has a large variety of data types, and concise syntax for effecting conversions and indirections. Also, it has a very high degree of portability from one machine to another. These capabilities make C powerful, efficient, and well-suited to PC applications that involve user interfaces. An IBM PS/2 model 30-286 running at 10 MHz with a math co-processor was employed.

Program Setup

The software is designed so that the user can interactively edit the input parameters, perform the numerical computations, and display the output results of selected variables. The program also allows the user to create data files to save the output results.

The basic screen layout of this program is shown in Figure 2. The screen is divided into three areas; a panel menu which shows the available options, an input strip and parameters display area. The first command "AIRFOIL", when selected, activates the

[AIRFOIL]	Input >
COMPUTE	Aircraft Wing Response in Steady Flow
GRAPH	
FILE	Flow speed (m/s) ==> 10.0
HELP	Wing angle of attack (deg) = 10.0
EXIT	Maximum time (sec) = 10.0
	Translational stiffness (N/m) = 1000.0
	Translational damping (N/m/s) = 1.0
	Rotational stiffness (Nm/rad) = 100.0
	Rotational damping (Nm/rad/s) = 0.5
	Elastic axis location (m) = 0.05

Fig. 2 Basic computer screen layout for user interface.

