Teaching Computer-Aided Mechanism Design and Analysis Using a High-Level Mechanism Toolkit

MATT CAMPBELL, HARRY H. CHENG

Integration Engineering Laboratory, Department of Mechanical and Aeronautical Engineering, University of California, Davis, California 95616

Received 19 July 2005; accepted 21 August 2006

ABSTRACT: A pedagogically effective teaching strategy that integrates computer-aided design and programming into a course on mechanism analysis and design is presented. Mechanism analysis is enhanced when coupled with computer programming that allows students to find solutions to more complex systems than would otherwise be possible. Web-based distance learning is part of the class and students also learn how to create these kinds of materials themselves. Students can better understand the course material through an integrated computing environment. By solving mechanism design problems in C/C++, the programming skills gained in the course are widely applicable in other areas of engineering. Ch, a C/C++ interpreter, is used to incorporate programming and mechanism design because of its high-level numerical and graphical plotting capabilities, scripting capability, and a mechanism toolkit with easy and quick animation. A student project is presented as an example to show how computer programming is integrated for effective learning. This teaching strategy has been actively used at the University of California, Davis for several years in an undergraduate course in computer-aided mechanism design and has been adopted by other universities as well. © 2007 Wiley Periodicals, Inc. Comput Appl Eng Educ 15: 277–288, 2007; Published online in Wiley InterScience (www.interscience.wiley.com); DOI 10.1002/cae.20156

Keywords: computer-aided mechanism design; C/C++ interpreter; Ch; Web-based design

INTRODUCTION

Mechanism design is a basic course taught to most mechanical engineering students that covers various planar linkages. Starting with the theory behind them, students learn the equations governing their motions, use these equations to analyze different types of mechanisms, and eventually move towards solving design problems. In most cases, simple loop closure equations are used to solve for link positions at a given time. By taking the derivative of the loop closure equations, the velocities and accelerations can be found. With accelerations, Newton's second law

Correspondence to H. H. Cheng (hhcheng@ucdavis.edu).
© 2007 Wiley Periodicals Inc.

can be used to find the force acting between the links thus completing the dynamic analysis of the mechanism. A computer-based approach to solve these equations becomes necessary as the shear number of equations can grow quite large.

There are many programs available to engineers that aid the design process. Some examples are Automated Dynamic Analysis of Mechanical Systems (ADAMS) and Working Model by MSC Software, complete analysis solutions for many types of problems or systems [1]; Synthetica, a spatial mechanism design package [2]; WATT by Heron [3], and SAM (Simulation and Analysis of Mechanisms) by Artas [4], used for the synthesis and analysis of planar mechanisms. With these types of GUI driven programs, users are required to learn the specifics of these program interfaces. Also, in the process of using such a software package, the equations are hidden from the user. All that is seen is the input and output. This is appropriate for many applications. For example in drafting or 3D solid modeling where much of the computation is graphical in nature. But in a learning environment, when dealing with mechanical design and analysis, hiding the equations from the students might keep them from learning the entire design process happening within the software. In most applications, it is beneficial for a mechanical designer to have a feel for the equations so that problems that invariably crop up are easier to troubleshoot. This is not to say that mechanism design softwares are not useful. They can give students an idea of different types or ranges of mechanisms, what they might be useful for, and an appreciation of the physical model [5]. It becomes a starting point for a class in mechanism design. Then students can move forward, analyzing mechanisms and working on mechanism designs themselves.

To strike a balance between the pencil and paper approach, which is burdensome due to repeated computation, and using purely GUI-based software that is often time consuming to learn with many unnecessary features, a pedagogically effective programming environment has been developed and used for teaching and teaching mechanism design and analysis [6]. The teaching strategy presented in reference [6] is based on low-level C programming. In this article, the integration of high-level programming for teaching mechanism analysis and design is described. Using the basics of object-oriented programming, it is easier to obtain graphical output, animation, and analytical results using built-in function libraries. Students can write very simple programs, about a half dozen lines, that can animate the mechanism and produce some graphs to give them an idea of how different mechanisms function. Once those basics are learned, the programs can be modified to perform more complicated analysis. The final goal would be for students to create a library of functions to perform analysis on their own and even using a Web browser to interface with these libraries for mechanism design and analysis.

In this article, a pedagogically effective method with computer programming to teach mechanism analysis and design is presented. Several advantages for this approach from the teaching standpoint are detailed, as are many of the benefits for including programming in the curriculum. Also, an argument for adding Web-based computing on a basic level is made. Ch [7−9], a C/C++ interpreter, and its Mechanism Toolkit [10] are presented as teaching tools that will easily enable the computer-aided approach to be combined with the traditional materials. A student project is presented to show how a C++ class is developed to perform mechanism analysis. The program is then extended to interface with a Web server so that analysis and animation can be performed through a Web browser interactively. The ideas presented are applicable to teaching other courses in engineering.

TEACHING MECHANISM DESIGN AND ANALYSIS WITH COMPUTER PROGRAMMING

An ideal teaching environment would allow students to see a variety of mechanisms and reduce the computational complexity while still giving them access to the pertinent equations. Furthermore, students should be allowed to plug in their own equations to see how well they work within the framework of the software. With this approach, the theory taught can be applied in design and analysis. Students can even write their own programs. This allows them to explore different aspects of the design process without the need for repetitive computations. The translation of formulas and equations to analysis software is transparent and nothing is hidden. The program will most likely be specialized for one set of equations or type of mechanism.

The process of creating the program, however, is also a valuable learning experience. Looking at the numerical methods for solving equations from a data handling perspective can be helpful. It forces students to think about the order of operations, what the necessary pieces of information are, and where to begin when starting a new design. This principle is also applicable to other areas of engineering as well.

Students will appreciate and better understand the comprehensive commercially available software packages. They will seem less like a black box once students have some experience designing their own software [6].

Having students integrate the necessary equations into their own program works toward improving both mechanical analysis skills and the ability to see a path to the solution of a set of equations. Coding the equations reinforces the theory. It helps students remember, not only what the equations are, but how to go about solving them because they literally need to plug one equation into the next. Follow-on courses in mechanisms may tackle more difficult design tasks, but because of the framework for analysis set up through this technique, more advanced analysis has a firm foundation. By having a solid starting point, adding functionality and complexity to mechanism design problems becomes much easier.

It is important for students in mechanical engineering to master some basic programming skills. Programming, in general, can be applied in all areas of engineering to solve many different problems. Basic programming skills can be taught quickly and utilized in combination with the course work. Programming is often a prerequisite in many engineering curriculums. To make these skills as widely applicable as possible, standard programming languages are the best choice to teach students who have not had a lot of programming experience. By selecting a commonly used programming language, instead of a commercial software package, the computing skills learned are not limited to one program and time is not wasted traversing an unfamiliar interface. There is no guarantee that a particular piece of software will be used at the company where the student ends up working, no matter how popular it is.

Once the student has a handle on some programming basics, they can begin to build software by starting with the simplest part: just solving for the position of the mechanism. From here a building block approach can be used to add different parts of the analysis as they are learned in class. Functions, as the blocks, reused with minor changes to the equations at their core, make it easy to add velocity and acceleration solving capability to an existing program. As the class progresses, students will learn more about mechanism analysis and can add to their own functions to analyze what it is that they just learned. In this case, the programs that the students create mirror the material covered in class and avoids being too complex in the beginning. This allows students to add just the functionality that they want or need to the program, expanding it with their knowledge in the subject. Because they are building each part, they can understand how it works [6]. Fixing errors in the calculations when they come up, turns out to be a part of the learning process. When using commercial software packages, one bad habit that is developed is the innate trust in the results. The student should feel the need to double check the results and not just accept them as a fact. One of the desired outcomes of doing some or all of the programming themselves, is a routine of using test cases to verify results.

The programming does not need to be complex with fancy user interfaces or lots of unnecessary options. Simple function driven programs with a minimum of error checking can accomplish the desired goal. This relieves students of the burden of having to put a multitude of features into a program. They can make it interactive, where values used for the calculations are provided at run time, or assign the values to variables before program execution.

WEB-BASED DISTANCE LEARNING AND INFORMATION SHARING

Beyond showing examples on the blackboard or on paper, having a way for students to interact with the subject matter is always preferable. Interaction and collaboration are becoming more common in learning. The Web is a natural choice to implement these ideas. Being platform and location independent, the Web allows distance learning where all that is needed is an Internet connection. As most students use the Web nowadays for a multitude of tasks related to school activities, it seems showing them how to make the Web a truly interactive learning tool makes sense.

The creation of examples aids in the basic understanding of the subject at hand. Mechanisms come in a wide variety of types and classes. Demonstrating all of these physically would be quite difficult. Software provides an easy way for students to see different examples and how they could ultimately be utilized. Again, this can be done with commercially available software, but it becomes necessary for each student to run his/her own copy of that software. Also, picking which software bundle becomes an issue because they all invariably have pros and cons. The Web can be used with some simple tools to provide interactive mechanism examples that students can explore on their own [11–13]. Much of this has already been done with the Ch Mechanism Toolkit, as shown in the next section. By using this set of high-level, simple tools, it provides a means for students to turn around and create something similar

by themselves. Most other software packages only let students see what others have done. It is more desirable to provide a means for students to create their own tools that can then be shared with others.

HyperText Markup Language (HTML) already provides the basic ability to create interaction. This can be done utilizing Common Gateway Interface (CGI) [14,15]. CGI can get user input via a fill-out form on the Web browser and pass it along to the program running on the Web server that will use it to perform the necessary calculations. The results can then be written into an HTML file by the same program doing the calculations and sent back to the user. HTML can be used in a very simple way that anyone with some prior programming experience can pick up with just some introductory material and a few examples. With a few basic commands in HTML and CGI, a Web page can be created that allows students to add a Web User Interface (WUI) to the program. The nature of the Web and its global accessibility then translates into collaboration between students, allowing them to share their work with others and work together. By providing an interface that is both easy to understand and to share, students can help each other to increase their level of understanding of the course material.

DESIGN AND ANALYSIS USING Ch MECHANISM TOOLKIT

To aid students starting out in a mechanism design class, Ch Mechanism Toolkit provides functions and classes that help students transition between strictly theory oriented coursework and example or application inclusive learning. This toolkit comprises several C++ classes with functions for designing and analyzing several of the most common planar linkages taught in a mechanism design course. Students can use these classes to write small programs, for example, half a dozen lines, which solve the equations of motion for the mechanisms and provide visual feedback in the form of graphs and animation. Beyond simply solving the types of mechanisms included in the toolkit, it is possible to copy and modify the source code for building a new program or class that would analyze different mechanisms.

Ch Language Environment

Ch is a C/C++ interpreter that encompasses ANSI C with extensions to make it more useful in an engineering design environment [7–9]. It can be used to run functions or single lines of C code interactively from the command line. It is an interpreter and therefore runs scripts written in C/C++. Object-oriented programming is also possible with Ch as it supports classes in C++ bringing along many of the advantages that come with it. What really boosts C's capabilities in an engineering environment is its graphical plotting and numerical computing functions. Among other extensions, Ch includes complex numbers and computational arrays as built-in data types. A couple of the benefits they bring are that linear algebra computations, such as array multiplication, no longer require loops, and standard math functions can use these types without special handling. This is only a brief example of many built-in numerical functions in Ch. This makes the process of coding computations in Ch much easier while still allowing the use of the standard language as a whole. A plotting class is also a part of Ch to create visual data output. When combined with computational arrays, getting plots can be as easy as one function call [8].

The fact that Ch is interpretive has two major consequences when dealing with mechanism design and Web-based interfaces. First, programs do not need to be compiled to run. This is especially attractive in a learning environment where students do not want to spend their time in the tedious compiling and debugging cycle. Making changes and seeing the outcome is a much faster process. Second, Ch allows scripting which in turn enables CGI to work effortlessly inside what is essentially a C program [15,16]. Ch is a platform that students can use with a minimum of programming, but still provides a complete set of tools to create useful analysis programs that can be used over the Web [13,17,18].

Mechanism Toolkit

Included in the Ch Mechanism Toolkit are classes that can be used to analyze several types of linkages [10]. These include **CFourbar**, **CCrankSlider**, **CGeared-Fivebar**, **CFourbarSlider**, **CStevSixbarI**, **CStevSixbarIII**, **CWattSixbarI**, **CWattSixbarII**, and **CCam**, for analyzing fourbar, slider-crank, geared fivebar, fourbar-slider, Stephenson sixbar (I & III), Watt sixbar (I & II) linkages, and cam follower systems, respectively. To use the Ch Mechanism Toolkit and examine its open source implementation, students just need some basic knowledge about mechanisms and computer programming. The interface to the toolkit is a set of member functions used to set the geometry and state of the particular mechanism and to perform the analysis desired. Along with the linkage parameters, variables of the appropriate data type are defined and used to return the computational results back to the calling program. From there, numerical or

graphical results can be obtained with a few function calls. There are also open source programs available for kinematic synthesis [19] based on Ch and its Mechanism Toolkit.

Lots of common types of mechanisms can use the above classes as a basis for their analysis. For example, straight-line linkages have many useful applications and quite a few types are just a specific fourbar linkage configuration. Shown in Figure 1 is an example of a Hoekens straight-line linkage. This animation is done using the **CFourbar** class through three function calls. The animation can also be done through a Web browser without any programming. By starting out using the Mechanism Toolkit, studying various types of planar linkages can be done quickly with the added benefit of the visual feedback.

Position, Velocity, and Acceleration Analysis

As with most mechanism design classes, the usual starting point is position analysis. The system can be defined by link lengths and ground link position. Given the angle of the input link, the state of the entire mechanism can be determined. The velocity of the links can be found if the input link angular velocity is defined. Similarly, accelerations for the mechanism can be found.

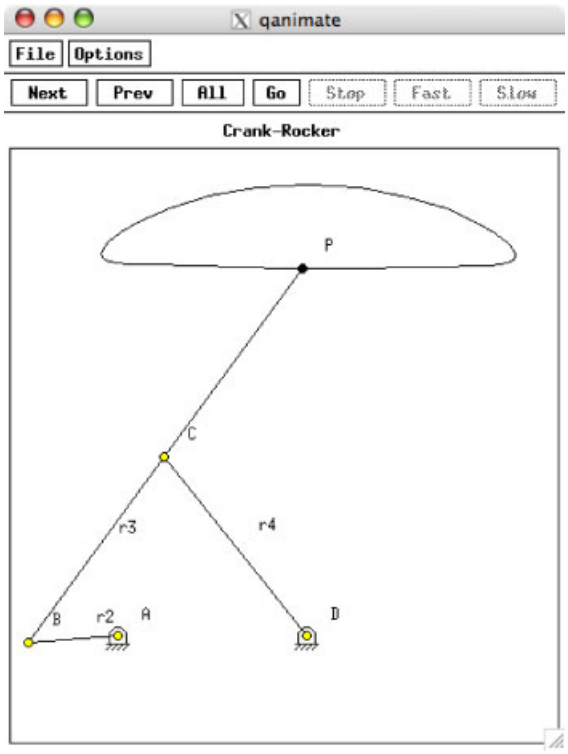

Figure 1 Hoekens straight-line linkage. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

Dynamic Analysis

Even dynamic analysis of these mechanisms is possible with a few function calls. The additional pieces of information needed are the masses, center of gravity locations, and mass moments of inertia. Because these values are all entered as separate numbers and not calculated by assuming the links are all uniform geometric shapes, students can find the forces for a wide variety of link materials and geometries by simply changing these numbers. Building this type of analysis from scratch can be complicated. With this toolkit, students can see complete sets of forces acting between the different links.

Plotting

Built into all of the mechanism classes are functions to plot a coupler curve. With nothing but link length, the path that the coupler point traces can be seen. It becomes relatively quick and easy to see the changes in output by varying the input parameters.

Animation

For students without a lot of experience with different types of linkages, the hardest concept to get across is the different types of motion that the different classes of mechanisms can achieve. In view of this, perhaps the most useful and visually appealing function is the animation of the mechanisms. The animation function uses the link length and calculates the position of all the links at evenly spaced input angles for one whole revolution of the input. This gives students an immediate feel for how these linkages would work far beyond anything a graph could show.

Web-Based Mechanism Design and Analysis

These predefined linkage classes are ready to be used without much programming at all. All that is needed is Ch and its Mechanism Toolkit. With the Ch CGI library and some knowledge of basic HTML, a Web page can be created that pulls data entered by the user into the mechanism functions and shows the output on the Web. To see how these classes work for mechanism analysis, Web pages have already been created for analyzing different mechanisms [10,11,13].

A STUDENT PROJECT ON COMPUTER-AIDED MECHANISM DESIGN AND ANALYSIS

To appreciate how mechanism analysis is done, students can use Ch and its graphical and numerical capabilities to create their own set of functions. The

ability to use object-oriented programming adds the possibility of creating a C++ class that can be accessed from many different user programs for the analysis of a mechanism. The open source Ch Mechanism Toolkit is a starting point, or template, for students to get an idea of the programming structure that can be used. This section demonstrates this type of class for mechanism analysis with a student project in a mechanism design course. In this project, students develop a user-friendly C++ class to analyze a Whitworth quick return mechanism. The program is then extended for Web-based analysis and animation. The project description, a sample student implementation, including code for the C++ class, CGI scripts, and HTML files, as well as the project report are available for download in Reference [20].

Quick Return Mechanism

A Whitworth quick return mechanism allows a large force to be applied by the slider in one direction, while quickly resetting it back to its initial position. The main components, shown in Figure 2, are the ground link (r_1), input link (r_2), input slider (body 3), rocking link (r_3), connector link (r_5), and output slider (body 6). The output slider moves horizontally with varying velocities driven by a constant angular velocity of the input link. Like most mechanisms, this one works to change the direction and force of the motion of the input to provide the desired output.

The first step is to solve the equations that govern the motion of the mechanism. Two loop-closure equations are used to analyze the position with a given angle of the input.

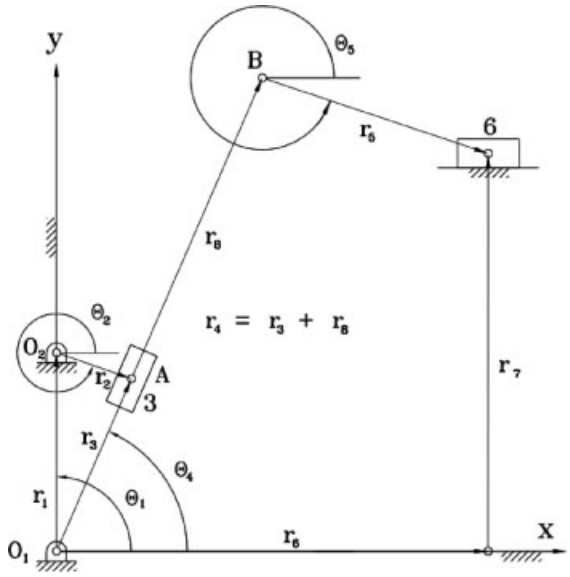

Figure 2 The Whitworth quick return mechanism.

$$\mathbf{r}_1 + \mathbf{r}_2 = \mathbf{r}_3 \tag{1a}$$

$$\mathbf{r}_4 + \mathbf{r}_5 = \mathbf{r}_6 + \mathbf{r}_7 \tag{1b}$$

Using complex numbers, Equation (1) can be represented as follows:

$$r_1 e^{i\theta_1} + r_2 e^{i\theta_2} = r_3 e^{i\theta_4} \tag{2a}$$

$$r_4 e^{i\theta_4} + r_5 e^{i\theta_5} = r_6 e^{i\theta_6} + r_7 e^{i\theta_7} \tag{2b}$$

Equation (2) are then simplified and broken into real and imaginary parts which are finally solved for the unknowns.

In a typical use of this mechanism, the input link would rotate with a constant angular velocity for $\dot{\theta}_2$ and zero angular acceleration for $\ddot{\theta}_2$. With these additional known values, a similar procedure can be used to find velocities by starting with the differentiation of Equation (2). A second differentiation of the resulting equations is then used to find accelerations.

Once the accelerations are known and the mass properties are given, Newton's Laws are used to solve for the forces. For this, a set of equations is derived from each link's free body diagram. The resulting set of 13 equations are solved simultaneously using a matrix of the form $[\mathbf{A}]\mathbf{x} = \mathbf{b}$, where $\mathbf{x}$ is the vector of unknown forces, $\mathbf{b}$ is the vector of gravitational and inertia forces, and $[\mathbf{A}]$ is the 13×13 matrix that relates the two vectors. These equations are quite a bit more complex to solve than the ones for the position, velocity, and acceleration. Thus, the need for a computer in solving these types of mechanism problems is evident. Although students could solve these equations with a variety of methods, repeated computations become quite burdensome and changing a few parameters means starting over at the very beginning. A program, while time consuming initially, will greatly speed the subsequent calculations.

A C++ Class for Analysis of a Quick Return Mechanism

After the governing equations of the mechanism are derived, a class **CQuickReturn** can be created to perform the necessary calculations. The major components are reviewed below to give an overview of how the class is structured. First, private data members are used to store all the variables used in the analysis. These include the parameters provided by the user that define the mechanism as well as the results from the calculations in the program. One of the major advantages of object-oriented programming is that all the data, whether entered or calculated, is available to all the other member

functions without the complication of passing the variables each time a function is called. Private data is hidden from the user so they are not burdened with knowing which piece of information each function needs when called.

The member functions that perform the actual calculations are also private members of the class. They are called as necessary by the public member functions. This is where the numerical capabilities of Ch help greatly. The position, velocity, and acceleration calculations are performed by using complex numbers. The force calculations, as they are solved through linear algebra, are greatly simplified. Once all the variables are assigned to the matrices, it is only one function call in the program to solve for $\mathbf{x} = [\mathbf{A}]^{-1}\mathbf{b}$.

The user interface to the class is through public member functions. All the parameters of the mechanism are provided through member functions. Link geometries, mass properties, and input angle and velocity, among others, are entered as necessary for the desired analysis. This is another advantage of the object-oriented programming. Not all the calculations have to be performed and only the necessary

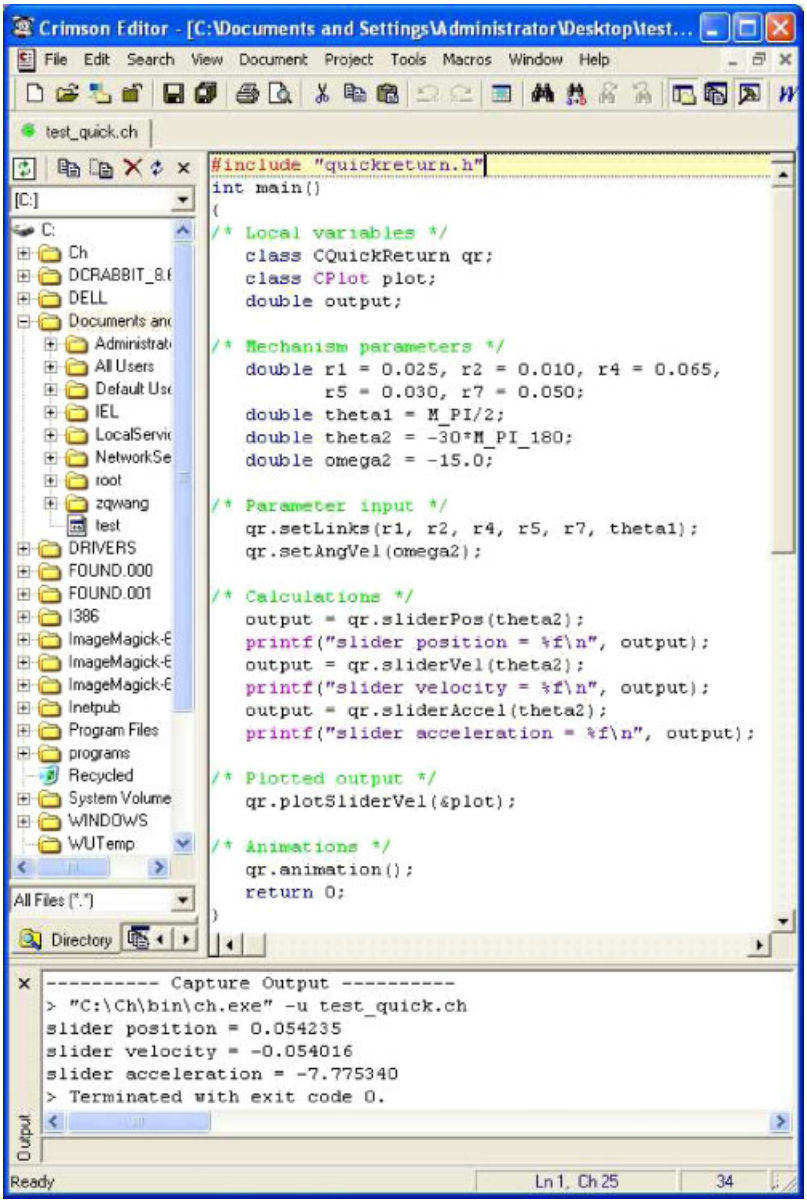

Figure 3 An application program using the **CQuickReturn** class within a Crimson Editor. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

information needs to be given. Output is also handled through public member functions. The desired information can be called for without needing to see the results of all the previous calculations.

An example program is shown in Figure 3 that uses the developed quick return mechanism class **CQuickReturn**. It is presented within Crimson Editor, an Integrated Development Environment (IDE) that works with Ch. In the first section of the program, labeled /* Local variables */, three local variables are defined for use in the program. The first one defines the class used for the actual mechanism analysis, which contains all the user created code for solving the equations of motion for the system. The next one defines the plotting class used to create visual output. The last variable here is used for passing the output of the calculation functions back to the program. The second section, labeled /* Mechanism parameters */, defines all the necessary mechanism parameters used in the analysis. This includes: link lengths ($\mathbf{r}_{1,2,4,5,7}$), ground link angle (θ_1), input link angle (θ_2), and input link angular velocity (ω_2). All angles are calculated in radians internally. θ_2 is therefore multiplied by a conversion factor to keep all

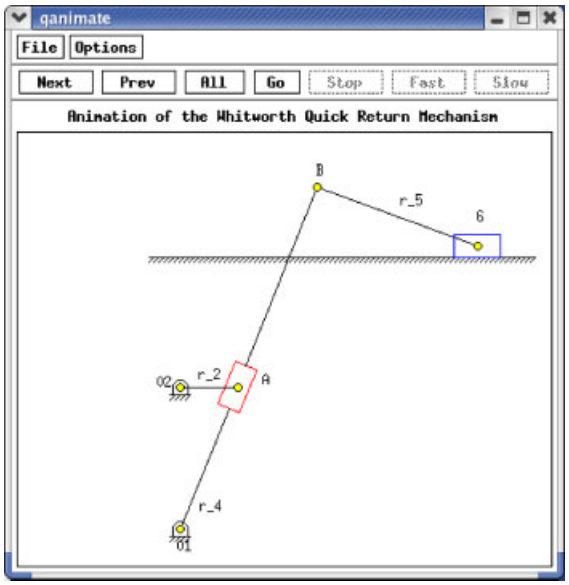

Figure 5 An animation snapshot of a quick return mechanism. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

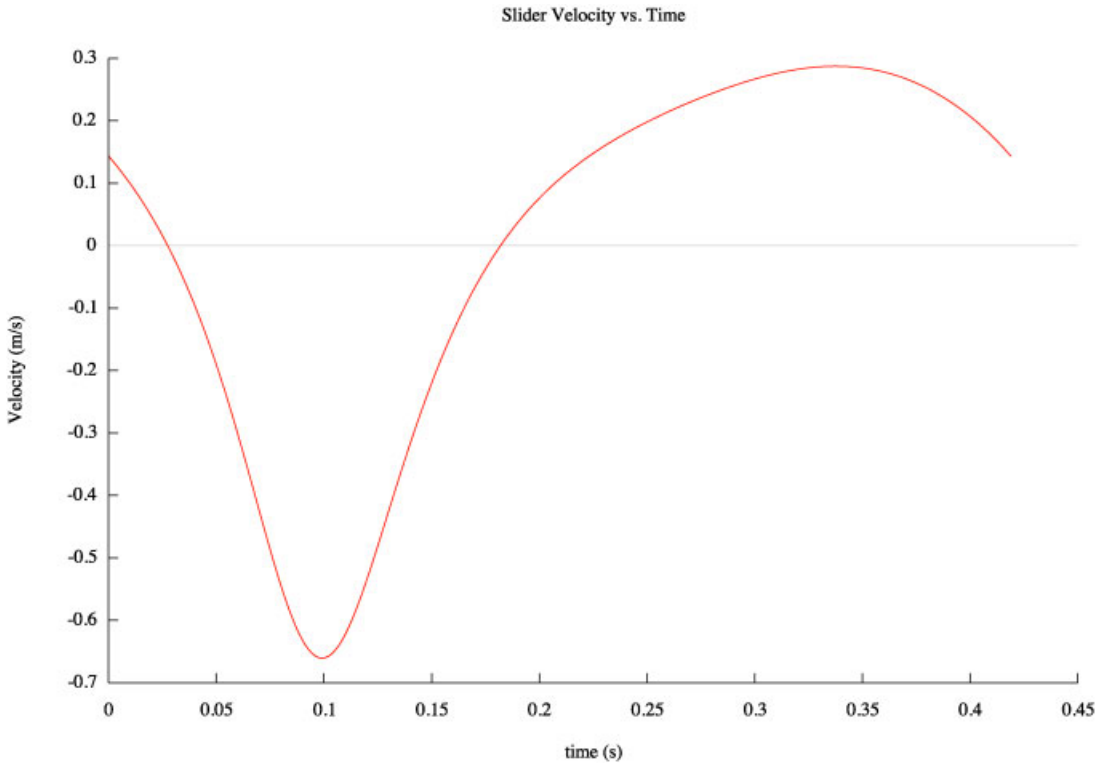

Figure 4 The velocity of the slider as plotted by **CQuickReturn** class. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

angles in the same units. The section labeled /* Parameter input */ calls the public member functions **setLinks** and **setAngVel** to enter the constant mechanism parameters into the private data members. The calculations section calls several member functions that take in the value of the input link angle, calculate all necessary parameters, and return the slider position, velocity, and acceleration which

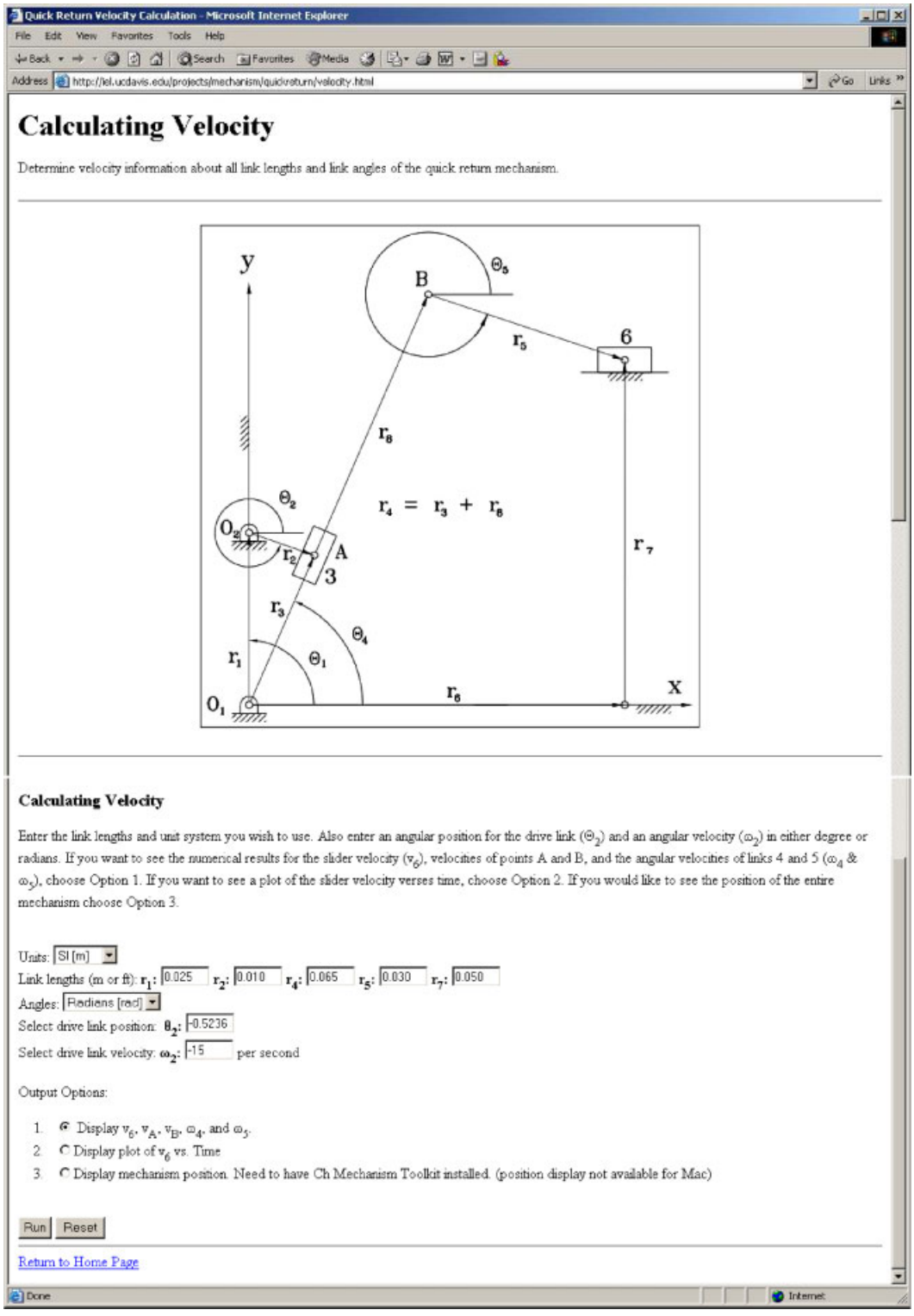

Figure 6 The Web interface for velocity calculation of a quick return mechanism.

are displayed after each function call. The last two sections plot a graph of the output slider velocity verses time for the given input angular velocity and display an animation of the system in motion.

The numerical output from the program is in the lower window of Figure 3. The plot of the output slider's velocity is shown in Figure 4. The animation of the mechanism created by the **CQuickReturn** class is shown in Figure 5. Only 34 lines of code are necessary to get numerical and two kinds of graphical output. This is a simple example that shows only some of the capabilities of the developed class. More complicated analysis is possible by defining additional mechanism parameters and calling other analysis functions. The source code used for the above example, including the **CQuickReturn** class and the HTML and CGI code described in the next section, is available to download [20].

Web-Based Interface to the Mechanism Class

As mentioned earlier, once the **CQuickReturn** class is developed, it can be interfaced from Web browsers [20]. The information is passed through the use of a simple fill-out form. As an example, the Web page for calculating velocities is shown in Figure 6.

The interface of the Web page is kept simple, just showing the necessary information needed for the analysis. To make the form less cluttered, different analyses are placed on different pages. Therefore, to calculate position, one would go to the first page; to find velocities, one would go to another; etc. It is not necessary to visit any particular page, but only to go to the desired page.

A CGI program is created that receives the data input via a Web browser, calls the member functions

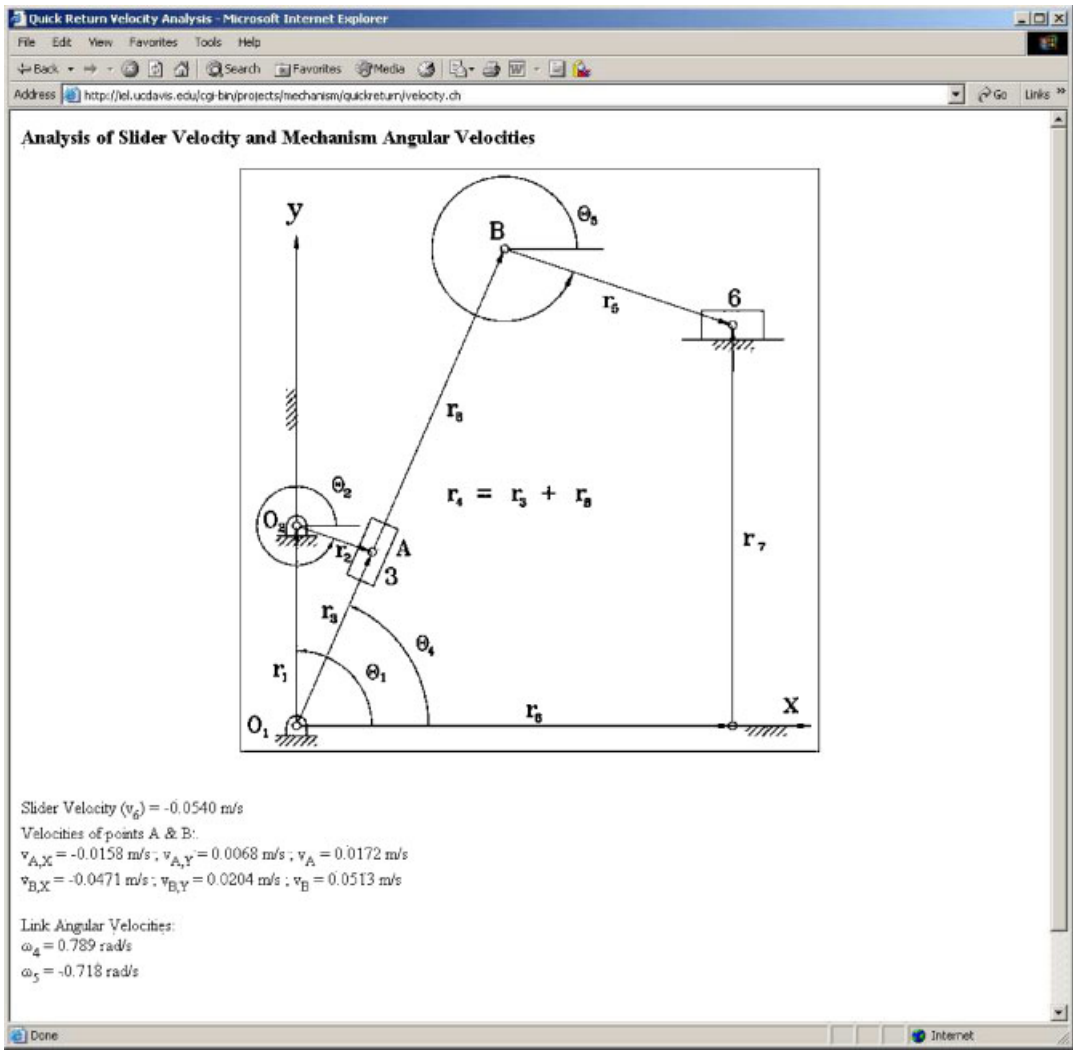

Figure 7 The results from the velocity calculation of a quick return mechanism.

of **CQuickReturn** to perform the calculations, and formats the output for display. The Ch CGI code includes the classes **CRequest** and **CResponse** which are similar to those in JavaServer Pages (JSP) and Active Server Pages (ASP). The member function **getForm()** is used to extract data from an HTML file, and the function **streval()** obtains a numerical value from a string passed from an HTML file. Figure 7 displays the output of the velocity analysis. In addition to numerical data, the results can be displayed in the form of a graph similar to Figure 4, or as an animation as shown in Figure 8.

The interface with fill-out forms in Ch is much simpler in comparison with other mathematical software packages. For example, MATLAB, does have functions for retrieving data from fill-out forms. Several structures need to be defined to aid in the data extraction [21]. The interface to the **CQuickReturn** class is an HTML file and one Ch CGI program.

Students who have to develop a program to run a mechanism analysis should have no problem picking up a few extra functions to interact through the Web. Output in the sample code was also kept to a minimum, just providing the pertinent feedback for the mechanism analysis. Therefore, once students get the basic concept for this type of Web-based user interface, the capabilities could be extended.

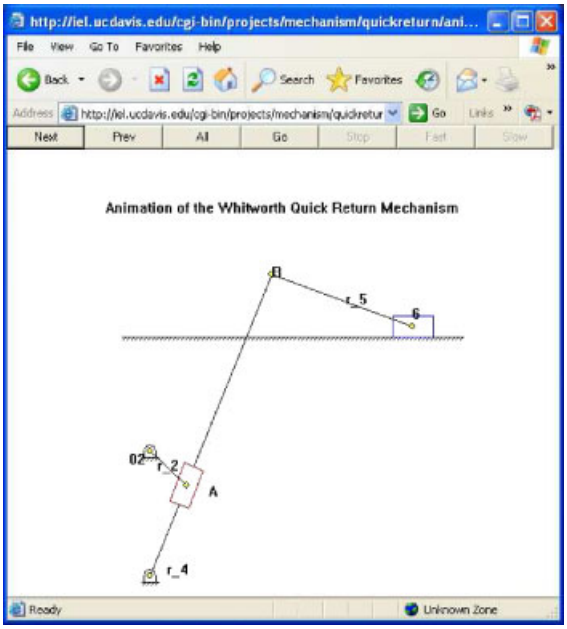

Figure 8 The quick return mechanism animation seen through a Web browser. [Color figure can be viewed in the online issue, which is available at www.interscience.wiley.com.]

CONCLUSION

A pedagogically effective teaching methodology for computer-integrated mechanism analysis and design has been presented in this article. As a superset of C, Ch with classes in C++ has high-level extensions for graphical plotting, numerical computing, and script computing. Without tedious compilation and linking, this C/C++ interpreter is ideal for teaching and learning subjects in engineering, especially for interactive presentation in a class. The Ch Mechanism Toolkit contains classes for analysis and design of commonly used mechanisms. High-level member functions in these classes can be used conveniently for kinematic and dynamical analysis as well as animation of mechanisms. Using QuickAnimation in this toolkit, animations of other mechanisms can be developed. By examining the source code of these classes, students can develop their own classes to design and analyze other mechanisms. The system can also be used for Web-based computing in mechanism design and analysis. Ch and Ch Mechanism Toolkit have been used for many years as an effective teaching tool in an undergraduate course *Computer-Aided Mechanism Design* taught at the University of California, Davis as well as by many other universities. In our teaching, mechanism design is interwoven with object-oriented engineering software design. A sample student project of designing a software package for the analysis of a Whitworth quick return mechanism has been presented in the article. The feedback from students is very positive. They enjoy using object-oriented programming to solve mechanism design problems, especially Web-based mechanism design and analysis. Both the principle of teaching and problem solving skills students learned are applicable to a wide range of engineering subjects.

ACKNOWLEDGMENTS

This work was supported in part by the University of California, Davis through an undergraduate instructional improvement grant and an educational grant from Intel Corporation.

REFERENCES

[1] MSC Software, http://www.mscsoftware.com/.
[2] A. Perez, H. J. Su, and M. McCarthy, SYNTHETICA 2.0:Software for the synthesis of constrained serial chains, In Proceedings ASME Design Engineering

Technical Conferences, No. DETC2004/57524, Salt Lake City, Utah, September (2004).
[3] WATT 1.6 User's Guide, Heron Technologies (2002). http://www.heron-technologies.com.
[4] SAM 5.0 User's Guide, Artas—Engineering Software (2003). http://www.artas.nl.
[5] T.-T. Fu, Applications of computer simulation in mechanism teaching, Comput Appl Eng Educ 11 (2003), 156−165.
[6] H. H. Cheng, Pedagogically effective programming environment for teaching mechanism design, Comput Appl Eng Educ 2 (1994), 23−39.
[7] H. H. Cheng, Scientific computing in the Ch programming language, Sci Program 2 (1993), 49−75.
[8] H. H. Cheng, Extending C and FORTRAN for design automation, ASME Trans J Mech Des 117 (1995), 390−395.
[9] Ch—An Embeddable C/C++ interpreter, http://www.softintegration.com.
[10] Ch Mechanism Toolkit, Softintegration, Inc. http://www.softintegration.com/products/toolkit/mechanism/.
[11] J. Larson and H. H. Cheng, Object-oriented cam design through the Internet, Intell Manuf 11 (2000), 515−534.
[12] H. H. Cheng and D. Trang, Object-oriented interactive mechanism design and analysis, Eng Comput 21 (2006), 237−246.
[13] H. H. Cheng and D. Trang, Web-based interactive analysis and animation of mechanisms, ASME Trans J Comput Inform Sci Eng 6 (2006), 84−90.
[14] The Common Gateway Interface, MCSA Software Development Group (1996).
[15] Ch CGI Toolkit, SoftIntegration, Inc. http://www.softintegration.com/products/toolkit/cgi/
[16] H. H. Cheng, CGI Programming in C, C/C++ Users J 14(11) (1996), 17−21.
[17] Y. Zhu, B. Chen, and H. H. Cheng, An object-based software package for interactive control system design and analysis, ASME Trans J Comput Inform Sci Eng 3 (2003), 366−371.
[18] Q. Yu, B. Chen, and H. H. Cheng, Web-based control system design and analysis, IEEE Control Syst Mag 24 (2004), 366−367.
[19] E. Pennestri, Kinematic Synthesis of Mechanisms, http://www.ingegneriameccanica.org/mechanisms.htm.
[20] Design and analysis of Whitworth quick return mechanism http://iel.ucdavis.edu/projects/mechanism/quickreturn.
[21] P. S. Shiakolas, V. Chandra, and J. Kebrle, Environment for engineering design, analysis, and simulation for education using MATLAB via the World Wide Web, Comput Appl Eng Educ 10 (2002), 99−120.

BIOGRAPHIES

Matt Campbell graduated with an MS in mechanical engineering from the University of California, Davis. His BS in mechanical engineering came from California State University, Chico. While in Davis he conducted research in the Integration Engineering Laboratory in the Department of Mechanical and Aeronautical Engineering. Currently his research includes the integration of hardware and software systems, sensor fusion in distributed systems, and mobile computing.

Harry H. Cheng is a professor and director of the Integration Engineering Laboratory at the University of California, Davis. He received the PhD degree in mechanical engineering and the MS degree in mathematics from the University of Illinois at Chicago in 1989 and 1986, respectively. He is the chief architect of an embeddable C/C++ interpreter Ch for script computing, which is being widely used in both academia and industry. He holds one US patent and has published over 120 articles in refereed journals and conference proceedings. He is a fellow of ASME and a senior member of IEEE, IEEE Computer Society, and IEEE Robotics and Automation Society. His current research interests include mobile multiagent systems, information-driven mechatronic systems, computer-aided design and manufacturing, and intelligent transportation systems.