E-learning in Process Control Education
Proceedings of European Congress of Chemical Engineering (ECCE-6)
Copenhagen, 16-20 September 2007

# E-learning in Process Control Education

M. Bakošová, M. Fikar, Ľ. Čirka

*Department of Information Engineering and Process Control, Institute of Information Engineering, Automation and Mathematics, Faculty of Chemical and Food Technology, Slovak University of Technology, Radlinského 9, SK-812 37 Bratislava, Slovakia*

## Abstract

The paper presents some new features that have been introduced in the last years to the course Automatic Control Fundamentals at the FCFT STU in Bratislava. These involve intensive use of information and communication technologies and reduction of repetitive pedagogic work using e-learning techniques. The work also shows how to employ MATLAB as a scripting engine coupled with XML technology to generate multiple outputs using single input source.

Keywords: e-learning, LMS, education, process control, information technologies

## 1. Introduction

In the last two decades computer technology revolutionized the world of computing and teaching. Tools are continually being developed to explore various ways that improve teaching and solve challenges that were unsolvable before. One of the challenges is to deal with a large number of students. If they have the same quiz or assignment they tend to cheat. However, it is reasonable to give them quizzes of the same complexity so that some of them are not in advantage.

In this paper, we discuss creation and maintenance of large sets of questions for e-learning course on Automatic Control Fundamentals. This includes generation of on-line and off-line quizzes for the course with approximately 250 students. The approach taken includes MATLAB as a scripting language combined with XML templates to produce input files. These can be further processed using XSLT to obtain HTML representation, input to LMS system Moodle, or various versions of PDF files produced by LATEX.

## 2. Automatic Control Fundamentals

The course Automatic Control Fundamentals is compulsory for all undergraduate students at the FCFT STU in Bratislava. There are altogether 13 lectures (two hours each) and 13 exercises in multiple groups (two hours each), where students have 10 problems to solve.

The new self-learning package covering all topics of the course has been established on the Internet. The idea is that the net course will in the future substitute the presentation form of lectures and exercises, so that the whole course could be studied totally as a self-study. Although the course is still lectured in the normal way, students are encouraged to master the exercises individually.

The Internet suite has several components:

- study materials, files for download, information about the course,
- basic operations (as described in Section 3),
- Internet version of all course topic problems,
- on-line tests and preparation for written tests,
- Moodle e-learning portal for grade books, attendances, quizzes, etc.

All this can be found in a freely accessible web page at http://www.kirp.chtf.stuba.sk/lcza (in Slovak).

The textbook by Bakošová et al. (2003) is used in the course. Several versions are available: paper, interactive PDF, and lecture notes PDF. The main topics of the course are modeling of chemical processes, closed-loop, stability, PID controller design and control of chemical processes. MATLAB and Simulink are used as a programming and simulation environment.  Virtual animated processes are used in simulations as in Fig. 1 and their number will increase in the future.

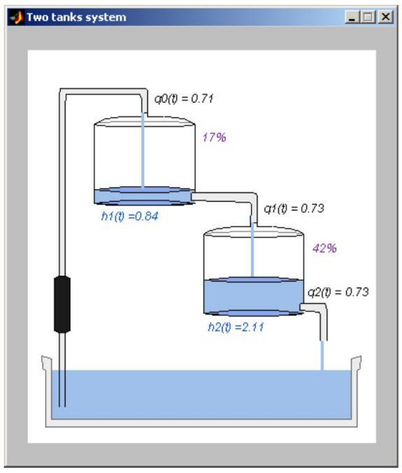


Figure 1: Animated storage tanks in MATLAB

## 3. MATLAB Web Server

Almost all tasks in the course are based on MATLAB/Simulink. During the laboratory course the students can use PC with MATLAB where the tasks can be solved. To enable the students to use the same or similar working environment individually on their home computers, or even using an Internet connection, we use the MATLAB Web Server (MWS). MWS is a cgi-bin application that is used to launch MATLAB script files (m-files) remotely. The user can give parameters that are used during the m-file execution. With MWS, almost all built-in and self-written MATLAB functions can be called from the m-file, including functions that produce plots to be shown to the user. We have implemented a series of relatively self-contained modules. It is possible to use these modules also in other subjects taught. Modules can be changed easily with a minimal cost. Further, we develop our modules using fairly standard MATLAB features that do not change between versions. MWS component is not developed anymore by Mathworks. Therefore we keep MATLAB R2006a on the web server.

The following set of automatic control related problems can be solved:
- polynomial mathematical operations,
- polynomial roots finding,
- solution of a matrix equation $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$,
- pole-zero map of LTI models,
- step response of LTI models,
- process model simulations (storage tanks and heat exchangers),
- closed-loop simulations.

## 4. E-learning

Starting with the academic year 2004/05, it has been decided that each student should have an individual assignment and that all students have assignments of a similar difficulty. With the number of students about 250 this quickly becomes impossible to manage from the teachers point of view. Also the evaluation of students varied among the different teachers. Therefore, we have decided to build up an Internet e-learning module and implemented Open Source LMS project Moodle (Dougiamas, 2005). The basic functionality of Internet Suite is as follows.
- Students are organized on groups.
- Each student can see only his/her assignments and evaluation during the course.
- Teachers can evaluate, see, and modify evaluation of the students.
- Teachers can also login as students.
- There are several discussion boards available: announcement (only teachers can write), general (any course participant can write or respond) and teachers forum (not seen by students). Discussions can be limited to a group or to all participants.
- Unified evaluation of students. Each laboratory exercise is evaluated with 20% for entrance test and 80% for the final report where time delay penalization system is introduced.

- User authorization. Each person involved in the course has unique login, password and role (student, teacher, administrator).

## 5. Assignments

The second part of the module is scientific and deals with assignment generation for all students. Types of assignments are fixed and given in the course book. We have then implemented a series of MATLAB scripts and functions that generate random numbers for a particular problem, solve it, and generate the result as a HTML file suitable to be included in the first part of the module. To explain in some more details, the procedure consists of the following parts:

1. A function that generates random input values for one student.
2. A function that takes the assignment and calculates not only its solution but also some intermediary results to aid the teacher with the evaluation of the student.
3. A script that calls both the preceding functions for the total number of students and stores the data in a file.
4. A script that reads the input file from the preceding step and outputs two series of HTML files – one for students and one for teachers. The latter contains the same information as for the student as well as the solution.

This division into independent modules has several advantages:

- Modules can be tested independently.
- If there is an error discovered in the middle of the semester within the assignments or solutions, the input data files assure that new HTML files can easily be generated with the same students data.

HTML Files are then uploaded to server with LMS Moodle that contains a special module that handles individual assignments. Teacher's files with solution are then protected using double authorization procedure:

- based on IP numbers – teacher can access the solution only at the departmental network that is secured using firewalls,
- Moodle user authentification module.

Although this solution works well in practice, its implementation is complicated. Even if a series of functions with different purposes is programmed, it is inevitable that program logic is mixed with output formatting – in this case HTML representation. If the output format should have been changed, one should carefully study the MATLAB source code to find out the necessary changes. Therefore, we have looked at some alternative ways to describe desired tasks with format neutral representation so that inputs can easily be reused and changed. A new procedure has been found based on XML representation and has been implemented in quizzes.

## 6. Quizzes

Examination of the course is divided into written and oral parts. The written part consists of a series of computational exercises. The questions can either be multiple-choice questions with five choices where each student marks one correct choice or short answers where no choices are given. The advantage of the first approach is that quizzes can be graded automatically. However, students are sometimes able to make a qualified guess only by looking at the answers or even find the correct answer by reverse engineering. On the other hand, short answer type of quizzes has to be graded manually, but make the quiz a little harder. Students can use any printed references (books, tables, etc).

Preparation of the written examination has been automated. It is using MATLAB as the computational engine, XML/XSLT as the transformation engine, and LATEX, HTML, and Moodle as formatting tools. Currently, there are about 50 different problem choices with different levels of difficulty that can be given to students and that cover the whole range of the course. In the first run, MATLAB generates a random combination of them so that the required total number of points is achieved. Also problems with similar topics are grouped together and always only one of them can be in the quiz. Next, MATLAB generates random values for each problem and outputs them in XML representation. This is repeated for a given number of groups so that students cannot just copy the results of their neighbors. Thus each group has the same type of problems but with different values. Finally, XML file is transformed into a final form. This can include printed version, html version, or a Moodle quiz. The whole procedure will now be explained on a simple example.

## 7. Example for a quiz generation

Let us consider this simple example problem together with its solution.

### *7.1. Problem Definition*

The closed loop system consists of a controlled system with transfer function of the form $G(s) = \frac{b_0}{s^2 + a_1 s + a0}$ and a PID controller of the form $G_c(s) = P + \frac{I}{s} + Ds$. If the setpoint value is changed at $t$ = 0 from 0 to $w$, the permanent tracking error is given as:

$$e(\infty) = \begin{cases} w\left(1 - \frac{b_0 P}{a_0 + b_0 P}\right) & \text{if } I = 0 \\ 0 & \text{otherwise} \end{cases} \qquad (1)$$

The problem will be in sequel denoted by acronym TRO.

*7.2. Template XML File*

Each problem consists of a template file where items that are to be changed are marked with **`##ID##`** where ID identifies name of the particular variable part. The template file **`tro.tpl`** for our problem is as follows

```
<problem name="tro">
<subproblem score="3" shortans="yes">
<problemtext>
The closed loop system consists of a
controlled system with transfer function
of the form <span CLASS="math">
G_p(s) = \frac{##b0##}{s^2 +
  ##a1## s + ##a0##}</span>
and a controller of the form
<span class="math">##gr##</span>.
If the setpoint value is changed at
<span CLASS="smath">t=0</span>
from 0 to ##w##, the permanent tracking
error is given as
</problemtext>
<answers>
<choice order="##1##" ans="gooditem">
  ##tross##</choice>
<choice order="##2##" ans="baditem">
  ##VAL2##</choice>
<choice order="##3##" ans="baditem">
  ##VAL3##</choice>
<choice order="##4##" ans="baditem">
  ##VAL4##</choice>
<choice order="##5##" ans="baditem">
  no other choice is correct</choice>
</answers>
</subproblem>
</problem>
```

*7.3. XML File Creation*

The first step is to generate some random and admissible data. MATLAB is used for this task. Each problem has a function associated with it that prepares data that are to be changed each time. A typical function consists of three parts.

1. Generation of data. This is demonstrated below and closely follows mathematic problem definition.

```
function [str,template, valstr] = tro
w=fix(rand(1,1)*10)+1;
num=fix(rand(1,1)*10)+1; b0=num(1,1);
den=fix(rand(1,2)*10)+1;
a1 = den(1,2); a0 = den(1,1);
```

```
xx=fix(rand(1,3)*10)+1; pp=xx(1,1);
ii=xx(1,2);dd=xx(1,3);
if (randn(1,1)<0)
  ii=0;
  tros = w*(1 - (b0 * pp)/(a0+b0*pp));
else
  tros = 0;
end
```

2. Generation of false data, preparation of parameters for the template file. In our example, we need to be sure that any of other three false answers is not the same as the true one. In addition, if the permanent tracking error is not zero, we give one of the false answers equal to zero. Finally, the transfer function of the controller is prepared to take care between PID and PD cases.

```
tross = sprintf('%.2f',tros);
choic = randn(1,4);
choi = sprintf('%.2f',choic);
while sum(findstr(choi,tross))>0
  choic = randn(1,4);
  choi = sprintf('%.2f',choic);
end
ss2 = sprintf('%.2f',choic(2));
ss3 = sprintf('%.2f',choic(3));
ss4 = sprintf('%.2f',choic(4));
if tros ~=0
   ss2 = '0.00';
end
if ii~=0
   [gr,e]=sprintf('G_R(s) =
   %.0f %+.0f s %+.0f/s ',pp,dd,ii);
 else
   [gr,e]=sprintf('G_R(s) =
   %.0f %+.0f s ',pp,dd);
end
```

3. Finally, values for the template file have to be specified. At first, they are converted to strings. These are then substituted to the template.

```
valstr= {'b0', int2str(b0)
        'a1', int2str(a1)
        'a0', int2str(a0)
        'gr', gr
        'w', int2str(w)
        'tross', tross
        'VAL2', ss2
        'VAL3', ss3
        'VAL4', ss4};
template = 'tro.tpl';
str = writetpl(template, valstr);
```

The result is returned in string **`str`** to the calling routine. In our case the result can be as follows.

```
<problem name="tro">
<subproblem score="3" shortans="yes">
<problemtext>
The closed loop system consists of a
controlled system with transfer function
of the form <span CLASS="math">
G_p(s) = \frac{3}{s^2 + 5 s + 7}</span>
and a controller of the form
<span class="math">G_R(s) =9+5s</span>.
If the setpoint value is changed at
<span CLASS="smath">t=0</span>
from 0 to 8, the permanent tracking
error is given as
</problemtext>
<answers>
<choice order="4" ans="gooditem">
  2.06</choice>
<choice order="2" ans="baditem">
  -1.15</choice>
<choice order="5" ans="baditem">
  0.13</choice>
<choice order="1" ans="baditem">
  0.29</choice>
<choice order="3" ans="baditem">
  no other choice is correct</choice>
</answers>
</subproblem>
</problem>
```

As we can see, the problem has been transformed into a multiple-choice question with one correct answer. MATLAB has provided suitable random values and results.

## *7.4. XML Structure*

The XML structure for one problem is designed in such a way that either multiple-choice or short answer questions can be produced.

The main element is **`<problem>`** with attribute name defining the name of the problem. Each problem can have one or more tags of type **`<subproblem>`**. Its attributes define score and flag **`shortans`** that specifies whether it is possible to generate short answer type of question. Subproblem consists of two parts: **`<problemtext>`** and **`<answer>`**. Finally, answers contain five times **`<choice>`** where its attributes define whether the choice is correct (**`gooditem`**) and what is its position after XSLT.

Problem text can contain one image of the form **`<img src='file'/>`** that should exist in the current directory.
Mathematics can be written in LATEX syntax enclosed in tags **`<span>`** with attributes either **`class='math'`** or **`class='smath'`**. The second one can be used for formulas that can be in HTML representation typeset without special mathematical formulae (for example t=4 versus $t = 4$ ).

## *7.5. Function* **`writetpl`**

This function called from each problem is responsible for several subtasks:

- Randomize the order of answers. This is important as some of the presentation formats cannot shuffle the answers. However, as template files have correct answer on a fixed place that cannot be moved, each answer becomes a random number in the attribute order. Then XSLT routine will be responsible to process the answers in sorting order, thus actually randomizing them. For this purpose, placeholders **`##1##`** to **`##5##`** are reserved for **`writetpl`** and cannot be used elsewhere.
- It tests whether the designer has filled out all templates and that all are necessary - there must be one-to-one relationship between MATLAB and template files.

## *7.6. XML File*

All problem functions return a string that holds the whole problem in XML form. However, all problems need to be put together and a valid XML file needs to be produced. This can again be done within MATLAB. Consider for example a script that produces 20 different questions of type **`tro`**:

```
n=20;
str=cell(n);
for i=1:n
  str{i}='tro';
end
writemain('quiz.xml', str);
```

The script uses function **`writemain`** and outputs a file **`quiz.xml`**. The function **`writemain`** includes proper XML header and a root element **`<quiz>`** and calls repeatedly all functions that have been requested.

## *7.7. Transformation to Presentation Formats*

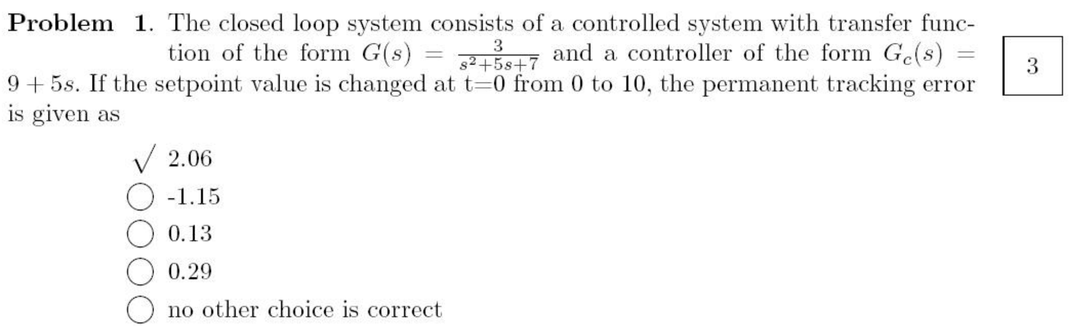
Further transformation of format independent XML file can be performed using standard XSLT engines (e.g. Kay (2006), Clark (2005), etc.) or built-in function **`xslt`** in MATLAB. See Fikar (2007) for more details. In our case, three output formats have been specified:

- PDF – to be used in written examination. It should exist in versions for teacher and student.
- HTML – to be shown on a usual web page, together with a hint to a correct answer.
- Moodle – to be used in e-learning environment with automatic links to students grade books.

These three output formats correspond to different situations: students can access some of the questions freely at the web pages together with a solution. There, they can test the knowledge without any stressing conditions. The Moodle output can be used at the beginning of seminars to check the students. And finally, paper output is used in written part of examinations. As all questions originate from a single source pool, students get familiar with them, and the final written examination results have been greatly improved. Two output format variations are possible: either the default multiple choice or a short answer type of question. In the latter case, incorrect answers in XML file are simply ignored and student gets only a placeholder to write the answer (whereas teacher only the correct answer). However, some types of problems do not make it possible to have short answer type of question. Therefore, each subproblem contains an attribute shortans with value yes or no that specifies whether short answer type of question or the default multiple-choice type will be generated in case that request for short answer has been made. The paper (PDF) version uses LATEX as it can create high quality PDF files from plain ASCII input and thus can act as a filter without user intervention. This would be very difficult if not impossible to achieve with WYSIWYG programs (MS Word, etc). The formatting engine LATEX is invoked with modified class exams (Van der Meer, 2002) that can in two runs produce two PDF files – one for students and one for the evaluator where the correct answers are shown. The resulting part of the PDF available to teacher for multiple-choice type of question is shown in Fig. 2.

One of the major problems with typesetting of output formats is mathematics on the web. Although the MathML standard exists, it is still not implemented in all web browsers. Therefore, the LATEX notation is used and transformed either using MimeTeX (creates pictures of mathematical objects, Forkosh (2006)) or jsMath (uses javascript engine, Cervone (2006)). Both approaches have some advantages and drawbacks but serve well their purpose. The purpose of plain HTML format is to produce a free pool of quiz questions that serve for preparation before the actual exam. These are published on the Internet site of the course. Here, no evaluation is performed. The questions and responses are coupled with a javascript so that student can click on a response to be sure that his/her response is correct. An example of this solution coupled with jsMath to typeset mathematic expression is shown in Fig. 3.

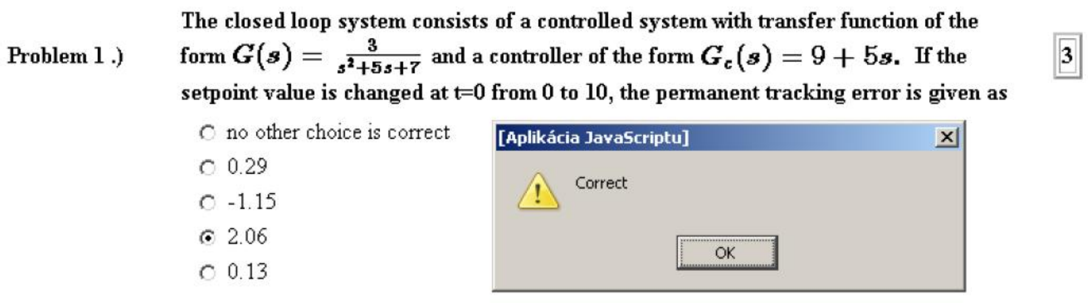
The last output format is needed for LMS Moodle. The idea is to prepare a large number of similar questions that could be used in Moodle quizzes in a random way – so that every student gets possibly different questions. Quizzes are used in each week to test preparation of students for the course. Moodle provides several ways of importing questions from a text file. These include WebCT, GIFT, etc. As import files are plain ASCII, it is not very difficult to create XSLT template that transforms the original XML file to the desired format. We have chosen Moodle XML import and our problem in Moodle is shown in Fig. 4. Here, the standard mathematical tool is MimeTeX. However, JsMath support can be installed as well.



**Problem 1.** The closed loop system consists of a controlled system with transfer function of the form $G(s) = \frac{3}{s^2+5s+7}$ and a controller of the form $G_c(s) = 9 + 5s$. If the setpoint value is changed at t=0 from 0 to 10, the permanent tracking error is given as $\boxed{3}$

- [✓ 2.06]
- ○ -1.15
- ○ 0.13
- ○ 0.29
- ○ no other choice is correct

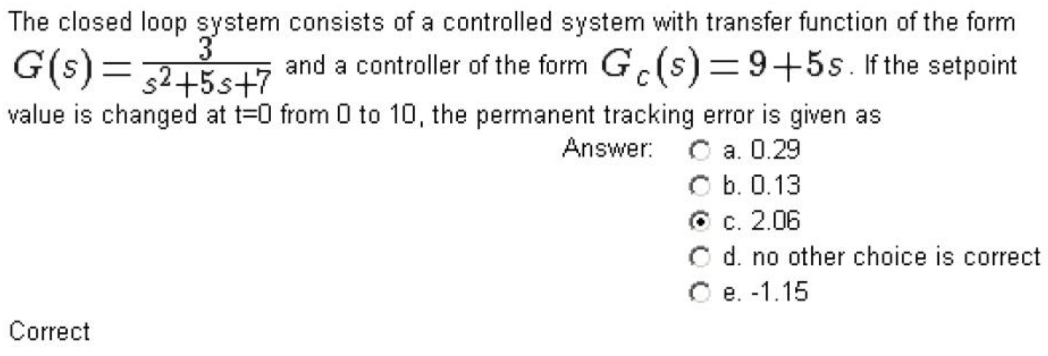
Figure 2: Example of a PDF file generated by LATEX that shows a multiple-choice type question with answer for a teacher



**Problem 1 .)** **The closed loop system consists of a controlled system with transfer function of the form $G(s) = \frac{3}{s^2+5s+7}$ and a controller of the form $G_c(s) = 9 + 5s$. If the setpoint value is changed at t=0 from 0 to 10, the permanent tracking error is given as** $\boxed{3}$

- ○ no other choice is correct
- ○ 0.29
- ○ -1.15
- [✓ 2.06]
- ○ 0.13

Figure 3: Example of a HTML file with jsMath



The closed loop system consists of a controlled system with transfer function of the form $G(s) = \frac{3}{s^2+5s+7}$ and a controller of the form $G_c(s) = 9+5s$. If the setpoint value is changed at t=0 from 0 to 10, the permanent tracking error is given as

Answer:
- ○ a. 0.29
- ○ b. 0.13
- [✓ c. 2.06]
- ○ d. no other choice is correct
- ○ e. -1.15

Correct

Figure 4: Example of Moodle question with MimeTeX

## 8. Conclusions

This paper has shown some new features in the course Automatic Control Fundamentals that have been developed in the last few years at FCFT STU in Bratislava. The changes were introduced in order to bring the course in line with current trends in information technologies and to increase its rating and attraction among students. These aims have been fulfilled. The developed solutions have shown that information technologies can reduce and sometimes even remove the repeating tasks from human and transfer them to computer. Moreover, it increases the productivity significantly. On the other side, the amount of work and time spent have been significant, it required understanding of many new ideas and technologies, and continuing learning of teachers. We have described a procedure of creating quizzes in output neutral XML representation. This has advantages and has not only greatly simplified creation of new questions but also made it possible to generate quizzes in many different ways. In the future, we will increase the portion of virtual processes and their control and introduce on-line access to remote laboratory of real processes.

## 9. Acknowledgements

The authors are pleased to acknowledge the financial support of the Cultural and Educational Grant Agency KEGA of the Slovak Republic under grant No. 3/3121/05 and Scientific Grant Agency VEGA of the Slovak Republic under grants No. 1/3081/06 and 1/4055/07.

## References

Bakošová, M., Fikar, M. and Čirka, Ľ., *Automatic Control Fundamentals. Laboratory Exercises of Automatic Control Fundamentals* (in Slovak), STU Press Bratislava, Slovakia (2003).

Dougiamas, M., *Moodle - a free, open source course management system for online learning*, http://moodle.org (2005).

Kay, M., *Saxon – the XSLT and XQuery processor*, http://saxon.sourceforge.net/ (2006).

Clark, J., *XT – XSLT processor*, http://jclark.com/xml/xt.html (2005).

Fikar, M., *On automatic generation of quizzes using MATLAB and XML in control engineering education*, Technical Report, IIEAM FCFT STU in Bratislava, Slovakia (2007).

Van der Meer, H., *The exams package for LaTeX*, http://www.ctan.org (2002).

Forkosh, J., *MimeTeX: embedded LaTeX equations in HTML pages*, http://www.forkosh.com/mimetex.html (2006).

Cervone, D., *jsMath: A method of including mathematicsin web pages*, http://www.math.union.edu/~dpvc/jsMath/ (2006)