

Using WIMS for Mathematical Education

Xiao Gang and André Galligo(*)

June 23 2001

Laboratoire de Mathématiques J.A. Dieudonné
Université de Nice, Parc Valrose, 06108 Nicedex 02, France
(*) and ORCCA, University of Western Ontario, London, N6A 5B7, Canada
Gang.Xiao@math.unice.fr, galligo@math.unice.fr, galligo@scl.csd.uwo.ca

Abstract

We describe some of the features of the system WIMS, an interactive mathematics server accessible at <http://wims.unice.fr>. We also illustrate via an example the use of the tools provided to implement new exercises.

Keywords: Internet, Mathematical software, Education, Interactive server, WIMS, Virtual classes, Computer algebra, Online teaching, distant learning.

Introduction

WWW Interactive Mathematics Server (WIMS) is a system providing Internet-accessible mathematical computations. It is developed by Xiao Gang at the University of Nice.

Because of the importance of practice in mathematics learning, emphasis has been put on online exercises. Mathematical softwares is used either for the generation of exercises, for example for the generation of random unimodular matrices of different sizes, or for the analysis of user answers. If the question requires finding a basis of a vector space, there are infinitely many correct answers. PARI/GP is used to analyse dynamically the answers instead of comparing them with a prepared database of correct answers.

Analyses of students' progress can be easily achieved using an Internet server for generating the exercises and processing the answers. Moreover data and results can be well-protected against cheating.

To obtain a high degree of mathematical sophistication, the basic design concept of the WIMS project is server-side interaction via http protocol, with a modular structure at two levels. On the one hand, each application (exercise or tool) under the system can be independently created, modified or removed, allowing the system to host a large number of such applications. On the other hand, the system calls various mathematical software (numerical or symbolic computation, visualization, proof assistant, etc) as background engines, with an independent interface designed for each such software.

These online tools on WIMS are not intended to replace traditional use of mathematical software. They are intended for 'casual' users who only want to carry out 'simple' computations. For such users, on the one hand it is often not cost-effective to learn a whole syntax set of software. On the other hand, web user interfaces can be made much more user friendly. Moreover, it is possible to create applications on WIMS combining the computing power of several different software systems.

The paper is organized as follows.

In the first section we describe the design of the system and we list the main tools currently interfaced with WIMS. In the second section, we describe our concept of virtual classes and their use for teaching mathematics. In the third section, we describe examples of WIMS modules. In the fourth section, we describe an example of development of a new exercise in WIMS.

1 Design and interfaced software

1.1 Design

The center of the system is a cgi program, `wims.cgi`, which is written in C. That program serves as the unique entry point for all user requests, and directs these requests to the appropriate applications.

Each application under the system is an independent unit, called a module. Modules are written in a proprietary scripting language, and are interpreted by `wims.cgi`. Inter-module communication is achieved via the usual http links. Many capabilities of the system are available to modules via special commands in this scripting language, so that these capabilities can be enhanced or upgraded without modification at the modules' level.

The principle of WIMS is to enable all the sophisticated mathematical computations to be carried out by exterior programs, in order to gain maximal power and versatility with minimal cost of development. With this consideration in mind, `wims.cgi` can only carry out very basic computations by itself. To some extent, the system behaves like a special operating system, with a user interface through the Internet, a programming interface through the modules' scripting language, and interfaced software as the system's resources.

The home site of WIMS (<http://wims.unice.fr>) is freely accessible to all. An online documentation WIMSdoc can also be found there, where technical details such as the scripting language for writing modules are documented. The source code of the system (server and modules) can be downloaded from the home site, distributed under the GNU General Public License.

1.2 Interfaced software

In principle, any program allowing batch mode execution can be interfaced with WIMS. The main restriction is that softwares with restrictive licenses should not be made freely accessible.

The following interfaces are among the currently available ones.

- PARI/GP. This is an excellent arbitrary-precision package for computations not involving transcendental functions. The header of PARI/GP output is automatically stripped off by the interface.
- MAXIMA is now the basic symbolic algebra system upon which WIMS relies. Its greatest advantage is being completely free (GNU).
- MuPAD. This is a symbolic algebra package whose copyright holder's allows this kind of use on the Internet.
- Gnuplot. The interface includes adaptation making it possible to plot animated sequences, by adding a few extra parameters.
- Povray, a popular ray tracing package.
- TeX.
- COQ. This is a proof assistant package built upon CAML. A WIMS module using this package can support step by step operations.

One of the problems which occurs when several programs are called in a single process is that each program has its particular syntax for mathematical expressions. In order to make the mathematical expressions directly

usable from program to program, routines translate syntaxes into the syntax of to the interfaced program.

2 Teaching mathematics in Virtual classes

2.1 Virtual classes

WIMS is designed to support intensive classroom applications. To this end, a structure of virtual classes is incorporated. A virtual class under WIMS can be created and managed entirely online by the teacher (supervisor) of the class. The content of a virtual class is a number of work sheets. Work sheets are created and maintained online by the teacher. Each work sheet contains a number of items. Each item is the address of a WIMS module, which may be an exercise (for which the teacher has determined the difficulty level), an online tool, or an online lesson text. The design of the system requires that the teacher of the virtual class takes the responsibility of choosing the exercises for his students. Example work sheets will also be created, which can be directly inserted to virtual classes, helping teachers to make their choices.

Each time a student answers an exercise, the score he gets as well as the resulting change to his averages are shown on the page. Experiments show that this instantaneous result is very efficient in inciting students to work hard. Ultimately, it is possible to generate intelligent personalized guides indicating the performance of each student. The structure can host efficient electronic discussions between teacher and students, or among students.

Security and anti-cheating are also very important aspects in the design of WIMS. As the system allows remote visitors to execute various programs on the server, much care has been taken to avoid risks related to security issues. WIMS is only implemented on the Linux environment which provides a high level of security protections.

2.2 Teaching Mathematics

We have made a large-scale use of virtual classes (400 students) at the University of Nice during the last years. Practice shows that students start to work easily with WIMS very quickly after registration. The behavior of the students under this environment is usually quite different from that in a classical environment. They become very active, trying to find out the most efficient way to get points. When the teacher is not immediately available, discussions among students start quickly, with the most advanced students

helping those in difficulty. However, nobody can copy another, because everybody is working on a different exercise (automatically generated), and the differences in the questions may often lead to radical differences in solutions.

WIMS is widely open. On one hand, it already contains a large number of online exercises classified by subject and easy reachable through a research engine. On the other hand, it provides friendly tools for developing modules for new exercises. In order to illustrate these capabilities, in the next section we list some current applications, then we present an exercise. In the following section, we relate, step by step, the construction of 2 module in WIMS, using different tools; both address the same pedagogical question.

3 Examples of WIMS modules

3.1 A wide variety of applications

- One of the most popular activities on the WIMS server seems to be the *Shifting Puzzle*, partly because no mathematical sophistication is needed to play with it. This is a typical step-by-step interface: It presents a scrambled mn puzzle, and the user can successively shift a row or a column of pieces, until the puzzle is recovered.
- Another step-by-step example is Gauss, which can generate linear systems or matrices over various coefficient rings, including finite ones. The user can make successive modifications of rows by clicking or by returning forms. Since PARI/GP is involved in all of the computations, a change of coefficient ring amounts to some minor changes in the script sent to PARI. This is controlled by a parameter.
- The system also allows "interactive" exercises, in which the user is encouraged to ask questions in order to solve a problem. *Linear system dialog* is such an example.

The server can detect situations such as pure speculation – where the user does not have enough information to determine the answer to the problem – or unnecessary information, such as the number of equations, which is not directly related to the number of solutions of the system. At the end, it attributes a score to the user, which depends on the efficiency of the questions asked.

- *Graphic functions* is a classical multiple-choice exercise, but one that generates interest by constructing both the question and the choices as random graphics. There is no database of pictures behind the exercise:

Every curve is plotted on the fly by Gnuplot, using randomly generated functions, taking care that no ambiguous situations (in which several choices may look good) are generated.

To give an idea of the variations offered by the application Graphic functionse, here is a list of its capabilities.

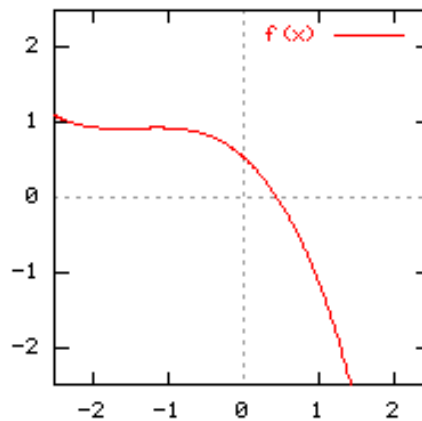
1. Graphic derivative, recognize the graph of the derivative of a function.
2. Graphic functions, recognize the graph of $f(-x)$ from that of $f(x)$, etc.
3. Graphic addition, recognize the graph of $f(x) + g(x)$.
4. Graphic inverse, recognize the graph of an inverse function.
5. Graphic integral, recognize the graph of the integral of a function.
6. Graphic abs, recognize the graph of $f(|x|)$.
7. Graphic multiplication, recognize the graph of $f(x)g(x)$.
8. Graphic inequalities 2D, recognize a region described by inequalities.
9. Graphic subsets, recognize a graphically described subset.
10. Graphical convergence, determine the limit of a recursive sequence according to the graph of the function.
11. Graphic ODE, graphically recognize solution in of a linear ordinary differential equation.
12. Graphic ODE phase graph, recognize the phase graph of a given ODE.
13. Fourier development, graphical search of development of a function.
14. Graphical decrypt of a picture crypted by a pseudo-random sequence.
15. CoincidenceDev, graphically find the Taylor expansion of a function.
16. CoincEqDif, graphical exercise about differential equations.
17. Parametric choice, from a parametric curve, recognize values or derivatives of functions.
18. Animated graphics.

3.2 A typical exercise : Simple geometric transformations on graphs of functions

The pedagogical aim of this exercise (number 2 in the previous list) is to train the student to recognize simple geometric transformations on the graph of a function and express it on its formal expression. The figure 1 is a copy of the corresponding web page:

Graphic functions

Here is the graph of a function $f(x)$.



Question. Among the following figures, which one represents the function $-f(x)$? Click on it.

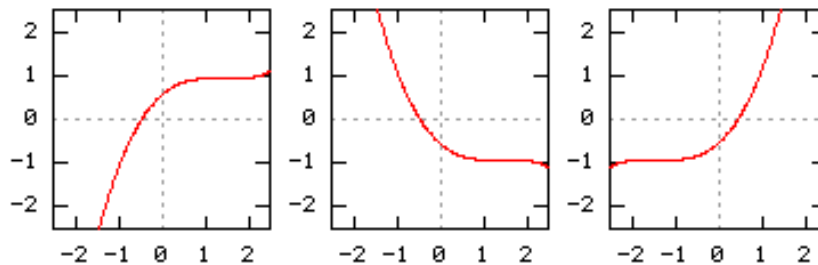


Figure 1: a Wims exercise

4 Development of new exercises

4.1 Specification step

Once a teacher has specified a pedagogical aim, he (or she) has to transform his idea into one or several exercises which can be easily answered and evaluated. The student's answer is going to be automatically checked, errors detected and the information delivered.

Let us take an example in elementary analysis at a high school level, the subject is approximate evaluation of functions.

An (ambitious) pedagogical aim is to develop a set of exercises which:

- review the use of basic rules on inequalities of real numbers,
- develop intuition to distinguish between sharp and rough bounds,
- experimentally show the difference between necessary and sufficient conditions,
- review elementary monotonic functions.

An example of a simple scheme of exercise which addresses all these points is the following.

“Two real numbers a and b are constrained to belong to two intervals. The student has to determine if a given inequality $f(a, b) > c_0$ is always satisfied or provide a counter-example.”

f is chosen to be a polynomial of degree at most 1 in A and of degree at most 1 in B ; where A (resp. B) is a monotonic function of a (resp. b). So the solution can be easily computed by the system.

For the students, the difficulty depends on the choice of f and of the intervals.

In a more advanced version (that we will not consider here), when the claim is True, the student should provide a step by step proof.

In order to implement this exercise in WIMS, the teacher has to describe more closely the functions, the intervals and the random variables to be used. Then she (or he) may chose one of the two formats proposed by the system.

We will describe in detail a quick construction of a simplified version of the exercise using the format “OEF” then, we will describe a complete implementation. We begin with a few words on modules.

4.2 About modules in WIMS

The system offers great freedom in the design of modules. In the simplest (but not very useful) case an ordinary html file may be put into a directory, to make this directory behave as a WIMS module.

On the other hand, there is no limit on the complexity of a module. `main.phtml` can call other files in the module; inline mathematical symbols and commands can be inserted into these files; in order that the module be correctly indexed on the site, a file named `INDEX` should be created; auxiliary (static) graphics files may be included; variables which the user is allowed to define when requesting the module should be declared in `var.def`; etc.

The output of the module is defined by `phtml` files (`phtml` stands for ‘programmable html’). The format of `phtml` is a proprietary extension to the `html` language by allowing WIMS commands and variable substitutions in the file.

In order to achieve maximal independence between WIMS commands and `html` tags, the command language is line-oriented. The system only interprets lines whose first non-space character is the exclamation mark ‘!’, it will consider the word following this ‘!’ as a WIMS command. All other lines are simply sent to the user, after eventual variable substitutions. This allows these commands to take place anywhere in the `html` result: within `html` tags, `html` header fields, `http` references, `javascript` scripts, etc. As a result, the interaction between WIMS commands and `html` tags is very flexible. The result sent to the user is a pure `html` file, with all WIMS-proprietary codes replaced by their outputs.

Although it is possible to put all the processing of user requests into a `phtml` file, it is recommended that computational codes be put into variable processing files which have a slightly different syntax. This syntax will be more convenient for variable processing. Such a distinction between computation and output also helps multi-language support: when a module is translated to another language, only `phtml` files need to be translated.

There can be two variable processing files: `var.init` which is interpreted at the module’s initialization, and `var.proc` which is interpreted at every user request. An arbitrary number of other variable processing files can be present, to be called by the above two.

4.3 A quick construction

WIMS proposes a simple construction mode for simple modules. The teacher has to complete a so called OEF (Online Exercise Format) file which will be translated into html. The syntax is straightforward and is indicated in the help menu.

Here is the file for a simplified version of our exercise, and some comments.

First, we provide administrative data :

```
\title{Inequality I}
\author{Andr Galligo}
\email{galligo@math.unice.fr}
\computeanswer{no}
\format{html}
```

Then we declare the variables for the 2 intervals $a_0 < a < a_1$ and $b_0 < b < b_1$ which are represented here via their middle points aa and bb . Note the use of the command `randint` which returns a random integer in the indicated range.

```
\precision{10000}

\real{aa=randint(-50..50)/10}
\real{az=randint(6..20)/10}
\real{ba=randint(-50..50)/10}
\real{bz=randint(6..20)/10}
\real{a0=aa-az}
\real{a1=aa+az}
\real{b0=ba-bz}
\real{b1=ba+bz}
```

Then we declare the function f , which here has the simple form $la + mb$, the value c , and the orientation of the inequality ($>$ or $<$) together with their display.

```
\real{l=randint(10..50)/10*random(1,-1)}
\real{m=randint(10..50)/10*random(1,-1)}
\real{c0=randint(-200..200)/10}
\function{f=\l*a + \m*b}
\text{tf=htmlmath(\f)}
\text{lt=lt}
```

```

\text{nbsp=nbsp}
\integer{orient=random(1,2)}
\text{ineq=\orient=1?lt:gt}
\text{i1=\orient=1?A:B}
\text{i2=\orient=1?B:A}

```

Then we perform the computation (evaluation at the 4 summits of the rectangle) needed for the solution called “good”.

```

\real{v1=evaluate(\f,a=\a0,b=\b0)}
\real{v2=evaluate(\f,a=\a0,b=\b1)}
\real{v3=evaluate(\f,a=\a1,b=\b0)}
\real{v4=evaluate(\f,a=\a1,b=\b1)}
\real{min=min(\v1,min(\v2,min(\v3,\v4)))}
\real{max=max(\v1,max(\v2,max(\v3,\v4)))}
\text{good=C}
\text{good=\min<\c0 and \max<\c0?\i1}
\text{good=\min>=\c0 and \max>=\c0?\i2}

```

Then we write the statement of the exercise, which will appear on the web page, this is written with a simple html syntax.

```

\statement{We have two real numbers, a and b, with
<p><center>
\ao &\lt; a &\lt; \a1 &\nbsp; and &\nbsp;
\b0 &\lt; b &\lt; \b1 .
</center><p>
Do we have \tf &\ineq; \c0 ?
<ul>
<p><li>A. Yes, for any values of a and b satisfying the conditions.
<p><li>B. No, it is never true for any a and b with the conditions.
<p><li>C. That depends. It is true for some values of a and b and
false for some other values.
</ul>}

```

Then we store the answer.

```

\choice{Good answer}{\good}{A,B,C}

```

Modtool

List of files

Your module *Inequality zone* (determine an inequality for variables in a given zone.) [[Propert](#)

Name	Action	Explanation
about.phtml	show edit delete	About file. You can leave it alone.
answer.phtml	show edit delete	Output result of user reply analysis.
filedesc	show edit delete	Explanation of each file in the module.
form.phtml	show edit delete	User reply form.
help.phtml	show edit delete	Help to users.
intro.phtml	show edit delete	Introduction and configuration of the exercise.
main.phtml	show edit delete	Main output organization file.
msg.phtml	show edit delete	Error messages.
present.phtml	show edit delete	Presentation of the problem.
var.def	show edit delete	User variable declarations.
var.init	show edit delete	Variable initialization file.
var.proc	show edit delete	General variable processing file.

Add a new file:

[Test the module.](#) [Welcome page of Modtool.](#)

Figure 2:

4.4 Implementation of the complete exercise

The help program provides a tool for online development of full-power WIMS modules, via 10 pre-written files which will be updated (see the figure 2).

To program our exercise we completed the files in the following order.

1. var.def contain the list of variables and their statutes: either “deny” if the student cannot change them, or “reply”.
2. var.init contains the description (declarations and assignments) of the variables as in the previous subsection but with a slightly different syntax.
3. present.phtml contains the description of the web page.
4. var.proc contains the calculations and prepares the analysis of the answer.
5. answer.phtml will provide the diagnostic and the scoring.
6. msg.phtml enables the treatment of the error messages.
7. intro.phtml contains the title of the exercise, a short text of presentation and the variable managing the level of difficulty.

The other files are left unchanged.

To give a better idea of the task, we print hereafter the file var.proc, the lenghtiest. The syntax is rather straightfoward (e.g. !mathsubst means substitution in an expression and !evalsubst means evaluation of a univariate function. We note the required error management in the recovering of answers. Once we have in mind the exercise and the program of the simplified version (see previous subsection) the meaning is obvious.

```
sign1=<
sign2=>
sign=$(sign$orient)

!if $cmd=reply and $status!=waiting
  error=double_reply
  !exit
!endif

!if $cmd=reply and $status=waiting

  choice=!trim $choice
  !bound choice within A,B,C default $
  !if $choice=$empty
    error=empty_data
```

```

!exit
!endif

f1=!mathsubst a=($a0) in $f
f2=!mathsubst a=($a1) in $f
v1=!evalsubst b=($b0) in $f1
v2=!evalsubst b=($b0) in $f2
v3=!evalsubst b=($b1) in $f1
v4=!evalsubst b=($b1) in $f2
max=$[max($v1,max($v2,max($v3,$v4)))]
min=$[min($v1,min($v2,min($v3,$v4)))]
good=C
!if $min>=$c0
!if $orient=1
good=A
!else
good=B
!endif
!endif
!if $max<=$c0
!if $orient=1
good=B
!else
good=A
!endif
!endif

diagnostics=good
!if $choice!=$good
diagnostics=bad
!endif
!if $choice=C and $good=C
!for i in a1,a2,b1,b2
ex$i=!trim $(ex$i)
!if $(ex$i)=$empty
error=empty_data
!exit
!endif
ex$i=${$(ex$i)}
!if NaN isin $(ex$i)

```

```

        error=bad_data
        !exit
    !endif
!next i
!if $exa1<=$a0 or $exa1>=$a1 or $exa2<=$a0 or $exa2>=$a1 or\
    $exb1<=$b0 or $exb1>=$b1 or $exb2<=$b0 or $exb2>=$b1
    error=out_of_range
    !exit
!endif
f1=!mathsubst a=($exa1) in $f
f1=!evalsubst b=($exb1) in $f1
f2=!mathsubst a=($exa2) in $f
f2=!evalsubst b=($exb2) in $f2
!if $orient=1 and ($f1>=$c0 or $f2<$c0) or\
    $orient=2 and ($f1<=$c0 or $f2>$c0)
    diagnostics=bad
!endif
!endif

status=done
!if $diagnostics=good
    module_score=10
!else
    module_score=0
!endif
wims_module_log=score $module_score/10
!endif

```

Conclusion

In this paper, we have presented the aims and some of the capabilities of the system WIMS. We have illustrated our presentation by the implementation in WIMS of a simple exercise. Of course more complicated exercise will require the use of interfaced computer algebra systems or numerical or graphics tools. The strategy of resolution and analysis of the answer can also be much more intricate. Now, the best way to evaluate the interest and the current capabilities of WIMS is just to go to the web site and use the system.

References

- [1] Xiao Gang: *WIMS: An interactive Mathematics Server* Journal of Online Mathematics and its Applications, 1, 1, January 2001. (see www.joma.org).