

# Mathematical Systems Accessed on the Web: Preliminary Overview

Paul Libbrecht  
paul@activemath.org

June 28, 2002

**Keywords:** web-based learning environment, content modeling, re-usability, course-management, personalization

## 1 Introduction

Mathematical Systems offer interesting capabilities to explore the *mathematical reality* with the help of properly triggered computations and display. As soon as mathematical content is placed on the Web, the capability to share the computation methods with others appears.

The first method to share mathematical computations is by distributing so-called worksheets. These files are made of code specifying to the mathematical system the sequence of operations and its data. They require the receiving side to be able to run the same system which may be a problem in many situations: some mathematical systems require specific platforms, some others have a high price, all require a non-trivial installation, etc.

A more elaborate method to share mathematical computations is to offer the mathematical system accessible through a Web interface. We call a session of mathematical computations *accessible through a Web-interface* if it is triggered by few mouse-clicks while browsing and reading related documents.

The following are essential parts of such systems:

- A mathematical system to perform computations
- A user-interface to display the results of computations and to receive the input of the user
- A connection mechanism between the two

We call systems that contain all these parts “mathematical-systems-connection”. The sole fact that they are *on the Web* makes them different from the classical computer-algebra desktop applications as the interactions happening within the session are related, most-probably tightly, to the content the user is browsing.

**Usages:** Sessions of such connections can be used for plain exploration purposes as is generally done with computer-algebra-systems. The user is then free to use the computations and their visible results to understand, conjecture, or verify mathematical facts.

Within learning environments, however, such connections can be used to have users perform exercises whose solutions are automatically evaluated. For these sessions, interaction is expected to be richer than a normal mathematical system: the user’s actions are to be evaluated and feedback is to be provided.

**Contents:** This preliminary article attempts to introduce the basic building blocks found in most of the available mathematical-systems-connection available, technologies associated to them are described followed by a list of characteristics. We conclude with the start of a list of available systems with a little description of their characters under the lights of the concepts introduced.

This article is preliminary in the sense that the set of available solutions is not extensively covered and, as such, the study may appear superficial. It is the intent of the author, however, to enrich a web-based version of this overview.

## 2 Typical Components

Allow us to remember that a session of a mathematical-systems-connection is triggered by a click while *web-browsing* (this could be something different than an HTML browser, in particular, such tools as Acrobat Reader or Xdvi (with hypertextual navigation enabled) can be considered such. In this scenario a client computer exists, the machine where the *web-client* runs, a server exists, the machine(s) that serve the documents.

Running on the client computer, our first component is the **client user-interface**: it should appear at this click in a reasonable amount of time. This can be an application opening a document downloaded (like a computer-algebra worksheet) in which case the client is understood to be the document-window within this application. This can also be a *web-client* window. From now on, the user is expected to perform actions that will trigger computations and whose results can be displayed by this client user-interface.

These computations are sent to the mathematical-system. The **communication** involved to send this is our second concern: the client and servers running for this task need (if remote is desired) to communicate through network. Characteristics of this communication include the ability to have multiple communication going on at the same time, or the reliance on a common standard.

Connected to the server is a component which we shall call the **proxy**. It serves as wrapper for the mathematical system and can be viewed as the responsible component to manage the life of the mathematical system and the connection.

Finally, the **mathematical system** running on the back is connected to the proxy by some appropriate ways.

## 3 Technologies Involved

We overview quickly major technologies that can be used in the components described in the previous section.

**Proxy Technologies** A proxy is a simple piece of code which can, presumably, be written in any language as long as it can be triggered by a web-server framework. Common names include Java Servlets, Python with Zope, plain Common Gateway Interface program, or others.

The other side of the proxy is the connection to the mathematical system. Some mathematical systems offer programmatic access to their kernel, this is the case of Mathematica with their JLink and MathLink products, of Pari [11], of Maple with their socket server, ... A common practice, however, for this task is to use the standard-input and standard-output of the mathematical system, run as a process. The proxy job is, then, to feed the system's input with the appropriate commands and listen to the results recognizing when the system's output is finished. Although this is mostly an easy string-matching task, the behaviour can become tricky when the expressions given to the mathematical system can be arbitrary e.g. when the user enters them. A typical example of such problems occurs, for example, with the ACTIVEMATH MAPLE console or the RIACA GAP server: if an expression of the form “? groups” is sent, the mathematical system will attempt to display a help page which, as a normal terminal manual page, will display page per page, expecting the user to press a key to go further. In most observed cases, this leads to a freeze of the system which has no other choice than to be entirely destroyed.

**Communication Technologies** Between the client and the server, a two way communication has to be made, running over a network. As web-servers are being used, it is quite common to use simple HTTP GET or POST methods to transmit the results and answers. Depending on network and server conditions, this can be rather limited: several web-servers limit the space allocated to the reception of POST method to 4 kilobytes, and parameters of the GET methods are generally even more restricted. For this reason, different methods may be appropriate. In particular, platform neutral technologies (such as XML-RPC, Corba, or SOAP) can be considered as a good solution, providing flexibility in implementations of both servers and clients.

**Client technologies** Server technology choices are generally rather unproblematic as they involve machines that are controlled by the owner. Client technologies are where the most delicate choices have to be made: responsiveness, freedom of programming, cross-platform characters and many other factors will depend on such choice. Moreover, the client machines can be very diverse.

We shall not cover the delivery basic HTML files which, if the HTML is simple, requires nothing special on the client. This may become much more elaborate when dynamic HTML is required (with dynamically updated form elements and/or content) but the problem is well-known in web-publishing practices. Once leader in the client technologies, the Java-applet platforms was considered as the most flexible cross-platform solutions. The under-specification of the graphical user-interface library (the so-called Abstract Windowing Toolkit) turned out, however, to be a nightmare when testing applications on different platforms. Since then, Java-applets have been dropped by many many commercial parties which tend to prefer dynamically-generated HTML even though these their responsiveness is very restricted.

Since then, java versions have kept being enhanced, and, though the cross-platform promise is only half true and testing on multiple platforms is still a requirement, the latest java versions start to fulfill the promise.<sup>1</sup>

Just as requiring an advanced Java may require an installation on the client-side, several other client technologies for multimedia are emerging as valid solutions: among others the Macromedia Flash platform (ready-built in most current platforms of the world), and the promising Scalable Vector Graphics standard both offer programmatic server communication, a real development environment, and open-source libraries to generate them.

The most widespread way to enable the open-on-click behavior expected by a Web environment is to use document-type configurations, telling the browser it should use a given application on the client to open a document of a given type previously downloaded by the browser. This technique allows to build quite rich applications and to re-use existing applications in a Web environment, like the CCP project does [10]. The approach has, however, several drawbacks as it only allows the request to generate the document, then, most of the time, stops communicating to the server. The first problem is that the application is not told the URL the document was from, hence, if further connection is desired, it must be encoded in the document (thus preventing proprietary encoded documents like most commercial applications use). The second problem is that only hand-crafted applications can report to the server a result of the user's interaction; for pure exploration purposes, this is not a problem, but for a learning environment, where a report about the user's success or failure is expected, is essential. These two problems explain well why the CCP project (and generally the computer-algebra worksheet download technique) could not go any further although it has an interesting content collection.

## 4 General Characteristics of a Mathematical-Systems-Connection

In this section we describe typical characters of Mathematical-Systems-Connection. Using these characters, we shall be able to make a little zoology of the realizations currently known by the author. Our character-lists starts on the client-side aspects and ends on the architectural choices qualities.

---

<sup>1</sup>A late newcomer in the java world may also be considered, the Java Web Start extension (a separate download) which allows the same Web-based launch together with a better caching, and the possibility for a relaxed security policy.

**Expressivity of the Client Interface** The user-interface displayed on the client is the visible part for the user. Its characteristics will determine how the user will enjoy its use for the purpose of explorations or learning.

Mathematical objects can be described by many visual appearances. The formula display is the most widely used. The ability to display properly a formula is a first important characteristics for any mathematical user-interface and is an issue in itself (the development of the MATHML-presentation language illustrates this well).

A user-interface to display mathematical objects to be manipulated can however offer richer visual representations and interactions. The ability to view the graph-plot of a function, to view a geometrical illustration of a set of equations, to manipulate these views, or even to manipulate the mathematical objects by acting on the views (such as the elementary geometry systems) are all examples of such characteristics. Generally, the more expressive a user-interface is, the more constrained its application domain is.

**Ease of client input** Mathematical systems are expected to react to user input providing an answer to user gestures. The issue is in the amount of mouse-movements and keys typed to input a typical mathematical object or transformation.

Classical Computer Algebra Systems are based on an linear input syntax where the a textual input is then interpreted by the mathematical system, as commands or mathematical objects. This approach allows many type of inputs and commands but requires the user to know the syntax of the system. When used only rarely within a learning-environment, this can be a bottleneck for the learner. Moreover, the mapping between graphical display of formulae and their input syntax is often not-obivous, though some systems allow the copy-and-paste paradigm from (parts of) a displayed formula to the linear syntax.

Formula editors (such as the WebEQ input-control or JOME attempt to make the input of mathematical objects as accessible as possible, using many palettes of buttons and menus supplemented by contextual help. Their usage is well suited for beginner but can be very quickly awkward.

**Responsiveness of the user-interface:** A side facet of these two characters is the responsiveness of the user-interface: as a reaction to a user gesture is probably involving remote communication, the speed factors have to be seriously considered when designing a whole architecture.

**Client Configuration Issue** A famous principle of the current web states that the client machine should not require any installation. This principle is rarely achievable on a large scale and a complete statement of the browsers-requirements is part of every normal deployments.

It is to be noted that most of the users may be extremely deranged, if not loose the taste to use the application, if unexpected or buggy behaviour occurs. As a result, the requirement to install a better browser or perform other large download, though it may be a major burden to some users, may turn out to a much greater overall satisfaction. Moreover, the assumption to cross-platform browsers or other client technology can be only rarely assumed.<sup>2</sup>

**Security Related Issues** The configuration required so that the single click is enough to trigger the system generally involves the installation plug-ins and/or the automatic opening of certain document-types by other applications. It is important to note that performing such configurations may endanger the system of the client as long as the applications or plug-ins that open the document allow almost any *script* to be opened: as is the case in many applications, the script interpreters often have no *sandbox* mechanism (like the java-applet security restrictions) hence the application makes no difference between a trusted script or a script downloaded from a location that was just discovered.

The same issues threaten the mathematical system and possibly other scripting interpreter on the server: it is quite common practice, when a system starts to be scriptable, to allow users to run any kind of scripts,

---

<sup>2</sup>As of today, the only browser that has been showing a somewhat consistent behaviour between platforms is the latest versions of Mozilla. This uniformity could only be reached with a complete rewrite of all basic components of the systems. Opposed to this one, the best counter-example is Internet Explorer whose code-base is entirely split between the Macintosh and Windows editions.

including some that execute other processes or read any file. Moreover, most mathematical systems consider the mathematical evaluation as an execution process which, typically, triggers the execution of any script. As a result, a textfield in a Web-page could, with the help of mathematical-systems-connection, invoke the deletion or reading of some files on the server. In the computer-algebra systems world, only the very last versions allow them to be run in a *safe mode*, which, however, offers no guarantee.

**Feedback capabilities** When mathematical-systems-connection are used within learning environments (which may be static web-pages), it is expected that the sessions are directed towards a precise goal. To achieve this goal, the learner may be helped by some indications as of the correctness of a computation-step or by some hints indicating the next steps.

Such feedback is most probably easier provided with a complete mathematical-systems-connection as it requires some evaluation of the computation-states and possibly some intelligent engine.

**Connectivity Issues** The communication between the client and the server-part is expected to happen remotely through a form of network, most probably over a TCP/IP network on which the current Internet is based. Various configurations of TCP/IP connectivity can occur, however: among the most common limitations are the firewalls which may prevent a client from being reached by any incoming socket connection and which may, even, prevent completely the connections made outside of a precise port-number, generally the HTTP port number 80. Other such issues include the fact that the connections may be made over interruptible connections (such as a dial-up connection) and may be attempted to be resumed later from a different IP address.

The robustness to these issues is a characteristics very rarely seen in mathematical-systems-connections which is probably due to the fact that implementations have not reached the full maturity.

**Maintaining the Session** We now reach to the characteristics of the communication and proxy components of the architecture.

A first important character of the connection a mathematical system is the lifecycle of the mathematical system. It is generally simpler and more secure to open the mathematical system at each user's request. This allows the system to be predictably clean at each requests.

Opposed to this one is the behaviour imitating the computer-algebra systems sessions where the connection to the system's kernel is maintained and state can be stored on the system. Such states include library loading and variable or function definitions which the user or system may be able to user later.

**Resource Management** As the ease-of-use of mathematical systems grows, complicated computations can very often reach the computation limits of the underlying systems. One such usage is when manipulating algebraic geometry objects (such as varieties) or computing very high prime numbers.

When the mathematical system is on the client, the user alone is responsible for such management and can, hopefully, react to interrupt a computation that is too demanding.

When the mathematical system is on the server, it is probably accessed by several users at the same time. Moreover, it should be always available, at least to perform non-demanding manipulations. For this purpose, the server-side architecture ideally needs to be aware of CPU consumption so as to prevent denial of service attacks. Very few systems implement such a feature until now.

**Ease of Authoring** As developing a mathematical-systems-connection may take quite an effort in development, especially for the user-interface. As a result, the ability to use a common development for several purposes arises naturally. It is even more important when non technically-skilled mathematicians are "authors" of these exercises.

How elaborate and how easily a mathematical-systems-connection can be authored are issues only mildly touched by most of the available software which, mostly, rely on a classical scripting or programming architecture. How user-friendly such a programming language can be is an important character. For example, the Python or Javascript languages are probably best examples of such languages.

The mathematical systems, themselves, may provide such a language, though not aimed at external connectivity and lacking everything in the direction of external interfaces. Programming within the mathematical systems, however, can offer a lot when evaluating and defining mathematical objects.

Authoring the user-interface is another major development that has only found decent solutions, currently, in the scripts-in-HTML-page server frameworks.

**Application Domain Restriction** When the mathematical-systems-connection is generic enough to be authored, the obvious question comes as to which mathematical domain can be covered. This can be, of course, limited by the user-interface which may be only capable of displaying special kinds of mathematical objects. Such a limit can also be imposed by the mathematical system itself.

Many mathematical-systems-connection tend to cover the so-called “K-12” education-level. Very few mathematical-systems-connection components available tend to be extensible. It is, however, a character that can be very important. Among others, there exists a complete zoology of specialized mathematical systems which cover most of the mathematical domains that can be put on a computer. These systems, many of which are open-source, would take great advantage of being displayed from a server as their usage would not require an installation for each new system. Having a Formula editors or viewers, would be put to good use when creating a mathematical-systems-connection if they were extensible..

**Re-usability of Components and their Pieces** A classical concern in software development is the ability to re-use parts of the code written. With the current component-oriented technology, this concern is even more important as it opens the door to programs created visually where little knowledge is required from the user of such an environment which could be a mathematical author.

Domains where some parts may be re-used include especially the user-interface component(s). Re-using long-matured and long-tested user-interface components (such as a 3D viewer or a formula editor) can be of invaluable help as the multi-platform test-procedures may consume a lot of resources. For this to happen, however, standard practices need to be defined and used. The lack of such standards is, again, probably due to the youth of the world of mathematical-systems-connection.

**Uniform Content Encoding** Among the interface-standards that have to be worked, one has been particularly studied: the encoding of mathematical objects. Two major players attempt to make a uniform encoding that could support (using extension mechanisms) all mathematical knowledge: the OpenMath [2] and MATHML-content [3] standards.

Using such uniform content encoding is one of the key points when designing extensible formulae viewers or editors or when developing parts of a mathematical system on the Web when re-uses in completely different systems is used. Currently, however, most of the systems use proprietary encodings; major computer algebra systems have started adopting MATHML (partly presentation, partly) content) but interoperability among them is still very restricted.<sup>3</sup>

## 5 A Little List

This list cannot, at least for the purpose of this paper, be complete. The author will try to include most of the relevant ones at his knowledge as of this writing. The paper will, however, be converted to a Web format which will be updated regularly from [http://www.activemath.org/math\\_sys\\_through\\_web](http://www.activemath.org/math_sys_through_web).

Thus, we shall omit many existing software and other projects which shall be inserted progressively in the web-pages.

---

<sup>3</sup>Most impressive results obtained using MATHML are results focused on MATHML-presentation.

## 5.1 Integrated Solutions

The integrated solutions provide the whole range of components in mathematical-systems-connection architecture, working together in a precise setting. It is only using such frameworks that one has hopes to author sessions connected mathematical systems with a responsive user-interface but without programming.

**The IAMC/Dragonfly pair** [8]: The pair provides raw access to classical mathematical systems sessions using a protocol called IAMC. Technologies are Java, using Swing (and WebEQ) on the client. The resulting user-interface is close to be as comfortable as that of the mathematical systems themselves, providing layout-view of the formulas (using the WebEQ component) and graph-views. Transport mechanism is made through simple TCP sockets, hence connectivity issues may arise.

**The ACTIVEMATH learning environment** [9] provides access to mathematical systems sessions with a goal to reach as part of an exercise. The interaction with the mathematical system is transparent but is supplemented by a feedback evaluating the user's progress (with the evaluation taking place in the mathematical system). Content for such feedback is authored in an XML file, for which, currently, no authoring tool exists. Supported systems include MUPAD, GAP, and MAPLE. The technologies used are Java on both server and client. The client Java applet is an elementary terminal-like applet, where formulas are displayed in "ASCII-art". The remote connection uses the XML-RPC protocol, an extension of the HTTP protocol; it has thus been possible to package a "port-dispatcher" that allows all connections to be made through the same port. Sessions, however, don't survive connection failures.

Currently, although ACTIVEMATH content is based on OpenMath, the interactions with the mathematical systems are plain character-strings. It is planned, once phrasebooks are mastered, to use uniform encoding to provide more elaborate user-interface components that can be used with many mathematical system.

## 5.2 Server Frameworks

The server frameworks differentiate themselves from the integrated solutions in that they only rely on the server to provide the access to the mathematical system. They are based on the classical web-server or CGI script paradigm, where the client is generally a simple web-browser, which displays texts and images and offers elementary form-elements for input.

They are all based on the "script-in-page", now classical, paradigm, which interleaves HTML content with a script programming-language, executed on request. This language includes mathematical system's language function-calls and object declarations. Authoring content for such interactions requires a good knowledge of the HTML language and can only be written by hand. Such authoring also allows elementary feedback on single answers.

These systems, typically have no as everything is downloaded from one HTTP connection.

**The Worldwide-web Interactive Mathematical System (WIMS):** [17] This server is an Open-Source initiative that provides connections to many systems (including rendering engines). It uses a proprietary scripting language which supports, for a limited domain, a uniform encoding among mathematical systems. The server is developed in C and runs on Linux systems. WIMS has an elementary resource-management system, allowing to prioritize a group of users. No sessions are supported by WIMS and input on the client can only be done using one or another syntax.

**The webMATHEMATICAServer:** [16] This commercial server, built on top of the MATHEMATICA J/Link connection system allows a connection to MATHEMATICA kernels but only within the time of the requests (the kernel used is then reset to be re-used later). As a result, webMATHEMATICASupports no sessions.

**The RIACA tag-library:** [4] This emerging Open-Source framework is probably going to be the richest in its category: based on the Java-Server-Pages [7] technology, this collection of tags supports the connection to a few mathematical systems (MAPLE, GAP, MATHEMATICA). Using Java Servlets sessions, the connection to the mathematical system can live through several requests. Its usage as a tag-library allows such objects as embedded OpenMath or MATHML-content elements for use in different settings.

### 5.3 Mathematical Software Busses

The category of mathematical software busses provides to the application programmer (hence to the server programmer), facilities to connect to mathematical systems. Connections may or may not maintain a state through a series of calls. Translations to and from a uniform encoding may be offered.

**The MATHWEB Software Bus:** [15] The goal of this Open-Source software bus is, originally, to allow (semi-automated) theorem provers to request mathematical services offered by computation systems, such as computer algebra systems, or other reasoning systems, such as first-order automated theorem provers or model checkers. The MATHWEB Software Bus is written in the Mozart-Oz language and can be accessed remotely through either the XML-RPC or Oz-private remote calls.

The bus does not support resource management but offers interesting remote dispatching, enabling a *web* of mathematical services to be offered. Some of the services in the MathWeb-SB are session-less, e.g., automated theorem provers. These services die as soon as the result is returned. Some translation services are available in MATHWEB, others are planned; In the near future, the software bus will be enriched with agent-oriented approaches to query the existence of services with an abstract description of the task and problem.

**JavaMath:** [1] The JavaMath project is an Open-Source server solution to enable the connection to mathematical systems over the web. Typically, it is meant to run well (and provides a documented solution) with applets. The interest of JavaMath is that it bundles a few of the OpenMath phrasebooks (namely the ones for the MAPLE and GAP computer-algebra systems).

JavaMath does not have resource management. The communication protocol is originally the Java Remote Method Invocation (which is very fragile) but has been enriched with plain HTTP requests.

**System-specific connection methods:** The three major computer algebra systems, start to offer the ability to behave as a server and be requested computations using a programmatic interface. This the case of MATHEMATICA J/Link, of the MAPLE socket-server, and of the MUPAD computing server. They tend to offer a good control over the system's operations.

The latter is a separate product, being tested currently. The approach is to develop such a server to facilitate all the resource management issues so as to guarantee, for example, the permanent access to the server. This product seems to be the only one to go in this direction.

### 5.4 Client Components

The client components are distributed stand-alone and are meant to be either integrated in a larger framework or be simply embedded as a web-page element.

These components can be configured using elementary parametrization (typically in HTML pages. Some can even be "authored" using a form of authoring tool.

This category is extremely rich when one thinks of the large amount of mathematical applets reachable through the Web. The diversity of components found around the world shows that re-uses is barely a practice among these component developers. This diversity starts to be presented in the Journal of Online Mathematics and its Applications MathLets collection [12], many lists are to be found as well, most of which have troubles being kept up-to-date.

As a result, we shall not be able to cover all examples available. We present typical examples of components meant to be re-used in larger projects.

**The Java OpenMath Editor (JOME):** [5] This Java Bean, meant to become OpenSource, is a flexible component to simply view and edit mathematical formulas. Its interest lies in the fact that it can edit any OpenMath formula, which makes it suitable for many developers to re-use in a larger applet. Among the possibly built applications is a complete formula editor (with all the necessary buttons). The domain is extensible (but currently limited to the standard OpenMath content dictionaries, close to be equivalent to MATHML-content).

The fact that JOME edits OpenMath trees make it particularly suitable to be used in connections to mathematical systems. The uniform encoding allows it to be used in different fields and with different mathematical systems.

The fact the JOME is a Java bean makes it suitable to use in a *visual building* environment (allowing a user to determine visually properties of the bean). This technique represents, most probably, one of the interesting futures of such re-usable components. As an example, a possible work-flow could be an author working on a visual appearance of a user-interface using such visual builder, followed, possibly, by a developer packaging the supplementary behaviors.

**The WebEQ input control:** [6] This commercial Java applet (free for some uses) allows the view and editing of mathematical formulas. In this sense, it is similar to JOME. The advantage of WebEQ is in the packaging of the product: the WebEQ developers suite comes together with a good documentation and tools to deploy and encoded web-pages that can take advantage of this component. The disadvantage is in the limited domain: the internal data-structure of WebEQ is a MATHML-presentation data-structure. It can generate MATHML-content (which can then be used in communication to the mathematical system) but does so with a heuristics and has no intent to be extensible. This makes it usable for college level mathematics but breaks easily as soon as one reaches slightly elaborated mathematical concepts.

**The JGV three-dimensional viewer:** [14] <http://www.geomtech.com/products/JGV/index.shtml> Just as the WebEQ component, this viewer is another product that originated at the now dead Geometry Center. It is an Open-Source viewer for three-dimensional models encoded using a few of the classical formats, especially the GeomView produced formats.

Being a Java (1.1) component makes it slightly slow but widely runnable. As only a small interaction is possible using this component, no real issues exist in this component which performs its job honestly.

**The MuPad ActiveX controls:** [13] These components are delivered as part of the *professional* version of the MUPAD computer-algebra system for Windows and are specific to this family of platforms. They exist in several flavors: a formula-viewer (similar to JOME), a two-dimensional and a three-dimensional graph viewer. They are produced from within MUPAD, using the expressivity of its syntax for mathematical objects. These components can be integrated in a larger component and connected using Visual Basic. Thanks to the object linking architecture, they can be inserted in most documents viewed on this platform.

ActiveX controls are simply another kind of plug-ins, hence can be embedded into HTML pages previewed on Windows. The only security involved in this procedure is dialog popping up on download asking whether it should be opened. This is pale compared to the sand-boxing facilities of Java applets for which untrusted code can be tried. In exchange, the native character of these controls allows a faster execution which allows, for example, the 3-dimensional views to be very responsive.

## 6 Conclusion

We have attempted to provide an overview of the current mathematical-systems-connection. Using this overview, we intend to pursue the classification of existing products on [http://www.activemath.org/math\\_sys\\_through\\_web](http://www.activemath.org/math_sys_through_web).

From all the solution-types we described, one configuration has been entirely forgotten: as learning environments emerge, the usage of mathematical systems' in a more constrained fashion appears more and more

needed. Mathematical systems, however, are currently only made for explorations: the user has no requirement to follow given rules, or provide an precise answer before pursuing any further. For the purposes of providing feedback, a mathematical system that could be completely controlled with the help of the built-in macro-language (possibly controlled by a server as well) does not exist yet. Such a setting would, however, allow to combine the resource-distribution needed by heavy demand mathematical computations, with the dynamic character of mathematics-on-the-web.

When browsing around the web about mathematical applets or other forms of mathematics-on-the-web presentations, it is already today impossible to keep track of all the efforts around. Sadly, most of the development efforts are very close to be *hackers' tricks* which are developed for a while, are then deployed on the web, rarely with a license, and are then often forgotten and not re-used anymore. The quality of these productions is often low and tends to stay such. Documentation, if any, will remain incomplete.

To reach another level of maturity, the community of mathematicians working to publish things on the web should use more exchange resources so as to start building a real community of collaborating developers. It is to be hoped, one day, that a collective effort will emerge providing to mathematicians around the world the ability, in five lines of code, to deploy a graph-plotting component or an editable formula-display to be used with any mathematical system.

Another issue which is becoming more and more patent is the license: generally, mathematical systems are licensed on a private, company, or campus basis. With the world accessibility of the knowledge on the Web, however, the world-wide access to a mathematical computation comes as an experience-rich illustration to knowledge. Mathematical systems makers have simply not allowed this until very recently. The only such license found is the webMATHEMATICA *Amateur* license which has strong requirements such as the need for a banner of the product and the requirement to be world available (!).

## References

- [1] A. C. Andrew Solomon, Craigh A. Struble and S. A. Linton. The javamath api, an architecture for internet accessible mathematical services. see <http://javamath.sourceforge.net>, 2001.
- [2] O. Caprotti and A. M. Cohen. Draft of the open math standard. Open Math Consortium, <http://www.nag.co.uk/projects/OpenMath/omstd/>, 1998.
- [3] D. Carlisle, P. Ion, R. Miner, and N. Poppelier. Mathematical markup language, version 2.0, 2001. <http://www.w3.org/TR/MathML2/>.
- [4] H. Cuypers and H. Sterk. Mathbook, web-technology for mathematical documents. In *Proceedings of the BITE 2001 conference*, 2001. See also from <http://www.riaca.win.tue.nl/>.
- [5] L. Dirat. The java openmath editor, June 2002. <http://mainline.essi.fr/jome/jome-en.html>.
- [6] D. S. Inc. Webeq developers suite features, June 2002. <http://www.dessci.com/webmath/webeq/features.stm>.
- [7] S. M. Inc. Java server pages, June 2002. <http://java.sun.com/products/jsp/>.
- [8] W. Liao and P. Wang. Dragonfly: A java-based iamc client prototype. Technical report, ICM Kent State University, January 2001. Available at <http://icm.mcs.kent.edu/reports/2001/ICM-200101-0001.pdf>.
- [9] P. Libbrecht, A. Frischauf, E. Melis, M. Pollet, and C. Ullrich. Interactive exercises in the activemath learning environment. In *ISSAC-2001 Workshop on Internet Accessible Mathematical Computation*, 2001. <http://icm.mcs.kent.edu/research/iamc01proceedings.html>.
- [10] modules@math.duke.edu. The connected curriculum project, June 1999. <http://www.math.duke.edu/education/ccp/>.
- [11] G. Niklasch. Pari-gp home, June 2002. <http://www.parigp-home.de/>.

- [12] J. of Online Mathematics and I. Applications. The mathets collections, May 2002. <http://www.joma.org/browse.html>.
- [13] SciFace Inc. Flexible gui-komponenten für mathematische arbeitsumgebungen, April 2001. <http://www.mupad.de/PROJECTS/GERMAN/bmbf.shtml>.
- [14] G. Technologies. Jgv: 3d viewing in java, June 2002. <http://www.geomtech.com/products/JGV/>.
- [15] A. Voronkov, editor. *System Description: The MathWeb Software Bus for Distributed Mathematical Reasoning*, number 2392 in Lecture Notes in Artificial Intelligence. Springer Verlag, 2002.
- [16] Wolfram Research Inc. webmathematica, the way the web calculates, June 2002. <http://www.wolfram.com/products/webmathematica/>.
- [17] G. Xiao. Wims: An interactive mathematics server. *Journal of Online Mathematics and its Applications*, 1(1), January 2001. <http://www.joma.org/articles/xiao/xiaotop.html> .