Efficient Integration Of Real-Time Hardware And Web Based Services Into MATLAB

Stefan Müller, Heinz Waller
AG für Numerische Methoden in der Mechanik und Simulationstechnik
Ruhr-Universität Bochum
44780 Bochum, Germany
E-mail: Stefan.Mueller@ruhr-uni-bochum.de
E-mail: Heinz.Waller@num.bi.ruhr-uni-bochum.de

KEYWORDS
Simulation interfaces, interactive simulation, automatic control, internet access, java integration.

ABSTRACT

This contribution describes the integration of real-time hardware, a data analysis tool and web based services. The general idea is to use MATLAB as a pivot in a design and measuring and data processsing environment in connection with real-time hardware. The ML8-Toolbox integrates the powerful MATLAB capabilities with a Sorcus MODULAR-4 real-time control board. As it occured during the realisation of this project the hardware platform can be situated on a different location than the design computer. The easiest way to accomplish this task was to wrap a java library arround MATLAB's engine and hook up java with the apache web server.

REAL-TIME HARDWARE AND SIMULATION ENVIRONMENT

The available real-time hardware in our lab consists of Sorcus MODULAR-4 extension boards, which are plugged into ISA slots on a host personal computer. These boards contain a real time operation system running on a 80486 processor. Various external moduls can be attached to the board in order to add a greater number of analog or digital input-ouput devices to the system, which interact with the surrounding physical plants.

The software necessary to control a physical plant so far has always consisted of *two separate parts*. One is the *real time task* which measures all available input- and output data and which does all real-time number crunching (fig. 1). The second one resides as an *control application* on the hosting personal computer and does all the necessary loading, unloading, task controlling, hard disc operations, parameter handling and user interactions. There has not been any other possibility to connect to MATLAB than to use the hard disc and the above mentioned application to save the measured data in the appropriate data formats. Up to now it has not been possible to use the powerful computational engine of MATLAB *and* the real-time hardware interactively and intuitively as the users are used to conduct measurements, simulations and identifications.

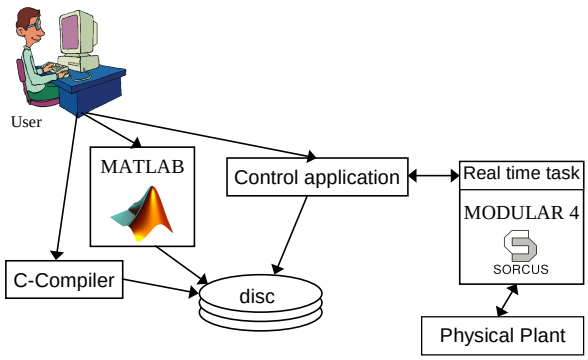

Fig. 1: Classical system environment

To conduct measurements and data analysis so far has been a step by step process. The first thing is to measure the data using the real-time control application and then save it to disc. Then load the data into MATLAB and work on it. As a next step it has been necessary to compute new settings for the real-time controller in MATLAB and to save all data to disc again. The control application has had to load the data again from disc and transfer it to the real-time task. Finally the whole process would start up again from the beginning.

This was a very tedious process. To overcome these limitations the ML8-toolbox was created.

ML8-TOOLBOX

MATLAB (MATLAB, 1997) is an extensable simulation environment. An application programm interface allows the user to write any program in C or Fortran and make it usable from within MATLAB. These extensions are called MEX-files and may connect to any dynamic link libraries and device drivers on a Windows95/NT platform.

We have programmed *132 MEX-files* and one *SIMULINK block* in detail to connect MATLAB and the virtual device driver of the real time hardware (fig. 2). All functions of the device drivers can be used directly at the prompt in MATLAB or from any m-file or script.

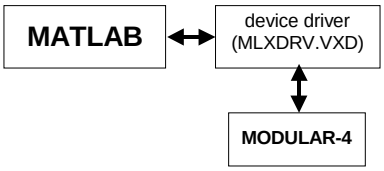

Fig. 2: MATLAB - real-time hardware

The mentioned *control application* is no longer needed. The toolbox has fully replaced the control application, all commands, settings, data handling and task controlling can be conducted from within MATLAB's workspace (fig3.).

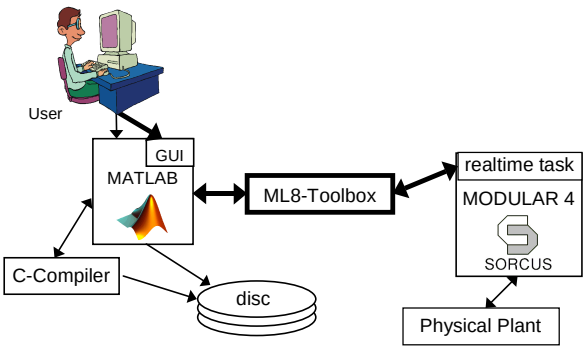

Fig. 3: Integrated environment

An additional DDE-interface (Windows messaging service) makes it possible to connect to the real-time driver from any capable application (e.g. office appications). Since the real-time hardware can be hooked up with SIMULINK one can perform hardware-in-the-loop-simulations, where one part is a real physical plant and the other part is only a simulation.

Now one can use all available GUIs in MATLAB (fig. 4) to conduct real-time experiments or simulations with respect to the Sorcus MODULAR-4 real-time hardware and the powerful MATLAB simulation environment.

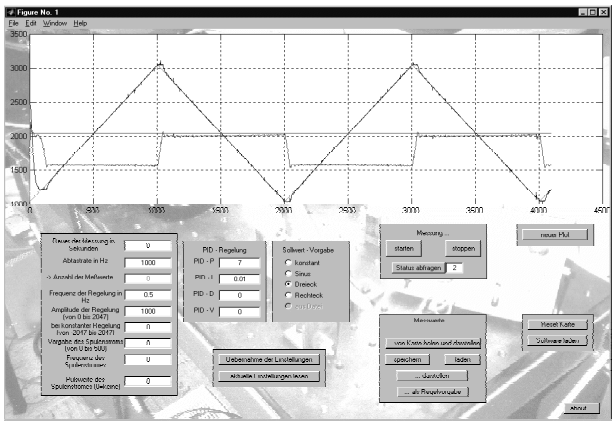

Fig. 4: MATLAB GUI for plant control

An additional java library called JMatLink (Müller, 1999) was developed, which allows users to connect to MATLAB from java applications as well as from java enabled web-servers and servlets.

WEB-SERVER INTEGRATION

In many technical applications the physical plant is located at a remote location and the user or operator is far away from it.

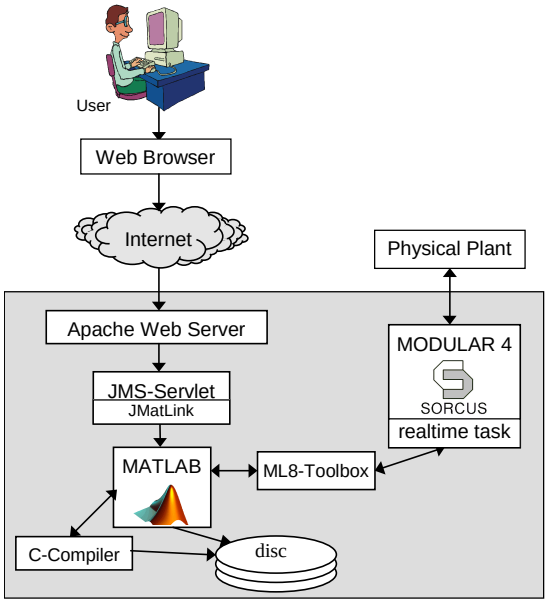

Fig. 5: System setup for internet usage

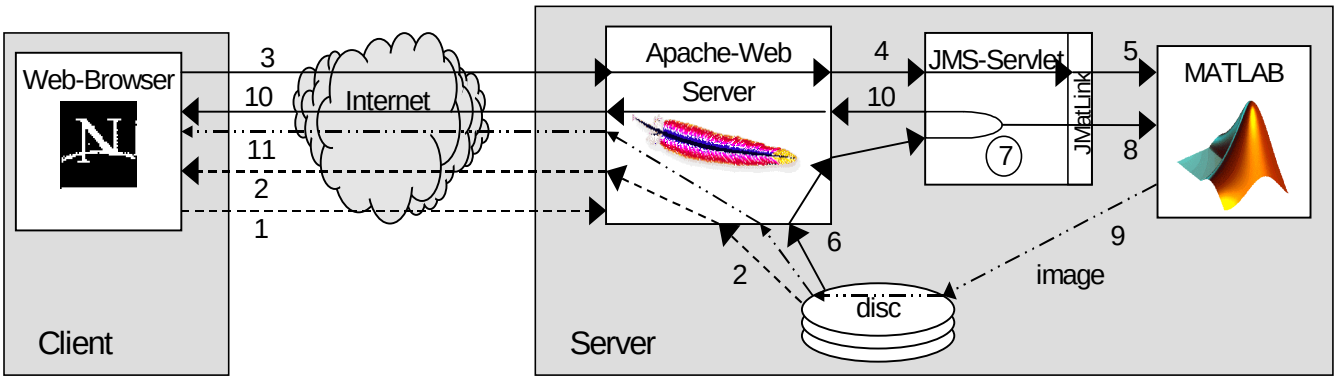

Fig. 6: data flow between client – server - MATLAB

To solve this problem the apache web server (Apache, 1999) for Windows95/NT with its java extension (Apache JServ, 1999) was used. A java servlet (Hunter, 1998) was developed which uses *form data* sent over the internet to connect to MATLAB's engine with the use of the JMatLink library (fig. 5). In connection with the ML8-Toolbox and the real-time hardware any physical plant with the appropriate interfaces can be controlled remotely over the world wide web now.

The following paragraphs will give a detailed description of the process of obtaining data from MATLAB through a web server. Fig. 6 is used to show the paths of the flowing data.

The client, which can be any advanced web browser, requests (1) an HTML-file from the web server. The web server will fetch the file from its discs and send it back (2) to the client over the internet (dashed lines). This is the common process for all world wide web accesses. The real interesting part is coming next.

```
<html>
...
<form action="/servlets/jms"
      method=get>
<input type=text name=engEvalString
       size=50   onEnter=submit()>
<input type=hidden name=outputFile
       value="matframe.html">
<input type=submit
value="EvalString">
</form>
...
</html>
```

Fig. 7: html-input file

The received html-file contains html forms which set variables to trigger certain actions later on, see fig. 7.

Figure 8 shows the appearance of this code in the web browser. This form data is sent back (3) to the web server. On the web server the data will be forwarded (4) to a java servlet. The *JMS servlet* checks on all available commands one by one. The first time the servlet is loaded during a server session, it will start MATLAB and establish a connection to MATLAB's engine through the JMatLink java library. This connection is static and persistent during the whole servlet life cycle.

Fig. 8: Appearance in browser window

The servlet sends all commands to MATLAB (5) for evaluation. Even the use of graphics is supported. More about graphics will follow later on.

The servlet can sent an instantaneous html-code back to the client or, as a more complex solution, it can ask the web-server to fetch an *.jhml file* from its discs (6), see fig. 9. The suffix .jhtml indicates to the server that the html file contains so called *server side includes*. These are additional commands which are included in the code and which are processed before the page is sent back to a client.

```
<html>
...
<servlet code="jms">
  <param name="figure" value=1>
  <param name="height" value=300>
  <param name="width"  value=400>
</servlet>
...
</html>
```

Fig. 9: .jhtml-ouput file on disc

This web server will parse the .jhml file and process all the servlets it contains (7) before sending back the output (10). The generated output may consist of variables, arrays or text output, which are requested from MATLAB (8). The `<servlet>`-tags in the .jhtml file are replaced by that output. As mentioned before graphics can be used, too. A figure will be saved to disc as a jpeg-image file und its location will be in the web servers document tree, where it can be requested by the client later on. All different figures produced by MATLAB can be requested. The generated file names will be unique. A special html `<img>`-tag containing a random filename will be included in the servlets response, see fig. 10.

```
<html>
...
 <image src=5x40369.jpeg
        height=300 width=400>
...
</html>
```

Fig. 10: jhtml-output file in brower

When the client receives the html coded output file it finds an `<img>`-tag with the aforementioned random image filename and it requests this image from the server (11) (dashed double dotted line). The reason for the random filenames is to make sure that the client or a proxy between client and web server will not send an image from its caches under any circumstances. It forces them always to request new image data from the server and hinders them to show older images, since all filenames are unique and random.

With the use of all those new features it is even possible to create a MATLAB GUI like the one presented in figure 4, but one written for a browser in the html language, see figure 11.

Some security considerations will have to be taken into account in a future version. Right now the implementation of the servlets makes it possible to execute all code on the host computer. MATLAB should run in a secure shell and access to the servlet should be restricted to users on a password basis in order to compensate for the security hazards.

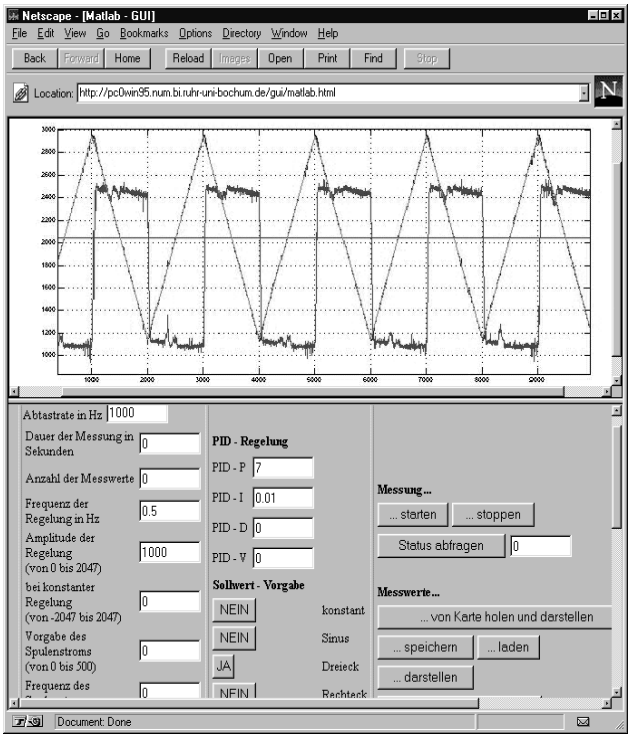

Fig. 11: GUI in html showed inside browser

CONCLUSION

This project shows how to integrate real time hardware and physical plants into MATLAB as an integrated simulation *and* data processing application. All data can now be accessed over the World Wide Web. It is also possible to conduct simulations *and* control a plant over the internet. Since only standard components and java are used, the described web integration should compile on all systems that support MATLAB, java and the apache web server.

Up to now only static data can be sent back to the client. Using MATLAB's capability to save 3D-graphs as VRML models, these could be sent to the client, too. For real time data the solution appears to be more difficult, e.g. a SIMULINK

block could be created which connects to a java virtual machine, which in turn opens up a TCP socket in order to send streaming data back to an applet on the client.

These tools also offer the advantage to hide the complexity of MATLAB from a potential user. This feature could be used to create learning material for dynamic system lectures (anything that can be processed by MATLAB) where the control theory is the object of study, rather than the command sets of a simulation tool.

REFERENCES

Apache Consortium. 1999: Apache Webserver Documentation: http://www.apache.org.

Apache JServ: 1999. Apache Webserver Java Servlet. http://java.apache.org

Hunter, J.: 1998. *Java Servlet Programming*. O'Reilly & Associates, Sebastopol, CA.

Müller, S.: 1999. JMatLink library. http://homepage.ruhr-uni-bochum.de/Stefan.Mueller/JMatLink/.

MATLAB: 1997. MATLAB Version 5.2. TheMathworks Inc., Natick, MA, USA.

Sorcus. GmbH: 1997. *MODULAR-4®/486 Programmierhandbuch*. Real-time libraries and API-reference.

BIOGRAPHY

Stefan Müller received the degree „Diplom Ingenieur" from the faculty of electrical engineering of the Ruhr-Universität Bochum, Germany in 1997. Since 1997 he is a Ph.D. scholar in mechanical engineering of the „Arbeitsgruppe numerische Methoden in der Mechanik und Simulationstechnik" of the Ruhr-Universität Bochum, Germany. His main interests include semi-active damping control of mechanical structures and web-based services combined with MATLAB.