Semi-Automatic Generation of Web-Based Computing Environments for Software Libraries

Pedher Johansson[1] and Daniel Kressner[2]

[1] Department of Computing Science, Umeå University, SE-901 87 Umeå, Sweden
pedher@cs.umu.se
[2] Institut für Mathematik, MA 4-5, Technische Universität Berlin, D-10623 Berlin, Germany
kressner@math.tu-berlin.de

Abstract. A set of utilities for generating web computing environments related to mathematical and engineering library software is presented. The web interface can be accessed from a standard world wide web browser with no need for additional software installations on the local machine. The environment provides a user-friendly access to computational routines, workspace management, reusable sessions and support of various data formats, including MATLAB binaries. The creation of new interfaces is a straightforward process. All necessary web pages are automatically generated from XML description files. The integration of the control and systems library SLICOT demonstrates the efficacy of this approach.

1 Introduction

Highly reliable and efficient library software is of particular importance for sophisticated engineering solutions. However, there is a gap between the number of existing mathematical routines and those actually used by engineers. A main obstacle for potential users is that, in order to benefit from new software, they typically have to go through the painful process of searching, downloading, installing and understanding. This requires a substantial amount of time; often even *before* the usefulness of the software can be evaluated. The user may moreover not have access to computing facilities and proprietary software.

We have developed a new collection of utilities, referred to as the *Web Computing Utilities* (or briefly, *Webcut*), that addresses these concerns. It provides a solution in the following way. The programmer can make new mathematical and engineering software available *on the web* where it is accessible and executable through standard web browsers. The programmer provides information about the routine parameters and standardized calling routines in so called description files. From this input, Webcut automatically generates HTML pages offering a user-friendly web computing environment. The essential prerequisites of software users are then, to know the type of the problem to be solved and to provide the input data in a convenient way. The use of web computing does not require any installation of software on the local computer nor does it require any documentation besides that which is integrated into the user interface.

This work is related to a wide variety of projects providing easy access to mathematical software. Among those, the most popular and successful is certainly MATLAB[1] [1]. There exists a web environment for MATLAB [2], its interfacing functionalities, however, are very limited. Others are ATLAS [3], JSPICE [4], MMM [5], Paraweb [6] and WOS [7]; those facilitate Java applets. We feel that such applets unnecessarily limit the range of admissible browsers or devices and thus preclude their use in Webcut. The network-computing system PUNCH [8] is possibly the most closely related project. Here, interfaces are generated by HTML templates quite similar to the XML strategy described in Section 3. While being more mature in general, PUNCH lacks a sound workspace management, i.e., an easy way for the user to administrate and reuse input / output data among several computational tasks. It should be noted that Webcut is not aimed to be a tool for accessing and using globally distributed resources. Although it could be coupled with projects like NetSolve [9] the objectives of this work are focused on handy interfaces.

The paper is organized as follows. Section 2 gives a short description of the web computing environment, from a user's point of view. This includes presentation of the major functionalities and a step-by-step illustration of a computing session. In Section 3, we describe the way in which the programmer has to provide information about the routines to be implemented. Details about the internal design are given in Section 4. Conclusions and future work are presented in Section 5, mainly to show how the software library SLICOT [10, 11] benefited from this work and how other libraries could benefit, too.

2 A User's Point of View

The web computing environment can be accessed from a standard web browser, with no need for additional software installations on the local computer.

In the following, we illustrate how the environment can be used to solve a sample problem, namely to compute the circular convolution of two signals

$$c_k = \sum_{j=1}^{n} a_j b_{k+1-j}, \quad k = 1, \ldots, n \; . \tag{1}$$

The vectors defining the problem can be entered by the user in the web interface. However, default values are provided so that computations can be performed without the need for the user to define an own problem. Input data can also be previously used data or be uploaded from MATLAB binary or plain text files. The underlying software is the SLICOT routine `DE01PD` based on discrete Hartley transforms [12].

Figure 1 shows the user interface for the routine `DE01PD`. Here, the vectors have already been entered and the user may choose between computing the

[1] MATLAB is a registered trademark of The MathWorks, Inc.

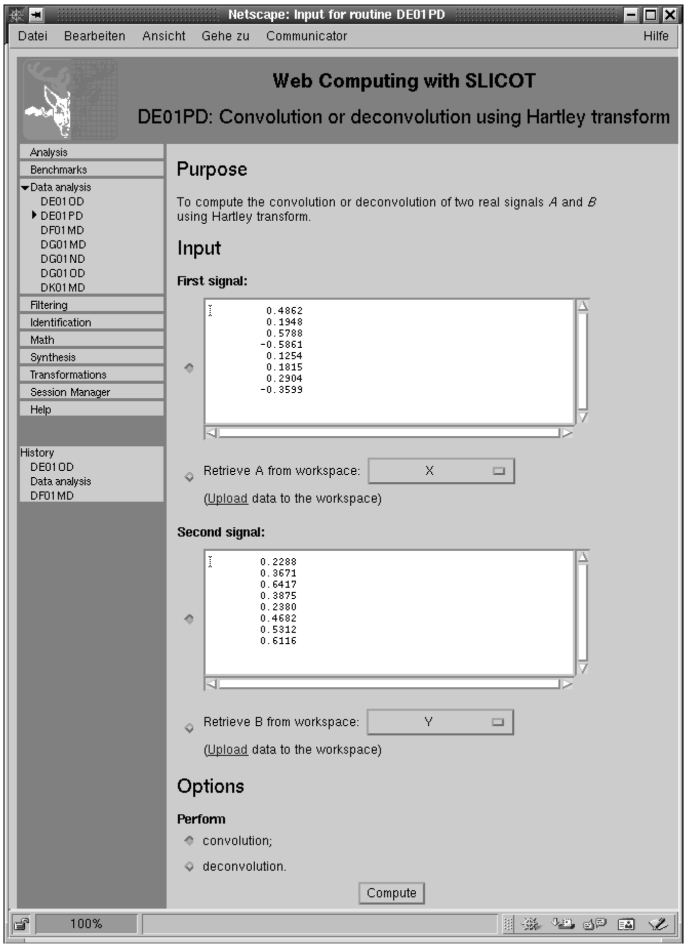

Fig. 1. Web interface for computing the convolution of real signals.

convolution or its inverse function, deconvolution. This is a simple example with only two input parameters and one option. With more complex computational tasks, the number of input parameters and options can be much larger, including scalar parameters and multiple choice lists.

After pressing the compute button in the window of Fig. 1, the convoluted signal is presented as in Fig. 2. It is also possible to download the input/output data as plain text files or MATLAB binaries. During the web computing session, matrices are stored in a workspace, making the data from previous tasks available as input in subsequent computations. For example, the discrete Fourier transform of the signal c can be computed using the web interface of the SLICOT routine `DG01ND`.

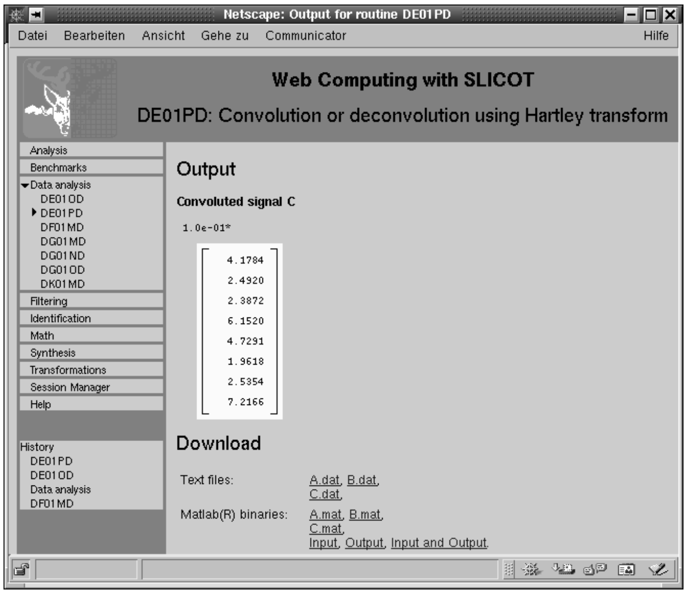

Fig. 2. Convoluted signal and possibilities for downloading data.

A convenient interface enables administrating this workspace, as illustrated in Fig. 3. One of the features of the session manager is that users can login and reuse data of previous computations. All information is stored on a remote machine so that data computed in Umeå, Sweden could easily be recovered, e.g., in an Internet cafè in Berlin.

3 A Programmers Point of View

Apart from being user-friendly it is important to keep web computing facilities as programmer-friendly as possible. The motivation for putting routines on-line will remain low if software producers have to struggle with HTML dialects and CGI techniques. Our approach only assumes the access to a web server that is able to parse the HTML-embedded scripting language PHP and to call executable binaries. The integration of new routines in the environment is then a process divided into two parts.

External routines to be included are first formalized in an XML description file. The syntax is well-defined and deals with a wide range of attributes of the routine, e.g., parameters, conditions, options, default values, description of the routine and so on. When dealing with software libraries which have very restrictive in-line documentation standards the production of the description file could be automatized.

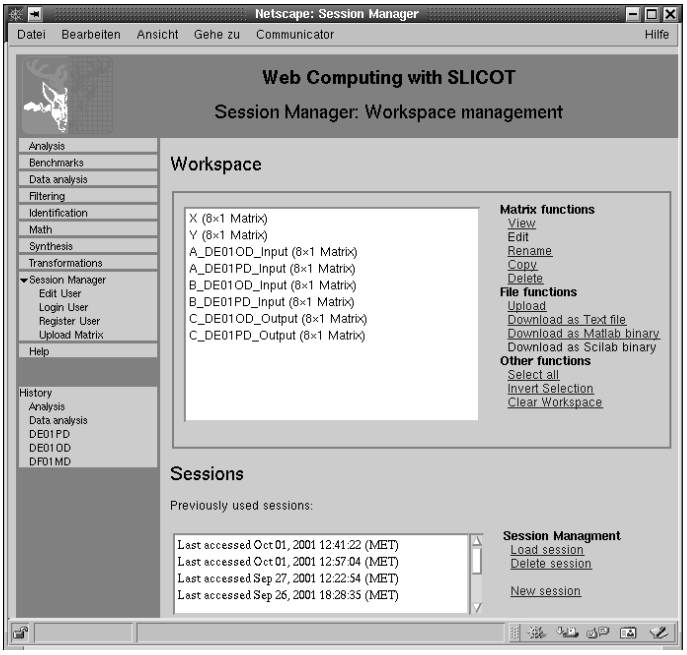

Fig. 3. The workspace manager.

The XML-format currently used in Webcut differs slightly from the format described below. Considerations concerning generality and reusability made it necessary to develop a modified version, together with improved parsers.

For the example from Section 2, the convolution of two signals, the description in the reviewed format is shown below.

```
<routine name = "DE01PD" supgroup = "DE - Covariances">
  <description>
    Convolution or deconvolution using Hartley transform
  </description>
  <parameters>
    <matrix name = "A">
      <description>First signal:</description>
      <default>
        \t0.4862\n\t0.1948\n\t0.5788\n\t-0.5861\n
        \t0.8254\n\t0.1815\n\t0.2904\n\t-0.3599
      </default>
    </matrix>
    <matrix name = "B">
      <description>Second signal:</description>
      <default>
        \t0.2288\n\t0.3671\n\t0.6417\n\t0.3875\n
```

```
      \t0.2380\n\t0.4682\n\t0.5312\n\t0.6116
    </default>
  </matrix>
  <optionlist name = "conv">
    <description>Perform:</description>
    <option name = "C" description = "convolution;"/>
    <option name = "D" description = "deconvolution."/>
    <default> C </default>
  </optionlist>
  <functional type = "length">
    <argument> A </argument>
  </functional>
</parameters>
<conditions>
  <condition type = "is_vector">
    <argument> A </argument>
  </condition>
  <condition type = "is_vector">
    <argument> B </argument>
  </condition>
  <condition type = "equal_length">
    <argument> A, B </argument>
  </condition>
</conditions>
</routine>
```

The top entry `routine`, specifies the name and description of the routine. It contains specifications of name, location and parameters to the callable routine together with possible conditions on those. Restricting parameters is convenient, especially with routines that are not designed for parameter control and may therefore crash without any further information. More about the currently used syntax and semantics of the description files can be found in the documentation accompanying Webcut [13].

The description files are then used by a parser, currently implemented in Perl, that regenerates and modifies the environment to include new routines or modify old ones. An important aspect is that each description file contains exactly the information related to one routine. There are no relations to other parts of the environment so that modifications in single routines keep local.

One weakness with our choice of platform for the server, In the current version, most added routines need an additional routine that handles the parameters in a for the Webcut compatible manner.

Data transfer follows at the moment a simple strategy; input data is supplied to the standard input stream and output data is read from the standard output stream. Hence, the input/output behavior of calling routines must satisfy a well-defined standard.

4 Implementation Details

The implementation of the Webcut environment is mainly done with PHP, a server side hypertext preprocessing language. PHP is a suitable choice for writing dynamic web content and it is intuitive and easy to integrate into existing HTML. Furthermore, a crucial aspect for our project, data bases are well supported. The overall design of the web computing environment is in a schematic way described in Fig. 4.

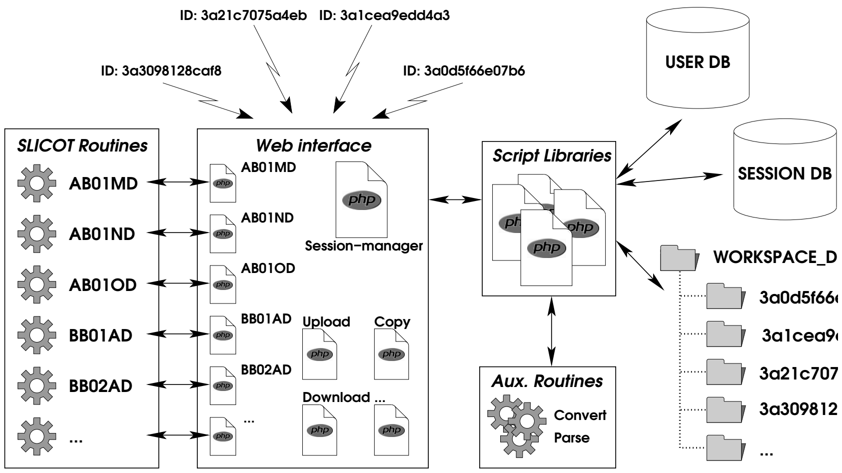

Fig. 4. A schematic overview of the Webcut design.

Within Webcut, several PHP libraries have been developed for the tasks performed by the environment. The following parts are provided.

Layout rendering. Functions that render HTML code and thereby the layout of Webcut are gathered in one library. The structure of the library makes it easy to modify parts or the entire layout.

Session Identification. Contains functions that interact with the browser and maintain the identification of the user during one session.

Invocation and interaction with external routines. Here, the execution of external routines is handled together with the passing of input parameters and catching output data.

Session, user and workspace management. The functions of this library take care of user, session and workspace data by providing high level access to the corresponding data bases.

Based on these libraries, a wide range of user-interactive pages has been developed. Those include workspace and user management, interaction with MATLAB and so on. The main part however consists of pages that call external routines.

As above described in Section 3, the PHP files are generated from the XML-description files, and included into the environment. Since they are based on the developed libraries it is easy to change their layout or functionality.

The session life time starts when a web interface is opened for the first time. The user is then provided with a unique ID that links her/him with a new session. The session itself consists of information about the user and a workspace for produced or uploaded data. This session is accessible as long as the user's web browser keeps track of the provided ID. Since there is a time limit, the ID expires after a period of inactivity. Registered users can restore previously used sessions at any time and at any place.

5 Conclusions and Future Work

In this paper we described tools which automatically generate web computing environments from given description files and calling routines. The resulting interfaces are intuitive to use and provide a sound workspace and session management.

The SLICOT library consists of about 250 user callable routines and benchmark collections in various domains of systems and control. Webcut has been used to provide large parts of this library with web interfaces. For example, Riccati benchmark collections [14] as well as solvers for the algebraic Riccati equation [15] can be tested on-line. The SLICOT web computing project can be found under `http://wc2.hpc2n.umu.se`.

Future developments will concentrate on the coupling of Webcut with grid computing environments. The web interfaces will enable users to direct computations to a heterogeneous set of computers. A major step to be taken is to improve the input/output data management. For instance, the data transfer between computational routines and the web server must be generalized. It is further planned to apply Webcut to other software libraries relevant to the engineering community. Of particular interest is software designed for parallel computations like ScaLapack [16] or PSLICOT [17].

6 Acknowledgments

The authors wish to thank Erik Elmroth, Bo Kågström and Volker Mehrmann for not only taking initiative for this project, but also for useful discussions and help throughout this work. We are also grateful to participate in the NICONET project [18], which develops and distributes the SLICOT library [10], and to the High Performance Computing Center North (HPC2N) [19] which provides the web computing server resources.

References

[1] The MathWorks, Inc.: MATLAB User's Guide. Natick, MA (1992)

[2] The MathWorks, Inc.: MATLAB web server, version 1.2.1. (2001) http://www.mathworks.com/products/webserver/

[3] Baldeschwieler, J., Blumofe, R., Brewer, E.: ATLAS: An infrastructure for global computing. Proceedings of the Seventh ACM SIGOPS European Workshop on System Support for Worldwide Applications (1996)

[4] Souder, D., Herrington, M., Garg, R. P., Deryke, D.: JSPICE: a component-based distributed Java front-end for SPICE. Concurrency: Practice and Experience **10** (1998) 1131–1141

[5] Günther, O., Müller, R., Schmidt, P., Bhargava, H. K., Krishnan, R.: MMM: A web-based system for sharing statistical computing modules. IEEE Internet Computing **1** (1997) 59–68

[6] Brecht, T., Sandhu, H., Shan, M., and Talbot, J.: ParaWeb: Towards word-wide supercomputing. Proceedings of the Seventh ACM SIGOPS European Workshop, Connemara, Ireland (1996) 181–188

[7] Kropf, P. G.: Overview of the web operating system (WOS) project. Advanced Simulation Technologies Conference, San Diego, CA (1999) 350–356

[8] Kapadia, N., Fortes, J.: PUNCH: An architecture for web-enabled wide-area network-computing. Cluster Computing **2** (1999) 153–164

[9] Arnold, D., Agrawal, S., Blackford, S., Dongarra, J., Miller, M., Sagi, K., Shi, Z., Vadhiyar, S.: Users' Guide to NetSolve V1.4. Computer Science Dept. Technical Report CS-01-467, University of Tennessee, Knoxville, TN (2001)

[10] Benner, P., Mehrmann, V., Sima, V., Van Huffel, S., Varga, A.: SLICOT—a subroutine library in systems and control theory. Applied and computational control, signals, and circuits, Vol. 1, Birkhäuser Boston, Boston, MA (1999) 499–539

[11] Elmroth, E., Johansson, P., Kågström, B. Kressner, D.: A web computing environment for the SLICOT library. The Third NICONET Workshop on Numerical Control Software (2001) 53–61

[12] Van Loan, C. F.: Computational frameworks for the fast Fourier transform. Society for Industrial and Applied Mathematics, Philadelphia, PA (1992)

[13] Johansson, P. and Kressner, D.: Webcut - a documentation. Preliminary version available from http://wc2.hpc2n.umu.se (2001)

[14] Abels, J. and Benner, P.: CAREX - a collection of benchmark examples for continuous-time algebraic Riccati equations (version 2.0). SLICOT working note 1999-14, WGS (1999)

[15] Laub, A. J.: A Schur method for solving algebraic Riccati equations. IEEE Trans. Automat. Control, **24** (1979) 913–921

[16] Blackford, S., Choi, J., Cleary, A., D'Azevedo, E., Demmel, J., Dhillon, I., Dongarra, J., Hammarling, S., Henry, G., Petitet, A., Stanley, K., Walker, D., Whaley, R. C.: ScaLAPACK Users' Guide. Society for Industrial and Applied Mathematics, Philadelphia, PA (1997)

[17] Benner, P., Quintana-Ortí, E. S., Quintana-Ortí, G.: PSLICOT routines for model reduction of stable large-scale systems. The Third NICONET Workshop on Numerical Control Software (2001) 39–44

[18] NICONET. Numerics in Control Network. http://www.win.tue.nl/wgs/niconet.html

[19] HPC2N. High Performance Computing Center North, Umeå University, Sweden. http://www.hpc2n.umu.se.