Real-Time Prototyping of Matlab-Java Code Integration for Water Sensor Networks Applications

S. Roubakis
Institute of Computer Science
FORTH
Heraklion, Greece
roub@ics.forth.gr

G. Tzagkarakis
Institute of Computer Science
FORTH
Heraklion, Greece
gtzag@ics.forth.gr

P. Tsakalides
Institute of Computer Science
FORTH
Computer Science Department
University of Crete
Heraklion, Greece
tsakalid@ics.forth.gr

***Abstract*—Industrial applications typically necessitate the interaction of heterogeneous software components, which makes the design of an integrated system a demanding task. Specifically, although Matlab® and Java are among the most commonly used programming languages in industrial practice, with each one offering its own advantages, however, their integration for real-time code prototyping is not straightforward. Motivated by this problem, this work proposes an efficient method based on the use of sockets to integrate Matlab and Java code for designing a data processing platform tailored to smart water sensor networks scenarios. The performance of the proposed approach is evaluated on two distinct tasks, namely, the recovery of missing values and the temporal super-resolution from streaming data. Experimental evaluation with real pressure data reveals the superiority of our methodology, in terms of reduced execution times, when compared against two well-established alternatives, namely, the use of standalone applications using input-output files for executing Matlab code in Java-based environments and socket-based solutions implemented directly in a Matlab environment.**

***Index Terms*—Matlab-Java code integration, client-server model, real-time prototyping, water sensor networks**

I. Introduction

One of the most demanding tasks in industrial practice is the integration of heterogeneous software components for the design of data processing platforms based on real-time code prototyping. The SmartWater2020 project[1] is a characteristic example of an R&D project, tailored to an industrial application, necessitating the close synergy between the involved partners, who develop the individual software modules in different programming languages. More specifically, SmartWater2020 aims at modernizing four water supply organizations in Cyprus and Crete through the development of intelligent software and smart water network monitoring devices for detecting leakages and problems in the quality of water, as well as the implementation of an innovative system for controlling the network pressure in real time.

This work is funded by the Interreg V-A Greece-Cyprus 2014-2020 programme, co-financed by the European Union (ERDF) and National Funds of Greece and Cyprus, under the project SmartWater2020.

[1]SmartWater2020, Interreg V-A Greece-Cyprus 2014-2020 programme, https://www.smartwater2020.eu

In particular, the individual components of the platform are developed in Matlab® and Java. Although they are among the most commonly used programming languages in industrial practice, with each one offering its own advantages, however, their integration for real-time code prototyping is not straightforward. The typical approach to combine the two languages is based on the generation of standalone applications.

Despite the importance of Matlab-Java code integration for real-time prototyping, only a few studies are publicly available, providing basic information about how to combine these two programming languages efficiently in practical applications [1]–[5]. Motivated by the above limitations, this work proposes an efficient alternative method for integrating Matlab and Java codes, based on the use of a *client-server model* [6], [7]. This approach enables an efficient distributed communication framework between two or more independent participants, who support two-way communication.

The performance of our proposed approach is evaluated on two distinct problems, that are also among the key functionalities supported by our developed data processing platform: (i) recovery of missing values in streaming data, and (ii) artificial increase of the temporal resolution for a received data stream. More specifically, the use of wireless sensor networks poses issues and challenges to the system's design and topology. Any malfunction of the equipment, or the presence of errors during transmission often yield missing measurements, that should be recovered prior to performing a high-level data analysis and/or statistical inference. Furthermore, the sampling frequency (or, equivalently, temporal resolution), which is associated with the telemetry costs, is often different from the frequency that is necessary to process the data with high accuracy, thus enhancing the reliability of the system. To this end, we increase artificially the temporal resolution of a given data stream, by reducing the problem of temporal super-resolution into a problem of missing data recovery, where the missing entries are introduced in a structured way in the received data stream. Both functionalities are important for water resource management applications, where leak detection is a critical task, and the reaction time is a crucial issue to guarantee the system's robustness.

978-9-0827-9703-9/19/$31.00

In summary, the contribution of this paper is twofold: (i) we describe in detail an efficient cost-free way to integrate Matlab and Java for real-time applications by exploiting the technology of sockets, and we provide a comparison with the well-established counterparts, revealing the high performance of our proposed method; (ii) we evaluate the performance of the proposed code integration method in two distinct problems arising in a real industrial use case, namely, the recovery of missing data and the temporal super-resolution from real pressure data recorded by a smart water network.

The rest of the paper is organized as follows: Section II overviews the typical approach for integrating Matlab and Java codes. Our proposed socket-based integration method is analyzed in Section III, and its performance is evaluated and compared against its competitors in Section IV. Finally, Section VI summarizes the main outcomes of this work and gives directions for further extensions.

II. Conventional Matlab-Java Integration Approach

This section overviews the typical approach for integrating Matlab and Java codes, which is primarily based on the concept of generating standalone applications combined with input-output file operations.

A. Matlab Standalone Applications

The distribution of a Matlab system is often hampered by the dependencies it creates. As such, the installation and license payment of a, more or less, common version of Matlab was initially required, so as to be able to exchange and integrate the modular software components of a system written by several partners. The Matlab compiler (MC) was introduced to overcome this limitation by producing Matlab standalone applications (MSA) that do not rely on such dependencies. These applications include all the necessary pre-written and built-in functions, with the MC producing an installer that installs both the application and all the required dependencies on a target system. Functions can be called with their usual parameters, but they need to be converted to the desired type before using them internally. The main advantage of an MSA-based design is that it can be used by systems where there is no pre-installed version of Matlab or related licenses.

B. Java Standalone Applications

Similarly to Matlab, Java allows the generation of Java standalone applications (JSA). Specifically, any Java class including a `main` method can produce a specific .jar file that can be executed independently on a target machine. The only requirement for execution is the existence of a compatible version of Java Runtime Environment (JRE). These systems can accept parameters, output the results on the JVM console, as well as generate graphical interfaces that enable the user to interact with the system. The main limitation of a Java-based environment is the lack of advanced mathematical libraries for facilitating complex scientific calculations. This drawback can be eliminated either by writing from scratch a Java version of our numerical algorithms or by integrating Java with a programming language which is better suited for scientific calculations, as is the case of Matlab.

C. Matlab-Java Integration

As mentioned above, there is often a need for interconnecting systems or parts of them that have been developed in different programming environments. Focusing on Matlab and Java integration, this can be accomplished in several ways. Matlab provides built-in support for loading Java classes, accessing fields, and invoking functions. Correspondingly, a Java-based system can invoke Matlab methods synchronously or asynchronously through the provided application programming interface (API). However, the response of such an architecture is typically prohibitive for designing systems that rely on streaming data. An alternative way is to use an MSA that integrates the functionality we want to exploit. However, the use of such an application requires the creation of a new process for execution.

Simply embedding the different parts that have been developed in the two programming environments does not meet the needs of a streaming application. At the same time, more efficient integration methods, such as the use of Matlab builder JA [4], are paid solutions. Thus, for the specific application, it is convenient to separate the two implementations into independent applications and focus on the communication between them. This architecture enhances the agility of the participants as the independent Matlab application will be the common center of advanced mathematical calculations, while each participant will be able to design its own data collection and visualization system customized to its own needs.

These two independent entities of the information producer and processing system can run locally on a machine. The usual way of communication is through input/output (I/O) operations on files. The data and parameter values are written by one entity in the appropriate form and used by the other. After each operation, it is necessary to consider deleting the files involved. Fig. 1 shows the steps for the temporal super-resolution functionality, which is applied on one-dimensional time series written in a single file. This file is the input to the standalone application for the associated functionality. The resulting output is written in an output file which, when used, is deleted along with the input file. On the contrary, the recovery of missing values from tensor-structured data is performed on three-dimensional arrays involving multiple time series. Fig. 2 shows the corresponding execution pipeline, which requires writing in multiple files for calling the function, as well as the production of multiple output files. Both the input and ouput files are deleted after the operations are completed.

III. Proposed Socket-based Matlab-Java Integration Method

The restrictions of I/O communication between standalone applications can be overcome by using a different architecture. Concerning the two targeted applications, namely,

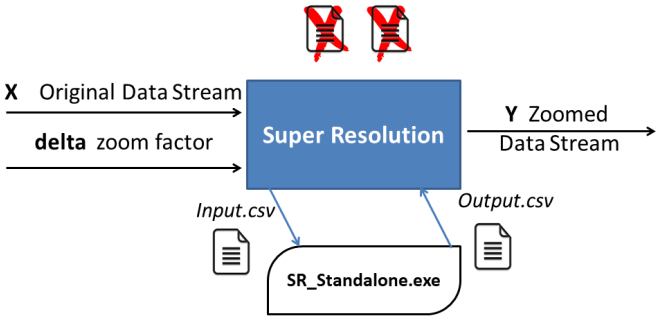


Fig. 1: I/O file-based Matlab-Java code interaction for temporal super-resolution.

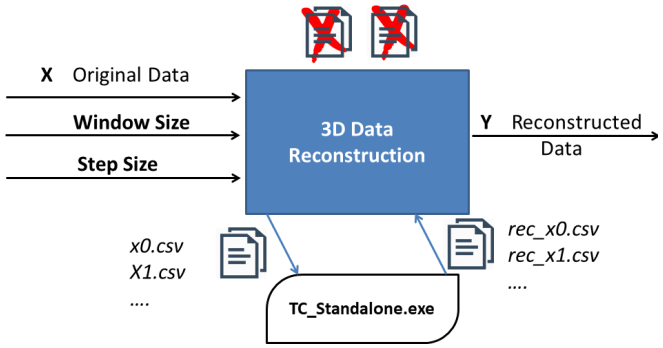


Fig. 2: I/O file-based Matlab-Java code interaction for recovering missing values from tensor-structured data.

tensor completion and temporal super-resolution, they can be implemented in both Java and Matlab environments. However, for precise and complex calculations, it is preferable to use the Matlab implementations of the algorithms, which avoids their re-writing in Java. As existing systems of participants consist of Java-based components, it is more efficient to implement and incorporate a component, which will be responsible for calling the Matlab functions to be employed.

Specifically, our approach is based on the *client-server model*. In this model, there are two entities for the purpose of allocating a resource or providing a service upon a request. The entity that makes the requests is called the client, while the one that provides a service is called the server. The server waits until a request arrives to serve it. As long as the result of the request is calculated, the client is blocked. Then, the server responds to the request of the client, which unblocks and receives the result. Clients and servers are often deployed on separate hardware and communicate via a computer network. However, they can also be implemented in the same system.

In our application, the server entity is assigned to a Matlab implementation of the functions, whilst the client entity and the rest of the system are implemented in Java. The client-server models must define their communication protocol, that is, the set of rules governing the exchanged messages. The two entities run on the same system and their communication is performed through Transmission Control Protocol (TCP) sockets. TCP sockets were selected against the User Datagram Protocol (UDP) due to the guarantees they provide in sending and receiving data as a connection-oriented technology. Messages exchanged through these sockets are in the form of strings and maintain a stable structure. For the client's requests the structure is as follows: [function-id, size, data], where "function-id" is the identifier of the function we want to run on the server, "size" is the length of the data stream after the increment of its temporal resolution or the length of each data stream that forms the tensor, and "data" is the representation of measurements as comma-separated strings for both functions. When the operation is completed, the values are written in the form of a string and sent to the client.

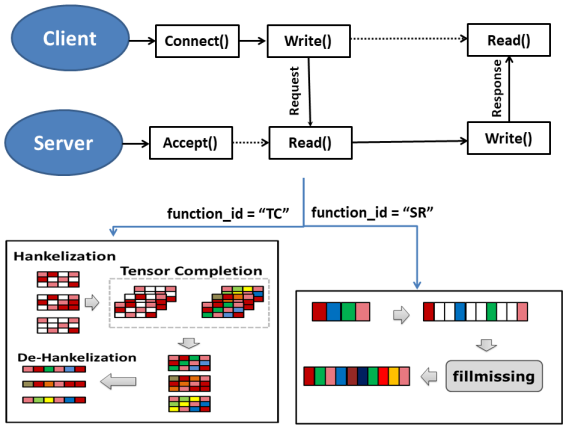


Fig. 3: Proposed socket-based Matlab-Java code interaction for (i) recovering missing values from tensor-structured data (TC problem), and (ii) increasing the temporal resolution (SR problem).

Fig. 3 shows the proposed architecture based on the server-client model for (i) recovering missing values from multiple sensor streams (TC problem) and (ii) increasing the temporal resolution of a single time series (SR problem). Focusing on the more complex TC problem, the first step is to establish the connection between the two entities based on the TCP protocol. Then, the client sends a request to the server with the structure described above. Although the data streams are obtained from the pressure sensors in a one-dimensional structure, the data recovery problem is solved more efficiently by converting the one-dimensional time series into third-order tensors. To this end, a Hankelization function is applied, where by using a sliding window we convert vectors to matrices, which are second-order data structures. Combining multiple Hankelized matrices we create the three-dimensional tensor to be completed. Next, the reverse processes are carried out (i.e., separating the tensor slices and applying a de-Hankelization process) such that the completed time series are sent from the server to the client as a response to the preceding request. Notice that the dashed lines in the figure symbolize the time intervals where the entities are blocked and expect some message to awake them based on the TCP protocol, thus synchronizing the communication between them. A similar approach can be followed to execute the SR function for temporal super-resolution of single data streams.

IV. Performance Evaluation

In this section, the performance of our proposed socket-based approach using standalone Matlab and Java applications, hereafter called "Standalone Sockets", is evaluated and compared against two alternatives, namely, (i) the use of standalone applications based on I/O operations on files, hereafter denoted by "Standalone Files", and (ii) a socket-based execution of the code via the Matlab's environment, hereafter called "Matlab Sockets". Although the second case is expected to yield the optimal performance, however, it necessitates the use of a paid license for using Matlab, thus increasing the expense for deploying an integrated system. On the other hand, the first case is expected to result in higher execution times, due to the use of read/write/delete operations on data files. Ideally, we would like the performance of our approach, which does not depend on paid software, to approximate as much as possible the performance of "Matlab Sockets".

As mentioned already, the above three Matlab-Java integration methods are evaluated in two distinct problems, namely, (a) the recovery (a.k.a completion) of missing values in tensor-structured data (TC), and (b) the temporal super-resolution (SR) of one-dimensional time series. More specifically, the Matlab toolbox TMac[2] is used to solve the TC problem, whilst the built-in Matlab function `fillmissing` is employed to perform SR using a piecewise cubic spline interpolation. The necessary standalone applications are extracted in both cases. Furthermore, both problems are solved for a set of real pressure data collected by a water distribution network, in the framework of the SmartWater2020 project.

More specifically, according to the project's specifications, the data is examined once a day and the sampling period of the sensors is fixed at 15 minutes. We examine three different window sizes w for the received data, $w \in \{256, 512, 1024\}$. For the TC problem, a Hankelization process is performed first, to express the received time series data in a matrix form, as it is required by TMac. For this, each time series is scanned using a sliding window of 32 samples with a step size of 16 samples. The subsequent tests make use of Matlab 2017a on a Windows 10 64-bit computer featuring an Intel Core i3-5005U processor at 2 GHz and 4 GB of RAM.

The performance of the three integration methods is measured in terms of the average execution time over 100 requests. As a request, we consider the call to one of the functions for TC or SR. We emphasize that all the requests are called with the same parameters and data to maintain their homogeneity. Notice also that, although the absolute execution times may be highly dependent on the target machine, however, we are only interested in the relative times between the three methods, which are evaluated on the same machine. For the "Standalone Files" method specifically, a synchronization between writing files and running the application is required. Thus a requester with a busy wait loop expects to record all the values of each data stream and write all the files before calling the corresponding function. Instead, for the other two methods, which are based on a client-server model, the server is always able to apply the desired functionality when submitting a request. As such, the timing requirement is satisfied by the architecture itself.

Test-case 1: Tensor completion. The TC problem is solved for a set of 4 pressure streams, whose missing values are recovered simultaneously. We test two different fill ratios, namely, 40% and 60%, by artificially generating streams with 60% and 40%, respectively, of randomly missing entries. The overall execution time of the TC function refers to the time period during which the request is made, the TC operation is completed, and the output results are obtained.

The completion of the TC function is common to both the file- and the socket-based architectures, whereas the export of the results differs. In the former case, the results are written in files and read by the applicant, which has the responsibility to delete them. In the latter case, it suffices to send the results through the socket.

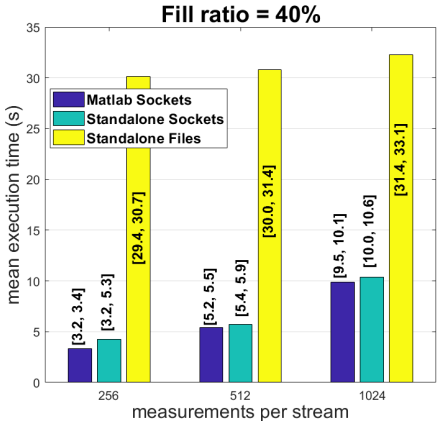
Figs. 4a-4b depict the average execution times for the TC functionality for the three integration methods and the three window lengths, with the 95% confidence intervals for the range of mean execution time shown on top of each bar. As expected, the "Standalone Files" method yields the worst performance for both fill ratio values and for all window lengths. On the other hand, the two socket-based methods achieve a significant reduction of the execution times. Most importantly, the performance of our proposed "Standalone Sockets" approach is very close to that of its "Matlab Sockets" counterpart, while offering the advantage of a non-paid, license-free, system design, in contrast to "Matlab Sockets". Furthermore, as the window length increases, more resources are required by the system, which increases the average execution times for all the methods compared. Finally, as the fill ratio increases (i.e., fewer entries are missing) all the methods are executed in slightly reduced times, as expected.

Test-case 2: Temporal super-resolution. The SR problem is solved for each one of the 4 pressure streams separately. Specifically, an increase of the time resolution is performed for data windows of length 512, with a zoom factor equal to two. Doing so, the intermediate flow created and sent to the server has a length of 1024 samples with the actual and missing entries alternating. Fig. 5 shows the average execution times for the three integration methods and for each one of the 4 pressure streams. As in the case of TC, the "Standalone Files" architecture results in the worst performance for all the streams. On the other hand, our proposed "Standalone Sockets" method yields a similar performance with, or even it outperforms, the optimal "Matlab Sockets" scheme.

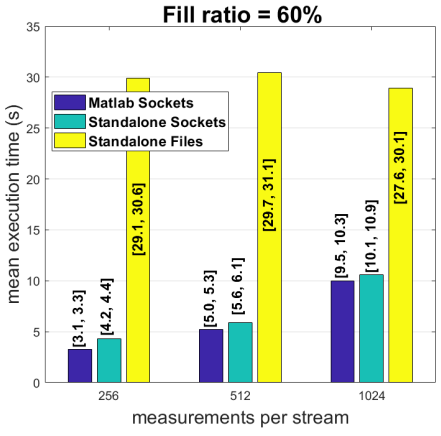
V. Related Work

Matlab aims at reusing legacy code providing a two-way integration with other programming languages. Such an architecture is often found in the creation of virtual laboratories to support distance learning. The independence of the entities used enhances their easy replacement with different technologies, without losing the functionality of the system. The server

[2]Code available at https://xu-yangyang.github.io/TMac/.



(a)



(b)

Fig. 4: Average execution times for the TC problem and for a fill ratio of (a) 40% and (b) 60%, using the three Matlab-Java integration methods.

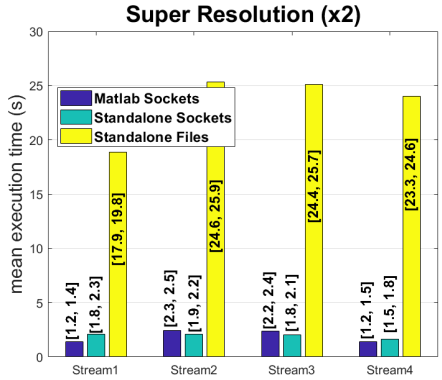


Fig. 5: Average execution times for the SR problem and the four streams, using the three Matlab-Java integration methods.

entity is used, among other things, as a math engine for making calculations or producing simulation components [2]–[4]. The client entity is dedicated to the development of the graphical interface from which the user can interact with system and receive feedback. Depending on the requirements of each application, the graphical interface can be implemented as a clean Java GUI [3] or a web-based GUI [2], [4]. The independence of the involved entities is such that it even applies to the underlying protocol for data exchange. Although the most popular standard we encounter is TCP, depending on system specifications, we can also find ActiveX, DDE [3]. In [5], the problem of not existing proper support for recalling Matlab functions from within Java VM integrated with Matlab is highlighted. The need for two-way asynchronous communication was resolved by an adjective producer-consumer model. Furthermore, a complete Java-based framework for carrying out DSP tasks for wireless sensor networks applications has been proposed in [9], [10].

VI. Conclusions and Future Work

This paper examined a socket-based method for integrating Matlab and Java codes in an industrial setting, whilst relying on a fully non-paid, license-free implementation. A comparison with two alternative methods, namely, a method based on I/O file operations and an optimal socket-based method that executes the codes directly in a Matlab environment, revealed the high performance of our adopted scheme. More specifically, an experimental evaluation by solving the tensor completion and temporal super-resolution problems using real pressure data demonstrated the inefficiency of the file-based approach. On the contrary, our socket-based method, whose performance approximates closely, or even outperforms, the one of the Matlab-based counterpart, enables the design of a robust, independent and portable system, based on the client-server model, without requiring a paid Matlab license.

A critical issue in industrial applications is related to data security. For this, we are interested in extending the socket-based approach by incorporating a data encryption mechanism applied to the requests sent from the client, as well as to the corresponding functions when sending the response to the request from the server.

References

[1] J. Sanchez, "Virtual and remote control labs using Java: A qualitative approach", *IEEE Control Syst. Mag.*, **22**(2):8–20, 2002.

[2] C. Röhrig and A. Jochheim, "Java-based framework for remote access to laboratory experiments," *IFAC Proc. Volumes*, **33**(31):67–72, 2000.

[3] C. Schmid, "A remote laboratory using virtual reality on the web," *Simulation*, **73**(1):13–21, 1999.

[4] K. Magnusson, S. Scandpower, and M. Hogskola, "Integrating Java with a Matlab environment at Studsvik Scandpower," Dissertation, 2002 (https://bit.ly/2tN5ovP).

[5] A. Naderlinger, J. Templ, S. Resmerita, and W. Pree, "An asynchronous Java interface to MATLAB," *in Proc. 4th Intl. ICST Conf. on Simulation Tools and Techniques*, Barcelona, Spain, 21-25 Mar., 2011.

[6] M. Xue and C. Zhu, "The socket programming and software design for communication based on client/server," *in Proc. IEEE Pacific-Asia Conf. on Circuits, Comm. and Systems*, Chengdu, China, 16-17 May, 2009.

[7] D. Ruble, *Practical Analysis and Design for Client/Server and GUI Systems*, Prentice Hall, 1997.

[8] S. D. Indu *et al.*, "Wireless sensor networks: Issues & challenges," *Intl. J. of Computer Science and Mobile Comp.*, **3**(6):681–685, 2014.

[9] H. Kwon *et al.*, "Experiments with sensor motes and Java-DSP," *IEEE Trans. Education*, **52**(2):257–262, 2009.

[10] F. Aiello *et al.*, "A java-based agent platform for programming wireless sensor networks," *The Computer Journal*, **54**(3):439–454, 2011.