The Universal Java Matrix Package: Everything is a Matrix!

Holger Arndt, Department of Computer Science, Technical University of Munich, 85747 Garching, Germany, Email: mail@holger-arndt.com

I. Introduction

Matrix operations form the basis of many machine learning algorithms and have a great impact on their total performance. Therefore, it is essential to utilize fast algorithms for matrix operations such as multiplication or decomposition. As for Java, however, programmers have to use additional libraries for this purpose, since Sun's JDK does not provide a built-in matrix implementation.

The most popular Java matrix libraries, JAMA [8] and Colt [5], are slower than necessary, as they cannot exploit multi-threading on modern hardware. Moreover, they have not received updates for a long time and are limited to certain use cases. For example, JAMA can only deal with two-dimensional dense matrices with double values and is unuseable for sparse or multi-dimensional data, or other cell types.

II. The Universal Approach

UJMP, the Universal Java Matrix Package [3] is intended to be the "swiss army knife" for linear algebra and data processing in Java. It introduces a consistent and unified data representation, which is applicable to *any* kind of data with a rectangular shape. It supports:

- dense and sparse matrices,
- multi-dimensional matrices or tensors,
- arbitrary cell types including generics,
- up to 2^{63} rows or columns,
- data sizes exceeding the capacity limit of main memory.

III. Data Storage

This great flexibility comes from the fact, that UJMP introduces a hierarchy of interfaces, abstract matrix classes and implementations which is illustrated in Fig. 1:

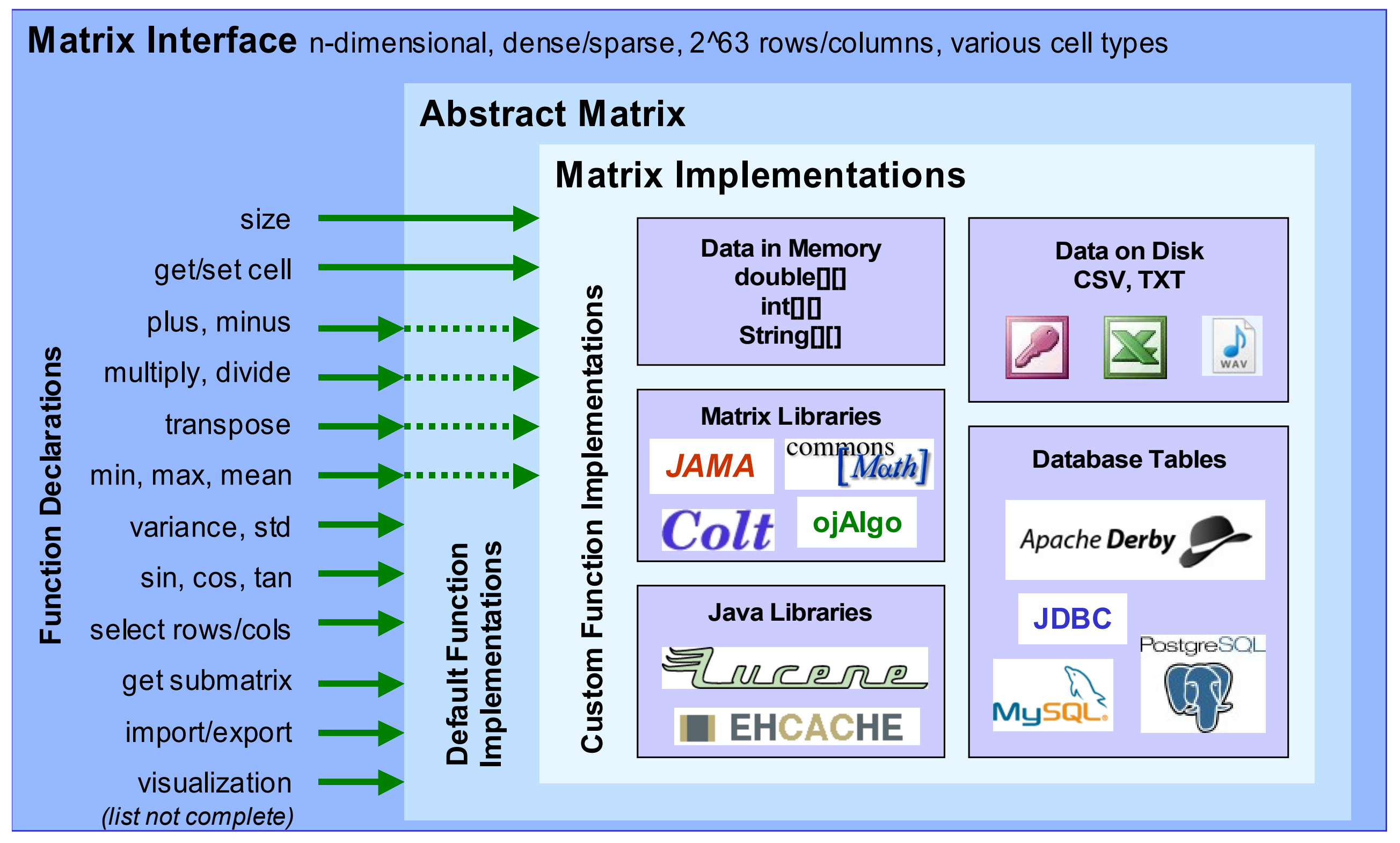

Figure 1: Interface, abstract class and matrix implementations.

In fact, the underlying storage implementation becomes secondary and we can say that, for UJMP, *everything* is a matrix:

arrays of values in memory, binary files on disk, JDBC database tables, Excel files, CSV files, images, audio files, network graphs, Java collection classes, and even other matrix libraries!

IV. Features

To facilitate data processing, UJMP does not only offer basic methods for linear algebra (plus, minus, transpose, inverse, solving equations) but also a large number of additional functions important in machine learning and scientific research (mean, variance, replacement of missing values, mutual information, etc.), interfaces to Matlab, Octave and R, import/export filters for many file types and visualization methods. To deliver the best possible performance, UJMP supports parallel execution with multiple threads on modern hardware.

All matrix implementations can store additional meta-information such as row or column labels. Most operations can be executed in three different ways to reduce the memory footprint if necessary (create a copy of the data, modify the original matrix, or link an operation to the original data). Iterators and a mechanism for automatic entry conversion facilitates data access.

V. Integrating Other Libraries

Numerical algorithms may exhibit a very different runtime performance, depending on matrix size, number of CPUs, CPU type, memory architecture, operating system and Java version. It is virtually impossible to tune e.g. an algorithm for singular value decomposition to work equally well on all platforms. Therefore, UJMP pursues a different approach and integrates 20 matrix libraries within one toolbox. If necessary, UJMP can thus redirect matrix operations to faster libraries to provide the best possible performance. An example is illustrated in Fig. 2, where JAMA [8] is used for calculating a singular value decomposition on smaller matrices, and ojAlgo [7] for sizes greater than 100x100:

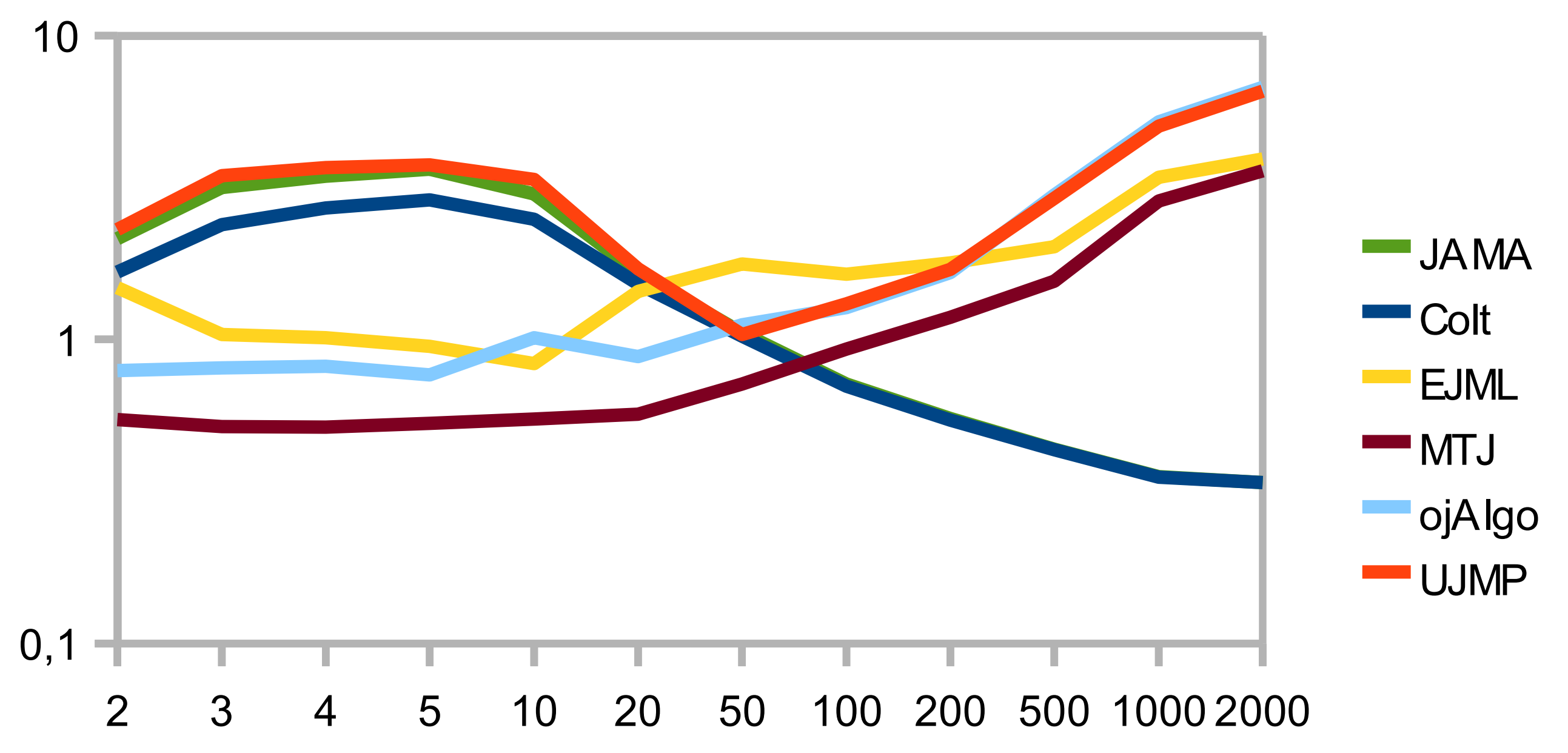

Figure 2: Relative performance of SVD for matrix sizes between 2x2 and 2000x2000 on an Intel Xeon with 4 cores (through UJMP interface).

Other outstanding libraries, which can be used to perform matrix decompositions are Parallel Colt [9], Apache commons-math [2], MTJ [6], EJML [1], ojAlgo [7], and jblas [4].

VI. Summary

The Universal Java Matrix Package is a novel and innovative matrix library for Java. Its extendable architecture makes it ready for different data types and large amounts of data. It can integrate other matrix libraries for maximal performance. Flexible calculation methods beyond standard linear algebra functions are provided, which can be applied in different ways to save memory.

However, the availability of documentation remains to be improved, which would make it easier for new users to become acquainted with the concepts of this library. As compensation, there is a public online forum for Q&A.

It should be emphasized, that the Universal Java Matrix Package is licensed under GNU LGPL, which allows its integration into commercial applications. Source code and jar files are freely available through our website [3] in the hope that it will attract many new users and developers. Everyone is welcome to contribute!

References

[1] Peter Abeles, *Efficient Java Matrix Library (EJML)*, http://code.google.com/p/efficient-java-matrix-library/, 2010.

[2] Apache Software Foundation, *Apache commons mathematics library*, http://commons.apache.org/math/, 2008.

[3] Holger Arndt, Markus Bundschus, and Andreas Nägele, *Towards a next-generation matrix library for Java*, COMPSAC: International Computer Software and Applications Conference (2009), http://ujmp.org.

[4] Mikio Braun, *jblas: Linear Algebra for Java*, http://jblas.org/, 2010.

[5] CERN – European Organization for Nuclear Research, *Colt*, http://acs.lbl.gov/~hoschek/colt/, 1999.

[6] B.-O. Heimsund, *Matrix toolkits for Java (MTJ)*, http://ressim.berlios.de/, 2006.

[7] Anders Petersson, *ojAlgo*, http://ojalgo.org/, 2010.

[8] The MathWorks and the National Institute of Standards and Technology (NIST), *JAMA: A Java matrix package*, http://math.nist.gov/javanumerics/jama/, 2005.

[9] Piotr Wendykier, *Parallel Colt*, http://sites.google.com/site/piotrwendykier/software/parallelcolt, 2010.