Comparison of mathematical programs for data analysis

(Edition 5.04)

by Stefan Steinhaus
(stefan@steinhaus-net.de)

Munich / Germany
17 July 2008

Location: http://www.scientificweb.de/ncrunch/

History of modification

Edition	Date	Modification
4.0	22/07/2002	Release of new edition 4.0
5.0	24/02/2008	Release of new edition 5.0
5.01	27/02/2008	Minor corrections for Maple and Mathematica based on after publication feedback.
5.02	01/03/2008	Minor correction in debugging part of Mathematica
5.03	15/04/2008	Add $ for developer engine at Matlab
5.04	17/07/2008	Added missing Scilab functions from Grocer toolbox.

Contents

1. Introduction

The aim of this report is to compare mathematical programming languages on a fair level. The intention is to show only facts about the tested programs and to avoid making subjective remarks. Therefore it could be used as base information to make your own decision.

The main focus of the report is to have a closer look on mathematical programming, data analysis and simulation functionality for huge and very huge data sets. That type of functionality is of great interest for econometrics, the financial sector in general, biology, chemistry, physics and also several other businesses where the numerical analyze of data is very important.

1.1. Contents

The report consists of tables which are listing the availability of functions for each program. It is divided in functional sections of mathematical, graphical functionality and programming environment, a data import/export interface section, the availability for several operating systems, a speed comparison and finally a summary of the whole information.

To rate all these information a simple scoring system has been used. Available functions received 1 point (implemented or as free addon), available functions by additional modules 0.8 points and not available functions equal 0 points. The summation of all points per section gives the rating. In the summary all the ratings of the subsections have been weighted in the following relation:

Installation, learnability, usability	15%
Mathematical functions	35%
Graphical functions	10%
Programming environment	11%
Data import/export	5%
Available operating systems	2%
Speed comparison	22%

1.2. Globally used symbols

+	-	Function is implemented in the Program
m	-	Function is supported by an additional module which is free of charge.
$	-	Function is supported by an additional module which is not free of charge.
-	-	Function is not implemented

Listed functions are all based on commercial products (except Scilab) which have a guaranteed maintenance and support. Of course there is a huge amount of freeware add-ons, modules available but with no guarantee for maintenance or support. This is a very important point for several types of business (i.e. for the usage in a bank).

1.3. Tested programs

GAUSS from Aptech Systems Inc.	(www.aptech.com)
Maple from Waterloo Maple Software Inc.	(www.maplesoft.com)
Mathematica from Wolfram Research Inc.	(www.wolfram.com)
Matlab from The Mathworks Inc.	(www.mathworks.com)
O-Matrix from Harmonic Software	(www.omatrix.com)

OxMetrics(Ox Prof.) from Timberlake Consultants Ltd. (www.oxmetrics.net)
W Scilab from INRIA (www.scilab.org)

Macsyma is a mathematical program which has been included in test reports in the past. However the commercial development and distribution has been stopped. Currently it is available as Freeware under the name Maxima. It is still a very interesting product but I was not able to include it for the test report this time.

2. Installation, learnability and usability

2.1. Introduction

A huge amount of mathematical programs are available on the market. Advertisement is promising a variety of features which should cover any kind of usage scenario. However before we are going to have a detailed look on these features, let us have a look on the ROI. ROI[1] is a short cut for the “Return of investment”. Someone is going to invest money in expectation to make a profit. If we transfer the expression ROI to a software product this means that someone is going to buy a software product to improve company processes, optimized and/or reduce costs on another side or similar situations.

Now how are we going to receive an ROI out of a new software product? Looking on worst case scenarios software will be bought and stays in a book shelf with no usage at all[2]. Or maybe one step ahead the software should be used but the software distribution and administration has not been planned. Even if this is clear what happens if the software is finally installed but it is so complicated that the users need to be send on cost intensive seminars or the usability keeps more time then the situation before.

To be sure an ROI could only be reached if a software investment is well planned, the software itself is easy to manage, simple to learn and to use. The saved time needs to be in healthy relation to the time being spent before the investment else there will be no ROI.

To be honest a huge amount of time could be spend into a correctly planned software investment. That is a work which is often been done by consultants when it comes to high cost investments and which is of course also time intensive. This report cannot and will not cover this very interesting topic!!!

2.2. Installation

Installation is something which appears to be trivial. Get a CD and start the SETUP routine. Well if you work in a small company the user is mostly also administrator and responsible for the installation. In a huge company or in companies which have a high demand for IT security there are certain rules about installation of software and clear guidelines who is responsible for this. So often a software installation is handled by

[1] http://en.wikipedia.org/wiki/Return_on_Investment
[2] This is no joke or fictive scenario. This really happens.

a central service which makes remote installations necessary. Another scenario would be to do computation on a central server whereas yourself you are working/developing on a client.

Therefore the following table is only a short summary of trivial features around the subject installation.

Functions (Version)	**GAUSS** (8.0)	**Maple** (V11)	**Mathematica** (6.0)	**Matlab** (2008a)	**O-Matrix** (6.3)	**Ox Prof.** (5.0)	**Scilab** (4.1.2)
Standard OS installation (i.e. MSI under Windows)	+	+	+	+	+	+	+
Customizable installation	+	+	+	+	+	+	+
Silent installation mode	-	+	+	+	-	+	+
Client/Server installation	+	+	+	+	-	+	+
License management for client/server usage	+	+	+	+	-	-	*[3]
Online check for updates	-	+	-	+	-	-	-
Available features	66.67% (4/6)	100.00% (6/6)	83.33% (5/6)	100.00% (6/6)	33.33% (2/6)	66.67% (4/6)	80.00% (4/5)

2.3. Learnability

Learning the usage or programming under a software product is essential for beginners. Professionals mostly don't need help for learning but just for lookups of certain commands or methods. The following table shows parameters which can help people learning how to use those programs.

Functions (Version)	**GAUSS** (8.0)	**Maple** (V11)	**Mathematica** (6.0)	**Matlab** (2008a)	**O-Matrix** (6.3)	**Ox Prof.** (5.0)	**Scilab** (4.1.2)
Online help (content/index/search for command/topic/keyword)	+/+/+/+/+	+/+/+/+/+	+/+/+/+/+	+/+/+/+/+	+/+/+/+/+	+/+/-/-/-	+/-/+/+/+
Manual (printed/electronic)	$/+	+/+	+/+	+/+	-/+[4]	+/+	$[5]/+
Additional books	$	$	$	$	-	$	+
Math. background explanation	-	+	+	+	-	+	-
Web based training (WBT)							
Program usage	-	+	+	+	-	-	-
Mathematical usage	-	+	+	-	-	-	-

[3] License management is not necessary since SciLab is a freeware product.
[4] Electronically manual is identical with online help.
[5] Documentations for Scilab are mostly available for free over the INRIA website (PDF or HTML format).

Functions (Version)	GAUSS (8.0)	Maple (V11)	Mathematica (6.0)	Matlab (2008a)	O-Matrix (6.3)	Ox Prof. (5.0)	Scilab (4.1.2)
Tutorials							
Program usage	-	+	+	+	-	+	-
Mathematics	-	+	+	-	-	-	-
Available features	58.46% (7.6/13)	98.46% (12.8/13)	98.46% (12.8/13)	83.08% (10.8/13)	46.15% (6/13)	52.31% (6.8/13)	52.31% (6.8/13)

2.4. Usability

Usability is a subject which is important for every user independent from his/her skill level. Problems which appear during work need to be solved or case specific search for solutions have to be satisfied. The search for the "right" solution could become very time intensive and should never be underestimated. Real development time is nothing in comparison to the search of solutions or solving of technical problems.

Functions (Version)	GAUSS (8.0)	Maple (V11)	Mathematica (6.0)	Matlab (2008a)	O-Matrix (6.3)	Ox Prof. (5.0)	Scilab (4.1.2)
Newsgroups / mailing lists	+	+	+	+	+	+	+
FAQ lists	+	+	+	+	-	+	+
Support hotline	$[6]	+	+	+	+[7]	+	+[8]
Use cases							
Demonstration	-	+	+	$	+	-	-
Source code	-	-	+	$	+	-	-
Explanation	-	-	+	$	+	-	-
Enhanced UI support							
UI support for math input	-	+	$	+	+[9]	-	+[10]
Assistant	-	+	$	-	-	-	-
Palettes for symbols and commands	-	+	+	-	-	-	-
Work documentation							
Documentation functionality	-	+	+	+	-	-	-
In place documentation	-	+	+	-	-	-	-
Document export to RTF / LaTex / PDF / HTML	- / - / - / -	+ / + / - / +	+ / + / + / +	+ / + / + / +	- / - / - / -	- / - / - / -	- / - / - / -
Available features	18.67% (2.8/15)	80.00% (12/15)	97.33% (14.6/15)	69.33% (10.4/15)	40.00% (6/15)	20.00% (3/15)	26.67% (4/15)

[6] Surpport via email available for Platinum Premier Support contract
[7] Support via email.
[8] Support via email.
[9] Support for UI programming not totally equivalent to "UI support for math input"
[10] Support for UI programming not totally equivalent to "UI support for math input"

2.5. Summarizing the comparison of the mathematical functions

The following table summarizes the results with the mentioned weighting:

Installation	10%
Learnability	30%
Usability	60%

Functions (Version)	**GAUSS (8.0)**	**Maple (V11)**	**Mathematica (6.0)**	**Matlab (2008a)**	**O-Matrix (6.3)**	**Ox Prof. (5.0)**	**Scilab (4.1.2)**
Installation (10%)	66.67%	100.00%	83.33%	100.00%	33.33%	66.67%	80.00%
Learnability (30%)	58.46%	98.46%	98.46%	83.08%	46.15%	52.31%	52.31%
Usability (60%)	18.67%	80.00%	97.33%	69.33%	40.00%	20.00%	26.67%
Overall result (100% = Best)	35.41%	87.54%	96.27%	76.52%	41.18%	34.36%	39.69%

3. Comparison of the mathematical functionality

Actually there are a lot of different mathematical and statistical programs on the market which are covering a huge amount of functions. The following tables should give an overview about the functionality for analyzing data in numerical ways and should mark out which functions are supported by which program and whether these functions are already implemented in the base program or whether you need an additional module. The functions are sorted by the categories:

Standard mathematics
Linear algebra
Analysis
Numerical mathematics
Stochastic
Statistics
Other mathematics

3.1. Standard mathematics

Standard mathematics functions are an essential part of any kind of mathematical work. Not necessary to mention that these type of functions should be available in all programs. Therefore the following results are not very surprising.

Functions (Version)	**GAUSS** (8.0)	**Maple** (V11)	**Mathematica** (6.0)	**Matlab** (2008a)	**O-Matrix** (6.3)	**Ox Prof.** (5.0)	**Scilab** (4.1.2)
BesselI	+	+	+	+	+	+	+
Bessel J	+	+	+	+	+	+	+
BesselK	-	+	+	+	+	+	+
Bessel Y	+	+	+	+	+	+	+
Beta function	+	+	+	+	+	+	+
Binomial	-	+	+	+	-	+	-
Factorial	+	+	+	+	+	+	+
FresnelC	-	+	+	$	-	-	-
FresnelS	-	+	+	$	-	-	-
Gamma function	+	+	+	+	+	+	+
Hyperbolic trig. function	+	+	+	+	+	+	+
Incomplete Gammafunc.	+	+	+	+	+	+	-
Log / Ln / Exp	+ / + / +	+ / + / +	+ / + / +	+ / + / +	+ / + / +	+ / + / +	+ / + / +
Log-Gammafunc.	+	+	+	+	+	+	+

Functions (Version)	GAUSS (8.0)	Maple (V11)	Mathematica (6.0)	Matlab (2008a)	O-Matrix (6.3)	Ox Prof. (5.0)	Scilab (4.1.2)
Poly gamma	-	+	+	+	-	+	+
Square root	+	+	+	+	+	+	+
Sum / Product	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +
Trig. / arg trig. functions	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +
Implemented functions	77.27% (17/22)	100.00% (22/22)	100.00% (22/22)	98.18% (21.6/22)	81.82% (18/22)	90.91% (20/22)	81.82% (18/22)

3.2. Algebra

Algebra and especially linear algebra offers a basic functionality for any kind of matrix oriented work. I.e. optimization routines are widely used in the financial sector but also very useful for logistic problems (remember the traveling salesman problem). Most simulation and analyzing routines are relying on decompositions, equation solving and other routines from algebra.

Functions (Version)	GAUSS (8.0)	Maple (V11)	Mathematica (6.0)	Matlab (2008a)	O-Matrix (6.3)	Ox Prof. (5.0)	Scilab (4.1.2)
Eigenvalues							
Eigenvalues	+	+	+	+	+	+	+
Eigenvectors	+	+	+	+	+	+	+
Matrix analysis							
Characteristic polynom	+	+	+	+	-	+	+
Determinant	+	+	+	+	+	+	+
Hadamard matrix	-	-	m	+	-	-	-
Hankel matrix	-	+	+	+	+	-	-
Hilbert matrix	-	+	+	+	+	-	+
Householder matrix	$	+	-	+	-	+	+
Inverse matrix	+	+	+	+	+	+	+
Kronecker product	+	+	+	+	+	+	+
Pascal matrix	-	-	-	+	-	-	-
Toeplitz matrix	+	+	+	+	+	+	+
Upper Hessenberg form	+	+	+	+	-	-	+
Decompositions							
Cholesky decomposition	+	+	+	+	+	+	+
Crout decomposition	+	-	-	+	-	+	-
Dulmage-Mendelsohn decomposition	-	-	-	+	-	-	-
LU decomposition	+	+	+	+	+	+	+
QR decomposition	+	+	+	+	+	+	+
Schur form of quadratic matrix	+	+	+	+	+	+	+
Smith normal form	-	+	-	$	+	-	-
Singular value decomposition	+	+	+	+	+	+	+

Functions (Version)	GAUSS (8.0)	Maple (V11)	Mathematica (6.0)	Matlab (2008a)	O-Matrix (6.3)	Ox Prof. (5.0)	Scilab (4.1.2)
Optimization							
Optimization - linear models (Unconstr. / Constr.)	+ / +	+ / +	+ / +	+ / +	+ / +	+ / -	+ / +
Optimization - nonlinear models (Unconstr. / Constr.)	+ / +	+ / +	+ / +	$ / $	+ / +	+ / +	+ / +
Optimization - quadratic models (QP) (Unconstr. / Constr.)	+ / +	+ / +	+ / +[11]	$ / $	+ / +	+ / +	+ / +
Equation solver							
Linear equation solver	+	+	+	+	+	+	+
Non-linear equation solver	$	+	+	$	+	+	+
Ordinary Differential Equation solver	$	+	+	+	+	-	+
Partial Differential Equation solver	-	+	+	+	-	-	-
Miscellaneous							
Moore-Penrose pseudo-inverse	+	+	+	+	+	+	+
Sparse matrices handling	+	+	+	+	-	-	+
Implemented functions	76.97% (25.4/33)	87.88% (29/33)	84.85% (28/33)	93.94% (31/33)	72.73% (24/33)	66.67% (22/33)	78.79% (26/33)

3.3. Analysis

The Analysis offers functions which are commonly used in technical areas. Functions like Fourier transformation for example are used in data acquisition. The following table shows only a small extract of important functions. It is important to mention that symbolic math becomes very important in connection with Analysis. However symbolic math is not subject of this report.

Functions (Version)	GAUSS (8.0)	Maple (V11)	Mathematica (6.0)	Matlab (2008a)	O-Matrix (6.3)	Ox Prof. (5.0)	Scilab (4.1.2)
Dirac Delta	-	+	+	+	-	-	-
Limit	-	+	+	+	-	-	-
Numerical integration (single / double / triple)	+ / + / +	+ / + / +	+ / + / +	+ / + / +	+ / + / -	+ / + / -	+ / + / +
Numerical differentiation (1st deriv. / 2nd deriv.)	+ / +	+ / +	+ / +	+ / +	+ / -	+ / +	+ / +
Fourier transf. (1D / 2D / multidim.)	+ / + / +	+ / + / +	+ / + / +	+ / + / +	+ / + / -	+ / - / -	+ / + / +
Inverse Fourier transformation (1D / 2D / multidim.)	+ / + / +	+ / + / +	+ / + / +	+ / + / +	+ / + / -	+ / - / -	+ / + / +

[11] Quadratic programming is available by an additional module but also by Mathematica itself. However it must be marked out that the Method->"QuadraticProgramming" option for FindMinimum is undocumented and therefore only usable for those who know it.

Functions (Version)	GAUSS (8.0)	Maple (V11)	Mathematica (6.0)	Matlab (2008a)	O-Matrix (6.3)	Ox Prof. (5.0)	Scilab (4.1.2)
Implemented functions	84.62% (11/13)	100.00% (13/13)	100.00% (13/13)	100.00% (13/13)	53.85% (7/13)	46.15% (6/13)	84.62% (11/13)

3.4. Numerical mathematics

The numerical mathematics offers fundamental algorithms for several appliances. It is marked out that especially any kind of interpolation algorithms are commonly used in technical and non technical businesses. Without really recognizing interpolation routines are used in nearly any kind of graphical representation.

Function (Version)	GAUSS (8.0)	Maple (V11)	Mathematica (6.0)	Matlab (2008a)	O-Matrix (6.3)	Ox Prof. (5.0)	Scilab (4.1.2)
Interpolation							
B-Spline interpolation	-	+	+	$	-	+	+
Classical interpolation (1D/2D/3D/n D)	+/+/-/-	+	+/+/+/+	+/+/+/+	+/-/-/-	-	+/-/-/-
k-Spline interpolation	+	+	+	$	+	+	+
Pade interpolation	$	+	+	-	+	-	-
Piecewise cubic hermite polynomial interpolation	-	-	+	+	-	-	-
Piecewise polynomial interpolation	-	-	+	+	-	-	-
Other functions							
Bisection	$	-	+	+	-	+	-
Newton method for finding roots	+	+	+	$	+	+	+
Runge Kutta method for solving ODE	$	+	+	$	+	-	+
Implemented functions	53.33% (6.4/12)	75.00% (9/12)	100.00% (12/12)	85.00% (10.2/12)	41.67% (5/12)	33.33% (4/12)	41.67% (5/12)

3.5. Descriptive statistic, stochastic and distribution functions

Very important to get familiar with data and to understand samples of data are stochastic and descriptive statistic routines. Distribution functions, their CDF and PDF function are commonly used to figure out what are representative samples and what are outliers. A typical “simple” but common usage might be in a productive area to take samples of the manufactured product and to see whether the faulty parts in a sample are within a normal range. More complex usages might be in load balancing simulations of telecommunication hardware. However it might be possible to mention example usages for nearly all kind of business.

Functions (Version)	GAUSS (8.0)	Maple (V11)	Mathematica (6.0)	Matlab (2008a)	O-Matrix (6.3)	Ox Prof. (5.0)	Scilab (4.1.2)
General functions							

Functions (Version)	**GAUSS** (8.0)	**Maple** (V11)	**Mathematica** (6.0)	**Matlab** (2008a)	**O-Matrix** (6.3)	**Ox Prof.** (5.0)	**Scilab** (4.1.2)
Contingency tables	$	-	-	-	-	-	-
Correlation	+	+	+	+	+	+	+
Cross tabulation	$	-	-	$	-	-	-
Deviation	+	+	+	+	+	+	+
Kurtosis	-	+	+	$	+	+	+
Markov models	$	$	$	$	-	+	+
Mean /geometric Mean / Mode	+ / + / $	+ / + / +	+ / + / +	+ / + / -	+ / + / +	+ / + / +	+ / + / +
Min / Max	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +
Quantile / Percentile	+ / -	+ / +	+ / -	$ / $	+ / +	+ / -	+ / +
Skewness	-	+	+	$	+	+	m
Variance	+	+	+	+	+	+	+
Variance-covariance matrix	+	+	+	+	+	+	+
Distribution functions *(PDF / CDF / iCDF/ randomnumber)*							
Bernoulli	-/-/-/-	+/+/+/+	+/+/+/+	$/$/$/$	-/-/-/-	-/-/-/-	-/-/-/-
Beta	$/+/$/+	+/+/+/+	+/+/+/+	$/$/$/$	-/-/-/-	+/+/+/+	m/+/-/+
Binomial	$/$/$/$	+/+/+/+	+/+/+/+	$/$/$/$	-/-/-/+	+/+/+/+	-/+/-/+
Brownian motion	-/-/-/-	$/$/$/$	-/-/-/$	-/-/-/-	-/-/-/-	-/-/-/+	-/-/-/-
Cauchy	$/$/$/$	+/+/+/+	+/+/+/+	-/-/-/-	-/$/$/+	+/+/+/+	-/-/-/-
Chi-squared	$/+/+/$	+/+/+/+	+/+/+/+	$/$/$/$	$/-/-/-	+/+/+/+	m/+/-/+
Chi-squared (non-central)	$/+/$/$	+/+/+/+	+/+/+/+	$/$/$/$	-/-/-/-	-/+/-/-	m/+/-/+
Dirichlet	-/-/-/-	-/-/-/-	$/$/$/$	-/-/-/-	-/-/-/-	-/-/-/+	-/-/-/-
Erlang	$/$/$/$	+/+/+/+	+/+/+/+	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
Exponential	$/$/$/$	+/+/+/+	+/+/+/+	$/$/$/$	+/+/$/+	+/+/+/+	-/-/-/+
Extreme value	-/-/-/-	-/-/-/-	+/+/+/+	$/$/$/$	-/-/-/-	+/+/+/+	-/-/-/-
F	+/+/+/$	+/+/+/+	+/+/+/+	$/$/$/$	+/-/-/-	+/+/+/+	m/+/-/+
F (non-central)	+/+/$/$	-/+/+/+	+/+/+/+	$/$/$/$	-/-/-/-	-/+/-/-	m/+/-/+
Gamma	$/+/$/+	+/+/+/+	+/+/+/+	$/$/$/$	-/-/-/+	+/+/+/+	m/+/-/+
Geometric	$/$/$/$	-/-/-/-	+/+/+/+	$/$/$/$	-/-/-/-	+/+/+/+	-/-/-/-
Gumbel	$/$/$/$	+/+/+/+	+/+/+/+	-/-/-/-	-/$/$/+	-/-/-/-	-/-/-/-
Half-normal	-/-/-/-	-/-/-/-	+/+/+/+	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
Hotelling T2	$/$/-/-	-/-/-/-	+/+/+/+	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
Hyper-exponential	$/$/$/$	-/-/-/-	$/$/$/$	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
Hypergeometric	$/$/$/$	+/+/+/+	+/+/+/+	$/$/$/$	-/-/-/-	+/+/+/+	m/m/-/-
Kernel	-/-/-/-	-/-/-/-	$/$/$/$	-/-/-/-	-/-/-/-	-/+/-/-	-/-/-/-
Laplace	$/$/$/$	+/+/+/+	+/+/+/+	-/-/-/-	-/-/-/+	-/-/-/-	-/-/-/-
Logarithmic	-/-/-/-	-/-/-/-	+/+/+/+	-/-/-/-	-/-/-/-	+/+/+/+	-/-/-/-
Logistic	$/$/$/$	+/+/+/+	+/+/+/+	-/-/-/-	-/$/-/-	+/+/+/+	m/m/-/m
Log-normal	+/+/+/$	+/+/+/+	+/+/+/+	$/$/$/$	-/-/-/+	+/+/+/+	m/m/m/m
Log-normal (multivariate)	+/+/$/$	-/-/-/-	+/+/+/+	-/-/-/-	-/-/-/-	-/-/-/-	m/m/m/m
Negative binomial	$/$/$/+	+/+/+/+	+/+/+/+	$/$/$/$	-/-/-/-	+/+/+/+	-/+/-/+
Normal	+/+/+/+	+/+/+/+	+/+/+/+	+/$/$/+	+/+/+/+	+/+/+/+	m/+/-/+

Functions (Version)	GAUSS (8.0)	Maple (V11)	Mathematica (6.0)	Matlab (2008a)	O-Matrix (6.3)	Ox Prof. (5.0)	Scilab (4.1.2)
Normal (bivariat)	-/+/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	+/+/-/+	-/-/-/-
Normal (multivariate)	$/+/$/$	-/-/-/-	+/+/+/+	$/$/$/$	-/-/-/-	+/+/-/+	-/-/-/+
Pareto	$/$/$/$	+/+/+/+	+/+/+/+	-/-/-/-	-/-/-/-	+/+/+/+	-/-/-/-
Poisson	$/$/$/+	+/+/+/+	+/+/+/+	$/$/$/$	-/-/-/+	+/+/+/+	m/+/-/+
Rayleigh	-/-/-/-	+/+/+/+	+/+/+/+	$/$/$/$	-/-/-/-	-/-/-/-	-/-/-/-
S	-/-/-/-	-/-/-/-	$/$/$/$	-/-/-/-	-/-/-/-	-/-/-/+	-/-/-/-
Student's t	+/+/+/$	+/+/+/+	+/+/+/+	$/$/$/$	+/+/+/-	+/+/+/+	m/+/-/m
Student's t (non-central)	+/+/+/$	-/-/-/-	+/+/+/+	$/$/$/$	-/-/-/-	-/+/-/-	-/-/-/-
Student's t (multivariate)	$/$/$/$	-/-/-/-	+/+/+/+	$/$/$/$	-/-/-/-	-/-/-/-	-/-/-/-
Uniform	$/$/-/+	+/+/+/+	+/+/+/+	$/$/$/+	+/+/$/+	-/-/-/+	-/-/-/+
Von Mises	$/$/$/+	+/+/+/+	$/$/$/$	-/-/-/-	-/-/-/-	+/+/+/+	-/-/-/-
Weibull	$/$/$/$	+/+/+/+	+/+/+/+	$/$/$/$	+/+/-/+	+/+/+/+	-/-/-/-
Wishart	$/$/$/$	-/-/-/-	+/+/+/+	-/-/-/$	-/-/-/-	-/-/-/+	-/-/-/m
Implemented functions	63.78% (114.8/180)	64.44% (116/180)	92.00% (165.6/180)	46.89% (84.4/180)	23.56% (42.4/180)	60.00% (108/180)	35.00% (62/180)

3.6. Statistics

Statistical functions are fundamental for any kind of data analysis. Routines like regression or time series are commonly used to find out trends or to predict future values i.e. for stock market courses. Filter routines are used to smooth or filter effects in data acquisition. Multivariate statistics are used to find patterns or common characteristics in data i.e. for market basket analysis by clustering routines.

Functions (Version)	GAUSS (8.0)	Maple (V11)	Mathematica (6.0)	Matlab (2008a)	O-Matrix (6.3)	Ox Prof. (5.0)	Scilab (4.1.2)
Regression models							
Linear	+	+	+	+	+	+	+
Loess	+	-	$	+	m	m	-
Logistic Regression	$	-	$	$	-	-	-
LOGIT / PROBIT	$ / $	- / -	m- / m	$ / $	- / -	m / m	m / m
Nonlinear / Polynomial	$ / $	+ / +	+ / +	$ / +	+ / +	+ / +	+ / +
PSN	$	-	-	-	-	-	-
Tobit models	+	-	-	-	-	-	m
Test statistics							
Ansari-Bradley test	-	-	-	$	-	-	-
Bartlett multiple-sample test	$	-	-	$	-	-	-
Besley test	$	-	-	-	-	-	-
Breusch-Pagan test for homoscedasticity	$	-	-	-	-	+	m
Chow Test for stability	$	-	-	-	-	$	m
CUSUM test for stability	$	-	-	-	-	-	m

Functions (Version)	GAUSS (8.0)	Maple (V11)	Mathematica (6.0)	Matlab (2008a)	O-Matrix (6.3)	Ox Prof. (5.0)	Scilab (4.1.2)
Davidson-MacKinnon J-Test	$	-	-	-	-	-	-
Dickey Fuller test	$	-	-	-	-	$	m
Durbin-Watson test	+	-	+	$	-	$	m
Engle's LM test	$	-	-	$	-	+	-
Friedman's test	$	-	-	$	-	-	-
F-Test	+	+	+	$	+	+	-
Goodness of fit test	$	+	-	$	-	$	-
Goldfeld-Quandt test for homoscedasticity	$	-	-	-	-	-	-
Granger's causality test	$	-	-	-	$	-	-
Hausman's specification test	$	-	-	-	-	-	-
Kolmogorov-Smirnov test	-	-	-	$	$	-	-
Kruskal-Wallis test	$	-	-	$	-	-	-
Kuh test	$	-	-	-	-	-	-
Lagrange multiplier test	$	-	-	-	-	+	m
Lilliefors test	-	-	-	$	-	-	-
Ljung-Box Q-Test	$	-	-	$	$	$	m
Mann-Whitney U test	$	-	-	-	-	-	-
Sign test	$	-	-	$	-	-	-
T-Test	+	+	+	$	+	+	-
Wald test	$	-	-	-	-	+	m
Walsh test	$	-	-	-	-	-	-
Wilcoxon rank sum / sign test	$ / $	- / -	- / -	$ / $	- / -	- / -	- / -
Z-Test	-	+	+	$	-	-	-
Filter / smoothing models							
Bandpass / Lowpass / Highpass / Multiband / Bandstop	$/$/$/-/$	-/-/-/-/-	$/$/$/-/$	$/$/$/$/$	$/$/$/-/$	-/-/-/-/-	+/+/+/+/+
Battle-Lemarie	-	-	$	-	-	-	-
Bessel	$	-	$	$	$	-	-
Butterworth	$	-	$	$	$	-	+
Chebyshev	$	-	$	$	+	-	+
Coiflet	$	-	$	-	-	-	-
Daubechies	$	-	$	$	-	-	-
Elliptic	-	-	$	$	-	-	+
Haar	$	-	$	$	-	-	-
Hodrick-Prescott	-	-	-	-	-	+	m
IIR / FIR	$ / $	- / -	$ / $	$ / $	$ / $	- / -	+ / +
Kernel	-	+	-	$	$	+	-
Linear	$	+	$	$	-	+	+
Meyer	-	-	$	$	-	-	-
Pollen	$	-	-	-	-	-	-
Riccati	-	-	$	$	-	-	+
Shannon	-	-	$	-	-	-	-

Functions (Version)	GAUSS (8.0)	Maple (V11)	Mathematica (6.0)	Matlab (2008a)	O-Matrix (6.3)	Ox Prof. (5.0)	Scilab (4.1.2)
Savitzky-Golay	$	-	-	$	$	-	-
Time series models							
ARMA / ARIMA / ARFIMA / ARMAX	+/$/$/$	- / - / - / -	$/$/-/-	$/-/$/$	$ / $ / $ / -	m/m/m/m	m/-/-/+
GARCH / ARCH / AGARCH / EGARCH / FIGARCH / IGARCH / MGARCH / PGARCH / TGARCH models	$/$/$/$/$/$/$/$/$	-/-/-/-/-/ -/-/-/-	$/$/-/-/-/ -/-/-/-	$/$/-/$/ -/-/-/-/-	$/-/-/-/-/ -/-/-/-	m/m/m/m/m/ m/m/m/m	m/-/-/-/-/ -/-/-/-
Holt's Winter additive / multiplicative	$ / $	- / -	- / -	- / -	$ / -	m / m	- / -
Multivariate GARCH models (Diagonal VEC / BEKK / Matrix Diagonal / Vector Diagonal)	$/$/$/$	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	$/$/$/$	-/-/-/-
Partial autocorrelation	+	-	$	$	$	+	m
Spectral analysis	$	-	$	$	$	+	+
State space models	$	-	$	$	$	m	+
Time series analysis (Stationary / Non-stat.)	$ / $	- / -	$ / $	$ / $	$ / $	+ / +	+ / +
Wavelets	$	-	$	$	+	-	+
Multivariate statistics							
ANOVA / MANOVA	$ / -	+ / -	+ / -	$ / $	- / -	- / -	- / -
Cluster analysis (hierarchical/k-means)	$ / -	- / -	$ / $	+ / $	- / -	- / -	- / -
Discriminant analysis	-	-	-	$	-	-	m
Factor analysis	-	-	-	$	$	-	m
Fuzzy clustering	-	-	-	$	-	-	-
Procrustes analysis	-	-	-	$	-	-	-
Principal component analysis	+	-	-	$	$	-	+
Principal coordinate analysis	-	-	-	$	-	-	-
Survival analysis	-	-	-	-	-	-	-
Design of Experiments							
Box-Behnken design	-	-	-	$	-	-	-
Central composite design	-	-	-	$	-	-	-
D-Optimal design	-	-	-	$	-	-	-
Full / Fractional factorial design	- / -	- / -	- / -	$ / $	- / -	- / -	- / -
Hadamard design	-	-	-	$	-	-	-
Response surface design	-	-	-	$	-	-	-
Other statistical functions & models							
Bootstrapping	$	+	$	$	-	+	-
Duration models	$	-	-	-	-	-	-
Entropy models	$	-	-	$	-	-	-
Event count models	$	-	-	-	-	$	-
Heckman two step estimation	$	-	-	-	-	-	-
Heteroscedasticity	$	-	-	-	-	+	-
Jacknife estimation	$	-	-	$	-	-	-
Lagrange multiplier test	$	+	-	-	-	+	-
Markowitz efficient frontier	$	-	$	$	-	-	-

Functions (Version)	GAUSS (8.0)	Maple (V11)	Mathematica (6.0)	Matlab (2008a)	O-Matrix (6.3)	Ox Prof. (5.0)	Scilab (4.1.2)
Maximum Likelihood (Unconstr. / Constr.)	$ / $	+ / -	+ / +	$ / -	$ / -	+ / +	- / -
Monte Carlo simulation	$	-	$	$	-	+	-
Implemented functions	64.17% (73.8/115)	9.57% (11/115)	34.96% (40.2/115)	53.39% (61.4/115)	25.04% (28.8/115)	41.74% (48/115)	33.91% (40/115)

3.7. Other mathematics

Functions which are not fitting to any of the other categories are listed under other mathematics. Those functions are mostly topic specific like for stock market simulations, data mining or social sciences.

Functions (Version)	GAUSS (8.0)	Maple (V11)	Mathematica (6.0)	Matlab (2008a)	O-Matrix (6.3)	Ox Prof. (5.0)	Scilab (4.1.2)
American binomal method / Black and Scholes - call / put	+/+/+/+	-/-/+/-	$/$/$/$	$/$/$/$	-/-/-/-	m/m/m/m	-/-/-/-
Amortization schedule	$	+	$	$	-	m	-
Cash flow model	$	+	$	$	-	m	-
Cointegration models	$	-	-	-	-	+	m
Decision trees	-	-	+	$	-	-	-
Dynamic rational expectation models	$	-	-	-	-	-	-
European binomal method / Black and Scholes - call / put	+/+/+/+	-/-/-/-	$/$/$/$	$/$/$/$	-/-/-/-	m/m/m/m	-/-/-/-
Fibonacci / prime numbers	- / -	+ / +	+ / +	- / +	- / +	- / -	- / -
Kalman filter	$	-	$	$	+	m	+
LISREL models	$	-	-	-	-	-	-
Neural networks (Forward propagation/Backward propagation)	$ / $	- / -	$ / $	$ / $	- / -	- / -	- / -
Panel models	$	-	-	-	-	m	m
Portfolio analysis	$	-	$	$	-	m	-
Rational Expectation models linear / non-linear	$ / $	- / -	- / -	$ / -	- / -	- / -	- / -
Regressive-autoregressive models	$	-	$	$	+	+	-
Social network models	$	-	$	-	-	-	-
List of constants	-	+	+	-	-	-	-
Implemented functions	73.85% (19.2/26)	23.08% (6/26)	64.62% (16.8/26)	56.15% (14.6/26)	11.54% (3/26)	57.69% (15/26)	11.54% (3/26)

3.8. Summarizing the comparison of the mathematical functions

Altogether 401 functions have been listed. The following table summarizes the results with the mentioned weighting:

Standard mathematics	5%	Descriptive statistic	20%
Linear Algebra	15%	Stochastic and distribution functions	20%
Analysis	10%	Statistics	20%
Numerical mathematics	10%	Other mathematics	20%

Functions (Version)	**GAUSS** (8.0)	**Maple** (V11)	**Mathematica** (6.0)	**Matlab** (2008a)	**O-Matrix** (6.3)	**Ox Prof.** (5.0)	**Scilab** (4.1.2)
Standard mathematics (5%)	77.27%	100.00%	100.00%	98.18%	81.82%	90.91%	81.82%
Algebra (15%)	76.97%	87.88%	84.85%	93.94%	72.73%	66.67%	78.79%
Analysis (10%)	84.62%	100.00%	100.00%	100.00%	53.85%	46.15%	84.62%
Numerical mathematics (10%)	53.33%	75.00%	100.00%	85.00%	41.67%	33.33%	41.67%
Descriptive statistics, stochastic and distribution functions (20%)	63.78%	64.44%	92.00%	46.89%	23.56%	60.00%	35.00%
Statistics (20%)	64.17%	9.57%	34.96%	53.39%	25.04%	41.74%	33.91%
Other mathematics (20%)	73.85%	23.08%	64.62%	56.15%	11.54%	57.69%	11.54%
Overall result (100% = Best)	69.56%	55.10%	76.04%	68.79%	36.58%	54.38%	44.63%

4. Comparison of the graphical functionality

Not only for presentation purposes it might be very useful to visualize numerical data (source or resulting data) by using graphical routines. Therefore every mathematical program should support at least typical graphic types like curve plots or histograms but also important statistical chart types like Box plots, dendograms, cluster graphs, multivariate plots. Also addition graphical information on top of trivial graphic types like error bars are very important. Some of the tested programs (most especially CAS) include a real huge amount of graphical functions and graphic types but due to the weighting of the test report on statistical and econometrical functions the following sections will only mention the most important graphic functions.

4.1. Chart types

The following tables show a list of available standard 2D / 3D and specialized graphic types for statistical and econometrical usage.

Functions (Version)	**GAUSS** (8.0)	**Maple** (V11)	**Mathematica** (6.0)	**Matlab** (2008a)	**O-Matrix** (6.3)	**Ox Prof.** (5.0)	**Scilab** (4.1.2)
2D-Graphics							
Area charts	$	+	+	+	-	-	-
Bar charts	+	+	+	+	+	+	+
Bubble Plot	$	+	+	+	+	+	-
Other charts	+	+	+	+	+	+	+
Error bars	$	+	+	+	+	+	+
High-Low-Average Plot	$	-	$	$	-	+	-
Histograms	+	+	+	+	+	+	+
Log Plot	+	+	+	+	+	+	+
Log-log Plot	+	+	+	+	+	+	+
Pie charts	$	+	+	+	-	-	+
Polar Plot	+	+	+	+	+	-	+
Radar Plot	$	-	-	-	-	-	-
Weibull Plot	-	-	-	$	-	-	-
XY Plot	+	+	+	+	+	+	+
3D-Graphics							
Charts	$	+	+	+	-	+	+
Contour Plot	+	+	+	+	+	+	+
Error bars	$	-	-	-	-	-	-
Height colors	+	+	+	+	+	-	+
Spectral Plots	$	-	$	+	-	+	-
Surface Plot	+	+	+	+	+	+	+

Functions (Version)	**GAUSS** (8.0)	**Maple** (V11)	**Mathematica** (6.0)	**Matlab** (2008a)	**O-Matrix** (6.3)	**Ox Prof.** (5.0)	**Scilab** (4.1.2)
XYZ Plot	+	+	+	+	+	+	+
Polygon and surface plots							
Ellipsoid	-	+	+	+	-	-	+
Ribbon plot	-	-	+	+	-	-	-
Spherical plot	-	+	+	+	-	-	-
Voronoi diagram	-	$	+	+	-	-	-
Special graphic types and functions							
Animations	$	+	+	+	-	-	+
Bollinger bands	-	-	+	$	-	-	-
Box & Whisker Plots	+	+	$	$	-	+	-
Candlestick charts	-	-	$	$	-	-	-
Cluster graphs	-	-	-	+	-	-	-
Control charts	-	-	-	$	-	-	-
Dendograms	$	-	+	$	-	-	-
Pareto Charts	$	+	+	$	-	-	-
Periodograms	-	-	-	$	$	+	-
Pyramide Plot	$	-	-	-	$	-	-
QQ Plot	$	+	+	$	$	+	-
Quantile Plot	+	+	+	$	-	+	-
Scatter Plot Matrix	+	+	+	+	-	+	-
Smith chart	-	-	$	$	+	-	-
Velocity plot	-	-	-	+	-	-	+
Overall result ***(100% = Best)***	63.00% (25.2/40)	64.50% (25.8/40)	79.50% (31.8/40)	86.50% (34.6/40)	41.00% (16.4/40)	47.50% (19/40)	42.50% (17/40)

4.2. Graphics import/export formats

Although mathematical programs are often used for calculations, simulations or graphical presentations most reports are made in word processing or spreadsheet programs. Exporting facilities for graphics are therefore a basic requirement for every software. In addition importing facilities are also very helpful for combining existing graphics with newly created once. All of the tested programs do support a way of exporting graphics via the clipboard but also it is essential to have the possibility to export and import graphics to a file. The following table therefore shows the supported file formats for graphic export and import which are of importance.

Export / Import formats (Version)	**GAUSS** (8.0)	**Maple** (V11)	**Mathematica** (6.0)	**Matlab** (2008a)	**O-Matrix** (6.3)	**Ox Prof.** (5.0)	**Scilab** (4.1.2)
AVI	- / -	- / -	+ / +	+ / +	- / -	- / -	m / m
BMP	+ / -	+ / +	+ / +	+ / +	+ / $	- / -	+ / m
EMF	+ / $	- / -	+ / +	+ / -	- / -	+ / +	+ / -
GIF	$ / $	+ / +	+ / +	+ / +	$ / $	- / -	+ / -
HDF	- / -	- / -	+ / +	+ / +	+ / +	- / -	- / -
JPG	$ / $	+ / +	+ / +	+ / +	+ / $	- / -	m / m
PNG	$ / -	+ / -	+ / +	+ / +	+ / $	+ / +	m / m
PS / EPS (export only)	+ / +	- / +	+ / +	+ / +	- / +	+ / +	+ / +
TIFF	+ / -	+ / -	+ / +	+ / +	+ / $	- / -	m / m
Overall result ***(100% = Best)***	54.44% (9.8/18)	50.00% (9/18)	100.00% (18/18)	94.44% (17/18)	65.56% (11.8/18)	33.33% (6/18)	77.78% (14/18)

4.3. Summarizing the comparison of the graphical functionality

Altogether 40 plot types and diagrams have been listed in the tables above. The following table summarizes the results calculated in percentage with a weighting of 75% for ‘Graphic types’ and 25% for ‘Graphic import/export formats’.

Functions (Version)	**GAUSS** (8.0)	**Maple** (V11)	**Mathematica** (6.0)	**Matlab** (2008a)	**O-Matrix** (6.3)	**Ox Prof.** (5.0)	**Scilab** (4.1.2)
Graphic types (75%)	63.00%	64.50%	79.50%	86.50%	41.00%	47.50%	42.50%
Graphic import/export formats (25%)	54.44%	50.00%	100.00%	94.44%	65.56%	33.33%	72.78%
Overall result (100% = Best)	60.86%	60.88%	84.63%	88.49%	47.14%	43.96%	51.32%

5. Functionality of the programming environment

For a lot of scientists it is very important to define complex models and to do simulations or to define complex mathematical problems as a standalone, ready to use application so that even none trained person can use it for their models. For this type of users it is not only essential to have the facilities of a mathematical program but also a powerful programming environment which provides development tools and interface functionality like debuggers, tracing routines, foreign language interfaces etc.

The following table should give an overview about the supported features of each programming environment.

Programming facilities	**GAUSS**	**Maple**	**Mathematica**	**Matlab**	**O-Matrix**	**Ox Prof.**	**Scilab**
	(8.0)	**(V11)**	**(6.0)**	**(2008a)**	**(6.3)**	**(5.0)**	**(4.1.2)**
Editing features							
Built-in editor[12]	+	+	+	+	+	+	+
External editor configurable	+	+	+	+	-	+	-
Source code formatting	+	-	+	+	-	-	+
Syntax highlighting	+	-	+	+	-	+	+
Command completion	-	+	+	+	-	-	+
Debugging							
Breakpoints	+	+	+	+	+	+	+
Function Tracer	+	+	+	-	-	+	-
Line Tracing	+	-	+	+	+	+	-
Profiler	+	+	$	+	+	-	+
Stack inspection	+	-	+	+	+	-	+
Variable inspection	+	-	+	+	+	+	+
Code advisory / best practice report	-	-	-	+	-	-	-
Language features							
API-interface	+	+	+	+	+	+	+
Compiler metacommands	+	-	-	-	-	+	-
Fuzzy conditional functions	+	-	$	$	-	+	-
GUI programming	-	+	+	+	+	+	+
Loops head / foot controlled	+ / +	+ / +	+ / -	+ / -	+ / -	+ / +	- / +
N-dimensional arrays (> 3)	+	+	+	+	-	+	+
Object oriented programming	-	-	+	+	+	+	+
OLE support	-	+	+	+	-	+	m
P code compiling	+	+	+	+	$	+	+
Web hosting[13]	-	$	$[14]	-	-	-	m[15]

[12] Having an interface where to type, test and run programs has become self-evidence however a few years ago it wasn't.

[13] Webhosting is a way of publishing applications on the web (either Internet or Intranet) by still having a possibility to interact. Mathematical and/or graphical functions are still executable.

Programming facilities	GAUSS	Maple	Mathematica	Matlab	O-Matrix	Ox Prof.	Scilab
	(8.0)	(V11)	(6.0)	(2008a)	(6.3)	(5.0)	(4.1.2)
Language interfaces							
C/C++	+	+	+	+	-	+	+
GAUSS	+	-	-	-	-	+	-
Maple	$	+	-	$	-	-	+
Mathematica	$	+16	+	$	+	-	-
Matlab	-	+	$	+	-	-	+
O-Matrix	-	-	$	-	+	-	-
Ox	-	-	-	-	-	+	-
Scilab	-	-	-	-	-	-	+
DLL-Calls	+	+	+	+	+	+	+
Miscellaneous							
Developer Engine	$	-	-	$	$	+	+
Redistr. with runtime licenses	$	-	-	$	$	$	+
Standalone application compilation	-	-	-	$	-	-	-
Source code optimization	-	-	-	-	-	-	-
Interface to source control system (i.e. Visual SourceSafe)	-	-	-	+	-	-	-
Overall result ***(100% = Best)***	62.70% (23.2/37)	50.81% (18.8/37)	64.86% (24/37)	72.43% (26.8/37)	41.62% (15.4/37)	72.43% (26.8/37)	62.16% (23/37)

It should be mentioned that there are different philosophies how to create and handle program independent applications. One way is to p compile an application and to redistribute it together with a runtime license. Quite a simple handling but mostly not so easy to integrate in the application environment and also the standard control mechanism like the Windows event log will not work. Another possibility is to create "EXE" files which are real Windows standard executables. This makes the application independent from any runtime license issue but it causes mostly high initial costs. A last possibility which is very nice for "real" developers is to integrate the mathematical functionality in own programs over an engine functionality (i.e. DLL file or shared library). Which way of building and distributing an application is up to the usage!

[14] webMathematica which enables web hosting is available for non commercial usage for free.
[15] Available via XMLLab module
[16] Maple can load Mathematica notebooks and convert them to Maple format. Basically this works for complex routines it is not recommendable.

6. Data handling

The data import/export and functionality for transforming and handling data is of significant importance. Imagine a program which offers all needed functions but there is no way to get the analysis data in the program. What is this type of program worth?

6.1. Data import/export options

In most cases the data which is being used for the analysis is based on external data sources like files from spreadsheet programs, databases or any other type of applications. Therefore it is necessary to have interfaces between the mathematical program and the database or spreadsheet program where the original data is located. Most mathematical programs have the possibility to import and export ASCII-based data which of course supposes that you have to convert your original data file into this general format. However it would be much easier to have direct access to the original data. The following table should give an overview of which direct import/export possibilities the tested mathematical programs do have.

Data import/export possibilities	**GAUSS**	**Maple**	**Mathematica**	**Matlab**	**O-Matrix**	**Ox Prof.**	**Scilab**
	(8.0)	**(V11)**	**(6.0)**	**(2008a)**	**(6.3)**	**(5.0)**	**(4.1.2)**
ACCESS	-	-	+	-	-	-	-
ASCII	+	+	+	+	+	+	+
Binary	+	+	+	+	+	+	+
Excel	+	+	+	+	+	+	+
Labview	-	$	$	-	-	-	m
Lotus 1-2-3	+	-	-	+	-	+	-
ODBC / OLE DB / JDBC-connections	m / - / -	- / - / $	+ / - / +	$ / - / $	$ / - / -	- / - / -	- /- / -
XML	-	+	+	+	+	-	-
Overall result ***(100% = Best)***	50.00% (5/10)	56.00% (5.6/10)	78.00% (7.8/10)	66.00% (6.6/10)	48.00% (4.8/10)	40.00% (4/10)	40.00% (4/10)

In some businesses it might be very helpful to have a connection to SAP. Unluckily none of the tested programs do have a real connection for SAP. So for those who need a connection to SAP it should be mentioned that there is an SAP connector available for OLE DB. So having OLE DB connectivity for several programs enables it also to connect to SAP.

6.2. Data preprocessing

Beside the import of the raw data it is also of importance to prepare data for the analysis. Typical steps before doing an analysis are filtering data, sorting, handling of missing values, outliers and several other routines. The following table shows an extraction of some important functions of this type.

Data preprocessing functions	**GAUSS** (8.0)	**Maple** (V11)	**Mathematica** (6.0)	**Matlab** (2008a)	**O-Matrix** (6.3)	**Ox Prof.** (5.0)	**Scilab** (4.1.2)
General							
Amount of columns / rows	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +
Deletion of data by criteria (filter)	+	+	+	+	+	+	+
Deletion of rows / columns by coordinates	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +
Existence of infinity values	+	+	-	+	+	+	+
Matrix editor	+	+	-	+	-	+	+
Missing values existence / replace / delete	+ / + / +	- / - / +	- / - / -	+ / + / +	+ / - / -	+ / + / +	+ / + / +
Recoding	+	-	-	-	-	+	-
Remove duplicate elements	+	-	+	+	-	+	+
Selection of data by criteria (filter)	+	+	+	-	-	+	+
Sorting	+	+	+	+	+	+	+
Matrix constructors							
Band matrix	+	+	+	+	-	+	-
Condition number	+	+	+	+	+	-	+
Diagonalization	+	+	+	+	+	+	+
Difference of 2 vectors	+	+	+	+	-	+	-
Identity matrix	+	+	+	+	+	+	+
Intersection of 2 vectors	+	+	+	+	-	+	+
Merge of matrices and vectors	+	+	+	+	+	+	+
Norm	-	+	+	+	+	+	+
Orthogonalization (constr.)	+	+	+	+	-	-	+
Permutation of columns / rows	- / +	+ / +	+ / +	+ / +	- / -	+ / +	+ / +
Rank	+	+	-	+	+	+	+
Reshape	+	+	+	+	+	+	+
Stack columns of a matrix	+	+	-	+	-	+	+
Submatrix extraction	+	+	+	+	+	+	+
Trace	+	+	+	+	+	+	+
Triangular extraction (upper / lower)	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +
Union of 2 vectors	+	+	+	+	-	+	+
Unstack column of a vector by factor / symetric	- / +	- / -	- / -	- / -	- / -	- / +	- / -
Overall result ***(100% = Best)***	91.43% (32/35)	82.86% (29/35)	71.43% (25/35)	88.57% (31/35)	54.29% (19/35)	91.43% (32/35)	85.71% (30/35)

6.3. Summary

Functions (Version)	GAUSS (8.0)	Maple (V11)	Mathematica (6.0)	Matlab (2008a)	O-Matrix (6.3)	Ox Prof. (5.0)	Scilab (4.1.2)
Data import/export (70%)	50.00%	56.00%	78.00%	66.00%	48.00%	40.00%	40.00%
Data handling and preparation (30%)	91.43%	82.86%	71.43%	88.57%	54.29%	91.43%	85.71%
Overall result (100% = Best)	62.43%	64.06%	76.03%	72.77%	49.89%	55.43%	53.71%

7. Available operating systems

For several types of problems it is important that software is also available for different types of computer platforms. Performance reasons or very high requirements for hardware might make it necessary to solve problems on high equipped workstations or mainframes. Also non objective reasons might have influence on this decision. Today most users work with Windows on Intel or AMD platforms however there are also very popular alternatives like Linux, Mac OS or any type of workstation UNIX. It is therefore necessary to respect these demands as well. The following table gives an overview about the availability of each mathematical program for several common platforms.

Platform (Version)	**GAUSS** (8.0)	**Maple** (V11)	**Mathematica** (6.0)	**Matlab** (2008a)	**O-Matrix** (6.3)	**Ox Prof.** (5.0)	**Scilab** (4.1.2)
HP 9000 (HP-UX)	+	-	+	-	-	+	+
IBM RISC (IBM AIX)	-	-	+	-	-	+	-
Intel / AMD 32 Bit (Windows)	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +
Intel / AMD 64 Bit (Windows)	+ / -	- / -	+ / +	+ / +	- / -	+ / +	- / -
Intel / AMD 32 Bit (Linux)	+ / +	+ / +	+ / +	+ / +	- / -	+ / +	+ / +
Intel / AMD 64 Bit (Linux)	+ / +	+ / +	+ / +	+ / +	- / -	+ / +	- / -
Intel 32 Bit (MAC OS)	+	+	+	+	-	+	+
Intel 64 Bit (MAC OS)	-	+	+	-	-	-	-
SUN (Solaris)	+	+	+	+	-	+	-
Total amount	76.92% (10/13)	69.23% (9/13)	100.00% (13/13)	76.92% (10/13)	15.38% (2/13)	92.31% (12/13)	46.15% (6/13)

The assessment for this part of the test report is also calculated by the key amount of available platforms divided by the total amount of listed platforms and will be displayed in percentage.

8. Speed comparison

The following speed comparison has been performed on a Intel Quad Core Q6600 processor with 2.4 GHz and 2 GB RAM running under Windows Vista Home (all timings are displayed in seconds).

The speed comparison tests 15 functions which are very often used within mathematical models. It is necessary to interpret the timing results in contents with whole models as then small differences in timings of single functions might result in timing differences of minutes up to several hours. However it is not possible to use complete models for this benchmark test as the work for porting the models on each mathematical program and also the running times would be dramatically high.

Functions	**GAUSS**	**Maple**	**Mathematica**	**Matlab**	**O-Matrix**	**Ox Prof.**	**Scilab**
	(8.0)	**(V11)**	**(6.0)**	**(2008a)**	**(6.3)**	**(5.0)**	**(4.1.2)**
IO test and descriptive statistics	2.479	127.483	6.968	10.375	14.583	5.485	21.036
Loop test 15.000 x 15.000	29.427	259.763	323.828	67.258	58.927	32.229	511.323
2000x2000 random matrix^1000	122.829	292.570	31.395	57.393	9.618	15.178	257.784
Sorting of 1000000 random values	67.439	236.543	38.604	14.217	8.985	14.238	36.274
FFT over 1048576 (= 2^20) random values	166.332	47.083	16.864	11.660	7.895	13.386	16.261
Determinant of a 1500x1500 random matrix	504.648	43.456	24.304	31.209	19.615	138.781	56.155
Inverse of a 1500x1500 random matrix	562.355	150.109	74.452	73.247	54.286	342.961	152.899
Eigenvalues of a 1200x1200 random matrix	28.943	27.140	10.562	4.910	4.713	14.484	12.000
Cholesky decomposition of a 1500x1500 random matrix	128.197	414.401	19.638	12.980	14.018	43.515	36.566
1500x1500 cross product matrix	779.001	100.698	48.264	23.459	35.590	107.149	121.302
Calculation of 10000000 Fibonacci numbers	3.334	821.140	2.172	1.434	1.042	2.261	3.193
Principal component factorization over a 10000x1000 matrix	60.578	422.140	37.875	27.107	4.369		51.771
Gamma function on a 1500x1500 random matrix	53.906	10081.174	252.987	27.473	26.585	6.881	26.104
Gaussian error function on a 1500x1500 random matrix	63.755	4815.049	272.294	27.255	1.985	5.327	27.787
Linear regression over a 1000x1000 random matrix	154.978	179.907	10.787	13.798	7.005	43.970	20.646
Overall performance	21.848%	11.159 %	39.072%	54.676%	83.422%	42.364%	24.510%

The overall performance has been calculated in the following way:

The best timing result of a benchmark function makes 100%; for calculating the results for each function I'll take the fastest timing and divide it by the timing of the tested program (*the formula will look MIN(A1;A2;…)/A2 for example*) and that makes the ranking in percentage. To

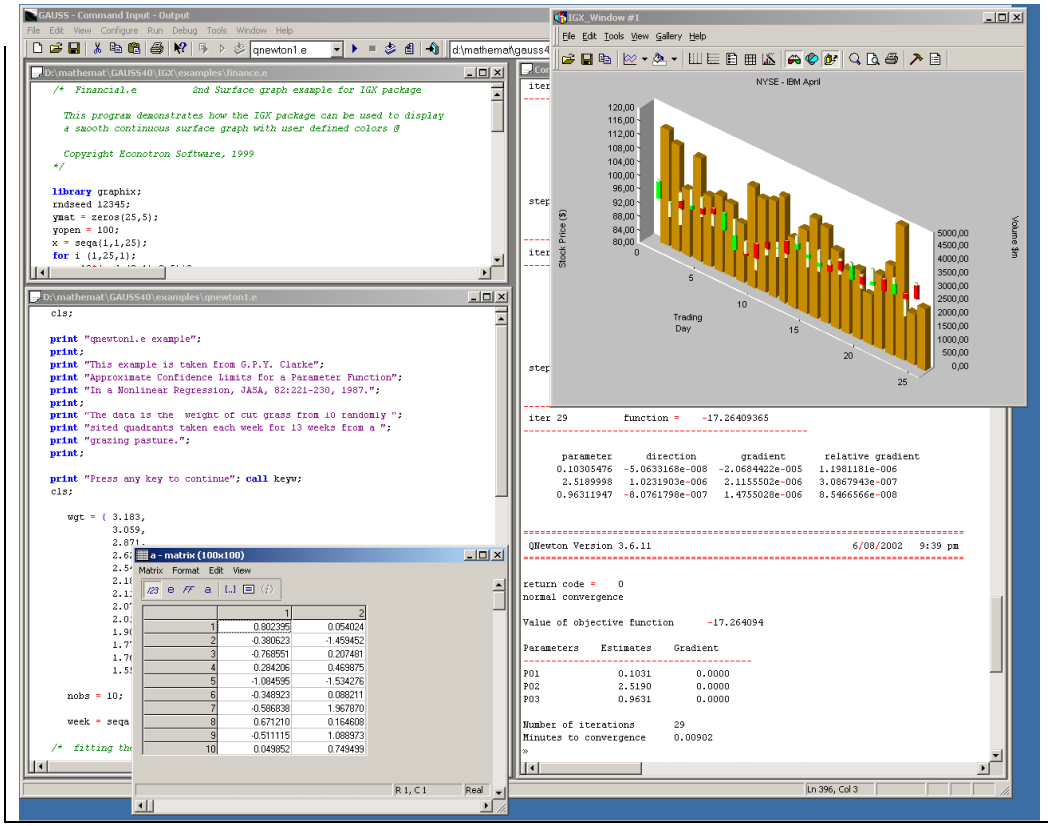
calculate the final „Overall performance” I have then added the percentage values for each tested program and divided it by the amount of tested functions (*in the moment 15*) which gives again a result as a percentage.

Functions which are not supported by a program have not been judged in these calculations and have therefore no influence on the overall timing result. Also the realization of each function for each mathematical program has been optimized as far as it has been possible. Also it is necessary to mention, that the performance tests should reflect an ordinary work which means with no specific compilation of the code and by avoiding any kind of “remembrance” functionality. If a compile function would be used most especially CAS could increase since their internal integer routines would automatically switch to floating point usage.

9. Summary and interpretation of the test results

9.1. General product information

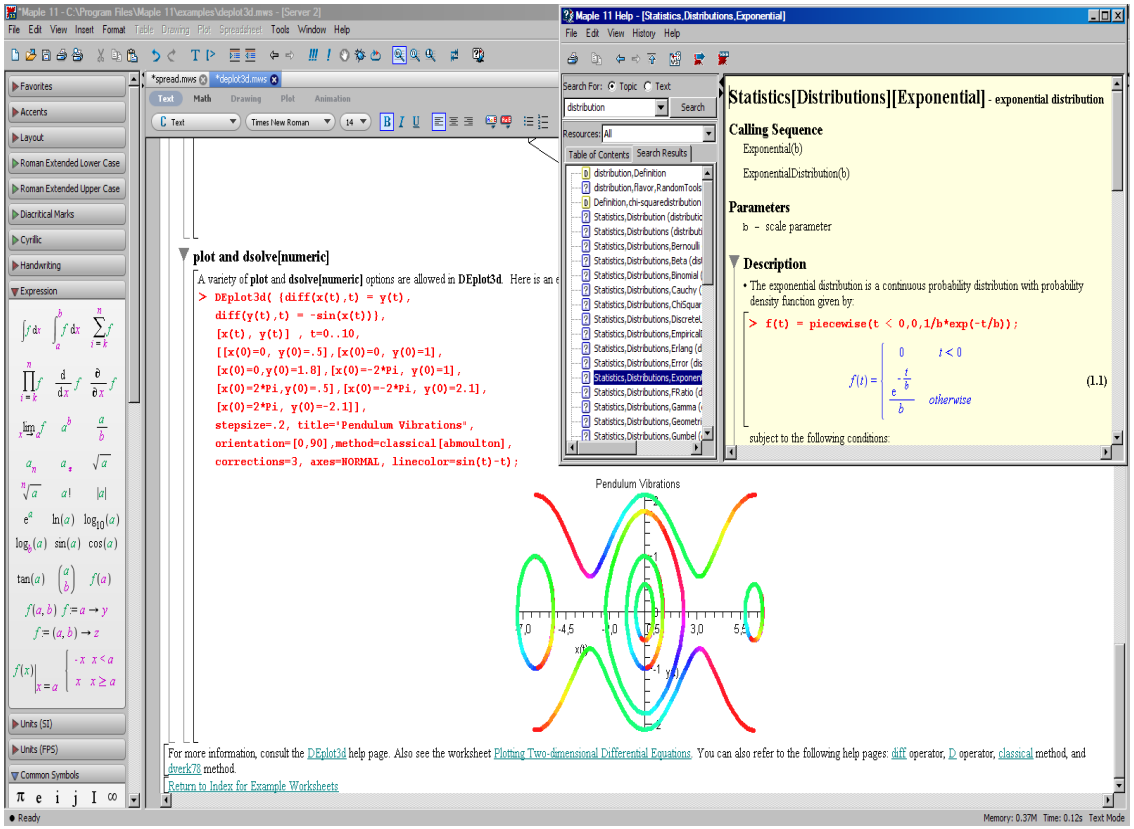
9.1.1. GAUSS



Product	GAUSS
Version	8.0
Producer	Aptech Systems Inc.
Location of head-quarter	Maple Valley, USA
Internet	www.aptech.com
Price (commercial)	≈ 2799 US$ (2969 €)
Price (academic)	≈ 707 US$ (708 €)
Price (student)[17]	≈ 49 US$ (52 €)
FAQ list	FAQ list exists at the website of Aptech Systems but there are not much information available.
Mailing lists	Available (Subscribe GAUSSIANS at majordomo@eco.utexas.edu)
Newsgroup	---
Archives	Several archives are available i.e. http://gurukul.ucc.american.edu/econ/gaussres/GAUSSIDX.HTM
Books	Only a very small amount of additional books are available.
Remark	GAUSS has very long tradition. Its development started in the 80s and the focus is on its programming facilities, speed and economic functionality.

[17] Light version is memory limited. Student version is a light version.

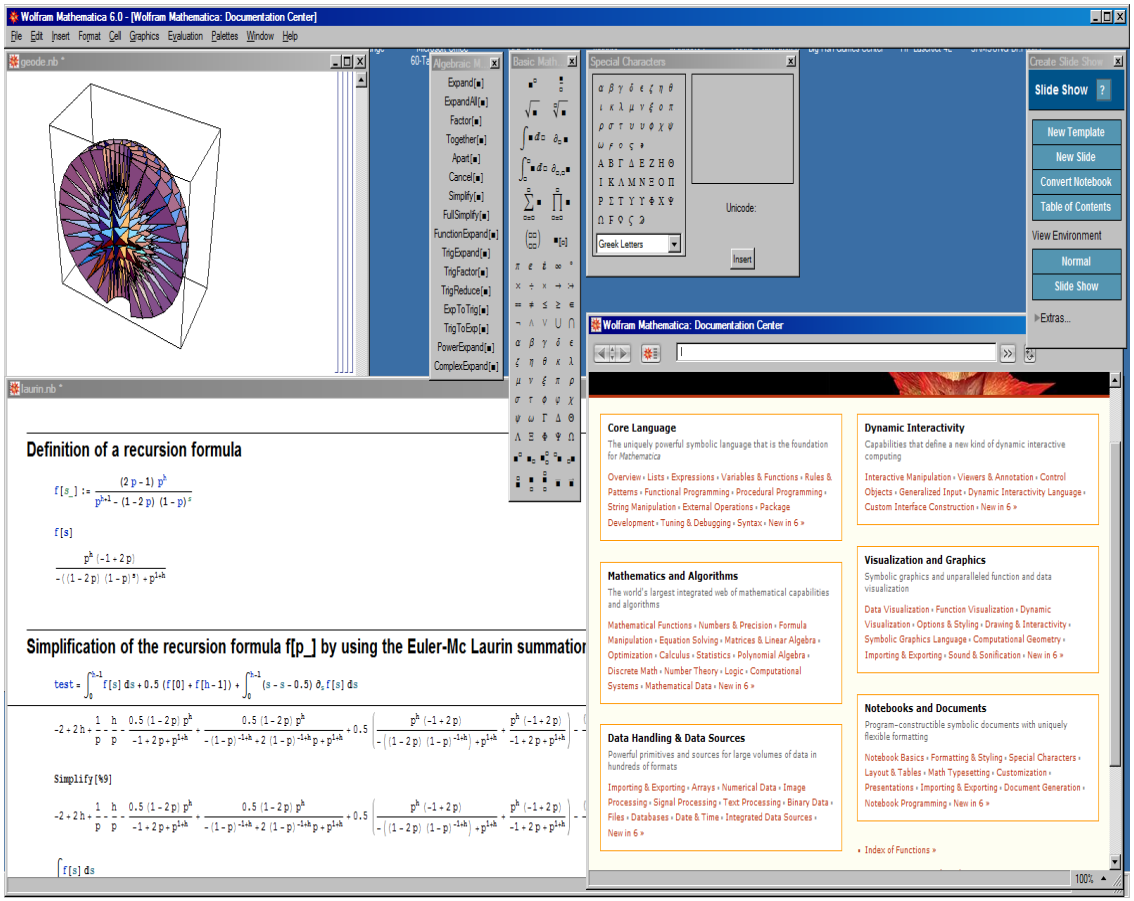
9.1.2. Maple



Product	Maple
Version	11
Producer	Waterloo Maple Software Inc.
Location of head-quarter	Waterloo, Canada
Internet	www.maplesoft.com
Price (commercial)	1895 US$ (1995 €)
Price (academic)[18]	995 US$ (995 €)
Price (student)	99 US$ (159 €)
FAQ list	A very informative FAQ list is available over the website of the producer.
Mailing lists	Several mailing lists are available i.e. MUG (Subscribe by Subscribe maple-list at majordomo@daisy.uwaterloo.ca)
Newsgroup	Available at comp.soft.sys.math.maple
Archives	Archives could be found on a lot of academic websites. The software producer itself supports an archive at: http://www.mapleapps.com/
Books	A huge amount of books for every kind of subject around Maple are available from nearly every big publishing company.
Remark	Maple is already for a very long time on the market. Its development is focused on the CAS sector and is believed to be one of the market leaders.

[18] Different pricing for research and academic.

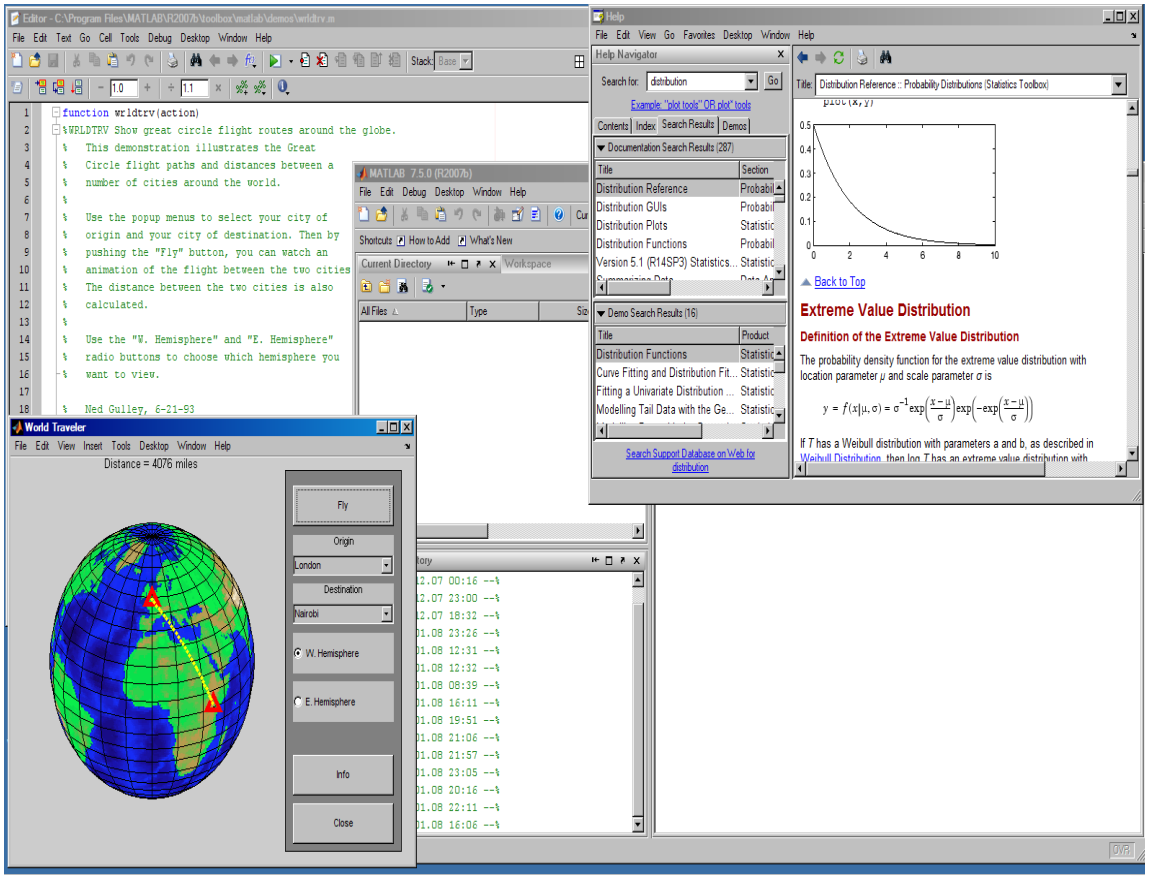
9.1.3. Mathematica



Product	Mathematica
Version	6.0
Producer	Wolfram Research Inc.
Location of head-quarter	Champaign, USA
Internet	www.wolfram.com
Price (commercial)	2495 US$[19] (3790 €)
Price (academic)	1095 US$ (1600 €)
Price (student)	140 US$ (152 €)
FAQ list	A very informative FAQ list is available over the website of the producer.
Mailing lists	Several mailing list are available i.e. DMUG (Subscribe by Subscribe dmug at majordomo@mathematica.ch)
Newsgroup	Available at comp.soft.sys.mathematica
Archives	Archives could be found on a lot of academic websites. The software producer itself supports an archive at: http://mathworld.wolfram.com/
Books	A huge amount of books for every kind of subject around Mathematica are available from nearly every big publishing company.
Remark	Similar like Maple is Mathematica one of the longest available CAS on the market (since 1988). It is also one of the market leaders, with the aim of supporting functionality and competence on several subjects, it is also a CAS which commonly usable.

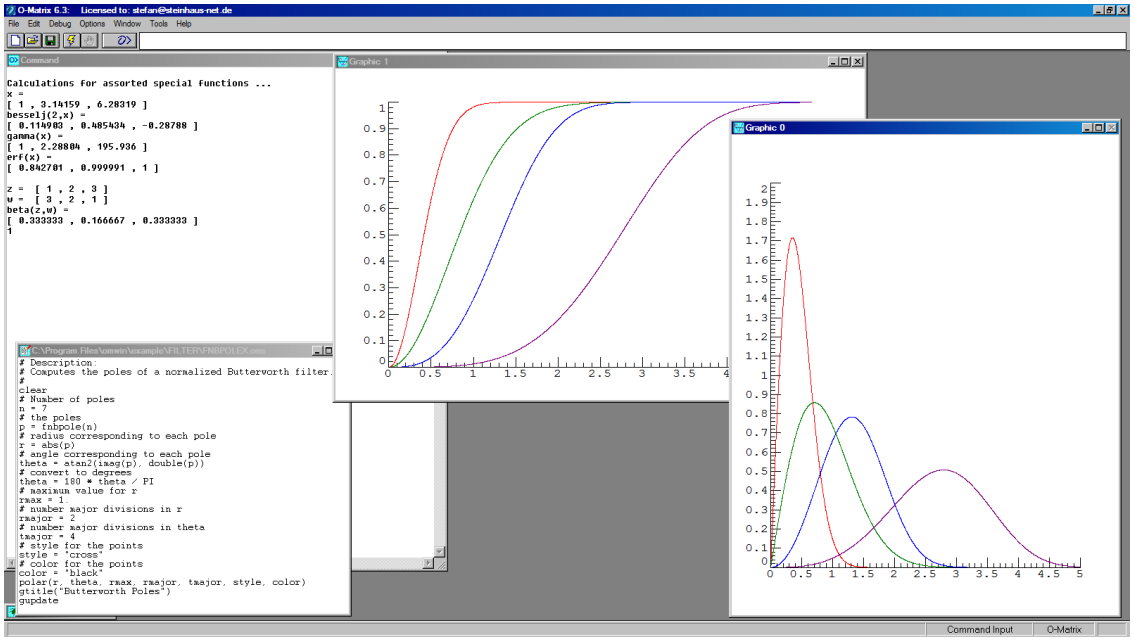
[19] Commercial price in the USA and Canada includes 1 year of Premier Service.

9.1.4. Matlab



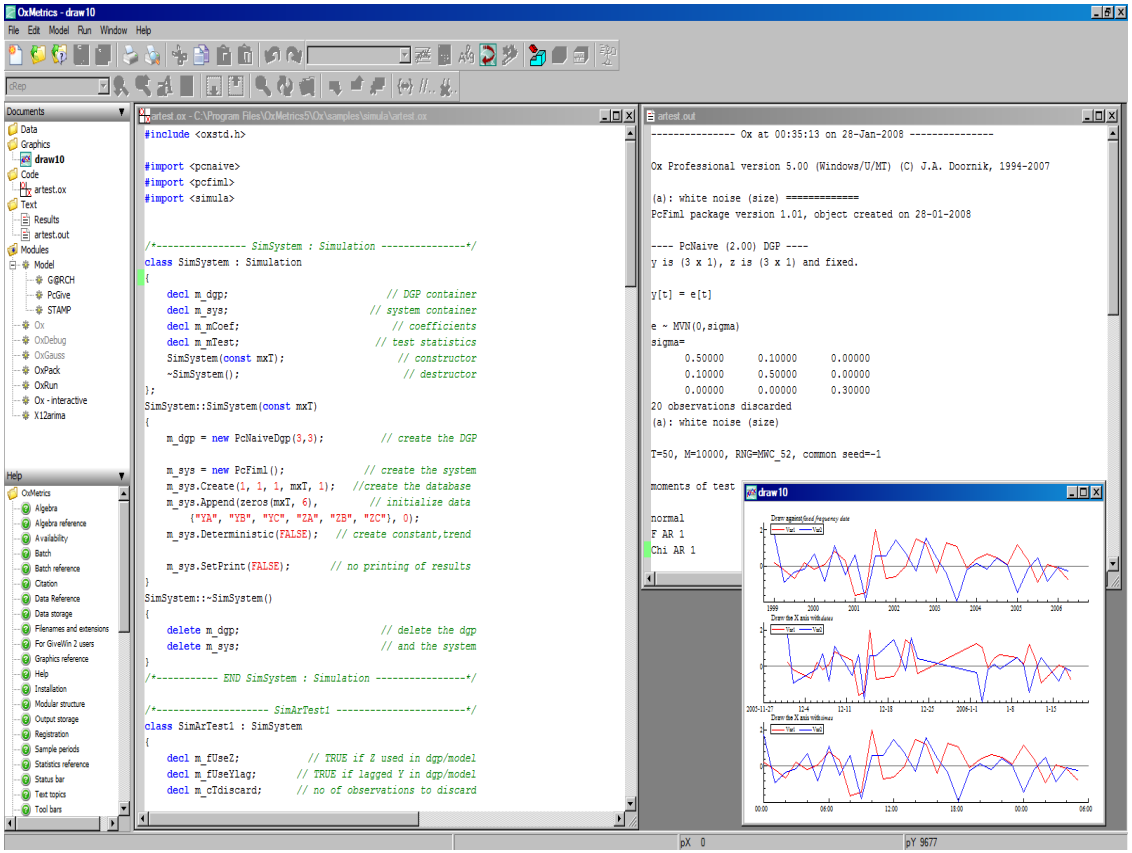
Product	Matlab
Version	2008a
Producer	The Mathworks Inc.
Location of head-quarter	Natick, USA
Internet	www.mathworks.com
Price (commercial)	1900 US$ (1950 €)
Price (academic)	varies
Price (student)	99 US$ (90 €)
FAQ list	In general a support FAQ list is available over Mathworks website but there are also several other available i.e. t http://www.mit.edu/%7Epwb/cssm/
Mailing lists	---
Newsgroup	Available at comp.soft-sys.matlab
Archives	Best archive for Matlab routines and programs are available at http://www.mathtools.net/ or http://www.mathworks.com/matlabcentral but archives are also widely available at Universities and commercial institutions.
Books	A huge amount of books for every kind of subject around Matlab are available from nearly every big publishing company.
Remark	The numerical mathematics program Matlab started its development in 1984 with mainly technical aims. Today Matlab offers functions and competence for nearly all important topics.

9.1.5. O-Matrix



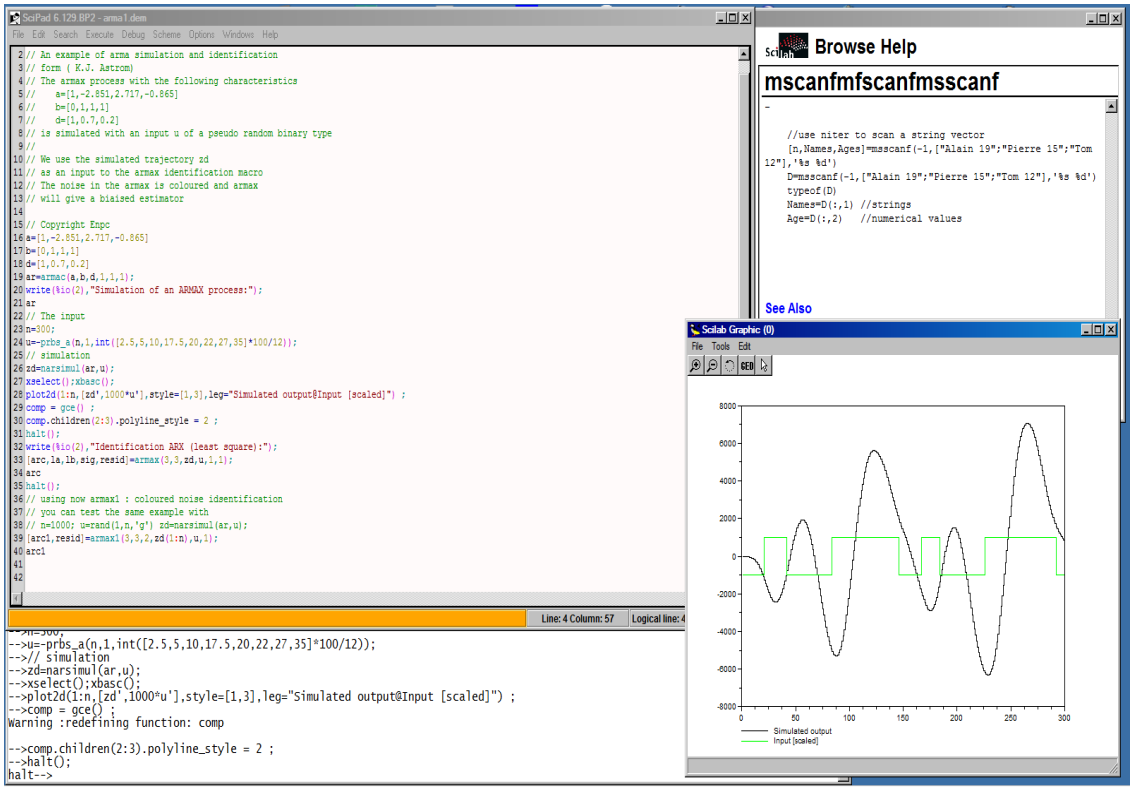
Product	O-Matrix
Version	6.3
Producer	Harmonics Software Inc.
Location of head-quarter	Breckenridge, USA
Internet	www.omatrix.com
Price (commercial)	265 US$
Price (academic)	145 US$
Price (student)	---
FAQ list	---
Mailing lists	---
Newsgroup	Newsgroup is available at http://tech.groups.yahoo.com/group/omatrix/
Archives	---
Books	---
Remark	O-Matrix is a very young product which is specialized for technical usage. Harmonic Software likes to compare O-Matrix with Matlab and sees its strength in the excellent performance, technical routines and its compatibility to Matlab.

9.1.7. Ox Professional



Product	Ox Professional
Version	5.0
Producer	OxMetrics
Location of head-quarter	United Kingdom
Internet	www.oxmetrics.com
Price (commercial)	2.000 US$ (1.500 €)
Price (academic)	500 US$ (375 €)
Price (student)	–
FAQ list	A comprehensive FAQ list is available at http://www.doornik.com/support.html
Mailing lists	A mailing list is available. Message posting is available for those who have joined the list at ox-users@jiscmail.ac.uk. To join the list simply send an email with the message body join ox-users firstname lastname.
Newsgroup	---
Archives	---
Books	A small amount of books are available mostly provided by one the developers of the product. All of them are available over the OxMetrics bookshop.
Remark	Ox Prof. is a combination of the products Ox professional, PcGive, G@rch and STAMP, specialized on economic topics. Its strength is the very good performance and a big range of routines for econometrics. The combined products are also available separately as single products. Ox Professional has been the subject of previous editions of this test report.

9.1.8. Scilab



Product	Scilab
Version	4.1.2
Producer	INRIA
Location of head-quarter	France
Internet	www.scilab.org
Price (commercial)	free
Price (academic)	free
Price (student)	free
FAQ list	---
Mailing lists	A majordomo mailing list is available. Messages are processed over users@lists.scilab.org. A subscription is possible by sending “Just click SEND” to users-subscribe@lists.scilab.org.
Newsgroup	Available at comp.soft-sys.math.scilab
Archives	As Scilab is a free available product there are also several users providing their developments for the community. Searching in Internet there are several packages available but first choice would be to look at the website of INRIA itself. Available at http://www.scilab.org/contrib/ there are already several packages available.
Books	Beside several e-Books there are also a few amounts of commercially distributed books available i.e. over Springer or Birkhauser publishing.
Remark	Scilab is a Freeware product specialized on numerical mathematics. For a freely available product it offers really a wide range of functionality.

9.1. Miscellaneous information

Several information like pricing, support, newsgroups, books, etc. are of significant importance for users of mathematical or statistical software. Due to the fact that this type of information cannot be characterized objectively I will only mention them without a judgment for the final summary of the test report. Also it should be mentioned that the following marks are not comparable to previous editions of this test report as it is a non objective evaluation against today's technical standards. It's up to each reader of this report to make his/her own decisions on this information

Functions (Version)	**GAUSS** (8.0)	**Maple** (V11)	**Mathematica** (6.0)	**Matlab** (2008a)	**O-Matrix** (6.3)	**Ox Prof.** (5.0)	**Scilab** (4.1.2)
Operation / Programming handling							
User interface	3	2	2	2	3	3	3
Graphics	2[20]	2	2	2	5	3	4
Programming language (similar to)	2 (Basic, Fortran)	2 (Pascal)	2 (Lisp, APL)	2 (Basic, Fortran)	2 (Basic, Fortran)	2 (C, C++)	2 (Basic, Fortran)
Program archives							
Program archives by the software producer	6	2	1	2	4	4	3
Program archives by external institutions	2	1	1	3	5	4	3

The information in this table are judge by school marks from 1 until 6 (1 - best, 6 - worst) and represent my own subjective opinion. The mark 6 normally means that something is not supported the fact that something is supported so badly that the mark 6 is justified never occurred. Conversely it was mostly also not justified to give the mark 1 as only very rare features are supported so excellent.

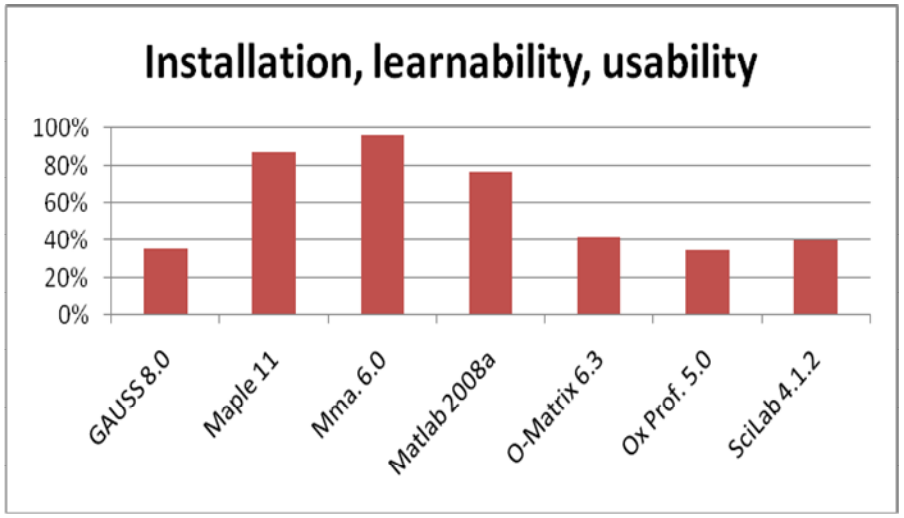
[20] GAUSS deserved the mark 2 for graphics only if used together with the module IGX or GAUSSPlot otherwise it should be 4.

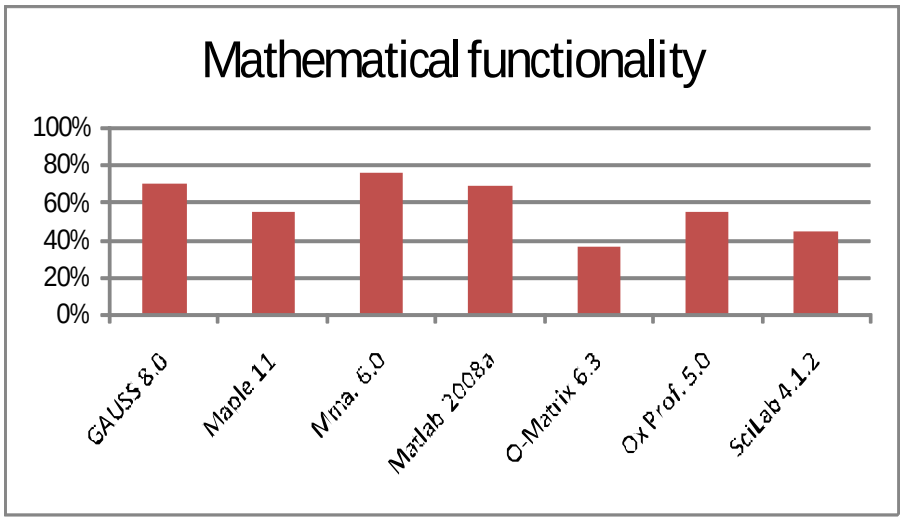
9.2. Summary

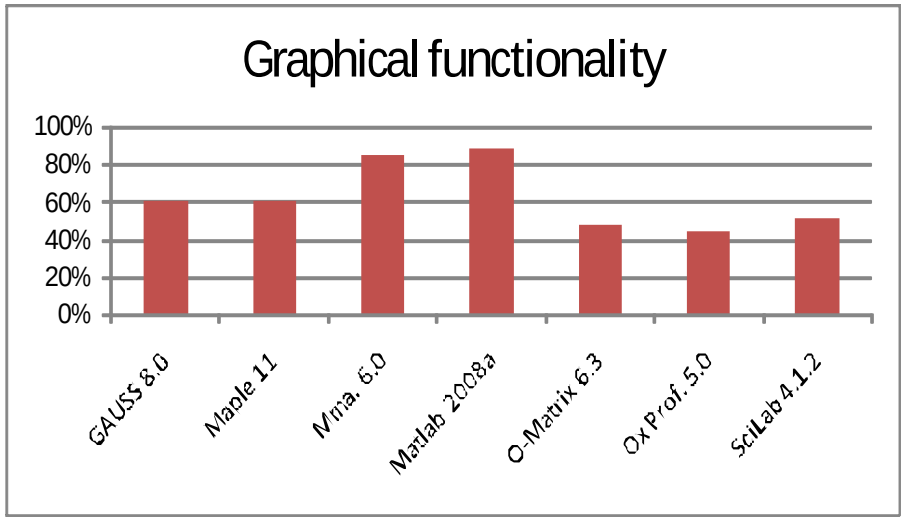
The summary should set the results of the speed comparison, the functionality of the programming environment, the data import/export facilities and the availability for different platforms in relation to the results of the comparison of the mathematical and graphical functionality. Also it includes a first attempt to describe and validate the “effects” of installation, learnability and usability.

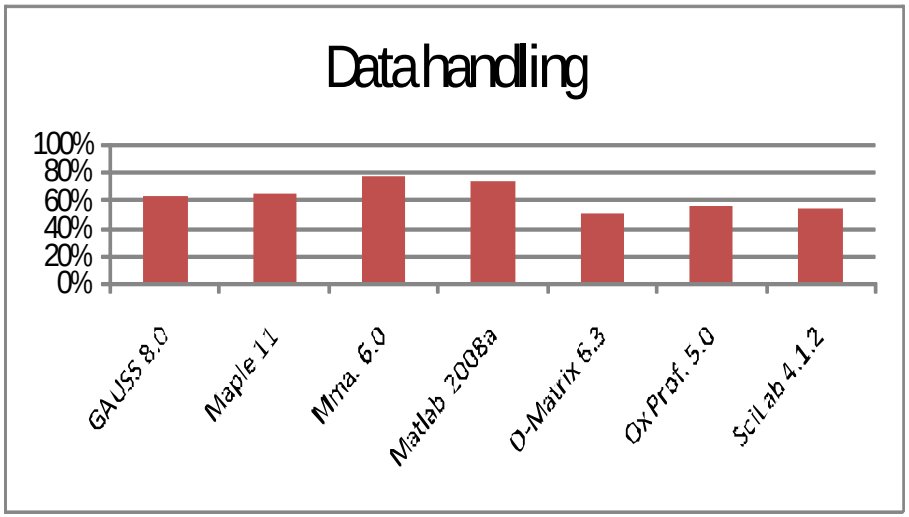
Test	GAUSS (8.0)	Maple (V11)	Mathematica (6.0)	Matlab (2008a)	O-Matrix (6.3)	Ox Prof. (5.0)	Scilab (4.1.2)
Installation, learnability and usability (15%)	35.41%	87.54%	96.27%	76.52%	41.18%	34.36%	39.69%
Comparison of the mathematical functionality (35%)	69.56%	55.10%	76.04%	68.79%	36.58%	54.38%	44.63%
Comparison of the graphical functionality (10%)	60.86%	60.88%	84.63%	88.49%	47.14%	43.96%	51.32%
Functionality of the programming environment (11%)	62.70%	50.81%	64.86%	72.43%	41.62%	72.43%	62.16%
Data handling (5%)	62.43%	64.06%	76.03%	72.77%	49.89%	55.43%	53.71%
Available platforms (2%)	76.92%	69.23%	100.00%	76.92%	15.38%	92.31%	46.15%
Speed comparison (22%)	21.85%	11.16%	39.07%	54.68%	83.42%	42.36%	24.51%
Overall result	52.11%	51.13%	71.05%	69.58%	49.43%	50.49%	42.54%

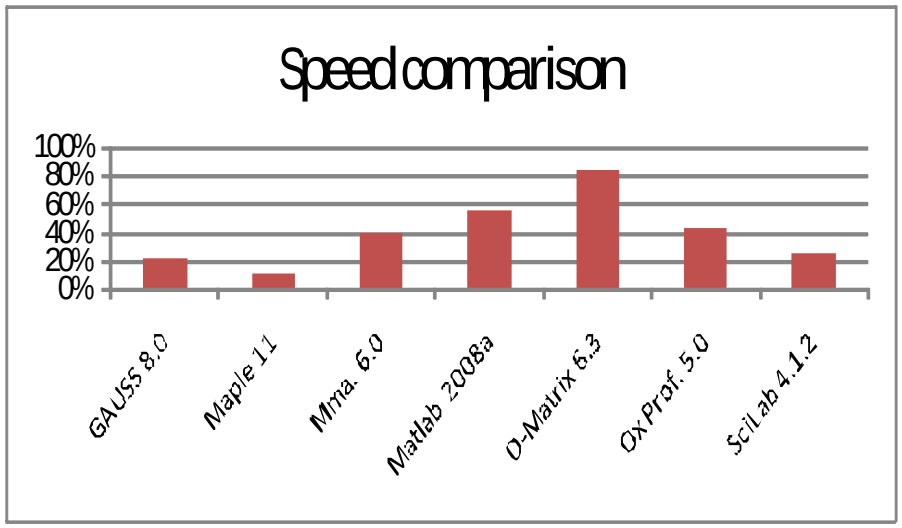
Note: The overall results of some tested programs are pretty bad due to the specific weighting of this test report. I would like to mention that this does of course not mean that the software is bad in general but that the programs are maybe not perfect for the specific usage mentioned in this test report, for other weightings/usages they might be much better or even leading.

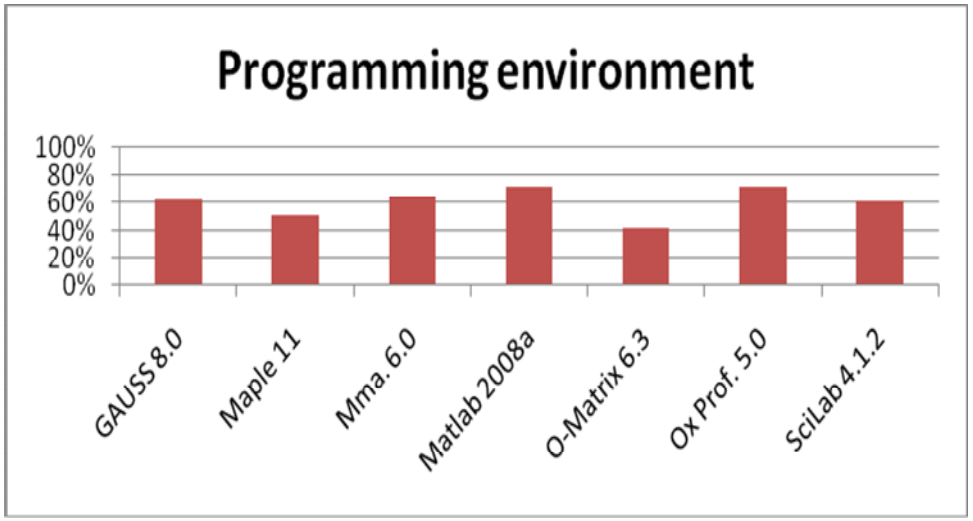

Installation, learnability, usability
100%
80%
60%
40%
20%
0%
GAUSS 8.0
Maple 11
Mma. 6.0
Matlab 2008a
O-Matrix 6.3
Ox Prof. 5.0
SciLab 4.1.2

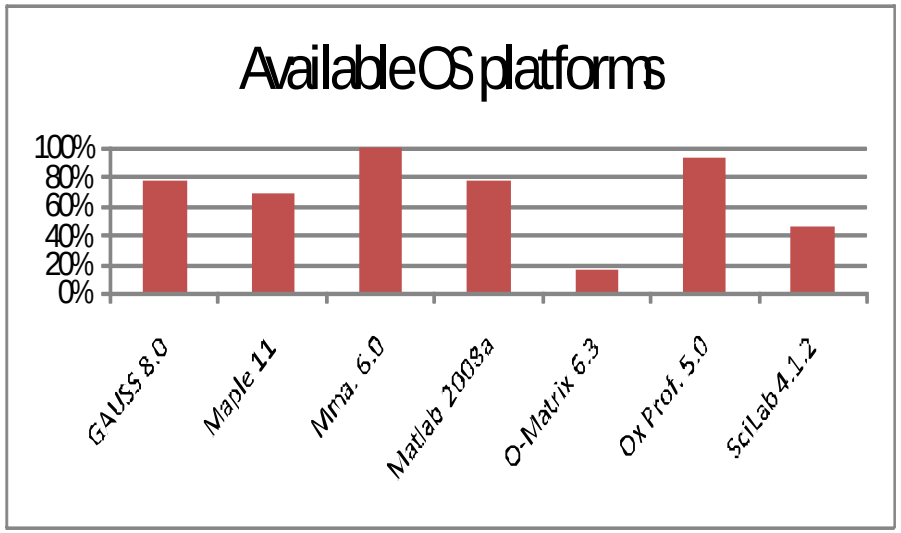

Mathematical functionality
100%
80%
60%
40%
20%
0%
GAUSS 8.0
Maple 11
Mma. 6.0
Matlab 2008a
O-Matrix 6.3
Ox Prof. 5.0
SciLab 4.1.2

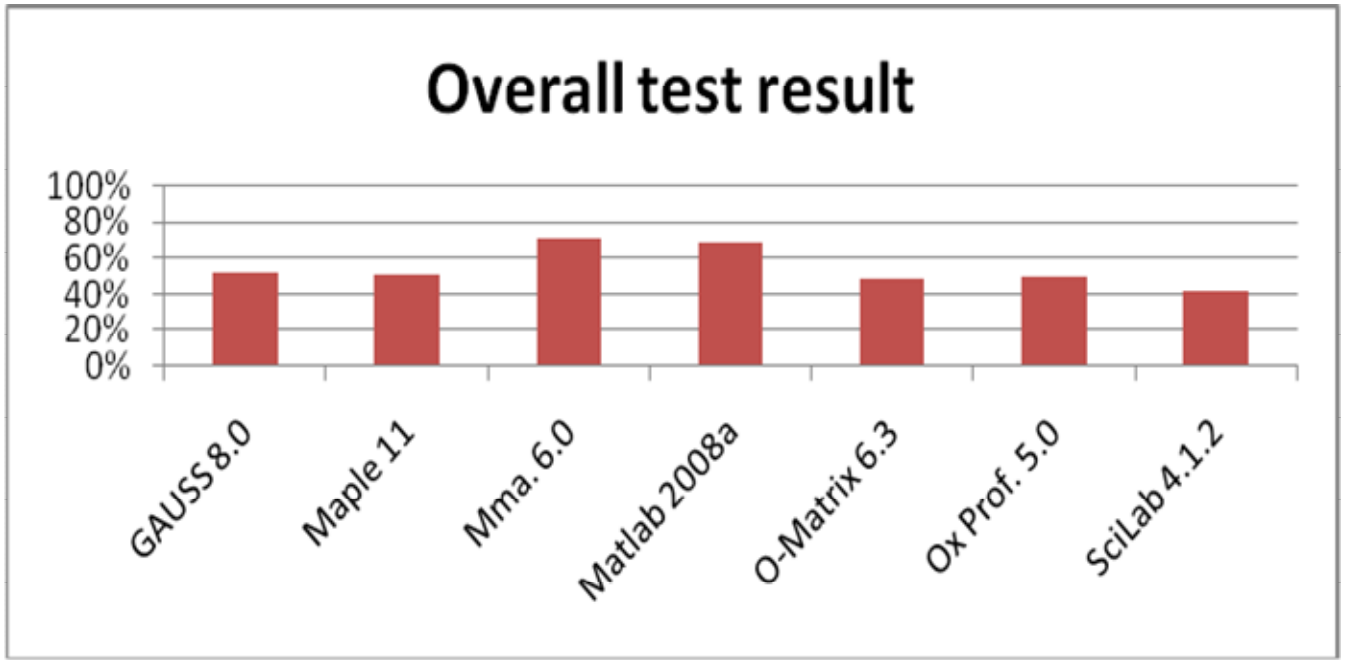

Graphical functionality
100%
80%
60%
40%
20%
0%
GAUSS 8.0
Maple 11
Mma. 6.0
Matlab 2008a
O-Matrix 6.3
Ox Prof. 5.0
SciLab 4.1.2


Data handling
100%
80%
60%
40%
20%
0%
GAUSS 8.0
Maple 11
Mma. 6.0
Matlab 2008a
O-Matrix 6.3
Ox Prof. 5.0
SciLab 4.1.2


Speed comparison
100%
80%
60%
40%
20%
0%
GAUSS 8.0
Maple 11
Mma. 6.0
Matlab 2008a
O-Matrix 6.3
Ox Prof. 5.0
SciLab 4.1.2


Programming environment
100%
80%
60%
40%
20%
0%
GAUSS 8.0
Maple 11
Mma. 6.0
Matlab 2008a
O-Matrix 6.3
Ox Prof. 5.0
SciLab 4.1.2


Available OS platforms
100%
80%
60%
40%
20%
0%
GAUSS 8.0
Maple 11
Mma. 6.0
Matlab 2008a
O-Matrix 6.3
Ox Prof. 5.0
SciLab 4.1.2



Other mathematical programming languages: At this point I would like to mention that beside the mathematical programs which I tested there are also some more interesting freeware and commercial products. Unluckily there have been several reasons why I couldn't include them into my current test report. Mostly the problem was that I didn't receive any support for my work by the software producers. However I think it is nevertheless worth to mention them at this point. The following software products are, as far as I believe, also of interest:

- Gretl (Freeware available at http://gretl.sf.net)
- MathView by MathWizards Inc. (http://www.mathwizards.com)
- Maxima (Freeware available at http://maxima.sourceforge.net/)
- Octave (Freeware available at http://bevo.che.wisc.edu/octave/)
- R (Freeware available at http://stat.auckland.ac.nz/r/r.html)
- S-Plus[21] (http://www.insightful.com)

[21] S-Plus has been tested in previous editions of this test report. Due to organizational changes it was impossible to find a supporting person neither by any distributor nor the software producer itself.

Appendix A: Listings of the benchmark tests

GAUSS Benchmark program:

```
/*************************************************************************/
/* GAUSS Benchmark program version 5.0                                   */
/* Author : Stefan Steinhaus                                             */
/* EMAIL  : stefan@steinhaus-net.de                                      */
/* This program is public domain. Feel free to copy it freely.           */
/*************************************************************************/

/* Test of IO function - Reading of data from ASCII file */
/* Test of Extraction of submatrices and calculation of descriptive stats */
/* Doing the whole process 2000 times do get higher timings              */

proc 1 = IOTestASCII(runs);
  local result,zeit,timing,i,j,k;
  local fname,datmat,ColName;
  local Year_Data, Limits_Area;
  local Min_Year,Max_Year,Mean_Year,GainLoss_Year;
  result=0;
  Limits_Area={1,261,522,784,1045,1305,1565,1827,2088,2349,2610,2871,3131,3158};
  fname="d:\\mathematik\\ncrunch5\\currency2.txt";
  _dxaschdr=1; _dxprint=0;
  for i (1,runs,1);
    timing=0; zeit=hsec;
    {datmat,ColName}=import(fname,0,0);
    for k (1,2000,1);
      for j (1,13,1);
        Year_Data=datmat[Limits_Area[j]:Limits_Area[j+1]-1,4:37];
        Min_Year=minc(Year_Data);
        Max_Year=maxc(Year_Data);
        Mean_Year=meanc(Year_Data);
        GainLoss_Year=100-
(Year_Data[1,.]+0.00000001)./(Year_Data[rows(Year_Data),.]+0.00000001)*100;
      endfor;
    endfor;
    timing=(hsec-zeit)/100;
    result=result+timing;
  endfor;
  retp(result/runs);
endp;

/* Test of 2 loops 15000x15000 */

proc 1 = LoopTest(runs);
  local a,b,result,i,timing,zeit;
  result=0;
  for i (1,runs,1);
    a=1;
    timing=0;
    zeit=hsec;
    for x (1,15000,1);
      for y (1,15000,1);
        a=a+x+y;
      endfor;
    endfor;
    timing=(hsec-zeit)/100;
    result=result+timing;
  endfor;
  retp(result/runs);
endp;

/* 2000x2000 normal distributed random matrix^1000 */

proc 1 = CreationMatrix(runs);
  local a,b,result,i,j,timing,t;
  result=0;
  for i (1,runs,1);
    timing=0;
    for j (1,200,1);
      b=1+(rndu(2000,2000)/100);
      t=hsec;
      a=b^1000;
      timing=timing+hsec-t;
    endfor;
    timing=timing/100;
    result=result+timing;
  endfor;
  retp(result/runs);
endp;

/* 100 x Sorting of 1000000 values */

proc 1 = SortFunc(runs);
  local a,b,result,i,j,timing,t;
  result=0;
  for i (1,runs,1);
    timing=0;
    for j (1,100,1);
      a=rndu(1000000,1);
      t=hsec;
      b=sortc(a,1);
```

```
      timing=timing+hsec-t;
    endfor;
    timing=timing/100;
    result=result+timing;
  endfor;
  retp(result/runs);
endp;

/* 100 x FFT of 1048576 random values (2^20) */

proc 1 = FFTFunc(runs);
  local a,b,result,i,j,timing,t;
  result=0;
  for i (1,runs,1);
    timing=0;
    for j (1,100,1);
      a=rndu(1048576,1);
      t=hsec;
      b=fft(a);
      timing=timing+hsec-t;
    endfor;
    timing=timing/100;
    result=result+timing;
  endfor;
  retp(result/runs);
endp;

/* 100 x Determinant of a 1500x1500 matrix */

proc 1 = Determinant(runs);
  local a,b,result,i,j,timing,t;
  result=0;
  for i (1,runs,1);
    timing=0;
    for j (1,100,1);
      a=rndu(1500,1500);
      t=hsec;
      b=det(a);
      timing=timing+hsec-t;
    endfor;
    timing=timing/100;
    result=result+timing;
  endfor;
  retp(result/runs);
endp;

/* 100 x Inverse of a 1500x1500 matrix */

proc 1 = InverseFunc(runs);
  local a,b,result,i,j,timing,t;
  result=0;
  for i (1,runs,1);
    timing=0;
    for j (1,100,1);
      a=rndu(1500,1500);
      t=hsec;
      b=inv(a);
      timing=timing+hsec-t;
    endfor;
    timing=timing/100;
    result=result+timing;
  endfor;
  retp(result/runs);
endp;

/* Eigenvalues of a 1200x1200 matrix */

proc 1 = EigenvaluesFunc(runs);
  local a,b,result,i,timing,zeit;
  result=0;
  for i (1,runs,1);
    timing=0;
    a=rndu(1200,1200);
    zeit=hsec;
    b=eig(a);
    timing=(hsec-zeit)/100;
    result=result+timing;
  endfor;
  retp(result/runs);
endp;

/* 100 x Cholesky decomposition of a 1500x1500-matrix */

proc 1 = CholeskyFunc(runs);
  local a,b,result,i,j,timing,t;
  result=0;
  for i (1,runs,1);
    timing=0;
    for j (1,100,1);
      a=moment(rndu(1500,1500),0);
      t=hsec;
      b=chol(a);
      timing=timing+hsec-t;
    endfor;
    timing=timing/100;
    result=result+timing;
  endfor;
  retp(result/runs);
```

```
endp;

/* 100 x 1500x1500 cross-product matrix */

proc 1 = CrossProductFunc(runs);
  local a,b,result,i,j,timing,t;
  result=0;
  for i (1,runs,1);
    timing=0;
    for j (1,100,1);
        a=rndu(1500,1500);
        t=hsec;
        b=moment(a,0);
        timing=timing+hsec-t;
    endfor;
    timing=timing/100;
    result=result+timing;
  endfor;
  retp(result/runs);
endp;

/* Calculation of 10000000 fibonacci numbers */

proc 1 = FibonacciFunc(runs);
  local a,b,phi,result,i,timing,zeit;
  result=0;
  phi = 1.6180339887498949;
  for i (1,runs,1);
    timing=0;
    a=floor(1000*rndu(10000000,1));
    zeit=hsec;
    b=(phi^a-(-phi)^(-a))/sqrt(5);
    timing=(hsec-zeit)/100;
    result=result+timing;
  endfor;
  retp(result/runs);
endp;

/* Principal Component Analysis over a 10000x1000 matrix */

proc 1 = PCAFunc(runs);
  local a,p,v,result,i,timing,zeit;
  result=0;
  for i (1,runs,1);
    a=rndu(10000,1000);
    timing=0;
    zeit=hsec;
    {p,v,a}=princomp(a,1000);
    timing=(hsec-zeit)/100;
    result=result+timing;
  endfor;
  retp(result/runs);
endp;

/* 100 x Gamma function over a 1500x1500 matrix */

proc 1 = GammaFunc(runs);
  local a,b,result,i,j,timing,t;
  result=0;
  for i (1,runs,1);
    timing=0;
    for j (1,100,1);
        a=rndu(1500,1500);
        t=hsec;
        b=gamma(a);
        timing=timing+hsec-t;
    endfor;
    timing=timing/100;
    result=result+timing;
  endfor;
  retp(result/runs);
endp;

/* 100 x Gaussian error function over a 1500x1500 matrix */

proc 1 = GaussianFunc(runs);
  local a,b,result,i,j,timing,t;
  result=0;
  for i (1,runs,1);
    timing=0;
    for j (1,100,1);
        a=rndu(1500,1500);
        t=hsec;
        b=erf(a);
        timing=timing+hsec-t;
    endfor;
    timing=timing/100;
    result=result+timing;
  endfor;
  retp(result/runs);
endp;

/* 100 x Linear regression over 1000x1000 matrix */

proc 1 = LinearRegressionFunc(runs);
  local a,b,result,i,j,timing,t;
  result=0;
  for i (1,runs,1);
```

```
   timing=0;
   for j (1,100,1);
      a=rndu(1000,1000);
      b=seqa(1,1,1000);
      t=hsec;
      b=olsqr(b,a);
      timing=timing+hsec-t;
   endfor;
   timing=timing/100;
   result=result+timing;
 endfor;
 retp(result/runs);
endp;


/* Main program */

cls;
format /RD 8,3;
runs=3;             /* Amount of runs for avg. timing -> suggestion at least 2 */
datmat=0;
ver=sysstate(1,0);
print; print; print;
print "               !!! GAUSS - Benchmark program !!!";
print "               ================================"; print; print;
print "Platform: GAUSS " ftos(ver[1],"%*lf",2,0) "." ftos(ver[2],"%*lf",1,0) "." ftos(ver[3],"%*lf",2,0);
print "Displaying the avg. timings over " runs " runs.";
print; print;
print "Function / timing [sec.]                                    result (sec.)";
print "-------------------------------------------------------------------------------";

{timing}=IOTestASCII(runs);
print /flush "IO test and descriptive statistics_____________________ : " timing;
{timing}=LoopTest(runs);
print /flush "Test of loops__________________________________________ : " timing;
{timing}=CreationMatrix(runs);
print /flush "2000x2000 uniform distributed random matrix^1000_______ : " timing;
{timing}=SortFunc(runs);
print /flush "1000000 values sorted ascending________________________ : " timing;
{timing}=FFTFunc(runs);
print /flush "FFT over 1048576 values (2^20)_________________________ : " timing;
{timing}=Determinant(runs);
print /flush "Determinant of a 1500x1500 random matrix_______________ : " timing;
{timing}=InverseFunc(runs);
print /flush "Inverse of a 1500x1500 uniform distr. random matrix____ : " timing;
{timing}=EigenvaluesFunc(runs);
print /flush "Eigenvalues of a uniform distr. 1200x1200 randommatrix_ : " timing;
{timing}=CholeskyFunc(runs);
print /flush "Cholesky decomposition of a 1500x1500-matrix__________ : " timing;
{timing}=CrossProductFunc(runs);
print /flush "1500x1500 cross-product matrix_____________________ : " timing;
{timing}=FibonacciFunc(runs);
print /flush "Calculation of 10000000 fibonacci numbers_____________ : " timing;
{timing}=PCAFunc(runs);
print /flush "Principal component analysis over a 10000x1000 matrix_ : " timing;
{timing}=GammaFunc(runs);
print /flush "Gamma function over a 1500x1500 matrix_____________ : " timing;
{timing}=GaussianFunc(runs);
print /flush "Gaussian error function over a 1500x1500 matrix_____ : " timing;
{timing}=LinearRegressionFunc(runs);
print /flush "Linear regression over a 1000x1000 matrix_____________ : " timing;
end;
```

Maple Benchmark program:

```
Maple Benchmarktest

(Version 5.0)
Initialisation
restart;
with(LinearAlgebra):
with(DiscreteTransforms):
with(combinat, fibonacci):
with(Statistics):
Seed=randomize():
> currentdir("D:\\Mathematik\\Ncrunch5");
Misc. operations
IO Test & descriptive statistics
LimitsArea:={1,261,522,784,1045,1305,1565,1827,2088,2349,2610,2871,3131,3158}:
t:=time():
datmat:=convert(readdata("Currency2.txt",[integer$4,float$38]),Matrix):
for k from 1 by 1 to 2000 do
  for j from 1 by 1 to 13 do
    YearData:=datmat[LimitsArea[j]..LimitsArea[j+1]-1,5..38]:
    amountrows:=LimitsArea[j+1]-LimitsArea[j]:
    for i from 1 by 1 to 34 do
      YD:=YearData[1..amountrows,i]:
      Mi-
nYear,MaxYear,MeanYear:=rtable_scanblock(YD,[rtable_dims(YD)],'Minimum','Maximum','Average');
      GainLossYear:=100-(YearData[1,i]+0.00000001)/(YearData[amountrows,i]+0.00000001)*100:
    end do:
  end do:
end do:
time()-t;
Loop 15000x15000
a:=1:
t:=time():
for x from 1 by 1 to 15000 do
  for y from 1 by 1 to 15000 do
    a:=a+x+y:
  end do:
end do:
time()-t;
2000x2000 normal distributed random matrix^1000
TotalTime:=0:
for i from 1 by 1 to 200 do
  a:=RandomMatrix(2000,outputoptions=[datatype=float[8]],generator=1.0..1.01):
  t:=time():
  (Array(a)^1000):
  TotalTime:=TotalTime+time()-t:
end do:
print(TotalTime);
1000000 values sorted ascending
TotalTime:=0:
for i from 1 by 1 to 100 do
  L := [seq(rand(0..1),i=1..1000000)]:
  t:=time():
  sort(L):
  TotalTime:=TotalTime+time()-t:
end do:
print(TotalTime);
Analysis
FFT over 1048576 values
m:=20:
TotalTime:=0:
for i from 1 by 1 to 100 do
  x:=RandomVector[row](2^m,generator=0.0..1.0,outputoptions=[datatype=float[8]]):
  t:=time():
  FourierTransform(x):
  TotalTime:=TotalTime+time()-t:
end do:
print(TotalTime);
Algebra
Determinant of a 1500x1500 random matrix
TotalTime:=0:
for i from 1 by 1 to 100 do
  a:=RandomMatrix(1500,outputoptions=[datatype=float[8]],generator=0.0..1.0):
  t:=time():
  Determinant(a):
  TotalTime:=TotalTime+time()-t:
end do:
print(TotalTime);
Inverse of a 1500x1500 uniform distr. random matrix
TotalTime:=0:
for i from 1 by 1 to 100 do
  a:=RandomMatrix(1500,outputoptions=[datatype=float[8]],generator=0.0..1.0):
  t:=time():
  MatrixInverse(a):
  TotalTime:=TotalTime+time()-t:
end do:
print(TotalTime);
Eigenvalues of a normal distr. 1200x1200 randommatrix
a:=RandomMatrix(1200,outputoptions=[datatype=float[8]],generator=0.0..1.0):
time(Eigenvalues(a));
Cholesky decomposition of a 1500x1500-matrix
TotalTime:=0:
for i from 1 by 1 to 100 do
  S:=RandomMatrix(1500,outputoptions=[datatype=float[8]],generator=0.0..1.0):
```

```
  A := Transpose(S).S:
  t:=time():
  LUDecomposition(A,method='Cholesky'):
  TotalTime:=TotalTime+time()-t:
end do:
print(TotalTime);
1500x1500 cross-product matrix
TotalTime:=0:
for i from 1 by 1 to 100 do
  S:=RandomMatrix(1500,outputoptions=[datatype=float[8]],generator=0.0..1.0):
  t:=time():
  Transpose(S).S:
  TotalTime:=TotalTime+time()-t:
end do:
print(TotalTime);
Number theory
Calculation of 10000000 fibonacci numbers
frnd:=rand(100..1000):
A := [seq(frnd(),i=1..10000000)]:
time(evalf(Map(a->fibonacci(a), A)));
Stochastic-statistic
Principal component analysis over a 10000x1000 matrix
> t := time(); a := RandomMatrix(10000, 1000, generator = 0. .. 1.0, outputoptions = [datatype = float[8]]);

cdim := ColumnDimension(a);

rdim := RowDimension(a); numcomp := cdim;

data_centered := Matrix(rdim, cdim);

for i to cdim do mean[i] := Mean(Column(a, i)); for j to rdim do data_centered[j, i] := a[j, i]-mean[i] end
do end do; cov := CovarianceMatrix(a);

evals := map(Re, Eigenvectors(cov)[1]);

evecs := map(Re, Eigenvectors(cov)[2]); SortMatByRow := proc (Mat, r) options operator, arrow; Ma-
trix(sort([seq(Column(Mat, i), i = 1 .. LinearAlgebra[ColumnDimension](Mat))], proc (x, y) options oper-
ator, arrow; evalb(y[r] < x[r]) end proc)) end proc; eigenVecsSorted := SubMa-
trix(SortMatByRow(convert(linalg[stackmatrix](evecs, evals), Matrix), RowDimension(evecs)+1), 1 ..
RowDimension(evecs), 1 .. ColumnDimension(evecs)); fv := SubMatrix(eigenVecsSorted, 1 .. cdim, 1 ..
numcomp); prinCom := `.`(Transpose(fv), Transpose(data_centered)); time()-t;
Gamma function over a 1500x1500 matrix
TotalTime:=0:
for i from 1 by 1 to 100 do
  a:=RandomMatrix(1500,generator=0.01..1.0,outputoptions=[datatype=float[8]]):
  t:=time():
  Map(GAMMA,a):
  TotalTime:=TotalTime+time()-t:
end do:
print(TotalTime);
Gaussian error function over a 1500x1500 matrix
TotalTime:=0:
for i from 1 by 1 to 100 do
  a:=RandomMatrix(1500,density=0.5,generator=0.0..1.0,outputoptions=[datatype=float[8]]):
  t:=time():
  Map(erf,a):
  TotalTime:=TotalTime+time()-t:
end do:
print(TotalTime);
Linear regression over a 1000x1000 matrix
TotalTime:=0:
for i from 1 by 1 to 100 do
  A:=RandomMatrix(1000,density=0.5,generator=0.0..1.0,outputoptions=[datatype=float[8]]):
  B:=RandomVector(1000,density=0.5,generator=0.0..1.0,outputoptions=[datatype=float[8]]):
  t:=time():
  LeastSquares(A,B):
  TotalTime:=TotalTime+time()-t:
end do:
print(TotalTime);
```

Mathematica Benchmark program:

Comparison benchmarktest for number cruncher testreport
(Version 5)
created 30th of December 2007 by Stefan Steinhaus

Misc. operation

IO Test and descriptive statistics

```
fn = "D:\\mathematik\\ncrunch5\\currency2.txt";

MyFile[fn_] :=
 Developer`ToPackedArray@
  N@Import[fn, "Table", "HeaderLines" -> 1, "IgnoreEmptyLines" -> True]

LimitsArea = {1, 261, 522, 784, 1045, 1305, 1565, 1827, 2088, 2349, 2610,
   2871, 3131, 3159};

Timing[
      datamat = MyFile[fn];
      Do[
           Do[
                YearData =
    Take[datamat, {LimitsArea[[j]], LimitsArea[[j + 1]] - 1}, {1, 37}];
                YDcols = Transpose[YearData];
                MinYear = Map[Min, YDcols];
                MaxYear = Map[Max, YDcols];
                MeanYear = Mean[Transpose[YDcols]];
           GainLossYear =
    100 - (First[YearData] + 0.00000001)/(Last[YearData] + 0.00000001)*100,
           {j, 1, 13}];,
      {k, 1, 2000}]
      ]
```

Loop 15000x15000

```
a = 1;
Timing[
 Do[
   Do[a = a + i + j, {i, 1, 15000, 1}];,
   {j, 1, 15000, 1}];
 ]
```

2000x2000 random matrix ^ 1000

```
Totaltime = 0;
Do[
      a = RandomReal[1/100, {2000, 2000}] + 1;
      t = AbsoluteTime[];
      a^1000;
      Totaltime = Totaltime + AbsoluteTime[] - t;
  , {j, 1, 200, 1}];
Print[Totaltime]
```

1000000 values sorted in ascending order

```
Totaltime = 0;
Do[
      a = RandomReal[{0, 1}, 1000000];
      t = AbsoluteTime[];
      Sort[a];
      Totaltime = Totaltime + AbsoluteTime[] - t;,
  {j, 1, 100, 1}];
Print[Totaltime]
```

Analysis

FFT over 1048576 random values

```
Totaltime = 0;
Do[
      a = RandomReal[{0, 1}, 1048576];
      t = AbsoluteTime[];
      Fourier[a];
      Totaltime = Totaltime + AbsoluteTime[] - t;,
  {j, 1, 100, 1}];
Print[Totaltime]
```

Algebra

Determinant of a 1500x1500 random matrix

```
Totaltime = 0;
Do[
      a = RandomReal[{0, 1}, {1500, 1500}];
      t = AbsoluteTime[];
      Det[a];
      Totaltime = Totaltime + AbsoluteTime[] - t;,
  {j, 1, 100, 1}];
Print[Totaltime]
```

Inverse of a 1500x1500 random matrix

```
Totaltime = 0;
Do[
```

```
    a = RandomReal[{0, 1}, {1500, 1500}];
    t = AbsoluteTime[];
    Inverse[a];
    Totaltime = Totaltime + AbsoluteTime[] - t;,
  {j, 1, 100, 1}];
Print[Totaltime]
```

Eigenvalues of a 1200x1200 randommatrix

```
a = RandomReal[{0, 1}, {1200, 1200}];

Timing[Eigenvalues[a];]
```

Cholesky decomposition of a 1500x1500-matrix

```
Totaltime = 0;
Do[
    a = RandomReal[{0, 1}, {1500, 1500}];
    a = Transpose[a].a;
    t = AbsoluteTime[];
    CholeskyDecomposition[a];
    Totaltime = Totaltime + AbsoluteTime[] - t;
  , {j, 1, 100, 1}];
Print[Totaltime]
```

1500x1500 cross-product matrix

```
Totaltime = 0;
Do[
    a = RandomReal[{0, 1}, {1500, 1500}];
    t = AbsoluteTime[];
    Transpose[a].a;
    Totaltime = Totaltime + AbsoluteTime[] - t;
    , {j, 1, 100, 1}];
Print[Totaltime]
```

Number theory

Calculation of 10000000 fibonacci numbers

```
gr = N[GoldenRatio]; w5 = N[Sqrt[5]];
fiboN[n_] = (gr^n - (-gr)^-n)/w5;

SeedRandom[16]; a = RandomInteger[{100, 1000}, 10000000];

Timing[fiboN[a];]
```

Stochastic & statistic

Principal component analysis over a 10000x1000 matrix

```
<< "MultivariateStatistics`";

a = RandomReal[{0, 1}, {10000, 1000}];

Timing[PrincipalComponents[a];]
```

Gamma function over a 1500x1500 matrix

```
Totaltime = 0;
Do[
    a = RandomReal[{0, 1}, {1500, 1500}];
    t = AbsoluteTime[];
    Gamma[a];
    Totaltime = Totaltime + AbsoluteTime[] - t;
    , {j, 1, 100, 1}];
Print[Totaltime]
```

Gaussian error function over a 1500x1500 matrix

```
Totaltime = 0;
Do[
    a = RandomReal[{0, 1}, {1500, 1500}];
    t = AbsoluteTime[];
    Erf[a];
    Totaltime = Totaltime + AbsoluteTime[] - t;,
    {j, 1, 100, 1}];
Print[Totaltime]
```

Linear regression over a 1000x1000 matrix

```
Totaltime = 0;
Do[
    a = RandomReal[NormalDistribution[], {1000, 1000}];
    b = Range[1000];
    t = AbsoluteTime[];
    LinearSolve[a, b];
    Totaltime = Totaltime + AbsoluteTime[] - t;,
    {j, 1, 100, 1}];
Print[Totaltime]
```

Matlab Benchmark program:

```
%**************************************************************************
%* Matlab Benchmark program version 5.0                                   *
%* Author : Stefan Steinhaus                                              *
%* EMAIL  : stefan@steinhaus-net.de                                       *
%* This program is public domain. Feel free to copy it freely.            *
%**************************************************************************

clc
disp('The following benchmark program will print the average timings')
disp('to calculate the functions by 3 runs.')
disp(' ')
disp(' ')
disp('!!! MATLAB - Benchmarkprogram !!!')
disp('==================================')
disp(' ')

%* Misc. operation *

result=0; runs=3;
for i=1:runs;
  tic;
  datmat=importdata('d:\\mathematik\\ncrunch5\\currency2.txt',' ');
   Limits_Area=[1,261,522,784,1045,1305,1565,1827,2088,2349,2610,2871,3131,3158];
  for k=1:2000;
    for j=1:13;
      Year_Data=datmat.data(Limits_Area(j):Limits_Area(j+1)-1,4:37);
      Min_Year=min(Year_Data);
      Max_Year=max(Year_Data);
      Mean_Year=mean(Year_Data);
      GainLoss_Year=100-(Year_Data(1,:)+0.00000001)./(Year_Data(end,:)+0.00000001)*100;
    end;
  end;
  result=result+toc;
end;
result=result/runs;
disp(['IO test & descriptive statistics_____________________ : ' num2str(result) ' sec.'])

result=0; a=1;
for i=1:runs
  tic;
  for x=1:15000;
     for y=1:15000;
        a=a+x+y;
     end;
  end;
  result=result+toc;
end;
result=result/runs;
disp(['Loop testing_____________________________________________ : ' num2str(result) ' sec.'])
result=0;
for i=1:runs
  for j=1:200
     b=1+(rand(2000,2000)/100);
     tic;
     a=b.^1000;
     result=result+toc;
  end
end
result=result/runs;
disp(['2000x2000 normal distributed random matrix^1000_____ : ' num2str(result) ' sec.'])
clear a; clear b; result=0;
for i=1:runs
  for j=1:100
     a=rand(1000000,1);
     tic;
     b=sort(a);
     result=result+toc;
  end
end
result=result/runs;
disp(['1000000 values sorted ascending_____________________ : ' num2str(result) ' sec.'])

%* Analysis *

clear a; clear b; result=0;
for i=1:runs
  for j=1:100
     a=rand(1048576,1);
     tic;
     b=fft(a);
     result=result+toc;
  end
end
result=result/runs;
disp(['FFT over 1048576 values (2^20)_____________________ : ' num2str(result) ' sec.'])

%* Algebra *

clear a; clear b; result=0;
for i=1:runs
  for j=1:100
     a=rand(1500,1500);
     tic;
     b=det(a);
```

```
      result=result+toc;
  end
end
result=result/runs;
disp(['Determinant of a 1500x1500 random matrix_____ : ' num2str(result) ' sec.'])
clear a; clear b; result=0;
for i=1:runs
  for j=1:100
      a=rand(1500,1500);
      tic;
      b=inv(a);
      result=result+toc;
  end
end
result=result/runs;
disp(['Inverse of a 1500x1500 uniform distr. random matrix__ : ' num2str(result) ' sec.'])
clear a; clear b; result=0;
for i=1:runs
  a=rand(1200,1200);
  tic;
  c=eig(a);
  result=result+toc;
end
result=result/runs;
disp(['Eigenval. of a normal distr. 1200x1200 randommatrix___ : ' num2str(result) ' sec.'])
clear a; clear b; result=0;
for i=1:runs
  for j=1:100
    a=rand(1500,1500);
    a=a'*a;
    tic;
    b=chol(a);
    result=result+toc;
  end
end
result=result/runs;
disp(['Cholesky decomposition of a 1500x1500-matrix_____ : ' num2str(result) ' sec.'])
clear a; clear b; result=0;
for i=1:runs
  for j=1:100
    a=rand(1500,1500);
    tic;
    b=a'*a;
    result=result+toc;
  end
end
result=result/runs;
disp(['1500x1500 cross-product matrix_____ : ' num2str(result) ' sec.'])
```

```
%* Number theory *

clear a; clear b;
phi = 1.6180339887498949; result=0;
for i=1:runs;
  a=floor(1000*rand(10000000,1));
  tic;
  b=(phi.^a-(-phi).^(-a))/sqrt(5);
  result=result+toc;
end
result=result/runs;
disp(['Calculation of 10000000 fibonacci numbers_____ : ' num2str(result) ' sec.'])

%* Stochastic-statistic *

result=0; a=0; b=0;
for i=1:runs;
   a=rand(10000,1000);
   tic;
      b=princomp(a);
   result=result+toc;
end;
result=result/runs;
disp(['Calc. of the principal components of a 10000x1000 matrix : ' num2str(result) ' sec.'])
clear a; clear b; result=0;
for i=1:runs
  for j=1:100
    a=rand(1500,1500);
    tic;
    b=gamma(a);
    result=result+toc;
  end
end
result=result/runs;
disp(['Gamma function over a 1500x1500 matrix_____ : ' num2str(result) ' sec.'])
clear a; clear b; result=0;
for i=1:runs
  for j=1:100
    a=rand(1500,1500);
    tic;
    b=erf(a);
    result=result+toc;
  end
end
result=result/runs;
disp(['Gaussian error function over a 1500x1500 matrix_____ : ' num2str(result) ' sec.'])
clear a; clear b; result=0;
for i=1:runs
  for j=1:100
```

```
      a=randn(1000,1000);
      b=1:1000;
      tic;
      b=a\b';
      result=result+toc;
   end
end
result=result/runs;
disp(['Linear regression over a 1000x1000 matrix_____ : ' num2str(result) ' sec.'])
```

O-Matrix Benchmark program:

```
# O-Matrix benchmark test version 5.0
# Stefan Steinhaus stefan@steinhaus-net.de
#

clear

include .\stats\vcov.oms

function [r,v,e,w] = pca(z,k,tp) begin
      [m,C0,R0,Cn,Rn,G0,Q0] = vcov(z,k)
      if tp == 0 then C = C0
      else if tp == 1 then C = R0
      else if tp == 2 then C = G0
      else if tp == 3 then C = Q0
      else begin
            error("[pca] error: tp flag should be between 0 and 3")
            halt
      end
      e = novalue
      r = eigsym(C,e)
      r = reverse(r)
      r = abs(r)
      d = prod(r)
      v = cumsum(r)/colsum(r)
      e = reverse(e')'
      w = z*e
end

# By default O-Matrix multi-threads the various windows within the
# application.  This allows editing, graphics etc. to continue
# while a calculation is running.  This reduces performance
# about 10-15% so we are turning it off here
interrupt(0)

begin
      runs=3
      result=0
      print
      print
      print "              !!! O-Matrix - Benchmark program !!!"
      print "              ==================================="
      print
      print
      print "Function / timing                                  result (sec.)"
      print "-------------------------------------------------------------------------------"

      # Misc. operation

      for counter=1 to runs begin
        t0 = time
        file="D:\\Mathematik\\Ncrunch5\\Currency2.txt"
        header=read(file, "char",1)
        datmat=read(file, "double",3159,38)
        close(file)
        Limits_Area={1,261,522,784,1045,1305,1565,1827,2088,2349,2610,2870,3131}
        Diff_Area={260,261,262,261,260,260,262,261,261,261,260,260,27}
        for i=1 to 2000 begin
          for j=1 to 13 begin
            Year_Data=datmat.blk(Limits_Area(j),5,Diff_Area(j),34)
            for k=1 to 34 begin
              Min_Year=mins(Year_Data.col(k))
              Max_Year=max(Year_Data.col(k))
              Mean_Year=colmean(Year_Data.col(k))
              GainLoss_Year=100-
(Year_Data(1,k)+0.00000001)/(Year_Data(Diff_Area(j),k)+0.00000001)*100
            end
          end
        end
        t1 = time - t0
        result=result+t1
      end
      result=result/runs
      print "IO test & descriptive statistics_____________________: ",result

      result=0
      N = 15000
      for counter=1 to runs begin
        a=1
        t0 = time
        for i = 1 to N begin
          for j = 1 to N begin
            a=a+i+j
          end
        end
        t1 = time - t0
        result=result+t1
      end
      result=result/runs
      print "Loop test___________________________________________: ",result

      N = 2000
      result=0
      for counter=1 to runs begin
       for counter2=1 to 200 begin
```

```
   x=unirnd(double(1.0),double(1.01),N,N)
    t0 = time
    x = x^1000.
    result = result + time-t0
 end
end
result=result/runs
print "2000x2000 random matrix^1000_____________________: ",result

N = 1000000
result=0
for counter=1 to runs begin
 for counter2=1 to 100 begin
    y=unirnd(double(0.0),double(1.0),N,1)
    t0 = time
    X = sort(y)
    result = result + time-t0
 end
end
result=result/runs
print "1000000 values sorted ascending__________________: ", result

# Analysis

N = 1048576
result=0
for counter=1 to runs begin
 for counter2=1 to 100 begin
   y=complex(unirnd(double(0.0),double(1.0),N,1))
    t0 = time
    X = fft(y)
    result = result + time-t0
 end
end
result=result/runs
print "FFT over 1048576 (2^20) values___________________: ", result

# Algebra

N = 1500
result=0
for counter=1 to runs begin
 for counter2=1 to 100 begin
    x=unirnd(double(0.0),double(1.0),N,N)
    t0 = time
    y=det(x)
    result = result + time-t0
 end
end
result=result/runs
print "Determinant of a 1500x1500 random matrix_____________: ",result

N = 1500
result=0
for counter=1 to runs begin
 for counter2=1 to 100 begin
    y=unirnd(double(0.0),double(1.0),N,N)
    t0 = time
    X = inv(y)
    result = result + time-t0
 end
end
result=result/runs
print "Inverse of 1500x1500 random matrix_____________________: ", result

N = 1200
result=0
for counter=1 to runs begin
  y=unirnd(double(0.0),double(1.0),N,N)
  t0 = time
  X = eigen(y)
  t1 = time - t0
  result=result+t1
end
result=result/runs
print "Eigenvalues of a 1200x1200 random matrix_____________: ",result

N = 1500
result=0
for counter=1 to runs begin
 for counter2=1 to 100 begin
    y=unirnd(double(0.0),double(1.0),N,N)
    y = y'*y
   t0 = time
    X = cholesky(y)
    result = result + time-t0
  end
end
result=result/runs
print "Cholesky decompositon of 1500x1500 matrix___________: ",result

N = 1500
result=0
for counter=1 to runs begin
  for counter2=1 to 100 begin
    y=unirnd(double(0.0),double(1.0),N,N)
   t0 = time
    X = y'*y
```

```
    result = result + time-t0
  end
end
result=result/runs
print "1500x1500 cross-product matrix_____________________: ",result

# Number theory

N = 10000000
result=0
phi = 1.6180339887498949
for counter=1 to runs begin
  a=unirnd(double(100.0),double(1000.0),N,1)
  t0=time
  b=(phi^a-(-phi)^(-a))/sqrt(5.)
  result = result + time-t0
end
result=result/runs
print "Calculation of 10000000 fibonacci numbers__________: ",result

# Stochastic-statistic

N = 10000
result=0
for counter=1 to runs begin
      a=unirnd(double(0.0),double(1.0),N,N/10)
      t0=time
      [r,v,e,w] = pca(a,0,0)
      result = result + time-t0
end
result=result/runs
print "Principal component analysis over 10000x1000 matrix: ",result

# O-Matrix implements ln(gamma(x)) instead of gamma(x),
# since the latter will overflow many computers' floating-point
# representation at modest values of x.
N = 1500
result=0
for counter=1 to runs begin
  for counter2=1 to 100 begin
    y=unirnd(double(0.0),double(1.0),N,N)
   t0=time
    X = exp(gammln(y))
    result = result + time-t0
  end
end
result=result/runs
print "Gamma function over a 1500x1500 matrix_____________: ",result

      N = 1500
      result=0
      for counter=1 to runs begin
        for counter2=1 to 100 begin
          y=unirnd(double(0.0),double(1.0),N,N)
         t0=time
          X = erf(y)
          result = result + time-t0
        end
      end
      result=result/runs
      print "Gaussian error function over a 1500x1500 matrix____: ",result

      N = 1000
      result=0
      for counter=1 to runs begin
        for counter2=1 to 100 begin
          a = unirnd(double(0.0),double(1.0),N,N)
          b = 1::N
         t0=time
          X = a\b;
          result = result + time-t0
        end
      end
      result=result/runs
      print "Linear regression over 1000x1000 matrix_____________: ",result
end
```

Ox Professional Benchmark program:

```
/*--------------------------------------------------------------------------*/
/* Ox Benchmark program version 5.0 from Stefan Steinhaus                   */
/* EMAIL  : stefan@steinhaus-net.de                                         */
/*                                                                          */
/* This program is public domain. Feel free to copy it freely.              */
/*                                                                          */
/*--------------------------------------------------------------------------*/


#include <oxstd.h>

main()
{
    decl runs = 3;                          /* number of repeats */
    decl i, k,j,time, a, b, c, x,y,secs,phi,datmat,header;
    decl LimitsArea, YearData, MinYear, MaxYear, MeanYear, GainLossYear;
    print("\n\n\n");
    print("!!! Ox - Benchmark program !!!\n");
    print("==============================\n\n\n");
    print("Displaying the avg. timings over ",runs," runs.\n\n");
    print("Function / timing                                    result (sec.)\n");
    print("-------------------------------------------------------------------------------\n");
    format("%8.3f");

    /* IO operation */
    /* Test of Extraction of submatrices and calculation of descriptive stats */
    /* Doing the whole process 2000 times do get higher timings             */

    LimitsArea={0,260,521,783,1044,1304,1564,1826,2087,2348,2609,2870,3130,3157};
    for (i = secs = 0; i < runs; ++i)
    {
        time = timer();
        decl file = fopen("Currency2.txt", "rb");
        fscan(file, "%z", &header);
        fscan(file, "%#m", 3159, 38, &datmat);
        fclose(file);
        for (k=0; k<2000; ++k)
        {
            for (j=0;j<13; ++j)
            {
                YearData=datmat[LimitsArea[j]:LimitsArea[j+1]-1][4:37];
                MinYear=minc(YearData);
                MaxYear=maxc(YearData);
                MeanYear=meanc(YearData);
                GainLossYear=100-(YearData[0][]+0.00000001)./(YearData[rows(YearData)-
1][]+0.00000001)*100;
            }
        }
        secs += (timer() - time) / 100;
    }
    secs /= runs;
    print("IO test & descriptive statistics_____________________________: ", secs, "\n");

    /* Misc. operation */

    x=0;y=0;a=1;
    for (i = secs = 0; i < runs; ++i)
    {
        time = timer();
        for (x = 0; x < 15000; ++x)
        {
            for (y = 0; y < 15000; ++y)
            {
                a = a+x+y;
            }
        }
        secs += (timer() - time) / 100;
    }
    secs /= runs;  delete a;  delete b;
    print("Loop test_____________________________________________________: ", secs, "\n");

    for (i = secs = 0; i < runs; ++i)
    {
        for (j = 0; j < 200; ++j)
        {
            b=1+fabs(ranu(2000,2000)/100);
            time = timer();
            a = b .^ 1000;
            secs += (timer() - time) / 100;
        }
    }
    secs /= runs;  delete a;  delete b;
    print("2000x2000 random matrix .^ 1000_____________________________: ", secs, "\n");

    for (i = secs = 0; i < runs; ++i)
    {
        for (j = 0; j < 100; ++j)
        {
            a = ranu(1,1000000);
            time = timer();
            b = sortr(a);
            secs += (timer() - time) / 100;
        }
```

```
}
secs /= runs;  delete a;  delete b;
print("Sorting of 1,000,000 random values_____: ", secs, "\n");

/* Analysis */

for (i = secs = 0; i < runs; ++i)
{
     for (j = 0; j < 100; ++j)
     {
          a = ranu(1, 1048576);
          time = timer();
          b = fft(a);
          secs += (timer() - time) / 100;
     }
}
secs /= runs;  delete a;  delete b;
print("FFT over 1,048,576 random values_____: ", secs, "\n");

/* Algebra */

for (i = secs = 0; i < runs; i++)
{
     for (j = 0; j < 100; j++)
     {
          a = ranu(1500,1500)/10;
          time = timer();
          b = determinant(a);
          secs += (timer() - time) / 100;
     }
}
secs /= runs;  delete a;  delete b;
print("Determinant of a 1500x1500 random matrix_____: ", secs, "\n");

for (i = secs = 0; i < runs; ++i)
{
     for (j = 0; j < 100; ++j)
     {
          a = ranu(1500,1500);
          time = timer();
          b = invert(a);
          secs += (timer() - time) / 100;
     }
}
secs /= runs;  delete a;  delete b;
print("Inverse of a 1500x1500 random matrix_____: ", secs, "\n");

for (i = secs = 0; i < runs; ++i)
{
     a = ranu(1200,1200);
     time = timer();
     eigen(a, &b);
     secs += (timer() - time) / 100;
}
secs /= runs;  delete a;  delete b;
print("Eigenvalues of a 1200x1200 random matrix_____: ", secs, "\n");

for (i = secs = 0; i < runs; ++i)
{

     for (j = 0; j < 100; ++j)
     {
          a = ranu(1500,1500); a = a'a;
          time = timer();
          b = choleski(a);
          secs += (timer() - time) / 100;
     }
}
secs /= runs;  delete a;  delete b;
print("Choleski decomposition of a 1500x1500 random matrix_____: ", secs, "\n");

for (i = secs = 0; i < runs; ++i)
{
     for (j = 0; j < 100; ++j)
     {
          a = ranu(1500,1500);
          time = timer();
          b = a'a;
          secs += (timer() - time) / 100;
     }
}
secs /= runs;  delete a;  delete b;
print("Creation of 1500x1500 cross-product matrix_____: ", secs, "\n");

/* Number theory */

phi = 1.6180339887498949;
for (i = secs = 0; i < runs; i++)
{
     a = floor(1000*ranu(10000000,1));
     time = timer();
     b = (phi^a-(-phi)^(-a))/sqrt(5);
     secs += (timer() - time) / 100;
}
secs /= runs;  delete a;  delete b;
print("Calculation of 10000000 fibonacci numbers_____: ",secs, "\n");
```

```
    /* Stochastic-statistic */

    // n! = Gamma(n + 1)
    for (i = secs = 0; i < runs; ++i)
    {
        for (j = 0; j < 100; ++j)
        {
            b=ranu(1500,1500);
            time = timer();
            a = gammafact(b);
            secs += (timer() - time) / 100;
        }
    }
    secs /= runs;  delete a;  delete b;
    print("Gamma function on a 1500x1500 random matrix_____________: ", secs, "\n");

    for (i = secs = 0; i < runs; ++i)
    {
        for (j = 0; j < 100; ++j)
        {
            a = ranu(1500,1500);
            time = timer();
            b = erf(a);
            secs += (timer() - time) / 100;
        }
    }
    secs /= runs;  delete a;  delete b;
    print("Gaussian error function over a 1500x1500 random matrix_____: ", secs, "\n");

    for (i = secs = 0; i < runs; ++i)
    {
        for (j = 0; j < 100; ++j)
        {
            a = ranu(1000,1000); b = range(1,1000);  c = 0;
            time = timer();
            ols2r(b, a, &c);
            secs += (timer() - time) / 100;
        }
    }
    secs /= runs;  delete a;  delete b;
    print("Linear regression over a 1000x1000 random matrix_____: ", secs, "\n");
}
```

SciLab Benchmark program:

```
stacksize(80000000)

cd "C:\Program Files\scilab-4.1.2\intfftw";
exec "loader.sce";

disp('The following benchmark program will print the average timings');
disp('to calculate the functions by 3 runs.');
disp(' ');
disp(' ');
disp('!!! SCILAB - Benchmarkprogram !!!');
disp('==================================');
disp(' ');

//* Test of IO function and descriptive statistics *

result = 0;
a = 0;
b = 0;
runs = 3;
for i=1:runs
   tic();
   [fd,err]=mopen('D:\Mathematik\Ncrunch5\currency2.txt','r',1);
   header=mgetstr(154,fd);
   datmat=fscanfMat('D:\Mathematik\Ncrunch5\currency2.txt');
   LimitsArea={1,261,522,784,1045,1305,1565,1827,2088,2349,2610,2871,3131,3158};
  for k=1:2000
    for j=1:13
      Year_Data=datmat(LimitsArea(j):LimitsArea(j+1)-1,5:38);
      Min_Year=min(Year_Data,1);
      Max_Year=max(Year_Data,1);
      Mean_Year=mean(Year_Data,1);
      GainLoss_Year=100-(Year_Data(1,:)+0.00000001)./(Year_Data($,:)+0.00000001)*100;
    end
  end
   zeit=toc();
   result=result+zeit;
end
result=result/runs;
disp('IO test and descriptive statistics_____________________: '+string(result)+' sec.');


//* Misc. operation *

result = 0;
a=1;
for i = 1:runs
  tic();
  for x = 1:15000
    for y = 1:15000
      a=a+x+y;
    end
  end
  zeit = toc();
  result = result+zeit;
end
result = result/runs;
disp('Loop test_____________________________________________: '+string(result)+' sec.');
clear('a');
clear('b');
for i = 1:runs
  for j=1:200
    b = 1+(rand(2000,2000,'uniform')/100);
    tic();
    a = b.^1000;
    result = result+toc();
  end
end
result = result/runs;
disp('2000x2000 normal distributed random matrix^1000_____: '+string(result)+' sec.');
clear('a');
clear('b');
result = 0;
for i = 1:runs
  for j=1:100
    a = rand(1000000,1,'uniform');
    tic();
    b = sort(a);
    result = result+toc();
  end
end
result = result/runs;
disp('1000000 values sorted ascending_____________________: '+string(result)+' sec.');

//* Analysis *

clear('a');
clear('b');
result = 0;
for i = 1:runs
  for j=1:100
    a = rand(1048576,1,'uniform');
    tic();
    b = fftw(a);
    result = result+toc();
```

```
  end
end
result = result/runs;
disp('FFT over 1048576 values_____: '+string(result)+' sec.');

//* Algebra *

clear('a');
clear('b');
result = 0;
for i = 1:runs
  for j=1:100
    a = rand(1500,1500,'uniform');
    tic();
    b = det(a);
    result = result+toc();
  end
end
result = result/runs;
disp('Determinant of a 1500x1500 random matrix_____: '+string(result)+' sec.');
clear('a');
clear('b');
result = 0;
for i = 1:runs
  for j=1:100
    a = rand(1500,1500,'uniform');
    tic();
    b = inv(a);
    result = result+toc();
  end
end
result = result/runs;
disp('Inverse of a 1500x1500 uniform distr. random matrix__: '+string(result)+' sec.');
clear('a');
clear('b');
result = 0;
for i = 1:runs
  a = rand(1200,1200,'uniform');
  tic();
  c = spec(a);
  result = result+toc();
end
result = result/runs;
disp('Eigenval. of a normal distr. 1200x1200 randommatrix__: '+string(result)+' sec.');
clear('a');
clear('b');
result = 0;
for i = 1:runs
  for j=1:100
    a = rand(1500,1500,'uniform');
    a = a'*a;
    tic();
    b = chol(a);
    result = result+toc();
  end
end
result = result/runs;
disp('Cholesky decomposition of a 1500x1500-matrix_____: '+string(result)+' sec.');
clear('a');
clear('b');
result = 0;
for i = 1:runs
  for j=1:100
    a = rand(1500,1500,'uniform');
    tic();
    b = a'*a;
    result = result+toc();
  end
end
result = result/runs;
disp('1500x1500 cross-product matrix_____: '+string(result)+' sec.');

//* Number theory *

clear('a');
clear('b');
phi = 1.61803398874989;
result = 0;
for i = 1:runs
  a = floor(1000*rand(10000000,1,'uniform'));
  tic();
  b = (phi.^a-(-phi).^(-a))/sqrt(5);
  result = result+toc();
end
result = result/runs;
disp('Calculation of 10000000 fibonacci numbers_____: '+string(result)+' sec.');

clear('a');
clear('b');
// * pca without output */
function [lambda,facpr,comprinc]=pca2(varargin)

//This  function performs  several  computations known  as
//"principal component  analysis". It includes  drawing of
//"correlations circle",  i.e. in the  horizontal axis the
//correlation  values  r(c1;xj)  and  in  the  vertical
//r(c2;xj). It is an extension of the pca function.
//
```

```
//The  idea  behind this  method  is  to  represent in  an
//approximative  manner a  cluster of  n individuals  in a
//smaller  dimensional subspace.  In order  to do  that it
//projects the cluster onto a subspace.  The choice of the
//k-dimensional projection subspace is  made in such a way
//that  the distances  in  the projection  have a  minimal
//deformation: we are looking for a k-dimensional subspace
//such that the squares of the distances in the projection
//is  as  big  as  possible  (in  fact  in  a  projection,
//distances can only stretch).  In other words, inertia of
//the projection  onto the k dimensional  subspace must be
//maximal.
//
//x is a nxp (n  individuals, p variables) real matrix.
//
//lambda is a px2 numerical matrix. In the first column we
//find the  eigenvalues of V,  where V is  the correlation
//pxp matrix  and in the  second column are the  ratios of
//the  corresponding  eigenvalue  over  the  sum  of
//eigenvalues.
//
//facpr  are  the principal  factors:  eigenvectors of  V.
//Each  column is an  eigenvector element  of the  dual of
//R^p. Is an orthogonal matrix.
//
//comprinc  are  the  principal components.  Each  column
//(c_i=Xu_i)  of  this  nxn  matrix  is  the  M-orthogonal
//projection of individuals onto principal axis.  Each one
//of this columns is a linear combination of the variables
//x1,  ...,xp  with  maximum  variance  under  condition
//u'_iM^(-1)u_i=1.
//
//Verification: comprinc*facpr=x
//
//References: Saporta, Gilbert, Probabilites,  Analyse des
//Donnees et Statistique, Editions Technip, Paris, 1990.
//
//author: carlos klimann
//
//date: 2002-02-05
//commentary fixed 2003-19-24 ??
// update disable graphics output Allan CORNET 2008

 [lhs,rhs]=argn(0)

 x = [];
 N = [];
 no_graphics = %f;

select rhs,
    case 1 then
      x = varargin(1);
      N=[1 2];
    case 2 then
      x = varargin(1);
      N = varargin(2);
    case 3 then
      x = varargin(1);
      N = varargin(2);
      no_graphics = varargin(3);
    else
    error(sprintf("%s: Wrong number of input argument(s).\n","pca"));
    end

if ( type(no_graphics) <> 4 ) then
  error(sprintf("%s: Wrong type for third input argument: Boolean expected.\n" ,"pca"));
end

if x==[] then
  lambda=%nan;
  facpr=%nan;
  comprinc=%nan;
  return;
end

[rowx colx]=size(x)
if size(N,'*')<>2 then error('Second parameter has bad dimensions'), end,
if max(N)>colx then disp('Graph demand out of bounds'), return, end

xbar=sum(x,'r')/rowx
y=x-ones(rowx,1)*xbar
std=(sum(y .^2,'r')) .^ .5
V=(y'*y) ./ (std'*std)
[lambda facpr]=bdiag(V,(1/%eps))
[lambda order]=sort(diag(lambda))
lambda(:,2)=lambda(:,1)/sum(lambda(:,1))
facpr=facpr(:,order)
comprinc=x*facpr

if ~no_graphics then
  w = winsid();
  if (w == []) then
    w = 0;
  else
    w = max(w)+1;
  end
  fig1 = scf(w);
```

```
    rc = (ones(colx,1)* sqrt((lambda(N,1))')) .* facpr(:,order(N)) ;
    rango = rank(V);
    ra = [1:rango]';
    if ( rango <= 1 ) then return, end
    plot2d(-rc(ra,1),rc(ra,2),style=-10);
    legend('(r(c1,xj),r(c2,xj)');
    ax=gca();
    ax.x_location="middle";
    ax.y_location = "middle";
    blue=color('blue')
    for k=1:rango,
      xstring(rc(k,1),rc(k,2),'X'+string(k));
      e=gce();
      e.foreground=blue;
    end
    title(' -Correlations Circle- ');
    fig2 = scf(w+1);
    plot2d3([0;ra;rango+1]',[0; lambda(ra,2);0]);
    plot2d(ra,lambda(ra,2),style=9);
    ax=gca();
    ax.grid=[31 31];
    plot2d3([0;ra;rango+1]',[0; lambda(ra,2);0])
    plot2d(ra,lambda(ra,2),style=9)
    for k=1:rango,
      xstring(k,0,'l'+string(k)),
      e=gce();e.font_style=1
    end
    title(' -Eigenvalues (in percent)- ')
    ylabel('%')
  end
endfunction
// * END pca without output */


result=0;

for i=1:runs;
    a=rand(10000,1000,'uniform');
    tic();
    b=pca2(a,[1 2],%t);
    result=result+toc();
end;


result=result/runs;
disp(['Calc. of the principal components of a 10000x1000 matrix : '+string(result)+' sec.'])
clear pca_without_graphics;

//* Stochastic-statistic *
clear('a');
clear('b');
result = 0;
for i = 1:runs
  for j=1:100
    a = rand(1500,1500,'uniform');
    tic();
    b = gamma(a(:));
    result = result+toc();
  end
end
result = result/runs;
disp('Gamma function over a 1500x1500 matrix_______________: '+string(result)+' sec.');
clear('a');
clear('b');
result = 0;
for i = 1:runs
  for j=1:100
    a = rand(1500,1500,'uniform');
    tic();
    b = erf(a(:));
    result = result+toc();
  end
end
result = result/runs;
disp('Gaussian error function over a 1500x1500 matrix______: '+string(result)+' sec.');
clear('a');
clear('b');
result = 0;
for i = 1:runs
  for j=1:100
    a = rand(1000,1000,'uniform');
    b = 1:1000;
    tic();
    b = a\b';
    result = result+toc();
  end
end
result = result/runs;
disp('Linear regression over a 1000x1000 matrix_____________: '+string(result)+' sec.');
```

Appendix B: References

Thanks to the help of the following persons it was possible to correct and improve the benchmark test and to correct the information within the report:

GAUSS: Nathan Lohonen, Aptech Systems Inc. (USA)
GAUSS: Sam Jones, Aptech Systems Inc. (USA)
GAUSS: Ron Schoenberg, Aptech Systems Inc. (USA)
Maple: Thomas Richard, Scientific Computers GmbH (Germany)
Mathematica: Marcus Wynne, Wolfram Research Inc. (USA)
Mathematica: Darren Glosemeyer, Wolfram Research Inc. (USA)
Mathematica: Daniel Lichtblau, Wolfram Research Inc. (USA)
Mathematica: Roger Germundsson, Wolfram Research Inc. (USA)
Mathematica: Jon McLoone, Wolfram Research Inc. (England)
Matlab: Cleve Moler, The Mathworks Inc. (USA)
Matlab: Bill McKeeman, The Mathworks Inc. (USA)
Matlab: Bobby Cheng, The Mathworks Inc. (USA)
O-Matrix: Beau Paisley, Harmonic Software Inc. (USA)
Ox Professional: Dr. Jurgen Doornik, University of Oxford (England)
Scilab: Allan Cornet, INRIA (France)