Comparison of mathematical programs for data analysis

(Edition 4)

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

München / Germany
27 Juli 2002

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

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. Therefor 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. Those 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, available functions by additional modules 0.9 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:

- Mathematical functions 38%
- Graphical functions 10%
- Programming environment 9%
- Data import/export 5%
- Available operating systems 2%
- Speed comparison 36%

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 are 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)
- ***MuPAD*** from the SciFace Software GmbH & Co. KG
  (www.sciface.com)
- ***O-Matrix*** from Harmonic Software
  (www.omatrix.com)
- ***Ox*** from Jurgen Doornik
  (www.nuff.ox.ac.uk/Users/Doornik/ & www.oxmetrics.net)
- ***Scilab*** from Dr. Scilab
  (www.scilab.org)
- ***S-Plus*** from Insightful Inc.
  (www.insightful.com)

The programs Macsyma and R-Lab have not been tested in this edition of the test report anymore. The development of both programs have been stopped. Even so Macsyma is still available over some dealers, there is no intention for any further development. Both products are obviously dead and are therefor of no interest for any further reporting.

2. 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

2.1. 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. Therefor the following results are not very surprising.

Functions (Version)	**GAUSS**	**Maple**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**Scilab**	**S-Plus**
	(5.0)	**(V8.0)**	**(4.2)**	**(6.5)**	**(2.5)**	**(5.2)**	**(3.2)**	**(2.6)**	**(V6.1)**
BesselI	+	+	+	+	+	-	+	+	-
Bessel J	+	+	+	+	+	+	+	+	-
BesselK	-	+	+	+	+	-	+	+	-
Bessel Y	+	+	+	+	+	+	+	+	-
Beta function	+	+	+	+	+	+	+	-	-
Binomial	-	+	+	+	+	-	+	-	+
Factorial	+	+	+	+	+	+	+	-	+
Gamma function	+	+	+	+	+	+	+	+	+
Hyperbolic trig. function	+	+	+	+	+	+	+	+	+

Functions (Version)	GAUSS (5.0)	Maple (V8.0)	Mathematica (4.2)	Matlab (6.5)	MuPAD (2.5)	O-Matrix (5.2)	Ox (3.2)	Scilab (2.6)	S-Plus (V6.1)
Incomplete Gammafunc.	+	+	+	+	+	+	+	-	-
Log / Ln / Exp	+ / + / +	+ / + / +	+ / + / +	+ / + / +	+ / + / +	+ / + / +	+ / + / +	+ / + / +	+ / + / +
Log-Gammafunc.	+	+	+	+	+	+	+	+	+
Poly gamma	-	+	+	+	+	-	+	-	-
Square root	+	+	+	+	+	+	+	+	+
Sum / Product	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +
Trig. / arg trig. functions	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +
Implemented functions	85.00% (17/20)	100.00% (20/20)	100.00% (20/20)	100.00% (20/20)	100.00% (20/20)	80.0% (16/20)	100.00% (20/20)	80.00% (16/20)	65.00% (13/20)

2.2. 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, equations solvings and other routines from algebra.

Functions (Version)	GAUSS (5.0)	Maple (V8.0)	Mathematica (4.2)	Matlab (6.5)	MuPAD (2.5)	O-Matrix (5.2)	Ox (3.2)	Scilab (2.6)	S-Plus (V6.1)
Eigenvalues									
Eigenvalues	+	+	+	+	+	+	+	+	+
Eigenvectors	+	+	+	+	+	+	+	+	+
Matrix analysis									
Cross product matrix	+	+	+	-	+	-	+	-	+
Characteristic polynom	+	+	+	+	+	-	+	+	-
Determinant	+	+	+	+	+	+	+	+	+
Hadamard matrix	-	-	$	+	-	-	-	-	-
Hankel matrix	-	+	+	+	$	+	-	+	-
Hilbert matrix	-	+	+	+	+	+	-	-	-

Functions (Version)	**GAUSS**	**Maple**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**Scilab**	**S-Plus**
	(5.0)	**(V8.0)**	**(4.2)**	**(6.5)**	**(2.5)**	**(5.2)**	**(3.2)**	**(2.6)**	**(V6.1)**
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	+	+	+	+	+	+	+	+	+
Optimization									
Optimization - linear models (Unconstr. / Constr.)	+ / $	+ / +	+ / +	+ / +	+ / +	+ / +	+ / -	+ / +	+ / +
Optimization - nonlinear models (Unconstr. / Constr.)	+ / $	+ / -	+ / +	$ / $	$ / $	+ / +	+ / -	+ / +	$ / $
Optimization - quadratic models (QP) (Unconstr. / Constr.)	+ / +	+ / -	$ / $	$ / $	$ / $	+ / +	+ / +	+ / +	$ / $
Equation solver									
Linear equation solver	+	+	+	+	+	+	+	+	+
Non-linear equation solver	$	+	+	$	+	+	+	+	+
Ordinary Differential Equation solver	$	+	+	+	+	+	-	+	+
Partial Differential Equation solver	-	+	+	+	+	-	-	-	-

Functions (Version)	GAUSS (5.0)	Maple (V8.0)	Mathematica (4.2)	Matlab (6.5)	MuPAD (2.5)	O-Matrix (5.2)	Ox (3.2)	Scilab (2.6)	S-Plus (V6.1)
Miscellaneous									
Moore-Penrose pseudo-inverse	+	+	+	+	+	+	+	+	+
Sparse matrices handling	+	+	+	+	+	-	-	+	-
Implemented functions	77.94% (26.5/34)	82.35% (28/34)	84.12% (28.6/34)	95.29% (32.4/34)	83.24% (28.3/34)	67.65% (23/34)	64.71% (22/34)	76.47% (26/34)	60.59% (20.6/34)

2.3. Analysis

Functions (Version)	GAUSS (5.0)	Maple (V8.0)	Mathematica (4.2)	Matlab (6.5)	MuPAD (2.5)	O-Matrix (5.2)	Ox (3.2)	Scilab (2.6)	S- Plus (V6.1)
Numerical integration (single / double / triple)	+ / + / +	+ / + / +	+ / + / +	+ / + / +	+ / + / +	+ / + / -	+ / + / -	+ / + / +	+ / - / -
Numerical differentiation (1st deriv. / 2nd deriv.)	+ / +	+ / +	+ / +	+ / +	+ / +	+ / -	+ / +	+ / +	+ / -
Fourier transf. (1D / 2D / multidim.)	+ / + / +	+ / - / -	+ / + / +	+ / + / +	+ / + / +	+ / + / -	+ / - / -	+ / + / +	+ / + / +
Inverse Fourier transformation (1D / 2D / multidim.)	+ / + / +	+ / - / -	+ / + / +	+ / + / +	+ / + / +	+ / + / -	+ / - / -	+ / + / +	+ / + / +
Implemented functions	100.00% (11/11)	63.64% (7/11)	100.00% (11/11)	100.00% (11/11)	100.00% (11/11)	63.64% (7/11)	54.55% (6/11)	100.00% (11/11)	72.73% (8/11)

2.4. Numerical mathematics

Function (Version)	GAUSS (5.0)	Maple (V8.0)	Mathematica (4.2)	Matlab (6.5)	MuPAD (2.5)	O-Matrix (5.2)	Ox (3.2)	Scilab (2.6)	S-Plus (V6.1)
Interpolation									
B-Spline Interpolation	-	+	+	$	-	-	+	-	+
Classical Interpolation	-	+	+	+	+	+	-	+	+
k-Spline Interpolation	+	+	+	$	+	+	+	+	+
Pade Interpolation	$	+	+	-	+	+	-	-	-

Function (Version)	GAUSS (5.0)	Maple (V8.0)	Mathematica (4.2)	Matlab (6.5)	MuPAD (2.5)	O-Matrix (5.2)	Ox (3.2)	Scilab (2.6)	S-Plus (V6.1)
Other functions									
Bisection	$	-	+	+	+	-	+	-	-
Newton method for finding roots	+	+	+	$	+	+	+	+	-
Runge Kutta method for solving ODE	$	+	+	$	+	+	-	+	-
Implemented functions	67.14% (4.7/7)	85.71% (6/7)	100.00% (7/7)	80.00% (5.6/7)	85.71% (6/7)	71.43% (5/7)	57.14% (4/7)	57.14% (4/7)	42.86% (3/7)

2.5. Descriptive statistic, stochastic and distribution functions

Functions (Version)	GAUSS (5.0)	Maple (V8.0)	Mathematica (4.2)	Matlab (6.5)	MuPAD (2.5)	O-Matrix (5.2)	Ox (3.2)	Scilab (2.6)	S-Plus (V6.1)
General functions									
Contingency tables	$	-	-	-	-	-	-	-	+
Correlation	+	+	+	+	+	+	+	+	+
Cross tabulation	$	-	-	$	-	-	-	-	+
Deviation	+	+	+	+	+	+	+	+	-
Kurtosis	-	+	+	$	+	-	+	+	+
Markov models	$	-	$	-	$	-	+	+	-
Mean / Mean / Mode	+ / + / $	+ / + / +	+ / + / +	+ / + / -	+ / + / +	+ / + / -	+ / + / +	+ / + / +	+ / + / -
Min / Max	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +
Skewness	-	+	+	$	+	-	+	-	+
Variance	+	+	+	+	+	+	+	+	+
Variance-covariance matrix	+	+	+	+	$	+	+	+	+
Distribution functions *(PDF / CDF / iCDF/ randomnumber)*									
Bernoulli	-/-/-/-	-/-/-/-	+/+/+/+	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
Beta	$/+/$/+	+/+/+/+	+/+/+/+	$/$/$/$	+/+/+/+	-/-/-/-	+/+/+/+	-/+/-/+	+/+/-/+
Binomial	$/$/$/$	+/+/+/+	+/+/+/+	$/$/$/$	+/+/+/+	-/-/-/-	+/+/+/+	-/+/-/+	+/+/-/+

Functions (Version)	GAUSS (5.0)	Maple (V8.0)	Mathe-matica (4.2)	Matlab (6.5)	MuPAD (2.5)	O-Matrix (5.2)	Ox (3.2)	Scilab (2.6)	S-Plus (V6.1)
Brownian motion	-/-/-/-	-/-/-/-	-/-/-/$	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/+	-/-/-/-	-/-/-/-
Cauchy	$/$/$/$	+/+/+/+	+/+/+/+	-/-/-/-	+/+/+/+	-/-/-/-	+/+/+/+	-/-/-/-	+/+/-/+
Chi-squared	$/+/+/$	+/+/+/+	+/+/+/+	$/$/$/$	+/+/+/+	-/-/-/-	+/+/+/+	-/+/-/+	+/+/-/+
Chi-squared (non-central)	$/+/$/$	-/-/-/-	+/+/+/+	$/$/$/$	-/$/-/$	-/-/-/-	-/+/-/-	-/+/-/+	-/-/-/-
Dirichlet	-/-/-/-	-/-/-/-	$/$/$/$	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/+	-/-/-/-	-/-/-/-
Erlang	$/$/$/$	-/-/-/-	+/+/+/+	-/-/-/-	+/+/+/+	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
Exponential	$/$/$/$	+/+/+/+	+/+/+/+	$/$/$/$	+/+/+/+	-/-/-/-	+/+/+/+	-/-/-/+	+/+/-/+
Extreme value	-/-/-/-	-/-/-/-	+/+/+/+	-/-/-/-	-/-/-/-	-/-/-/-	+/+/+/+	-/-/-/-	-/-/-/-
F	+/+/+/$	+/+/+/+	+/+/+/+	$/$/$/$	+/+/+/+	+/-/-/-	+/+/+/+	-/+/-/+	+/+/-/+
F (non-central)	+/+/$/$	-/-/-/-	+/+/+/+	$/$/$/$	-/$/-/$	-/-/-/-	-/+/-/-	-/+/-/+	-/-/-/-
Gamma	$/+/$/+	+/+/+/+	+/+/+/+	$/$/$/$	+/+/+/+	-/-/-/-	+/+/+/+	-/+/-/+	+/+/-/+
Geometric	$/$/$/$	-/-/-/-	+/+/+/+	$/$/$/$	+/+/+/+	-/-/-/-	+/+/+/+	-/-/-/-	+/+/-/+
Gumbel	$/$/$/$	-/-/-/-	$/$/$/$	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
Half-normal	-/-/-/-	-/-/-/-	+/+/+/+	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
Hotelling T2	$/$/-/-	-/-/-/-	+/+/+/+	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
Hyper-exponential	$/$/$/$	-/-/-/-	$/$/$/$	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
Hypergeometric	$/$/$/$	+/+/+/+	+/+/+/+	$/$/$/$	+/+/+/+	-/-/-/-	+/+/+/+	-/-/-/-	+/+/-/+
Kernel	-/-/-/-	-/-/-/-	$/$/$/$	-/-/-/-	+/+/-/+	-/-/-/-	-/+/-/-	-/-/-/-	-/-/-/-
Laplace	$/$/$/$	+/+/+/+	+/+/+/+	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
Logarithmic	-/-/-/-	-/-/-/-	$/$/$/$	-/-/-/-	-/-/-/-	-/-/-/-	+/+/+/+	-/-/-/-	-/-/-/-
Logistic	$/$/$/$	+/+/+/+	+/+/+/+	-/-/-/-	+/+/+/+	-/-/-/-	+/+/+/+	-/-/-/-	+/+/-/+
Log-normal	+/+/+/$	+/+/+/+	+/+/+/+	$/$/$/$	-/-/-/-	-/-/-/-	+/+/+/+	-/-/-/-	+/+/-/+
Log-normal (multivariate)	+/+/$/$	-/-/-/-	$/$/$/$	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
Negative binomial	$/$/$/+	+/+/+/+	+/+/+/+	$/$/$/$	-/$/-/$	-/-/-/-	+/+/+/+	-/+/-/+	+/+/-/+
Normal	+/+/+/+	+/+/+/+	+/+/+/+	$/$/$/+	+/+/+/+	-/+/+/+	+/+/+/+	-/+/-/+	+/+/-/+
Normal (bivariat)	-/+/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/+/-/+	-/-/-/-	-/-/-/-
Normal (multivariate)	$/+/$/$	-/-/-/-	+/+/+/+	-/-/-/-	-/-/-/$	-/-/-/-	-/+/-/+	-/-/-/+	+/+/-/+

Functions (Version)	GAUSS (5.0)	Maple (V8.0)	Mathematica (4.2)	Matlab (6.5)	MuPAD (2.5)	O-Matrix (5.2)	Ox (3.2)	Scilab (2.6)	S-Plus (V6.1)
Pareto	$/$/$/$	-/-/-/-	+/+/+/+	-/-/-/-	-/-/-/-	-/-/-/-	+/+/+/+	-/-/-/-	-/-/-/-
Poisson	$/$/$/+	+/+/+/+	+/+/+/+	$/$/$/$	+/+/+/+	-/-/-/-	+/+/+/+	-/+/-/+	+/+/-/+
Rayleigh	-/-/-/-	-/-/-/-	+/+/+/+	$/$/$/$	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
S	-/-/-/-	-/-/-/-	$/$/$/$	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/+	-/-/-/-	+/+/-/+
Student's t	+/+/+/$	+/+/+/+	+/+/+/+	$/$/$/$	+/+/+/+	+/+/+/-	+/+/+/+	-/+/-/-	+/+/-/+
Student's t (non-central)	+/+/+/$	-/-/-/-	+/+/+/+	$/$/$/$	-/-/-/-	-/-/-/-	-/+/-/-	-/-/-/-	-/-/-/-
Student's t (multivariate)	$/$/$/$	-/-/-/-	+/+/+/+	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
Uniform	$/$/-/+	+/+/+/+	+/+/+/+	$/$/$/+	+/+/+/+	-/-/-/+	-/-/-/+	-/-/-/+	+/+/-/+
Von Mises	-/-/-/+	-/-/-/-	$/$/$/$	-/-/-/-	-/-/-/-	-/-/-/-	+/+/+/+	-/-/-/-	-/-/-/-
Weibull	$/$/$/$	+/+/+/+	+/+/+/+	$/$/$/$	+/+/+/+	-/-/-/-	+/+/+/+	-/-/-/-	+/+/-/+
Wishart	$/$/$/$	-/-/-/-	+/+/+/+	-/-/-/$	-/-/-/-	-/-/-/-	-/-/-/+	-/-/-/-	-/-/-/-
Implemented functions	70.06% (124.7/178)	44.38% (79/178)	93.03% (165.6/178)	45.06% (80.2/178)	47.25% (84.1/178)	8.99% (16/178)	58.99% (105/178)	19.66% (35/178)	38.20% (68/178)

2.6. Statistics

Functions (Version)	GAUSS (5.0)	Maple (V8.0)	Mathematica (4.2)	Matlab (6.5)	MuPAD (2.5)	O-Matrix (5.2)	Ox (3.2)	Scilab (2.6)	S-Plus (V6.1)
Regression models									
Linear	+	+	+	+	+	+	+	+	+
Loess	+	-	$	+	-	-	m	-	+
Logistic Regression	-	-	-	-	-	-	-	-	+
LOGIT / PROBIT	$ / $	- / -	- / -	$ / $	- / -	- / -	- / m	- / -	+ / +
Nonlinear / Polynomial	$ / $	+ / +	+ / +	$ / +	+ / $	+ / +	- / -	+ / +	+ / +
PSN	$	-	-	-	-	-	-	-	-
Quantile	-	-	-	-	-	-	m	-	-
Tobit models	+	-	-	-	-	-	-	-	-

Functions (Version)	**GAUSS**	**Maple**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**Scilab**	**S-Plus**
	(5.0)	**(V8.0)**	**(4.2)**	**(6.5)**	**(2.5)**	**(5.2)**	**(3.2)**	**(2.6)**	**(V6.1)**
Test statistics									
Besley test	$	-	-	-	-	-	-	-	-
Breusch-Pagan test for homoscedasticity	$	-	-	-	-	-	+	-	-
Chow Test for stability	$	-	-	-	-	-	+	-	-
CUSUM test for stability	$	-	-	-	-	-	-	-	-
Davidson J-Test	$	-	-	-	-	-	-	-	-
Dickey Fuller test	$	-	-	-	-	-	-	-	-
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	$	-	-	-	-	-	+		$
Ljung-Box Q-Test	$	-	-	$	-	-	+	-	-
MacKinnon's J-Test	$	-	-	-	-	-	-	-	-
Sign test	-	-	-	$	-	-	-	-	-
T-Test	+	-	+	$	+	+	+	-	+
Wald test	$	-	-	-	-	-	+	-	-
Walsh test	$	-	-	-	-	-	-	-	-
Wilcoxon rank sum / sign test	- / -	- / -	- / -	$ / $	- / -	- / -	- / -	- / -	+ / +
Z-Test	-	-	+	$	-	-	-	-	+

Functions (Version)	GAUSS (5.0)	Maple (V8.0)	Mathe-matica (4.2)	Matlab (6.5)	MuPAD (2.5)	O-Matrix (5.2)	Ox (3.2)	Scilab (2.6)	S-Plus (V6.1)
Filter / smoothing models									
Bandpass / Lowpass / Highpass / Multiband / Bandstop	$/$/$/-/$	-/-/-/-/-	$/$/$/-/$	$/$/$/$/$	$/$/$/$/$	+/+/-/-/-	-/-/-/-/-	+/+/+/+/+	-/+/-/-/-
Battle-Lemarie	-	-	$	-	-	-	-	-	-
Bessel	$	-	$	$	-	-	-	-	-
Butterworth	$	-	$	$	$	+	-	+	-
Chebyshev	$	-	$	$	$	+	-	+	-
Coiflet	$	-	$	-	-	-	-	-	-
Daubechies	$	-	$	$	-	-	-	-	-
Elliptic	-	-	$	$	$	-	-	+	-
Haar	$	-	$	$	-	-	-	-	-
Hodrick-Prescott	-	-	-	-	-	-	+	-	-
IIR / FIR	$ / $	- / -	$ / $	$ / $	$ / $	- / -	- / -	+ / +	+ / +
Kernel	-	-	-	-	-	-	+	-	+
Linear	$	-	$	$	$	-	+	+	-
Meyer	-	-	$	$	-	-	-	-	-
Pollen	$	-	-	-	-	-	-	-	-
Riccati	-	-	$	$	$	-	-	+	-
Shannon	-	-	$	-	-	-	-	-	-
Savitzky-Golay	$	-	-	$	-	-	-	-	-
Time series models									
ARMA / ARIMA / ARFIMA / ARMAX	+/$/$/-	- / - / - / -	$/$/-/-	$/-/$/$	-/-/-/$	+ / - / - / -	+/+/+/+	-/-/-/+	+/+/-/-
GARCH / ARCH / AGARCH / EGARCH / IGARCH / MGARCH / PGARCH / TGARCH models	$/$/$/$/$/$ /$/$	-/-/-/-/ -/-/-/-	$/$/-/-/ -/-/-/-	$/$/-/-/ -/-/-/-	-/-/-/-/ -/-/-/-	-/-/-/-/ -/-/-/-	m/m/-/m/ m/-/-/-	-/-/-/-/ -/-/-/-	$/$/$/$/ -/$/$/$
Holt's Winter additive / multiplicative	$ / $	- / -	- / -	- / -	- / -	- / -	m / m	- / -	- / -
Multivariate GARCH models (Diagonal VEC / BEKK / Matrix Diagonal / Vector Diagonal)	$/$/$/$	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	$/$/$/$
Partial autocorrelation	+	-	-	$	-	-	+	-	-
Spectral analysis	$	-	$	$	$	-	+	+	+

Functions (Version)	GAUSS (5.0)	Maple (V8.0)	Mathematica (4.2)	Matlab (6.5)	MuPAD (2.5)	O-Matrix (5.2)	Ox (3.2)	Scilab (2.6)	S-Plus (V6.1)
State space models	$	-	$	$	$	-	m	+	-
Time series analysis (Stationary / Non-stat.)	$ / $	- / -	$ / $	$ / $	$ / $	- / -	+ / +	+ / +	+ / +
Wavelets	$	-	$	$	$	+	-	+	$
Multivariate statistics									
ANOVA / MANOVA	- / -	+ / -	+ / -	$ / $	- / -	- / -	- / -	- / -	+ / +
Discriminant analysis	-	-	-	$	-	-	-	-	+
Fuzzy clustering	-	-	-	$	-	-	-	-	+
Hierachical cluster analysis	$	-	-	$	-	-	-	-	+
K-means cluster analysis	-	-	-	-	-	-	-	-	+
Procrustes analysis	-	-	-	$	-	-	-	-	+
Principal component analysis	+	-	-	$	-	-	-	-	+
Survival analysis	-	-	-	-	-	-	-	-	+
Design of Experiments									
Box-Behnken design	-	-	-	$	-	-	-	-	-
Central composite design	-	-	-	$	-	-	-	-	-
D-Optimal design	-	-	-	$	-	-	-	-	-
Full / Fractional factorial design	- / -	- / -	- / -	$ / $	- / -	- / -	- / -	- / -	- / -
Hadamard design	-	-	-	$	-	-	-	-	-
Other statistical functions & models									
Bootstrapping	$	-	-	$	-	-	+	-	+
Duration models	$	-	-	-	-	-	-	-	-
Entropy models	$	-	-	$	-	-	-	-	-
Event count models	$	-	-	-	-	-	-	-	+
Heckman two step estimation	$	-	-	-	-	-	-	-	-
Heteroscedasticity	$	-	-	-	-	-	+	-	+
Jacknife estimation	$	-	-	-	-	-	-	-	+

Functions (Version)	GAUSS (5.0)	Maple (V8.0)	Mathematica (4.2)	Matlab (6.5)	MuPAD (2.5)	O-Matrix (5.2)	Ox (3.2)	Scilab (2.6)	S-Plus (V6.1)
Lagrange multiplier test	$	-	-	-	-	-	+	-	-
Markowitz efficient frontier	$	-	$	$	-	-	-	-	-
Maximum Likelihood (Unconstr. / Constr.)	$ / $	- / -	+ / $	$ / -	- / -	- / -	+ / -	- / -	+ / +
Monte Carlo simulation	$	-	$	$	-	-	+	-	-
Implemented functions	63.55% (69.9/110)	3.64% (4/110)	32.64% (35.9/110)	51.00% (56.1/110)	20.09% (22.1/110)	10.91% (12/110)	32.73% (36/110)	19.09% (21/110)	46.09% (50.7/110)

2.7. 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 (5.0)	Maple (V8.0)	Mathematica (4.2)	Matlab (6.5)	MuPAD (2.5)	O-Matrix (5.2)	Ox (3.2)	Scilab (2.6)	S-Plus (V6.1)
American binomal method / Black and Scholes - call / put	+/+/+/+	-/-/+/-	$/$/$/$	$/$/$/$	-/-/-/-	-/-/-/-	m/m/m/m	-/-/-/-	-/-/-/-
Amortization schedule	$	+	$	$	-	-	m	-	-
Cash flow model	$	+	$	$	-	-	m	-	-
Cointegration models	$	-	-	-	-	-	+	-	$
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	-	-
Portfolio analysis	$	-	$	$	-	-	m	-	$

Functions (Version)	GAUSS	Maple	Mathe-matica	Matlab	MuPAD	O-Matrix	Ox	Scilab	S-Plus
	(5.0)	(V8.0)	(4.2)	(6.5)	(2.5)	(5.2)	(3.2)	(2.6)	(V6.1)
Rational Expectation models linear / non-linear	$ / $	- / -	- / -	$ / -	- / -	- / -	- / -	- / -	- / -
Regressive-autoregressive models	$	-	$	$	-	+	+	-	+
Social network models	$	-	$	-	-	-	-	-	-
Implemented functions	82.40% (20.6/25)	20.00% (5/25)	65.60% (16.4/25)	61.60% (15.4/25)	7.60% (1.9/25)	12.00% (3/25)	60.00% (15/25)	4.00% (1/25)	15.20% (3.8/25)

2.8. Summarizing the comparison of the mathematical functions

Altogether 385 functions have been listed. The following table summarizes the results with the mentioned weighting:

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

Functions (Version)	GAUSS	Maple	Mathe-matica	Matlab	MuPAD	O-Matrix	Ox	Scilab	S- Plus
	(5.0)	(V8.0)	(4.2)	(6.5)	(2.5)	(5.2)	(3.2)	(2.6)	(V6.1)
Standard mathematics (5%)	85.00%	100.00%	100.00%	100.00%	100.00%	80.00%	100.00%	80.00%	65.00%
Algebra (15%)	77.94%	82.35%	84.12%	95.29%	83.24%	67.65%	64.71%	76.47%	60.59%
Analysis (10%)	100.00%	63.64%	100.00%	100.00%	100.00%	63.64%	54.55%	100.00%	72.73%
Numerical mathematics (10%)	67.14%	85.71%	100.00%	80.00%	85.71%	71.43%	57.14%	57.14%	42.86%
Descriptive statistics, stochastic and distribution functions (20%)	70.06%	44.38%	93.03%	45.06%	47.25%	8.99%	58.99%	19.66%	38.20%
Statistics (20%)	63.55%	3.64%	32.64%	51.00%	20.09%	10.91%	32.73%	19.09%	46.09%
Other mathematics (20%)	82.40%	20.00%	65.60%	61.60%	7.60%	12.00%	60.00%	4.00%	15.20%
Overall result (100% = Best)	75.86%	45.89%	75.87%	68.83%	51.04%	34.03%	56.22%	39.74%	43.80%

3. 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 testreport on statistical and econometrical functions the following sections will only mention the most important graphic functions.

3.1. Chart types

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

Functions (Version)	**GAUSS**	**Maple**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**Scilab**	**S-Plus**
	(5.0)	**(V8.0)**	**(4.2)**	**(6.5)**	**(2.5)**	**(5.2)**	**(3.2)**	**(2.6)**	**(V6.1)**
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	+	+	+	+	+	+	+	+	+

Functions (Version)	**GAUSS**	**Maple**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**Scilab**	**S-Plus**
	(5.0)	**(V8.0)**	**(4.2)**	**(6.5)**	**(2.5)**	**(5.2)**	**(3.2)**	**(2.6)**	**(V6.1)**
3D-Graphics									
Charts	$	+	+	+	+	-	+	+	+
Contour Plot	+	+	+	+	+	+	+	+	+
Error bars	$	-	-	-	-	-	-	-	-
Height colors	+	+	+	+	+	+	-	+	+
Spectral Plots	$	-	$	+	-	-	+	-	-
Surface Plot	+	+	+	+	+	+	+	+	+
XYZ Plot	+	+	+	+	+	+	+	+	+
Special graphic types and functions									
Animations	$	+	+	+	+	-	-	+	-
Bollinger bands	-	-	$	$	-	-	-	-	-
Box & Whisker Plots	+	+	$	$	+	-	+	-	+
Candlestick charts	-	-	$	$	-	-	-	-	-
Cluster graphs	-	-	-	+	-	-	-	-	+
Dendograms	$	-	-	$	-	-	-	-	$
Pareto Charts	$	+	-	$	-	-	-	-	+
Periodograms	-	-	-	$	-	-	+	-	+
Pyramide Plot	$	-	-	-	-	-	-	-	-
QQ Plot	$	+	-	$	-	-	+	-	+
Quantile Plot	+	+	-	$	-	-	+	-	+
Scatter Plot Matrix	+	+	-	+	-	-	+	-	+
Smith chart	-	-	$	-	-	+	-	-	+
Overall result ***(100% = Best)***	78.24% (26.6/34)	50.00% (17/34)	62.94% (21.4/34)	82.35% (28/34)	38.24% (13/34)	38.24% (13/34)	52.94% (18/34)	41.18% (14/34)	76.18% (25.9/34)

3.2. 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 all the supported file formats for graphic export and import.

Export / Import formats (Version)	**GAUSS**	**Maple**	**Mathematica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**Scilab**	**S-Plus**
	(5.0)	**(V8.0)**	**(4.2)**	**(6.5)**	**(2.5)**	**(5.2)**	**(3.2)**	**(2.6)**	**(6.0)**
BMP	+ / -	+ / -	+ / +	+ / +	+ / +	- / -	- / -	m / m	+ / +
GIF	$ / $	+ / -	+ / +	+ / +	- / -	- / -	- / -	+ / -	+ / +
JPG	$ / $	+ / -	+ / +	+ / +	+ / -	- / -	- / -	m / m	+ / +
PCX	+ / -	+ / -	- / -	+ / +	+ / -	- / -	- / -	m / m	+ / +
PS / EPS (export only)	+ / +	- / +	+ / +	+ / +	+ / -	- / -	+ / +	- / +	+ / +
TIFF	+ / -	- / -	+ / +	+ / +	+ / -	- / -	- / -	m / m	+ / +
WMF	+ / $	+ / -	+ / +	+ / -	+ / +	- / -	+ / -	- / -	+ / +
Overall result (100% = Best)	75.00% (10.5/14)	42.86% (6/14)	85.71% (12/14)	92.86% (13/14)	57.14% (8/14)	0.00% (0/14)	21.43% (3/14)	71.43% (10/14)	100.00% (14/14)

3.3. Summarizing the comparison of the graphical functionality

Altogether 34 functions 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**	**Maple**	**Mathematica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**Scilab**	**S-Plus**
	(5.0)	**(V8.0)**	**(4.2)**	**(6.5)**	**(2.5)**	**(5.2)**	**(3.2)**	**(2.6)**	**(6.0)**
Graphic types (75%)	78.24%	50.00%	62.94%	82.35%	38.24%	38.24%	52.94%	41.18%	76.18%
Graphic import/export formats (25%)	75.00%	42.86%	85.71%	92.86%	57.14%	0.00%	21.43%	71.43%	100.00%
Overall result (100% = Best)	77.43%	48.21%	68.63%	84.98%	42.96%	28.68%	45.06%	48.74%	82.13%

4. 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 non trained person can use it for their models. For this type 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 programming environment.

Programming facilities	**GAUSS**	**Maple**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**Scilab**	**S-Plus**
	(5.0)	**(V8.0)**	**(4.2)**	**(6.5)**	**(2.5)**	**(5.2)**	**(3.2)**	**(2.6)**	**(V6.1)**
Editing features									
Built-in editor	+	+	+	+	+	+	+	-	+
External editor configurable	+	+	+	+	+[1]	-	+	-	+
Source code formatting	+	-	+	+	-	-	-	m	+
Syntax highlighting	+	-	+	+	+	-	+	m	-
Debugging									
Breakpoints	+	+	-	+	+	+	+	+	+
Function Tracer	-	+	+	-	+	-	+	-	-
Line Tracing	+	-	-	+	+	+	+	-	+
Profiler	-	+	$	+	+	+	-	+	-
Stack inspection	+	-	+	+	+	+	-	+	+
Variable inspection	+	-	+	+	+	+	+	+	+
Language features									
API-interface	+	-	+	+	-	+	+	+	$
Compiler metacommands	+	-	-	-	-	-	+	-	-
DDE support	-	-	-	+	+	-	-	-	+
Fuzzy conditional functions	+	-	$	$	-	-	+	-	-
GUI programming	-	+	+	+	+	+	m	+	+
Loops head / foot controlled	+ / +	+ / +	+ / -	+ / -	+ / +	+ / -	+ / -	- / +	+ / +

[1] Actually only available under UNIX.

Programming facilities	GAUSS	Maple	Mathe-matica	Matlab	MuPAD	O-Matrix	Ox	Scilab	S-Plus
	(5.0)	(V8.0)	(4.2)	(6.5)	(2.5)	(5.2)	(3.2)	(2.6)	(V6.1)
N-dimensional arrays (> 3)	+	+	+	+	+	-	+	+	+
Object oriented programming	-	-	+	+	+	+	+	+	+
OLE support	-	+	+	+	+	-	+	-	+
P code compiling	+	-	+	+	-	-	+	+	-
Language interfaces									
Assembler	+	-	+	-	+	-	+	-	-
C/C++	+	+	+	+	+	-	+	+	+
FORTRAN	+	+	+	+	+	-	+	+	+
GAUSS	+	-	-	-	-	-	+	-	-
Maple	$	+	-	$	+	-	-	+	-
Mathematica	$	-	+	$	-	+	-	-	-
Matlab	-	+	$	+	-	-	-	+	-
MuPAD	-	-	-	-	+	-	-	-	-
O-Matrix	-	-	$	-	-	+	-	-	-
Ox	-	-	-	-	-	-	+	-	-
Scilab	-	-	-	-	$	-	-	+	-
S-Plus	-	-	-	-	-	-	-	-	+
DLL-Calls	+	+	+	+	-	+	+	+	+
Miscellaneous									
Developer Engine	$	-	-	-	-	-	+	-	$
Redistr. with runtime licenses	$	-	-	$	-	$	$	-	-
Overall result ***(100% = Best)***	65.56% (23.6/36)	41.67% (15/36)	62.78% (22.6/36)	68.33% (24.6/36)	60.83% (21.9/36)	38.61% (13.9/36)	66.39% (23.9/36)	50.00% (18/36)	55.00% (19.8/36)

5. 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?

5.1. 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**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**Scilab**	**S-Plus**
	(5.0)	**(V8.0)**	**(4.2)**	**(6.5)**	**(2.5)**	**(5.2)**	**(3.2)**	**(2.6)**	**(V6.1)**
ACCESS	-	-	$	-	-	-	-	-	+
ASCII	+	+	+	+	+	+	+	+	+
Binary	+	+	+	+	+	+	+	+	-
dBase	+	-	-	-	-	-	-	-	+
Excel	+	-	$	+	-	-	+	-	+
FoxPro	+	-	-	-	-	-	-	-	+
Labview	-	-	$	-	-	-	-	-	-
Lotus 1-2-3	+	-	-	+	-	-	+	-	+
ODBC-connections	m	-	$	$	-	-	-	-	+
Paradox	+	-	-	-	-	-	-	-	+
SAS / SPSS / STATA	- / - / -	- / - / -	- / - / -	- / - / -	- / - / -	- / - / -	- / - / +	- / - / -	+ / + / +
XML	-	+	+	+	+	-	-	-	+
Overall result ***(100% = Best)***	57.14% (8/14)	21.43% (3/14)	50.77% (6.6/14)	45.39% (5.9/14)	21.43% (3/14)	14.29% (2/14)	35.71% (5/14)	14.29% (2/14)	85.71% (12/14)

5.2. 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 value, outliers and several other routines. The following tables shows an extraction of some important functions of this type.

Data preprocessing functions	GAUSS	Maple	Mathematica	Matlab	MuPAD	O-Matrix	Ox	Scilab	S-Plus
	(5.0)	(V8.0)	(4.2)	(6.5)	(2.5)	(5.2)	(3.2)	(2.6)	(V6.1)
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	- / +	+ / +	+ / +	+ / +	+ / +	- / -	- / -	+ / +	+ / +

Data preprocessing functions	GAUSS	Maple	Mathe-matica	Matlab	MuPAD	O-Matrix	Ox	Scilab	S-Plus
	(5.0)	(V8.0)	(4.2)	(6.5)	(2.5)	(5.2)	(3.2)	(2.6)	(V6.1)
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)	77.14% (27/35)	62.86% (22/35)	82.86% (29/35)	82.29% (28.8/35)	42.86% (15/35)	82.86% (29/35)	68.57% (24/35)	71.43% (25/35)

5.3. Summary

Functions (Version)	GAUSS	Maple	Mathe-matica	Matlab	MuPAD	O-Matrix	Ox	Scilab	S-Plus
	(5.0)	(V8.0)	(4.2)	(6.5)	(2.5)	(5.2)	(3.2)	(2.6.1)	(6.1)
Data import/export (70%)	57.14%	21.43%	50.77%	45.39%	21.43%	14.29%	35.71%	14.29%	85.71%
Data handling and preparation (30%)	91.43%	77.14%	62.86%	82.86%	82.29%	42.86%	82.86%	68.57%	71.43%
Overall result (100% = Best)	67.43%	38.14%	54.40%	56.63%	39.69%	22.86%	49.86%	30.57%	81.43%

6. 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 PC 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 tables therefore gives an overview about the availability of each mathematical program for several common platforms.

Platform (Version)	**GAUSS**	**Maple**	**Mathematica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**Scilab**	**S-Plus**
	(5.0)	**(V8.0)**	**(4.2)**	**(6.5)**	**(2.5)**	**(5.2)**	**(3.2)**	**(2.6)**	**(V6.1)**
HP 9000 (HP-UX)	+	+	+	+	-	-	+	+	-
IBM RISC (IBM AIX)	+	+	+	+	-	-	+	+	-
Intel (Win. 9x,NT,2000, XP, Me)	+	+	+	+	+	+	+	+	+
Intel (Linux)	+	+	+	+	+	-	+	+	+
Motorola (MAC OS)[2]	-	+	+	+	-	-	-	+	-
SGI (SGI IRIX)	+	+	+	+	-	-	+	+	-
SUN (Solaris)	+	+	+	+	+	-	+	+	+
Total amount	85.71% (6/7)	100.00% (7/7)	100.00% (7/7)	100.00% (7/7)	42.86% (3/7)	14.29% (1/7)	85.71% (6/7)	100.00% (7/7)	42.86% (3/7)

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.

[2] Mac OS X or higher.

7. Speed comparison

The following speed comparison have been performed on a Pentium-III with 550 MHz and 384 MB RAM running under Windows 2000 Prof. (all timings are displayed in seconds). As it could be expected that modern computers could solve the given problems within a short time, the maximum running time for each function has been limited to 10 minutes.

The speed comparison tests 18 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 results 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	MuPAD[3]	O-Matrix	Ox	Scilab	S-Plus
	(5.0)	(V8.0)	(4.2)	(6.5)	(2.5)	(5.2)	(3.2)	(2.6)	(V6.1)
Reading data from an ASCII data file	0.891	6.079	3.435	2.437	10.466	0.631	0.688	0.908	2.113
Reading data from a database over ODBC interface	2.263[4]	-	3.215	10.455[5]	-	-	-	-	1.582
Extraction of submatrices and descriptive statistic	3.732	*	50.633	7.491	410.631	5.057	4.860	12.440	*
Loop test 5000 x 5000	28.310	230.822	298.350	208.780	327.090	63.331	46.076	*	*
3800x3800 random matrix^1000	13.313	*	9.443	23.184	*	9.353	12.474	133.402	30.724
Sorting of 3000000 random values	13.115	41.820	8.322	3.044	45.044	2.954	13.460	8.462	10.145
FFT over 1048576 (= 2^20) random values	8.195	182.963	2.534	1.572	41.309	2.514	4.630	4.717	4.977
Triple integration	0.024	57.593	*	59.536	444.600	-	-	204.474	-
Determinant of a 1000x1000 random matrix	15.630	60.226	14.942	2.834	*	2.003	8.556	24.976	6.009
Inverse of a 1000x1000 random matrix	44.131	17.094	78.913	7.361	*	5.258	18.553	71.993	50.713
Eigenvalues of a 600x600 random matrix	35.124	41.179	28.160	15.722	*	8.723	31.065	73.526	27.279
Cholesky decomposition of a 1000x1000 random matrix	3.949	*	4.777	1.242	*	0.921	2.113	9.603	8.372
1000x1000 crossproduct matrix	26.776	20.428	26.398	5.488	*	4.807	5.508	127.113	7.021
Calculation of 1000000 Fibonacci numbers	1.980	*	25.426	4.587	*	2.163	1.205	2.714	2.103
Principal component factorization over a 500x500 matrix	26.435	-	160.571	23.754	*	-	-	-	18.437
Gamma function on a 1500x1500 random matrix	2.504	*	26.007	4.266	*	6.430	1.489	5.438	6.739
Gaussian error function on a 1500x1500 random matrix	3.211	*	16.424	2.593	339.849	2.864	1.519	4.566	-

[3] MuPAD is also available with Scilab plugin. In this case the timings of MuPAD are approximately the same as Scilab.
[4] The Database Connection Kit (DCK) is needed to run this funcion.
[5] Database Toolbox is necessary to run this function.

Functions	GAUSS	Maple	Mathe-matica	Matlab	MuPAD[3]	O-Matrix	Ox	Scilab	S-Plus
	(5.0)	(V8.0)	(4.2)	(6.5)	(2.5)	(5.2)	(3.2)	(2.6)	(V6.1)
Linear regression over a 1000x1000 random matrix	15.750	28.101	4.736	3.095	*	1.942	7.611	25.296	19.007
Overall performance	49.63%	7.27%	24.52%	53.40%	1.550%	80.84%	59.78%	20.50%	35.36%

* ***Maximum running time of 10 minutes has been exceeded.***

The overall performance have been calculated in the following way:

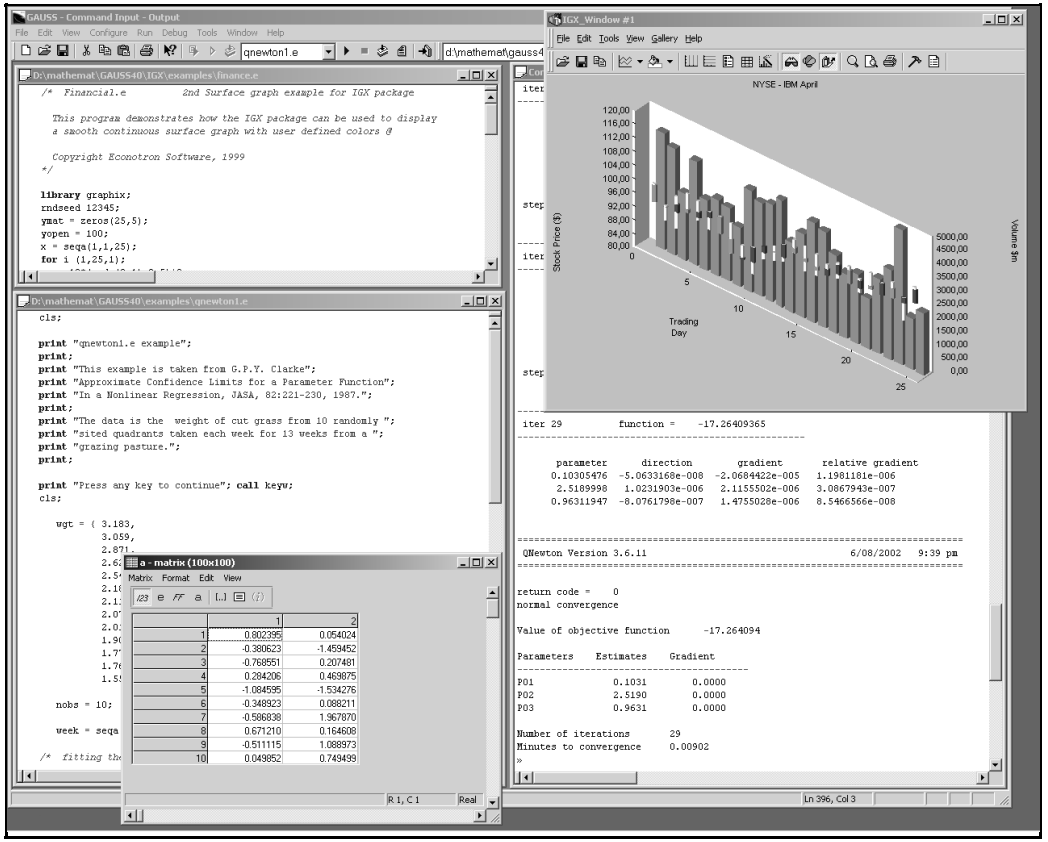
The best timing result of a benchmark function makes 100%; for calculating the results for each function I'll take the fastest timing and divide it by the timing of the tested program (*the formula will look MIN(A1;A2;...)/A2 for example*) and that makes the ranking in percentage. To calculate the final „Overall performance" I'll than added the percentage values for each tested program and divided by the amount of tested functions (*in the moment 18*) which gives again a result as a percentage.

Functions which are not supported by a Program have not been judged in this 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.

8. Summary and interpretation of the test results

8.1. General product information

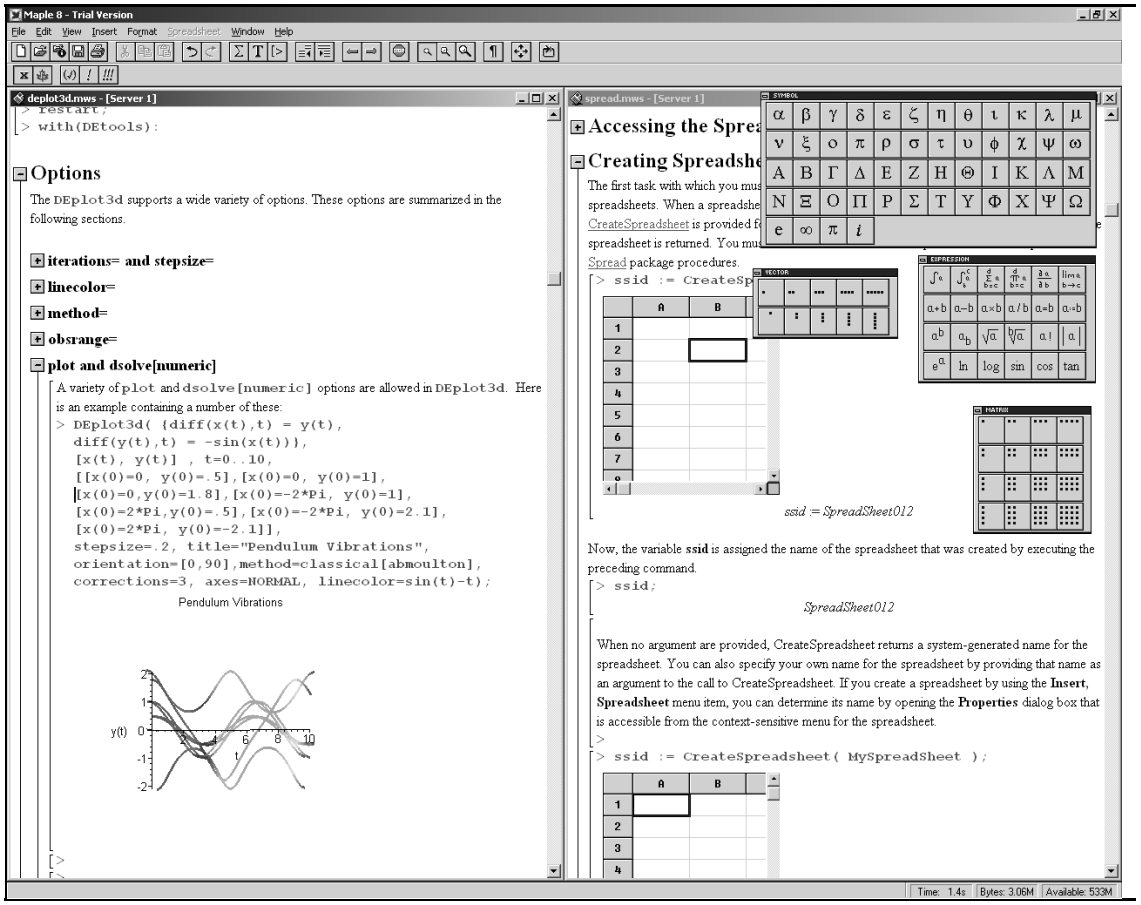
8.1.1. GAUSS



Product	GAUSS
Version	5.0
Producer	Aptech Systems Inc.
Location of headquarter	Maple Valley, USA
Internet	www.aptech.com
Price (commercial)	≈ 2799 US$ (2994 €)
Price (academic)	≈ 707 US$ (756 €)
Price (student)[6]	≈ 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.

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

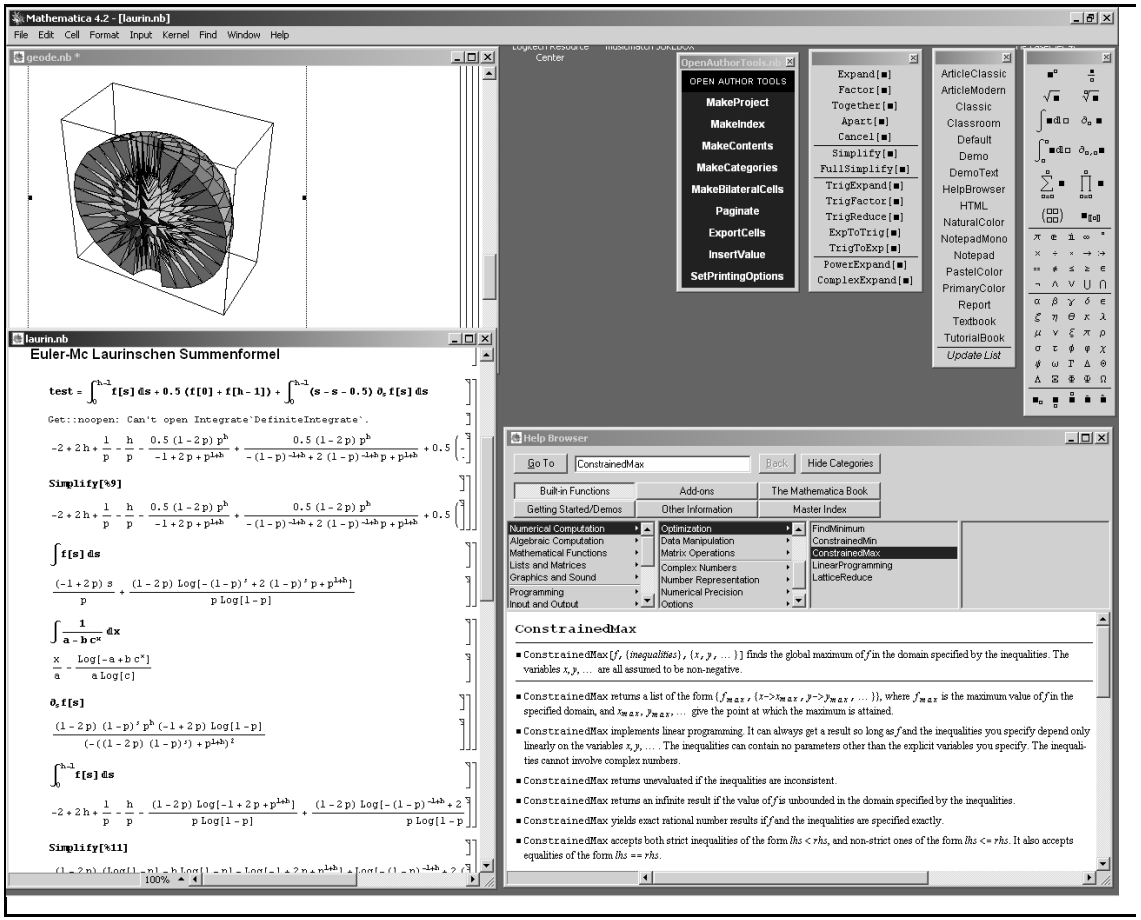
8.1.2. Maple



Product	Maple
Version	8.0
Producer	Waterloo Maple Software Inc.
Location of headquarter	Waterloo, Canada
Internet	www.maplesoft.com
Price (commercial)	≈1989 US$ (2125 €)
Price (academic)[7]	(≈1766 US$ / ≈1112 US$) (1890 € / 1190 €)
Price (student)	≈186 US$ (199 €)
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. It's development is focused on the CAS sector and is believed to be one of the market leaders.

[7] Different pricing for research and academic.

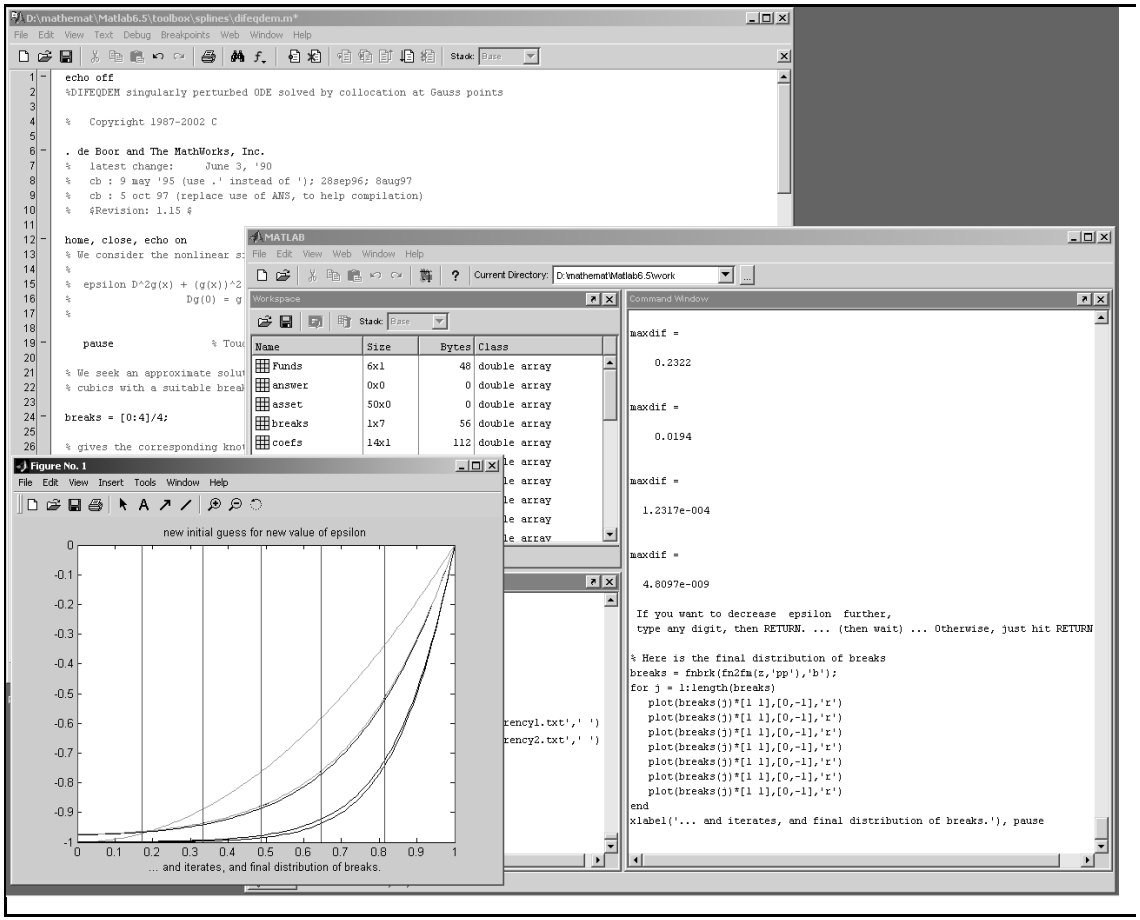
8.1.3. Mathematica



Product	Mathematica
Version	4.2
Producer	Wolfram Research Inc.
Location of headquarter	Champaign, USA
Internet	www.wolfram.com
Price (commercial)	1880 US$[8] (2120 €)
Price (academic)	895 US$ (1220 €)
Price (student)	140 US$ (128 €)
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://www.mathsource.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 alaso a CAS which commonly usable.

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

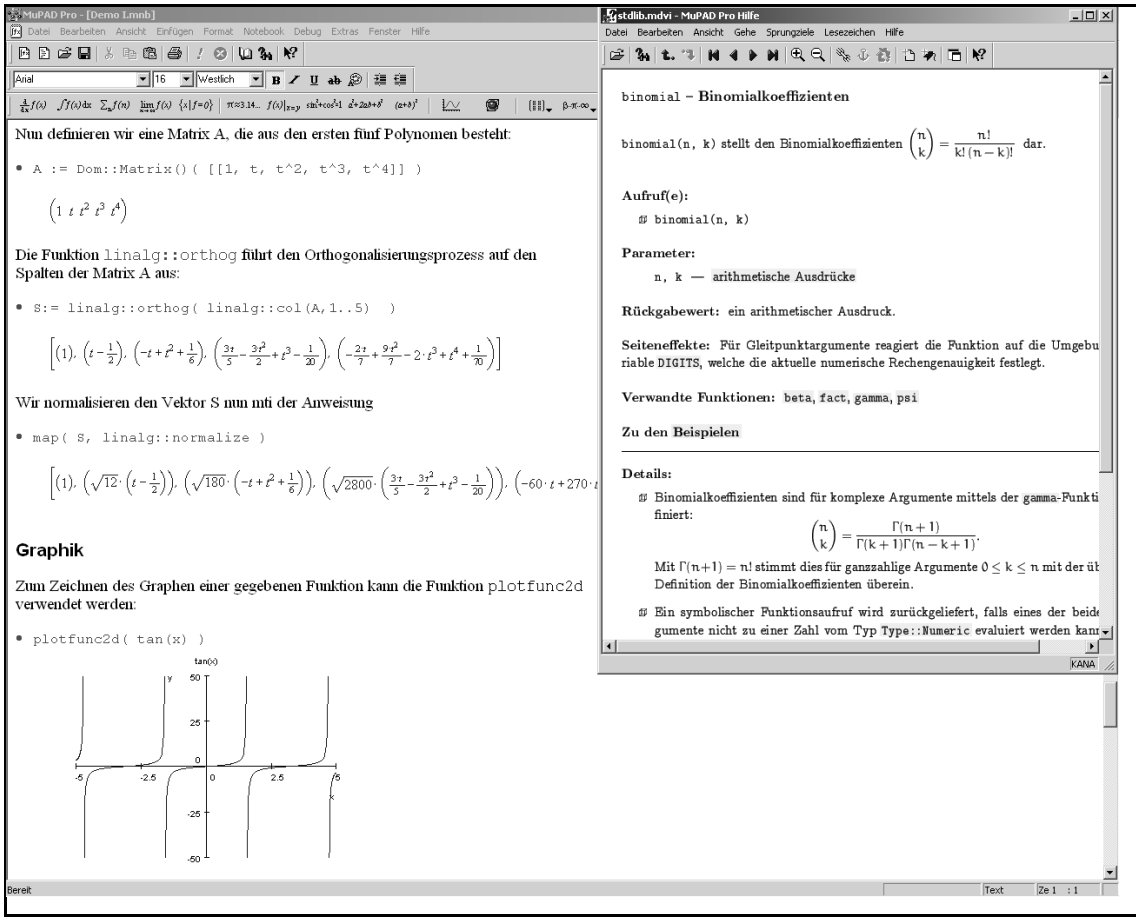
8.1.4. Matlab



Product	Matlab
Version	6.5
Producer	The Mathworks Inc.
Location of headquarter	Nattick, USA
Internet	www.mathworks.com
Price (commercial)	1900 US$ (≈ 2033 €)
Price (academic)	varies
Price (student)	99 US$[9] (≈ 106 €)
FAQ list	In general a support FAQ list is available over Mathworks at http://www.mathworks.com/support/general/ 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 is available at http://www.mathtools.net/ 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 math.program Matlab started its development in 1984 with mainly technical aims. Today Matlab offers functions and competance for nearly all important topics.

[9] Student version is restricted in functionality.

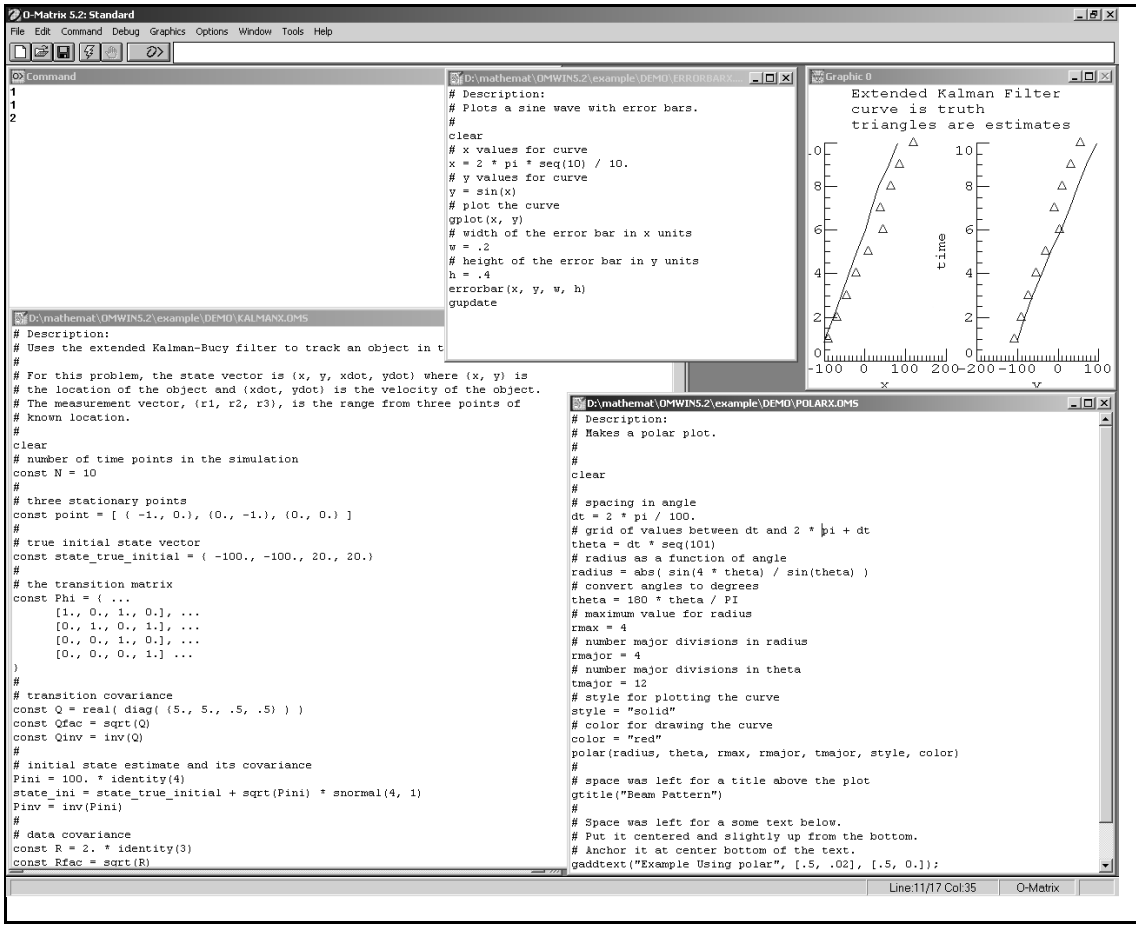
8.1.5. MuPAD



Product	MuPAD
Version	2.5
Producer	SciFace Software GmbH&Co.KG
Location of headquarter	Paderborn, Germany
Internet	www.sciface.com
Price (commercial)	500 US$ / 675 US$[10] (≈ 535 € / 722 €)
Price (academic)	300 US$ / 405 US$ (≈ 321 € / 433 €)
Price (student)	100 US$ / 135 US$ (≈ 107 € / 144 €)
FAQ list	Available at the website of SciFace.
Mailing lists	A mailing list is available by sending Subscribe to owner-mupad@informatik.uni-oldenburg.de
Newsgroup	---
Archives	---
Books	Books for MuPAD are rare. Only a few publisher offer books for MuPAD.
Remark	MuPAD started its development at the University of Paderborn in Germany and became a commercial product in 1997. During its University times the focus of the development was lying on mathematical topics and since it became a commercial product also the user interface has developed. Today it looks like a promising product which is worth watching in the future.

[10] Price including SciLab plugin.

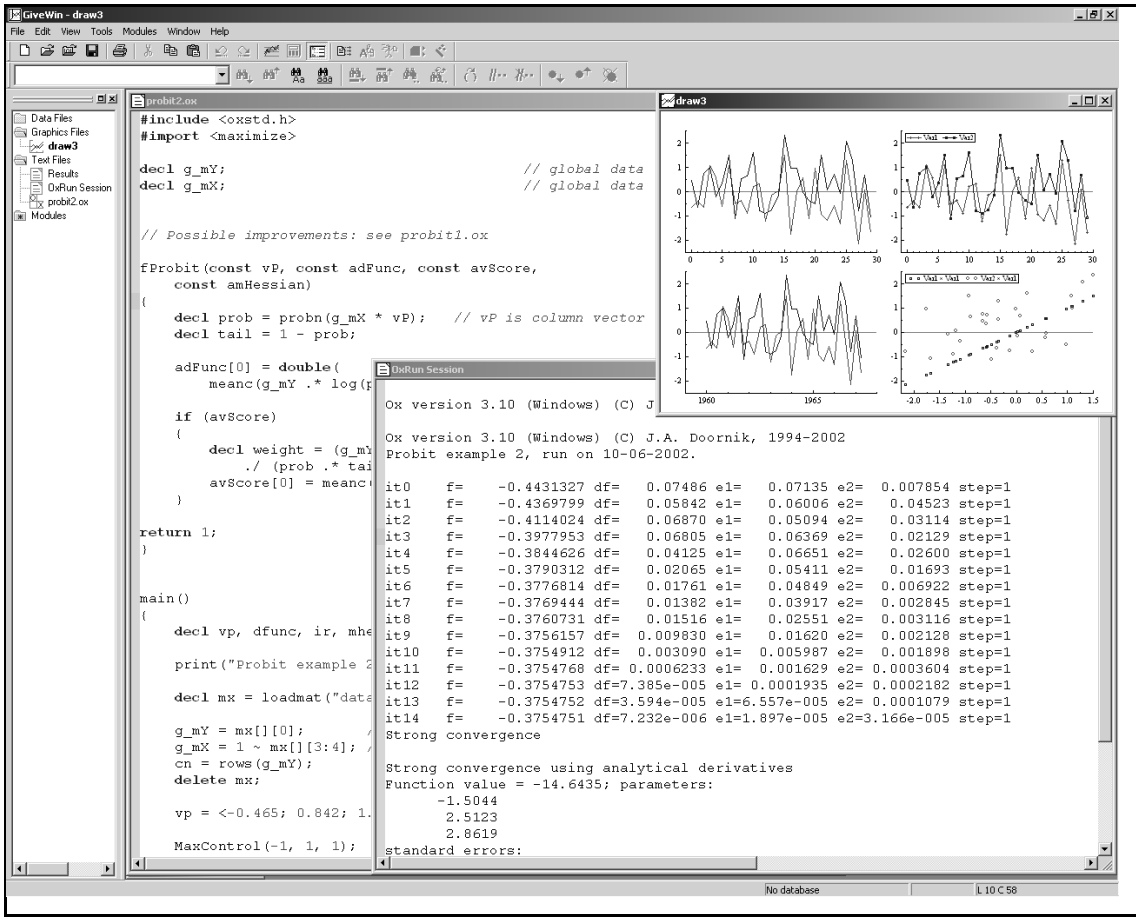
8.1.6. O-Matrix



Product	O-Matrix
Version	5.2
Producer	Harmonics Software Inc.
Location of headquarter	Seattle, USA
Internet	www.omatrix.com
Price (commercial)	265 US$[11]
Price (academic)	49 US$
Price (student)	49 US$
FAQ list	---
Mailing lists	---
Newsgroup	---
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.

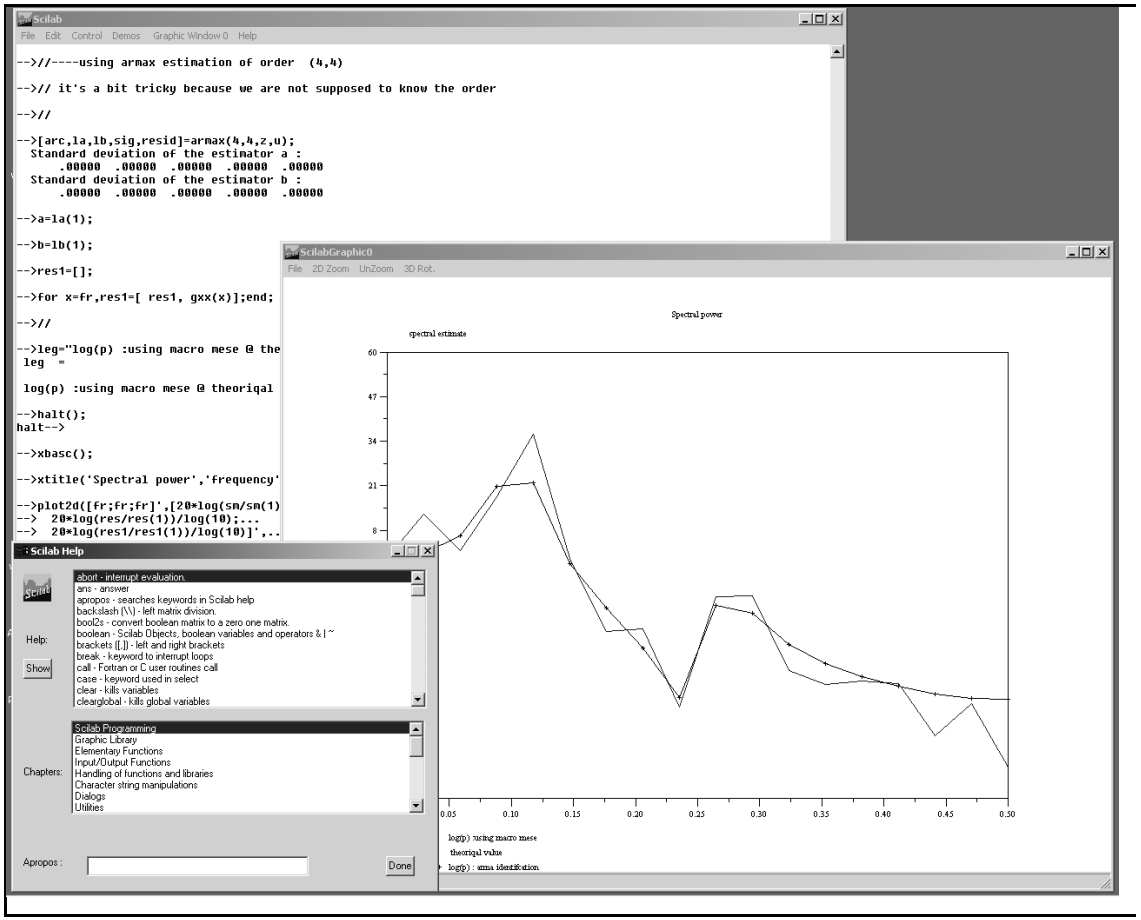
[11] Light version is memory limited. Light version available for free

8.1.7. Ox



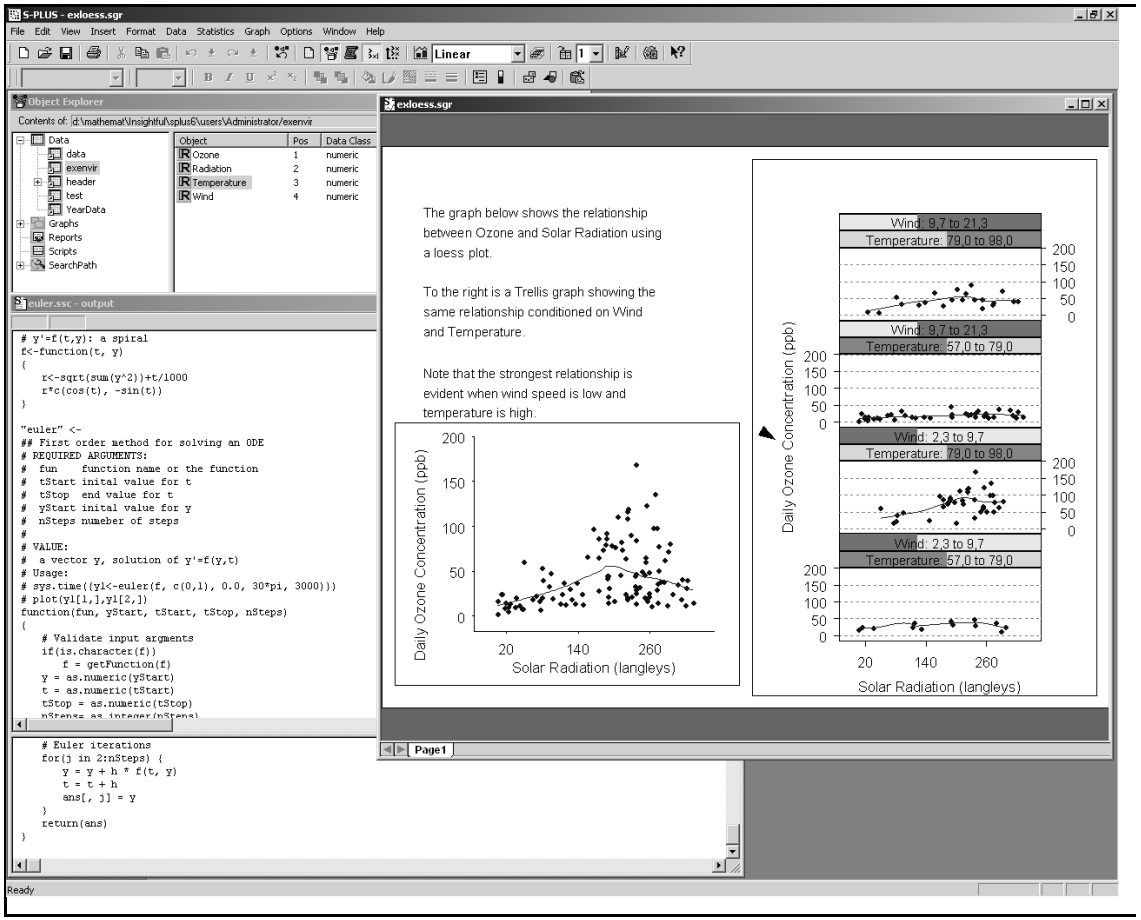
Product	Ox
Version	3.2
Producer	Jurgen Doornik / OxMetrics
Location of headquarter	United Kingdom
Internet	www.oxmetrics.com
Price (commercial)	≈ 748 US$ (850 €)
Price (academic)	≈ 374 US$ (400 €)
Price (student)	≈ 127 US$ (140 €)
FAQ list	---
Mailing lists	A mailing list is available by join ox-users firstname lastname at jiscmail@jiscmail.ac.uk
Newsgroup	---
Archives	---
Books	---
Remark	Ox is a very young product, specialized on economic topics. Its strength is the very good performance and a big range of routines for econometrics.

8.1.8. Scilab



Product	SciLab
Version	2.6
Producer	INRIA / Dr. Scilab
Location of headquarter	France
Internet	www.scilab.org
Price (commercial)	free
Price (academic)	free
Price (student)	free
FAQ list	An FAQ list is available at http://www-rocq.inria.fr/scilab/
Mailing lists	---
Newsgroup	Available at comp.soft-sys.math.scilab
Archives	---
Books	---
Remark	Scilab is a Freeware product specialized on numerical mathematics. For a freely available product it offers really a wide range of functionality.

8.1.9. S-Plus



Product	S-Plus
Version	6.1
Producer	Insightful Inc.
Location of headquarter	Seattle, USA
Internet	www.insightful.com
Price (commercial)	1695 £ (2770 US$)
Price (academic)	795 £ (1299 US$)
Price (student)	unknown
FAQ list	A very detailed FAQ list is available over the website of Insightful.
Mailing lists	Available by sending Subscribe to s-news-request@utstat.toronto.edu
Newsgroup	---
Archives	There are only a few amount of archives available, most of them are at Universities i.e. http://lib.stat.cmu.edu/S/
Books	A few books for programming and statistics topics are available for S-Plus.
Remark	S-Plus startet its development with the programming language S. Within several years of development it became a powerful tool with the focus on biomedical routines.

8.1. Miscellaneous information

Several informations 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 testreport. It's up to each reader of this report to make his/her own decisions on these information

Functions (Version)	**GAUSS** (5.0)	**Maple** (V8.0)	**Mathematica** (4.2)	**Matlab** (6.5)	**MuPAD** (2.5)	**O-Matrix** (5.2)	**Ox** (3.2)	**Scilab** (2.6)	**S-Plus** (6.1)
Operation / Programming handling									
User interface	2	3	2	2	3	3	4	4	2
Graphics	2[12]	3	2	3	3	5	3	3	2
Programming language (similar to)	2 (Basic, Fortran)	2 (Pascal)	3 (Lisp, APL)	2 (Basic, Fortran)	3 (Pascal)	2 (Basic, Fortran)	3 (C, C++)	2 (Basic, Fortran)	3 (C++)
Books / Support									
Online help / Electronic handbook	3 (2)	2	1	2	3	2	3	5 (3)	4 (2)
Additional books	5	1	1	3	5	6	6	5	3
FAQ lists	5	2	2	2	4	6	5	4	2
Newsgroups / mailing lists	1	2	1	1	5	6	2	3	2
Program archives by the software producer	6	3	1	2	5	4	4	4	6
Program archives by external institutions	1	1	1	3	5	6	5	5	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 have been justified never occurred. Conversely it was mostly also not justified to give the mark 1 as only very rare features are supported so excellent.

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

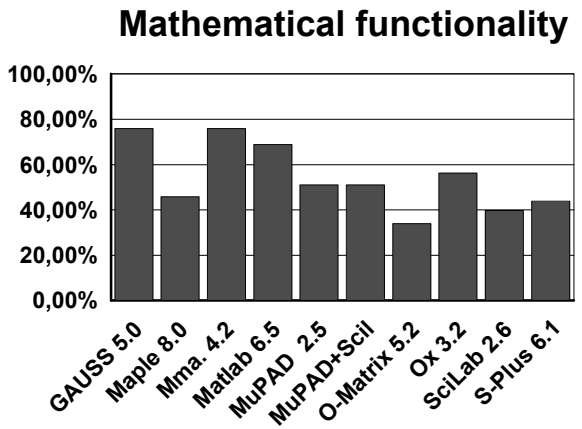
8.2. Summary

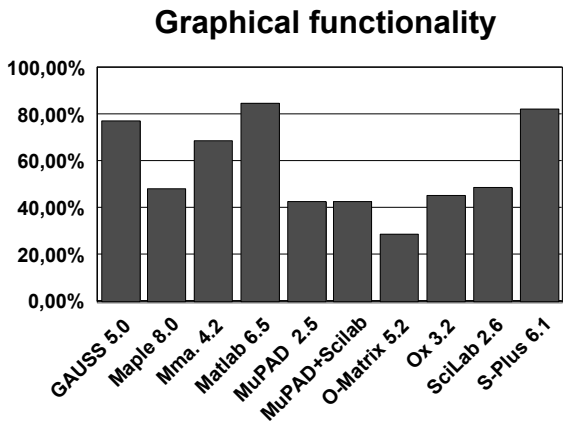
The summary should set the results of the speed comparison, the functionality of the programming environment, the data import/export facilities and the availability for different platforms in relation to the results of the comparison of the mathematical and graphical functionality. The relation between these four tests is 38:10:9:5:2:36.

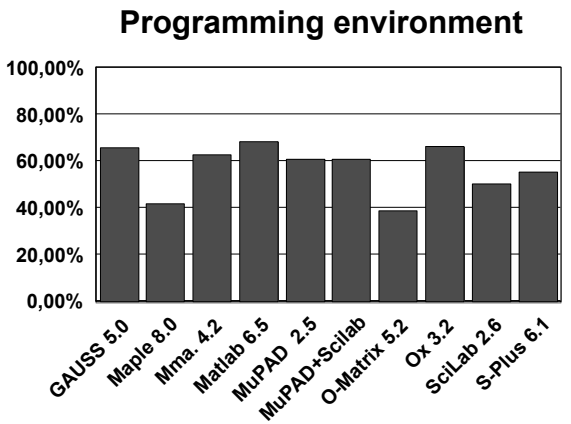
Test	GAUSS	Maple	Mathe-matica	Matlab	MuPAD	O-Matrix	Ox	Scilab	S-Plus
	(5.0)	(V8.0)	(4.2)	(6.5)	(2.5)	(5.2)	(3.2)	(2.6)	(V6.1)
Comparison of the mathematical functionality (38%)	75.86%	45.89%	64.24%	68.83%	51.04%	34.03%	56.22%	39.74%	43.80%
Comparison of the graphical functionality (10%)	77.43%	48.21%	68.63%	84.98%	42.96%	28.68%	45.06%	48.74%	82.13%
Functionality of the programming environment (9%)	65.56%	41.67%	62.78%	68.33%	60.83%	38.61%	66.39%	50.00%	55.00%
Data handling (5%)	67.43%	38.14%	54.40%	56.63%	39.69%	22.86%	49.86%	30.57%	81.43%
Available platforms (2%)	85.71%	100.00%	100.00%	100.00%	42.86%	14.29%	85.71%	100.00%	42.86%
Speed comparison (36%)	49.63%	7.27%	24.52%	53.40%	1.55% (20.50%)[13]	80.84%	59.78%	20.50%	35.36%
Overall result	65.42%	32.53%	54.89%	64.86%	32.57% (39.39%)[14]	49.81%	57.57%	35.38%	47.46%

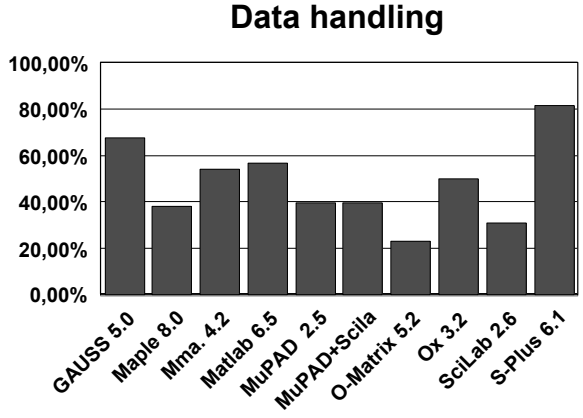
Note: The overall results of some tested programs are pretty bad due to the specific weighting of this testreport. I would like to mention that this does of course not mean that the software is bad in general but that the programs are maybe not perfect for the specific usage mentioned in this testreport,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

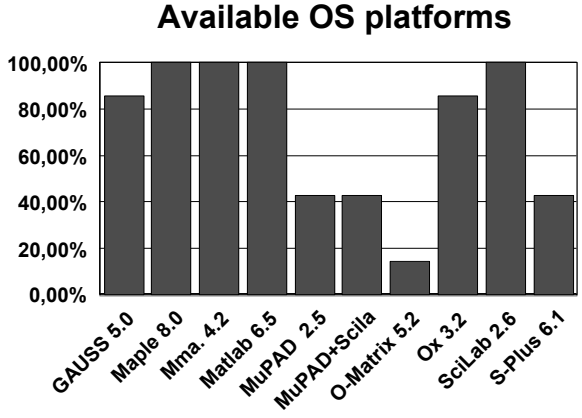
[13] Timing could be reached when using the Scilab component.
[14] Overall result when Scilab component was used.

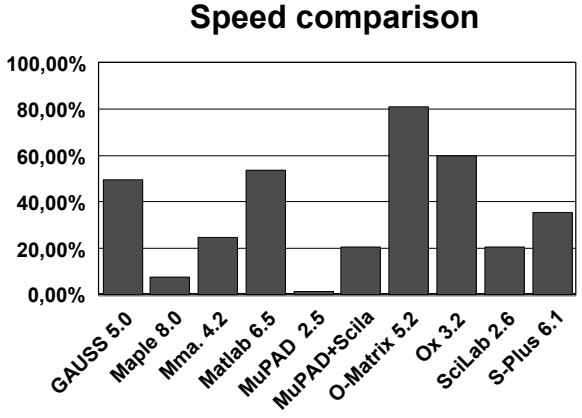

Mathematical functionality
100,00%
80,00%
60,00%
40,00%
20,00%
0,00%
GAUSS 5.0
Maple 8.0
Mma. 4.2
Matlab 6.5
MuPAD 2.5
MuPAD+Scil
O-Matrix 5.2
Ox 3.2
SciLab 2.6
S-Plus 6.1

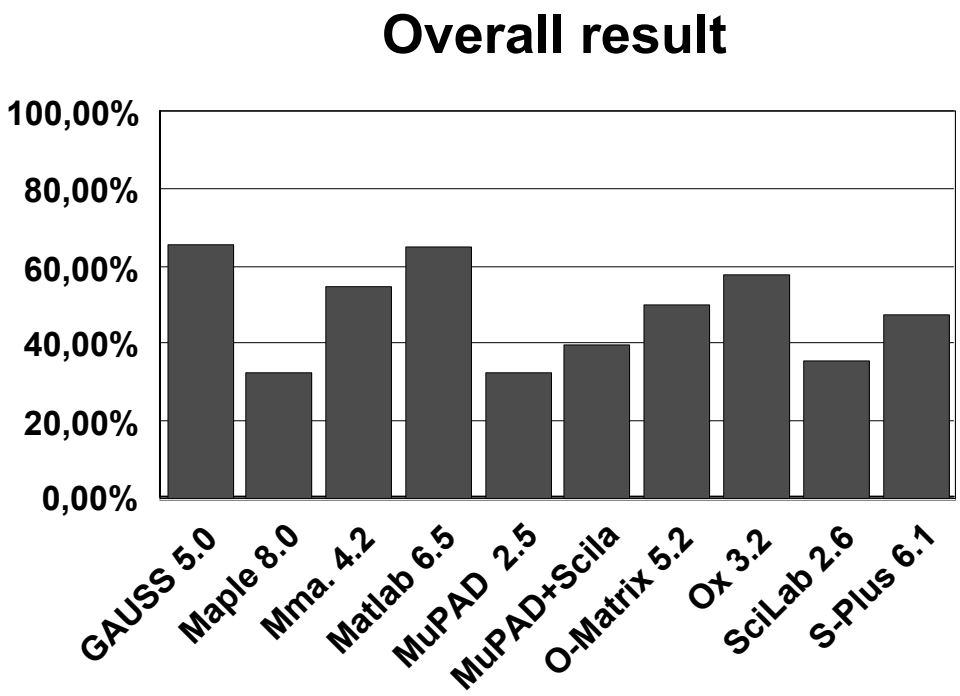

Graphical functionality
100,00%
80,00%
60,00%
40,00%
20,00%
0,00%
GAUSS 5.0
Maple 8.0
Mma. 4.2
Matlab 6.5
MuPAD 2.5
MuPAD+Scilab
O-Matrix 5.2
Ox 3.2
SciLab 2.6
S-Plus 6.1


Programming environment
100,00%
80,00%
60,00%
40,00%
20,00%
0,00%
GAUSS 5.0
Maple 8.0
Mma. 4.2
Matlab 6.5
MuPAD 2.5
MuPAD+Scilab
O-Matrix 5.2
Ox 3.2
SciLab 2.6
S-Plus 6.1


Data handling
100,00%
80,00%
60,00%
40,00%
20,00%
0,00%
GAUSS 5.0
Maple 8.0
Mma. 4.2
Matlab 6.5
MuPAD 2.5
MuPAD+Scila
O-Matrix 5.2
Ox 3.2
SciLab 2.6
S-Plus 6.1


Available OS platforms
100,00%
80,00%
60,00%
40,00%
20,00%
0,00%
GAUSS 5.0
Maple 8.0
Mma. 4.2
Matlab 6.5
MuPAD 2.5
MuPAD+Scila
O-Matrix 5.2
Ox 3.2
SciLab 2.6
S-Plus 6.1


Speed comparison
100,00%
80,00%
60,00%
40,00%
20,00%
0,00%
GAUSS 5.0
Maple 8.0
Mma. 4.2
Matlab 6.5
MuPAD 2.5
MuPAD+Scila
O-Matrix 5.2
Ox 3.2
SciLab 2.6
S-Plus 6.1



Other mathematical programming languages: 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 testreport. Mostly the problem was that I didn't received 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:

- MathView by MathWizards Inc. (http://www.mathwizards.com)
- Octave (Freeware available at http://bevo.che.wisc.edu/octave/)
- R (Freeware available at http://stat.auckland.ac.nz/r/r.html)

Appendix A: Listings of the benchmark tests

GAUSS Benchmark program:

```
/*************************************************************************/
/* GAUSS Benchmark program version 3.0                                   */
/* Author : Stefan Steinhaus                                             */
/* EMAIL  : stefan@steinhaus-net.de                                      */
/* This program is public domain. Feel free to copy it freely.           */
/*************************************************************************/

proc g1(x,y,z);
      retp(sin(x^2)+sin(y^2)+sin(z^2));
endp;

proc g2(x,y,z);
      retp(2^sin(x)+3^cos(y)+4^sin(z));
endp;

proc g3(x,y,z);
      retp(2^log(x^2+y^2+z^2));
endp;

proc g4(x,y,z);
      retp(sin(x^2-y^2-z^2));
endp;

proc g5(x,y,z);
      retp(cos(x^2+y^2+z^2));
endp;

proc g6(x,y,z);
      retp(sin(x^2+y^2+z^2));
endp;

/* Test of IO function - Reading of data from ASCII file */

proc 1 = IOTestASCII(runs);
  local result,zeit,timing,i;
  local fname,datmat,ColName;
  result=0;
  fname="e:\\mathematik\\ncrunch4\\currency2.txt";
  _dxaschdr=1; _dxprint=0;
  for i (1,runs,1);
    timing=0; zeit=hsec;
    {datmat,ColName}=import(fname,0,0);
    timing=(hsec-zeit)/100;
    result=result+timing;
  endfor;
  retp(result/runs);
endp;

/* Test of IO function - Reading of data over ODBC */

proc 2 = IOTestODBC(runs,datmat);
  local result,zeit,timing,i;
  local fname,ret,ColName,currencies,selstmt,t1,t2,t3,t4;

currencies="Exchange_Rate_Year,Exchange_Rate_Month,Exchange_Rate_Day,AUD,ATS,BEF,BRL,CA
D,CNY,DKK,EUR,FIM,FRF,DEM,GRD,HKD,INR,IEP,ITL,JPY,KRW,MYR,MXP,NLG,NOK,NZD,PT
E,SEK,ZAR,SGD,SRI,ESP,CHF,TWD,THB,GBP,VEB";
  result=0;
  selstmt="select "$+currencies$+" from [Currency];";
  fname="e:\\mathematik\\ncrunch4\\database.mdb";
  ret = ODBCAddDataSource("Microsoft Access Driver (*.mdb)","benchmark",fname);
  for i (1,runs,1);
    timing=0; zeit=hsec;
    ODBCOpen(1,"benchmark","","");
    ODBCExecuteDirect(1,selstmt);
    datmat=ODBCReadComplete(1);
    {ColName,t1,t2,t3,t4}=ODBCDescribeCol(1);
    ODBCClose(1);
    timing=(hsec-zeit)/100;
    result=result+timing;
  endfor;
  ret = ODBCDelDataSource("Microsoft Access Driver (*.mdb)","benchmark");
  retp(result/runs,datmat);
endp;

/* Test of Extraction of submatrices and calculation of descriptive stats */
/* Doing the whole process 100 times do get higher timings               */

proc 1 = ExtractTest(runs,datmat);
  local result,zeit,timing,i,j,k;
  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};
  for i (1,runs,1);
    timing=0; zeit=hsec;
    for k (1,100,1);
```

```
    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 5000x5000 */

proc 1 = LoopTest(runs);
 local a,b,result,i,timing,zeit;
 result=0;
 for i (1,runs,1);
  timing=0;  a=1;
  zeit=hsec;
  for x (1,5000,1);
     for y (1,5000,1);
        a=a+x+y;
     endfor;
  endfor;
  timing=(hsec-zeit)/100;
  result=result+timing;
 endfor;
 retp(result/runs);
endp;

/* 3800x3800 normal distributed random matrix^1000 */

proc 1 = CreationMatrix(runs);
 local a,b,result,i,timing,zeit;
 result=0;
 for i (1,runs,1);
  timing=0; b=abs(rndn(3800,3800)/10);
  zeit=hsec; a=(b+1)^1000; timing=(hsec-zeit)/100;
  result=result+timing;
 endfor;
 retp(result/runs);
endp;

/* Sorting of 3000000 values */

proc 1 = SortFunc(runs);
 local a,b,result,i,timing,zeit;
 result=0;
 for i (1,runs,1);
  timing=0; a=rndn(3000000,1);
  zeit=hsec; b=sortc(a,1); timing=(hsec-zeit)/100;
  result=result+timing;
 endfor;
 retp(result/runs);
endp;

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

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

/* Numerical solution of 6 3D integrals */

proc 1 = Integration(runs);
 local a,result,i,xl,yl,zl,timing,zeit;
 result=0;
 for i (1,runs,1);
  timing=0; xl=pi|-pi;yl=xl;zl=xl;
    _intord=4;
  zeit=hsec;
  a=intquad3(&g1,xl,yl,zl); a=intquad3(&g2,xl,yl,zl);
  a=intquad3(&g3,xl,yl,zl); a=intquad3(&g4,xl,yl,zl);
  a=intquad3(&g5,xl,yl,zl); a=intquad3(&g6,xl,yl,zl);
  timing=(hsec-zeit)/100;
  result=result+timing;
 endfor;
 retp(result/runs);
endp;

/* Determinant of a 1000x1000 matrix */

proc 1 = Determinant(runs);
```

```
  local a,b,result,i,timing,zeit;
  result=0;
  for i (1,runs,1);
    timing=0; a=rndn(1000,1000);
    zeit=hsec; b=det(a); timing=(hsec-zeit)/100;
    result=result+timing;
  endfor;
  retp(result/runs);
endp;

/* Inverse of a 1000x1000 matrix */

proc 1 = InverseFunc(runs);
  local a,b,result,i,timing,zeit;
  result=0;
  for i (1,runs,1);
    timing=0; a=rndu(1000,1000);
    zeit=hsec; b=inv(a); timing=(hsec-zeit)/100;
    result=result+timing;
  endfor;
  retp(result/runs);
endp;

/* Eigenvalues of a 600x600 matrix */

proc 1 = EigenvaluesFunc(runs);
  local a,b,result,i,timing,zeit;
  result=0;
  for i (1,runs,1);
    timing=0; a=rndn(600,600);
    zeit=hsec; b=eig(a); timing=(hsec-zeit)/100;
    result=result+timing;
  endfor;
  retp(result/runs);
endp;

/* Cholesky decomposition of a 1000x1000-matrix */

proc 1 = CholeskyFunc(runs);
  local a,b,result,i,timing,zeit;
  result=0;
  for i (1,runs,1);
    timing=0; a=moment(rndn(1000,1000),0);
    zeit=hsec; b=chol(a); timing=(hsec-zeit)/100;
    result=result+timing;
  endfor;
  retp(result/runs);
endp;

/* 1000x1000 cross-product matrix */

proc 1 = CrossProductFunc(runs);
  local a,b,result,i,timing,zeit;
  result=0;
  for i (1,runs,1);
    timing=0; a=rndn(1000,1000);
    zeit=hsec; b=moment(a,0); timing=(hsec-zeit)/100;
    result=result+timing;
  endfor;
  retp(result/runs);
endp;

/* Calculation of 1000000 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(1000000,1));
    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 500x500 matrix */

proc 1 = PCAFunc(runs);
  local a,p,v,result,i,timing,zeit;
  result=0;
  for i (1,runs,1);
    a=rndn(500,500);
    timing=0;
    zeit=hsec;
    {p,v,a}=princomp(a,500);
    timing=(hsec-zeit)/100;
    result=result+timing;
  endfor;
  retp(result/runs);
endp;

/* Gamma function over a 1500x1500 matrix */
```

```
proc 1 = GammaFunc(runs);
  local a,b,result,i,timing,zeit;
  result=0;
  for i (1,runs,1);
    timing=0; a=rndn(1500,1500)^2;
    zeit=hsec; b=gamma(a); timing=(hsec-zeit)/100;
    result=result+timing;
  endfor;
  retp(result/runs);
endp;

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

proc 1 = GaussianFunc(runs);
  local a,b,result,i,timing,zeit;
  result=0;
  for i (1,runs,1);
    timing=0; a=rndn(1500,1500)^2;
    zeit=hsec; b=erf(a); timing=(hsec-zeit)/100;
    result=result+timing;
  endfor;
  retp(result/runs);
endp;

/* Linear regression over 1000x1000 matrix */

proc 1 = LinearRegressionFunc(runs);
  local a,b,result,i,timing,zeit;
  result=0;
  for i (1,runs,1);
    timing=0; a=rndn(1000,1000); b=seqa(1,1,1000);
    zeit=hsec; b=olsqr(b,a); timing=(hsec-zeit)/100;
    result=result+timing;
  endfor;
  retp(result/runs);
endp;



/* Main program */

format /RD 8,3;
runs=3;             /* Amount of runs for avg. timing -> suggestion at least 2 */
_GUI=0; datmat=0;
ver=sysstate(1,0);
dllibrary -a odbc;
library odbc;

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 (ASCII file)_____________________________________________  : " timing;
{timing,datmat}=IOTestODBC(runs,datmat);
print /flush "IO test (ODBC)_____________________________________________________  : " timing;
{timing}=ExtractTest(runs,datmat);
print /flush "Extraction of submatrices & descriptive statistics___________ : " timing;
{timing}=LoopTest(runs);
print /flush "Test of loops_____________________________________________________  : " timing;
{timing}=CreationMatrix(runs);
print /flush "3800x3800 normal distributed random matrix^1000_____________ : " timing;
{timing}=SortFunc(runs);
print /flush "3000000 values sorted ascending_____________________________  : " timing;
{timing}=FFTFunc(runs);
print /flush "FFT over 1048576 values (2^20)_____________________________ : " timing;
{timing}=Integration(runs);
print /flush "Triple integration_____________________________________________  : " timing;
{timing}=Determinant(runs);
print /flush "Determinant of a 1000x1000 random matrix_____________________ : " timing;
{timing}=InverseFunc(runs);
print /flush "Inverse of a 1000x1000 uniform distr. random matrix_________ : " timing;
{timing}=EigenvaluesFunc(runs);
print /flush "Eigenvalues of a normal distr. 600x600 randommatrix_________ : " timing;
{timing}=CholeskyFunc(runs);
print /flush "Cholesky decomposition of a 1000x1000-matrix_________________ : " timing;
{timing}=CrossProductFunc(runs);
print /flush "1000x1000 cross-product matrix_____________________________  : " timing;
{timing}=FibonacciFunc(runs);
print /flush "Calculation of 1000000 fibonacci numbers____________________ : " timing;
{timing}=PCAFunc(runs);
print /flush "Principal component analysis over a 500x500 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 3)

Initialisation

restart;
with(LinearAlgebra):
with(linalg):
with(stats):
Seed=randomize():

Misc. operations
IO Test

t:=time():
fd := fopen("E:\\Mathematik\\Ncrunch4\\Currency1.txt",READ,TEXT):
header:=readline(fd):
datmat:=readdata(fd,[integer,integer,integer,integer,float,float,float,float,float,float,float,float,float,float,fl
oat,float,float,float,float,float,float,float,float,float,float,float,float,float,float,float,float,float,floa
t,float,float,float,float,float,float,float,float,float]):
fclose(fd):
time()-t;

Extraction of submatrices & descriptive statistics

LimitsArea:={1,261,522,784,1045,1305,1565,1827,2088,2349,2610,2871,3131,3158}:
t:=time():
for k from 1 by 1 to 100 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
            MinYear:=min(op(YearData[1..amountrows,i])):
            MaxYear:=max(op(YearData[1..amountrows,i])):
            MeanYear:=describe[mean](YearData[1..amountrows,i]):

GainLossYear:=100-(YearData[1,i]+0.00000001)/(YearData[amountrows,i]+0.00000001)*100:
        end do;
    end do;
end do;
time()-t;

Loop 5000x5000

a:=1;
t:=time():
for x from 1 by 1 to 5000 do
    for y from 1 by 1 to 5000 do
        a:=a+x+y:
    end do;
end do;
time()-t;

3800x3800 normal distributed random matrix^1000

a:=evalf(evalm(RandomMatrix(3800,3800)/1000)):
time(map(x->x^1000,evalm((a+1))));

3000000 values sorted ascending

L := [seq(rand(),i=1..3000000)]:
time(sort(L));

Analysis
FFT over 1048576 values

m:=20:
x:=RandomVector[row](2^m,generator=-1.0..1.0):
y:=Vector[row](2^m,shape=zero):
time(evalhf(FFT(m,x,y)));

Triple integration

t:=time():
evalf(Int(Int(Int(sin(x^2)+sin(y^2)+sin(z^2),x=-Pi..Pi),y=-Pi..Pi),z=-Pi..Pi,method=_d01gbc),5):
evalf(Int(Int(Int(2^sin(x)+3^cos(y)+4^sin(z),x=-Pi..Pi),y=-Pi..Pi),z=-Pi..Pi,method=_d01gbc),5):
evalf(Int(Int(Int(2^log(x^2+y^2+z^2),x=-Pi..Pi),y=-Pi..Pi),z=-Pi..Pi,method=_d01gbc),5):
evalf(Int(Int(Int(sin(x^2-y^2-z^2),x=-Pi..Pi),y=-Pi..Pi),z=-Pi..Pi,method=_d01gbc),5):
evalf(Int(Int(Int(cos(x^2+y^2+z^2),x=-Pi..Pi),y=-Pi..Pi),z=-Pi..Pi,method=_d01gbc),5):
evalf(Int(Int(Int(sin(x^2+y^2+z^2),x=-Pi..Pi),y=-Pi..Pi),z=-Pi..Pi,method=_d01gbc),5):
time()-t;

Algebra
Determinant of a 1000x1000 random matrix

a:=RandomMatrix(1000,1000)/100:
a[1,1]:=evalf(a[1,1]):
time(Determinant(a));

Inverse of a 1000x1000 uniform distr. random matrix
```

```
a:=RandomMatrix(1000,1000)/100:
a[1,1]:=evalf(a[1,1]):
time(MatrixInverse(a));
```

Eigenvalues of a normal distr. 600x600 randommatrix

```
a:=RandomMatrix(600,600)/100:
a[1,1]:=evalf(a[1,1]):
time(Eigenvalues(a));
```

Cholesky decomposition of a 1000x1000-matrix

```
S:=RandomMatrix(1000,1000)/100:
A := evalf(Transpose(S).S):
time(LUDecomposition(A,method='Cholesky'));
```

1000x1000 cross-product matrix

```
S:=RandomMatrix(1000,1000)/100:
S[1,1]:=evalf(S[1,1]):
time(Transpose(S).S);
```

Number theory
Calculation of 1000000 fibonacci numbers

```
with(combinat, fibonacci):
frnd:=rand(1..1000):
A := [seq(frnd(),i=1..1000000)]:
time(evalm(fibonacci(A)));
```

Stochastic-statistic
Principal component analysis over a 500x500 matrix

Gamma function over a 1500x1500 matrix

```
a:=evalm(RandomMatrix(1500,1500)/100+0.01):
b:=evalm(abs(a)):
time(evalf(map(b->GAMMA(b),evalm(b)))):
```

Gaussian error function over a 1500x1500 matrix

```
A:=evalf(RandomMatrix(1500,density=0.5,generator=-1.0..1.0)):
time(evalf(map(A->erf(A),evalm(A))));
```

Linear regression over a 1000x1000 matrix

```
A:=evalf(RandomMatrix(1000,1000)):
B:=evalf(RandomMatrix(1000,1)):
time(LeastSquares(A,B));
```

Mathematica Benchmark program:

```
Comparison benchmarktest for number cruncher testreport
(Version 3)
created 3rd of March 2002 by Stefan Steinhaus

Misc. operation

IO Test - Reading of data from ASCII file

In[1]:=
fn="E:\\mathematik\\ncrunch4\\currency1.txt";

In[2]:=
MyFile[fn_]:=Module[{rl,f,r,r2},rl=ReadList[fn,Word,RecordLists\[Rule]True];
   f=First[rl];
   r=Rest[rl];
   r2=r/.{"."\[Rule]"",x_String\[RuleDelayed]ToExpression[x]};
   Join[{f},r2]]

In[3]:=
Timing[data=MyFile[fn];]

IO Test - Reading of data from ODBC data source

Needs["DatabaseAccess`"]
Timing[Module[{},dbselect = OpenDataSource["currency"];
   datmat=DataSourceEvaluate[dbselect,
      SQLSelect[
       SQLTable["[currency]"], {SQLColumn["Exchange_Rate_Year"],
         SQLColumn["Exchange_Rate_Month"], SQLColumn["Exchange_Rate_Day"],
         SQLColumn["AUD"],SQLColumn["ATS"],SQLColumn["BEF"],
         SQLColumn["BRL"],SQLColumn["CAD"],SQLColumn["CNY"],
         SQLColumn["DKK"],SQLColumn["EUR"],SQLColumn["FIM"],
         SQLColumn["FRF"],SQLColumn["DEM"],SQLColumn["GRD"],
         SQLColumn["HKD"],SQLColumn["INR"],SQLColumn["IEP"],
         SQLColumn["ITL"],SQLColumn["JPY"],SQLColumn["KRW"],
         SQLColumn["MYR"],SQLColumn["MXP"],SQLColumn["NLG"],
         SQLColumn["NOK"],SQLColumn["NZD"],SQLColumn["PTE"],
         SQLColumn["SEK"],SQLColumn["ZAR"],SQLColumn["SGD"],
         SQLColumn["SRI"],SQLColumn["ESP"],SQLColumn["CHF"],
         SQLColumn["TWD"],SQLColumn["THB"],SQLColumn["GBP"],
         SQLColumn["VEB"]}]];
   colname=DataSourceColumns[dbselect, TableName -> "currency"];
   CloseDataSource[dbselect];]]
Remove[DatabaseAccess]

Data Extraction Test - Extraction of data and descriptive statistics

<<Statistics`DescriptiveStatistics`
LimitsArea={1,261,522,784,1045,1305,1565,1827,2088,2349,2610,2871,3131,3159};
Timing[Do[
   Do[YearData=Take[datmat,{LimitsArea[[j]],LimitsArea[[j+1]]-1},{1,37}];
     YDcols=Transpose[YearData];
     MinYear=Map[Min,YDcols];
     MaxYear=Map[Max,YDcols];
     MeanYear=Map[Mean,YDcols];
     GainLossYear=
       100-(First[YearData]+0.00000001)/(Last[YearData]+0.00000001)*100,{j,1,
       13}],{k,1,100}]]

Loop 5000x5000

a=1;
Timing[Do[Do[a=a+i+j,{i,1,5000,1}],{j,1,5000,1}]]

3800x3800 random matrix ^ 1000

a = Table[Random[Real]\/10, {3800}, {3800}];

Timing[(a + 1)^1000;]

3000000 values sorted in ascending order

a=Table[Random[Real],{3000000}];
Timing[Sort[a];]

Analysis

FFT over 1048576 random values

a=Table[Random[Real],{1048576}];
Timing[Fourier[a];]

Triple integration

t0=SessionTime[];
NIntegrate[Sin[x^2]+Sin[y^2]+Sin[z^2],{x,-Pi,Pi},{y,-Pi,Pi},{z,-Pi,Pi}];
NIntegrate[2^Sin[x]+3^Cos[y]+4^Sin[z],{x,-Pi,Pi},{y,-Pi,Pi},{z,-Pi,Pi}];
NIntegrate[2^Log[x^2+y^2+z^2],{x,-Pi,Pi},{y,-Pi,Pi},{z,-Pi,Pi}];
NIntegrate[Sin[x^2-y^2-z^2],{x,-Pi,Pi},{y,-Pi,Pi},{z,-Pi,Pi}];
NIntegrate[Cos[x^2+y^2+z^2],{x,-Pi,Pi},{y,-Pi,Pi},{z,-Pi,Pi}];
NIntegrate[Sin[x^2+y^2+z^2],{x,-Pi,Pi},{y,-Pi,Pi},{z,-Pi,Pi}];
```

```
SessionTime[]-t0
```

Algebra

Determinant of a 1000x1000 random matrix

```
a=Table[Random[Real],{1000},{1000}];
Timing[Det[a];]
```

Inverse of a 1000x1000 random matrix

```
a=Table[Random[Real],{1000},{1000}];
Timing[Inverse[a];]
```

Eigenvalues of a 600x600 randommatrix

```
a=Table[Random[Real],{600},{600}];
Timing[Eigenvalues[a];]
```

Cholesky decomposition of a 1000x1000-matrix

```
<<"LinearAlgebra`Cholesky`"
a=Table[Random[Real],{1000},{1000}];
a += Transpose[a] + 100. IdentityMatrix[1000];
Timing[CholeskyDecomposition[a];]
Remove[LinearAlgebra`Cholesky]
```

1000x1000 cross-product matrix

```
Timing[Transpose[a].a;]
```

Number theory
Calculation of 1000000 fibonacci numbers

```
a=Table[Random[Integer,{100,1000}],{1000000}];
Timing[Fibonacci[a];]
```

Stochastic & statistic
Principal component analysis over a 500x500 matrix

```
<<Statistics`MultiDescriptiveStatistics`
a=Table[Random[Real],{500},{500}];
Timing[PrincipalComponents[a];]
Remove[Statistics`MultiDescriptiveStatistics]
```

Gamma function over a 1500x1500 matrix

```
a=Table[Random[Real],{1500},{1500}];
Timing[Gamma[a];]
```

Gaussian error function over a 1500x1500 matrix

```
a=Table[Random[Real],{1500},{1500}];
Timing[Erf[a];]
```

Linear regression over a 1000x1000 matrix

```
a=Table[Random[Real],{1000},{1000}];

Timing[Table[Fit[a[[i]],{1,x},x],{i,1,1000}];]
```

Matlab Benchmark program:

```
%************************************************************************
%* Matlab Benchmark program version 3.0                                 *
%* 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;
  importdata('e:\\mathematik\\ncrunch4\\currency2.txt',' ');
  timing=toc;
  result=result+timing;
end;
result=result/runs;
disp(['IO Reading test (ASCII file)_____ : ' num2str(result) ' sec.'])

result=0; runs=1;
for i=1:runs;
  tic;
  conn=database('currency','','');
  curs=exec(conn,'select
Exchange_Rate_Year,Exchange_Rate_Month,Exchange_Rate_Day,AUD,ATS,BEF,BRL,CAD,CNY,DK
K,EUR,FIM,FRF,DEM,GRD,HKD,INR,IEP,ITL,JPY,KRW,MYR,MXP,NLG,NOK,NZD,PTE,SEK,ZAR,
SGD,SRI,ESP,CHF,TWD,THB,GBP,VEB from [currency];');
  curs=fetch(curs,4000);
  colnames=columnnames(curs);
  close(curs);
  close(conn);
  timing=toc;
  result=result+timing;
end;
result=result/runs;
disp(['IO Reading test (ODBC)_____ : ' num2str(result) ' sec.'])

result=0;
Limits_Area=[1,261,522,784,1045,1305,1565,1827,2088,2349,2610,2871,3131,3158];
datmat=zeros(size(curs.data));
datmat(:)=[curs.data{:}];
for i=1:runs;
  tic;
  for k=1:100;
    for j=1:13;
      Year_Data=datmat(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;
  timing=toc;
  result=result+timing;
end;
result=result/runs;
disp(['Extraction of submatrices & descriptive statistics___ : ' num2str(result) ' sec.'])

result=0; a=1;
for i=1:runs;
  tic;
  for x=1:5000;
    for y=1:5000;
      a=a+x+y;
    end;
  end;
  timing=toc;
  result=result+timing;
end;
result=result/runs;
disp(['Loop testing_____ : ' num2str(result) ' sec.'])
result=0; a=0;
for i=1:runs
  b=abs(randn(3800,3800)/2);
  tic;
  a=b.^1000;
  timing=toc;
  result=result+timing;
end
result=result/runs;
disp(['3800x3800 normal distributed random matrix^1000_____ : ' num2str(result) ' sec.'])
clear a; clear b; result=0;
```

```
for i=1:runs
  a=randn(3000000,1);
  tic; b=sort(a); timing=toc;
  result=result+timing;
end
result=result/runs;
disp(['3000000 values sorted ascending_____ : ' num2str(result) ' sec.'])

%* Analysis *

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

clear a; clear b; result=0;
for i=1:runs
  tic;
  Q = triplequad('integrnd1',-pi,pi,-pi,pi,-pi,pi,1.0e-2);
  Q = triplequad('integrnd2',-pi,pi,-pi,pi,-pi,pi,1.0e-2);
  Q = triplequad('integrnd3',-pi,pi,-pi,pi,-pi,pi,1.0e-2);
  Q = triplequad('integrnd4',-pi,pi,-pi,pi,-pi,pi,1.0e-2);
  Q = triplequad('integrnd5',-pi,pi,-pi,pi,-pi,pi,1.0e-2);
  Q = triplequad('integrnd6',-pi,pi,-pi,pi,-pi,pi,1.0e-2);
  timing=toc;
  result=result+timing;
end
result=result/runs;
disp(['Triple integration over 6 functions_____ : ' num2str(result) ' sec.'])

%* Algebra *

clear a; clear b; result=0;
for i=1:runs
  a=rand(1000,1000);
  tic; b=det(a); timing=toc;
  result=result+timing;
end
result=result/runs;
disp(['Determinant of a 1000x1000 random matrix_____ : ' num2str(result) ' sec.'])
clear a; clear b; result=0;
for i=1:runs
  a=rand(1000,1000);
  tic; b=inv(a); timing=toc;
  result=result+timing;
end
result=result/runs;
disp(['Inverse of a 1000x1000 uniform distr. random matrix__ : ' num2str(result) ' sec.'])
clear a; clear b; result=0;
for i=1:runs
  a=randn(600,600);
  tic; c=eig(a); timing=toc;
  result=result+timing;
end
result=result/runs;
disp(['Eigenval. of a normal distr. 600x600 randommatrix____ : ' num2str(result) ' sec.'])
clear a; clear b; result=0;
for i=1:runs
  a=randn(1000,1000);
  a=a'*a;
  tic; b=chol(a); timing=toc;
  result=result+timing;
end
result=result/runs;
disp(['Cholesky decomposition of a 1000x1000-matrix_____ : ' num2str(result) ' sec.'])
clear a; clear b; result=0;
for i=1:runs
  a=randn(1000,1000);
  tic; b=a'*a; timing=toc;
  result=result+timing;
end
result=result/runs;
disp(['1000x1000 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(1000000,1));
  tic;
  b=(phi.^a-(-phi).^(-a))/sqrt(5);
  timing=toc;
  result=result+timing;
end
result=result/runs;
disp(['Calculation of 1000000 fibonacci numbers_____ : ' num2str(result) ' sec.'])

%* Stochastic-statistic *
```

```
result=0; a=0; b=0;
for i=1:runs;
    a=randn(500,500);
    tic;
       b=princomp(a);
    timing=toc;
    result=result+timing;
end;
result=result/runs;
disp(['Calc. of the principal components of a 500x500 matrix : ' num2str(result) ' sec.'])
clear a; clear b; result=0;
for i=1:runs
   a=randn(1500,1500)^2;
   tic; b=gamma(a); timing=toc;
   result=result+timing;
end
result=result/runs;
disp(['Gamma function over a 1500x1500 matrix_________________ : ' num2str(result) ' sec.'])
clear a; clear b; result=0;
for i=1:runs
   a=randn(1500,1500)^2;
   tic; b=erf(a); timing=toc;
   result=result+timing;
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
   a=randn(1000,1000); b=1:1000;
   tic; b=a\b'; timing=toc;
   result=result+timing;
end
result=result/runs;
disp(['Linear regression over a 1000x1000 matrix_____________ : ' num2str(result) ' sec.'])
```

MuPAD Benchmark program:

```
History:=3:
M:=Dom::Matrix():
Seed:=1:
randn:= stats::uniformRandom(-1,1):
randu:= float@random(0..10^10)/10^10:
```

IO Test

```
Zeit:=time():
Data:=import::readdata("E:\\Mathematik\\Ncrunch4\\Currency2.txt"):
Header:=data[1]:
delete data[1]:
(time()-zeit)*sec/10.0^3;
```

Extraction of submatrices & descriptive statistics

```
LimitsArea:={1,261,522,784,1045,1305,1565,1827,2088,2349,2610,2871,3131,3158}:
T:=time():
for k from 1 to 100 do
    for j from 1 to 13 do
        YearData:= [op(data, LimitsArea[j]..LimitsArea[j+1]-1)]:
        YearData:= map(YearData, row -> [op(row, 5..38)]):
        Amountrows:=LimitsArea[j+1]-LimitsArea[j]:
        for i from 1 to 34 do
            ithcol:= map(YearData, row -> row[i]);
            MinYear:=min(op(ithcol)):
            MaxYear:=max(op(ithcol)):
            MeanYear:=stats::mean(op(ithcol)):
            GainLossYear:=100-(ithcol[1]+0.00000001)/(ithcol[amountrows]+0.00000001)*100:
        end_for:
    end_for:
End_for:
(time()-t)*sec/10.0^3;
```

Test of loops

```
Reset():
HISTORY:= 0:
a:=1:
Zeit:=time():
for j from 1 to 5000 do for i from 1 to 5000 do a:=a+i+j: end_for: end_for:
(time()-zeit)*sec/10.0^3;
```

3800 x 3800 random matrix^1000

```
N:=3800:
a:= array(1..n,1..n,[[randn()+1 $ j=1..n] $i=1..n]):
Time(map(a,_power,1000))*sec/10.0^3;
```

Sorting of 3000000 random values

```
Reset():
HISTORY:= 0:
r:= random(0..10^10):

n:= 3000000:
L:=[r() $ i=1..n]:
time((L:= sort(L)))*sec/10.0^3;
delete L;
```

FFT over 1048576 random values

```
Reset():
HISTORY:= 0:

r:= float@random(0..10^7):
m:= 20:
a:=[r() $ i = 1..2^m]:
time((a:= numeric::fft(a)))*sec/10.0^3;
```

Triple integration

```
Reset():
DIGITS:= 4:
Q:= numeric::quadrature:
PP:= -PI..PI:

Zeit:=time():
Q1 = Q(Q(Q(sin(x^2)+sin(y^2)+sin(z^2),x=PP),y=PP),z=PP):
Q2 = Q(Q(Q(2^sin(x)+3^cos(y)+4^sin(z),x=PP),y=PP),z=PP):
Q3 = Q(Q(Q(2^log(10,x^2+y^2+z^2+0.00001),x=PP),y=PP),z=PP):
Q4 = Q(Q(Q(sin(x^2-y^2-z^2),x=PP),y=PP),z=PP):
Q5 = Q(Q(Q(cos(x^2+y^2+z^2),x=PP),y=PP),z=PP):
Q6 = Q(Q(Q(sin(x^2+y^2+z^2),x=PP),y=PP),z=PP):
(time()-zeit)*sec/10.0^3;
```

Determinant of a 1000x1000 random matrix

```
n:= 1000:
a:= array(1..n,1..n,[[randn() $ j=1..n] $i=1..n]):
```

```
time(numeric::det(a))*sec/10.0^3;
```

Inverse of a 1000 x 1000 random matrix

```
n:= 1000:
a:= array(1..n,1..n,[[randn() $ j=1..n] $i=1..n]):

time(numeric::inverse(a))*sec/10.0^3;
```

Eigenvalues of a 600 x 600 random matrix

```
n:= 600:
a:= array(1..n,1..n,[[randn() $ j=1..n] $i=1..n]):
time(numeric::eigenvalues(a))*sec/10.0^3;
```

Cholesky decomposition of a 1000 x 1000 random matrix

```
A:= M(1000,1000,float(random(0..100)/1000)):
A:=(linalg::transpose(A)*A);
time(numeric::factorCholesky(A))*sec/10.0^3;
```

Creation of a 1000 x 1000 cross-product matrix

```
A:= M(1000,1000,float(random(0..100)/1000)):

time(linalg::transpose(A)*A)*sec/10.0^3;
```

Calculation of 1000000 fibonacci numbers

```
Reset():
HISTORY:= 0:
r:= random(100..1000):

n:= 1000000:
time(([numlib::fibonacci(r()) $ i=1..n]))*sec/10.0^3;
```

Principal component analysis over a 500x500 matrix

```
Zeit:=time():
n:= 500:
alldata:= array(1..n,1..n,[[randn() $ j=1..n] $i=1..n]):
for i from 1 to n do zeitreihe[i] := [float(alldata[i, j]) $ j=1..n]: end_for:
for i from 1 to n do
    zeitmittel[i]:= stats::mean(zeitreihe[i]):
    for j from 1 to n do zeitreihe[i][j] := zeitreihe[i][j] - zeitmittel[i]: end_for:
End_for:
Sigma:= array(1..n, 1..n):
C   := array(1..n, 1..n):
for i from 1 to n do
    for j from 1 to n do
        Sigma[i, j]:= stats::covariance(zeitreihe[i], zeitreihe[j]):
        C[i, j]:= stats::correlation(zeitreihe[i], zeitreihe[j]):
    end_for:
End_for:
[Eigenwerte, Eigenvektoren, residuen]:= numeric::eigenvectors(Sigma):
for j from 1 to n do Hauptvektor[j]:= array(1..n, [Eigenvektoren[i, j] $ i=1..n]): end_for:
(time()-zeit)*sec/10.0^3;
```

Gamma function on a 1500 x 1500 random matrix

```
n:= 1500:
a:= array(1..n,1..n,[[randu() $ j=1..n] $i=1..n]):

time(map(a,gamma))*sec/10.0^3;
```

Gaussian error function over a 1500 x 1500 random matrix

```
n:= 1500:
a:= array(1..n,1..n,[[randu() $ j=1..n] $i=1..n]):

time(map(a,erf))*sec/10.0^3;
```

Linear regression over a 1000 x 1000 random matrix

```
m:= 1000: n:= 1000:
A:= array(1..m,1..n,[[randn() $ j=1..n] $ i=1..m]):
b:= array(1..m, [randn() $ j=1..m]):

time(([solution, kernel, residues]:= numeric::leastSquares(A, b)))*sec/10.0^3;
```

O-Matrix Benchmark program:

```
# O-Matrix benchmark test version 3.0
# Stefan Steinhaus stefan@steinhaus-net.de
#

clear

# 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
    print
    print
    print "              !!! O-Matrix - Benchmark program !!!"
    print "              ====================================="
    print
    print
    print "Function / timing                                result (sec.)"
    print "-------------------------------------------------------------------------"

    # Misc. operation

    t0 = time
    file="E:\\Mathematik\\Ncrunch4\\Currency1.txt"
    header=read(file, "char",1)
    datmat=read(file, "double",3159,38)
    close(file)
    t1 = time - t0
    print "IO Test_____________________________________________:  ",t1

    t0 = time
    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 100 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
    print "Extraction of submatrices & descriptive statistics_: ",t1

    N = 5000
    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
    print "Loop test___________________________________________:  ",t1

    N = 3800
    x=(1-rand(N,N)/100)
    t0 = time
    x = x^1000.
    t1 = time - t0
    print "3800x3800 random matrix^1000_____________________:  ",t1

    N = 3000000
    y=rand(N,1)
    t0 = time
    X = sort(y)
    t1 = time - t0
    print "3000000 values sorted ascending__________________:  ", t1

    # Analysis

    N = 1048576
    y=complex(rand(N,1))
    t0 = time
    X = fft(y)
    t1 = time - t0
    print "FFT over 1048576 (2^20) values___________________:  ", t1

    # Algebra

    N = 1000
    x=rand(N,N)
    t0 = time
    y=det(x)
     t1 = time - t0
```

```
print "Determinant of a 1000x1000 random matrix___________: ",t1

N = 1000
y=rand(N,N)
t0 = time
X = inv(y)
t1 = time - t0
print "Inverse of 1000x1000 random matrix_____________________: ", t1

N = 600
y=snormal(N,N)
t0 = time
X = eigen(y)
t1 = time - t0
print "Eigenvalues of a 600x600 random matrix_____________: ",t1

N = 1000
y=abs(snormal(N,N))
y = y'*y
t0 = time
X = cholesky(y);
t1 = time - t0
print "Cholesky decompositon of 1000x1000 matrix__________: ", t1

N = 1000
y=rand(N,N)
t0 = time
X = y'*y
t1 = time - t0
print "1000x1000 cross-product matrix_____________________: ", t1

# Number theory

N = 1000000
phi = 1.6180339887498949
a=floor(1000*rand(N,1))
t0=time
b=(phi^a-(-phi)^(-a))/sqrt(5.)
t1=time - t0
print "Calculation of 1000000 fibonacci numbers___________: ", t1

# Stochastic-statistic

# 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
    y=rand(N,N)^2
    t0 = time
    X = exp(gammln(y))
    t1 = time-t0
    print "Gamma function over a 1500x1500 matrix_____________: ", t1

    N = 1500
    y=rand(N,N)^2
    t0 = time
    X = erf(y)
    t1 = time - t0
    print "Gaussian error function over a 1500x1500 matrix____: ", t1

    N = 1000
    a = snormal(N,N)
    b = 1::N
    t0 = time
    X = a\b;
    t1 = time - t0
    print "Linear regression over 1000x1000 matrix____________: ", t1
end
```

Ox Benchmark program:

```
/*--------------------------------------------------------------------------*/
/* Ox Benchmark program version 3.0 from Stefan Steinhaus                   */
/* EMAIL  : stefan@steinhaus-net.de                                         */
/*                                                                          */
/* This program is public domain. Feel free to copy it freely.              */
/*                                                                          */
/*--------------------------------------------------------------------------*/


#include <oxstd.h>

main()
{
    decl cRep = 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 ",cRep," runs.\n\n");
    print("Function / timing                                    result  (sec.)\n");
    print("-------------------------------------------------------------------------------\n");
    format("%8.3f");

    /* IO operation */

    for (i = secs = 0; i < cRep; ++i)
    {
        time = timer();
        decl file = fopen("Currency1.txt", "rb");
        fscan(file, "%z", &header);
        fscan(file, "%#m", 3159, 38, &datmat);
       fclose(file);
        secs += (timer() - time) / 100;
    }
    secs /= cRep;
    print("IO test_____________________________________________________________:  ", secs, "\n");

    /* Test of Extraction of submatrices and calculation of descriptive stats */
    /* Doing the whole process 100 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 < cRep; ++i)
    {
        time=timer();
      for (k=0; k<100; ++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 /= cRep;
    print("Extraction of submatrices & descriptive statistics_________ : ", secs, "\n");

    /* Misc. operation */

    x=0;y=0; a=1;
    for (i = secs = 0; i < cRep; ++i)
    {
        time = timer();
        for (x = 0; x < 5000; ++x)
        {
            for (y = 0; y < 5000; ++y)
            {
                a = a+x+y;
            }
        }
        secs += (timer() - time) / 100;
    }
    secs /= cRep;  delete a;  delete b;
    print("Loop test___________________________________________________________:  ", secs, "\n");

    for (i = secs = 0; i < cRep; ++i)
    {
        b=1+fabs(rann(3800,3800)/10);
        time = timer();
        a = b .^ 1000;
        secs += (timer() - time) / 100;
    }
    secs /= cRep;  delete a;  delete b;
    print("3800x3800 random matrix .^ 1000_____________________________: ",  secs, "\n");

    for (i = secs = 0; i < cRep; ++i)
```

```
{
      a = rann(1,3000000);
      time = timer();
      b = sortr(a);
      secs += (timer() - time) / 100;
}
secs /= cRep;  delete a;  delete b;
print("Sorting of 3,000,000 random values_____________________: ", secs, "\n");

/* Analysis */

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

/* Algebra */

for (i = secs = 0; i < cRep; i++)
{
      a = rann(1000,1000)/10;
      time = timer();
      b = determinant(a);
      secs += (timer() - time) / 100;
}
secs /= cRep;  delete a;  delete b;
print("Determinant of a 1000x1000 random matrix_____________: ", secs, "\n");

for (i = secs = 0; i < cRep; ++i)
{
      a = ranu(1000,1000);
      time = timer();
      b = invert(a);
      secs += (timer() - time) / 100;
}
secs /= cRep;  delete a;  delete b;
print("Inverse of a 1000x1000 random matrix_________________: ", secs, "\n");

for (i = secs = 0; i < cRep; ++i)
{
      a = rann(600,600);
      time = timer();
      eigen(a, &b);
      secs += (timer() - time) / 100;
}
secs /= cRep;  delete a;  delete b;
print("Eigenvalues of a 600x600 random matrix_______________: ", secs, "\n");

for (i = secs = 0; i < cRep; ++i)
{
      a = rann(1000,1000); a = a'a;
      time = timer();
      b = choleski(a);
      secs += (timer() - time) / 100;
}
secs /= cRep;  delete a;  delete b;
print("Choleski decomposition of a 1000x1000 random matrix_____: ", secs, "\n");

for (i = secs = 0; i < cRep; ++i)
{
      a = rann(1000,1000);
      time = timer();
      b = a'a;
      secs += (timer() - time) / 100;
}
secs /= cRep;  delete a;  delete b;
print("Creation of 1000x1000 cross-product matrix___________: ", secs, "\n");

/* Number theory */

phi = 1.6180339887498949;
for (i = secs = 0; i < cRep; i++)
{
      a = floor(1000*ranu(500000,1));
      time = timer();
      b = (phi^a-(-phi)^(-a))/sqrt(5);
      secs += (timer() - time) / 100;
}
secs /= cRep;  delete a;  delete b;
print("Calculation of 500000 fibonacci numbers_____________: ",secs, "\n");


/* Stochastic-statistic */

// n! = Gamma(n + 1)
for (i = secs = 0; i < cRep; ++i)
{
      b=rann(1500,1500).^2;
      time = timer();
```

```
        a = gammafact(b);
        secs += (timer() - time) / 100;
    }
    secs /= cRep;  delete a;  delete b;
    print("Gamma function on a 1500x1500 random matrix_____: ", secs, "\n");

    for (i = secs = 0; i < cRep; ++i)
    {
        a = rann(1500,1500).^2;
        time = timer();
        b = erf(a);
        secs += (timer() - time) / 100;
    }
    secs /= cRep;  delete a;  delete b;
    print("Gaussian error function over a 1500x1500 random matrix_____: ", secs, "\n");

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

SciLab Benchmark program:

```
function i=myf(xyz,nf)
  x=xyz(1); y=xyz(2); z=xyz(3);
  i=[sin(x^2)+sin(y^2)+sin(z^2)
  2^sin(x)+3^cos(y)+4^sin(z)
  2^log(x^2+y^2+z^2+0.00001)
  sin(x^2-y^2-z^2)
  cos(x^2+y^2+z^2)
  sin(x^2+y^2+z^2)]
endfunction

function [X,Y,Z]=cube2tetrahedrons()
//given a cube defined by
// -1<=x<=1
// -1<=y<=1
// -1<=z<=1
// this function gives a 12 tetrahedrons decomposition of the cube volume
// X(:,i),Y(:,i),Z(:,i) are the coordinates of the ith  tetrahedron edges
//bottom tetra's
  X1=[0;-1; 1; 1];
  Y1=[0;-1;-1; 1];
  Z1=[0;-1;-1;-1];

  X2=[0;-1;-1; 1];
  Y2=[0;-1; 1; 1];
  Z2=[0;-1;-1;-1];
//top tetra's
  X3= X1
  Y3= Y1
  Z3=-Z1

  X4= X2
  Y4= Y2
  Z4=-Z2
//right tetra's
  X5=[0; 1; 1; 1];
  Y5=[0;-1; 1;-1];
  Z5=[0;-1;-1; 1];

  X6=[0; 1; 1; 1];
  Y6=[0; 1; 1;-1];
  Z6=[0; 1;-1;-1];
//left tetra's
  X7=-X5
  Y7= Y5
  Z7= Z5

  X8=-X6
  Y8= Y6
  Z8= Z6
//front tetra's
  X9=[0;-1; 1; 1];
  Y9=[0;-1;-1;-1];
  Z9=[0;-1;-1; 1];

  X10=[0;-1;-1; 1];
  Y10=[0;-1;-1;-1];
  Z10=[0;-1; 1; 1];
//rear tetra's
  X11= X9
  Y11=-Y9
  Z11= Z9

  X12= X10
  Y12=-Y10
  Z12= Z10

  X=[X1 X2 X3 X4 X5 X6 X7 X8 X9 X10 X11 X12]
  Y=[Y1 Y2 Y3 Y4 Y5 Y6 Y7 Y8 Y9 Y10 Y11 Y12]
  Z=[Z1 Z2 Z3 Z4 Z5 Z6 Z7 Z8 Z9 Z10 Z11 Z12]
endfunction



stacksize(80000000)

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 - Reading of data from file (either ODBC or flat file) *

result = 0;
a = 0;
b = 0;
runs = 3;
for i=1:runs
  timer();
```

```
  [fd,err]=mopen('E:\Mathematik\Ncrunch4\currency2.txt','r',1);
  header=mgetstr(154,fd);
  datmat=fscanfMat('E:\Mathematik\Ncrunch4\currency2.txt');
  zeit=timer();
  result=result+zeit;
end
result=result/runs;
disp('IO test_____________________________________________:  '+string(result)+' sec.');

//* Test of Extraction of submatrices and calculation of descriptive stats */
//* Doing the whole process 100 times do get higher timings                */

result=0;
LimitsArea={1,261,522,784,1045,1305,1565,1827,2088,2349,2610,2871,3131,3158};
for i=1:runs
  timer();
  for k=1:100
    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=timer();
  result=result+zeit;
end
result=result/runs;
disp('Extraction of submatrices & descriptive statistics___: "+string(result)+' sec.');


//* Misc. operation *

result = 0;
for i = 1:runs
  a=1;
  timer();
  for x = 1:5000
    for y = 1:5000
      a=a+x+y;
    end
  end
  zeit = timer();
  result = result+zeit;
end
result = result/runs;
disp('Loop test___________________________________________:  '+string(result)+' sec.');
clear('a');
clear('b');
for i = 1:runs
  b = abs(rand(3800,3800,'n')/2);
  timer();
  a = b.^1000;
  zeit = timer();
  result = result+zeit;
end
result = result/runs;
disp('3800x3800 normal distributed random matrix^1000_____: '+string(result)+' sec.');
clear('a');
clear('b');
result = 0;
for i = 1:runs
  a = rand(3000000,1,'n');
  timer();
  b = -sort(-a);
  zeit = timer();
  result = result+zeit;
end
result = result/runs;
disp('3000000 values sorted ascending_____________________:  '+string(result)+' sec.');

//* Analysis *

clear('a');
clear('b');
result = 0;
for i = 1:runs
  a = rand(1048576,1,'n');
  timer();
  b = fft(a,-1);
  zeit = timer();
  result = result+zeit;
end
result = result/runs;
disp('FFT over 1048576 values_____________________________:  '+string(result)+' sec.');

clear('a');
result=0;
intresult=0;
x=[-%pi,%pi,%pi;%pi,%pi,-%pi;-%pi,-%pi,-%pi;-%pi,-%pi,-%pi];
y=[-%pi,-%pi,-%pi;-%pi,-%pi,%pi;%pi,-%pi,-%pi;-%pi,%pi,%pi];
z=[-%pi,-%pi,%pi;-%pi,%pi,%pi;-%pi,%pi,%pi;%pi,-%pi,-%pi];
deff('v1=f1(xyz,numfun)','v1=sin(xyz"*xyz)')
```

```
for i=1:runs
  timer();
  [X,Y,Z]=cube2tetrahedrons()
  [intresult,err]=int3d(X*%pi,Y*%pi,Z*%pi,myf,6,[0,1000000,1.d-1,1.d-4])
  zeit=timer();
  result=result+zeit;
end
result=result/runs;
disp('Triple integration_____________________________________  : '+string(result)+' sec.')

//* Algebra *

clear('a');
clear('b');
result = 0;
for i = 1:runs
 a = rand(1000,1000,'u');
 timer();
 b = det(a);
 zeit = timer();
 result = result+zeit;
end
result = result/runs;
disp('Determinant of a 1000x1000 random matrix_____________: '+string(result)+' sec.');
clear('a');
clear('b');
result = 0;
for i = 1:runs
 a = rand(1000,1000,'u');
 timer();
 b = inv(a);
 zeit = timer();
 result = result+zeit;
end
result = result/runs;
disp('Inverse of a 1000x1000 uniform distr. random matrix__: '+string(result)+' sec.');
clear('a');
clear('b');
result = 0;
for i = 1:runs
 a = rand(600,600,'n');
 timer();
 c = spec(a);
 zeit = timer();
 result = result+zeit;
end
result = result/runs;
disp('Eigenval. of a normal distr. 600x600 randommatrix____: '+string(result)+' sec.');
clear('a');
clear('b');
result = 0;
for i = 1:runs
 a = rand(1000,1000,'n');
 a = a'*a;
 timer();
 b = chol(a);
 zeit = timer();
 result = result+zeit;
end
result = result/runs;
disp('Cholesky decomposition of a 1000x1000-matrix_________: '+string(result)+' sec.');
clear('a');
clear('b');
result = 0;
for i = 1:runs
 a = rand(1000,1000,'n');
 timer();
 b = a'*a;
 zeit = timer();
 result = result+zeit;
end
result = result/runs;
disp('1000x1000 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(1000000,1,'u'));
 timer();
 b = (phi.^a-(-phi).^(-a))/sqrt(5);
 zeit = timer();
 result = result+zeit;
end
result = result/runs;
disp('Calculation of 1000000 fibonacci numbers_____________: '+string(result)+' sec.');

//* Stochastic-statistic *

clear('a');
clear('b');
```

```
result = 0;
for i = 1:runs
  a = rand(1500,1500,'n')^2;
  timer();
  b = gamma(a(:));
  zeit = timer();
  result = result+zeit;
end
result = result/runs;
disp('Gamma function over a 1500x1500 matrix_________________: '+string(result)+' sec.');
clear('a');
clear('b');
result = 0;
for i = 1:runs
  a = rand(1500,1500,'n')^2;
  timer();
  b = erf(a(:));
  zeit = timer();
  result = result+zeit;
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
  a = rand(1000,1000,'n');
  b = 1:1000;
  timer();
  b = a\b';
  zeit = timer();
  result = result+zeit;
end
result = result/runs;
disp('Linear regression over a 1000x1000 matrix_____________: '+string(result)+' sec.');
```

S-Plus Benchmark program:

```
options(echo=F)
options(object.size=120000000)
print("!!! S-Plus - Benchmarkprogram !!!")
print("================================")
print

## Test of IO function - Reading of data from an ASCII file ##

print("IO test - ASCII (Sec.)                    : ")
zeit<-proc.time()
datmat<-read.table('e:\\mathematik\\ncrunch4\\currency1.txt',header=T)
print((proc.time()-zeit)[1])

## Test of IO function - Reading of data via ODBC ##

print("IO test - ODBC (Sec.)                     : ")
zeit<-proc.time()
myConnection <- "DSN=currency"
header<-importData(,type="ODBC",startRow=1,endRow=1,odbcConnection=myConnection,
odbcSqlQuery="select TOP 1 * from [currency];")[0,1:39]
datmat<-importData(,type="ODBC",odbcConnection=myConnection, odbcSqlQuery="select * from
[currency];")
print((proc.time()-zeit)[1])

## Test of Extraction of submatrices and calculation of descriptive stats ##
## Doing the whole process 100 times do get higher timings              ##

print("Extraction of submatrices + descriptive statistics (Sec.) : ")
zeit<-proc.time()
LimitsArea<-c(1,261,522,784,1045,1305,1565,1827,2088,2349,2610,2871,3131,3158)
for (k in 1:100)
{
   for (j in 1:13)
      {
          for (i in 1:33)
          {
               YearData=datmat[LimitsArea[j]:LimitsArea[j+1]-1,6:39]
               MinYear=min(YearData[,i],na.rm =T)
               MaxYear=max(YearData[,i],na.rm =T)
               MeanYear=mean(YearData[,i],na.rm =T)

GainLossYear=100-(YearData[1,i]+0.00000001)/(YearData[LimitsArea[j+1]-LimitsArea[j],i]+0.0000000
1)*100
          }
      }
}
print((proc.time()-zeit)[1])

## Test of 2 loops 5000x5000 ##

print("Test of loops (Sec.)                      : ")
a<-1
zeit<-proc.time()
for(i in 1:5000)
{
     for(j in 1:5000)
     {
          a<-a+i+j
     }
}
print((proc.time()-zeit)[1])

## Misc. operation ##

print("3800x3800 random matrix^1000 (sec.) : ")
x<-matrix(runif(14440000),3800,3800)
print(dos.time(x^1000))
print("3000000 values sorted ascending (sec.) : ")
x<-runif(3000000)
print(dos.time(sort(x)))

## Analysis ##

print("FFT over 1048576 values (sec.) :")
x<-runif(1048576)
print(dos.time(fft(x)))
print("Triple integration (sec.) : ")

## Algebra ##


print("Determinant of a 1000x1000 random matrix (sec.) : ")
x<-matrix(runif(1000000),1000,1000)
print(dos.time(det(x)))
print("Inverse of a 1000x1000 uniform distr. random matrix (sec.) :")
x<-matrix(runif(1000000),1000,1000)
print(dos.time(solve(x)))
print("Eigenval. of a normal distr. 600x600 randommatrix (sec.) :")
x<-matrix(runif(360000),600,600)
print(dos.time(eigen.default(x, only.values = T)$values))
print("Cholesky decomposition of a 1000x1000-matrix (sec.) :")
```

```
x<-matrix(runif(1000000),1000,1000)
x<-crossprod(x,x)
print(dos.time(chol(x)))
print("1000x1000 cross-product matrix (sec.) :")
x<-matrix(runif(1000000),1000,1000)
print(dos.time(crossprod(x,x)))

## Number theory ##

print("Calculation of 500000 fibonacci numbers (sec.) : ")
phi<-1.6180339887498949
x<-floor(1000*matrix(runif(500000),500000,1))
print(dos.time((phi^x-(-phi)^(-x))/sqrt(5)))

## Stochastic-statistic ##

print("Principal Comp. An. over a 500x500 matrix(sec.):")
x<-matrix(runif(250000),500,500)
print(dos.time(princomp(x)))
print("Gamma function over a 1500x1500 matrix (sec.) :")
x<-matrix(runif(2250000),1500,1500)
print(dos.time(gamma(x)))
print("Gaussian error function over a 1500x1500 matrix (sec.) :")
print("Linear regression over a 1000x1000 matrix (sec.) :")
x<-matrix(runif(1000000),1000,1000)
y<-runif(1000)
print(dos.time(lsfit(x,y)))
```

Appendix B : References

The following persons helped to transfer the benchmark test and support information about their mathematical programs:

- **GAUSS:** Jörg Hohlfeld, Additive GmbH (Germany)
- **Maple:** Thomas Richard, Scientific Computers GmbH (Germany)
- **Mathematica:** Robert Knapp, Wolfram Research Inc. (USA)
- **Mathematica:** Ben Wilson, Wolfram Research Inc. (USA)
- **Mathematica:** Jörg Hohlfeld, Additive GmbH (Germany)
- **Matlab:** Cleve Moler, The Mathworks Inc. (USA)
- **MuPAD:** Oliver Kluge, SciFace GmbH / University of Paderborn (Germany)
- **MuPAD:** Walter Oevel, SciFace GmbH / University of Paderborn (Germany)
- **MuPAD:** Christopher Creutzig, SciFace GmbH / University of Paderborn (Germany)
- **O-Matrix:** Brad Bell, Harmonic Software Inc. (USA)
- **O-Matrix:** Beau Paisley, Harmonic Software Inc. (USA)
- **Ox:** Jurgen Doornik, Nuffield College (England)
- **Ox:** Ana Timberlake, Timberlake Consulting (England)
- **Scilab:** Dr. Scilab, INRIA (France)
- **Scilab:** Serge Steer, INRIA (France)
- **S-Plus:** Reinhard Sy, Insightful GmbH (Germany)