Comparison of mathematical programs for data analysis

(Edition 3)

by Stefan Steinhaus
(stst@informatik.uni-frankfurt.de)

University of Frankfurt / Germany
April 1999

Location: http://www.informatik.uni-frankfurt.de/~stst/ncrunch/

Contents

1997-1999 by Stefan Steinhaus, stst@informatik.uni-frankfurt.de

1. Introduction

This comparison should give an overview about the functionality, the availability for different operating systems and the speed of mathematical programs for analyzing huge or very huge data sets in mathematical, statistical or graphical ways. The focus of this test is therefore set to mathematical functions which are mainly used in economics, financial analysis, biology, chemistry, physics and some other subjects where the numerical analyze of data is very important.

In the first part of this test report you can find an overview over the mathematical functionality divided into different subsections, in the second part you can find an overview about the graphical functionality, in the third part you can find an overview about the programming environment, in the fourth part you can find a list of available export/import options, in the fifth part there is a list of operating systems for which the programs are available and in the sixth part you can find a speed comparison on these systems. In the seventh part there will be a final summary to relate all the different parts of the test report. The relation between these six parts will be 38 : 10 : 9 : 5 : 2 : 36 but for those who want to have their own weighting I will describe in detail the calculation for the summary so that you can easily compute the result with your own weighting.

The following symbols have been used all over the whole testreport within the comparison tables:

+	-	Function is implemented in the Program
m	-	Function is supported by an additional module
-	-	Function is not implemented

Actually the following programs have been tested:

- ***GAUSS*** from Aptech Systems Inc.
  (www.aptech.com)
- ***Macsyma*** from Macsyma Inc.
  (www.macsyma.com)
- ***Maple*** from Waterloo Maple Software Inc.
  (www.maplesoft.com)
- ***Mathematica*** from Wolfram Research Inc.
  (www.wolfram.com)
- ***Matlab*** from The Mathworks Inc.
  (www.mathwork.com)
- ***MuPAD*** from the University of Paderborn / SciFace GmbH
  (www.sciface.com)
- ***O-Matrix*** from Harmonic Software
  (www.omatrix.com)
- ***Ox*** from Jurgen Doornik
  (www.nuff.ox.ac.uk/Users/doornik)
- ***R-Lab*** from Ian Searle
  (www.eskimo.com/~ians/rlab.html)
- ***Scilab*** from Dr. Scilab
  (www-rocq.inria.fr/scilab/)
- ***S-Plus*** from MathSoft Inc.
  (www.mathsoft.com)

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 table should give an overview about the functionality for analyzing data in numerical ways and should mark out which function is 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" and „Other mathematics".

2.1. Standard mathematics

Functions (Version)	**GAUSS**	**Macsyma**	**Maple**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**R-Lab**	**Scilab**	**S-Plus**
	(3.2.38)	**(2.4)**	**(V5.1)**	**(4.0)**	**(5.3)**	**(1.4.1)**	**(4.0)**	**(2.1)**	**(2.0.12)**	**(2.4.1)**	**(V4.5)**
Square root	+	+	+	+	+	+	+	+	+	+	+
Log / Ln / Exp	+ / + / +	+ / + / +	+ / + / +	+ / + / +	+ / + / +	- / + / +	+ / + / +	+ / + / +	+ / + / +	+ / + / +	+ / + / +
Trig. functions	+	+	+	+	+	+	+	+	+	+	+
Hyperbolic trig. function	+	+	+	+	+	+	+	+	-	+	+
Gamma function	+	+	+	+	+	+	-	+	-	+	+
Incomplete Gammafunc.	+	+	+	+	+	+	-	+	-	-	-
Poly gamma	-	+	+	+	-	+	-	+	-	-	-
Log-Gammafunc.	+	+	+	+	+	+	+	+	-	+	+
Beta function	+	+	+	+	+	+	+	+	-	-	-
BesselI	+	+	+	+	+	-	-	+	-	+	-
Bessel J	+	+	+	+	+	+	+	+	-	+	-
BesselK	-	+	+	+	+	-	-	+	-	+	-
Bessel Y	+	+	+	+	+	+	+	+	-	+	-
Implemented functions	86.67% (13/15)	100.00% (15/15)	100.00% (15/15)	100.00% (15/15)	93.33% (14/15)	80.00% (12/15)	66.67% (10/15)	100.00% (15/15)	33.33% (5/15)	80.00% (12/15)	53.33% (8/15)

2.2. Algebra

Functions (Version)	**GAUSS**	**Macsyma**	**Maple**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**R-Lab**	**Scilab**	**S-Plus**
	(3.2.38)	**(2.4)**	**(V5.1)**	**(4.0)**	**(5.3)**	**(1.4.1)**	**(4.0)**	**(2.1)**	**(2.0.12)**	**(2.4.1)**	**(V4.5)**
Eigenvalues											
Eigenvalues	+	+	+	+	+	+	+	+	+	+	+
Eigenvectors	+	+	+	+	+	+	+	+	+	+	+
Matrix constructors / analysis											
Band matrix	+	+	+	+	+	-	-	-	-	-	-
Condition number	+	+	-	+	+	-	+	-	+	+	-
Cross product matrix	+	-	-	-	-	-	-	-	-	-	+
Determinant	+	+	+	+	+	+	+	+	+	+	+
Identity matrix	+	+	+	+	+	+	+	+	+	+	-
Inverse matrix	+	+	+	+	+	+	+	+	+	+	+
Kronecker product	+	+	+	+	+	-	+	+	-	+	+
Norm	-	+	+	+	+	+	-	+	+	+	+
Orthogonalization (constr.)	+	+	+	+	+	+	-	-	-	+	-
Rank	+	+	+	-	+	+	+	+	+	+	+
Toeplitz matrix	+	+	+	m	+	-	-	+	+	+	-
Trace	+	+	+	+	+	+	+	+	+	+	-
Upper Hessenberg form	+	+	-	+	+	-	-	-	+	+	-
Decompositions											
Cholesky decomposition	+	+	+	+	+	+	+	+	+	+	+
Crout decomposition	+	-	-	-	-	-	-	-	-	-	-
LU decomposition	+	+	+	+	+	+	-	+	+	+	+
QR decomposition	+	+	+	+	+	+	+	+	+	+	+
Schur form of quadratic matrix	+	+	-	+	+	-	-	-	+	+	+
Singular value decomposition	+	+	+	+	+	+	+	+	+	+	+

Functions (Version)	**GAUSS**	**Macsyma**	**Maple**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**R-Lab**	**Scilab**	**S-Plus**
	(3.2.38)	**(2.4)**	**(V5.1)**	**(4.0)**	**(5.3)**	**(1.4.1)**	**(4.0)**	**(2.1)**	**(2.0.12)**	**(2.4.1)**	**(V4.5)**
Optimization											
Optimization - linear models (Unconstr. / Constr.)	m / m	+ / -	+ / -	+ / +	m / m	+ / +	+ / +	+ / -	- / -	+ / +	+ / +
Optimization - nonlinear models (Unconstr. / Constr.)	m / m	- / -	+ / -	+ / m	m / m	- / -	+ / +	- / -	- / -	+ / +	m / m
Optimization - quadratic models (QP) (Unconstr. / Constr.)	+ / +	- / -	+ / -	- / -	m / m	- / -	+ / +	- / -	- / -	+ / +	m / m
Equation solver											
Linear equation solver	+	+	+	+	+	+	+	+	+	+	+
Non-linear equation solver	m	+	+	+	m	+	+	-	+	+	+
Ordinary Differential Equation solver	m	+	+	+	+	+	+	-	+	+	+
Partial Differential Equation solver	-	m	+	+	m	-	-	-	-	-	-
Miscellaneous											
Moore-Penrose pseudo-inverse	+	+	-	+	+	-	-	+	-	+	+
Sparse matrices handling	+	+	-	+	+	-	-	-	m	+	-
Implemented functions	93.94% (31/33)	78.79% (26/33)	69.70% (23/33)	84.85% (28/33)	93.94% (31/33)	54.55% (18/33)	63.64% (21/33)	51.52% (17/33)	60.61% (20/33)	87.88% (29/33)	69.70% (23/33)

2.3. Analysis

Functions (Version)	**GAUSS**	**Macsyma**	**Maple**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**R-Lab**	**Scilab**	**S- Plus**
	(3.2.38)	**(2.4)**	**(V5.1)**	**(4.0)**	**(5.3)**	**(1.4.1)**	**(4.0)**	**(2.1)**	**(2.0.12)**	**(2.4.1)**	**(V4.5)**
Numerical integration (single / double / triple)	+ / + / +	+ / + / +	+ / + / +	+ / + / +	+ / + / -	+ / + / +	+ / + / -	m / - / -	- / - / -	+ / + / +	+ / - / -
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)	63.64% (7/11)	100.00% (11/11)	81.82% (9/11)	63.64% (7/11)	63.64% (7/11)	45.46% (5/11)	18.18% (2/11)	100.00% (11/11)	72.73% (8/11)

2.4. Numerical mathematics

Function (Version)	**GAUSS**	**Macsyma**	**Maple**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**R-Lab**	**Scilab**	**S-Plus**
	(3.2.38)	**(2.4)**	**(V5.1)**	**(4.0)**	**(5.3)**	**(1.4.1)**	**(4.0)**	**(2.1)**	**(2.0.12)**	**(2.4.1)**	**(V4.5)**
Interpolation											
B-Spline Interpolation	-	-	+	+	m	-	-	+	-	-	+
Classical Interpolation	-	+	+	+	+	+	+	-	-	+	+
k-Spline Interpolation	+	+	+	+	m	+	+	+	-	+	+
Pade Interpolation	m	+	+	+	-	+	+	-	-	-	-
Other functions											
Bisection	m	+	-	+	m	+	-	-	m	-	-
Newton method for finding roots	+	+	+	+	m	+	+	-	m	-	-
Runge Kutta method for solving ODE	m	+	+	+	m	+	+	-	m	+	-
Implemented functions	71.43% (5/7)	85.71% (6/7)	85.71% (6/7)	100.00% (7/7)	85.71% (6/7)	85.71% (6/7)	71.43% (5/7)	28.57% (2/7)	42.86% (3/7)	42.86% (3/7)	42.86% (3/7)

2.5. Descriptive statistic, stochastic and distribution functions

Functions (Version)	**GAUSS**	**Macsyma**	**Maple**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**R-Lab**	**Scilab**	**S-Plus**
	(3.2.38)	**(2.4)**	**(V5.1)**	**(4.0)**	**(5.3)**	**(1.4.1)**	**(4.0)**	**(2.1)**	**(2.0.12)**	**(2.4.1)**	**(V4.5)**
General functions											
Contingency tables	m	-	-	-	m	-	-	-	-	-	+
Correlation	+	+	+	+	+	+	+	+	+	+	+
Cross tabulation	m	-	-	-	m	-	-	-	-	-	+
Deviation	+	+	+	+	+	-	+	-	+	+	-
Kurtosis	-	+	+	+	m	-	-	-	-	-	-
Markov models	m	+	-	m	-	-	-	+	-	+	-
Mean	+	+	+	+	+	+	+	+	+	+	+
Median	+	+	+	+	+	+	+	+	+	+	+
Min / Max	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +	- / -
Mode	m	-	+	+	-	+	-	+	-	+	-
Skewness	-	+	+	+	m	-	-	-	-	-	-
Variance	+	+	+	+	+	+	+	+	-	+	+
Variance-covariance matrix	+	+	+	+	+	-	+	+	-	+	+
Distribution functions *(PDF / CDF / iCDF/ randomnumber)*											
Bernoulli	-/-/-/-	+/+/-/-	-/-/-/-	+/+/+/+	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
Beta	m/+/ m/+	+/+/+/-	+/+/+/+	+/+/+/+	m/m/m/m	-/-/-/-	-/-/-/-	+/+/+/+	-/-/-/+	-/+/-/+	+/+/-/+
Binomial	m/m/m/m	+/+/-/-	+/+/+/+	+/+/+/+	m/m/m/m	-/-/-/-	-/-/-/-	-/-/-/+	-/-/-/+	-/+/-/+	+/+/-/+
Cauchy	m/m/m/m	+/+/+/-	+/+/+/+	+/+/+/+	-/-/-/-	-/+/-/-	-/-/-/-	-/-/-/-	-/-/-/+	-/-/-/-	+/+/-/+
Chi-squared	m/+/m/m	+/+/+/-	+/+/+/+	+/+/+/+	m/m/m/m	-/-/-/-	-/-/-/-	+/+/+/+	-/-/-/-	-/+/-/+	+/+/-/+
Chi-squared (non-central)	m/+/m/m	-/-/-/-	-/-/-/-	+/+/+/+	m/m/m/m	-/-/-/-	-/-/-/-	+/-/-/-	-/-/-/+	-/+/-/-	-/-/-/-
Erlang	m/m/m/m	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
Exponential	m/m/m/m	+/+/+/-	+/+/+/+	+/+/+/+	m/m/m/m	-/-/-/-	-/-/-/-	-/-/-/+	-/-/-/+	-/-/-/+	+/+/-/+
Extreme value	-/-/-/-	-/-/-/-	-/-/-/-	+/+/+/+	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
F	+/+/m/m	+/+/+/-	+/+/+/+	+/+/+/+	m/m/m/m	-/-/-/-	+/-/-/-	+/+/+/+	-/-/-/+	-/+/-/+	-/-/-/-

Functions (Version)	**GAUSS**	**Macsyma**	**Maple**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**R-Lab**	**Scilab**	**S-Plus**
	(3.2.38)	**(2.4)**	**(V5.1)**	**(4.0)**	**(5.3)**	**(1.4.1)**	**(4.0)**	**(2.1)**	**(2.0.12)**	**(2.4.1)**	**(V4.5)**
F (non-central)	+/+/m/m	-/-/-/-	-/-/-/-	+/+/+/+	m/m/m/m	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/+	-/+/-/+	-/-/-/-
Gamma	m/+/m/+	+/+/+/-	+/+/+/+	+/+/+/+	m/m/m/m	-/-/-/-	-/-/-/-	+/+/+/+	-/-/-/+	-/+/-/+	+/+/-/+
Geometric	m/m/m/m	+/+/-/-	-/-/-/-	+/+/+/+	m/m/m/m	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	+/+/-/+
Gumbel	m/m/m/m	+/+/+/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
Half-normal	-/-/-/-	-/-/-/-	-/-/-/-	+/+/+/+	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
Hotelling T2	m/m/-/-	-/-/-/-	-/-/-/-	+/+/+/+	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
Hyper-exponential	m/m/m/m	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
Hypergeometric	m/m/m/m	-/-/-/-	+/+/+/+	+/+/+/+	m/m/m/m	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	+/+/-/+
Laplace	-/-/-/-	+/+/+/-	+/+/+/+	+/+/+/+	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
Logistic	m/m/m/m	+/+/+/-	-/-/-/-	+/+/+/+	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/+	-/-/-/-	-/-/-/-	+/+/-/+
Log-normal	+/+/m/m	+/+/+/-	+/+/+/+	+/+/+/+	m/m/m/m	-/-/-/-	-/-/-/-	-/-/-/+	-/-/-/-	-/-/-/-	+/+/-/+
Log-normal (multivariate)	+/+/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
Maxwell	-/-/-/-	+/+/+/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
Negative binomial	m/m/m/m	+/+/-/-	+/+/+/+	+/+/+/+	m/m/m/m	-/-/-/-	-/-/-/-	-/-/-/+	-/-/-/-	-/+/-/+	+/+/-/+
Normal	+/+/m/+	+/+/+/-	+/+/+/+	+/+/+/+	m/m/m/+	-/+/-/-	-/+/+/+	+/+/+/+	-/-/-/+	-/+/-/+	+/+/-/+
Normal (multivariate)	-/+/m/m	+/-/-/-	-/-/-/-	+/+/+/+	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/+	+/+/-/+
Pareto	m/m/m/m	+/+/+/-	-/-/-/-	+/+/+/+	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
Poisson	m/m/m/+	+/+/-/-	+/+/+/+	+/+/+/+	m/m/m/m	-/-/-/-	-/-/-/-	+/+/+/+	-/-/-/+	-/+/-/+	+/+/-/+
Rayleigh	-/-/-/-	+/+/-/-	-/-/-/-	+/+/+/+	m/m/m/m	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
Student’s t	+/+/m/m	+/+/+/-	+/+/+/+	+/+/+/+	m/m/m/m	-/+/-/-	+/+/+/-	+/+/+/+	-/-/-/-	-/+/-/-	+/+/-/+
Student’s t (non-central)	+/+/m/m	-/-/-/-	-/-/-/-	+/+/+/+	m/m/m/m	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
Student’s t (multivariate)	-/-/-/-	+/+/-/-	-/-/-/-	+/+/+/+	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-
Uniform	m/m/-/+	+/+/+/+	+/+/+/+	+/+/+/+	m/m/m/+	-/-/-/+	-/-/-/+	-/-/-/+	-/-/-/+	-/-/-/-	+/+/-/+
Von Mises	-/-/-/+	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	+/+/+/+	-/-/-/-	-/-/-/-	-/-/-/-
Weibull	m/m/m/m	+/+/+/-	+/+/+/+	+/+/+/+	m/m/m/m	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	+/+/-/+
Wishart	-/-/-/-	+/+/+/-	-/-/-/-	+/+/+/+	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/+	-/-/-/-	-/-/-/-	-/-/-/-
Implemented functions	72.79% (115/158)	49.37% (78/158)	47.47% (75/158)	83.54% (132/158)	55.70% (88/158)	6.96% (11/158)	10.13% (16/158)	31.01% (49/158)	10.76% (17/158)	20.25% (32/158)	36.71% (58/158)

2.6. Statistics

Functions (Version)	**GAUSS**	**Macsyma**	**Maple**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**R-Lab**	**Scilab**	**S-Plus**
	(3.2.38)	**(2.4)**	**(V5.1)**	**(4.0)**	**(5.3)**	**(1.4.1)**	**(4.0)**	**(2.1)**	**(2.0.12)**	**(2.4.1)**	**(V4.5)**
Regression models											
Linear	+	+	+	+	+	-	+	+	-	+	+
Loess	+	-	-	m	+	-	-	-	-	-	+
LOGIT	m	-	-	-	-	-	-	-	-	-	+
Nonlinear	m	+	+	+	m	-	+	-	-	+	+
Polynomial	m	+	+	+	+	-	+	-	-	+	+
PROBIT	m	-	-	-	-	-	-	m	-	-	+
PSN	m	-	-	-	-	-	-	-	-	-	-
Quantile	-	-	-	-	-	-	-	+	-	-	-
Tobit models	+	-	-	-	-	-	-	-	-	-	-
Test statistics											
Davidson J-Test	m	-	-	-	-	-	-	-	-	-	-
F-Test	+	+	-	+	-	-	+	-	-	-	+
J-Test	m	-	-	-	-	-	-	-	-	-	-
Goodness of fit test	m	-	-	-	-	-	-	+	-	-	+
Q-Test	m	-	-	-	-	-	-	-	-	-	-
Sign test	-	-	-	-	m	-	-	-	-	-	-
T-Test	+	+	-	+	m	-	+	-	-	-	+
Wald test	m	-	-	-	-	-	-	-	-	-	-
Wilcoxon rank sum test	-	-	-	-	m	-	-	-	-	-	+
Wilcoxon sign rank test	-	-	-	-	m	-	-	-	-	-	+
Z-Test	-	-	-	+	m	-	-	-	-	-	+
Filter / smoothing models											
Bandpass / Lowpass / Highpass / Multiband / Bandstop	m/m/-/-/-	-/-/-/-/-	-/-/-/-/-	m/m/m/-/m	m/m/m/ m/m	-/-/-/-/-	+/+/-/-/-	-/-/-/-/-	-/-/-/-/-	+/+/+/+/+	-/+/-/-/-
Battle-Lemarie	-	-	-	m	-	-	-	-	-	-	-

Functions (Version)	GAUSS	Macsyma	Maple	Mathe-matica	Matlab	MuPAD	O-Matrix	Ox	R-Lab	Scilab	S-Plus
	(3.2.38)	(2.4)	(V5.1)	(4.0)	(5.3)	(1.4.1)	(4.0)	(2.1)	(2.0.12)	(2.4.1)	(V4.5)
Bessel	m	-	-	m	-	-	-	-	-	-	-
Butterworth	m	-	-	m	m	-	+	-	-	+	-
Chebyshev	m	-	-	m	m	-	+	-	-	+	-
Coiflet	m	-	-	m	-	-	-	-	-	-	-
Daubechies	m	-	-	m	m	-	-	-	-	-	-
Haar	m	-	-	m	m	-	-	-	-	-	-
Hodrick-Prescott	-	-	-	-	-	-	-	+	-	-	-
IIR / FIR	m / m	- / -	- / -	m / m	m / m	- / -	- / -	- / -	- / -	+ / +	- / -
Kernel	-	-	-	-	-	-	-	+	-	-	+
Linear	m	-	-	m	m	-	-	-	-	+	-
Meyer	-	-	-	m	m	-	-	-	-	-	-
Pollen	m	-	-	-	-	-	-	-	-	-	-
Shannon	-	-	-	m	-	-	-	-	-	-	-
Savistky-Golay	m	-	-	-	m	-	-	-	-	-	-
Time series models											
ARIMA / ARFIMA / ARMAX	m / m / -	- / - / -	- / - / -	m / - / -	m / - / m	- / - / -	- / - / -	m / m / m	- / - / -	- / - / +	+ / - / -
GARCH / ARCH / AGARCH / EGARCH / IGARCH / MGARCH / PGARCH / TGARCH models	m/m/m/m/ m/m/m/m	-/-/-/- -/-/-/-	-/-/-/- -/-/-/-	m/m/-/- -/-/-/-	m/m/-/- -/-/-/-	-/-/-/- -/-/-/-	-/-/-/- -/-/-/-	-/m/-/m/ -/-/-/-	-/-/-/- -/-/-/-	-/-/-/- -/-/-/-	m/m/m/m/ -/-/m/m
Multivariate GARCH models (Diagonal VEC / BEKK / Matrix Diagonal / Vector Diagonal)	m/m/m/m	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	-/-/-/-	m/m/m/m
Spectral analysis	m	+	-	m	m	-	-	+	-	+	+
State space models	m	-	-	m	m	-	-	m	-	+	-
Time series analysis (Stationary / Non-stat.)	m / m	- / -	- / -	m / m	m / m	- / -	- / -	+ / +	- / -	+ / +	+ / +
Wavelets	m	-	-	m	m	-	+	-	-	+	m
Multivariate statistics											
Discriminant analysis	-	-	-	-	m	-	-	-	-	-	-
Fuzzy clustering	-	-	-	-	-	-	-	-	-	-	+

Functions (Version)	**GAUSS**	**Macsyma**	**Maple**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**R-Lab**	**Scilab**	**S-Plus**
	(3.2.38)	**(2.4)**	**(V5.1)**	**(4.0)**	**(5.3)**	**(1.4.1)**	**(4.0)**	**(2.1)**	**(2.0.12)**	**(2.4.1)**	**(V4.5)**
Hierachical cluster analysis	-	-	-	-	m	-	-	-	-	-	+
K-means cluster analysis	-	-	-	-	-	-	-	-	-	-	+
Principal component analysis	+	-	-	-	m	-	-	-	-	-	+
Survival analysis	-	-	-	-	-	-	-	-	-	-	+
Other statistical functions & models											
Bootstrapping	m	-	-	-	m	-	-	+	-	-	+
Duration models	m	-	-	-	-	-	-	-	-	-	-
Entropy models	m	-	-	-	-	-	-	-	-	-	-
Event count models	m	-	-	-	-	-	-	-	-	-	+
Granger causality test	m	-	-	-	-	-	-	-	-	-	-
Heckman two step estimation	m	-	-	-	-	-	-	-	-	-	-
Heteroscedasticity	m	-	-	-	-	-	-	+	-	-	+
Jacknife estimation	m	-	-	-	-	-	-	-	-	-	+
Lagrange multiplier test	m	-	-	-	-	-	-	-	-	-	-
Markowitz efficient frontier	m	-	-	m	m	-	-	-	-	-	-
Maximum Likelihood (Unconstr. / Constr.)	m / m	- / -	- / -	+ / -	m / -	- / -	- / -	- / -	- / -	- / -	+ / +
Monte Carlo simulation	m	-	-	m	m	-	-	+	-	-	-
Implemented functions	76.25% (61/80)	7.50% (6/80)	3.75% (3/80)	42.50% (34/80)	48.75% (39/80)	0.00% (0/80)	12.50% (10/80)	22.50% (18/80)	0.00% (0/80)	23.75% (19/80)	50.00% (40/80)

2.7. Other mathematics

Functions (Version)	**GAUSS**	**Macsyma**	**Maple**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**R-Lab**	**Scilab**	**S-Plus**
	(3.2.38)	**(2.4)**	**(V5.1)**	**(4.0)**	**(5.3)**	**(1.4.1)**	**(4.0)**	**(2.1)**	**(2.0.12)**	**(2.4.1)**	**(V4.5)**
Amortization schedule	m	-	+	m	-	-	-	-	-	-	-
Cash flow model	m	+	+	m	m	-	-	-	-	-	-
Cointegration models	m	-	-	-	-	-	-	+	-	-	-
Black Scholes model	m	+	+	m	m	-	-	-	-	-	-
Dynamic rational expectation models	m	-	-	-	-	-	-	-	-	-	-
Linear rational expectation models	m	-	-	-	m	-	-	-	-	-	-
LISREL models	m	-	-	-	-	-	-	-	-	-	-
Non-linear rational expectation models	m	-	-	-	-	-	-	-	-	-	-
Social network models	m	-	-	-	-	-	-	-	-	-	-
Kalman filter	m	-	-	m	m	-	+	m	-	+	-
Neural networks (Forward propagation/Backward propagation)	m / m	- / -	- / -	- / -	m / m	- / -	- / -	- / -	- / -	- / -	- / -
Panel models	m	-	-	-	-	-	-	m	-	-	-
Regressive-autoregressive models	m	-	-	m	m	-	-	+	-	-	+
Portfolio analysis	m	-	-	m	m	-	-	-	-	-	-
Implemented functions	100.00% (15/15)	13.33% (2/15)	20.00% (3/15)	40.00% (6/15)	53.33% (8/15)	0.00% (0/15)	6.67% (1/15)	26.67% (4/15)	0.00% (0/15)	6.67% (1/15)	6.67% (1/15)

2.8. Summarizing the comparison of the mathematical functions

Altogether 315 functions have been listed in the tables above. The following table summarizes the results calculated in percentage with a weighting of 5% for ‘Standard mathematics’, 15% for ‘Linear Algebra’, 10% for ‘Analysis’, 10% for ‘Numerical mathematics’, 20% for ‘Descriptive statistic, stochastic and distribution functions’, 20% for ‘Statistics’ and 20% for ‘Other mathematics’.

Functions (Version)	**GAUSS**	**Macsyma**	**Maple**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**R-Lab**	**Scilab**	**S- Plus**
	(3.2.38)	**(2.4)**	**(V5.1)**	**(4.0)**	**(5.3)**	**(1.4.1)**	**(4.0)**	**(2.1)**	**(2.0.12)**	**(2.4.1)**	**(V4.5)**
Standard mathematics (5%)	86.67%	100.00%	100.00%	100.00%	93.33%	80.00%	66.67%	100.00%	33.33%	80.00%	53.33%
Linear Algebra (15%)	93.94%	78.79%	69.70%	84.85%	93.94%	54.55%	63.64%	51.52%	60.61%	87.88%	69.70%
Analysis (10%)	100.00%	63.64%	63.64%	100.00%	81.82%	63.64%	63.64%	45.46%	18.18%	100.00%	72.73%
Numerical mathematics (10%)	71.43%	85.71%	85.71%	100.00%	85.71%	85.71%	71.43%	28.57%	42.86%	42.86%	42.86%
Descriptive statistics, stochastic and distribution functions (20%)	72.76%	49.37%	47.47%	83.54%	55.70%	6.96%	10.13%	31.01%	10.76%	20.25%	36.71%
Statistics (20%)	76.25%	7.50%	3.75%	42.50%	48.75%	0.00%	12.50%	22.50%	0.00%	23.75%	50.00%
Other mathematics (20%)	100.00%	13.33%	20.00%	40.00%	53.33%	0.00%	6.67%	26.67%	0.00%	6.67%	6.67%
Overall result (100% = Best)	85.37%	45.79%	44.63%	70.94%	67.07%	28.51%	32.24%	36.17%	19.01%	41.60%	43.36%

100% would mean that all listed functions are implemented (319 of 319).

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 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. Graphic types

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

Functions (Version)	**GAUSS**	**Macsyma**	**Maple**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**R-Lab**	**Scilab**	**S-Plus**
	(3.2.38)	**(2.4)**	**(V5.1)**	**(4.0)**	**(5.3)**	**(1.4.1)**	**(4.0)**	**(2.1)**	**(2.0.12)**	**(2.4.1)**	**(V4.5)**
2D-Graphics											
Area charts	-	-	-	+	+	-	-	-	-	-	+
Bar charts	+	+	+	+	+	+	+	-	-	-	+
Bubble Plot	-	-	-	-	-	-	-	+	-	-	+
Other charts	+	-	-	+	+	-	+	+	-	+	+
Error bars	-	-	-	+	+	-	+	+	-	+	+
High-Low-Average Plot	-	-	-	m	m	-	-	-	-	-	+
Histograms	+	+	+	+	+	-	+	+	-	+	+
Log Plot	+	+	+	+	+	+	+	-	-	+	+
Log-log Plot	+	+	+	+	+	+	+	-	-	+	+
Pie charts	-	-	-	+	+	+	-	-	-	-	+
Polar Plot	+	+	+	+	+	+	+	-	-	-	+
Weibull Plot	-	-	-	-	m	-	-	-	-	-	-
XY Plot	+	+	+	+	+	+	+	+	-	+	+

Functions (Version)	**GAUSS**	**Macsyma**	**Maple**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**R-Lab**	**Scilab**	**S-Plus**
	(3.2.38)	**(2.4)**	**(V5.1)**	**(4.0)**	**(5.3)**	**(1.4.1)**	**(4.0)**	**(2.1)**	**(2.0.12)**	**(2.4.1)**	**(V4.5)**
3D-Graphics											
Charts	-	-	-	+	+	-	-	-	-	+	+
Contour Plot	+	+	+	+	+	+	+	-	-	+	+
Error bars	-	-	-	-	-	-	-	-	-	-	-
Height colors	+	+	+	+	+	+	+	-	-	+	+
Surface Plot	+	+	+	+	+	+	+	-	-	+	+
XYZ Plot	+	+	+	+	+	+	+	-	-	+	+
Special graphic types and functions											
Animations	-	+	+	+	+	+	-	-	-	+	-
Bollinger bands	-	-	-	m	m	-	-	-	-	-	-
Box & Whisker Plots	+	-	+	m	m	-	-	+	-	-	+
Candlestick charts	-	-	-	m	m	-	-	-	-	-	-
Cluster graphs	-	-	-	-	-	-	-	-	-	-	m
Dendograms	-	-	-	-	m	-	-	-	-	-	m
Periodograms	-	-	-	-	-	-	-	+	-	-	+
QQ Plot	m	-	+	-	m	-	-	+	-	-	+
Quantile Plot	+	-	+	-	m	-	-	+	-	-	+
Scatter Plot Matrix	+	-	-	-	+	-	-	+	-	-	+
Overall result ***(100% = Best)***	51.72% (15/29)	37.93% (11/29)	48.28% (14/29)	68.97% (20/29)	86.21% (25/29)	37.93% (11/29)	41.38% (12/29)	34.48% (10/29)	0.00% (0/29)	41.38% (12/29)	82.76% (24/29)

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**	**Macsyma**	**Maple**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**R-Lab**	**Scilab**	**S-Plus**
	(3.2.38)	**(2.4)**	**(V5.1)**	**(4.0)**	**(5.3)**	**(1.4.1)**	**(4.0)**	**(2.1)**	**(2.0.12)**	**(2.4.1)**	**(4.5)**
BMP	+ / -	+ / -	+ / -	+ / +	+ / +	- / -	- / -	- / -	- / -	- / -	+ / -
GIF	- / -	+ / -	+ / -	+ / +	- / -	+ / -	- / -	- / -	- / -	- / -	+ / -
HPGL (export only)	+	-	+	-	+	-	-	-	-	-	+
JPG	- / -	- / -	+ / -	+ / +	+ / +	- / -	- / -	- / -	- / -	- / -	+ / -
PCX	+ / -	+ / -	+ / -	- / -	+ / +	- / -	- / -	- / -	- / -	- / -	- / -
PS / EPS (export only)	+ / +	- / -	+ / -	+ / +	+ / +	+ / -	- / -	+ / +	- / -	+ / -	+ / -
TIFF	+ / -	- / -	- / -	+ / +	+ / +	- / -	- / -	- / -	- / -	- / -	+ / -
WMF	+ / -	- / -	- / -	+ / +	+ / -	- / -	- / -	+ / -	- / -	- / -	+ / -
Overall result ***(100% = Best)***	46.67% (7/15)	20.00% (3/15)	40.00% (6/15)	80.00% (12/15)	80.00% (12/15)	13.33% (2/15)	0.00% (0/15)	20.00% (3/15)	0.00% (0/15)	6.67% (1/15)	46.67% (7/15)

3.3. Summarizing the comparison of the graphical functionality

Altogether 41 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**	**Macsyma**	**Maple**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**R-Lab**	**Scilab**	**S-Plus**
	(3.2.38)	**(2.4)**	**(V5.1)**	**(4.0)**	**(5.3)**	**(1.4.1)**	**(4.0)**	**(2.1)**	**(2.0.12)**	**(2.4.1)**	**(4.5)**
Graphic types (75%)	51.72%	37.93%	48.28%	68.97%	86.21%	37.93%	41.38%	34.48%	0.00%	41.38%	82.76%
Graphic import/export formats (25%)	46.67%	20.00%	40.00%	80.00%	80.00%	13.33%	0.00%	20.00%	0.00%	6.67%	46.67%
Overall result (100% = Best)	50.46%	33.45%	46.21%	71.72%	84.66%	31.78%	31.03%	30.86%	0.00%	32.70%	73.74%

4. Functionality of the programming environment

For a lot of scientists it is very important to define complex models and to do simulations on them 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**	**Macsyma**	**Maple**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**R-Lab**	**Scilab**	**S-Plus**
	(3.2.38)	**(2.4)**	**(V5.1)**	**(4.0)**	**(5.3)**	**(1.4.1)**	**(4.0)**	**(2.1)**	**(2.0.12)**	**(2.4.1)**	**(V4.5)**
Editing features											
Built-in editor	+	+	+	+	+	+	+	+	-	-	+
External editor configurable	+	-	+	+	+	-	-	-	-	-	+
Source code formatting	m	+	-	+	+	-	-	-	-	-	+
Syntax highlighting	-	-	-	+	+	-	-	m	-	-	-
Debugging											
Breakpoints	+	+	+	-	+	+	+	+	-	+	+
Function Tracer	-	+	+	+	-	+	-	+	-	-	-
Line Tracing	+	-	-	-	+	+	+	+	-	-	+
Profiler	-	+	+	m	+	+	+	-	-	-	-
Stack inspection	+	+	-	+	+	+	+	-	-	+	+
Variable inspection	+	+	-	+	+	+	+	+	-	+	+
Language features											
API-interface	+	-	-	+	+	-	+	+	-	+	m
DDE support	-	-	-	-	+	+	-	-	-	-	+
GUI programming	-	-	-	+	+	-	+	m	-	+	+
N-dimensional arrays (> 3)	-	+	+	+	+	+	-	+	-	+	+
Object oriented programming	-	-	-	+	+	+	+	+	+	-	+
OLE support	-	-	+	+	+	+	-	-	-	-	+
P code compiling	+	+	-	+	+	-	-	+	-	+	-

Programming facilities	**GAUSS**	**Macsyma**	**Maple**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**R-Lab**	**Scilab**	**S-Plus**
	(3.2.38)	**(2.4)**	**(V5.1)**	**(4.0)**	**(5.3)**	**(1.4.1)**	**(4.0)**	**(2.1)**	**(2.0.12)**	**(2.4.1)**	**(V4.5)**
Language interfaces											
Assembler	+	-	-	+	-	+	-	+	-	-	-
C/C++	+	-	-	+	+	+	-	+	-	+	+
FORTRAN	+	+	+	+	+	+	-	+	-	+	+
GAUSS	+	-	-	-	-	-	-	+	-	-	-
Macsyma	-	+	-	-	-	-	-	-	-	-	-
Maple	m	-	+	-	m	+	-	-	-	+	-
Mathematica	m	-	-	+	m	-	+	-	-	-	-
Matlab	-	+	-	m	+	-	-	-	-	-	-
MuPAD	-	-	-	-	-	+	-	-	-	-	-
O-Matrix	-	-	-	m	-	-	+	-	-	-	-
Ox	-	-	-	-	-	-	-	+	-	-	-
Scilab	-	-	-	-	-	-	-	-	-	+	-
S-Plus	-	-	-	-	-	-	-	-	-	-	+
DLL-Calls	+	-	-	+	+	-	+	+	-	+	+
Miscellaneous											
Developer Engine	m	-	-	-	-	-	-	+	-	-	m
Redistr. with runtime licenses	m	-	-	-	m	-	-	-	-	-	-
Overall result ***(100% = Best)***	54.55% (18/33)	36.36% (12/33)	27.27% (9/33)	63.64% (21/33)	69.70% (23/33)	48.48% (16/33)	36.36% (12/33)	54.55% (18/33)	3.03% (1/33)	36.36% (12/33)	54.55% (18/33)

The final result is display in percent, 100% means that every programming facility has been supported.

5. 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**	**Macsyma**	**Maple**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**R-Lab**	**Scilab**	**S-Plus**
	(3.2.38)	**(2.4)**	**(V5.1)**	**(4.0)**	**(5.3)**	**(1.4.1)**	**(4.0)**	**(2.1)**	**(2.0.12)**	**(2.4.1)**	**(V4.5)**
ACCESS	-	-	-	m	-	-	-	-	-	-	+
Applixware	-	-	-	-	m	-	-	-	-	-	-
ASCII	+	+	+	+	+	+	+	+	+	+	+
AutoCAD	-	+	-	+	-	-	-	-	-	-	-
Binary	+	-	+	+	+	+	+	+	-	+	-
dBase	+	-	-	-	-	-	-	-	-	-	+
Excel	+	-	-	m	m	-	-	+	-	-	+
FoxPro	+	-	-	-	-	-	-	-	-	-	+
GAUSS	+	-	-	-	-	-	-	+	-	-	+
Informix	-	-	-	m	-	-	-	-	-	-	-
Labview	-	-	-	m	-	-	-	-	-	-	-
Lotus 1-2-3	+	-	-	-	+	-	-	+	-	-	+
Lotus Symphony	+	-	-	-	-	-	-	-	-	-	+
Matlab	-	+	-	+	+	-	+	-	-	+	+
ODBC-connections	m	-	-	-	m	-	-	-	-	-	+
Ox	-	-	-	-	-	-	-	+	-	-	-
Paradox	+	-	-	-	-	-	-	-	-	-	+
Quattro Pro	+	-	-	-	-	-	-	-	-	-	+
SAS	-	-	-	-	-	-	-	-	-	-	+

Data import/export possibilities	**GAUSS**	**Macsyma**	**Maple**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**R-Lab**	**Scilab**	**S-Plus**
	(3.2.38)	**(2.4)**	**(V5.1)**	**(4.0)**	**(5.3)**	**(1.4.1)**	**(4.0)**	**(2.1)**	**(2.0.12)**	**(2.4.1)**	**(V4.5)**
SigmaPlot	-	-	-	-	-	-	-	-	-	-	+
S-Plus	-	-	-	-	-	-	-	-	-	-	+
SPSS	-	-	-	-	-	-	-	-	-	-	+
STATA	-	-	-	-	-	-	-	+	-	-	+
Systat	-	-	-	-	-	-	-	-	-	-	+
Transform	-	-	-	m	+	-	-	-	-	-	-
Overall result ***(100% = Best)***	44.00% (11/25)	12.00% (3/25)	8.00% (2/25)	36.00% (9/25)	32.00% (8/25)	8.00% (2/25)	12.00% (3/25)	28.00% (7/25)	4.00% (1/25)	12.00% (3/25)	72.00% (18/25)

The final result is displayed in percent, 100% means that every import/export format has been supported.

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**	**Macsyma**	**Maple**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**R-Lab**	**Scilab**	**S-Plus**
	(3.2.38)	**(2.4)**	**(V5.1)**	**(4.0)**	**(5.3)**	**(1.4.1)**	**(4.0)**	**(2.1)**	**(2.0.12)**	**(2.4.1)**	**(V4.5)**
DEC (UNIX)	+	-	+	+	+	+	-	+	-	+	+
HP 9000 (HP-UX)	+	+	+	+	+	+	-	+	-	+	+
IBM RISC (IBM AIX)	+	+	+	+	+	+	-	+	-	+	+
Intel (OS/2)	+	-	-	-	-	+	-	-	+	-	-
Intel (Win. 9x,NT)	+	+	+	+	+	+	+	+	+	+	+
Intel (Linux)	+	+	+	+	+	+	-	+	+	+	+
Motorola (MAC OS)	-	-	+	+	-	+	-	-	+	-	-
SGI (SGI IRIX)	+	+	+	+	+	+	-	+	-	+	+
SUN (Solaris)	+	+	+	+	+	+	-	+	+	+	+
Total amount	88.89% (8/9)	66.67% (6/9)	88.89% (8/9)	88.89% (8/9)	77.78% (7/9)	100.00% (9/9)	11.11% (1/9)	77.78% (7/9)	55.56% (5/9)	77.78% (7/9)	77.78% (7/9)

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.

7. Speed comparison

The following speed comparison have been performed on a Pentium-II with 400 MHz and 384 MB RAM running under Windows NT 4.0 with Windows versions of the programs (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 30 minutes.

The speed comparison tests 16 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**	**Macsyma**	**Maple**	**Mathe-matica**	**Matlab**	**MuPAD**	**O-Matrix**	**Ox**	**R-Lab**	**Scilab**	**S-Plus**
	(3.2.38)	**(2.4)**	**(V5.1)**	**(4.0)**	**(5.3)**	**(1.4.1)**	**(4.0)**	**(2.1)**	**(2.0.12)**	**(2.4.1)**	**(V4.5)**
Creation, transposing and deformation of a 1000x1000 matrix	0.998	*	***	1.963	0.671	383.392	0.891	0.995	1.112	2.483	2.150
1000x1000 random matrix^1000	1.165	1447.360	**	0.781	1.843	21.771	0.841	1.165	1.021	1.953	1.030
Sorting of 2000000 random values	11.543	815.292	***	7.090	19.459	31.265	2.474	7.798	6.730	5.949	3.600
FFT over 1048576 (= 2^20) random values	9.013	**	*	2.734	4.192	*	2.554	4.266	5.521	4.987	5.020
Double integration	0.004	2.174	1.412	31.295	1.162	1.832	0.110	-	-	0.480	-
Triple integration	0.190	3.404	0.471	*	-	90.210	-	-	-	42.270	-
Determinant of a 1000x1000 random matrix	21.741	77.491	*	22.342	17.204	*	21.431	9.16	13.697	23.433	744.910
Creation of a 1400x1400 Toeplitzmatrix	0.955	*	314.783	2.624	1.636	66.856	10.435	0.160	27.696	2.674	3.040
Inverse of a 1000x1000 random matrix	55.422	1413.001	*	97.460	43.997	*	26.908	26.969	68.933	69.910	126.520
Eigenvalues of a 600x600 random matrix	78.516	3.284	*	33.107	49.087	*	9.413	39.203	81.196	87.937	66.510
Cholesky decomposition of a 1000x1000 random matrix	4.564	*	*	5.067	4.358	*	1.182	4.423	5.631	9.183	13.030
1000x1000 crossproduct matrix	28.554	*	*	29.413	26.143	*	6.139	13.142	65.200	130.477	45.870
Calculation of 500000 Fibonacci numbers	1.202	*	*	19.027	2.467	2.454	1.452	1.385	1.719	1.702	2.280
Gamma function on a 1000x1000 random matrix	1.562	*	*	16.504	17.531	842.832	2.764	0.814	-	2.924	2.500
Gaussian error function on a 1000x1000 random matrix	1.933	*	*	10.866	11.641	1267.172	1.142	0.931	-	2.454	-
Linear regression over a 1000x1000 random matrix	44.134	**	*	10.636	17.621	*	16.464	17.939	16.924	24.579	35.900
Overall performance	47.97%	7.49%	2.54%	31.95%	34.64%	3.84%	67.29%	68.12%	41.13%	27.85%	30.52%

***** ***Maximum running time of 30 minutes has been exceeded.***
****** ***Memory allocation problems***
******* ***Function doesn't support huge data sizes.***

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 16*) 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

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

Some of the prices below are based on information of distributors in Germany or software producers in UK and are therefore not mentioned in US$. However for comparison purposes I have recalculated the pricing in US$ which are mentioned in brackets.

Functions	**GAUSS**	**Macsyma**	**Maple**	**Mathematica**	**Matlab**
(Version)	**(3.2.38)**	**(2.4)**	**(V5.1)**	**(4.0)**	**(5.3)**
Software producer	Aptech Systems Inc. (www.aptech.com)	Macsyma Inc. (www.macsyma.com)	Waterloo Maple Software Inc. (www.maplesoft.com)	Wolfram Research Inc. (www.wolfram.com)	The Mathworks Inc. (www.mathworks.com)
Pricing for Windows					
Commercial license	2495 US$	1195 DM (687 US$)	2610 DM (1496 US$)	3285 DM (1883 US$)	unknown
Non-commercial license	695 US$	575 DM (249 US$)	2225 DM (1276 US$)	2085 DM (1195 US$)	unknown
Student license	40 US$	199 US$	155 DM (89 US$)	257 DM (147 US$)	117 DM (68 US$)
Remarks	Light version is memory limited. Student version is a light version.	Lite version available for a special pricing but with restricted capabilities	Student version is restricted in functionality.		Student version is restricted in functionality.
Operation / Programming handling					
User interface	4	3	3	2	3
Graphics	4	2	3	2	3
Programming language (similar to)	2 (Basic, Fortran)	3	2 (Pascal)	3 (Lisp)	2 (Basic, Fortran)

Functions (Version)	**GAUSS**	**Macsyma**	**Maple**	**Mathematica**	**Matlab**
	(3.2.38)	**(2.4)**	**(V5.1)**	**(4.0)**	**(5.3)**
Books / Support					
Online help / Electronic handbook	3	3	2	1	3 (3)
Additional books	5	6	1	1	3
FAQ lists	5	5	2	2	2
Newsgroups / mailing lists	1	6	2	1	1
Programm archives by the software producer	6	6	3	1	2
Programm archives by external institutions	1	6	1	1	3

Functions (Version)	**MuPAD**	**O-Matrix**	**Ox**	**R-Lab**	**Scilab**	**S-Plus**
	(1.4.1)	**(4.0)**	**(2.1)**	**(2.0.12)**	**(2.4.1)**	**(4.5)**
Software producer	SciFace GmbH (www.sciface.de)	Harmonic Software Inc. (www.omatrix.com)	Jurgen Doornik (www.nuff.ox.ac.uk/Users/doornik)	Ian Searle (www.eskimo.com/~ians/rlab.html)	INRIA / Dr. Scilab (www-rocq.inria.fr/scilab/)	MathSoft Inc. (www.mathsoft.com)
Pricing for Windows						
Commercial license	578 DM (220 US$)	265 US$	500 £ (817 US$)	free	free	1695 £ (2770 US$)
Non-commercial license	98 DM (56 US$)	49 US$	250 £ (409 US$)	free	free	795 £ (1299 US$)
Student license	98 DM (56 US$)	49 US$	85 £ (139 US$)	free	free	unknown
Remarks	Light version is available for non-commercial use for free. Macintosh and Linux version is free.	Light version is memory limited. Light version available for free	Command line version without interface is available for non-commercial use for free.			

Functions	**MuPAD**	**O-Matrix**	**Ox**	**R-Lab**	**Scilab**	**S-Plus**
(Version)	**(1.4.1)**	**(4.0)**	**(2.1)**	**(2.0.12)**	**(2.4.1)**	**(4.5)**
Operation / Programming handling						
User interface	4	2	4	6	4	2
Graphics	3	5	3	6	3	2
Programming language (similar to)	3 (Pascal)	2 (Basic, Fortran)	3 (C, C++)	4 (C)	2 (Basic, Fortran)	3 (C++)
Books / Support						
Online help / Electronic handbook	3	2	3	6	5 (3)	4 (2)
Additional books	5	6	6	6	6	3
FAQ lists	4	6	5	6	4	3
Newsgroups / mailing lists	5	6	5	6	5	2
Programm archives by the software producer	5	4	4	4	4	6
Programm archives by external institutions	5	6	5	5	5	3

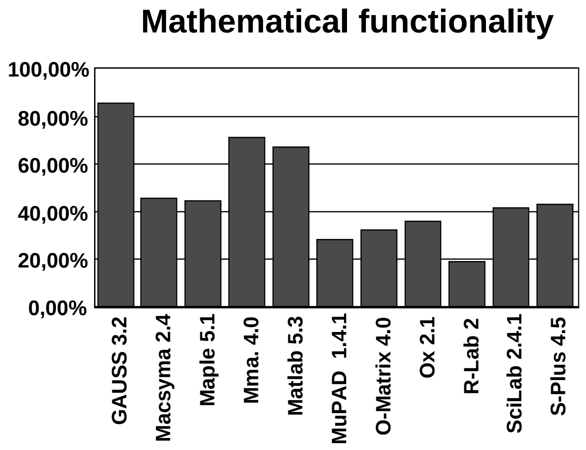
The informations in this table are judge by school marks from 1 until 6 (1 - best, 6 - worst) and represent my own subjective opinion. The mark 6 normally means that something is not supported, the fact that something is supported so badly that the mark 6 have been justified never occurred. Conversely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

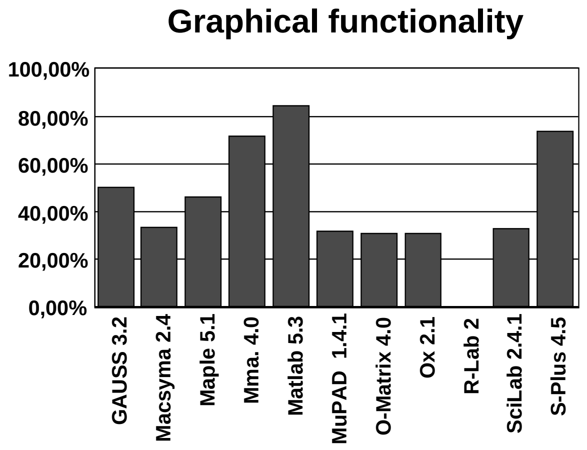
8.2. Summary

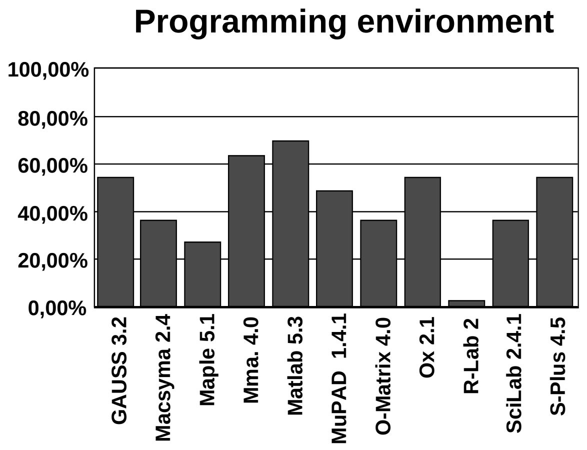
The summary should set the results of the speed comparison, the functionality of the programming environment, the data import/export facilities and the availability for different platforms in relation to the results of the comparison of the mathematical and graphical functionality. The relation between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

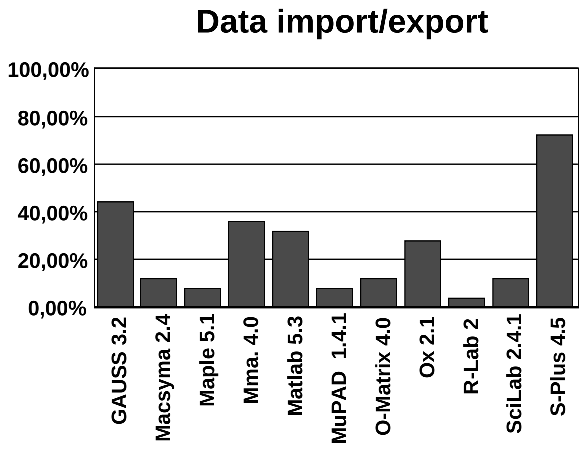
Test	GAUSS	Macsyma	Maple	Mathe-matica	Matlab	MuPAD	O-Matrix	Ox	R-Lab	Scilab	S-Plus
	(3.2.38)	(2.4)	(V5.1)	(4.0)	(5.3)	(1.4.1)	(4.0)	(2.1)	(2.0.12)	(2.4.1)	(V4.5)
Comparison of the mathematical functionality (38%)	85.37%	45.79%	44.63%	70.94%	67.07%	28.51%	32.24%	36.17%	19.01%	41.60%	43.35%
Comparison of the graphical functionality (10%)	50.46%	33.45%	46.21%	71.72%	84.66%	31.78%	31.03%	30.86%	0.00%	32.70%	73.74%
Functionality of the programming environment (9%)	54.55%	36.36%	27.27%	63.64%	69.70%	48.48%	36.36%	54.55%	3.03%	36.36%	54.55%
Data import/export (5%)	44.00%	12.00%	8.00%	36.00%	32.00%	8.00%	12.00%	28.00%	4.00%	12.00%	72.00%
Available platforms (2%)	88.89%	66.67%	88.89%	88.89%	77.78%	100.00%	11.11%	77.78%	55.56%	77.78%	77.78%
Speed comparison (36%)	47.96%	7.49%	2.54%	31.95%	34.64%	3.84%	67.29%	68.12%	41.13%	27.85%	30.51%
Overall result	63.64%	28.65%	27.13%	54.93%	55.85%	22.16%	43.68%	49.22%	23.62%	34.53%	44.90%

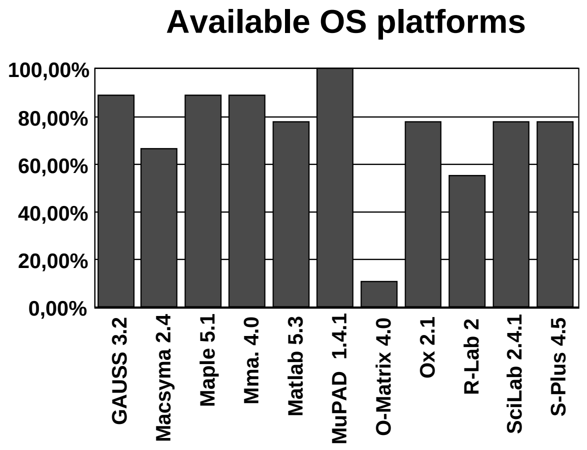
Note: The overall results of some tested programs are pretty bad due to the specific weighting of this testreport. I would like to mention that this does of course not mean that the software is bad but that the programs are maybe not perfect for the specific usage mentioned in this testreport, for other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

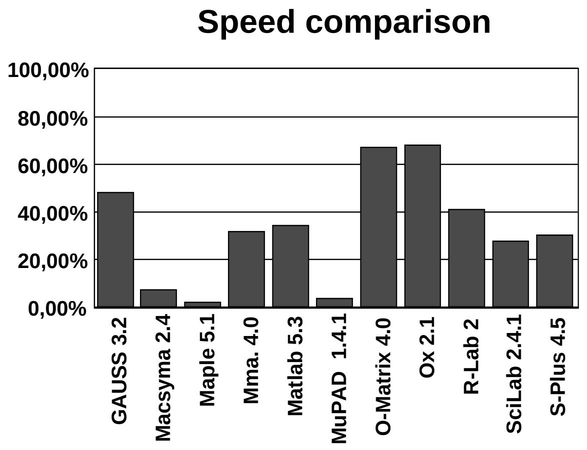

Mathematical functionality
100,00%
80,00%
60,00%
40,00%
20,00%
0,00%
GAUSS 3.2
Macsyma 2.4
Maple 5.1
Mma. 4.0
Matlab 5.3
MuPAD 1.4.1
O-Matrix 4.0
Ox 2.1
R-Lab 2
SciLab 2.4.1
S-Plus 4.5

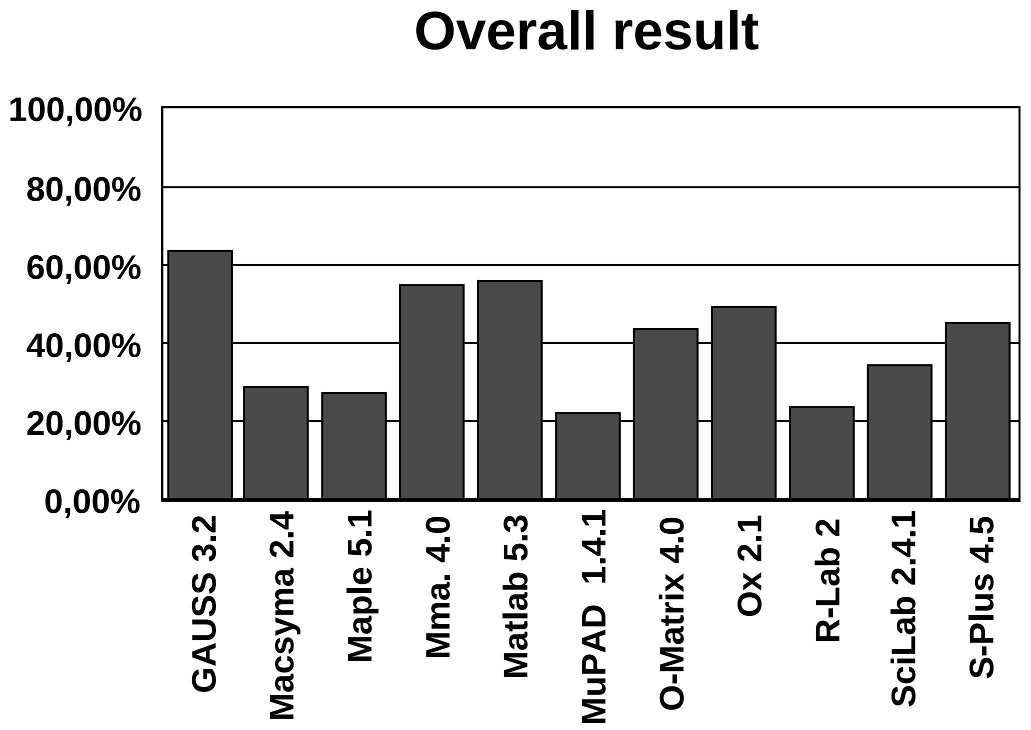

Graphical functionality
100,00%
80,00%
60,00%
40,00%
20,00%
0,00%
GAUSS 3.2
Macsyma 2.4
Maple 5.1
Mma. 4.0
Matlab 5.3
MuPAD 1.4.1
O-Matrix 4.0
Ox 2.1
R-Lab 2
SciLab 2.4.1
S-Plus 4.5


Programming environment
100,00%
80,00%
60,00%
40,00%
20,00%
0,00%
GAUSS 3.2
Macsyma 2.4
Maple 5.1
Mma. 4.0
Matlab 5.3
MuPAD 1.4.1
O-Matrix 4.0
Ox 2.1
R-Lab 2
SciLab 2.4.1
S-Plus 4.5


Data import/export
100,00%
80,00%
60,00%
40,00%
20,00%
0,00%
GAUSS 3.2
Macsyma 2.4
Maple 5.1
Mma. 4.0
Matlab 5.3
MuPAD 1.4.1
O-Matrix 4.0
Ox 2.1
R-Lab 2
SciLab 2.4.1
S-Plus 4.5


Available OS platforms
100,00%
80,00%
60,00%
40,00%
20,00%
0,00%
GAUSS 3.2
Macsyma 2.4
Maple 5.1
Mma. 4.0
Matlab 5.3
MuPAD 1.4.1
O-Matrix 4.0
Ox 2.1
R-Lab 2
SciLab 2.4.1
S-Plus 4.5


Speed comparison
100,00%
80,00%
60,00%
40,00%
20,00%
0,00%
GAUSS 3.2
Macsyma 2.4
Maple 5.1
Mma. 4.0
Matlab 5.3
MuPAD 1.4.1
O-Matrix 4.0
Ox 2.1
R-Lab 2
SciLab 2.4.1
S-Plus 4.5



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 2.0                                  */
/************************************************************************/

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

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

print;
print "After a single test has been run the amount of times you have choosen";
print "the program will print the average elapsed time and continues with";
print "the next test calculation."; print;
print "Dependent on the amount of runs you will choose now it could take some";
print "time until you see the first results."; print;
print "How much times do you want to run the test (I suggest at least 3 times) :";
zaehl=con(1,1);
print; print; print;
print "!!! GAUSS - Benchmarkprogram !!!";
print "================================"; print;

/* Misc. operation */

ergeb=0; zeit=0; a=0;b=0;
format /RD 8,3;
for i (1,zaehl,1);
      timing=0; zeit=hsec;
      b=abs(rndn(1000,1000)/10); a=b’; b=reshape(a,500,2000); a=b’; timing=(hsec-zeit)/100;
      ergeb=ergeb+timing;
endfor;
ergeb=ergeb/zaehl;
print /flush "Creation, trans. & reshaping of a 1000x1000 matrix___ :" ergeb " sec.";
clear a,b,ergeb;
for i (1,zaehl,1);
      timing=0; b=abs(rndn(1000,1000)/10);
      zeit=hsec; a=(b+1)^1000; timing=(hsec-zeit)/100;
      ergeb=ergeb+timing;
endfor;
ergeb=ergeb/zaehl;
print /flush "1000x1000 normal distributed random matrix^1000______ :" ergeb " sec.";
clear a,b,ergeb;
for i (1,zaehl,1);
      timing=0; a=rndn(2000000,1);
      zeit=hsec; b=sortc(a,1); timing=(hsec-zeit)/100;
      ergeb=ergeb+timing;
endfor;
ergeb=ergeb/zaehl;
print /flush "2000000 values sorted ascending______________________ :" ergeb " sec.";

/* Analysis */

clear a,b,ergeb;
for i (1,zaehl,1);
      timing=0; a=rndn(1048576,1);
      zeit=hsec; b=fft(a); timing=(hsec-zeit)/100;
      ergeb=ergeb+timing;
endfor;
ergeb=ergeb/zaehl;
print /flush "FFT over 1048576 values______________________________ :" ergeb " sec.";
clear a,b,ergeb;
for i (1,10,1);
      timing=0; xl=3.1415926535897932|-3.1415926535897932;yl=xl;
      _intord=40;
      zeit=hsec; a=intquad2(&f,xl,yl); timing=(hsec-zeit)/100;
      ergeb=ergeb+timing;
endfor;
ergeb=ergeb/10;
print /flush "Double integration___________________________________ :" ergeb " sec.";
clear a,b,ergeb;
for i (1,zaehl,1);
      timing=0; xl=3.1415926535897932|-3.1415926535897932;yl=xl;zl=xl;
      _intord=40;
      zeit=hsec; a=intquad3(&g,xl,yl,zl); timing=(hsec-zeit)/100;
      ergeb=ergeb+timing;
endfor;
ergeb=ergeb/zaehl;
print /flush "Triple integration___________________________________ :" ergeb " sec.";

/* Algebra */

clear a,b,ergeb;
for i (1,zaehl,1);
      timing=0; a=rndn(1000,1000); zeit=hsec; b=det(a);timing=(hsec-zeit)/100;
      ergeb=ergeb+timing;
endfor;
```

```
ergeb=ergeb/zaehl;
print /flush "Determinant of a 1000x1000 random matrix______________ :" ergeb " sec.";
clear a,b,ergeb;
for i (1,zaehl,1);
      timing=0; a=seqa(1,1,1400);
      zeit=hsec; b=toeplitz(a); timing=(hsec-zeit)/100;
      ergeb=ergeb+timing;
endfor;
ergeb=ergeb/zaehl;
print /flush "1400x1400 Toeplitzmatrix_______________________________ :" ergeb " sec.";
clear a,b,ergeb;
for i (1,zaehl,1);
      timing=0; a=rndu(1000,1000);
      zeit=hsec; b=inv(a); timing=(hsec-zeit)/100;
      ergeb=ergeb+timing;
endfor;
ergeb=ergeb/zaehl;
print /flush "Inverse of a 1000x1000 uniform distr. random matrix__ :" ergeb " sec.";
clear a,b,ergeb;
for i (1,zaehl,1);
      timing=0; a=rndn(600,600);
      zeit=hsec; c=eig(a); timing=(hsec-zeit)/100;
      ergeb=ergeb+timing;
endfor;
ergeb=ergeb/zaehl;
print /flush "Eigenvalues of a normal distr. 600x600 randommatrix__ :" ergeb " sec.";
clear a,b,ergeb;
for i (1,zaehl,1);
      timing=0; a=moment(rndn(1000,1000),0);
      zeit=hsec; b=chol(a); timing=(hsec-zeit)/100;
      ergeb=ergeb+timing;
endfor;
ergeb=ergeb/zaehl;
print /flush "Cholesky decomposition of a 1000x1000-matrix_________ :" ergeb " sec.";
clear a,b,ergeb;
for i (1,zaehl,1);
      timing=0; a=rndn(1000,1000);
      zeit=hsec; b=moment(a,0); timing=(hsec-zeit)/100;
      ergeb=ergeb+timing;
endfor;
ergeb=ergeb/zaehl;
print /flush "1000x1000 cross-product matrix________________________ :" ergeb " sec.";

/* Number theory */

clear a,b,ergeb;
phi = 1.6180339887498949;
for i (1,zaehl,1);
      timing=0; a=floor(1000*rndu(500000,1));
      zeit=hsec; b=(phi^a-(-phi)^(-a))/sqrt(5); timing=(hsec-zeit)/100;
      ergeb=ergeb+timing;
endfor;
ergeb=ergeb/zaehl;
print /flush "Calculation of 500000 fibonacci numbers______________ :" ergeb " sec.";

/* Stochastic-statistic */

clear a,b,ergeb;
for i (1,zaehl,1);
      timing=0; a=rndn(1000,1000)^2;
      zeit=hsec; b=gamma(a);timing=(hsec-zeit)/100;
      ergeb=ergeb+timing;
endfor;
ergeb=ergeb/zaehl;
print /flush "Gamma function over a 1000x1000 matrix_______________ :" ergeb " sec.";
clear a,b,ergeb;
for i (1,zaehl,1);
      timing=0; a=rndn(1000,1000)^2;
      zeit=hsec; b=erf(a); timing=(hsec-zeit)/100;
      ergeb=ergeb+timing;
endfor;
ergeb=ergeb/zaehl;
print /flush "Gaussian error function over a 1000x1000 matrix______ :" ergeb " sec.";
clear a,b,ergeb;
for i (1,zaehl,1);
      timing=0; a=rndn(1000,1000); b=seqa(1,1,1000);
      zeit=hsec; b=olsqr(b,a); timing=(hsec-zeit)/100;
      ergeb=ergeb+timing;
endfor;
ergeb=ergeb/zaehl;
print /flush "Linear regression over a 1000x1000 matrix____________ :" ergeb " sec.";
end;
```

Macsyma Benchmark program:

Macsyma Benchmarktest
(Version 2)

```
Showtime:true
```

Misc. operations

Creation, trans. & reshaping of a 1000x1000 matrix

```
b:abs(mat_rand(1000,1000,1.0)/10)^‘$
a:mat_reshape(b,500,2000)^‘$
```

1000x1000 random matrix^1000

```
A:(1+MAT_RAND(1000,1000))$
A^1000$
```

Sorting of 2000000 random values

```
A:makelist(random(1.d0),j,1,2000000)$
sort(A)$
```

Analysis

FFT over 1048576 random values

```
A:MAT_RAND(2^20,1)$
fft(true,-1,A)$
```

Double integration

```
realpart(dfloat(integrate(integrate(sin(x^2+y^2),x,-%pi,%pi),y,-%pi,%pi)))$
```

Triple integration

```
realpart(dfloat(integrate(integrate(
integrate(sin(x^2+y^2+z^2),x,-%pi,%pi),y,-%pi,%pi),z,-%pi,%pi)))$
```

Algebra

Determinant of a 1000x1000 random matrix

```
load(nkaux)
A:MAT_RAND(1000,1000)$
det_by_lu(A)$
```

Creation of a 1400 x 1400 Töplitzmatrix

```
toeplitz(1..1400)$
```

Inverse of a 1000 x 1000 uniform distributed random matrix

```
A:MAT_RAND(1000,1000)$
Invert(A)$
```

Eigenvalues of a 600 x 600 normal distributed random matrix

```
A:MAT_RAND(600,600,1.d0)$
eigenvalues_by_schur(A)$
```

Cholesky decomposition of a 1000 x 1000 random matrix

```
A:MAT_RAND(1000,1000)$
A:transpose(A).A$
cholesky_decomp(A)$
```

Creation of a 1000 x 1000 cross-product matrix

```
A:MAT_RAND(1000,1000)$
transpose(A).A$
```

Number theory

Calculation of 500000 fibonacci numbers

```
A:floor(MAT_RAND(1,500000)*1000)$
matrixmap(fib,A);
```

Stochastic - statistic

Gamma function on a 1000 x 1000 random matrix

```
A:MAT_RAND(1000,1000)$
matrixmap(gamma,A)$
```

Gaussian error function over a 1000 x 1000 random matrix

```
A:MAT_RAND(1000,1000)$
matrixmap(erf,A)$
```

Linear regression over a 1000 x 1000 random matrix

```
A:MAT_RAND(1000,1000)$
B:vec2list(dfloat(1..1000))$
for i: 1 step 1 thru 1000 do block([tmp],
  tmp:vec2list(a[..,i]),C[i]:lsq1(B,tmp,1))$
```

Maple Benchmark program:

Maple Benchmarktest
(Version 2)

Initialisation

```
restart;
with(linalg, randmatrix):
with(stats):
with(linalg):
readlib(FFT):
```

Misc. operations

Creation, trans. & reshaping of a 1000x1000 matrix

```
t:=time():
b:=evalf(evalm(abs(randmatrix(1000,1000)/10))):
a:=transpose(b):
b:=linalg[matrix](500,2000,convert(convert(convert(a,table),list),array)):
a:=transpose(b):
time()-t;
```

1000x1000 normal distributed random matrix^1000

```
a:=evalf(evalm(randmatrix(1000,1000)/1000)):
t:=time():
map(x->x^1000,evalm((a+1))):
time()-t;
```

2000000 values sorted ascending

```
L := [seq(rand(),i=1..2000000)]:
t:=time():
sort(L):
time()-t;
```

Analysis

FFT over 1048576 values

```
m:=20:
x:=convert(convert(convert(evalf(evalm(randmatrix(2^m,1)/100)),table),list),array):
y:=convert([seq(0,i=1..2^m)],array):
t:=time():
FFT(m,x,y):
time()-t;
```

Double integration

```
t:=time():
evalf(int(int(sin(x^2+y^2), x=-Pi..Pi),y=-Pi..Pi)):
time()-t;
```

Triple integration

```
t:=time():
evalf(int(int(int(sin(x^2+y^2+z^2), x=-Pi..Pi),y=-Pi..Pi),z=-Pi..Pi)):
time()-t;
```

Algebra

Determinant of a 1000x1000 random matrix

```
a:=evalm(randmatrix(1000,1000)/100):
t:=time():
b:=evalf(det(a)):
time()-t;
```

1400x1400 Toeplitzmatrix

```
L := [seq(i,i=1..1400)]:
t:=time():
toeplitz(L):
time()-t;
```

Inverse of a 1000x1000 uniform distr. random matrix

```
a:=evalm(randmatrix(1000,1000)/100):
t:=time():
evalf(inverse(a)):
time()-t;
```

Eigenvalues of a normal distr. 600x600 randommatrix

```
a:=evalm(randmatrix(600,600)/100):
t:=time():
evalf(Eigenvals(a)):
time()-t;
```

Cholesky decomposition of a 1000x1000-matrix

```
S:=randmatrix(1000, 1000)/100:
A := evalf(evalm(transpose(S)&*S)):
t:=time():
evalf(cholesky(A)):
time()-t;
```

1000x1000 cross-product matrix

```
S:=randmatrix(1000, 1000)/100:
t:=time():
evalf(evalm(transpose(S)&*S)):
time()-t;
```

Number theory

Calculation of 500000 fibonacci numbers

```
with(combinat, fibonacci):
frnd:=rand(1..1000):
A := [seq(frnd(),i=1..500000)]:
t:=time():
evalm(fibonacci(A)):
time()-t;
```

Stochastic-statistic

Gamma function over a 1000x1000 matrix

```
a:=randmatrix(1000,1000):
b:=evalm(abs(a)):
t:=time():
evalf(evalm(gamma(b))):
time()-t;
```

Gaussian error function over a 1000x1000 matrix

```
A:=randmatrix(1000, 1000)/100:
t:=time():
evalf(evalm(erf(A))):
time()-t;
```

Linear regression over a 1000x1000 matrix

```
A:=evalf(evalm(randmatrix(10,1000))):
B:=[seq(rand(),i=1..1000)]:
t:=time():
leastsqrs(A,B):
time()-t;
```

Mathematica Benchmark program:

Comparison benchmarktest for number cruncher testreport
(Version 2)

Misc. operation

Creation, trans. & reshaping of a 1000x1000 matrix

```
Timing[Transpose[
Partition[Flatten[Transpose[Abs[Table[Random[Real],{1000},{1000}]]]],500]];]
```

1000x1000 random matrix ^ 1000

```
a = Table[Random[Real]/10, {1000}, {1000}];
Timing[(a + 1)^1000;]
```

2000000 values sorted in ascending order

```
a=Table[Random[Real],{2000000}];
Timing[Sort[a];]
```

Analysis

FFT over 1048576 random values

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

Double integration

```
Timing[NIntegrate[Sin[x^2+y^2],{x,-Pi,Pi},{y,-Pi,Pi}];]
```

Triple integration

```
Timing[NIntegrate[Sin[x^2+y^2+z^2],{x,-Pi,Pi},{y,-Pi,Pi},{z,-Pi,Pi}];]
```

Algebra

Determinant of a 1000x1000 random matrix

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

Creation of a 1400x1400 Toeplitzmatrix

```
Timing[Table[Abs[x-i]+1,{i,1,1400},{x,1,1400}];]
```

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 500000 fibonacci numbers

```
a=Table[Random[Integer,{100,1000}],{500000}];
Timing[Fibonacci[a];]
```

Stochastic & statistic

Gamma function over a 1000x1000 matrix

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

Gaussian error function over a 1000x1000 matrix

```
a=Table[Random[Real],{1000},{1000}];
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 2.0                        *
%***************************************************************

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 *

ergeb=0; a=0; b=0; zaehl=3;
for i=1:zaehl;
    tic;
    b=abs(randn(1000,1000)/10); a=b'; b=reshape(a,500,2000); a=b';
    zeit=toc;
    ergeb=ergeb+zeit;
end;
ergeb=ergeb/zaehl;
disp(['Creation, trans. & reshaping of a 1000x1000 matrix___ : ' num2str(ergeb) ' sec.'])
ergeb=0; a=0; zaehl=5;
for i=1:zaehl
    b=abs(randn(1000,1000)/2);
    tic; a=b.^1000; zeit=toc;
    ergeb=ergeb+zeit;
end
ergeb=ergeb/zaehl;
disp(['1000x1000 normal distributed random matrix^1000______ : ' num2str(ergeb) ' sec.'])
clear a; clear b; ergeb=0;
for i=1:zaehl
    a=randn(2000000,1);
    tic; b=sort(a); zeit=toc;
    ergeb=ergeb+zeit;
end
ergeb=ergeb/zaehl;
disp(['2000000 values sorted ascending______________________ : ' num2str(ergeb) ' sec.'])

%* Analysis *

clear a; clear b; ergeb=0;
for i=1:zaehl
    a=randn(1048576,1);
    tic; b=fft(a); zeit=toc;
    ergeb=ergeb+zeit;
end
ergeb=ergeb/zaehl;
disp(['FFT over 1048576 values______________________________ : ' num2str(ergeb) ' sec.'])
clear a; ergeb=0;
for i=1:zaehl
    tic; a=dblquad('fun',-pi,pi,-pi,pi,1.e-3,'quad8');zeit=toc;
    ergeb=ergeb+zeit;
end
ergeb=ergeb/zaehl;
disp(['Double integration___________________________________ : ' num2str(ergeb) ' sec.'])

%* Algebra *

clear a; clear b; ergeb=0;
for i=1:zaehl
    a=rand(1000,1000);
    tic; b=det(a); zeit=toc;
    ergeb=ergeb+zeit;
end
ergeb=ergeb/zaehl;
disp(['Determinant of a 1000x1000 random matrix_____________ : ' num2str(ergeb) ' sec.'])
clear a; clear b; ergeb=0;
for i=1:zaehl
    a=1:1400;
    tic; b=toeplitz(a); zeit=toc;
    ergeb=ergeb+zeit;
end
ergeb=ergeb/zaehl;
disp(['1400x1400 Toeplitzmatrix_____________________________ : ' num2str(ergeb) ' sec.'])
clear a; clear b; ergeb=0;
for i=1:zaehl
    a=rand(1000,1000);
    tic; b=inv(a); zeit=toc;
    ergeb=ergeb+zeit;
end
ergeb=ergeb/zaehl;
disp(['Inverse of a 1000x1000 uniform distr. random matrix__ : ' num2str(ergeb) ' sec.'])
clear a; clear b; ergeb=0;
for i=1:zaehl
    a=randn(600,600);
    tic; c=eig(a); zeit=toc;
    ergeb=ergeb+zeit;
end
ergeb=ergeb/zaehl;
```

```
disp(['Eigenval. of a normal distr. 600x600 randommatrix____ : ' num2str(ergeb) ' sec.'])
clear a; clear b; ergeb=0;
for i=1:zaehl
        a=randn(1000,1000);
        a=a'*a;
        tic; b=chol(a); zeit=toc;
        ergeb=ergeb+zeit;
end
ergeb=ergeb/zaehl;
disp(['Cholesky decomposition of a 1000x1000-matrix_________ : ' num2str(ergeb) ' sec.'])
clear a; clear b; ergeb=0;
for i=1:zaehl
        a=randn(1000,1000);
        tic; b=a'*a; zeit=toc;
        ergeb=ergeb+zeit;
end
ergeb=ergeb/zaehl;
disp(['1000x1000 cross-product matrix_______________________ : ' num2str(ergeb) ' sec.'])

%* Number theory *

clear a; clear b;
phi = 1.6180339887498949; ergeb=0;
for i=1:zaehl;
        a=floor(1000*rand(500000,1));
        Tic; b=(phi.^a-(-phi).^(-a))/sqrt(5); zeit=toc;
        ergeb=ergeb+zeit;
end
ergeb=ergeb/zaehl;
disp(['Calculation of 500000 fibonacci numbers______________ : ' num2str(ergeb) ' sec.'])

%* Stochastic-statistic *

clear a; clear b; ergeb=0;
for i=1:zaehl
        a=randn(1000,1000)^2;
        tic; b=gamma(a); zeit=toc;
        ergeb=ergeb+zeit;
end
ergeb=ergeb/zaehl;
disp(['Gamma function over a 1000x1000 matrix_______________ : ' num2str(ergeb) ' sec.'])
clear a; clear b; ergeb=0;
for i=1:zaehl
        a=randn(1000,1000)^2;
        tic; b=erf(a); zeit=toc;
        ergeb=ergeb+zeit;
end
ergeb=ergeb/zaehl;
disp(['Gaussian error function over a 1000x1000 matrix______ : ' num2str(ergeb) ' sec.'])
clear a; clear b; ergeb=0;
for i=1:zaehl
        a=randn(1000,1000); b=1:1000;
        tic; b=a\b'; zeit=toc;
        ergeb=ergeb+zeit;
end
ergeb=ergeb/zaehl;
disp(['Linear regression over a 1000x1000 matrix____________ : ' num2str(ergeb) ' sec.'])

function z = fun(x,y)
z = sin(x.^2+y.^2);
```

MuPAD Benchmark program:

```
HISTORY:=3:
M:=Dom::Matrix():
SEED:=1:
randn:= float@random(-10^10..10^10)/10^10:
randu:= float@random(0..10^10)/10^10:
```

Creation, trans. & reshaping of a 1000x1000 matrix

```
n:= 1000:
t0:=time():
a:= array(1..n,1..n,[[randn() $ i=1..n] $ j=1..n]):
a:= map(a,_mult,1/10):
a:= map(a, abs):
a:= array(1..n,1..n,[[a[j,i]$j=1..n]$i=1..n]):
N:=n/2: M:=2*n:
a:= array(1..N,1..M,[[a[i,j]$j=1..n, a[N+i,j]$j=1..n] $i=1..N]):
a:= array(1..M,1..N,[[a[j,i]$j=1..N]$i=1..M]):
t1:=time():
Total = (t1-t0)*msec;
```

1000 x 1000 random matrix^1000

```
n:=1000:
a:= array(1..n,1..n,[[randn()+1 $ j=1..n] $i=1..n]):
time(map(a,_power,1000))*msec;
```

Sorting of 2000000 random values

```
L:=[0 $ 2000000]:
L:=map(L,random(0..10^10)):
time(sort(L))*msec;
```

FFT over 1048576 random values

```
m:= 20:
a:=[0 $ 2^m]:
a:=map(a,(random(0..10^10))/1000):
time(fft(a,m))*msec;
```

Double integration

```
f:=numeric::quadrature:
time(f(f(sin(x^2+y^2),x=-PI..PI),y=-PI..PI))*msec;
```

Triple integration

```
time(f(f(f(sin(x^2+y^2+z^2),x=-PI..PI),y=-PI..PI),z=-PI..PI))*msec;
```

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))*msec;
```

Creation of a 1400 x 1400 Töplitzmatrix

```
toeplitz:= proc(Liste) local n, A, i, j, tmp;
    begin  n:= nops(Liste);
    A:=array(1..n,1..n);
    ((A[i,i+j-1]:= Liste[j]) $ j=1..n-i+1) $ i=1..n;
    ((A[i,j]:= A[j,i];) $ j=1..i-1) $ i=2..n;
    A;
end_proc:
n:= 1400:
L:= [i $ i=1..n]:
time(toeplitz(L))*msec;
```

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))*msec;
```

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))*msec;
```

Cholesky decomposition of a 1000 x 1000 random matrix

```
A:= M(1000,1000,float(random(0..100)/1000)):
A:=(linalg::transpose(A)*A);
time(linalg::cholesky(A))*msec;
```

Creation of a 1000 x 1000 cross-product matrix

```
A:= M(1000,1000,float(random(0..100)/1000)):
time(linalg::transpose(A)*A)*msec;
```

Calculation of 500000 fibonacci numbers

```
sqrt5:= float(sqrt(5)):
p:= float( (sqrt5+1)/2):
fib:= proc(n) option remember;
   begin (p^n-(-p)^(-n))/sqrt5 end_proc:
n:=500000:
a:= [0$n]:
a:= map(a, random(100..1000)):
time(map(a, fib))*msec;
```

Gamma function on a 1000 x 1000 random matrix

```
n:= 1000:
a:= array(1..n,1..n,[[randu() $ j=1..n] $i=1..n]):
time(map(a,gamma))*msec;
```

Gaussian error function over a 1000 x 1000 random matrix

```
n:= 1000:
a:= array(1..n,1..n,[[randu() $ j=1..n] $i=1..n]):
time(map(a,erf))*msec;
```

Linear regression over a 1000 x 1000 random matrix

```
linreg:=
   proc(A, b) local m,n,i,j,k;
   begin
      m:= op(A, [0,2,2]):
      n:= op(A, [0,3,2]):
      b:= array(1..n,[_plus(A[k,j]*b[k] $ k=1..m) $j=1..n]):
      A:= array(1..n,1..n,[[_plus(A[k,i]*A[k,j]$k=1..m)$j=1..n]$i=1..n]):
      A:= numeric::inverse(A):
      array(1..n,[_plus(A[j,k]*b[k] $ k=1..n) $j=1..n]):
   end_proc:

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(linreg(A, b))*msec;
```

O-Matrix Benchmark program:

```
# O-Matrix benchmarks corresponding the gauss benchmark from
# Stefan Steinhaus stst@informatik.uni-frankfurt.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

    # Misc. operation

    N = 1000
    t0=time
    a=abs(snormal(N,N)/10)
    a=a’
    dim a(500,2000)
    a=a’
    t1=time-t0;
    print "Creation, trans. & reshaping of a ",N,"x",N," matrix:",t1

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

    N = 2000000
    y=rand(N,1)
    t0 = time
    X = sort(y)
    t1 = time - t0
    print N, " values sorted ascending: ", t1

    # Analysis

    N = 1048576
    y=complex(rand(N,1))
    t0 = time
    X = fft(y)
    t1 = time - t0
    print N, " element FFT: ", t1
    # Double integration

    function fvec(x, yvec) begin
         return sin(x^2 + yvec^2)
    end
    function limy(x) begin
         y1   = -3.14159265358979324
         y2   = 3.14159265358979324
         ybnd = .1
         return {y1, y2, ybnd}
    end

    x1   = -3.14159265358979324
    x2   = 3.14159265358979324
    xbnd = .1
    limx = {x1, x2, xbnd}

    t0 =time
    y = gaussq2d(function fvec, limx, function limy)
    t1 = time - t0
    print "Double integration: ", t1

    # Algebra

    N = 1000
    x=rand(N,N)
    t0 = time
    y=det(x)
   t1 = time - t0
    print "Determinant of a ",N," x ",N," random matrix:",t1

    N = 1400
    x=identity(N)
    t0 = time
    for i=1 to N begin
        for y=1 to N begin
            x(y,i)=abs(i-y)+1
        end
    end
    t1 = time - t0
    print N, "x",N," Toeplitz matrix: ", t1

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

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

    N = 1000
    y=rand(N,N)
    y = y'*y
    t0 = time
    X = cholesky(y);
    t1 = time - t0
    print "Cholesky decompositon of ",N, "x",N, " matrix: ", t1

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

    # Number theory

    N = 500000
    phi = 1.6180339887498949
    a=floor(1000*rand(N,1))
    t0=time
    b=(phi^a-(-phi)^(-a))/sqrt(5.)
    t1=time - t0
    print "Calculation of ",N," 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 = 1000
    y=rand(N,N)^2
    t0 = time
    X = exp(gammln(y))
    t1 = time-t0
    print N, " x ", N, " gamma function: ", t1

    N = 1000
    y=rand(N,N)^2
    t0 = time
    X = erf(y)
    t1 = time - t0
    print N," x ", N, " gaussian error function", t1

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

Ox Benchmark program:

```
/*-------------------------------------------------------------------------*/
/* Ox Benchmark program version 2.0                          */
/*-------------------------------------------------------------------------*/

#include <oxstd.h>
Main()
{
      decl cRep = 3;                              /* number of repeats */
      decl i, time, a, b, c, secs,phi;
      print("number of repeats: ", cRep, " (timings in seconds)\n\n");
      Format("%8.3f");

      /* Misc. operation */

      for (i = secs = 0; i < cRep; i++)
      {
            time = timer();
            B=fabs(rann(1000,1000)/10); a=b';b=reshape(a,500,2000); a=b';
            secs += (timer() - time) / 100;
      }
      secs /= cRep;  delete a;  delete b;
      print("Creation, trans. & reshaping of a 1000x1000 matrix:          ", secs, "\n");

      for (i = secs = 0; i < cRep; ++i)
      {
            B=fabs(rann(1000,1000)/10);
            time = timer(); a = (b+1) .^ 1000; secs += (timer() - time) / 100;
      }
      secs /= cRep;  delete a;  delete b;
      print("1000x1000 random matrix .^ 1000:                          ", secs, "\n");

      for (i = secs = 0; i < cRep; ++i)
      {
            a = rann(1,2000000);
            time = timer(); b = sortr(a); secs += (timer() - time) / 100;
      }
      secs /= cRep;  delete a;  delete b;
      print("Sorting of 2,000,000 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 1,000,000 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 = range(1,1400);
            time = timer(); b = toeplitz(a,1400); secs += (timer() - time) / 100;
      }
      secs /= cRep;  delete a;  delete b;
      print("Creation of an 1400x1400 Toeplitz 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), Ox only has loggamma, so need to take exp(.)
for (i = secs = 0; i < cRep; ++i)
{
      b=rann(1000,1000).^2;
      time = timer(); a = gammafact(b); secs += (timer() - time) / 100;
}
secs /= cRep;  delete a;  delete b;
print("Gamma function on a 1000x1000 random matrix:               ", secs, "\n");

// use erf(x) = 2 * \Phi(x*sqrt(2)) - 1;
for (i = secs = 0; i < cRep; ++i)
{
      a = rann(1000,1000).^2;
      time = timer(); b = 2 * probn(a*sqrt(2)) - 1; secs += (timer() - time) / 100;
}
secs /= cRep;  delete a;  delete b;
print("Gaussian error function over a 1000x1000 random matrix:    ", secs, "\n");

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

R-Lab Benchmark program:

```
printf("\nThe following benchmark program will print the average timings\n");
printf("to calculate the functions by 3 runs.\n");
printf("\n\n");
printf("!!! SCILAB - Benchmarkprogram !!!\n");
printf("=================================\n\n");
format(32);
srand();

## Misc. operation ##

ergeb = 0;
zaehl = 3;
for (i in 1:zaehl)
{
  tic();
  b = abs(rand(1000,1000)/10);
  a = b’;
  b = reshape(a,500,2000);
  a = b’;
  zeit = toc();
  ergeb = ergeb+zeit;
}
ergeb = ergeb/zaehl;
printf("Creation, trans. & reshaping of a 1000x1000 matrix (sec.)___: %f\n",ergeb);
ergeb = 0;
for (i in 1:zaehl)
{
  b = abs(1+rand(1000,1000)/100);
  tic();
  a = b.^1000;
  zeit = toc();
  ergeb = ergeb+zeit;
}
ergeb = ergeb/zaehl;
printf("1000x1000 normal distributed random matrix^1000 (sec.)______: %f\n",ergeb);
ergeb = 0;
for (i in 1:zaehl)
{
  a = rand(2000000,1);
  tic();
  b = sort(a).val;
  zeit = toc();
  ergeb = ergeb+zeit;
}
ergeb = ergeb/zaehl;
printf("2000000 values sorted ascending (sec.)______________________:  %f\n",ergeb);

## Analysis ##

ergeb = 0;
for (i in 1:zaehl)
{
  a = rand(1048576,1);
  tic();
  b = fft(a);
  zeit = toc();
  ergeb = ergeb+zeit;
}
ergeb = ergeb/zaehl;
printf("FFT over 1048576 values (sec.)______________________________:  %f\n",ergeb);

printf("Double integration (sec.)___________________________________  : n.a.\n");
printf("Triple integration (sec.)___________________________________  : n.a.\n");

## Algebra ##

ergeb = 0;
for (i in 1:zaehl)
{
  a = rand(1000,1000)/5;
  tic();
  b = det(a);
  zeit = toc();
  ergeb = ergeb+zeit;
}
ergeb = ergeb/zaehl;
printf("Determinant of a 1000x1000 random matrix (sec.)_____________: %f\n",ergeb);
ergeb = 0;
for (k in 1:zaehl)
{
       tic();
       x = eye(1400,1400);
       for (i in 1:1400)
       {
          for (j in 1:1400)
          {
             x[j;i]=abs(i-j)+1;
          }
       }
       zeit = toc();
       ergeb = ergeb+zeit;
}
```

```
ergeb = ergeb/zaehl;
printf("1400x1400 Toeplitzmatrix (sec.)______________________________:  %f\n",ergeb);
ergeb = 0;
for (i in 1:zaehl)
{
  a = rand(1000,1000);
  tic();
  b = inv(a);
  zeit = toc();
  ergeb = ergeb+zeit;
}
ergeb = ergeb/zaehl;
printf("Inverse of a 1000x1000 uniform distr. random matrix (sec.)__: %f\n",ergeb);
ergeb = 0;
for (i in 1:zaehl)
{
  a = rand(600,600);
  tic();
  c = eig(a).val;
  zeit = toc();
  ergeb = ergeb+zeit;
}
ergeb = ergeb/zaehl;
printf("Eigenval. of a normal distr. 600x600 randommatrix (sec.)____: %f\n",ergeb);
ergeb = 0;
for (i in 1:zaehl)
{
  a = rand(1000,1000);
  a = a'*a;
  tic();
  b = chol(a);
  zeit = toc();
  ergeb = ergeb+zeit;
}
ergeb = ergeb/zaehl;
printf("Cholesky decomposition of a 1000x1000-matrix (sec.)_________: %f\n",ergeb);
ergeb = 0;
for (i in 1:zaehl)
{
  a = rand(1000,1000);
  tic();
  b = a'*a;
  zeit = toc();
  ergeb = ergeb+zeit;
}
ergeb = ergeb/zaehl;
printf("1000x1000 cross-product matrix (sec.)_______________________:  %f\n",ergeb);
```

```
## Number theory ##

phi = 1.61803398874989;
ergeb = 0;
for (i in 1:zaehl)
{
  a = floor(1000*rand(500000,1));
  tic();
  b = ceil((phi.^a-(-phi).^(-a))/sqrt(5));
  zeit = toc();
  ergeb = ergeb+zeit;
}
ergeb = ergeb/zaehl;
printf("Calculation of 500000 fibonacci numbers (sec.)______________: %f\n",ergeb);

## Stochastic-statistic ##

printf("Gamma function over a 1000x1000 matrix (sec.)_______________: n.a.\n",ergeb);
printf("Gaussian error function over a 1000x1000 matrix (sec.)______: n.a.\n",ergeb);
ergeb = 0;
for (i in 1:zaehl)
{
  b = zeros(1000,1)+(1:1000)';
  for (j in 1:4) {b[j;1]=j;}
  a = rand(1000,1000);
  b = 1:1000;
  tic();
  b = a\b';
  zeit = toc();
  ergeb = ergeb+zeit;
}
ergeb = ergeb/zaehl;
printf("Linear regression over a 1000x1000 matrix (sec.)____________: %f\n",ergeb);
```

SciLab Benchmark program:

```
stacksize(8000000)

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(' ');

//* Misc. operation *

ergeb = 0; a = 0; b = 0; zaehl = 1;
for i = 1:zaehl
    timer(); b = abs(rand(1000,1000,'n')/10);a = b'; b = matrix(a,500,2000); a = b'; zeit = timer();
    ergeb = ergeb+zeit;
end
ergeb = ergeb/zaehl;
disp('Creation, trans. & reshaping of a 1000x1000 matrix___: '+string(ergeb)+'sec.');
ergeb = 0; a = 0;
for i = 1:zaehl
    b = abs(rand(1000,1000,'n')/2);
    timer(); a = b.^1000; zeit = timer();
    ergeb = ergeb+zeit;
end
ergeb = ergeb/zaehl;
disp('1000x1000 normal distributed random matrix^1000______: '+string(ergeb)+'sec.');
clear('a');clear('b');ergeb = 0;
for i = 1:zaehl
    a = rand(2000000,1,'n');
    timer();  b = -sort(-a); zeit = timer();
    ergeb = ergeb+zeit;
end
ergeb = ergeb/zaehl;
disp('2000000 values sorted ascending______________________:  '+string(ergeb)+'sec.');

//* Analysis *

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

clear('a');ergeb=0;
x=[-%pi,-%pi;%pi,%pi;%pi,-%pi];  y=[-%pi,-%pi;-%pi,%pi;%pi,%pi];
deff('z=f1(x,y)','z=sin(x^2+y^2)')
for i=1:zaehl
    timer(); a=int2d(x,y,f1); zeit=timer();
    ergeb=ergeb+zeit;
end
ergeb=ergeb/zaehl;
disp('Double integration___________________________________  : '+string(ergeb)+'sec.')

clear('a');ergeb=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('v=f2(xyz,numfun)','v=sin(xyz''*xyz)')
for i=1:zaehl
    timer(); [RESULT,ERROR]=int3d(x,y,z,f2,1,[0, 500000, 0.01, 1.d-3]); result=2*RESULT;
    zeit=timer();
    ergeb=ergeb+zeit;
end
ergeb=ergeb/zaehl;
disp('Triple integration___________________________________  : '+string(ergeb)+'sec.')

//* Algebra *

clear('a');clear('b');ergeb = 0;
for i = 1:zaehl
    a = rand(1000,1000,'u');
    timer(); b = det(a); zeit = timer();
    ergeb = ergeb+zeit;
end
ergeb = ergeb/zaehl;
disp('Determinant of a 1000x1000 random matrix_____________:  '+string(ergeb)+'sec.');
clear('a');clear('b');ergeb = 0;
for i = 1:zaehl
    a = 1:1400;
    timer(); b = toeplitz(a); zeit = timer();
    ergeb = ergeb+zeit;
end
ergeb = ergeb/zaehl;
disp('1400x1400 Toeplitzmatrix_____________________________:  '+string(ergeb)+'sec.');
clear('a');clear('b');ergeb = 0;
for i = 1:zaehl
    a = rand(1000,1000,'u');
    timer(); b = inv(a); zeit = timer();
    ergeb = ergeb+zeit;
end
ergeb = ergeb/zaehl;
disp('Inverse of a 1000x1000 uniform distr. random matrix__: '+string(ergeb)+'sec.');
clear('a');clear('b');ergeb = 0;
for i = 1:zaehl
    a = rand(600,600,'n');
    timer(); c = spec(a); zeit = timer();
    ergeb = ergeb+zeit;
end
ergeb = ergeb/zaehl;
disp('Eigenval. of a normal distr. 600x600 randommatrix____: '+string(ergeb)+'sec.');
clear('a');clear('b');ergeb = 0;
for i = 1:zaehl
    a = rand(1000,1000,'n');a = a'*a;
    timer(); b = chol(a); zeit = timer();
    ergeb = ergeb+zeit;
end
ergeb = ergeb/zaehl;
disp('Cholesky decomposition of a 1000x1000-matrix_________: '+string(ergeb)+'sec.');
clear('a');clear('b');ergeb = 0;
for i = 1:zaehl
    a = rand(1000,1000,'n');
    timer(); b = a'*a; zeit = timer();
    ergeb = ergeb+zeit;
end
ergeb = ergeb/zaehl;
disp('1000x1000 cross-product matrix________________________: '+string(ergeb)+'sec.');

//* Number theory *

clear('a');clear('b');phi = 1.61803398874989; ergeb = 0;
for i = 1:zaehl
    a = floor(1000*rand(500000,1,'u'));
    timer(); b = (phi.^a-(-phi).^(-a))/sqrt(5); zeit = timer();
    ergeb = ergeb+zeit;
end
ergeb = ergeb/zaehl;
disp('Calculation of 500000 fibonacci numbers______________: '+string(ergeb)+'sec.');

//* Stochastic-statistic *

clear('a');clear('b');ergeb = 0;
for i = 1:zaehl
    a = rand(1000,1000,'n')^2;
    timer(); b = gamma(a(:)); zeit = timer();
    ergeb = ergeb+zeit;
end
ergeb = ergeb/zaehl;
disp('Gamma function over a 1000x1000 matrix_______________: '+string(ergeb)+'sec.');
clear('a');clear('b');ergeb = 0;
for i = 1:zaehl
    a = rand(1000,1000,'n')^2;
    timer(); b = erf(a(:)); zeit = timer();
    ergeb = ergeb+zeit;
end
ergeb = ergeb/zaehl;
disp('Gaussian error function over a 1000x1000 matrix______: '+string(ergeb)+'sec.');
clear('a');clear('b');ergeb = 0;
for i = 1:zaehl
    a = rand(1000,1000,'n');b = 1:1000;
    timer(); b = a\b'; zeit = timer();
    ergeb = ergeb+zeit;
end
ergeb = ergeb/zaehl;
disp('Linear regression over a 1000x1000 matrix____________: '+string(ergeb)+'sec.');
```

S-Plus Benchmark program:

```
print("!!! S-Plus - Benchmarkprogram !!!")
print("================================")
print
library(Matrix)
options(object.size=100000000)

## Misc. operation ##

print("Creation, trans. & reshaping of a 1000x1000 matrix (sec.) : ")
print(dos.time(t(matrix(t(abs(matrix(rnorm(1000*1000),ncol=1000,nrow=1000))),ncol=500,nrow=2000))
))
print("1000x1000 random matrix^1000 (sec.) : ")
x<-Matrix(runif(1000000),1000,1000)
Matrix.class(x)
print(dos.time(x^1000))
print("2000000 values sorted ascending (sec.) : ")
x<-runif(2000000)
print(dos.time(sort(x)))

## Analysis ##

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

## Algebra ##


print("Determinant of a 1000x1000 random matrix (sec.) : ")
determinant <- function(x) prod(eigen(x)$values)
x<-Matrix(runif(1000000),1000,1000)
Matrix.class(x)
print(dos.time(determinant(x)))
print("1400x1400 Toeplitzmatrix (sec.) :")
"top" <-
function (n = 4)
{
  mat <- matrix(ncol = n, nrow = n, byrow = F)
      mat[, 1] <- 1:n
      mat[, n] <- n:1
      for (i in seq(2, n - 1)) mat[, i] <- c(seq(i, 1), seq(2, n - i + 1))
      mat
}
print(dos.time(top(1400)))
print("Inverse of a 1000x1000 uniform distr. random matrix (sec.) :")
x<-Matrix(runif(1000000),1000,1000)
Matrix.class(x)
print(dos.time(solve(x)))
print("Eigenval. of a normal distr. 600x600 randommatrix (sec.) :")
x<-Matrix(runif(360000),600,600)
Matrix.class(x)
print(dos.time(eigen(x, vectors = F)$Value))
print("Cholesky decomposition of a 1000x1000-matrix (sec.) :")
x<-Matrix(runif(1000000),1000,1000)
x<-crossprod(x,x)
Matrix.class(x)
print(dos.time(chol(x)))
print("1000x1000 cross-product matrix (sec.) :")
x<-Matrix(runif(1000000),1000,1000)
Matrix.class(x)
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("Gamma function over a 1000x1000 matrix (sec.) :")
x<-Matrix(runif(1000000),1000,1000)
Matrix.class(x)
print(dos.time(gamma(x)))
print("Gaussian error function over a 1000x1000 matrix (sec.) :")
print("Linear regression over a 1000x1000 matrix (sec.) :")
x<-Matrix(runif(1000000),1000,1000)
Matrix.class(x)
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:** Gordon Stone, Aptech Systems Inc. (USA)
- **Macsyma:** Robert H. Berman, Macsyma Inc. (USA)
- **Macsyma:** Jeffrey P. Golden, Macsyma Inc. (USA)
- **Macsyma:** Richard J. Petti, Macsyma Inc. (USA)
- **Maple:** Andreas Himmeldorf, Scientific Computers GmbH (Germany)
- **Maple:** Ralf Kraume, Visual Analysis GmbH (Germany)
- **Mathematica:** Brett H. Barnhart, Wolfram Research Inc. (USA)
- **Mathematica:** Roger Germundsson, Wolfram Research Inc. (USA)
- **Mathematica:** Cliff Hastings, Wolfram Research Inc. (USA)
- **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:** Stefan Wehmeier, 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)
- **Scilab:** Dr. Scilab, INRIA (France)
- **Scilab:** Serge Steer, INRIA (France)
- **S-Plus:** Sven Bevermann, MathSoft Inc. (Germany)
- **S-Plus:** David Smith, MathSoft Inc. (England)
- **S-Plus:** Reinhard Sy, GraS GmbH (Germany)