Abstract

In this document, we present a proposal for a mathematical roadmap for Scilab and analyse the features of Scilab which may require an update. We begin the analysis by a complete, but brief, overview of the computational features of Scilab. We also describe (a subset of the) missing mathematical features in Scilab. Emphasis is on high level features and not on detailed specifications, which are managed by Scilab Enhancement Proposals (SEPs). In the conclusion, we try to select some developments and projects which appear to be reasonable choices.

Contents

1 Introduction 5
  1.1 Features vs use cases .............................................. 5
  1.2 What for? ......................................................... 7
  1.3 The current state .................................................. 8
  1.4 Other softwares ................................................... 8
  1.5 Benchmarking speed ................................................. 9
  1.6 Benchmarking accuracy ............................................. 9
  1.7 What not to do ..................................................... 10
  1.8 Users and features ............................................... 10
  1.9 Methodology used in this document ................................. 10

2 Overview of Scilab 11
  2.1 Basics ............................................................ 12
      2.1.1 Features ..................................................... 12
      2.1.2 Libraries ................................................... 13
      2.1.3 Strong points ................................................. 13
      2.1.4 Issues ....................................................... 13
  2.2 Classics in numerical analysis ................................. 14
      2.2.1 Features ..................................................... 14
      2.2.2 Libraries ................................................... 17
      2.2.3 Strong points ................................................. 17
      2.2.4 Issues ....................................................... 17
## 12 Geometry

12.1 Features .................................................. 29
12.2 Suggested libraries to connect to Scilab ................. 29
12.3 Existing features in Matlab ............................... 29
12.4 Existing tools which may require an update .......... 29
12.5 Why it fits in Scilab scope ............................. 29

## 13 Partial Differential Equations

13.1 Why it fits in Scilab scope .............................. 30
13.2 Existing features in Matlab ............................. 30
13.3 Suggested library to connect ............................ 30

## 14 Neural Networks

14.1 Existing features in Matlab ............................. 30
14.2 Suggested libraries to connect .......................... 31
14.3 Discussion ................................................. 31

## 15 Others .................................................... 31

## 16 Conclusion ................................................ 31

## 17 Acknowledgments ........................................ 33

Bibliography ..................................................... 33

Index ............................................................ 35
1 Introduction

The goal of this document is to give an overview of the future possible computational (mathematical) developments in Scilab. Two main types of possible evolutions are analyzed:

- update of existing features,
- creation of new features.

This document is an open-source project. The \LaTeX sources are available on the Scilab Forge:

http://forge.scilab.org/index.php/p/docmathroadmap

The \LaTeX sources are provided under the terms of the Creative Commons Attribution-ShareAlike 3.0 Unported License:

http://creativecommons.org/licenses/by-sa/3.0

We emphasize that this document is an exploration of possible developments. It is not a definitive planning for these developments. It is draft and may remain a dynamical draft depending on the new developments made in Scilab. Hence, if you feel that the proposals suggested in this document do not match your particular needs, please contact the authors and share your point of view.

In this introduction, we analyze the context of the computational features in Scilab. We begin by presenting the reasons for the update or the creation of new computational features in Scilab. Then we present the difficulties raised by the complexity of Scilab, which makes it difficult to have an overview of its current state. We briefly present a list of commercial or open source softwares which may serve as a comparison with Scilab and present resources which provide benchmarks for speed or accuracy of numerical softwares. In the final section, we present the methodology used in this document.

1.1 Features vs use cases

It is completely impossible to discuss a mathematical roadmap without analysing the whole purpose of Scilab. As any other software, Scilab provide features which allow to solve use cases.

This is presented in figure 1, where the two dimensions of this situation are detailed. One the computational features axis, we find, for example, elementary functions, linear algebra, control, ordinary differential equations (ODE), optimization, statistics, control and many other features available in Scilab. On the use case axis, we find the automotive industry, aerospace, chemistry, finance and many other application domains for Scilab.

Historically, the first features available in Matlab, the ancestor of Scilab, were linear algebra, driven by the will to use linear algebra libraries such as Lapack and Eispack. Then, engineers realized that this was a tremendous tool for Control, and add the associated features, plus ordinary differential equations. The Simulink
tool and its Scilab counterpart, XCos, were created for that particular purpose. Afterwards, engineers in other fields found that it was possible to use this tool for many other use cases and asked for new features such as optimization and statistics. All in all, use cases are driving the need for computational features in Scilab, under the condition that these features can be easily provided by Open Source libraries.

The question that this document tries to analyze is the "..." presented in the figure 1 in both dimensions. More specifically, we try to find

- new use cases,
- new features or an update of existing ones.

Indeed, it may happen that existing features have performance or accuracy limitations. In some cases, existing features are not allowing to completely solve a particular use case.

This situation is presented in the figure 2, where the round shapes represent the areas defined by existing features, the "U" symbols represent existing features and the question marks "?" represent the new features to be created.

The need for update comes in two flavors: sometimes only a part of the feature is to be updated, while sometimes the whole component is to be replaced. For example, the ODE solver in Scilab is based on OdePack, which is efficient, but may be completely replaced by the up-to-date Sun Dials library. In some cases, there is a complete domain which is not covered by Scilab. For example, there is currently no component to solve systems of partial differential equations associated with conservation laws (that is, non linear hyperbolic systems). In some other cases,
there are existing features which are almost completely filling a domain, but not exactly. For example, Scilab provide iterative sparse linear algebra solvers, but not the associated preconditioners. There may be situations where there are a small number of existing features which fills basic situations, but more advanced needs are unsatisfied. For example, the current statistics features are existing in Scilab, but there are many more advanced statistics features which are missing.

1.2 What for?

There may be a variety of reasons for the update or the creation of new features in Scilab.

- A feature is completely missing. For example, there is currently no nonlinear optimization solver with non linear constraints. This feature is provided by the fmincon function in Matlab, for example.

- Accuracy: the feature exists in Scilab, but its accuracy is poor. For example, we may wonder why the value of $\sin(2^b4)$ is not the same in Scilab on all operating systems.

- Speed: the feature exists in Scilab, but it is slow. For example, the sparse linear algebra "lufact" function is slow, compared to the sparse LU decomposition provided in the UMFPACK module of Scilab.

- Documentation: the feature exists, but is so poorly documented that it is, eventually, unusable. For example, the previous versions of Scilab did not provide enough examples for the nonlinear unconstrained "optim" function. The
tolerances for this functions still lack of documentation, because we actually
do not know much about their actual use in the source code.

- Bugs / limitations / lack of tests : the feature exists, but has bugs. For example, the bug #7101 shows that there is a problem with the use of the "roots" function, depending on the operating system. For example, the bug #7569 shows that several inverse distribution functions are associated with a limited precision of 8 digits (instead of the upper limit 17 digits).

1.3 The current state

In order to suggest new features, it is (obviously) necessary to have the knowledge of the existing features. This is much more complicated that it looks.

For example, it is not straightforward to fully understand the features available in Scilab with respect to sparse linear algebra, or optimization. The reason of this problem is that several authors have developped several tools, each one devoted to a particular feature. This work has not been systematically been made consistent with existing features.

Moreover, a large amount of features is available in toolboxes. These toolboxes cover many computational areas and are developped by many authors, most of them being outside the Consortium. The toolboxes do not always provide the features missing in Scilab (in some cases, the features are the same). Another issue is that the toolboxes are not always maintained by their authors, which leads to packages which do not work with the latests versions of Scilab.

For example, Scilab provides iterative sparse linear algebra solvers, but do not provide preconditionners. Still, preconditionners are available in the Scilin toolbox, but this toolbox has not been maintained and does not work for the latests versions of Scilab.

As another example, Scilab provides several optimization features. But there is also a large number of features which are available in toolboxes. The simple task of managing the list of existing optimization features available either in Scilab or in toolboxes is complicated. This is why the document "Optimization in Scilab" [15] has been written. Obviously, the current work would be much more easy if such a document was available for each domain covered by Scilab.

1.4 Other softwares

Updating existing computational features or integrating new features in Scilab may be driven by comparisons with other softwares.

As expressed by Alan Edelman, we can first focus on the 800 pound gorilla, Matlab. One of the reason of the success of Matlab is the number of features which are available in this tool. Moreover, Cleve Moler is a scientific leader in applied mathematics. This is why, in the next section, we analyze the features provided by Matlab in each domain.

There are other softwares that we can consider for comparison purposes, including two major open source projects :
• Octave
• R : The R Project for Statistical Computing
• Gauss : Aptech
• LabVIEW : National Instruments

Alternatively, we can also consider the open source project FreeMat : http://freemat.sourceforge.net/

In practice, some features in Scilab can often be compared with the features provided by spreadsheet programs. Indeed, many users are making use of both types of softwares (that is, interpreted languages such as Scilab, or spreadsheet programs such as OpenOffice) during the same work session or inside a workflow. In the case where the two softwares provide exactly the same features, such as generating random numbers, or computing a distribution function, the accuracy of the softwares can be compared. This topic is covered is the next sections.

In order to suggest to update existing features or to introduce new ones, we may get some help from benchmarking Scilab. The most common type of benchmark is measuring the performance. But some authors also measure the quality of the documentation, the graphics, etc... One point which is specific to numerical softwares is the accuracy of their computations. These topics are covered in the next sections.

1.5 Benchmarking speed

Stefan Steinhaus did a deep comparison of mathematical programs for data analysis. The work has been sustained from 1997 to 2008. He reported his work at : http://www.scientificweb.com/ncrunch/ and the latest version (2008) of his benchmark is available in [23].

His benchmark included Scilab and numerical and symbolic commercial softwares, such as Gauss, Maple, Mathematica, Matlab, O-Matrix and OxMetric. He considered the cost (in $ or EUR), the installation, learnability, usability, math. features, graphics, data management, portability and speed.

There are other benchmarks as well :

• Benchmark by Philippe Grosjean in 2003 and its older version Benchmark #1

• Simple Benchmarks for Matlab and Clones by Derek O’Connor in 2006

1.6 Benchmarking accuracy

It may happen that there is a trade-off between speed and accuracy. This is why authors like Edelman or Mc Cullough considered the accuracy of softwares like Scilab, Matlab or Excel.

In ”The accuracy of statistical distributions in Microsoft Excel 2007” - A. Talha Yalta, Computational Statistics and Data Analysis 52 (2008) 4579-4586, the author provides an assessment of the statistical distributions in Microsoft Excel versions 97 through 2007 along with two competing spreadsheet programs, namely Gnumeric 1.7.11 and OpenOffice.org Calc 2.3.0. He finds that the accuracy of various statistical
functions in Excel 2007 range from unacceptably bad to acceptable but significantly inferior in comparison to alternative implementations. In particular, for the binomial, Poisson, inverse standard normal, inverse beta, inverse student’s t, and inverse F distributions, it is possible to obtain results with zero accurate digits as shown with numerical examples.

Other references on this topic are provided in Contributor - stats.

In "An Analysis of Scientific Computing Environments: A Consumer’s View" by Amit Soni [22], the author compare commercial and free open source languages and softwares. The author compares various simple and complex problems and how they can be approached in various languages. This work is done in the context of the Numerical Mathematical Consortium. The analysis addresses some incompatibility issues within languages and their impact. This work is continued by Alan Edelman and presented in "Open Source and Traditional Technical Computing" by Alan Edelman [20].

1.7 What not to do

Symbolic computations were perceived as being out of the scope of Scilab, which focuses mainly on numerical computations. Whether this choice should remain in the next years is subject to discussion.

TODO: finish that section.

1.8 Users and features

The figure 3 presents a rather abstract view of users compared the features of Scilab. Obviously, all users make use of the algebra of matrices, may be without even noticing it. Indeed, many expression in Scilab, such as a*b for example, are involving matrices. Hence, it is not technically possible to remove the linear algebra features of Scilab, because they are at the core of the design of the language. By contrast, not all users are interested by signal processing or by XCos.

What features would users want in the coming years and for what use cases? Anyone willing to share his comments is invited to contact the authors of this report.

1.9 Methodology used in this document

For each mathematical problem, we analyze the following points:

• what is the current state of Scilab?
• what features are to provide?
• why it fits in Scilab scope?
• is there existing library to connect to Scilab?
• is there a Scilab toolbox providing the feature?
• does Matlab provide an equivalent feature?
Detailed feature requests should be written as "Scilab Enhancement Proposals" and not in this document.

When a library is suggested for connection, we pay attention to at least the following points:

- licence: is the license authorizes the inclusion in Scilab?
- language: is the language compatible with Scilab?
- mathematical algorithms: is the algorithm efficient, robust, fast, up-to-date?
- documentation: is the library correctly documented with respect to Scilab requirements?
- tests: is the library tested?

2 Overview of Scilab

In this section, we try to make a complete overview of Scilab. The goal of this section is to give a brief description of all the computational features of Scilab and to emphasize the main design choices and the main issues.

We separate three levels of features in Scilab, each associated with a set of functions.

- Basics: the features used by all users,
- Classics in numerical analysis: the features used by many users,
- Advanced methods: the features used by some users, but obviously not all.
For each set of functions, we make a brief overview of:

- the computational features in this set,
- the libraries which supports the features,
- the strong points of Scilab in this set,
- the issues.

In this section, we choose a style which is the one usually practiced in slides rather than in technical reports: this is on purpose, in order to produce a shorter document.

2.1 Basics

In this section, we present the features which cannot be removed from Scilab. We consider that 100% of the users actually use all these features. These features are typical in interpreted languages such as Matlab, Octave, Python (SciPy) or even in symbolic computational tools such as Maxima, Mathematica or Maple.

This section is partially based on the presentation and analysis presented in [9, 11, 13, 7, 8, 12, 10, 14, 17, 16].

2.1.1 Features

- Matrix algebra: +, -, *, ./ Typical statement is $C=A*B$ where $A, B$ are real or complex matrices.

- Elementary functions: $\cos, \sin, \exp, \log$, power ($^\cdot$)... Typical statement is $y=\sin(x)$ where $x$ is real or complex matrix.

- Matrix functions: $\expm, \pow, \logm, \cosm$, etc...

- Special functions: $\gamma, \beta, \erf$, etc...

- Dense linear algebra: $\backslash, lu, \spec, \svd, \chol$, least squares Typical statement is $x=A^b$ where $A,b$ are real or complex matrix.

- Floating point number functions: $%\eps, \frexp, \numprop, \ieee, \nearfloat$

- Test matrices: $\text{testmatrix}$ (Franck, Hilbert, Magic squares), Toeplitz

- Discrete maths: $\text{factorial}, \text{permute}, \text{perms}, ...$
2.1.2 Libraries

The libraries which are providing the basic computational features in Scilab are the following.

- BLAS [2]
- LAPACK [3]
- SLATEC [19]: e.g. bessel
- SPECFUN [18]: e.g. calerf
- macros: e.g. erfinv
- Fortran (e.g. complex functions such as complex tan)

2.1.3 Strong points

In this section, we describe the main advantages of using the basic features of Scilab.

- Optimized BLAS optimized implementations:
- As a consequence, Scilab can be faster than what a typical user could easily make, even in a compiled language such as C or Fortran: vectorization provides the performance, at the library level.
- Rich set of elementary functions: \( \text{sind} \) (degree), \( \text{sinh} \) (hyperbolic).
- Matrix language: simple, fast with vectorized statements.

2.1.4 Issues

In this section, we present the main weak points with Scilab.

- The lack of a Just In Time component in the interpreter forces the user to write get efficiency from vectorization only.
- Missing functions: \( \text{hypot} \), \( \text{norm2} \), \( \text{expm1} \), \( \text{tan2} \), ... The specfun Forge project [6] tries to solve this issue (project unfinished yet).
- No int64.
- No multi-core sin.
- BLAS/LAPACK not used everywhere: Linpack and Eispack still in the sources.
- No direct use of ATLAS on Linux: still possible to use Debian ATLAS, packaged by S.L.
• Portability issue: we use the math. library (sin, cos, exp, etc...) of the compiler (gcc on Linux, Visual on Windows). We should use the same portable elementary functions library on all systems, e.g. fdlibm [24].

• Partial lack of floating point support: not all functions manage \%inf or \%nan, no way to set IEEE rounding mode, the \texttt{format} function does not display the number of digits associated with the precision of doubles.

• Partial lack of unit tests of elementary, special functions, including accuracy.

• Missing test matrices: Wilkinson, Ding-Dong, VanDerMonde, ... The Forge/Atoms testmatrix project [5] tries to solve this issue (project unfinished yet).

• No support for numerical testing. The Forge/Atoms assert project [4] tries to solve this issue.

2.2 Classics in numerical analysis

In this section, we focus on features which could technically be removed from Scilab, but make this tool interesting. At least 50% of the users actually use at least one of these features daily. These features are classics in numerical analysis and are typically found in Matlab, Octave, R or in numerical libraries such as Numerical Recipes, IMSK, NAG, GSL.

2.2.1 Features

In this section, we describe the following features of Scilab:

• Differential equations,

• Integration,

• Nonlinear equations,

• Optimization,

• Probabilities / Statistics,

• Interpolation.

The following is an overview the functions in these classic fields of computational mathematics.

• Differential equations : \texttt{ode}, \texttt{bvode}, \texttt{dassl}, etc... Computes the solution $y(t)$ of $dy(t)/dt = f(y)$ with $x$ real vector. This module includes a collection of algorithms:
  
  - Stiff problems : Backward Differentiation Formula (BDF) from Odepack,
  
  - Non-stiff problems : Adams method from Odepack,
  
  - Adaptive Runge-Kutta of order 4 (RK4),
- Shampine and Watts program based on Fehlberg’s Runge-Kutta pair of order 4 and 5,
- Root finding from Odepack,
- Discrete time simulation.

This module can manage the following type of right hand side of the first order differential equation:

- Macros (functions) with header \( ydot = f(t,y) \), with, if required, optional arguments.
- Fortran or C dynamically linked functions with header \( \text{fex}(\text{int } \ast n, \text{double } \ast t, \text{double } \ast y, \text{double } \ast ydot) \).

• Integration: \( \text{integ}, \text{integrate} \) (macro for string expressions)
  - Computes \( y = \int_{a}^{b} f(x) \, dx \) with \( x \) real scalar and \( f \) a function.
  - Typical use: \( I = \text{intg}(a, b, f) \) where \( y = f(x) \).
  - The \text{intg} function can manage macros (functions) or Fortran or C dynamically linked functions.

• Nonlinear equations: \( \text{fsolve} \).
  - Computes \( x \) real vector such that \( f(x) = 0 \) with \( f \) a multivariable function.
  - Typical use: \( x = \text{fsolve}(x0, f) \) where \( y = f(x) \).
  - The \text{fsolve} function can manage macros (functions) or Fortran or C dynamically linked functions.
  - The Jacobian can be provided, but is optional.

• Nonlinear unconstrained or bound constrained optimization (needs derivatives): \( \text{optim} \)
  - Typical use: \( [f_{opt}, x_{opt}] = \text{optim}(f, x0) \) where \( [f, g, \text{ind}] = f(x, \text{ind}) \).
  - The \text{optim} function can manage macros (functions) or Fortran or C dynamically linked functions.
  - The \text{derivative} function computes the gradient of the objective function by finite differences.

• Other Optimisation solvers
  - Nonlinear least squares (needs derivatives): \( \text{leastsq}, \text{lsqrsolve} \)
  - Nonlinear objective, without constraints or with bounds (does not need derivatives): \( \text{fminsearch} \)
  - Linear, quadratic optimization: \( \text{qld}, \text{karmarkar}, \text{qpsolve}, \text{qp_solve} \)
  - Global optimisation: genetic algorithms \( \text{optim_ga}, \text{simulated annealing optim_sa} \)
- Semidefinite programming: `semidef`
- Linear Matrix Inequality Solver: `lmisolver`

- Probability / Statistics
  - 6 Uniform Random Number generators: `u = grand(1000,5,"def")`
  - Including Mersenne Twister by Matsumoto and Nishimura: large period about $2^{19937}$ so that any simulation is guaranteed to use only a small fraction of the overall period.
  - 14 Non-uniform Random Number Generators: Normal, Exponential, Poisson, Geometric, etc...
  - Typical use: `Y=grand(m,n,"nor",av,sd)`
  - 11 Cumulated Distribution Functions and their inverse: Normal, Exponential, Beta, etc...
  - Typical use is `[P,Q]=cdfnor("PQ",X,Mean,Std)`.
  - Provides $P,Q$ for increased accuracy for small probabilities.
  - Many other functions: `mean, variance, st_deviation, pca`, etc...

- Interpolation.
  - 15 functions for 1D, 2D or 3D interpolation
  - linear interpolation with `interpln`
  - spline interpolation with `splin`

We emphasize that several toolboxes are available which are useful complements to the built-in functions.


- the Forge/Atoms Low Discrepancy Sequences Toolbox [http://forge.scilab.org/index.php/p/lowdisc/](http://forge.scilab.org/index.php/p/lowdisc/) which allows to create sequences such as Halton, Sobol, etc...
2.2.2 Libraries

The libraries which are providing the classic computational features in Scilab are the following.

- Quadpack: http://www.netlib.org/quadpack, a package for the numerical computation of definite one-dimensional integrals.

- Odepack: http://www.netlib.org/odepack, a collection of Fortran solvers for the initial value problem for ordinary differential equation systems.


- Minpack: http://www.netlib.org/minpack/, a Fortran 77 code for solving nonlinear equations and nonlinear least squares problems.

- Dcdflib: http://www.netlib.org/random/, a Fortran 77 code for cumulative distribution functions, inverses, and parameters.

- Many macros: e.g. derivative

2.2.3 Strong points

In this section, we describe the main advantages of using the classic features of Scilab.

- Accurate algorithms: floating point aware.

- Flexible functions: can drive either Scilab-defined or external codes in C or Fortran.

- Robust algorithms: most algorithms are used for decades.

2.2.4 Issues

- Partial lack of documentation for complicated algorithms such as ode.

- No odes with delay.

- No use of Sundials https://computation.llnl.gov/casc/sundials/, which is used in XCos.

- No Brent algorithm such as Matlab’s fminbnd to minimize \( f(x) \) when \( x \) is scalar.

- No Dekker/Bus/Brent algorithm such as Matlab’s fzero for \( f(x) = 0 \) when \( x \) is scalar.


• It is not possible to have a callback containing the primitive as in the callee [http://bugzilla.scilab.org/show_bug.cgi?id=1657](http://bugzilla.scilab.org/show_bug.cgi?id=1657). For example, assume that \( f \) is the right hand side of an ODE. We call \( \text{ode} \) with \( f \) as input argument, then the body of \( f \) cannot contain a call to the \( \text{ode} \) function.

2.3 Advanced numerical methods and tools

In this section, we present advanced numerical methods and tools. Less than 10% of the users actually use at least one of these features daily. These features are rather advanced tools in numerical analysis and are not found in all technical computing environments.

• Sparse matrices.
  − Storage of sparse matrices
  − Basic algebra +,−,∗,. / and functions number of nonzeros: \texttt{nnz}
  − Conversion from other formats: \texttt{full}, \texttt{adj2sp}, \texttt{spcompak}, \texttt{mtlb_sparse}, Harwell-Boeing: \texttt{ReadHBSparse}, Matrix Market: \texttt{mminfo}, \texttt{mmread}, \texttt{mmwrite}.
  − Generation of random sparse matrices: \texttt{sprand}
  − Sparse LU decomposition and solver: \texttt{lufact}, \texttt{lusolve}
  − Sparse Cholesky decomposition: \texttt{spchol}
  − Iterative methods to solve linear equations: \texttt{gmres}, \texttt{pcg}, \texttt{qmr}
  − Sparse LU decomposition with UMFPACK: \texttt{umf_lufact}, \texttt{umf_lusolve}
  − Sparse Cholesky decomposition with TAUCS: \texttt{taucs_chfact}, \texttt{taucs_chsolve}
  − Eigenvalue problem by Implicit Restarted Arnoldi Method

• Polynomials.
  − definition of polynomials with \texttt{poly}
  − algebra: +,−,∗,. / by overloading of operations
  − \texttt{root}: fast RPOLY routine or eigenvalues of companion matrix
  − Greatest Common Divisor \texttt{gcd}, Least Common Multiple \texttt{lcm}
  − fast evaluation with \texttt{horner}
  − many other algorithms related to CACSD.
- Control: the CACSD module - many, many functions.
- Signal Processing, including filters, \texttt{fft} and \texttt{fftw}
- Sound file handling, including reading and writing of .au and .wav files.

3 Floating point support

At the very basic level, Scilab manages double precision floating point numbers. Hence, any feature which improves the management of doubles improves Scilab as a whole. In this section, we present a short list of possible tasks on this topic.

- \texttt{hypot}: square root of sum of squares. Avoids unnecessary overflow.
- \texttt{set rounding mode}: set the processor rounding mode to one of the four IEEE rounding modes: to-nearest (default), towards zero, towards \(+\)infinity, towards \(-\)infinity. This allows to detect sensitivity to rounding errors in some cases.
- \texttt{format double, format single}: allows to display the exact number of digits to be used for that precision. The current format function only takes as input argument the total number of characters, which is useless. What we need is 17 digits after the decimal point for doubles, 9 digits for single.
- move the default to \texttt{ieee(2)}.
- \texttt{norm with dnrm2}: \url{http://www.netlib.org/blas/dnrm2.f} The current norm function is a macro, using a basic trick for scaling and avoiding overflows. This trick costs more than it should.
- standard ISO C functions: \texttt{copysign, scalbn, signbit, isfinite, isinf, isnormal, fpclassify, fma}

4 Complex support

In this section, we present some issues which could be solved related to the management of complex variables (for which the imaginary part is nonzero).

- Provide a tool to visualize the branch cuts. \url{http://www.mathworks.com/company/newsletters/news_notes/clevescorner/sum98cleve.html} \url{http://www.mathworks.com/company/newsletters/news_notes/clevescorner/summer98.cleve.html}
- Provide a document with branch cuts of all functions. They are currently completely unknown. A work should be done to make this point clearer.
- complex division algorithm. It has been proved in ”Scilab is not naive” that the current algorithm can fail. Better algorithms exist and should be used.
5 Elementary and special functions

In this section, we present several projects which may allow to solve issues related to elementary functions.

- Portability and fdlibm : TODO
- Missing special functions : nchoosek, binomialln, permutations, permutationsln, FresnelC, FresnelS, log1p, expm1, lambertW
- Create a sufficiently large testing framework to assess the numerical quality and stability of a Scilab release with respect to elementary functions. There is currently almost nothing.

5.1 The current state

The elementary functions such as sin, cos, tan, asin, acos, atan, sqrt, exp, log are provided by the compiler. Therefore, they are operating-system dependent. For example, we may get different results for the computation of $\sin(2^{64})$ on Windows and on Linux, or on Linux 32 bits and Linux 64 bits.

gammainc is not in Scilab

 TODO : finish that section.

5.2 Why it fits in Scilab scope

TODO : finish that section.

5.3 Existing features in Matlab

- hypot
- airy
- ellipj
- gamma

5.4 Suggested library to connect

TODO : finish that section.

5.5 Existing tools which may require an update

TODO : finish that section.
6 Dense linear algebra

Dense linear algebra is what Scilab is designed for. Hence, it should remain at the center of our math focus.

A first step should be to analyze the current state of dense linear algebra in Scilab.

- Why is Linpack, Eispac libraries in the sources ?
- Why are there patched Lapack routines in the sources ? Are these patches still required ?
- What is the exact problem solved by unsfdcopy ?
- Is the current version of the Lapack library still sufficiently covered by the existing gateways ?
- Provide sufficient information for using optimized libraries for example: how to use ATLAS for Gnu/Linux. Provide sufficient information about the utility of optimized libraries.
- Create a sufficiently large testing framework to assess the numerical quality of a Scilab release with respect to linear algebra. There is currently almost nothing.
- In the optimization libraries behind optim, the dense linear algebra routines are hard-coded. We should updated them and connect BLAS/LAPACK instead.

TODO : finish that section.

7 Sparse linear algebra

In this section, we present developments which could be done for the management of sparse matrices in Scilab.

7.1 Why it fits in Scilab scope

Efficient linear algebra is the main topic in Scilab. Among other applications, sparse matrices can be used in Finite Elements computations. Large sparse matrices require to use iterative linear algebra solvers, although direct sparse methods can also handle sparse matrices in some cases.

7.2 Existing features in Matlab

- Linear Algebra
  - choline Sparse incomplete Cholesky and Cholesky-Infinity factorizations
  - condest 1-norm condition number estimate
- **eigs** Largest eigenvalues and eigenvectors of matrix
- **ilu** Sparse incomplete LU factorization
- **luinc** Sparse incomplete LU factorization
- **normest** 2-norm estimate
- **spaugment** Form least squares augmented system
- **sprank** Structural rank
- **svds** Find singular values and vectors

• **Linear Equations (Iterative Methods)**
  - **bicg** Biconjugate gradients method
  - **bicgstab** Biconjugate gradients stabilized method
  - **bicgstabl** Biconjugate gradients stabilized (l) method
  - **cgs** Conjugate gradients squared method
  - **gmres** Generalized minimum residual method (with restarts)
  - **lsqr** LSQR method
  - **minres** Minimum residual method
  - **pcg** Preconditioned conjugate gradients method
  - **qmr** Quasi-minimal residual method
  - **symmlq** Symmetric LQ method
  - **tfqmr** Transpose-free quasi-minimal residual method

### 7.3 The current state

Internally, the backslash and slash operators are associated with the Sparse package written by Kenneth S. Kundert and Alberto Sangiovanni-Vincentelli: [http://www.netlib.org/sparse/](http://www.netlib.org/sparse/)

This library is used by the functions lufact, lusolve, luget and ludel. In other words, when the backslash operator is used on a sparse matrix, it is as if the lufact, lusolve and ludel functions were called (in this order). This is done by overloading, in the

The Arnoldi package is a reverse communication module which allows to compute some eigenvalues of a sparse matrix: [http://www.scilab.org/product/man/dnaupd.html](http://www.scilab.org/product/man/dnaupd.html)

On another hand, we find two packages:

• the Taucs package, which provides a direct Cholesky algorithm for sparse matrices: [http://www.scilab.org/product/man/taucs_chfact.html](http://www.scilab.org/product/man/taucs_chfact.html)

• the UMFPACK package, which provides a direct LU algorithm for sparse matrices: [http://www.scilab.org/product/man/umf_lufact.html](http://www.scilab.org/product/man/umf_lufact.html)
The Benchmark shown at: `scilab/modules/umfpack/examples/bench.txt` shows that the UMFPACK functions are much faster than the backslash operator from the Sparse package.


Some routines developed in Scilin have already been integrated into Scilab:

- `pcg`: preconditioned conjugate gradient
- `qml`: quasi minimal residual method with preconditioning

There are some questions which are left to be determined:

- what is the purpose of `spchol` (sparse cholesky factorization): what is the library used by `spchol`?

The current state of Scilab with respect to sparse linear algebra is confusing in the sense of many tools are providing nearly similar features (like `lufact` / `umf_lufact`, `spchol`/`taucs_chfact`), or are iterative solvers (`pcg`, `qmr`) using preconditioners which are not provided by Scilab.

## 8 Optimization

In this section, we present a list of missing optimization features in Scilab. These features are not available in current Scilab toolboxes.

- Non-linear objective with non-linear constraints

### 8.1 Why it fits in Scilab scope

Optimization features are already part of current Scilab features and several Scilab toolboxes.

### 8.2 Existing features in Matlab

Matlab provides the `fmincon` function, which searches for the local minimum of a nonlinear function with nonlinear constraints: `fmincon`

Matlab provides the `bintprog`, which solves binary integer programming problems: `bintprog`

### 8.3 Bug fixes in genetic algorithms

There are design, documentation and testing issues with this component. The following is a list of bug to be fixed.

- The `optim_ga` function does not protect against unexpected parameters. [http://bugzilla.scilab.org/show_bug.cgi?id=7885](http://bugzilla.scilab.org/show_bug.cgi?id=7885)
• There is a design issue with optim ga. [http://bugzilla.scilab.org/show_bug.cgi?id=7705](http://bugzilla.scilab.org/show_bug.cgi?id=7705)

• The optim_moga, optim_nsga and optim_nsga2 functions have no examples. [http://bugzilla.scilab.org/show_bug.cgi?id=8414](http://bugzilla.scilab.org/show_bug.cgi?id=8414)

• The optim_moga, optim_nsga and optim_nsga2 functions do not take into account for the additional arguments in the cost function. [http://bugzilla.scilab.org/show_bug.cgi?id=8415](http://bugzilla.scilab.org/show_bug.cgi?id=8415)

8.4 Bug fixes in fminsearch

There are limitations and design issues with fminsearch. The following is a list of bug to be fixed.

• The fminsearch function does not allow to interrupt the algorithm. [http://bugzilla.scilab.org/show_bug.cgi?id=7890](http://bugzilla.scilab.org/show_bug.cgi?id=7890)

• The neldermead component has too many options. [http://bugzilla.scilab.org/show_bug.cgi?id=7891](http://bugzilla.scilab.org/show_bug.cgi?id=7891)

• The demonstrations of the nmplot functions generate warnings. [http://bugzilla.scilab.org/show_bug.cgi?id=7723](http://bugzilla.scilab.org/show_bug.cgi?id=7723)

8.5 Bug fixes in simulated annealing

There are limitations and design issues with optim_sa. The following is a list of bug to be fixed.

• The optim_sa example does not work comp_t_params should be replaced by comp_t_params. [http://bugzilla.scilab.org/show_bug.cgi?id=5548](http://bugzilla.scilab.org/show_bug.cgi?id=5548)

• Also, the same design issues of the genetic algorithms are true for the simulated annealing. This includes the management of callbacks [http://bugzilla.scilab.org/show_bug.cgi?id=7705](http://bugzilla.scilab.org/show_bug.cgi?id=7705) and the protection against unexpected parameters [http://bugzilla.scilab.org/show_bug.cgi?id=7885](http://bugzilla.scilab.org/show_bug.cgi?id=7885).

8.6 Suggested library to connect

OPT++ opt++ is a library of nonlinear optimization algorithms written in C++. The motivation for this package is to build an environment for the rapid prototyping and development of new optimization algorithms. In particular, the focus is on robust and efficient algorithms for problems in which the function and constraint evaluations require the execution of an expensive computer simulation. Currently, OPT++ includes the classic Newton methods, a nonlinear interior-point method, parallel direct search, generating set search, a trust region - parallel direct search hybrid, and a wrapper to NPSOL. Between these methods, a wide range of problems can be solved, e.g. with or without constraints, with or without analytic gradients, simulation based, etc. OPT++ 2.4 is now available on a variety of platforms under
the GNU Lesser General Public License” Opt++ is developed at CSMR Department Projects at Sandia National Labs in California by Patty Hough, Juan Meza, Ricardo Oliva, Pam Williams.

OPT ++ is available under the GNU Lesser General Public License. It is ported for Linux, Cygwin.

Coin-OR (more specifically CLP and CBC) is a linear programming library can do mixed integer programming, supports sparse matrixes and was before 2004 a commercial product from IBM (OSL). This library was competing (and is still competing) with CPLEX. Interface to CLP is the subject of SEP #6 by Yann Collette: SEP clp

8.7 Existing tools which may require an update

ConMinis a fortran routine written by G. Vanderplaats at NASA for Nastran. ConMin is behind every optimization performed with Nastran. This optimization method allows to perform constrained optimization and is based on the feasible directions method (no problem with a Hessian matrix, only vectors are stored). A first toolbox has been written to interface this method to Scilab. Contribution 1086

8.8 Current projects to provide the feature

The sci-ipopt project aims at providing an interface to IPOPT: sci-ipopt

The fmincon project aims at providing the fmincon function in Scilab, by using the ipopt solver: fmincon

9 Modelization

This is a list of missing features in Scilab:

• non linear modelling: neural networks, Kriging, statistical modeling;

• linear modelling: LASSO for example;

9.1 Why it fits in Scilab scope

Non linear models are used in almost all scientific disciplines like systems control, optimization, simulation, etc ...

9.2 Existing features in Matlab

TODO: finish this section

9.3 Suggested libraries to connect

• LOLIMOT (LOcal LInear MOdel Tree);

• DACE (A Kriging toolbox);
• MARS and PolyMARS (modelization based on piecewise linear models);
• CART (Classification And Regression Tree);

9.4 Discussion

• What is missing for Kriging? The DACE module is already available in ATOMS: http://atoms.scilab.org/toolboxes/dace_scilab
• Lolimot. There exists the Lolimot module: http://atoms.scilab.org/toolboxes/lolimot What work exactly is necessary?
• MARS and PolyMARS (modelization based on piecewise linear models): What library?
• CART (Classification And Regression Tree): What library?

10 Statistics

Statistics are partially covered by current Scilab features, but some tools are still missing:

• bootstrap algorithms to compute the sensitivity of one statistic measure against the data
• more probability density functions, more CDF

The function is provided in Scilab with the cdfbet function: http://www.scilab.org/product/man/cdfbet.html but the Matlab functions allow to expand the input arguments, in the case where one argument is a scalar and another argument is a matrix.

10.1 Why it fits in Scilab scope

Scilab already include several statistical functions.

10.2 Existing features in Matlab

Matlab provides the haltonset, sobolset and scramble functions in the Statistics toolbox:

• haltonset
• sobolset
• qrandset.scramble

Matlab provides a collection of PDFs, CDFs and associated random number generators. For example, it provides the betapdf, betacdf, betainv and betarnd functions:
The binopdf function in Matlab: binopdf or the normpdf functions normpdf are easy to develop in Scilab, but this is a loss of time. Even more, naive implementations leads to wrong numerical results. This topic is developped on the wiki at: Contributor - stats

Matlab provide several fitting functions, based on maximum log-likelihood principle:

- betafit
- binofit
- lognfit
- normfit

Matlab provides negative likelihood functions for parameter estimations:

- normlike
- betalike

The ksdensity function in Matlab provides a kernel smoothing density estimate:

- ksdensity

Matlab provides statistics associated with common distribution functions:

- normstat
- hygestat

Matlab provides several plotting features, including boxplots: boxplot and DOE plots: interactionplot

Matlab provides hypothesis testing functions, such as Kolmogorov-Smirnov one sample test: kstest or one sample T-test: ttest

10.3 Existing tools which may require an update

Low Discrepancy Sequences The module:
http://atoms.scilab.org/toolboxes/lowdisc
provides a first implementation for these sequences. The work which remains to be done is:

- improving the speed by using vectorized statements,
- providing scrambling algorithms (such as Kocis and Whiten’s).
11 Design of Experiments

Design of experiments are a method to approximate the behavior of a system with polynomials.

DOE are completely uncovered by current Scilab features. To include it into Scilab, the following features should be added:

- management of integer uplets (combinations),
- management of several classes of polynomials: order 1, order 2, reduced order 2,
- management of several classes of DOEs: full factorial, centered composite.

Matlab currently provides the following DOE features:

- Full factorial
- Fractional factorial
- Response surface (central composite and Box-Behnken)
- D-optimal
- Latin hypercube

11.1 Suggested libraries to connect to Scilab

Lolimot [http://lolimot.sourceforge.net/] is a project developed by Yann Collette for the management of DOEs in Scilab. It is available with LGPL license: The scilab package is dedicated to design of experiments for:

- polynomials model (Full factorial, D optimal, etc ...);
- non linear models (Latin Hypercube Sampling, etc ...).

11.2 Why it fits in Scilab scope

DOEs would particularly fit in the Scilab environment because of the following reasons.

- The Response Surface Method (RSM) allows to approximate a model based on a polynomial meta-model. The points used to compute the coefficients of the meta-model are produced by the DOE. The computation of the coefficients of the polynomials are based on linear algebra algorithms, in particular least squares, which already exist in Scilab.
- Once created, the user can optimize the meta-model with the optimization features provided by Scilab.
11.3 Existing features in Matlab

The `ff2n` function provides a two-level full factorial design:

```matlab
ff2n
```

while the `fullfact` function provides the same for an arbitrary number of levels:

```matlab
fullfact
```

The `bbdesign` function provides a Box-Behnken design:

```matlab
bbdesign
```

The `ccdesign` function provides a Central Composite design:

```matlab
ccdesign
```

The `lhsdesign` provides a Latin Hypercube sampling design:

```matlab
lhsdesign
```

12 Geometry

12.1 Features

TODO: finish this section

12.2 Suggested libraries to connect to Scilab

TODO: finish this section

12.3 Existing features in Matlab

TODO: finish this section

12.4 Existing tools which may require an update

That work may be done by providing a connection to existing geometry libraries, such as the open source project CGAL:

```text
http://www.cgal.org/
```

The CGLAB toolbox is available at:

```text
Contribution 323
```

12.5 Why it fits in Scilab scope

13 Partial Differential Equations

PDEs appears in the following mathematical fields:

- conservation laws,
- finite volumes,
- finite elements.
There is currently no Scilab component to solve PDEs. Introducing PDE solvers may require the following software components:

- 1/2/3D meshes,
- dense or sparse linear algebra solvers (already available in Scilab),
- PDE solvers.

### 13.1 Why it fits in Scilab scope

TODO: finish this section

### 13.2 Existing features in Matlab

TODO: finish this section

### 13.3 Suggested library to connect

Summary:

- FreeFem: may be included in Scilab (LGPL)
- Clawpack: may not be included in Scilab (research-only license), but may be provided as a Scilab toolbox

FreeFem is a Finite Element library in C++ under LGPL: [http://www.freefem.org/ff++/index.htm](http://www.freefem.org/ff++/index.htm)

FreeFem is developed at Laboratoire Jacques-Louis Lions by highly recognized experts Olivier Pironneau, Frédéric Hecht, Antoine Le Hyaric. FreeFem is about to be released with a support for 3d problems. The 3.0beta can be downloaded at [http://www.freefem.org/ff++/ftp/](http://www.freefem.org/ff++/ftp/).

FreeFem is available as a Scilab toolbox: [Contribution 149](http://www.freefem.org/ff++/ftp/).

The toolbox SciFreefem by Frederic Hecht (modified by Yann Collette) is available: [Contribution 883](http://www.freefem.org/ff++/ftp/).

CLAWPACK is a software package designed to compute numerical solutions to hyperbolic partial differential equations using a wave propagation approach. It is developed by Randall J. LeVeque, Jan Olav Langseth, Marsha Berger, David McQueen, Donna Calhoun, Peter Blossey, Sorin Mitran. License is "research and instructional use only." [http://www.amath.washington.edu/~claw/](http://www.amath.washington.edu/~claw/)

The new version of clawpack 5 (under development beta) is licenced under GPLv3+ [Clawpack](http://www.amath.washington.edu/~claw/)

### 14 Neural Networks

#### 14.1 Existing features in Matlab

Matlab provides the Neural Networks toolbox:

Neural Networks
14.2 Suggested libraries to connect

- FANN (Fast Artificial Neural Network);
- SNNS (Stuttgart Neural Network System);

14.3 Discussion

What would be the advantage of using FANN instead of the existing ANN toolbox:

http://atoms.scilab.org/toolboxes/ANN_Toolbox

What would be a library for SNNS?

TODO: finish that section.

15 Others

There are many!
Some of which are:

- control - CACSD module: ODEs with delays
- ODE: replace Odepack by Sundials
- Optimization: interior point solvers for LMI problems
- Multi-Precision Computations: http://wiki.scilab.org/Contributor_-_MPFR.
- Using the Boost.graph library to replace some of the algorithms in Metanet:

TODO: finish that section.

16 Conclusion

This is a rough, unfinished, incomplete, filled with spell checking and basic errors, draft of a proposal of a mathematical roadmap for Scilab.

We have analyzed Scilab by decomposing it by fundamental mathematical domains, including elementary and special functions, sparse linear algebra, optimization, modelization, statistics, design of experiments, geometry, partial differential equations, neural networks, dense linear algebra, ordinary differential equations and control.

The problems associated with completing this document is nontrivial. Indeed, defining the current state of Scilab in each of its computational domains is a difficult task. In order to define the features to update, we need to determiner their performance and their accuracy: this is unknown to us for many features. By performance, we mean the mathematical performance of the algorithms (that is, their state-of-the-art nature) as well as their CPU speed. Obtaining fast computations requires to
use efficiently the software and hardware currently available (e.g. multi-core architectures). Moreover, gathering the user feedback is difficult, perhaps because of the open source and free nature of Scilab. We emphasize that the user feedback would be a very efficient way of driving Scilab, since some users have a strong expertise in their particular domain.

In an hypothetical context with unlimited time and unlimited task force (with unlimited expertise), the following sequence of steps may be followed, in that order.

1. Create or update technical reports describing the current state of Scilab. The goal of these reports would not be to analyze the features in great detail, but only to give an overview of the existing features, the current implementations, there performance and accuracy. This includes the test of the current implementations in order to know their performance and accuracy.

2. Update this roadmap and take decisions.

3. Update the existing features which may provide poor performance or poor accuracy.

4. Create new functions which may provide missing features.

In order to updating existing features or to create new ones, we have to find (and select) a open-source libraries providing the feature under study, or to develop ourself the missing feature. This may be based on the internal work of the Consortium or the work of external contributors. These tasks are not straightforward.

Driving the roadmap might be driven by:

- the existing use cases provided by the users,
- the will to explore new use cases.

The reason behind this is that choosing based on the only proposals of the Consortium is a somewhat impossible task, leading either to arbitrary choices (may be because of limited knowledge), or to a large and heterogeneous list of existing bug reports to fix or new mathematical domains to explore.

The practical situation is that we have limited time, limited task force, limited feedback from the users about existing use cases and limited will to explore new use cases. In this context, we suggest to focus on the following main numerical computing domains, already covered by existing features in Scilab:

- elementary functions,
- dense linear algebra,
- control,
- ordinary differential equations,
- optimization,
- statistics.
We feel that Symbolic computations, Design of Experiments, Partial Differential Equations, Finite Elements, Neural Networks, Geometry, Sparse linear algebra should be considered as secondary Scilab goals. This is highly questionable. We suggest the following features updates and creations.

- **Elementary functions**: test the accuracy of existing elementary and update the elementary functions by connecting the Fdlibm library.

- **Sparse iterative solvers**: provide pre-conditioners for sparse matrices based on an update of the Scilin project.

- **Statistics**: test existing distribution functions and create new PDFs, CDFs, inverse CDFs, RNGs and other statistics features (smoothing density estimate by kernel, maximum log-likelihood fitting functions, test statistics).

- **Ordinary Differential Equations**: update the Odepack solvers and replace them by Sundials solvers.

- **Design of Experiments**: create full factorial, fractional factorial, D-optimal and LHS designs functions. This domain is not covered by existing features. Still, we forecast that the ratio between the technical difficulty and the practical applications favors this topic.

Moreover, we feel that creating a technical report describing the current state of dense linear algebra should be a secondary priority. Indeed, this is the core of Scilab and this primary goal cannot be ignored. In particular, we should provide a version of Atlas for Gnu/Linux the same way that we provide the Intel MKL for Windows. Moreover, we should clarify specific points, such as the presence of the Eispack sources in Scilab and the future evolutions required to use multi-core architectures, including the analysis of the open-source projects candidates for parallel dense linear algebra.

17 Acknowledgments

This document has evolved from a wiki page managed on Scilab’s wiki [http://wiki.scilab.org](http://wiki.scilab.org) into a LATEXdocument. We would like to thank the authors who participated, at various degrees, to the original wiki page, including Yann Collette, Sylvestre Ledru and Vincent Couvert. We would like to thank Serge Steer, whose input has been reported in the ”Others” section.

References


