HOW WELL IS FREE SOFTWARE ABLE TO REPLACE MATLAB IN SOLUTION OF COMMON CONTROL PROBLEMS? (SURVEY)

Martin Ondera, Mikuláš Huba

Slovak University of Technology in Bratislava, Faculty of Electrical Engineering & IT,
Institute of Control and Industrial Informatics
Ilkovičova 3, 812 19 Bratislava, Slovak Republic

Martin.Ondera@stuba.sk

Abstract

Nowadays it is hardly possible to imagine solution of control engineering problems without a high-quality computational and simulation software. A large amount of computation (both numerical and symbolic) as well as simulation is required at almost every step of control analysis and/or design. Although there are several other candidates, a vast majority of control engineers will most likely agree that from all software dedicated to solution of control problems MATLAB is the most suitable – it provides a large scale of control-related toolboxes, the Simulink module for an easy simulation of control systems and (last but not least) it has a large community of users and a long tradition of usage in the field. And yet, there is something which makes many control engineers consider using for their purposes software different from MATLAB – the high price for the licence. At the same time, however, they would like to keep the comfort MATLAB gives them and to use the other software in the same way as MATLAB. The aim of this paper is, therefore, to "map the territory" of available freeware packages that are generally regarded as MATLAB substitutes and to judge their ability to replace MATLAB in solution of everyday control engineering problems. Unlike most of similar comparison studies, it does not describe the basic mathematical and/or graphical functions of the software. Instead, it looks for features that facilitate control system analysis and design within each particular package and tries to compare them with those of MATLAB, Simulink and Control System Toolbox. Five different MATLAB alternatives are introduced, two of them – Scilab and Octave – are discussed in detail.

Keywords: control engineering, MATLAB, free software, Scilab, Octave.

Presenting Author's biography

Mikuláš Huba received the MSc. and PhD. Degrees in technical cybernetics from Slovak University of Technology in Bratislava in 1974 and 1982, respectively. From 1989 he is a Senior Lecturer and Head of the Control Theory Group of the Institute of Control and Industrial Informatics at the Faculty of Electrical Engineering and Information Technology. From 1996 he is also Head of the university Distance Education Centre. He is an author of more than 180 papers, monographs on Constrained Control and Constrained PID Control and co-author of two monographs on Flexible and Web-Based Learning.

1 Introduction

Over the last decade, thanks to its suitability and a wide range of toolboxes, MATLAB (together with its simulation module Simulink) has become a #1 computational and simulation software used in control engineering. It is being successfully applied to solution of control problems both in industry and at universities. However, all of its numerous advantages are somewhat diminished by a single (but very significant) disadvantage: MATLAB is not cheap. In fact, the price for a licence is enormous, besides, each of the toolboxes is charged separately. Therefore, it is quite reasonable to look for alternatives.

Nowadays, there are several different free software packages that are being declared (either by their authors or by users) as MATLAB substitutes. There are even several studies dedicated to the comparison of the features of these packages with those of MATLAB, e.g. [1], but they are usually limited only to basic mathematical and/or graphical functions and do not account for their use in control engineering. This paper tries to fill the gap, concentrating on the possible features that might be useful in control applications. Above all, the two most famous packages, Scilab and Octave are examined but other options are also briefly mentioned. MATLAB itself is taken as a benchmark to judge the abilities of the free software.

The paper is organized as follows. In Section 2, all of the packages are introduced one-by-one and the basic facts about them are given. Sections 3 and 4 then provide a closer look at the control-related features of Scilab and Octave, the two major packages discussed in the paper, and compare these features with their MATLAB equivalents as well as with one another. Section 5 contains a “case study” consisting of a few examples from automatic control theory that were solved sequentially in MATLAB, Scilab and Octave. Finally, section 6 concludes the paper by an attempt to give the outlook to the future.

2 Available MATLAB substitutes

This section offers a short overview of five free MATLAB alternatives, namely Scilab, Octave, RLab, Pylab and SysQuake LE. The basic information mentioned for each of the software includes its origin, list of supported computer platforms and a general level of similarity to MATLAB.

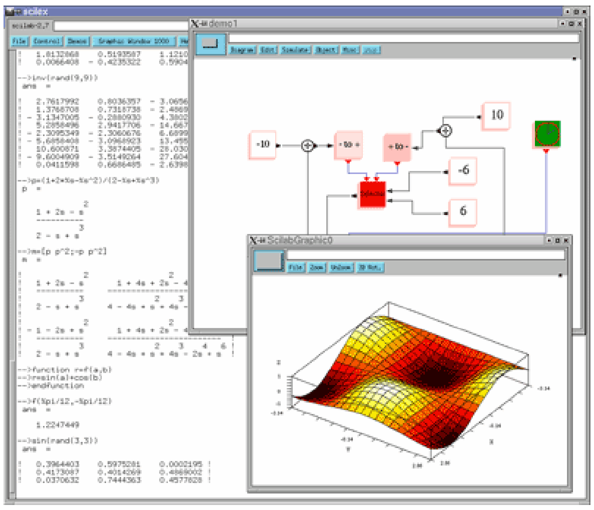
2.1 Scilab

Scilab is an open-source numerical computational package developed by the French National Institute for Research in Computer Science and Control (INRIA). The first version of the software dates back to 1989 and the latest version so far (4.1.1) was released in May 2007. Scilab 4.1.1 is currently available for use with Windows 2000/XP/Vista, GNU/Linux and HP-UX, however, previous versions also worked with Windows 9x (Scilab 4.0) and Windows NT (3.0). Finally, although not officially supported, there are also versions for other platforms, such as Mac OS X.

Scilab resembles MATLAB in many ways – both products are primarily dedicated to numerical computations, both work with matrices as basic data types and both use a modular system of toolboxes to widen their application opportunities. From a control engineering point of view it is especially useful that (unlike any other free package) Scilab also contains a Simulink equivalent (named Scicos) for modelling and simulation of dynamic systems. The Scilab programming language also bears a high resemblance to MATLAB code.

In spite of all this, Scilab and MATLAB are not entirely compatible and although Scilab even contains a converter for translation of MATLAB code to Scilab language, a user intervention is usually necessary in order to make MATLAB programs work correctly in Scilab. Moreover, unfortunately, so far there is no tool able to convert Simulink block diagrams to Scicos.

Besides the official program documentation, there are several other reference guides on Scilab/Scicos, e.g. [2, 3, 4, 5].



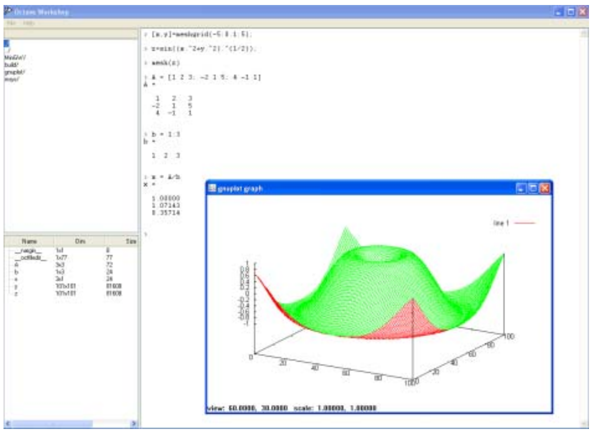
```
!   1.8132868    0.5193587    1.1210
!   0.0066408  - 0.4235322    0.5904

-->inv(rand(9,9))
 ans  =

!   2.7617992    0.8036357  - 3.0656
!   1.3768708    0.7316738  - 2.4869
! - 3.1347005  - 0.2880930    4.3802
!   5.2858496    2.9417706  - 14.667
! - 2.3095349  - 2.3060676    6.6099
! - 5.6858408  - 3.0968923    13.455
!   10.600871    3.3874405  - 28.030
! - 9.6004909  - 3.5149264    27.604
!   0.0411598    0.6686485  - 2.6398

-->p=(1+2*%s-%s^2)/(2-%s+%s^3)
 p  =

              2
    1 + 2s - s
    ----------
              3
    2 - s + s

-->m=[p p^2;-p p^2]
 m  =

!             2                  2
!   1 + 2s - s        1 + 4s + 2s - 4
!   ----------        ---------------
!             3                2   3
!   2 - s + s         4 - 4s + s + 4s -
!
!             2                  2
! - 1 - 2s + s        1 + 4s + 2s - 4
!   ----------        ---------------------------
!             3                2   3    4   6 !
!   2 - s + s         4 - 4s + s + 4s - 2s + s !

-->function r=f(a,b)
-->r=sin(a)+cos(b)
-->endfunction

-->f(%pi/12,-%pi/12)
 ans  =

    1.2247449

-->sin(rand(3,3))
 ans  =

!   0.3964403    0.5975281    0.0002195 !
!   0.4173087    0.4014269    0.4869002 !
!   0.0370632    0.7444363    0.4577828 !
```

Fig. 1 Scilab – working environment

2.2 Octave

Octave (or more precisely GNU Octave) is a high-level language primarily intended for numerical computations that is “mostly compatible with MATLAB” [6], originally created and up to now maintained by John W. Eaton from the University of Wisconsin. The development of the language began in 1992 and the first official version (1.0) was released in early 1994. Currently there are two different “latest” versions available for download – Octave 2.1.73 (March 2006) declared as “stable” and Octave 2.9.12 (May 2007) declared as “testing”. It was also announced that Octave 3.0 should be released shortly.

Unlike MATLAB or Scilab, Octave in its pure form is not a package but merely a programming language, which means that there are more than one environments implementing it. Therefore, apart from "official" Octave binary files that are available for use with Linux, Mac OS X and Windows [6], there is also e.g. a currently unmaintained distribution named Octave Workshop [7].



```
> [x,y]=meshgrid(-5:0.1:5);
> z=sin((x.^2+y.^2).^(1/2));
> mesh(z)
> A = [1 2 3; -2 1 5; 4 -1 1]
A =

   1   2   3
  -2   1   5
   4  -1   1

> b = 1:3
b =

  1  2  3

> x = A/b
x =

  1.00000
  1.07143
  0.35714

>
```

Fig. 2 Octave Workshop – working environment

From all of the MATLAB substitutes, Octave is probably the one whose syntax is the most similar to MATLAB – e.g. [8] states that the syntax is "practically identical" while in [9] it is written that "carefully programming a script will allow it to run on both Octave and MATLAB". However, although the previous statements are generally true as far as basic mathematical and/or graphical functions are concerned, this is unfortunately not the case with Octave's control-related features, as will be illustrated in sections 4 and 5. Besides, Octave so far does not provide any Simulink-like tool for simulation of dynamic systems.

The official program documentation is available both in a book form [10] as well as online; there is also a guide to Octave in Slovak language [8].

2.3 RLab

RLab is an interactive interpreted numerical computation program and its core programming language written by Ian R. Searle [9]. The development of RLab had started in 1989 and continued until 2000, when its author abandoned the project. The latest version of the original RLab (2.1) is available both in binary form (for Linux and Windows) and as a source code (under the General Public License). In 2004, the RLab project was taken over by Marijan Kostrun from the University of Connecticut who renamed it RLaBplus and added several new features. However, this new version is not available for the Windows platform.

Despite the fact that RLab is generally considered a MATLAB alternative, its similarity to MATLAB is restricted only to the problems handled. According to [11] the original intention of its author was to "borrow the best features from the MATLAB language and provide improved language syntax and semantics". Although this intention might seem advantageous at the beginning, nowadays it rather disqualifies RLab from becoming an easy-to-use MATLAB substitute. Moreover, as far as dedicated control-related features are concerned, RLab has practically nothing to offer.

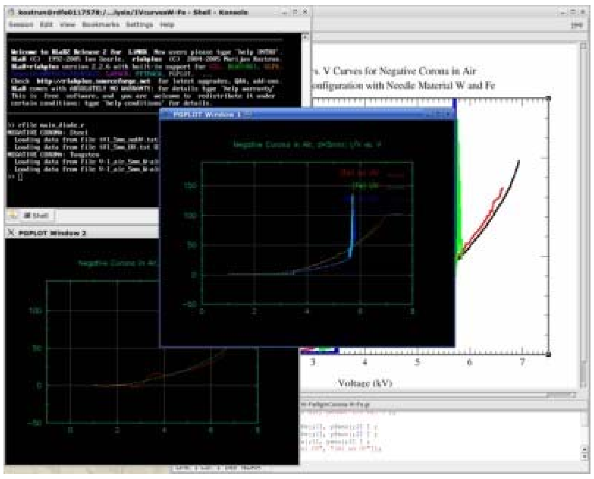
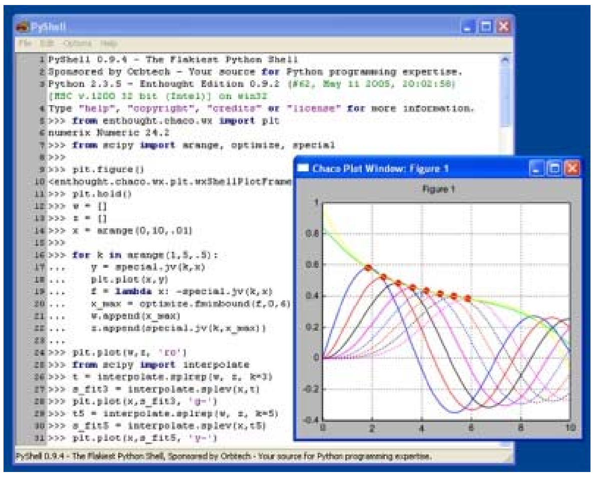


Fig. 3 RLab – working environment



```
PyShell 0.9.4 - The Flakiest Python Shell
Sponsored by Orbtech - Your source for Python programming expertise.
Python 2.3.5 - Enthought Edition 0.9.2 (#62, May 11 2005, 20:02:58)
[MSC v.1200 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from enthought.chaco.wx import plt
numerix Numeric 24.2
>>> from scipy import arange, optimize, special
>>>
>>> plt.figure()
<enthought.chaco.wx.plt.wxShellPlotFrame
>>> plt.hold()
>>> w = []
>>> z = []
>>> x = arange(0,10,.01)
>>>
>>> for k in arange(1,5,.5):
...     y = special.jv(k,x)
...     plt.plot(x,y)
...     f = lambda x: -special.jv(k,x)
...     x_max = optimize.fminbound(f,0,6)
...     w.append(x_max)
...     z.append(special.jv(k,x_max))
...
>>> plt.plot(w,z, 'ro')
>>> from scipy import interpolate
>>> t = interpolate.splrep(w, z, k=3)
>>> s_fit3 = interpolate.splev(x,t)
>>> plt.plot(x,s_fit3, 'g-')
>>> t5 = interpolate.splrep(w, z, k=5)
>>> s_fit5 = interpolate.splev(x,t5)
>>> plt.plot(x,s_fit5, 'y-')
```

PyShell 0.9.4 - The Flakiest Python Shell, Sponsored by Orbtech - Your source for Python programming expertise.

Fig. 4 Python/Pylab – a screenshot

2.4 Pylab

Pylab is an interactive, matrix-oriented system for scientific computation based on Python programming language inspired by MATLAB [12]. Unlike MATLAB or any of its previously mentioned freeware alternatives, Pylab is not a single coherent package but rather a collection of several cooperating modules (IPython, NumPy, SciPy, Matplotlib, etc.) interconnected through the Python programming language.

Pylab is available for use with both Unix-based operating systems as well as Windows, however, its use with the latter is according to [12] more likely to cause problems and therefore somewhat discouraged.

Pylab's compatibility level with MATLAB code is questionable and by no means able to compete with that of Octave or Scilab. Also, similarly as in the case of RLab, so far there is no known Pylab module dedicated to control systems analysis, design and/or simulation, which makes Pylab rather uninteresting from a control engineering perspective.

2.5 SysQuake LE

SysQuake is a numerical computing environment based on a programming language mostly compatible with MATLAB that offers facilities for interactive graphics which give insight into the problems being analyzed [9]. The origin of the software dates back to 1997 while the latest version so far (3.5) was released in October 2006. Developed and distributed by a Swiss company Calerga Sarl, SysQuake is commercial software. A limited version of the package, named SysQuake LE, can be obtained free of charge, though. The currently supported platforms are Windows 2000/XP, Mac OS X and Linux.

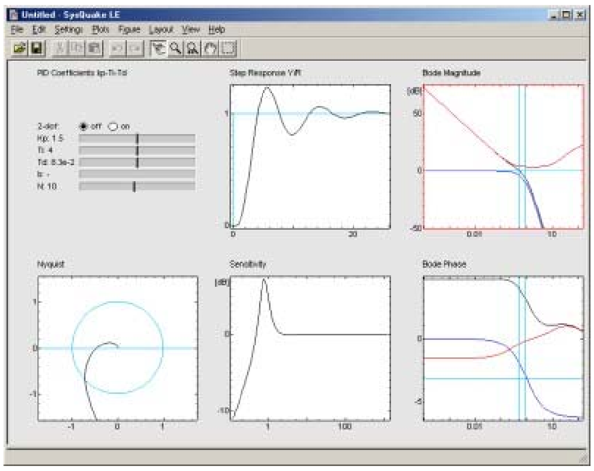


Fig. 5 SysQuake LE – a screenshot

In contrast to all the previously mentioned software, SysQuake is, in fact, not a general MATLAB substitute – such an aim is not stated anywhere in its documentation [13] nor is it a "public opinion". Nevertheless, its MATLAB-compatible language (named LME), as well as some of its features make it worth mentioning to the control engineering community.

SysQuake's strongest point is its interactivity – in this aspect, it seems, SysQuake outperforms not only any of the MATLAB substitutes, but even MATLAB itself. For example, it is able to display several different characteristics for a given control system (e.g. open- and closed-loop step response, Bode and Nyquist frequency plots, etc.) at the same time, allows to change the system's parameters by adjusting either of the plots and the results of such changes are immediately reflected in all of the characteristics. There are several control-related demos/applications bundled with SysQuake, e.g. continuous- and discrete-time PID control, Generalized Predictive Control, ARW structures, etc.

On the other hand, in comparison with Simulink, simulations in SysQuake miss the high flexibility and the clear structure provided by a diagram composed of logically interconnected blocks (in other words, whereas Simulink allows a user to change both the structure and the parameters of a simulation model equally easily, SysQuake provides only an elegant way to adjust parameters but the structure of the simulation model is not so easy to change). Therefore, it is more probable that SysQuake LE will find its position in the control engineering world as an addition to MATLAB, rather than as its replacement.

2.6 Other options

Apart from the five MATLAB alternatives mentioned above, there are also other software tools that could be used to replace MATLAB in several partial tasks from control engineering – e.g. Modelica or DYNAST could be used instead of Simulink for simulation of some control systems. However, as none of these tools aims to substitute MATLAB as a whole, they are considered out of the scope of this paper.

3 Scilab control-related features

This section presents a list of the features of the latest version of Scilab (4.1.1) that are more or less directly related to control engineering. The first subsection is dedicated to Scilab's control-related toolboxes (especially to the "General System and Control" toolbox) and the second to Scicos, Scilab's module for simulation of dynamic systems (i.e. a Simulink equivalent).

3.1 Scilab control toolboxes

So far, there are four main toolboxes in Scilab devoted to control engineering:

- General System and Control
- Robust Control Toolbox
- ARMA Modelisation and Simulation Toolbox
- Identification

Besides, there are also others, which (although not primarily dedicated to the purpose) might help in solution of control problems as well, such as "Optimization and Simulation" or "Signal Processing Toolbox".

The "General System and Control" toolbox of Scilab (version 4.1.1) – according to [14] – contains 77 functions, many of which are equivalent to those of MATLAB's Control System Toolbox. Unfortunately, to find a function that would be completely identical is rather an exception than a rule – even in the case where the functions' names are the same (e.g. *ss2ss*), their parameters are usually different. A list of some of the functions together with their possible MATLAB equivalents is available in Tab. 1 below.

Tab. 1 "General System and Control" toolbox (selected functions)

Scilab function	**Description**	**MATLAB equivalents**
abcd	state-space matrices	*ssdata*
canon	canonical controllable form	*canon*
cls2dls	bilinear transform	*c2d*
cont_mat	controllability matrix	*ctrb*
csim	simulation (time response) of a linear system	*lsim, step, impulse, …*
dscr	discretization of a linear system	*c2d*
dsimul	state-space discrete time simulation	*lsim, dstep, dimpulse, …*
/.	feedback operation	*feedback*
freq	frequency response	*freqresp*
lin	linearization	*linmod*
linmeq	Sylvester and Lyapunov equation solver	*lyap, dlyap*
lqe	linear quadratic estimator (Kalman filter)	*kalman*
lqg	LQG compensator	*lqgreg*
lqr	LQ compensator (full state)	*lqr, dlqr*
lyap	Lyapunov equation	*lyap, dlyap*
obsv_mat	observability matrix	*obsv*
ppol	pole placement	*place, acker*
repfreq	frequency response	*freqresp*
ricc	Riccati equation	*care, dare*
ss2ss	state-space to state-space conversion	*(ss2ss)*
ss2tf	conversion from state-space to transfer-function	*(ss2tf)*
tf2ss	conversion from transfer-function to state-space	*(tf2ss)*
trzeros	transmission zeros and normal rank	*zero (tzero)*

One of the many important differences between MATLAB and Scilab regarding their basic control toolbox that may be a source of difficulties to MATLAB users trying Scilab is a different representation of linear time-invariant systems – while MATLAB uses for the purpose special object-data types (*tf*, *ss*, *zpk*, …), Scilab represents them as lists of a predefined type (the *syslin* command is used for a linear system definition). Another source of discomfort rests in the fact Scilab lacks *help*- and *lookfor*-like commands, which complicates the search for MATLAB equivalents.

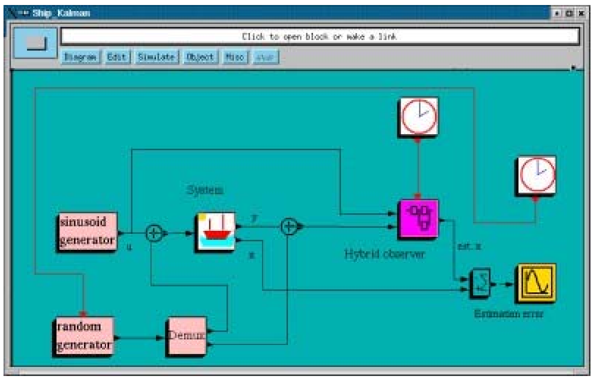


Fig. 6 Scicos – Kalman estimator demo

3.2 Scicos

As was already written in section 2, Scicos is Scilab's Simulink-equivalent, i.e. a tool for simulation of dynamic systems. Unfortunately, it is not compatible with Simulink at all, which means that if one wants to do a simulation of a system in Scicos based on an existing Simulink model, the only way to do so is to manually re-create the simulation model from blocks that are available in Scicos.

Similarly as in Simulink, Scicos blocks are grouped in "blocksets" (in Scicos these are called "palettes") that are accessible through the *Edit / Palettes* menu. According to [15] there are 13 palettes in the latest version of Scicos – they are listed in Tab. 2 below.

Tab. 2 Scicos – list of palettes

Scicos palette	**Contents**	**Simulink equivalents**
Sources	constants, function generators, clocks, file inputs	*Sources*
Sinks	outputs, scopes, meters, file output	*Sinks*
Linear	state space, sum, integral, differential, TFs, gain, …	*Continuous, Discrete*
Non-linear	math expr., saturation, log, product, exp., trig., …	*Discontinuities, Math Oper.*
Events	clock, delays, halt, binary ops	*Sinks, Sources*
Threshold	upward, downward, zero-crossing	*Discontinuities*
Others	logical, generic, rate, deadband, constraints, …	*Math Oper., Discontinuities*
Branching	if-then-else, switch, mux, demux, relay, …	*Signal Routing*
Electrical	meters, supplies, discrete components	*Sim PowerSystems*
Thermo Hydraulics	pipe, tap, wells, container	*SimHydraulics*
Old Blocks	blocks for backward compatibility	-----
Demo Blocks	bouncing balls, scope	*Simulink demos*
RTAI-Lib	hard real-time acquisition card interfaces, scope, meter, FIFO, mailboxes	*RT Workshop RT WinTarget*

Scicos doesn't support the Simulink drag&drop practice – to place a block in a simulation diagram one should first click its icon in the palette and then the position where it should be placed. Connecting blocks is carried out in a very similar way – after selecting the *Edit / Link* command (or even without it) one first has to click the source block's output and then the destination block's input. Finally, moving and deleting blocks is also different from Simulink – instead of dragging or pressing the *Del* key, one has to right-click a block and select the appropriate command (i.e. *Move* or *Delete*) from its context-menu.

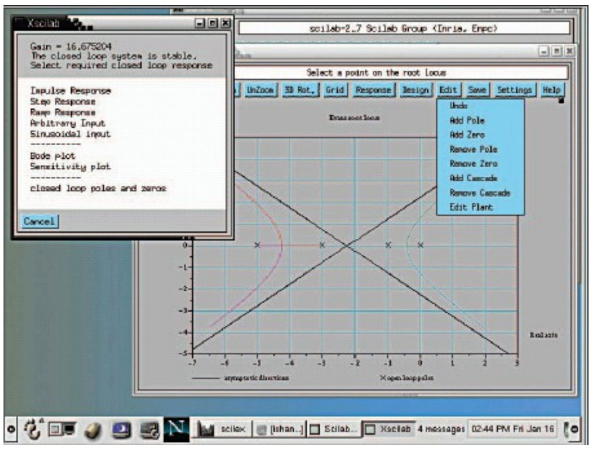
Unlike Simulink, Scicos uses two different types of links – regular ones that transmit signals (drawn in black and placed at the sides of a block by default) and activation ones (red, placed at the top/bottom of a block) that transmit activation information. The activation information determines at which time instants the input(s) and/or the output(s) of a particular discrete-time block are activated (i.e. the signal is read from the inputs or sent to the outputs of the block) – in Simulink this is usually carried out by setting the "sampling time" parameter of the block.

Scicos allows a user to specify 8 different parameters of the simulation (*Simulate / Setup*). The most important of them are *Final integration time* (this is equivalent to Simulink's *Stop time*), *Integrator absolute* and *relative tolerance* (equivalent to their Simulink counterparts) and *Maximum step size* (*Max step size* in Simulink). The solver to be employed for the simulation can be specified by setting a numeric value on a scale from 0 to 100 – unfortunately, the difference between different values is not clearly stated.

Scicos simulation diagrams can be saved in two file formats – binary (default, the file extension is *.cos*) and text (*.cosf* extension). However, an inexperienced user might overlook the latter as the indication of the text-file-format option in the title of a file-save dialog box is rather unusual.

3.3 Control-related demos

In Scilab, Scicos, their documentation and guide-books there are several control examples and demos. For example, in [2] a simulation of a discrete-time state observer for a continuous-time linear system is used to demonstrate usage of Scicos blocks. Demos on PID, LQG, robust control, tracking, mixed sensitivity as well as several others are accessible through the *? / Scilab Demos* menu.


Gain = 16.675204
The closed loop system is stable.
Select required closed loop response

Select a point on the root locus

Fig. 7 RLTool for Scilab – a screenshot

3.4 User-donated extensions

Apart from official Scilab control-related features which are bundled with the official distribution of the software, there are also several user-donated Scilab extensions and toolboxes (of different quality) that might be useful in control engineering applications. These include e.g. RLTool for Scilab: Scilab's equivalent of MATLAB's SISO Design Tool [16] (Fig. 7), sciFLT: Scilab Fuzzy Logic Toolbox [17], ANN: toolbox for artificial neural networks or REALTERM: an interface to serial port. A complete list of the user-donated extensions can be found at Scilab Home Page [14], Toolboxes Center.

4 Octave control-related features

4.1 Octave Control Systems Toolbox

Octave Control Systems Toolbox is the only control-related toolbox that Octave possesses. Originally developed and described in [18] it contains commands for dealing with linear time-invariant control systems that are similar to those of MATLAB's Control System Toolbox. Some of the functions included in the toolbox are listed in Table 3 below.

Tab. 3 Octave Control Systems Toolbox (selected functions)

Octave function	Description	MATLAB equivalents
are	continuous-time algebraic Riccati equation	*care*
bode	Bode plots of a system	*bode*
c2d	continuous-to-discrete-time conversion	*c2d*
ctrb	controllability matrix	*ctrb*
d2c	discrete-to-continuous-time conversion	*d2c*
dare	discrete-time algebraic Riccati equation	*dare*
dkalman	discrete-time Kalman predictor	*kalman*
dlqe	discrete-time Kalman filter	*dlqe*
dlqr	discrete-time LQ regulator	*dlqr*
lqe	continuous-time Kalman filter	*lqe*
lqr	continuous-time LQ regulator	*lqr*
lsim	simulation (time response)	*lsim*
nyquist	Nyquist plot of a system	*nyquist*
obsv	observability matrix	*obsv*
place	pole placement control design	*place*
ss	state-space system representation	*ss*
step	step response	*step, dstep*
sysadd	parallel connection of blocks	*parallel*
sysmin	minimal realization	*minreal*
sysmult	series connection of blocks	*series*
sysout	print system in a desired format	*printsys*
tf	transfer-function system representation	*tf*
zp	zero-pole-gain system representation	*zpk*

4.2 Control-related demos

There are 7 demos in Octave Control Systems Toolbox illustrating the use of the functions for system representation, block-diagram manipulation, frequency response, state space analysis, system model manipulations, root locus and LQG/H_2/H_∞. All of the demos are accessible through the *DEMOcontrol* command.

4.3 User-donated extensions

In contrast to MATLAB or Scilab, user-donated extensions for Octave devoted to solution of control problems are rare and rather difficult to find as there is no list of user-donated files neither in [10] nor in [6]. One of such infrequent tools is The NMPC Control and Estimation Tool [19] dedicated to nonlinear model predictive control.

5 A "case study"

This section utilizes three simple examples from control theory to demonstrate the use of Scilab and Octave in solution of control problems and to examine possible difficulties that might arise from their use. Each example is solved sequentially in MATLAB, Scilab and Octave, the results are compared and the similarities and/or differences are discussed.

5.1 Linear SISO pole-placement control design

Let us consider a very simple SISO linear system described by the state-space equation

$$\dot{x} = Ax + Bu \tag{1}$$

where

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, B = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{2}$$

To this system we want to design a state feedback

$$u = -Kx \tag{3}$$

such that the poles of the resulting closed loop were a) *[-2, -1]* or b) *[-2, -2]*.

With MATLAB, the solution can be calculated using the well-known *place* and *acker* commands:

```
>> A=[0 1; 0 0];
>> B=[0; 1];
>> Ka=place(A,B,[-2 -1])

Ka =

     2     3

>> eig(A-B*Ka)

ans =
    -1
    -2

>> Kb=acker(A,B,[-2 -2])

Kb =
     4     4

>> eig(A-B*Kb)

ans =
    -2
    -2
```

It is worthy of note that while in the a) case we can choose arbitrary of the two functions, in the b) case only *acker* can be used since *place* is not able to assign double real pole to a single-input system – this results from different algorithms used for calculation of the feedback (*acker* is based on Ackermann's formula whereas *place* employs singular value decomposition).

In order to do the same calculation in Scilab we have to replace the *place* and *acker* commands with their Scilab equivalent – *ppol.* Similarly, we also have to use the *spec* command instead of MATLAB's *eig*:

```
-->A=[0 1; 0 0];
-->B=[0; 1];
-->Ka=ppol(A,B,[-2 -1])

Ka  =
    2.    3.

-->spec(A-B*Ka)

ans  =
  - 2.
  - 1.

-->Kb=ppol(A,B,[-2 -2])

Kb  =
    4.    4.

-->spec(A-B*Kb)

ans  =
  - 2.
  - 2.
```

For an easier conversion of MATLAB code to Scilab language the *Matlab to Scilab conversion tool* integrated in Scilab can be used. Unfortunately, the tool is by far not perfect and cannot properly handle commands other than the most-basic ones. In this particular example, for instance, the tool replaces *spec* with *eig* but is unable to substitute *ppol* for *place* or *acker*.

Octave solution is a bit more complicated because of different system representation required by the *place* command:

```
octave:1> A=[0 1; 0 0];
octave:2> B=[0; 1];
octave:3>         C=[1          0];
octave:4> sys=ss(A,B,C);
octave:5> Ka=place(sys,[-2 -1])

Ka =
  2  3

octave:6> eig(A-B*Ka)

ans =
  -1
  -2

octave:7> Kb=place(sys,[-2 -2])

Ka =
  4  4

octave:8> eig(A-B*Kb)

ans =
  -2
  -2
```

Although the *C* matrix plays no role in the design, it has to be specified, otherwise the *ss* command produces an error.

5.2 Linear MIMO pole-placement control design

Let us now consider a multiple-input multiple-output case with the system matrices being

$$A = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, B = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{4}$$

The MATLAB solution now is

```
>> A=[0 1; 0 0];
>> B=eye(2);
>> Ka=place(A,B,[-2 -1])

Ka =
     2     1
     0     1

>> eig(A-B*Ka)

ans =
    -2
    -1

>> Kb=place(A,B,[-2 -2])

Kb =
     2     1
     0     2

>> eig(A-B*Kb)

ans =
    -2
    -2
```

and the Scilab one

```
-->A=[0 1; 0 0];
-->B=eye(2,2);
-->Ka=ppol(A,B,[-2 -1])

Ka  =
    2.    0.
    0.    1.

-->spec(A-B*Ka)

ans  =
  - 2.
  - 1.

-->Kb=ppol(A,B,[-2 -2])

Kb  =
    2.    0.
    0.    2.

-->spec(A-B*Kb)

ans  =
  - 2.
  - 2.
```

One can see that this time the *Ka*, *Kb* gains calculated by Scilab are different from those of MATLAB, although, of course, both solutions are correct as both ensure the desired poles of the closed loop. Nevertheless, this fact indicates different algorithmic implementations behind MATLAB's *place* and Scilab's *ppol*.

Unlike MATLAB or Scilab, Octave's *place* function cannot handle MIMO systems so far. Therefore, Octave solution of the above example cannot be presented here.

5.3 Wind-up effect simulation

In order to compare simulation capabilities of MATLAB, Scilab and Octave let us consider the Simulink block diagram in Fig. 8. Simulations like this are often used in introductory courses on nonlinear and/or constrained control as a demonstration of wind-up effect resulting from inappropriate use of controllers with integral action.

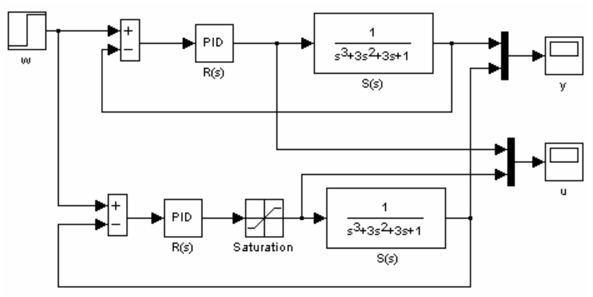


Fig. 8 Wind-up simulation in Simulink – – simulation diagram

In the block diagram in Fig. 8 the 3rd-order linear system

$$S(s) = \frac{1}{(s+1)^3} \tag{5}$$

is controlled by the linear PID controller

$$R(s) = 1{,}75\left(1 + \frac{1}{2{,}6s} + \frac{0{,}65s}{0{,}065s+1}\right) \tag{6}$$

The desired value is w = 1,5 and the saturation of the controller output is set to $|u| \le 2$. Both the case with and without saturation are simulated and from the

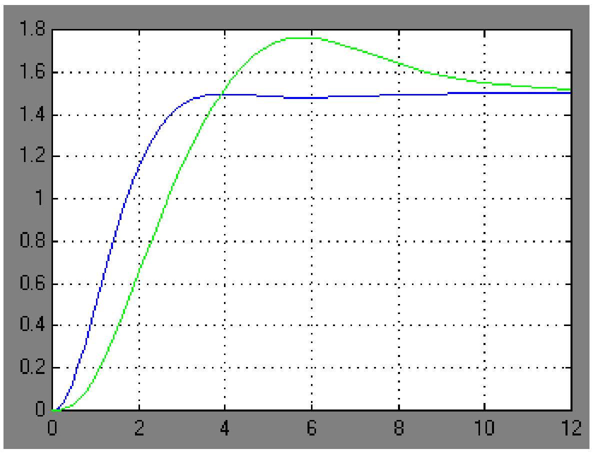


Fig. 9 Wind-up simulation in Simulink – – simulation results (output variable *y*)

results (Fig. 9-10) it is possible to see the influence of the constraint to the output variable y as well as to the control action u.

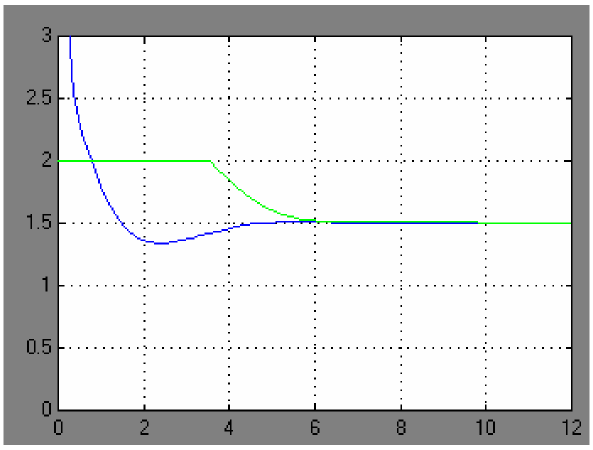


Fig. 10 Wind-up simulation in Simulink – – simulation results (control action u)

The same simulation can also be accomplished with Scicos – the corresponding block diagram is in Fig. 11 and the results in Fig. 12-13. One can see that although the block icons are different, the overall simulation scheme is almost the same, up to few details: in comparison with Simulink, Scicos does not offer the *PID controller* block (the *transfer function* blocks were used instead) and unlike Simulink ones, *Scope* blocks in Scicos have to be connected to the source of activation signal (clock).

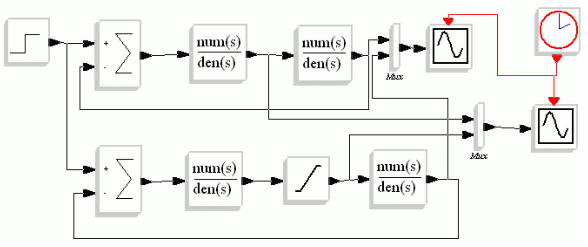


Fig. 11 Wind-up simulation in Scicos – – simulation diagram

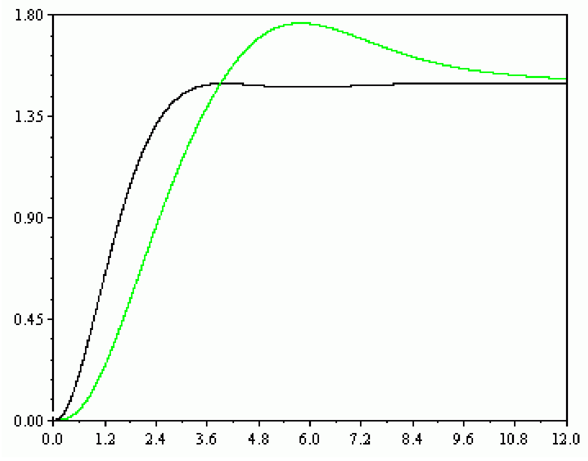


Fig. 12 Wind-up simulation in Scicos – – simulation results (output variable y)

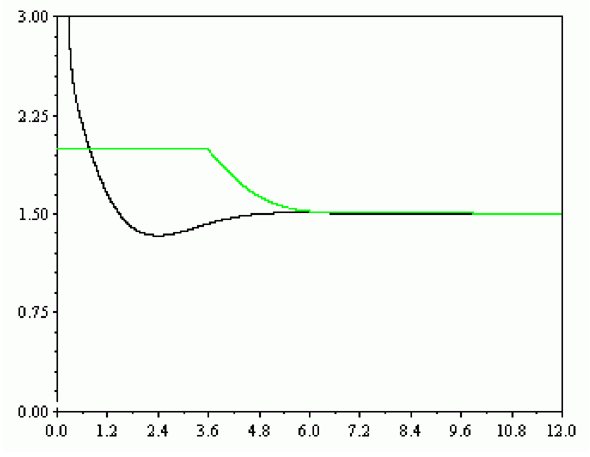


Fig. 13 Wind-up simulation in Scicos – – simulation results (control action u)

Because Octave does not posses any block-diagram simulation module, in order to do a simulation in Octave one needs to write custom program code based on commands like *sysadd*, *sysmult* and *lsim*. However, all these commands are dedicated only to linear system simulation and thus cannot handle the nonlinear element (saturation) in the loop. As a result, the only way of solving the above example in Octave is to find a nonlinear differential equation describing the system and solve it using one of the available ODE solvers, e.g. *ode45*. Obviously, such a "low-level" solution is extremely uncomfortable and time-consuming.

6 Conclusions

In this paper a comparison of the control-related features of several free MATLAB substitutes was presented. Five different packages were briefly introduced and two most famous of them – Scilab and Octave – were examined in detail. Currently, it seems that none of the packages has the potential to replace MATLAB in solution of control problems fully and easily – even if some of them provide almost the same functionality as MATLAB, their use is less comfortable and therefore not too appealing to long-term MATLAB users.

A mutual comparison of the two major candidates, Scilab and Octave is (and in future will very probably remain) very much in favour of the former. Apart from the most important fact that Scilab has more to offer in the field of control applications, there are also other factors that will most likely cause that Scilab will keep its today's dominance. First, although both the software are distributed free of charge, there is a considerable difference in their "backing" – while the development of Scilab is still (at least partially) organised and financed through INRIA, Octave has practically no financial support and its development is entirely upon GNU-software enthusiasts. Next, (although die-hard Unix/Linux fans may argue the importance of this), Scilab is also more Windows-user-friendly than Octave – whereas in order to use the most recent version of Octave with Windows one has to go through a somewhat complicated installation procedure, Scilab is nowadays primarily dedicated to Windows platform (the default download button at Scilab's website retrieves the Windows version), while at the same time its use with Unix-based operating systems is not affected in any way. And finally, Scilab is gradually building a growing community of users – a thing that according to many played a key role in fighting MATLAB its today's position in the control engineering world.

7 Acknowledgments

The authors would like to gratefully acknowledge the support provided by:

- ESF Project 13120200015 "Human Resource Development in Automation and Control",
- KEGA Project 3/3121/05 "Network of Virtual Laboratories for Real Process Control",
- VEGA Project 1/3089/06 "Development and Integration of Methods of the Nonlinear System Theory".

8 References

[1] G. E. Urroz. Comparing SCILAB and Matlab. http://www.engineering.usu.edu/cee/faculty/ /gurro/Software_Calculators/Scilab_Docs/ /SciMatLabDiff.pdf

[2] S. L. Campbell, J.P. Chancelier and R. Nikoukhah Modeling and Simulation in Scilab/Scicos. Springer. New York. 2006.

[3] C. Bunks, J. P. Chancelier, F. Delebecque, C. Gomez, M. Goursat, R. Nikoukhah and S. Steer. Engineering and Scientific Computing with Scilab. Birkhäuser. Boston. 1999.

[4] R. Nikoukhah and S. Steer. SCICOS – A Dynamic System Builder and Simulator. User's Guide. http://www.scicos.org/p.pdf.

[5] J. Pribiš. SCILAB [in Slovak]. FEI TU. Košice. 2006.

[6] http://www.gnu.org/software/octave/octave.html

[7] http://www.math.mcgill.ca/loisel/octave-workshop/

[8] J. Buša. OCTAVE. An extended introduction [in Slovak]. FEI TU. Košice. 2006.

[9] http://en.wikipedia.org/

[10] J. W. Eaton. GNU Octave Manual. Network Theory Ltd. 1997.

[11] http://rlab.sourceforge.net/

[12] M. Kaukič. The basics of Pylab programming [in Slovak]. FEI TU. Košice. 2006.

[13] http://www.calerga.com/doc/SQ3_main.htm

[14] http://www.scilab.org/

[15] http://www.scicos.org/

[16] I. Pendharkar. RLTool for Scilab: a public domain tool for SISO system design. *IEEE Control Systems Magazine,* 25, 1, 2005.

[17] http://es.geocities.com/jaime_urzua/sciFLT/ /sciflt.html

[18] A. S. Hodel, R. B. Tenison, D. A. Clem and J.E. Ingram. The Octave Control Systems Toolbox: a MATLAB™-like CACSD environment. In: *Proceedings of the 1996 IEEE International Symposium on Computer-Aided Control System Design*, 15-18 Sep 1996, pp. 386-391.

[19] http://jbrwww.che.wisc.edu/home/tenny/nmpc/ /index.html