PolyxGui : A new Graphical User Interface (GUI) for the Polynomial Toolbox POLYX

Dimitris Varsamis and Nicholas P. Karampetakis

***Abstract*— It is well known that the polynomial toolbox *Polyx* is a package for systems, signals and control analysis and design based on advanced polynomial methods. *Polyx* consists of as many as 222 M-files in *Matlab* code, which implements new original algorithms that are fast and reliable. The aim of this work is to present a new Graphical User Interface (GUI), build in *Matlab*, for the easy use of the polynomial toolbox *Polyx*. This new interface gives the user, who is not very familiar with all these functions of *Polyx*, the opportunity to handle easily polynomial matrices and systems.**

I. Introduction

IT is well known that the Computer-Aided-Design (CAD) package of *Matlab* (MathWorks Inc.) [1] provides a dynamic, user-friendly and open computational environment for implementation of several scientific applications in a wide range of knowledge fields, such as linear algebra, statistics, applied mathematics, numerical analysis, processing signal, control theory.

The environment of *Matlab* supports a vast number of inner operations and functions, as well as toolboxes for specialized areas of scientific applications. It also supports a flexible, simple and structured programming language, which matches language C. The operations of *Matlab* are separated into : a) *numerical*, that is the ones which manipulate numerical data and extract numerical results, and b) *symbolic*, which manipulate and calculate symbolic expressions, that is they calculate with mathematical symbols.

The Polynomial Toolbox *Polyx* [2] is a *Matlab* package for systems, signals and control analysis and design based on advanced polynomial methods. It consists of as many as 222 M-files in MATLAB code and is easy to use.

More specifically, *Polyx* has the following characteristic features and abilities:

- Simple manipulation with polynomials matrices and legible presentation of these matrices with the aid of a new object (pol-object).
- Overloaded operations and functions for polynomial matrices, solvers for numerous linear and quadratic matrix polynomial equations.
- Polynomial matrices with complex coefficients for applications in signal processing.
- New generation of numerical algorithms : easy, fast and reliable.
- Continuous-time and discrete-time system and signal models based on polynomial matrix fractions.
- Easy conversion of a pol-object to and from LTI objects of the *Control System Toolbox* [3] and polynomial objects defined in the *Symbolic Math Toolbox* [4].
- and much more (http://www.polyx.com/download/handout.pdf)

D. Varsamis and N. P. Karampetakis are with the Department of Mathematics, Aristotle University of Thessaloniki, Thessaloniki 54006, Greece, (corresponding author e-mail: karampet@math.auth.gr).

Matlab offers to the programmer the ability to create a graphical user interface (GUI). A GUI is a graphical display that contains devices, or components, that enable a user to perform interactive tasks. The usefulness of this operation is great, because the programs or application that has been implemented with graphical user interfaces become more user-friendly to the final user. Many *Matlab* toolboxes now come with GUIs. The *Control System Toolbox*, for example comes with GUIs like the *LTI Viewer* (is a GUI that simplifies the analysis of linear, time-invariant systems), the *Root Locus* design GUI, the *Graphical Tuning Window* (a GUI for displaying and manipulating the Bode, root locus, and Nichols plots for the controller currently being designed). The Polynomial Toolbox *Polyx* comes also with a GUI like the *Polynomial Matrix Editor* for creating and editing polynomial and standard *Matlab* matrices.

By using the extended capabilities of *Matlab* in creating GUIs, we have build a new GUI, named *PolyxGui*, that handles the most of the existing functions of the polynomial toolbox *Polyx*. This new GUI, will give the opportunity to the users of *Polyx* to easily handle the most of its functions, without great experience in the syntax of the functions.

II. Program Description

This work has been implemented in a Graphical User Interface of *Matlab*. The development of a GUI in *Matlab* is separated in two stages.

The first stage is the design of application with the aid of guide *layout editor*. In *layout editor* the programmer can design the objects that are necessary to execute the application. The objects which apply in application are pushbuttons, popup menus, text boxes, edit boxes, check boxes etc. Most objects from them are accompanied by a callback function. Also the programmer can create the menus which are in the menu bar. All these features are saved in a figure, wherein is the initial figure of application.

The second stage is the programming of the application.

Initially, the programmer adds the commands that are necessary in each object which is accompanied by a callback function. In the callback function the programmer can set the operations that the object will do after its execution. Except the programming of the callback function, the programmer can create a new object of any type and can handle all properties of all objects.

The application *PolyxGui* follows the above description. In the initial figure there exists a triple (text box, popup menu and text box) of objects. The text box and the menus in the menu bar were created by the *layout editor*. All the other objects were created with programming.

The objects designed in *layout editor* are named *constants* because they are used for only one operation and are accompanied by only one callback function. The objects created with programming are named *temporary,* since each of these is used for many operations, and are accompanied by many different callback functions. *PolyxGui* has four temporary pushbuttons that are accompanied by more than 50 callback functions. *PolyxGui* has also a temporary popup menu which is accompanied by many callback functions. The data and mainly the polynomial matrices are presented with as many text boxes as are the elements of the matrix and the size of text boxes is shaped according to the elements of the matrix.

PolyxGui uses many types of dialog boxes such as *inputdlg* (used for input or define the parameters of some operations), *errordlg* (used for informing the user for some wrong execution in operation), *warndlg* (used for informing the user for some wrong setting of parameters in operation) and *questdlg* (used for the user to select the action which he wants). For all operations there is control checking if the data or the parameters are correct.

Data in *PolyxGui* are stored in a double structure. The polynomial matrices created by the user or saved as a result are stored in the first structure. The systems defined by the user or saved as a result are stored in the second structure.

PolyxGui has m-files that initialize the properties of all objects or initialize the values of popup menus.

Finally *PolyxGui* has text-files which bear the help of each operation.

III. Guide Application

PolyxGui on the initial figure has three main menus: *File, Matrices* and *Systems* (see Figure 1).

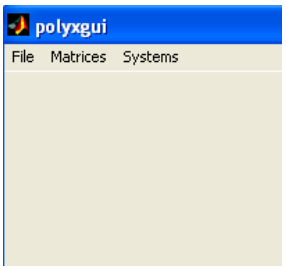

Figure 1. Main menu

The menu *File* (see Figure 2) consists of the submenus:

- The submenu *New* help the user to create a polynomial/constant matrix or to define a new system.´
- The submenu *Editor* that helps the user to edit the saved matrix.
- The submenu *Delete* that helps the user to delete the saved matrix or system.
- The submenu *Save Work* that helps the user to save the current work, which are all the matrices and systems saved in application.
- The submenu *Load Work* is used for loading the saved work.
- The submenu *Exit* is used for the exit from the application.

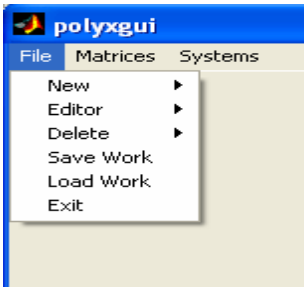

Figure 2. File menu

The menu *Matrices* (see Figure 3) consists of the submenus:

- The submenu *Basic Operations* is separated into another two submenus:
  - The submenu *Operations with 2 Matrices* consists of the following operations: addition, subtraction and multiplication of polynomial matrices.
  - The submenu *Operations for 1 Matrix* consists of the following operations: 1) simple operations for matrix, such as the calculation of transpose, conjugate, adjoint, inverse, pseudoinverse, nullspace, minimal basis and dual matrix; 2) properties for matrix, such as the calculation of determinant, rank, roots and degree and 3) test for a polynomial matrix, such as if the matrix is fullrank, singular, unimodular, prime, stable, row reduced and column reduced.

- The submenu *Advanced Operations* has five submenus:
  - The first submenu *Special Constant Matrices* is used for the creation of the Hurwitz, Sylvester and companion matrix.
  - The second submenu *Divisors and Multiples* is used for the division of polynomial matrices, the computation of the greatest common divisor and the least common multiple and for test if two matrices are co-primes and proper.
  - The third submenu *Reduced and Canonical Forms* calculate the row reduced form, column reduced form, hermite form, echelon form and Smith form of a polynomial matrix.
  - The fourth submenu *Factorizations* calculates the factorizations LU, spectral and co-spectral of a polynomial matrix.
  - The fifth submenu *Matrix Pencil Routines* calculates the Kronecker canonical form and the Clements form.

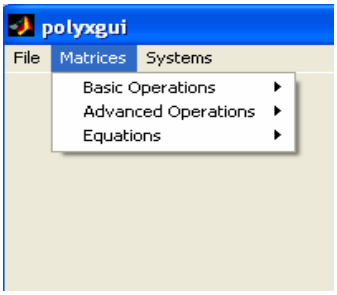

Figure 3. Matrices menu

- The submenu *Equations* has four submenus: *Linear, Diophantine, Symmetric* and *Lyapunov Equations*. In all submenus the respective equation are solved.

The menu *Systems* (see Figure 4) consists of two submenus:

- The submenu *System Conversions* that gives the user the ability to convert a system into another form: State Space, Descriptor State Space, Left Matrix Fraction, Right Matrix Fraction, Transfer Function.
- The submenu *System Design* that is used for stabilization of a system or pole placement.

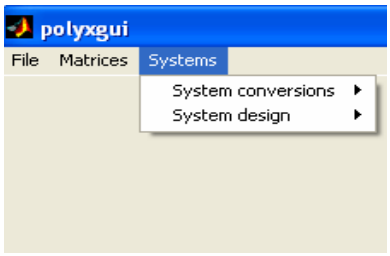

Figure 4. Systems menu.

When the user selects an operation, the objects (handles) which are necessary to execute this operation are presented on the figure. Each operation has a help text (coming from *Polyx*) in the top right corner of figure wherein exist information about how to execute this operation (see Fig. 5).

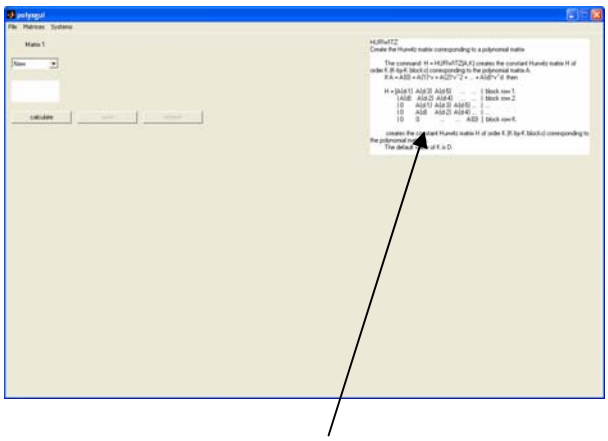

Figure 5. Help text for each operation.

Each operation of the program is checked for any wrong input given by the user or any wrong calculation during the operation. If an error has been occurred then a message is displayed to the user, giving information for the error and how to fix this problem (see Figure 6).

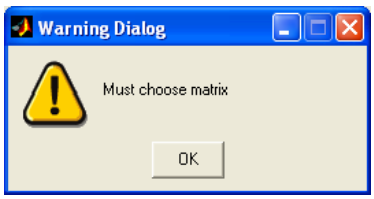

Figure 6. Warning dialog for wrong input data.

IV. Example

Consider a system described by the left matrix fraction (LMF) description $H = Q^{-1}P$ with matrices

$$P(s) = \begin{bmatrix} 1 & s \\ -1 & 1 \end{bmatrix} \text{ and } Q(s) = \begin{bmatrix} 0 & s^2 - s \\ 2 - s & s - 1 \end{bmatrix}$$

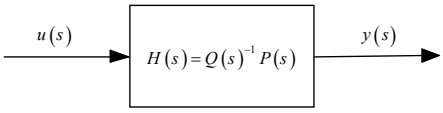

Figure 7. Open loop system

In order to declare the above system in *PolyxGui*, first we define the two polynomial matrices P and Q. We create the polynomial matrices P and Q from menu *File→ New→ New Matrix→ Polynomial Matrix* and following the guidelines from help text, we set the dimensions and the name of matrix (see Figure 8) and then we fill the edit boxes with elements of matrix (see Figure 9).

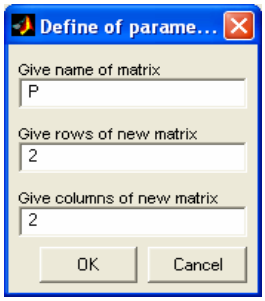

Figure 8. Define the dimensions and the name of matrix

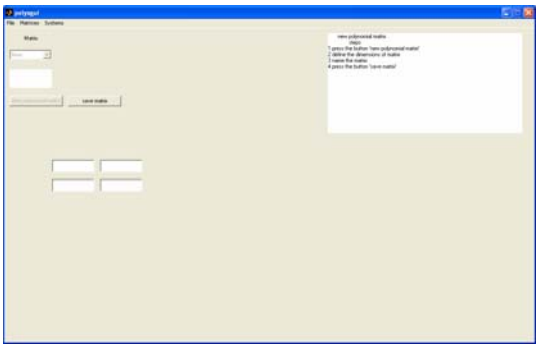

Figure 9. Define the elements of matrix

Then we define the LMF description of the system from menu *File→ New→ New System→ Left Matrix Fraction* and following the guidelines from help text, we define the matrices P and Q of the LMF. The program checks if these matrices are constitute a LMF description i.e compatible dimensions and $\det[Q] \neq 0$. When the system is saved, a figure with information for the system is presented (see Figure 10).

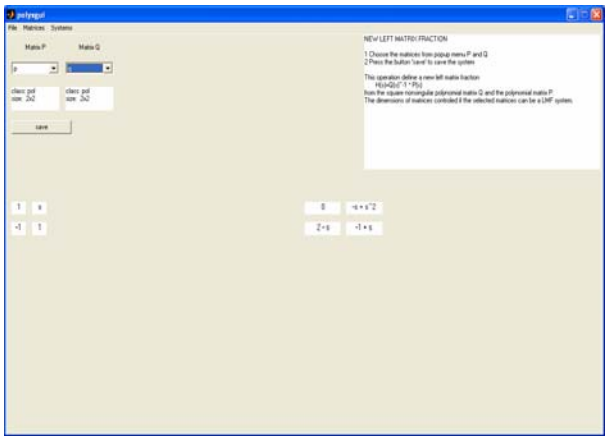

Figure 10. Define the system

We can easily check if our open loop system is stable by finding the roots of the determinant of the polynomial matrix $Q(s)$. First we select *Matrices→ Basic operations→ Operations for 1 Matrix* and then *Roots* from the popup menu *Choose Property*. Finally we select the matrix $Q(s)$ in respective popup menu. The roots that *PolyxGui* give us, are the following $\{0,2,1\}$ and thus the system is unstable (see Figure 11).

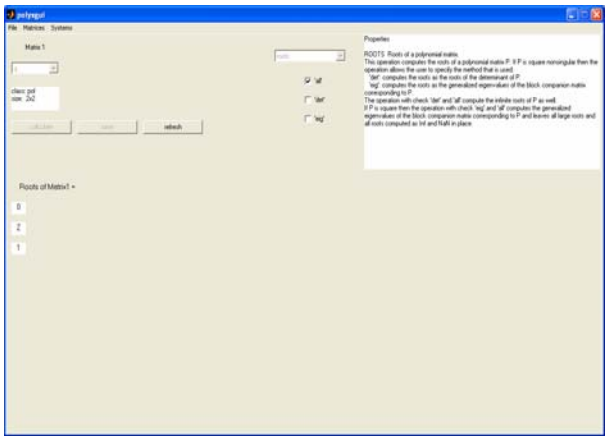

Figure 11. Roots of matrix $Q(s)$

Suppose now that we are interested to design a compensator $C(s) = N_C(s) D_C(s)^{-1}$ such that the closed loop given in Figure 12, has all its poles in the left half plane.

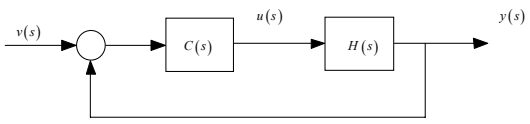

Figure 12. Closed loop system

First we follow the menu *System→ System design→ Pole Placement→ for Lmf*. Then, we select the LMF system from the respective popup menu (in this popup menu only LMF system is presented). In the sequel, we press the button design and a figure which we set the number of poles is

presented. Then we set the desired poles in edit boxes and we press the button continue to view the result (see Figure 13). Let the new poles be $\{-1,-2,-3\}$.

In the result we watch the compensator $C(s) = N_C(s)D_C(s)^{-1}$ where

$$N_C = \begin{bmatrix} 6+7s & -4s \\ 11 & 4 \end{bmatrix},\ D_C = \begin{bmatrix} 1+s & -1-s \\ 7+s & 0 \end{bmatrix}$$

and the diagonal matrix

$$R(s) = \begin{bmatrix} 6+11s+6s^2+s^3 & 0 \\ 0 & 2+3s+s^2 \end{bmatrix}$$

The matrix R is created from *Polyx* by using the desired poles. Finally the user has the ability to save the results or with the form of system or as matrices.

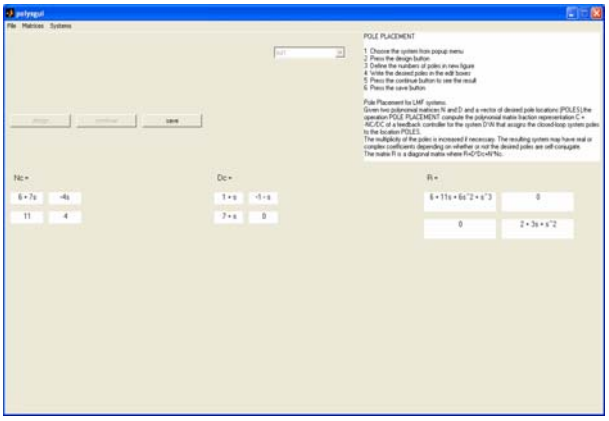

Figure 13. Result of pole placement.

The same problem is solved with the use of Diophantine equation. This operation exists in menu *Matrices→ Equations→ Diophantine*. Initially, we select the type of Diophantine equation from respective popup menu and afterwards we select the matrices of equation. Define $A = Q$, $B = P$ and $C = R$ and the Diophantine equation $AX + BY = C$ will be $QX + PY = R$ (see Figure 14).

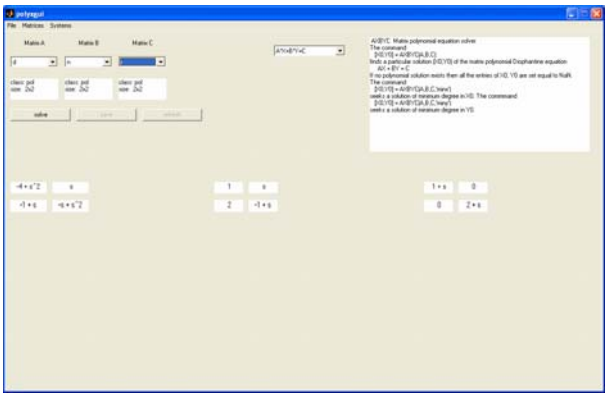

Figure 14. Define parameters of Diophantine equation.

Then we press the button solve and the results are presented in figure (see Figure 15).

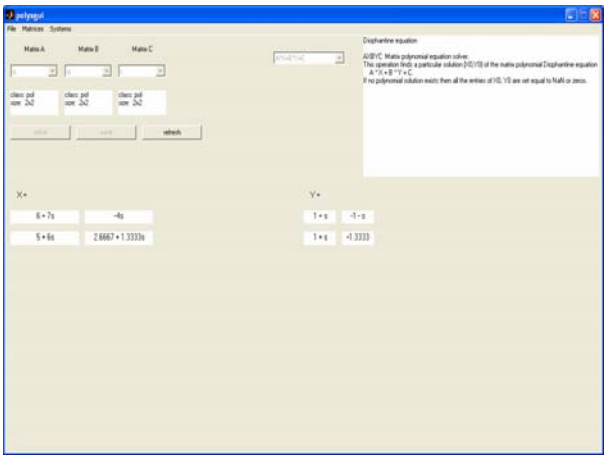

Figure 15. Result of Diophantine equation.

The solutions are $X = \begin{bmatrix} 1+s & -1-s \\ 1+s & -1,333 \end{bmatrix}$ and $Y = \begin{bmatrix} 6+7s & -4s \\ 5+6s & 2,67+1,33s \end{bmatrix}$ where $X = D_C$ and $Y = N_C$ and belong to the same family of solutions with the ones that we find above.

Finally, the user has the ability to save the results or to replay this operation.

V. Conclusion

A graphical user interface, named *PolyxGui*, has been developed. This new program gives the user the opportunity to use the toolbox *Polyx* and the software *Matlab* with out the user having to know neither the programming environment of Matlab nor the functions of *Polyx*. This program is addressed to scientists whose field of knowledge includes polynomials, polynomial matrices and their applications in systems, signals and control. Our next target is to include much more functions in the *PolyxGui* from the new versions of *Polyx* and *Matlab*. The interested user may find this new program in the URL : http://anadrasis.web.auth.gr/karampet/Proceedings/ecc2007.rar .

Acknowledgment

It is a great pleasure to acknowledge the assistance we have received from the people working in Polyx Ltd. and especially to Martin Hromčík and Michael Sebek.

References

[1] The MathWorks, Matlab 6, release 13, http://www.mathworks.com.
[2] Polyx, Ltd. Polynomial Toolbox 2.5, http://www.polyx.com .
[3] The MathWorks, Control System Toolbox 7.1, http://www.mathworks.com/products/control/ .
[4] The MathWorks, Symbolic Math Toolbox 3.1.5, http://www.mathworks.com/products/symbolic/ .

[5] THE MATHWORKS, Creating graphical user interfaces, Version 7, http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/buildgui.pdf .