DEVELOPING OPEN AND FLEXIBLE COMPUTING ENVIRONMENTS FOR TEACHING MATHEMATICS AND SCIENCE

Miroslaw MAJEWSKI
Zayed University, United Arab Emirates
mirekmajewski@yahoo.com

Fred SZABO
Concordia University, Canada
szabo@alcor.concordia.ca

ABSTRACT

Most software packages for the teaching of mathematics contain only a limited set of tools, utilities, and procedures. Therefore we often have to use more than one computer program to teach different subjects. This situation makes teaching very inefficient: We need to teach our students how to use each tool, we need to teach different, environment-dependent strategies for solving problems, and sometimes we even need to adapt to completely different computing philosophies. An ideal teaching package, on the other hand, would allow teachers and students to customize the program by modifying the resources within the program, adding their own procedures, functions and operations, and might even allow them to build their own mathematical libraries. In this paper, we show that *MuPAD*, a computer algebra system from SciFace Software and the University of Paderborn in Germany, is gradually becoming such an ideal electronic teaching environment in this sense since it already meets several of the mentioned requirements. We will show how teachers can build their own libraries, add and integrate them with *MuPAD* resources, and use them in their teaching with both a standalone and an online version of *MuPAD*. Finally, we will discuss some of the advantages of this open and flexible environment.

Desiderata

Let us begin by quoting from a letter by Carlos Fleitas, a teacher of mathematics from Spain, who mentions that he uses *Cabri* and *CarbiWeb* to make interactive geometry, uses *Derive* to study functions and graphs, tries to present the elementary ideas of probability with the help of *Excel*, and has recently begun to investigate *MuPAD* as a tool for generating L-systems. We know of other teachers of mathematics who are using even more expanded sets of tools in their classroom. Such approaches are difficult to follow.

In all of these cases, both the teachers and the students need to master several computing tools in addition to having to learn the mathematical concepts involved. There is a good reason for using such nonintegrated sets of tools in the classroom. Not one of these packages can be used to teach all or almost all topics in undergraduate mathematics. However, recent advances in the development of the *MuPAD* computer algebra system, for example, are giving us hope that the situation might change in the near future. Before we begin to analyze some of the promising aspect of *MuPAD*, we will identify some of the features we expect a computer package to have to be suitable for the teaching of undergraduate mathematics.

The package should have a broad and easily expandable mathematical base, be easy to learn and use, and fit naturally into most modern teaching and learning environments. By this we mean the following:

1. The package should provide an environment for the teaching of the widest possible range of topics in undergraduate mathematics: abstract algebra, linear algebra, geometry, calculus, differential equations, probability and statistics, as well other standard topics.
2. The package should provide means for teachers to enrich and expand the functionality of the package through customized libraries and software extensions.
3. The package should be easy to maintain and update by teachers and their assistants, even in schools with modest computer facilities, and should be portable across computer platforms such as the Windows, Linux and the Macintosh environments.
4. The package should be easy to use. In particular, it should be compatible with mainstream electronic course management systems such as WebCT. It should have a natural and easily learned interface and help facility. Moreover, it should be easy to learn by average students with relatively little supervision.
5. The package should provide for both command-line programming as well as for the menu-driven manipulation of mathematical objects, especially graphs and surfaces. Its programming language should be easy to learn and have as natural a syntax as possible.

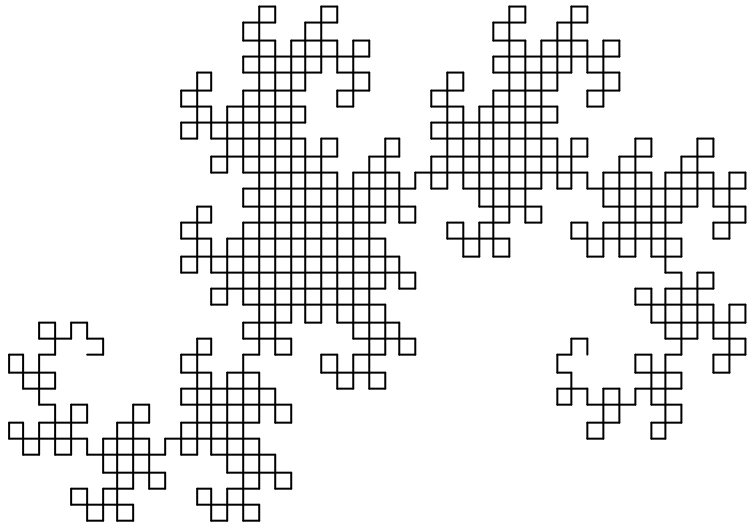
This list can certainly be expanded and does not encompass all relevant features. Some of them were discussed in detail in our earlier paper (see [2]). Here we will concentrate mainly on the problem of developing *MuPAD* customized libraries and using them as teaching tools.

The Role of Libraries in Computer Algebra Systems

Libraries for computer algebra systems are sets of mathematical procedures usually grouped by topic. For example, a library may consist of procedures for teaching calculus, number theory, probability and statistics, and so on. The procedures defined in a library may range from simple tools for solving quadratic equations or calculating greatest common divisors to sophisticated methods for finding shortest paths in oriented graphs.

Computer algebra systems that allow teachers to build their own libraries are invaluable tools for advancing the teaching of mathematics. Using such systems, teachers and students can

collaboratively build a wide range of toolboxes for their courses, modify them as needed, and improve and expand them over time. This provides a live environment for experimenting with mathematics. Teachers can exchange libraries with their colleagues and build extensive educational systems. Moreover, the ability to customize libraries may inspire enterprising students to solve mathematical problems that are often beyond of their school curriculum. The paper on L-systems mentioned below (see [1]), is one such example of a student project that went considerably beyond the standard undergraduate curriculum. Its author, *Michelle Raimbert*, began with an undergraduate paper on L-systems, written under the supervision of the second author, an introduction to L-systems by the first author, and created a *MuPAD* notebook on L-systems containing a beautiful collection of programs for generating fractals and other branching structures. The main objective of this paper was to investigate the suitability of MuPAD for the representation of such algorithms and techniques and to illustrate this suitability by creating some of the most famous fractal curves. One of them is a well-known Harter-Heightway Dragon curve,



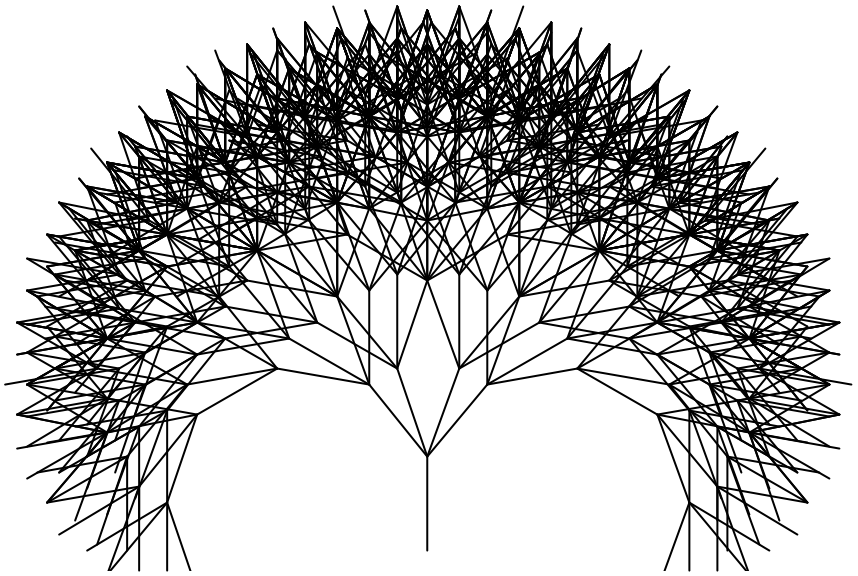
generated by the MuPAD code

```
L:=plot::Lsys(90, "BL",
   "L" = "L+R+",
   "R" = "-L-R",
   "L" = Line, "R" = Line,
   "B" = RGB::Black
):
L::generations:=11:
plot(L,Axes=None)
```

It is pedagogically valuable that the rule to generate the Harter-Heightway Dragon curve is conceptually quite simple. Start with a single segment of the length L. In each of the subsequent steps, replace any obtained segment by a semi-triangle, i.e. the figure that contains two equal segments separated by the right angle. This construction can be mimicked physically to some extent, by folding a piece of paper. However, for larger numbers of steps the use of computer is necessary. In most of the known cases, the creation of the Harter-Heightway Dragon curve requires a good knowledge of a programming language and programs creating this curve are

usually not simple. With MuPAD, however, we can create this curve by typing eight lines of simple code without losing sight of the algorithm involved.

Another interesting example of a fractal structure from the same student's project is a beautiful tree like shape where she experimented with multiple colors, here for printing purposes presented as a black-and-white picture,



generated by the code

```
L:=plot::Lsys (20, "L",
    "L"="BR[++YL][+OL][--YL][-OL]+",
    "R"=Line, "L"=Line,
    "Y"=RGB::OrangeRed,
    "O"=RGB::Pink,
    "B"=RGB::CadmiumRedDeep
  ):
  L::generations:=6:
plot(L,Axes=None)
```

Again, the creation of this type of fractal shape requires a good knowledge of a programming language and recursive programming techniques. With MuPAD nine lines of code suffice and only two lines require a bit more explanation.

The two above examples show that computer algebra systems provide us with entirely new tools for visualization of intriguing mathematical objects: elaborate curves and surfaces, and representations of biological phenomena such as the growth of algae, the veins in leaves, and the bronchi in the lungs. As such, they help us to instill a new physical meaning into mathematical objects.

As the above code shows, many of these objects are generated easily and intuitively. The *MuPAD* project on L-systems [1], for example, provides an enjoyable starting point in one such direction. We know of many other examples where students have discovered interesting mathematical facts by experimenting with mathematical concept by using a computer program.

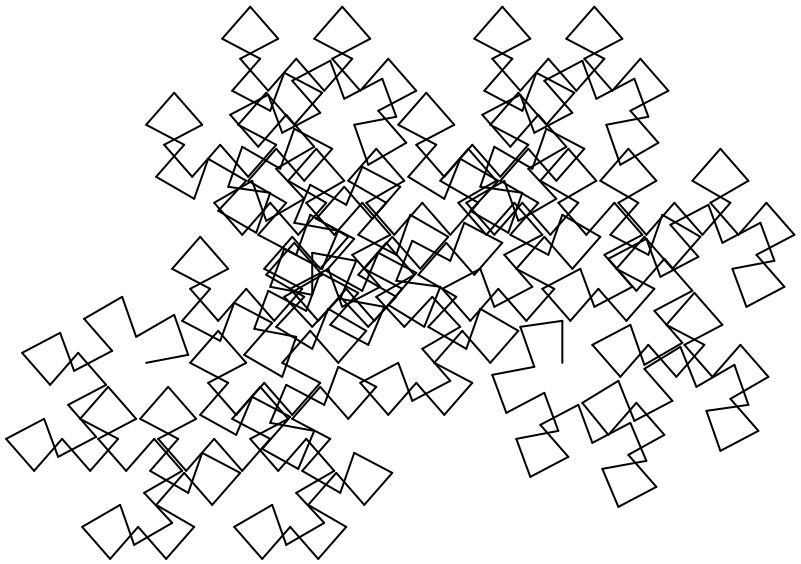
From an experimental student's project or classroom works there is one step to organizing the most interesting pieces of work and saving them later as reusable library of procedures.

For example, with a very basic knowledge of MuPAD programming student or teacher can convert the code presented in [1] into a library of new L-system procedures. Here we show how it can be done for the mentioned above dragon curve and the tree like shape.

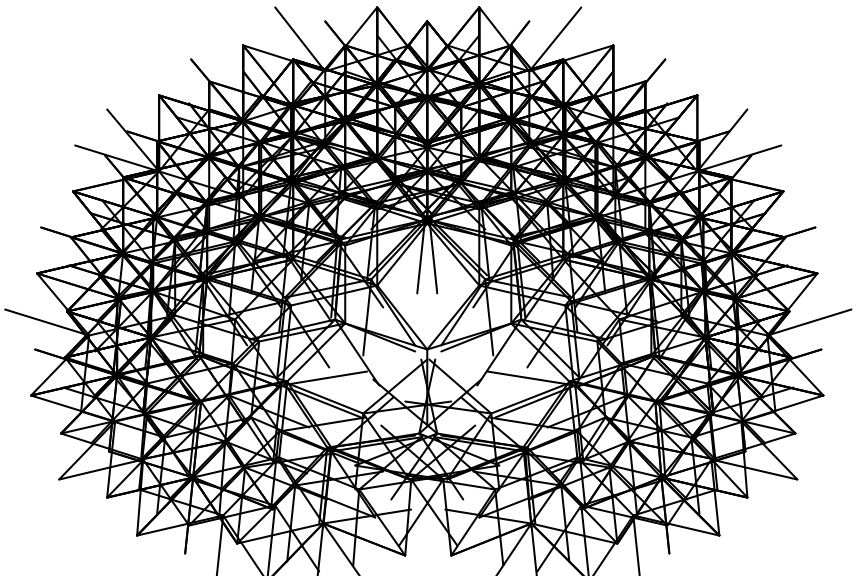
```
dragon := proc(angle, steps)
begin
   L:=plot::Lsys(angle, "BL",
      "L" = "L+R+",
      "R" = "-L-R",
      "L" = Line, "R" = Line,
      "B" = RGB::Black
   ):
   L::generations:=steps:
   plot(L,Axes=None);
end;

tree := proc(angle, steps)
begin
   L:=plot::Lsys (angle, "L",
      "L"="BR[++YL][+OL][--YL][-OL]+",
      "R"=Line, "L"=Line,
      "Y"=RGB::OrangeRed,
      "O"=RGB::Pink,
      "B"=RGB::CadmiumRedDeep
   ):
   L::generations:=steps:
   plot(L,Axes=None)
end
```

Later such procedures can be used in the classroom to experiment with dragon curves using various input parameters. For example,

- `dragon(100,10)`



- `tree(35,6)`



Developing Libraries for *MuPAD*

As our simple examples in this paper show, MuPAD is a command-line-based computer algebra system. It has a programming language similar in many aspects to Pascal. The package is highly universal. It can be used for almost any undergraduate mathematical topic, from logic to sophisticated problems in linear and abstract algebra. We would now like to discuss how the openness and flexibility of *MuPAD* make this system a promising environment for the teaching of mathematics.

There are many ways of developing a customized library for *MuPAD*. Teachers can easily write a set of procedures, test them on appropriate input data, and save them as reusable files on their computers. Here is a simple example that illustrates this process. Suppose we would like to create a library for basic statistical routines. We could start by writing a procedure for calculating the average of *n* numbers and save this procedure in a new library. The following steps accomplish this task.

```
average:=proc()
local n, i, result;
begin
   n:=args(0); result:=0;
   for i from 1 to n do result:=result+args(i) end;
   result:=result/n;
   return (result)
end:

WRITEPATH := "userlib";
write("mylibrary.mb", average)
```

We could then assign our students the task of extending this library[1] by writing simple procedures for other familiar statistical routines. Files such as **`mylibrary.mb`** can be saved in folders such as **`userlib`** in the *MuPAD* directory. To use this library, all a student has to do is to load the file into *MuPAD* when required. This can be as simple as invoking the following two commands:

```
READPATH := “userlib”;
read("mylibrary.mb")
```

The maintenance of a computer lab in a school can be a time-consuming process. By specifying a location on the school network for *MuPAD* libraries, we can reduce this task to the maintenance of a single folder on a network server. This can be done by adding to each local installation of *MuPAD* a small configuration file with both **`READPATH`** and **`WRITEPATH`** commands that point to a network folder. From that moment on, all a teacher needs to do is to update and maintain a single folder on a network server. Recent innovations in the design of *MuPAD* lead us to believe that in the future, the development of customized libraries will be even simpler than in the given example. For more information on using and developing *MuPAD* libraries, we refer to chapter 5 of [2], where the benefits of these features of *MuPAD* are discussed in detail.

MuPAD Computing on the Web

In addition to making it easy to create customized libraries, the *MuPAD* Computing Server is a feature that may completely change how we use computer algebra systems in schools in the future. The *MuPAD* Computing Server is a special server application consisting of the *MuPAD* computing engine and *MuPAD* libraries. Users access the *MuPAD* Computing Server through a web page using standard browsers and perform calculations and solve problems directly on the Web. From the perspective of students and teachers, this tool is completely platform-independent. It can be accessed from a Macintosh, Windows-based PC, or even from a Linux-driven computer. Since all calculations are performed remotely on the server, the needs for specific hardware configurations at the user end are now redundant. Anyone connected to the Internet with any browser will be able to work with *MuPAD*. In order to maintain the *MuPAD* Computing Server, all teachers need to do is to update and load their libraries on the server. In addition, they can post on web pages all management aspects of working with their libraries: a listing of available topics and procedures, instructions on how to use them, tutorials, quizzes, glossaries, and so on. Students can use such *MuPAD* installations in the classroom as well as at home. They do not need to have *MuPAD* installed on their computers.

Conclusion

We have illustrated with simple examples how teachers and students can use the intuitive programming language of *MuPAD* to build mathematical libraries and use them in their teaching and research by integrating them with existing *MuPAD* resources. In our own teaching, we have incorporated such libraries in *WebCT*, where students can explore and experiment on the Web using several computing platform. At this point, our libraries encompass various fields of

[1] The statistical procedures in the basic MuPAD library are quite well developed. However, we will undoubtedly always need new additional routines for specific applications.

mathematics and some applications to other sciences. We have also shown how student projects can be used to expand our repertoire of libraries. Competing environments, based on other computer algebra systems with other features, are being developed elsewhere. What distinguishes the *MuPAD* system from most, if not all, of the other systems, is that it is open, flexible, user-friendly, portable across most computing platforms, and very affordable. We therefore already have here a tool that meets many of our listed needs. The era of device-independent, Web-based teaching and learning of mathematics and science has begun. Let us embrace it and use it for the betterment of global education.

BIBLIOGRAPHY

1. Michelle Baryliuk-Raimbert, Understanding *MuPAD* and Using it to Generate L-Systems, Science College Project, Concordia University, 2002.
2. Jürgen Gerhard et al., *MuPAD* Tutorial (English Edition), Springer, Berlin, 2000.
3. Miroslaw L. Majewski, M. E. Fred Szabo, Integrating *MuPAD* into the Teaching of Mathematics, Proceedings of the 5th Conference on Technology in Mathematics Teaching, Klagenfurt, Austria, August 2001.
4. Miroslaw L. Majewski, *MuPAD Pro* Computing Essentials, Springer Verlag, 2002.
5. M. E. Fred Szabo, Linear Algebra: An Introduction Using Mathematica, Harcourt/Academic Press, Boston, 2001.
6. M. E. Fred Szabo, Linear Algebra: An Introduction Using Maple, Harcourt/Academic Press, Boston, 2002.