International Journal of Computer Discovered Mathematics (IJCDM)
ISSN 2367-7775 ©IJCDM
March 2016, Volume 1, No.1, pp.62-75
Received 25 January 2016. Published on-line 30 January 2016.
web: `http://www.journal-1.eu/`

Discover Euler Math Toolbox

René Grothmann
Catholic University of Eichstätt Germany
e-mail: rene.grothmann@ku.de
web: `http://www.rene-grothmann.de`

Abstract. Euler Math Toolbox (EMT) is a complete toolbox for numerical and symbolic mathematics. The purpose of this paper is to introduce into this program, its features, its design goals, and its intended usage. We also discuss mathematical education in general facing of a modern world with mathematical online tools available from everywhere.

Keywords. Matrix Language, Symbolic Mathematics, Numerical Mathematics.

Mathematics Subject Classification (2010). 97U50

1. Modern Mathematics

If you asked a person on the street what mathematics is you would get a rather disappointing answer. In public opinion, mathematics is all about numbers and computations. If you said you do mathematical research they would surprise you with a complete lack of understanding what could be left for research. How to add and multiply has been known since a long time, hasn't it? No matter how educated they are they would add that they have always been bad in math, with a tone of pride in the voice. Mathematics is nothing to care about or to waste time with.

Why did I start with such frustrating observations? Because it is our job to improve math education. The times have changed. We no longer need to teach tedious computing. None of our students will ever do a division on paper in their lifetime again. In fact, I myself haven't done it for thirty years. I appreciate that we need not forget cultural techniques. But we must be careful not to get blind for the real life. Take a look at the free page of Wolfram Alpha ([4]) to see what is available at the fingertips of our students.

[1]This article is distributed under the terms of the Creative Commons Attribution License which permits any use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.

So what else should we teach? Well, we should concentrate on the hard stuff, the weaknesses, the things you do not learn that easy. The following come to my mind.

- Modelling of a real life problem in mathematics.
- Interpreting and judging the results of our computers.
- Estimating the result beforehand.
- Telling right from wrong.

There certainly are lots more, including some basic knowledge in programming. And, not to forget, there is the visualization of data and of geometrical objects.

Teachers will say that this is not feasible. There is not enough time they claim. And don't you have to teach the basic of mathematics before going into the hard stuff? The point of this paper about mathematical software, however, is to show how to teach mathematics right from the start with the more advanced stuff in mind. There is no longer a need *to waste time with tedious computations.*

This paper is about a software package which can help to free the student from the paper work, and to concentrate on the mathematics. We will also mention and discuss other packages. But let us see what Euler Math Toolbox (see the home page at [1]) can do for math education and also for mathematical research.

2. The Euler Math Toolbox

First of all, we give a very brief introduction into the software. Euler Math Toolbox (EMT) should be considered a simple, but effective interface to a set of tools. The necessary core of these tools is installed with EMT. Some other tools need to be installed separately.

The following components are installed with EMT.

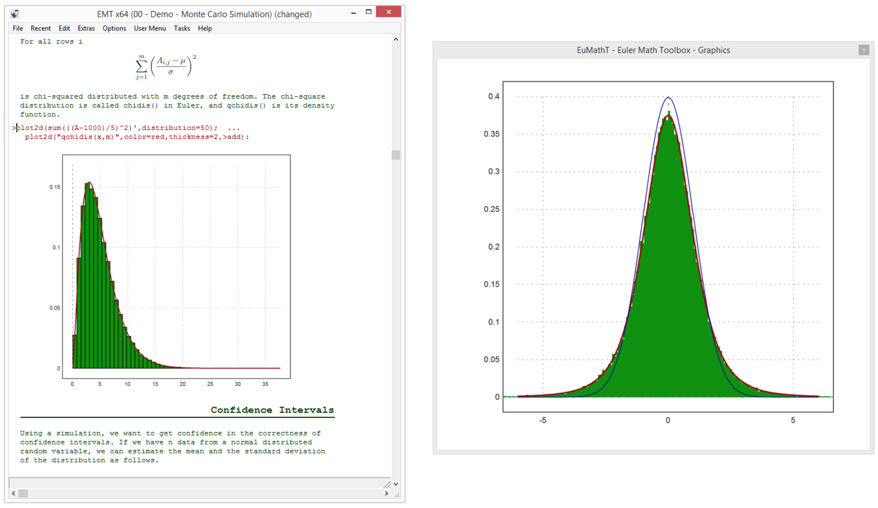
- The core program and its numerical algorithms, partly written in C/C++ and partly in the language of EMT.
- The Maxima system for symbolic computations. Maxima is a free and mature software which is maintained by a group of enthusiast.
- The notebook interface, featuring fully editable notebooks with the commands, the output, comments and images that were inserted into the notebook, and optionally a window to display the graphics separately (see Figure 1).
- A small C compiler for routines that need maximal efficiency. An external compiler for Windows DLL can be used alternatively.
- The LPSOLVE library for optimization.

The following tools need to be installed separately.

- The Latex system for the best display of formulas in notebooks and in plots. Maxima does a simple 2D display of formulas, but Latex looks a lot better. Notebooks can also exported in PDF format if Latex is installed.
- Python can be integrated into EMT, including a full set of libraries. E.g., Anaconda ([6]) can directly be used by EMT from the command line. Python scripts are more efficient than EMT scripts, and EMT can use Python functions for computations.

- Povray scenes can be generated in EMT, and the program can call Povray and integrate the photo realistic images into notebooks.
- Scilab can be called from EMT in a very simple form via pipes. This may be useful in some instances. We talk about the relation of EMT to other matrix languages (especially Matlab) later.

Students today are not used very much to command line interfaces. Even some researchers do not care about learning a command language, just to do some simple computations. There have been attempts to create systems with icons for each basic tool the user may want to use. The program Derive is an example for this ([5]). But it turns out that such an interface is even more difficult to learn and to handle. So EMT sticks with a conservative interface (see Figure 1).


EMT x64 (00 - Demo - Monte Carlo Simulation) (changed)
File Recent Edit Extras Options User Menu Tasks Help
EuMathT - Euler Math Toolbox - Graphics

For all rows i

$$\sum_{j=1}^{m} \left(\frac{A_{i,j} - \mu}{\sigma} \right)^2$$

is chi-squared distributed with m degrees of freedom. The chi-square distribution is called chidis() in Euler, and qchidis() is its density function.

```
>plot2d(sum(((A-1000)/5)^2)',distribution=50);  ...
  plot2d("qchidis(x,m)",color=red,thickness=2,>add):
```

Confidence Intervals

Using a simulation, we want to get confidence in the correctness of confidence intervals. If we have n data from a normal distributed random variable, we can estimate the mean and the standard deviation of the distribution as follows.

FIGURE 1. Euler Math Toolbox, Interface

Consequently, EMT provides a way for the user to find examples and explanations for the things he or she wishes to do. EMT has an extensive help system for this, and additionally a set of tutorial notebooks and examples are available on the computer or in the web (browse [1] for this).

3. NUMERICAL AND SYMBOLIC MATHEMATICS

In the school, all mathematics is symbolic. But life is not like this. Symbolic mathematics is important to study mathematics, but the computations have to be done numerically in the end. And many problems cannot be solved symbolically at all. Some can, but the solution is so tedious it needs a computer anyway. And the symbolic results are often of no use.

Example: . We solve the equation

$$x^3 + x = 4$$

for x. The symbolic solution to such equations has been found centuries ago. But the methods are not taught in schools and seldom in universities, because they fail completely with equations of degree 5. (This fact, of course, is very interesting.) And even if we know the method, we would not be satisfied with the answer. It is a useless expression.

```
>p &= x^3+x-4

                                3
                               x  + x - 4

>plot2d(p,-1,2,grid=6):
>&solve(p,x)[3]

                  sqrt(109)     1/3             1
             x = (--------- + 2)    - --------------------
                     3/2                 sqrt(109)     1/3
                    3                 3 (--------- + 2)
                                            3/2
                                           3

>solve(p,1)
 1.37879670013
```

In EMT, the ampersand & denotes a symbolic term (as in `&solve(p,x)[3]`) or a symbolic definition (as in `p &= x^3+x-4`). In fact, it calls Maxima to handle the expression.

For the numerical core of EMT, symbolic expressions are just strings. Strings, however, can be used to for many numerical functions like `solve` or `plot2d`. By convention, such strings are interpreted as expressions in the variable `x`. This is easier than using functions (which are also available, of course). Conventions like this make EMT easier to handle.

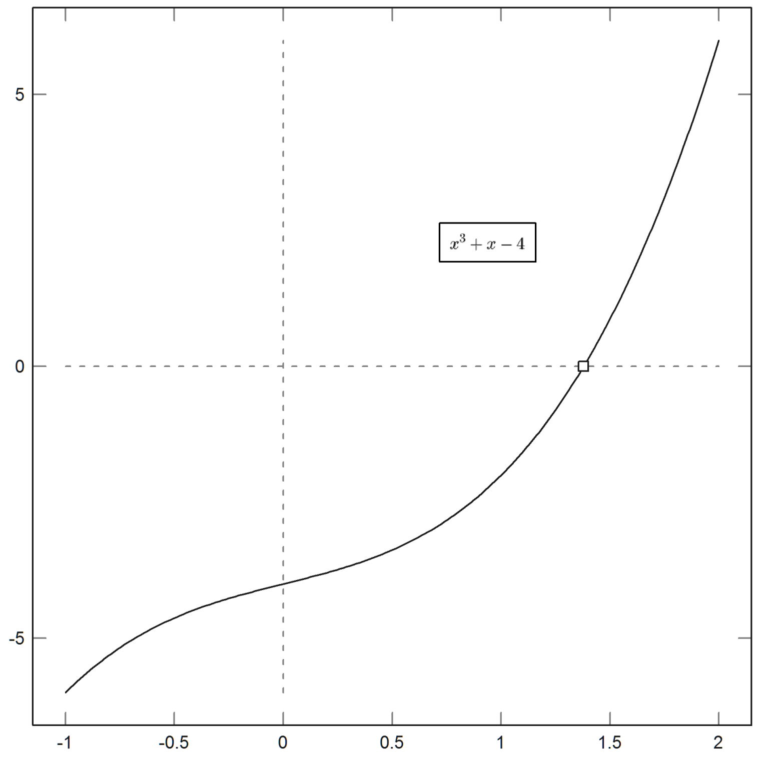


Figure 2. Plot and Zero of $p(x) = x^3 + x - 4$

From the example, you see that symbolic expressions are integrated seamlessly into EMT. There are two copies of the expression, one in the external program

Maxima (running as a separate process), and one in the numerical kernel. EMT takes care to update both when `&=` is used.

This example also shows how useless symbolic results can be. After plotting the function, we clearly see a zero close to 1.4, and we can easily find it numerically. But the symbolic solution of Maxima has no obvious meaning.

But symbolic analysis is not always useless. We can demonstrate that too within this example. For the derivative of the polynomial is $3x^2 + 1$ which is obviously positive, and thus we haver proved that the polynomial has exactly one real zero.

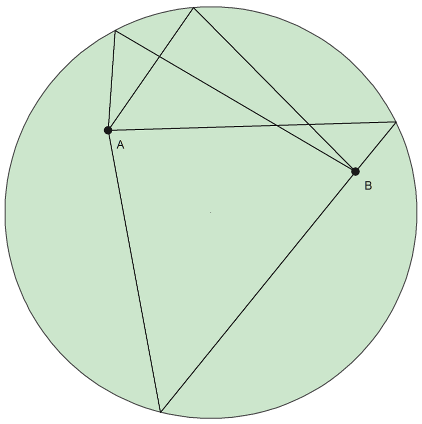


Figure 3. Billiard on a circular Table

For an applied mathematician it is clear that most problems cannot be treated by symbolic means at all. Expanding the previous example, we could simply take polynomials of degree higher than 4. But even if it is possible to find a symbolic solution it is highly impractical.

Example: We take the geometric example of Figure 3. In this figure, we play billiard on a circular table. The aim is to play a ball from A to B. There are two or four possible solutions. Besides the obvious cases none of them can be computed easily. They are the solutions of a fourth order equation.

A geometrical way to imagine the four solution requires some knowledge about ellipses with foci A and B. You are looking for ellipses which are tangential to the circle in some point on the circle. There are two ellipses which touch the circle from the inside, and two which touch it from the outside.

The numerical EMT code to derive the numerical solutions and to plot the figure is the following.

```
>function f(t) &= sqrt((cos(t)-A[1])^2+(sin(t)-A[2])^2) ...
>   + sqrt((cos(t)-B[1])^2+(sin(t)-B[2])^2);
>A=[-0.5,0.4]; B=[0.7,0.2];
>{tmin,tmax}=fextrema("f",0,2pi); tmin, tmax,
 [0.456609,  2.05497]
 [1.6561,  4.46422]
>plot2d("x^2+y^2",level=[0;1],r=1.05,<grid,color=rgb(0.8,0.9,0.8));  ...
>plot2d(A[1],A[2],>points,style="o#",>add); label("A",A[1],A[2]);  ...
>plot2d(B[1],B[2],>points,style="o#",>add); label("B",B[1],B[2]); ...
>tx=tmin|tmax; ...
>for t=tx; plot2d([A[1],cos(t),B[1]],[A[2],sin(t),B[2]],>add); end;
```

As you see, this involves a lot more thinking, modelling and programming than the simple discussion of a polynomial function in the previous example. We will later discuss weather or not the introduction to the details of a system like EMT is worth the effort in schools.

It is this interaction between numerical and symbolic mathematics which is the main idea of Euler Math Toolbox. The second important aspect of EMT are the plots.

4. Visualizing Mathematics

There are is hardly any math that cannot be visualized in some way or another. Besides algebraic skills and logical reasoning, a visual imagination of the situation is one of the columns of mathematical thinking. From the beginning, EMT was also a visualization tool.

It should be emphasized that there are special tools that create much better plots for special purposes. E.g., EMT has statistical plots too, but a package like R has a lot more (see [7]). For graphs and other representations of big data, there are special packages in Python that produce very nice images. For geometry, there are geometry programs like C.a.R. (see [2], and the report in [3]). So it is important that programs can exchange data. Of course, EMT can export and import data in the usual formats like CVS (comma separated values). But, again, that there are specialized formats for specialized programs.

EMT contains a graphical engine in its core which can handle planar and 3D graphs. The 3D part has some restrictions since it is not based on hardware rendering. But it is nevertheless very useful and produces great results. Browse the examples in 1 for the numerous 2D and 3D plots that EMT can do. Some examples use Povray for photo realistic scenes of mathematical content.

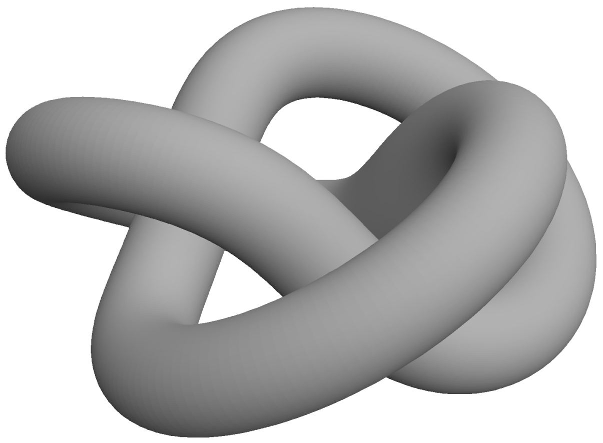


Figure 4. A Knot computed and visualized with EMT

Example: The knot in figure 4 has been computed with a mixture of symbolic and numerical methods. This 3D plot is done with the graphical kernel of EMT. In the examples on [1] you can find the same example rendered by Povray.

Example: Let us study a more simple example which can be handled by elementary analysis. Imagine a ball that is thrown with the same speed but at different angles. What points can be reached with the ball? For simplicity and to be able to handle the problem symbolically, we neglect air friction which would need advanced numerical solvers of EMT.

With the angle α, the speed v and the gravitational acceleration g, we have

$$x(t) = tv\cos(\alpha), \quad y(t) = tv\sin(\alpha) - \frac{gt^2}{2}$$

from which we derive

$$y(x) = \frac{x\sin(\alpha)}{\cos(\alpha)} - \frac{gx^2}{2\cos(\alpha)^2 v^2}.$$

for the curve of the ball. We can now differentiate to α and find the highest point a ball can reach at the distance x.

$$\alpha_x = \arctan\left(\frac{v^2}{gx}\right)$$

These are all non-trivial computations for students at a lower level. Even less trivial is to derive the function $h(x)$ which describes the maximal height we can reach with a ball at distance x.

$$h(x) = \frac{v^2}{2g} - \frac{gx^2}{2v^2}.$$

That is a parabola. The formulas here can be computed with the symbolic part of EMT, Maxima, and you can find the example in 1.

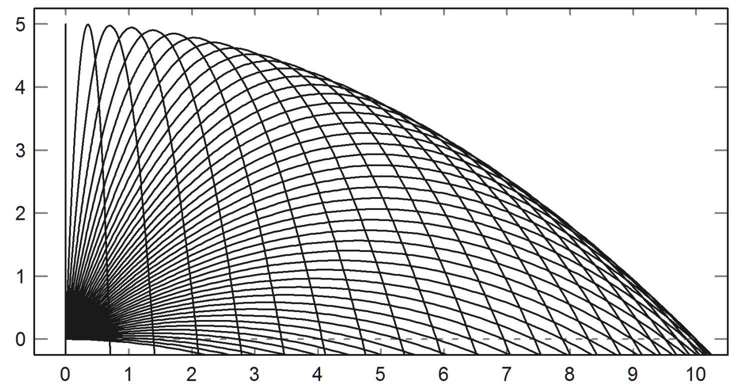


Figure 5. Throwing a ball at various Angles

While the derivations above are nice for a mathematician they have no meaning for a non-mathematician. We need to visualize the trajectories of the balls at various angles. This is very easy. In Figure 5 you see the trajectories. It is immediately clear what is meant by "points we can reach", and it is also easy to conjecture that the function h is a parabola.

```
>v=10; g=10; t=0:0.01:3; al=rad(0:2:90);  ...
>aspect(2);  ...
>plot2d(v*cos(al')*t,v*sin(al')*t-g*t^2/2, ...
>   a=0,b=10,c=0,d=5,grid=6):
```

This is a very simple application of the matrix language we are going to discuss in the next section. As a side remark, EMT understands $0° : 2° : 90°$ instead.

5. Matrix Languages and Matlab

When the project EMT was started in 1988 it was about complex numbers. The primary aim was to be able to handle and visualize computations with complex polynomials and their zeros. It was interactive from the start with a simple to use syntax. At that time, Matlab (see [8]) came up and introduced a nice way of doing mathematics interactively on a computer.

The main idea was to apply operations and functions to each element of a matrix or vector. This is called a matrix language (but the term is now also used in another area of computer science). Together with tools for handling matrices, modifying and extracting sub-matrices, and tools for numerical Linear Algebra, this makes a powerful software with a short command syntax. Maxima and some other algebra system use a similar feature, or they have at least the function "map" as a weak replacement.

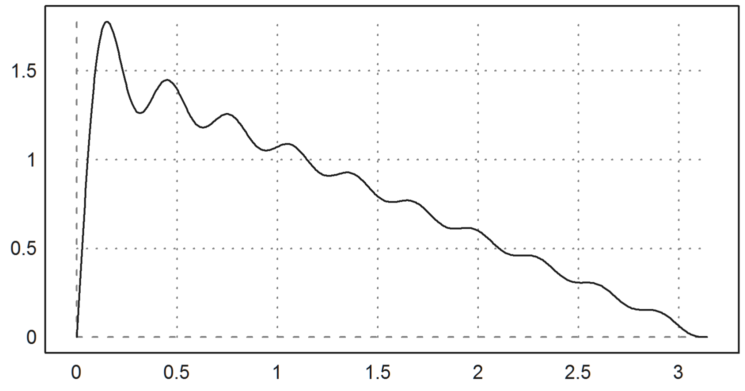


Figure 6. Trigonometric Sum

Example: As an example, we compute

$$f(t) = \sum_{k=1}^{20} \frac{\sin(kt)}{k}$$

for a vector of values simultaneously in EMT.

```
>t=linspace(0,pi,500); k=(1:20)';
>s=colsum(sin(k*t)/k);
>aspect(2); plot2d(t,s):
```

The result is figure 6. For an explanation, we have a row vector `t` and a column vector `k`. The term `sin(k*t)/k` combines the row vector and the column vector to a matrix M applying the functions and operators to each element. I.e.,

$$m_{i,j} = \frac{\sin(k_j t_i)}{k_j}.$$

Then `colsum` takes the sums of the columns and forms a new row vector with the desired values $f(t_i)$. Note that the first matrix generated will be `k*t`. Then the sine function is applied to all elements of this matrix. And after that, each row is divided by k_j.

The syntax of Matrix is a bit different, and some things may not work in Matlab out of the box. EMT expanded the capabilities of matrix generation quite a bit. EMT was never designed to be compatible to Matlab. In fact, many ideas of Matlab were considered to be confusing, e.g., the use of round brackets for functions and matrix elements.

The plot function `plot2d` can plot all sorts of things. We used it to plot a function by values stored in vectors. However, in EMT, we could also use an expression, if the function is given by an expression or by a programmed function in the language of EMT. Of course, it is possible to plot many functions, either one by one, or all at once using a matrix with the values of each function in the rows of the matrix.

Example: In figure 7 we plot the first 5 Chebyshev Polynomials. We use an expression in the variable `x` and a column vector `k` to force `plot2d` to draw all functions for all k_j. The vector `3:7` of colors is applied one by one to the functions.

```
>k=(1:5)'; plot2d("cheb(x,k)",-1,1,color=3:7):
```

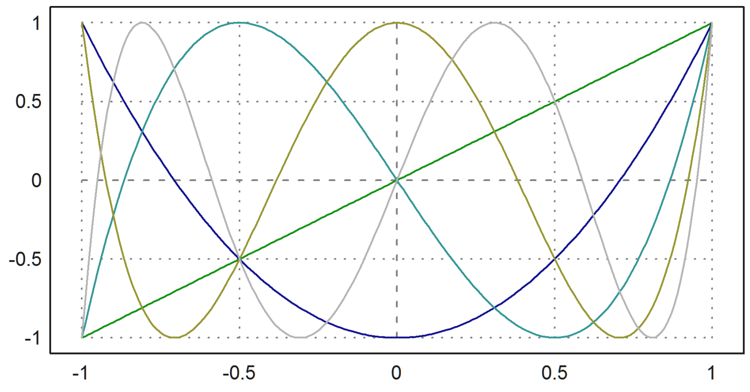


FIGURE 7. Chebyshev Polynomials

Example: A nice thing that EMT can do is Monte-Carlo simulation. One of its many statistical functions generates normal distributed random variables.

In the following, we count how often a random walk starting in 0 with 0-1-normal distributed steps exceeds the value 10 before the 100-th step. The following function utilizes the matrix language effectively to simulate a 100 thousand random walks. It does also plot the first 20 random walks and the maximal and minimal values any random walk ever attains (in figure 8).

```
>N=100000; A=cumsum(normal(N,100));
>sum(sum(A>10)'>0)/N -> " %"
 28.997 %
>vmax=max(A')'; vmin=min(A')';
>plot2d(vmax_vmin,color=red,thickness=2);
>plot2d(A[1:20],>add):
```

A bit of explanation for the EMT code should be provided.

- The function `cumsum` takes the cumulative sum of the random values in each row. So each row is now a random walk.
- The comparison `A>10` is vectorized by the matrix language, and yields 1 (true) in each element of `A` that is larger than 10, else 0 (false).
- If we sum up the rows then the rows with sum not equal to 0 are the ones that ever exceeded the value 10.

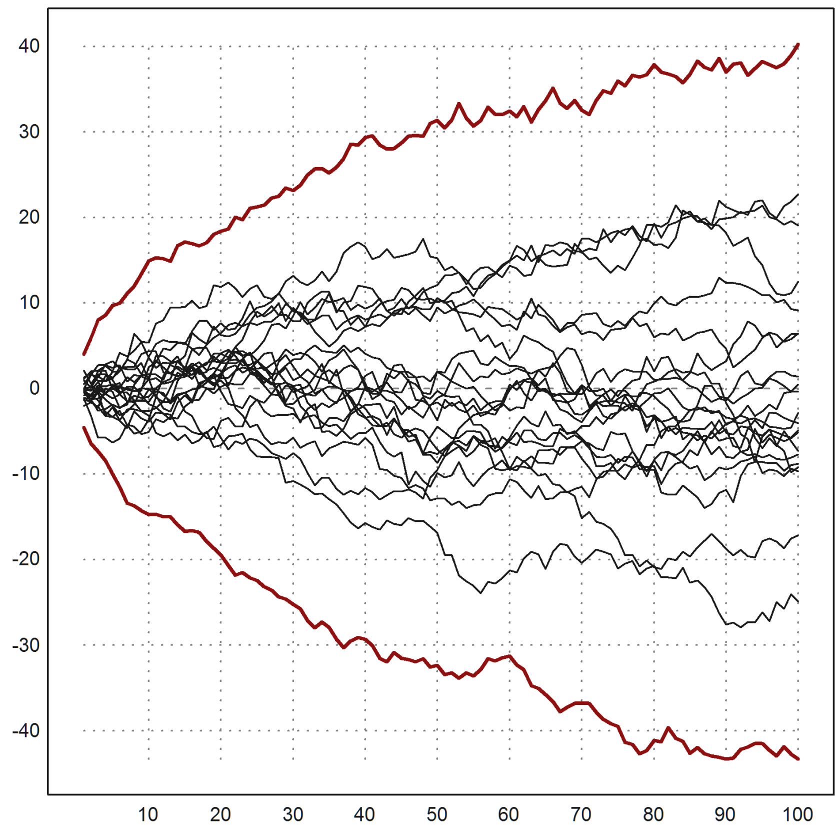


Figure 8. Random Walks

- We make this a row vector (by transposing) and check how many are not 0.
- The result is divided by the number of runs and converted and printed to percentage.

This looks all very efficient and mighty, *but also tricky*. Because of that, we have our doubts that matrix languages are really the wonderful thing they seem to be at first sight. Obviously, we could just as well fill our matrix in a loop. Loops are easy to understand and easy to learn. Moreover, they are indispensable for any programmer. So students do well to learn about loops. Then, why don't we teach loops instead of matrix languages?

The reason is that Matlab and EMT both are inefficient when it comes to loopy programming. Of course, we could use C to help out, and EMT contains a small C compiler. Or, alternatively, we could use Python, which is available for scripting in EMT too.

But then, we will ask ourselves: Why not use Python to start with? The answer to this is the Sage Math project (see [9]). But it has its shortcomings too. We will delve further into this discussion later.

6. Euler Math Toolbox and Math Education

As explained in the first section, software packages can change the curricula of schools, even if they are used only at the basic level, not much more advanced than a calculator with graphics capabilities. EMT can be used on just this level, either for symbolic or for numerical computations.

A typical category of school problems involve geometric applications of calculus like computing tangent functions to curves. Since the students have to compute everything by pen and paper the problems are usually restricted to very simple functions. Moreover, we should remember that the students have to do the modelling *and* the computation. The limited time restricts the tractable problems even further.

What if we have a mighty symbolic and numerical tool at our hands? As the following example shows we can suddenly do a lot more.

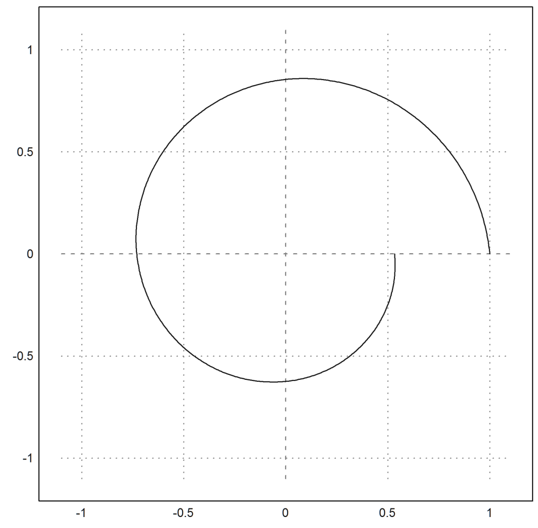


FIGURE 9. Logarithmic Spiral

Example: This example is just a little bit above the high school level. The reason is not the complexity of the concept, but the inability to compute specific examples. This is a clear indication that we should use computers to get things done, even in schools, but at least in low level studies in universities.

The idea of the length of a curve $\gamma(t)$ is not difficult geometrically, and also its physical interpretation is easy to understand. The speed vector is simply $\gamma'(t)$. Consequently, the length of the curve is the integral of the speed $\|\gamma'(t)\|$ over the time passed.

Let us compute and plot an example in EMT (see figure 9).

```
>a=0.1;
>function gx(t) &= cos(t)*exp(-a*t);
>function gy(t) &= sin(t)*exp(-a*t);
>t=linspace(0,2pi,500); sx=gx(t); sy=gy(t);
>plot2d(sx,sy,r=1.1);
```

We can also compute the curve length in symbolic form with Maxima. For this, we define a symbolic function for the speed at time `t`. The functions is not printed here. We can then symbolically integrate this function, and in this example, the integral can be evaluated in symbolic form.

If that weren't the case we would have to integrate numerically. The example shows the command for this too. Note, that we derived the speed with the symbolic help, but evaluated the integral numerically.

```
>&sqrt(diff(gx(t),t)^2+diff(gy(t),t)^2);
>function speed(t) &= trigsimp(%);
>&integrate(speed(t),t,0,2*pi), %()

                                   - 2 pi a
                       2       1   E
                 sqrt(a  + 1) (- - ---------)
                               a       a

 4.68838666031
>integrate(&speed(x),0,2*pi)
 4.68838666031
```

The manipulations by `trigsimp` are not difficult, even at a low level. They require only basic knowledge of trigonometry. However, this example can only be done exactly by chance. For an ellipse, the same becomes impossible.

```
>a=0.1;
>function gx(t) &= cos(t);
>function gy(t) &= sin(t)*a;
>t=linspace(0,2pi,500); sx=gx(t); sy=gy(t);
>plot2d(sx,sy,r=1.1);
>&sqrt(diff(gx(t),t)^2+diff(gy(t),t)^2); function speed(t) &= trigsimp(%)

                          2          2
                  sqrt((a  - 1) cos (t) + 1)

>integrate(&speed(x),0,2*pi)
 4.0639741801
```

We cannot integrate this function exactly. But of course we can use a numerical method to get a good result. The adaptive method of EMT yields 16 correct digits. You do not need EMT for this value. You could just enter the numerical integral into Wolfram Alfa (see [4]) and get the same value.

But with EMT, we can also demonstrate that the value is reasonable. We take some points on the ellipse and add the lengths of the line segments between the points (see figure 10). In the matrix language of EMT, this is can be done with a very compact code.

```
>t=linspace(0,2pi,20);
>dx=differences(gx(t)); dy=differences(gy(t));
>sum(sqrt(dx^2+dy^2))
 4.05260997603
>aspect(2);
>plot2d(gx(t),gy(t),a=-1,b=1,c=-0.5,d=0.5,color=blue,>addpoints):
```

The matrix language sure is efficient here. But the same could be achieved with just a little code in any computer language with a loop, or even an Excel sheet. The main point is that we can compute it. We do not have to trust in error prone symbolic calculations. Yes, in fact, numerical results are often more reliable than symbolic calculations which have to be evaluated numerically in the end anyways.

There is one more important aspect which we did not mention yet. Programs such as EMT are enormously useful for the teacher.

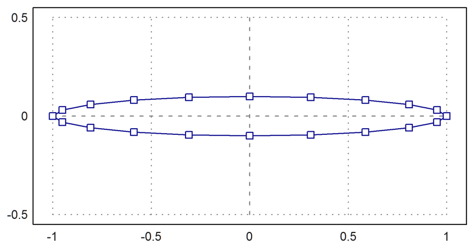


FIGURE 10. Estimating the Length of an Ellipse

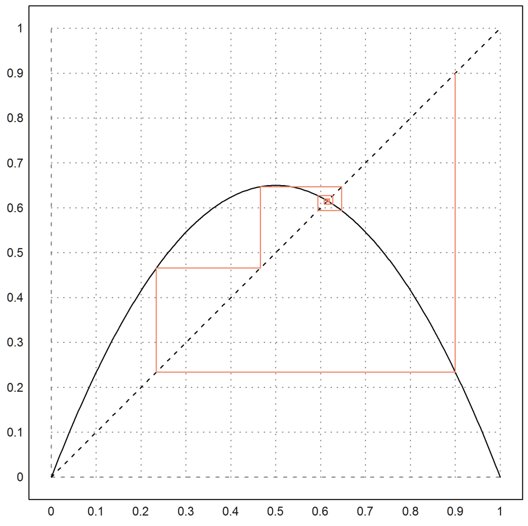


FIGURE 11. Convergence of $x_{n+1} = ax_n(1 - x_n)$, $x_0 = 0.9$

- Mathematical concepts can be demonstrated visually. For this, enough examples have been given here. Plots are more exact and more beautiful than a sketch on the blackboard. The necessary mathematical analysis of the situation can be related to the visual demonstration later to ensure the students of their understanding. We should never underestimate the value of visual representations.
- Even purely analytical concepts like the convergence of sequences can be visualized (see figure 11), or the convergence of the partial sums of a power series. This can also be animated if necessary.
- Tedious and error prone computations can be demonstrated with the computer. E.g., EMT has a function `pivotize` which can be used in the Gauß algorithm and in the Simplex algorithm. Together with a preselected fractional format the teacher can quickly demonstrate these algorithms step by step without getting lost in error prone computations at the blackboard.
- Answers given by students in tests can be checked for correctness, either symbolically or numerically. On a higher level, there are often different ways to represent a result which are equivalent, but not obviously so. This can also be checked with a software.

7. Verdict and Chances

To the author it has long been obvious that software will enter our schools and our universities sooner or later. The current state of the modern school is still in its infancy. One reason is the ongoing lack of proper technical equipment. The other reason is a our inability to imagine other forms of teaching.

At this point, it is important to stress that we must not let *the primary education of mathematics go in favor of software skills*. EMT is not meant like this. It is meant to facilitate, not to hinder understanding. In fact, the logical side of a problem and the modelling is what students are able to do the least. So we need to concentrate on this. A software can help, but it is no replacement of proper thinking and teaching.

Furthermore, it is important that *we do not educate for specific software systems*, how wide spread they may be in the industry. If we replace education by training, there is no end. There are way to many systems out there to be taught and the future will bring many more. We need to concentrate on the common basics. These are visual and logical understanding as well as basic computational skills.

Now, Euler Math Toolbox is not the only package available for the purpose. Matlab, Maple or Mathematica are other choices. The author tried to convince you in this paper that EMT is a good choice. Your mileage may vary, however.

For research, all mentioned systems are of little use. Researchers either use specialized software or a general programming language. In the end, it is far more rewarding for a math student to learn a programming language than the intricacies of Matlab. In fact, *Matlab has hindered the development of open libraries* just by its presence.

So, why not go ahead and use Euler Math Toolbox and its mighty companion Maxima? You will benefit from it, even if you use it only on a very basic level. Maybe you benefit the most if you keep it on a basic level, for the benefit of mathematical education.

References

[1] Euler Math Toolbox, `http://www.euler-math-toolbox.de/`.
[2] C.a.R., `http://car.rene-grothmann.de/`.
[3] René Grothmann, The Geometry Program C.a.R., International Journal of Computer Discovered Mathematics, Volume 1 (2016), No. 1, 45-61, `http://www.journal-1.eu/2016-1/Grothmann-CaR-pp.45-61.pdf`
[4] Wolfram Alpha, `http://www.wolframalpha.com/`.
[5] Derive, `http://www.chartwellyorke.com/derive.html`.
[6] Anaconda Python, `https://www.continuum.io/why-anaconda`.
[7] The R Project, `https://www.r-project.org/`.
[8] Matlab by MathWorks, `http://www.mathworks.com/`.
[9] Sage Math Project, `http://www.sagemath.org/`.