Bulletin of the *Transilvania* University of Braşov • Vol 8(57), No. 1 - 2015
Series III: Mathematics, Informatics, Physics, 129-137

FROM THE NUMERICAL SOLUTION TO THE SYMBOLIC FORM

Ernest SCHEIBER[1]

Abstract

We present the possibility to obtain the closed form of the solution of a differential equation problem from the numerical solution using the *Eureqa* software. The procedure succeeds when the closed form exits and it is relatively simple.

2012 *ACM Subject Classification:* G.1.7, G.1.8, I.1.1
Key words: closed form, initial value problem, partial differential equation, *Eureqa.*

1 Introduction

In this note we present the possibility to obtain the closed form of the solution of an ordinary differential equation (ODE) problem and of a partial differential equation (PDE) problem from the numerical solution, at least when this closed form exists and is relatively *simple.*

The tool that we shell use is a very exciting *symbolic regression* ([3]) software called *Eureqa* from *Nutonian, Inc.,* [7]. The license of the product is free for academic and an accredited university. In [5] there is given a very good presentation of *Eureqa.* There is presented an example of simplifying a trigonometrical expression with *Eureqa.*

The cases when a differential equation problem has a closed form of the solution are not very large. Usually the numerical solution satisfies the required practical needs.

The procedure followed by us consists in:

1. The numerical results are stored in a text file, as comma separated values. The numerical solution is a table whose columns are the functions arguments and the values of the functions, while each row is a data row. Each row will be a line in the text file.

2. The file is imported to *Eureqa,* it is defined the search (a column is the unknown function of some other columns) and it is launched the search of the closed form. All these actions are manually operated. The *Eureqa* documentation details these operations.

This approach is not an universal panacea. It works only when the expression of the solution is relative simple. Several Computer Algebra Systems (CAS, [6]) offer alternatives that are often more effective, but the approach of *Eureqa* is related to genetic programming.

[1]Faculty of Mathematics and Informatics, *Transilvania* University of Braşov, Romania, e-mail: scheiber@unitbv.ro

The following examples presents cases when *Eureqa* obtained the closed form. For that the examples can be reproduced there are given the codes to generate the numerical solution of the problems, but any other procedure or software may be used.

2 The case of ODE

To solve an ODE we use *Scilab,* a *Matlab* type software, freely distributed, [9]. Two examples treats an initial value problem (IVP) and a boundary value problem (BVP), respectively.

Exemple 2.1. *The initial value problem*

$$\begin{aligned} y'(x) &= \frac{1}{\cos x} - y(x)\tan x, \\ y(0) &= 1. \end{aligned}$$

The solution is $y(x) = \sin x + \cos x$. The *Scilab* script to solve the IVP is

```
1 deff('dy=f(x,y)','dy=1.0./cos(x)-y.*tan(x)')
2 x=0:0.01:1;
3 x0=0;y0=1;
4 y=ode(y0,x0,x,f);
5 csvWrite([x',y'],'. . .\ode.csv')
```

From *Eureqa,* the obtained results are given in Figure 1.

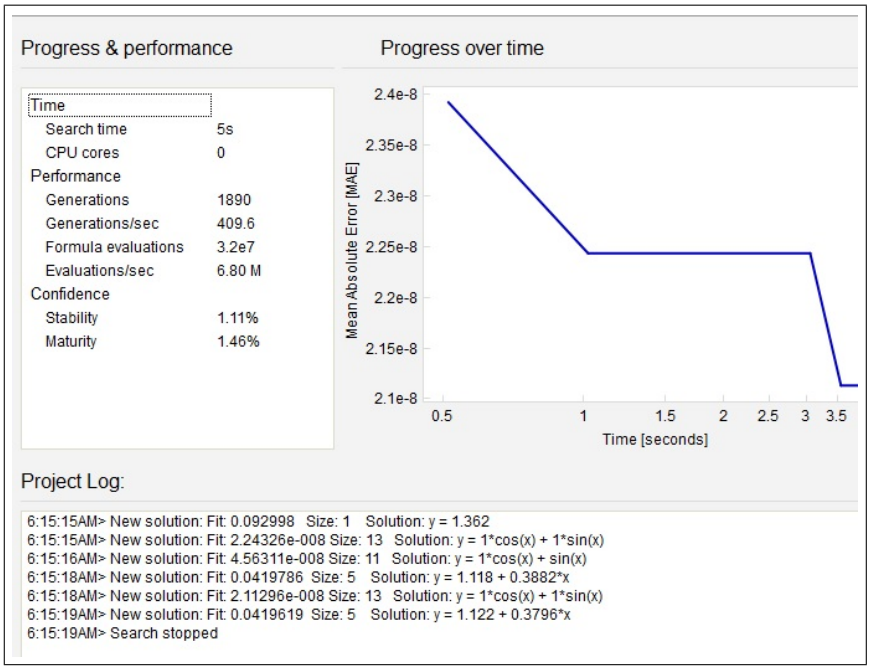

Figure 1: *Eureqa* results for Example 2.1.

Exemple 2.2. *The boundary value problem*

$$\begin{aligned} y''(x) &= 2 - y(x), \quad x \in [0, \frac{\pi}{2}], \\ y(0) &= 0, \\ y(\frac{\pi}{2}) &= 1. \end{aligned}$$

The solution is $y(x) = 2 - \sin x - 2\cos x$. The *Scilab* function **bvode** is used, [1]. The *Scilab* script to solve the BVP is

```
exec('. . .\bvp.sci',-1)
[x,y]=bvp();
csvWrite([x',y'],'. . .\bvp.csv')
```

where *bvp.sci* is the function

```
function [x,w]=bvp()
    deff('f=fsub(x,z)','f=2-z(1)');
    deff('df=dfsub(x,z)',['df(1)=-1','df(2)=0']);
    deff('g=gsub(i,z)',['a=[z(1),z(1)-1]','g=a(i)']);
    deff('dg=dgsub(i,z)',['a=[1,0;1,0]','dg=a(i,:)']);
    deff('[z,dmval]=guess(x)',['z=0','dmval=0']);
    n=1;
    m=2;
    a=0;
    b=%pi/2;
    x=a:0.01:b;
    fixpnt=0;
    zeta=[0,%pi/2];
    ipar=zeros(1,11);
    ipar(3)=1;ipar(4)=2;ipar(5)=2000;ipar(6)=200;ipar(7)=1;
    ltol=[1,2];
    tol=[1.e-7,1.e-7];
    z=bvode(x,n,m,a,b,zeta,ipar,ltol,tol,fixpnt,fsub,dfsub,gsub,dgsub,guess);
    w=z(1,:)
endfunction
```

From *Eureqa,* the obtained results are given in Figure 2.
This BVP may be reformulated as a Dirichlet problem and be solved for example, by the finite element method, [4] p.46,

$$\begin{aligned} \triangle u + u &= 2, \quad \text{in } [0, \frac{\pi}{2}] \times [0, 1], \\ u(0, y) &= 0, \quad y \in [0, 1], \\ u(\frac{\pi}{2}, y) &= 1, \quad y \in [0, 1], \\ \frac{\partial u}{\partial n}(x, y) &= 0, \quad x \in [0, \frac{\pi}{2}], y \in \{0, 1\}. \end{aligned}$$

3 The case of PDE

We shall use *FreeFem++* to solve PDE with the finite element method (FEM), [8, 2]. The software defines a C++ idiom with an extension to handle the concepts of FEM. It runs on Windows, Unix, Macs machines. For Windows a compiled version is provided.

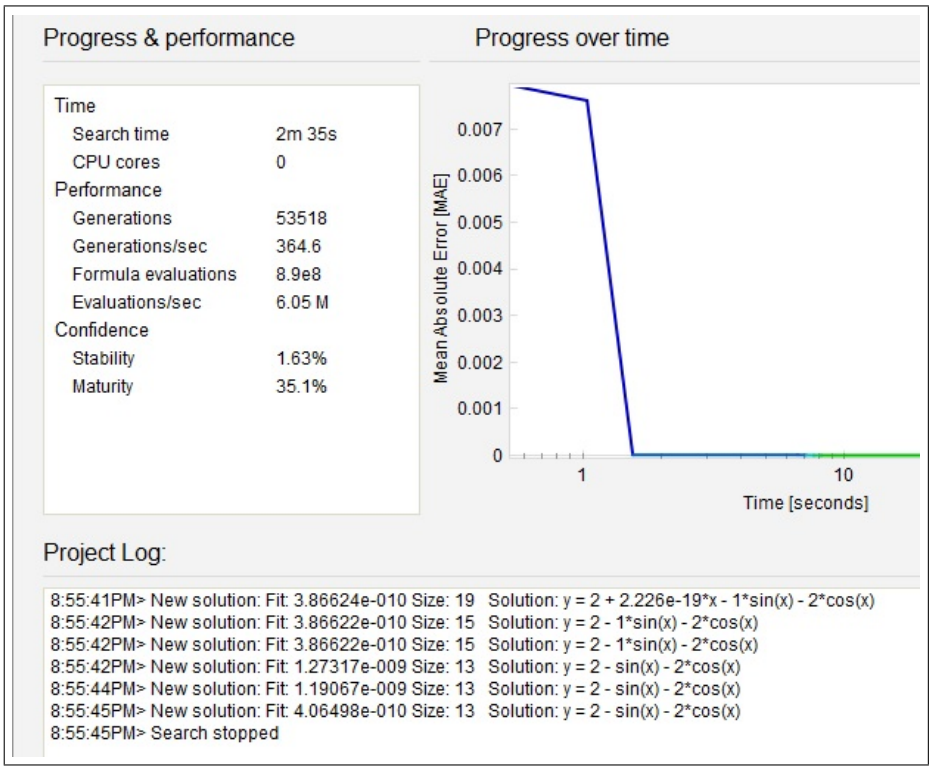

Figure 2: *Eureqa* results for Example 2.2.

This extension allows to easily translate the mathematical formula into *FreeFem++* statements. *FreeFem++* scripts can solve problems in 2D and 3D. *FreeFem++* is distributed with a free software license, GNU Lesser General Public License (LGPL).

Exemple 3.1. *The Dirichlet problem for the Laplace equation*

$$\begin{aligned} \triangle u &= 0 \quad \text{in } \Omega = \{(x,y) : x^2 + y^2 < 1\}, \\ u|_{\partial\Omega} &= 3x^2 + y^2 \end{aligned}$$

The solution is $u(x,y) = 2 + x^2 - y^2$. Thel *FreeFem++* script to solve is

```
1  border C(t=0,2*pi){x=cos(t);y=sin(t);}
2  mesh Th=buildmesh(C(50));
3  fespace Vh(Th,P1);
4  func s=2+x^2-y^2;
5  Vh u,v,ps=s;
6  solve prob(u,v)=int2d(Th)(dx(u)*dx(v)+dy(u)*dy(v))+on(C,u=3*x^2+y^2);
7  real error=sqrt(int2d(Th)((ps-u)^2));
8  cout<<"Error = "<<error<<endl;
9  plot(u,value=true);

11 savemesh(Th,"ex1",[x,y,u]);
```

The function `savemesh` generates two files *ex1.points* and *ex1.faces*. In a simple Java program, we extract from *ex1.points* the needed data and store them as a comma separated values.

The displayed error is 0.00129556.

The results produced by *Eureqa* are given in Figure 3.

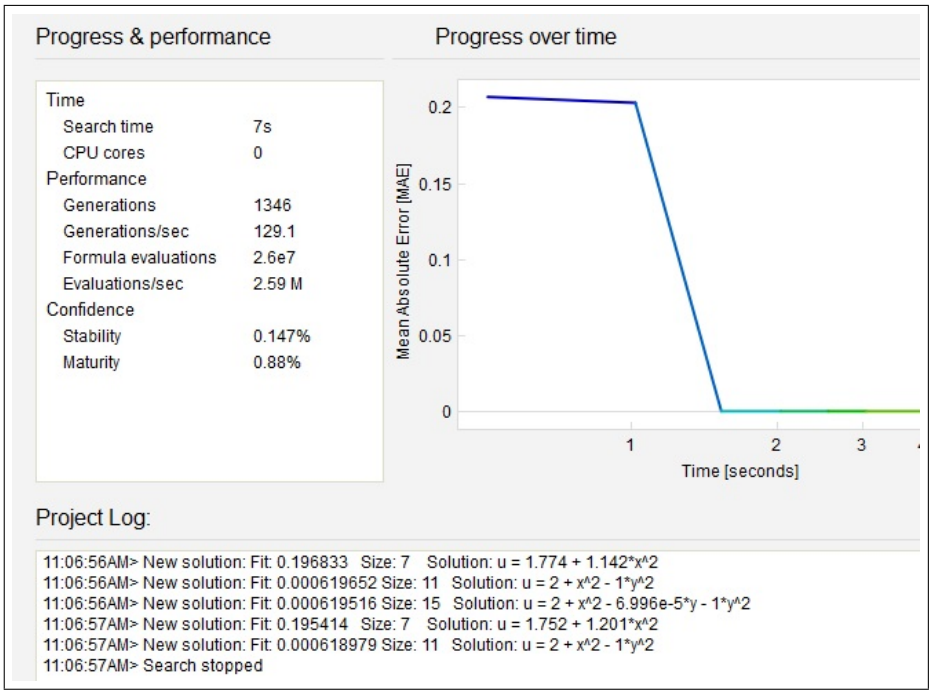

Figure 3: *Eureqa* results for Example 3.1.

Exemple 3.2. *The boundary value problem for a parabolic PDE*

$$\begin{aligned} \frac{\partial u}{\partial t} - a^2 \triangle u &= x^2 + y^2 - 4a^2 t, \quad \text{in } \Omega = [0,1]^2,\ t \in [0,T] \\ u(0,x,y) &= 0, \quad (x,y) \in \Omega, \\ u(t,x,y) &= t(x^2+y^2), \quad (x,y) \in \partial\Omega, t \in [0,T]. \end{aligned}$$

with $a = 0.1, T = 1$.

The solution is $u(t,x,y) = t(x^2+y^2)$. The *FreeFem++* script ([2]) to solve this problem is

```
1 mesh Th=square(16,16);
2 fespace Vh(Th,P1);
3 real dt=0.1,a=0.1,T=3;
4 Vh u,v,uu,f,g;
5 problem eq(u,v)=int2d(Th)(u*v+dt*a^2*(dx(u)*dx(v)+dy(u)*dy(v)))-
6       int2d(Th)(uu*v+dt*f*v)+on(1,2,3,4,u=g);
7 real t=0;
8 uu=0;
```

```
savemesh(Th,"ex0",[x,y,uu]);
for(int i=0;i<T/dt;i++){
  t=t+dt;
  f=x^2+y^2-4*a^2*t;
  g=t*(x^2+y^2);
  eq;
  real error=sqrt(int2d(Th)((u-g)^2));
  uu=u;
  cout<<"t="<<t<<" Error="<<error<<endl;
  savemesh(Th,"ex"+t,[x,y,u]);
}
```

At each time level, the *FreeFem++* generated results are saved. Again, a Java program extracts from all these files the needed data and generates a single file for *Eureqa.*

From *Eureqa* we obtain the results given in Figure 4.

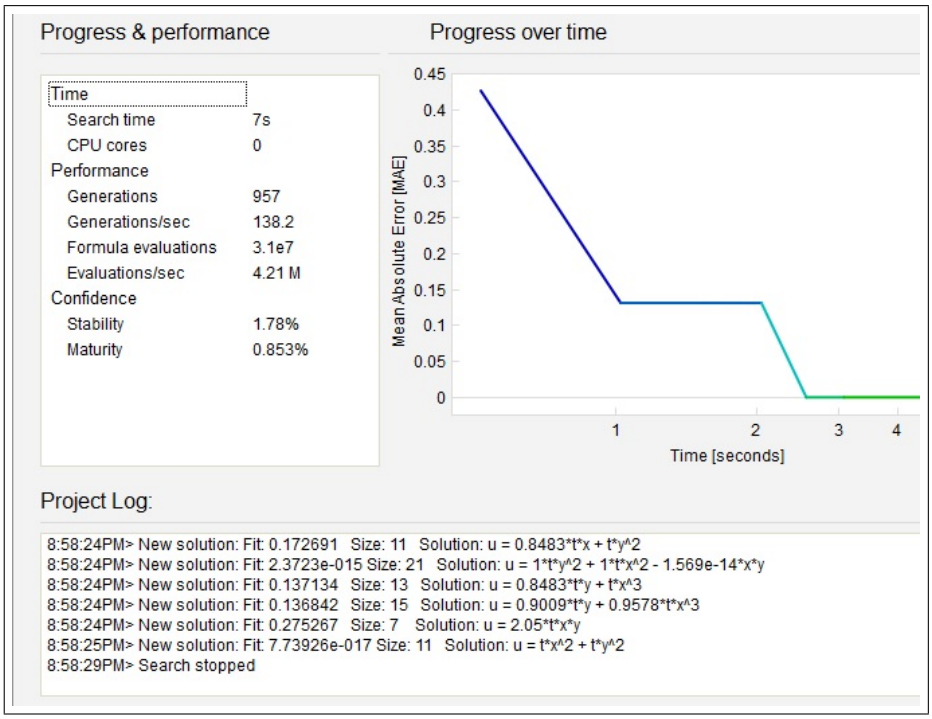

Figure 4: *Eureqa* results for Example 3.2.

Exemple 3.3. *The boundary value problem for a hyperbolic PDE*

$$\begin{aligned}
\frac{\partial^2 u}{\partial t^2} - a^2 \triangle u &= 2x^2 y - 2a^2(t^2+t)y, \quad \text{in } \Omega = [0,1]^2,\ t \in [0,T] \\
u(0,x,y) &= 0, \quad (x,y) \in \Omega, \\
\frac{\partial u}{\partial t}(0,x,y) &= x^2 y, \quad (x,y) \in \Omega, \\
u(t,x,y) &= (t^2+t)x^2 y, \quad (x,y) \in \partial\Omega, t \in [0,T].
\end{aligned}$$

with $a = 0.1, T = 1$.

The solution is $u(t, x, y) = (t^2 + t)x^2y$. With a similar calculation scheme to that used in the previous example, the *FreeFem++* script is

```
mesh Th=square(16,16);
fespace Vh(Th,P1);
real dt=0.001,a=0.1,T=1;
Vh u,v,u1,u2,f,g,sol;
problem eq(u,v)=int2d(Th)(u*v+dt^2*a^2*(dx(u)*dx(v)+dy(u)*dy(v)))-
     int2d(Th)(2*u1*v-u2*v+dt^2*f*v)+on(1,2,3,4,u=sol);
real t=dt,error;
u2=0;
g=x^2*y;
u1=u2+dt*g(x,y);
error=sqrt(int2d(Th)((u1-sol)^2));
ofstream e("errors.txt");
e<<t<<" "<<error<<endl;
sol=(t^2+t)*x^2*y;
savemesh(Th,"ex0",[x,y,u2]);
savemesh(Th,"ex"+t,[x,y,u1]);
for(int i=1;i<T/dt;i++){
  t=t+dt;
  f=2*x^2*y-2*a^2*(t^2+t)*y;
  sol=(t^2+t)*x^2*y;
  eq;
  error=sqrt(int2d(Th)((u-sol)^2));
  u2=u1;
  u1=u;
  e<<t<<" "<<error<<endl;
  savemesh(Th,"ex"+t,[x,y,u]);
}
```

At each time level the error of the numerical solution is computed. The plot of the evolution of the errors is given in Figure 5.

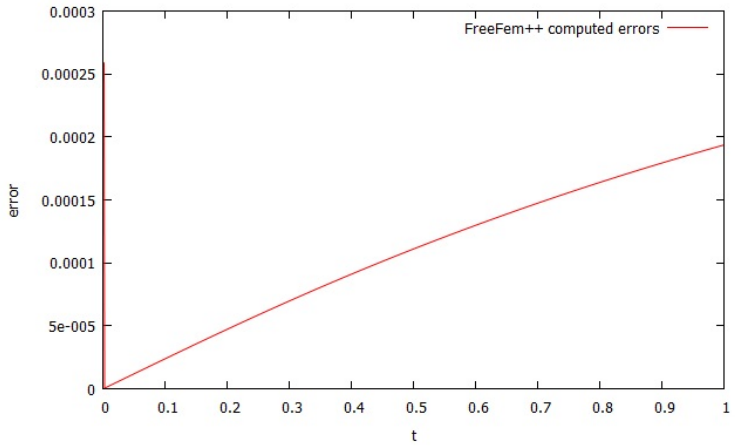

Figure 5: The errors in Example 3.3.

The results of *Eureqa* are given in Figure 6

It may be observed that *Eureqa* computes a better approximation of the numerical

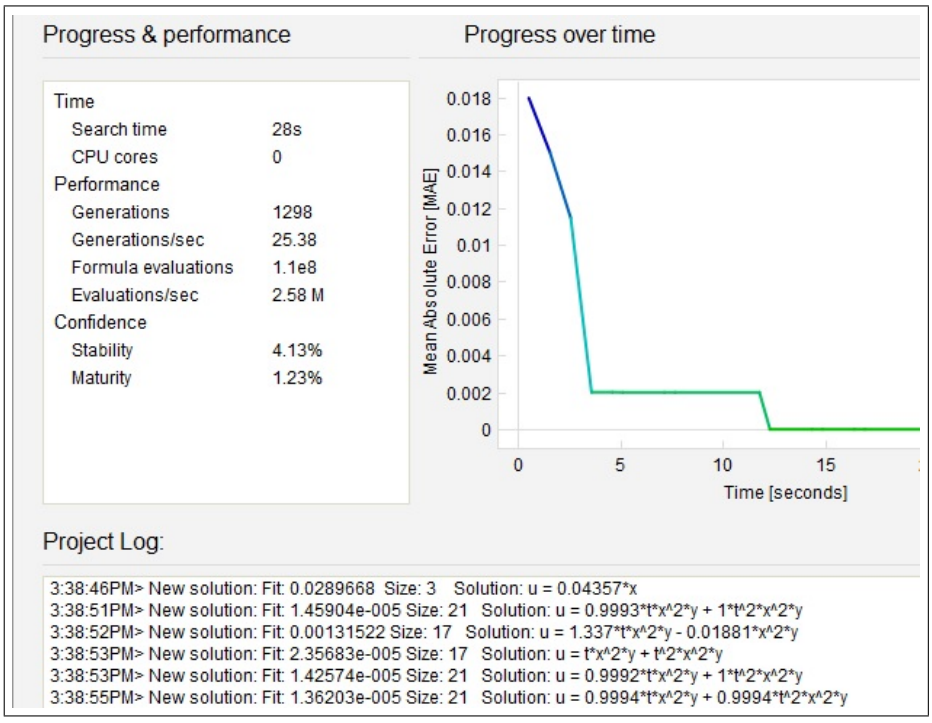

Figure 6: *Eureqa* results for Example 3.3.

solution than the exact solution. The reason is that the errors in the *FreeFem++* computation are larger than the tolerance used by Eureqa.

4 Conclusions

The possibility to obtain the closed form of the solution of a differential equation problem, at least when this closed form exists and is relatively *simple,* is exemplified. The *Eureqa* software accomplishes this task.

In a real scenario the solution of the problem is unknown and the results given by *Eureqa* require further verifications. It would be useful that *Eureqa* offers the possibility to export the results and / or the possibility to attach a user defined callback method.

References

[1] Ascher U., Christiansen J., Russell R. D., *Collocation software for boundary-value ODEs.* ACM trans. math software, **7** (1981), no. 2, 209-222.

[2] Hecht, F., *New development in FreeFem++.* J. Numer. Math. **20** (2012), no. 3-4, 251-265.

[3] Koza J. R., Example of a run of genetic programming, `www.genetic-programming.com/gpquadraticexample.html`

[4] Langtangen H. P., *A FEniCS Tutorial.* `http://fenicsproject.org/_static/tutorial/fenics_tutorial_1.0.pdf`, 2011.

[5] Stoutemeyer R. D., *Can the Eureqa symbolic regression program, computer algebra and numerical analysis help each other?* Notices AMS, **60** (2013), no. 6, 713-724.

[6] * * *, `en.wikipedia.org/wiki/List_of_computer_algebra_systems`

[7] * * *, `www.nutonian.com`

[8] * * *, `www.freefem.org/ff++`

[9] * * *, `www.scilab.org`

[10] * * *, `http://en.wikipedia.org/wiki/List_of_finite_element_software_packages`