Proposal of A New Elements for the Mathematica Program to Solve Partial Differential Equations

Asst. Prof. Sadiq A. Mehdi

Al-Mustansiriya University

College of Education Computer science

Sadiqmehdi71@yahoo.com

Eng. Methaq M. Ali

Ministry of Youth and Sport

Methaqmahdi@yahoo.com

Abstract: *In this paper we have designed new proposal elements for the Mathematica program in order to support model with partial differential equations (PDEs). Using these, we can specify initial and boundary values problem. We have also implemented a prototype that generates input to a PDE-solver from a specification in Mathematica using our extensions.*

Keywords: Object-oriented modeling language, partial differential equations.

1- Introduction

Mathematical models containing differential equations are used. In models where functions of one dependent variable occur, these equations are ordinary differential equations (ODEs). Such models can be written in a high-level, object-oriented modeling language called Mathematica [1][2], where models are defined by writing the equations using a special operator for time derivatives of functions. For example, an equation stating the relationship between the voltage over and the current through an ideal capacitor is:

$$I(t) = C \frac{d}{dt} V \qquad (1)$$

This equation states that the function I, the current, is proportional to the time derivative of the function V, the voltage. In Mathematica, equation (1) can be written as:

```
I = C*der(V);
```

If the voltage V is known and we want to determine the current I, an ODE must be solved. This kind of modeling with ODEs is used in many engineering applications. However, there are many other areas, such as fluid dynamics, structural mechanics etc., where mathematical models contain functions of several variables, for example a function that gives the flow rate of a fluid in different parts of a pipe or a function that gives the amplitude of the water surface when modeling water waves. Models involving such functions contain partial derivatives, i.e. derivatives with respect to one of several variables, thus partial differential equations (PDEs) must be solved to find searched functions. A typical example of a PDE is the wave equation, which can be used to model for example water waves:

$$\frac{\partial^2}{\partial t^2} U(x, y, t) = \frac{\partial^2}{\partial x^2} U(x, y, t) + \frac{\partial^2}{\partial y^2} U(x, y, t) \qquad (2)$$

Currently, the Mathematica language has no support for expressing spatial variables and spatially distributed functions. There is also no support for describing domains, which are needed to define the problem domain for PDEs and functions of several variables. In this paper, we will attempt to design new language

elements to solve these issues. We also present an example model that uses the wave equation in (2) and show how it can be modeled using Mathematica with our extensions.

This paper is organized as follows: Section 2 introduces and discusses the new elements. Section 3 presents the results by showing an example that was modeled using these extensions and solved using our prototype implementation. Finally, some conclusions are made in Section 4.

2 -Language description

In order to write PDE-based models, we need to describe geometric shapes and regions that define the problem domain. For this, we need a notion of spatially distributed variables which does not vary over time, but represent dimensions in space. These are called space variables. When defining geometric regions, we need to give some constraints or equations to be applied on the space variables. We also need to define sub-domains in order to apply different conditions on different subsets of a domain. Furthermore, when we use functions of several variables in expressions or equations, we need to specify the domain for the space variables involved. In this section, we present language extensions that deal with these issues.

2.1- Space variables

In Mathematica, variable declarations may contain type attributes, such as parameter, constant ,etc. We introduce a new keyword space to declare space variables. An example of a declaration is:

space Real x,y;

The space variables have by default no limits. Here, the variables x and y belong to the range $(-\infty, \infty)$, and thus represent the entire R2. In order to describe smaller domains, we introduce domain classes with constraints.

2.2- Domains

A domain is a set, in our case a set of tuples of real numbers. For example, a two-dimensional domain is a set of pairs of real numbers. Thus, R2 is a two-dimensional domain. Subsets of a domain can be defined by putting constraints on the space variables. For example, a rectangular domain can be defined by limiting the x and y variables with lower and upper bounds. We introduce the keywords domain and constraint used as following :

```
domain Circular
space Real x,y;
constraint
-1 <= x <= 1 and
-sqrt(1-xˆ2) <= y <= sqrt(1-xˆ2)
end Circular;
```

In the current implementation, the constraints are specified as an and- separated list of intervals or an or-separated list of equality expressions involving the space variables. Generally, the constraints could be any Boolean expression which must be true for the space variables to belong to this domain. The same construction can be used to define sub-domains inside a domain, using equality constraints for the space variables and thereby decreasing the dimension. By defining a circle domain and using it as a component in the Circular domain above we can refer to the boundary of the domain. This is shown in the example model in Section 3, where we also use inheritance to simplify the domain description.

2.3- Domain specification for expressions

When space variables occur in expressions, the domain they belong to must be declared. One method of doing this is to use temporary space variables and specify the domain for these together with the expression. A well-known notation used in mathematical literature looks like this:

$$f(x; y) = \sin(x) + \cos(y) ; \quad (x, y) \in \Omega \qquad (3)$$

To use a syntax similar to (3), we introduce two keywords, where and in , corresponding to comma and ' ∈ '. An example may look like this:

f(x,y) = sin(x) + cos(y) where (x,y) in Ω ;

Here, a previously declared domain omega is used as the domain for the equation. The domain specification applies to the whole expression, here the equation. Thus, the where .. in construct has lower precedence than the equality operator. This construction can also be used to define parametric curves, which simplifies description of geometric shapes to be used as domains.

2.4- Partial Differential Equations models

Using the extensions described so far, general domains can be defined. These domains can then be used when modeling a PDE problem. Besides the domain, boundary conditions and the PDEs themselves need to be written in the model. To specify the boundary conditions, we introduce a new keyword boundary, which defines a new section in the model specification, containing the boundary conditions for the PDE problem.
For example:

```
modelMyPDEModel
space Real x,y;
Circular circdomain;
Real U(x,y,t);
boundary
U(x,y,t)=0 where (x,y) in circdomain.boundary;
endMyPDEModel;
```

Here, the value of the function Uis said to be equal to 0 on the boundary component of the domain circdomain. In some time-dependent PDE problems, the initial values of the unknown function must also be given. These equations can be stated in the boundary section as well, using a constant value 0 instead of the

time variable t as used above. Alternatively, they can be stated in another section called initial, as shown in Section 3.

The PDE itself is straightforward, but we need to define a new operator for partial derivatives. We use the name pder for this operator. An example is as follows:

```
pder(U(x,y,t),t,2)=pder(U(x,y,t),x,2)+ pder(U(x,y,t),y,2)
where (x,y) in circdomain;
```

This is the wave equation defined in (2). Arguments to pder are the function, the variable with respect to which derivations is done, and the order of the derivative.

3- Results

The prototype implementation was done in Mathematica, based on a Mathematicaprogram. The language constructs discussed here were implemented with the Mathematica syntax. For solving the PDE, a finite-difference solver [3] was used, which generates an iterative solver in Mathematica, from which C++ code is generated using MathCode. As a test case, we have modeled and solved a two-dimensional PDE problem, involving the wave equation, with our prototype implementation of the extensions described in the previous section. In this model, we use a hierarchical definition of the domain using inheritance:

```
domain Cartesian2D
space Real x,y;
end
domainCircleBase
extends Cartesian2D
parameter Real r=1;
parameter Real xc=0, yc=0;
endCircleBase;
domain Circle
extendsCircleBase
constraint
x==xc-sqrt(rˆ2-(y-yc)ˆ2) or
```

```
x==xc+sqrt(rˆ2-(y-yc)ˆ2)
end Circle;
domain Circular
extendsCircleBase
Circle boundary(xc=xc, yc=yc, r=r);
constraint
xc-sqrt(rˆ2-(y-yc)ˆ2) <= x <= xc+sqrt(rˆ2-(y-yc)ˆ2) and
yc-r<= y <= yc+r
end Circular;
```

Here, we define the base classes Cartesian2D and CircleBasefor two-dimensional domains in general and for circular domains, respectively, containing parameters for the center and radius of a circle. Then we define the domain Circle with the conditions for the x-variable, and use it as the boundary component in the domain Circular, where also the conditions that define the region inside the boundary are written. The model itself is then defined as follows:

```
modelTestModel
parameter Real r,xc,yc;
parameter Real Uzero(domain=circdom);
Circular circdom(r=r,xc=xc,yc=yc);
Real U(domain=circdom);
initial
U(x,y,t) = Uzero(x,y) where (x,y) in circdom;
pder(U(x,y,t),t,1) = 0 where (x,y) in circdom;
boundary
pder(U(x,y,t),x,1) = 0 where (x,y,) in circdom.boundary;
pder(U(x,y,t),y,1) = 0 where (x,y,) in circdom.boundary;
U(x,y,t) = 0 where (x,y) in circdom.boundary;
equation
pder(U(x,y,t),t,2)=pder(U(x,y,t),x,2)+ pder(U(x,y,t),y,2)
where (x,y) in circdom;
endTestModel;
```

Here, the initial values of the function U is given by the function *Uzer*o, which is provided as a parameter during the instantiation of the model.

4- Conclusion

We have presented new proposal elements to the Mathematicaprogram, which can be used for modeling of PDE-based models. We used a syntax similar to mathematical notation because it is already widely known. In general, the goal of the design decisions was to make it possible to reuse parts of the problem specification, such as the domains and the boundaries. Another goal was to be consistent with the current Mathematica program. The solution of the PDE is done with a numerical solver, using finite-difference methods.

References

[1] Jirstrand, M., " Mathematica a full system simulation tool". In Proc. of Mathematica Workshop .2002.

[2] Sadiq A. Mehdi , "Introduction To Mathematica" , Al-Zaky , Baghdad, 2010 .

[3] Jirstrand, M., J. Gunnarsson, and P. Fritzson ."Mathematica – a new modeling and simulation environment for Mathematica".http://www.mathemtica.org.

[4] Sheshadri, K. and P. Fritzson, "A Mathematica-based PDE-solver generator". In *Scandinavian Simulation Society (SIMS) Conferenc*e.Link¨ oping University, Link¨ oping, Sweden.2000.

اقتراح عناصر جديدة لبرنامج الـ Mathematica لحل المعادلات التفاضلية الجزئية

أ.م. صادق عبد العزيز مهدي
الجامعة المستنصرية
كلية التربية/قسم علوم الحاسبات
Sadiqmehdi71@yahoo.com

المهندس . ميثاق مهدي علي
وزارة الشباب والرياضة
مديرية الرعاية العلمية
Methaqmahdi@yahoo.com

المستخلص:

في هذا البحث قمنا باقتراح عناصر جديدة لبرنامج الـ Mathematica من اجل دعم حل النموذج الذي يتضمن معادلات تفاضلية جزئية والذي بامكاننا تحديد القيم الابتدائية والحدودية للمسألة ، ونفذنا ايضا النموذج المتولد في PDE–solver من خلال مواصفات لغة الـ Mathematica باستخدام العناصر الجديدة المقترحة للامتدادات.