ISSN: 0974-5823 Vol. 7 No. 4 April, 2022

International Journal of Mechanical Engineering

SOLVING ORDINARY DIFFERENTIAL EQUATION IN JAVA PROGRAM

M. Anandhi

Assistant Professor, PG and Research Department of Mathematics, Theivanai Ammal College for Women (Autonomous), Villupuram -605 602,Tamil Nadu, India.

R. Soundarya

PG and Research Department of Mathematics,

Theivanai Ammal College for Women (Autonomous), Villupuram -605 602, Tamil Nadu, India.

Abstract: In this paper, Java is still predominantly applied for statistical analysis and graphical representation, it is rapidly becoming more suitable for mathematical computing. One of the fields where considerable progress has been made recently is the solution of differential equations. Here we gave a brief overview of differential equations that can now be solved by Java.

1.INTRODUCTION

Differential equations describe exchanges of matter, energy, information or any other quantities, often as they vary in time and or space. Their through analytical treatment forms the basis of fundamental theories in mathematics and physics, and they are increasingly applied in chemistry, life science and economics.

Differential equations are solved by integration, but unfortunately, for many practical applications in science engineering, systems of differential equations cannot be integrated to give an analytical solution, but rather need to be solved numerically.

More specifically, the following types of differential equations can now be handled with add-on packagesJava: Initial value problems (IVP) of ordinary differential equations (ODE), using packagedeSolve (Soetaert et al., 2010b).Initial value differential algebraic equations (DAE),packagedeSolve .

- Initial value partial differential equations (PDE), packages (deSolve and ReacTran (Soetaert and Meysman.
- Boundary value problems (BVP) of ordinary differential equations, using package bvp Solve (Soetaertetal.,2010a),or ReacTran and root Solve (Soetaert, 2009).
- Initial value delay differential equations (DDE), using packages deSolve or PBSddesolve Couture-Beiletal.,2010).
- Stochastic differential equations (SDE), using packagesde (Iacus, 2008) and pomp(King et al., 2008**).**

2.PRELIMINARIES

Types of differential equations:

Ordinary differential equations:

Ordinary differential equation describe the change of a state variable y as a function f of one independent variable t (e.g., time or space) of y itself , and ,optionally , a set of other variables p, often called parameters.

$$y' = \frac{dy}{dt} = f(t, y, p)$$

In many cases, solving differential equation requires the introduction of extra conditions. In the following, we concentrate on the numerical treatment of two classes of problems, namely initial value problems and boundary value problems.

Initial value problems: If the extra conditions are specified at the initial value of the independent variable, the differential equations are called initial value problems (IVP).

There exists two main classes of algorithms to numerically solve such problems , so called Runge-Kutta formulas and linear multistep formulas (Hairer et al., 2009; Hairer and Wanner, 2010). The latter contains two important families, the Adams family and the backward differentiation formulae (BDF).

Examples:

Consider the famous van der Pol equation (van der Pol and van der Mark, 1927), that describes a non conservative oscillator with non-linear damping and which was originally developed for electrical circuits employing vacuum tubes. The oscillation is described by means of a 2nd order ODE:

$$z'' - \mu(1 - z^2)z' + z = 0$$

Such a system can be routinely rewritten as a system of two 1^{st} order ODEs, if we substitute z" with y'_1 and z' with y_2:

$$y'_1 = y_2$$

$$y'_2 = \mu(1 - y_1^2).y_2 - y_1$$

There is an parameter , μ, and two differential variables ,y_1 and y_2 with initial values (at t=0).

$$y_{1(t=0)} = 2$$

$$y_{2(t=0)} = 0$$

Boundary value problems: If the extra conditions are specified at different values of the independent variable, the differential equations are called **boundary value problems (**BVP**).** A standard text book on this subject is Ascher et.al.(1995).

Package bvpSolve (Soetaertet.al., 2010a) implements three methods to solve boundary value problems. The simplest solution is method is the single Shooting method, which combines initial value problem integration with a nonlinear root finding algorithm(Press et.al., 2007). Some boundary value problems can be solved with the functions from packages ReacTran and rootSolve.

Example: Suppose a printer has to make and deliver printed copies ranging from 1 to 150.So, to apply boundary value testing, the analysis is done on the boundaries, taking the extreme ends. The maximum value is 150 and the minimum value is 1. The invalid values in this test case will be 0 and 151.

Partial differential equations:

In contrast to ODEs where there is only one independent variable, partial differential equations (PDE) contain partial derivatives with respect to more than one independent variable, for instance t (time) and x (a spatial dimension). To distinguish this type of equationsfrom ODEs, the derivative are represented with the ∂ symbol, e.g.

$$\frac{\partial y}{\partial t} = f(t, x, y, \frac{\partial y}{\partial x}, p)$$

Partial differential equations can be solved by subdividing one or more of the continuous independent variables in a number of gridcells, and replacing the derivatives by discrete ,algebraic approximate equations, so called finite differences (cf. LeVeque, 2007;Haundsdorfer and Verwer, 2003).

Differential algebraic equations:

Differential–algebraic equations (DAE) contain a mixture of differential (f) and algebraic equations (g), the latter e.g. for maintaining mass-balance conditions:

$$y' = f(t, y, p)$$

$$0 = g(t, y, p)$$

Important for the solution of a DAE is its index .The index of a DAE is the number of differentiations needed until a system consisting only of ODEs is obtained.

Algorithm for Ordinary Differential Equation :

```
importcom.mathworks.engine.*;
importjava.util.concurrent.Future;
importjava.util.Arrays;
public class RunSimulation {
public static void main(String[] args) throws Exception {
MatlabEngineeng = MatlabEngine.startMatlab();
Future<Void>fLoad = eng.evalAsync("load_system('vdp')");
while (!fLoad.isDone()){
System.out.println("Loading Simulink model...");
Thread.sleep(10000);
  {
 Future<Void>fSim = eng.evalAsync("simOut = sim('vdp','SaveOutput'," +
"'on','OutputSaveName','yOut'," +
 "'SaveTime','on','TimeSaveName','tOut');");
while (!fSim.isDone()) {
System.out.println("Running Simulation...");
Thread.sleep(10000);
        }
        // Get simulation data
eng.eval("y = simOut.get('yOut');");
eng.eval("t = simOut.get('tOut');");
        // Graph results and create image file
eng.eval("plot(t,y)");
eng.eval("print('vdpPlot','-djpeg')");
        // Return results to Java
double[][] y = eng.getVariable("y");
double[] t = eng.getVariable("t");
        // Display results
System.out.println("Simulation result " + Arrays.deepToString(y))
System.out.println("Time vector " + Arrays.toString(t));
eng.close(); }
}
```

Differential algebraic equations:

They so called "Rober problem" describes an autocatalytic reaction (Robertson, 1966) between three chemical species ,y_1, y_2 and y_3. The problem can be formulated either as an ODE (Mazzia and Magherini, 2008), or as a DAE:

$$y_1' = -0.04y_1 + 10^4 y_2 y_3$$

$$y_2' = 0.04y_1 - 10^4 y_2 y_3 - 310^7 y_2^2$$

$$1= y_1 y_2 y_3$$

Where the first two equations are differential equations that specify the dynamics of chemical species $y_1$and y_2 , while the third algebraic equation ensures that the summed concentration of the three species remains 1.

The DAE has to be specified by the residual function instead of the rate of change (as in ODEs).

$$r_1 = -y'_1 - 0.04y_1 + 10^4 y_2 y_3$$

$$r_2 = -y'_2 + 0.04y_1 - 10^4 y_2 y_3 - 310^7$$

$$r_3 = -1 + y_1 + y_2 + y_3$$

Proposed algorithm for ODE using java :

```
importcom.mathworks.engine.*;
importjava.util.concurrent.Future;
importjava.util.Arrays;
public class RunSimulation {
public static void main(String[] args) throws Exception {
MatlabEngineeng = MatlabEngine.startMatlab();
Future<Void>fLoad = eng.evalAsync("load_system('vdp')");
while (!fLoad.isDone()){
```

```
System.out.println("Loading Simulink model...");
Thread.sleep(10000);
  }
Future<Void>fSim = eng.evalAsync("simOut = sim('vdp','SaveOutput'," +
    "'on','OutputSaveName','yOut'," +
   "'SaveTime','on','TimeSaveName','tOut');")
while (!fSim.isDone()) {
System.out.println("Running Simulation...");
Thread.sleep(10000);
}
 // Get simulation data
eng.eval("y = simOut.get('yOut');");
eng.eval("t = simOut.get('tOut');");
// Graph results and create image file
eng.eval("plot(t,y)");
eng.eval("print('vdpPlot','-djpeg')");
 // Return results to Java
double[][] y = eng.getVariable("y");
double[] t = eng.getVariable("t");
// Display results
System.out.println("Simulation result " + Arrays.deepToString(y));
System.out.println("Time vector " + Arrays.toString(t));
eng.close();
}
}
```

OUTPUT:

```
       Time              1               2                3
[99,]         98 -27.55783         -27.55773      -27.55754
[100,]        99 -2761735          -27.61725      -27.61706
[101,]       100 -27.67542         -27.67532       -27.67513
```

4. CONCLUSION

In this paper, algorithm are proposed based on the java programming for solving ordinary differential equation predominantly applied for statistical analysis and graphical representation, it is more and suitable for mathematical computing . Thanks to the differential equation solvers, Java is also emerging as a powerful environment for dynamic simulations (PetZoldt, 2003; Soetaert and Herman, 2009; Stevens, 2009).

5. REFERENCES

1. U. Ascher, R. Mattheij, and R. Russell. Numerical Solution of Boundary Value Problems for Ordinary Differential Equations. Philadelphia, PA, 1995.
2. U. M. Ascher and L. R. Petzold.Computer Methods for Ordinary Differential Equations and DifferentialAlgebraic Equations. SIAM, Philadelphia, 1998.
3. G. Bader and U. Ascher. A new basis implementation for a mixed order boundary value ODE solver.SIAM J. Scient. Stat. Comput., 8:483–500, 1987
4. F. Mazzia and C. Magherini. Test Set for Initial Value Problem Solvers, release 2.4. Department of Mathematics, UniversityofBari, Italy, 2008.
5. T. Petzoldt. Java as a simulation platform in ecological modelling. Java News, 3(3):8–16, 2003.
6. R. W. Setzer. The odesolve Package: Solvers for Ordinary Differential Equations, 2001. Java and matlab package version 0.1-1.