Program to Solve a First Order Differential Equation Using Picard Iterates

Gautam Krishnan

January 29, 2023 – February 26, 2023

---

1) Introduction

Consider an unspecified first order differential equation with initial data:

$$\frac{dy}{dx} = f(x, y), \qquad y(x_0) = y_0 \quad (\mathbf{1})$$

f is a function of the variables $x\ and\ y$. While one might be able to solve such a first order differential equation explicitly if some of its characteristics are known (for example, if it is a separable or exact equation), the method outlined below – Picard's Iteration – uses only what is given in the initial value problem.

There exists a solution to (1) which is defined as the limit $y_n(x)$ (which is part of a sequence) that comes closer and closer to solving (1) (that is, it converges) as n increases on some interval such that $t \in [t_0, t + \alpha]\ for\ some\ \alpha$. To find the solution, the equation can be written as a function L, whose dependence is on x and $y(x)$:

$$y(x) = L\big(x, y(x)\big) \quad (\mathbf{2})$$

The solution $y(x)$ can be obtained from (2) when inputted into the function L, it returns $y(x)$ itself. Therefore,

$$\int_{x_0}^{x} \frac{dy}{dt}(t)\,dt = \int_{x_0}^{x} f(x, y(t))dt \quad \text{or}$$

$$\boldsymbol{y(x) = y_0 + \int_{x_0}^{x} f(x, y(t))dt} \quad (\mathbf{3})$$

2) Picard Iteration

Equation (3) gives rise to a method by which a solution for (1) can be approximated. Starting with y_0, a better approximation be found in the following form:

$$y_1 = y_0 + \int_{x_0}^{x} f(x, y_0(t))dt$$

Then

$$y_2 = y_0 + \int_{x_0}^{x} f(x, y_1(t))dt$$

And so on till

$$y_n = y_0 + \int_{x_0}^{x} f(x, y_{n-1}(t))dt$$

With each $y_n(x)$ a better approximation is obtained. These approximations are called *Picard Iterates* after Charles Émile Picard.

With the algorithm described above, two programs were written in two programming languages – one in Java and one in Python. These have been described below.

3) A Simple Example

Consider the initial value problem:

$$\frac{dy}{dx} = 2xy \;\; , y(0) = 1$$

This is a separable, homogenous linear first order equation, which can be solved as follows:

$$\frac{1}{y}\,dy = 2x\,dx$$

$$\int \frac{1}{y}\,dy = \int 2x\,dx$$

$$lny = x^2 + c$$

$$lny(0) = \ln(1) = 0 = 0 + c$$

$$lny = x^2$$

$$y(x) = e^{x^2}$$

The Picard Iterates (calculated from the Python program below) is as follows:

```
Iteration number: 1
1

Iteration number: 2
x**2 + 1

Iteration number: 3
x**4/2 + x**2 + 1

Iteration number: 4
x**6/6 + x**4/2 + x**2 + 1

Iteration number: 5
x**8/24 + x**6/6 + x**4/2 + x**2 + 1

>>> |
```

The pattern seen can be written in the form:

$$y(x) = \sum \frac{x^{2n}}{n!}$$

Which is indeed the Taylor series approximation for e^{x^2} as $n \to \infty$.

4) The Program in Java

The code for the program in Java is given here.

Java DiffEq Program.docx

5) Comments and Shortcomings

The program above attempts to execute mathematical operations “manually” – primarily for evaluating a function and integrating a function. Here “manually” means executing these operations using a function (in programming construct, or loosely, a method) that is written by the programmer rather than by using a library of Java. Therefore, the program is heavily reliant on the use of Strings (arithmetic operations while present are insignificant). This causes some undesirable results (like runtime errors) and even causes the program to output incorrect results.

The program is lengthy, yet does not serve its purpose. With new data passed each time the program was run, new bugs were found. Debugging is a time-consuming process. Accounting for all exceptions and cases was difficult. Additionally, if the program did work, there are a number of constraints to the types of equations that can be entered (for example the differential equation can only be linear, and there are formatting requirements for the expression) as was seen in the initial instructions while running the code.

Why not use a library or package? Unlike the SymPy library in Python, Java does not have a library to perform algebra or calculus. This is one of the drawbacks of this programming language, which passes onto the code.

In spite of all the challenges of writing the program, the experience was quite thought provoking. It provided insight to the complexity behind packages and libraries in programming languages, and at a broader level, operating systems. The programmer is aware of the operations in this program because the code is their work, unlike in the case of a library in which only a function is called, but how it is implemented is hidden (through process abstraction).

6) The Program in Python

Click here for the Python program.

Python DiffEq Program.docx

7) Highlights

Unlike the Java program, this program uses the SymPy library for computer algebra and symbolic arithmetic. As a result, the program is only a few lines long. The use of this library also expands the kinds of differential equations it can solve.

Reading the code is simple and rather intuitive. However, one running the program does not know how functions of the SymPy class operate. This is a drawback of this program.

Note: To run this program, the SymPy module must be downloaded in addition to Python.

8) Applications

Estimating the value of *e*

Solving the following differential equation gives the value of *e*:

$$\frac{dy}{dx} = y, \qquad y(0) = 1$$

$$\frac{1}{y}dy = \ dx$$

$$\int \frac{1}{y}dy = \int dx$$

$$lny = x + c$$

$$y(x) = Ce^{x}$$

$$y(0) = 1 = C$$

$$\therefore y(x) = e^{x}$$

e can be estimated by finding $y(1)$. These requirements can be inputted into the Python program with a large number of iterations (say 200). The resulting value will be accurate to multiple significant figures.

This is shown in the screen shot below.

```
Enter right-hand side of the equation dy/dx = f(x,y) y
Enter number of iterations 200
Enter x0 of initial data (y(x0)=y0) 0
Enter y0 of initial data (y(x0)=y0) 1
 Squeezed text (75 lines).
2.71828182845905
>>> |
```

9) Conclusion

This project illustrated Picard's iteration using two programs. Though solving a given first order differential might not be possible in many situations, the Picard Iterates present a solution. Moreover, when the Picard Iterates become large or complicated to solve "manually", the program written here presents an approximation, whose accuracy can be increased by increasing the number of Picard Iterations.

10) Bibliography

Braun, M. (1983). The existence-uniqueness theorem; Picard iteration. In *Differential equations and their applications an introduction to applied mathematics* (pp. 67–80). essay, Springer New York.