9-30-2013

A Numerical Analysis of PSM with Applications to DDEs

Dustin Lehmkuhl
Rose-Hulman Institute of Technology, lehmkudc@rose-hulman.edu

Advisors:
Vincenzo Isaia

MSTR 13-01

A Numerical Analysis of PSM with Applications to DDEs

The Modified Picard method (PSM) for approximating IVPs (in a non-standard numerical fashion) involving ODEs or PDEs has been established as a viable option (see Sochacki and Parker et al.). The form of the approximating method allows itself to be used without much labor on delay differential equations (where the vector field at the current time relies on the state of the system at some earlier time as well as the current time). The properties of the solutions to the DDEs can be different depending on how the delay shows up, hence there are a myriad of subclasses (see Baker, Paul and Wille) and as a consequence, their numerical simulation can be delicate. This jump to DDEs via PSM appears to be possible without worrying about which subclass is involved.

I. INTRODUCTION

The Picard Iteration is a method of approximating simple differential equations easily, but is normally incapable of handling complicated operations without clever integration tricks. PSM is a method of preparing a system of equations for a Modified Picard method that has shown much promise in approximating complicated systems. Delay Differential Equations are a class of Differential Equations that normally involve much more complicated means of determining solutions, but the use of PSM appears show promise for modeling these Delay problems without much effort. For more information on PSM and the Modified Picard Method, see Sochacki and Parker et al. and information on Delay Differential Equations and their subclasses see Baker, Paul and Wille.

II. BRIEF DESCRIPTION OF PICARD ITERATION

The Picard iteration is a method of determining solutions of differential equations through the form

$$y_i = y_0 + \int_{t_0}^{t} f\big(s, y_{i-1}(s)\big)ds; \quad where \qquad \frac{dy}{dt} = f(t,y)$$

In order to carry out the Picard iteration, an initial value y_0 and an expression for y' are required.

Example 2.1

$$\frac{dy}{dt} = y' = ty; y(0) = y_0 = 1$$

$$y_1(t) = y_0 + \int_0^t s * y_0(s)ds = 1 + \int_0^t s * 1ds = 1 + \frac{t^2}{2}$$

$$y_2(t) = y_0 + \int_0^t s * y_1(s)ds = 1 + \int_0^t s * \left(1 + \frac{t^2}{2}\right)ds = 1 + \frac{t^2}{2} + \frac{t^4}{8}$$

$$y_3(t) = y_0 + \int_0^t s * y_2(s)ds = 1 + \int_0^t s * \left(1 + \frac{t^2}{2} + \frac{t^4}{8}\right)ds = 1 + \frac{t^2}{2} + \frac{t^4}{8} + \frac{t^6}{48}$$

Like Taylor expansions, Picard iterations provide a local approximation about a specified time t_0 and therefore incur some error the farther from that t_0 the procedure is carried out. Also like Taylor

expansions, this approximation becomes more precise with more iterations. Once this ith iteration is carried out, a working solution can be used or a large number of IVPs.

Picard Iterations can also be used for systems of Differential Equations.

Example 2.2

$$x' = xy; x(0) = 1$$

$$y' = -ty; y(0) = 2$$

$$x_1(t) = x_0 + \int x_0 * y_0 dt = 1 + \int (1) * (2) dt = 1 + 2t$$

$$y_1(t) = y_0 + \int -t * y_0 dt = 2 + \int -2t dt = 2 - t^2$$

$$x_2(t) = x_0 + \int x_1 * y_1 dt = 1 + \int (1 + 2t) * (2 - t^2) dt = 1 + 2t - 2t^2 - \frac{1}{3}t^3 - \frac{1}{2}t^4$$

$$y_2(t) = y_0 + \int -t * y_1 dt = 2 + \int -t * (2 - t^2) dt = 2 - t^2 + \frac{1}{4}t^4$$

$$x_3(t) = x_0 + \int x_2 * y_2 dt = 1 + 2t + 2t^2 + t^3 - \frac{2}{3}t^4 + \cdots$$

$$y_3(t) = y_0 + \int -t * y_2 dt = 2 - t^2 + \frac{1}{4}t^4 - \frac{1}{24}t^6$$

$$x_4(t) = 1 + 2t + 2t^2 + t^3 + 0t^4 + \cdots$$

$$y_4(t) = 2 - t^2 + \frac{1}{4}t^4 - \frac{1}{24}t^6 + \frac{1}{192}t^8$$

Unfortunately the Picard Iteration becomes difficult once more complicated terms are involved.

Example 2.3:

$$y' = \cos(y) + \sin(t)\,; y(1) = 0$$

$$y_1(t) = 0$$

$$y_2(t) = \int_1^t \big(\cos\big(y_1(s)\big) + \sin(s)\big) ds = t - \cos(t) - (1 - \cos(1))$$

$$y_3(t) = \int_1^t \big(\cos\big(y_2(s)\big) + \sin(s)\big)ds = \int_1^t \big(\cos\big(s - \cos(s) - (1 - \cos(1))\big) + \sin(s)\big)ds$$

As integration is used at every iteration, systems that involve difficult or impossible to close integrals are even more difficult to solve using computational software. However, a Modified Picard Method can be used instead by PSM.

III. BRIEF DESCRIPTION OF PARKER-SOCHACKI METHOD

The Parker-Sochacki method, PSM for Short, is a process of converting the series of differential equations to a polynomial form. Polynomials are remarkably well-behaved in addition, multiplication, and most importantly integration. The way this process work is by adding a new IVP to the system by replacing offending terms with variables.

Example 3.1

$$x' = xy + \cos(t)\,; x(0) = x_0$$

$$y' = -e^t y; y(0) = y_0$$

$$t(0) = 0$$

A few “renaming” of certain objectionable segments makes the iteration much more manageable. The derivative and initial values are determined manually to add these equations to the vector field.

$$u = \cos(t)\,; u' = -\sin(t)\,; \mathrm{u}(0) = \cos(0) = 1$$

$$v = \sin(t)\,; v' = \cos(t)\,; v(0) = \sin(0) = 0$$

$$w = e^t; w' = e^t; w(0) = e^0 = 1$$

With the correct terms replaced, the final system looks like this:

$$x' = xy + u; x(0) = x_0$$

$$y' = -wy; y(0) = y_0$$

$$u' = -v; u(0) = 1$$

$$v' = u; v(0) = 0$$

$$w' = w; w(0) = 1$$

Ideally, this method can be used for most systems of ODE’s

IV. APPLYING PSM TO COMPUTATIONAL METHODS

As the Modified Picard method involves numerous iterations and involves simple operations, it naturally lends itself towards computational methods. However, a few simplifications and processes must be made to adapt this method to programming.

As stated earlier, Picard iterations only approximate around a certain point, causing error the further the approximation is taken. To remedy this, a reset r can be applied at some time τ where a whole new Picard iteration can be applied.

$$y_{i,r-1}(t = \tau) = C$$

$$y_{0,r} = C$$

$$y_{i,r}(t) = C + \int_{\tau}^{t} F(s, y_{i-1})ds$$

These reset points can be determined by any myriad of methods, including an even spacing across the time span of calculation. The method used in the following examples was by evaluating an iteration i at a set of times until an error reaches a certain predefined tolerance. This error is approximated as the difference between evaluations at iterations $i-1$ and i. For solutions with alternating terms like solution y in Example 2.1, iterations $i-2$ and i are used otherwise there would be no calculated error for some iterations.

$$if\ |y_{i-1}(t_{tol} + dt) - y_i(t_{tol} + dt)| < tol, \qquad then\ t_{tol} = t_{tol} + dt$$

$$if\ |y_{i-1}(t_{tol} + dt) - y_i(t_{tol} + dt)| > tol, \qquad then\ t_0 = t_{tol}$$

While polynomials are relatively simple to multiply with an iterative computation, the exponentially increasing power and number of terms becomes unwieldy after only a few iterations as shown in example 2.1. For the following examples, all products of polynomials are truncated after the term with the ith power, where i is the iteration number.

For the ith iteration of Picard:

$$p_i(t) = c_0 + c_1 t + \cdots + c_i t^i$$

$$q_i(t) = d_0 + d_1 t + \cdots d_i t^i$$

$$p_i(t) * q_i(t) \approx k_0 + k_1 t + \cdots k_i t^i$$

Where c, d, and k are constants.

For the numerical analysis examples below, periodic functions are used for a few reasons. For one, error scales with the magnitude of a function, so a monotonic function would produce a misleading error the farther along an approximation would be taken. Also, models of periodic functions have a tendency to fail in both amplitude and phase, so a method that can be accurate in both ways is preferred.

Example 4.1

$$x' = -y; x(0) = 1$$

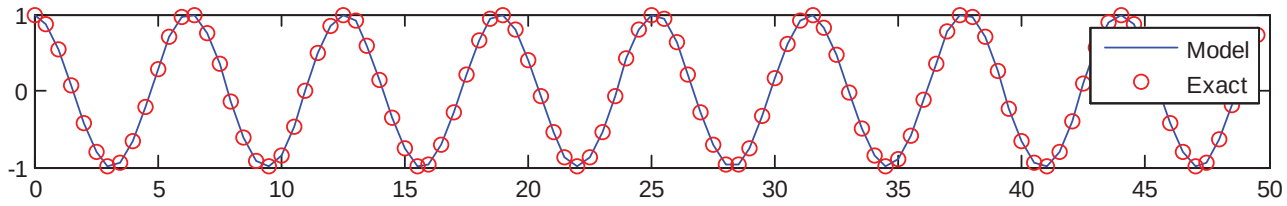
$$y' = x; y(0) = 0$$

This system of differential equations yields:

$$x = \cos(t)\,; y = \sin(t)$$

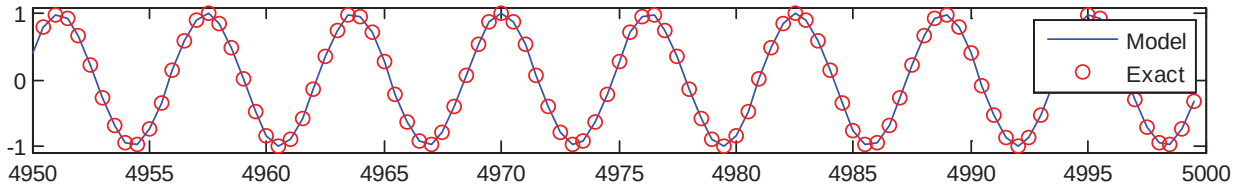
By utilizing an 15 degree approximation and allowing 50 resets with a reset tolerance of 10^{-10} a cosine graph up to t=50 is created. Data 1 is the calculated approximation through PSM and the red circles are the actual solution of y=cos(t).

Figure 4.1



To further illustrate the effectiveness of this process, below is the same solution with the same degree of approximation and tolerance but taken with many more resets to make the solution reach a time of 5000.

Figure 4.2



Time Reached	Resets Required	CPU Time	Integrated Error	Maximum Error
50	50	0.0624 s	3.7007 e-11	2.2861 e-12
5000	5000	6.5676 s	3.7976 e-7	2.3781 e-10

The CPU time reported was from the MATLAB (v.R2011a) program with a student laptop, and when repeated will yield different results. The Integrated error reported is a simple Riemann sum of the errors across all calculated points. The Maximum Error is the highest calculated error at any point. Notice that there was no delay in phase with this process even after a large time span.

Example 4.2

$$x_1' = \frac{1}{2}\frac{x_1}{t+1} - 2tx_2; x_1(0) = 1$$

$$x_2' = \frac{1}{2}\frac{x_2}{t+1} + 2tx_1; x_2(0) = 0$$

After PSM:

$$x_1' = 0.5x_1x_4 - 2x_2x_3; x_1(0) = 1$$

$$x_2' = 0.5x_2x_4 + 2x_1x_3; x_2(0) = 0$$

$$x_3' = 1; x_3(0) = 0$$

$$x_4' = -x_4^2; x_4(0) = 0$$

With the Exact Solution:

$$x_1(t) = \sqrt{t+1}\cos(t^2); x_2(t) = \sqrt{t+1}\sin(t^2)$$

For these plots, the degree of approximation and the tolerance are still the same.

Figure 4.3

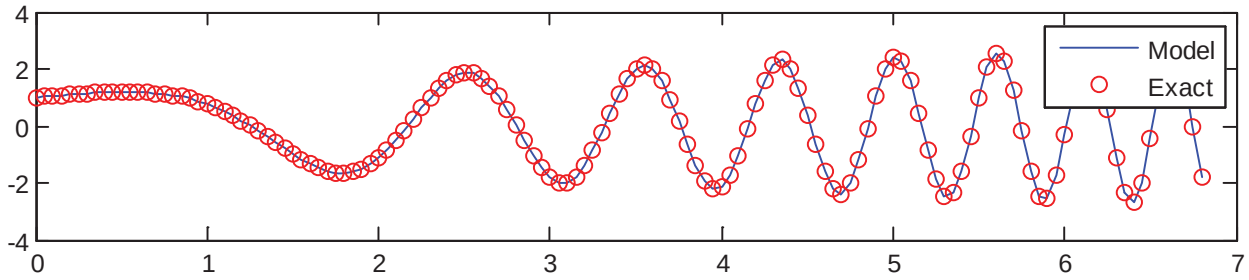
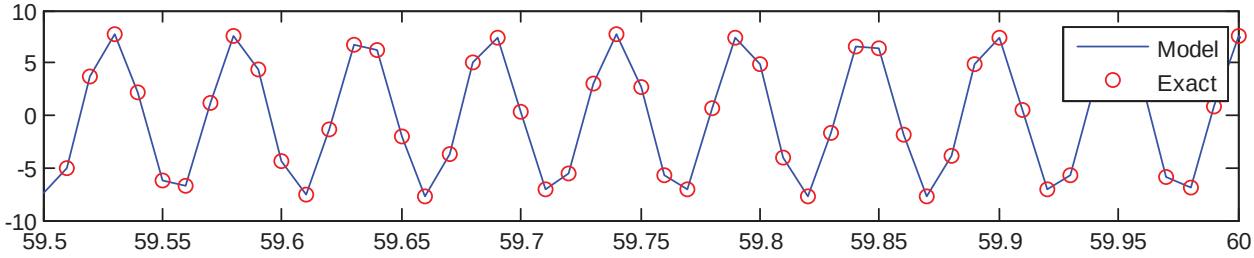


Figure 4.4



Time Reached	Resets Required	CPU Time	Integrated Error	Maximum Error
6.85	50	1.9344 s	1.4356 e-8	5.5689 e-9
60.38	5000	205.4533 s	5.9641 e-7	2.5032 e-8

This System of Differential equation is notably difficult to model due to its changes in amplitude and frequency, causing many modeling methods to fail incredibly quickly.

The error on this method is still significantly higher than that of a normal cosine graph, but as shown in the second plot there doesn't appear to be any appreciable shift in phase or altitude at large values of time. The time taken to perform this calculation was significantly higher than that of example 4.1

V. APPLICATION OF PSM TO DDE's