Discretized Picard's Method

James H. Money James S. Sochacki

March 19, 2007

Abstract

Using the Modified Picard Method of Parker and Sochacki [1, 2], we derive a hybrid scheme using the analytical Picard method with approximations to the differential operators. This new method, called Discretized Picard's Method, uses approximations in the space dimensions to compute derivatives but utilizes a continuous approximation in the time dimension. We illustrate the method using finite difference schemes on linear and nonlinear PDEs. We derive the stability condition for several examples and show the stability region is increasing up to the CFL condition. Finally, we demonstrate results in one and two dimensions for this method.

1 Introduction

One way to find the solution of an ordinary differential equation is to apply Picard's Method. Picard's Method is a method that has been widely studied since its' introduction by Emile Picard in [3]. The method was designed to prove existence of solutions of ordinary differential equations(ODEs) of the form

$$\begin{cases} y'(t) &= f(t,y) \\ y(t_0) &= y_0 \end{cases}$$

by defining the recurrence relation based on the fact

$$y(t) = y_0 + \int_{t_0}^{t} f(s, y(s))\, ds.$$

The only assumptions that are made are f and $\frac{\partial f}{\partial y}$ are continuous in some rectangle surrounding the point (t_0, y_0). In particular, the recurrence relation is given by

$$\begin{cases} \phi^{(0)}(t) &= y_0 \\ \phi^{(n)}(t) &= y_0 + \int_{t_0}^{t} f(s, \phi^{(n-1)}(s))\, ds, \quad n = 1, 2, \ldots \end{cases}. \tag{1}$$

While the recurrence relation results in a straight-forward algorithm to implement on the computer, the iterates become hard to compute after a few steps.

For example, consider the ODE

$$\begin{cases} y'(t) & = 1/y(t) \\ y(1) & = 1 \end{cases},$$

which has the solution $y(t) = \sqrt{2t-1}$. However, the Picard iterates are

$$\begin{array}{ll} \phi^{(0)}(t) & = 1 \\ \phi^{(1)}(t) & = 1 + \int_1^t 1\, ds = 1 + (t-1) = t \\ \phi^{(2)}(t) & = 1 + \int_1^t 1/s\, ds = 1 + \ln t \\ \phi^{(3)}(t) & = 1 + \int_1^t 1/(1+\ln s)\, ds \end{array},$$

and we note the last integral is difficult to calculate. Continuing beyond the fourth iterate only results in increasing problems with calculating the integral. As a result, Picard's Method is generally not used in this form.

Parker and Sochacki, in [1], considered the same problem, but restricted the problem to an autonomous ODE with $t_0 = 0$ and f restricted to polynomial form. In this setting, the iterates result in integration consisting of polynomials. They also showed that the n-th Picard iterate is the MacLaurin polynomial of degree n for $y(t)$ if $\phi^{(n)}(t)$ is truncated to degree n at each step. This form of Picard's method is called the Modified Picard Method(MPM).

In [1], Parker and Sochacki showed that a large class of ODEs could be converted to polynomial form using substitutions and using a system of equations. While this class of ODEs is dense in the analytic functions, it does not include all analytic functions. They also showed one can approximate the solution by a polynomial system and the resulting error bound when using these approximations. Parker and Sochacki also showed that if $t_0 \neq 0$, one computes the iterates as if $t_0 = 0$ and then the approximated solution to the ODE is $\phi^{(n)}(t + t_0)$.

In [2], Parker and Sochacki showed that the ODE based method can be applied to partial differential equations(PDEs) when the PDE is converted to an initial value problem form for PDEs. The resulting solution from MPM is the truncated power series solution from the Cauchy-Kovelsky theorem[4].

Both the ODE and PDE versions of MPM are now used to solve a number of problems including some stiff ODEs. Rudmin[5] describes how to use the MPM to solve the N-Body problem for the solar system accurately. Pruett, et. al. [6], analyzed how to adaptively choose the timestep size and the proper number of iterates for a smaller N-Body simulation and when a singularity was present.

Carothers, et. al., in [7], have proved some remarkable properties of these polynomial systems. They constructed a method by which an ODE could be analytic but could not be converted to polynomial form. They provide a method to convert any polynomial system to a quadratic polynomial system and show how to decouple any system of ODEs into a single ODE. Extending the work of Rudmin, they derive an algebraic method to compute the coefficients of the MacLaurin expansion using Cauchy products.

Warne, et. al. [8], computed an error bound when using the MPM that does not involve using the n-th derivative of the function. This explicit *a-priori*

bound was then used to adaptively choose the timestep size for several problems. They showed a way to generate the Pade approximation using the MacLaurin expansion from MPM.

The MPM has been extended to use parallel computations and adaptively choose the timesteps as the algorithm executes. In [9], the method is modified to include a generic form for ODEs and PDEs and allowed the computation in parallel for any system of equations using a generic text based input file. This method was later modified using the error bound result in [8] to choose adaptive timesteps while performing the parallel computations.

To illustrate the MPM, consider solving the ODE

$$\begin{cases} y'(t) &= y^2(t) \\ y(0) &= 1 \end{cases}, \tag{2}$$

where the solution to (2) is given by $y(t) = 1/(1-t)$. The Modified Picard iterates are

$$\phi^{(0)}(t) = 1$$

$$\phi^{(1)}(t) = 1 + \int_0^t 1^2 \, ds = 1 + t$$

$$\phi^{(2)}(t) = 1 + \int_0^t (1+s)^2 \, ds = 1 + t + t^2 + t^3/3.$$

We truncate to degree two since this is the second iterate and proceed as before, calculating

$$\phi^{(3)}(t) = 1 + \int_0^t (1+s+s^2)^2 \, ds = 1 + t + t^2 + t^3 + \dots$$

$$\phi^{(4)}(t) = 1 + \int_0^t (1+s+s^2+s^3)^2 \, ds = 1 + t + t^2 + t^3 + t^4 + \dots$$

To compare this to the MacLaurin expansion, we get, in fact, that

$$y(t) = 1/(1-t) = 1 + t + t^2 + t^3 + t^4 + \dots,$$

which matches precisely at each degree to the modified Picard iterates.

To highlight the implementation of MPM for PDEs [2], consider the Sine-Gordon equation

$$\begin{cases} u_{tt} = u_{xx} + \sin u \\ u(x,0) = \cos x \\ u_t(x,0) = 0 \end{cases}.$$

The right hand side of this PDE is not in polynomial form. In particular, $\sin u$ is not polynomial. Let $z = u_t$, $v = \cos u$, and $w = \sin u$. Then, the corresponding

system after substituting is

$$\begin{cases} u_t = z & u(x,0) = \cos x \\ z_t = u_{xx} + w & z(x,0) = 0 \\ v_t = -wz & v(x,0) = \cos(\cos x) \\ w_t = vz & w(x,0) = \sin(\cos x) \end{cases}.$$

Since the right hand side is polynomial and equivalent to the Sine-Gordon equation, we call the Sine-Gordon equation **projectively polynomial**. The MPM is applied on the polynomial system.

2 Modified Picard Method for PDEs

In the PDE version of Picard's Method[2], one considers

$$\begin{cases} u_t & = P(u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \ldots, \frac{\partial^2 u}{\partial x^2}, \frac{\partial^2 u}{\partial x \partial y}, \ldots) \\ u(\cdot, 0) & = q(\cdot) \end{cases},$$

where P and q are n variable polynomials. Parker and Sochacki's method is to compute the iterates

$$\begin{cases} \phi^{(0)}(t) & = q(\cdot) \\ \phi^{(n+1)}(\cdot, t) & = q(\cdot) + \int_0^t P(\phi^{(n)}(\cdot, s))\, ds, \quad n = 0, 1, 2, \ldots \end{cases}.$$

We truncate the terms with t-degree higher than n at each step since these terms do not contribute to the coefficient for the t^{n+1} term in the next iteration. We denote the *degree of the Picard iterate* as j for $\phi^{(j)}(t)$, given this truncation that is performed. This method is summarized below in Algorithm 1.

Algorithm 1 Modified Picard Method for PDEs

Require: q, the initial condition, and P the polynomial system
Require: Δt and *numtimesteps*
Require: *degree* the degree of the Picard approximation
  for i from 1 to *numtimesteps* **do**
    $\phi^{(0)}(\cdot, t) = q(\cdot)$
    for j from 1 to degree **do**
      $\phi^{(j)}(\cdots, t) = q(\cdot) + \int_0^t P(\phi^{(j-1)}(\cdot, s))\, ds$
      Truncate $\phi^{(j)}(\cdot, t)$ to degree j in t.
    end for
    $q(\cdot) = \phi^{(degree)}(\cdot, \Delta t)$
  end for

This algorithm is called the Modified Picard Method(MPM). While the MPM algorithm easily computes the approximates since it only depends on calculating derivatives and integrals of the underlying polynomials, it has some

limitations. In [2], the authors showed how to handle the PDE including the initial conditions. However, the method requires the initial conditions in polynomial form. While in some PDEs this is the case, many times one computes a Taylor polynomial that approximates the initial condition to high degree. This results in a substantial increase in computational time. For some problems, the initial condition is not explicitly known, but only a digitized form of the data. For example, in image processing, most of the data has already been digitized and we have to interpolate the data using polynomials in order to apply the MPM. If this is done, the resulting polynomial may not effectively approximate the derivatives of the original function. The polynomial approximation might contain large amounts of oscillations that does not represent the underlying data accurately. Finally, we would also like to be able to handle boundary conditions in a simple manner, but keep the extensibility of the MPM, which does not allow for a boundary condition.

3 Discretized Picard's Method

To overcome the deficiencies listed in section 2, we consider the underlying discrete data directly. We consider the initial condition $u_0 = u_{0_{i_1 i_2 \dots i_m}}$ where $u_0 \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_m}$ is a matrix of m dimensions. Instead of applying the derivatives directly, we consider a set of linear operators L_i where $i = 1, 2, \dots k$ that approximate the derivatives. Then, instead of solving the PDE

$$\begin{cases} u_t & = P(u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \dots, \frac{\partial^2 u}{\partial x^2}, \frac{\partial^2 u}{\partial x \partial y}, \dots) \\ u(\cdot, 0) & = q(\cdot) \end{cases},$$

we replace the various derivatives by L_i and solve

$$\begin{cases} u_t & = P(u, L_1 u, L_2 u, \dots, L_k u) \\ u(\cdot, 0) & = u_{0_{i_1 i_2 \dots i_m}} \end{cases}.$$

We define multiplication of two elements u and v component-wise, instead of using standard matrix multiplication. Then, we compute the iterates

$$\begin{cases} \phi^{(0)}(t) & = u_0 \\ \phi^{(n+1)}(t) & = u_0 + \int_0^t P(\phi^{(n)}(s), L_1\phi^{(n)}(s), L_2\phi^{(n)}(s), \dots, L_k\phi^{(n)}(s))\, ds, \quad n = 0, 1, 2, \dots \end{cases}.$$

The resulting method computes the discretized solution of the PDE, but is continuous in the time variable. In section 4, we illustrate the importance of requiring the operators L_i to be linear in order to get a similar result to the MPM. Given we are utilizing the underlying discrete data in the space variables, we call this new method the **Discretized Picard Method**(DPM). The new method is listed in Algorithm 2.

Algorithm 2 Discretized Picard Method

Require: u_0, the initial condition, and P the polynomial system
Require: $L_1, L_2, \ldots, L_k$, the linear approximations to the derivatives
Require: Δt and *numtimesteps*
Require: *degree* the degree of the Picard approximation
  for i from 1 to *numtimesteps* **do**
    $\phi^{(0)}(\cdot, t) = u_0$
    for j from 1 to degree **do**
      $\phi^{(j)}(t) = u_0 + \int_0^t P(\phi^{(j-1)}(s), L_1(\phi^{(j-1)}(s), \ldots, L_k(\phi^{(j-1)}(s))\, ds$
    end for
    $u_0 = \phi^{(degree)}(\Delta t)$
    Enforce boundary conditions on u_0.
  end for

3.1 Computation of L_i

For the linear operator, there are many discrete operators available for L_i[see [10, 11]]. For example, one could use finite differences, finite elements, or Galerkin methods. In this paper, the operator chosen is the finite difference(FD) operator. For example, if $u_t = u_{xx}$, we can choose the operator L to satisfy the central difference scheme

$$Lu_j = \frac{u_{j+1} - 2u_j + u_{j-1}}{\Delta x}.$$

The operator L is extended easily to the two and three dimension case. In section 5, we show how the choice of the operator determines the stability condition for the maximum timestep size. In addition, the first and last terms in the one dimension case, and all the boundary terms in the two and three dimension cases will have to be handled separately. We discuss this further in section 3.2.

Recall, from the introduction, that a PDE $u_t = f(u, \frac{\partial u}{\partial x}, \ldots)$, is considered projectively polynomial if it can be rewritten as a system of equations in n-variables so that $Y' = P(Y, \frac{\partial Y_1}{\partial x}, \ldots)$ where $Y = [Y_1, \ldots, Y_N]$ and P is polynomial.

For a general class of linear operators based on a linear finite difference(FD) scheme, we deduce that the system remains projectively polynomial, which is summarized by the lemma and theorem below.

Lemma 3.1 *Consider solving via the DPM the PDE*

$$\begin{cases} u_t & = Mu \\ u(\cdot, 0) & = u_0 \end{cases}$$

for some linear differential operator M and initial matrix u_0. Assume that $L \approx M$ is the corresponding linear finite difference operator. Assume L is defined by

$$Lu_{i_1 i_2 \ldots i_m} = \sum_{j_1, j_2, \ldots, j_m} \alpha_{j_1, j_2, \ldots, j_m} u_{i_1 + j_1, i_2 + j_2, \ldots, i_m + j_m}.$$

Then, the PDE is projectively polynomial.

Proof. This follows directly from the definition since Lu is the sum of degree one terms. ∎

Since the linear operator L is projectively polynomial, we see by extension, the general problem is also projectively polynomial.

Theorem 3.1 *Consider solving the PDE*

$$\begin{cases} u_t & = P(u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \dots, \frac{\partial^2 u}{x^2}, \dots) \\ u(\cdot, 0) & = u_0(\cdots) \end{cases}$$

by using the DPM method of

$$\begin{cases} u_t & = P(u, L_1 u, L_2 u, \dots, L_m u) \\ u(\cdot, 0) & = u_{0_{i_1 i_2 \dots i_m}} \end{cases}$$

where each L_i, $i = 1, \dots m$ is linear as in Lemma 3.1. Then, the system is projectively polynomial.

Proof. From Lemma 3.1, we know that each L_i is polynomial and in fact linear. The resulting system is the composition of polynomial terms and has to be projectively polynomial. ∎

As a result, the results of the MPM method with regards to truncating terms can be extended to DPM. Thus, after each iterate is computed, we truncate the terms to degree n, assuming we have computed the n-th iterate.

3.2 Boundary Conditions

The boundary conditions need to be handled carefully in DPM due to the use of higher degree iterates. When the degree of the iterate is one, normal boundary conditions are applied, similar to a FD scheme. However, since the degree one iterate is used to compute the second degree iterate, and similarly for degree three and higher, we must calculate the values at the boundary. The approach we take is to compute one sided derivatives for the FD scheme at the boundaries. Figure 1 illustrates the problem with boundary conditions. When using a degree one iterate, the terms at point x_1 and x_J need to be calculated, where J is the number of discrete data points and the linear operator has a 3 point stencil. If we do not enforce the one sided derivatives at this stage, the data at x_1 and x_J is invalid for the degree two iterate, and then, x_2 and x_{J-1} is invalid after the second iterate is computed. This continues, reducing the available data as the degree of the Picard iterate increases, unless we enforce one sided derivatives at each step.

As a result, we enforce the linear operator to compute one sided derivatives at the edges of the domain. For example, in the one dimension example of

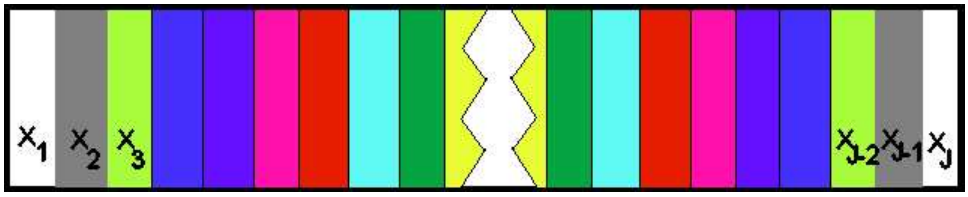

Figure 1: Boundary Conditions The similarly shaded regions are lost if one sided derivatives are not enforced as the degree of the iterates increase.

$u_t = u_{xx}$ with L being the centered difference scheme, we use the end condition in one dimension to be

$$Lu_J = \frac{u_J - 2u_{J-1} + u_{J-2}}{\Delta x^2}$$

and a similar term for Lu_1. Now, we have all the values, and there is no ambiguity in the values at the boundary for any of the degrees of the iterates.

4 Comparison of MPM with DPM and Finite Differences

In this section, we compare the MPM to the DPM. While the MPM computes the power series form for the function u, the DPM does the same computation, but with an approximation to the derivatives at each step. For example, we consider solving the following PDE

$$\begin{cases} u_t & = u_x \\ u(x,0) & = u_0(x) \end{cases}$$

compared to the DPM method of

$$\begin{cases} u_t & = Lu \\ u(x,0) & = u_0(x) \end{cases}, \tag{3}$$

where L is the operator for central difference scheme. If we compute the iterates for MPM we get,

$$\begin{array}{ll} p^{(0)}(t) & = u_0 \\ p^{(1)}(t) & = u_0 + u_{0_x} t \\ p^{(2)}(t) & = u_0 + u_{0_x} t + u_{0_{xx}} t^2/2 \\ p^{(3)}(t) & = u_0 + u_{0_x} t + u_{0_{xx}} t^2/2 + u_{0_{xxx}} t^3/6 \\ & \ldots\ldots \end{array},$$

while the DPM computes

$$\begin{aligned}
\phi^{(0)}(t) &= u_0 \\
\phi^{(1)}(t) &= u_0 + L(u_0)t \\
\phi^{(2)}(t) &= u_0 + L(u_0)t + L^2(u_0)t^2/2 \\
\phi^{(3)}(t) &= u_0 + L(u_0)t + L^2(u_0)t^2/2 + L^3(u_0)t^3/6 \\
&\ldots...
\end{aligned}$$

and we note that L^2 would be a 5 point approximation to u_{xx} and L^3 would be a 7 point approximation to u_{xxx}. By choosing L to be the centered difference scheme, (3) corresponds to the approximated derivatives.

If we consider a nonlinear example, the correspondence between derivatives and the linear operator is still true. If we consider Burger's equation

$$\begin{cases} u_t + (\frac{u^2}{2})_x = 0 \\ u(x,0) = \alpha(x) \end{cases},$$

we can first project to a simpler polynomial system to ease our calculations. Let $w = \frac{u^2}{2}$ to get the equivalent system

$$\begin{cases} u_t + w_x = 0 & u(x,0) = \alpha(x) \\ w_t + uw_x = 0 & w(x,0) = \frac{\alpha^2(x)}{2} = \beta(x) \end{cases}.$$

Consider the following integral form of this system

$$u(x,t) = \alpha(x) - \int_0^t w_x(x,\tau)d\tau$$

$$w(x,t) = \beta(x) - \int_0^t u(x,\tau)w_x(x,\tau)d\tau$$

and the Picard iteration for this system

$$u^{(k+1)}(x,t) = \alpha(x) - \int_0^t w_x^{(k)}(x,\tau)d\tau$$

$$w^{(k+1)}(x,t) = \beta(x) - \int_0^t u^{(k+1)}(x,\tau)w_x^{(k+1)}(x,\tau)d\tau.$$

Now let L be a linear approximation for $\frac{\partial}{\partial x}$. This leads to the following discrete in space approximation

$$u_j^{(k+1)}(t) = \alpha_j - \int_0^t L[w_j^{(k)}(\tau)]d\tau$$

and

$$w_j^{(k+1)}(t) = \beta_j - \int_0^t u_j^{(k+1)}(\tau)L[w_j^{(k+1)}(\tau)]d\tau$$

to this iteration where j indicates $x_j = j\Delta x$. We let

$$u_j^{(0)} = \alpha_j \text{ and } w_j^{(0)} = \beta_j.$$

The Picard iterates for $k = 0$ are

$$u_j^{(1)}(t) = \alpha_j - \int_0^t L[w_j^{(0)}(\tau)]d\tau = \alpha_j - L[w_j^{(0)}]t$$

$$w_j^{(1)}(t) = \beta_j - \int_0^t u_j^{(0)}(\tau)L[w_j^{(0)}(\tau)]d\tau = \beta_j - u_j^{(0)}L[w_j^{(0)}]t.$$

Similarly for $k = 1$, we get

$$\begin{aligned} u_j^{(2)}(t) = \quad & \alpha_j - \textstyle\int_0^t L[w_j^{(1)}(\tau)]d\tau = \alpha_j - \int_0^t L[\beta_j - u_j^{(0)}L[w_j^{(0)}]\tau]d\tau \\ & = \alpha_j - L[w_j^{(0)}]t + L[u_j^{(0)}L[w_j^{(0)}]]\tfrac{t^2}{2} \end{aligned}$$

and

$$\begin{aligned} w_j^{(2)}(t) = \quad & \beta_j - \textstyle\int_0^t u_j^{(1)}(\tau)L[w_j^{(1)}(\tau)]d\tau = \beta_j - \int_0^t(\alpha_j - L[w_j^{(0)}]\tau)L[\beta_j - u_j^{(0)}L[w_j^{(0)}]\tau)]d\tau \\ & = \beta_j - u_j^{(0)}L[w_j^{(0)}]t + (u_j^{(0)}L[u_j^{(0)}L[w_j^{(0)}]] + L[w_j^{(0)}]^2)\tfrac{t^2}{2} \end{aligned}.$$

Then for $k = 2$ we have

$$\begin{aligned} u_j^{(3)}(t) = \quad & \alpha_j - \textstyle\int_0^t L[w_j^{(2)}(\tau)]d\tau \\ = \quad & \alpha_j - \textstyle\int_0^t L[\beta_j - u_j^{(0)}L[w_j^{(0)}]\tau + (u_j^{(0)}L[u_j^{(0)}L[w_j^{(0)}]] + L[w_j^{(0)}]^2)\frac{\tau^2}{2}d\tau \\ = \quad & \alpha_j - L[w_j^{(0)}]t + L[u_j^{(0)}L[w_j^{(0)}]]\tfrac{t^2}{2} - L[u_j^{(0)}L[u_j^{(0)}L[w_j^{(0)}]] + (L[w_0^j])^2)]\tfrac{t^3}{3!} \end{aligned}$$

and

$$\begin{aligned} w_j^{(3)}(t) = \quad & \beta_j - \textstyle\int_0^t u_j^{(2)}(\tau)L[w_j^{(2)}(\tau)]d\tau = \beta_j - \int_0^t(\alpha_j - L[w_j^{(0)}]\tau + L[u_j^{(0)}L[w_j^{(0)}]]\frac{\tau^2}{2})* \\ & \qquad L[\beta_j - u_j^{(0)}L[w_j^{(0)}]\tau + (u_j^{(0)}L[u_j^{(0)}L[w_j^{(0)}]] + L[w_j^{(0)}]^2)\tfrac{\tau^2}{2}]d\tau \\ & = \beta_j - u_j^{(0)}L[w_j^{(0)}]t + (u_j^{(0)}L[u_j^{(0)}L[w_j^{(0)}]] + L[w_j^{(0)}]^2)\tfrac{t^2}{2} \\ & \qquad -(u_j^0L[u_j^{(0)}L[u_j^{(0)}L[w_j^{(0)}]] + L[w_j^{(0)}]^2]+ \\ & \qquad 3L[w_j^0]L[u_j^0L[w_j^0] + L[w_j^0]L[u_j^{(0)}L[w_j^{(0)}]])\tfrac{t^3}{3!} \end{aligned}.$$

And we can continue for higher values of k. However, we can now replace w_j^0 with $(u_j^0)^2/2$ and have

$$\begin{aligned} u_j^{(1)}(t) \quad & = \alpha_j - L[\tfrac{(u_j^0)^2}{2}]t \\ u_j^{(2)}(t) \quad & = \alpha_j - L[\tfrac{(u_j^0)^2}{2}]t + L[u_j^{(0)}L[\tfrac{(u_j^0)^2}{2}]]\tfrac{t^2}{2} \\ u_j^{(3)}(t) \quad & = \alpha_j - L[\tfrac{(u_j^0)^2}{2}]t + L[u_j^{(0)}L[\tfrac{(u_j^0)^2}{2}]]\tfrac{t^2}{2} - L[u_j^{(0)}L[u_j^{(0)}L[\tfrac{(u_j^{(0)})^2}{2}]] + (L[\tfrac{(u_j^{(0)})^2}{2}])^2]\tfrac{t^3}{3!} \end{aligned}.$$

We note that these iterates are the same as the MPM iterates, except with the linear approximation L applied instead of differentiating at each step. The pattern can now be extended as well for other nonlinear problems. This process also works on generating a space discretization with time Picard iteration on any equation of the form

$$\begin{cases} u_t + (f(u))_x = 0 \\ u(x,0) = \alpha \end{cases}$$

where f is polynomial.

The DPM method iterates of degree one and two are related to standard FD schemes. The forward time FD scheme is related to the degree one iterate of DPM. When the degree of DPM is two, we get the DPM method is equivalent to the Lax-Wendroff scheme when the appropriate operator is chosen. The following theorem illustrates the relations between the forward time difference scheme and the Lax-Wendroff scheme.

Theorem 4.1 *Consider applying the Discretized Picard Method to the equation*

$$\begin{cases} u_t & = Mu \\ u(\cdot, 0) & = u_0 \end{cases}$$

for some linear differential operator M and initial matrix u_0. Assume that $L \approx M$ is the corresponding linear finite difference operator. Then, the degree one Picard iterate is the same as the finite difference scheme using the operator L and the degree two Picard iterate is the Lax-Wendroff scheme, if the operator L is chosen to use a stencil with half steps.

Proof. For the degree one iterate, we compute the iterate

$$\phi^{(1)}(t) = u_0 + \int_0^t Lu_0 \, ds$$

Evaluating, we get

$$\phi^{(1)}(t) = u_0 + Lu_0 t$$

and by rearranging we get

$$\phi^{(1)}(t) = u_0 + Lu_0 t$$

$$\frac{\phi^{(1)}(t) - u_0}{t} = Lu_0$$

$$\frac{\phi^{(1)}(t) - \phi^{(0)}(t)}{t} = L[\phi^{(0)}(t)].$$

Letting $u^{n+1} = \phi^{(1)}(t)$ and $u^n = \phi^{(1)}(t)$ we get

$$\frac{u^{n+1} - u^n}{t} = Lu^n$$

Now letting $t = \Delta t$, we get the desired result.

For the second degree iterate, we compute

$$\phi^{(2)}(t) = u_0 + \int_0^t L(\phi^{(1)}(t)(s)) \, ds$$

By expanding and rearranging, we obtain:

$$\begin{aligned}\phi^{(2)}(t) &= u_0 + \int_0^t L(u_0 + Lu_0 s)\, ds \\ &= u_0 + \int_0^t Lu_0 + L^2 u_0 s\, ds \\ &= u_0 + Lu_0 t + L^2 u_0 t^2/2\end{aligned}$$

But, we note that the Lax Wendroff method computes

$$u_0 + u_t t + u_{tt} t^2/2$$

and using that $u_{tt} = L(Lu) = L^2 u$, and choosing the correct operator L with half step points for the stencil, the proof is complete. ∎

5 Stability

In this section, we consider the stability of the DPM as the degree of the Picard iterates increase. In general, we cannot determine a stability condition for any degree m, but the stability region increases with m for all our examples. For the first example, we consider solving the transport equation

$$\begin{cases} u_t & = u_x \\ u(\cdot, 0) & = u_0 \end{cases}$$

using the central difference scheme

$$Lu_j = \frac{u_{j+1} - u_{j-1}}{2\Delta x}$$

with one sided difference at the boundary and in one dimension. The first assertion we make is about the term $L^n u$, since this is needed to compute the Von-Neumann analysis for stability.

Lemma 5.1 *For the linear operator $Lu_j = \frac{u_{j+1} - u_{j-1}}{2\Delta x}$, we have that*

$$L^n u_j = \frac{\sum_{i=0}^n (-1)^i \binom{n}{i} u_{j-2i+n}}{(2\Delta x)^n}$$

Proof. We illustrate a method that is less algebraic and relies on functionology and combinatorics for a proof. For further reference, please see [12, 13]. We define a sequence (U_n) in $\mathbb{R}[[x]]$ by $U_0(x) = \sum_j u_j x^j$ and $U_n(x) = \sum_j L^n(u_j) x^j$. Since L is linear, we have the relation

$$L^n(u_j) = \frac{L^{n-1}(u_{j+1}) - L^{n-1}(u_{j-1})}{2\Delta x}$$

for $n > 0$. Multiplying by x^j and summing over all $j \in \mathbb{Z}^+$ we get that

$$\begin{aligned} U_n(x) &= \sum_j \left[\frac{L^{n-1}(u_{j+1}) - L^{n-1}(u_{j-1})}{2\Delta x}\right] x^j \\ &= \frac{1}{2\Delta x}\left[\frac{U_{n-1}(x)}{x} - xU_{n-1}(x)\right] \\ &= \frac{1}{2\Delta x}\frac{1-x^2}{x}U_{n-1}(x) \end{aligned}$$

Hence, we have $U_n(x) = \left(\frac{1}{2\Delta x}\frac{1-x^2}{x}\right)^n U_0(x)$. Thus, we have

$$\begin{aligned} L^n(u_j) &= [x^j]\left(\frac{1}{2\Delta x}\frac{1-x^2}{x}\right)^n U_0(x) \\ &= \left(\frac{1}{2\Delta x}\right)^n [x^{j+n}](1-x^2)^n U_0(x) \end{aligned}$$

where $[x^j]$ denotes the j-th coefficient of the expansion immediately to the right. If we apply the binomial theorem to the right hand side we see that

$$\begin{aligned} L^n(u_j) &= \left(\frac{1}{2\Delta x}\right)^n \sum_{i=0}^n \binom{n}{i}(-1)^i u_{(j+n)-(2i)} \\ &= \left(\frac{1}{2\Delta x}\right)^n \sum_{i=0}^n \binom{n}{i}(-1)^i u_{j-2i+n} \end{aligned}$$

which completes the proof. ∎

Now, given we have each term explicitly, we can now compute the stability polynomial for any degree of our Picard iterate.

Theorem 5.1 *The Picard iterates of degree m for*

$$\begin{cases} u_t & = u_x \\ u(\cdot, 0) & = u_0 \end{cases}$$

using the central scheme result in the stability polynomial

$$\lambda = 1 + \sum_{n=1}^m \left[\frac{\nu^n}{n!}\sum_{l=1}^n (-1)^l \binom{n}{l} e^{i(n-2l)}\right]$$

where $\nu = \frac{\Delta t}{2\Delta x}$.

Proof. From the Picard iterates, we compute the degree m iterate to be

$$\phi^{(m)}(t) = u_0 + Lu_0 t + L^2 u_0 t^2/2! + \ldots L^m u_0 t^m/m!$$

Let $u^m = \phi^{(m)}(t)$. Then, applying the formula above, we get

$$u_j^m = u_{0_j} + Lu_{0_j} t + \cdots + L^m u_{0_j} t^m/m!$$

If $t = \Delta t$ and $\nu = \frac{\Delta t}{2\Delta x}$, we obtain

$$u_j^{m,1} = u_{0_j} + \nu L u_{0_j} + \nu^2/2! L^2 u_{0_j} + \cdots + \nu^m/m! L^m u_{0_j}$$

or

$$u_j^{m,1} = u_{0_j} + \sum_{n=1}^{m} L^n u_{0_j} \nu^n / n!$$

By applying theorem 5.1, we obtain

$$u_j^{m,1} = u_{0_j} + \sum_{n=1}^{m} \frac{\nu^n}{n!} \left[\sum_{l=0}^{n} (-1)^l \binom{n}{l} u_{j-2l+n} \right]$$

Then, letting $u_j^{m,p} = \lambda^p e^{ijk\Delta x}$ we get

$$\lambda = 1 + \sum_{n=1}^{m} \frac{\nu^n}{n!} \left[\sum_{l=0}^{n} (-1)^l \binom{n}{l} e^{i(n-2l)} \right]$$

and this completes the proof. ∎

Now, let us consider the case of the first four iterates to illustrate the change in the stability condition as the degree increases:

Theorem 5.2 *The stability condition for the first four iterates of*

$$\begin{cases} u_t & = u_x \\ u(\cdot, 0) & = u_0 \end{cases}$$

using the central difference scheme are

Degree	**Stability Condition**
1	*unstable*
2	*unstable*
3	$\nu \leq \frac{\sqrt{3}}{2}$
4	$\nu \leq \sqrt{2}$

for $\nu = \frac{\Delta t}{2\Delta x}$.

Proof. While the result for the $m = 1$ case can be obtained by the usual means for FD scheme, we wish to illustrate an alternate method that makes the computation slightly easier and more straightforward. We consider the stability polynomial

$$\lambda = 1 + \nu \left[e^{ij\Delta x} - e^{-ij\Delta x} \right]$$

for degree one or

$$\lambda = 1 + 2i\nu \sin\theta$$

where $\theta = j\Delta x$. We have

$$|\lambda| = \lambda\overline{\lambda} = 1 + 4\nu^2 \sin^2\theta$$

showing the scheme is unstable. To complete our formal analysis, define

$$f(\nu, \theta) := 1 + 4\nu^2 \sin^2 \theta$$

Then, we fix ν and find the minimum with respect to θ by differentiating:

$$f_\theta = 8\nu^2 \sin\theta \cos\theta = 0$$

Hence, we have $\theta = 0, \pi, \pi/2, -\pi/2$. Filling in those values, we obtain the set of polynomials

$$f(\nu, 0) = f(\nu, \pi) = 1$$
$$f(\nu, \pi/2) = f(\nu, -\pi/2) = 1 + 4\nu^2$$

and we want both these to be less than one for $\nu \geq 0$, i.e.:

$$\begin{cases} 1 & \leq 1 \\ 1 + 4\nu^2 & \leq 1 \end{cases}$$

However, no choice of ν satisfies all these requirements and we conclude that the degree one polynomial is unstable.

Now, we complete a similar analysis on degree two and get the same result. But for degree $m = 3$, we have

$$\lambda = 1 + 2i\nu \sin\theta + \nu^2(\cos 2\theta - 1) + \frac{\nu^3}{3} i \left[\sin(3\theta) - 3\sin\theta\right]$$

We define

$$f(\nu, \theta) := |\lambda|^2$$

and compute $\frac{\partial f}{\partial \theta}(\nu, \theta) = 0$ and get the real solutions are

$$\theta = 0, -\frac{\pi}{2}, \frac{\pi}{2}.$$

Thus, we have the polynomial conditions

$$\begin{cases} f(\nu, 0) = f(\nu, \pi) = 1 \leq 1 \\ f(\nu, -\pi/2) = f(\nu, \pi/2) = 1 - 4/3\nu^4 + 16/9\nu^6 \leq 1 \end{cases}$$

which is satisfied when $\nu \leq \frac{\sqrt{3}}{2}$. The bound for the DPM iterate of degree four is similar to derive and the calculations result in $\nu \leq \sqrt{2}$. ∎

In the case of the degree three and four iterates, the CFL condition is violated. Thus, we need not choose any higher degree iterate than three for the DPM. As a result, we use a degree three iterate with $\nu \leq 1$.

For the heat equation in one dimension, a similar analysis can be completed and is listed below.

Theorem 5.3 *The stability condition for the first four iterates of*

$$\begin{cases} u_t & = u_{xx} \\ u(\cdot,0) & = u_0 \end{cases}$$

using the central difference scheme are

Degree	**Stability Condition**
1	$\nu \leq 0.5$
2	$\nu \leq 0.5$
3	$\nu \leq \frac{\sqrt[3]{4+\sqrt{17}}}{4} - \frac{1}{4\sqrt[3]{4+\sqrt{17}}} + \frac{1}{4} \approx 0.6281863317$
4	$\nu \leq \frac{1}{12}\sqrt[3]{172+36\sqrt{29}} - \frac{5}{3\sqrt[3]{172+36\sqrt{29}}} + \frac{1}{3} \approx 0.6963233909$

for $\nu = \frac{\Delta t}{(\Delta x)^2}$.

A similar analysis will work for the two dimension datasets. We consider the process of applying the heat equation in two dimensions and we get a corresponding analysis for stability from the theorem below.

Theorem 5.4 *The stability condition for the first four iterates for solving*

$$\begin{cases} u_t & = u_{xx} + u_{yy} \\ u(\cdot,0) & = u_0 \end{cases}$$

via DPM using the central difference scheme is

Degree	**Stability Condition**
1	$\nu \leq 0.25$
2	$\nu \leq 0.25$
3	$\nu \leq \frac{1}{2}\left[\frac{\sqrt[3]{4+\sqrt{17}}}{4} - \frac{1}{4\sqrt[3]{4+\sqrt{17}}} + \frac{1}{4}\right] \approx 0.3140931658$
4	$\nu \leq \frac{1}{2}\left[\frac{\sqrt[3]{172+36\sqrt{29}}}{12} - \frac{5}{3\sqrt[3]{172+36\sqrt{29}}} + \frac{1}{3}\right] \approx 0.3481616954$

for $\nu_x = \nu_y = \nu = \frac{\Delta t}{(\Delta x)^2}$.

Proof. We can handle the two dimension case similar to the one dimensional case. Here we need to form $f(\nu_x, \nu_y, \theta, \omega) = \lambda$ and then solve

$$\begin{cases} f_\theta(\nu_x, \nu_y, \theta, \omega) = 0 \\ f_\omega(\nu_x, \nu_y, \theta, \omega) = 0 \end{cases}$$

For the degree two iterate, we get

$$\begin{cases} \theta = 0 & \omega = 0 \\ \theta = 0 & \omega = \pi \\ \theta = \pi & \omega = 0 \\ \theta = \pi & \omega = \pi \end{cases}$$

Then we compute $f(\nu, \nu, \cdot, \cdot)$ for each value of θ and ω and we get

$$\begin{cases} -1 \leq 1 \leq 1 \\ -1 \leq 1 - 4\nu + 8\nu^2 \leq 1 \\ -1 \leq -1 \leq 1 - 4\nu + 8\nu^2 \leq 1 \\ -1 \leq 1 - 8\nu \leq 1 \end{cases}$$

Solving for all cases and combining the answer we get that $\nu \leq 1/4$. We can apply the same analysis and compute the result for degree three and four. ∎

We note here, that we can let $\nu_x \neq \nu_y$ by writing $\nu_y = c\nu_x$ for some constant c and apply the same analysis above and get a similar result when the space grid is not square.

6 Numerical Implementation and Examples

All the examples are implemented in Matlab using a 2Ghz Pentium IV. In order to implement the DPM, an object class for computing the iterates was developed that utilizes matrix coefficients. This object class implements all the basic mathematical operations and includes an integral operator over the time domain. The linear operators are implemented as pluggable modules for the DPM routine which makes the method versatile when considering different types of PDEs and testing different operators used for each derivative. All the floating point arithmetic is computed in double precision.

The first example we consider is

$$\begin{cases} u_t = u_x \\ u(x, 0) = \sin x \end{cases}.$$

We use the centered difference operator for the first derivative, which is $Lu_j = \frac{u_{j+1} - u_{j-1}}{2\Delta x}$. We chose $\Delta x = 0.01$, and ran the method for a total of 200 iterations using a degree three iterate with $\Delta t = \Delta x$, the maximum value allowed by the CFL condition. The result is shown in Figure 2 for times $t = 0, 2, 4$. We note that while the first two iterates are unstable, using the degree three iterate results in a stable method.

The second example is the heat equation in one dimension. We used the centered difference scheme $Lu_j = \frac{u_{j+1} - 2u_j + u_{j-1}}{(\Delta x)^2}$. The degree four iterate is used again for computation and the result is shown in Figure 3. We note the computational cost of computing using the higher degree iterate allows us to compute the final result in less timesteps.

The third example we present is the inviscid form of Burger's equation, which is

$$\begin{cases} u_t & = -uu_x \\ u(0, x) & = f(x) \end{cases}. \tag{4}$$

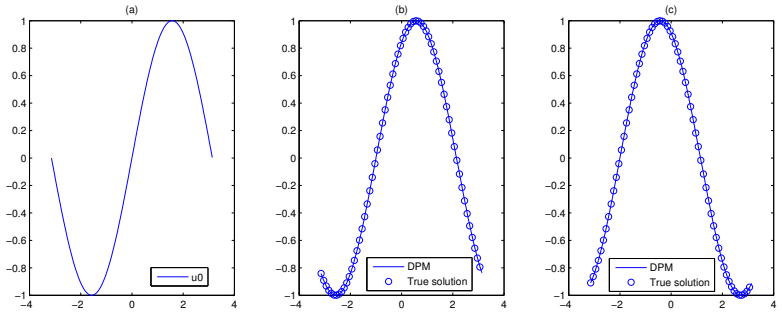

Figure 2: Degree 4 iterate for solving $u_t = u_x$ using a centered difference scheme.

We choose $f(x) = -3/\pi \tan^{-1} x + 1.5$. We see the computed result up to the start of the shock formation in Figure 4(a) using DPM. In (b), the same result is computed using the Lax-Wendroff scheme. However, the stability condition is $\mathcal{O}(\Delta t/(\Delta x)^2)$ for Lax-Wendroff, but the third degree DPM only requires $\Delta t/\Delta x \leq 0.25$. As a result, 21000 iterates must be computed for the Lax-Wendroff versus 420 for the DPM method. The computational savings, even with computing the higher degree iterates, is substantial.

The final example we present is an image smoothing example. Using the fourth degree iterate for solving $u_t = \Delta u$ with the noisy initial image in Figure 5(a), we compute the result in less time. The intermediate and final results are shown in Figure 5(b) and (c). Here, we chose the maximum value for $\nu = \Delta t/(\Delta x)^2$ in Theorem 5.4.

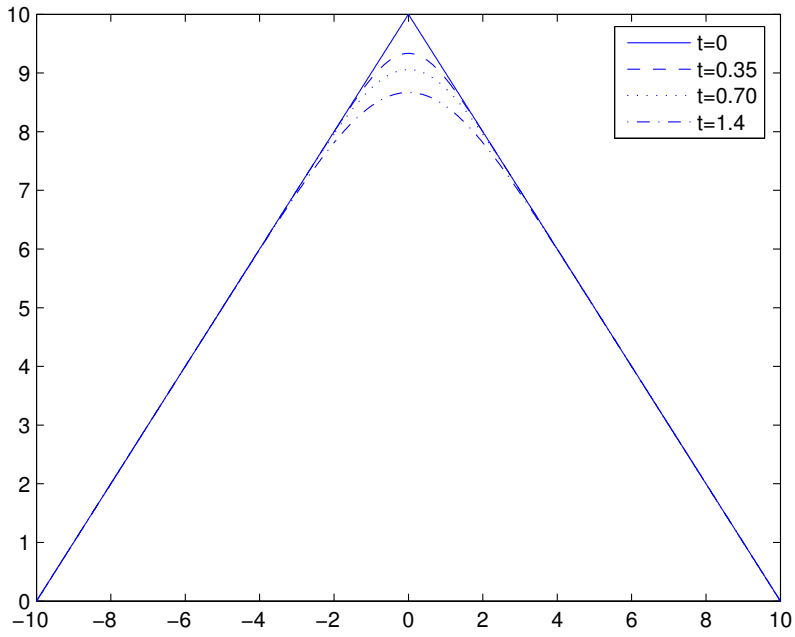

Figure 3: Degree 4 iterate for solving $u_t = u_{xx}$ using a centered difference scheme.

7 Concluding Remarks

We developed the Discretized Picard's Method using the MPM and finite difference schemes. We showed the relation of this new method with existing schemes and computed stability results. We showed for our examples the stability region increases as the degree of the Picard iterate increases in one and two dimensions. The results of this method easily generalizes to any dimension. Future work includes further analysis on stability in the general parabolic form and applications to problems with singularities.

References

[1] J. S. Sochacki, G. E. Parker, Implementing the Picard iteration, Neural, Parallel, and Scientific Computation 4 (1996) 97–112.

[2] J. S. Sochacki, G. E. Parker, A Picard-Maclaurin theorem for initial value PDE's, Abstract and Applied Analysis 5 (1) (2000) 47–63.

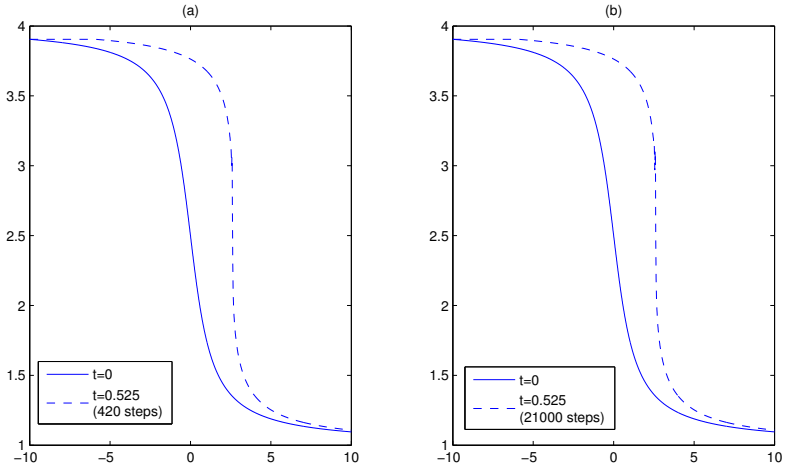

Figure 4: Degree 3 iterate for solving $u_t = -uu_x$ in the present of a shock. (a) is computed via DPM. (b) is the same result using Lax-Wendroff

[3] E. Picard, Traite D'Analyse, Vol. 3, Gauthier-Villars, 1922-1928.

[4] L. Evans, Partial Differential Equations, American Mathematical Society, 1998, pp. 221–233.

[5] J. Rudmin, Application of the Parker-Sochacki method to celestial mechanics, Tech. rep., James Madison University (1998).

[6] C. D. Pruett, J. W. Rudmin, J. M. Lacy, An adaptive N-body algorithm of optimal order, Journal of Computational Physics 187 (2003) 298–317.

[7] D. C. Carothers, G. E. Parker, J. S. Sochacki, P. G. Warne, Some properties of solutions of polynomial systems of differential equations, Electronic Journal of Differential Equations 2005 (40) (2005) 1–17.

[8] P. G. Warne, D. A. P. Warne, J. S. Sochacki, G. E. Parker, D. C. Carothers, Explicit a-priori error bounds and adaptive error control for approximation of nonlinear initial value differential systems, Computers and Mathematics with Applications.

[9] J. Money, A general ODE and PDE solver using Picard's method, MAA Sectional Meetings, 1998.

[10] K. W. Morton, D. F. Mayers, Numerical Solution of Partial Differential Equations, Cambridge University Press, 1994.

[11] A. Quarteroni, R. Sacco, F. Saleri, Numerical Mathematics, Springer, 2000.

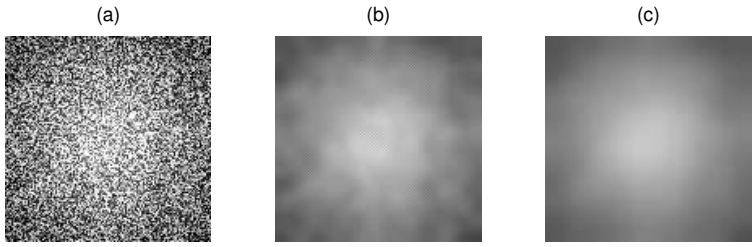

Figure 5: Degree 3 iterate for solving $u_t = \Delta u$ in 2D using a centered difference scheme. Image (a) is the initial noisy image. Image (b) is the result after 5 iterations. Image (c) is the result after 10 iterations.

[12] R. P. Stanley, Enumerative Combinatorics, Vol. 1, Cambridge Press, 1997.

[13] H. S. Wilf, generatingfunctionology, 2nd Edition, Academic Press, 1994.