Picard Iteration Methods

Introduction

The Picard Iteration Method, also known as **Picard's Method of Successive Approximations**, is an iterative technique used to solve ordinary differential equations (ODEs). It is particularly useful for solving initial value problems of the form:

$$y' = f(x, y), \quad y(x_0) = y_0$$

where f is continuous on a given interval. The method iteratively approximates the solution by constructing a sequence of functions that converge to the actual solution of the differential equation.

Formulation of Picard Iteration Method

Given the initial value problem:

$$y' = f(x, y), \quad y(x_0) = y_0,$$

the integral form of the differential equation can be written as:

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

The Picard Iteration Method constructs a sequence of approximations $y_n(x)$ as follows:

- **Initial Approximation**: Start with an initial guess $y_0(x) = y_0$.
- **Iteration Step**: Define the sequence of functions by:

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

The function $y_{n+1}(x)$ is obtained by integrating $f(t, y_n(t))$, the function evaluated using the previous approximation $y_n(x)$.

Convergence of Picard Iteration

The convergence of Picard's method depends on the **Lipschitz condition**. If there exists a constant $L > 0$ such that

$$|f(x, y_1) - f(x, y_2)| \leq L|y_1 - y_2| \quad \text{for all } x, y_1, y_2,$$

then the sequence $\{y_n(x)\}$ converges uniformly to the unique solution $y(x)$ of the differential equation on a given interval.

Theorem (Picard's Convergence Theorem) Let $f(x, y)$ be continuous and satisfy a Lipschitz condition in y on the interval $[x_0, x_0 + h]$. Then the sequence $\{y_n(x)\}$ generated by Picard's method converges to the unique solution $y(x)$ of the initial value problem.

Example

Consider the initial value problem:

$$y' = x + y, \quad y(0) = 1.$$

1. **Rewriting in Integral Form**: The integral form of the problem is

$$y(x) = 1 + \int_0^x (t + y(t))\, dt.$$

2. **Picard Iterations**:

- **First Approximation** $y_0(x) = 1$:

$$y_1(x) = 1 + \int_0^x (t + 1)\, dt = 1 + \left[\frac{t^2}{2} + t\right]_0^x = 1 + \frac{x^2}{2} + x.$$

- **Second Approximation** $y_1(x) = 1 + \frac{x^2}{2} + x$:

$$y_2(x) = 1 + \int_0^x \left(t + 1 + \frac{t^2}{2} + t\right) dt.$$

Calculating this integral yields a more accurate approximation of $y(x)$.

Summary

The Picard Iteration Method is a powerful technique for approximating solutions to differential equations. By constructing a sequence of iteratively refined approximations, the method converges to the unique solution under certain conditions. It is particularly useful for proving existence and uniqueness in theoretical studies of differential equations.

Introduction to Taylor Series Methods

Taylor Series methods are numerical techniques for solving initial value problems (IVPs) of ordinary differential equations (ODEs). Given an ODE:

$$y' = f(x, y), \quad y(x_0) = y_0,$$

the Taylor Series method approximates the solution by expanding $y(x)$ as a Taylor series around x_0.

Taylor Series Expansion

The Taylor series expansion of a function $y(x)$ around a point $x = x_0$ is given by:

$$y(x) = y(x_0) + (x - x_0)y'(x_0) + \frac{(x - x_0)^2}{2!}y''(x_0) + \frac{(x - x_0)^3}{3!}y'''(x_0) + \dots$$

For small increments $h = x_{n+1} - x_n$, this can be written as:

$$y(x_n + h) = y(x_n) + hy'(x_n) + \frac{h^2}{2!}y''(x_n) + \frac{h^3}{3!}y'''(x_n) + \dots$$

In the context of solving ODEs, we approximate $y(x_{n+1})$ by truncating this series after a finite number of terms, depending on the desired accuracy.

Taylor Series Method of Order 1

The first-order Taylor series method is equivalent to the **Euler method**. Using only the first derivative term, we have:

$$y_{n+1} = y_n + hf(x_n, y_n)$$

where h is the step size. This method has a local truncation error of $O(h^2)$.

Taylor Series Method of Order 2

To obtain a more accurate method, we include the second derivative term. For the second-order Taylor series method:

$$y_{n+1} = y_n + hf(x_n, y_n) + \frac{h^2}{2}(f_x(x_n, y_n) + f_y(x_n, y_n)f(x_n, y_n)).$$

The second-order method has a local truncation error of $O(h^3)$.

Higher-Order Taylor Series Methods

For higher accuracy, we can include higher-order derivatives. For instance:

- **Third-Order Taylor Series Method**:

$$\begin{aligned} y_{n+1} =& y_n + hf(x_n, y_n) + \frac{h^2}{2}(f_x(x_n, y_n) + f_y(x_n, y_n)f(x_n, y_n)) \\ &+ \frac{h^3}{6}\left(f_{xx} + 2f_{xy}f + f_{yy}f^2 + f_y(f_x + ff_y)_{(x_n, y_n)}\right) \end{aligned}$$

  This method has a local truncation error of $O(h^4)$.

- **Fourth-Order Taylor Series Method**:

$$\begin{aligned} y_{n+1} =& y_n + hf(x_n, y_n) + \frac{h^2}{2}(f_x(x_n, y_n) + f_y(x_n, y_n)f(x_n, y_n)) \\ &+ \frac{h^3}{6}\left(f_{xx} + 2f_{xy}f + f_{yy}f^2 + f_y(f_x + ff_y)_{(x_n, y_n)}\right) + \frac{h^4}{24}(\dots) \end{aligned}$$

  where the fourth-order terms include combinations of higher partial derivatives. This method has a local truncation error of $O(h^5)$.

Example Calculation

Consider the initial value problem:

$$y' = x + y, \quad y(0) = 1$$

Using a step size $h = 0.1$, compute $y(0.1)$ with the second-order Taylor series method.

1. First, compute $y' = x + y$. 2. Then, find $y'' = \frac{d}{dx}(x + y) = 1 + y' = 1 + (x + y)$. 3. Substitute values:

$$y_{n+1} \approx y_n + hf(x_n, y_n) + \frac{h^2}{2}f'(x_n, y_n)$$

Solution: At $x_0 = 0$ and $y_0 = 1$,

$$y(0.1) \approx y(0) + 0.1 \cdot (0 + 1) + \frac{0.1^2}{2} \cdot (1 + 1) = 1 + 0.1 + 0.01 = 1.11$$

Advantages and Limitations

Taylor Series methods are straightforward and can achieve high accuracy by including more terms. However:

- **Advantage**: Easy to implement for low orders and provides high accuracy with small step sizes.

- **Limitation**: Higher-order derivatives can be difficult to compute, especially for complex functions $f(x, y)$.
- **Limitation**: Not suitable for stiff ODEs, where errors can grow rapidly with small step sizes.

Conclusion

Taylor Series methods offer a systematic way to approximate the solution of ODEs by expanding in a series form. They are particularly useful for problems where higher-order derivatives are easy to compute, though they may become inefficient for complex functions or stiff equations.

Introduction

The Euler methods, both explicit and implicit, are foundational techniques for the numerical solution of ordinary differential equations (ODEs). They are commonly used to approximate solutions of initial value problems of the form:

$$\frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0$$

where $y(t)$ is the unknown function to be determined, and $f(t, y)$ is a given function.

Overview of Euler Methods The Euler methods provide simple one-step approximations for solving ODEs by progressing in discrete time steps. Given a time step $h > 0$, the time points t_n are defined as:

$$t_n = t_0 + n \cdot h, \quad n = 0, 1, 2, \ldots$$

Let $y_n \approx y(t_n)$ be the approximation to the true solution $y(t)$ at $t = t_n$.

Explicit Euler Method

The explicit Euler method is a simple, forward method for approximating $y(t)$ at each time step. It uses the derivative at the current point to estimate the value at the next point.

Formula The explicit Euler method updates the solution as:

$$y_{n+1} = y_n + h \cdot f(t_n, y_n)$$

where: - y_n is the current approximation of $y(t_n)$, - h is the step size, - $f(t_n, y_n)$ is the derivative at the current time step.

Properties 1. **Simplicity**: The explicit Euler method is easy to implement and requires only a function evaluation at each step. 2. **First-Order Accuracy**: The method is first-order accurate in h, meaning that the local

truncation error is $\mathcal{O}(h^2)$, while the global error is $\mathcal{O}(h)$. 3. **Stability**: The explicit Euler method can be unstable for stiff equations, which requires careful selection of the step size h.

Example

Consider the ODE:

$$\frac{dy}{dt} = -2y, \quad y(0) = 1$$

Using the explicit Euler method with $h = 0.1$, we have:

$$y_{n+1} = y_n - 0.2y_n = y_n(1 - 0.2)$$

Starting from $y_0 = 1$:

$$y_1 = 1 \cdot (1 - 0.2) = 0.8,$$
$$y_2 = 0.8 \cdot (1 - 0.2) = 0.64, \quad \text{and so on.}$$

Implicit Euler Method

The implicit Euler method, also known as the backward Euler method, is a stable alternative to the explicit Euler method, especially for stiff equations. Here, we use the derivative at the next point to determine the solution, leading to an implicit equation that must be solved at each step.

Formula

The implicit Euler method is given by:

$$y_{n+1} = y_n + h \cdot f(t_{n+1}, y_{n+1})$$

This equation is implicit in y_{n+1}, meaning that we need to solve for y_{n+1} at each step. For nonlinear f, this generally requires using a root-finding method like Newton's method.

Properties

1. **First-Order Accuracy**: Like the explicit Euler method, the implicit Euler method has first-order accuracy. 2. **Stability**: The implicit Euler method is **A-stable**, making it suitable for stiff equations. This means it can handle larger step sizes h without becoming unstable. 3. **Complexity**: The implicit Euler method requires solving an equation at each step, which may be computationally expensive for nonlinear problems.

Example Using the same ODE:

$$\frac{dy}{dt} = -2y, \quad y(0) = 1$$

we apply the implicit Euler method:

$$y_{n+1} = y_n + h(-2y_{n+1})$$

Rearranging, we get:

$$y_{n+1}(1 + 2h) = y_n \Rightarrow y_{n+1} = \frac{y_n}{1 + 2h}$$

With $h = 0.1$, this becomes:

$$y_{n+1} = \frac{y_n}{1 + 0.2} = \frac{y_n}{1.2}$$

Starting from $y_0 = 1$:

$$y_1 = \frac{1}{1.2} \approx 0.8333,$$
$$y_2 = \frac{0.8333}{1.2} \approx 0.6944, \quad \text{and so on.}$$

Comparison of Explicit and Implicit Euler Methods

- **Stability**: The implicit Euler method is more stable than the explicit method, especially for stiff equations.
- **Computational Cost**: The explicit Euler method is computationally cheaper, as it requires only a function evaluation at each step. The implicit method, however, requires solving an equation at each step.
- **Accuracy**: Both methods have the same order of accuracy, but the implicit method allows for larger time steps in stiff problems.

Conclusion

The Euler methods are fundamental tools in numerical ODE solving. The explicit method is simple and effective for non-stiff problems, while the implicit Euler method provides enhanced stability for stiff problems. Choosing between the two methods often depends on the nature of the differential equation and the computational resources available.

Runge-Kutta Methods

Runge-Kutta methods are a family of iterative techniques used to approximate solutions to ordinary differential equations (ODEs). Given an ODE of the form

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

Runge-Kutta methods approximate $y(x)$ at discrete points by using intermediate function evaluations to improve accuracy.

Second-Order Runge-Kutta Method (RK2)

The second-order Runge-Kutta method advances a solution by a step h as follows:

Method I

1. Calculate the first estimate using the slope at the starting point:
$$k_1 = h \cdot f(x_n, y_n).$$
2. Calculate the midpoint slope:
$$k_2 = h \cdot f\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right).$$
3. Update y using k_2:
$$y_{n+1} = y_n + k_2.$$

Method II

1. Calculate the first estimate using the slope at the starting point:
$$k_1 = h \cdot f(x_n, y_n), \quad k_2 = h \cdot f\left(x_n + h, y_n + k_1\right).$$
2. Update y using k_1 and k_2:
$$y_{n+1} = y_n + \frac{1}{2}(k_1 + k_2).$$

Example for RK2

Consider the ODE:
$$\frac{dy}{dx} = x + y, \quad y(0) = 1.$$
Using RK2 with $h = 0.1$ for a few steps:

$$\begin{aligned} k_1 &= 0.1 \cdot f(0, 1) = 0.1 \cdot (0 + 1) = 0.1, \\ k_2 &= 0.1 \cdot f(0.05, 1.05) = 0.1 \cdot (0.05 + 1.05) = 0.11, \\ y_1 &= 1 + 0.11 = 1.11. \end{aligned}$$

Fourth-Order Runge-Kutta Method (RK4)

The fourth-order Runge-Kutta method is more accurate and widely used. The update formula is given by:

1. Calculate intermediate slopes:
$$\begin{aligned} k_1 &= h \cdot f(x_n, y_n), \\ k_2 &= h \cdot f\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right), \\ k_3 &= h \cdot f\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right), \\ k_4 &= h \cdot f(x_n + h, y_n + k_3). \end{aligned}$$

2. Update y with a weighted average of these slopes:

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4).$$

Example for RK4

Consider the ODE:

$$\frac{dy}{dx} = x + y, \quad y(0) = 1.$$

Using $h = 0.1$:

$$\begin{aligned}
k_1 &= 0.1 \cdot f(0, 1) = 0.1, \\
k_2 &= 0.1 \cdot f(0.05, 1.05) = 0.11, \\
k_3 &= 0.1 \cdot f(0.05, 1.055) = 0.1105, \\
k_4 &= 0.1 \cdot f(0.1, 1.1105) = 0.12105, \\
y_1 &= 1 + \frac{1}{6}(0.1 + 2 \cdot 0.11 + 2 \cdot 0.1105 + 0.12105) \approx 1.1104.
\end{aligned}$$

Milne Predictor-Corrector Method

The Milne Predictor-Corrector Method is a multi-step technique for solving ODEs. It uses information from previous points to predict the value at a new point and then corrects this prediction using an average slope.

Given the ODE:

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

the goal is to approximate the solution $y(x)$ at discrete points over a range of x values.

Predictor Formula (Milne's Predictor)

The predictor formula extrapolates y at x_{n+1} based on previous values:

$$y_{n+1}^{\text{pred}} = y_{n-3} + \frac{4h}{3}\left(2f_{n-2} - f_{n-1} + 2f_n\right).$$

Corrector Formula (Milne's Corrector)

The corrector formula refines the predicted y-value using an implicit approach:

$$y_{n+1} = y_{n-1} + \frac{h}{3}\left(f_{n-1} + 4f_n + f_{n+1}^{\text{pred}}\right),$$

where $f_{n+1}^{\text{pred}} = f(x_{n+1}, y_{n+1}^{\text{pred}})$.

This corrected value is iterated with the predictor formula until the difference between consecutive values is below a certain tolerance.

Example Problem

Consider the ODE:

$$\frac{dy}{dx} = x + y, \quad y(0) = 1.$$

Using step size $h = 0.1$, assume we have the following values:

$$y_0 = 1, \quad y_1 \approx 1.11034, \quad y_2 \approx 1.23252, \quad y_3 \approx 1.36788.$$

Predictor Step

$$y_4^{\text{pred}} = y_0 + \frac{4h}{3}\left(2f_1 - f_2 + 2f_3\right).$$

Substitute $f_i = f(x_i, y_i)$:

$f_1 = f(0.1, 1.11034) = 1.21034, \quad f_2 = f(0.2, 1.23252) = 1.43252, \quad f_3 = f(0.3, 1.36788) = 1.66788.$

Then,

$$y_4^{\text{pred}} = 1 + \frac{4 \cdot 0.1}{3}\left(2 \cdot 1.21034 - 1.43252 + 2 \cdot 1.66788\right) \approx 1.51612.$$

Corrector Step

$$y_4 = y_2 + \frac{h}{3}\left(f_2 + 4f_3 + f_4^{\text{pred}}\right).$$

Compute $f_4^{\text{pred}} = f(0.4, y_4^{\text{pred}}) = f(0.4, 1.51612) = 1.91612.$

$$y_4 = 1.23252 + \frac{0.1}{3}\left(1.43252 + 4 \cdot 1.66788 + 1.91612\right) \approx 1.51756.$$

This process can be repeated iteratively until y_4^{pred} and y_4 converge to within a desired tolerance.

Picard's Iteration for a System of First-Order Equations

Consider a system of first-order ODEs given by:

$$\begin{aligned}
\frac{dy_1}{dx} &= f_1(x, y_1, y_2, \ldots, y_n), \quad y_1(x_0) = y_{1,0}, \\
\frac{dy_2}{dx} &= f_2(x, y_1, y_2, \ldots, y_n), \quad y_2(x_0) = y_{2,0}, \\
&\vdots \\
\frac{dy_n}{dx} &= f_n(x, y_1, y_2, \ldots, y_n), \quad y_n(x_0) = y_{n,0}.
\end{aligned}$$

The Picard iteration approach constructs a sequence of approximations $(y_1^{(k)}, y_2^{(k)}, \ldots, y_n^{(k)})$ that converges to the solution. Starting with an initial guess, each subsequent approximation is defined by integrating the previous iterate.

Iterative Formula

Let $y_i^{(0)}(x) = y_{i,0}$, and for $k \geq 1$, define the next approximation $y_i^{(k)}(x)$ as:

$$y_i^{(k)}(x) = y_{i,0} + \int_{x_0}^{x} f_i(t, y_1^{(k-1)}(t), y_2^{(k-1)}(t), \ldots, y_n^{(k-1)}(t))\, dt.$$

This process is repeated iteratively until convergence.

Example Problem

Consider the system:

$$\frac{dy_1}{dx} = y_2, \quad y_1(0) = 1,$$
$$\frac{dy_2}{dx} = -y_1, \quad y_2(0) = 0.$$

We will apply Picard's iteration method to find the approximate solution for $y_1(x)$ and $y_2(x)$.

Initial Guess

$$y_1^{(0)}(x) = 1, \quad y_2^{(0)}(x) = 0.$$

First Iteration

$$y_1^{(1)}(x) = 1 + \int_0^x y_2^{(0)}(t)\, dt = 1 + \int_0^x 0\, dt = 1.$$
$$y_2^{(1)}(x) = 0 + \int_0^x -y_1^{(0)}(t)\, dt = 0 + \int_0^x -1\, dt = -x.$$

Second Iteration

$$y_1^{(2)}(x) = 1 + \int_0^x y_2^{(1)}(t)\, dt = 1 + \int_0^x -t\, dt = 1 - \frac{x^2}{2}.$$
$$y_2^{(2)}(x) = 0 + \int_0^x -y_1^{(1)}(t)\, dt = 0 + \int_0^x -1\, dt = -x.$$

Repeating these steps will produce increasingly accurate approximations of the solution.

Taylor Series Method for Second-Order Equations

Consider a second-order ordinary differential equation (ODE) of the form:

$$y''(x) = f(x, y, y'), \quad y(x_0) = y_0, \quad y'(x_0) = y_0'.$$

The goal is to solve this equation numerically by approximating the solution using a Taylor series expansion. The Taylor series expansion around x_0 is given by:

$$y(x) = y(x_0) + y'(x_0)(x - x_0) + \frac{y''(x_0)}{2!}(x - x_0)^2 + \frac{y^{(3)}(x_0)}{3!}(x - x_0)^3 + \dots$$

We truncate this expansion to a finite number of terms for practical computation.

Iterative Formula

To solve the second-order ODE numerically, we use the following steps:

1. **Initial Conditions**:

$$y_0 = y(x_0), \quad y_0' = y'(x_0).$$

2. **First Iteration (Taylor Expansion)**:

$$y_1 = y_0 + y_0'(h) + \frac{y_0''(h^2)}{2!}.$$

   Here, h is the step size, and y_0'' is computed by substituting x_0, y_0, y_0' into the given ODE $y''(x) = f(x, y, y')$.

3. **Second Iteration**: Similarly, we use higher-order derivatives $y^{(3)}, y^{(4)}, \dots$ to compute $y_2, y_3, \dots$.

Example Problem

Consider the second-order ODE:

$$y'' = -y, \quad y(0) = 1, \quad y'(0) = 0.$$

This is the simple harmonic oscillator equation. We will use the Taylor Series Method to approximate the solution.

Initial Conditions

We have:

$$y(0) = 1, \quad y'(0) = 0, \quad y''(0) = -y(0) = -1.$$

First Iteration

Using the Taylor series expansion:

$$y_1 = y_0 + y_0'(h) + \frac{y_0''(h^2)}{2!} = 1 + 0(h) + \frac{-1(h^2)}{2} = 1 - \frac{h^2}{2}.$$

Second Iteration

The third derivative $y^{(3)}(x)$ is found by differentiating $y'' = -y$, which gives $y^{(3)} = -y'$. Therefore, $y^{(3)}(0) = 0$. Then:

$$y_2 = y_1 + y_1'(h) + \frac{y_1''(h^2)}{2!} = \left(1 - \frac{h^2}{2}\right) + 0(h) + \frac{-\left(1 - \frac{h^2}{2}\right)}{2}(h^2).$$