# Program for Picard's iterative method | Computational Mathematics

Difficulty Level: Medium • Last Updated: 28 Jun, 2019

The **Picard's** method is an iterative method and is primarily used for approximating solutions to differential equations.

This method of solving a differential equation approximately is one of successive approximation; that is, it is an iterative method in which the numerical results become more and more accurate, the more times it is used.



The Picard's iterative method gives a sequence of approximations Y1(x), Y2(x), ...Yk(x) to the solution of differential equations such that the nth approximation is obtained from one or more previous approximations.

The Picard's iterative series is relatively easy to implement and the solutions obtained

**Start Your Coding Journey Now!** 

Login

#### Picard's iteration method formula:

$$y(t) = y_0 + \int_{t_0}^t f(x, y(x)) dx$$

Picard's iteration formula.

## Steps involved:

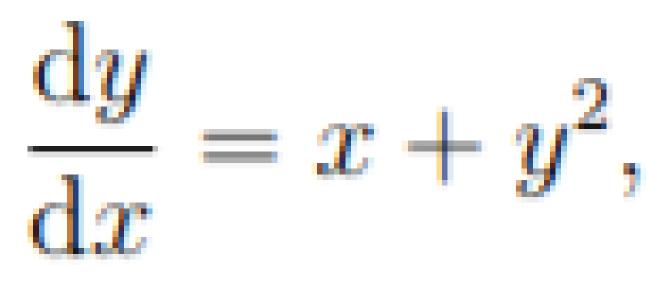
- Step 1: An approximate value of y (taken, at first, to be a constant) is substituted into the right hand side of the differential equation:
   dy/dx= f(x, y).
- Step 2: The equation is then integrated with respect to x giving y in terms of x as a second approximation, into which given numerical values are substituted and the result rounded off to an assigned number of decimal places or significant figures.
- Step 3: The iterative process is continued until two consecutive numerical solutions are the same when rounded off to the required number of decimal places.

## Picard's iteration example:

Given that:

**Start Your Coding Journey Now!** 

Login



and that y = 0 when x = 0, determine the value of y when x = 0.3, correct to four places of decimals.

#### **Solution:**

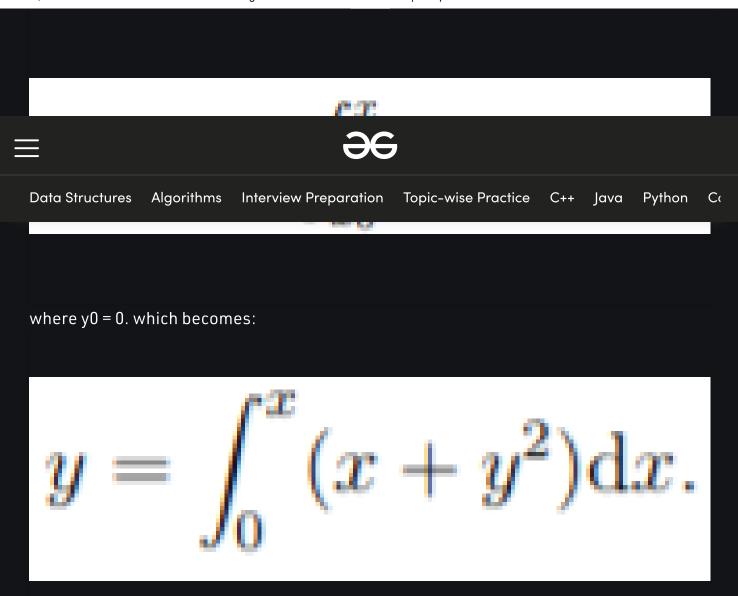
We may proceed as follows:

$$\int_{x_0}^x \frac{\mathrm{d}y}{\mathrm{d}x} \mathrm{d}x = \int_{x_0}^x (x + y^2) \mathrm{d}x,$$

where x0 = 0. Hence:

**Start Your Coding Journey Now!** 

Login



#### First Iteration:

We do not know y in terms of x yet, so we replace y by the constant value y0 in the function to be integrated.

The result of the first iteration is thus given, at x = 0.3, by:

**Start Your Coding Journey Now!** 

Logir

$$y_1 = \int_0^x x dx = \frac{x^2}{2} \simeq 0.0450$$

• Second Iteration:

Now, we use:

$$\frac{\mathrm{d}y}{\mathrm{d}x} = x + y_1^2 = x + \frac{x^4}{4}.$$

Therefore,

$$\int_0^x \frac{\mathrm{d}y}{\mathrm{d}x} \mathrm{d}x = \int_0^x \left( x + \frac{x^4}{4} \right) \mathrm{d}x,$$

**Start Your Coding Journey Now!** 

Logir

## which gives:

$$y - 0 = \frac{x^2}{2} + \frac{x^5}{20}.$$

The result of the second iteration is thus given by:

$$y_2 = \frac{x^2}{2} + \frac{x^5}{20} \simeq 0.0451$$

at x = 0.3.

• Third Iteration:

Now we use:

**Start Your Coding Journey Now!** 

Loain

## Therefore,

$$\int_0^x \frac{\mathrm{d}y}{\mathrm{d}x} \mathrm{d}x = \int_0^x \left( x + \frac{x^4}{4} + \frac{x^7}{20} + \frac{x^{10}}{400} \right) \mathrm{d}x,$$

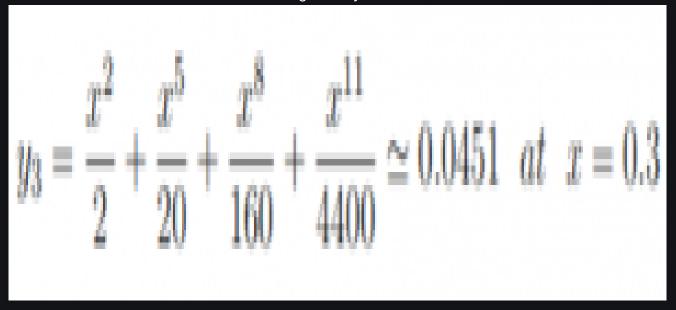
## which gives:

 $x^2$   $x^5$   $x^8$   $x^{11}$ 

**Start Your Coding Journey Now!** 

Logir

## The result of the third iteration is thus given by:



at x = 0.3.

• Hence, y = 0.0451, correct upto four decimal places, at x = 0.3.

## Program for Picard's iterative method:

## **Start Your Coding Journey Now!**

Loair

```
y2[count] = Y2(temp);
    y3[count] = Y3(temp);
}
printf("\nX\n");
for (temp = start_value;
     temp <= end value;</pre>
     temp = temp + allowed_error) {
    printf("%.41f ", temp);
}
printf("\n\nY(1)\n");
for (temp = start_value, count = 0;
     temp <= end_value;</pre>
     temp = temp + allowed_error, count++) {
    printf("%.4lf ", y1[count]);
}
printf("\n\nY(2)\n");
for (temp = start_value, count = 0;
     temp <= end_value;</pre>
     temp = temp + allowed_error, count++) {
    printf("%.41f ", y2[count]);
}
printf("\n\nY(3)\n");
for (temp = start_value, count = 0;
     temp <= end value;</pre>
     temp = temp + allowed_error, count++) {
    printf("%.41f ", y3[count]);
return 0;
```

### **Output:**

Χ

**Start Your Coding Journey Now!** 

Loai

Y(2) 1.0000 1.5045 2.3419 3.7552 6.0645 9.6667 15.0352 22.7205 Y(3) 1.0000 1.5053 2.3692 3.9833 7.1131 13.1333 24.3249 44.2335

> Like 0

> > Next

Page: 1 2 3

## Runge-Kutta 4th Order Method to **Solve Differential Equation**

## RECOMMENDED ARTICLES

- 01 **Program for Gauss Seidel Method** (Computational Mathematics)
- Iterative program to generate 02 distinct Permutations of a String 06, Aug 19
- Print all the permutation of length L using the elements of an array **Iterative**

21, Aug 19

Find all even length binary 06 sequences with same sum of first and second half bits | Iterative 26, Sep 19

**Start Your Coding Journey Now!**