UNIT V

Numerical solution of Ordinary Differential equations

Solution by Taylor's series-Picard's Method of successive Approximations-Euler's Method-Runge-Kutta Methods. Numerical solutions of Laplace equation using finite difference approximation.

1. Taylor's Series method

Consider the one dimensional initial value problem

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

where **f** is a function of two variables **x** and **y** and (x_0 , y_0) is a known point on the solution curve.

If the existence of all higher order partial derivatives is assumed for **y** at $x = x_0$, then by Taylor series the value of **y** at any neighboring point **x+h** can be written as

$$y(x_0+h) = y(x_0) + h\, y'(x_0) + h^2/2\, y''(x_0) + h^3/3!\, y'''(x_0) + \ldots\ldots$$

where ' represents the derivative with respect to **x.** Since at x_0, y_0 is known, **y'** at x_0 can be found by computing $f(x_0,y_0)$. Similarly higher derivatives of **y** at x_0 also can be computed by making use of the relation **y' = f(x,y)**

$y'' = f_x + f_y y'$

$y''' = f_{xx} + 2f_{xy}y' + f_{yy}\, y'^2 + f_y y''$ and so on.

Then

$$y(x_0+h) = y(x_0) + h\, f + h^2 \left(f_x + f_y y' \right) / 2! + h^3 \left(f_{xx} + 2f_{xy}y' + f_{yy}\, y'^2 + f_y y'' \right) / 3! + o(h^4)$$

Hence the value of **y** at any neighboring point $x_0 + h$ can be obtained by summing the above infinite series. However, in any practical computation, the summation has to be terminated after some finite number of terms. If the series has been terminated after the P^{th} derivative term then

the approximated formula is called the Taylor series approximation to **y** of order **P** and the error is of order **P+1**. The same can be repeated to obtain **y** at other points of **x** in the interval $[x_0, x_n]$ in a marching process.

Algorithm

1. Specify x_0, x_n, y_0, h ((x_0, y_0) Initial point, x_n point where the solution is required ,**h** the step length to be used in the marching process)
2. Repeat
   compute $f(x_i, y_i),\ f'(x_i, y_i),\ f''(x_i, y_i)\ \ldots$
   compute $y(x_i+h) = y(x_i) + h\, f(x_i, y_i) + h^2/2\, f'(x_i, y_i) + h^3/3!\, f''(x_i, y_i) + \ldots$
   $x_i = x_i + h$
   until $x_i = x_n$

However, in practice one usually computes two sets of approximations using step sizes **h** and **h/2** and compares the solutions
For $p = 4$, $E_4 = c * h^4$ and the same with step size **h/2**, $E_4 = c * (h/2)^4$, that is if the step size is halved the error is reduced by an order of **1/16**.

Example 1 **Solve the initial value problem $y' = -2xy^2$, $y(0) = 1$ for y at x = 1 with step length 0.2 using Taylor series method of order four.**

Solution:
Given $y' = f(x,y) = -2xy^2$ $y'' = -2y^2 - 4xyy'$

$y''' = -8yy' - 4xy'^2 - 4xyy''$

$y^{iv} = -12y'^2 - 12yy'' - 12xy'y'' - 4xyy'''$

$y^{v} = -48y'y'' - 16yy''' - 12xy''^2 - 16xy'y''' - 4xyy^{iv}$

The forth order Taylor's formula is

$y(x_i+h) = y(x_i) + h\, y'(x_i, y_i) + h^2\ y''(x_i, y_i)/2! + h^3\, y'''(x_i, y_i)/3! + h^4\, y^{iv}(x_i, y_i)/4! + \ldots$

given at x=0, y=1 and with h = .2 we have

$y' = -2(0)(1)^2 = 0.0$
$y'' = -2(1)^2 - 4(0)(1)(0) = -2$
$y''' = -8(1)(0) - 4(0)(0)^2 - 4(0)(1)(-2) = 0.0$
$y^{iv} = -12(0)^2 - 12(1)(-2) - 12(0)(0)(-2) - 4(0)(1)(0) = 24$

$y(0.2) = 1 + .2\,(0) + .2^2\,(-2)/2! + 0 + .2^4\,(24)/4! = .9615$

now at x = .2 we have y = .9615

$y' = -0.3699$, $y'' = -1.5648$, $y''' = 3.9397$ and $y^{iv} = 11.9953$

then $y(0.4) = 1 + .2\,(-.3699) + .2^2\,(-1.5648)/2! + .2^3(3.9397)/3! + .2^4\,(11.9953)/4! = 0.8624$

$y(0.6) = 1 + .2\,(-.5950) + .2^2\,(-0.6665)/2! + .2^3\,(4.4579)/3! + .2^4\,(-5.4051)/4! = 0.7356$

$y(0.8) = 1 + .2\,(-.6494) + .2^2\,(-0.0642)/2! + .2^3\,(2.6963)/3! + .2^4\,(-10.0879)/4! = 0.6100$

$y(1.0) = 1 + .2\,(-.5953) + .2^2\,(-0.4178)/2! + .2^3\,(0.9553)/3! + .2^4\,(-6.7878)/4! = 0.5001$

now at x = 1.0 we have y = .5001
$y' = -0.5001$, $y'' = 0.5004$, $y''' = -.000525$ and $y^{iv} = -3.0005$

Error in the approximation

$E_4 = h^4\,(y^4(1.0) - y^4(0.8))/\,5! = h^4\,(-3.0005-11.9953)/\,5! = -1.9994e-004$

Analytical solution $y(x) = 1/(1+x^2)$ at x = 1, y = .5

Alternate method:

By Taylor series method
$y(x_0 + x - x_0) = y_0 + (x-x_0)y'(x_0) + (x-x_0)^2y''(x_0)/2! + (x-x_0)^3y'''(x_0)/3! + (x-x_0)^4y^{iv}(x_0)/4! + \ldots$
Since $x_0 = y' = y''' = 0$, $y = 1$, $y'' = -2$ and $y^{iv} = 24$ at x = 0,
$y(\,x\,) = 1. - x^2 + x^4 + \ldots$

Example 2 **Using Taylor series method of order four solve the initial value problem y' = (x - y)/2, on [0, 3] with y(0) = 1. Compare solutions for h = 1, 1/2, 1/4 and 1/8.** S

Solution:

Given y' = f(x,y) = **(x - y)/2**

y" = (1 - y')/2, y''' = -y"/2, y^{iv} = -y'''/2

The forth order Taylor's formula is

$y(x_i+h) = y(x_i) + h\, y'(x_i, y_i) + h^2\, y''(x_i, y_i)/2! + h^3\, y'''(x_i, y_i)/3! + h^4\, y^{iv}(x_i, y_i)/4! + \ldots$

$= 1 + h\,(x_i - y_i)/2 - h^2\,(1 - y'_i)/(2*2!) - h^3\, y''_i/(2*3!) - h^4\, y'''_i/(2*4!) + o(h^5)$

$= .5*(\,2 + h\,(x_i - y_i) - .5h^2\,(1 - y'_i) - .1667h^3\, y''_i - .0417h^4\, y'''_i\,) + o(h^5)$

Comparison of the solution for various **h**

x_i	y_i				Exact
	h = 1	h = .5	h = .25	h = .125	y_i
0.000	1.000000	1.000000	1.000000	1.000000	1.000000
0.125				0.943239	0.943240
0.250			0.897492	0.897491	0.897492
0.375				0.862087	0.862087
0.500		0.836426	0.836404	0.836402	0.836402
0.750			0.811870	0.811868	0.811868
1.000	0.820315	0.819629	0.819594	0.819592	0.819592
1.500		0.917142	0.917102	0.917100	0.917100
2.000	1.104513	1.103683	1.103641	1.103639	1.103639
2.500		1.359558	1.359517	1.359515	1.359514
3.000	1.670186	1.669431	1.669393	1.669391	1.669391

2. Euler's Method :

Though in principle it is possible to use Taylor's method of any order for the given initial value problem to get good approximations, it has few draw backs like

- The scheme assumes the existence of all higher order derivatives for the given function **f(x,y)** which is not a requirement for the existence of the solution for any first order initial value problem.
- Even the existence of these higher derivatives is guaranteed it may not be easy to compute them for any given **f(x,y).**
- Because of the usage of higher order derivatives in the formula it is not convenient to write computer programs , that is the method is more suited for hand calculations.

To overcome these difficulties, Euler developed a scheme by approximating **y'** in the given**ivp**. The scheme is as follows:

The derivative term in the first order ivp

$$\mathbf{y' = f(x, y)\ ,\ y(x_0) = y_0}$$

is approximated by making use of Taylor series approximation of the dependent variable **y(x)** at the point $\mathbf{x_{i+1}}$.

Example 1 Find **y(0.5)** if **y** is the solution of IVP **y' = -2x-y, y(0) = -1** using Euler's method with step length **0.1**. Also find the error in the approximation.

Solution: **f(x, y) = -2x - y**,

$y_1 = y_0 + h\ f(x_0, y_0) = -1 + 0.1*(-2*0 - (-1)) = -0.8999$
$y_2 = y_1 + h\ f(x_1, y_1) = -0.8999 + 0.1*(-2*0 - (-0.8999)) = -0.8299$
$y_3 = y_2 + h\ f(x_2, y_2) = -0.8299 + 0.1*(-2*0 - (-0.8299)) = -0.7869$
$y_4 = y_3 + h\ f(x_3, y_3) = -0.7869 + 0.1*(-2*0 - (-0.7869)) = -0.7683$
$y_5 = y_4 + h\ f(x_4, y_4) = -0.7683 + 0.1*(-2*0 - (-0.7683)) = -0.7715$

Truncation error in the approximation = $(h^2/2)\ f''(\square)$, where $0 < \square < 0.5$
$= 0.005\ f''(\square)$

Example 2:
Use Eulers method to solve for **y[0.1]** from **y' = x + y + xy**, **y(0) = 1** with **h = 0.01** also estimate how small **h** would need to obtain four-decimal accuracy.

Solution :
f(x, y) = x + y + xy,

$y_1 = y_0 + h\ f(x_0, y_0) = 1.0 + .01*(0 + 1 + 0*1) = 1.01$
$y_2 = y_1 + h\ f(x_1, y_1) = 1.01 + .01*(0.01 + 1.01 + 0.01*1.01) = 1.02$
$y_3 = y_2 + h\ f(x_2, y_2) = 1.02 + .01*(0.02 + 1.02 + 0.02*1.02) = 1.031$
$y_4 = y_3 + h\ f(x_3, y_3) = 1.031 + .01*(0.03 + 1.031 + 0.03*1.031) = 1.042$
$y_5 = y_4 + h\ f(x_4, y_4) = 1.042 + .01*(0.04 + 1.042 + 0.04*1.042) = 1.053$
$y_6 = y_5 + h\ f(x_5, y_5) = 1.053 + .01*(0.05 + 1.053 + 0.05*1.053) = 1.065$
$y_7 = y_6 + h\ f(x_6, y_6) = 1.065 + .01*(0.06 + 1.065 + 0.06*1.065) = 1.076$

$y_8 = y_8 + h\, f(x_7, y_7) = 1.076 + .01*(0.07 + 1.076 + 0.07*1.076) = 1.089$
$y_9 = y_9 + h\, f(x_8, y_8) = 1.089 + .01*(0.08 + 1.089 + 0.08*1.089) = 1.101$
$y_{10} = y_{10} + h\, f(x_9, y_9) = 1.101 + .01*(0.09 + 1.101 + 0.09*1.101) = 1.114$

error in the approximation = $(h^2/2)\ f''(x)$, where $0 < x < 0.1$
$= 0.00005\ f''(x)$

Backward Method :

Instead of using Taylor's formula at the point $\mathbf{x_0+h}$ one can also use $\mathbf{x_0-h}$ and develop Euler's backward formula in the following manner. Consider the Taylor series at $\mathbf{x = x_0 - h}$

$$\mathbf{y(x_0 - h) = y(x_0) - hy'(x_0) + (h^2/2)\, y''(x_0) + \ldots}$$

If we truncate the infinite series on the right hand side after second term, we can write

$$\mathbf{y(x_0 - h) = y(x_0) - hy'(x_0)}$$
$$\mathbf{-y(x_0) = -y(x_0 - h) - h\, f(x_0, y_0)}$$
or $\mathbf{y_i = y_{i-1} + h\, f_i}$ for any **i**

since $\mathbf{f_i = f(x_i, y_i)}$ has also the unknown $\mathbf{y_i}$, the formula has more than one term which has the unknown and hence the scheme is an implicit one.
Local truncation error is $\mathbf{(h^2/2)\, y''(\square)}$ where $\mathbf{x_i < \square < x_{i+1}}$.

That is, the order of the scheme is same as Euler forward scheme, which is very convenient to use because of its explicit nature whereas the backward scheme is an implicit one in which an approximation to $\mathbf{y_i}$ has to be computed with help of some other scheme. Hence using Euler Forward scheme is time saving process than Euler backward scheme with the same order of accuracy.

Mid-point Method :

Euler's forward and backward schemes

$$y_{i+1} = y_i + h\, f(x_i, y_i)$$
$$y_i = y_{i-1} + h\, f(x_i, y_i)$$

can be clubbed together to get a second order scheme of the form

$$y_{i+1} = y_{i-1} + 2h\, f(x_i, y_i)$$

The only drawback of this scheme is at **i = 0**, the scheme uses the value of the dependent variable at **i = -1** which is not available. Hence one has to use some other scheme to compute the value of the dependent variable at first node (next to the initial point). However from second node onwards the solution can be obtained with second order accuracy using the mid point Nystrom method.

Modified Euler's Method :

The Euler forward scheme may be very easy to implement but it can't give accurate solutions. A very small step size is required for any meaningful result. In this scheme, since, the starting point of each sub-interval is used to find the slope of the solution curve, the solution would be correct only if the function is linear. So an improvement over this is to take the arithmetic average of the slopes at x_i and x_{i+1}(that is, at the end points of each sub-interval). The scheme so obtained is called modified Euler's method. It works first by approximating a value to y_{i+1} and then improving it by making use of average slope.

$$y_{i+1} = y_i + h/2\,(y'_i + y'_{i+1})$$
$$= y_i + h/2(f(x_i, y_i) + f(x_{i+1}, y_{i+1}))$$

If Euler's method is used to find the first approximation of y_{i+1} then

$$y_{i+1} = y_i + 0.5h(f_i + f(x_{i+1}, y_i + hf_i))$$

Truncation error:

$$y_{i+1} = y_i + h\, y'_i + h^2 y_i''/2 + h^3 y_i'''/3! + h^4 y_i^{iv}/4! + \ldots$$
$$f_{i+1} = y'_{i+1} = y'_i + h\, y''_i + h^2 y_{i'}'''/2 + h^3 y_i^{iv}/3! + h^4 y_i^{v}/4! + \ldots$$

By substituting these expansions in the Modified Euler formula gives

$y_i + h\,y'_i + h^2y_i''/2 + h^3y_i'''/3! + h^4y_i^{iv}/4! + \ldots = y_i + h/2\,(y'_i + y'_i + h\,y''_i + h^2y_i'''/2 + h^3y_i^{iv}/3! + h^4y_i^{v}/4! + \ldots)$

So the truncation error is: $-h^3y_i'''/12 - h^4y_i^{iv}/24 + \ldots$ that is, Modified Euler's method is of order two.

Example 1 **Find y(1.0) accurate upto four decimal places using Modified Euler's method by solving the IVP $y' = -2xy^2$, $y(0) = 1$ with step length 0.2.**

Solution: **$f(x, y) = -2xy^2$**
y' = -2*x*y*y, y[0.0] = 1.0 with h = 0.2

Given
y[0.0] = 1.0

Euler Solution: y(1) = y(0) + h*(-2*x*y*y)(1)
y[0.20] = 1.0
Modified Euler iterations:y(1) = y(0) + .5*h*((-2*x*y*y)(0) + (-2*x*y*y)(1)
y[0.20] = 1.0 y[0.20] = 0.9599999988079071 y[0.20] = 0.9631359989929199
y[0.20] = 0.9628947607919341 y[0.20] = 0.9629133460803093

Euler Solution: y(2) = y(1) + h*(-2*x*y*y)(2)
y[0.40] = 0.8887359638083165
Modified Euler iterations:y(2) = y(1) + .5*h*((-2*x*y*y)(1) + (-2*x*y*y)(2)
y[0.40] = 0.8887359638083165 y[0.40] = 0.8626358081578545
y[0.40] = 0.8662926943348495 y[0.40] = 0.8657868947404332
y[0.40] = 0.865856981554814

Euler Solution: y(3) = y(2) + h*(-2*x*y*y)(3)
y[0.60] = 0.7458966289094106
Modified Euler iterations:y(3) = y(2) + .5*h*((-2*x*y*y)(2) + (-2*x*y*y)(3)
y[0.60] = 0.7458966289094106 y[0.60] = 0.7391085349039348
y[0.60] = 0.7403181774980547 y[0.60] = 0.7401034281837107
y[0.60] = 0.7401415785278189

Euler Solution: y(4) = y(3) + h*(-2*x*y*y)(4)
y[0.80] = 0.6086629119889084
Modified Euler iterations:y(4) = y(3) + .5*h*((-2*x*y*y)(3) + (-2*x*y*y)(4)
y[0.80] = 0.6086629119889084 y[0.80] = 0.6151235687114084
y[0.80] = 0.6138585343771569 y[0.80] = 0.6141072871136244
y[0.80] = 0.6140584135348263

Euler Solution: y(5) = y(4) + h*(-2*x*y*y)(5)
y[1.00] = 0.49340256427369866
Modified Euler iterations:y(5) = y(4) + .5*h*((-2*x*y*y)(4) + (-2*x*y*y)(5)
y[1.00] = 0.49340256427369866 y[1.00] = 0.5050460713552334
y[1.00] = 0.5027209825340415 y[1.00] = 0.5031896121302805
y[1.00] = 0.5030953322323046 y[1.00] = 0.503114306721248

Example 2 **Find y in [0,3] by solving the initial value problem y' = (x - y)/2, y(0) = 1. Compare solutions for h = 1/2, 1/4 and 1/8.**

Solution: **f(x, y) = (x - y)/2**

Case(i) : **y' = (x - y)/2, y(0) = 1.0** with **h = 1/2**

Given
y[0.0] = 1.0

Euler Solution: y(1) = y(0) + h*((x-y)/2)(1)
y[0.50] = 0.75
Modified Euler iterations:y(1) = y(0) + .5*h*(((x-y)/2)(0) + ((x-y)/2)(1)
y[0.50] = 0.75 y[0.50] = 0.84375 y[0.50] = 0.83203125 y[0.50] = 0.83349609375 y[0.50] = 0.83331298828125 y[0.50] = 0.8333358764648438

Euler Solution: y(2) = y(1) + h*((x-y)/2)(2)
y[1.00] = 0.7499997615814209
Modified Euler iterations:y(2) = y(1) + .5*h*(((x-y)/2)(1) + ((x-y)/2)(2)
y[1.00] = 0.7499997615814209 y[1.00] = 0.8229164183139801 y[1.00] = 0.8138018362224102 y[1.00] = 0.8149411589838564 y[1.00] = 0.8147987436386757 y[1.00] = 0.8148165455568233

Euler Solution: y(3) = y(2) + h*((x-y)/2)(3)
y[1.50] = 0.8611107402377911
Modified Euler iterations:y(3) = y(2) + .5*h*(((x-y)/2)(2) + ((x-y)/2)(3)
y[1.50] = 0.8611107402377911 y[1.50] = 0.9178236877476991 y[1.50] = 0.9107345693089606 y[1.50] = 0.9116207091138029 y[1.50] = 0.9115099416381975 y[1.50] = 0.9115237875726483

Euler Solution: y(4) = y(3) + h*((x-y)/2)(4)
y[2.00] = 1.0586415426231315

Modified Euler iterations:

y(4) = y(3) + .5*h*(((x-y)/2)(3) + ((x-y)/2)(4)
y[2.00] = 1.0586415426231315 y[2.00] = 1.1027516068990952 y[2.00] = 1.0972378488645997 y[2.00] = 1.0979270686189118 y[2.00] = 1.0978409161496228 y[2.00] = 1.0978516852082838

Euler Solution: y(5) = y(4) + h*((x-y)/2)(5)
y[2.50] = 1.3233877543069634
Modified Euler iterations:y(5) = y(4) + .5*h*(((x-y)/2)(4) + ((x-y)/2)(5)
y[2.50] = 1.3233877543069634 y[2.50] = 1.357695577403087 y[2.50] = 1.3534070995160716 y[2.50] = 1.3539431592519484 y[2.50] = 1.3538761517849638

Euler Solution: y(6) = y(5) + h*((x-y)/2)(6)
y[3.00] = 1.6404133957887526
Modified Euler iterations:y(6) = y(5) + .5*h*(((x-y)/2)(5) + ((x-y)/2)(6)
y[3.00] = 1.6404133957887526 y[3.00] = 1.6670972872799508 y[3.00] = 1.663761800843551 y[3.00] = 1.664178736648101 y[3.00] = 1.6641266196725322

Case(ii) : **y' = (x - y)/2, y(0) = 1.0** with **h = 1/4**

Given
y[0.0] = 1.0

Euler Solution: y(1) = y(0) + h*((x-y)/2)(1)
y[0.250] = 0.875
Modified Euler iterations:y(1) = y(0) + .5*h*(((x-y)/2)(0) + ((x-y)/2)(1)
y[0.250] = 0.875 y[0.250] = 0.8984375 y[0.250] = 0.89697265625 y[0.250] = 0.897064208984375

Euler Solution: y(2) = y(1) + h*((x-y)/2)(2)
y[0.500] = 0.816176176071167
Modified Euler iterations:y(2) = y(1) + .5*h*(((x-y)/2)(1) + ((x-y)/2)(2)
y[0.500] = 0.816176176071167 y[0.500] = 0.8368563205003738 y[0.500] = 0.8355638114735484 y[0.500] = 0.835644593287725

Euler Solution: y(3) = y(2) + h*((x-y)/2)(3)
y[0.750] = 0.7936846013712966
Modified Euler iterations:y(3) = y(2) + .5*h*(((x-y)/2)(2) + ((x-y)/2)(3)
y[0.750] = 0.7936846013712966 y[0.750] =

0.8119317853121117 y[0.750] = 0.8107913363158108 y[0.750] = 0.8108626143780796
Euler Solution: y(4) = y(3) + h*((x-y)/2)(4)

y[1.000] = 0.8032508895617894
Modified Euler iterations:y(4) = y(3) + .5*h*(((x-y)/2)(3) + ((x-y)/2)(4)
y[1.000] = 0.8032508895617894 y[1.000] = 0.8193513439328768 y[1.000] = 0.8183450655346838 y[1.000] = 0.8184079579345709
Euler Solution: y(5) = y(4) + h*((x-y)/2)(5)
y[1.250] = 0.8411035237646307
Modified Euler iterations:y(5) = y(4) + .5*h*(((x-y)/2)(4) + ((x-y)/2)(5)
y[1.250] = 0.8411035237646307 y[1.250] = 0.8553098052268149 y[1.250] = 0.8544219126354284 y[1.250] = 0.8544774059223901

Euler Solution: y(6) = y(5) + h*((x-y)/2)(6)
y[1.500] = 0.9039146953929605
Modified Euler iterations:y(6) = y(5) + .5*h*(((x-y)/2)(5) + ((x-y)/2)(6)
y[1.500] = 0.9039146953929605 y[1.500] = 0.9164496480303976 y[1.500] = 0.9156662134905579 y[1.500] = 0.9157151781492978

Euler Solution: y(7) = y(6) + h*((x-y)/2)(7)
y[1.750] = 0.9887481031258607
Modified Euler iterations:y(7) = y(6) + .5*h*(((x-y)/2)(6) + ((x-y)/2)(7)
y[1.750] = 0.9887481031258607 y[1.750] = 0.9998083540466274 y[1.750] = 0.9991170883640794 y[1.750] = 0.9991602924692387

Euler Solution: y(8) = y(7) + h*((x-y)/2)(8)
y[2.000] = 1.093012893186083
Modified Euler iterations:y(8) = y(7) + .5*h*(((x-y)/2)(7) + ((x-y)/2)(8)
y[2.000] = 1.093012893186083 y[2.000] = 1.1027719368752444 y[2.000] = 1.1021619966446718 y[2.000] = 1.1022001179090826
Euler Solution: y(9) = y(8) + h*((x-y)/2)(9)
y[2.250] = 1.2144230184137998
Modified Euler iterations:y(9) = y(8) + .5*h*(((x-y)/2)(8) + ((x-y)/2)(9)
y[2.250] = 1.2144230184137998 y[2.250] = 1.223033938221066 y[2.250] = 1.2224957557331118 y[2.250] = 1.222529392138609

Euler Solution: y(10) = y(9) + h*((x-y)/2)(10)
y[2.500] = 1.3509613786303571
Modified Euler iterations:y(10) = y(9) + .5*h*(((x-y)/2)(9) + ((x-y)/2)(10)
y[2.500] = 1.3509613786303571 y[2.500] = 1.3585592480824138 y[2.500] = 1.3580843812416603 y[2.500] = 1.3581140604192075
Euler Solution: y(11) = y(10) + h*((x-y)/2)(11)
y[2.750] = 1.5008481797867843
Modified Euler iterations:y(11) = y(10) + .5*h*(((x-y)/2)(10) + ((x-y)/2)(11)
y[2.750] = 1.5008481797867843 y[2.750] = 1.5075521813920236 y[2.750] = 1.5071331812916962 y[2.750] = 1.5071593687979665

Euler Solution: y(12) = y(11) + h*((x-y)/2)(12)
y[3.000] = 1.6625130155689716
Modified Euler iterations:y(12) = y(11) + .5*h*(((x-y)/2)(11) + ((x-y)/2)(12)
y[3.000] = 1.6625130155689716 y[3.000] = 1.6684283103508373 y[3.000] = 1.6680586044269707 y[3.000] = 1.6680817110472124

Case(iii) : **y' = (x - y)/2, y(0) = 1.0** with **h = 1/8**

Given
y[0.0] = 1.0

Euler Solution: y(1) = y(0) + h*((x-y)/2)(1)
y[0.1250] = 0.9375
Modified Euler iterations:y(1) = y(0) + .5*h*(((x-y)/2)(0) + ((x-y)/2)(1)
y[0.1250] = 0.9375 y[0.1250] = 0.943359375 y[0.1250] = 0.94317626953125
Euler Solution: y(2) = y(1) + h*((x-y)/2)(2)
y[0.2500] = 0.8920456171035767
Modified Euler iterations:y(2) = y(1) + .5*h*(((x-y)/2)(1) + ((x-y)/2)(2)
y[0.2500] = 0.8920456171035767 y[0.2500] = 0.8975498788058758 y[0.2500] = 0.8973778706276789

Euler Solution: y(3) = y(2) + h*((x-y)/2)(3)
y[0.3750] = 0.8569217930155446
Modified Euler iterations:y(3) = y(2) + .5*h*(((x-y)/2)(2) + ((x-y)/2)(3)
y[0.3750] = 0.8569217930155446 y[0.3750] = 0.8620924634176603 y[0.3750] = 0.8619308799675942

Euler Solution: y(4) = y(3) + h*((x-y)/2)(4)
y[0.5000] = 0.8315024338597582
Modified Euler iterations:y(4) = y(3) + .5*h*(((x-y)/2)(3) + ((x-y)/2)(4)
y[0.5000] = 0.8315024338597582 y[0.5000] = 0.836359730596966 y[0.5000] = 0.8362079400739283

Euler Solution: y(5) = y(4) + h*((x-y)/2)(5)
y[0.6250] = 0.8151993908072874
Modified Euler iterations:y(5) = y(4) + .5*h*(((x-y)/2)(4) + ((x-y)/2)(5)
y[0.6250] = 0.8151993908072874 y[0.6250] = 0.8197623062048026 y[0.6250] = 0.8196197150986302
Euler Solution: y(6) = y(5) + h*((x-y)/2)(6)
y[0.7500] = 0.8074601603787794
Modified Euler iterations:y(6) = y(5) + .5*h*(((x-y)/2)(5) + ((x-y)/2)(6)
y[0.7500] = 0.8074601603787794 y[0.7500] = 0.8117465357129019 y[0.7500] = 0.8116125864837106

Euler Solution: y(7) = y(6) + h*((x-y)/2)(7)
y[0.8750] = 0.8077657241223026
Modified Euler iterations:y(7) = y(6) + .5*h*(((x-y)/2)(6) + ((x-y)/2)(7)
y[0.8750] = 0.8077657241223026 y[0.8750] = 0.8117923193808908 y[0.8750] = 0.8116664882790599

Euler Solution: y(8) = y(7) + h*((x-y)/2)(8)
y[1.0000] = 0.8156285192196802
Modified Euler iterations:y(8) = y(7) + .5*h*(((x-y)/2)(7) + ((x-y)/2)(8)
y[1.0000] = 0.8156285192196802 y[1.0000] = 0.8194110786347212 y[1.0000] = 0.8192928736530011

Euler Solution: y(9) = y(8) + h*((x-y)/2)(9)
y[1.1250] = 0.8305905320862623
Modified Euler iterations:y(9) = y(8) + .5*h*(((x-y)/2)(8) + ((x-y)/2)(9)
y[1.1250] = 0.8305905320862623 y[1.1250] = 0.8341438456947754 y[1.1250] = 0.8340328046445094

Euler Solution: y(10) = y(9) + h*((x-y)/2)(10)
y[1.2500] = 0.852221507509997
Modified Euler iterations:y(10) = y(9) + .5*h*(((x-y)/2)(9) + ((x-y)/2)(10)
y[1.2500] = 0.852221507509997 y[1.2500] = 0.8555594689839763 y[1.2500] = 0.8554551576879144

Euler Solution: y(11) = y(10) + h*((x-y)/2)(11)
y[1.3750] = 0.8801172663274216
Modified Euler iterations:y(11) = y(10) + .5*h*(((x-y)/2)(10) + ((x-y)/2)(11)
y[1.3750] = 0.8801172663274216 y[1.3750] = 0.883252927298937 y[1.3750] = 0.8831549378935771

Euler Solution: y(12) = y(11) + h*((x-y)/2)(12)
y[1.5000] = 0.9138981250585888
Modified Euler iterations:y(12) = y(11) + .5*h*(((x-y)/2)(11) + ((x-y)/2)(12)
y[1.5000] = 0.9138981250585888 y[1.5000] = 0.9168437461524608 y[1.5000] = 0.9167516954932773

Euler Solution: y(13) = y(12) + h*((x-y)/2)(13)
y[1.6250] = 0.9532074113216032
Modified Euler iterations:y(13) = y(12) + .5*h*(((x-y)/2)(12) + ((x-y)/2)(13)
y[1.6250] = 0.9532074113216032 y[1.6250] = 0.95597451009519 y[1.6250] = 0.9558880382585153

Euler Solution: y(14) = y(13) + h*((x-y)/2)(14)
y[1.7500] = 0.9977100692219482
Modified Euler iterations:y(14) = y(13) + .5*h*(((x-y)/2)(13) + ((x-y)/2)(14)
y[1.7500] = 0.9977100692219482 y[1.7500] = 1.000309465199494 y[1.7500] = 1.0002282340751956
Euler Solution: y(15) = y(14) + h*((x-y)/2)(15)
y[1.8750] = 1.0470913492635905
Modified Euler iterations:y(15) = y(14) + .5*h*(((x-y)/2)(14) + ((x-y)/2)(15)
y[1.8750] = 1.0470913492635905 y[1.8750] = 1.049533206241223 y[1.8750] = 1.049456898210672

Euler Solution: y(16) = y(15) + h*((x-y)/2)(16)
y[2.0000] = 1.1010555776593376
Modified Euler iterations:y(16) = y(15) + .5*h*(((x-y)/2)(15) + ((x-y)/2)(16)
y[2.0000] = 1.1010555776593376 y[2.0000] = 1.1033494434461277 y[2.0000] = 1.1032777601402906

Euler Solution: y(17) = y(16) + h*((x-y)/2)(17)
y[2.1250] = 1.1593250002283733
Modified Euler iterations:y(17) = y(16) + .5*h*(((x-y)/2)(16) + ((x-y)/2)(17)
y[2.1250] = 1.1593250002283733 y[2.1250] = 1.161479843978849 y[2.1250] = 1.1614125051116466

Euler Solution: y(18) = y(17) + h*((x-y)/2)(18)
y[2.2500] = 1.221638696360544
Modified Euler iterations:y(18) = y(17) + .5*h*(((x-y)/2)(17) + ((x-y)/2)(18)
y[2.2500] = 1.221638696360544 y[2.2500] = 1.2236629436446282 y[2.2500] = 1.2235996859170006

Euler Solution: y(19) = y(18) + h*((x-y)/2)(19)
y[2.3750] = 1.2877515588009272
Modified Euler iterations:y(19) = y(18) + .5*h*(((x-y)/2)(18) + ((x-y)/2)(19)
y[2.3750] = 1.2877515588009272 y[2.3750] = 1.289653124548429 y[2.3750] = 1.2895937006188196
Euler Solution: y(20) = y(19) + h*((x-y)/2)(20)
y[2.5000] = 1.357433335265581
Modified Euler iterations:y(20) = y(19) + .5*h*(((x-y)/2)(19) + ((x-y)/2)(20)
y[2.5000] = 1.357433335265581 y[2.5000] = 1.359219654714051 y[2.5000] = 1.3591638322312865

Euler Solution: y(21) = y(20) + h*((x-y)/2)(21)
y[2.6250] = 1.4304677281411309
Modified Euler iterations:y(21) = y(20) + .5*h*(((x-y)/2)(20) + ((x-y)/2)(21)

y[2.6250] = 1.4304677281411309 y[2.6250] = 1.4321457859080915 y[2.6250] = 1.432093346602874

Euler Solution: y(22) = y(21) + h*((x-y)/2)(22)
y[2.7500] = 1.5066515487479644
Modified Euler iterations:y(22) = y(21) + .5*h*(((x-y)/2)(21) + ((x-y)/2)(22)
y[2.7500] = 1.5066515487479644 y[2.7500] = 1.5082279061411892 y[2.7500] = 1.508178644972651

Euler Solution: y(23) = y(22) + h*((x-y)/2)(23)
y[2.8750] = 1.5857939228601574
Modified Euler iterations:y(23) = y(22) + .5*h*(((x-y)/2)(22) + ((x-y)/2)(23)
y[2.8750] = 1.5857939228601574 y[2.8750] = 1.5872747435327825 y[2.8750] = 1.5872284678867632

Euler Solution: y(24) = y(23) + h*((x-y)/2)(24)
y[3.0000] = 1.6677155443756573
Modified Euler iterations:y(24) = y(23) + .5*h*(((x-y)/2)(23) + ((x-y)/2)(24)
y[3.0000] = 1.6677155443756573 y[3.0000] = 1.66910661842644 y[3.0000] = 1.669063147362353

Example 3 Find y(0.1) for $y' = x - y^2$, y(0) = 1 correct upto four decimal places.

Solution: **$f(x, y) = x - y^2$**

Case(i) : **$y' = (x - y)/2$, y(0) = 1.0** with **h = 1/2**

Given
y[0.0] = 1.0

Euler Solution: y(1) = y(0) + h*((x-y)/2)(1)
y[0.50] = 0.75
Modified Euler iterations:y(1) = y(0) + .5*h*(((x-y)/2)(0) + ((x-y)/2)(1)
y[0.50] = 0.75 y[0.50] = 0.84375 y[0.50] = 0.83203125 y[0.50] = 0.83349609375 y[0.50] = 0.83331298828125 y[0.50] = 0.8333358764648438
Euler Solution: y(2) = y(1) + h*((x-y)/2)(2)
y[1.00] = 0.7499997615814209
Modified Euler iterations:y(2) = y(1) + .5*h*(((x-y)/2)(1) + ((x-y)/2)(2)
y[1.00] = 0.7499997615814209 y[1.00] = 0.8229164183139801 y[1.00] = 0.8138018362224102 y[1.00] = 0.8149411589838564 y[1.00] = 0.8147987436386757 y[1.00] = 0.8148165455568233

Euler Solution: y(3) = y(2) + h*((x-y)/2)(3)
y[1.50] = 0.8611107402377911

Modified Euler iterations:y(3) = y(2) + .5*h*(((x-y)/2)(2) + ((x-y)/2)(3)
y[1.50] = 0.8611107402377911 y[1.50] = 0.9178236877476991 y[1.50] = 0.9107345693089606 y[1.50] = 0.9116207091138029 y[1.50] = 0.9115099416381975 y[1.50] = 0.9115237875726483

Euler Solution: y(4) = y(3) + h*((x-y)/2)(4)
y[2.00] = 1.0586415426231315
Modified Euler iterations:y(4) = y(3) + .5*h*(((x-y)/2)(3) + ((x-y)/2)(4)
y[2.00] = 1.0586415426231315 y[2.00] = 1.1027516068990952 y[2.00] = 1.0972378488645997 y[2.00] = 1.0979270686189118 y[2.00] = 1.0978409161496228 y[2.00] = 1.0978516852082838

Euler Solution: y(5) = y(4) + h*((x-y)/2)(5)
y[2.50] = 1.3233877543069634
Modified Euler iterations:y(5) = y(4) + .5*h*(((x-y)/2)(4) + ((x-y)/2)(5)
y[2.50] = 1.3233877543069634 y[2.50] = 1.357695577403087 y[2.50] = 1.3534070995160716 y[2.50] = 1.3539431592519484 y[2.50] = 1.3538761517849638

Euler Solution: y(6) = y(5) + h*((x-y)/2)(6)
y[3.00] = 1.6404133957887526
Modified Euler iterations:y(6) = y(5) + .5*h*(((x-y)/2)(5) + ((x-y)/2)(6)
y[3.00] = 1.6404133957887526 y[3.00] = 1.6670972872799508 y[3.00] = 1.663761800843551 y[3.00] = 1.664178736648101 y[3.00] = 1.6641266196725322

Case(ii) : **y' = (x - y)/2, y(0) = 1.0** with **h = 1/4**

Given
y[0.0] = 1.0
Euler Solution: y(1) = y(0) + h*((x-y)/2)(1)
y[0.250] = 0.875
Modified Euler iterations:y(1) = y(0) + .5*h*(((x-y)/2)(0) + ((x-y)/2)(1)
y[0.250] = 0.875 y[0.250] = 0.8984375 y[0.250] = 0.89697265625 y[0.250] = 0.897064208984375

Euler Solution: y(2) = y(1) + h*((x-y)/2)(2)
y[0.500] = 0.816176176071167
Modified Euler iterations:y(2) = y(1) + .5*h*(((x-y)/2)(1) + ((x-y)/2)(2)
y[0.500] = 0.816176176071167 y[0.500] = 0.8368563205003738 y[0.500] = 0.8355638114735484 y[0.500] = 0.835644593287725

Euler Solution: y(3) = y(2) + h*((x-y)/2)(3)
y[0.750] = 0.7936846013712966
Modified Euler iterations:y(3) = y(2) + .5*h*(((x-y)/2)(2) + ((x-y)/2)(3)

y[0.750] = 0.7936846013712966 y[0.750] = 0.8119317853121117 y[0.750] = 0.8107913363158108 y[0.750] = 0.8108626143780796

Euler Solution: y(4) = y(3) + h*((x-y)/2)(4)
y[1.000] = 0.8032508895617894
Modified Euler iterations:y(4) = y(3) + .5*h*(((x-y)/2)(3) + ((x-y)/2)(4)
y[1.000] = 0.8032508895617894 y[1.000] = 0.8193513439328768 y[1.000] = 0.8183450655346838 y[1.000] = 0.8184079579345709

Euler Solution: y(5) = y(4) + h*((x-y)/2)(5)
y[1.250] = 0.8411035237646307
Modified Euler iterations:y(5) = y(4) + .5*h*(((x-y)/2)(4) + ((x-y)/2)(5)
y[1.250] = 0.8411035237646307 y[1.250] = 0.8553098052268149 y[1.250] = 0.8544219126354284 y[1.250] = 0.8544774059223901

Euler Solution: y(6) = y(5) + h*((x-y)/2)(6)
y[1.500] = 0.9039146953929605
Modified Euler iterations:y(6) = y(5) + .5*h*(((x-y)/2)(5) + ((x-y)/2)(6)
y[1.500] = 0.9039146953929605 y[1.500] = 0.9164496480303976 y[1.500] = 0.9156662134905579 y[1.500] = 0.9157151781492978

Euler Solution: y(7) = y(6) + h*((x-y)/2)(7)
y[1.750] = 0.9887481031258607
Modified Euler iterations:y(7) = y(6) + .5*h*(((x-y)/2)(6) + ((x-y)/2)(7)
y[1.750] = 0.9887481031258607 y[1.750] = 0.9998083540466274 y[1.750] = 0.9991170883640794 y[1.750] = 0.9991602924692387

Euler Solution: y(8) = y(7) + h*((x-y)/2)(8)
y[2.000] = 1.093012893186083
Modified Euler iterations:y(8) = y(7) + .5*h*(((x-y)/2)(7) + ((x-y)/2)(8)
y[2.000] = 1.093012893186083 y[2.000] = 1.1027719368752444 y[2.000] = 1.1021619966446718 y[2.000] = 1.1022001179090826

Euler Solution: y(9) = y(8) + h*((x-y)/2)(9)
y[2.250] = 1.2144230184137998
Modified Euler iterations:y(9) = y(8) + .5*h*(((x-y)/2)(8) + ((x-y)/2)(9)
y[2.250] = 1.2144230184137998 y[2.250] = 1.223033938221066 y[2.250] = 1.2224957557331118 y[2.250] = 1.222529392138609

Euler Solution: y(10) = y(9) + h*((x-y)/2)(10)
y[2.500] = 1.3509613786303571
Modified Euler iterations:y(10) = y(9) + .5*h*(((x-y)/2)(9) + ((x-y)/2)(10)
y[2.500] = 1.3509613786303571 y[2.500] = 1.3585592480824138 y[2.500] =

1.3580843812416603 y[2.500] = 1.3581140604192075

Euler Solution: y(11) = y(10) + h*((x-y)/2)(11)
y[2.750] = 1.5008481797867843
Modified Euler iterations:y(11) = y(10) + .5*h*(((x-y)/2)(10) + ((x-y)/2)(11)
y[2.750] = 1.5008481797867843 y[2.750] = 1.5075521813920236 y[2.750] = 1.5071331812916962 y[2.750] = 1.5071593687979665

Euler Solution: y(12) = y(11) + h*((x-y)/2)(12)
y[3.000] = 1.6625130155689716
Modified Euler iterations:y(12) = y(11) + .5*h*(((x-y)/2)(11) + ((x-y)/2)(12)
y[3.000] = 1.6625130155689716 y[3.000] = 1.6684283103508373 y[3.000] = 1.6680586044269707 y[3.000] = 1.6680817110472124

Case(iii) : **y' = (x - y)/2, y(0) = 1.0** with **h = 1/8**

Given
y[0.0] = 1.0

Euler Solution: y(1) = y(0) + h*((x-y)/2)(1)
y[0.1250] = 0.9375
Modified Euler iterations:y(1) = y(0) + .5*h*(((x-y)/2)(0) + ((x-y)/2)(1)
y[0.1250] = 0.9375 y[0.1250] = 0.943359375 y[0.1250] = 0.94317626953125

Euler Solution: y(2) = y(1) + h*((x-y)/2)(2)
y[0.2500] = 0.8920456171035767
Modified Euler iterations:y(2) = y(1) + .5*h*(((x-y)/2)(1) + ((x-y)/2)(2)
y[0.2500] = 0.8920456171035767 y[0.2500] = 0.8975498788058758 y[0.2500] = 0.8973778706276789

Euler Solution: y(3) = y(2) + h*((x-y)/2)(3)
y[0.3750] = 0.8569217930155446
Modified Euler iterations:y(3) = y(2) + .5*h*(((x-y)/2)(2) + ((x-y)/2)(3)
y[0.3750] = 0.8569217930155446 y[0.3750] = 0.8620924634176603 y[0.3750] = 0.8619308799675942

Euler Solution: y(4) = y(3) + h*((x-y)/2)(4)
y[0.5000] = 0.8315024338597582
Modified Euler iterations:y(4) = y(3) + .5*h*(((x-y)/2)(3) + ((x-y)/2)(4)
y[0.5000] = 0.8315024338597582 y[0.5000] = 0.836359730596966 y[0.5000] = 0.8362079400739283

Euler Solution: y(5) = y(4) + h*((x-y)/2)(5)

y[0.6250] = 0.8151993908072874
Modified Euler iterations:y(5) = y(4) + .5*h*(((x-y)/2)(4) + ((x-y)/2)(5)
y[0.6250] = 0.8151993908072874 y[0.6250] = 0.8197623062048026 y[0.6250] = 0.8196197150986302

Euler Solution: y(6) = y(5) + h*((x-y)/2)(6)
y[0.7500] = 0.8074601603787794
Modified Euler iterations:y(6) = y(5) + .5*h*(((x-y)/2)(5) + ((x-y)/2)(6)
y[0.7500] = 0.8074601603787794 y[0.7500] = 0.8117465357129019 y[0.7500] = 0.8116125864837106

Euler Solution: y(7) = y(6) + h*((x-y)/2)(7)
y[0.8750] = 0.8077657241223026
Modified Euler iterations:y(7) = y(6) + .5*h*(((x-y)/2)(6) + ((x-y)/2)(7)
y[0.8750] = 0.8077657241223026 y[0.8750] = 0.8117923193808908 y[0.8750] = 0.8116664882790599

Euler Solution: y(8) = y(7) + h*((x-y)/2)(8)
y[1.0000] = 0.8156285192196802
Modified Euler iterations:y(8) = y(7) + .5*h*(((x-y)/2)(7) + ((x-y)/2)(8)
y[1.0000] = 0.8156285192196802 y[1.0000] = 0.8194110786347212 y[1.0000] = 0.8192928736530011

Euler Solution: y(9) = y(8) + h*((x-y)/2)(9)
y[1.1250] = 0.8305905320862623
Modified Euler iterations:y(9) = y(8) + .5*h*(((x-y)/2)(8) + ((x-y)/2)(9)
y[1.1250] = 0.8305905320862623 y[1.1250] = 0.8341438456947754 y[1.1250] = 0.8340328046445094

Euler Solution: y(10) = y(9) + h*((x-y)/2)(10)
y[1.2500] = 0.852221507509997
Modified Euler iterations:y(10) = y(9) + .5*h*(((x-y)/2)(9) + ((x-y)/2)(10)
y[1.2500] = 0.852221507509997 y[1.2500] = 0.8555594689839763 y[1.2500] = 0.8554551576879144

Euler Solution: y(11) = y(10) + h*((x-y)/2)(11)
y[1.3750] = 0.8801172663274216
Modified Euler iterations:y(11) = y(10) + .5*h*(((x-y)/2)(10) + ((x-y)/2)(11)
y[1.3750] = 0.8801172663274216 y[1.3750] = 0.883252927298937 y[1.3750] = 0.8831549378935771

Euler Solution: y(12) = y(11) + h*((x-y)/2)(12)
y[1.5000] = 0.9138981250585888

Modified Euler iterations:y(12) = y(11) + .5*h*(((x-y)/2)(11) + ((x-y)/2)(12)
y[1.5000] = 0.9138981250585888 y[1.5000] = 0.9168437461524608 y[1.5000] = 0.9167516954932773

Euler Solution: y(13) = y(12) + h*((x-y)/2)(13)
y[1.6250] = 0.9532074113216032
Modified Euler iterations:y(13) = y(12) + .5*h*(((x-y)/2)(12) + ((x-y)/2)(13)
y[1.6250] = 0.9532074113216032 y[1.6250] = 0.95597451009519 y[1.6250] = 0.9558880382585153

Euler Solution: y(14) = y(13) + h*((x-y)/2)(14)
y[1.7500] = 0.9977100692219482
Modified Euler iterations:y(14) = y(13) + .5*h*(((x-y)/2)(13) + ((x-y)/2)(14)
y[1.7500] = 0.9977100692219482 y[1.7500] = 1.000309465199494 y[1.7500] = 1.0002282340751956

Euler Solution: y(15) = y(14) + h*((x-y)/2)(15)
y[1.8750] = 1.0470913492635905
Modified Euler iterations:y(15) = y(14) + .5*h*(((x-y)/2)(14) + ((x-y)/2)(15)
y[1.8750] = 1.0470913492635905 y[1.8750] = 1.049533206241223 y[1.8750] = 1.049456898210672

Euler Solution: y(16) = y(15) + h*((x-y)/2)(16)
y[2.0000] = 1.1010555776593376
Modified Euler iterations:y(16) = y(15) + .5*h*(((x-y)/2)(15) + ((x-y)/2)(16)
y[2.0000] = 1.1010555776593376 y[2.0000] = 1.1033494434461277 y[2.0000] = 1.1032777601402906

Euler Solution: y(17) = y(16) + h*((x-y)/2)(17)
y[2.1250] = 1.1593250002283733
Modified Euler iterations:y(17) = y(16) + .5*h*(((x-y)/2)(16) + ((x-y)/2)(17)
y[2.1250] = 1.1593250002283733 y[2.1250] = 1.161479843978849 y[2.1250] = 1.1614125051116466

Euler Solution: y(18) = y(17) + h*((x-y)/2)(18)
y[2.2500] = 1.221638696360544
Modified Euler iterations:y(18) = y(17) + .5*h*(((x-y)/2)(17) + ((x-y)/2)(18)
y[2.2500] = 1.221638696360544 y[2.2500] = 1.2236629436446282 y[2.2500] = 1.2235996859170006

Euler Solution: y(19) = y(18) + h*((x-y)/2)(19)
y[2.3750] = 1.2877515588009272
Modified Euler iterations:y(19) = y(18) + .5*h*(((x-y)/2)(18) + ((x-y)/2)(19)

y[2.3750] = 1.2877515588009272 y[2.3750] = 1.289653124548429 y[2.3750] = 1.2895937006188196

Euler Solution: y(20) = y(19) + h*((x-y)/2)(20)
y[2.5000] = 1.357433335265581
Modified Euler iterations:y(20) = y(19) + .5*h*(((x-y)/2)(19) + ((x-y)/2)(20)
y[2.5000] = 1.357433335265581 y[2.5000] = 1.359219654714051 y[2.5000] = 1.3591638322312865

Euler Solution: y(21) = y(20) + h*((x-y)/2)(21)
y[2.6250] = 1.4304677281411309
Modified Euler iterations:y(21) = y(20) + .5*h*(((x-y)/2)(20) + ((x-y)/2)(21)
y[2.6250] = 1.4304677281411309 y[2.6250] = 1.4321457859080915 y[2.6250] = 1.432093346602874

Euler Solution: y(22) = y(21) + h*((x-y)/2)(22)
y[2.7500] = 1.5066515487479644
Modified Euler iterations:y(22) = y(21) + .5*h*(((x-y)/2)(21) + ((x-y)/2)(22)
y[2.7500] = 1.5066515487479644 y[2.7500] = 1.5082279061411892 y[2.7500] = 1.508178644972651

Euler Solution: y(23) = y(22) + h*((x-y)/2)(23)
y[2.8750] = 1.5857939228601574
Modified Euler iterations:y(23) = y(22) + .5*h*(((x-y)/2)(22) + ((x-y)/2)(23)
y[2.8750] = 1.5857939228601574 y[2.8750] = 1.5872747435327825 y[2.8750] = 1.5872284678867632

Euler Solution: y(24) = y(23) + h*((x-y)/2)(24)
y[3.0000] = 1.6677155443756573
Modified Euler iterations:y(24) = y(23) + .5*h*(((x-y)/2)(23) + ((x-y)/2)(24)
y[3.0000] = 1.6677155443756573 y[3.0000] = 1.66910661842644 y[3.0000] = 1.669063147362353

Runge-Kutta Method :

Runge-Kutta method here after called as RK method is the generalization of the concept used in Modified Euler's method.

In Modified Eulers method the slope of the solution curve has been approximated with the slopes of the curve at the end points of the each sub interval in computing the solution. The natural generalization of this concept is computing the slope by taking a weighted average of the slopes taken at more number of points in each sub interval. However, the implementation of the scheme differes from Modified Eulers method so

that the developed algorithm is explicit in nature. The final form of the scheme is of the form

y_{i+1} = y_i + (weighted average of the slopes) for **i = 0, 1, 2 . . .**
where **h** is the step length and y_i and y_{i+1} are the values of **y** at x_i and x_{i+1} respectively.

In genaral, the slope is computed at various points x_s in each sub interval $[x_i, x_{i+1}]$ and multiplied them with the step length **h** and then weighted average of it is then added to y_i to compute y_{i+1}. Thus the RK method with **v** slopes called as v-stage RKmethod can be written as

$$K_1 = h\, f(x_i, y_i)$$
$$K_2 = h\, f(x_i + c_2 h, y_i + a_{21} K_1)$$
$$K_3 = h\, f(x_i + c_3 h, y_i + a_{31} K_1 + a_{32} K_2)$$
. . .
. . .
. . .

$$K_v = h\, f(x_i + c_v h, y_i + a_{v1} K_1 + a_{v2} K_2 + \ldots + a_{vv-1} K_{v-1})$$
and

$$y_{i+1} = y_i + (W_1 K_1 + W_2 K_2 + \ldots + W_v K_v) \quad \text{for } \mathbf{i = 0, 1, 2 \ldots}$$

To determine the parameters c's, a's and W's in the above equation, y_{i+1} defined in the scheme is expanded interms of steplengh h and the resultant equation is then compared with Taylor series expansion of the solution of the differential equation upto a certain number of terms say **p**. Then the v-stage RKmethod will be of order **p** or is an p^{th} order RK method. Here for any **v>4** the maximum possible order **p** of the R Kmethod is always less than **v**. However, for any **v** lessthan or equal to **4**, it is possible derive an RK method of order **p = v**. Now, consider the case v = 2 to derive the 2-stage RK method. For this

$$K_1 = h\, f(x_i, y_i)$$
$$K_2 = h\, f(x_i + c_2 h, y_i + a_{21} K_1)$$
$$y_{i+1} = y_i + (W_1 K_1 + W_2 K_2) \quad \text{for } \mathbf{i = 0, 1, 2 \ldots}$$

Now by Taylor series expansion

$$y(x_{i+1}) = y(x_i) + h\, y'(x_i) + h^2 y''(x_i)/2! + h^3 y'''(x_i)/3! + o(h^4)$$
$$= y(x_i) + h\, f + h^2 (f_x + f_y f)/2! + h^3 (f_{xx} + 2f_{xy} f + f_{yy} f^2 + f_y (f_x + f_y f))/3! + o(h^4)$$

Also

$$K_1 = h\, f_i$$
$$K_2 = h\, f(x_i + c_2 h, y_i + a_{21} K_1)$$
$$= h(f_i + c_2 h f_x + a_{21} K_1 f_y + (c_2 h)^2 f_{xx}/2! + (a_{21} K_1)^2 f_{yy}/2! + c_2 h$$

$a_{21}K_1 f_{xy} + o(h^4)$)

$= h(f_i + c_2h f_x + a_{21}h f_i f_y + (c_2h)^2 f_{xx} /2! + (a_{21}h f_i)^2 f_{yy} /2! + c_2h a_{21}h f_i f_{xy} + o(h^4)$)

$y_{i+1} = y_i + (W_1+W_2) h f_i + h^2(W_2c_2f_x + W_2a_{21}f f_y) + h^3 W_2(c_2^2a_{21}f f_{xy} + a^2_{21}f^2 f_{yy})/2 + o(h^4)$

Now by comparing the equal powers of **h** in$y_{i+1}$and **y(x_{i+1})** we get

$W_1 + W_2 = 1$ $c_2W_2 = 1/2$ and $a_{21}W_2 = 1/2$

The solution of this system is

$a_{21} = c_2$, $W_2 = 1/(2c_2)$ **and** $W_1 = 1 - 1/(2c_2)$

where c_2 is any arbitrary constant not equal to zero. For these values of a_{21}, W_2 , W_1, since 2-stage RK method compares with Taylor series upto $h^2$for any value of c_2 the 2-stage RK method is of order two and hence this scheme is denoted in many text books as a second order RK method. Now, to give some numerical values to a_{21}, W_2 , W_1 first the valuec_2 of has to be fixed. Generally the value of c_2 is fixed such that the values of a_{21}, W_2 , W_1 are integers or some real numbers which easy to remember. Two of such cases are $c_2 = 1/2$ and $c_2=1$.

Case (i): $c_2 = 1/2$ $a_{21}= 1/2$, $W_2 = 1$, $W_1 = 0$. The corresopnding 2-stage (second order) RK method is

$K_1 = h f(x_i, y_i)$
$K_2 = h f(x_i + h/2, y_i + K_1 /2)$
$y_{i+1} = y_i + (K_2)$ for **i = 0, 1, 2 . . .**

or equivalently

$y_{i+1} = y_i + h f(x_i + h/2, y_i + h f(x_i, y_i) /2)$ for **i = 0, 1, 2 . . .**

Which is knothing but Eulers method with step length **h = 1/2.**

Case (ii): $c_2 = 1$ $a_{21}= 2$, $W_2 = W_1 = 1/2$. The corresopnding 2-stage (second order) RK method is

$K_1 = h f(x_i, y_i)$
$K_2 = h f(x_i + h, y_i + K_1)$

$y_{i+1} = y_i + (K_1 + K_2)/2$ for **i = 0, 1, 2 . . .**
or equivalently

$y_{i+1} = y_i + .5 h (f(x_i, y_i) + f(x_i + h, y_i + h f(x_i, y_i)))$ for **i = 0, 1, 2 . . .**

Which is knothing but the Modified Eulers method.

Following the same procedure one can develope the higher order RK methods by giving various values to v and comparing the obtained y_{i+1} with the same obtained by Taylor series method. Classical RK methods of order three and four are

1	RK method of order three (v = 3)	$K_1 = h\, f(x_i, y_i)$ $K_2 = h\, f(x_i + h/2, y_i + K_1/2)$ $K_3 = h\, f(x_i + h, y_i - K_1 + 2K_2)$ $y_{i+1} = y_i + (K_1 + 4K_2 + K_3)/6$
2	RK method of order three (v = 4)	$K_1 = h\, f(x_i, y_i)$ $K_2 = h\, f(x_i + h/2, y_i + K_1/2)$ $K_3 = h\, f(x_i + h/2, y_i + K_2/2)$ $K_4 = h\, f(x_i + h, y_i + K_3)$ $y_{i+1} = y_i + (K_1 + 2K_2 + 2K_3 + K_4)/6$

Example 1 **Find y(1.0) using RK method of order four by solving the IVP $y' = -2xy^2$, y(0) = 1 with step length 0.2. Also compre the solution obtained with RK methods of order three and two.**

Solution:
Given y' = -2*x*y*y, y[0] = 1.0
(Using RK method of order 4) with step length = 0.2

K1 = -0.0 K2 = -0.040000001192092904 K3 = -0.03841600109815598
K4 = -0.07397150516004751

y[0.20] = 0.9615327483765758

K1 = -0.07396362030033653 K2 = -0.10257533554202282 K3 = -0.09942553577510745
K4 = -0.11891661890710704

y[0.40] = 0.8620524180696251

K1 = -0.11890150298349086 K2 = -0.12883389087826705 K3 = -0.12724447323187424
K4 = -0.12958625565317425

y[0.60] = 0.7352783369268004

K1 = -0.12975221972783935 K2 = -0.12584296464465622

K3 = -0.1265778509537273
K4 = -0.11856521365315237

y[0.80] = 0.6097518261638406

K1 = -0.11897513618897959 K2 = -0.10900467458369312

K3 = -0.11098872146906143
K4 = -0.09950585680741567

y[1.00] = 0.5000071953135232

Comparison of the solution with RK method of orders two, three and four:

	x = 0.0	x = 0.2	x = 0.4	x = 0.6	x = 0.8	x = 1.0
2nd Order	1.0	0.9600	0.8603	0.7350	0.6116	0.5033
3rd Order	1.0	0.9616	0.8622	0.7355	0.6099	0.5
4th Order	1.0	0.9615	0.8620	0.7350	0.6098	0.5
Analytical	1.0	0.9615	0.8621	0.7353	0.6098	0.5

Example 2:
Find y in [0,3] by solving the initial value problem y' = (x - y)/2, y(0) = 1 using RK method of order four with h = 1/2 and 1/4.

Solution:
Given y' = (x-y)/2, y[0.000] = 1.0
(Using RK method of order 4) **with step-length = 0.5**

K1 = -0.25 K2 = -0.15625 K3 = -0.16796875
K4 = -0.0830078125
y[0.500] = 0.83642578125

K1 = -0.0841064453125 K2 = -0.0110931396484375

K3 = -0.020219802856445312
K4 = 0.04594850540161133
y[1.000] = 0.8196284770965576

K1 = 0.045092880725860596 K2 = 0.10195627063512802

K3 = 0.09484834689646959
K4 = 0.1463807940017432
y[1.500] = 0.9171422953950241

K1 = 0.14571442615124397 K2 = 0.19000012288233847

K3 = 0.18446441079095166
K4 = 0.22459832345350605
y[2.000] = 1.1036825982202458

K1 = 0.22407935044493854 K2 = 0.2585694316393212

K3 = 0.2542581714900234
K4 = 0.2855148075724327
y[2.500] = 1.3595574922662559

K1 = 0.28511062693343603 K2 = 0.3119717985667565

K3 = 0.30861415211259147
K4 = 0.3329570889052882
y[3.000] = 1.6694307617991593
Given y' = (x-y)/2, y[0.0000] = 1.0
(Using RK method of order 4) **with step-length = 0.25**

K1 = -0.125 K2 = -0.1015625 K3 = -0.10302734375 K4 = -0.08087158203125
y[0.2500] = 0.897491455078125

K1 = -0.08093643188476562 K2 = -0.06025290489196777

K3 = -0.06154562532901764
K4 = -0.04199322871863842
y[0.5000] = 0.8364036682372292

K1 = -0.04205045852965365 K2 = -0.023797304871550296

K3 = -0.02493812697518176
K4 = -0.007683192657755938
y[0.7500] = 0.8118695824237503

K1 = -0.007733697802968786 K2 = 0.00837465830971676

K3 = 0.007367886052673911
K4 = 0.022595316440446975
y[1.0000] = 0.8195940336507935

K1 = 0.02255074579365081 K2 = 0.03676632418154763 K3 = 0.03587785053230408
K4 = 0.0493160144771128
y[1.2500] = 0.8557865519338713

K1 = 0.049276681008266085 K2 = 0.061821888445249454

K3 = 0.06103781298043799
K4 = 0.07289695438571134
y[1.5000] = 0.9171020583080967

K1 = 0.07286224271148792 K2 = 0.08393335254201992

K3 = 0.08324140817761168
K4 = 0.09370706668928647
y[1.7500] = 1.0005885301147697

K1 = 0.09367643373565379 K2 = 0.10344665662717542

K3 = 0.10283601769645534
K4 = 0.11207193152359687
y[2.0000] = 1.1036408157658 55

K1 = 0.11204489802926812 K2 = 0.12066709190243885

K3 = 0.12012820478536568
K4 = 0.12827887243109742
y[2.2500] = 1.2239598764051842

K1 = 0.12825501544935197 K2 = 0.13586407698376748

K3 = 0.1353885106378665
K4 = 0.14258145161961866
y[2.5000] = 1.3595168167905574

K1 = 0.14256039790118033 K2 = 0.14927537303235655

K3 = 0.14885568708665806
K4 = 0.15520343701534808
y[2.7500] = 1.5085211426496503

K1 = 0.1551848571687937 K2 = 0.1611108035957441

K3 = 0.1607404319440597
K4 = 0.16634230317578624
y[3.0000] = 1.669392747887015

The Method of Successive Approximations for First Order Differential Equations(PICARD'S METHOD)

The Method of Successive Approximations (Picard's Iterative Method), if $dydt=f(t,y)$is a first order differential equation and with the initial condition $y(0)=0$ (if the initial condition is not $y(0)=0$ then we can apply a substitution to translate the differential equation so that $y(0)=0$ becomes the initial condition) and if both f and $\partial f \partial y$ are both continuous on some rectangle R for
which $-a\leq t\leq a$ and $-b\leq y\leq b$ then $\lim n\to\infty \phi n(t)=\lim n\to\infty \int t0 f(s,\phi n-1(s))ds=\phi(t)$ where $y=\phi(t)$ is the unique solution to this initial value problem.

Furthermore, recall that the functions $\{\phi 0,\phi 1,\phi 2,...,\phi n,...\}$ are successively better approximations of the unique solution $y=\phi(t)$. We start with $\phi 0(t)=0$ and the rest of the functions, $\phi 1,\phi 2,...,\phi n,...$ can be obtained with the following recursive formula:

rsive formula:

(1)

$$\phi n+1(t)=\int t0 f(s,\phi n(s))ds$$

We also noted that if $\phi k(t)=\phi k+1(t)$ for some k, then we have that $y=\phi k(t)$ is the unique solution we're looking for.

We will now look at another example of applying the method of successive approximations to solve first order initial value problems.

Example 1

Find the functions $\phi 1$, $\phi 2$, and $\phi 3$ using the Method of Successive Approximations for the differential equation $dydt=t2y-t$ with the initial condition $y(0)=0$.
Let $f(t,y)=t2y-t$. Note that f is continuous on all of R2 and $dfdt=2ty-1$ is continuous on all of R2 so a unique solution exists. Define $\phi 0(t)=0$. We will now compute the first three approximation functions:
(2)

$$\phi 1(t)=\int t0f(s,\phi 0(s))ds\phi 1(t)=\int t0s2(0)-sds\phi 1(t)=\int t0-sds\phi 1(t)=-t22$$

(3)

$$\phi 2(t)=\int t0f(s,\phi 1(s))ds\phi 2(t)=\int t0(s2(-s22)-s)ds\phi 2(t)=\int t0(-s42-s)ds\phi 2(t)=-s52\cdot 5-s22$$

(4)

$$\phi 3(t)=\int t0f(s,\phi 2(t))ds\phi 3(t)=\int t0(s2(-s52\cdot 5-s22)-s)ds\phi 3(t)=\int t0(-s72\cdot 5-s42-s)ds\phi 3(t)=-t 82\cdot 5\cdot 8-t52\cdot 5-t22$$

Example 2

Find the functions $\phi 1$, $\phi 2$, and $\phi 3$ using the Method of Successive Approximations for the differential equation $dydt=y-t+1$ with the initial condition $y(0)=0$. Then find the exact solution to this initial value problem by taking the limit of the sequence of approximations $\{\phi 0,\phi 1,\phi 2,...\}$ as $n\to\infty$.
Let $f(t,y)=y-t+1$. Clearly this function is continuous on all of R2 and $\partial f\partial t=-1$ is continuous on all of R2 as well, and so there exists a unique solution $\phi(t)$ to this differential equation. Define $\phi 0(t)=0$.
(5)

$$\phi 1(t)=\int t0f(s,\phi 0(s))ds\phi 1(t)$$

$$=\int t0(-s+1)d$$

$$s\phi 1(t)=-t22+t$$

(6)

$$\phi_2(t)=\int t0f(s,\phi 1(s))ds\phi 2(t)=\int t0(-s22+s-s+1)ds\phi 2(t)=\int t0(-s22+1)ds\phi 2(t)=-t33\cdot 2+t$$

(7)

$$\phi_3(t)=\int t0f(s,\phi 2(s))ds\phi 3(t)=\int t0(-s33\cdot 2+s-s+1)ds\phi 3(t)=\int t0(-s33\cdot 2+1)ds\phi 3(t)=-t44!+t$$

It's not hard to see that for $n\in$N we have that $\phi n(t)=-tn+1(n+1)!+t$. Taking the limit as $n\to\infty$ gives us that:
(8)

$$\lim n\to\infty\phi n(t)=\lim n\to\infty(-tn+1(n+1)!+t)\lim n\to\infty\phi n(t)=t$$

Therefore $\phi(t)=t$ is the unique solution to this initial value problem.
Clearly $\phi(t)=t$ satisfies the initial condition of $\phi(0)=0$. Furthermore, a quick substitution of this function into our differential equation shows that indeed $\phi(t)=t$ is a solution.

Finite Difference Method for the Solution of Laplace Equation

Finite Difference Method

The finite difference techniques are based on the replacement of differential equations with approximately equivalent finite difference equations. Whenever solving differential equations analytically is not easy then the differential equations are replaced by corresponding finite difference equations and those will be solved. Further, the differential equations provide exact values while the difference equations provide approximate values for the variables. Approximation of the solution relates to the number of grids given region is divided into. More are the grids and better is the approximation.

The finite difference equations are algebraic in nature and their approximate solutions are related to number of grid points of the region R.A finite difference solution basically involves the following three steps:

(i) Divide the solution region using grids and nodes into small rectangles

(ii) Approximate the given differential equation by equivalent finite difference equations that relate the solutions to the grid points.

(iii) Solve the difference equations subject to the prescribed boundary and/or initial conditions.

To find the solution of the function $u(x,y)$ on the region R, we divide the solution region R of xy – plane into equal rectangles or meshes of sides h and $k$along x and y directions respectively, as shown in Figure 2.We now construct a finite difference equation to represent equation (1) and boundary conditions (2).This will help us to calculate the values of $u(x,y)$at the nodes of the rectangles in the region R. The detailed derivation of finite difference equation from (1) is given in Appendix 1.

Here we directly consider the difference equation as

$$u_{i,j} = \frac{1}{4}\left(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} - h^2 f_{i,j}\right), \quad i,j = 2,3,4 \quad (3)$$

Also, the boundary conditions as

$$u(x_i, y_1) = 0, u(x_i, y_5) = 0, u(x_1, y_j) = 0, u(x_5, y_j) = 0, \quad i,j = 1,2,3,4,5 \quad (4)$$

Equation (3) represents a difference equation and (4) represents the boundary conditions equivalent respectively to the differential equation (1) and the boundary conditions (2).Here in (3) we have used the notations as $u_{i,j} = u(x_i, y_j)$ and

$f_{i,j} = f(x_i, y_j)$.Also, the set $\{x_1, x_2, x_3, x_4, x_5\}$is the set of partition points of the interval [0, 1] along x – axis with step length h. Similarly, the set $\{y_1, y_2, y_3, y_4, y_5\}$ is the set of partition points of the interval [0, 1] along y – axis with step length k. Of course in our present study we choose $h = k$.The numerical solutions for $u(x, y)$ is computed at all the nine free nodes of the region Rlocated at the locations $\{(i, j): i = 2, 3, 4 \; and \; j = 2, 3, 4\}$. These nine free nodes are numbered as shown in Figure 3.

Gauss Seidel method is applied to solve (3) numerically for the specific value of the function $f(x, y) = 1$.The function $f(x, y)$in fact admits any constant or expression in terms of $x$and y. For simplicity and to illustrate the procedure of numerical solution here we have selected that $f(x, y) = 1$. Other expressions for $f(x, y)$will be considered as and when it demands in our further work.

At the nodal points as shown in Figure 3 we compute the successive approximate solutions for $u(x, y)$using spreadsheet. The iterations and the approximations of the solutions are provided in a tabular form in Table 1.The Gauss – Seidel iterations for every node are continued till the approximate solution values converge to a constant.

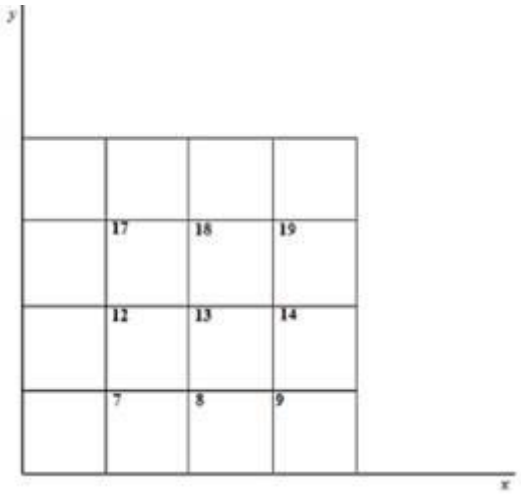

Figure 3. Numberedfree nodes of the region R.

Table 1. Solution using Gauss Seidel method.

Iterations	**Node numbers**								
	7	**8**	**9**	**12**	**13**	**14**	**17**	**18**	**19**
1	0	0	0	0	0	0	0	0	0
2	-0.0156	-0.0195	-0.0205	-0.0195	-0.0254	-0.0271	-0.0205	-0.0271	-0.0292
3	-0.0254	-0.0334	-0.0308	-0.0334	-0.0459	-0.0421	-0.0308	-0.0421	-0.0367
4	-0.0323	-0.0429	-0.0369	-0.0429	-0.0581	-0.0485	-0.0369	-0.0485	-0.0399
5	-0.0371	-0.0486	-0.0399	-0.0486	-0.0642	-0.0516	-0.0399	-0.0516	-0.0414
6	-0.0399	-0.0516	-0.0414	-0.0516	-0.0673	-0.0532	-0.0414	-0.0532	-0.0422
7	-0.0414	-0.0532	-0.0422	-0.0532	-0.0688	-0.0539	-0.0422	-0.0539	-0.0426