محاضرات التحليل العددي

Numerical Analyses Methods

Numerical Analyses Methods

Solution of Nonlinear Equations:-

1) Bisection Method:

The bisection method (sometimes called the midpoint method for equations) is a method used to estimate the solution of an .equation

we approach this problem by writing the equation in the form f(x) = 0 for some function f(x). This reduces the problem to .(finding a root for the function f(x))

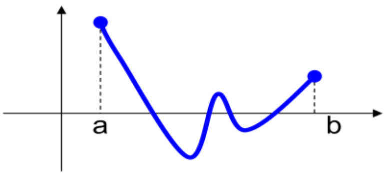
the Bisection Method needs a closed interval [a,b] for which the function f(x) is positive at one endpoint and negative at the other. In other words f(x) must satisfy the condition f(a)*f(b) < 0. This means that this algorithm can not be applied .to find tangential roots

There are several advantages that the Bisection method

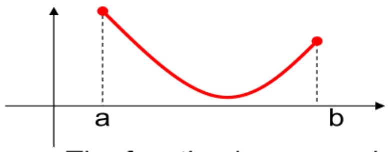
The number of steps required to estimate the root within the desired error can be easily computed before the algorithm is applied. This gives a way to compute how long the algorithm (will compute. (Real-time applications

NOTE

- If f(a) and f(b) have the same sign, the function may have an even number of real zeros or no real zeros in the interval [a, b].
- Bisection method can not be used in these cases.



The function has four real zeros



The function has no real zeros

<u>Bisection Algorithm</u>

The idea for the Bisection Algorithm is to cut the interval [a,b] you are given in half (bisect it) on each iteration by computing the midpoint P. The midpoint will replace either a or b depending on if the sign of $f(P)$ agrees with $f(a)$ or $f(b)$.

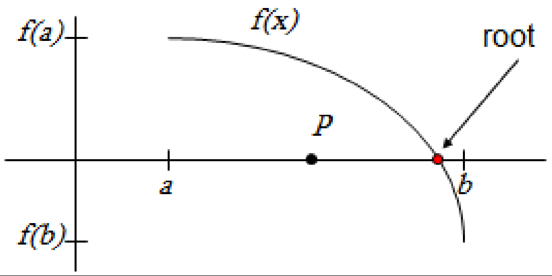
Step 1: Compute $P = (a+b)/2$

Step 2: If $sign(f(P)) = 0$ then end algorithm

else If $sign(f(P)) = sign(f(a))$ then $a = P$

else b =P

Step 3: Return to step 1



This shows how the points a, b and P are related.

<u>**Assumptions:**</u>

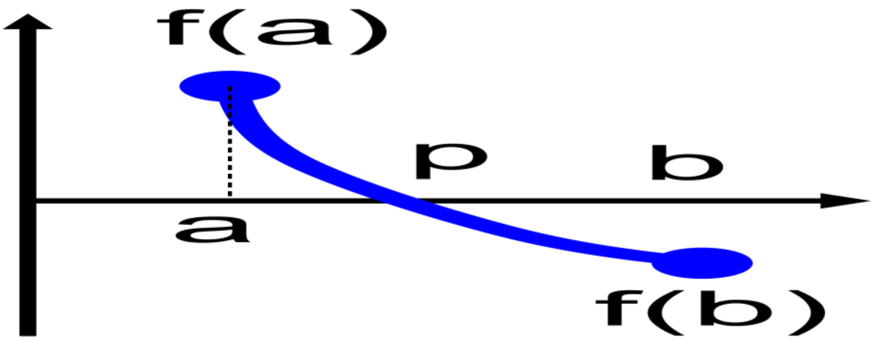
- f(x) is continuous on [a,b]
- f(a) f(b) < 0

<u>**Algorithm:**</u>

Loop

1. Compute the mid point p=(a+b)/2
2. Evaluate f(c)
3. If f(a) f(p) < 0 then new interval [a, p]
   If f(a) f(p) > 0 then new interval [p, b]

End loop



Example

Can you use Bisection method to find a zero of :

$f(x) = x^3 - 3x + 1$ in the interval $[0,1]$?

Answer:

$f(x)$ is continuous on $[0,1]$

and $f(0)*f(1) = (1)(-1) = -1 < 0$

$\Rightarrow$ Assumptions are satisfied

$\Rightarrow$ Bisection method can be used

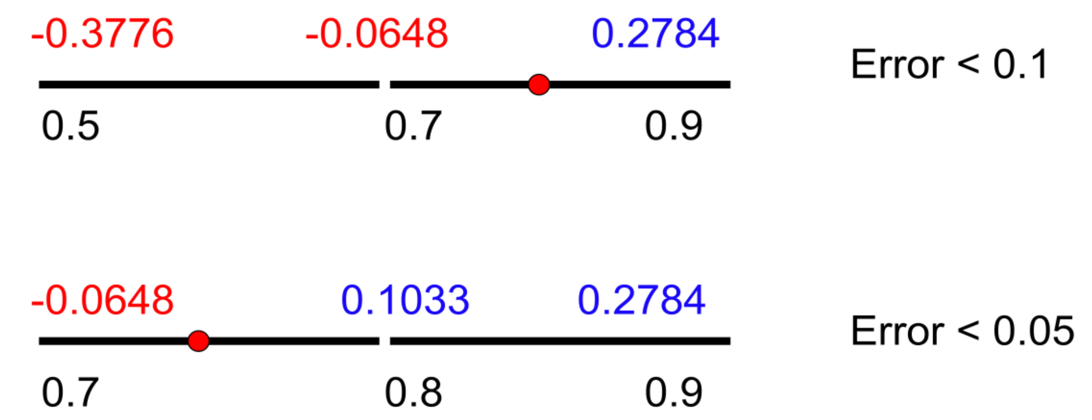
Example

Use Bisection method to find a root of the equation x = cos (x) with absolute error <0.02

(assume the initial interval [0.5, 0.9])

f(a)=-0.3776 f(b) =0.2784 Error < 0.2

a =0.5 c= 0.7 b= 0.9



Iteration	a	b	c= $\frac{(a+b)}{2}$	f(c)
1	0	1	0.5	-0.375
2	0	0.5	0.25	0.266
3	0.25	0.5	.375	-7.23E-3
4	0.25	0.375	0.3125	9.30E-2
5	0.3125	0.375	0.34375	9.37E-3

In order to compute the number of iteration we use the following equation :-

$$n \geq \frac{\log(b-a)-\log(\varepsilon)}{\log(2)}$$

Example

Lets apply the Bisection Method to the same function as we did for the Regula-Falsi Method. The equation is: x^3-2x-3=0, the function is: $f(x)=x^3-2x-3$.

This function has a root on the interval [0,2]

Iteration	a	b	x_{mid}	$f(a)$	$f(b)$	$f(x_{mid})$
1	0	2	1	-3	1	-4
2	1	2	1.5	-4	1	-2.262
3	1.5	2	1.75	-2.262	1	-1.140
4	1.75	2	1.875	-1.140	1	-.158

Advantages

- Simple and easy to implement
- One function evaluation per iteration
- The size of the interval containing the zero is reduced by 50% after each iteration
- The number of iterations can be determined a priori
- No knowledge of the derivative is needed
- The function does not have to be differentiable

Disadvantage

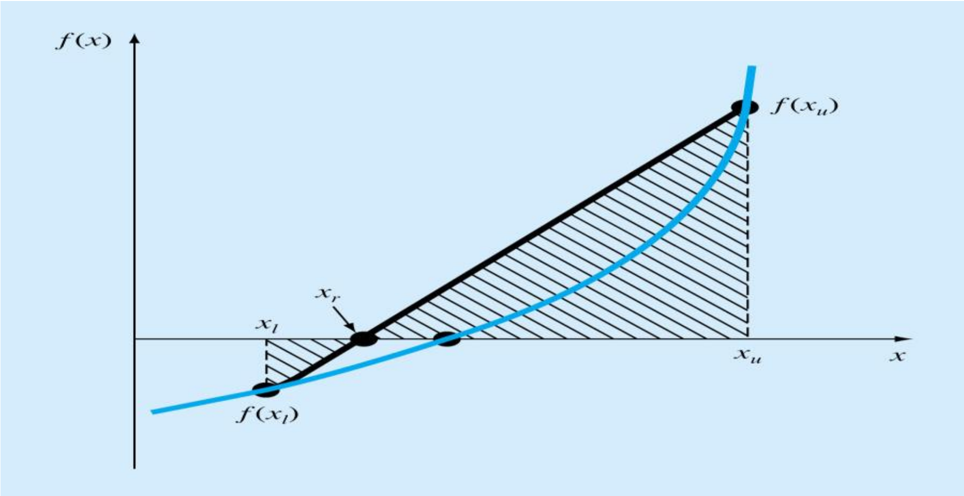
- Slow to converge

Good intermediate approximations may be discarded

2) False position method

If a real root is bounded by a and b of f(x)=0, then we can approximate the solution by doing a linear interpolation between the points

[a, f(a)] and [b, f(b)] to find the c value such that f(c)=0, where f(x) is the linear approximation of f(x).



1. Find a pair of values of a, b and x such that f_l=f(a) <0 and f_u=f(b) >0.
2. Estimate the value of the root from the following formula:-

$$c = \frac{af_u - bf_l}{f_u - f_l}$$

and evaluate f(c).

3. Use the new point to replace one of the original points, keeping the two points on opposite sides of the x axis.

If f(c)<0 then a=c == > f_l=f(c)

If f(c)>0 then b=c == > f_u=f(c)

If f(c)=0 then you have found the root and need go no further!

4. See if the new x_l and x_u are close enough for convergence to be declared. If they are not go back to step 2.

example1:- Root of $f(x) = x^3 - 0.165x^2 + 3.993 \times 10^{-4} = 0$ for false-position method.

Iteration	x_L	x_U	x_r	$\lvert\in_a\rvert\%$	$f(x_m)$
1	0.0000	0.1100	0.0660	----	-3.1944×10^{-5}
2	0.0000	0.0660	0.0611	8.00	-1.1320×10^{-5}
3	0.0611	0.0660	0.0624	2.05	-1.1313×10^{-7}

Example 2

Find the root of $f(x) = (x-4)^2(x+2) = 0$, using the initial guesses of $x_L = -2.5$ and $x_U = -1.0$, and a pre-specified tolerance of $\in_s = 0.1\%$.

Solution

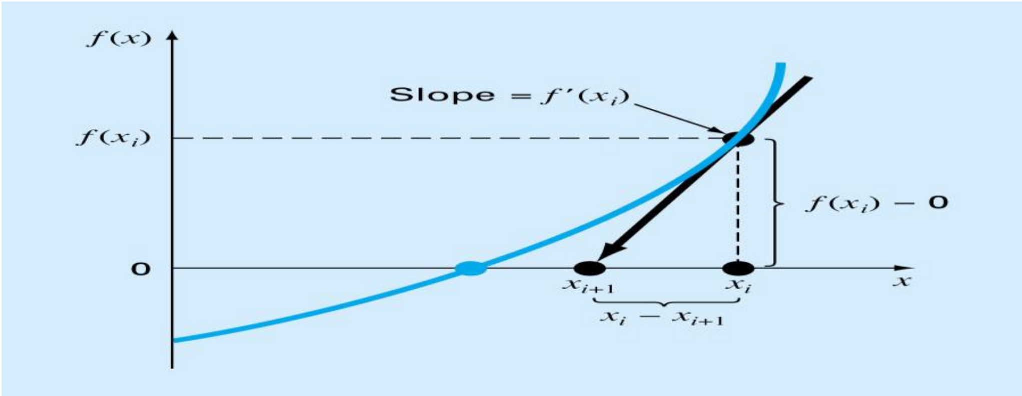
The individual iterations are not shown for this example, but the results are summarized in Table 2. It takes five iterations to meet the pre-specified tolerance.

Table 2 Root of $f(x) = (x-4)^2(x+2) = 0$ for false-position method.

Iteration	x_L	x_U	$f(x_L)$	$f(x_U)$	x_r	$\lvert\in_a\rvert\%$	$f(x_m)$
1	-2.5	-1	-21.13	25.00	-1.813	N/A	6.319
2	-2.5	-1.813	-21.13	6.319	-1.971	8.024	1.028
3	-2.5	-1.971	-21.13	1.028	-1.996	1.229	0.1542
4	-2.5	-1.996	-21.13	0.1542	-1.999	0.1828	0.02286
5	-2.5	-1.999	-21.13	0.02286	-2.000	0.02706	0.003383

3) Newton-Raphson Method:-

A convenient method for functions whose derivatives can be evaluated analytically. It may not be convenient for functions whose derivatives cannot be evaluated analytically.



Algorithm:-

algorithm

1)Evaluate $f'(x)$ symbolically.

2) Use an initial guess of the root, x_i, to estimate the new value of the root, x_{i+1}, as

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

3) Find the absolute relative approximate error $|\in_a|$ as

$$|\in_a| = \left|\frac{x_{i+1} - x_i}{x_{i+1}}\right| \times 100$$

Example 1 Cont.

Solution

To aid in the understanding of how this method works to find the root of an equation, the graph of f(x) is shown to the right,

where

$$f(x)=x^3-0.165x^2+3.993\times10^{-4}$$

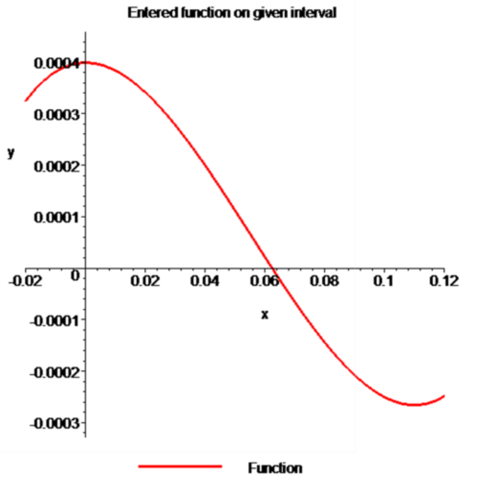


Figure 4 Graph of the function f(x)

Example 1 Cont.

<u>Iteration 1</u>
The estimate of the root is

$$\begin{aligned}
x_1 &= x_0 - \frac{f(x_0)}{f'(x_0)} \\
&= 0.05 - \frac{(0.05)^3 - 0.165(0.05)^2 + 3.993\times10^{-4}}{3(0.05)^2 - 0.33(0.05)} \\
&= 0.05 - \frac{1.118\times10^{-4}}{-9\times10^{-3}} \\
&= 0.05 - (-0.01242) \\
&= 0.06242
\end{aligned}$$

Example 1 Cont.

The absolute relative approximate error $\left|\in_a\right|$ at the end of Iteration 1 is

$$\left|\in_a\right| = \left|\frac{x_1 - x_0}{x_1}\right| \times 100$$

$$= \left|\frac{0.06242 - 0.05}{0.06242}\right| \times 100$$

$$= 19.90\%$$

The number of significant digits at least correct is 0, as you need an absolute relative approximate error of 5% or less for at least one significant digits to be correct in your result.

Example 1 Cont.

<u>Iteration 2</u>
The estimate of the root is

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

$$= 0.06242 - \frac{(0.06242)^3 - 0.165(0.06242)^2 + 3.993 \times 10^{-4}}{3(0.06242)^2 - 0.33(0.06242)}$$

$$= 0.06242 - \frac{-3.97781 \times 10^{-7}}{-8.90973 \times 10^{-3}}$$

$$= 0.06242 - \left(4.4646 \times 10^{-5}\right)$$

$$= 0.06238$$

Example 1 Cont.

The absolute relative approximate error $|\in_a|$ at the end of Iteration 2 is

$$
\begin{aligned}
|\in_a| &= \left|\frac{x_2 - x_1}{x_2}\right| \times 100 \\
&= \left|\frac{0.06238 - 0.06242}{0.06238}\right| \times 100 \\
&= 0.0716\ \%
\end{aligned}
$$

The maximum value of *m* for which $|\in_a| \le 0.5 \times 10^{2-m}$ is 2.844. Hence, the number of significant digits at least correct in the answer is 2.

Example 1 Cont.

<u>Iteration 3</u>
The estimate of the root is

$$
\begin{aligned}
x_3 &= x_2 - \frac{f(x_2)}{f'(x_2)} \\
&= 0.06238 - \frac{(0.06238)^3 - 0.165(0.06238)^2 + 3.993 \times 10^{-4}}{3(0.06238)^2 - 0.33(0.06238)} \\
&= 0.06238 - \frac{4.44 \times 10^{-11}}{-8.91171 \times 10^{-3}} \\
&= 0.06238 - \left(-4.9822 \times 10^{-9}\right) \\
&= 0.06238
\end{aligned}
$$

Example 1 Cont.

The absolute relative approximate error $|\in_a|$ at the end of Iteration 3 is

$$|\in_a| = \left|\frac{x_2 - x_1}{x_2}\right| \times 100$$

$$= \left|\frac{0.06238 \quad - 0.06238}{0.06238}\right| \times 100$$

$$= 0\%$$

The number of significant digits at least correct is 4, as only 4 significant digits are carried through all the calculations.

Newton's method

- Use Newton's method to find root of

$$f(x) = x^2 - 4\sin(x) = 0$$

- Derivative is

$$f'(x) = 2x - 4\cos(x)$$

so iteration scheme is

$$x_{k+1} = x_k - \frac{x_k^2 - 4\sin(x_k)}{2x_k - 4\cos(x_k)}$$

- Taking $x_0 = 3$ as starting value, we obtain

x	$f(x)$	$f'(x)$	h
3.000000	8.435520	9.959970	−0.846942
2.153058	1.294772	6.505771	−0.199019
1.954039	0.108438	5.403795	−0.020067
1.933972	0.001152	5.288919	−0.000218
1.933754	0.000000	5.287670	0.000000

.Example: Find the root of $e^{-x} - 3x = 0$

:Solution

$f(x) = e^{-x} - 3x$

$f(x) = -e^{-x} - 3$

With these, the Newton-Raphson solution

can be updated as

$$x_{i+1} = x_i + \frac{e^{-x_i} - 3x_i}{-e^{-x_i} - 3}$$

$-1 \longrightarrow 0{:}2795 \longrightarrow 0{:}5680 \longrightarrow 0{:}6172 \longrightarrow$
$0{:}6191 \longrightarrow 0{:}6191$

Converges much faster than the bisection

Example :-By using the Newton-Raphson's method find the positive root of the quadratic equation

x2 + 11x – 17 = 0 correct to 3 significant figures

.

Numerical analyses of integration

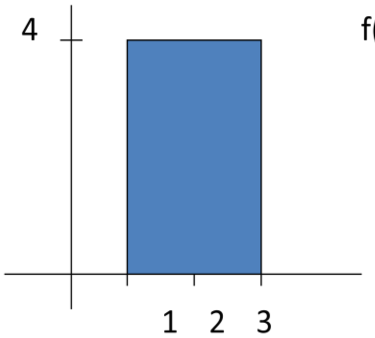
The Definite integral as the area of a region

If f is continuous and nonnegative on the closed interval [a, b], then the area of the region bounded by the graph of f, the x-axis, and the vertical lines x = a and x = b is given by

$$\boxed{area = \int_a^b f(x)dx}$$

Areas of common geometric figures.

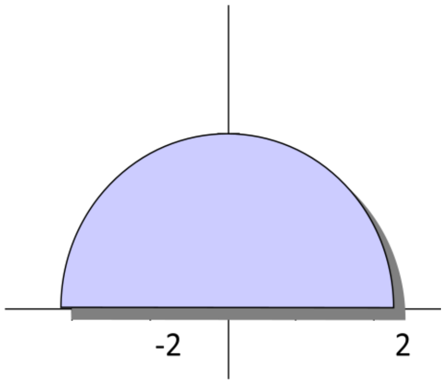
$$\int_1^3 4dx$$



f(x) = 4

Area = 4(2) = 8

$$\int_{-2}^{2} \sqrt{4-x^2}\,dx$$



$$A = \frac{1}{2}\pi r^2 = \frac{1}{2}\pi 2^2 = 2\pi$$

Riemann Sums and Definite Integrals

Definition of the Definite Integral

If f is defined on the closed interval [a, b] and the limit of a Riemann sum of f exists, then we say f is integrable on [a, b] and we denote the limit by

$$\lim_{\Delta x \to 0} \sum_{i=1}^{n} f(c_i)\Delta x_i = \int_a^b f(x)dx$$

The limit is called the <u>definite integral</u> of f from a to b. The number a is the <u>lower limit</u> of integration, and the number b is the <u>upper limit</u> of integration.

Introduction to area under a curve (cont.)

There are two methods we can use to find the area under a curve: **the trapezoidal rule, and Simpson's rule.**

For each method we must know:
f(x)- the function of the curve
n- the number of partitions or rectangles
(a, b)- the boundaries on the x-axis between which we are finding the area

$$\int_a^b f(x)$$

Trapezoidal Rule

TRAPEZOIDAL RULE ALWAYS begins with:

$f(x_0)$ and ends with $f(x_n)$

Within the brackets with every" f " being multiplied by 2 EXCEPT for the first and last terms

Trapezoidal Rule- Example

Remember: Trapezoidal Rule Only

$$A = \frac{b-a}{2n}[f(x_0) + 2f(x_1) + 2f(x_2) + 2f(x_3) + \ldots f(x_n)]$$

Given this problem below, what all do we need to know in order to find the area under the curve using Trapezoidal Rule?

$$f(x) = \int_0^4 x^3$$

4 partitions

Simpson's Rule

Simpson's rule is the most accurate method of finding the area under a curve. It is better than the trapezoidal rule because instead of using straight lines to model the curve, it uses parabolic arches to approximate each part of the curve. The equation that is used for Simpson's Rule ALWAYS begins with:

$f(x_0)$ And ends with $f(x_n)$

Within the brackets with every "f" being multiplied by alternating coefficients of 4 and 2 EXCEPT the first and last terms.

In Simpson's Rule, n MUST be even.

Simpson's Rule- Example

Remember: Simpson's Rule Only

$$A = \frac{b-a}{3n}[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \ldots f(x_n)]$$

Given this problem below, what all do we need to know in order to find the area under the curve using Simpson's Rule?

$$f(x) = \int_0^4 x^3$$

4 Partitions

Example: Simpson's Rules

Evaluate the integral $\int_0^4 xe^{2x}dx$

- **Simpson's 1/3-Rule**

$$I=\int_0^4 xe^{2x}dx \approx \frac{h}{3}\left[f(0)+4f(2)+f(4)\right]$$
$$=\frac{2}{3}\left[0+4(2e^4)+4e^8\right]=8240.411$$
$$\varepsilon=\frac{5216.926-8240.411}{5216.926}=-57.96\%$$

- **Simpson's 3/8-Rule**

$$I=\int_0^4 xe^{2x}dx \approx \frac{3h}{8}\left[f(0)+3f(\frac{4}{3})+3f(\frac{8}{3})+f(4)\right]$$
$$=\frac{3(4/3)}{8}\left[0+3(19.18922)+3(552.33933)+11923.832\right]=6819.209$$
$$\varepsilon=\frac{5216.926-6819.209}{5216.926}=-30.71\%$$

USC

Example

Approximate $\int_0^4 x^2dx$ using trapizume rule , n = 8 subintervals.

Δx = (4-0)/8 = 1/2 x0 = 0 x1 = 0.5 , x2 = 1 ,x3=1.5 , x4=2 ,x5=2.5 ,x6=3 ,x7=3.5 ,x8=4

$$\int_0^4 x^2dx \approx \frac{\Delta x}{2}\left[f(x_0)+2f(x_1)+2f(x_2)+\cdots+f(x_n)\right]$$
$$=\frac{1/2}{2}\left[f(0)+2f(0.5)+2f(1)+\cdots+f(4)\right]$$
$$=0.25\left[0+2(0.25)+2(1)+\cdots+16\right]$$
$$=21.5$$

Example

Estimate $\int_0^4 x^2 dx$ using Simpson's Rule and n = 4.
Here, Δx = (4-0)/4 = 1.

$$\begin{aligned}
\int_a^b f(x)\,dx &\approx \frac{\Delta x}{3}\left[f(x_0)+4f(x_1)+2f(x_2)+4f(x_3)+f(x_4)\right] \\
&= \frac{1}{3}\left[f(0)+4f(1)+2f(2)+4f(3)+f(4)\right] \\
&= \frac{1}{3}\left[0^2+4(1)^2+2(2)^2+4(3)^2+(4)^2\right] \\
&= 64/3 \approx 21.333
\end{aligned}$$

NUMERICAL SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS

Differential equations are among the most important mathematical tools used in producing models in the physical sciences, biological sciences, and engineering. In this text, we consider numerical methods for solving ordinary differential equations, that is, those differential equations that have only one independent variable.
The differential equations we consider in most of the book are of the form

$Y'(t) = f(t, Y(t))$,

where $Y(t)$ is an unknown function that is being sought. The given function $f(t, y)$ of two variables defines the differential equation.

- A first order initial value problem of ODE may be written in the form

$$y'(t) = f(y,t), \qquad y(0) = y_0$$

- Example:

$$y'(t) = 3y + 5, \qquad y(0) = 1$$
$$y'(t) = ty + 1, \qquad y(0) = 0$$

- Numerical methods for ordinary differential equations calculate solution on the points, $t_n = t_{n-1} + h$ where h is the steps size .

Methods to find appraoximate solution of ORDINARY DIFFERENTIAL EQUATIONS

- **Euler Methods**
- **Modified Euler Method**
- **Runge-Kutta Methods** Second Order

1) EULER 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 $\mathbf{t_i}$ and $\mathbf{t_{i+1}}$(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 $\mathbf{y_{i+1}}$ and then improving it by making use of average slope.

Consider the forward difference approximation for first derivative

$$y_n' \cong \frac{y_{n+1} - y_n}{h}, \qquad h = t_{n+1} - t_n$$

- **Rewriting the above equation we have**

$$y_{n+1} = y_n + hy_n', \qquad y_n' = f(y_n, t_n)$$

- **So,** y_n **is recursively calculated as**

$$
\begin{aligned}
y_1 &= y_0 + hy_0' = y_0 + h\, f(y_0, t_0) \\
y_2 &= y_1 + h\, f(y_1, t_1) \\
&\vdots \\
y_n &= y_{n-1} + h\, f(y_{n-1}, t_{n-1})
\end{aligned}
$$

Example: solve

$$y' = ty + 1, \quad y_0 = y(0) = 1, \quad 0 \le t \le 1, \quad h = 0.25$$

Solution:

for $t_0 = 0, \quad y_0 = y(0) = 1$

for $t_1 = 0.25, \quad y_1 = y_0 + hy_0'$

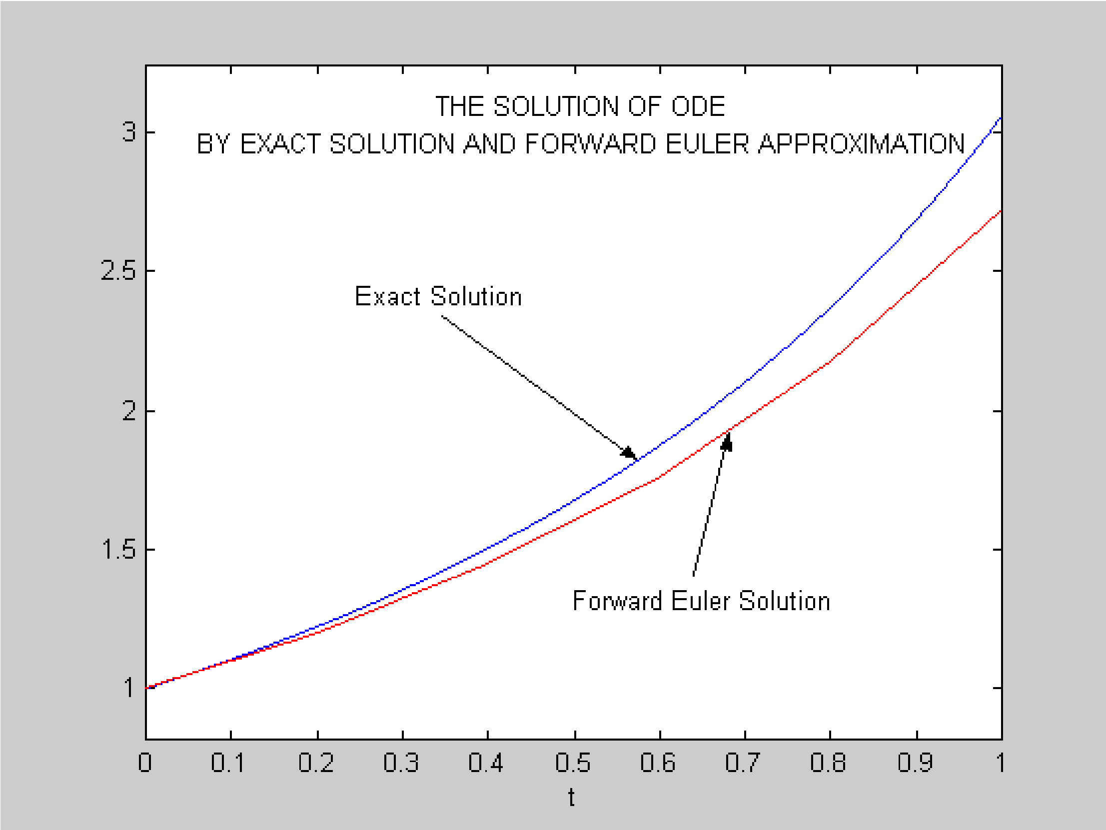
$$= y_0 + h(t_0 y_0 + 1)$$

$$= 1 + 0.25(0*1 + 1) = 1.25$$

for $t_2 = 0.5, \quad y_2 = y_1 + hy_1'$

$$= y_1 + h(t_1 y_1 + 1)$$

$$= 1.25 + 0.25(0.25*1.25 + 1) = 1.5781$$



2)Modified Euler Method

- Modified Euler method is derived by applying the trapezoidal rule to integrating $y_n' = f(y,t)$; So, we have

$$y_{n+1} = y_n + \frac{h}{2}(y'_{n+1} + y'_n), \qquad y_n' = f(y_n, t_n)$$

- If *f* is linear in *y*, we can solved for y_{n+1} similar as backward euler method
- If *f* is nonlinear in *y*, we necessary to used the method for solving nonlinear equations i.e. successive substitution method (*fixed point*)

Example: solve

$$y' = ty + 1, \quad y_0 = y(0) = 1, \quad 0 \le t \le 1, \quad h = 0.25$$

Solution:

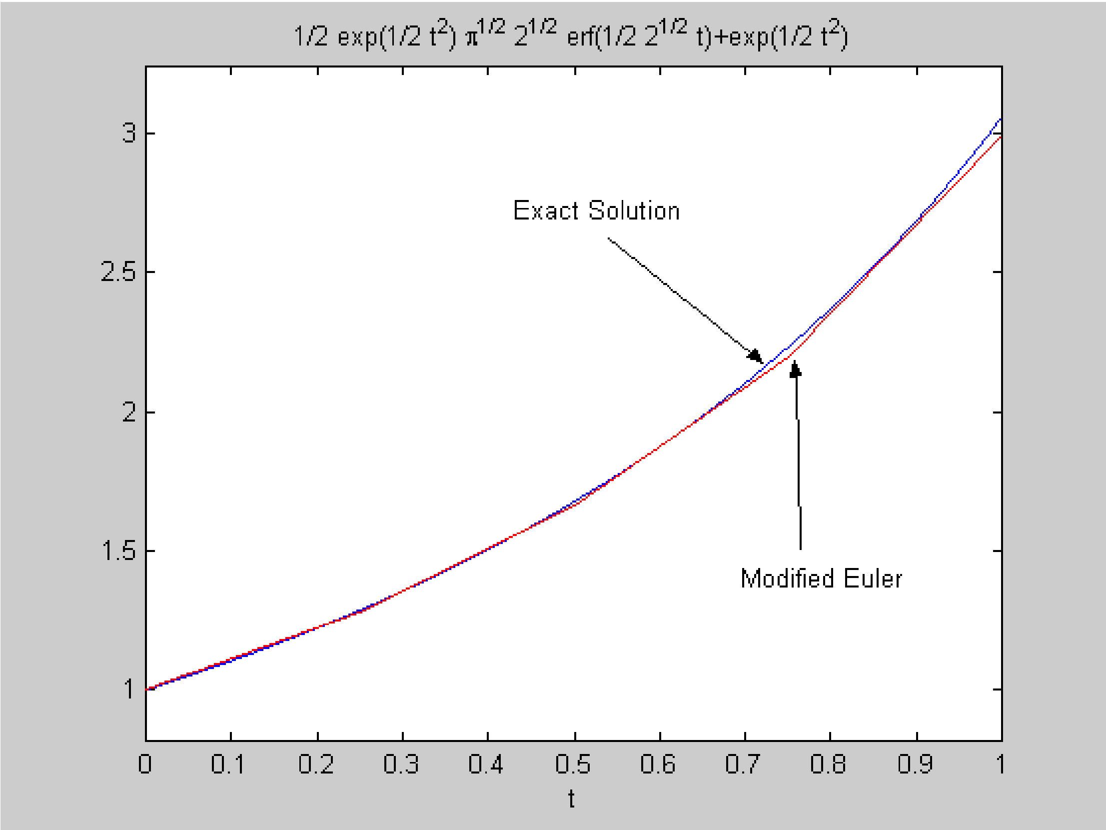
f is linear in y. So, solving the problem using modified Euler method for y_n yields

$$y_n = y_{n-1} + \frac{h}{2}(y'_{n-1} + y'_n)$$

$$= y_{n-1} + \frac{h}{2}(t_{n-1}y_{n-1} + 1 + t_n y_n + 1)$$

$$\Leftrightarrow \quad y_n(1 - \frac{h}{2}t_n) = y_{n-1}(1 + \frac{h}{2}t_{n-1}) + h$$

$$\Leftrightarrow \quad y_n = \frac{(1 + \frac{h}{2}t_{n-1})}{(1 - \frac{h}{2}t_n)} y_{n-1} + h$$



Example 1:

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

Solution: $\mathbf{f(x, y) = -2xy^2}$

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

<u>**Case(ii)**</u> : **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

3)Second Order Runge-Kutta Method

- The second order Runge-Kutta (RK-2) method is derived by applying the trapezoidal rule to integrating $y' = f(y,t)$ over the interval $[t_n, t_{n+1}]$. So, we have

$$y_{n+1} = y_n + \int_{t_n}^{t_{n+1}} f(y,t)dt$$

$$= y_n + \frac{h}{2}\left(f(y_n,t_n) + f(\bar{y}_{n+1}, t_{n+1})\right)$$

We estimate $\bar{y}_{n+1}$ by the forward euler method.

So, we have

$$y_{n+1} = y_n + \frac{h}{2}\left(f(y_n,t_n) + f(y_n + hf(y_n,t_n), t_{n+1})\right)$$

Or in a more standard form as

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

where $k_1 = hf(y_n, t_n)$

$$k_2 = hf(y_n + k_1, t_{n+1})$$