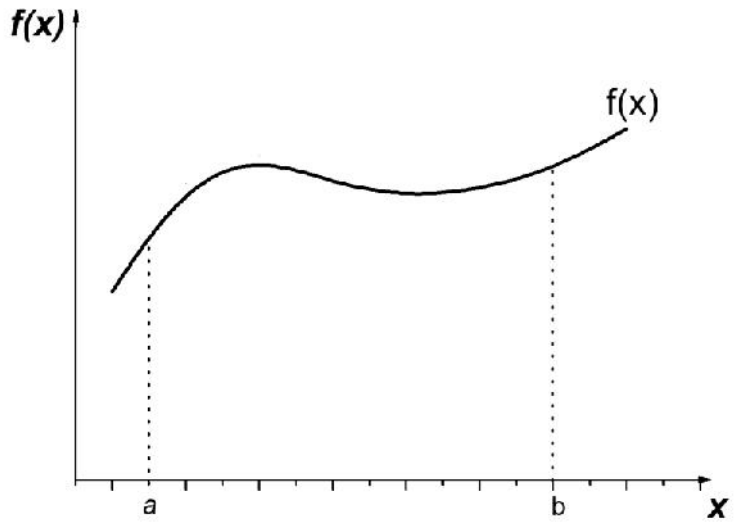
Sayısal İntegral

Bir f(x) fonksiyonunun tanımlı olduğu bir [a,b] aralığındaki belirli aşağıdaki gibi tanımlanır;

S, [a,b] aralığında f(x) eğrisi ile x-ekseni arasındaki kalan yüzeyin alanıdır.

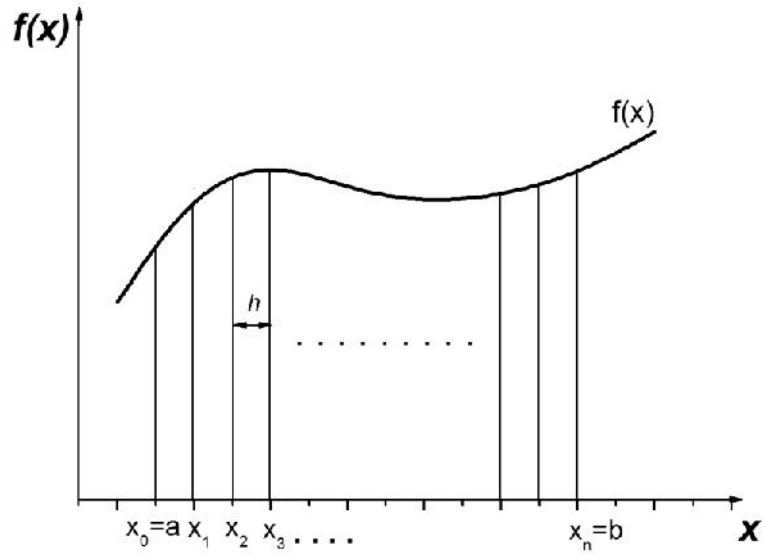


Şekilde [a,b] aralığında eşit aralıklarla N sayıda nokta belirlersek, adım uzunluğu aşağıdaki gibi olur;

$$\frac{(b-a)}{N} = h$$

a ve b noktalarını da katarak bo noktaları şöyle adlandırabiliriz.

$x_0=a, \quad x_i=a+ih, \quad x_N=b \quad (i=1,2,3,\ldots,N-1)$



N tane yamuk alanı toplanırsa, sayısal integral için trapez formülü bulunmuş olur.

$$S = \frac{(f_0 + f_1)}{2} h + \frac{(f_1 + f_2)}{2} h + ... + \frac{(f_{N-1} + f_N)}{2} h$$

$$\int_a^b f(x)dx = h\left[\frac{1}{2} f_0 + f_1 + f_2 + f_2 + ... + f_{N-1} + f_N\right] + O(h^2)$$

bulunur.

Soru:

$$\int_{0.5}^{1} \sqrt{x}\cos(x)\,dx$$

İntegralini trapez yöntemiyle hesaplayınız. (Aralığı 4 eşit parçaya bölünüz n=4)

$$h = \frac{1-0.5}{4} = 0.125$$

x	**f(x)**
0.5	0.6205
0.625	0.6411
0.75	0.6337
0.875	0.5996
1	0.5403

$$I \approx h\left[\frac{f_0}{2} + f_1 + f_2 + f_3 + \frac{f_4}{2}\right]$$

$$I \approx 0.125\left[\frac{0.6205}{2} + 0.6411 + 0.6337 + 0.5996 + \frac{0.5403}{2}\right] = 0.3072$$

```
from numpy import *
def yamuk(a,b,n):
    if n<1 or a>b:
        print("n,a ve b yi kontrol ediiz!")
    else:
        h=(b-a)/n
        s=0.5*(f(a)+f(b))
        for i in range(1,n):
            x=a+i*h
            s=s+f(x)
        return h*s
def f(x):
    return sqrt(x)*cos(x)
a=0.5
b=1.0
n=4
integral= float(yamuk(a,b,n))
print ("integralin degeri=", "%.10f" % integral)
```

Sayısal Türev

f(x) fonksiyonunu x civavarında ileri ve geri taylor seri açılımları aşağıdaki gibidir.

$$
\begin{aligned}
f(x+h) &= f(x)+hf'(x)+\frac{h^2}{2!}f''(x)+\frac{h^3}{3!}f'''(x)+\frac{h^4}{4!}f^{(4)}(x)+\cdots \\
f(x-h) &= f(x)-hf'(x)+\frac{h^2}{2!}f''(x)-\frac{h^3}{3!}f'''(x)+\frac{h^4}{4!}f^{(4)}(x)+\cdots \\
f(x+2h) &= f(x)+2hf'(x)+\frac{(2h)^2}{2!}f''(x)+\frac{(2h)^3}{3!}f'''(x)+\frac{(2h)^4}{4!}f^{(4)}(x) \\
f(x-2h) &= f(x)-2hf'(x)+\frac{(2h)^2}{2!}f''(x)-\frac{(2h)^3}{3!}f'''(x)+\frac{(2h)^4}{4!}f^{(4)}(x)
\end{aligned}
$$

Yukardaki ifadeler kullanılarak,

$$
\begin{aligned}
f(x+h)+f(x-h) &= 2f(x)+h^2f''(x)+\frac{h^4}{12}f^{(4)}(x)+\cdots \\
f(x+h)-f(x-h) &= 2hf'(x)+\frac{h^3}{3}f'''(x)+\cdots \\
f(x+2h)+f(x-2h) &= 2f(x)+4h^2f''(x)+\frac{4h^4}{3}f^{(4)}(x)+\cdots \\
f(x+2h)-f(x-2h) &= 4hf'(x)+\frac{8h^3}{3}f'''(x)+\cdots
\end{aligned}
$$

yazabiliriz.

f(x+h) - f(x-h) ‘tan:

$$f'(x)=\frac{f(x+h)-f(x-h)}{2h}+O(h^2)$$

f(x+h) + f(x-h) ‘tan:

$$f''(x)=\frac{f(x+h)-2f(x)+f(x-h)}{h^2}+O(h^2)$$

Soru:

Yanda verilen f(x) = cos(x) fonksiyonunun değerlerini kullanarak

h=0.01 için x=0.5 için fonksiyonun birinci türevinin değerini hesaplayınız.

x_i	$\cos(x_i)$
0.45	0.900447
0.46	0.896052
0.47	0.891568
0.48	0.886995
0.49	0.882333
0.50	0.877583
0.51	0.872745
0.52	0.867819
0.53	0.862807
0.54	0.857709
0.55	0.852525

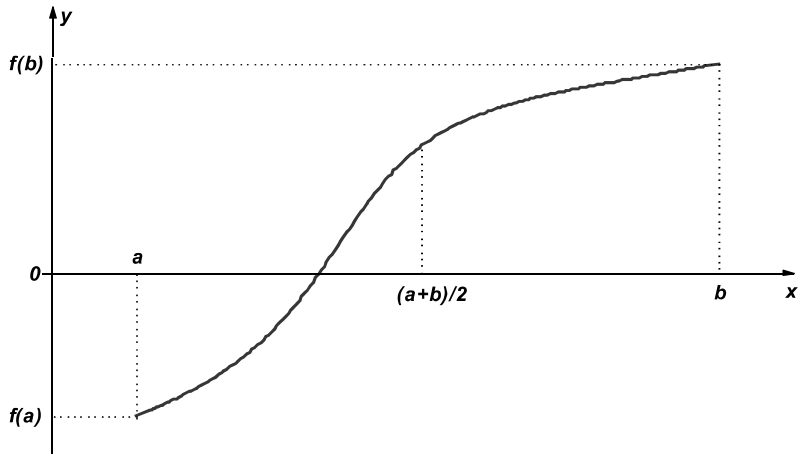
$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

$$\begin{aligned} f'(0.50) &\approx \frac{f(0.51) - f(0.49)}{2 \times 0.01} \\ &= \frac{0.872745 - 0.882333}{0.02} = -0.479418 \\ f'(0.5) &= -\sin(0.5) = -0.479426 \end{aligned}$$

```
from math import *
def f(x):
    return cos(x)
a=0.5
h=0.01
f1=(f(a+h)-f(a-h))/(2*h)
print("%.10f" % float(f1),"%.10f" % -sin(a))
```

Yarılama Yöntemi:

[a,b] aralığında sürekli bir fonksiyon için f(a)•f(b)<0 oluyorsa, yani fonksiyon işaret değiştiriyorsa, bu [a,b] aralığında bir kökü vardır.



Foksiyonun bu aralıkta kökü varsa, aralığın orta noktası olan

$$x_m = \frac{(a+b)}{2}$$

noktasına bakılır.

- Eğer f(a)•f(x_m) < 0 oluyorsa, kök solda demektir ve yeni aralık olarak [a,x_m] alınır.
- f(x_m)•f(b) < 0 oluyorsa, kök sağda demektir. O zaman yeni aralık [x_m,b] olur.

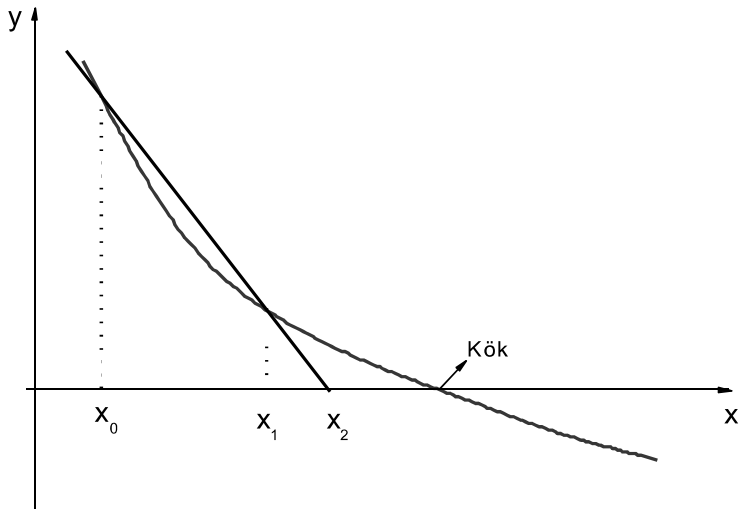
Bu işlem tekrarlandıkça(iterasyon), giderek köke yaklaşılır ve önceden belirlenmiş olan tolerans değerinde (ε) kök bulunur.

```
from math import *
def f(x):
    return exp(x)*log(x)-x*x
def yarila(a,b,tol):
    if f(a)*f(b)>0.0:
        print("Bu aralikta kok yok.")
    else:
        dx=b-a
```

```
    while(abs(dx)>tol):
        xm=(a+b)/2
        if(f(a)*f(xm))<0.0:
            b=xm
            dx=b-a
        else:
            a=xm
            dx=b-a
    return xm
a=1.0
b=3.0
tol=1.0e-8
print("X = " ,yarila(a,b,tol))
```

Kiriş Yöntemi:

Yarılama yönteminde kökün bulunduğu [a,b] aralığının hep ortası alınarak kök'e yaklaşılıyordu. Buna karşın, f(a) ve f(b) değerlerine bakılarak kökün hangi tarafa daha yakın olacağını tahmin edebiliriz. |f(a)|>|f(b)| ise kök b değerine daha yakın olacaktır. Kiriş yöntemi bu özelliği kullanan hızlı bir yöntemdir.



Kiriş yönteminde verilen iki noktadan geçen kirişin x-eksenine ulaştığı noktayla devam edilir. Birbirine yakın x_0 ve x_1 gibi iki nokta ele alalım . Bu noktaların kökü iki taraftan sarması şart değildir.

Öncelikle bu iki noktayı birleştiren doğrunun denklemini yazalım;

$$\frac{y-f(x_1)}{f(x_1)-f(x_0)}=\frac{x-x_1}{x_1-x_0}$$

Bu doğru f(x) fonksiyonu için x_0 ve x_1 noktaları arasındaki kiriş olur. Kirişin x-eksenini kestiği x_2 noktasını bulmak için doğru denkleminde y=0 alırsak,

$$\frac{0-f(x_1)}{f(x_1)-f(x_0)}=\frac{x_2-x_1}{x_1-x_0}$$

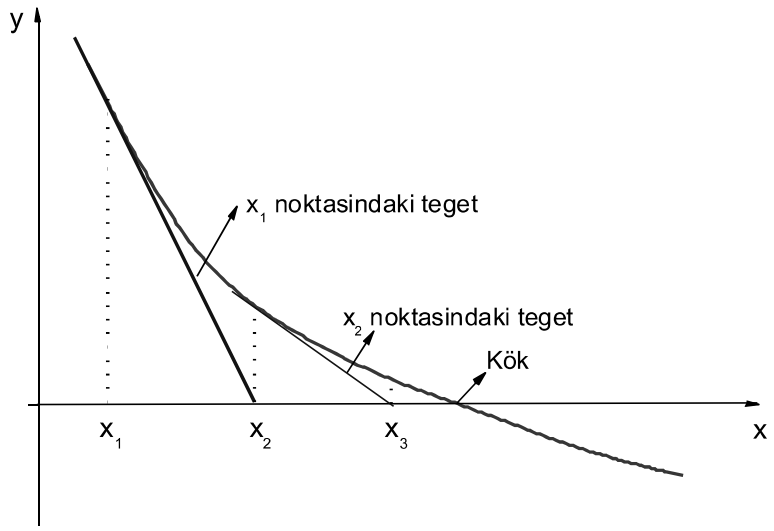
$$x_2=x_1-\frac{(x_1-x_0)f(x_1)}{f(x_1)-f(x_0)}$$

x_2 noktası köke daha yakın olacaktır. Bu kez x_1 ve x_2 noktalarından geçen kirişi esas alıp, aynı işlemlerle yeni x_3 noktası hesaplayabiliriz. Bulunan her yeni x_i değeri köke daha yakın olur ve sonunda belli hata payı içinde kök bulunur.

```
from math import *
def f(x):
    return exp(x)*log(x)-x**2
def kiris(a,b,tol):
    x0=a
    x1=b
    dx=b-a
    while(abs(dx)>tol):
        x2=x1-(x1-x0)*f(x1)/(f(x1)-f(x0))
        x0=x1
        x1=x2
        dx=x1-x0
    return x2
a=3.0
b=5.0
tol=1.0e-8
x=kiris(a,b,tol)
print( "x = ", x)
```

Newton-Raphson Yöntemi:

Kiriş yönteminde eğrinin iki noktadan geçen kirişin uzantısı alınıyordu. Bunun için, başlangıçta iki nokta verilmesi gerekiyordu. Buna benzer bir yöntemi, tek bir noktadan başlayıp, o noktadaki teğeti kullanarak da yapabiliriz. Sadece bir noktayla başlatılan ve f'(x) türevinin de bilinmesi gereken bu hızlı yöntem "Newton-Raphson Yöntemi" olarak bilinir. Bu yöntemde teğet uzantısının x-eksenine ulaştığı nokta ile devam edilerek köke ulaşılır.



Şekilde görüldüğü gibi, x_1 noktasında f(x) eğrisine teğet olan doğrunun denklemi;

$$y - f(x_1) = f'(x_1)(x - x_1)$$

Bu teğetin x-eksenini kestiği x_2 noktasını bulmak için denklemde y=0 alınırsa;

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

Bu kez x_2 noktasındaki teğeti esas alıp yeni bir x_3 noktası bulunur. Buradan devamla, x_i noktasından sonraki nokta aşağıdaki gibi olur.

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

Bulunan her yeni nokta köke daha da yaklaşır ve sonunda belli bir hata payı içinde köke erişilmiş olur.

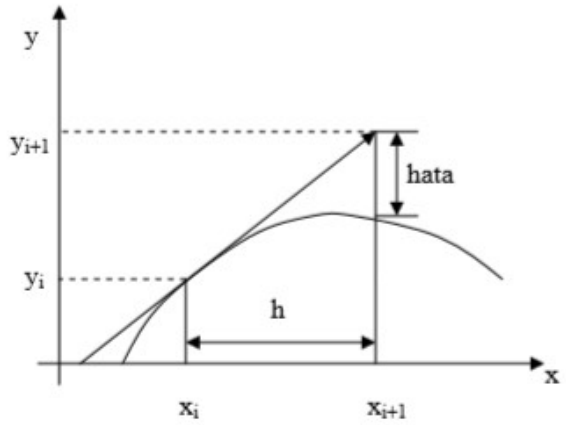
Kiriş yönteminde olduğu gibi, bu yöntemde de çift katlı kökler de bulunabilir.

Newton-Raphson yöntemi hem interval yarılama hem de kiriş yöntemine göre çok daha hızlıdır. Her iterasyonda anlamlı sayı iki katına çıkar.

```
from math import *
def f(x):
    return cos(x)-x
def f1(x):
    return -sin(x)-1
def newton(a,tol):
    x1=a
    x=x1-f(x1)/f1(x1)
    while(abs(x-x1)>tol):
        x1=x
        x=x1-f(x1)/f1(x1)
    return x
a=1.0
tol=1.0e-8
x=newton(a,tol)
print("x = " ,x)
```

Euler yöntemi

Euler yöntemi Taylor-serisinin sadece birinci dereceden terimini kullanan bir yöntemdir.



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

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

yeni değer = eski değer + eğim * adım

Runge-Kutta Yöntemi

Sayısal analizde Runge-Kutta yöntemi, adi diferansiyel denklemlerin çözüm yaklaşımları için bir yötemdir.

Aşağıdaki gibi tanımlanan bir başlangıç değer problemini ele alalım.

$y^{l}=dy/dx=f(x,y)$

$y(x_0) = y_0$

ve bu problem için 4.mertebe Runge-Kutta yöntemi aşağıdaki denklemlerle verilir.

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h$$

$$k_1 = f(x_i, y_i)$$

$$k_2 = f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1 h)$$

$$k_3 = f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_2 h)$$

$$k_4 = f(x_i + h, y_i + k_3 h)$$

Böylece bir sonraki değeri o anki değerine aralığının büyüklüğüyle tahmini eğimin çarpımının eklenmesiyle elde edilir. Bu eğim, eğimlerin ağırlıklı ortalamasıdır.

k_1 aralığın başlangıcındaki eğimdir.

k_2 aralığın orta noktasındaki eğimdir. Bu k_2 eğimi, Euler Yöntemi kullanılarak y'nin $x_n+h/2$ noktasındaki değerinden elde edilir.

k_3 yine orta noktadaki eğimdir. Ama bu sefer y değeri k_2 eğiminden elde edilir.

k_4 aralığın sonundaki eğimdir ve y değeri k_3 eğimi kullanılarak bulunur.

RUNGE-KUTTA YÖNTEMLERİ:

$$\frac{dy}{dx}=f(x,y)\Rightarrow\frac{d^2y}{dx^2}=f'(x,y)\Rightarrow\frac{d^3y}{dx^3}=f''(x,y)\Rightarrow\frac{d^4y}{dx^4}=f'''(x,y)$$

$$y_{n+1}=y_n+\frac{h}{1!}f(x_n,y_n)+\frac{h^2}{2!}f'(x_n,y_n)+\frac{h^3}{3!}f''(x_n,y_n)+\frac{h^4}{4!}f'''(x_n,y_n)+\dots$$

TAYLOR Serisi açınımını istenilen mertebeye kadar sürdürüp kalan terimlerden ilkini hata terimi olarak kabul edelim. Bu seri açınımının ilk dört terimini kullanarak hesaba devam edelim.

$$y_{n+1}=y_n+\frac{h}{1!}f(x_n,y_n)+\frac{h^2}{2!}f'(x_n,y_n)+\frac{h^3}{3!}f''(x_n,y_n)+\underline{\frac{h^4}{4!}f'''(x_n,y_n)}$$

Buradaki beşinci terimi artık ***O(h^4)*** *olarak göreceğiz. Yukarıda yaptığımız gibi:*

$$f'(x_n,y_n)=\frac{f(x_{n+1},y_{n+1})-f(x_n,y_n)}{h}=\frac{f_{n+1}-f_n}{h}$$

$$f''(x_n,y_n)=\frac{f(x_{n+2},y_{n+2})-2f(x_{n+1},y_{n+1})+f(x_n,y_n)}{h^2}=\frac{f_{n+2}-2f_{n+1}+f_n}{h^2}$$

$$\Rightarrow y_{n+1}=y_n+hf_n+\frac{h}{2}[f_{n+1}-f_n]+\frac{h}{6}[f_{n+2}-2f_{n+1}+f_n]$$

$$\Rightarrow y_{n+1}=y_n+\frac{h}{6}[6-3+1]f_n+\frac{h}{6}[3-2]f_{n+1}+\frac{h}{6}[f_{n+2}]=y_n+\frac{h}{6}[4f_n+f_{n+1}+f_{n+2}]$$

$$\Rightarrow y_{n+1}=y_n+\frac{h}{6}\left[f_n+4(f_n+f_{n+1})/2+\underline{f_n-f_{n+1}+f_{n+2}}\right]$$

Son formülde k_1 ve k_2 büyüklükleri hemen görülüyor. Altı çizili terimler için ise aşağıdaki gibi bir k_3 seçiminin mümkün olduğu gösterilebilir. Böylece üçüncü mertebeden RUNGE-KUTTA Yöntemine erişmiş olduk.

$$k_1=hf(x_n,y_n)$$

$$k_2=hf\left(x_n+\frac{h}{2},y_n+\frac{k_1}{2}\right)$$

$$k_3=h(x_n+h,y_n-k_1+2k_2)$$

$$y_{n+1}=y_n+\frac{1}{6}(k_1+4k_2+k_3)$$

Uygulamada en çok kullanılan biçimi ile 4. Mertebeden RUNGE-KUTTA Yöntemi, Yani yukarıdaki seride h^4 katsayılı terim de kullanılarak elde edilen yöntem aşağıdaki gibi özetlenebilir. x_n noktasındaki bilinen y_n değerini kullanarak x_{n+1} noktasındaki y_{n+1} değerini bulmak için uygulanacak işlemler şöylece özetlenebilir.

$$k_1 = hf(x_n, y_n)$$
$$k_2 = hf\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right)$$
$$k_3 = hf\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right)$$
$$k_4 = hf(x_n + h, y_n + k_3)$$
$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$
$$x_{n+1} = x_n + h$$

4. Mertebeden RUNGE-KUTTA Yöntemi dört adımda yapmaktadır. Bu metodun hatası, TAYLOR Serisinde kullanmadığımız ilk terim olan beşinci mertebeden türev içeren terim yardımıyla şöyle belirlenebilir.

$$\underline{\frac{h^5}{5!}\left(\frac{dy}{dx}\right)_a^{(5)}} \approx \underline{\underline{O(h^5)}}$$

Örnek :

$$\frac{dy}{dx} = -\frac{y}{x}, x = 1 y(1) = 1 \Rightarrow y_{analitik}(x) = \frac{1}{x}, (1 \le x \le 3, h = 0.2)$$

İlk adımı hesaplayalım.

$$k_1 = hf(x_n, y_n) = 0.2\left[-\frac{1}{1}\right] = -0.2$$

$$k_2 = hf\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right) = 0.2\left[-\frac{1-0.2/2}{1+0.2/2}\right] \cong -0.164$$

$$k_3 = hf\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right) = 0.2\left[-\frac{1-0.164/2}{1+0.2/2}\right] \cong -0.167$$

$$k_4 = hf(x_n + h, y_n + k_3) = 0.2\left[-\frac{1-0.167}{1+0.2}\right] \cong -0.139$$

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \cong 1 + \frac{1}{6}[-0.2 - 2(0.164) - 2(0.167) - 0.139]$$

$$y_{n+1} \cong 1 - 0.1668 \cong 0.8332$$

$$x_{n+1} = x_n + h = 1 + 0.2 = 1.2$$

Böylece ilk adımımızı tamamlamış olduk. Şimdi $1 \le x \le 3$ için çözümü tablo haline getirebiliriz.

x	y=y(RK)	k1	k2	k3	k4		y=1/x
1	1	-0,2	-0,16364	-0,16694	-0,13884		1
1,2	0,833333	-0,13889	-0,11752	-0,11917	-0,10202		0,833333
1,4	0,714286	-0,10204	-0,08844	-0,08934	-0,07812		0,714286
1,6	0,625	-0,07813	-0,06893	-0,06947	-0,06173		0,625
1,8	0,555556	-0,06173	-0,05523	-0,05557	-0,05		0,555556
2	0,5	-0,05	-0,04524	-0,04546	-0,04132		0,5
2,2	0,454545	-0,04132	-0,03773	-0,03789	-0,03472		0,454545
2,4	0,416667	-0,03472	-0,03194	-0,03206	-0,02959		0,416667
2,6	0,384615	-0,02959	-0,02739	-0,02748	-0,02551		0,384615
2,8	0,357143	-0,02551	-0,02375	-0,02381	-0,02222		0,357143
3	0,333333						0,333333

$O(h^5) = O(0,00032)$

n inci mertebeden bir diferansiyel denklem n tane birinci mertebeden diferansiyel denklemden oluşan bir sisteme dönüştürülebilir.

$$\frac{dy_1}{dx} = f_1(x, y_1, y_2, \cdots, y_n)$$
$$\frac{dy_2}{dx} = f_2(x, y_1, y_2, \cdots, y_n)$$
$$\vdots$$
$$\frac{dy_n}{dx} = f_n(x, y_1, y_2, \cdots, y_n)$$

Bu sistemin çözümü için x in bir noktasındaki $y_1, y_2, \cdots, y_n$ değerlerinin (koşullarının) verilmesi gerekir.

$$\frac{dy}{dx} = x - y$$
$x = 0 \Rightarrow y(0) = 1$ *için python kodu:*
$(0 \le x \le 1, h = 0.2)$

```
from math import *
def rk4m(x0,y0,h, n):
    x=x0;y=y0;xd=[x0];yd=[y0]
    for i in range(n+1):
        k1 = h * f(x, y)
        k2 = h * f(x + 0.5*h, y + 0.5*k1)
        k3 = h * f(x + 0.5*h, y + 0.5*k2)
        k4 = h * f(x + h, y + k3)
        y= y + (k1 + 2*(k2 + k3) + k4)/6.0
        yd.append(y)
        x=x+h
        xd.append(x)
    return(xd,yd)
def f(x,y):
    return x-y
n=5
h=0.2
x0=0.0
y0=1.0
x,y=rk4m(x0,y0,h,n)
for i in range(n+1):
    print(x[i],y[i])
```