CENG 235 ALGORİTMALARLA SAYISAL ÇÖZÜMLEME

Prof. Dr. Tufan TURACI
tturaci@pau.edu.tr

- Pamukkale Üniversitesi
- Mühendislik Fakültesi
- Bilgisayar Mühendisliği Bölümü

- Hafta 3

3. Hafta Konular

- **Lineer Olmayan Denklemlerin Yaklaşık Çözüm Yöntemleri**

  --- Basit İterasyon Yöntemi (Sabit Nokta İterayonu)

  --- Newton-Raphson Yöntemi (Teğetler Yöntemi)

Lineer Olmayan Denklemlerin Yaklaşık Çözüm Yöntemleri

--- Basit İterasyon Yöntemi (Sabit Nokta İterayonu)

--- Newton-Raphson Yöntemi (Teğetler Yöntemi)

--- Bisection (Yarılama) Yöntemi

--- Regula-Falsi Yöntemi (Kirişler Yöntemi)

--- Sekant Yöntemi (Değişken Kesen Yöntemi)

--- Teğet-Kiriş Yöntemi

Lineer Olmayan Denklemler

Mühendisliğin birçok alanında karşılaşılan problemlerden biri lineer olmayan denklem veya denklem sistemlerdir. İki veya daha yüksek dereceli polinomlar veya trigonometrik, üstel ve logaritmik gibi lineer olmayan terimler içeren denklemler lineer olmayan denklemlere örnektir.

Genelde lineer olmayan denklemler $f(x) = 0$ kapalı formunda yazılırlar. Karşılaşılan denklemlerin çoğu tek değişkenli olmakla beraber çok değişkenli $f(x_1, x_2, x_{3,\ldots\ldots}) = 0$ denklemlerin çözümü de söz konusu olabilir.

Lineer Olmayan Denklemler

Kök bulma işlemi, verilen $f(x)$ denkleminde $f(x_k) = 0$ değerini sağlayan x_k değerlerinin bulunması işlemidir. Tek değişkenli bir fonksiyon için bu değerler aynı zamanda eğrinin x eksenini kestiği noktalardır. Kök bulma işlemlerinde öncelikle kökün hangi aralıkta olduğu belirlenir.

--- Bolzano Teoremi

--- Grafik ile aralık belirleme

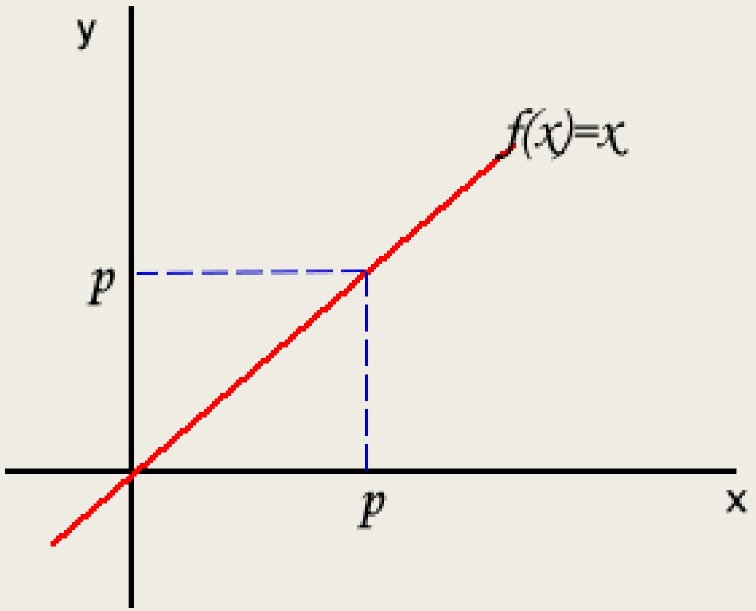
Daha sonra kök bulmaya en uygun yöntem seçilerek köke en yakın değere yakınsanır.

1- Basit İterasyon Yöntemi (Sabit Nokta İterayonu)

Tanım: Verilen bir $f(x)$ fonksiyonu ve p sayısı için,

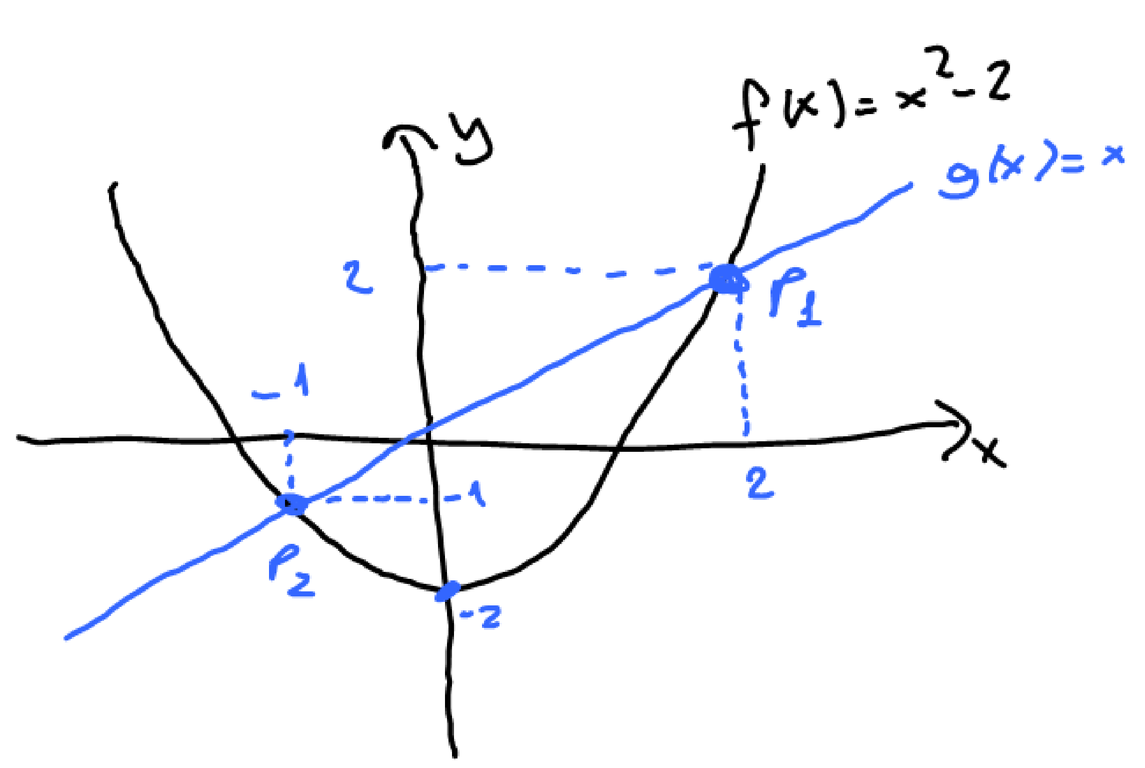
$$f(p) = p$$

oluyorsa, p noktası sabit bir noktadır.



Örnek: $f(x) = x^2 - 2$ için sabit nokta incelemesi yapınız.

Grafik:



Çözüm:

P_1 ve P_2 noktaları sabit noktalardır.

Bu noktaları bulmak için;

$$g(x) = f(x)$$

$$x = x^2 - 2$$

$$x^2 - x - 2 = 0$$

$$x = 2 \text{ ve } x = -1$$

sabit noktalardır.

Teorem:

(i) Eğer $f(x) \in C[a,b]$ ve her $x \in [a,b]$ için $f(x) \in [a,b]$ oluyorsa $f(x)$'in [a,b]'de en azından bir sabit noktası vardır.

(ii) her $x \in [a,b]$ için $f'(x)$ mevcut ve $f'(x) \in (a,b)$ oluyorsa, $|f'(x)| \leq k$ (k<1 olmak üzere) ise [a,b]'de ***tek bir sabit nokta*** vardır.

Örnek:

$[-2,+2]$ tanım aralığında $f(x)=x^2-2$ inceleyelim.

$$f'(x)=2x$$

$$|f'(x)|=|2x|=2.|x|\leq 4$$

$k=4>1$ old. dan $f(x)$ in tek bir sabit noktası yoktur.

Örnek:

$[-1, +1]$ tanım aralığındaki $f(x) = \frac{x^2-1}{3}$

fonk.nun sabit nokta incelemesini yapınız.

$$|f'(x)| = \left| \frac{2x}{3} \right| = \frac{2}{3}|x| \leq \frac{2}{3}$$

$k = \frac{2}{3} < 1$ old. dan $g(x)$ tek bir

sabit noktaya sahiptir.

$$x = \frac{x^2-1}{3}$$

$$x^2 - 3x - 1 = 0$$

$$x_{1,2} = \frac{3 \mp \sqrt{13}}{2}$$

Sabit nokta $x = \frac{3-\sqrt{13}}{2}$ 'dir.

Sabit Nokta Teoremi:

$f(x) \in C[a,b]$ ve her $x \in [a,b]$ için ;

$f(x) \in [a,b]$ ve $f'(x)$ mevcut ve $f'(x) \in (a,b)$ olsun.

Her $x \in (a,b)$ için $0 < k < 1$ olmak üzere $|f'(x)| \leq k$ olsun.

Bu durumda [a,b] aralığındaki herhangi bir x_0 başlangıç noktası için

$$x_n = f(x_{n-1}), \qquad n \geq 1$$

dizisi tek bir sabit noktaya yakınsar.

Sabit Nokta İterasyonu:

Adım 1: Verilen $f(x) = 0$ fonksiyonu $x = F(x)$ formunda yazılır.

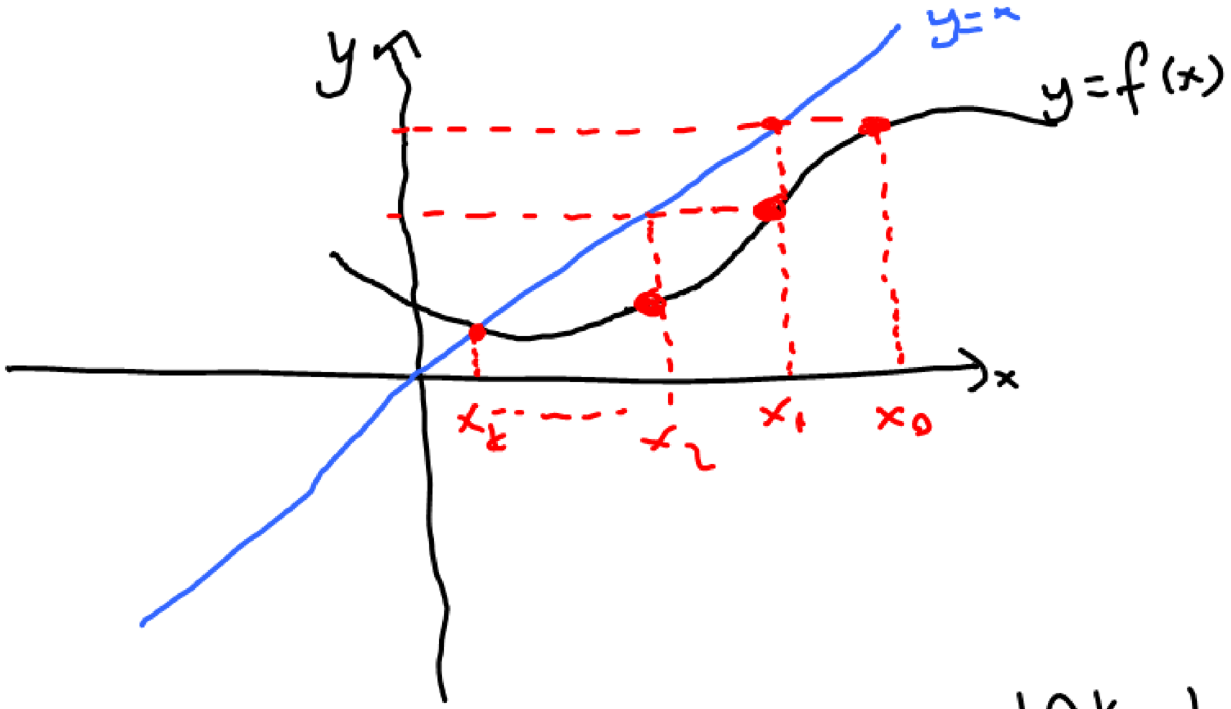
Adım 2: İterasyon başlangıcı için tahimini bir x_0 başlangıç değeri alınır ve yakınsama şartı sağlanırsa x_0 başlangıç değeri $F(x)$'de yerine yazılarak x_1, daha sonra x_1 noktası $F(x)$' de tekrar yazılarak x_2 bulunur.

Bu işlem n defa tekrarlandığında n. iterasyon için genel denklem

$$x_{n+1} = F(x_n) \text{ olur.}$$

Adım 3: İterasyona $|x_{n+1} - x_n| < \varepsilon(Hata\ Tolerans\ Değeri)$ oluncaya kadar devam edilir. Bu şart sağlanıyorsa aranan kök x_{n+1}'dir.

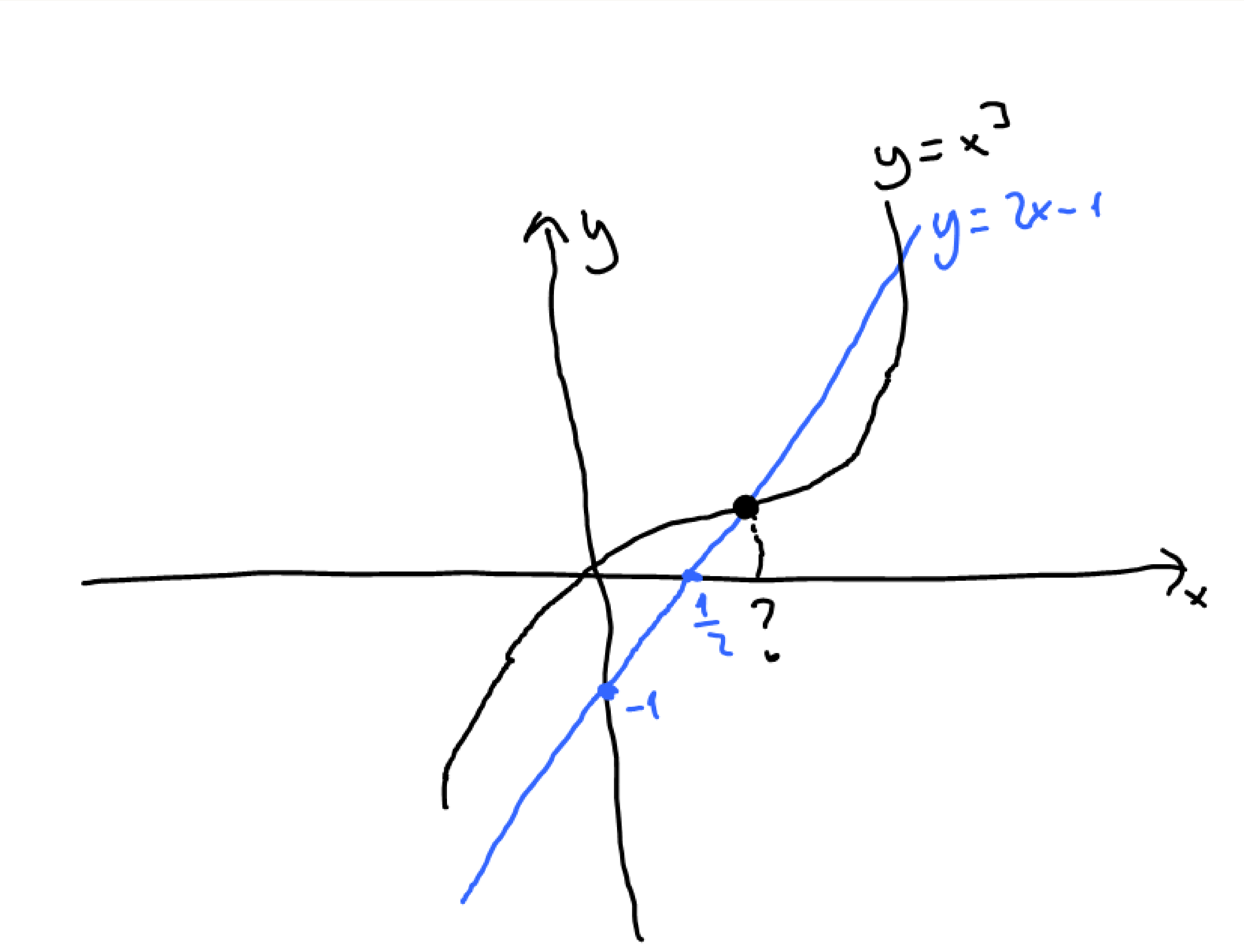
Başlangıç Noktasının Yakınsama Şartı:



... x_0 başlangıç noktası civarında $|f'(x)|<1$ olursa köke yakınsama olur.

... $i \to \infty$ iken $|x_{i+1} - x_i| \to 0$ ise (i+1). iterasyonda hata i. iterasyondaki hataya bağlıdır.

Örnek! $x^3 - 2x + 1 = 0$ denkleminin kökünün bulunduğu aralığı grafik yöntemi ile belirleyerek kökü $\varepsilon = 0.002$ hata ile basit iterasyon yöntemi ile bulunuz.
(4 ondalık kullanınız.)



y = x^3
y = 2x - 1
y
x
1/2
?
-1

$x_0 = \frac{1}{2}$ başlangıç noktası ile iterasyona başlanabilir.

$$f(x) = x^3 - 2x + 1 = 0$$

$$\underbrace{\frac{x^3+1}{2}}_{F(x)} = x$$

$$F'(x) = \frac{3}{2}x^2 = \frac{3}{2}\left|x^2\right| = \frac{3}{2}\left|\left(\frac{1}{2}\right)^2\right|$$

$$= \frac{3}{2}\cdot\frac{1}{4} = \boxed{\frac{3}{8} < 1}$$

$x_0 = 0.5$ başlangıç noktası olur.

$x_0 = 0.5$

$x_1 = 0.5625$

$x_2 = 0.5890$

$x_3 = 0.6022$

$x_4 = 0.6092$

$x_5 = 0.6130$

$x_6 = 0.6152$

$x_7 = 0.6164$

$\Delta x_1 = |x_1 - x_0| = 0.0625 > \varepsilon$

$\Delta x_2 = |x_2 - x_1| = 0.0265 > \varepsilon$

$\Delta x_3 = |x_3 - x_2| = 0.0132 > \varepsilon$

$\Delta x_4 = |x_4 - x_3| = 0.0070 > \varepsilon$

$\Delta x_5 = |x_5 - x_4| = 0.0038 > \varepsilon$

$\Delta x_6 = |x_6 - x_5| = 0.0022 > \varepsilon$

$\Delta x_7 = |x_7 - x_6| = \boxed{0.0012 < \varepsilon}$

$\varepsilon = 0.002$ hata ile yaklaşık kök

$x_7 = 0.6164$

C kodu:

```c
#include<stdio.h>
#include<conio.h>
#include<locale.h>
#include<math.h>
#define hata 0.002
float F(float x)
{return (pow(x,3)+1)/2;}

int main()
{setlocale(LC_ALL, "Turkish");
float x1=0.5,x2; int i=0;
printf("Yönteme başladığımız nokta= %.4f\n",x1);
do
{x2=x1;
x1=F(x1);
i++;
printf("%d. adımda yaklaşık değer= %.4f\n",i,x1);
}while (fabs(x1-x2)>hata);
printf("yaklaşık kök =%.4f",x1);
printf("f(%.4f)=%.4f\n",x1,pow(x1,3)-2*x1+1);
getch ();
return 0;
}
```

Ekran Çıktısı:

```
Yönteme başladığımız nokta= 0,5000
1. adımda yaklaşık değer= 0,5625
2. adımda yaklaşık değer= 0,5890
3. adımda yaklaşık değer= 0,6022
4. adımda yaklaşık değer= 0,6092
5. adımda yaklaşık değer= 0,6130
6. adımda yaklaşık değer= 0,6152
7. adımda yaklaşık değer= 0,6164
yaklaşık kök =0,6164
f(0,6164)=0,0014


--------------------------------

Process exited with return value 0
Press any key to continue . . .
```

Örnek! $e^x - 3x = 0$ denkleminin kökünü $[0,1]$ aralığında $\varepsilon = 10^{-5}$ hata ile bulunuz. (6 ondalık kullanınız.)

$$e^x - 3x = 0$$

$$x = \underbrace{\frac{e^x}{3}}_{f(x)}$$

$$f'(x) = \frac{e^x}{3}$$

$$f'(0) = \frac{1}{3} < 1$$

$$f'(1) = \frac{e}{3} < 1$$

Yakınsama şartı sağlandı.

$x_0 = 0$ başlangıç değeri olsun.

$x_1 = f(x_0) = f(0) = 0.333333$

$x_2 = f(x_1) = f(0.333333) = 0.465204$

$\vdots$

$x_{20} = f(x_{19}) = 0.619035$

$x_{21} = f(x_{20}) = 0.619047$

$\Delta x_{21} = 0.000008 < 10^{-5}$

Yaklaşık kök $= x_{21} = 0.619047$

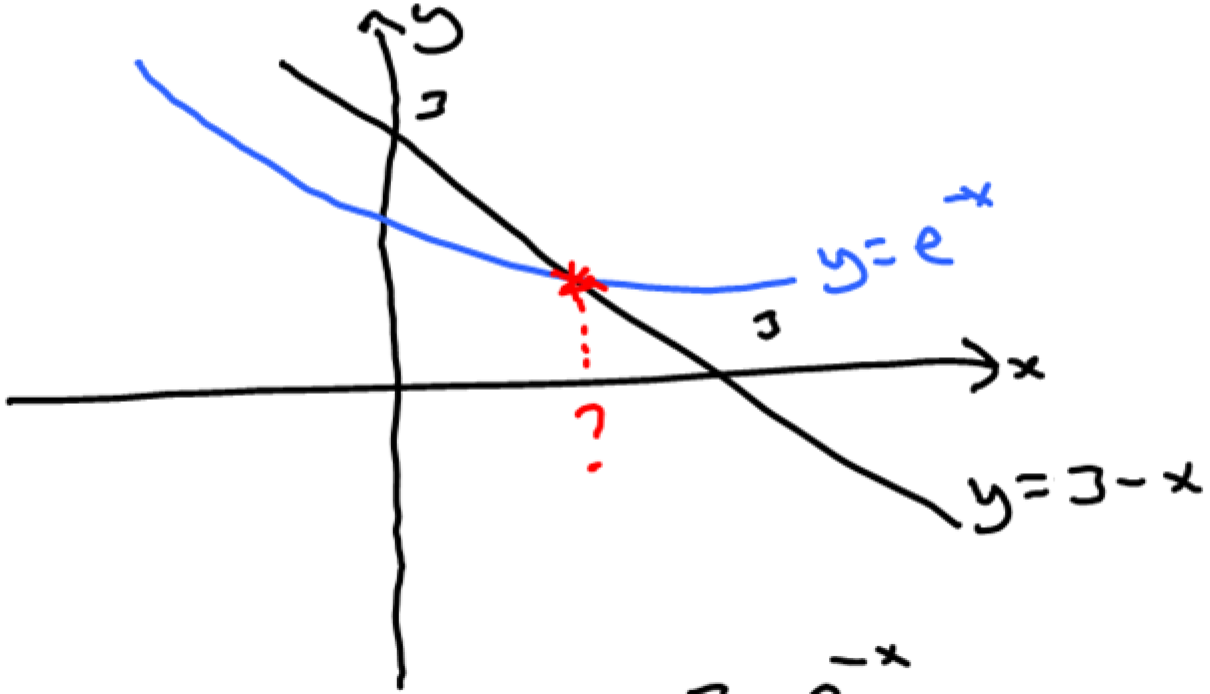
C kodu:

```c
#include<stdio.h>
#include<conio.h>
#include<locale.h>
#include<math.h>
#define hata 0.00001
float F(float x)
{return exp(x)/3;}

int main()
{setlocale(LC_ALL, "Turkish");
float x1=0,x2; int i=0;
printf("Yönteme başladığımız nokta= %.6f\n",x1);
do
{x2=x1;
x1=F(x1);
i++;
printf("%d. adımda yaklaşık değer= %.6f\n",i,x1);
}while (fabs(x1-x2)>hata);
printf("yaklaşık kök =%.6f\n",x1);
printf("f(%.6f)=%.6f\n",x1,exp(x1)-3*x1);
getch ();
return 0;
}
```

Ekran Çıktısı:

```
Yönteme başladığımız nokta= 0,000000
1. adımda yaklaşık değer= 0,333333
2. adımda yaklaşık değer= 0,465204
3. adımda yaklaşık değer= 0,530780
4. adımda yaklaşık değer= 0,566752
5. adımda yaklaşık değer= 0,587511
6. adımda yaklaşık değer= 0,599835
7. adımda yaklaşık değer= 0,607273
8. adımda yaklaşık değer= 0,611806
9. adımda yaklaşık değer= 0,614586
10. adımda yaklaşık değer= 0,616297
11. adımda yaklaşık değer= 0,617352
12. adımda yaklaşık değer= 0,618004
13. adımda yaklaşık değer= 0,618407
14. adımda yaklaşık değer= 0,618657
15. adımda yaklaşık değer= 0,618811
16. adımda yaklaşık değer= 0,618906
17. adımda yaklaşık değer= 0,618965
18. adımda yaklaşık değer= 0,619002
19. adımda yaklaşık değer= 0,619025
20. adımda yaklaşık değer= 0,619039
21. adımda yaklaşık değer= 0,619047
yaklaşık kök =0,619047
f(0,619047)=0,000016

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

Örnek! $3 - x - e^{-x} = 0$ denkleminin küçük kökünün bulunduğu aralığı grafik çizerek belirleyip basit iterasyon yöntemi ile $\varepsilon = 10^{-5}$ hata ile bulunuz. (6 ondalık kullanınız.)



$$x = \underbrace{3 - e^{-x}}_{F(x)}$$

$$F'(x) = e^{-x}$$

$$F'(3) = e^{-3} = \frac{1}{e^{3}} < 1$$

$x_0 = 3$ başlangıç noktası olur.

$x_0 = 3$

$x_1 = F(3) = 2.950213$

$x_2 = F(x_1) = 2.947671$

$x_3 = F(x_2) = 2.947538$

$x_4 = F(x_3) = 2.947531$

$|x_4 - x_3| = 0.000007 < 10^{-5}$

Yaklaşık kök $= 2.947531$

C kodu:

```c
#include<stdio.h>
#include<conio.h>
#include<locale.h>
#include<math.h>
#define hata 0.00001
float F(float x)
{return 3-exp(-x);}

int main()
{setlocale(LC_ALL, "Turkish");
float x1=3,x2; int i=0;
printf("Yönteme başladığımız nokta= %.6f\n",x1);
do
{x2=x1;
x1=F(x1);
i++;
printf("%d. adımda yaklaşık değer= %.6f\n",i,x1);
}while (fabs(x1-x2)>hata);
printf("yaklaşık kök =%.6f\n",x1);
printf("f(%.6f)=%.6f\n",x1,3-exp(-x1)-x1);
getch ();
return 0;
}
```

Ekran Çıktısı:

```
Yönteme başladığımız nokta= 3,000000
1. adımda yaklaşık değer= 2,950213
2. adımda yaklaşık değer= 2,947671
3. adımda yaklaşık değer= 2,947538
4. adımda yaklaşık değer= 2,947531
yaklaşık kök =2,947531
f(2,947531)=-0,000000


--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

Çalışma Sorusu: $f(x) = \cos x - x$ fonksiyonunun bir kökünü basit iterasyon yöntemini kullanarak $x_0 = \frac{\pi}{4}$ başlangıç değeri için 7 iterasyon ile hesaplayınız. (6 ondalık kullanınız.)

Yanıt: $x_7 = 0.736128$

Çalışma Sorusu: $f(x) = x^3 - 3x - 20$ fonksiyonunun $[1,4]$ aralığındaki kökünü basit iterasyon yöntemiyle $\varepsilon = 10^{-5}$ hata ile 6 ondalık kullanarak bulunuz.

Yanıt: 3.080859

2- Newton-Raphson Yöntemi (Teğetler Yöntemi)

$f(x) \in C^2[a,b]$ iken $f(x) = 0$ denkleminin [a,b] aralığındaki köküne yaklaşmak için $f(x)$'in $x_0 \in [a,b]$ civarındaki Taylor seri açılımını kullanıldığında:

$$f(x) \cong f(x_0) + f'(x_0)(x - x_0) + f''(x_0)\frac{(x - x_0)^2}{2!} + \dots\dots\dots$$

3. terim ve daha sonrası kalan olarak ele alınır ve ihmal edilsin.

$f(x) = 0$ olduğunda:

$$f(x) \cong f(x_0) + f'(x_0)(x - x_0) = 0$$

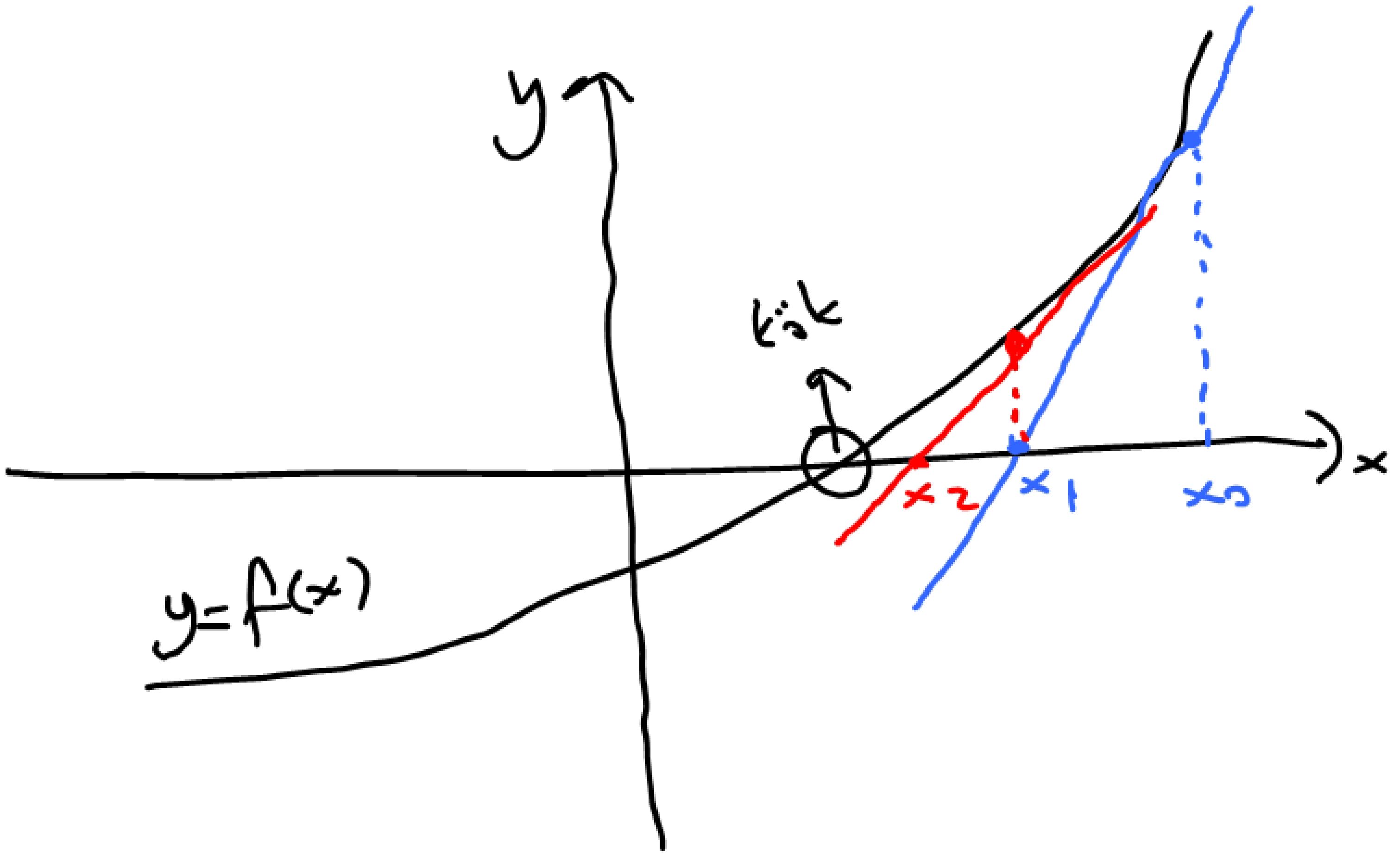
$$x \cong x_0 - \frac{f(x_0)}{f'(x_0)} \qquad f'(x_0) \neq 0$$

Burada bir $\{X_n\}_{n=0}^{\infty}$ dizisi oluşturulabilir:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}; \qquad n = 0,1,2,3,\ldots.$$

Teorem: $f(x) \in C^2[a,b]$ iken, $x \in [a,b]$ olmak üzere $f(x) = 0$ ve $f'(x) \neq 0$ ise öyle bir $\delta > 0$ sayısı vardır ki $x_0 \in [x-\delta, x+\delta]$ başlangıç noktasıyla $\{X_n\}_{n=0}^{\infty}$ dizisi Newton yöntemi ile x'ye yakınsar.

Newton-Raphson Yöntemi Geometrik Yorumu



Teğetler çizerek köke yaklaşılır.

İterasyon: $$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

İterasyona başlama noktası olarak:

$$f''(x) \cdot f(x) > 0$$

şartını sağlayan nokta alınabilir.

x_0 civarında köke yakınsama şartı:

$$\left| \frac{f(x_0) \cdot f''(x_0)}{[f'(x_0)]^2} \right| < 1$$

Örnek: $e^x - 3x = 0$ denkleminin $[0,1]$ aralığındaki kökünü Newton-Raphson yöntemi ile 10^{-5} hata olacak şekilde 6 ondalıkta hesaplayınız.

$f(0) = 1 > 0$

$f(1) = e^1 - 3 = e - 3 < 0$

$f(0) \cdot f(1) < 0$ kök var.

$f'(x) = e^x - 3$

$f''(x) = e^x$

$\underbrace{f(x)}_{>0} \cdot \underbrace{f''(x)}_{>0} > 0$

$x_0 = 0$ için $\quad x_0 = 0$ için

Başlangıç noktası $x_0 = 0$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

$$x_0 = 0$$

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 0 - \frac{f(0)}{f'(0)} = 0.5$$

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = 0.610060$$

$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)} = 0.618597$$

$$x_4 = x_3 - \frac{f(x_3)}{f'(x_3)} = 0.619061$$

$$x_5 = x_4 - \frac{f(x_4)}{f'(x_4)} = 0.619061$$

$$|x_5 - x_4| = 0$$

Böylece yaklaşık kök:

$$x_5 = 0.619061$$

C kodu:

```c
#include<stdio.h>
#include<conio.h>
#include<locale.h>
#include<math.h>
#define hata 0.00001
float F(float x)
{return exp(x)-3*x;}

float FT(float x)
{return exp(x)-3;}

int main()
{setlocale(LC_ALL, "Turkish");
float x0=0,x; int i=0;
printf("Yönteme başladığımız nokta= %.6f\n",x0);
do
{x=x0;
x0=x-F(x)/FT(x);
i++;
printf("%d. adımda yaklaşık değer= %.6f\n",i,x0);
}while (fabs(x0-x)>hata);
printf("yaklaşık kök =%.6f\n",x0);
printf("f(%.6f)=%.6f\n",x0,F(x0));
getch ();
return 0;
}
```

Ekran Çıktısı:

```
Yönteme başladığımız nokta= 0,000000
1. adımda yaklaşık değer= 0,500000
2. adımda yaklaşık değer= 0,610060
3. adımda yaklaşık değer= 0,618997
4. adımda yaklaşık değer= 0,619061
5. adımda yaklaşık değer= 0,619061
yaklaşık kök =0,619061
f(0,619061)=0,000000


--------------------------------

Process exited with return value 0
Press any key to continue . . .
```

Örnek: $x+\ln x-5=0$ denkleminin $[3.2, 4]$ aralığındaki kökünü Newton-Raphson yöntemi ile 10^{-3} hata olacak şekilde 4 ondalıklı hesaplayınız.

$$f(3.2) = -0.6368 < 0$$

$$f(4) = 0.3863 > 0$$

$$f'(x) = 1 + \frac{1}{x}$$

$$\boxed{f''(x) = \frac{-1}{x^2} < 0}$$

$f(x) \cdot f''(x) > 0$

$x_0 = 3.2$ için sağlanır.

Başlangıç Noktası

iterasyon:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

$$x_0 = 3.2$$

$$x_1 = 3.2 - \frac{f(3.2)}{f'(3.2)} = 3.6852$$

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = 3.6934$$

$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)} = 3.6934$$

$$|x_3 - x_2| = 0$$

$\varepsilon = 10^{-3}$ hata ile yaklaşık kök $= 3.6934$

C kodu:

```c
#include<stdio.h>
#include<conio.h>
#include<locale.h>
#include<math.h>
#define hata 0.001
float F(float x)
{return x+log(x)-5;}

float FT(float x)
{return 1+1/x;}

int main()
{setlocale(LC_ALL, "Turkish");
float x0=3.2,x; int i=0;
printf("Yönteme başladığımız nokta= %.4f\n",x0);
do
{x=x0;
x0=x-F(x)/FT(x);
i++;
printf("%d. adımda yaklaşık değer= %.4f\n",i,x0);
}while (fabs(x0-x)>hata);
printf("yaklaşık kök =%.4f\n",x0);
printf("f(%.4f)=%.4f\n",x0,F(x0));
getch ();
return 0;
}
```

Ekran Çıktısı:

```
Yönteme başladığımız nokta= 3,2000
1. adımda yaklaşık değer= 3,6852
2. adımda yaklaşık değer= 3,6934
3. adımda yaklaşık değer= 3,6934
yaklaşık kök =3,6934
f(3,6934)=0,0000


--------------------------------

Process exited with return value 0
Press any key to continue . . .
```

Örnek: $\cos x - x = 0$ denkleminin bir kökünü $x_0 = \frac{\pi}{4}$ başlangıç noktasından başlayarak Newton-Raphson yöntemi ile $\varepsilon = 10^{-5}$ hata ve 6 ondalık ile hesaplayınız.

$$f(x) = \cos x - x$$
$$f'(x) = -\sin x - 1$$
$$f''(x) = -\cos x$$

$$x_0 = \frac{\pi}{4}$$

$$\left| \frac{f(\frac{\pi}{4}) \cdot f''(\frac{\pi}{4})}{[f'(\frac{\pi}{4})]^2} \right| \overset{?}{<} 1$$

$$\left| \frac{\left(\frac{\sqrt{2}}{2} - \frac{\pi}{4}\right) \cdot \left(-\frac{\sqrt{2}}{2}\right)}{\left(-\frac{\sqrt{2}}{2} - 1\right)^2} \right| \overset{?}{<} 1$$

$$0.018997 < 1$$

$x_0 = \frac{\pi}{4}$ başlangıç noktası olabilir.

İterasyon:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 0.785398$$

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = 0.739536$$

$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)} = 0.739085$$

$$x_4 = x_3 - \frac{f(x_3)}{f'(x_3)} = 0.739085$$

$|x_4 - x_3| = 0$ old.dan

$\varepsilon = 10^{-5}$ hata ile yaklaşık kök:

$x_4 = 0.739085$

C kodu:

```
#include<stdio.h>
#include<conio.h>
#include<locale.h>
#include<math.h>
#define hata 0.00001

float F(float x)
{return cos(x)-x;}

float FT(float x)
{return (-1)*sin(x)-1;}
```

```
int main()
{setlocale(LC_ALL, "Turkish");
float x0=M_PI/4,x; int i=0;
printf("Yönteme başladığımız nokta= %.6f\n",x0);
do
{x=x0;
x0=x-(F(x)/FT(x));
i++;
printf("%d. adımda yaklaşık değer= %.6f\n",i,x0);
}while (fabs(x0-x)>hata);
printf("yaklaşık kök =%.6f\n",x0);
printf("f(%.6f)=%.6f\n",x0,F(x0));
getch ();
return 0;
}
```

Ekran Çıktısı:

```
Yönteme başladığımız nokta= 0,785398
1. adımda yaklaşık değer= 0,739536
2. adımda yaklaşık değer= 0,739085
3. adımda yaklaşık değer= 0,739085
yaklaşık kök =0,739085
f(0,739085)=-0,000000


--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

Çözme Sorusu: $x^3+7x^2-5x-20=0$ denkleminin kökünü Newton-Raphson yöntemi ile $x_0=8$ başlangıç noktası ile 10^{-6} hata ile 7 ondalık ile hesaplayınız.

Yanıt: 1.8162012

Çalışma Sorusu: $\sqrt{28}$ sayısının yaklaşık değerini $\varepsilon = 10^{-5}$ hata ile 6 ondalık kullanarak Newton-Raphson yöntemi ile hesaplayınız.

Yanıt: 5.291502

Kaynaklar

- Numerical Analysis, Richard L. Burden, Brooks/Cole Cengage Learning, Boston., 2009.
- Numerical Methods for Mathematics, Science, and Engineering, 2nd Edition, John H. Mathews, Prentice Hall International Edition, 1992.
- Nümerik Analiz, (Numerical Analysis, D. Kincaid, W. Cheney, 3rd ed.(2002)), Nuri Özalp, Elif Demirci, Gazi Kitabevi Yayınları, 2012.
- Sayısal Analiz ve Mühendislik Uygulamaları, İrfan Karagöz, Nobel Yyıncılık, 2011.
- Sayısal Çözümleme, Recep Tapramaz, Literatür yayıncılık, 2002.
- Bilgisayar Uygulamalı Sayısal Analiz Yöntemleri, Eyüp Sabri Türker, Engin Can, II. Baskı, Değişim Yayınları.