Numerikus Módszerek

10. Gyakorlat

Nemlineáris egyenletek numerikus megoldása

Kétféle módszert alkalmazunk:

Kétféle módszert alkalmazunk:

1. Zérushelykeresés $F(x) = 0$,

Kétféle módszert alkalmazunk:

1. Zérushelykeresés $F(x) = 0$,
2. Fixpontiterációs $G(x) = x$.

Kétféle módszert alkalmazunk:

1. Zérushelykeresés $F(x) = 0$,
2. Fixpontiterációs $G(x) = x$.

Egy f függvényt vizsgálunk. A problémát természetétől függően visszavezetjük zéruskeresésre, vagy fixpontiterációra. Az F vagy G függvényeket az f függvényből saját magunknak kell megkonstruálnunk. Persze erre nézve vannak régi, jól bevált receptek.

Feladatok

Feladatok

A fenti módszerek valamelyikével oldjuk meg az alábbi egyenleteket:

1. $f(x) = 3x^3 - 12x + 4$ zérushelyének a megkeresése ($x \in [4, 5]$). Javasolt módszer az intervallumfelezés, fixpontiteráció $g(x) = \frac{3x^3+4}{12}$.
2. $f(x) = x^3 - 100$ zérushelyének a megkeresése ($x \in [4, 5]$). Javasolt módszer a Newton-Raphson.
3. $\ln(x) = \frac{1}{x}$ egyenlet megoldása, tehát $f(x) = \ln(x) - \frac{1}{x} = 0$, azaz zérushelykeresés ($x \in [1, 2]$). Javasolt módszer a Newton-Raphson.
4. $g(x) = \frac{1}{2}\cos(x - 1)$ fixpontjának a meghatározása.
5. Hová kössük a kecskét?

Zérushelykeresés

Zérushelykeresés

1. Intervallumfelezés

Zérushelykeresés

1. Intervallumfelezés
2. Húrmódszer

Zérushelykeresés

1. Intervallumfelezés
2. Húrmódszer
3. Newton módszer

Zérushelykeresés

1. Intervallumfelezés
2. Húrmódszer
3. Newton módszer
4. Szelő módszer

A Matlab függvény megírásának a módja

A Matlab függvény megírásának a módja

Két lehetőségünk van:

A Matlab függvény megírásának a módja

Két lehetőségünk van:

1. Univerzális függvény írunk, amelynek paraméterei között függvénynév szerepel.

A Matlab függvény megírásának a módja

Két lehetőségünk van:

1. Univerzális függvény írunk, amelynek paraméterei között függvénynév szerepel.
2. Olyan függvényt írunk, amely célirányosan az adott feladatot oldja meg.

Az *EVAL* Matlab függvény használata

Az *EVAL* Matlab függvény használata

```
1    function y=Kiszamol1(f)
2 -    a=3;
3 -    b=5;
4 -    y=eval(f);
5 -  end
```

Az *EVAL* Matlab függvény használata

```
1   function y=Kiszamol1(f)
2 -   a=3;
3 -   b=5;
4 -   y=eval(f);
5 -   end
```

Hívás: y=Kiszamol1('a^2 + 1')

Eredmény: $y = 10$

Az *EVAL* Matlab függvény használata

```
1   function y=Kiszamol1(f)
2 -   a=3;
3 -   b=5;
4 -   y=eval(f);
5 -   end
```

Hívás: `y=Kiszamol1('a^2 + 1')`

Eredmény: $y = 10$

Hívás: `y=Kiszamol1('b^2 + 1')`

Eredmény: $y = 26$

Az *FEVAL* Matlab függvény használata

Az *FEVAL* Matlab függvény használata

```
1     function y=Kiszamol2(f,a)
2 -     y=feval(f,a);
3 -   end
```

Az *FEVAL* Matlab függvény használata

```
1     function y=Kiszamol2(f,a)
2 -     y=feval(f,a);
3 -   end
```

Hívás:

```
f=@(x) x^2+1
a=3
y=Kiszamol2(f,a)
```

Eredmény: $y = 10$

Iterációk

Iterációk

Egy pontra támaszkodó stacionér iteráció:

Iterációk

Egy pontra támaszkodó stacionér iteráció:

$$
\begin{aligned}
x_1 &= a \\
x_k &= T(x_{k-1}) \quad k = 2, 3, \ldots
\end{aligned}
$$

Iterációk

Egy pontra támaszkodó stacionér iteráció:

$$
\begin{aligned}
x_1 &= a \\
x_k &= T(x_{k-1}) \quad k = 2, 3, \ldots
\end{aligned}
$$

Két pontra támaszkodó stacionér iteráció:

Iterációk

Egy pontra támaszkodó stacionér iteráció:

$$x_1 = a$$
$$x_k = T(x_{k-1}) \quad k = 2, 3, \ldots$$

Két pontra támaszkodó stacionér iteráció:

$$x_1 = a$$
$$x_2 = b$$
$$x_k = T(x_{k-1}, x_{k-2}) \quad k = 3, 4, \ldots$$

Feladat

Feladat

Írjunk függvényt, amely kiszámolja a Fibonacci-sorozat 100-dik elemét.

Feladat

Írjunk függvényt, amely kiszámolja a Fibonacci-sorozat 100-dik elemét.

Fibonacci-sorozat:

$$
\begin{aligned}
x_1 &= 1 \\
x_2 &= 1 \\
x_k &= x_{k-1} + x_{k-2} \quad (k = 3, 4, \dots)
\end{aligned}
$$

Feladat

Írjunk függvényt, amely kiszámolja a Fibonacci-sorozat 100-dik elemét.

Fibonacci-sorozat:

$$
\begin{aligned}
x_1 &= 1 \\
x_2 &= 1 \\
x_k &= x_{k-1} + x_{k-2} \quad (k = 3, 4, \dots)
\end{aligned}
$$

Leonardo Pisano (1170-1250)

```
 1      function y=Fibo1(n)
 2 -      a=1;
 3 -      b=1;
 4 -      if (n==1 || n==2)
 5 -          y=1;
 6 -          return;
 7 -      end
 8 -      for k=3:1:n
 9 -          y=a+b;
10 -          a=b;
11 -          b=y;
12 -      end
13 -  end
```

Feladat

Feladat

Írjunk függvényt, amely kiírja a képernyőre a Fibonacci-sorozat első n elemét.

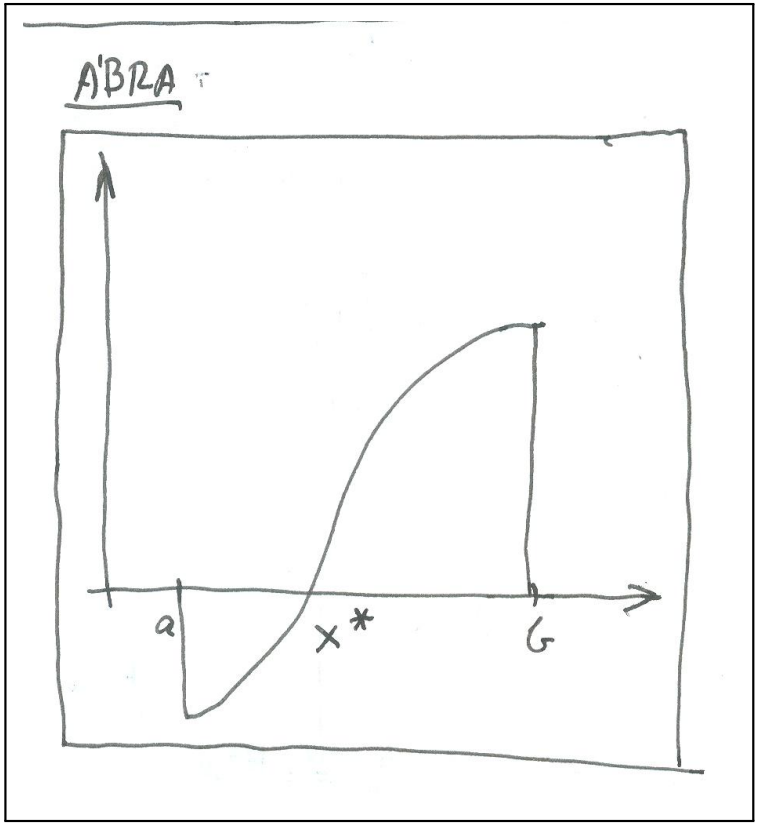
Feladat

Írjunk függvényt, amely kiírja a képernyőre a Fibonacci-sorozat első n elemét.

```
1      function y=Fibo2(n)
2 -      y=ones(1,n);
3 -      for i=3:1:n
4 -          y(1,i)=y(1,i-1)+y(1,i-2);
5 -      end
6 -    end
```

1. Intervallumfelezés

1. Intervallumfelezés



Matematikai alapok:

Matematikai alapok:

Bolzano-Darboux tétel: *Ha az* $f : [a, b] \to \mathbb{R}$ *folytonos függvény és az* $y \in \mathbb{R}$ *olyanok, hogy*

$$f(a) < y < f(b) \qquad \textit{vagy} \qquad f(b) < y < f(a)$$

akkor létezik olyan $x \in [a, b]$, *amelyre* $f(x) = y$.

Matematikai alapok:

Bolzano-Darboux tétel: *Ha az* $f : [a, b] \to \mathbb{R}$ *folytonos függvény és az* $y \in \mathbb{R}$ *olyanok, hogy*

$$f(a) < y < f(b) \qquad \textit{vagy} \qquad f(b) < y < f(a)$$

akkor létezik olyan $x \in [a, b]$*, amelyre* $f(x) = y$*.*

Egy $f : [a, b] \to \mathbb{R}$ *függvény zérushelyét keressük. A zérushelyet* x^* *módon jelöljük, azaz* $f(x^*) = 0$*.*

Az algoritmus

Az algoritmus

Az elindulás feltétele, hogy az f függvény az intervallum végpontjaiban ellentétes előjelű értékeket vegyen fel, azaz $f(a)f(b) < 0$.

Az algoritmus

Az elindulás feltétele, hogy az f függvény az intervallum végpontjaiban ellentétes előjelű értékeket vegyen fel, azaz $f(a)f(b) < 0$.

Az algoritmus $[a_k, b_k]$ intervallumok sorozatát állítja elő, úgy hogy

Az algoritmus

Az elindulás feltétele, hogy az f függvény az intervallum végpontjaiban ellentétes előjelű értékeket vegyen fel, azaz $f(a)f(b) < 0$.

Az algoritmus $[a_k, b_k]$ intervallumok sorozatát állítja elő, úgy hogy

- $[a_1, b_1] = [a, b]$.

Az algoritmus

Az elindulás feltétele, hogy az f függvény az intervallum végpontjaiban ellentétes előjelű értékeket vegyen fel, azaz $f(a)f(b) < 0$.

Az algoritmus $[a_k, b_k]$ intervallumok sorozatát állítja elő, úgy hogy

- $[a_1, b_1] = [a, b]$.
- Az $[a_k, b_k]$ intervallumot az $[a_{k-1}, b_{k-1}]$ intervallumból úgy állítjuk elő, hogy azt az

$$x_{k-1} = \frac{a_{k-1} + b_{k-1}}{2}$$

felezőponttal két részintervallumra bontjuk, ezek egyike lesz az $[a_k, b_k]$ intervallum.

Az algoritmus

Az elindulás feltétele, hogy az f függvény az intervallum végpontjaiban ellentétes előjelű értékeket vegyen fel, azaz $f(a)f(b) < 0$.

Az algoritmus $[a_k, b_k]$ intervallumok sorozatát állítja elő, úgy hogy

- $[a_1, b_1] = [a, b]$.
- Az $[a_k, b_k]$ intervallumot az $[a_{k-1}, b_{k-1}]$ intervallumból úgy állítjuk elő, hogy azt az

$$x_{k-1} = \frac{a_{k-1} + b_{k-1}}{2}$$

felezőponttal két részintervallumra bontjuk, ezek egyike lesz az $[a_k, b_k]$ intervallum.
- Minden $[a_k, b_k]$ intervallum megőrzi az $[a_1, b_1]$ intervallumnak azt a tulajdonságát, hogy az f függvény az intervallum végpontjaiban ellentétes előjelű értéket vesz fel, azaz $f(a_k)f(b_k) < 0$.

A k-adik lépés:

$$x_{k-1} := \frac{a_{k-1} + b_{k-1}}{2}.$$

A k-adik lépés:

$$x_{k-1} := \frac{a_{k-1} + b_{k-1}}{2}.$$

3 eset van:

A k-adik lépés:

$$x_{k-1} := \frac{a_{k-1} + b_{k-1}}{2}.$$

3 eset van:

- $f(x_{k-1}) = 0$, akkor megtaláltuk a zérushelyet.

A k-adik lépés:

$$x_{k-1} := \frac{a_{k-1} + b_{k-1}}{2}.$$

3 eset van:

- $f(x_{k-1}) = 0$, akkor megtaláltuk a zérushelyet.
- $f(a_{k-1})f(x_{k-1}) < 0$, akkor

$$a_k := a_{k-1}$$
$$b_k := x_{k-1}$$

A k-adik lépés:

$$x_{k-1} := \frac{a_{k-1} + b_{k-1}}{2}.$$

3 eset van:

- $f(x_{k-1}) = 0$, akkor megtaláltuk a zérushelyet.
- $f(a_{k-1})f(x_{k-1}) < 0$, akkor

$$a_k := a_{k-1}$$
$$b_k := x_{k-1}$$

- $f(x_k)f(b_k) < 0$, akkor

$$a_k := x_{k-1}$$
$$b_k := b_{k-1}$$

Leállás: Ha az x^* zérushelyet előre adott ε-nál jobban közelítjük.

Leállás: Ha az x^* zérushelyet előre adott ε-nál jobban közelítjük.
Hibabecslés: $[a_k, b_k]$ intervallumok sorozatát állítjuk elő.
$x^*, x_k \in [a_k, b_k]$, így

$$|x^* - x_k| < \frac{b-a}{2^k}.$$

(Az x^* zérushelynek az intervallum felezőpontjától való távolságát nézzük.)

2. Húrmódszer

2. Húrmódszer

Most is intervallumok egy $[a_k, b_k]$ sorozatát definiáljuk úgy, hogy

2. Húrmódszer

Most is intervallumok egy $[a_k, b_k]$ sorozatát definiáljuk úgy, hogy

- $[a_1, b_1] = [a, b]$.

2. Húrmódszer

Most is intervallumok egy $[a_k, b_k]$ sorozatát definiáljuk úgy, hogy

- $[a_1, b_1] = [a, b]$.
- Az $[a_k, b_k]$ intervallumot az $[a_{k-1}, b_{k-1}]$ intervallumból úgy állítjuk elő, hogy azt egy x_{k-1} osztóponttal két részintervallumra bontjuk, ezek egyike lesz az $[a_k, b_k]$ intervallum.

2. Húrmódszer

Most is intervallumok egy $[a_k, b_k]$ sorozatát definiáljuk úgy, hogy

- $[a_1, b_1] = [a, b]$.
- Az $[a_k, b_k]$ intervallumot az $[a_{k-1}, b_{k-1}]$ intervallumból úgy állítjuk elő, hogy azt egy x_{k-1} osztóponttal két részintervallumra bontjuk, ezek egyike lesz az $[a_k, b_k]$ intervallum.
- Minden $[a_k, b_k]$ intervallum megőrzi az $[a_1, b_1]$ intervallumnak azt a tulajdonságát, hogy az f függvény az intervallum végpontjaiban ellentétes előjelű értéket vesz fel, azaz $f(a_k)f(b_k) < 0$.

2. Húrmódszer

Most is intervallumok egy $[a_k, b_k]$ sorozatát definiáljuk úgy, hogy

- $[a_1, b_1] = [a, b]$.
- Az $[a_k, b_k]$ intervallumot az $[a_{k-1}, b_{k-1}]$ intervallumból úgy állítjuk elő, hogy azt egy x_{k-1} osztóponttal két részintervallumra bontjuk, ezek egyike lesz az $[a_k, b_k]$ intervallum.
- Minden $[a_k, b_k]$ intervallum megőrzi az $[a_1, b_1]$ intervallumnak azt a tulajdonságát, hogy az f függvény az intervallum végpontjaiban ellentétes előjelű értéket vesz fel, azaz $f(a_k)f(b_k) < 0$.

Az x_{k-1} osztópontot a

$$P_{k-1}\left(a_{k-1}, f(a_{k-1})\right), \qquad Q_{k-1}\left(b_{k-1}, f(b_{k-1})\right)$$

pontokon átmenő szelő metszi ki az x tengelyből.

A k-adik lépés: Középiskolai módszerekkel megmutatható, hogy

$$x_{k-1} = \frac{b_{k-1}f(a_{k-1}) - a_{k-1}f(b_{k-1})}{f(a_{k-1}) - f(b_{k-1})}.$$

A k-adik lépés: Középiskolai módszerekkel megmutatható, hogy

$$x_{k-1} = \frac{b_{k-1}f(a_{k-1}) - a_{k-1}f(b_{k-1})}{f(a_{k-1}) - f(b_{k-1})}.$$

Geometriai megfontolások alapján könnyű látni, hogy az x_{k-1} osztópont az $[a_{k-1}, b_{k-1}]$ intervallumot a z intervallum végpontjaiban felvett függvényértékek abszolútértékeinek az arányában osztja, ezért az osztópont ahhoz a végponthoz lesz közelebb, amelyikben a az f függvény értékének az abszolút értéke kisebb. Ezért úgy tűnik, hogy a húrmódszer jobb, mint az intervallumfelezéses módszer.

Ekkor 3 eset van

Ekkor 3 eset van

- $f(x_{k-1}) = 0$, akkor megtaláltuk a zérushelyet.

Ekkor 3 eset van

- $f(x_{k-1}) = 0$, akkor megtaláltuk a zérushelyet.
- $f(a_{k-1})f(x_{k-1}) < 0$, akkor

$$a_k := a_{k-1}$$
$$b_k := x_{k-1}$$

Ekkor 3 eset van

- $f(x_{k-1}) = 0$, akkor megtaláltuk a zérushelyet.
- $f(a_{k-1})f(x_{k-1}) < 0$, akkor

$$a_k := a_{k-1}$$
$$b_k := x_{k-1}$$

- $f(x_{k-1})f(b_{k-1}) < 0$, akkor

$$a_k := x_{k-1}$$
$$b_k := b_{k-1}$$

Ekkor 3 eset van

- $f(x_{k-1}) = 0$, akkor megtaláltuk a zérushelyet.
- $f(a_{k-1})f(x_{k-1}) < 0$, akkor

$$a_k := a_{k-1}$$
$$b_k := x_{k-1}$$

- $f(x_{k-1})f(b_{k-1}) < 0$, akkor

$$a_k := x_{k-1}$$
$$b_k := b_{k-1}$$

Leállás: Ha az $[a_k, b_k]$ intervallum hossza előre adott ε-nál kisebbé válik, vagy az iterációk száma egy előre megadott itmax értéket túllép. (Persze, ha megtaláltuk a zérushelyet, akkor is leállunk.)

3. Newton-Raphson módszer

3. Newton-Raphson módszer

Isac Newton (1642-1727)
Joseph Raphson (1648?-1715?)

3. Newton-Raphson módszer

Isac Newton (1642-1727)
Joseph Raphson (1648?-1715?)

Ötlet: *(k-adik lépés)* x_{k-1} *már elég közel van az* x^**-hoz. ekkor az* x_k*-t az* f *függvény* x_{k-1}*-beli érintője metszi ki az* x *tengelyből.*

3. Newton-Raphson módszer

Isac Newton (1642-1727)
Joseph Raphson (1648?-1715?)

Ötlet: *(k-adik lépés)* x_{k-1} *már elég közel van az* x^**-hoz. ekkor az* x_k*-t az* f *függvény* x_{k-1}*-beli érintője metszi ki az* x *tengelyből.*
Az érintő egyenlete: $y = f'(x_{k-1})(x - x_{k-1}) + f(x_{k-1})$
$y = 0$, *akkor*

$$x_k = x_{k-1} - \frac{f(x_{k-1})}{f'(x_{k-1})}.$$

3. Newton-Raphson módszer

Isac Newton (1642-1727)
Joseph Raphson (1648?-1715?)

Ötlet: *(k-adik lépés)* x_{k-1} *már elég közel van az* x^**-hoz. ekkor az* x_k*-t az* f *függvény* x_{k-1}*-beli érintője metszi ki az* x *tengelyből.*
Az érintő egyenlete: $y = f'(x_{k-1})(x - x_{k-1}) + f(x_{k-1})$
$y = 0$, *akkor*

$$x_k = x_{k-1} - \frac{f(x_{k-1})}{f'(x_{k-1})}.$$

Más módon is el lehet jutni ehhez az iterációhoz:

3. Newton-Raphson módszer

Isac Newton (1642-1727)
Joseph Raphson (1648?-1715?)

Ötlet: *(k-adik lépés)* x_{k-1} *már elég közel van az* x^**-hoz. ekkor az* x_k*-t az* f *függvény* x_{k-1}*-beli érintője metszi ki az* x *tengelyből.*
Az érintő egyenlete: $y = f'(x_{k-1})(x - x_{k-1}) + f(x_{k-1})$
$y = 0$, *akkor*

$$x_k = x_{k-1} - \frac{f(x_{k-1})}{f'(x_{k-1})}.$$

Más módon is el lehet jutni ehhez az iterációhoz:
Elsőfokú Taylor polinom:

$$0 = f(x^*) \approx f(x_{k-1}) + f'(x_{k-1})(x^* - x_{k-1})$$

$$x^* \approx x_{k-1} - \frac{f(x_{k-1})}{f'(x_{k-1})}$$

$$x_k = x_{k-1} - \frac{f(x_{k-1})}{f'(x_{k-1})}$$

Elegendő feltétel a konvergenciára

Elegendő feltétel a konvergenciára

Monoton konvergencia tétel *Legyen* $f : [a, b] \to \mathbb{R}$ *kétszer folytonosan differenciálható függvény úgy, hogy*

$$f'(x) \neq 0, \qquad f''(x) \neq 0 \qquad (x \in [a, b])$$

továbbá létezik $x^* \in]a, b[$ *úgy, hogy* $f(x^*) = 0$. *Legyen* $x_0 \in [a, b]$ *olyan, hogy* $f(x_0)f''(x_0) > 0$. *Ekkor az* x_0 *pontból indított Newton iteráció monoton konvergál* x^* *hoz.*

A konvergencia sebességéről

A konvergencia sebességéről

$$|x^* - x_k| \leqslant \frac{M}{2m} |x^* - x_{k-1}|^2 .$$

A konvergencia sebességéről

$$|x^* - x_k| \leqslant \frac{M}{2m} |x^* - x_{k-1}|^2 .$$

ahol

A konvergencia sebességéről

$$|x^* - x_k| \leqslant \frac{M}{2m} |x^* - x_{k-1}|^2 .$$

ahol

$$\begin{aligned} \left|f''(x)\right| &\leqslant M \qquad (x \in [a,b]) \\ \left|f'(x)\right| &\geqslant m \qquad (x \in [a,b]), \end{aligned}$$

A leállás feltétele

A leállás feltétele

$$|f(x_k)| < \varepsilon \qquad \text{vagy} \qquad k > \mathrm{itmax}.$$

A leállás feltétele

$$|f(x_k)| < \varepsilon \qquad \text{vagy} \qquad k > \mathrm{itmax}.$$

Érdemes megjegyezni, hogy a Newton-Raphson iteráció egy fixpont iteráció.

4. Szelő módszer

4. Szelő módszer

A szelő módszer egy két pontra támaszkodó stacionér iteráció. x_1, x_2 *kell az elinduláshoz.*

$$x_k = x_{k-1} - \frac{f(x_{k-1})(x_{k-1} - x_{k-2})}{f(x_{k-1}) - f(x_{k-2})} = \frac{f(x_{k-1})x_{k-2} - x_{k-1}f(x_{k-2})}{f(x_{k-1}) - f(x_{k-2})}$$

amit úgy kapunk, hogy a Newton-Raphson módszerben az $f'(x_k)$ *differenciálhányados helyére az*

$$f'(x_{k-1}) \approx \frac{f(x_{k-1}) - f(x_{k-2})}{x_{k-1} - x_{k-2}}$$

differenciahányadost írjuk.

Fixpontiteráció

Fixpontiteráció

Matematikai alapok:

Fixpontiteráció

Matematikai alapok:

1. **Kontrakció fogalma** $f : [a, b] \to [a, b]$ *kontrakció, ha*

$$|f(x) - f(y)| \leqslant q|x - y| \qquad (x, y \in [a, b])$$

teljesül valamilyen $q \in [0, 1]$ *szám esetén. A* q *számot a* ***kontrakciós faktor****nak nevezzük.*

2. Banach-féle fixponttétel *Ha* $f : [a, b] \to [a, b]$ *egy kontrakció* q *kontrakciós faktorral, akkor egyértelműen létezik* $x^* \in [a, b]$ *úgy, hogy* $x^* = f(x^*)$, *sőt az*

$$x_1 \in [a, b] \qquad \text{tetszőleges}$$
$$x_k = f(x_{k-1}) \qquad (k = 2, 3, \dots)$$

rekurzióval definiált (x_k) *sorozat konvergens és* $x_k \to x^*$.

3. Lagrange tétele *Ha az* $f : [a, b] \to \mathbb{R}$ *függvény*

- *folytonos az* $[a, b]$*-n*
- *differenciálható az* $]a, b[$ *pontjaiban,*

akkor létezik olyan $\xi \in]a, b[$*, amellyel*

$$f(b) - f(a) = f'(\xi)(b - a).$$

4. Elegendő feltétel kontrakcióhoz *Ha* $f : [a, b] \to [a, b]$ *olyan, hogy teljesíti a Lagrange-tétel feltételeit, továbbá létezik olyan* $q \in]0, 1[$, *amellyel*

$$\left|f'(x)\right| \leqslant q \qquad (x \in]a, b[),$$

akkor f *kontrakció* q *kontrakciós faktorral.*

5. Mértani sor összegképlete *Ha* $|q| < 1$, *akkor*

$$\sum_{k=0}^{\infty} q^k = \frac{1}{1-q}.$$

6. A posteriori hibabecslés *az előzőekből már könnyen összebarkácsolható:*

$$|x_k - x^*| \leqslant \frac{q}{1-q} |x_k - x_{k-1}|,$$

ahol az $f : [a, b] \to [a, b]$ *q-kontrakció egyértelműen létező* x^* *fixpontját keressük az*

$$\begin{aligned} x_1 &\in [a, b] \qquad \text{tetszőleges} \\ x_k &= f(x_{k-1}) \qquad (k = 2, 3, \dots) \end{aligned}$$

iterációval.

Érdemes észrevenni, hogy a $q \to \frac{q}{1-q}$ $(q \in]0,1[)$ függvény felülről nem korlátos, sőt

$$\lim_{q \to 1-0} \frac{q}{1-q} = +\infty,$$

ami ha q-t figyelmen kívül hagyjuk, erősen el tudja rontani a leállási feltételt.

Az algoritmus: *Az* $f : [a, b] \to [a, b]$ *q-kontrakció egyértelműen létező* x^* *fixpontját keressük.*

Az algoritmus: *Az* $f : [a, b] \to [a, b]$ *q-kontrakció egyértelműen létező* x^* *fixpontját keressük.*

- ***A*** q ***kontrakciós faktor***

$$\left|f'(x)\right| \leqslant q \qquad (x \in]a, b[)$$

módon számolható, ha f *differenciálható* $]a, b[$*-n és létezik ilyen* $q \in]0, 1[$ *szám.*

Az algoritmus: *Az* $f : [a, b] \to [a, b]$ *q-kontrakció egyértelműen létező* x^* *fixpontját keressük.*

- ***A*** q ***kontrakciós faktor***

$$\left|f'(x)\right| \leqslant q \qquad (x \in]a, b[)$$

módon számolható, ha f *differenciálható* $]a, b[$*-n és létezik ilyen* $q \in]0, 1[$ *szám.*

- ***Az iteráció***

$$x_1 \in [a, b] \qquad \textit{tetszőleges}$$
$$x_k = f(x_{k-1}) \qquad (k = 2, 3, \dots)$$

módon van definiálva.

Az algoritmus: *Az* $f : [a, b] \to [a, b]$ *q-kontrakció egyértelműen létező* x^* *fixpontját keressük.*

- ***A*** q ***kontrakciós faktor***

$$\left|f'(x)\right| \leqslant q \qquad (x \in]a, b[)$$

módon számolható, ha f *differenciálható* $]a, b[$*-n és létezik ilyen* $q \in]0, 1[$ *szám.*

- ***Az iteráció***

$$x_1 \in [a, b] \qquad \textit{tetszőleges}$$
$$x_k = f(x_{k-1}) \qquad (k = 2, 3, \dots)$$

módon van definiálva.

- ***A leállási feltétel****t az*

$$|x_k - x^*| \leqslant \frac{q}{1-1} |x_k - x_{k-1}|$$

egyenlőtlenség adja.

Matlab programok

```
1   function [x,it]=IntFel1(f,a,b,epsilon)
2   x=(a+b)/2;
3   hiba=(b-a)/2;
4   it=1;
5   while(hiba>epsilon)
6       fx=eval(f);
7       if fx==0
8           return;
9       end
10      u=x;
11      x=a;
12      fa=eval(f);
13      if fa*fx<0
14          b=u;
15      else
16          a=u;
17      end
18      x=(a+b)/2;
19      hiba=hiba/2;
20      it=it+1;
21  end
22  end
```

Hívás: *Például az* $f : [0,1] \to \mathbb{R}$ $f(x) = 3x^3 - 12x + 4$ *függvény esetén.*

```
a=0
b=1
f='3*x^3-12*x+4
epsilon=0.0001
[x, it] =IntFel1(f,a,b,epsilon)
```

Eredmény: $x = 0.3434$,
it=14.

```
function [x, it] = intfel(f, a, b, p)
x = a;
fa = eval(f);
x = (a+b)/2;
fx = eval(f);
pk = (b-a)/2;
it = 1;
while pk>p && fx~=0
    if fx*fa<0
        b = x;
    else
        a = x;
    end
x = a;
fa = eval(f);
x = (a+b)/2;
fx = eval(f);
pk = pk/2;
it = it+1;
end
end
```

Hívás: *Például az* $f : [0,1] \to \mathbb{R}$ $f(x) = 3x^3 - 12x + 4$ *függvény esetén.*

```
f='3*x^3-12*x+4'
a=0
b=1
p=0.0001
[x, it] =intfel(f,a,b,p)
```

Eredmény: $x = 0.3434$, it=14.

Newton-Raphson módszer

```
1     function [x,it]=NewtonRaphson(f,f1,x,epsilon)
2 -     fx=eval(f);
3 -     hiba=abs(fx);
4 -     it=1;
5 -   while(hiba>epsilon)
6 -         f1x=eval(f1);
7 -         x=x-fx/f1x;
8 -         fx=eval(f);
9 -         it=it+1;
10 -        hiba=abs(fx);
11 -    end
12 -  end
```

Hívás:

```
f='x^3-100'
f1='3*x^2'
x=5
epsilon=0.0001
[x, it] =Newton-Raphson(f,f1,x,epsilon)
```

Eredmény: $x = 4.6416$,
it=6.

(jó!)

Fixpontiteráció

Fixpontiteráció

```
1     function [x,it]=Fixpontit(g,x,q,epsilon)
2 -     u=x;
3 -     x=eval(g);
4 -     K=q/(1-q);
5 -     it=1;
6 -   while(K*abs(u-x)>epsilon)
7 -         u=x;
8 -         x=eval(g);
9 -         it=it+1;
10 -    end
11 -    end
```

Hívás:

```
g='0.5*cos(x-1)'
x=0.5
q=0.5
epsilon=0.0001
[x, it] =Fixpontit(f,x,q)
```

Eredmény: $x = 0.4175$, it=10.

(ez is jó!)

KÖSZÖNÖM A FIGYELMET!