Numerikus módszerek

Bevezetés

Most készül, kérem a türelmüket.

Miskolc, 2020. Glavosits Tamás

Tartalomjegyzék

1. fejezet

Gyakorlatok

1.1. Numerikus integrálás

1.1.1. A gyakorlat célja

Az $\int_a^b f(x)dx$ határozott integrál numerikus közelítése.

1.1.2. Fogalmak

Legyen $[a, b]$ egy intervallum. A

$$B = \{x_1, x_2, \ldots, x_{n+1}\}$$

halmazt az $[a, b]$ intervallum egy ***beosztásá****nak nevezzük, ha*

$$a = x_1 < x_2 < \cdots < x_{n+1} = b.$$

Az $x_1, x_2, \ldots, x_{n+1}$ számokat ***osztópont****oknak nevezzük. Az osztópontok az $[a, b]$ intervallumot*

$$I_1 = [x_1, x_2], \quad I_2 = [x_2, x_3], \quad \ldots \quad , I_n = [x_n, x_{n+1}]$$

részintervallumokra bontják. Tehát az osztópontok száma egyel több, mint a részintervallumok száma. (Szokás az osztópontokat 0-tól kezdve indexelni, ebben az esetben az utolsó osztópont x_n és n számú részintervallumunk van. A Matlab azonban a vektorok indexeléséz 1-gyel kezdi, így mi is ehhez igazodunk.)

Az $[a, b]$ intervallum egy B beosztását ***ekvidisztáns beosztás****nak nevezzük, ha az osztópontok által meghatározott részintervallumok azonos hosszúságúak.*

Ha B az $[a, b]$ intervallum egy ekvidisztáns beosztása $n + 1$ osztóponttal, akkor

- $h = \frac{b-a}{n}$ *a lépésköz,*
- $x_k = x_{k-1} + h \qquad (k = 2, 3, \ldots, n+1)$
- $x_k = a + (k-1)h \qquad (k = 2, 3, \ldots, n+1)$
- *Így* $x_1 = a$ *és* $x_{n+1} = b$.

Ismétlés: *A beosztást végző* ***Matlab függvény****:*

$$z = linspace(a, b, n).$$

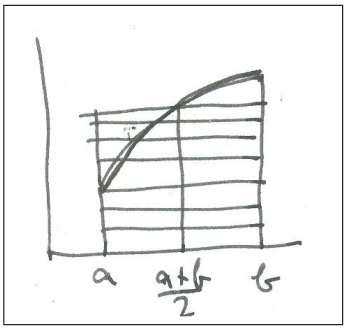
Az $[a, b]$ *intervallum egy ekvidisztáns beosztását adja úgy, hogy az osztópontok száma* n*, így a kapott részintervallumok száma* $n - 1$*. Hatása azonos a*

$$z = a : (b-a)/(n-1) : b$$

parancs hatásával. A z=linspace(a,b) esetén n értéke automatikusan 100.

1.1.3. Téglalap módszer

Egyszerű téglalap módszer:



$$\int_a^b f(x)dx \approx (b-a) f\left(\frac{a+b}{2}\right).$$

Összetett téglalap módszer:

Az ekvidisztáns beosztás minden részintervallumára alkalmazzuk az egyszerű téglalap módszert.

Az összetett téglalap formula matematikai képlete:

$$\int_a^b f(x)dx \approx h \sum_{k=2}^{n+1} f\left(\frac{x_{k-1} + x_k}{2}\right).$$

Az összetett téglalap formulát megvalósító Matlab függvény:

```
function I=OsszTeglalapInt(h,y)
I=h*sum(y);
end
```

Input: h= lépésköz,

$$y = \left[f\left(\frac{x_1 + x_2}{2}\right), \ldots, f\left(\frac{x_n + x_{n+1}}{2}\right) \right],$$

Output: Az integrálközelítő érték.

Feladat: *Határozzuk meg az*

$$\int_0^1 e^{-x^2} dx$$

integrálközelítő értékét összetett téglalap formulával.

Mo: *Motiváció: az $x \to e^{-x^2}$ függvény primitív függvénye nem elemi függvény, úgyhogy ha kíváncsiak vagyunk az integrálközelítő értékére, nincs más választásunk, mint az, hogy numerikusan integráljunk.*
Hívás: *10 részintervallum esetén (n=10)*

```
h=0.1
u=0.05:0.1:1
y=exp(-u.^2)
I=OsszTeglalapInt(h,y)
```

Eredmény: $n = 10$ *esetén* 0.7471,
$n = 100$ *esetén* 0.7468, *ami már elég pontos.*

1.1.4. A Newton-Cotes kvadratúrák

Isac Newton (1642-1727)
Roger Cotes (1682-1716)

Egyszerű formula: $n + 1$ *alappontra Lagrange-féle interpolációs polinomot illesztünk, majd az eredeti integrandus helyett az interpolációs polinomot integráljuk.*

Összetett formula: *Ekvidisztáns beosztást alkalmazunk, a kapott beosztás megfelelő részintervallumaira alkalmazzuk az egyszerű formulát, majd összegezzük.*

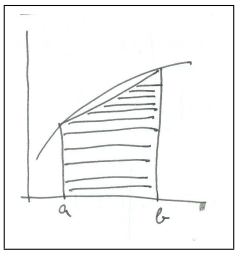
2 alappontot alkalmazunk, a két alappontra egyenest illesztünk, így kapjuk az egyszerű trapéz formulát. Az egyszerű trapéz formulából könnyen származtatható az összetett trapéz formula.

Ha 3 alappontra parabolát illesztünk, akkor az egyszerű Simpson formulát kapjuk. Az egyszerű Simpson formulából származtatjuk az összetett Simpson formulát. Tehát az összetettSimpson formula esetén a részintervallumok száma páros.

Az egyszerű trapéz formula *Mivel ismerjük a trapéz területképletét,*

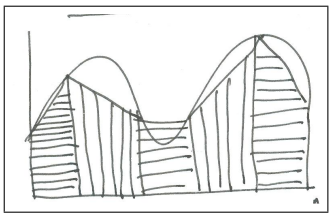
$$T = \frac{m}{2}(a+c)$$

így integrálnunk sem kell. Az egyszerű trapéz formulát:



$$T(f,a,b) = \frac{b-a}{2}\left(f(a)+f(b)\right)$$

Összetett trapéz formula: *A részintervallumok száma: n, az osztópontok száma: $n+1$.*



$$\begin{aligned}
T_n(f,a,b) &= \sum_{k=2}^{n+1} T(f,x_{k-1},x_k) = \sum_{k=2}^{n+1} \frac{x_k - x_{k-1}}{2}\left(f(x_{k-1})+f(x_k)\right) = \\
&= \frac{h}{2}\sum_{k=2}^{n+1}\left(f(x_{k-1})+f(x_k)\right) = \\
&= \frac{h}{2}\left(\left(f(x_1)+f(x_2)\right)+\left(f(x_2)+f(x_3)\right)+\cdots+\left(f(x_n)+f(x_{n+1})\right)\right) = \\
&= \frac{h}{2}\left(f(x_1)+2\sum_{k=2}^{n} f(x_k)+f(x_{n*1})\right).
\end{aligned}$$

Így kapjuk, hogy

$$T_n(f,a,b) = \frac{h}{2}\left(f(x_1) + 2\sum_{k=2}^{n} f(x_k) + f(x_{n+1})\right)$$

Matlab függvény az összetett trapéz formulára

```
function I=OsszTrapInt(x,y)
n=length(x);
n=n-1;
h=x(2)-x(1);
s=0;
for i=2:1:n
    s=s+y(i);
end
I=(h/2)*(y(1)+2*s+y(n+1));
end
```

Input: $x = [x_1, x_2, \ldots, x_{n+1}]$, $y = [f(x_1), \ldots, f(x_{n+1})]$.
Output: I az integrál közelítő értéke összetett trapéz formulával.

Feladat *: Összetett trapéz formula alkalmazásával határozzuk meg az*

$$\int_a^b e^{-x^2} dx$$

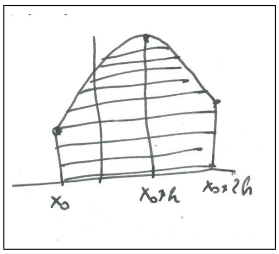
integrál közelítő értékét.
***Hívás**: 10 részintervallum esetén (n=10)*

```
x=linspace(0,1,11)
y=exp(-x.^2)
I=OsszTrapInt(x,y)
```

***Eredmény**: $n = 10$ esetén 0.7462,*
$n = 100$ esetén 0.7468.

Az egyszerű Simpson formula:
Thomas Simpson (1710-1761) angol matematikus, Cotes eredményét publikálta.

Az alappontok és függvényértékek:



x_0	x_0+h	x_0+2h
y_0	y_1	y_2

$$P_3(x) = y_0 l_1(x) + y_1 l_2(x) + y_2 l_3(x)$$

$$\int_{x_0}^{x_0+2h} P_3(x) = \frac{h}{3}\left(y_0 + 4y_1 + y_2\right).$$

Az egyszerű Simpson formula levezetése: *(Kihagyható, bár rutinfeladat.)*

Gyakorlásképpen nem árt felírni az l_1, l_2, l_3 polinomokat:

$$l_1(x) = \frac{(x-(x_0+h))\,(x-(x_0+2h))}{(x_0-(x_0+h))\,(x_0-(x_0+2h))} = \frac{1}{2h^2}\,(x-(x_0+h))\,(x-(x_0+2h))$$

$$l_2(x) = \frac{(x-x_0)\,(x-(x_0+2h))}{(x_0+h-x_0))\,(x_0+h-(x_0+2h))} = -\frac{1}{h^2}\,(x-x_0)\,(x-(x_0+2h))$$

$$l_3(x) = \frac{(x-x_0)\,(x-(x_0+h))}{(x_0+2h-x_0))\,(x_0+2h-(x_0+h))} = \frac{1}{2h^2}\,(x-x_0)\,(x-(x_0+h))$$

A helyettesítéses integrálás képletét alkalmazva kapjuk, hogy

$$\int_{x_0}^{x_0+2h} (x-(x_0+h))\,(x-(x_0+2h))\,dx = \int_{-h}^{h} u(u-h)du =$$

$$\left[\begin{array}{l} \textit{régi határok } x: x_0 \to x_0+2h \\ \textit{új határok } u = x-(x_0+h): -h \to h \\ x = u+(x_0+h) \\ \dfrac{dx}{du} = 1 \;\Rightarrow\; dx = du \end{array}\right]$$

$$= \int_{-h}^{h} (u^2 - uh)du = \left[\frac{u^3}{3} - \frac{u^2 h}{2}\right]_{u=-h}^{u=h} = \frac{2h^3}{3}$$

$$\int_{x_0}^{x_0+2h} (x-x_0)\,(x-(x_0+2h))\,dx = \int_{0}^{2h} u(u-2h)du =$$

$$\left[\begin{array}{l} \textit{régi határok } x: x_0 \to x_0+2h \\ \textit{új határok } u = x-x_0: 0 \to 2h \\ x = u+x_0 \\ \dfrac{dx}{du} = 1 \;\Rightarrow\; dx = du \end{array}\right]$$

$$= \int_{0}^{2h} (u^2 - 2uh)du = \left[\frac{u^3}{3} - \frac{2hu^2}{2}\right]_{u=0}^{u=2h} = -\frac{4}{3}h^3$$

$$\int_{x_0}^{x_0+2h} (x-x_0)\,(x-(x_0+h))\,dx = \int_0^{2h} u(u-h)du =$$

$$\left[\begin{array}{l} \textit{régi határok } x: x_0 \to x_0+2h \\ \textit{új határok } u = x-x_0: 0 \to 2h \\ x = u+x_0 \\ \dfrac{dx}{du} = 1 \;\Rightarrow\; dx = du \end{array}\right]$$

$$= \int_0^{2h} (u^2 - uh)du = \left[\frac{u^3}{3} - \frac{hu^2}{2}\right]_{u=0}^{u=2h} = -\frac{2}{3}h^3.$$

Így kapjuk, hogy

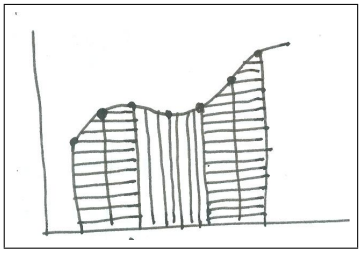
$$\int_{x_0}^{x_0+2h} P_3(x) = y_0 \frac{1}{2h^2}\frac{2h^3}{3} + y_1\left(-\frac{1}{h^2}\right)\left(-\frac{4}{3}h^3\right) + y_2\frac{1}{2h^2}\frac{2h^3}{3} =$$

$$= \frac{h}{3}\left(y_0 + 4y_1 + y_2\right)$$

azaz

$$S\left(f, x_0, x_0+h, x_0+2h\right) = \frac{h}{3}\left(y_0 + 4y_1 + y_2\right).$$

Az összetett Simpson formula *Ekvidisztáns beosztást alkalmazunk páros sok részintervallummal, majd 3 alappontonként parabolát illesztünk.*



$$S_n\left(f,a,b\right) = \sum_{i=0}^{\frac{n}{2}-1} S\left(f, x_1+2i, x_2+2i, x_3+2i\right) =$$

$$= \frac{h}{3}\left(\left(y_1+4y_2+y_3\right) + \left(y_3+4y_4+y_5\right) + \cdots + \left(y_{n-1}+4y_n+y_{n+1}\right)\right) =$$

$$= \frac{h}{3}\left(y_1 + 4\sum_{\substack{k=2 \\ k\ Ps}}^{n} y_k + 2\sum_{\substack{k=3 \\ k\ Pt}}^{n-1} y_k + y_{n+1}\right).$$

A k Ps illetve a k Pt azt jelenti, hogy az összegzés páros illetve páratlan indexekre vonatkozik. Tehát

$$S_n(f,a,b) = \frac{h}{3}\left(y_1 + 4\sum_{\substack{k=2\\ k\ Ps}}^{n} y_k + 2\sum_{\substack{k=3\\ k\ Pt}}^{n-1} y_k + y_{n+1} \right).$$

Ezek után már könnyű megírni az Összetett Simpson Szabályt alkalmazó Matlab függvényt. (n a részintervallumok száma.)

```
function I=OsszSimpInt(x,y)
n=length(x);
n=n-1;
h=x(2)-x(1);
s=0;
for i=2:2:n
    s=s+y(i);
end
t=0;
for i=3:2:n-1
    t=t+y(i);
end
I=(h/3)*(y(1)+4*s+2*t+y(n+1));
end
```

Feladat: *Számoljuk ki az*

$$\int_0^1 e^{-x^2} dx$$

integrál értékét $n = 100$ részintervallum esetén (osztópontok száma:101).

***Hívás**: 100 részintervallum esetén (n=100)*

```
x=linspace(0,1,101)
y=exp(-x.^2)
I=OsszSimpInt(x,y)
```

***Eredmény**: $n = 10$ esetén 0.7468*
$n = 100$ esetén 0.7468.

Feladat: *Számoljuk ki az*

$$\int_0^{\pi} \sin(x) dx$$

integrál értékét.

$$\int_0^{\pi} \sin(x)dx = \Big[-\cos(x) \Big]_{x=0}^{x=\pi} = (-cos(\pi) + cos(0)) = 1 + 1 = 2.$$

Ez az integrál remek alkalmat ad arra, hogy a tanult két módszert összehasonlítsuk. Írjunk Matlab függvényt, amely $n = 2, 4, 8, 10, 20, 100$ *értékek esetén kiszámolja a*

$$\int_0^{\pi} \sin(x)dx$$

integrál értékét összetett trapéz formulával és összetett Simpson formulával, majd n *értékét, illetve a kapott értékeket elhelyezi egy mátrixban.*

```
1   function A=CompTrapSimp()
2 -   A=zeros(6,3);
3 -   A(:,1)=[2;4;8;10;20;100];
4 -   for i=1:1:6
5 -     x=linspace(0,pi,A(i,1)+1);
6 -     y=sin(x);
7 -     A(i,2)=OsszTrapInt(x,y);
8 -     A(i,3)=OsszSimpInt(x,y);
9 -   end
10 -  end
```

```
>> A=CompTrapSimp()

A =

    2.0000    1.5708    2.0944
    4.0000    1.8961    2.0046
    8.0000    1.9742    2.0003
   10.0000    1.9835    2.0001
   20.0000    1.9959    2.0000
  100.0000    1.9998    2.0000
```

Hibaképletek összetett trapéz és összetett Simpson formula esetén:

A priori hibabecslés

$$\left| \int_a^b f(x)dx - T_n\left(f, a, b\right) \right| \leqslant \frac{M_2(b-a)^3}{12n^2},$$

ahol $M_2 = \sup\left\{\left|f''(x)\right| \middle| x \in [a,b]\right\}$.

$$\left|\int_a^b f(x)dx - S_n(f)\right| \leqslant \frac{M_4(b-a)^5}{180n^4},$$

ahol $M_4 = \sup\left\{\left|f^{(4)}(x)\right| \middle| x \in [a,b]\right\}$.

A posteriori hibabecslés

$$\left|\int_a^b -T_{2n}\right| \leqslant |T_{2n} - T_n|$$

$$\left|\int_a^b -S_{2n}\right| \leqslant |S_{2n} - S_n|$$

Feladat:

$$\int_1^2 \ln(x)dx = \Big[x\ln(x) - x\Big]_{x=1}^{x=2} = 2\ln(2) - 1 = 0.3863.$$

Ez az integrál remek alkalmat teremt az a priori és az a posteriori hibabecslés elvégzésére. (Önálló feladat és egyben remek Zh feladat.)

Feladat *π értékét közelítjük az alábbi integrál segítségével:*

$$\int_0^1 \frac{4}{1+x^2}dx = 4\Big[\arctan(x)\Big]_{x=0}^{x=1} = \pi.$$

```
1    function I=Pikozelites(n)
2 -    x=linspace(0,1,n+1);
3 -    y=(1+x.^2).^(-1)*4;
4 -    I=OsszSimpInt(x,y);
5 -    end
```

***Hívás**: n részintervallum esetén*

```
format long
I=Pikozlelites(10)
pi
```

***Eredmény**: A π értéke:* 3.141 592 653 589 793 . . .
$n = 10$ *esetén* 3.141 592 6|13 939 215 *azaz 7 jegyre pontos.*
$n = 20$ *esetén* 3.141 592 65|2 969 785 *azaz 8 jegyre pontos.*
$n = 100$ *esetén* 3.141 592 653 589 7|53 *azaz 13 jegyre pontos.*

Feladat: *A statisztikában a 68-95-99.7 szabály mondja meg, hogyha az adatok m várható értékű és σ^2 szórásnégyzetű normális eloszlásból származnak, akkor az adatok 68%-a 95%-a, illetve 99.7%-a esik az m várható érték 1σ, 2σ, 3σ sugarú környezetébe. Ezeket az értékeket most numerikusan kiszámoljuk.*

Tehát $\xi \sim \mathcal{N}(m, \sigma^2)$ és a

$$P_k = \mathbb{P}\left(m - k\sigma < \xi < m + k\sigma\right) \qquad (k = 1, 2, 3)$$

értékét keressük. Standardizálással kapjuk, hogy

$$\mathbb{P}\left(m - k\sigma < \xi < m + k\sigma\right) = \mathbb{P}\left(-k < \frac{\xi - m}{\sigma} < k\right),$$

ahol $\eta \doteq \frac{\xi - m}{\sigma} \sim \mathcal{N}(0, 1)$ (standard normális eloszlású valószínűségi változó), továbbá

$$\mathbb{P}\left(-k < \eta < k\right) = \Phi(k) - \Phi(-k) = \frac{1}{\sqrt{2\pi}} \int_{-k}^{k} e^{\frac{-x^2}{2}} dx \qquad (k = 1, 2, 3).$$

Ez utóbbi integrál numerikus közelítése a feladat.

Mo: *Először rajzoltassuk ki a képernyőre a*

$$\varphi(x) = \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}} \qquad (x \in \mathbb{R})$$

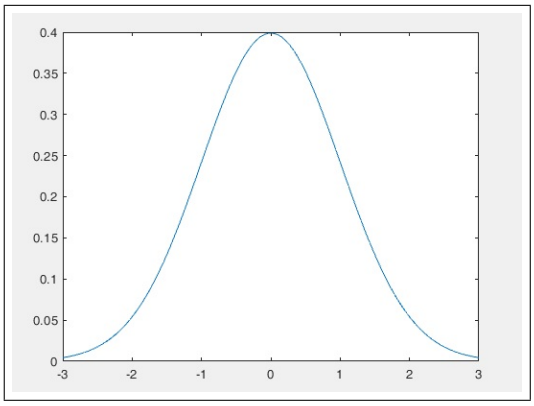
függvényt, ami a standard normális eloszlás sűrűségfüggvénye amit Gauss-féle haranggörbének szokás nevezni.

```
1    function phi=GaussHarang()
2 -  x=linspace(-3,3);
3 -  y=(1/sqrt(2*pi))*exp((-1/2)*x.^2);
4 -  phi=plot(x,y);
5 -  end
```

***Hívás**:*

```
format short
phi=GaussHarang()
```

***Eredmény**:*



Szigma Szabály

```
1   function p=SzigmaSzabaly(n)
2   p=zeros(3,1);
3   x1=linspace(0,1,n+1);
4   x2=linspace(1,2,n+1);
5   x3=linspace(2,3,n+1);
6   phi=@(x) (1/sqrt(2*pi))*exp((-1/2)*x.^2);
7   y1=phi(x1);
8   y2=phi(x2);
9   y3=phi(x3);
10  a=OsszSimpInt(x1,y1);
11  b=OsszSimpInt(x2,y2);
12  c=OsszSimpInt(x3,y3);
13  p(1)=2*a;
14  p(2)=p(1)+2*b;
15  p(3)=p(2)+2*c;
16  end
```

***Hívás**:*

```
format long
P=SzigmaSzabaly(20)
```

***Eredmény**: Először ismertetjük a Wikipedia-n található eredményt:*
0.682 689 492 190 859
0.954 499 736 091 643
0.997 300 203 927 876
Az általunk számolt eredmény $n = 20$ esetén:
0.682 889 525 . . .

0.954 499 728 . . .
0.997 300 198 . . .
$n = 100$ *esetén*
0.682 689 492 . . .
0.954 499 736 . . .
0.997 300 203 . . . .

Léteznek még más kvadratúra családok, például a Gauss kvadratúrák.

A numerikus integrálás nagyon ötletes módszere a Monte-Carlo módszer, amely több dimenzióban is alkalmazható, könnyen programozható. Véletlen számokat alkalmaz az integrál közelítésére.

Érdemes utánanézni a

1.2. Nemlineáris egyenletek numerikus megoldása

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: *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?*

1.2.1. Zérushelykeresés

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

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.*

A magam részéről a második utat javaslom, ugyanis a Matlab nyelv használata a számunkra inkább segédeszköz, mint cél. Ennek ellenére - leginkább lustaságból - a kidolgozott feladatokban univerzális függvényeket írtam.

Azok számára, akik kíváncsiak az (1)-ben leírtak Matlab-beli megvalósítására, bemutatom az $EVAL$ és a $FEVAL$ parancsok használatát.

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:

```
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: $y = 10$

1.2.2. Iterációk

Egy pontra támaszkodó stacionér iteráció:

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

Két pontra támaszkodó stacionér iteráció:

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

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

Fibonacci-sorozat:

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

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: *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
```

1.2.3. 1. Intervallumfelezés

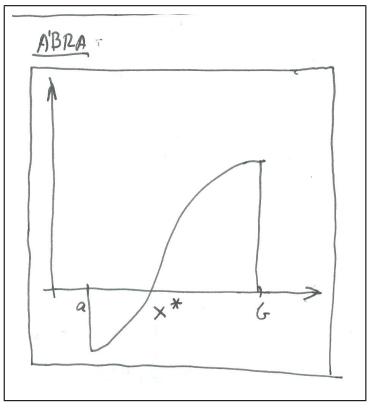
Matematikai alapok

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

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

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 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:*

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

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.*
Hibabecslés: $[a_k, b_k]$ *intervallumok sorozatát állítjuk elő.* $x^*, x_k \in [a_k, b_k]$, *így*

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

mivel az x^**-nak az intervallum felezőpontjától mért távolságát figyeljük.*

1.2.4. 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} *osztópónttal 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:

Irányvektor: $(b_{k-1} - a_{k-1}, f(b_{k-1}) - f(a_{k-1}))$

Normálvektor: $(f(a_{k-1}) - f(b_{k-1}), b_{k-1} - a_{k-1}) = (A, B)$

A húr egyenlete: $Ax + By = Ax_0 + By_0 \qquad (P_0 = (a_{k-1}, f(a_{k-1})) = (x_0, y_0)$

$(f(a_{k-1}) - f(b_{k-1}))\, x + (b_{k-1} - a_{k-1})\, y = (f(a_{k-1}) - f(b_{k-1}))\, a_{k-1} + (b_{k-1} - a_{k-1})\, f(a_{k-1})$.

$y = 0$, akkor

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

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

- *$f(x_{k-1}) = 0$, 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.)

1.2.5. 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:

$$\begin{aligned} 0 = f(x^*) &\approx f(x_{k-1}) + f'(x_{k-1})(x^* - x_{k-1}) \\ x^* &\approx x_{k-1} - \frac{f(x_{k-1})}{f'(x_{k-1})} \\ x_k &= x_{k-1} - \frac{f(x_{k-1})}{f'(x_{k-1})} \end{aligned}$$

Elegendő feltétel a konvergenciára *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 *Írjuk fel az* f *függvény elsőrendű Taylor polinomját.*

$$0 = f(x^*) = f(x_{k-1}) + f'(x_{k-1})(x^* - x_{k-1}) + \frac{f''(\xi)}{2}(x^* - x_{k-1})^2,$$

ahol ξ x^* *és* x_k *között van. Ebből rendezéssel kapjuk, hogy*

$$|x^* - x_k| \leqslant \frac{f''(\xi)}{2f'(x_{k-1})} \left|x^* - x_{k-1}\right|^2. \qquad (k = 1, 2, \dots)$$

Ha tehát

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

akkor

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

A leállás feltétele:

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

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

1.2.6. 4. Szelő módszer

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.

1.2.7. 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*

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

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 ahhoz, hogy egy függvény kontraktív legyen *Ha* $f : [a, b] \to [a, b]$ *olyan, hogy teljesíti a Lagrange-tétel feltételeit, továbbá létezik olyan* $q \in]0, 1[$*, amellyel*

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

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*

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

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.*

- ***A*** q ***kontrakciós faktor***

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

  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.

1.2.8. Matlab programok

Intervallumfelezés:

```
function [x,it]=IntFel1(f,a,b,epsilon)
x=(a+b)/2;
hiba=(b-a)/2;
it=1;
while(hiba>epsilon)
    fx=eval(f);
    if fx==0
        return;
    end
    u=x;
    x=a;
    fa=eval(f);
    if fa*fx<0
        b=u;
    else
        a=u;
    end
    x=(a+b)/2;
    hiba=hiba/2;
    it=it+1;
end
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:

```
function [x,it]=NewtonRaphson(f,f1,x,epsilon)
fx=eval(f);
hiba=abs(fx);
it=1;
while(hiba>epsilon)
    f1x=eval(f1);
    x=x-fx/f1x;
    fx=eval(f);
    it=it+1;
    hiba=abs(fx);
end
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.

Fixpontiteráció:

```
function [x,it]=Fixpontit(g,x,q,epsilon)
u=x;
x=eval(g);
K=q/(1-q);
it=1;
while(K*abs(u-x)>epsilon)
    u=x;
    x=eval(g);
    it=it+1;
end
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.