Eötvös Loránd Tudományegyetem
Természettudományi Kar

Véletlen iterációk statisztikai elemzése

BSc Szakdolgozat

Írta: Mészáros Ádám István
Matematika BSc, Matematikai elemző szakirány

Témavezető: Sigray István
Műszaki Gazdasági Tanár, Analízis Tanszék

2013
Budapest

Tartalomjegyzék

1. Bevezetés **3**

2. A Newton-módszer **4**

2.1. A módszer leírása 4

2.2. Konvergencia vizsgálat 5

2.3. Komplex eset 8

3. Módosított módszer polinomokra **13**

3.1. Motiváció 13

3.2. A módszer 14

3.3. Elemzés 16

3.3.1. Egy példán keresztül 19

3.3.2. Érdekes esetek 27

3.3.3. Ellenpélda 28

4. Összefoglalás **34**

A. A dolgozatban használt programok bemutatása **36**

A.1. A felhasznált módszerek megvalósítása 36

A.2. Konvergencia szemléltetése 38

Irodalomjegyzék **41**

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani témavezetőmnek, Sigray Istvánnak, a kiváló témáért és, hogy folytonos tanácsaival és ötleteivel elengedhetetlen segítséget nyújtott a szakdolgozatom elkészítéséhez. Továbbá köszönettel tartozom családomnak, akik az utolsó pillanatokban egyéb elfoglaltságuk ellenére vették a fáradtságot, hogy átnézzék a szakdolgozatomat. Végül pedig páromnak, Zsanettnek, aki a legnehezebb pillanataimban is mellettem állt.

1. fejezet

Bevezetés

A mindennapok során gyakran szembesülünk olyan problémákkal amelyek megoldásához, vagy optimalizálásához különböző egyenleteket kell megoldanunk. Az ilyen egyenletek pontos megoldása nem mindig könnyű, vagy egyáltalán lehetséges feladat. Ilyen probléma például általános megoldóképlet hiányában az ötöd vagy magasabb fokú polinomok gyökeinek keresése. Ezen feladatok gyors megoldásánál numerikus módszerek alkalmazására kell hagyatkoznunk.

Ezeknek a módszereknek a gyorsaságát több tényező is kedvezőtlenül alakíthatja, így például be fogom mutatni, hogy az egyik legismertebb iterációs eljárásnál: a Newton-módszernél csak milyen esetekben garantált a konvergencia. A dolgozatom során megpróbálok egy olyan iterációs módszert konstruálni a Newton-módszer alapján, mellyel gyors konvergenciát érhetünk el olyan helyekről is ahonnan a Newton-módszer nem, vagy csak lassan konvergál.

A módszer konstruálásánál egy véletlen változót fogok felhasználni, amely miatt az eredmény több futtatás során, több különböző eredményt is adhat. Emiatt a dolgozatom második felében ezt a módszert több példán általam írt számítógépes programokkal fogom elemezni.

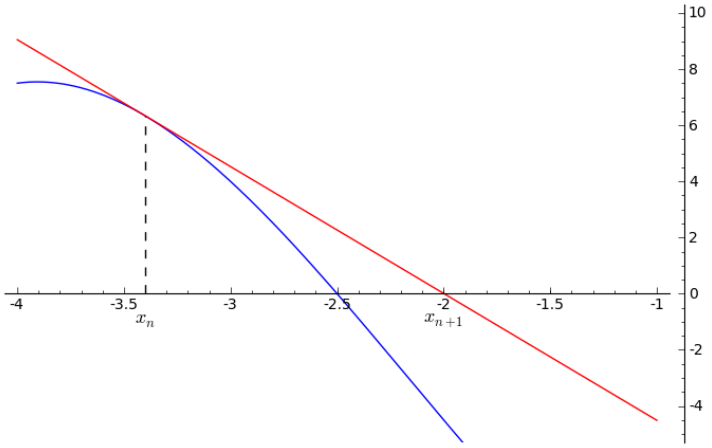
Az eredmények rekonstruálhatósága végett minden felhasznált nagyobb programot a függelékben ismertetek, és a kisebb kódokat pedig a dolgozat során kiírom és elmagyarázom.

2. fejezet

A Newton-módszer

2.1. A módszer leírása

A Newton-módszert, mint a neve is mutatja, először Isaac Newton dolgozta ki, azonban a mai formáját Joseph Raphson írta le 1690-ben, ezért Newton-Raphson-módszernek is hívják. Newton a módszert eredetileg polinomokra alkalmazta, ahogy a későbbiekben majd én is teszem. A Newton-módszer egy iterációs eljárás nemlineáris egyenletek gyökeinek megtalálására, közelítésére. A módszer - valós esetben - egy geometriailag nagyon egyszerű meggondoláson alapul, amit egy kép segítségével fogok bemutatni.



2.1. ábra. Egy lépés a Newton-módszerrel

Első lépésként vegyünk egy kezdőpontot az x tengelyen, amely elég közel van a gyökünkhöz. Jelöljük x_0-lal. Tegyük fel, hogy n lépés után eljutottunk az x_n pontig. Húzzunk érintőt a grafikonhoz ebben a pontban. Az érintő és az x tengely metszéspontja legyen a következő pontunk, x_{n+1}, majd így folytassuk amíg elég közel nem kerülünk az egyenletünk gyökéhez.

Tegyük fel, hogy x_k közel van az $f(x) = 0$ egyenlet x^* gyökéhez, és f kétszer folytonosan differenciálható. Ekkor a következőképpen írhatjuk fel f elsőfokú Taylor-polinomját maradéktaggal

$$f(x^*) = f(x_k) + f'(x_k)(x^* - x_k) + O((x^* - x_k)^2)$$

$f(x^*)$ lineáris közelítéséhez nincsen szükségünk az $(x^* - x_k)^2$-es tagokra.

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

Definiáljunk egy lépést a közelítés jobb oldalával, így előállíthatjuk a következő iterációt

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} \tag{2.1}$$

2.1.1. Példa. *Az alábbiakban a 2.1. ábrán látható polinommal fogok számolni.*

$$f(x) = x^3 + 6{,}5x^2 + 5x - 12{,}5$$

$f(x) = 0$ *egyenletnek gyöke az* $x^* = -2{,}5$. *Induljunk el az* $x_0 = -1$ *pontból.*

k	x_k	$f(x_k)$	$\lvert x_k - x^* \rvert$
0	$-1{,}0000$	$-1{,}2000e+01$	$1{,}5000e+00$
1	$-3{,}4000$	$6{,}3360e+00$	$9{,}0000e-01$
2	$-1{,}9982$	$-4{,}5159e+00$	$5{,}0177e-01$
3	$-2{,}5001$	$8{,}6664e-04$	$9{,}9045e-05$
4	$-2{,}5000$	$-9{,}8121e-09$	$1{,}1214e-09$

2.2. Konvergencia vizsgálat

Legyen $[a, b] \subset \mathbb{R}$ egy intervallum, $f : [a, b] \to \mathbb{R}$ kétszer folytonosan differenciálható és $\exists! x^* \in [a, b]$ amelyre $f(x^*) = 0$. Ebben az alfejezetben azt vizsgáljuk, hogy milyen esetben tudjuk garantálni azt, hogy a Newton-módszer f x^* gyökéhez konvergál.

2.2.1. Jelölés. $m_1 := \min\limits_{x \in [x_k, x^*]} |f'(x)|$, $M_2 := \max\limits_{x \in [x_k, x^*]} |f''(x)|$, $C := \frac{M_2}{2m_1}$

2.2.2. Tétel. *Tegyük fel, hogy* $m_1 > 0$ *és a Newton-módszer konvergens, ekkor másodrendben konvergál a gyökhöz.*

Bizonyítás. Írjuk fel f x_k körüli elsőfokú Taylor-polinomját Lagrange maradéktaggal az x^* pontban

$$\begin{aligned} f(x^*) &= f(x_k) + f'(x_k)(x^* - x_k) + \frac{f''(\xi)}{2}(x^* - x_k)^2 && (2.2)\\ 0 &= \frac{f(x_k)}{f'(x_k)} + (x^* - x_k) + \frac{\frac{f''(\xi)}{2}(x^* - x_k)^2}{f'(x_k)} && (2.3)\\ x^* &= x_k - \frac{f(x_k)}{f'(x_k)} - \frac{f''(\xi)}{2f'(x_k)}(x^* - x_k)^2 && (2.4) \end{aligned}$$

Ahol $\xi \in (x_k, x^*)$. Vonjuk ki a 2.1. egyenletből a 2.4.-t

$$\begin{aligned} x_{k+1} - x^* &= \left(x_k - \frac{f(x_k)}{f'(x_k)}\right) - \left(x_k - \frac{f(x_k)}{f'(x_k)} - \frac{f''(\xi)}{2f'(x_k)}(x^* - x_k)^2\right)\\ x_{k+1} - x^* &= \frac{f''(\xi)}{2f'(x_k)}(x^* - x_k)^2 \end{aligned}$$

Vegyük mindkét oldal abszolút értékét

$$\begin{aligned} |x_{k+1} - x^*| &= \frac{|f''(\xi)|}{2|f'(x_k)|}|x^* - x_k|^2 && (2.5)\\ |x_{k+1} - x^*| &\leq C|x^* - x_k|^2 && (2.6) \end{aligned}$$

Vagyis hogyha a Newton-módszer konvergens, akkor másodrendben konvergál. □

A másodrendű konvergencia azt jelenti, hogy a hiba legalább négyzetesen csökken, vagyis a helyes jegyek száma nagyjából megduplázódik a lépésenként.

Hogyan tudjuk garantálni a konvergenciát? 2.6 egyenlőtlenséget a következőképpen tudjuk iterálni x^* közelében

$$\begin{aligned} |x_k - x^*| &\leq C|x^* - x_{k-1}|^2 && (2.7)\\ &\leq C\left|C|x^* - x_{k-2}|^2\right|^2 = C^3|x^* - x_{k-2}|^4 && (2.8)\\ &\leq \ldots \leq C^{2^k-1}|x^* - x_0|^{2^k} && (2.9) \end{aligned}$$

Vagyis $|x_k - x^*| \leq C^{2^k-1}|x^* - x_0|^{2^k}$. Szeretnénk, ha a következő teljesülne

$$\lim_{k\to\infty} |x^* - x_k| = 0$$

Ha a 2.9. egyenletben az egyenlőtlenség jobb oldala tart nullához, akkor a Rendőr-elv alapján ez teljesül.

$$C^{2^k-1}|x^* - x_0|^{2^k} = q^{2^k-1}|x^* - x_0|$$

Ez akkor és csak akkor tart nullához, ha $q < 1$.

$$q = C|x^* - x_0| = \frac{M_2}{2m_1}|x^* - x_0| < 1$$

Átrendezve

$$|x_0 - x^*| < \frac{2m_1}{M_2}$$

Ezek alapján a következő tételt mondhatjuk ki

2.2.3. Tétel. *Legyen* $m_1 := \min_{x\in[a,b]} |f'(x)|$, *és* $M_2 := \max_{x\in[a,b]} |f''(x)|$. *Tegyük fel, hogy* $m_1, M_2 > 0$ *és* $|x_0 - x^*| < \frac{2m_1}{M_2}$. *Ekkor*

$$\lim_{k\to\infty} x_k = x^*$$

2.2.4. Tétel. *Tegyük fel, hogy* $x_0 \in [a, b]$, $\forall x \in (x^*, x_0)$*-re* $f'(x)$,$f''(x) \neq 0$, *és* x_0*-ra teljesül, hogy* $f(x_0)f''(x_0) > 0$, *akkor* x_k *szigorúan monoton tart a gyökhöz.*

Bizonyítás. Négy esetünk van.

1. $x_0 > x^*$, $f(x_0) > 0$
2. $x_0 > x^*$, $f(x_0) < 0$
3. $x_0 < x^*$, $f(x_0) > 0$
4. $x_0 < x^*$, $f(x_0) < 0$

Az esetek bizonyítása nagyban hasonlít, ezért csak az elsőt látjuk be.

Mivel $f(x_0) > 0$ és $f(x_0)f''(x_0) > 0$, ezért $f''(x_0) > 0$. Továbbá (x^*, x_0)-n $f'(x) \neq 0$, ezért az vagy mindenhol pozitív, vagy mindenhol negatív.

$$\begin{aligned} 0 &< f(x_0) && (2.10)\\ &= f(x_0) - f(x^*) && (2.11)\\ &= f'(\xi)\underbrace{(x_0 - x^*)}_{>0} && (2.12) \end{aligned}$$

2.11 $\Rightarrow$ 2.12 a Lagrange-féle középérték tétel miatt, valamilyen $\xi \in (x^*, x_0)$-re. Ez azt jelenti, hogy $f'(\xi) > 0$, amiből következik, hogy $\forall x \in (x^*, x_0) : f'(x) > 0$. Ez alapján a

következőt tudjuk

$$\begin{aligned} x_{k+1} &= x_k - \underbrace{\frac{f(x_k)}{f'(x_k)}}_{>0} \\ x_{k+1} &< x_k \end{aligned}$$

Vagyis az x_k sorozat szigorúan monoton csökkenő. Használjuk ismét a Lagrange-féle középérték tételt

$$0 < f(x_k) = f(x_k) - f(x^*) = \underbrace{f'(\xi)}_{>0}(x_k - x^*)$$

Tehát $x_k > x^*$ azaz x_k sorozat korlátos. A korlátosságból és a monotonitásból következik, hogy x_k konvergens. Tegyük fel, hogy $x_k \to \tilde{x}^*$.

$$\begin{aligned} \lim_{k\to\infty} x_{k+1} &= \lim_{k\to\infty} x_k - \frac{\lim\limits_{k\to\infty} f(x_k)}{\lim\limits_{k\to\infty} f'(x_k)} \\ \tilde{x}^* &= \tilde{x}^* - \frac{f(\tilde{x}^*)}{f'(\tilde{x}^*)} \\ \frac{f(\tilde{x}^*)}{f'(\tilde{x}^*)} &= 0 \end{aligned}$$

Ez csak akkor lehetséges, ha $f(\tilde{x}^*) = 0$, viszont x^* gyök egyértelmű, ezért $x^* = \tilde{x}^*$. □

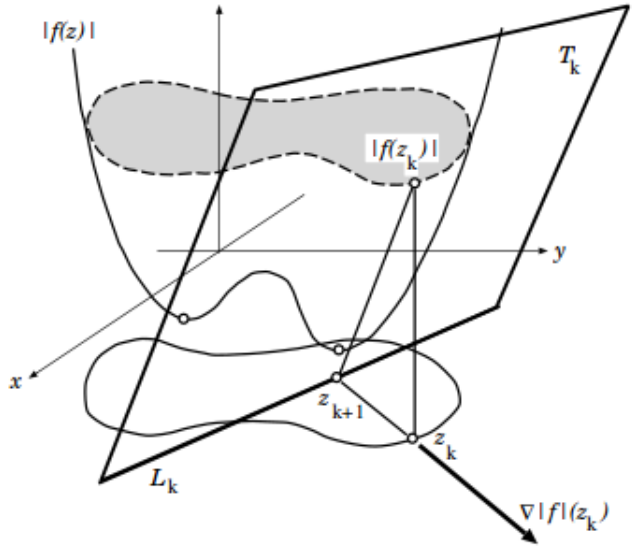
2.3. Komplex eset

Érdemes megvizsgálni a módszert komplex gyökkel rendelkező egyenletekre is. Ekkor a fentiekkel analóg állítások mondhatók ki. A komplex eset geometriai meggondolása is hasonló. Feleltessük meg a $z = x + yi \in \mathbb{C}$ számokat $(x, y) \in \mathbb{R}^2$ vektorokkal. Legyen $f(z) = u + vi = u(x, y) + v(x, y)i$ analitikus függvény. $|f(z)| = \sqrt{u^2(x, y) + v^2(x, y)}$. $z_k = x_k + y_k i$ ahol f és f' sem 0. Legyen T_k $|f(z)|$ érintő síkja a (x_k, y_k) helyen, és legyen L_k az xy sík és T_k metszésvonala. Belátható, hogy ekkor

$$z_{k+1} = x_{k+1} + y_{k+1}i = z_k - \frac{f(z_k)}{f'(z_k)}$$

megfelel annak az (x_{k+1}, y_{k+1}) pontnak, amelyik az L_k egyenesen legközelebb van (x_k, y_k) ponthoz. (2.2) [4, p. 809]. A lépés iránya ezáltal ellentétes $|f(z)|$ gradiensével a z_k pontban.

Meggondolható, hogy ez valós esetben ugyanazt jelenti mint eddig, hiszen a két sík metszésvonala helyett, az x tengely és $|f(x)|$ érintő egyenesének metszéspontját vesszük,



2.2. ábra. A komplex eset geometriai meggondolása [4, p. 807]

ami megegyezik az x tengely és $f(x)$ érintőjének metszéspontjával.

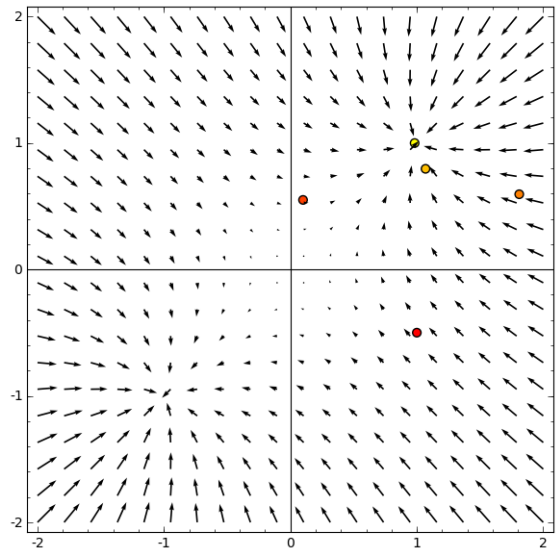
Vegyük például a $f(x) = x^2 - 2i$ komplex együtthatós polinomot, amelynek az $1 + i$ és a $-1 - i$ a gyökei. $|f|$ gradiensénél (2.3(a). ábra) valamivel jobb szemléltetés végett definiáljuk a következő vektormezőt

$$g(x, y) = \left(\mathrm{Re} \left\{ -\frac{f(x + yi)}{f'(x + yi)} \right\}, \mathrm{Im} \left\{ -\frac{f(x + yi)}{f'(x + yi)} \right\} \right)$$

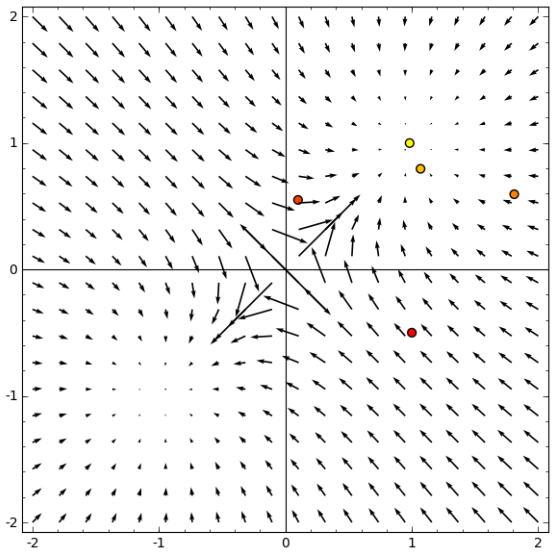
Ezt ábrázolva láthatjuk, hogy az esetünkben hogyan fog haladni a komplex síkon a módszer. A 2.3. ábrákon jelöltem $1 - 0{,}5i$-ből indított módszer pontjait is. A kezdőpont pirossal van ábrázolva, ami az iterációs lépésekkel átvált sárgába.

2.3.1. Példa. *2.3. ábra számokkal, vagyis $f(x) = x^2 - 2i$ gyökét keressük $1 - 0{,}5i$ kezdőponttal.*

k	z_k	$\lvert f(z_k)\rvert$	$\lvert z_k - z^*\rvert$
0	$1{,}00000 - 0{,}50000i$	$3{,}0923e + 00$	$1{,}5000e + 00$
1	$0{,}10000 + 0{,}55000i$	$1{,}9125e + 00$	$1{,}0062e + 00$
2	$1{,}81000 + 0{,}59500i$	$2{,}9261e + 00$	$9{,}0561e - 01$
3	$1{,}06891 + 0{,}79611i$	$5{,}8966e - 01$	$2{,}1522e - 01$
4	$0{,}98262 + 0{,}99980i$	$4{,}8935e - 02$	$1{,}7377e - 02$
5	$1{,}00008 + 0{,}99992i$	$3{,}0464e - 04$	$1{,}0771e - 04$
6	$1{,}00000 + 1{,}00000i$	$1{,}1601e - 08$	$4{,}1015e - 09$



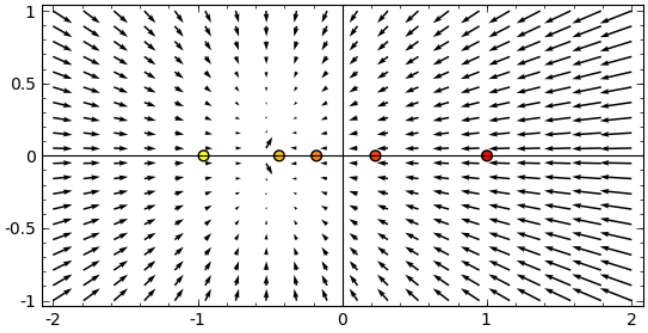
(a) $-\nabla|f(z)|$ és négy iterációs lépés az $1-0{,}5i$-ből



(b) $g(x,y)$ és négy iterációs lépés az $1-0{,}5i$-ből

2.3. ábra. A lépés irányán túl a (b) ábra a lépés nagyságát is szemlélteti

Láthatjuk, a módszer nem csak komplex gyökökkel rendelkező valós polinomokra működik, hanem komplex együtthatós polinomokra is. Azonban ha egy valós polinom komplex gyökét szeretnénk meghatározni, akkor szükséges, hogy a kezdőpontunk képzetes része eltérjen nullától, különben az iterációs lépés tulajdonsága miatt $\forall z_k \in \mathbb{R}$ (2.4. ábra).



2.4. ábra. $(x-(-1/2-i/4))(x-(-1/2+i/4))$ komplex gyökeinek keresése valós ($x_0=1$) kezdőpontból

A komplex síkon minden gyökhöz tartozik egy vonzási tartomány, amely halmazból az iterációt indítva az adott gyökhöz konvergál a módszer. A teljes komplex síkhoz, ezenkívül hozzátartoznak azon ponthalmazok, ahonnan a módszer divergens. Érdekes kérdés, hogy meg tudjuk-e határozni a gyökökhöz tartozó vonzási tartományokat adott függvényekre. A másodfokú polinomokra a következő tételt tudjuk

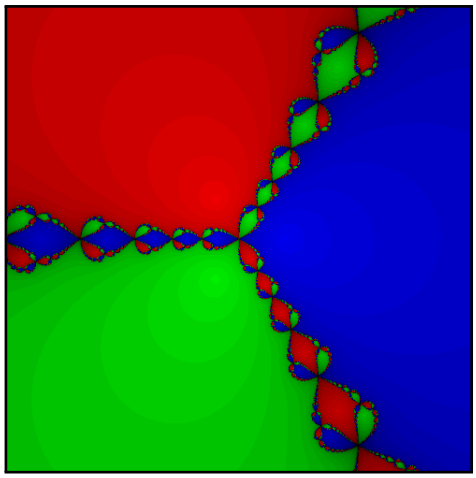
2.3.2. Tétel (Arthur Cayley, 1897). *Legyen p komplex másodfokú polinom, melynek α és β két különböző gyöke. Legyen L az α-t és β-t összekötő szakasz merőleges felezővonala. Ekkor, ha a Newton-módszert alkalmazzuk p-re, a gyökökhöz tartozó $B(\alpha)$ és $B(\beta)$ vonzási tartományok, az L egyenes által szétválasztott komplex félsíkok.*

Cayley nem tudta megállapítani, hogy mi teljesül nagyobb fokszámú polinomok esetén az azokon fellépő kaotikus viselkedés miatt. Ugyanis ehhez hasonló szabályos viselkedés nem figyelhető meg harmad vagy magasabb fokú polinomok esetén.

2.3.3. Tétel (Gauss-Lucas). *Legyen $p(z)$ egy nemkonstans komplex együtthatós polinom, $z_1, \ldots, z_n$ gyökökkel és $p'(z)$ $w_1, \ldots, w_{n-1}$ gyökökkel. Legyen $H = \{\sum_{i=1}^{n} c_i z_i \ 0 \leq c_i \leq 1, \ \sum_{i=1}^{n} c_i = 1\}$ p komplex gyökeinek konvex burka a komplex síkon. Ekkor $\forall i \ w_i \in H$.*

Mivel a Newton-módszer kiszámíthatatlanul viselkedik a polinom deriváltja gyökeinek a közelében, semmi biztosíték nincs arra, hogy a kezdőponthoz legközelebbi gyökhöz konvergál. Így míg a 2.3.2. tétel alapján erre az általánosításra számíthatnánk, a 2.3.3. tétel következményeként előfordulhat, hogy az egyik gyökhöz közel elhelyezkedő derivált gyöke, az egyik iterációs lépés során átsodorja a módszert egy másik gyökhöz közelebb.

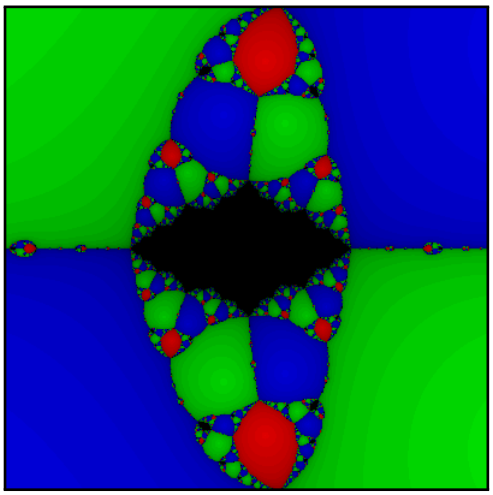
Valójában minden komplex differenciálható függvényre ezen halmazok határai egy fraktált, úgynevezett Julia halmazt határoznak meg. (2.5. ábra)



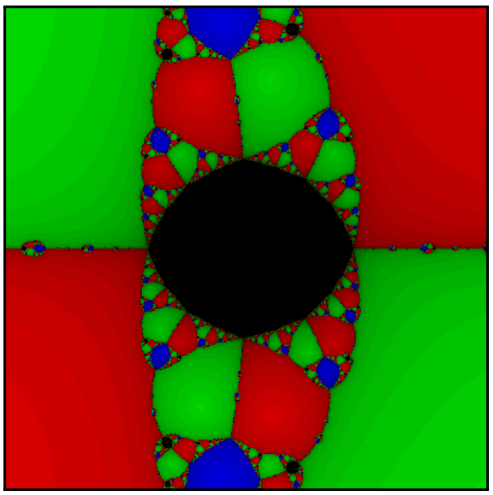
2.5. ábra. $x^3 - 1$ komplex gyökeihez tartozó halmazok a $\{a + bi : a, b \in [-5, 5]\}$ síkrészen. A piros részből indított iteráció $-1/2 + i\sqrt{3}/2$-höz konvergál, a zöld részről ennek komplex konjugáltjához, és a kék részről 1-hez. A sötétebb részekről lassabb a módszer.

Ismert olyan harmadfokú polinom, amelyeknek egy adott komplex számra, és annak egy bizonyos környezetéből választott tetszőleges pontra alkalmazott Newton-módszer

divergens. Az ilyen halmazokat fekete színnel szokás jelölni. Erre mutat két példát a 2.6. ábra.



(a) Az $f(x) = (x-1)(x+0{,}5+0{,}33i)(x+0{,}5-0{,}33i)$ polinom a $\{a+bi : a,b \in [-0{,}2; 0{,}2]\}$ halmazon.



(b) Az $f(x) = x^3 - 2x + 2$ polinom a $\{a+bi : a,b \in [-0{,}3; 0{,}3]\}$ halmazon.

2.6. ábra. Példa olyan komplex halmazokra, amelyekről egy adott polinomon a Newton-módszer divergens.

3. fejezet

Módosított módszer polinomokra

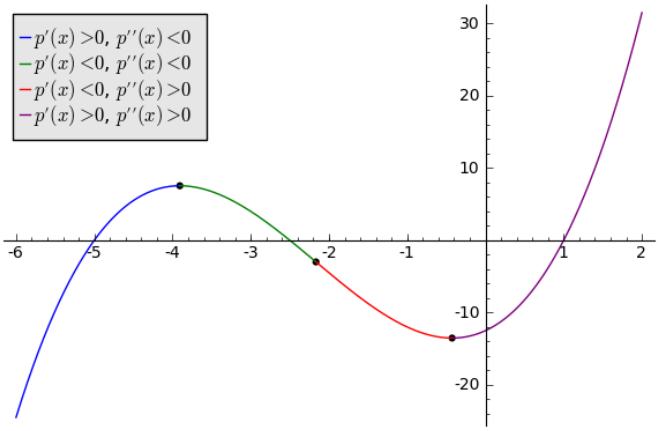
Adott a Newton-módszer, amit az előző fejezetben vizsgáltunk. Tudjuk, hogyha az x_k sorozat konvergens, akkor másodrendben konvergál a gyökhöz (2.2.2), sőt bizonyos feltételekkel monoton konvergenciát tudtunk biztosítani (2.2.4). Azonban sok esetben a módszer nem konvergens, ugyanis az említett tételek feltételei könnyen megsérülnek, ha a kezdőpontot nem választjuk a gyöknek egy kellően kis sugarú környezetéből. Ezt véletlenszerűen választott kezdőponttal nem tudjuk garantálni, így ilyenkor csak remélhetjük, hogy az iteráció az egyik lépés során véletlenül belelép egy megfelelő intervallumba, azzal biztosítva a további konvergenciát. Szeretnénk egy olyan módszert, amely ennek ellenére konvergens viszonylag gyorsan és sok helyről.

3.1. Motiváció

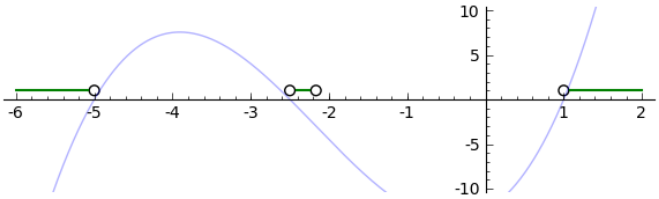
Vizsgáljuk meg, hogy egy polinom hogyan viselkedik a 2.2.4. tétel alapján, vagyis honnan indulva garantált a monoton konvergencia.

A tétel feltételei alapján a kezdőpontunk és a gyök közötti intervallumon sem a függvény deriváltja, sem a második deriváltja nem lehet nulla. Ez négy lehetőséget ad melyet a 3.1. ábrán szemléltettem. Mint az ábrán is látható, előfordulhat olyan eset, amikor egy ilyen intervallumon nincsen gyöke a polinomnak. Azonban amelyik intervallumon van, ott a gyök előtti, vagy utáni részről véve a kezdőpontunkat, a feltételek teljesülnek.

3.2. ábrán láthatjuk, hogy esetünkben a kezdőpontunknak alkalmas intervallum majdnem lefedi a számegyenest, és bár monoton és másodrendű konvergencia garantált, a két szélső intervallumban, ha távolról indulunk, a nagy abszolút értékű derivált miatt, előfordulhat, hogy lassú lesz a módszer.



3.1. ábra. Egy polinom felosztása az első és második deriváltjának viselkedése alapján



3.2. ábra. Megfelelő intervallumok a kezdőpont választáshoz 2.2.4. tétel alapján

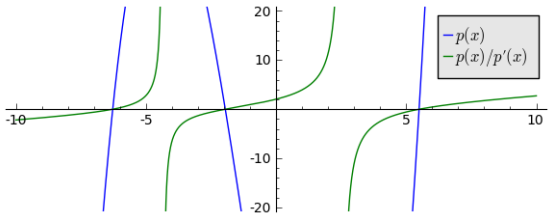
Ha egy polinomot leosztunk egy másik polinommal, és a két polinomnak nem egyezik meg valamelyik gyöke, akkor egy olyan függvényt kapunk amelynek a gyökei megegyeznek az eredeti polinoméval. Ugyanis legyen $p_1(x)$ a polinomunk amelyiknek a gyökeit keressük, és osszuk le $p_2(x)$ polinommal. Legyen x^* gyöke $p_1(x)$-nek. Ekkor

$$\frac{p_1(x^*)}{p_2(x^*)} = \frac{0}{p_2(x^*)} = 0$$

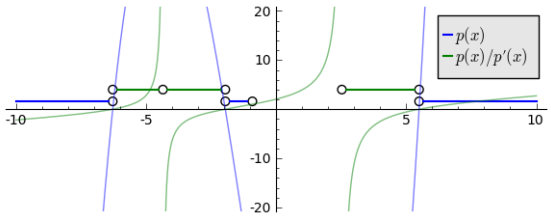
Felvetődik tehát az ötlet, hogy hajtsuk végre a Newton-módszert a $\frac{p_1(x)}{p_2(x)}$ egyenlet gyökeinek keresésére. Vegyük azt a speciális esetet, amikor $p_2(x) = p_1'(x)$. Ha $p_1(x)$-nek nincs többszörös gyöke, akkor a módosított egyenletünknek ugyanazok a gyökei.

3.2. A módszer

Ahogyan a 3.3(b). ábrán látható a két egyenlet konvergenciát garantáló intervallumai jól kiegészítik egymást. Ez alapján az az ötletünk támadhat, hogyha véletlenszerűen váltogatjuk az egyenletet, amire a módszert alkalmazzuk, jó eséllyel eljutunk az egyenletünk valamelyik gyökéhez.



(a) Egy polinom és a módosított egyenlete



(b) Megfelelő intervallumok a kezdőpont választáshoz 2.2.4. tétel alapján

3.3. ábra

Tehát adott p polinom, amelynek a gyökeit keressük. Ekkor vegyük azt az iterációt, aminek egy lépését 0,5 valószínűséggel a (3.1.) iterációval tesszük, és 0,5 valószínűséggel (3.2.) iterációval tesszük.

$$x_{k+1} = x_k - \frac{p(x_k)}{p'(x_k)} \tag{3.1}$$

$$x_{k+1} = x_k - \frac{\frac{p(x_k)}{p'(x_k)}}{\left(\frac{p(x_k)}{p'(x_k)}\right)'} = x_k - \frac{p(x_k)p'(x_k)}{p'(x_k)^2 - p''(x_k)p(x_k)} \tag{3.2}$$

A későbbiekben ezekre első (3.1.), és második (3.2.) iterációs lépésként, a módszerre pedig véletlen iterációként fogok hivatkozni.

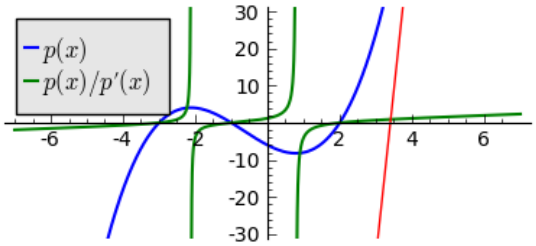
3.2.1. Példa. *Az alábbiakban egy olyan harmadfokú polinomon végzem el az iterációt, aminek az együtthatóit* -50 *és* 50 *között véletlenszerűen generáltam.*

$$p(x) = -49{,}6776x^3 + 36{,}1231x^2 - 1{,}1662x - 20{,}4780$$

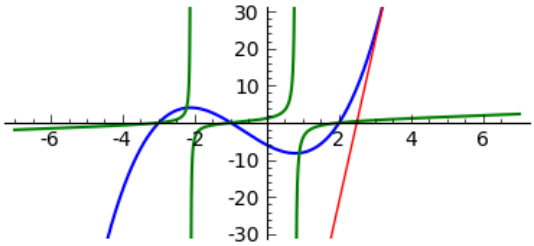
k	*iterációs lépés*	x_k	$\lvert f(x_k)\rvert$	$\lvert x_k - x^*\rvert$
0	-	$1+i$	$8.2695e+01$	$5.5794e-01$
1	*1.*	$0{,}78752 + 0{,}72275i$	$2{,}0696e+01$	$2{,}0946e-01$
2	*2.*	$0{,}60295 + 0{,}54236i$	$3{,}4869e+00$	$4{,}9152e-02$
3	*1.*	$0{,}64583 + 0{,}57250i$	$2{,}7337e-01$	$3{,}6089e-03$
4	*2.*	$0{,}64227 + 0{,}57184i$	$1{,}3524e-03$	$1{,}7911e-05$
5	*1.*	$0{,}64228 + 0{,}57183i$	$3{,}3361e-08$	$4{,}4183e-10$

Szükségesnek tartom megjegyezni, hogy bár az iterációs lépések felváltva követték egymást, ez kizárólag a véletlenszerűségből ered.

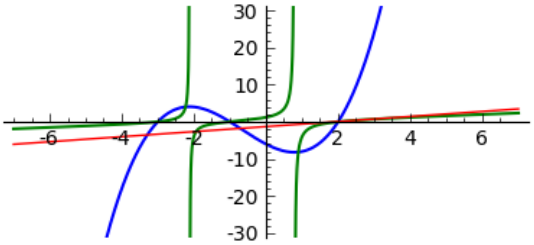
3.2.2. Példa. *Egy szemléletes példaként hajtsuk végre a módszert a következő polinomon:* $p(x) = x^3 + 2x^2 - 5x - 6$, $x_0 = 5$ *kezdőponttal.*



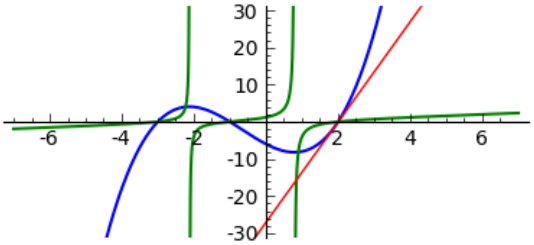
(a) Az első iterációs lépéssel lépünk



(b) Az első iterációs lépéssel lépünk



(c) Az második iterációs lépéssel lépünk



(d) Az első iterációs lépéssel lépünk

Az ábrán a következő lépések láthatók: $x_1 = 3{,}4$, $x_2 = 2{,}4891$, $x_3 = 1{,}9040$, $x_4 = 2{,}0053$. *Megfigyelhető az ábrákon, hogy a negyedik lépéssel elég közel kerültünk a gyökhöz, az ábrákat továbbfolytatni ezáltal fölösleges helyfoglalás lenne, viszont a következő lépés megtalálja az* $x^* = 2$ *gyököt. (3.3.1. megjegyzés)*

3.3. Elemzés

A dolgozat következő szakaszában az imént bevezetett módszert fogjuk több szempontból megvizsgálni. A vizsgálathoz több helyen a GNU Octave programnyelvet használom, ami szinte teljesen kompatibilis a MATLAB-bal, emellett ingyenes. A programok bemutatására ebben a fejezetben nem szánok sokat, csak éppen annyit, hogy érthető legyen az elvégzett művelet. A programok részletesebb bemutatását bővebben az A. függelékben végzem.

3.3.1. Megjegyzés. *Szükségesnek tartom megjegyezni a programokkal számolt közelítések hibájának mértékét. Minden program a* `Newton.m` *programot használja fel a Newton-módszer elvégzéséhez. Ennek a felhasználó az 5. argumentumában be tud állítani egy hibakorlátot, amelyet ha nem adunk meg akkor a* 0,001 *alapbeállítással dolgozik. A programokban is végig ezt az értéket használtuk. A hibakorlát az abszolút relatív közelítési hibát korlátozza vagyis akkor mondjuk, hogy megtalálta a gyököt, ha*

$$\frac{|x_i - x_{i-1}|}{|x_i|} < 0{,}001$$

Miért jó ez a leállási feltétel? Ha csak a Newton-módszer definíciójára gondolunk:

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

Láthatjuk, ha x_{i-1} közel van a gyökhöz, akkor az egyenlet jobb oldala közel van nullához (későbbi 3.3.3. állítás alapján), így a konvergencia tudatában a feltétel a gyök közelében tényleg biztosítja a leállást. Polinomok esetén az állítás megfordítása is igaz, azonban ez nem triviális. $|x_i|$ leosztásával nagyobb abszolút értékű számoknál nagyobb hibával dolgozunk, amivel azt érjük el, hogy nagyjából ugyanannyi helyes számjegyet kapjunk eredményeinkben annak nagyságrendjétől függetlenül. Ez természetesen nem minden esetben jó. Megoldandó feladat alapján eldönthető, hogy az abszolút közelítési hibát ($|x_i - x_{i-1}|$) használjuk helyette. Esetünkben ez nem lesz olyan fontos, mivel általában 0-hoz közeli gyökökkel fogunk számolni.

Egy iterációs eljárás vizsgálatakor elsősorban az érdekelhet minket, hogy konvergens-e a módszer, és ha igen, akkor milyen gyorsan konvergál. Ezek megvizsgálásához egy olyan módszernél, amelyben egy véletlen változó befolyásolja az eredményt, a statisztika módszereire lesz szükségünk.

3.3.2. Állítás. *Tegyük fel, hogy p polinomra és egy adott x^* gyökére teljesülnek a 2.2.2. tétel feltételei, p-nek x^* nem többszörös gyöke, és $p''(x)p(x) \neq p'(x)^2$. Ekkor 3.2 másodrendben konvergál a gyökhöz.*

Bizonyítás. A második iterációs lépés nem más, mint a Newton-módszer a p/p' egyenletre, aminek adott gyöke megegyezik p gyökével, így ha p/p'-re teljesülnek 2.2.2. tétel feltételei, akkor mivel p/p' adott gyökéhez másodrendben konvergál, így p gyökéhez is. p/p' folytonos azon a helyen kívül, ahol $p' = 0$. Ezáltal kiválaszthatók a megfelelő $[a, b]$ intervallumok a gyökök körül, amely nem tartalmazzák p' gyökhelyeit.

$m_1 > 0$-hoz azt kell belátnunk, hogy $(p/p')' \neq 0$

$$\left(\frac{p(x)}{p'(x)}\right)' = 1 - \frac{p''(x)p(x)}{p'(x)^2}$$

Tehát amennyiben $p''(x)p(x) = p'(x)^2$ teljesül valamilyen x-re, ott a függvényünk deriváltja 0. Ezért tettük fel, hogy ez nem teljesül. Ehhez létezik megfelelő $[a, b]$ intervallum, hiszen ahol $p/p' = 0$ ott a derivált 1.

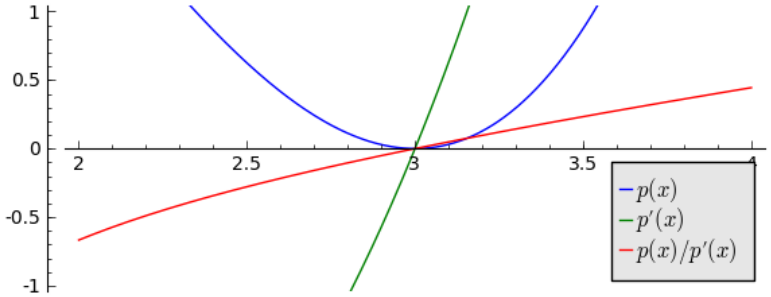
p/p' kétszer folytonosan differenciálható, ugyanis

$$\left(\frac{p(x)}{p'(x)}\right)'' = \left(1 - \frac{p''(x)p(x)}{p'(x)^2}\right)' = -\frac{p'''(x)p'(x)^2p(x) + p''(x)p'(x)^3 - 2p''(x)^2p'(x)p(x)}{p'(x)^4}$$

A számlálóban végzett folytonos függvények közötti műveletek nem befolyásolják a függvény folytonosságát, így az csak ott nem folytonos, ahol $p'(x) = 0$, vagyis p/p' $[a, b]$-n kétszer folytonosan differenciálható. $\square$

Tehát ha megfelelő a polinomunk akkor (a megfelelő helyeken) két másodrendű módszert használunk véletlenszerűen váltakozva.

Azonban ha a polinomnak van többszörös gyöke, akkor sem kizárt, hogy a második iterációs lépés megtalálja a gyököt. Vegyük például a $p(x) = x(x-3)^2$ polinomot melynek a 3 kétszeres gyöke.



3.4. ábra. $p(x) = x(x-3)^2$ polinom, annak deriváltja és a hányadosuk ábrázolása.

3.4. ábrán látható, hogy bár $p'(x)$-nek is gyöke a 3, annak környezetében abszolút értékben nagyobb értékeket vesz fel mint $p(x)$, ami miatt a hányados bár nem értelmezett 3-ban, de

$$\lim_{x\to 3} \frac{p(x)}{p'(x)} = \lim_{x\to 3} \frac{x(x-3)^2}{(x-3)(x-1)} = \lim_{x\to 3} \frac{x(x-3)}{(x-1)} = 0$$

Ez nekünk annyira pont elég, hogy a módszer, ahogyan közeledik 3-hoz úgy egy idő után megálljon a hibahatár elérésekor.

Ezt általánosabban is kijelenthetjük.

3.3.3. Állítás. *Legyen $p(x)$ polinomnak x^* (n-szeres) gyöke. Ekkor*

$$\lim_{x\to x^*} \frac{p(x)}{p'(x)} = 0$$

Bizonyítás. A L'Hospital-szabály alapján, ha létezik határérték, akkor

$$\lim_{x\to x^*} \frac{p(x)}{p'(x)} = \lim_{x\to x^*} \frac{p^{(n-1)}(x)}{p^{(n)}(x)} = \lim_{x\to x^*} \frac{0}{p^{(n)}(x)}$$

A polinomnak $n-1$ deriválás után még gyöke az x^*, ezért lett a számláló határértéke 0. Ahhoz, hogy az egész kifejezás határértéke is 0 legyen $p^{(n)}(x) \neq 0$ teljesülése szükséges. Ez

teljesül, ugyanis tudjuk, hogy a polinom foka legalább n. Ha a polinom foka nagyobb mint n, akkor az n. derivált különböző helyeken vesz fel 0 értéket. Ha a polinom foka n, akkor az n. derivált egy konstansfüggvény, melynek értéke $p(x)$ főegyütthatója megszorozva $n!$-sal, ami nem egyenlő 0-val. □

A további statisztikai elemzéshez különböző adatok generálására lesz szükségünk. Ezelőtt vizsgáljuk meg a módszerek konvergenciáját és annak idejét egy-két példában, amiből későbbiekben intuitív ki tudunk indulni. Érdemes lenne olyan példákat találni amelyeken működik a módszer és olyanokat is amelyeken nem, ezáltal össze tudnánk vetni, hogy mi befolyásolhatja a módszer helyességét.

3.3.1. Egy példán keresztül

Vegyünk egy harmadfokú polinomot és vegyünk olyan kezdőpontot amely elég távol van a gyöktől.

3.1. Listing. Bemenet

```
1 p=poly([1,6+3i,6-3i])
2 [gyok,it]=Newton(p,2e+4)
```

Az első sorban létrehozom a p polinomot melynek három gyöke 1, $6+3i$ és $6-3i$. A második sorban egy saját függvényt használok fel (A.1), amely a p polinomon végzi el a Newton-módszert $x_0 = 20000$ kezdőponttal.

3.2. Listing. Eredmény

```
1 gyok =   1.00000
2 it =   29
```

Vagyis a Newton-módszer 29 lépés alatt találta meg az $x^* = 1$ gyököt. Nézzük a Newton-módszert p/p' egyenleten!

3.3. Listing. Bemenet

```
1 [gyok,it]=Newton(p,2e+4,0,2)
```

3.4. Listing. Eredmény

```
1 gyok =   1.0000
2 it =   9
```

Vagyis a módosított egyenletünkkön végrehajtott módszer gyorsabban találta meg ugyanazt a gyököt. Nézzük a véletlen iterációt!

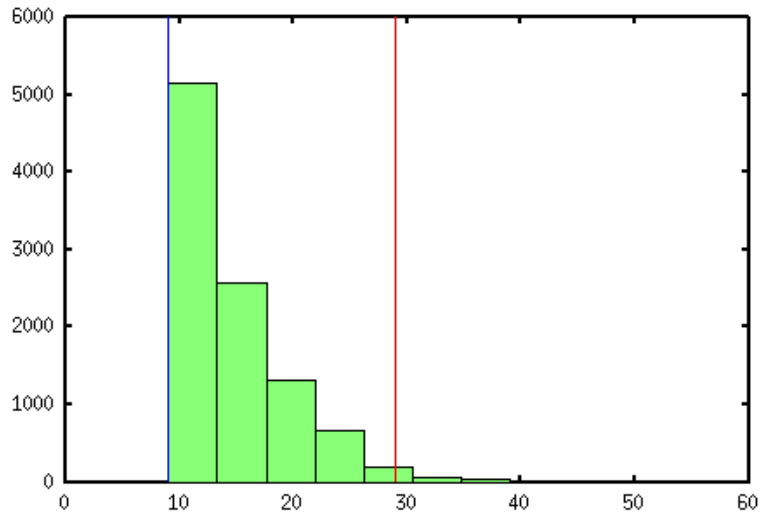
3.5. Listing. Bemenet

```
x=zeros(1e+4,1);
for i=1:(1e+4)
    [gyok,it]=Newton(p,2e+4,0,3);
    x(i)=it;
end
m=mean(x)
s=std(x)
figure('Position',[10 10 450 300])
hax=axes;
hold on
hist(x)
line([29 29],get(hax,'YLim'),'Color',[1 0 0]) % Szukseges lepesszam p-n
line([9 9],get(hax,'YLim'),'Color',[0 0 1]) % Szukseges lepesszam p/p'-n
```

Elvégzem 10000-szer a módosított módszert ugyanazon bemenő paraméterekkel, majd kiszámolom a szükséges lépések számának átlagát és szórását, ezenkívül egy hisztogramon is ábrázolom, hogy lássuk az eloszlását (3.5. ábra).

3.6. Listing. Eredmény

```
m =  14.298
s =  4.9452
```



3.5. ábra. 3.5. listázás 11. sorának eredménye, amely szemlélteti a véletlen iterációhoz szüksges lépések számának eloszlását, az adott esetben. A piros vonal a Newton-módszer szükséges lépésszámát jelzi p-n, a kék vonal p/p'-n

A lépések száma átlagosan megfeleződött, ami egyelőre jó eredménynek tűnik, bár megjegyzem, hogy nem vizsgáltam, hogy minden futtatás elért-e egy gyököt. Ami a hisztogramról elsőre feltűnik, hogy nagyobb a valószínűsége, hogy hamar véget ér az iteráció, és egyszer sem volt gyorsabb mint a két iterációs lépésből a gyorsabb külön futtatva, viszont előfordult,

hogy lassabb mint a lassabb iteráció. Persze ebből még ne vonjunk le semmit. Futtassuk le a fenti kódokat egy távoli komplex kezdőponttal: $x_0 = 20000 - 10000i$

3.7. Listing. Eredmény

```
gyok =  6.0000 - 3.0000i
it =  26
m =  14.319
s =  4.9417
```

Az első két sor a Newton-módszer eredménye, a harmadik, negyedik a véletlen iterációé. Az előzőtől lényeges különbséget egyelőre nem látunk, érdemes lenne sokkal több pontot egyszerre vizsgálni.

Nézzük meg a véletlen iteráció milyen eloszlásban iterált a gyökökhöz.

3.8. Listing. Bemenet

```
r=roots(p); % [6.0000 + 3.0000i,6.0000 - 3.0000i,1.0000 + 0.0000i]
n=length(r)+1;
y=zeros(1,n);
for j=1:(1e+4)
    [gyok,it]=Newton(p,1e+4*(2-i),0,3);
    [MINE,MINH]=min(abs(r-gyok));
    if MINE<0.001
        y(MINH)=y(MINH)+1;
    else
        y(n)=y(n)+1;
    end
end
y/1e+4 % vagy abrzolashoz: x=1:n; bar(x,y/1e+4);
```

A GNU Octave beépített függvényével kiszámolom a polinom gyökeit (ami egyébként a Frobenius kísérő mátrix sajátértékeivel határozza meg őket), majd elvégzem 10000-szer a véletlen iterációt, és eltárolom melyik esetben melyik gyökhöz konvergált a módszer. Végül pedig kiszámolom a relatív gyakoriságokat.

3.9. Listing. Eredmény

```
ans =

   0   0   1   0
```

Az eredmény szerint a módszer 1 valószínűséggel tart a 3. gyökhöz, ami az $1 + 0i$. Ez azt jelenti, hogy az előzőekben kapott eredményeknél, de legalábbis a komplex kezdőpontnál, a véletlen iteráció mindig elérte a gyököt, ami ráadásul nem egyezik meg a Newton-módszer által megtalált gyökkel. A 4. elem annak a relatív gyakorisága, hogy nem konvergál a módszer.

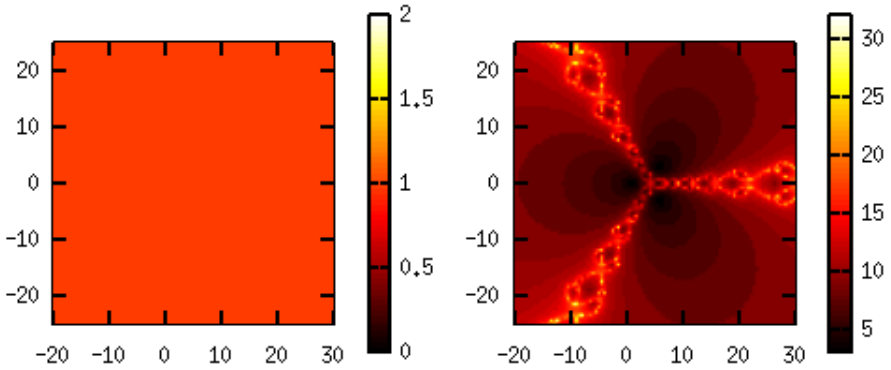
Az előző példákra egy általánosabb módszert is felépíthetünk, amely egy polinomnál egyszerre több pontból is megvizsgálja, hogy konvergál-e a módszer és átlagosan milyen gyorsan. Továbbá érdekes lehet a véletlen iteráció és az eredeti módszerek konvergencia sebességének összefüggése, mint minden véletlen iteráció konvergencia sebessége lassabb mint valamelyik módszeré, vagy van-e esély arra, hogy ahonnan semelyik módszer nem konvergált, ott a véletlen iteráció igen.

A következőkben a A.2. függelékben ismertetett függvénnyel szemléltetem az adott polinomon a véletlen iteráció jóságát.

3.10. Listing. Bemenet

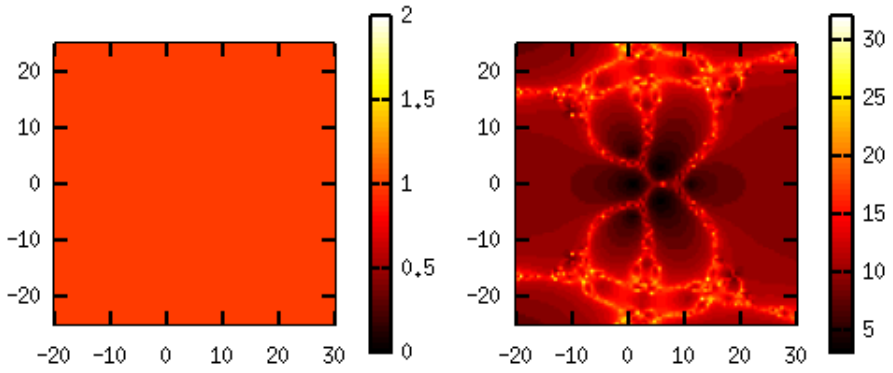
```
1 IteracioAbraKomplex(p,-20-25i,30+25i,100,1,1)
2 IteracioAbraKomplex(p,-20-25i,30+25i,100,1,2)
3 IteracioAbraKomplex(p,-20-25i,30+25i,100,100,3)
```

A hívott program az adott polinomhoz készít egy szemléltető ábrát a $\{a + bi : a \in [-20, 30], b \in [-25, 25]\}$ komplex számhalmazon az adott módszerekhez. Bár ez a halmaz nem tartalmaz a gyöktől távoli számokat, mégis érdekesebbnek tűnt ezen összehasonlítást végezni. Az első ábra a sikeresen gyökhöz ért iterációk számát szemlélteti. Ez az első két esetben 0 vagy 1, így csak az utolsó esetben érdekes, ahol a véletlen változó miatt előfordulhat, hogy egy pontból egyszer konvergens a módszer, egyszer nem, itt 100 ismétlést végzünk. A második ábra a sikeres esetekhez szükséges iterációs lépések számát mutatja. A listázás három sorával mindkét ábrát kirajzoljuk mindegyik módszerhez $100 \cdot 100$ kezdőpontból indulva.



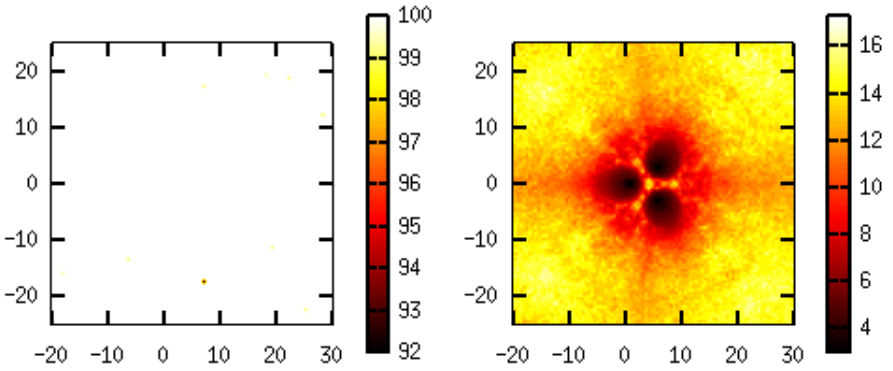
3.6. ábra. 3.10. listázás 1. sorának eredménye, amely szemlélteti a Newton-módszer konvergenciáját az adott polinomon, és az ahhoz szükséges lépések számát. A módszer mind a $100 \cdot 100$ kezdőpontból konvergens volt

Láthatjuk, hogy a különböző egyenletekre futtatott Newton-módszer (3.6. ábra és 3.7. ábra) máshol éri el a kritikusabb, lassabb pontjait. Amik a gyökök vonzási tartományának



3.7. ábra. 3.10. listázás 2. sorának eredménye, amely szemlélteti a Newton-módszer konvergenciáját a p/p' egyenleten, és az ahhoz szükséges lépések számát. A módszer mind a $100 \cdot 100$ kezdőpontból konvergens volt.

határain helyezkednek el.

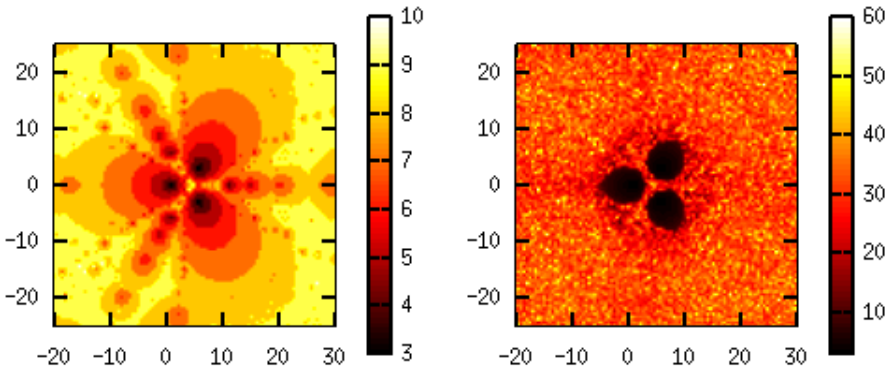


3.8. ábra. 3.10. listázás 3. sorának eredménye, amely szemlélteti a véletlen iteráció konvergenciáját az adott polinomon, és az ahhoz szükséges lépések átlagos számát. A módszer $100 \cdot 100$ kezdőpontból indulva pár pontból nem konvergált minden esetben, de a legrosszabb esetben is 0,92 valószínűséggel igen. Cserébe a leglassabb esetben is átlagosan 17 lépés alatt megtalált egy gyököt.

Bár előfordulhat, hogy a képek alapján nem megfelelő következtetéseket vonunk le, érdemes szemügyre venni az eredményeket mielőtt az adatokkal foglalkoznánk. Ami feltűnő, hogy a véletlen iterációnak a leglassabb esetében is átlagosan csak 17 iterációs lépésre volt szüksége, míg a különböző egyenletekre külön néhol 30 fölé emelkedik ez a szám. Ha a felosztást finomítjuk az ábrákon látható tendencia mentén megtalálna olyan kezdőpontokat, ahonnan lassabb az iteráció, míg a véletlen iteráció esetén a máshol elhelyezkedő kritikus kezdőpontok miatt a szükséges lépések száma, meglátásom szerint, nem fog lényegesen megnőni. Viszont megfigyelhető, hogy a 3.6. és 3.7. ábráktól eltérően itt az ábra nagy

része világos, vagyis a pontok nagy részénél szükség is volt 13-17 lépésre, míg a különálló módszereknél a 10 lépéshez tartozó árnyalatok dominálnak.

A.2.1. megjegyzés alapján rajzoljuk ki a véletlen iterációhoz szükséges lépések számának minimumát és maximumát. (3.9. ábra)



3.9. ábra. A két ábra nem tartalmazza a sikeres iterációk számát, mivel azok megegyeznek a 3.8. ábrán találhatóval. A bal oldali ábra a szükséges iterációs lépések minimumát, a jobb oldali ábra a maximumát szemlélteti.

A képekből arra következtethetünk, hogyha a módszert nem kellő közelről indítjuk a gyökhöz, akkor a szükséges lépésszám kis valószínűséggel, de akármilyen nagyra megnőhet, hiszen a maximumot szemléltető ábra elég homogénnek tűnik.

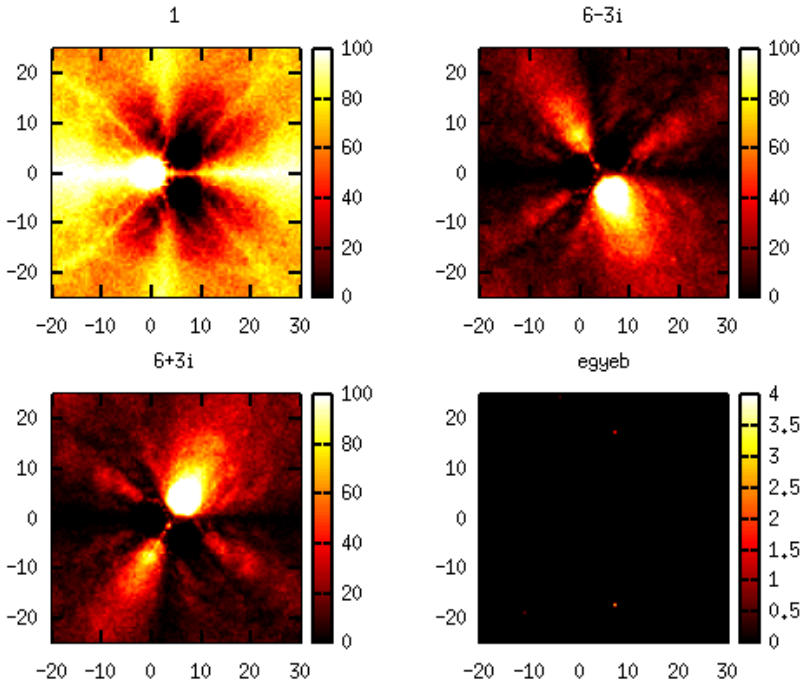
Azt, hogy az adott pontokból indított véletlen iteráció milyen eloszlásban iterál a gyökökhöz szintén egy ábrával a legegyszerűbb szemléltetnünk. Az ehhez szükséges programot a A.2. függelékben részletezem. Ez a Newton-módszernél a gyökökhöz tartozó vonzási tartományok megfelelője a véletlen iterációnál.

3.11. Listing. Bemenet

```
1 KonvergenciaAbraKomplex(p,-20-25i,30+25i,100,100,3)
```

Bár az eredmény a 3.10. ábrán érdekes szabályosságot mutat, sok mindent erről a részletesebb ábráról sem tudunk megállapítani. Feltünő viszont, hogy a valós gyökhöz mennyivel többször konvergál a módszer.

Amennyiben a A.2. program által generált adatokat nem kirajzoljuk, hanem egy változóba mentjük, úgy további elemzéseket tudunk végrehajtani. Legyen 3.10. listázás eredménye a `zs` változóba mentve a következőképpen. `zs` 6 db mátrixot tartalmaz. Az első mátrix (`zs(:,:,1)`) a Newton-módszerhez szükséges lépések száma, a második a Newton-módszerhez szükséges lépések száma p/p' egyenleten, a harmadik a véletlen iterációhoz szükséges lépések számának minimuma, negyedik az átlaga, ötödik a maximuma.



3.10. ábra. 3.11. listázás eredménye. Adott gyökhöz tartozó ábra minél világosabb egy adott pontban, annál többször konvergál a véletlen iteráció hozzá. Míg a véletlenül kiválasztott kezdőpontunknál véletlenül 1 valószínűséggel konvergált egy adott gyökhöz a módszer, itt látható, hogy ez nincs mindig így.

Vizsgáljuk meg lehet-e a véletlen iteráció gyorsabb, mint valamelyik egyenleten alkalmazott Newton-módszer.

3.12. Listing. Bemenet

```
t=min(zs(:,:,1),zs(:,:,2))>zs(:,:,3);
[x y]=meshgrid(linspace(-20,30,100),linspace(-25,25,100));
figure
pcolor(x,y,double(t))
axis equal tight;
colorbar;
colormap hot;
shading flat;
sum(sum(t))/10000
```

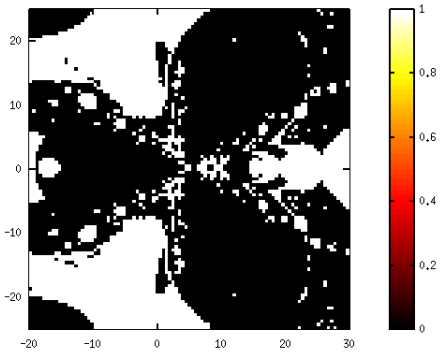
Ezzel kirajzoltuk, hogy hol, és megszámoltuk, hogy milyen arányban lett a véletlen iteráció gyorsabb mindkét eljárásnál.

3.13. Listing. Eredmény

```
ans =  0.32470
```

Ez egész jó eredmény, de figyelembe kell venni, hogy minden pontban 100 futtatásnak a

minimumát vettük, vagyis nem azt jelenti, hogy 1/3 az esélyünk arra, hogy gyorsabb lesz a módszerünk az adott síkrészen.



3.11. ábra. 3.12. listázás eredménye. A fehér mezőkben található komplex számokból indított véletlen iterációk gyorsabbak lettek mint külön-külön alkalmazva az első és második iterációs lépéseket. Ami nem egyezik meg a 3.5. ábra alapján várt eredménnyel. Ezek a pontok a adott síkrészen a $100 \cdot 100$ pontnak a 32,47%-át teszik ki.

A 3.5. listázás eredményéből úgy tűnik, hogy a véletlen iteráció szükséges lépésszámának az átlaga a gyorsabb iterációs lépés szükséges lépésszámához van közelebb. Ezt az elmentett adatokból meg tudjuk vizsgálni.

3.14. Listing. Bemenet

```
1 t=abs(min(zs(:,:,1),zs(:,:,2))-zs(:,:,4))<abs(max(zs(:,:,1),zs(:,:,2))-
      zs(:,:,4));
2 sum(sum(t))/10000
```

Az 1. sorban létrehozok egy olyan mátrixot melynek elemei logikai változók: igaz, ha a gyorsabb iterációs lépéshez, hamis ha a lassabb iterációs lépéshez van közel az átlag. a 2. sorban kiszámoljuk az igazok arányát.

3.15. Listing. Eredmény

```
1 0.10780
```

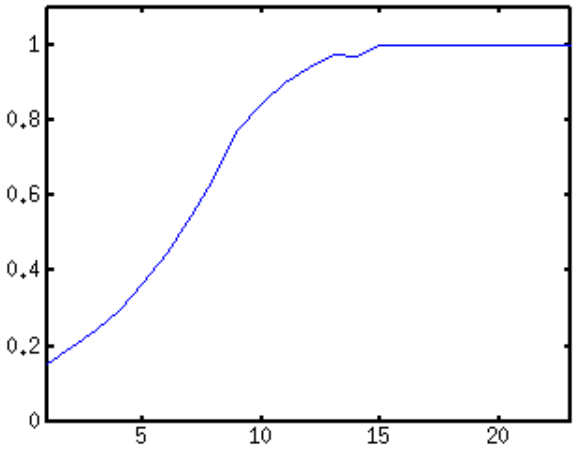
Eszerint a hivatkozott eredmény félrevezető volt. Vizsgáljuk meg, hogy mikor fordul elő, hogy a lassabb iterációs lépéshez közelít az átlag.

3.16. Listing. Bemenet

```
1 A=abs(min(zs(:,:,1),zs(:,:,2))-zs(:,:,4))<abs(max(zs(:,:,1),zs(:,:,2))-
      zs(:,:,4));
2 t=max(zs(:,:,1),zs(:,:,2))-min(zs(:,:,1),zs(:,:,2));
3 x=1:23;
4 y=zeros(1,23);
5 for i=x
```

```
6       B=t>i;
7       pAB=sum(sum(A & B))/10000;
8       pB=sum(sum(B))/10000;
9       y(i)=pAB/pB;
10  end
11  plot(x,y)
```

Megnéztük, hogy mi a valószínűsége annak, hogy a minimumtól való távolság kisebb mint a maximumtól való távolság, feltéve, hogy a minimum és a maximum távolsága nagyobb mint x, ahol $x \in [1, 23]$ egész szám. Ezt minden x-re kirajzoltuk.



3.12. ábra. 3.16. listázás eredménye, mely azt ábrázolja, hogy adott x-re mekkora a feltételes valószínűsége, hogyha az első és második iterációs lépéshez szükséges lépésszámok különbsége nagyobb mint x, akkor a véletlen iterációhoz szükséges lépésszámok átlaga a gyorsabb iterációs lépésnek a szükséges lépésszámához van közel. Az ábra szerint $x = 15$-től ennek feltételes valószínűsége 1.

Eszerint ez a tendencia a gyöktől távol vagy megváltozik, vagy az először véletlenül kiválasztott pontunk éppen olyan pont volt, ahol a két iterációs lépéshez szükséges lépésszám között elég nagy volt a különbség. Valószínűnek tartom, hogy mindkettő igaz, és a megváltozott tendencia nem más, mint a gyöktől távol a két iterációs lépéshez szükséges lépésszám egymástól való eltávolodása, ami egyébként intuitív logikus, hiszen két különböző iterációs lépés miért lenne mindenhonnan nagyjából ugyanolyan gyors.

3.3.2. Érdekes esetek

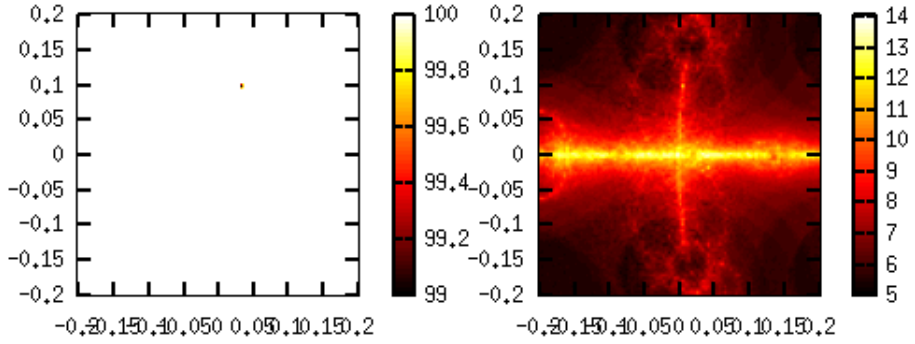
Érdekes esetnek tartom amikor a két iterációs lépés közül legalább az egyik divergens egy adott síkrészen. Vizsgáljuk meg a véletlen iteráció ilyen esetekben, hogyan viselkedik.

Vegyük az egyik példát amit 2. fejezet végén a komplex esetnél bemutattam, és nézzük meg az adott tartományokon milyen hányszor és milyen gyorsan konvergál.

3.17. Listing. Bemenet

```
1 IteracioAbraKomplex(poly([1,-0.5-0.33i,-0.5+0.33i]),-0.2-0.2i,0.2+0.2i
    ,100,100,3)
```

A $p(x) = (x-1)(x+0{,}5+0{,}33i)(x+0{,}5-0{,}33i)$ polinomot az egyszerűség kedvéért ugyanazon a halmazon ábrázoltam mint 2.6. ábrán.



3.13. ábra. 3.17. listázás eredménye, melyről látható, hogy a véletlen iteráció olyan helyről is konvergálhat, ahol az első iterációs lépés nem.

3.18. Listing. Bemenet

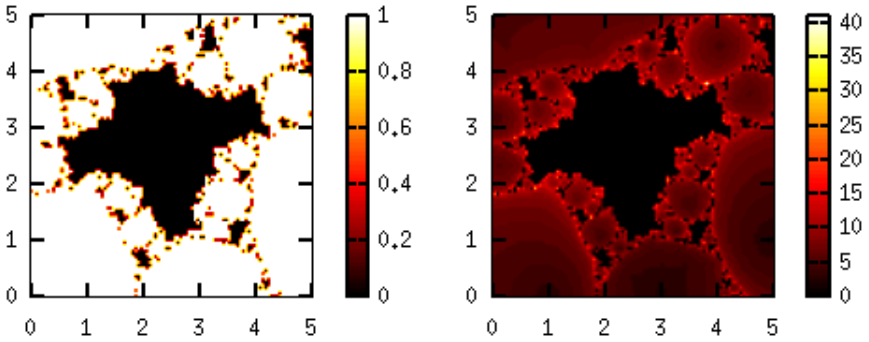
```
1 IteracioAbraKomplex(poly([1,3,5+j]),0,5+5j,100,1,2)
2 IteracioAbraKomplex(poly([1,3,5+j]),0,5+5j,100,100,3)
```

Ábrázoljunk a $p(x) = (x-1)(x-3)(x-5-i)$ polinomot a $\{a+bi : a,b \in [0,5]\}$ síkrészen, ahol a második iterációs lépés egy nagyobb ponthalmazon divergens, és nézzük meg ugyanitt a véletlen iteráció 100 futtatásából a sikeres iterációk számát és a szükséges lépések számának átlagát. (3.14. ábra)

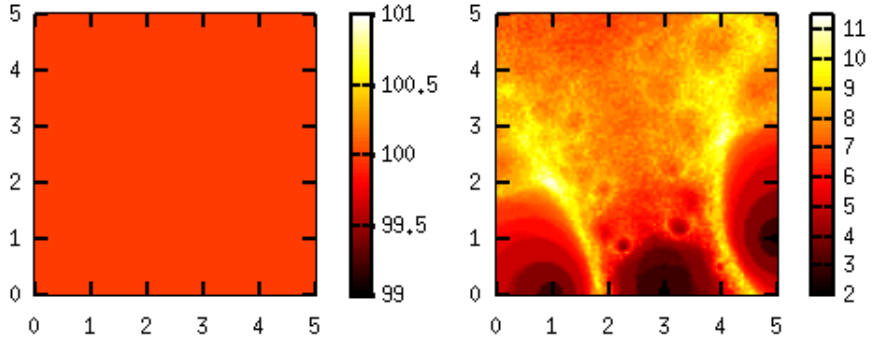
3.3.3. Ellenpélda

Az előzőekben bemutattunk egy példát, amin a véletlen iteráció viszonylag jól viselkedett, azonban könnyen találhatunk olyan példát, ahol a módszer meglehetősen rosszul viselkedik. A következőkben egy ilyet vizsgálunk.

Vegyük az x^3-1 polinomot, melyet korábban már ábrázoltam a 2.5. ábrán. Ennek gyökei a következők: $1, -1/2+i\sqrt{3}/2, -1/2-i\sqrt{3}/2$. Ez a gyökök elhelyezkedését nézve hasonló az előzőekben vizsgált polinomhoz, azonban itt a gyökök a komplex síkon egy szabályos



(a) Az második iterációs lépéssel



(b) A véletlen iterációval

3.14. ábra. 3.18. listázás eredménye, melyről látható, hogy a véletlen iteráció olyan helyről is konvergálhat, ahol a második iterációs lépés nem.

háromszög csúcsain helyezkednek el. E hasonlóság miatt érdekes összehasonlítani a két példát.

3.19. Listing. Bemenet

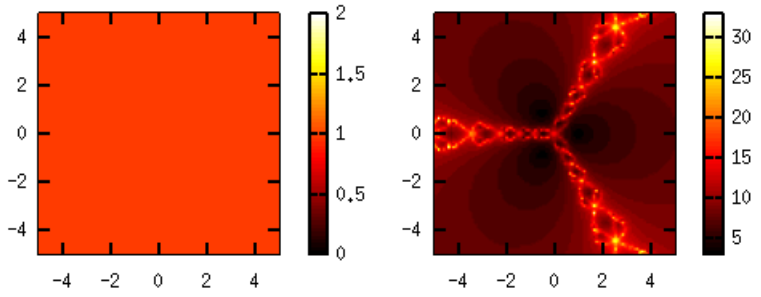
```
1 p=[1 0 0 -1];
2 IteracioAbraKomplex(p,-5-5i,5+5i,100,1,1)
3 IteracioAbraKomplex(p,-5-5i,5+5i,100,1,2)
```

Lefuttatjuk a már ismert programunkat amely kirajzolja a szükséges iterációs lépések számát az adott módszerekhez a $\{a + bi : a, b \in [-5, 5]\}$ síkrészen. Láthatjuk, hogy a két iterációs lépés külön-külön minden vizsgált kezdőpontból konvergens volt, azonban a véletlen iterációnál ez megváltozik.

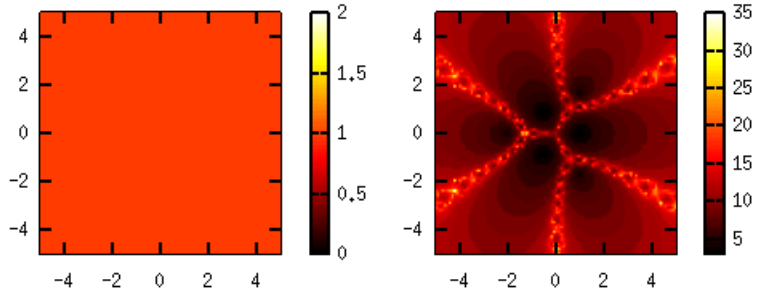
3.20. Listing. Bemenet

```
1 KonvergenciaAbraKomplex(p,-5-5i,5+5i,100,100,3)
```

Kirajzoljuk a gyökökhöz tartozó vonzási tartományok véletlen iterációhoz tartozó megfelelőjét.

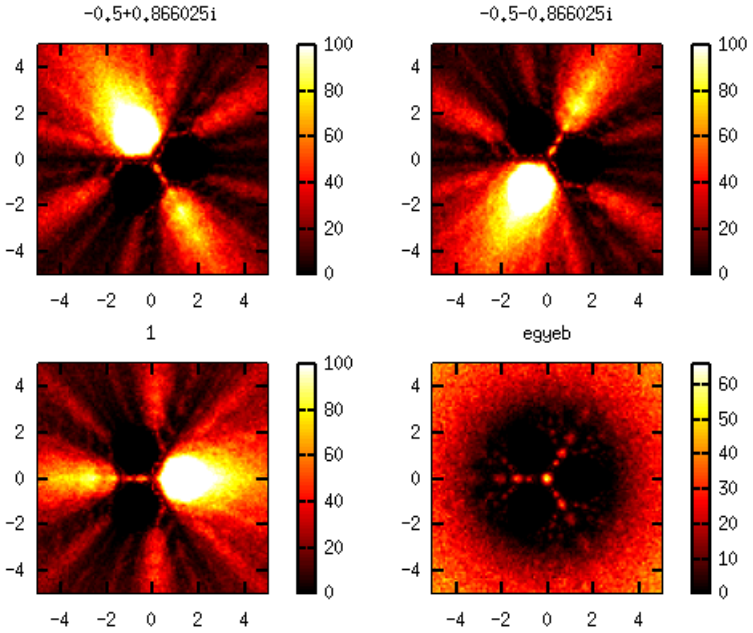


(a) Az első iterációs lépéssel



(b) A második iterációs lépéssel

3.15. ábra. 3.19. listázás eredménye



3.16. ábra. 3.20. listázás eredménye, mely azt ábrázolja, hogy melyik komplex számból melyik gyökhöz hányszor konvergált a véletlen iteráció. Látható a különbség 3.10. ábrától. Míg a jelenlegi ábrán minden gyökhöz azonos ábra tartozik egy megfelelő 120 fokos forgatás erejéig, addig a másik ábrán a valós gyök magához vonzotta azokat az iterációkat amik most az egyéb halmazba estek.

Az eredmény az eddigiek alapján meglepő lehet, hiszen a polinomunkon a véletlen iteráció annak három gyökéhez közeli intervallumon is már rosszabbul viselkedik mint azt elvárnánk, ráadásul egy olyan polinomon, aminek gyökei hasonlóan helyezkedtek el egymáshoz viszonyítva, a módszer meglehetősen jól viselkedett. A különbség az volt, hogy az előző polinomunk gyökei nem egy szabályos háromszögben helyezkedtek el a komplex síkon, míg most igen. Felvetődik tehát a kérdés, hogy vajon a szabályos háromszöggel van-e a baj, és ha igen akkor miért rontja el a módszerünket.

Futtassunk le egy kis programot amely véletlenszerűen generál 3 gyököt az $\{a+bi : a,b \in [-10,10]\}$ halmazból, és nézzük meg ezeken hogyan viselkedik a módszer. Ehhez szükség van a A.2. program kisebb módosítására: adjuk a z mátrixot visszatérési értékül. A z mátrix tartalmazza, hogy adott pontból hányszor konvergált gyökhöz az iteráció.

3.21. Listing. Bemenet

```
k=0;
for i=1:100
    gyok=(rand(6,1)*(20))-10;
    x=IteracioAbraKomplex(poly([gyok(1)+gyok(2)*j,gyok(3)+gyok(4)*j,gyok
        (5)+gyok(6)*j]),-100-100j,100+100j,10,10,3);
    if sum(sum(x))==10^3
        k=k+1;
    end
end
k/100
```

A program 6 valós számot generál -10 és 10 között, amiből létrehoz 3 komplex számot, ezekből létrehoz egy polinomot, és lefuttatja rá a A.2. programot, a fent említett módosítással. A program a $\{a+bi : a,b \in [-100,100]\}$ halmazon fut, mivel az ábrán az figyelhető meg, hogyha a véletlen iteráció rossz, akkor a gyököktől távolabb kezd el romlani. $10 \cdot 10$ pontból indítjuk a véletlen iterációt, amit 10-szer megismétel adott polinomra, majd az egészet még 100-szor, vagyis 100 ilyen polinomra.

3.22. Listing. Eredmény

```
ans =  0.98000
```

Vagyis a polinomok 98%-án minden esetben annyiszor érte el a gyököt a módszer, ahányszor futtattuk. Most generáljunk olyan polinomokat amiknek a gyökei szabályos sokszögként helyezkednek el.

3.23. Listing. Bemenet

```
k=0;
for i=1:100
    gyok=(rand(4,1)*(20))-10;
```

```
4       gyok(5:6)=[cos(pi/3),-sin(pi/3);sin(pi/3),cos(pi/3)]*[gyok(3)-gyok
            (1);gyok(4)-gyok(2)]+[gyok(1);gyok(2)];
5       x=IteracioAbraKomplex(poly([gyok(1)+gyok(2)*j,gyok(3)+gyok(4)*j,gyok
            (5)+gyok(6)*j]),-100-100j,100+100j,10,10,3);
6       if sum(sum(x))==10^3
7           k=k+1;
8       end
9   end
10  k/100
```

A program 4 valós számot generál -10 és 10 között, amiből létrehoz két komplex számot. Ezeket vektorként alkalmazva egy forgásmátrix segítségével létrehozzuk a harmadik gyököt, majd az előzőekben ismertetett módon folytatjuk az eljárást.

3.24. Listing. Eredmény

```
1   ans = 0
```

Látható, ha céltudatosan olyan polinomot generálunk amelynek gyökei a komplex síkon szabályos háromszögként helyezkednek el, akkor ez az eredmény nemcsak drasztikusan lecsökken, de egyenesen egy ilyen típusú polinomra sem volt konvergens minden esetben amikor futtattuk rajta a módszert. Persze ez egy erős feltétel, de láthatjuk, hogy a véletlenszerűen elhelyezkedő gyököknél ekkor is magas volt a százalék. Ha a 6. sort átírjuk a következőre:

```
6   if sum(sum(x))>=10^3*0.5
```

ami azt jelenti, hogy egy adott polinomnál csak azt szeretnénk, hogy a rajta indított véletlen iterációk fele legyen konvergens (az adott feltételek mellett), akkor az eredmény 0,12-re emelkedik.

Érdemes megvizsgálni negyedfokú polinomok körében is ezt a viselkedést. Helyettesítsük 3.21. listázás 3-4. sorát a következővel:

```
3   gyok=(rand(8,1)*(20))-10;
4   x=IteracioAbraKomplex(poly([gyok(1)+gyok(2)*j,gyok(3)+gyok(4)*j,gyok(5)+
        gyok(6)*j,gyok(7)+gyok(8)*j]),-100-100j,100+100j,10,10,3);
```

amivel 4 darab komplex gyökkel rendelkező polinomokra futtatjuk a módszert.

3.25. Listing. Eredmény

```
1   ans =   0.97000
```

Hasonlóan jó eredményt kapunk. Most generáljunk szabályos négyszögben elhelyezkedő gyököket.

3.26. Listing. Bemenet

```
k=0;
for i=1:100
    gyok=(rand(4,1)*(20))-10;
    elt=[0,-1;1,0]*[gyok(3)-gyok(1);gyok(4)-gyok(2)];
    gyok(5:6)=elt+[gyok(1);gyok(2)];
    gyok(7:8)=elt+[gyok(3);gyok(4)];
    x=IteracioAbraKomplex(poly([gyok(1)+gyok(2)*j,gyok(3)+gyok(4)*j,gyok
        (5)+gyok(6)*j,gyok(7)+gyok(8)*j]),-100-100j,100+100j,10,10,3);
    if sum(sum(x))==10^3
        k=k+1;
    end
end
k/100
```

Mint az előbb, egy eltolásvektort hozunk létre, a derékszögű forgásmátrix segítségével, amivel létrehozunk 2 pontból 4 szabályos négyszögben elhelyezkedőt.

3.27. Listing. Eredmény

```
ans =  0
```

Az eredmény megfelel a harmadfokú polinomokon tapasztalttal. Ha itt is módosítjuk, hogy adott polinomra csak az indított iterációk 50%-a legyen konvergens akkor az eredmény 0,64-re emelkedik. Ez nagyobb emelkedés mint amit harmadfokú esetben láthattunk.

Ezáltal úgy sejtjük, hogy ha a egy polinom gyökei szabályos sokszögként helyezkednek el a komplex síkon a módszer, a gyököktől megfelelően távolról indulva, pozitív valószínűséggel divergál.

4. fejezet

Összefoglalás

Dolgozatom során egy olyan iterációs módszer vizsgálatával foglalkoztam, amely véletlenszerűen lép két iterációs lépés segítségével. A két iterációs lépés igazából nem más, mint az egyenletünkön, és annak egy olyan módosításán végzett iterációs módszer, amelynek gyökei megegyeznek az eredeti egyenletünkével. Ennek a koncepciónak azt a speciális esetét vizsgáltam amikor a két egyenleten a Newton-módszert hajtom végre. Ehhez szükség volt a Newton-módszer alapos bemutatására és megismerésére.

A második fejezet során láthattuk, hogy a Newton-módszer milyen egyszerű koncepción alapszik, és ha bizonyos feltételek teljesülnek és a módszer konvergens, akkor a módszer másodrendben tart a gyökhöz. Megállapítottuk, milyen feltételek szükségesek a konvergenciához, és azt is, hogy bizonyos feltételekkel monoton konvergenciát tudunk biztosítani. Megvizsgáltuk, a komplex gyökökkel rendelkező egyenletekre mi a geometriai meggondolás, melyből kiderült, hogy a Newton-módszer egy gradiens módszer $|f|$ abszolút érték függvényre nézve. A komplex síkon minden gyökhöz tartozik egy vonzási tartomány, melyek másodfokú polinomok esetén két félsíkot, magasabb fokú polinomok és egyéb komplex differenciálható függvények esetén különböző fraktálokat határoznak meg.

A következőkben a módszerünket polinomokra specializáltuk, amelyeknél a módosított egyenletet úgy állítottuk elő, hogy az eredeti polinomot leosztottuk egy másikkal, amire a saját deriváltját választottuk. Megmutattuk, hogy egy-két plusz feltétellel a második iterációs lépés is másodrendben konvergens, és még a többszörös gyök sem biztos, hogy megakadályozza a konvergenciát. Ezután a módszer elemzéséhez általam írt számítógépes programokat használtunk fel. A következő fontosabb eredményeket tapasztaltuk: A módszer gyorsasága átlagosan a két iterációs lépés gyorsasága között mozog, azonban van esélyünk arra is, hogy gyorsabb, és arra is, hogy lassabb legyen mindkét iterációs lépésnél. Ha a két iterációs lépéshez szükséges lépésszám között nagy a különbség, akkor a

véletlen iterációhoz szükséges átlagos lépésszáma a gyorsabb iterációs lépésnek a szükséges lépésszámához van közelebb, ami kedvező hiszen semmi sem garantálja a két iterációs lépés gyorsaságának a hasonlóságát. Azonban a módszer valamilyen oknál fogva rosszul viselkedik olyan polinomokon amelyeknek a gyökei szabályos sokszögként helyezkednek el a komplex síkon.

A. függelék

A dolgozatban használt programok bemutatása

A dolgozatban felhasznált programok GNU Octave nyelven íródtak ami egy ingyenes MATLAB klón, ezáltal a következő programok jó eséllyel lefutnak MATLAB-ban, viszont előfordulhatnak kisebb kompatibilitási hibák. Ezeket azonban általában könnyen és hamar ki lehet küszöbölni. További probléma, hogy az említett programnyelvek lassan kezelik a `for` ciklusokat, helyette a műveletek vektorokkal való megoldását javasolják. Az alábbi programok több egymásba ágyazott `for` ciklust használnak, ezáltal meglehetősen lassúak lehetnek.

A.1. A felhasznált módszerek megvalósítása

A.1. Listing. Newton.m

```
function [gyok it]=Newton(p,x0,seq=0,m=1,h=0.001,maxit=100,prob=0.5)
    x(1)=x0;
    pd=polyder(p);
    pdd=polyder(pd);
    if m==3
        r=1;
    else
        r=0;
    end
    for i=2:(maxit+1)
        px=polyval(p,x(i-1));
        pdx=polyval(pd,x(i-1));
        if r==1
            if(rand>prob)
                m=1;
            else
```

```
17                  m=2;
18              end
19          end
20
21          if m==1
22              if pdx==0
23                  i=i-1;
24                  break;
25              end
26              x(i)=x(i-1)-px/pdx;
27          elseif m==2
28              pddx=polyval(pdd,x(i-1));
29              if pdx^2-pddx*px==0
30                  i=i-1;
31                  break;
32              end
33              x(i)=x(i-1)-(px*pdx)/(pdx^2-pddx*px);
34          end
35
36          if ( abs(x(i)-x(i-1)) < h*abs(x(i)) )
37              break;
38          end
39      end
40      it=i-1;
41      if seq==0
42          gyok=x(i);
43      else
44          gyok=x;
45      end
```

A listázott program végzi minden felhasznált programban a Newton-módszert egy adott polinomra és annak módosított egyenletére, és a módosított módszert is.

A bemeneti változók:

p: Adott polinom együtthatói vektorként, ahol a legnagyobb kitevő együtthatója az első elem, és utána a sorban a többi.

x0: Kezdőpont

seq: Ha 1, akkor a gyok visszatérési típusa a lépések tömbje, ha 0 akkor csak az utolsó lépés

m: Módszer sorszáma. 1: Newton-módszer p-re, 2: Newton-módszer p/p'-re, 3: Véletlen iteráció

h: A leállási feltételhez használt hibahatár

maxit: Maximum megengedett iteráció

prob: A véletlen iterációnál a módosított egyenletre történő Newton-módszer valószínűsége

A felsoroltakból csak az első két paramétert kötelező megadni, ez esetben alapbeállításokkal végzi a módszert.

A.2. Konvergencia szemléltetése

A.2. Listing. IteracioAbraKomplex.m

```
function IteracioAbraKomplex(p,a,b,f,ism,m=3,h=0.001)
    [x y]=meshgrid(linspace(real(a),real(b),f),linspace(imag(a),imag(b),
       f));
    r=roots(p);
    n=length(r)+1;
    z=zeros(f,f);
    zs=zeros(f,f);
    it_temp=zeros(ism);
    for k=1:f
        for l=1:f
            temp=0;
            for i=1:ism
                [gyok,it]=Newton(p,x(k,l)+y(k,l)*j,0,m,h,60);
                [MINE MINH]=min(abs(r-gyok));
                if MINE<h
                    z(k,l)=z(k,l)+1;
                    temp=temp+1;
                    it_temp(temp)=it;
                end
            end
            if temp!=0
                zs(k,l)=mean(it_temp(1:temp));
            else
                zs(k,l)=0;
            end
        end
    end
    figure
    subplot(1,2,1)
    pcolor(x,y,z)
    colorbar;
    colormap hot;
    shading interp;
    axis equal tight;
    subplot(1,2,2)
    pcolor(x,y,zs)
    colorbar;
    colormap hot;
    shading interp;
    axis equal tight;
```

A listázott program elvégzi az adott iterációs eljárást egy a komplex síkról vett téglalapon megadott sűrűséggel és ismétlésszámmal, majd két ábrát generál belőle. Az első ábra szemlélteti, hogy az ismétlésszámból mennyi iteráció ért célt valamelyik gyöknél. A második ábra az iterációhoz szükséges átlagos lépésszámot szemlélteti. Az ábrákon a világosabb szín jelöli a nagyobb számot.

p: Adott polinom együtthatói vektorként, ahol a legnagyobb kitevő együtthatója az első elem, és utána a sorban a többi.

a: A kívánt téglalap bal alsó sarkában elhelyezkedő komplex szám

b: A kívánt téglalap jobb felső sarkában elhelyezkedő komplex szám

f: f^2 kezdőpontra osztjuk a téglalapot

ism: Adott kezdőpontra hányszor ismételjük meg az iterációs módszert

m: Módszer sorszáma. 1: Newton-módszer p-re, 2: Newton-módszer p/p'-re, 3: Véletlen iteráció

h: A leállási feltételhez használt hibahatár

A.2.1. Megjegyzés. *Az generált ábrából egy kevés módosítással más információt is kinyerhetünk. A 21. sorban a* `mean` *függvényt helyettesíthetjük különböző statisztikai függvényekkel mint a* `min`*,* `max`*, vagy az* `std` *függvénnyel mely szórást számol.*

A.3. Listing. KonvergenciaAbraKomplex.m

```
1  function KonvergenciaAbraKomplex(p,a,b,f,ism,m=3,h=0.001)
2      [x y]=meshgrid(linspace(real(a),real(b),f),linspace(imag(a),imag(b),
           f));
3      r=unique(roots(p));
4      n=length(r)+1;
5      z=zeros(f,f,n);
6      for i=1:ism
7          for k=1:f
8              for l=1:f
9                  [gyok,it]=Newton(p,x(k,l)+y(k,l)*j,0,m,h,120);
10                 [MINE MINH]=min(abs(r-gyok));
11                 if MINE<h
12                     z(k,l,MINH)=z(k,l,MINH)+1;
13                 else
14                     z(k,l,n)=z(k,l,n)+1;
15                 end
16             end
17         end
18     end
```

```
19
20      figure
21      for j=1:n
22          subplot(ceil(n/2),2,j)
23          pcolor(x,y,z(:,:,j))
24          colorbar;
25          colormap hot;
26          shading interp;
27          axis equal tight;
28          if j!=n
29              title(num2str(r(j)));
30          else
31              title('egyeb');
32          end
33      end
```

A listázott program elvégzi az adott iterációs eljárást egy a komplex síkról vett téglalapon megadott sűrűséggel és ismétlésszámmal, majd gyökszám plusz egy ábrát generál belőle. Mindegyik gyökhöz tartozott ábra azt szemlélteti hányszor konvergált hozzá a módszer. Az utolsó ábra szemlélteti azokat az iterációkat amik nem értek el egy gyököt.

A metódus paraméterei megegyeznek A.2. listázás paramétereivel.

Irodalomjegyzék

[1] E. T. Whittaker, G. Robinson, „The Newton-Raphson Method" *The Calculus of Observations: A Treatise on Numerical Mathematics, 4th ed.*, New York: Dover, pp. 84-87, 1967.

[2] Faragó István, *Alkalmazott analízis 1*, előadás jegyzet 2012.

[3] Faragó István, Horváth Róbert, „Nemlineáris egyenletek és egyenletrendszerek megoldása" *Numerikus módszerek*, Typotex 2011.

[4] Lily Yau, Adi Ben-Israel „The Newton and Halley Methods for Complex Roots", *The American Mathematical Monthly* Vol. 105, No. 9, November 1998.

[5] P.D. Straffin, C.C.T. Benson, *Calculus Problems for a New Century*, Mathematical Association of America 1993.

[6] William J. Gilbert, „Generalizations of Newton's method", *Fractals* Vol. 9, No. 3, Szeptember 2001.

[7] Veress Krisztián, *A Newton és Gauss-Newton módszerek alkalmazása egyenletrendszerek megoldására és nemlineáris optimalizálásra* [online], 2007, [2013.05.18.], <http://www.inf.u-szeged.hu/~verkri/bin/newton-gauss.pdf>

Nyilatkozat

Név:
ELTE Természettudományi Kar, szak:
NEPTUN azonosító:
Diplomamunka címe:

A **diplomamunka** szerzőjeként fegyelmi felelősségem tudatában kijelentem, hogy a dolgozatom önálló munkám eredménye, saját szellemi termékem, abban a hivatkozások és idézések standard szabályait következetesen alkalmaztam, mások által írt részeket a megfelelő idézés nélkül nem használtam fel.

Budapest, 20 _____

a hallgató aláírása