Eötvös Lóránd Tudomány Egyetem
Természettudományi Kar

Gremsperger Dávid

Többértékű függvények gyökkeresése

BSc Alkalmazott matematikus szakdolgozat

Témavezető:
Sigray István
Analízis Tanszék

2017. május 31.

Tartalomjegyzék

1. Bevezetés **3**

2. Newton-módszer **4**

2.1. Algoritmus valós esetben 5

2.2. Komplex esetben 5

3. Négyzetgyökös összegek **7**

3.1. $\sqrt{z+7}+\sqrt{z-4}=4\sqrt{z+5}$ vizsgálata 7

3.1.1. Gyökök keresése 7

3.1.2. Hibás esetek bemutatása 10

3.2. $\sqrt{z+3}+\sqrt{z-1}=3\sqrt{z+5}$ vizsgálata 12

3.2.1. Gyökök keresése 12

4. A $z=\sqrt[5]{az+b}$ alakú egyenletek **15**

4.1. $z=\sqrt[5]{z+1}$ vizsgálata 16

4.2. $z=\sqrt[5]{z-1}$ vizsgálata 19

4.3. A $z=\sqrt[5]{2z+1}$ vizsgálata 21

4.4. Egyéb együtthatóval ellátott esetek 24

5. Összefoglalás **31**

A. Newtonitsajat.m **32**

B. otod.m **34**

C. javotod.m **36**

D. Ábrázoló programok **38**

Irodalomjegyzék **42**

Köszönetnyilvánítás

Ezúton szeretném megköszönni témavezetőmnek, Sigray Istvánnak, hogy nagyban segítette a szakdolgozatom létrejöttét, és hasznos tanácsokkal látott el a témában. Valamint köszönettel tartozom a családomnak is a segítségért s a támogatásért, amit a dolgozatom megírásában kaptam.

1. fejezet

Bevezetés

Dolgozatom célja különböző komplex értelemben vett többértékű függvények gyökkeresése és vizsgálata egy numerikus módszerrel, a Newton-módszer segítségével. Több példán keresztül megvizsgáljuk a módszer hatékonyságát, különböző tagszámú, illetve magasabb rendű gyököket tartalmazó esetekre is. Először többtagú négyzetgyökös függvényeket, majd ötödik gyököt tartalmazó függvényeket tisztán valós, illetve komplex együtthatókkal.
A módszer implementálásához a MatLab programcsomagot használjuk, s a megoldásokat négy tizedesjegy pontossággal keressük és adjuk meg. A példákhoz felhasznált programok részletes használati utasításai s dokumentációi megtalálhatóak a függelékekben, hogy a dolgozat olvasója maga is megismételhesse a felsorolt példák futását, illetve hogy saját egyenleteket előállítva kísérletezhessen. A programok a következő linken keresztül bárkinek elérhetőek:
`https://drive.google.com/file/d/0B8yJ5m_glHrUSXBob2ZWX256M3M/view?usp=sharing`

2. fejezet

Newton-módszer

A konkrét számolások előtt ejtsünk szót a felhasznált algoritmusról is. Newton-módszer, vagy másnéven Newton-Raphson-módszer egy gyökkereső algoritmus, amellyel valós vagy komplex differenciálható függvények zérushelyeit találhatjuk meg. Működésének alapja a függvény hatványsorba fejtése a megsejtett gyök körül, és a pontba húzott érintőjének meredekségének kiszámolása. A módszer konvergenciájának sebességéről valós értékű függvényekre a következő tételt mondhatjuk ki:

Tétel. *Legyen f kétszer folytonosan differenciálható $\mathbb{R} \to \mathbb{R}$ függvény az x^* egy pontos megoldás zárt környezetében, amire $f(x^*) = 0$ és $f'(x^*) \neq 0$. Ekkor a Newton-módszer minden elég jó kezdeti x_0 közelítés esetén kvadratikusan konvergál.*

Bizonyítás.

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

Használjuk a Lagrange-féle középértéktételt $\xi \in (x^*, x_n)$ -re,

$$= \frac{f'(x_n) - f'(\xi)}{f'(x_n)}(x_n - x^*) =$$

majd újra $\theta \in (x_n, \xi)$ -re,

$$= \frac{f''(\theta)(x_n - \xi)}{f'(x_n)}(x_n - x^*)(x_n - \xi)$$

Tehát azt kaptuk, hogy

$$x_{n+1} - x^* = \frac{f''(\theta)(x_n - \xi)}{f'(x_n)}(x_n - x^*)(x_n - \xi)$$

Vegyük az abszolútértékét, amiből a következő felső becslés következik:

$$|x_{n+1} - x^*| \leq |x_n - x^*||x_n - \xi| max|\frac{f''(\theta)}{f'(x_n)}|$$

Amiből következik:

$$|x_{n+1} - x^*| \leq \frac{max|f''|}{max|f'|}|x_n - x^*|^2$$

Így beláttuk, hogy a módszer kvadratikusan konvergens. □

2.1. Algoritmus valós esetben

Vegyük az $f(x)$ függvény $x = x_0 + \epsilon$ pont körüli Taylor sorát, ahol ϵ a nulladik tag hibája:

$$f(x_0 + \epsilon) = f(x_0) + f'(x_0)\epsilon + ...$$

Majd csak az első két taggal közelítve:

$$f(x_0 + \epsilon) \approx f(x_0) + f'(x_0)\epsilon$$

A fenti közelítés egyenlőséggel teljesülve az $f(x)$ függvény $(x_0, f(x_0))$ ponthoz húzott érintőjének az egyenletét adja. $(x_1, 0)$ legyen az a pont, ahol az érintő metszi az x-tengelyt.
Az $f(x_0 + \epsilon) = 0$ egyenletet megoldva és $\epsilon \equiv \epsilon_0$ -t helyettesítve és rendezve:

$$\epsilon_0 = -\frac{f(x_0)}{f'(x_0)}$$

ami az elsőrendű közelítés a keresett gyökhöz. Majd $x_1 = x_0 + \epsilon$ helyettesítve kiszámoljuk az új ϵ_1-t és ismételjük az eljárást, amíg nem konvergál a gyökhöz. Általánosan:

$$\epsilon_n = -\frac{f(x_n)}{f'(x_n)}$$

A gyök jó megválasztásával az algoritmus iteratívan is használható:

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

Vagy egyszerűbben a valós: $f(x)$-nek az $x = x_n$ pontbeli érintője:

$$y = f'(x_n)(x - x_n) + f(x_n)$$

Ha $y = 0$ és $x = x_{n+1}$ közelítést használva:

$$0 = f'(x_n)(x_{n+1} - x_n) + f(x_n)$$

Ezt rendezve a (2.1)-ben leírt iteratív képletet kapjuk, ahol x_n-nek sorozata tart x^* pontos megoldáshoz. **Megjegyzés**: A módszer instabil lokális szélsőértékek vagy aszimptoták közelében, illetve nem használható ha a derivált az adott pontban 0.

2.2. Komplex esetben

Komplex értékű $f : \mathbb{C} \to \mathbb{C}$ holomorf függvények esetén is érvényes a fenti konvergenciatétel, az alábbi bizonyítással:

Bizonyítás. Legyen $g(z) = z - \dfrac{f(z)}{f'(z)}$ és $g^{\circ k}(z_0)$ pedig jelölje azt, hogy k iterációs lépés után z_0-ból indítva mit kapunk. Tudjuk, hogy ha z^* a pontos megoldása f-nek, akkor $f(z^*) = 0$ és $g(z^*) = z^*$, és $g'(z^*) = 0$. Legyen $n \in \mathbb{N}^+$ a legkisebb olyan szám, melyre $g^{(n)}(z^*) \neq 0$. Böttcher tétele szerint kellően kis $\epsilon > 0$ és $\delta > 0$ esetén van olyan $B(z^*, \epsilon) \to B(0, \delta)$ konform leképezés, melyre $\phi(z) = 0$ és

$$(\phi \circ g \circ \phi^{-1})(z) = z^k$$

Emiatt

$$(\phi \circ g \circ \phi^{-1})^{\circ k} = (\phi \circ g^{\circ k} \circ \phi^{-1}) = w^{n^k},$$

tehát

$$g^{\circ k}(z_0) = \phi^{-1}(w_0)^{n^k},$$

ahol $w_0 = \phi(z_0)$. $w_0^{n^k}$ n rendben tart a 0-hoz, és a 0 közelében a konformitás miatt ϕ^{-1} jól közelíthető egy lineáris függvénnyel, tehát $g^{\circ k}(z_0)$ is n rendben tart z^*-hoz.

□

Így beláttuk, hogy komplexben is igaz a Newton-iteráció konvergenciatétele. A valósban levezetett képletet az iterációra, egy az egyben alkalmazhatjuk $f : \mathbb{C} \to \mathbb{C}$ holomorf függvények esetén is, komplex kezdőpontokból indítva,s a továbbiakban ezt implementálva fogjuk keresni ilyen f-ek gyökeit $\mathbb{C}$-ben.

3. fejezet

Négyzetgyökös összegek

Ebben a fejezetben a $\pm j\sqrt{a} \pm k\sqrt{b} = \pm l\sqrt{c}$ alakú komplex kifejezés gyökeit keressük több példán is megvizsgálva, ahol a, b, c mind $mz \pm n$ alakúak és z komplex, j, k, l, m, n pedig valós számok. De még a konkrét feladat leprogramozása előtt vizsgáljuk meg az egyenletet. Egyszerű algebrai átalakításokat használva próbáljuk egyszerűbben számolható alakra hozni.
Tegyük fel, hogy j, k, l együtthatókat már bevittük a gyökjel alá. Ekkor

$$\pm\sqrt{a} \pm \sqrt{b} = \pm\sqrt{c} \ /^2 \tag{3.1}$$

$$a \pm 2\sqrt{ab} + b = c \tag{3.2}$$

$$a + b - c = \mp 2\sqrt{ab} \ /^2 \tag{3.3}$$

$$a^2 + b^2 + c^2 + 2ab - 2ac - 2bc = 4ab \tag{3.4}$$

$$a^2 + b^2 + c^2 - 2(ab - ac - bc) = 0 \tag{3.5}$$

Mivel a, b, c mind lineáris kifejezések, ezért (3.5) egy legfeljebb másodfokú kifejezés lehet, és így legfeljebb két különböző megoldása van. Komplex számtest felett ezek ekvivalens átalakítások, így feltételezhetjük, hogy (3.1)-nek is legfeljebb két különböző gyöke lehet. Kettő valós, vagy kettő képzetes, amik egymás konjugáltjai, vagy egy darab kétszeres valós gyök fordulhat elő.
Ezen előismeretek birtokában jobb elképzelésünk van arról, hogy a programunknak mennyi gyököt kellene megtalálnia.

3.1. $\sqrt{z+7} + \sqrt{z-4} = 4\sqrt{z+5}$ vizsgálata

3.1.1. Gyökök keresése

Az egyenlet keresett két gyöke a $\approx -5.5833 \pm 0.5385i$. Tekintsük a fenti egyenletet a baloldalra rendezve és indítsuk az első iterációt $z_0 = 1 + i$ pontból az $f(z) = \sqrt{z+7} +$

$\sqrt{z-4}-4\sqrt{z+5}$ függvényre. Az egyes iterációs lépésekben meghatározott következő iterációs pontok, és hozzájuk tartozó függvényértékeik:

[z,fz,it]=Newtonitsajat(10,10^{-4},1+i)
Nulladik iterációs lépésben:
$z_0 = 1 + i$
$f(z_0) = -6.7129 + 1.1181i$

Első iterációs lépésben:
$z_1 = -8.3761 + 6.4408i$
$f(z_1) = -3.0812 - 3.6061i$

Második iterációs lépésben:
$z_2 = -5.1840 - 4.3602i$
$f(z_2) = -3.2737 + 1.7158i$

Ötödik iterációs lépésben:
$z_5 = -4.8896 - 0.9929i$
$f(z_5) = -1.3225 - 0.6530i$

Kilencedik iterációs lépésben:
$z_9 = -5.5833 - 0.5385i$
$f(z_9) = -0.0000 - 0.0000i$ (Itt áll le a program, elértük a kívánt pontosságot.)

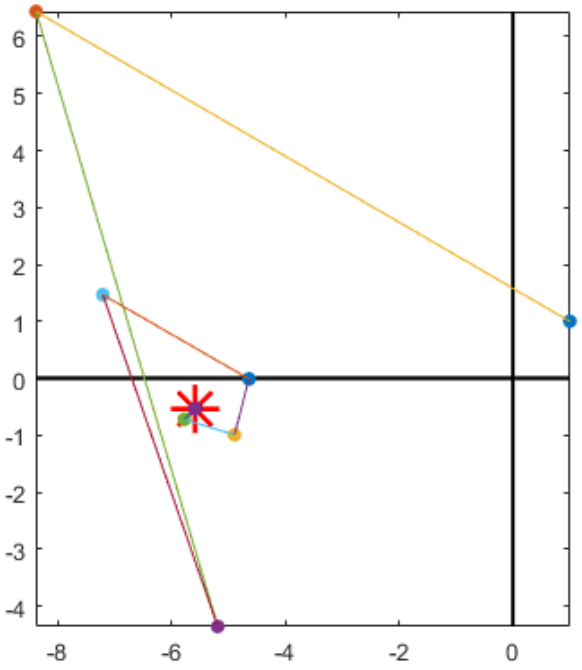
A program által megtalált gyök a $-5.5833 - 0.5385i$, ahogy azt a pár részletesen kiírt iterációs lépés mutatja. Az ábrán az nem látszik, mert a pontok annyira illeszkednek egymásra, hogy az utolsó lépésekben ugyan nem értük el a kívánt pontosságot, de már akkor is szinte a megoldással azonos értékeket kaptunk.

Az eredményeket a következő példákban is ebben a formátumban fogjuk kilistázni.

Mivel a fenti komplex szám megoldás, ezért előzetes ismeretek nélkül is joggal gondolhatjuk, hogy a konjugáltjának is megoldásnak kell lenni. Így jó másik kezdőpont választásnak tűnik az első kezdőpont konjugáltja. Indítsuk most a $z_0 = 1 - i$ pontból az iterációt.

[z,fz,it]=Newtonitsajat(10,10^{-4},1-i)
Nulladik iterációs lépésben:
$z_0 = 1 - i$
$f(z_0) = -6.7129 - 1.1181i$

Első iterációs lépésben:



3.1. ábra. Az $1 + i$ konvergálása

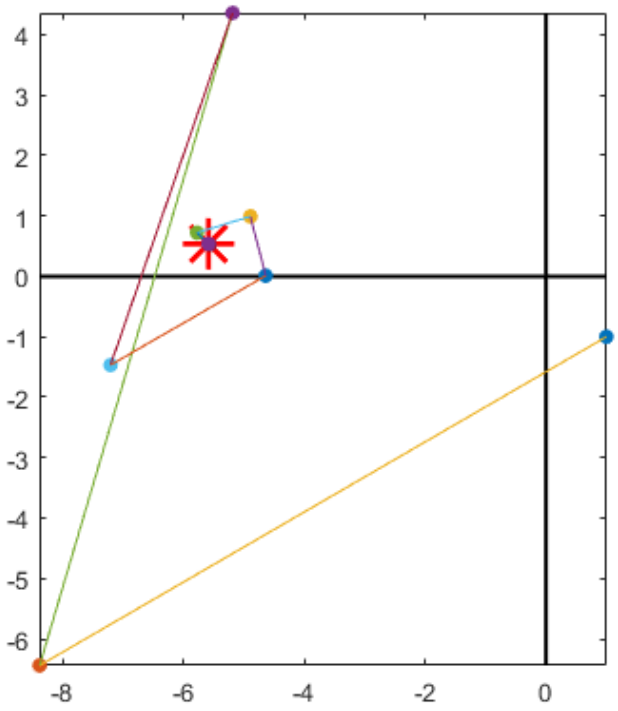
$z_1 = -8.3761 - 6.4408i$
$f(z_1) = -3.0812 + 3.6061i$

Második iterációs lépésben:
$z_2 = -5.1840 + 4.3602i$
$f(z_2) = -3.2737 - 1.7158i$

Ötödik iterációs lépésben:
$z_5 = -4.8896 + 0.9929i$
$f(z_5) = -1.3225 + 0.6530i$

Kilencedik iterációs lépésben:
$z_9 = -5.5833 + 0.5385i$
$f(z_9) = -0.0000 + 0.0000i$

A program által megtalált gyök a $-5.5833 - 0.5385i$. Azontúl, hogy konjugált kezdőpontból a konjugált megoldást találtuk meg, az egyes iterációs lépésekben talált pontok is konjugáltjaik egymásnak, sőt az ezeken a helyeken felvett függvényértékek is.



3.2. ábra. Az $1 - i$ konvergálása

3.1.2. Hibás esetek bemutatása

Ebben a részben leírt hibák analóg módon értelmezhetők bármely másik, a továbbiakban tárgyalt példák esetében is. Ha az iterációt olyan kezdőpontból indítjuk, mely valamelyik négyzetgyökjel alatti kifejezésbe behelyettesítve 0-t eredményez, ebben az esetben a $-7, 4, -5$, akkor a megfelelő tagban a deriváltban végtelennel kellene osztanunk. Ez a MatLab-ban nem hibát vagy kivételt fog eredményezni, hanem végtelent, azaz Inf-et fog visszaadni. Így a derivált értéke Inf+konstans értéket fog felvenni, ahol a konstans a többi tagból adódik. A következő pont kiszámításánál az iterációban ekkor végtelennel kell osztanunk az $f(z_n)$ véges számot, aminek a végeredménye 0. Ez azt eredményezi, hogy a kiindulási pont sosem fog elmozdulni, végig ugyanaz marad.
Példaként, a program következő hibás eredménnyel fut le, minden iterációs lépésben a $z_0 = 4$-ből indítva:

Tetszőleges iterációs lépésben:
$z_n = 4$
$f(z_n) = -8.6834$

A z értékek állandók, ezért megoldás is lehetne, de f(z) értékei nem tartanak nullához, ami szükséges lenne ahhoz, hogy valódi gyök legyen.

A probléma kézenfekvő megoldása a perturbálás, a kezdőpont elmozdítása egy kicsi számmal. A $z_0 = 3.9$-ből indítva az iterációt, már az egyik megoldáshoz fogunk konvergálni.

[z,fz,it]=Newtonitsajat(20,10^{-4},3.9)
Nulladik iterációs lépésben:
$z_0 = 3.9$
$f(z_0) = -8.6316 + 0.3162i$

Első iterációs lépésben:
$z_1 = 2.4630 + 4.9875i$
$f(z_1) = -6.9362 - 0.8559ii$

Második iterációs lépésben:
$z_2 = -16.8541 + 3.6306i$
$f(z_2) = -1.1202 - 6.1548i$

Hetedik iterációs lépésben:
$z_5 = 0.0093 + 0.5514i$
$f(z_5) = -6.1788 + 1.6145i$

Tizennegyedik iterációs lépésben:
$z_9 = -5.5833 + 0.5385i$
$f(z_9) = 0.0000 + 0.0000i$

A program által megtalált gyök a $-5.5833 + 0.5385i$. Mivel lényegesen messzebbről indítottuk, ezért a konvergencia lassabb, és több iterációs lépésre van szükségünk. Valamint az is igaz, hogy az ilyen jellegű nem megoldáshoz konvergáló pontoknak nem létezik konvergenciatartománya, azaz bármilyen picivel is mozdítjuk el onnan a kezdőpontunkat, akkor már az egyik valódi megoldáshoz fogunk konvergálni. Annak az állításnak a belátásával, hogy ezek instabil fixpontok később, az ötödikgyökös esetben foglalkozunk.
Hasonlonó, perturbálással megoldható problémával már harmadfokú polinomok esetében is találkozhatunk a Newton-iteráció kapcsán. Például a $z^3 + iz + (-1 - i)$ esetében is léteznek olyan pontok, amelyekből indulva nem kovergálunk, de megperturbálva a kezdőpontot már ismét valamelyik gyökét találjuk meg az egyenletnek.

3.2. $\sqrt{z+3}+\sqrt{z-1}=3\sqrt{z+5}$ vizsgálata

3.2.1. Gyökök keresése

A fenti egyenlet ránézésre hiába egyezik meg teljesen az előző példánkkal, a megoldása során mégis felvetődik egy eddigiekben nem tapasztalt probléma a gyökei keresése során.

Újfent rendezzük egy oldalra az egyenletet, és indítsuk el az iterációt az $z_0 = -1+i$-ból.

[z,fz,it]=Newtonitsajat(10,10^{-4},-1+i)
Nulladik iterációs lépésben:
$z_0 = -1.0000 + 1.0000i$
$f(z_0) = -4.2471 + 1.0546i$

Első iterációs lépésben:
$z_1 = -6.2040 + 9.0928i$
$f(z_1) = -2.7114 - 1.2327i$

Második iterációs lépésben:
$z_2 = -14.2193 - 3.3308i$
$f(z_2) = -0.7037 + 1.9423i$

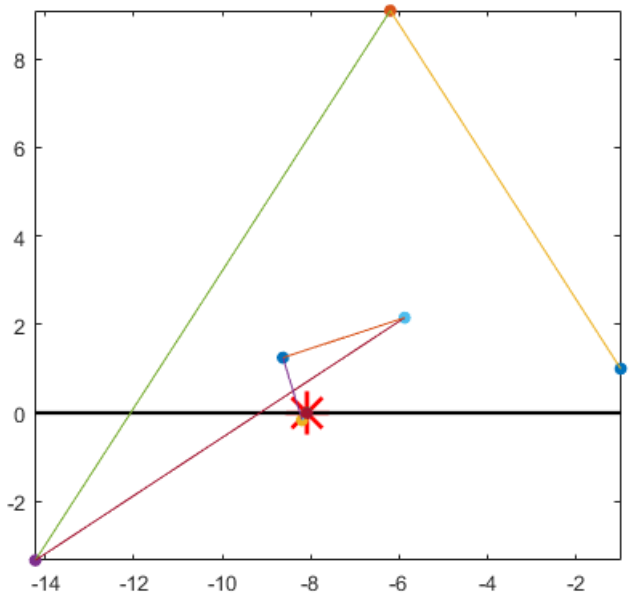
Negyedik iterációs lépésben:
$z_4 = -8.6283 + 1.2489i$
$f(z_4) = -0.5072 - 0.2999i$

Hetedik iterációs lépésben:
$z_7 = -8.0847 - 0.0000i$
$f(z_7) = -0.0000 + 0.0000i$

A program által megtalált gyök a -8.0847.

Most az eddig tapasztaltakhoz képest egy tisztán valós gyököt találtunk meg, így az az ötlet, hogy következő kezdőpontunknak válasszuk az $-1-i$-t azaz a konjugáltat biztosan nem célravezető. Sőt a komplex sík bármely pontjából indítva az iterációt, az ha konvergál, mindig csak a -8.0847-ba fog bekonvergálni, még akkor is, ha a másik gyök pontos értékét választjuk kiinduló pontnak. Ez a másik keresett gyök a $\frac{4\sqrt{31}}{15} - \frac{33}{5} \approx -5.1153$, mely természetesen szintén tisztán valós.

Ennek a hibának az oka abban keresendő, hogy a MatLabban is, mint sok másik programnyelvben vagy programcsomagban, az n-edik gyökvonás beépített művelete



3.3. ábra. A -1+i konvergenciája

csak a legnagyobb pozitív valósrészű (ha csak tisztán képzetes gyökei vannak, akkor a pozitív képzetest) gyököt adja meg kimenetként, s így mivel nem veszi figyelembe a többi lehetséges eredményt a négyzetgyökvonásnál, s így nem találja meg a második megoldást.

Egy egyszerű példa erre: a MatLabban kiadva a sqrt(4) parancsot, azt az eredményt kapjuk, hogy 2. Pedig a -2 is ugyanúgy jó megoldás lenne. A további gyökök megkereséséhez használjuk ki, hogy ha z megoldása a négyzetgyökvonásnak, akkor a $-z$ is. Így a programkódban minden gyökjeles tag úgy módosul, hogy vagy megtartja eddigi értékét, vagy -1-szeresére változik. Három tagunk van, így $2 \cdot 2 \cdot 2 = 8$ különböző eset jön létre, de ezeknek a felét eleve el is hagyhatjuk, ugyanis azok az egyenletek amik egymás -1-szeresei, pont ugyanazt a megoldást fogják adni.
Konkrétan az eseteink párokba állítva:

$$\sqrt{z+3}+\sqrt{z-1}-3\sqrt{z+5}=0 \qquad -\sqrt{z+3}-\sqrt{z-1}+3\sqrt{z+5}=0 \tag{3.6}$$

$$\sqrt{z+3}-\sqrt{z-1}+3\sqrt{z+5}=0 \qquad -\sqrt{z+3}+\sqrt{z-1}-3\sqrt{z+5}=0 \tag{3.7}$$

$$-\sqrt{z+3}+\sqrt{z-1}+3\sqrt{z+5}=0 \qquad \sqrt{z+3}-\sqrt{z-1}-3\sqrt{z+5}=0 \tag{3.8}$$

$$\sqrt{z+3}+\sqrt{z-1}+3\sqrt{z+5}=0 \qquad -\sqrt{z+3}-\sqrt{z-1}-3\sqrt{z+5}=0 \tag{3.9}$$

amikből elég csak a bal oldali oszlopot vizsgálni. A (3.6)-t már kiszámoltuk és az a már megtalált megoldáshoz konvergált. Nézzük a (3.7)-t és a Newtonitsajat.m fájlban írjuk át a harmadik és negyedik sorokban a függvényt és deriváltját az aktuálisan vizsgált esetre, majd futtassuk a $z_0 = -4$ kezdőpontból a számolást.

[z,fz,it]=Newtonitsajat(10,10^{-4},-4)

Nulladik iterációs lépésben:

$z_0 = -4.0000$

$f(z_0) = 3.0000 - 1.2361i$

Első iterációs lépésben:

$z_1 = -6.0812 + 0.4406i$

$f(z_1) = 0.6657 + 2.2785i$

Második iterációs lépésben:

$z_2 = -4.6589 - 0.2906i$

$f(z_2) = 1.9359 + 0.3929i$

Negyedik iterációs lépésben:

$z_4 = -4.9954 + 0.0529i$

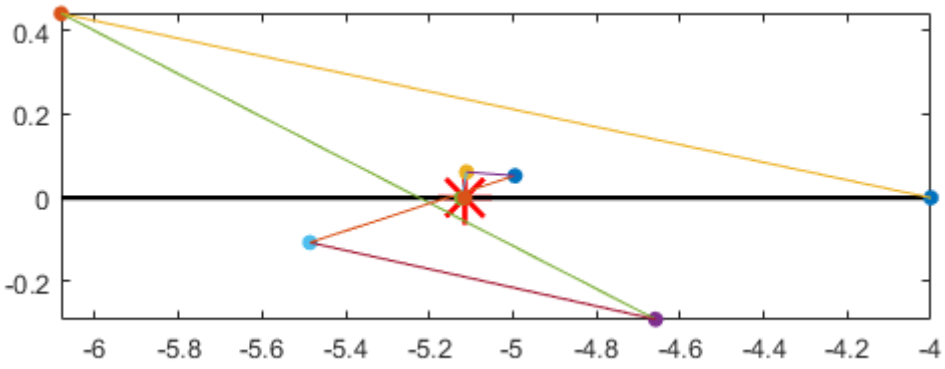
$f(z_4) = 0.5174 - 0.5690i$

Nyolcadik iterációs lépésben:

$z_8 = -5.1153 + 0.0000i$

$f(z_8) = 0.0000 - 0.0000i$

A megtalált gyök a -5.1153.



3.4. ábra. A -4 konvergenciája

Így rögtön megtaláltuk a második gyököt, és nem volt szükség a (3.8) és (3.9) eseteket is megvizsgálni. Ha azonban ezeket az egyenleteket megpróbáljuk bevinni és lefuttatni, azt tapasztalnánk, hogy bármely tetszőleges kezdőpontot választva az iterációk sosem konvergálnak sem meglévő, sem új megoldáshoz. Ha előfordulna, hogy esetleg már az eredetileg megadott egyenletnek sem találnánk gyökét egy másik tetszőleges példánál, akkor is meg kell nézni a többi három esetet, mert csak azok ellenőrzése után állapíthatjuk meg a különböző megoldások pontos számát biztosan.

4. fejezet

A $z = \sqrt[5]{az+b}$ alakú egyenletek

Miután az előző egyenlettípust alaposan megvizsgáltuk több szempontból is, azonos megközelítéssel használjuk a már meglévő programunkat a $z = \sqrt[5]{az+b}$ többértékű függvény zérushelyeinek megtalálására, ahol a és b nem nulla valós számok. Először most is vizsgáljuk meg algebrai átalakításokkal is az egyenletet, a feladatra való jobb rálátás kedvéért.

$$z = \sqrt[5]{az+b} \ /^5 \tag{4.1}$$

$$z^5 = az + b \tag{4.2}$$

$$z^5 - az - b = 0 \tag{4.3}$$

Ebben az esetben triviálisan, egy pontosan ötödfokú polinomra vezetett a számolás, aminek öt megoldása van. Mivel ekvivalens átalakításokat végeztünk, így (4.1) és (4.3) gyökei megegyeznek, tehát a (4.1) alakú egyenletekre is, ideális esetben, a Newton-módszer segítségével legfeljebb 5 megoldást kell találnunk.

Nézzük a (4.3)-ban felírt egyenletet csupán valós számok felett, és vizsgáljuk meg valós gyökeinek lehetséges számát. Ehhez nézzük az első deriváltját. Átrendezve:

$$x^4 = \frac{a}{5} \tag{4.4}$$

Ha a paraméter pozitív, akkor (4.4)-nek általánosan 2 darab valós megoldása van: $\pm\frac{\sqrt{a}}{\sqrt{5}}$. Mivel $a \neq 0$, ezért a derivált biztos előjelet vált a gyököknél, és az első egy lokális maximuma, a második pedig egy lokális minimuma lesz (4.3)-nak, ami $\pm\infty$-ben a $\pm\infty$-be tart. Így (4.3)-nak lehet 1, 2 (ekkor az egyik biztos kétszeres gyök), illetve 3 különböző valós gyöke is.

Ha a paraméter pedig negatív, akkor a deriváltjának nem létezik valós felett megoldása, grafikonja minden pontban pozitív értékeket vesz fel, tehát (4.3) szigorúan monoton nő, határértéke a $\pm\infty$-ben $\pm\infty$ ismét. Ebből adódóan csak 1 valós gyöke van.

4.1. $z = \sqrt[5]{z+1}$ vizsgálata

Rendezzük a bal oldalra a kifejezést, majd deriváltjával együtt írjuk be a programba, és első kezdőpontunknak a $z_0 = 0$-t válasszuk.

[z,fz,it]=otod(10,10^{-4},0,0)
Nulladik iterációs lépésben:
$z_0 = 0$
$f(z_0) = -1$

Első iterációs lépésben:
$z_1 = 1.2500$
$f(z_1) = 0.0739$

Második iterációs lépésben:
$z_2 = 1.1674$
$f(z_2) = 0.0001$

Harmadik iterációs lépésben:
$z_3 = 1.1673$
$f(z_3) = 0.0000$

A megtalált gyök az 1.1673

Azonban fellép a már az előzőekben megismert probléma, miszerint ha valós megoldást találunk elsőre, akkor bármely másik kezdőpontot választunk, nem találunk új valódi gyököt, akármilyen jó közelítést is használnánk. A megoldást újra az fogja nyújtani, hogy figyelembe vesszük az ötödik gyökvonás egyéb értékeit is a $\sqrt[5]{az+b}$ esetében, mint ahogy a négyzetgyökös esetben. A többi eredményhez a MatLab-ban a $\sqrt[5]{az+b}$-ból megkapott értéket forgassuk el a komplex számsíkon 72°-kal, mert a megoldások ugyanazon az origó középpontú körön helyezkednek el. Ezt négyszer tudjuk megtenni egymás után, mielőtt visszatérnénk a kiindulási pontba.

Ennek az implementálásához kézenfekvő megoldást nyújtana az origó körüli θ szögű forgatás mátrixa: $M = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix}$, ám a MatLab nem vektoros formában tárolja a komplex számokat, így ezt a szorzást ilyen formában nem tudjuk elvégezni egyéb átalakítások nélkül. Ehelyett szorozzuk egyből a számot $e^{i\theta}$, ahol θ a kívánt forgatás szöge radiánban megadva.
Legyen $z = a+ib$. A két számolásnak a végeredménye ugyanaz, mert az Euler-formulát használva látszik, hogy:

$$(a+ib) \cdot e^{i\theta} = (a+ib)(cos(\theta) + isin(\theta)) =$$

$$= a \cdot cos(\theta) + ia \cdot sin(\theta) + ib \cdot cos(\theta) - b \cdot sin(\theta) =$$

$$= a \cdot cos(\theta) - b \cdot sin(\theta) + i[a \cdot sin(\theta) + b \cdot cos(\theta)]$$

Míg vektoros alakban a mátrixszorzást elvégezve:

$$\begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} a \cdot cos(\theta) - b \cdot sin(\theta) \\ a \cdot sin(\theta) + b \cdot cos(\theta) \end{bmatrix}$$

Innen látszik, hogy a két számolási mód egyenértékű.

Egészítsük ki ezzel a művelettel a programunkat. Mind a függvénynél, mind a deriváltnál vezessünk be egy új szorzótagot az ötödik gyökvonást tartalmazó tagokhoz, ami egy 72°-os forgatásnak felel meg. Először váltsuk át a fokot radiánba: $\frac{72\pi}{180}$, majd legyen ez az új tag $e^{\frac{72\pi i}{180} forg}$, ahol a $forg$ változó az összes gyök megtalálásához szükséges és $forg = 0, 1, 2, 3, 4$. Keressünk tovább a megoldásokat a forgatásos módszer felhasználásával, és az egyszerűség kedvéért z_0-t válasszuk végig 0-nak, ami az összes megoldásunknak egy jó közelítése.

Ha a forgatás 72°-os:

[z,fz,it]=otod(10,10^{-4},0,1)

Nulladik iterációs lépésben:

$z_0 = 0$

$f(z_0) = -0.3090 - 0.9511i$

Első iterációs lépésben:

$z_1 = 0.1190 + 1.0378i$

$f(z_1) = -0.0594 - 0.0357i$

Második iterációs lépésben:

$z_2 = 0.1813 + 1.0842i$

$f(z_2) = 0.0001 + 0.0002i$

Harmadik iterációs lépésben:

$z_3 = 0.1812 + 1.0840i$

$f(z_3) = -0.0000 + 0.0000i$

A megtalált gyök a 0.1812 + 1.0840i

Ha a forgatás 144°-os:

[z,fz,it]=otod(10,10^{-4},0,2)

Nulladik iterációs lépésben:

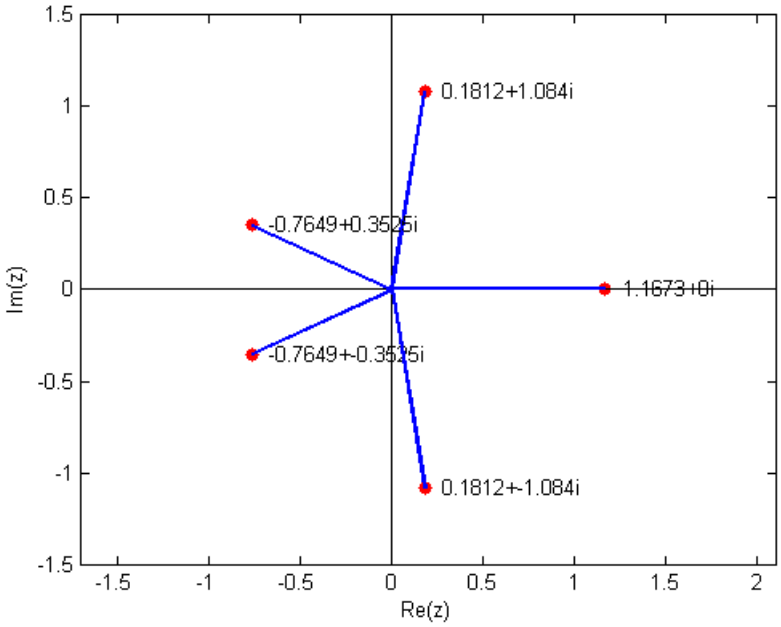
$z_0 = 0$
$f(z_0) = 0.8090 - 0.5878i$

Első iterációs lépésben:
$z_1 = -0.7400 + 0.4311i$
$f(z_1) = 0.0550 + 0.0734i$

Második iterációs lépésben:
$z_2 = -0.7666 + 0.3517i$
$f(z_2) = -0.0021 - 0.0001i$

Harmadik iterációs lépésben:
$z_3 = -0.7649 + 0.3525i$
$f(z_3) = 0.0000 + 0.0000i$

A megtalált gyök a -0.7649 + 0.3525i



4.1. ábra. A $z = \sqrt[5]{z+1}$ gyökeinek elhelyezkedése

A maradék forgatások eredményét összegezve, de nem részletezve, ha 216°-kal forgatunk, akkor a 144°-os eredmény konjugáltját, $-0.7649-0.3525i$-t, míg 288°-nál pedig a 72°-nál talált megoldás konjugáltját, $0.1812-1.0840i$-t kapjuk. A vártnál ellentétben, ennek a módszernek a felhasználásánál, nincs segítségünkre a kezdőpontok konjugálása, abban hogy a konjugált megoldáspárokat megtaláljuk egy elforgatáson belül. De így

ebben az esetben megtaláltuk mind az öt megoldását az egyenletnek, amik elhelyezkedését a fenti ábra szemlélteti. Figyeljük meg, hogy a gyököket pont olyan sorrendben találtuk meg az egyre nagyobb szögű forgatásokkal, mint ahogy azok szöge növekszik.

4.2. $z = \sqrt[5]{z-1}$ vizsgálata

Tekintsük a fenti feladatot. Az előzőekhez hasonlóan alakítsuk át, és első iterációt indítsuk a $z_0 = i$ pontból.

[z,fz,it]=otod(10,10^{-4},i,0)
Nulladik iterációs lépésben:
$z_0 = 0 + 1.0000i$
$f(z_0) = -0.9550 + 0.5134i$

Első iterációs lépésben:
$z_1 = 0.8290 + 0.3954i$
$f(z_1) = 0.0493 + 0.0696i$

Második iterációs lépésben:
$z_2 = 0.7629 + 0.3519i$
$f(z_2) = -0.0019 - 0.0014i$

Harmadik iterációs lépésben:
$z_3 = 0.7649 + 0.3525i$
$f(z_3) = -0.0000 + 0.0000i$

A megtalált gyök a 0.7649 + 0.3525i

Most elsőre egy komplex gyököt találtunk, így próbáljuk meg a konjugált kezdőpontból a megoldásunk konjugált párját megtalálni:

[z,fz,it]=otod(10,10^{-4},-i,0)
Nulladik iterációs lépésben:
$z_0 = 0 - 1.0000i$
$f(z_0) = -0.9550 - 0.5134i$

Első iterációs lépésben:
$z_1 = 0.8290 - 0.3954i$
$f(z_1) = 0.0493 - 0.0696i$

Második iterációs lépésben:

$z_2 = 0.7629 - 0.3519i$
$f(z_2) = -0.0019 + 0.0014i$

Harmadik iterációs lépésben:
$z_3 = 0.7649 - 0.3525i$
$f(z_3) = -0.0000 - 0.0000i$

A megtalált gyök a 0.7649 - 0.3525i

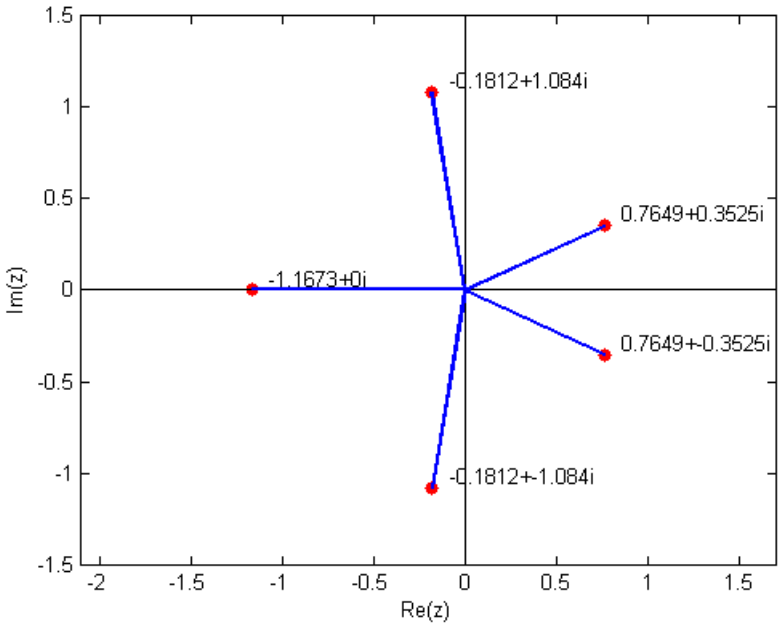
Láthatjuk, hogy ebben az esetben sikerült egy számolási lépésen belül két gyököt is találni, de ez csak akkor működik általánosan, ha a *forg* paraméterünk 0, és nem tisztán valós értékű megoldást találunk elsőre. Erre példa az előző esetben taglaltak.

A következő gyök megtalálásához állítsuk a *forg* paramétert 1, és válasszunk egy elég jó kezdőpontot. Legyen $z_0 = i$. Ekkor az előzőekben már megszokott módon a módszer konvergens, és a megtalált megoldás két iterációs lépés után a $z = -0.1812 + 1.0840i$. Tovább haladva, ha $forg = 2$ és $z_0 = -1$, akkor könnyedén jutunk el egy újabb megoldáshoz ismét kettő iteráción belül: $z = -1.1673$

Egy érdekes megfigyelés, hogy ebben az esetben, ha nem tisztán valós kezdőpontot választunk, vagy ha az a valós szám nagyobb, mint körülbelül 0.801, akkor nem konvergens a módszer, akármilyen kicsire is veszem a szám képzetes részét, az egy végtelen, két pont között oszcilláló ciklusba kerül. (Valójában ha a képzetes rész már 10^{-14} nagyságrendű, akkor már ismét konvergál jó valós kezdőpontot választva, de ez olyan kicsi, hogy szinte elhanyagolható.)

Az előző szakaszban bemutatott egyenlethez képest, most ha a *forg* paramétert 3-ra állítjuk, akkor azt tapasztaljuk, hogy nem tudunk olyan kezdőpontot megadni, hogy egy valódi gyökhöz konvergálna. Miután az *forg*=0 esetben rögtön 2 gyököt is találtunk, várható volt, hogy lesz olyan eset, ahol lényegében új eredménnyel nem fogunk találkozni. Az iteráció ismét két pont között fog oszcillálni egy végetlen ciklusban amik minden kezdőpont esetén ugyanazok. Ezek néhány lépést elvégezve fixen bekonvergálnak körülbelül a -0.2201 - 1.0448i és -1.1121 + 0.0351i értékekre, amik még meg is egyeznek az előbbi megfigyelésünkben fellépő két ponttal.

Az utolsó hiányzó gyök újra gond nélkül adódik a *forg*=4 esetben $z_0 = -i$ kezdőpontból indítva már két iterációs lépésen belül, s a kapott megoldás a
$z = -0.1812 - 1.0840i$. Így ebben az esetben is megtaláltuk mind az öt keresett értéket, csupán nem a várt és már bemutatott sorrendben oszlanak el a *forg* paraméter különböző értékei között.



4.2. ábra. A $z = \sqrt[5]{z-1}$ gyökeinek elhelyezkedése

4.3. A $z = \sqrt[5]{2z+1}$ vizsgálata

Az eddig taglalt egyenlettípus egy másik merőben más esete, az egy darab valós megoldást tartalmazón kívül, az amikor az egyenletnek három valós gyöke is van. Ebben az esetben, az eddig használt módszer, miszerint kiválasztunk egy ötödik gyökvonás értéket, és végig azzal számolunk, nem fogja megtalálni az összes megoldást, hiába közelítenénk azt tetszőlegesen közel (de nem a pontos értékből indítva), egy végtelen ciklusba jutunk, ami néhány pont között oszcillál. Ezért vezessünk be egy új módszert, ami minden iterációban kiszámolja az öt különböző gyökvonás értékének megfelelően az öt lehetséges következő pontot, és azt választja ki, amelynek a legkisebb az abszolútértéke az egyenlettel egyenrangú (4.3) alakú polinomba behelyettesítve.
Ez biztosítja, hogy ha már elég közel kerültünk egy megoldáshoz, akkor ne hagyjuk el a környezetét, és biztos bekonvergáljunk, valamint még azt is eredményezi, hogy bármilyen messzi kezdőpont választva a módszer véges legyen. Kivéve az $az+b=0$ egyenletet kielégítő $z = -\frac{b}{a}$ megoldásból indítva, ez továbbra sem konvergál egyik megoldáshoz sem, de megperturbálva már újra biztosan konvergálni fogunk.

Bizonyítás. Legyen $z_0 = -\frac{b}{a} + c$,ahol $|c| = \epsilon$ és $h(z) = z - \frac{f(z)}{f'(z)}$, valamint tegyük fel, hogy $a \neq 0$. Ekkor $h(z)$ a z_0 kezdőpontból indulva a következő pontunk az iterációban,

ahol K konstans tényezőt jelöl:

$$h(z_0) = z_0 - \frac{f(z_0)}{f'(z_0)} = -\frac{b}{a} + c + \frac{-\frac{b}{a} + \sigma(1)}{-\frac{a}{5}(ac)^{-4/5} + \sigma(c^{-4/5})} = -\frac{b}{a} + c + (-\frac{b}{a} - \frac{a}{5})(ac)^{4/5} + \sigma(c^{-4/5}) =$$

$$= -\frac{b}{a} + Kc^{4/5} + \sigma(c^{4/5})$$

Amiről tudjuk, hogyha ϵ elég kicsi, akkor $K\epsilon^{4/5} > \epsilon$ már teljesül, s így az új pontunk messzebb lesz a $-\frac{b}{a}$-tól, sőt még az is látszik, hogy lassú lesz az eltávolodás. □

A program jellegéből adódóan, már csak a z_n értékeket kívánjuk közölni lépésenként. Az egyenlet megoldásai: $1.2906, 0.1141 + 1.2167i, -1.0000, -0.5188,$
$0.1141 - 1.2167i$. (A $z^5 = 2z + 1$ megoldásai.)
Legyen a kezdőpontunk a $z_0 = 1$, és futtassuk az iterációt:

[z]=javotod(10,10^{-4},1)
Nulladik iterációs lépésben:
$z_0 = 1$

Első iterációs lépésben:
$z_1 = 1.2947$

Második iterációs lépésben:
$z_2 = 1.2906$

A megtalált megoldás a 1.2906

Hasonló egyszerűséggel és kevés iterációs lépésen belül adódik a többi megoldásunk is, akár nagyon rossz kezdőpont mellett is: $z_0 = 100i$

[z]=javotod(10,10^{-4},100i)
Nulladik iterációs lépésben:
$z_0 = 100i$

Első iterációs lépésben:
$z_1 = 0,0052 + 2.3217i$

Második iterációs lépésben:
$z_2 = 0.0891 + 1.2441i$

Harmadik iterációs lépésben:
$z_3 = 0.1140 + 1.2167i$

Negyedik iterációs lépésben:
$z_4 = 0.1141 + 1.2167i$

A megtalált megoldás a $0.1141 + 1.2167i$

Ezek alól kivételt képez minden esetben a három valós gyök közül az, amelyik középen helyezkedik el a számegyenesen. Mivel a másik négy gyök úgymond "körbeveszi", ezért ehhez csak egy elég közeli kezdőpontból fog konvergálni megfelelően a módszer, különben nem fogunk új eredményt kapni.

[z]=javotod(10,10^{-4},-0.6)
Nulladik iterációs lépésben:
$z_0 = -0.6$

Első iterációs lépésben:
$z_1 = -0.3224 - 0.0000i$

Második iterációs lépésben:
$z_2 = -0.5757 - 0.1962i$

Harmadik iterációs lépésben:
$z_3 = -0.7598 + 0.0189i$

Hatodik iterációs lépésben:
$z_4 = -1.0000 - 0.0000i$

A megtalált megoldás a -1, holott sokkal közelebb vagyunk a $-0,5188$-hez. Menjünk még közelebb a kívánt megoldáshoz és $z_0 = -0.51$ legyen, ami már megfelelő lesz a célunk eléréséhez. Az ezekhez hasonló esetekben a kellően jó kezdőpont megadása szinte elengedhetetlen a gyök megtalálásához.

[z]=javotod(10,10^{-4},-0.51)
Nulladik iterációs lépésben:
$z_0 = -0.51$

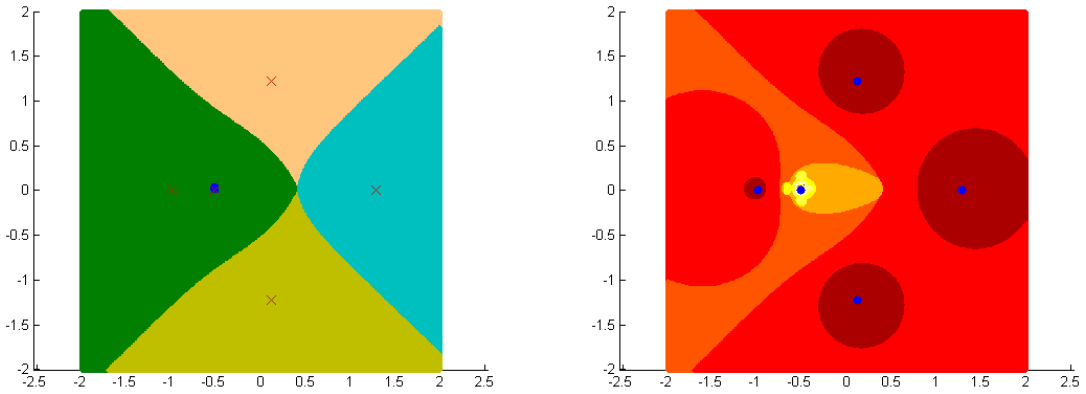
Első iterációs lépésben:
$z_1 = -0.5165 - 0.0000i$

Második iterációs lépésben:
$z_2 = -0.5186 - 0.0000i$

Harmadik iterációs lépésben:

$z_3 = -0.5188 - 0.0000i$

A megtalált megoldás a -0.5188



(a) Koncergenciák (b) Sebesség

4.3. ábra. $z = \sqrt[5]{2z+1}$ futása

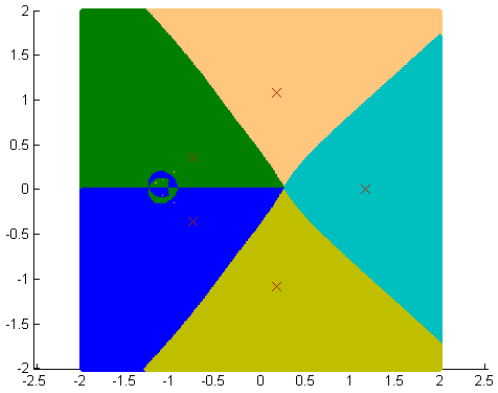
A 4.2/b ábrán a sötét, vörös árnyalatú színek gyors konvergenciát jelentenek, és a világos egyre fehéredő sárgák pedig a lassú,a kár 10 feletti lépésszámú iterációt jelölnek.

A konvergenciaábrán az is megfigyelhető, hogy ehhez a gyökhöz konvergáló kezdőpontok halmaza (sötétkékkel jelölt), egy korlátos tartományt képeznek, míg a többihez tartozók nem.

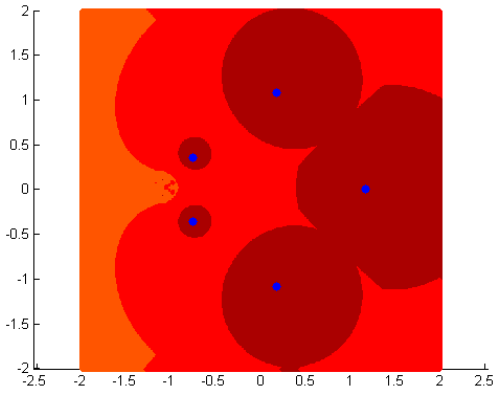
Még érdemes valamelyik előbbi példát is megnézni hogyan viselkedik a javotodben implementált módszerrel. Ha az előbbi $z = \sqrt[5]{z+1}$ példát vizsgáljuk ($z = \sqrt[5]{z-1}$ ábrái az y tengelyre való tükrözéssel állnak elő), akkor a következő lenti ábrákat kapjuk. Sőt, ha nagyobb tartományon tekintjük a konvergenciaábrát, azt tapasztaljuk, hogy a -1.25 környezetében lévő "hurok" megismétlődik nagyobb méretben, ugyanazokkal az egyéb színű ponthalmazokkal együtt körülötte, amik ebben a méretben jobban meg is figyelhetők. Ilyen különböző méretekben ismétlődő alakzatokat a polinomokra elvégzett Newton-iterációknál is megfigyelhettünk már.

4.4. Egyéb együtthatóval ellátott esetek

A javított módszert, akkor is gond nélkül tudjuk alkalmazni, ha most a $z = \sqrt[5]{az+b}$ alakú egyenletekre megengedjük, hogy a és b paraméterek ne csak tisztán valósak, hanem tetszőleges nem nulla komplex számok is lehessenek. Láthatjuk, hogy ilyenkor új jelenségeket is tapasztalhatunk.



(a) Koncergenciák



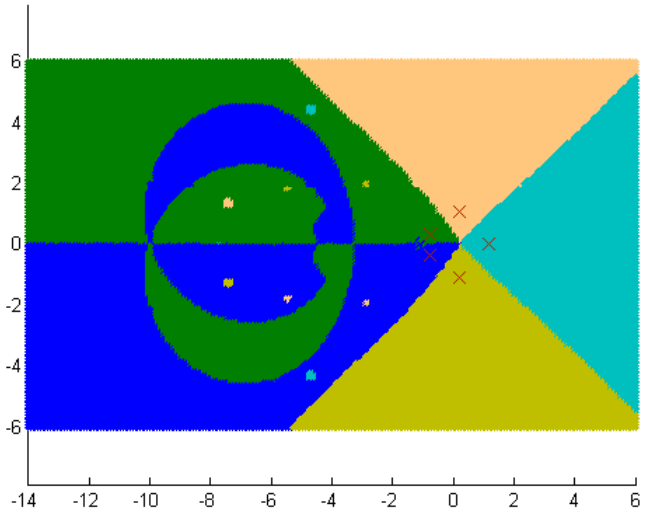
(b) Sebesség

4.4. ábra. $z = \sqrt[5]{z+1}$ futása

Nézzük a $z = \sqrt[5]{(2i-1)z+i}$ egyenletet. Mint az ábra is mutatja, a -0.3970 + 0.1931i gyök körül, melyhez a sötétkék pontok tartoznak, egy "örvény" jött létre a sárga színű pontokból, valamint ezt a gyököt csak elég jó közelítéssel találhatjuk meg, és akkor is csak lassan. Az "örvény" szélén lévő fekete pontok, olyan kezdőpontok, melyekben az iteráció kettő pont között oszcilláló végtelen ciklusba kerül. Egy konkrét példán bemutatva, ha az iterációt a $z_0 = -0.5 + 0.5i$ kezdőpontból indítjuk, akkor körülbelül harminc iterációs lépés múlva a z értékei állandósulnak -0.4561 - 0.3029i és -0.5730 + 0.4530i -ra egymást váltogatva. A nem konvergáló fekete pontokban mindig ez a ciklus lesz megfigyelhető.
Hasonló jelenség már harmadfokú polinomokra is megfigyelhető volt a Newton-módszer esetén, például a $(z-1)(z+0.5-0.33i)(z+0.5+0.33i)$ polinomnál is, ami a $z_0 = 0$ kezdőpontból indítva szintén egy kettő hosszú ciklusba jut. Ezúttal a perturbálás sem segít, mert ezek a végtelen ciklusok egész tartományokból vonzanak be pontokat mindkét esetben, s a kis számmal való elmozdítással továbbra is ezekben a tartományokban maradnánk. Így sajnos ez negatív a jelenség nem szűnt meg a vizsgált példáinkban.

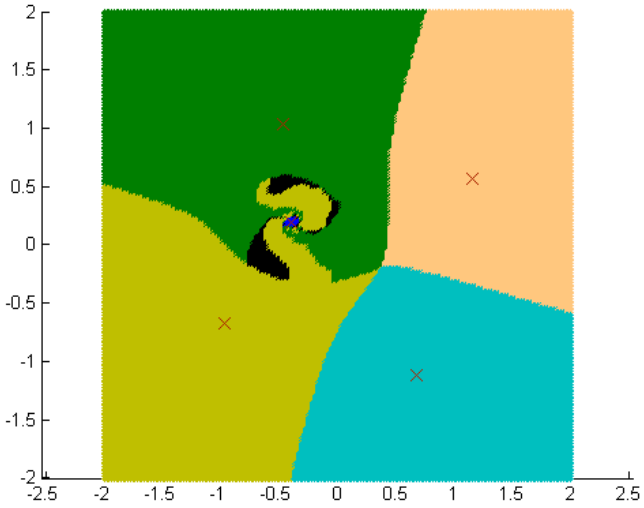
A $z = \sqrt[5]{(2i-5)z+1+4i}$ egyenletnek a megszokottnál nagyobb a korlátos tartománya, s elszórva egyéb megoldáshoz konvergáló, apró ponthalmazokat is tartalmaz. Hasonló jelenségekkel bíró, de szabályosabbnak tetsző ábrát kapunk a $z = \sqrt[5]{(2i)z+1+4i}$ egyenletre

Egy speciális esetet eredményez ha megengedjük, hogy $b = 0$ is lehessen. Ekkor ugyanis a 0 nem csak gyöke lesz az egyenletnek, hanem taszító pont is, mert az $az = 0$ egyenletnek is a $z = 0$ lesz a megoldása. Mivel most a 0 gyök egyben a korábbiakban belátott taszító pontja is az iterációnak, ezért nem lesz hozzá tartozó konvergenciatartomány. Így az ilyen típusú egyenleteknél a $z = 0$ megoldást csak akkor találjuk

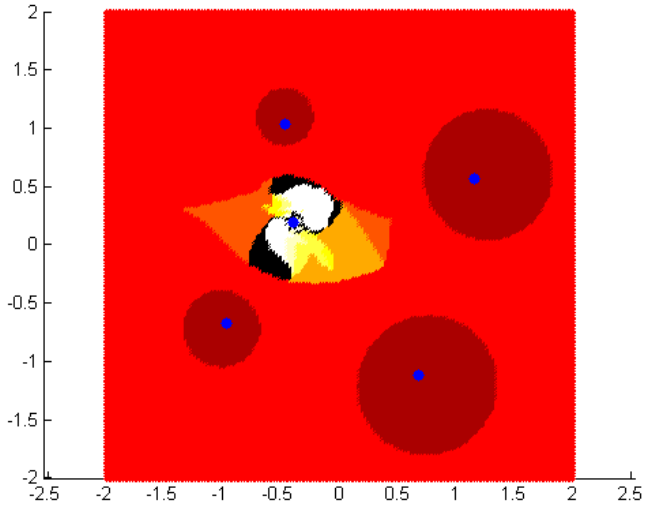


4.5. ábra. A $z = \sqrt[5]{z+1}$ konvergenciája

meg, ha pont a $z = 0$-ból indítjuk az iterációt. Erre az egyenletre futási példa a 4.14-es ábrák.

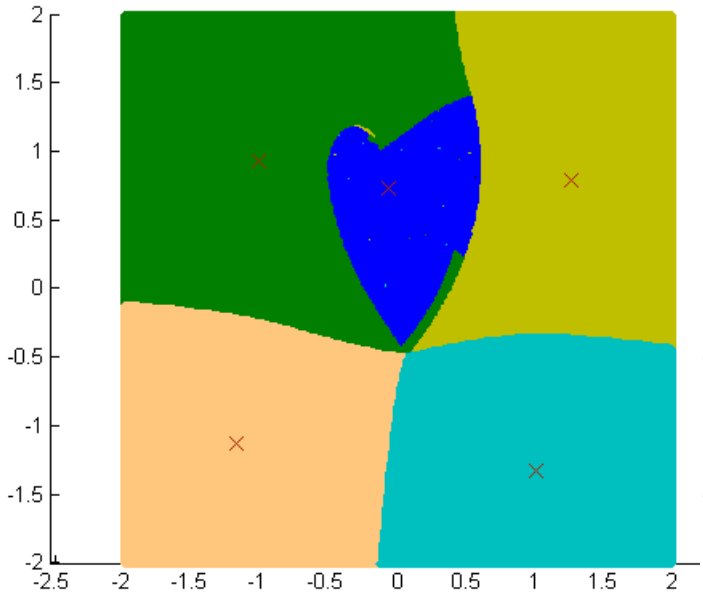


4.6. ábra. Konvergenciája

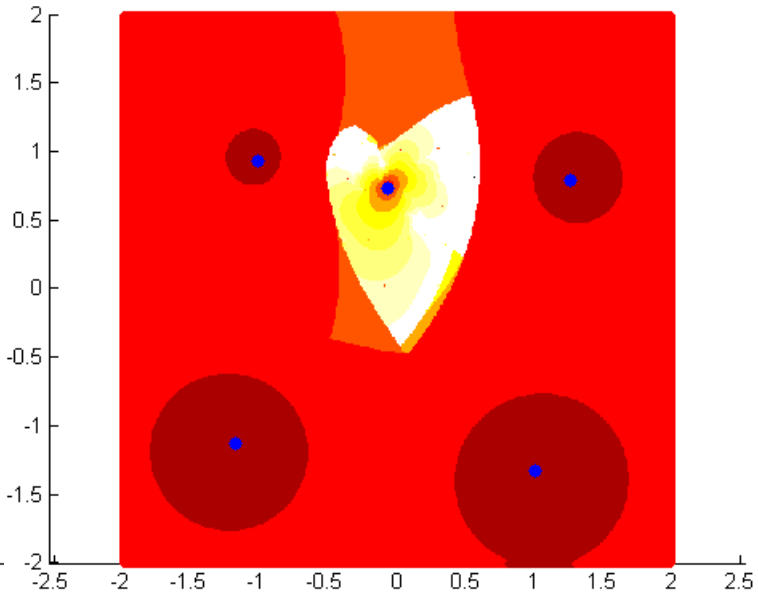


4.7. ábra. Sebessége

4.8. ábra. $z = \sqrt[5]{(2i-1)z+i}$ futása

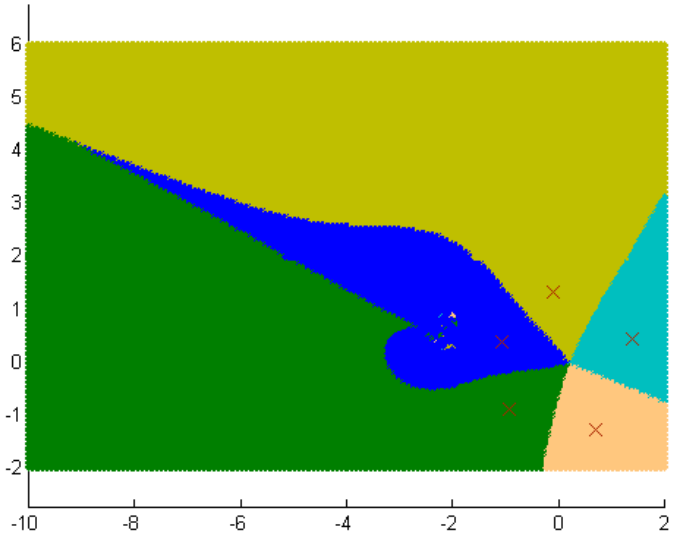


4.9. ábra. Konvergenciája

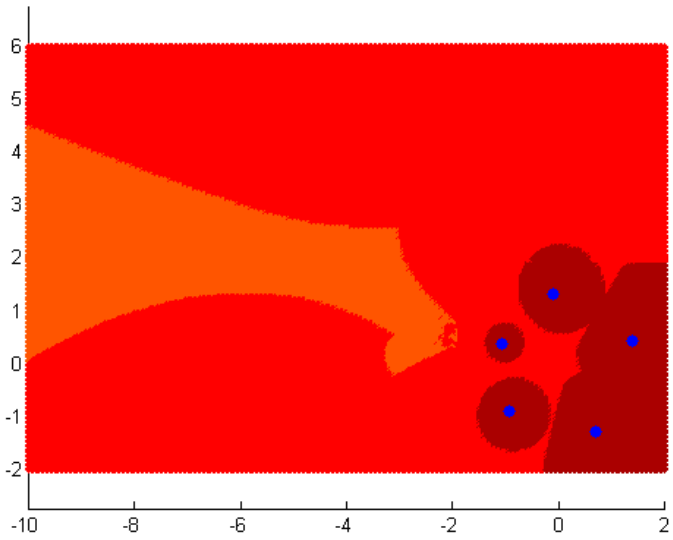


4.10. ábra. Sebessége

4.11. ábra. $z = \sqrt[5]{(2i-5)z+1+4i}$ futása

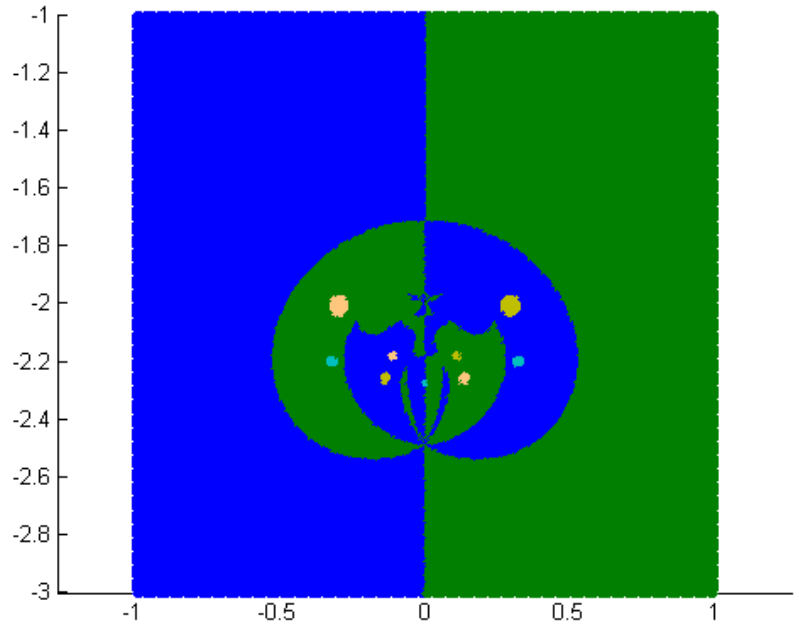


(a) Koncergenciák

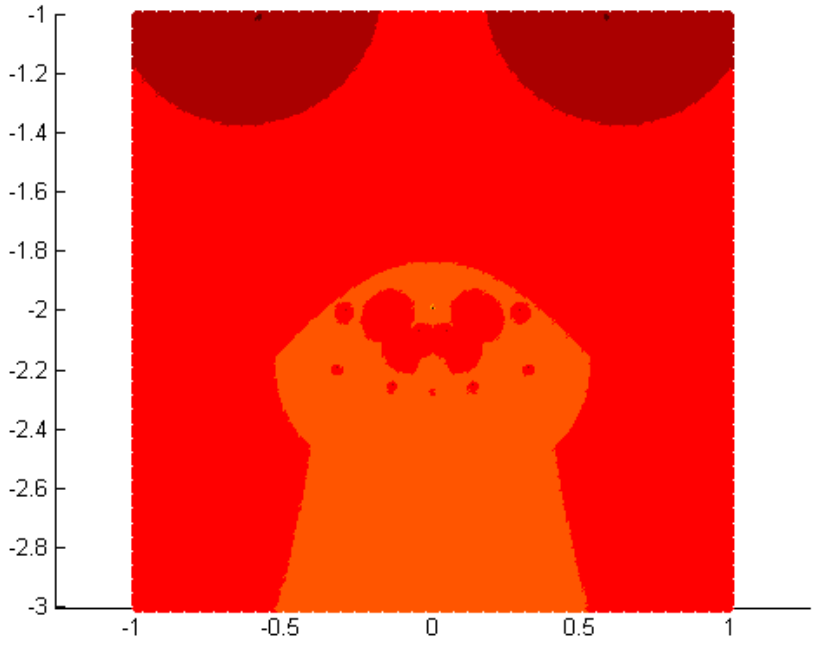


(b) Sebesség

4.12. ábra. $z = \sqrt[5]{(2i)z + 1 + 4i}$ futása

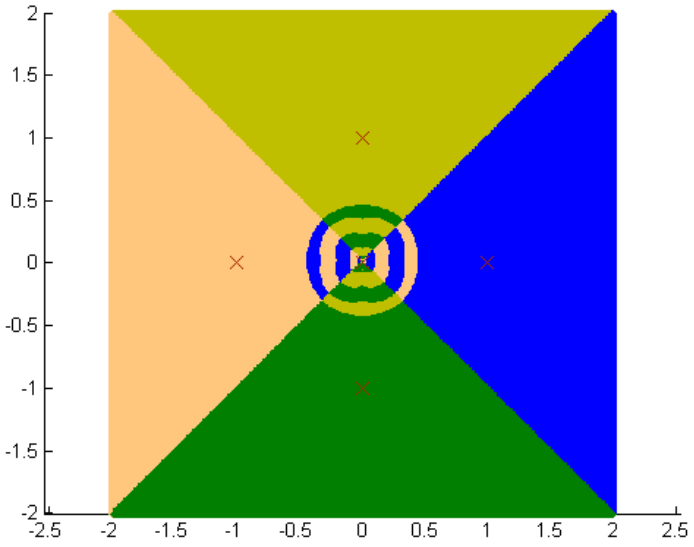


(a) Koncergenciák

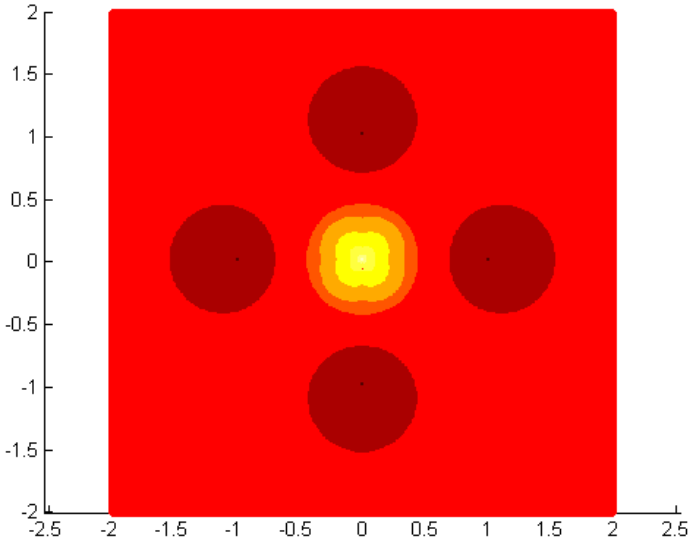


(b) Sebesség

4.13. ábra. $z = \sqrt[5]{2z + 4i}$ futása a gyökök ábrázolása nélkül a $-2i$ környezetében



(a) Koncergenciák



(b) Sebesség

4.14. ábra. $z = \sqrt[5]{z}$ futása

5. fejezet

Összefoglalás

A dolgozatban először bemutattuk a Newton-módszer működésének alapjait és alapvető tulajdonságait valós és komplex esetben egyaránt. Ezek után az általában vizsgált esetekkel ellentétben nem polinomokra, hanem különböző n-edik gyököket tartalmazó kifejezésekre vizsgáltuk a módszer futását. Először négyzetgyököket, majd ötödik gyököt tartalmazóakra. Mindkét esetben azt tapasztaltuk, hogy a klasszikus Newton-módszer nem alkalmazható meggondolások és módosítások nélkül, mert többértékű komplex függvényekről van szó, s figyelembe kell vennünk az adott n-edik gyök összes lehetséges értékét a megoldásunkban. Ezeket is figyelembe véve már tudtuk úgy módosítani a módszert, hogy a vizsgált egyenleteknek minden esetben megtalálhassuk az összes megoldását.

A. függelék

Newtonitsajat.m

A.1. Feladata

Tetszőleges, nem csak a második fejezetben vizsgált függvények zérushelyeit keresi meg Newton-módszer segítségével.

A.2. Használata

A MatLab parancs ablakába beírva a [z,fz,it]=Newtonitsajat(maxit,hiba,kezd) paranccsal hívhatjuk meg a függvényt.
Megjegyzés: A függvényt és deriváltját kézileg kell bevinni a program 3. és 4. soraiban (f és fder függvény fogantyúk z függvényében) a MatLab szintaxisnak megfelelően
Megjegyzés: ha nem adunk meg output paramétereket, akkor a visszakapott ans vektor a z vektornak felel meg.

A.2.1. Input paraméterek

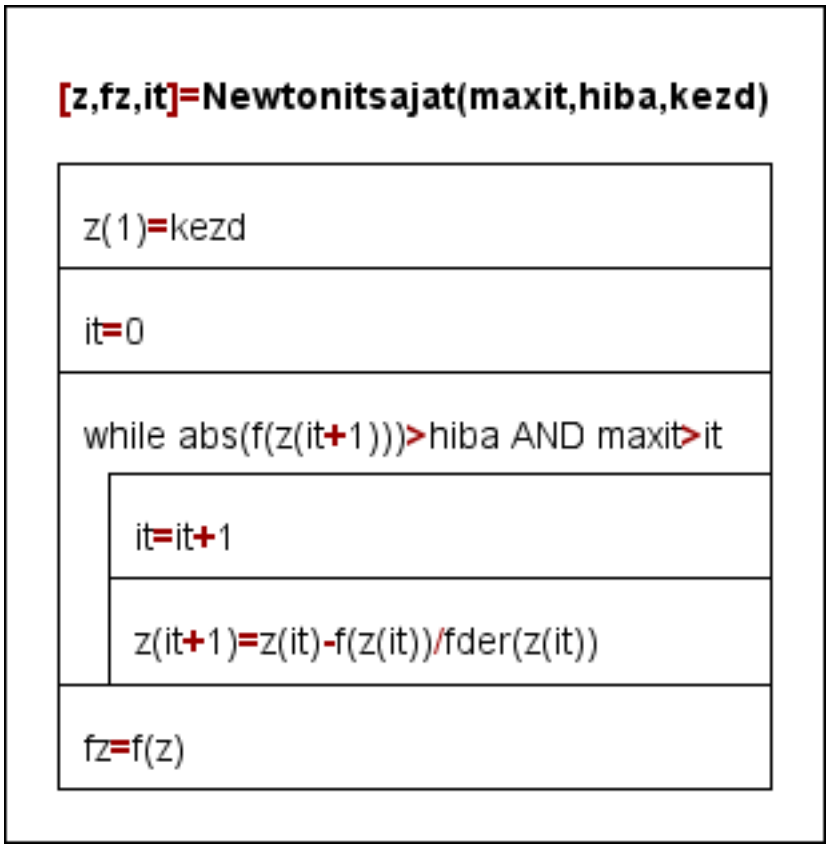
A megadás sorrendjében a következők:

- **maxit**: az iterációk maximális száma, célja, hogy nem konvergáló kezdőpontok ne hozzanak létre végtelen ciklust [pozitív egész szám]
- **hiba**: mekkora pontossággal kívánjuk megkapni az eredményt [pozitív szám]
- **kezd**: az iteráció kezdőpontja [komplex szám]

A.2.2. Output paraméterek

A megadás sorrendjében a következők:

- **z**: vektor, ami az egyes iterációs lépésekben kiszámolt pontokat tartalmazza, első tagja a kezdőpont
- **fz**: vektor, ami a z vektorban lévő pontokban felvett függvényértékeket tárolja
- **it**: az elvégzett iterációs lépések száma



A.1. ábra. Newton-módszer struktogramja

B. függelék

otod.m

B.1. Feladata

A $z = \sqrt[5]{az + b}$ alakú egyenletek Newton-módszerrel való megoldására szolgál. Alapjaiban az előző programmal egyezik meg, de nem célszerű más típusú egyenletek megoldására használni.

B.2. Használata

A MatLab parancs ablakába beírva a [z,fz,it]=otod(maxit,hiba,kezd,forg) parancsot hívhatjuk meg a függvényt.
Megjegyzés: A függvényt és deriváltját kézileg kell bevinni a program 7. és 8. (f és fder függvény fogantyúk z függvényében) soraiban a MatLab szintaxisnak megfelelően.

B.2.1. Input paraméterek

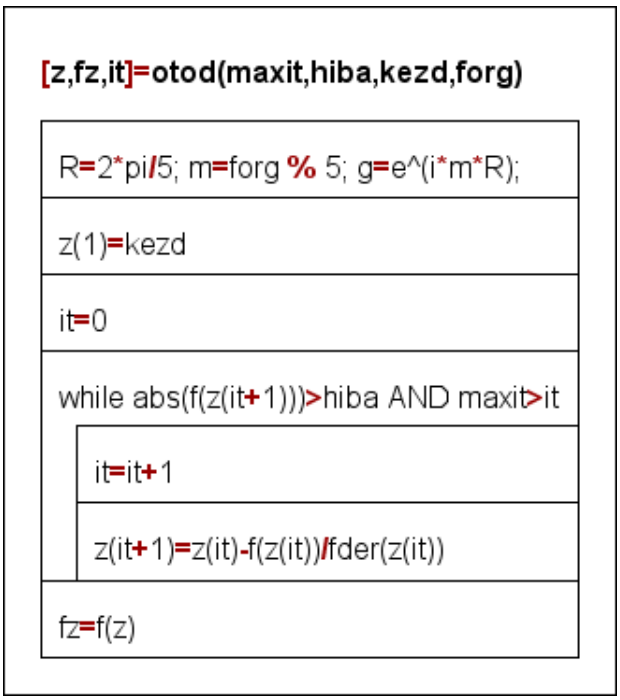
A megadás sorrendjében a következők:

- **maxit**: az iterációk maximális száma, célja, hogy nem konvergáló kezdőpontok ne hozzanak létre végtelen ciklust [pozitív egész szám]
- **hiba**: mekkora pontossággal kívánjuk megkapni az eredményt [pozitív szám]
- **kezd**: az iteráció kezdőpontja [komplex szám]
- **forg**: az ötödik gyökvonás MatLab szerinti alapértelmezett visszatérési értékét hányszor forgassuk el 72°-al [nemnegatív egész szám]

B.2.2. Output paraméterek

A megadás sorrendjében a következők:

- **z**: vektor, ami az egyes lépésekben kiszámolt pontokat tartalmazza, első tagja a kezdőpont
- **fz**: vektor, ami a z vektorban lévő pontokban felvett függvényértékeket tárolja
- **it**: az elvégzett iterációs lépések száma



B.1. ábra. otod.m struktogramja

C. függelék

javotod.m

C.1. Feladata

A $z = \sqrt[5]{az+b}$ alakú egyenletek Newton-módszerrel való megoldására szolgál, azzal módosítva, hogy minden lépésben mind az öt lehetséges következő pontot kiszámolja, és a legkisebb abszolútértékűt választja az f függvénynek megfelelő polinomba visszahelyettesítve. Az előző hasonló programhoz képest, ez már képes tetszőleges együtthatójú egyenleteket is megoldani pontosan.

C.2. Használata

A MatLab parancs ablakába beírva a z=javotod(maxit,hiba,kezd) parancsot hívhatjuk meg a függvényt.
Megjegyzés: A függvényt és deriváltját kézileg kell bevinni a program 14. és 15. (fk és fkder függvény fogantyúk) soraiban a MatLab szintaxisnak megfelelően. A g változójú szorzótagot ne módosítsuk.

C.2.1. Input paraméterek

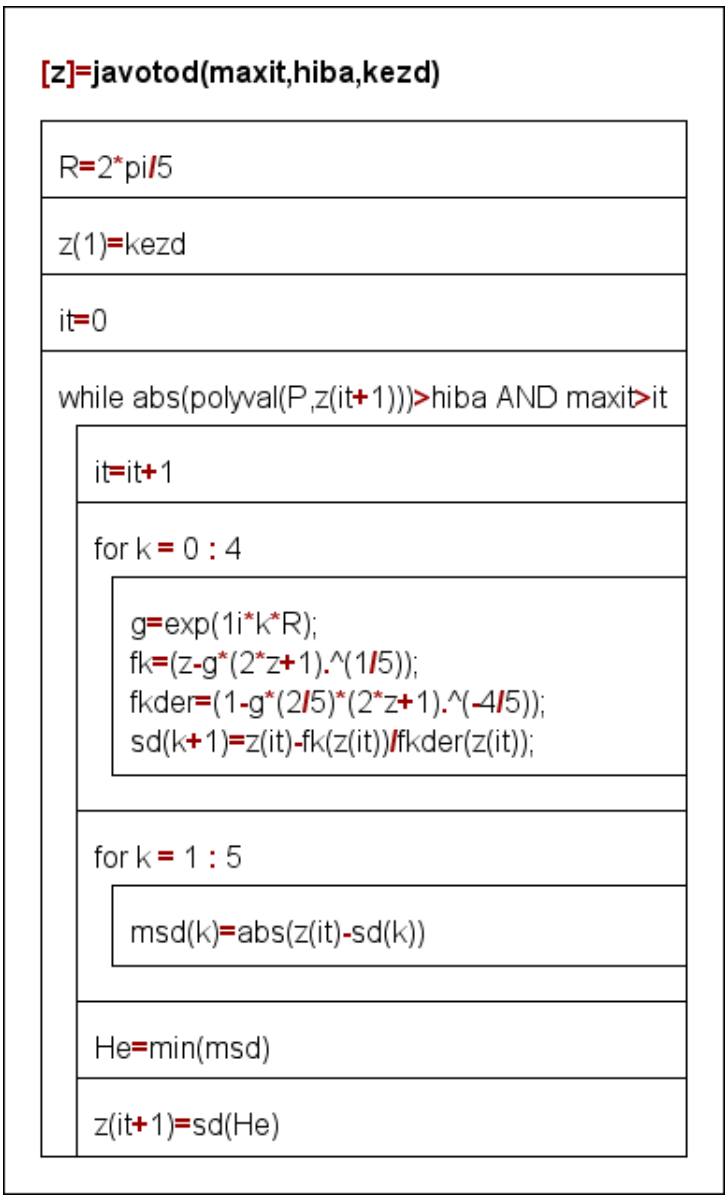
A megadás sorrendjében a következők:

- **maxit**: az iterációk maximális száma, célja, hogy nem konvergáló kezdőpontok ne hozzanak létre végtelen ciklust [pozitív egész szám]
- **hiba**: mekkora pontossággal kívánjuk megkapni az eredményt [pozitív szám]
- **kezd**: az iteráció kezdőpontja [komplex szám]

C.2.2. Output paraméterek

A megadás sorrendjében a következők:

- **z**: vektor, ami az egyes lépésekben kiszámolt pontokat tartalmazza, első tagja a kezdőpont



C.1. ábra. javotod.m struktogramja

D. függelék

Ábrázoló programok

D.1. newfrakt.m

D.1.1. Feladata

A komplex sík pontjait színezi ki különböző színűre, annak megfelelően, hogy a javotod.m programot abból a kezdőpontból indítva, adott iterációs lépésen belül, melyik gyökhöz fog konvergálni. A pontos megoldásokat vörös X jelöli, és a nem konvergáló kezdőpontokat fekete ponttal ábrázolja.
Megjegyzés: A program harmadik sorában szereplő P polinom meg kell hogy egyezzen, a javotodben vizsgált P polinommal a jó eredményhez.

D.1.2. Használata

A MatLab parancs ablakába beírva a newfrakt(xmin,xmax,ymin,ymax,pontok) parancsot hívhatjuk meg a függvényt. A javotod.m-ben használni kívánt maximum iterációs számot, és pontosságot a 20. programsorban módosíthatjuk.
Szép, sima ábrához a pontok paramétert elég nagynak, 200-500, és az ábrázolási tartomány végpontjait kellően köze érdemesl egymáshoz választani.

D.1.3. Input paraméterek

A megadás sorrendjében:

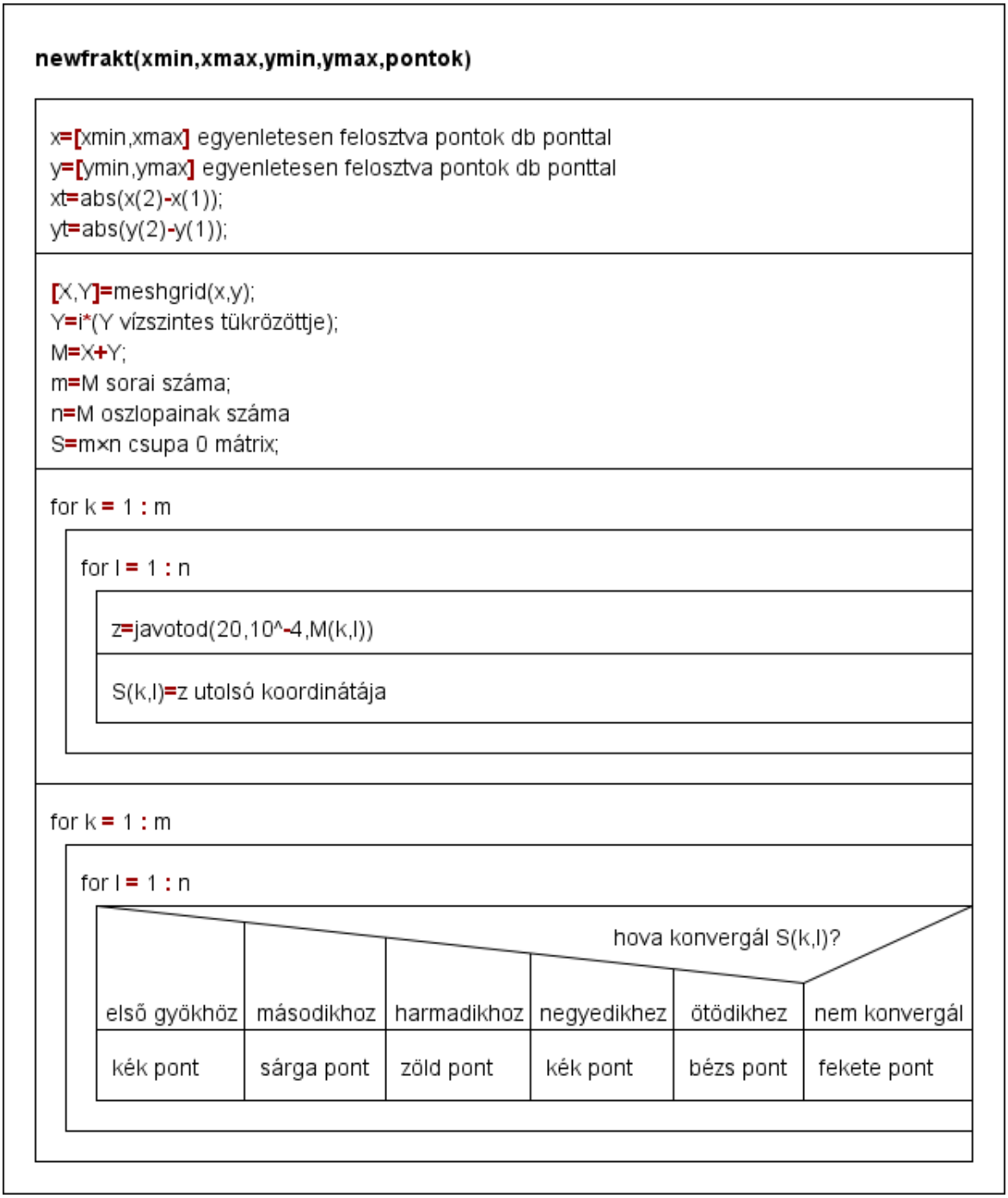
- **xmin**: vízszintes tengelyen az ábrázolási tartomány bal oldali végpontja
- **xmax**: vízszintes tengelyen az ábrázolási tartomány jobb oldali végpontja
- **ymin**: függőleges tengelyen az ábrázolási tartomány minimuma
- **ymax**: függőleges tengelyen az ábrázolási tartomány maximuma

- **pontok**: a min és max értékek között hány egymástól egyenlő távolságra lévő osztópontot vegyünk

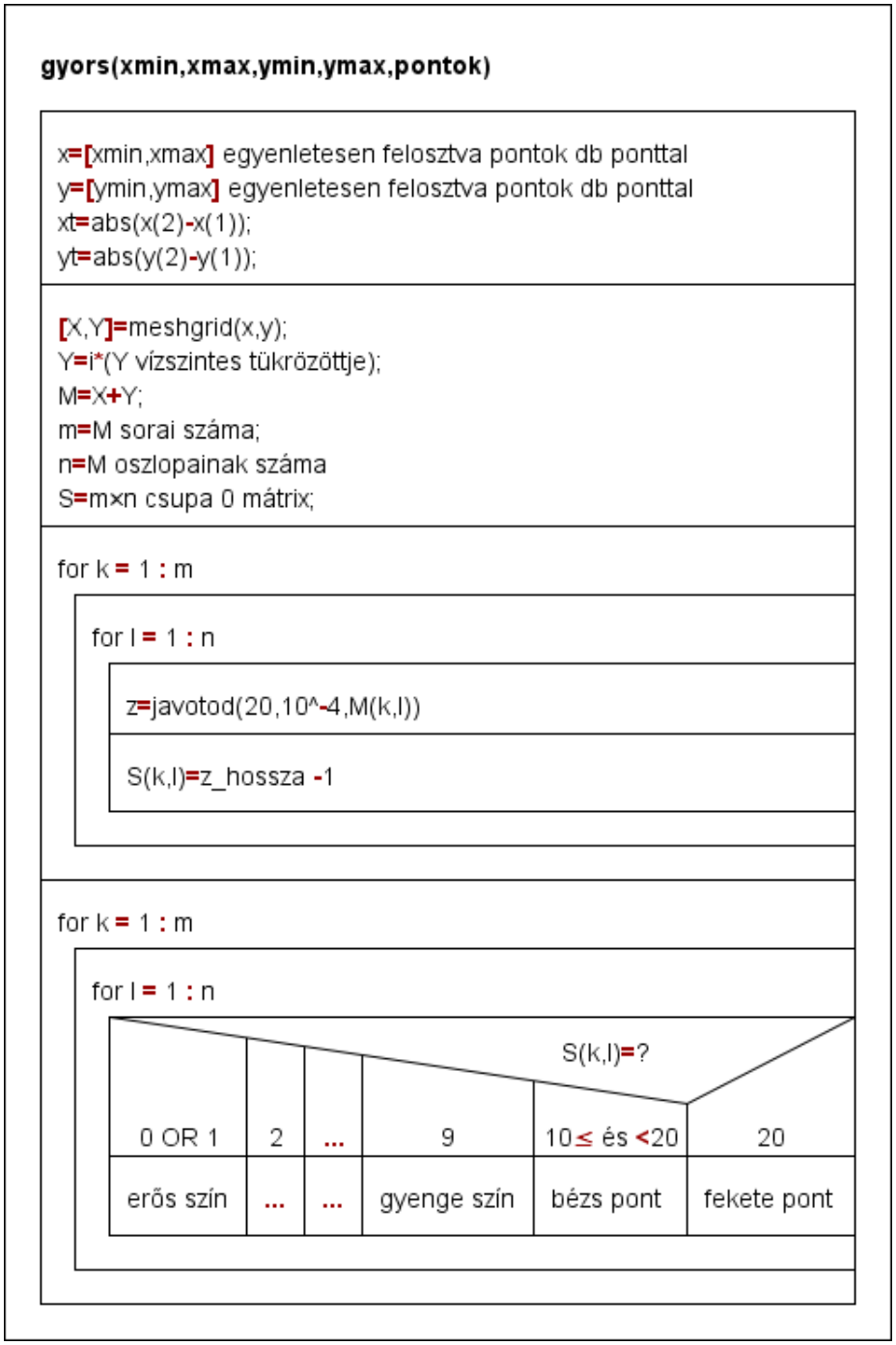
D.2. gyors.m

D.2.1. Feladata

A komplex sík pontjait színezi ki különböző színűre, annak megfelelően, hogy a javotod.m programot abból a kezdőpontból indítva, hány iterációs lépésre van szüksége ahhoz, hogy valamelyik gyökhöz odakonvergáljon. A pontos megoldásokat kék pont jelöli, és a nem konvergáló kezdőpontokat fekete ponttal ábrázolja.
A MatLab parancs ablakába beírva a gyors(xmin,xmax,ymin,ymax,pontok) paranccsal hívhatjuk meg a függvényt. A többi rávonatkozó használati utasítás és egyéb megjegyzés, illetve az input paraméterei teljes mértékben megegyeznek a newfrakt.m dokumentációjában szereplővel, azt kivéve, hogy a maximális iterációs szám nem változtatható.



D.1. ábra. newfrakt.m struktogramja



D.2. ábra. gyors.m struktogramja

Irodalomjegyzék

[1] Kerényi Péter: Gyökkeresés Iterációval (2011)
http://www.cs.elte.hu/ keppabt/documents/gyokkereses_iteracioval.pdf

[2] McMullen, Curt *Families of Rational Maps and Iterative Root-Finding Algorithms*, Annals of Mathematics 125 (1987), 467-493

[3] Gilbert, William J. *Generalizations of Newton's method*, Fractals, Vol 9., No.3 (2001), 251-262

[4] John Miller *Dynamics in one Complex Variable* (1990), 6-6

[5] Egyéb linkek
https://en.wikipedia.org/wiki/Newton's_method
https://www.mathworks.com/matlabcentral/newsreader/view_thread/6790
https://en.wikipedia.org/wiki/Newton_fractal