

Índex

RESUM	7
1 INTRODUCCIÓ	9
2 DESGLOSANT ELS FRACTALS	10
2.1 Definició de fractal	10
2.2 Història dels fractals	11
2.3 Benoît Mandelbrot	12
3 FORMES FRACTALS	16
3.1 Conjunt de Cantor	16
3.1.1 Construcció	16
3.2 Formes de Koch	17
3.2.1 La corba de Koch	17
3.2.2 El floc de neu de Koch	18
3.3 Formes de Sierpiński	20
3.3.1 Triangle de Sierpiński	20
3.3.2 Relació entre triangles: Sierpiński i Pascal	21
3.3.3 Catifa	24
3.3.4 Tetraedre	24
3.3.5 Esponja de Menger	25

3.4	Corba de drac	26
3.4.1	Construcció de la corba drac	26
3.5	Corbes que omplen l'espai	27
3.5.1	Construcció d'una corba de Peano	27
3.5.2	Altres corbes de Peano/corbes que omplen l'espai	28
3.6	Conjunt de Julia	29
3.7	Conjunt de Mandelbrot	32
3.7.1	Observem dos exemples	33
3.7.2	Matriu d'iteracions	36
4	CARACTERÍSTIQUES DELS FRACTALS	39
4.1	Característiques generals	39
4.2	Autosimilitud	39
4.2.1	Autosimilitud exacta	39
4.2.2	Autosimilitud aproximada	40
4.2.3	Autosimilitud estadística	41
4.3	Dimensions fraccionàries o fractals	41
4.3.1	Dimensió d'autosimilitud	41
4.3.2	Dimensió de comptatge de caixes/Haussdorff	43
4.4	Recursivitat	49
5	FRACTALS A LA NATURA	50
5.1	Geometria de la natura	50

5.1.1	Rius	50
5.1.2	Raigs	52
5.1.3	Muntanyes	52
5.2	La fractalitat dels éssers vius	53
5.2.1	Els pulmons	53
5.2.2	El sistema circulatori	54
5.3	Altres formacions naturals fractals	55
6	FRACTALS I APLICACIONS	56
6.1	L'impacte dels fractals a la tecnologia	56
6.1.1	Injectors de fluids fractals	56
6.1.2	Antenes fractals	57
6.1.3	Metamaterials fractals	58
6.1.4	Les innovacions de Fractal Antenna Systems	59
6.1.5	Computació i imatges	59
6.1.6	Fractus, SA	59
6.2	La cosmologia fractal	60
7	PART PRÀCTICA	62
7.1	Formes fractals amb Python	62
7.1.1	Llibreries i funcions	62
7.1.2	Conjunt de Cantor amb Python	63
7.1.3	Corba de Koch amb Python	63

7.1.4 Triangle de Sierpiński amb Python	64
7.1.5 Tetraedre de Sierpiński amb Python	65
7.1.6 Catifa de Sierpiński amb Python	65
7.1.7 Esponja de Menger amb Python	66
7.1.8 Corba de drac amb Python	66
7.1.9 Corba de Peano amb Python	67
7.1.10 Corba de Hilbert amb Python	68
7.1.11 Cojunt de Julia amb Python	68
7.1.12 Conjunt de Mandelbrot amb Python	69
7.2 Generació de paisatges amb fractals	70
7.2.1 Llibreries	70
7.2.2 Paisatges generats amb Mandelbrot	71
7.2.3 Paisatges generats amb Julia	74
7.2.4 Paisatges generats amb el fractal de Newton	78
8 CONCLUSIÓ	80
9 ANNEX	82
9.1 Rotació d'un segment en el pla	82
9.2 El conjunt dels nombres complexos	83
9.3 Mètode de Newton-Raphson	83
BIBLIOGRAFIA	84
Llibres	85

Pàgines web	85
-----------------------	----

Abans de començar amb el treball en si, donar agraïments a les persones que m'han ajudat a realitzar el TR. Primerament, al meu tutor de TR, per ajudar-me a plantejar el tema i donar-me algunes indicacions de com seguir. També, agrair al meu pare per ajudar-me a entendre i estudiar els conceptes matemàtics que es veuran en el treball.

RESUM

Els fractals representen un gir en el plantejament de la geometria natural. Aquests engloben teories i formes matemàtiques importants avui dia. Eventualment, es va començar a veure la seva viabilitat en aplicacions tecnològiques, formulació de teories i observació de la naturalesa i fenòmens naturals, les quals avui dia es continuen investigant i desenvolupant. En aquest treball es pretén estudiar a fons els fractals i respondre a la següent qüestió: És possible modelar en 3D un paisatge natural a partir d'un sol fractal? Es podran obtenir més d'un tipus de paisatge? Per a dur a terme aquest treball es va fer una recerca a fons sobre el concepte de fractal (característiques, tipus, història, etc.). Aquesta recerca es va realitzar revisant i mirant diferents fonts d'informació (llibres, textos, vídeos i llocs web). Posteriorment, usant programació Python, es va buscar construir diversos models de paisatges "naturals", que se semblessin a un paisatge natural real, a partir de seccions d'alguns fractals (conjunt de Mandelbrot, conjunts de Julia i el fractal de Newton). Els resultats obtinguts van ser la generació de diferents models de paisatge "semblants" a un model real. Després d'aquests resultats, es demostra que, a través dels conjunts de Mandelbrot, Julia i el fractal de Newton, observem que es poden generar paisatges semblants, a un paisatge natural a partir d'un fragment d'aquest fractal.

ABSTRACT

Fractals represent a major breakthrough in the approach to natural geometry. They include theories and mathematical forms that are important today. Eventually, their viability began to be seen in technological applications, formulation of theories and observation of nature and natural phenomena, which today continue to be investigated and developed. In this work we intend to study fractals in depth and answer the following questions: Is it possible to model in 3D a natural landscape from a single fractal? Is it possible to obtain more than one type of landscape? In order to carry out this work, a depth research on the concept of fractal (characteristics, types, history, etc.) was done. This research was done by reviewing and looking at different sources of information (books, texts, videos and websites). After that, using Python programming, we built several models of "natural" landscapes, resembling a real natural landscape, from sections of some fractals (Mandelbrot set, Julia sets and Newton's fractal). The results obtained were different landscape models "resembling" a real model. After these results, it is demonstrated that with the Mandelbrot, Julia and Newton fractal sets, we obtain that landscapes similar to a natural landscape can be generated from a fragment of this fractal.

1. INTRODUCCIÓ

El tema d'aquest treball de recerca (TR) se centra en els fractals i en el seu estudi. Les motivacions personals per escollir aquest camp de les matemàtiques com a temàtica del meu TR, va ser la meva fascinació per aquestes formes, les quals havia vist moltes vegades en llibres, vídeos i també a la natura. Això em va enfilar cap a un interès més acadèmic del tema, és a dir, estudiar sobre el concepte de fractal. Els meus objectius teòrics són estudiar les teories matemàtiques involucrades en els fractals. També es busca conèixer quines han sigut les seves repercussions o implicacions teòriques i pràctiques dins de la ciència, més enllà de les matemàtiques. Com a objectius pràctics es busca aprendre a utilitzar un llenguatge de programació (Python), per després elaborar els meus paisatges fractals i respondre a les següents qüestions: És possible modelar en 3D un paisatge natural a partir d'un sol fractal? Es podran obtenir més d'un tipus de paisatge? Amb aquesta premissa, el treball es va orientar inicialment (a part d'en un estudi explicatiu) en un àmbit més tecnològic, sent que es buscava construir una antena amb forma fractal, però, per diverses circumstàncies es va decidir orientar a la seva relació amb els paisatges.

Per desenvolupar aquest treball la metodologia a seguir va ser una primera part recopilatòria d'informació sobre el tema i els temes amb els quals es relaciona transversalment (annexos). Aquesta informació es va obtenir de diverses fonts d'informació: pàgines web de divulgació, vídeos de caràcter divulgatiu, i el més important llibres/textos científics. Per a la part pràctica, s'ha utilitzat programació de codi obert Python i algunes llibreries pròpies per a realitzar-les. En l'àmbit estilístic s'ha utilitzat el programa L^AT_EX, orientat en la creació de textos científics. L'estrucció del treball s'ha dividit en 7 parts: una primera part on es desglossa l'origen dels fractals, una segona en què es comenta individualment les formes fractals, una tercera on s'exposa les característiques comunes de tota forma fractal, seguit de dues parts, una orientada en la presència dels fractals a la natura i altre a la ciència. Les últimes dues parts se centren en la pràctica i els annexos.

2. DESGLOSANT ELS FRACTALS

2.1 Definició de fractal

Si s'ha de definir de manera simple i concreta "que és una fractal?" la resposta més senzilla seria la geometria del món. La curiositat per tot el que ens envolta és una característica intrínseca dels éssers humans, la qual està present dins nosaltres des que obtenim consciència a la nostra infantesa. Per fer-ho utilitzem descripcions, que obtenim d'allò que els nostres sentits ens permet veure.

La geometria clàssica euclidiana és suficient per explicar la forma dels elements, ara bé, la cosa es dificulta a l'hora d'afrontar formes més complexes: la silueta dels arbres en hivern, les formes de les ones en trencar-se, la ramificació pulmonar, recorregut de la costa, etc. Què són aquestes formes? Com les classifiquem? La resposta a aquestes preguntes són els fractals, formes geomètriques amb molta complexitat, resultat de la repetició d'una forma geomètrica específica, això, gràcies a una particularitat d'aquestes formes anomenada complexa d'autosimilitud, el qual tot i que es desglossarà més endavant, es pot definir com la propietat d'una forma de mantenir-se semblant o quasi semblant a diferents escales.



Figura 2.1: Les llavors d'un gira-sol, disposades en dues famílies d'espirals entrelaçades [7].

2.2 Història dels fractals

Els fractals són el resultat de la necessitat de categoritzar a aquelles formes o figures que la geometria euclidiana no podia definir. Històricament, els humans utilitzàvem la geometria clàssica o euclidiana, descoberta per Euclides fa 2500 anys, per definir a aquelles formes que veiem del nostre voltant. Tot i això, el pas del temps i el desenvolupament de les matemàtiques va necessitar l'aparició d'un nou concepte per aquelles figueres exageradament complexes o "informes".

Al segle XIX, la concepció de dimensió que es tenia en aquella època va ser qüestionada pels matemàtics quan investigant aquestes figueres complexes van descobrir que tenien la particularitat de tenir dimensió fraccionària. El principi d'aquest esdeveniment es va donar

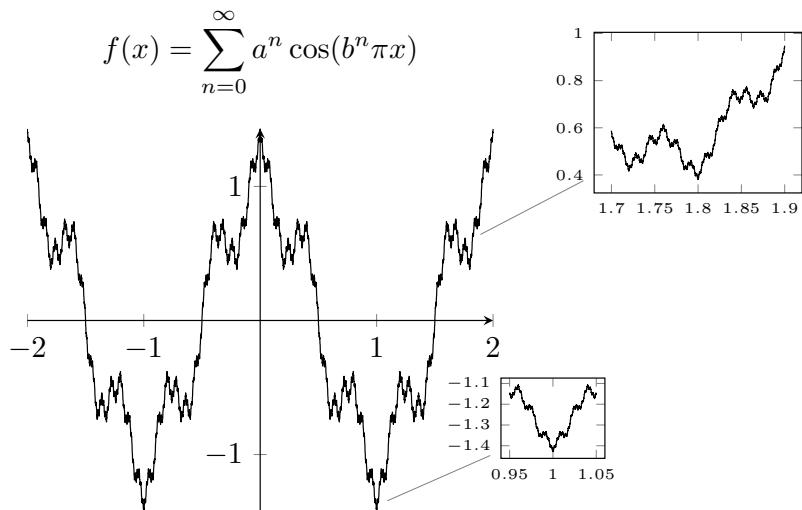


Figura 2.2: Representació de la funció de Weierstrass.

amb el matemàtic Karl Weierstrass, quan en 1872, va dibuixar la seva funció de Weierstrass (veure Figura 2.2). En el mateix segle, van començar a aparèixer conceptes cada vegada més geomètrics i menys algebraics. Aquests conceptes podien construir-se partint d'una figura inicial (iniciador), a la qual s'aplicaven una sèrie de construccions geomètriques senzilles. La sèrie de figures obtingudes s'aproximava a una figura límit que corresponia al que avui diem conjunt fractal. Així, en 1904, Helge Von Koch va definir una corba amb propietats similars a la de Weierstrass: la corba de Koch. En 1915, Waclaw Sierpiński va construir el seu triangle i, un any després, la seva catifa.

Altres dos grans matemàtics que es van endinsar en el camp dels fractals van ser Georg Cantor i Giuseppe Peano. Aquests dos matemàtics van pertànyer a un grup que per la seva

labor va ser reconegut cap al final de la "crisi dels fonaments" que acabaria en 1925. Un dels problemes tractats durant aquesta crisi dels fonaments va ser que les "matemàtiques clàssiques" eren les eines adequades per a estudiar les estructures regulars de la geometria d'Euclides i l'evolució contínua de la dinàmica de Newton, però que eren ineficients amb aquelles formes o estructures que no encaixaven en aquesta mena de geometria.

Sota aquesta premissa es necessitava una "nova matemàtica" en la qual encaixessin aquelles formes dissidents amb els patrons de Newton i Euclides, com són la corba de Cantor i la corba de Peano, les quals són capaces "d'omplir el pla". La qüestió sorgeix del següent fet: en ser una corba hauria de tenir dimensió 1, però, si pot emplenar un quadrat la seva dimensió ha de ser 2, per tant, quina és la dimensió d'aquestes estructures? Aquests nous elements no estaven contemplats en la matemàtica tradicional, van ser considerats com a "monstres matemàtics".

L'any 1919, Félix Hausdorff va introduir la primera manera d'observar i estudiar aquest tipus de formes en la vida real, la dimensió de Hausdorff-Besicovitch, actualment coneguda com a dimensió fractal. Uns anys més tard, el rus Andrei Kolgomórov describia una eina similar a la de Hausdorff que seria posteriorment coneguda com l'entropia de Kolmogórov. Si bé les aportacions d'aquests matemàtics van ser vitals per la formació del que serien els fractals, és al matemàtic Benoît Mandelbrot, a qui hem de mirar com a referent de la geometria fractal.

2.3 Benoît Mandelbrot



Benoît Mandelbrot [19] va néixer l'any 1924 a Varsòvia, en el si d'una família jueva lituana, en la qual, va tenir als seus dos oncles com a referents, els qui ho van introduir en les matemàtiques. Amb 11 anys, l'any 1936, abandona Polònia a causa de la Segona Guerra Mundial, traslladant-se a França, on la seva educació va caure en mans del seu oncle Szolem Mandelbrot. Aquí comença la trajectòria acadèmica de

Mandelbrot, que igual que la seva gran aportació a les matemàtiques, va ser irregular.

Va realitzar els seus estudis universitaris a la Universitat de Lió, per a després ingressar l'École polytechnique. Temps després, va emigrar cap als Estats Units on va obtenir un màster en aeronàutica a l'Institut Tecnològic de Califòrnia. Va tornar a París per obtenir el doctorat a la Universitat de París l'any 1952. Més endavant va tornar als EUA se'n va anar al MIT i després a l'Institut d'Estudis Avançats de Princeton, on va ser estudiant de John von Neumann. Més endavant va començar a treballar a l'IBM Research.

Paral·lelament, Mandelbrot es va dedicar a investigar sobre aquelles formes que escapaven de la geometria convencional, va examinar els treballs previs de Hausdorff i Gaston Julia, d'aquest últim, va ser alumne i va estudiar molts dels seus articles, els quals no el van interessar en aquell moment. Més endavant, l'any 1967, Mandelbrot va publicar l'article "Quina longitud té la costa de la Gran Bretanya?" per a la revista Science. En aquest article, Mandelbrot explicava les seves primeres idees respecte als fractals, que encara el concepte no estava ben definit. En el llibre es planteja el fet que depenen de la precisió de l'instrument de mesura que s'utilitza (si és més petit) s'obtindrà un resultat més proper a la realitat. Amb aquest context Mandelbrot va observar un fenomen inusual. A mesura que es feia zoom en una zona de costa trobem una porció semblant a la primera. Més endavant, aquest concepte que va ser batejat com autosimilitud. També parlava del concepte de la dimensió Hausdorff-Besicovitch, que postula l'existència de dimensions fraccionàries per aquestes formes. Mentrestant va seguir treballant a l'IBM on se li va encarregar examinar



Figura 2.3: En aquesta imatge es pot observar el fenomen que Mandelbrot va observar respecte a la precisió en la mesura.

les anomalies sonores que sofrien les transmissions electròniques. Allí, Mandelbrot va descobrir que, si es feia una gràfica amb les dades d'aquestes pertorbacions, es podia observar "autosimilitud" en totes les escales d'augment. El més curiós és que aquesta autosimilitud també podia apreciar-se en altres gràfics estadístics com per exemple els preus del cotó,

les variacions de les cotitzacions de la borsa, les freqüències de les paraules en anglès i les fluctuacions dels líquids turbulents. Amb aquest fet, Mandelbrot va tornar a examinar els treballs de Gaston Julia, aconseguint construir la seva funció de Julia. Per tant, en combinar els treballs de Julia amb els seus estudis, aplicats a l'ordinador va assentar les bases del que més endavant serien els fractals.

Mandelbrot va batejar al seu descobriment amb el nom de fractal, paraula originària del llatí *fractus*. El significat de *Fractus* és trencat o irregular, definició que concorda amb el que va descobrir Mandelbrot i el motiu pel qual va utilitzar aquesta paraula. Temps després va batejar aquest corrent com geometria fractal, posicionant-se com el principal creador de la geometria fractal. L'any 1982 va publicar el seu llibre "Fractal Geometry of Nature", on explicava les seves investigacions i tesis sobre els fractals.

D'altra banda, Mandelbrot també es va interessar pels patrons del mercat financer. Investigant sobre aquestes va descobrir que els canvis de preu en el mercat no segueixen els convencionals patrons gaussians, sinó que seguien un esquema més proper als postulats per Lévy en els seus articles sobre el mercat. El concepte de fractals es va aplicar posteriorment a l'estudi dels mercats financers en el context dels fractals.

A la fi de la dècada de 1980, Mandelbrot va utilitzar dades de ticks de Olsen & Associates a Zúric per a aplicar la seva teoria dels fractals a l'estructura microestructural dels mercats. Aquesta col·laboració va donar lloc a la publicació dels primers treballs exhaustius sobre la llei d'escala en finances, que confirma la idea de Mandelbrot sobre la naturalesa fractal de la microestructura dels mercats. Anys després, l'any 1985 va ser guardonat amb el premi Barnard Medal for Meritorious Service to Science. Posteriorment, va guanyar la medalla Franklin. L'any 1987 va guanyar el premi Alexander von Humboldt, la medalla Steindal l'any 1988, la medalla Nevada al 91 i de més premis. Finalment, va morir l'any 2010 amb 85 anys a causa de càncer de pàncrees.

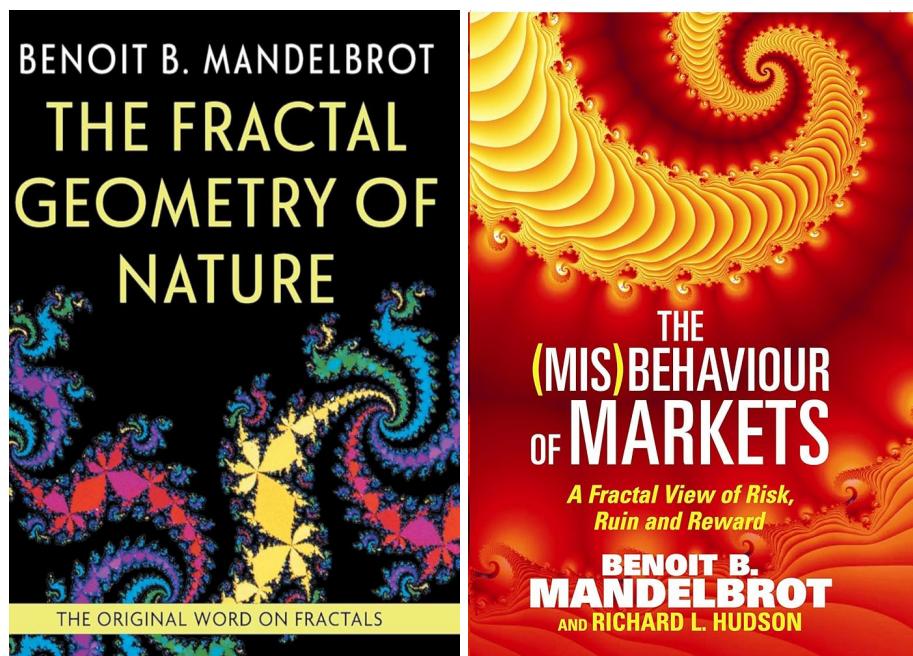


Figura 2.4: A l'esquerra el llibre "Fractal Geometry of Nature" (1982) i a la dreta "The (Mis)Behavior of Markets" (2004).

3. FORMES FRACTALS

3.1 Conjunt de Cantor

El conjunt de cantor, és un conjunt fractal batejat per Georg Cantor l'any 1883. Aquest conjunt és considerat com la primera estructura fractal registrada. El conjunt de Cantor destaca per ser un subconjunt de l'interval real $[0, 1]$ i contradir la intuïció relativa de la mida d'un cos geomètric, és a dir, és un conjunt de mesura nul, però no és buit.

3.1.1 Construcció

Comencem amb l'interval tancat $[0, 1]$, [5, p. 66].



Ara treurem l'interval (obert) $\left(\frac{1}{3}, \frac{2}{3}\right)$, és a dir, treurem el terç central de $[0, 1]$, però no els nombres $\frac{1}{3}$ i $\frac{2}{3}$. Això deixa dos intervals $\left[0, \frac{1}{3}\right]$ i $\left[\frac{2}{3}, 1\right]$ de longitud $\frac{1}{3}$ cadascun i es completa un pas bàsic de la construcció.



Ara repetim, ens fixem en els intervals restants $\left[0, \frac{1}{3}\right]$ i $\left[\frac{2}{3}, 1\right]$ i eliminem els seus terços mitjans, la qual cosa dona lloc a quatre intervals de longitud $\frac{1}{9}$.



Continuem així. En altres paraules, existeix un procés de retroalimentació en el qual es genera una seqüència d'intervals (tancats): un després del primer pas, dos després del segon, quatre després del tercer, vuit després del quart, etc. (és a dir, 2^n intervals de longitud $\frac{1}{3^n}$ després de l'enèsim pas). La figura 3.1 mostra la seva construcció.

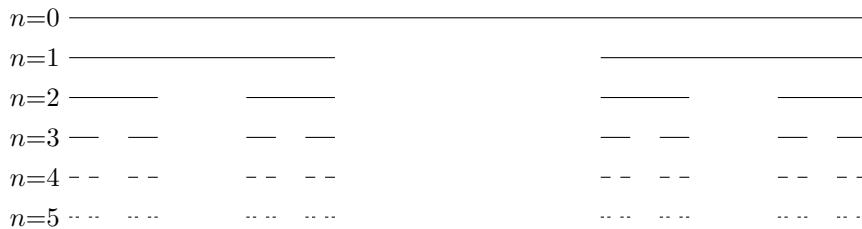


Figura 3.1: Passos inicials del conjunt de Cantor.

3.2 Formes de Koch

3.2.1 La corba de Koch

La corba de Koch [16] és un fractal que va ser conegut l'any 1904 en un article anomenat: *Sobre una corba continua que no té tangents i obtinguda per mètodes de la geometria elemental*. L'article va ser publicat per Helge von Koch, un matemàtic suec (1870-1924), conegut per la teoria de nombres.

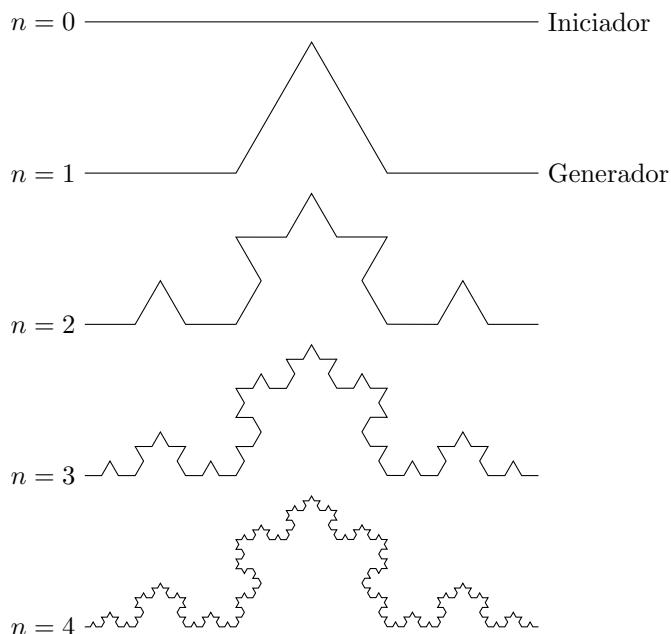


Figura 3.2: Construcció de la corba de Koch. La construcció de la corba de Koch es desenvolupa en etapes. En cada etapa, el nombre de segments de la línia augmenta per un factor de 4.

La construcció de la corba comença amb una recta al pla, anomenada **iniciador**. Aquesta es divideix en terços i després se substitueix el terç central per un triangle equilàter sense la seva base, obtenint una nova figura de quatre parts. D'aquesta figura obtinguda s'utilitzarà

una reducció en les següents etapes anomenada **generador**. Ara, s'agafa cada un dels segments de la línia obtinguda i es reproduceix el generador en cada una de les parts i així successivament. En tot el procés de construcció es pot observar la característica d'autosimilitud, sent que cadascuna de les quatre parts en el k -esim pas és de nou una versió reduïda de l'anterior (veure Figura 3.2).

3.2.2 El floc de neu de Koch

Relacionat amb la corba de Koch trobem el floc de neu de Koch. El floc de Koch, també anomenada illa o estrella de Koch, és una altra fractal obra de Helge von Koch. Es pot construir de dues formes, la primera i més senzilla (veure Figura 3.3), és ajuntant 3 corbes de Koch pels seus extrems. Així s'obté el famós floc de Koch, una figura d'àrea finita i perímetre infinit.

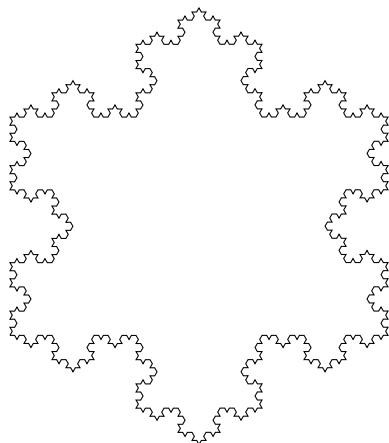
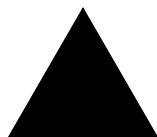
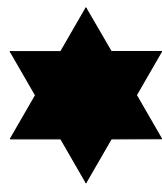


Figura 3.3: El floc de neu de Koch. Ajuntem 3 corbes de Koch pels seus extrems.

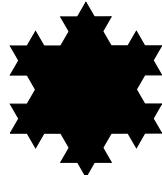
L'altra manera de desenvolupar el floc de Koch començaria amb un triangle equilàter (omplert) de longitud lateral igual a L . Aquest triangle serà en el nostre iniciador.



En el segon pas, al terç mitjà de cada costat del triangle afegim un nou triangle equilàter de longitud lateral igual a $\frac{1}{3}L$. D'aquesta manera obtenim el generador.



A continuació, al terç mitjà de cada costat del generador afegim triangles equilàters de longitud lateral igual a $\frac{1}{3} \left(\frac{L}{3} \right) = \frac{L}{9}$.



Finalment, repetim aquest procés de manera seqüencial fins a obtenir la Figura 3.4

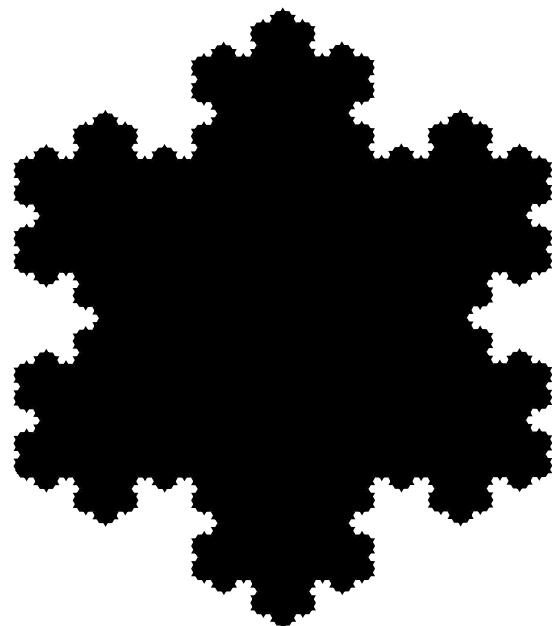


Figura 3.4: El floc de neu de Koch. 5 iteracions del floc de Koch.



Figura 3.5: Flocs naturals. El floc de neu de Koch comparteix similitud amb els flocs de la vida real.

3.3 Formes de Sierpiński

Un dels grans contribuïdors al camp dels fractals fou Sierpiński. Waclaw Sierpiński fou un matemàtic polonès (1883-1969), amb grans contribucions dins de les matemàtiques com la teoria de conjunts i la de números. També es va interessar pels fractals, molt abans que Mandelbrot creés el terme fractal. Sierpiński va construir diversos objectes fractals:

- Triangle de Sierpiński
- Catifa de Sierpiński
- Corba de Sierpiński
- Tetraedre de Sierpiński
- Piràmide de Sierpiński

3.3.1 Triangle de Sierpiński

El més conegut de les fractals de Sierpiński, i la seva obra magna, és el triangle de Sierpiński [5]. Aquest triangle es forma des d'un triangle equilàter al qual se li va removent recursivament triangles més petits fora del centre. A diferència de les altres formes de Sierpiński, la seva existència és milers d'anys a Sierpiński, ja que era utilitzat per antigues civilitzacions com a patró de decoració. Així i tot, va ser batejada com "de Sierpiński" pel fet que ell va ser el primer a estudiar les seves propietats matemàtiques.

La construcció geomètrica del triangle Sierpiński és la següent [5, p. 76]. Comencem amb un triangle ple en el pla i després li apliquem un esquema repetitiu d'operacions. Després es localitza els punts mitjans dels seus tres costats i s'ajunten amb els antics vèrtexs del triangle original, aquests punts mitjans defineixen quatre triangles congruents dels quals eliminem el del centre. Això completa el pas bàsic de la construcció. En altres paraules, després del primer pas tenim tres triangles congruents els costats dels quals tenen exactament la meitat de la grandària del triangle original i que es toquen en tres punts que són vèrtexs comuns de dos triangles contigus.

Ara seguim el mateix procediment amb els tres triangles restants i repetim el pas bàsic tantes vegades com ho desitgem. És a dir, comencem amb un triangle i després produïm 3, 9, 27, 81, 243... triangles (veure Figura 3.6), cadascun dels quals és una versió escalada

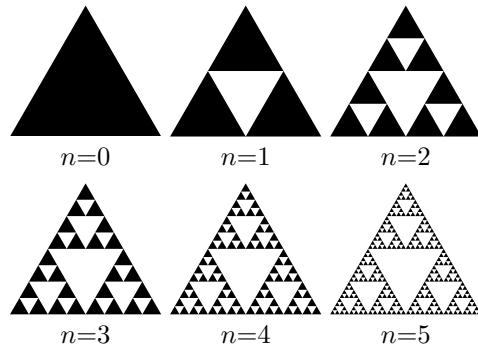


Figura 3.6: Triangle de Sierpiński. Els passos bàsics de construcció del triangle de Sierpiński.

exacta dels triangles en el pas anterior. El triangle de Sierpiński és el conjunt de punts en el pla que queden si es duu a terme aquest procés infinitament sovint. La característica de l'autosimilitud i recursivitat són presents en el procés de construcció, és a dir, cadascuna de les tres parts del pas és una versió lluïda - per un factor de 2 - de tota l'estructura del pas anterior.

3.3.2 Relació entre triangles: Sierpiński i Pascal

El triangle de Sierpiński és un objecte que ha aconseguit plasmar-se cap a altres conceptes dins de les matemàtiques. Tan així, que el triangle del matemàtic polonès està relacionat amb un altre triangle famós, el triangle de Pascal/Tartaglia.

El triangle de Pascal, o també anomenat de Tartaglia [5, p. 80], es defineix com una piràmide numèrica en la qual es troben representats els coeficients binominals de l'equació 3.1. La seva construcció comença amb el número 1 al vèrtex superior i s'expandeix cap avall amb la resta de nombres de manera seqüencial. D'aquesta manera, el triangle hauria de tenir l'estructura següent:

El disseny del triangle de Pascal s'obté amb la següent fórmula:

$$c(n, k) = \binom{n}{k} = \frac{n!}{(n-k)!k!}, \quad n \in \mathbb{N}, \quad k = 0, \dots, n. \quad (3.1)$$

Aquí, n i k representen el número files i columnes que té el triangle respectivament. En la Taula 3.1 es poden observar les 4 primeres files del triangle de pascal.

Doncs bé, una característica del triangle de Pascal és que es pot obtenir el triangle de

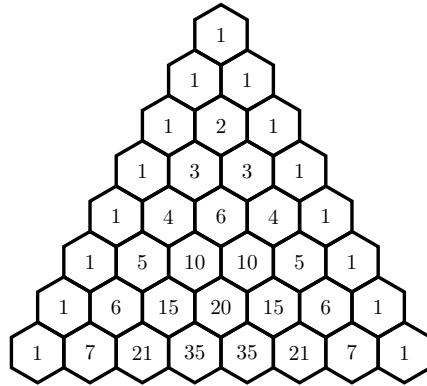


Figura 3.7: Triangle de Pascal/Tartaglia. Il·lustració de les primeres 8 files del triangle.

n	k	$c(n, k)$	valors
0	0	$c(0,0)$	1
1	0,1	$c(1,0), c(1,1)$	1, 1
2	0,1,2	$c(2,0), c(2,1), c(2,2)$	1, 2, 1
3	0,1,2,3	$c(3,0), c(3,1), c(3,2), c(3,3)$	1, 3, 3, 1

Taula 3.1: Les primeres 4 files del triangle de Pascal.

Sierpiński dins d'aquest si s'utilitza un codi de colors per diferenciar els nombres parells dels imparells (veure Figura 3.8).

També es pot formar utilitzant un codi de colors basant-se en els nombres divisors d'un nombre $m \in \mathbb{N}$ (veure Figura 3.9).

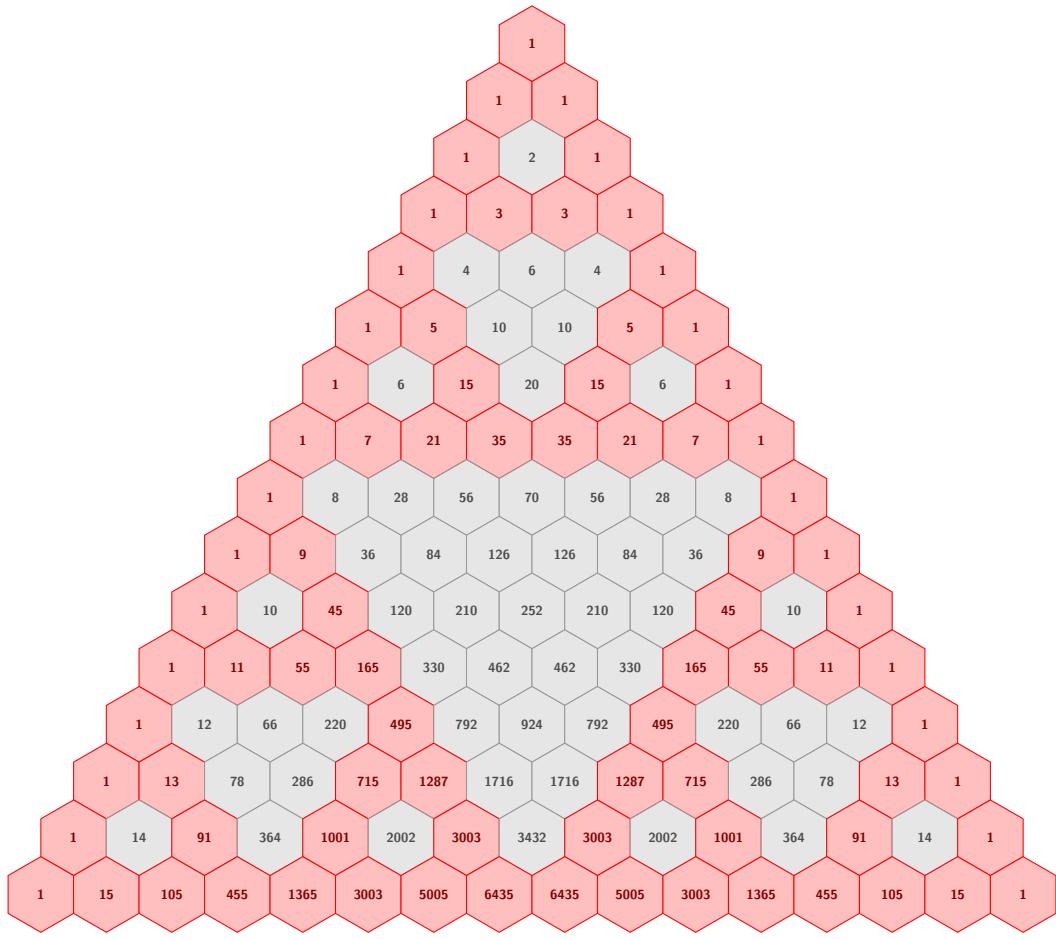


Figura 3.8: Codificació de color. Codificació per colors d'entrades paris (grises) i imparells (vermelles) en el triangle de Pascal amb 16 files.

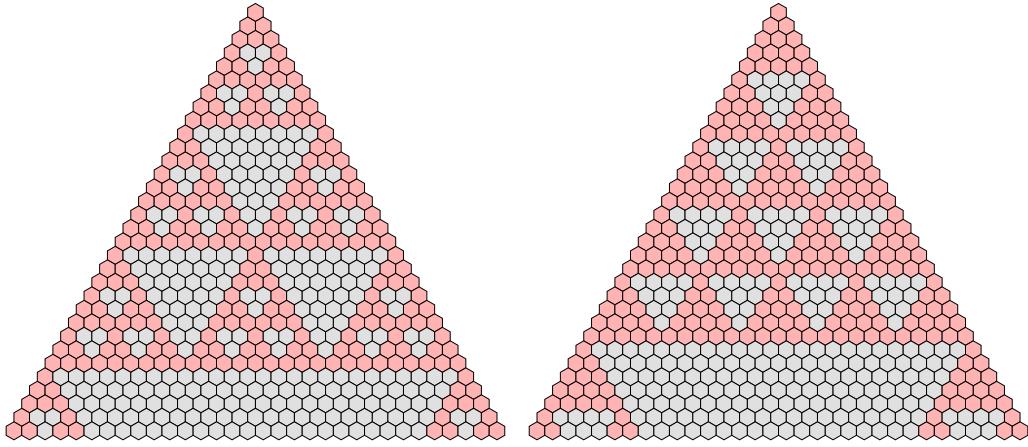


Figura 3.9: Codi de colors del triangle de Pascal. Les cel·les grises indiquen la divisibilitat entre 3 (a l'esquerra) i 5 (a la dreta).

3.3.3 Catifa

La catifa és el següent fractal que va crear Sierpiński, l'any 1916 [5, p. 79]. Aquesta figura es construeix a partir d'un quadrat en el pla, el qual subdividim en 9 quadrats més petits de costat $\frac{1}{3}$ del costat original, eliminant també el del centre. Després, aquest procés es repeteix amb cadascun dels quadrats successivament. Per cada iteració, el nombre de quadrats es multiplica per 8 i cada quadrat sempre tindrà de mida $\frac{1}{3}$ del quadrat de la iteració anterior.

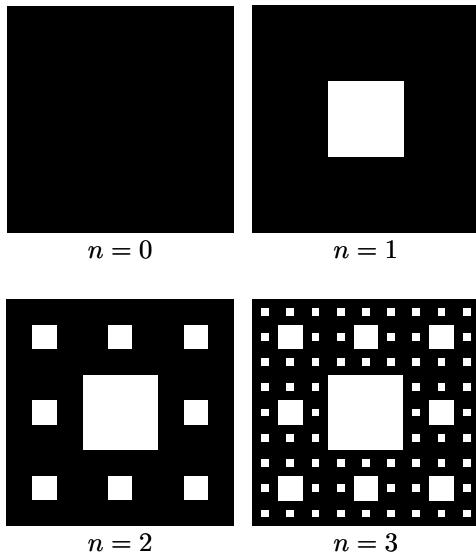


Figura 3.10: Catifa de Sierpiński. Els passos bàsics de construcció de la catifa de Sierpiński.

3.3.4 Tetraedre

Fins ara hem vist fractals que es representen al pla (d'entre 1 a 2 dimensions). Ara bé, no tots els fractals són "plans", existeixen diversos fractals amb dimensió irracional major a 2. Amb aquesta premissa apareix el tetraedre o piràmide de Sierpiński [3, p. 129]. Aquesta és una representació tridimensional del triangle de Sierpiński (veure Figure 3.11). El seu procés de construcció es realitza partint d'un tetraedre. Sobre cadascuna de les seves cares, marquem els punts mitjans de les arestes. En unir-les s'elimina la part central. D'aquesta manera obtenim un objecte format amb quatre tetraedres que seria la 1a iteració. Per a obtenir les següents iteracions es repeteix el mateix procés en cadascun dels quatre tetraedres obtenint un nou objecte.

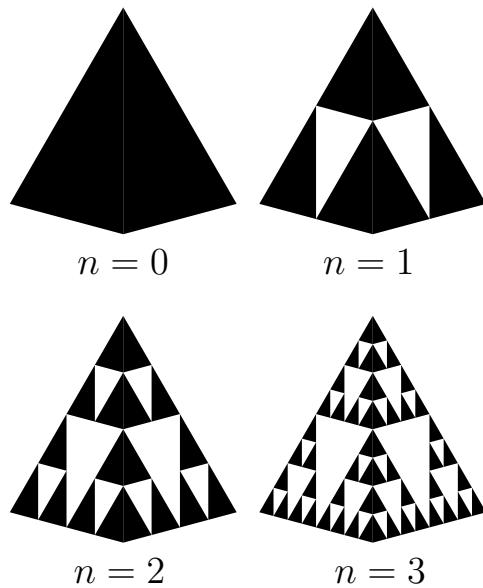


Figura 3.11: Tetraedre de Sierpiński.

3.3.5 Esponja de Menger

Un altre fractal amb dimensió superior a 2 és l'anomenada esponja de Menger [5, p. 106]. L'esponja de Menger és un fractal quasi tridimensional, batejat així pel matemàtic austríac Karl Menger, que el va descriure l'any 1926.

La seva construcció és semblant a la de la catifa de Sierpiński, però, amb un cos tridimensional. Es comença amb un cub sòlid, el qual és perforat repetidament en forats quadrats cada cop més petits. Amb cada iteració els forats tenen una mesura de $\frac{1}{3}$ dels forats de la iteració prèvia.

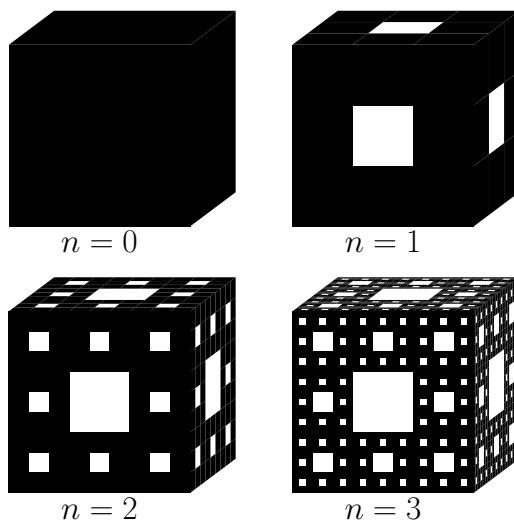


Figura 3.12: Construcció de l'esponja de Menger.

3.4 Corba de drac

La corba de drac [17] es una corba fractal que va ser obtinguda mitjançant un procés iteratiu. Va ser observada per primera vegada a la NASA pels físics Juan Heighway y Guillermo Harter, es per això, que també la corba es anomenada com el Drac d'Heighway-Harter.

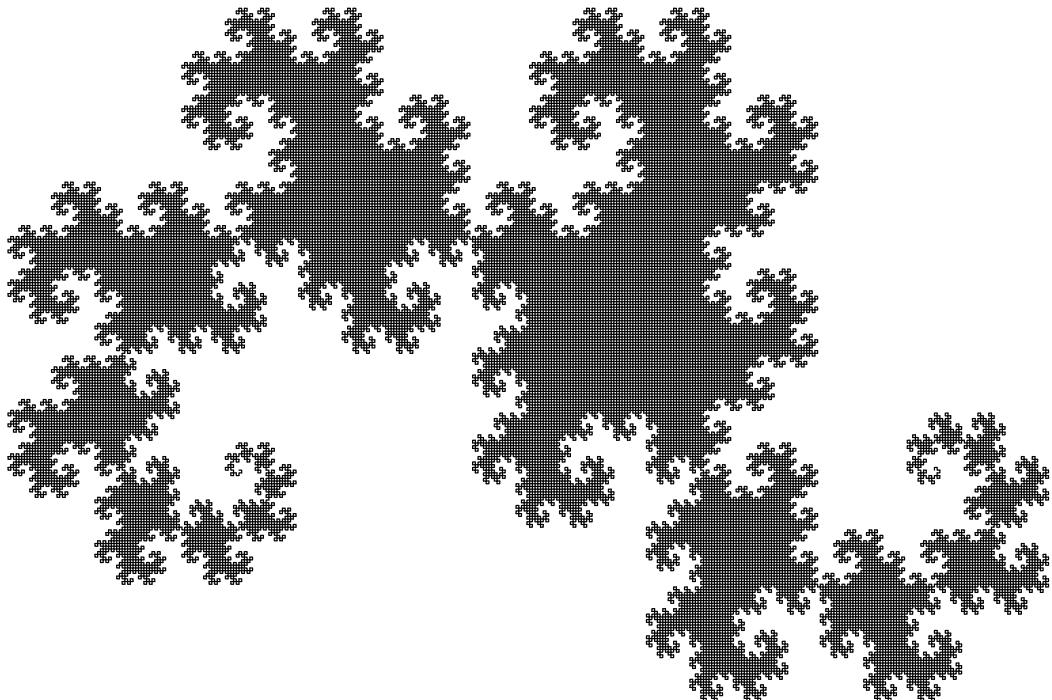


Figura 3.13: La corba de Drac.

Entre les seves propietats tenim les següents:

- La corba de drac té diverses variants, que es poden generar al canviar l'angle de gir, també si es generen més d'una corba de drac (Tetradrac y el drac bessó).

3.4.1 Construcció de la corba drac

Per a la seva construcció utilitzarem un triangle isòsceles sense base com iniciador. Després, generem un triangle orientat cap a dalt, en el costat esquerre del triangle, i un altre triangle orientat cap avall al costat dret, obtenint el generador. Aquest procés es repeteix

seqüencialment en tots els triangles isòsceles sense base de la figura anterior.

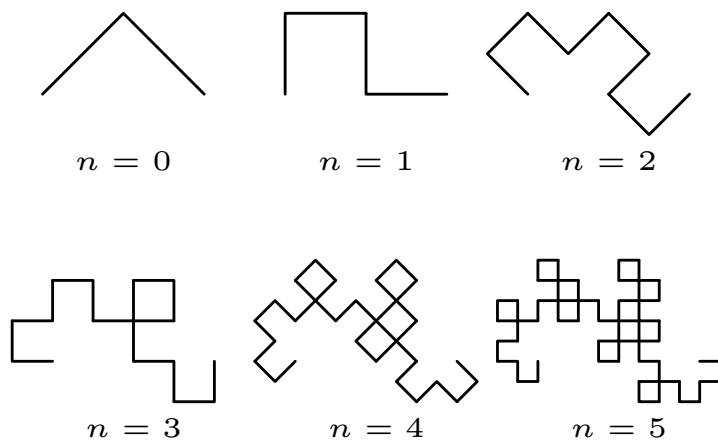


Figura 3.14: Construcció de la corba de Drac.

3.5 Corbes que omplen l'espai

L'any 1890, el matemàtic Giuseppe Peano [6], publica l'article "Sur une courbe qui remplit toute une aire plane", conegut en català com a "corbes que omplen l'espai". En aquest article, Peano explicava el descobriment d'una corba contínua, la qual tenia la peculiaritat d'"omplir" tot el pla, aquesta corba, seria batejada com corba de Peano l'any 1891. Un any després David Hilbert va fer una "variant" d'aquesta corba que seria coneguda com la corba de Hilbert.

Aquestes corbes que omplen l'espai, es construeixen mitjançant successions de corbes contínues sense interseccions, convergint en sola corba, que també s'anomena corba límit, la qual, passa per qualsevol punt de la superfície. És a dir, aquestes corbes són infinites i continues.

3.5.1 Construcció d'una corba de Peano

La construcció de la corba de Peano [5, p. 92] comença amb un segment de línia recta amb longitud 1, que serà l'iniciador (k). En aquest recte es tracen nous segments fins que queden 9 segments amb mesura $l = \frac{k}{3}$, com s'indica a la Figura 3.15 (generador). Aquest

procediment se'n va repetint en aquests 9 segments i així indefinidament en cada generador que es va obtenint com es mostra a la Figura 3.15.

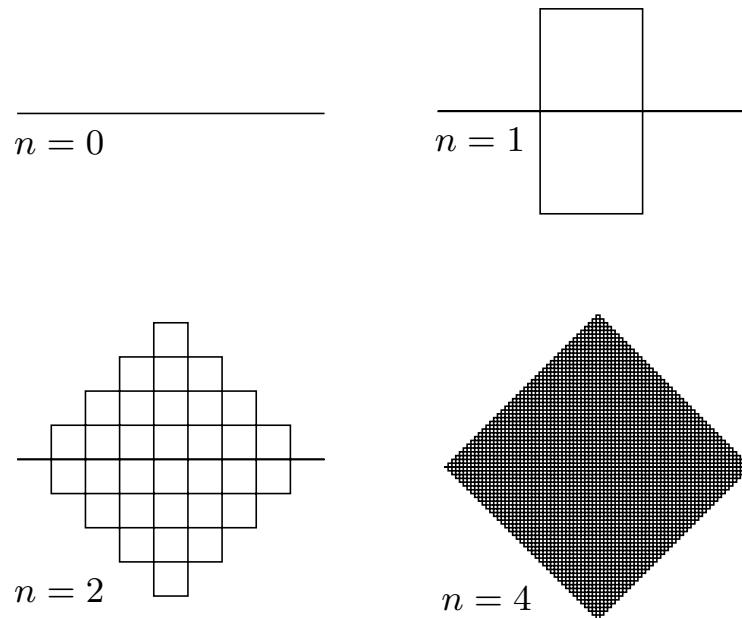


Figura 3.15: Construcció de la corba de Peano en 4 pasos. L'iniciador ($n = 0$) i tres generadors o iteracions ($n = 1, 2, 4$).

3.5.2 Altres corbes de Peano/corbes que omplen l'espai

Nom	Representació
Corba de Hilbert	
Corba de Moore	

Taula 3.2: Altres corbes que omplen l'espai o de Peano. Ambdues tenen dimensió fractal 2.

3.6 Conjunt de Julia

Els conjunts de Julia [12], batejats així en honor al matemàtic Gaston Julia, són una agrupació de fractals que s'obtenen en estudiar el comportament dels nombres complexos en ser iterats per una funció. La característica d'aquests conjunts es troba en el fet que la seva funció f només es troba constituïda per aquelles iteracions del nombre complex on els punts de la funció tenen una distribució "caòtica".

Un dels conjunts de Julia més coneguts, és aquell que s'obté de la funció quadràtica següent:

$$f_c(z) = z^2 + c. \quad (3.2)$$

On utilitzem un nombre c concret, el qual utilitzem per construir una successió de nombres, és a dir, fem iteracions d'aquesta funció:

$$\begin{aligned} z_0 &= z \\ z_1 &= f_c(z_0) = z_0^2 + c \\ z_2 &= f_c(z_1) = z_1^2 + c \\ &\vdots \\ z_n &= f_c(z_{n-1}) = z_{n-1}^2 + c \end{aligned}$$

El cas especial $c = 0$. Un cas especial es per a $c = 0$, on $f_0(z) = z^2$ és la funció més bàsica i el punt de sortida a un increïble catàleg de bells conjunts fractals de Julia.

z	òrbita 1		òrbita 2		òrbita 3	
	successió	mòdul	successió	mòdul	successió	mòdul
z_0	$0.8863+0.1563i$	0.9000	$0.9781+0.2079i$	1.0	$0.931+0.5375i$	1.0750
z_1	$0.7612+0.277i$	0.8100	$0.9135+0.4067i$	1.0	$0.5778+1.0008i$	1.1556
z_2	$0.5026+0.4217i$	0.6561	$0.6691+0.7431i$	1.0	$-0.6677+1.1566i$	1.3355
z_3	$0.0747+0.4239i$	0.4305	$-0.1045+0.9945i$	1.0	$-0.8917-1.5445i$	1.7835
z_4	$-0.1741+0.0634i$	0.1853	$-0.9781-0.2079i$	1.0	$-1.5904+2.7546i$	3.1808
z_5	$0.0263-0.0221i$	0.0343	$0.9135+0.4067i$	1.0	$-5.0587-8.762i$	10.1174
z_6	$0.0002-0.0012i$	0.0012	$0.6691+0.7431i$	1.0	$-51.1813+88.6487i$	102.3627

Taula 3.3: Dinàmica de $z \rightarrow z^2$. La iteració de tres punts inicials usant l'operació d'elevació al quadrat simple $z \rightarrow z^2$.

La Taula 3.3 mostra el comportament dinàmic de tres nombres complexos. En tots els casos observem que les successions de les òrbites giren al voltant de l'origen. No obstant això, el punt inicial de l'òrbita 1 que es troba dins del cercle unitari produeix una successió que convergeix cap a l'origen, el punt inicial de l'òrbita 2 que es troba exactament en el cercle unitari produeix una successió que roman allí per sempre, i el punt inicial de l'òrbita 3 exterior al cercle unitari produeix una successió que escapa a l'infinít, és a dir, el valor absolut de les iteracions augmenta cada vegada més (veure Figura 3.16).

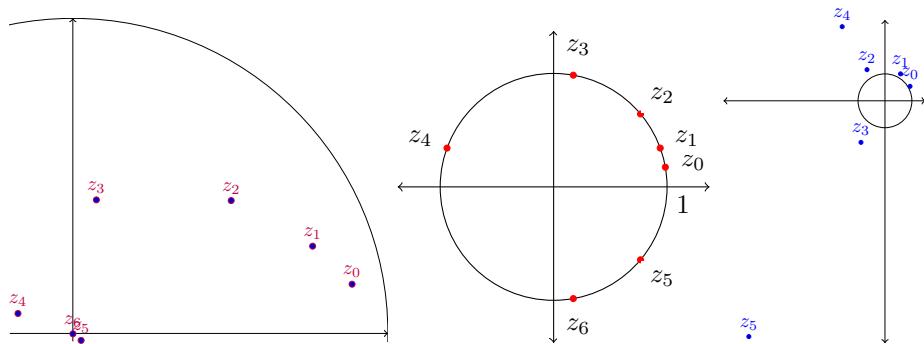


Figura 3.16: Iterant $z \rightarrow z^2$. S'iteren els tres punts inicials de la Taula 3.3.

Conjunt de Prisoners i Fuita. Això ens porta a una important dicotomia dinàmica: el pla complex de valors inicials z_0 se subdivideix en dos subconjunts.

- **El conjunt de fuita E .** El qual recull els punts z_0 per als quals la seva iteració s'escapa a l'infinít.
- **El conjunt de prisoners P .** Recull els punts z_0 per als quals tots els valors de la seva iteració romanen en una regió delimitada.

Observi que P és el disc al voltant de 0 amb radi 1 i que E és l'exterior d'aquest disc. La frontera entre E i P és el cercle unitari. Per tant, a aquesta frontera se'l denomina el **conjunt de Julia** i al conjunt de prisoners P el **conjunt ple de Julia**.

Conjunts de Julia per a la família quadràtica $f_c(z) = z^2 + c$. El cas especial $z \rightarrow z^2$ és la porta d'entrada a un increïble zoòlögic de bells conjunts fractals de Julia. Un departament d'aquest zoòlögic de conjunts de Julia es construeix sobre la iteració de $f_c(z) = z^2 + c$, on c és un paràmetre complex. El conjunt de Julia de $z \rightarrow z^2$ està just en el centre d'aquesta classe; $c = 0$. Com el cercle unitari és la frontera del conjunt de fuita per a $f_0(z) = z^2$, els altres conjunts de Julia són les fronteres del conjunt de fuita de $z \rightarrow z^2 + c$. Per tant, la

frontera comuna entre E_c i P_c és el conjunt de Julia.

Criteri de fuita. Que tan gran ha de ser una iteració perquè puguem decidir que l'òrbita definitivament escaparà a l'infinít? Es pot demostrar [5, p. 738] que si triem un z_0 arbitrari i si en la k -ésima iteració $|z_k| > 2$, llavors z_0 no pertany al conjunt de Julia. Per tant, basta trobar un sol terme de la successió que verifiqui $|z_k| > 2$ per a tenir la certesa que z_0 no pertany al conjunt.

z	òrbita 1	òrbita 2	òrbita 3
z_0	$1.0000+0.0000i$	$0.5000+0.2500i$	$0.0000+0.8800i$
z_1	$0.5000+0.5000i$	$-0.3125+0.7500i$	$-1.2744+0.5000i$
z_2	$-0.5000+1.0000i$	$-0.9648+0.0313i$	$0.8741-0.7744i$
z_3	$-1.2500-0.5000i$	$0.4210+0.4397i$	$-0.3357-0.8538i$
z_4	$0.8125+1.7500i$	$-0.5085+0.8781i$	$-1.1163+1.0732i$
z_5	$-2.9023+3.3438i$	$-1.0125-0.3930i$	$-0.4056-1.8960i$
z_6	$-3.2571-18.9094i$	$0.3707+1.2958i$	$-3.9302+2.0377i$
z_7	$-347.4578+123.6784i$	$-2.0416+1.4607i$	$10.7942-15.5173i$
z_8		$1.5346-5.4646i$	$-124.7697-334.4940i$
z_9		$-28.0066-16.2717i$	

Taula 3.4: Tres punts de fugida. La iteració de tres punts inicials per a $z \rightarrow z^2 + c$, $c = -0.5 + 0.5i$. Les tres òrbites escapan a l'infinít.

Un primer conjunt de Julia. Per a un primer exemple, triem $c = -0.5 + 0.5i$ i uns pocs punts inicials (veure Taules 3.4 i 3.5). Podem observar dos comportaments bàsics. En la primera taula, els punts iterats escapan a l'infinít. En la segona taula, els punts iterats no s'escapen, sinó que acaben convergint a un punt determinat, a saber, $z \approx -0.409 + 0.275i$. Això indica que tenim de nou dues conques d'atracció, però el zero ja no és un dels punts d'atracció.

z	òrbita 1	òrbita 2	òrbita 3
z_0	-0.500+0.500 <i>i</i>	-0.312+0.250 <i>i</i>	-0.688+0.250 <i>i</i>
z_1	-0.500+0.000 <i>i</i>	-0.465+0.344 <i>i</i>	-0.089+0.156 <i>i</i>
z_2	-0.250+0.500 <i>i</i>	-0.402+0.180 <i>i</i>	-0.516+0.472 <i>i</i>
z_3	-0.688+0.250 <i>i</i>	-0.371+0.355 <i>i</i>	-0.456+0.012 <i>i</i>
z_4	-0.090+0.156 <i>i</i>	-0.489+0.236 <i>i</i>	-0.292+0.489 <i>i</i>
z_{100}	-0.361+0.225 <i>i</i>	-0.430+0.272 <i>i</i>	-0.355+0.310 <i>i</i>
z_{200}	-0.423+0.281 <i>i</i>	-0.405+0.278 <i>i</i>	-0.417+0.263 <i>i</i>
z_{300}	-0.405+0.275 <i>i</i>	-0.409+0.274 <i>i</i>	-0.408+0.278 <i>i</i>
z_{400}	-0.409+0.275 <i>i</i>	-0.409+0.275 <i>i</i>	-0.408+0.274 <i>i</i>
z_{500}	-0.409+0.275 <i>i</i>	-0.409+0.275 <i>i</i>	-0.409+0.275 <i>i</i>

Taula 3.5: Tres punts de prisoners. La iteració de tres punts inicials per a $z \rightarrow z^2 + c$, $c = -0.5 + 0.5i$. Les tres òrbites no escapen a l'infinit. Més aviat semblen convergir a un punt $z \approx 0.41 + 0.28i$.

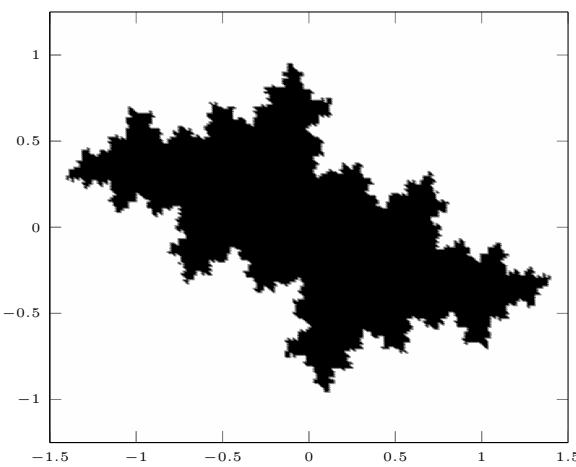


Figura 3.17: Conjunt de prisoners per a $c = -0.5 + 0.5i$. El conjunt de prisoners per a $z \rightarrow z^2 + c$, $c = -0.5 + 0.5i$ es mostra en negre. Els punts extiors escapen a l'infinit.

3.7 Conjunt de Mandelbrot

El conjunt de Mandelbrot és sens dubte el fractal més popular, probablement l'objecte més popular de les matemàtiques contemporànies. Algunes persones afirmen que no sols és l'objecte més bell sinó també el més complex que s'ha vist, és a dir, que s'ha fet visible. Des que Mandelbrot va realitzar el seu extraordinari experiment en 1979, ha estat duplicat per desenes de milers de científics aficionats de tot el món. A tots ells els agrada endinsar-se en la il·limitada varietat d'imatges que poden desenvolupar-se en la pantalla d'un ordinador. A vegades es necessiten moltes hores per a generar-les; però aquest és el preu que cal pagar per l'aventura de trobar una cosa nova i fantàstic on ningú ha mirat abans.

Aquesta riquesa és només un regal generós de les matemàtiques als qui els agrada meravellar-

se amb les boniques imatges, o aquesta aparent bellesa i complexitat tenen un significat més profund? Les aparents característiques pictòriques del conjunt de Mandelbrot tenen una contrapartida igual en la seva bellesa matemàtica? En altres paraules, el conjunt de Mandelbrot presenta una visió del que els matemàtics de vegades anomenen l'estètica de les matemàtiques? La resposta és un "sí" vigorós.

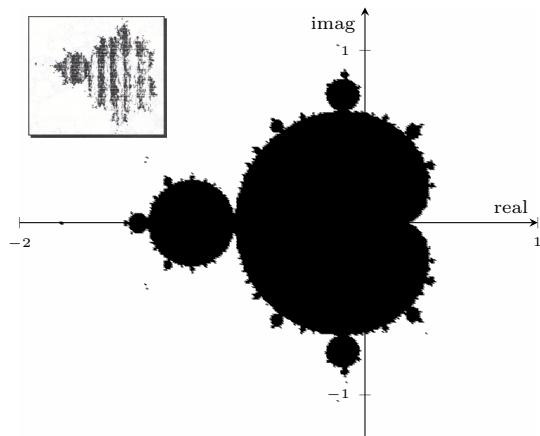


Figura 3.18: El conjunto de Mandelbrot - vella i nova representació. La inserció mostra una impressió original de l'experiment de Mandelbrot.

El conjunt de Mandelbrot s'obté de la següent funció:

$$f(z) = z^2 + c, \quad z, c \in \mathbb{C}. \quad (3.3)$$

Primer es fixa un nombre complex c qualsevol. A partir de c , es construeix una successió recursiva:

$$\begin{cases} z_0 = 0, & \text{termini inicial} \\ z_{n+1} = z_n^2 + c, & \text{successió recursiva.} \end{cases}$$

En general, els punts c pertanyen al subconjunt del pla complex $[-2, 2] \times [-2, 2]$.

3.7.1 Observem dos exemples

Primer implementem en Python una funció denominada **successio_c**, la qual crearà una successió d'un nombre complex arbitrari c .

```
# c és nombre complex arbitrari
# max_iter és el nombre màxim d'iteracions
def successio_c(c, max_iter):
    z=0
    dict_successio={ 'c[k]' :[] , 'successio_c[k]' :[] , 'mòdul_c[k]' :[] }
```

```

for i in range(0,max_iter):
    z=z**2+c
    dict_successio[ 'c[k]' ].append(f'c{[k]}')
    dict_successio[ 'successió_c[k]' ].append(z)
    dict_successio[ 'mòdul_c[k]' ].append(abs(z))
df=pd.DataFrame(dict_successio)
return df

```

Exemple 3.1. Prendre $c = -0.5 + 0.5i$

```

c1=complex(-0.5,0.5)
taula_01=successio_c(c1,20)
taula_01

```

Aquesta funció retorna un conjunt de dades amb les següents columnes:

- $c[k]$: índex de cadascun dels elements de la successió
- $\text{successió}_c[k]$: elements de la successió del nombre complex c
- $\text{mòdul}_c[k]$: mòdul de cadascun dels elements de la successió

$c[k]$	$\text{successió}_c[k]$	$\text{mòdul}_c[k]$
$c[0]$	$-0.5000+0.5000i$	0.7071
$c[1]$	$-0.5000+0.0000i$	0.5000
$c[2]$	$-0.2500+0.5000i$	0.5590
$c[3]$	$-0.6875+0.2500i$	0.7315
$c[4]$	$-0.0898+0.1562i$	0.1802
$c[5]$	$-0.5163+0.4719i$	0.6995
$c[6]$	$-0.4561+0.0127i$	0.4563
$c[7]$	$-0.2921+0.4885i$	0.5692
$c[8]$	$-0.6533+0.2146i$	0.6876
$c[9]$	$-0.1193+0.2196i$	0.2499
$c[10]$	$-0.5340+0.4476i$	0.6968
$c[11]$	$-0.4152+0.0220i$	0.4158
$c[12]$	$-0.3281+0.4817i$	0.5829
$c[13]$	$-0.6244+0.1839i$	0.6509
$c[14]$	$-0.1439+0.2704i$	0.3063
$c[15]$	$-0.5524+0.4222i$	0.6953
$c[16]$	$-0.3731+0.0336i$	0.3746
$c[17]$	$-0.3619+0.4749i$	0.5971
$c[18]$	$-0.5946+0.1562i$	0.6148
$c[19]$	$-0.1709+0.3142i$	0.3577

Taula 3.6: Taula amb la successió del nombre complex $-0.5+0.5i$ i els seus respectius mòduls. Es pot veure que en 19 iteracions tots aquests mòduls són menors que dos.

En la Taula 3.6 es pot veure que en 19 iteracions el mòdul de tots els nombres $c[k]$, $k = 0, \dots, 19$ és menor que dos. És a dir, $|c[k]| < 2$ per a tot $k = 0, \dots, 19$.

Exemple 3.2. Prendre $c = 0.5 + 0.5i$

```
c2=complex(0.5,0.5)
taula_02=successio_c(c2,9)
taula_02
```

c[k]	successió_c[k]	mòdul_c[k]
c[0]	0.50+0.50i	0.71
c[1]	0.50+1.00i	1.12
c[2]	-0.25+1.50i	1.52
c[3]	-1.69-0.25i	1.71
c[4]	3.29+1.34i	3.55
c[5]	9.49+9.33i	13.30
c[6]	3.47+177.50i	177.53
c[7]	-31493.03+1231.54i	31517.10
c[8]	990294278.68-77569977.40i	993327669.89

Taula 3.7: Taula amb la successió del nombre complex $0.5+0.5i$ i els seus respectius mòduls. Es pot ser que a partir de la iteració 4 els mòduls són majors que dos.

Per contra, en Exemple 3.2 es pot veure que en 8 iteracions a partir del nombre $c[4]$, el mòdul és major que dos. És a dir, $|c[k]| > 2$, per a $k = 4, \dots, 8$.

Mandelbrot va establir que tots els punts c que compleixin amb la condició de l'Exemple 3.1 per a un cert nombre màxim d'iteracions (per a aquest estudi s'ha establert en 9, però pot ser qualsevol, per exemple: 15, 20, 50, etc.) es pinten de color negre. Cas contrari, punts com l'Exemple 3.2, es pinten de color blanc (veure Figura 3.19).

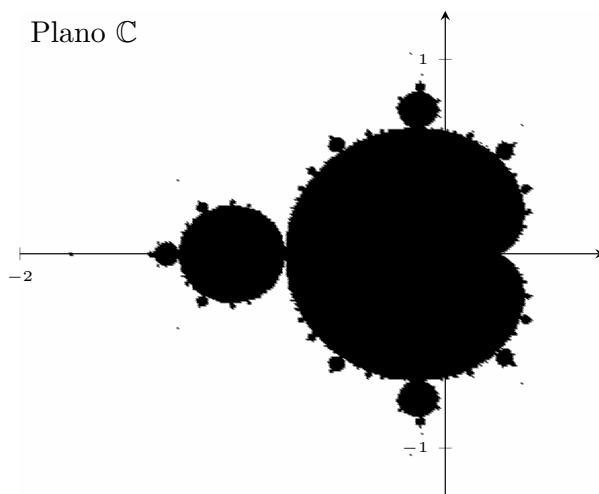


Figura 3.19: Conjunt de Mandelbrot.

3.7.2 Matriu d'iteracions

El nostre objectiu és crear una matriu que mostri el nombre d'iteracions d'un conjunt de nombres complexos c que s'obtenen de la funció $f(z) = z^2 + c$, amb $z = 0$.

Obtenir la iteració k . Crear una funció que retorni la iteració k on el mòdul del nombre complex c és major que dos.

```
# c és nombre complex arbitrari
# max_iter és el nombre màxim d'iteracions
def obtenir_iteracio(c, maxiter):
    z=complex(0,0)
    for k in range(maxiter):
        z=z*z+c
        if abs(z)>2:
            break
        pass
    pass
    return k
```

Taula amb diferents nombres complexos c . Farem una reixeta amb 81 nombres complexos en l'interval 2D $[-2, 2] \times [-2, 2]$ amb un espaiat de 0.5.

```
x_values=np.linspace(-2,2,9)
y_values=np.linspace(-2,2,9)
n=len(x_values)
m=len(y_values)
a=[]
for i in range(n):
    row=[]
    for j in range(m):
        c=complex(x_values[i].round(2),y_values[j].round(2))
        row.append(f'{c.real}+{c.imag}i')
    a.append(row)

data_complex=pd.DataFrame(a)
data_complex
```

	0	1	2	3	4	5	6	7	8
0	-2.0-2.0i	-2.0-1.5i	-2.0-1.0i	-2.0-0.5i	-2.0+0.0i	-2.0+0.5i	-2.0+1.0i	-2.0+1.5i	-2.0+2.0i
1	-1.5-2.0i	-1.5-1.5i	-1.5-1.0i	-1.5-0.5i	-1.5+0.0i	-1.5+0.5i	-1.5+1.0i	-1.5+1.5i	-1.5+2.0i
2	-1.0-2.0i	-1.0-1.5i	-1.0-1.0i	-1.0-0.5i	-1.0+0.0i	-1.0+0.5i	-1.0+1.0i	-1.0+1.5i	-1.0+2.0i
3	-0.5-2.0i	-0.5-1.5i	-0.5-1.0i	-0.5-0.5i	-0.5+0.0i	-0.5+0.5i	-0.5+1.0i	-0.5+1.5i	-0.5+2.0i
4	0.0-2.0i	0.0-1.5i	0.0-1.0i	0.0-0.5i	0.0+0.0i	0.0+0.5i	0.0+1.0i	0.0+1.5i	0.0+2.0i
5	0.5-2.0i	0.5-1.5i	0.5-1.0i	0.5-0.5i	0.5+0.0i	0.5+0.5i	0.5+1.0i	0.5+1.5i	0.5+2.0i
6	1.0-2.0i	1.0-1.5i	1.0-1.0i	1.0-0.5i	1.0+0.0i	1.0+0.5i	1.0+1.0i	1.0+1.5i	1.0+2.0i
7	1.5-2.0i	1.5-1.5i	1.5-1.0i	1.5-0.5i	1.5+0.0i	1.5+0.5i	1.5+1.0i	1.5+1.5i	1.5+2.0i
8	2.0-2.0i	2.0-1.5i	2.0-1.0i	2.0-0.5i	2.0+0.0i	2.0+0.5i	2.0+1.0i	2.0+1.5i	2.0+2.0i

Taula 3.8: Taula amb els 81 nombres complexos

En la fila 3 i columna 5 de la Taula 3.8 es pot veure el nombre complex $-0.5 + 0.5i$ (veure Exemple 3.1). Per a aquest número en 19 iteracions el mòdul de tots els seus elements eren menors que 2. Per tant, assignem amb un 19 a aquesta cel·la. D'altra banda, en la fila 5 i columna 5 es pot veure el nombre complex $0.5 + 0.5i$ (veure Exemple 3.2), per a aquest nombre a partir de la iteració 4 el mòdul de tots els seus elements eren majors que 2. Per tant, assignem amb un 4 a aquesta cel·la.

Visualitzar la matriu d'iteracions. Imprimir una matriu amb el nombre d'iteracions que compleix amb la condició mòdul major que 2 per a cadascun dels 81 nombres complexos.

```
matriu_iteracions=np.zeros((n,m))
for i in range(n):
    for j in range(m):
        c=complex(x_values[i],y_values[j])
        matriu_iteracions[i,j]=obtenir_iteracio(c,20)

data_iterations=pd.DataFrame(matriu_iteracions)
data_iterations
```

	0	1	2	3	4	5	6	7	8
0	0	0	0	0	19	0	0	0	0
1	0	0	1	2	19	2	1	0	0
2	0	1	2	4	19	4	2	1	0
3	0	1	3	19	19	19	3	1	0
4	1	1	19	19	19	19	19	1	1
5	0	1	1	4	4	4	1	1	0
6	0	1	1	1	2	1	1	1	0
7	0	0	1	1	1	1	1	0	0
8	0	0	0	0	1	0	0	0	0

Taula 3.9: Matriu d'iterations

Dibuixar la matriu d'iterations.

```
plt.figure(figsize=(8, 8))
plt.imshow(matriu_iteracions.T, interpolation="nearest")
plt.show()
```

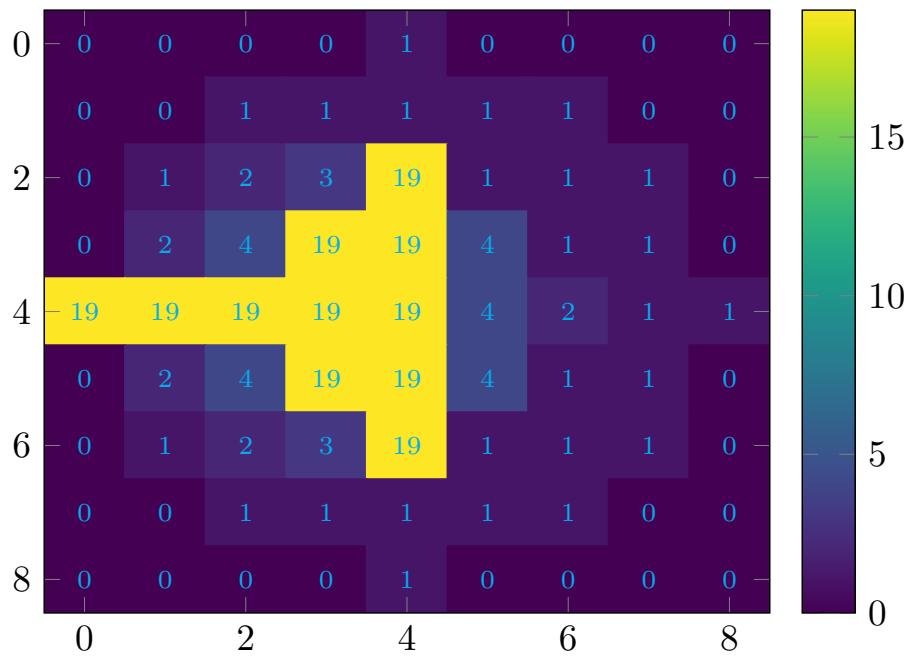


Figura 3.20: Gràfic de la matriu d'iteracions. La barra de colors indica el nombre d'iteracions d'un nombre complex fins que es dispara de 0 (morat) a 19 (groc). A la figura s'indica el nombre d'iteracions que té cada nombre complex de la Taula 3.8.

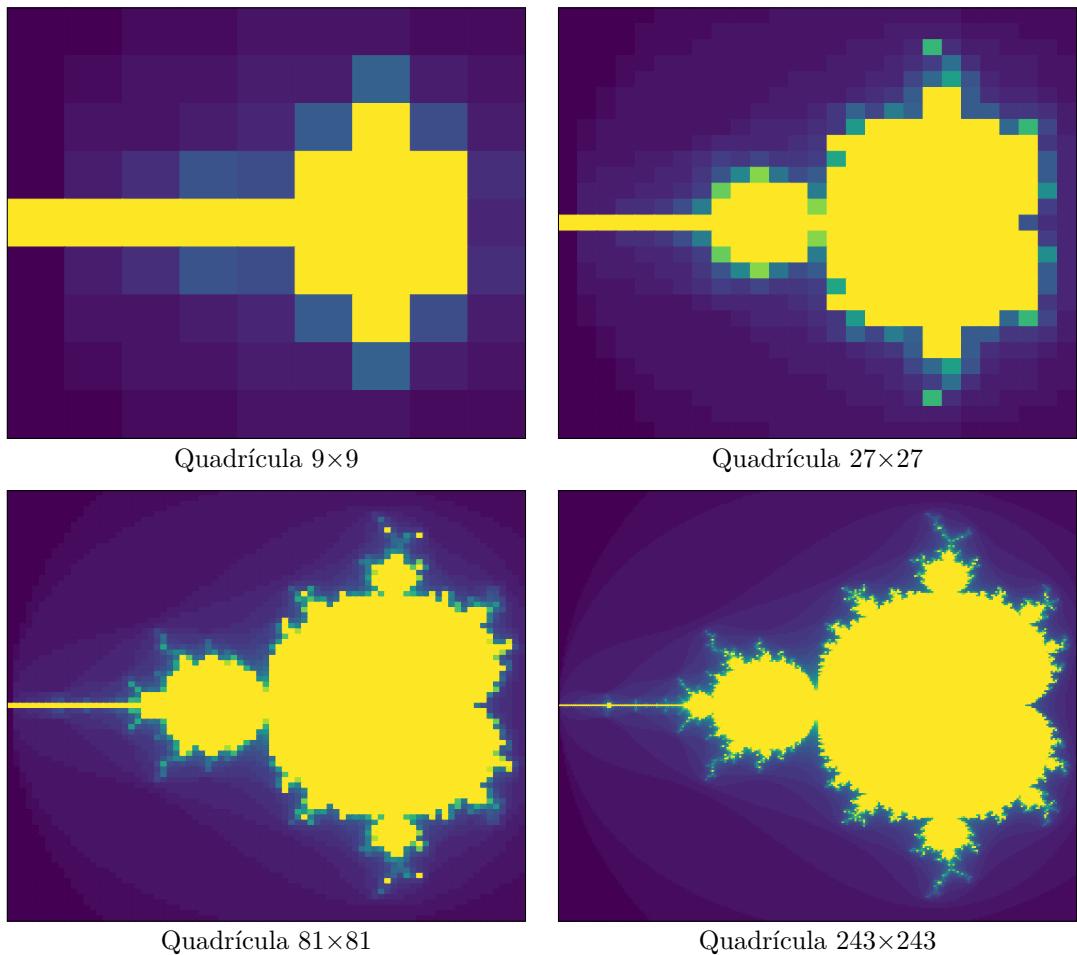


Figura 3.21: 4 conjunts de Mandelbrot. Cadascun amb un conjunt de punts major.

4. CARACTERÍSTIQUES DELS FRACTALS

4.1 Característiques generals

En ser figures tan diferents de les tradicionals formes euclidianes, els fractals presenten característiques úniques:

- Són formes recursives
- Presenten autosimilitud
- No tenen dimensió amb valor natural, sinó irracional

També, tenen la qualitat de ser figures amb superfície finita, però perímetre infinit, això, pel fet que són figures que es repeteixen a diferents escales.

4.2 Autosimilitud

El concepte d'autosimilitud o autosemblança és una propietat dels fractals el qual consisteix en el fet que tot un conjunt és igual o semblant en totes les parts de si mateix, de tal manera que el cos es pot engrandir o encongrir i encara seguirà sent igual. Un exemple de la natura d'autosimilitud són les ones del mar.

Matemàticament, l'autosimilitud és un terme amb diferents conceptes o tipus d'autosimilitud. Aquests tipus d'autosimilitud presenten diferències entre si mateixos, però, comparteixen la característica tenir semblança a escala.

4.2.1 Autosimilitud exacta

Es diu que hi ha autosimilitud exacta quan una o diverses parts d'un tot repeteixen exactament la seva similitud amb aquest tot. L'autosimilitud exacta permet l'amplificació successiva

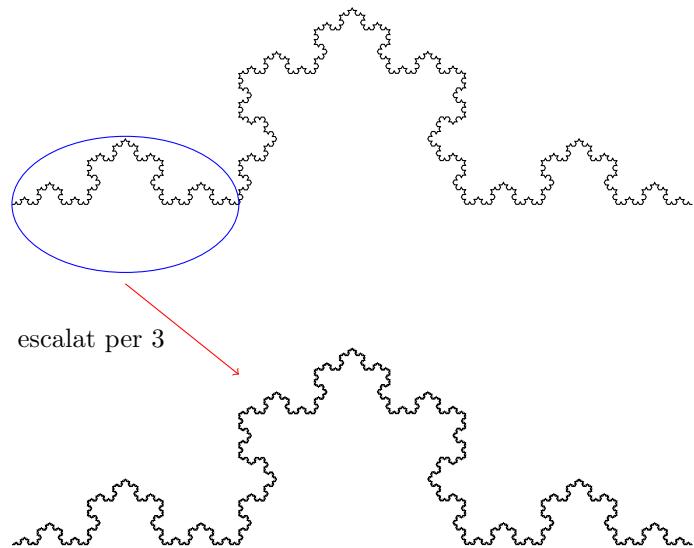


Figura 4.1: Ampliació de la corba de Koch. Un quart de la corba de Koch (a dalt) es magnifica per un factor de 3. A causa de l'autosimilitud de la corba de Koch, el resultat és una còpia de la corba completa.

amb repetició exacta única, múltiple o infinita de les propietats inicials.

4.2.2 Autosimilitud aproximada

L'autosimilitud aproximada o quasi-autosimilitud es troba sovint en la naturalesa (autosimilitud natural). Es caracteritza per no condensar la propietat d'autosimilitud en tot el conjunt, és a dir, hi existeixen petites diferències entre la part i el tot.

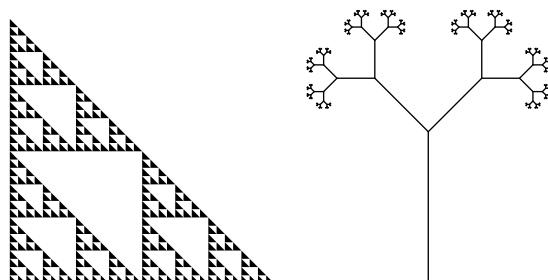


Figura 4.2: Dues diferents estructures autosemlants La junta de Sierpiński (esquerra) és autosemlant en tots els seus punts, mentre que l'arbre de dues branques (dreta) és semblant només en les fulles.

4.2.3 Autosimilitud estadística

L'autosimilitud estadística és la menys exigent. Només es conserven algunes propietats estadístiques durant el canvi d'escala, com a les muntanyes o en els cràters lunars.

4.3 Dimensions fraccionàries o fractals

Tota figura geomètrica o cos que es pugui representar al pla, per norma general, té el que anomenem dimensions. Per exemple una recta té una dimensió, un quadrat dos i un cub tres. Això és el que s'anomena dimensió topogràfica. Aquesta dimensió topogràfica, normalment, és un nombre natural (1, 2 i 3). Tot i això, aquesta norma no hi aplica pels fractals, els quals, tenen una dimensió topològica amb nombre irracional, aquesta característica va donar lloc a què molts matemàtics intentessin determinar on tenen cabuda els fractals, respecte al camp de les dimensions, és aquí on apareix el terme dimensió fractal o fraccionària. La dimensió fractal es va idear amb l'objectiu d'analitzar la dimensió irracional dels fractals en el pla, amb el temps, es van donar diverses definicions i es van elaborar diferents dimensions per aquesta, com poden ser la dimensió de Hausdorff-Besicovitch, dimensió de comptatge de caixes, dimensió d'euclidiana, dimensió de capacitat, dimensió d'informació, etc. Hi existeixen moltes, i totes estan relacionades, però no totes tenen la mateixa importància. De totes aquestes, destaquen la dimensió d'autosimilitud i la de comptatge de caixes.

4.3.1 Dimensió d'autosimilitud

Per a definir la dimensió d'autosimilitud [2, p. 194], primer analitzarem alguns cossos autosimilars triviais (una línia, un quadrat i un cub) de la següent manera:

Una línia de longitud 1 és un objecte molt autosimilar: Pot dividir-se en $n = n^1$ petites peces, cadascuna de les quals és exactament $\frac{1}{n}$ de la grandària de la línia original i cadascuna de les quals, quan s'amplia per un factor n , s'assembla exactament a la línia original (veure Figura 4.3).

Per a un quadrat de costat 1, si el dividim en quadrats de costat $\frac{1}{n}$ de l'original obtindrem

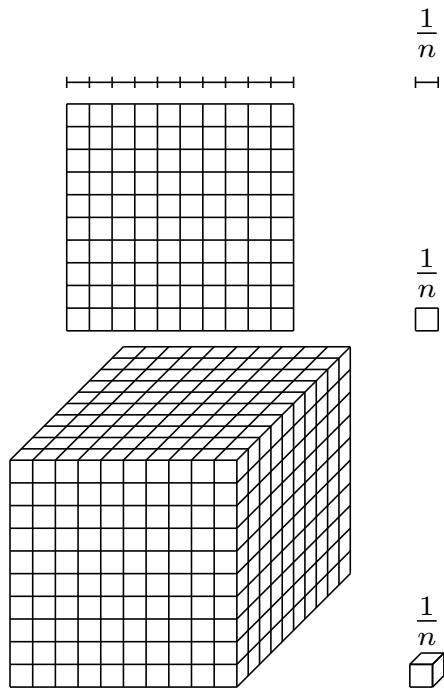


Figura 4.3: Il·lustrant el nombre de còpies petites d'un objecte que caben dins d'una còpia gran.

$n \times n$ quadrats petits. Per tant, es requereix n^2 quadrats petits per obtenir l'original.

En el cas d'un cub de costat 1, en dividir-ho per peces de costat $\frac{1}{n}$, s'obtindran n^3 peces més petites, les necessàries per reconstruir el cub original.

Coss	Factor d'augment	Nombre de copies petites que caben en una copia gran
Línia	3	$3 = 3^1$
Quadrat	3	$9 = 3^2$
Cub	3	$27 = 3^3$

Taula 4.1: Efectes de l'ampliació en diferents cossos.

En aquests 3 casos, l'exponent distingeix la dimensió de l'objecte en qüestió. En la taula 4.1 hem pres $n=3$ per a dividir la línia, el quadrat i el cub, en la tercera columna podem veure que cadascun dels exponents estan relacionats amb la dimensió dels cossos anteriors. A més, la base 3 representa el factor d'augment. Per tant, obtenim la següent igualtat.

$$\text{nombre de còpies petites} = (\text{factor d'augment})^D, \quad (4.1)$$

on D és la dimensió. Finalment, aplicant logaritme a banda i banda de la igualtat 4.1 obtenim

$$D = \frac{\log(k)}{\log(M)} = \frac{\log(\text{nombre de còpies petites})}{\log(\text{factor d'augment})}. \quad (4.2)$$

Exemple 4.1. Trobar la dimensió del triangle de Sierpiński amb el mètode d'autosimilitud.

Solució. Per al triangle de Sierpiński, recordem que podem subdividir aquesta figura en tres peces autosimilars, cadascun dels quals pot ampliar-se per un factor de 2 per a obtenir la figura completa. Així, tenim

$$D = \frac{\log(k)}{\log(M)} = \frac{\log(\text{nombre de triangles})}{\log(\text{factor d'augment})} = \frac{\log 3}{\log 2} = 1.584962501,$$

el qual no és en absolut un nombre enter! Provem de nou. El triangle de Sierpiński també es pot construir muntant nou peces més petites. Cadascuna d'aquestes peces més petites és exactament una quarta part de la mida de la figura original. Per tant,

$$D = \frac{\log 9}{\log 4} = \frac{\log 3^2}{\log 2^4} = \frac{2 \log 3}{2 \log 2} = \frac{\log 3}{\log 2} = 1.584962501,$$

i tenim la mateixa dimensió. ■

4.3.2 Dimensió de comptatge de caixes/Haussdorff

La dimensió de comptatge de caixes o de Haussdorff [3, p. 93], és una de les tantes nocions, respecte a la dimensió fractal, aquesta destaca per ser un la més eficient en aplicacions científiques. Aquesta noció de dimensió fractal empra quadràcules, que denominarem caixes, les quals són superposades a una forma fractal. En fer això, es fa un recompte del nombre de caixes ($N(r)$), de mesura r , necessàries per a cobrir una forma. Però, com depèn $N(r)$ de r ? Per a respondre aquesta pregunta, observem com es comporta aquesta relació en 1, 2 i 3 dimensions (veure Figura 4.4).

- Si la forma és unidimensional, com el segment de recta, veiem que $N(r) = \frac{1}{r}$.
- Si la forma és bidimensional, com el quadrat unitari (omplert), veiem que $N(r) = \left(\frac{1}{r}\right)^2$
- Si la forma és tridimensional, com el cub unitari (omplert), veiem que $N(r) = \left(\frac{1}{r}\right)^3$.

Per a formes més complexes, la relació entre $N(r)$ i $\frac{1}{r}$ pot no ser tan clara. Si suposem que $N(r)$ és aproximadament $k \left(\frac{1}{r}\right)^d$, una relació de llei de potència, com podem trobar d ?

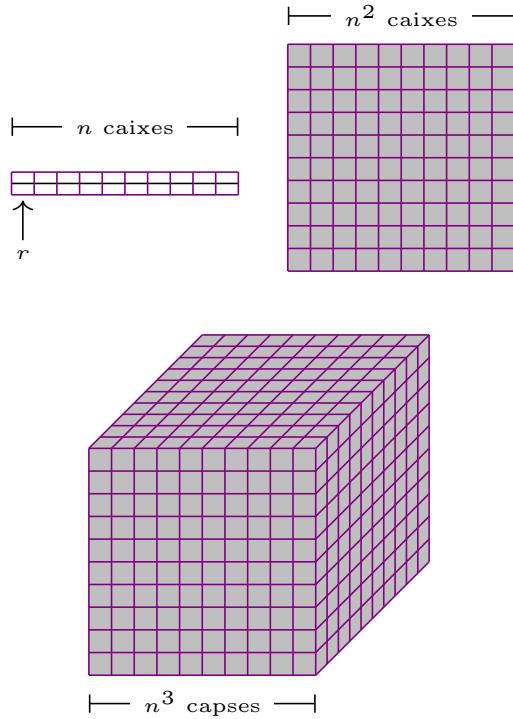


Figura 4.4: La línia, el quadrat i el cub, tots de costat 1. En tots els casos les caixes que els cobreixen tenen els seus costats com mesurada $r = \frac{1}{n}$.

Prenent el logaritme de tots dos costats de $N(r) = k \left(\frac{1}{r}\right)^d$, obtenim:

$$\begin{aligned} \log(N(r)) &= \log(k) + \log\left(\frac{1}{r}\right)^d \\ &= \log(k) + d \log\left(\frac{1}{r}\right). \end{aligned}$$

Amb l'expectativa que l'aproximació es fa millor per a r més petit. Resolent per a d i prenent el límit quan $r \rightarrow 0$, tenim:

$$d = \lim_{r \rightarrow 0} \frac{\log(N(r))}{\log(\frac{1}{r})}$$

Observi que quan $r \rightarrow 0$ tenim $\frac{1}{r} \rightarrow \infty$, llavors $\log\left(\frac{1}{r}\right) \rightarrow \infty$ i $\frac{\log(k)}{\log(\frac{1}{r})} \rightarrow 0$.

Si el límit existeix, es denomina dimensió de comptatge de caixes, d , de la forma. Aquest límit pot ser lent per a convergir; un enfocament alternatiu és observar que

$$\log(N(r)) = \log(k) + d \log\left(\frac{1}{r}\right),$$

és l'equació d'una recta amb pendent d i intercepció- y , $\log(k)$. Per tant, si representem $\log(N(r))$ enfront de $\log\left(\frac{1}{r}\right)$, els punts haurien de situar-se aproximadament en una línia recta amb pendent d . Aquest és el mètode $\log - \log$ per a trobar la dimensió de comptatge de caixes.

Exemple 4.2. *Trobar la dimensió del triangle de Sierpiński amb el mètode de comptatge per caixes.*

Solució. Per a aquest exemple utilitzarem 4 quadrículles amb caixes de costat $r_1 = \frac{1}{8}$, $r_2 = \frac{1}{16}$, $r_3 = \frac{1}{32}$ i $r_4 = \frac{1}{64}$. Després, per a cada quadrícula, comptarem el nombre de caixes que cobreixen el triangle de Sierpiński (vegeu Taula 4.2). Finalment, trobem els 4 punts $\left(\log\left(\frac{1}{r}\right), \log(N(r))\right)$ en el pla, els quals seran necessaris per a obtenir la recta que millor s'ajusta a aquests 4 punts.

Per tant, la dimensió de comptatge de caixes del triangle de Sierpiński és el pendent de la recta $y = 1.584963x$ (veure Figura 4.5), que és el mateix resultat que es va obtenir en l'Exemple 4.1.

r	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$	$\frac{1}{64}$
$N(r)$	27	81	243	729
$\log\left(\frac{1}{r}\right)$	2.0794	2.7726	3.4657	4.1589
$\log(N(r))$	3.2958	4.3944	5.4931	6.5917

Taula 4.2: 4 quadrículles del triangle de Sierpiński amb caixes de costat $r_1 = 1/8$, $r_2 = 1/16$, $r_3 = 1/32$ i $r_4 = 1/64$.

■

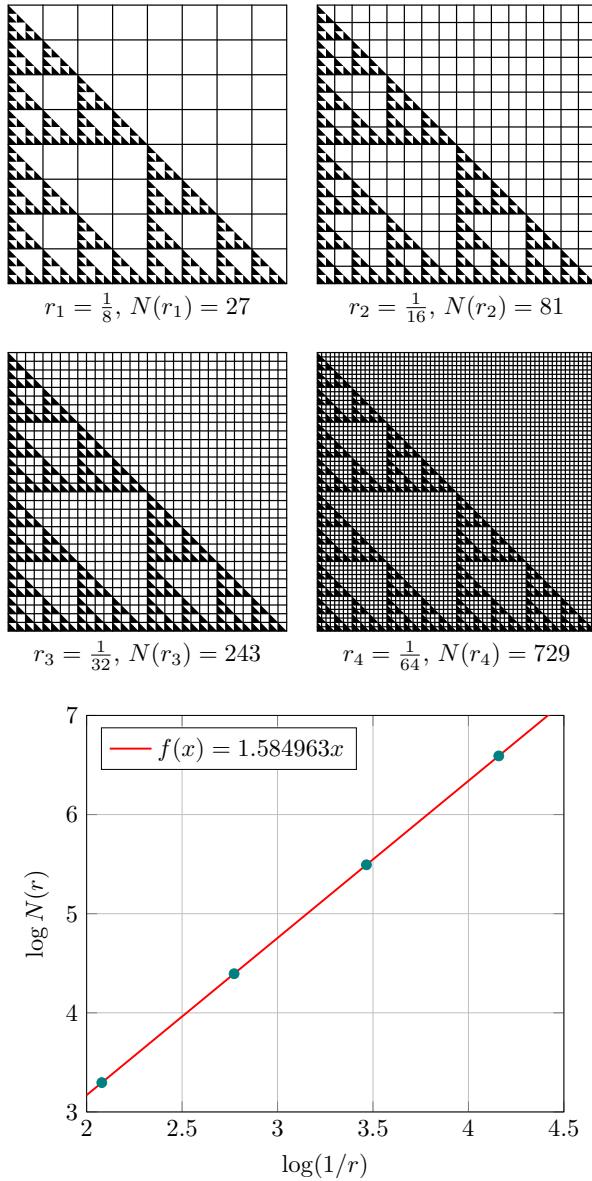


Figura 4.5: Triangle de Sierpinński. Comptatge de caixes utilitzant 4 quadràcules, el pendent de la recta que s'ajusta als 4 punts és 1.584963.

Exemple 4.3. Trobar la dimensió del contorn del mapa d'Espanya amb el mètode de comptatge de caixes.

Solució. Per a aquest exemple utilitzarem 3 quadràcules amb caixes de costat $r_1 = \frac{1}{6}$, $r_2 = \frac{1}{12}$ i $r_3 = \frac{1}{24}$. Després, per a cada quadràcula, comptarem el nombre de caixes que cobreixen el mapa d'Espanya que són $N(r_1) = 23$, $N(r_2) = 52$ i $N(r_3) = 121$ respectivament (veure Figura 4.6).

■

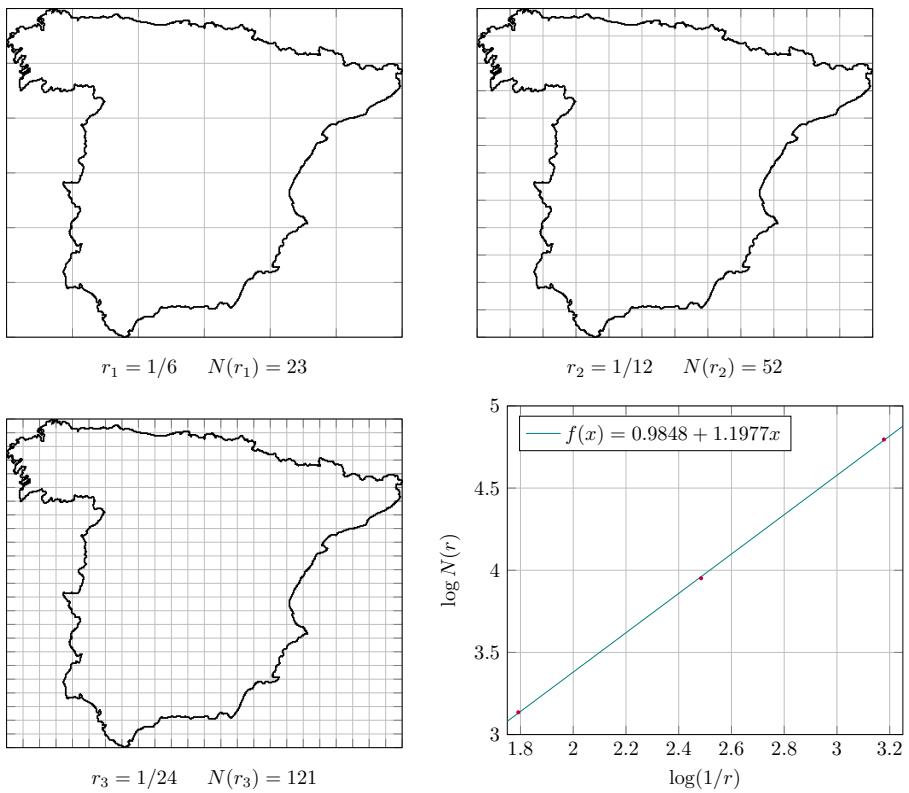


Figura 4.6: Comptatge de caixa utilitzant 3 quadràcules, el pendent de la recta que s'ajusta als 3 punts és 1.1977.

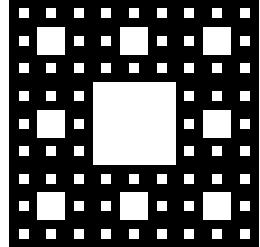
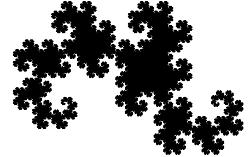
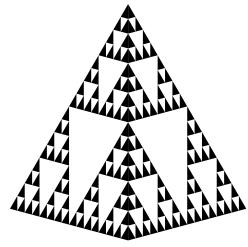
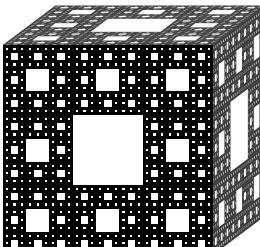
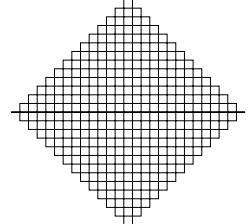
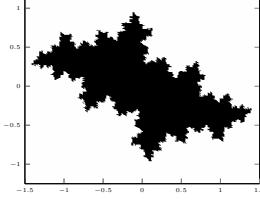
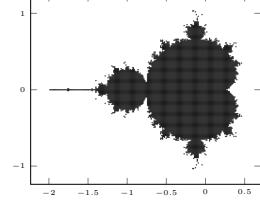
Actualment, gràcies als avanços en informàtica, podem deixar de costat aquest mètode tan rudimentari, però funcional, i usar mesures r més petites per a aproximar-nos cada vegada més al valor real de la dimensió fractal de qualsevol forma.

Taula 4.3: Dimensió dels fractals estudiats en aquest treball.

Valor real	Valor aproximat	Nom	Representació
$\frac{\log(2)}{\log(3)}$	0.6309	conjunt de Cantor	
$\frac{\log(4)}{\log(3)}$	1.2619	corba de Koch	
$\frac{\log(3)}{\log(2)}$	1.585	triangle de Sierpiński	

Continua en la pàgina següent

Taula 4.3 – Continua de la pàgina anterior

Valor real	Valor aproximat	Nom	Representació
$\frac{\log(2)}{\log(3)}$	1.8928	catifa de Sierpiński	
$\frac{\log(2)}{\log(\sqrt{2})}$	2	corba de drac	
$\frac{\log(4)}{\log(2)}$	2	tetraedre de Sierpiński	
$\frac{\log(20)}{\log(3)}$	2,7268	esponja de Menger	
2	2	corba de Peano	
2	2	conjunt de Julia	
2	2	conjunt de Mandelbrot	

4.4 Recursivitat

L'última característica important dels fractals és la seva recursivitat. La recursivitat es defineix com la propietat d'una unitat o procés de construir-se mitjançant la repetició de si mateix indefinidament.

5. FRACTALS A LA NATURA

5.1 Geometria de la natura

Com vam mencionar en apartats anteriors, les formes fractals són la geometria de la naturalesa, és a dir, la geometria fractal es pot trobar en la natura, en moltes parts i amb diverses formes.

La manera més comuna en la qual podem trobar els fractals a la natura són els paisatges i elements de la natura [4], els quals, presenten una forma que escapa de la concepció "perfecta" de la geometria euclidiana, ja que presenten autosimilitud, però, d'una manera no tan exacta, és a dir, els elements naturals amb forma fractal presenten quasi similitud.

5.1.1 Rius

Les xarxes o formacions d'aigua com rius són altre exemple de la fractalitat del món. Els fractals són emprats en l'estudi dels rius de dues maneres. El patró del rierol dels rius segueix una forma fractal, perquè un riu es divideix en corrents d'aigua més petites i així successivament. Aquest procés es repeteix en la riba de les conques on presenten límits fractals a diferents escales.



Figura 5.1: Doñana. Fotografies fetes per Héctor Garrido al Parc de Doñana [15]. Rius i maresmes amb estructura fractal.

L'exemple de riu "fractal" seria un riu trenat. Un riu trenat o riu entrelaçat té un llit que consisteix en una xarxa de canals separats per bancs o petites illes. Aquestes formacions

d'aigua presenten quasi similitud a diverses escales des dels punts més externs del riu fins al naixement d'aquest.



Figura 5.2: Riu entrellaçat. Fotografia del riu Tasman a Nova Zelanda.

5.1.2 Raigs

Els llamps presenten una estructura semblant a la d'un arbre fractal, ja que cada extensió d'aquest és una còpia d'una extensió inicial (iniciador). Per tant, cada extensió d'un raig és igual en tot el seu conjunt. La idea dels raigs com a fractals ve des de la concepció de les figures de Lichtenberg. Georg Christoph Lichtenberg va generar imatges produïdes descàrregues elèctriques ramificades sobre superfícies o interiors de materials aïllants, on es va observar que aquestes figueres presentaven ramificacions autosimilars.

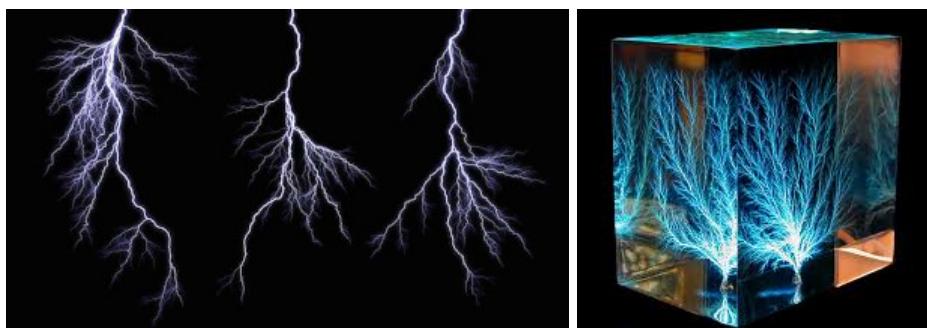


Figura 5.3: Raigs. A l'esquerra, imatges de 3 llamps diferents. A la dreta, una figura de Lichtenberg continguda en un vidre.

5.1.3 Muntanyes

Les muntanyes també presenten fractalitat en la seva estructura. Aquests accidents geogràfics, producte de les forces tectòniques i l'erosió de l'aigua, presenten en els seus pics i barris formacions més petites d'aquestes a diferents escales. Observant, la propietat d'autosimilitud. La fractalitat de les muntanyes va ser estudiada per Sig Handelman, Richard Voss i Ken Musgrave, els qui eren col·legues de Mandelbrot. Ells van mesurar la rugositat de les muntanyes i després van utilitzar aquests valors en programes, obtenint falsificacions fractals realistes i complexes. Aquestes falsificacions van ser utilitzades per Mandelbrot en les seves classes sobre geometria fractal en Yale, on les compara amb fotos de muntanyes reals, mostrant que només tenen lleugeres diferències. D'aquesta manera es va demostrar la naturalesa fractal de les muntanyes, perquè segueixen la mateixa "aleatorietat" per a formar-se.



Figura 5.4: Muntanyes. A l'esquerra, una muntanya feta per un programa mitjançant fractals. A la dreta, una muntanya dels Pirineus.

5.2 La fractalitat dels éssers vius

Els éssers vius també tenim certa naturalesa fractal [4, p. 55]. Les plantes presenten una estructura autosimilar en els seus raïms de flors i les seves arrels. El nostre interior també és una font d'estructures amb naturalesa fractal: sistema respiratori, circulatori i nervis. Els quals tenen models fractals.

5.2.1 Els pulmons

Els nostres pulmons són una estructura fractal per excel·lència. La finalitat dels pulmons és la de proporcionar oxigen pel nostre organisme, aquest òrgan té la capacitat de contenir mig litre d'aire i després intercanviar oxigen per CO_2 en menys d'un segon. Un procés tan exigent com aquest necessitaria una superfície gran, la qual, hauria d'estar continguda en un volum petit. Per fer-nos una idea, la superfície que ocupen els pulmons és de $130\ m^2$, mentre que els pulmons només poden contenir entre 5 a 6 litres d'oxigen.

Per dur a terme aquest procés els pulmons disposen de la tràquea i de les 23 generacions de ramificacions de l'arbre bronquial sumat els alvèols. Aquest sistema resulta eficient, però aquesta eficiència es deu a la disposició de la seva estructura, la qual és autosimilar. Les branques respiratòries es van dividint durant les 23 generacions en branques mare i filla. Aquestes estan dissenyades perquè en la zona d'intercanvi d'oxigen els seus diàmetres tinguin pocs canvis, alentint el flux d'aire.

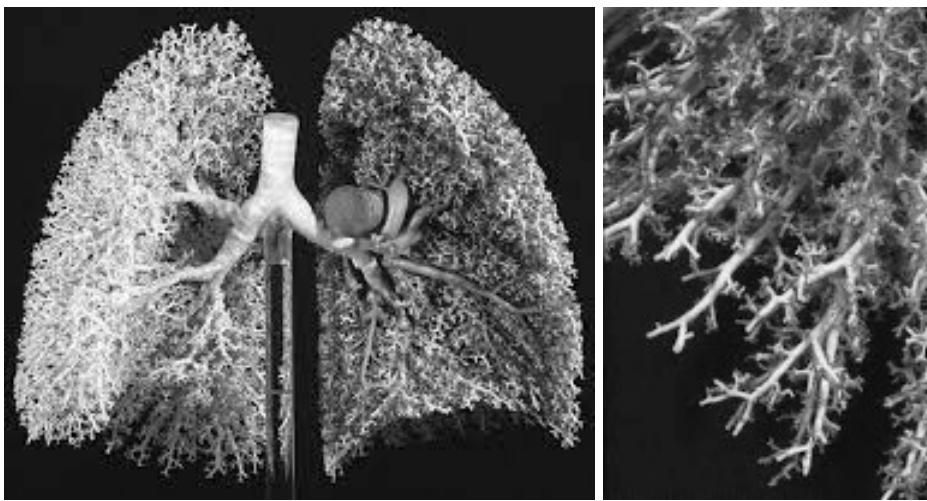


Figura 5.5: Pulmons. A l'esquerra, un model dels pulmons. A la dreta, una imatge ampliada d'un pulmó.

5.2.2 El sistema circulatori

El sistema cardiovascular és un altre exemple de geometria fractal dins del cos. Les ramifications del sistema circulatori (venes, artèries i capil·lars) tenen la missió de transportar sang per tot el cos. De la mateixa manera que amb el sistema respiratori, el sistema circulatori presenta unes dimensions exorbitades en comparació amb el nostre cos. Si es posa tots els vasos sanguinis d'un cos humà en línia recta, s'assoliria una longitud de 96000 km, una cosa impossible de contenir en un cos humà. Llavors "Com pot ser això?", la resposta és la mateixa que amb el sistema respiratori, una estructura amb naturalesa fractal. Les ramifications del sistema circulatori estan dissenyades per augmentar i disminuir-se a diferents escales, és a dir, presenta autosimilitud.

La presència dels fractals en el sistema circulatori també està present en el tractament de malalties coronàries. Molts estudis científics han demostrat que un cor malalt presenta una variació de la seva dimensió fractal, respecte a la d'un cor sa. Per fer-ho, s'utilitza el mètode de comptatge de capses, amb el qual, s'obté la dimensió fractal del cor. Un exemple, seria la malaltia de l'artèria oclusiva greu, on varia la dimensió fractal de l'arbre coronari esquerre.

5.3 Altres formacions naturals fractals

Existeixen moltes altres formacions naturals tant del medi ambient com dins nostre, que presenten la característica d'autosimilitud dels fractals. A continuació una taula amb més formacions fractals:

Nom	Representació
Ull	
Núvols	
ADN	
Falgueres	
Romanesco	

Taula 5.1: Formacions fractals naturals.

6. FRACTALS I APLICACIONS

6.1 L'impacte dels fractals a la tecnologia

De la mateixa manera que hem fet amb tots els coneixements que se n'han anat descobrint al llarg de la nostra història. Nosaltres, els humans, hem aprofitat els fractals i les seves propietats per fer uso d'aquest en la tecnologia. Apliquem els fractals en antenes multibanda, capes d'invisibilitat de microones, algoritmes de compressió d'imatges, fibra òptica i inclòs internet i la seva infraestructura.

6.1.1 Injectors de fluids fractals

Aquest dispositiu va ser ideat per Marc-Olivier [4, p. 90], un enginyer químic de la universitat de Londres, durant el projecte NICE (Nature Inspired Chemical Engineering). Aquest projecte, tenia l'objectiu de trobar solucions des de la natura als problemes de mescla, transport i reacció en dissenys d'enginyeria química.

La naturalesa fractal d'aquest dispositiu es troba en el modelatge de l'injector el qual segueix una estructura com la del sistema respiratori amb un conducte inicial, que es divideix en tubs més petits, iguals, que es van ramificant. Els injectors introdueixen el nou químic des de moltes sortides distribuïdes uniformement al voltant de l'espai desitjat. El fluid injectat surt per les sortides simultàniament en estar a la mateixa distància entre ells mateixos. Assegurant que les reaccions químiques es produixin de manera uniforme en tot el recipient.

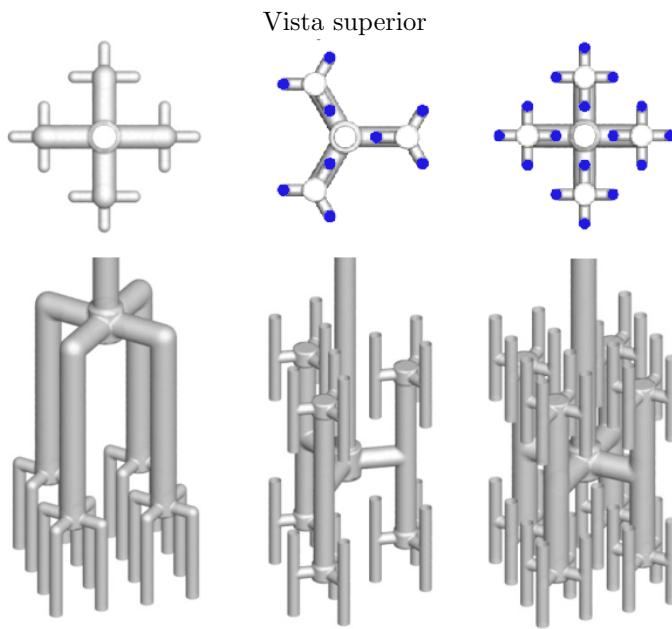


Figura 6.1: Representació 3D d'un injector fractal representat per iteracions.

6.1.2 Antenes fractals

Les antenes fractals són l'invent més destacat que incorpora fractals. Aquests dispositius són més flexibles que les abans convencionals, ja que no necessiten extensions electròniques (components) per augmentar la seva longitud. En comptes d'això incorporen diverses escales de longitud, fent-se més sensible a una gamma més àmplia de freqüències. Això, s'aconsegueix modificant la geometria de l'antena, utilitzant fractals, pel fet que la qualitat autosemblant dels fractals permet aconseguir aquest plantejament a la perfecció.

Les antenes fractals han demostrat tenir importants avantatges respecte a les antenes tradicionals. Aquests avantatges són les següents:

- L'estructura fractal d'aquestes antenes permeten reduir les dimensions de l'antena entre un 50 i un 75%, respecte a les antenes tradicionals.
- La seva estructura permet assolir diversos màxims de corrent.
- En ser antenes més petites requereixen menys components i circuits. També presenten menys elements radioactius.
- La seva estructura fractal combina la capacitància i la inductància de l'element.

Els principals desenvolupadors en antenes fractals són Nathan Cohen (Fractal Antenna Systems [14]) i Carlos Puente (Fractus. SA [9]). Aquestes empreses s'han encarregat de crear

diverses patents d'antena fractal. Aquestes antenes són utilitzades antenes captadores de metres, aeròports, instal·lacions i dispositius electrònics.

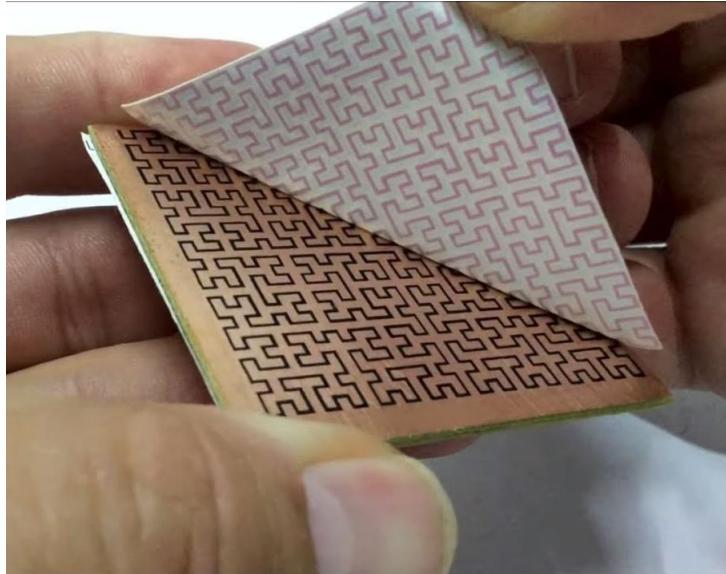


Figura 6.2: Antena fractal amb la forma de la corba de Peano.

6.1.3 Metamaterials fractals

Un altre invent que utilitza fractals són els anomenats metamaterials fractals. Els metamaterials fractals van ser obra de l'empresa Fractal Antenna Systems Inc., en la recerca de produir metamaterials electromagnètics millorats.

Els metamaterials fractals són una millora respecte als metamaterials electromagnètics convencionals, a causa dels seus ressonadors de banda ampla, els quals estan posats en un patró fractal. Aquesta característica, exclusiva d'aquests metamaterials, és la que permet que els metamaterials fractals poden aconseguir un rendiment de banda ampla i multibanda en els camps d'ocultació, apantallament, absorció i transmissió.

Els ressonadors absorbeixen la radiació electromagnètica, aconseguint un efecte d'ocultació o invisibilitat. De fet, l'absorció de radiació també s'encarrega de desviar la radiació a través de l'efecte d'ona superficial evanescent, la qual cosa, permet evitar la penetració electromagnètica externa. Aquestes propietats i processos permeten als metamaterials fractals proveir de diverses aplicacions:

- Càrrega sense fil
- Transferència de calor

- Sistema d'ocultació o "capa" d'invisibilitat
- Lents amb metamaterials
- Absorbidors de RF
- Sistemes de radar de vehicles

6.1.4 Les innovacions de Fractal Antenna Systems

Com ja hem vist abans, Fractal Antenna Systems ha produït tecnologies amb inspiració fractal, ara bé, aquests productes no són els únics, Fractal Antenna Systems ha intentat portar aquesta innovació fractal a més aparells.

Fractal Antenna ha treballat en la producció de bateries fractals, amb les quals, s'Intentarà reduir el nombre de dendrites de les bateries i millora la densitat de potència d'aquestes. També, mitjançant l'aplicació de geometria fractal, es busca obtenir electroimants petits amb un gran flux magnètic. D'altra banda, treballen en l'aplicació de plaques PCB amb patrons fractals, amb l'objectiu de reduir la corrosió de les plaques.

6.1.5 Computació i imatges

És en aquest àmbit on s'ha aprofitat un gran potencial dels fractals en la tecnologia. S'utilitzen tècniques amb fractals en la compressió d'imatges. En aquest procés el que es fa és trobar un sistema de funcions iterades, per a després, codificar la informació de la imatge en aquest. Per complementar aquest procés s'utilitza l'esquema de sistemes de funcions iterades fraccionades, amb el qual se subdivideix la imatge mitjançant particions.

Un altre ús dels fractals el trobem en el modelatge de paisatges entorns, on han sigut útils per produir renderitzats i efectes visuals que actualment són utilitzats al cinema o per generació d'imatges.

6.1.6 Fractus, SA

Com ja hem vist, els fractals han tingut una gran importància en el desenvolupament de noves tecnologies, sent que moltes empreses, utilitzen les propietats matemàtiques o estè-



Figura 6.3: Paisatge muntanyós amb una vall enmig fet amb fractals.

tiques dels fractals per la innovació tecnològica. És aquí on trobem a Fractus. SA. Aquesta empresa espanyola és un dels màxims exponents en tecnologia.

Aquesta empresa va fer-se coneguda, gràcies a la que fou, la seva patent més important, l'antena fractal. Fractus apareix com la idea de dos alumnes catalans de la UPC (Universitat Politècnica de Catalunya): Carles Puente i Rubén Bonet. Els quals es van especialitzar en el desenvolupament d'antenes multibanda i miniatura, amb l'objectiu de substituir les velles antenes.

És en aquest context que, durant els anys 1995 i 1998, Carles Puente donà conèixer el món la patent d'antena fractal. Aquest fet acabaria sent determinant tant a la història de fractus com per a la indústria dels mòbils, ja que va donar pas a una nova generació de telèfons mòbils menys toscos i amb proporcions més petites. Actualment, el 90 % dels mòbils presenten dins seu una antena fractal.

6.2 La cosmologia fractal

Tenint en compte que els fractals són formes associades a la natura en si mateixa, molts investigadors van començar a considerar models fractals com hipòtesis per resoldre diverses incògnites en cosmologia i astrofísica.

S'ha utilitzat hipòtesis fractals per resoldre la paradoxa d'Olbers, dins de la cosmologia observable. L'anomenada solució de Mandelbrot proposa que la lluminositat pot ser finita

i poden existir zones fosques en el cel si s'assumeix que la distribució de galàxies té una estructura fractal, sempre que a gran escala la dimensió fractal sigui inferior a 3. Aquesta hipòtesi també postula les estrelles en l'univers tindrien una distribució fractal similar al conjunt de Cantor, donant resposta a les àmplies àrees fosques.

En la cosmologia teòrica, els fractals s'han utilitzat en l'estudi de la hipotètica naturalesa irregular de l'espai-temps a petites escales a causa de fluctuacions quàntiques. En aquesta hipòtesi es proposa que a petites escales l'espai-temps hauria de tenir una estructura d'esponja quàntica.

També, s'ha emprat en l'estudi de la massa imaginària i els taquins. Un taquió és com s'anomena a una partícula teòrica que podria tenir velocitats superlumíniques.

7. PART PRÀCTICA

7.1 Formes fractals amb Python

La primera pràctica del treball és la construcció de formes fractals amb Jupyter, una eina que utilitza llenguatge de programació Python. Aquest apartat pràctic s'ha anat desenvolupant paral·lelament amb el marc teòric del treball amb l'objectiu d'obtenir un coneixement de programació amb fractals previ per a les pràctiques següents, cal destacar que la gran majoria de les formes fractals mostrades en aquest treball es van fer amb els codis que es mostraran en aquest apartat. Per tant, aquest apartat no està orientat a algun dels objectius presentats amb el treball, sinó a mostrar els programes i codis inicials que s'han utilitzat per produir les imatges del treball. També com a punt de partida en programació amb fractals.

7.1.1 Llibreries i funcions

Primerament, hem de definir algunes llibreries i funcions inicials que utilitzarem en els següents codis per programar les formes fractals. Les llibreries que utilitzarem són les següents:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

- **NumPy**: dissenyada pel llenguatge Python, NumPy ens proporcionarà funcions matemàtiques, vectors i matrius.
- **Matplotlib**: s'utilitza per graficar en 2D. Igual que NumPy, Matplotlib usa llenguatge Python.
- **Pandas**: aquesta llibreria Python l'utilitzarem per a l'anàlisi de dades i manipulació de taules, similar a excel.

Respecte les funcions, només utilitzarem una, que serà la funció **rotation()**. Per aquesta funció utilitzarem les fórmules obtingudes en l'annex 9.1.

```

# Primer definim la funció rotation i afegim els paràmetres següents (punt
# inicial , punt final , angle)
def rotation(start, end, angle):
    # Construïm la matriu
    M = np.array ([[np.cos(angle), -np.sin(angle)], [np.sin(angle), np.cos(angle)]])
    # Construïm un vector D, i l'igualem al punt inicial més + la matriu
    D = start + M@(end-start)
    # Quan s'executa la línia anterior guarda el vector rotat D
    return D

```

Aquesta funció ens permet fer rectes en 2D, facilitant la construcció d'algunes formes que requereixen desplaçaments per fer-se.

7.1.2 Conjunt de Cantor amb Python

```

# Comencem construint una funció , la qual , anomenarem cantor_set() . Els
# paràmetres d'aquesta funció seran els següents:
# a0 = coordenada x del punt 'a'
# a1 = coordenada y del punt 'a'
# b0 = coordenada x del punt 'b'
# b1 = coordenada y del punt 'b'
# ite = nombre d'iteracions
def cantor_set(a0,b0,a1,b1,ite):
    # Primer condicional .
    if ite==0:
        # Usem les coordenades 'x' i 'y' de 'a' i 'b' per construir l'iniciador.
        plt.plot([a0,b0],[a1,b1],color='black')
    # Segon condicional .
    else:
        # Mitjançant operacions , trobem les coordenades 'c0' i 'd0' en l'eix 'x'
        # .
        c0=a0+(b0-a0)/3.0
        d0=a0+2*(b0-a0)/3.0
        # Cridem a la funció cantor_set() i afegint els nous paràmetres
        # construïm el 1r generador.
        cantor_set(a0,c0,a1-0.05,a1-0.05,ite-1)
        cantor_set(d0,b0,b1-0.05,b1-0.05,ite-1)

    # Dibuixem el conjunt de Cantor amb 5 iteracions .
    for k in range(0,5):
        cantor_set(0,1,0,0,k)
        plt.axis('off')

```

7.1.3 Corba de Koch amb Python

```

# Començem definint la funció koch() amb els seus paràmetres
# ite = iteracions
# x,y = punts en el pla.
def koch(x,y,ite):

```

```

# Fem un condicional per la iteració 0, és a dir, l'iniciador.
if ite == 0:
    # Amb el plt.plot és traça una línia recta de a=(0,0) a b=(1,0) en el
    # pla.
    plt.plot([x[0],y[0]], [x[1],y[1]], color = 'red')
# Posem un segon condicional que servirà per construir altres punts i les
# iteracions del fractal.
else:
    # Definim els punts u, v, w. Per fer-ho operem amb els punts x i y.
    u = x + (y-x)/3
    v = x + 2 * (y-x)/3
    w = rotation(u,v,np.pi/3)
    # Un cop obtenim els punts utilitzem la funció koch() per traçar el 1r
    # generador
    koch(x,u,ite-1)
    koch(u,w,ite-1)
    koch(w,v,ite-1)
    koch(v,y,ite-1)

```

7.1.4 Triangle de Sierpiński amb Python

```

# Definim la funció triangle() amb els paràmetres següents:
# ite = iteracions
# x,y,z = punts en el pla
def triangle(x,y,z,ite):
    # Fem un condicional per a l'iniciador.
    if ite == 0:
        # Tracem un triangle amb els punts a, b, c
        plt.fill([x[0],y[0],z[0],x[0]],[x[1],y[1],z[1],x[1]], color = 'cyan')
    # Segon condicional per trobar altres punts.
    else:
        # Definim els punts u, v, w. Per fer-ho, operem amb els punts x i y.
        u = x + (z-x)/2
        v = z + (y-z)/2
        w = x + (y-x)/2
        # Tracem el 1r generador amb la funció triangle().
        triangle(x,w,u,ite-1)
        triangle(w,y,v,ite-1)
        triangle(u,v,z,ite-1)

# Agafem els punts a, b, c com a valors inicials al pla.
a = np.array([0,0])
b = np.array([1,0])
c = rotation(a,b,np.pi/3)

# Passem els punts com a valors de la funció triangle() i li assignem 4
# iteracions.
triangle(a,b,c,4)
plt.axis('equal')
plt.axis('off')

```

7.1.5 Tetraedre de Sierpiński amb Python

```
# Per a construir el tetraedre reutilitzarem la funció triangle() i afegirem
# algunes ordres més.
# Agafarem els punts 'b' i 'c' com a punts inicials.
b = np.array([0,0])
c = np.array([-0.5,2.5])
# Per trobar 'a' construirem una recta 'cb' i la rotarem amb la funció rotation
# -50°.
# Per a 'd' rotarem la recta 'cd' 30°.
a = rotation(c,b,-5*np.pi/18)
d = rotation(c,b,np.pi/6)
# Reutilitzem la funció triangle() amb els nous punts i li afegirem 4 iteracions
# .
triangle(a,b,c,4)
triangle(b,d,c,4)
plt.axis('equal')
plt.axis('off')
```

7.1.6 Catifa de Sierpiński amb Python

```
# Es defineix la funció com a carpet() i li posem els paràmetres següents:
# ite = iteracions.
# x,y,z,q = valors del pla.
def carpet(x,y,z,q,ite):
    # 1r Condicional per obtenir l'iniciador, en aquest cas, un quadrat.
    if ite == 0:
        # Amb el plt.fill es traça el quadrat seguint els punts x,y,z,q.
        plt.fill([x[0],y[0],z[0],q[0],x[0]],[x[1],y[1],z[1],q[1],x[1]],color =
# red)
    # Amb el 2n Condicional s'obtenen més punts mitjançant operacions.
    else:
        a = x + (y-x)/3
        b = x + 2*(y-x)/3
        c = y + (z-y)/3
        d = y + 2*(z-y)/3
        e = z + (q-z)/3
        f = z + 2*(q-z)/3
        g = q + (x-q)/3
        h = q + 2*(x-q)/3
        r = x + (z-x)/3
        s = y + (q-y)/3
        t = x + 2*(z-x)/3
        u = y + 2*(q-y)/3
        # Traçem el primer generador amb la funció carpet().
        carpet(x,a,r,h,ite-1)
        carpet(a,b,s,r,ite-1)
        carpet(b,y,c,s,ite-1)
        carpet(h,r,u,g,ite-1)
        carpet(s,c,d,t,ite-1)
        carpet(g,u,f,q,ite-1)
        carpet(u,t,e,f,ite-1)
        carpet(t,d,z,e,ite-1)

    # Utilitzarem els punts següents com punts inicials.
```

```

a = np.array([0,0])
b = np.array([1,0])
c = np.array([1,1])
d = np.array([0,1])

# Per últim, agafem els punts anteriors com a paràmetres de carpet() i els
# iterem 4 vegades.
carpet(a,b,c,d,4)
plt.axis('equal')
plt.axis('off')

```

7.1.7 Esponja de Menger amb Python

```

# Per construir l'esponja de Menger utilitzarem la funció carpet().
# Utilitzarem els punts a, b, c i d com a punts inicials. Aquests punts s'
# utilitzaran per generar els extrems del cub.
a = np.array([0,0])
b = np.array([1,0])
c = rotation(b,a,-np.pi/2)
d = rotation(a,b,np.pi/2)
# Per trobar 'e' es construeix una recta 'bc' i rotem -45°.
# En el cas de 'f' es construeix una recta 'cd' i es gira 135°.
# Per 'g' es gira una recta 'dc' 45°.
e = rotation(b,c,-np.pi/4)
f = rotation(c,b,3*np.pi/4)
g = rotation(d,c,np.pi/4)
# Mitjançant operacions, obtenim tres punts més, els quals ens serviran per
# tancar el cub.
m = b + 3*(e-b)/4
n = c + 3*(f-c)/4
o = d + 3*(g-d)/4
# Amb la funció carpet() dibuixarem les cares del cub utilitzant els punts
# obtinguts com a paràmetres i els iterem 4 vegades.
carpet(a,b,c,d,3)
carpet(b,m,n,c,3)
carpet(d,c,n,o,3)
plt.axis('equal')
plt.axis('off')

```

7.1.8 Corba de drac amb Python

```

# Definim la nostra funció com a drac(), i li agreguem els paràmetres següents:
# ite = iteracions
# x,y,z = valors al pla
def drac(x,y,z,ite):
    # El 1r condicional servirà per generar l'iniciador.
    if ite == 0:
        # Utilitzem els punts x, y i z.
        plt.plot([x[0],y[0],z[0]],[x[1],y[1],z[1]])
    # En el 2n condicional generem dos punts: 'u' i 'v'.
    else:
        u = x + (z-x)/2

```

```

v = rotation(x,u,np.pi/2)
# Posem els paràmetres a la funció drac() i tracem el 1r generador.
drac(x,v,y,ite-1)
drac(z,u,y,ite-1)

# Agafem els punts 'a', 'b' i 'c' com a punts inicials.
a = np.array([0,0])
b = np.array([.5,.5])
c = np.array([1,0])

# Agafem els punts anteriors com a paràmetres en drac(). Iterem els punts 10
# vegades.
drac(a,b,c,10)
plt.axis('equal')
plt.axis('off');

```

7.1.9 Corba de Peano amb Python

```

# Fem una nova funció anomenada peano(). Els seus paràmetres són els següents:
# ite = iteracions
# x,y = valors al pla
def peano(x,y,ite):
    # Amb el 1r condicional farem l'iniciador.
    if ite == 0:
        #Construim el nostre iniciador amb els punts x=(0,0) i y=(1,0). En
        # aquest cas l'iniciador és una recta.
        plt.plot([x[0],y[0]], [x[1],y[1]], color ='blue')
    # El 2n condicional servirà per trobar els punts per obtenir el 1r generador
    # .
    else:
        # Tracem els punts 'a' i 'b' mitjançant operacions.
        a = x + (y-x)/3
        b = x + 2*(y-x)/3
        # Per obtenir els següents punts utilitzarem rotacions.
        # Per obtenir 'c' girem una recta 'ab' 90º
        # Per obtenir 'd' girem una recta 'by' 90º
        # Per obtenir 'e' girem una recta 'ab' -90º
        # Per obtenir 'f' girem una recta 'by' -90º
        c = rotation(a,b,np.pi/2)
        d = rotation(b,y,np.pi/2)
        e = rotation(a,b,-np.pi/2)
        f = rotation(b,y,-np.pi/2)
        # El següent pas és traçar el 1r generador. Per fer-ho, utilitzarem la
        # funció peano() i usarem de paràmetres els valors obtinguts
        # anteriorment.
        peano(x,a,ite-1)
        peano(a,c,ite-1)
        peano(c,d,ite-1)
        peano(d,b,ite-1)
        peano(b,f,ite-1)
        peano(f,e,ite-1)
        peano(e,a,ite-1)
        peano(a,b,ite-1)
        peano(b,y,ite-1)

    # Utilitzem els punts 'x' i 'y' com a punts inicials.

```

```

x = np.array([0,0])
y = np.array([1,0])

# Iterem els punts 'x' i 'y' 4 vegades. Obtenint la corba.
peano(x,y,4)
plt.axis('equal')
plt.axis('off')

```

7.1.10 Corba de Hilbert amb Python

```

# Comencem fent una funció per trobar els punts per fer la corba. Aquesta funció
# ↪ l'anomenem points_hilbert().
def points_hilbert(ite):
    # Establim el 1r condicional per a la iteració 0 amb els valors de x i y.
    if ite == 0:
        x = 0
        y = 0
    # El 2n condicional servirà per totes les iteracions majors que 1.
    else:
        x0,y0 = points_hilbert(ite -1)
        # Establim una malla d'1X1 que va des de -0.5 fins a 0.5. En ella
        # ↪ trobarem tots els valors de x i y necessaris per construir la
        # ↪ corba.
        x = 0.5*np.array([-0.5+y0,-0.5+x0,0.5+x0,0.5-y0])
        y = 0.5*np.array([-0.5+x0,0.5+y0,0.5+y0,-0.5-x0])
    # Dins del return guardem tots els valors de x i y que s'obtenen.
    return x, y

# Fem una nova funció, anomenada plot_hilbert() i li afegim com a únic paràmetre
# ↪ n.
def plot_hilbert(n):
    # Utilitzem la funció points_hilbert() i li posem de paràmetre n.
    # n = nombre d'iteracions.
    x,y = points_hilbert(n)
    # Canviem la funció points_hilbert() perquè pugui imprimir els valors com un
    # ↪ espai 1-dimensional.
    X = np.reshape(x,4**n)
    Y = np.reshape(y,4**n)
    plt.plot(X,Y,color = 'purple')
    plt.axis('off')
    plt.axis('equal')

plot_hilbert(5)

```

7.1.11 Conjunt de Julia amb Python

```

# Fem una nova funció, la qual, anomenarem max_iteration(). Els paràmetres d'
# ↪ aquesta funció seran:
# z = nombres complexos.
# c = nombre d'un complex concret.
# ite = iteracions
def max_iteration(z,c,ite):

```

```

# Utilitzem un bucle for , amb el qual , obtindrem la iteració 'k' , on el
# → valor del mòdul és |z| > 2.
for k in range(ite):
    z = z**2+c
    if(abs(z)>2):
        break
# Amb el return k , guardem l'última iteració on z és major de 2.
return k

# Definim una nova funció com a plot_julia(). Els paràmetres de la funció són:
# n = nombre de punts en un rang de -2 a 2 per a 'x' i 'y'.
# c = nombre complex fix de la funció f(z) = z*z+c
# ite = nombre de iteracions
def plot_julia(n,c,ite):
    # Construïm dos eixos de -2 a 2 en 'x' i 'y'.
    x = np.linspace(-2,2,n)
    y = np.linspace(-2,2,n)
    # Construïm una matriu de zeros de dimensió nxn.
    A = np.zeros((n,n))
    # Amb dos bucles for , construïm una malla de nombres complexos 'z' amb els
    # → valors 'x' i 'y'.
    for p in range(n):
        for q in range(n):
            z = complex(x[p],y[q])
            # Actualitzem la matriu , substituint cada zero pel valor de 'k' que
            # → retorna la funció max_iteration().
            A[p,q] = max_iteration(z,c,ite)
    # Matriu actualitzada amb els valors de 'k' de cada complex 'z'.
    return A

# Dibuixem el conjunt amb les següents característiques :
# n = 500 punts
# c = -0.5-1j*0.5
# ite = 30
A = plot_julia(500,-0.5-1j*0.5,30)

# Amb la libreria matplotlib dibuixem la matriu A.
plt.figure(figsize=(10, 10))
plt.imshow(A.T, interpolation="nearest");

```

7.1.12 Conjunt de Mandelbrot amb Python

```

# Creem una nova funció , la qual anomenarem plot_mandelbrot() . Aquesta funció
# → tindrà els paràmetres següents:
# n = nombre de punts en un rang de -2 a 2.
# ite = nombre d'iteracions.
def plot_mandelbrot(n,ite):
    # Construïm dos eixos de -2 a 2 en 'x' i 'y'. També farem una matriu de
    # → zeros de dimensió nxn.
    x = np.linspace(-2,2,n)
    y = np.linspace(-2,2,n)
    A = np.zeros((n,n))
    # Amb dos bucles for , construïm una malla de nombres complexos 'c' amb els
    # → valors 'x' i 'y'.
    for p in range(n):
        for q in range(n):

```

```

c = complex(x[p],y[q])
# Actualitzem la matriu, substituint cada zero pel valor de 'k' que
    ↪ retorna la funció max_iteration(). En aquest cas, z = 0. Al
    ↪ definir 'z' com a 0 estem actualitzant el valor de c.
# c = tots els nombres complexos.
A[p,q] = max_iteration(0,c,ite)
# La matriu s'actualitzara amb els valors de 'A' obtinguts en cada nova
    ↪ iteració.
return A

# Ara imprimim els resultats amb 300 i 30 iteracions , per a la matriu A.
A = plot_mandelbrot(300,30)
# Amb la llibreria matplotlib dibuixem la matriu A.
plt.figure(figsize=(10, 10))
plt.imshow(A.T, interpolation="nearest");

```

7.2 Generació de paisatges amb fractals

Com vam explicar en el capítol 5 del treball, els fractals es troben presents en la natura, inclòs, són emprats en l'estudi de muntanyes. Amb la teoria ja interioritzada prèviament, toca posar això en practica. En aquesta pràctica, l'objectiu és generar petits paisatges utilitzant únicament fractals, i respondra a les següents preguntes:

- És possible crear un paisatge a partir d'un sol fractal?
- Es podran obtenir més d'un tipus de paisatge?

Igual que a la primera pràctica, utilitzarem el llenguatge de programació Python per generar aquests mini paisatges.

7.2.1 Llibreries

Per aquesta pràctica tornarem a utilitzar les mateixes llibreries de la Secció 7.1, però hi afegirem unes noves que ens serviran per fer models 3D.

```

import mayavi.mlab
import numpy as np
np.bool_ = np.bool_
import matplotlib.pyplot as plt
import scipy.ndimage
from matplotlib import colormaps
from matplotlib import colors
from matplotlib import cm

```

- **NumPy**: dissenyada pel llenguatge Python, NumPy ens proporcionarà funcions matemàtiques, vectors i matrius.
- **Matplotlib**: s'utilitza per graficar en 2D. Igual que NumPy, Matplotlib usa llenguatge Python.
- **Mayavi.mlab**: aquesta llibreria Python serveix per generar models en 3D. L'utilitzarem per crear els mini paisatges fractals.
- **scipy.ndimage**: scipy serveix per analitzar i processar imatges multidimensionals.

També, afegirem diferents patrons de color des de la llibreria Matplotlib:

- **colormaps**
- **colors**
- **cm**

7.2.2 Paisatges generats amb Mandelbrot

```
# Creem una nova funció , la qual anomenarem mandelbrot() . Aquesta funció tindrà
    ↪ els paràmetres següents:
# c = numero de nombres complexos en el pla en un rang de [x_min , x_max]X[y_min ,
    ↪ y_max].
# ite = nombre d'iteracions .
def mandelbrot(c,ite):
    # Usem com a punt d'inici el nombre complex z=0+0i
    z=complex(0,0)
    # Amb el següent bucle es crea una successió de nombres complexos c(0) , c(1)
        ↪ , c(2) , etc. Aquest procés s'executará fins que es trobi la iteració c
        ↪ (k) . Tal que
    # |c(k)|>2.
    for k in range(ite):
        Z=z*z+c
        if abs(z)>2:
            break
            pass
        pass
    return k

# Definim una altra funció , que servirà per dibuixar regions del conjunt de
    ↪ Mandelbrot , utilitzant un nombre complex 'z' concret.
# z = nombres complexos
# dx = rang de capacitat pels valors reals , per obtenir la regió a graficar .
# dy = rang de capacitat pels valors imaginaris , per obtenir la regió a graficar
    ↪ .
# n = nombre de punts en la regió en els eixos x i y .
def plot_mandelbrot3D(z,dx,dy,n):
    # Construïm la regió de [ real(z)-dx , real(z)+dx ]X[imag(z)-dy , imag(z)+dy ].
    x=np.linspace(np.real(z)-dx,np.real(z)+dx,n)
    y=np.linspace(np.imag(z)-dy,np.imag(z)+dy,n)
    # Fem una matriu de zeros de dimensió nxn.
    a=np.zeros((n,n))
```

```

# Amb dos bucles for, construïm una malla de nombres complexos 'c' amb els
# → valors 'x' i 'y'.
for k in range(n):
    for j in range(n):
        c=complex(x[k],y[j])
        # Actualitzem la matriu, substituint cada zero pel valor de 'k' que
        # → retorna la funció mandelbrot().
        # c = tots el nombres complexos.
        a[k,j]=mandelbrot(c,40)
        # La matriu s'actualitzarà amb els valors de 'a' obtinguts en cada
        # → nova iteració.
    pass
pass

# Amb la primera línia farem que mayavi ens doni els resultats en un fons
# → blanc
mayavi.mlab.figure(bgcolor=(1, 1, 1))
# A partir d'aquí tenim dos escenaris diferents:

#1: Si volem un model predeterminat utilitzem la següent línia.
mayavi.mlab.surf(a.T, warp_scale=5.5,colormap = 'GnBu')
#2: Si volem que el model tingui més detall, utilitzem les dues línies
# → següents i desactivem l'anterior amb #.
smoothed_atlas = scipy.ndimage.gaussian_filter(a.T, 2)
mayavi.mlab.surf(smoothed_atlas , warp_scale=5.5,colormap = 'terrain')
# Mostrar la imatge
mayavi.mlab.show()

```

Ara, per a construir un paisatge només fiquem els valors que vulguem als paràmetres següents.

Exemple 7.1.

```

dx= 0.029210087940185203
dy= 0.021907565955138875
z1=-0.2475791267965755+0.7568966408535892j
n=500
plot_mandelbrot3D(z1,dx,dy,n)

```

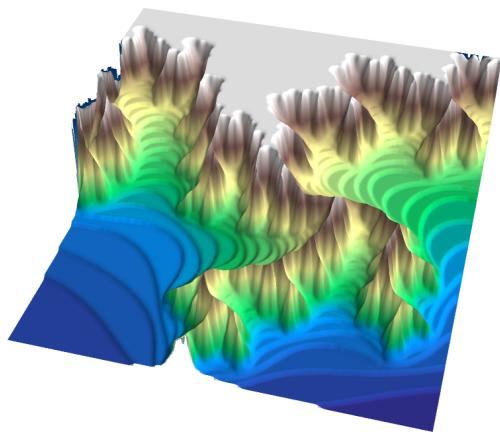


Figura 7.1: Cap generat amb el nombre complex $z_1 = 0.2475791267965755 + 0.7568966408535892i$. El mapa de colors utilitzat ha sigut el "terrain".

Exemple 7.2.

```

dx= 0.09253012048192777
dy= 0.0693975903614458
z2=-1.3363855421686746+0.0038585209003218005j
n=500
plot_mandelbrot3D(z2,dx,dy,n)

```

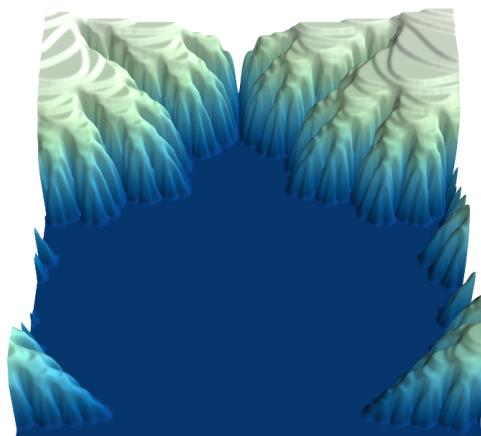


Figura 7.2: Glacera amb un llac enmig generat amb el nombre complex $z_2 = -1.3363855421686746 + 0.0038585209003218005i$. El mapa de colors utilitzat ha sigut el "GnBu".

7.2.3 Paisatges generats amb Julia

```
# Com a punt de partida , definim una funció com a julia () . Aquesta funció tindrà
    ↪ els paràmetres següents:
# c = nombre complex especific
# z = nombres complexos
# ite = nombre d'iteracions
def julia(c,z,ite):
    for k in range(ite):
        z=z*z+c
        if abs(z)>4:
            break
            pass
    pass
    return k

# Valors de la malla .
x_min = -1.5
x_max = 1.5
y_min = -1.5
y_max = 1.5

# A continuació definim una altra funció , la qual , anomenarem plot_julia3D () .
    ↪ Aquesta funció ens servirà per crear el paisatge 3D.
def plot_julia3D():
    # Construïm dos eixos de [x_min , x_max]X[y_min , y_max] . També farem una
        ↪ matriu de zeros de dimensió nxn.
    x=np.linspace(x_min,x_max,n)
    y=np.linspace(y_min,y_max,n)
    a=np.zeros((n,n))
    # Amb dos bucles for , construïm una malla de nombres complexos 'z' amb els
        ↪ valors 'x' i 'y'.
    for i in range(n):
        for j in range(n):
            z=complex(x[i],y[j])
            a[i,j]=julia(c,z,40)
            #print(a,c)
            pass
    pass

    # Amb la primera línia farem que mayavi ens doni els resultats en un fons
        ↪ blanc
    mayavi.mlab.figure(bgcolor=(1, 1, 1))
    # A partir d'aquí tenim dos escenaris diferents:

    #1: Si volem un model predeterminat utilitzem la següent línia .
    mayavi.mlab.surf(a.T, warp_scale=2.5,colormap = 'YIGn')
    #2: Si la volem amb més detall , utilitzem les dues línies següents i
        ↪ desactivem l'anterior.
    smoothed_atlas = scipy.ndimage.gaussian_filter(a.T, 2)
    mayavi.mlab.surf(smoothed_atlas , warp_scale=2.5,colormap = 'terrain')
    # Mostrar la imatge
    mayavi.mlab.show()
```

Ara, per a construir un paisatge només fiquem els valors que vulguem als paràmetres següents.

```
# Valors de la malla.  
x_min = -1.5  
x_max = 1.5  
y_min = -1.5  
y_max = 1.5  
  
# Nombre de punts en la malla  
n=500
```

Exemple 7.3.

```
# Nombre complex específic  
c = complex(0.4,0.4)  
  
# Dibuixem el paisatge amb l'opció 1.  
mayavi.mlab.surf(a.T, warp_scale=2.5,colormap = 'YIGn')  
plot_julia3D(x_min,x_max,y_min,y_max,c,n)
```

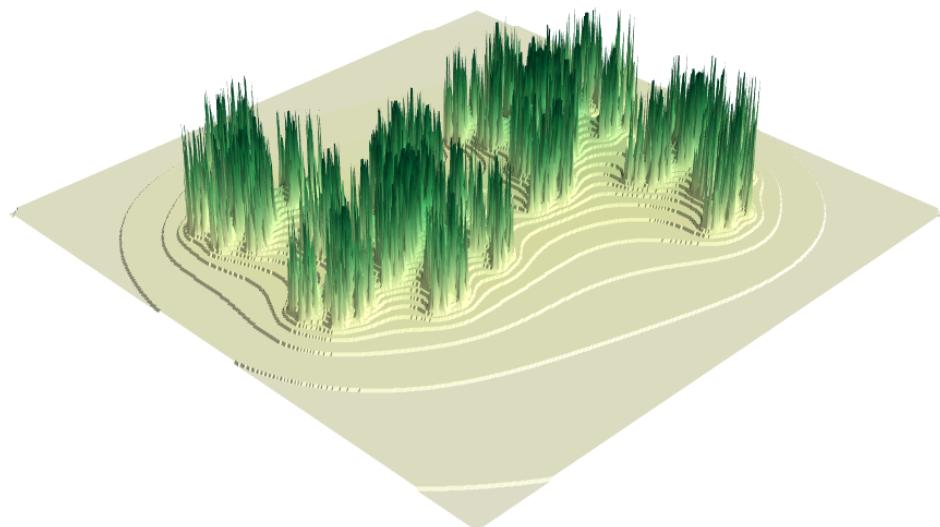


Figura 7.3: Petit bosc generat amb el conjunt de Julia de $c=0.4+0.4i$.

Exemple 7.4.

```
# Nombre complex específic  
c = complex(0,0.8)  
  
# Dibuixem el paisatge amb l'opció 1.  
mayavi.mlab.surf(a.T, warp_scale=2.5,colormap = 'YIGn')  
plot_julia3D(x_min,x_max,y_min,y_max,c,n)
```

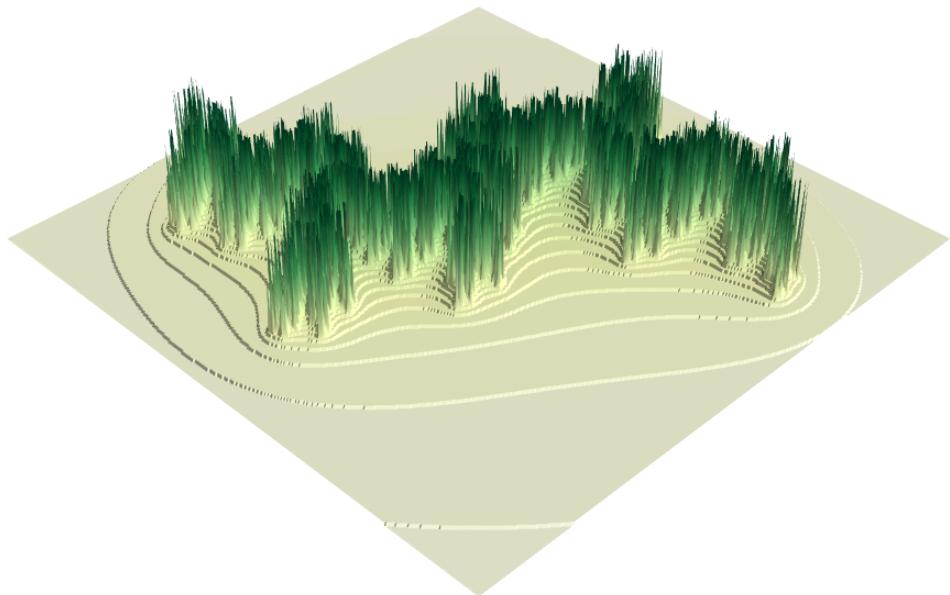


Figura 7.4: Altre bosc generat amb el conjunt de Julia de $c=0+0.8i$.

Exemple 7.5.

```
# Nombre complex específic
c = complex(0.4,0.4)

# Dibuixem el paisatge amb l'opció 2.
smoothed_atlas = scipy.ndimage.gaussian_filter(a.T, 2)
mayavi.mlab.surf(smoothed_atlas, warp_scale=2.5,colormap = 'terrain')
plot_julia3D(x_min,x_max,y_min,y_max,c,n)
```

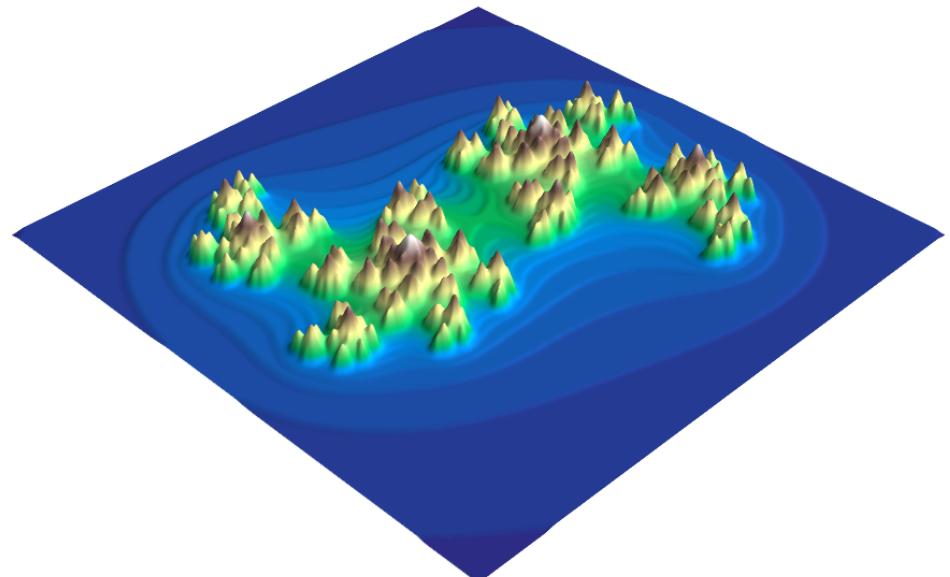


Figura 7.5: Petit arxipèlag generat amb el conjunt de Julia $c=0.4+0.4i$.

Exemple 7.6.

```
# Nomre complex específic
c = complex(0,0.8)

# Dibuixem el paisatge amb l'opció 2.
smoothed_atlas = scipy.ndimage.gaussian_filter(a.T, 2)
mayavi.mlab.surf(smoothed_atlas, warp_scale=2.5,colormap = 'terrain')
plot_julia3D(x_min,x_max,y_min,y_max,c,n)
```

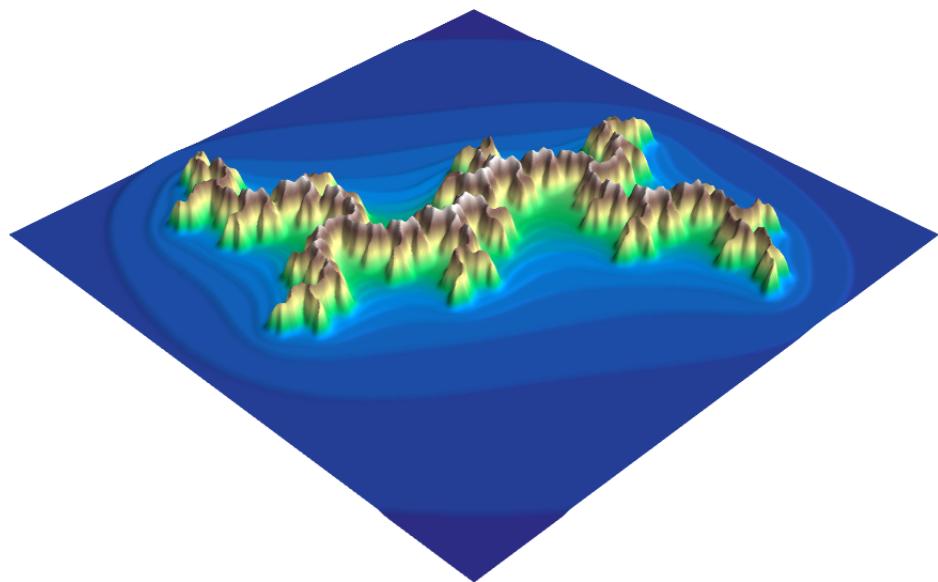


Figura 7.6: Illa generada amb el conjunt de Julia $c=0+0.8i$.

7.2.4 Paisatges generats amb el fractal de Newton

El codi següent genera una imatge fractal acolorint els punts del pla complex utilitzats com a estimació inicial per l'arrel trobada.

```
# Com a punt de partida , definim una funció com a newton_fractal() . Aquesta
#   ↪ funció tindrà els paràmetres següents:
# f = funció
# df = derivada de la funció f
# z0 = nombre complex arbitrari que s'apropi a un zero de f. Un zero és un
#   ↪ anàleg a un punt de tall en el pla (x, y) , però en els complexos.
# ite = nombre d'iteracions
def newton_fractal(f,df,z0,ite):
    # tol = La tolerància: un petit valor que serveix com a valor límit , és a
    #   ↪ dir , només agafem els valors que siguin majors a la tolerància.
    tol = 1e-8
    z=z0
    for k in range(ite):
        dz = f(z)/df(z)
        if abs(dz)<tol:
            return k
        z = z - 1.25*dz

# Definim la funció f i la seva derivada.
f=lambda z : z**4 + 3j - 1
df=lambda z : (4 + 3j)*z**3 + 3j

# Valors de la malla .
x_min = 0.4
x_max = 0.9
y_min = -0.9
y_max = -0.3

# Punts de la malla
n_space = 300
# Construïm dos eixos de [x_min , x_max]X[y_min , y_max]. També farem una matriu
#   ↪ de zeros de dimensió n_space X n_space.
x_values=np.linspace(x_min,x_max,n_space)
y_values=np.linspace(y_min,y_max,n_space)
n=len(x_values)
m=len(y_values)
# Fem una matriu de zeros de dimensió nxm.
a=np.zeros((n,m))

# Amb dos bucles for , construïm una malla de nombres complexos 'z' amb els
#   ↪ valors 'x' i 'y'.
for k in range(n):
    for j in range(m):
        z=complex(x_values[k],y_values[j])
        # Actualitzem la matriu , substituint cada zero pel valor del mòdul de 'z
        #   ↪ ' que retorna la funció newton_fractal() .
        # z = tots el nombres complexos.
        a[k,j]=newton_fractal(f,df,z,200)
        # La matriu s'actualitzarà amb els valors de 'a' obtinguts en cada nova
        #   ↪ iteració .
        pass
    pass
```

```

# Amb la primera línia farem que mayavi ens doni els resultats en un fons blanc
mayavi.mlab.figure(bgcolor=(1, 1, 1))
# Per obtenir més detall, utilitzem les dues línies següents.
smoothed_atlas = scipy.ndimage.gaussian_filter(a.T, 1.5)
mayavi.mlab.surf(smoothed_atlas, warp_scale=2.5,colormap = 'GnBu')
# Mostrar la imatge
mayavi.mlab.show()

```

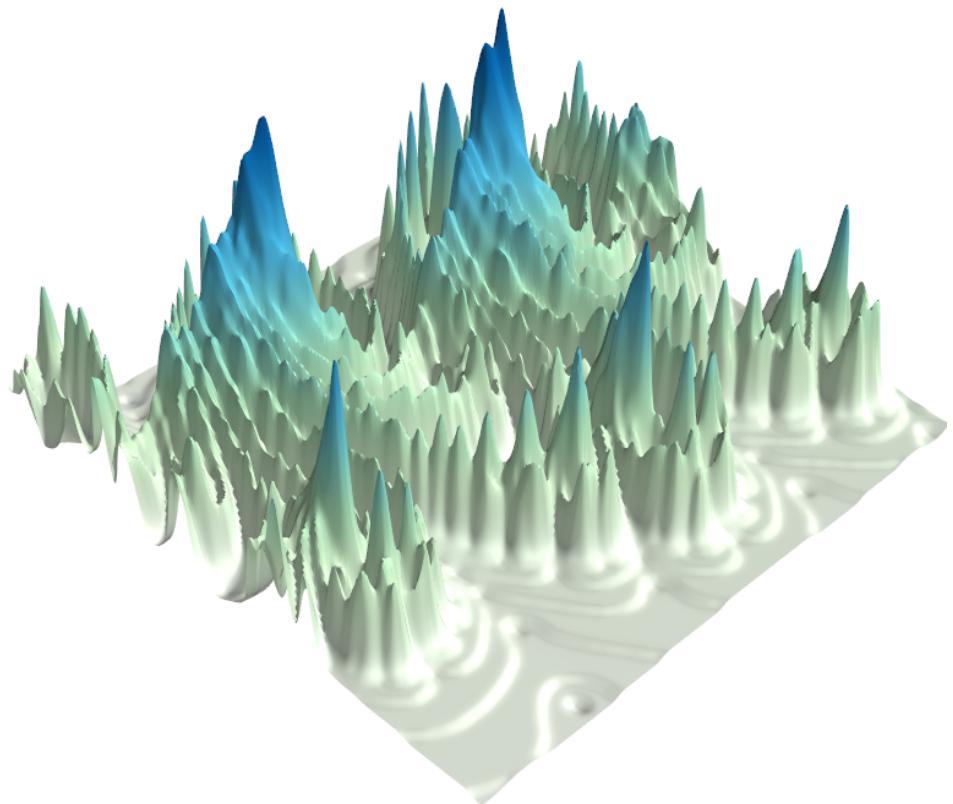


Figura 7.7: Serralada nevada obtinguda amb el polinomi de variable complexa $z^{4+3i} - 1$.

8. CONCLUSIÓ

Com a conclusió del treball, veiem que els resultats obtinguts de la pràctica han sigut un total de 7 paisatges: 2 illes, un penya-segat, una glacera, una serralada nevada i 2 boscos. Aquests resultats ens donen una resposta afirmativa a les qüestions plantejades, però no de la manera plantejada a l'inici. En el cas de la primera qüestió (És possible crear un paisatge a partir d'un sol fractal?), els resultats obtinguts mostren que per als conjunts de Julia, si es pot modelar un paisatge, però, per al conjunt de Mandelbrot i el fractal de Newton només es necessita una regió o secció d'aquest per programar un model 3D de paisatge. Per a la segona qüestió (Es podran obtenir més d'un tipus de paisatge?), els resultats mostren que es pot, ja que s'han obtingut diversos paisatges per a cada fractal utilitzat, sent la glacera i el penya-segat per a Mandelbrot, les illes i boscos per a Julia i la serralada nevada pel fractal de Newton.

En l'àmbit teòric la proposta inicial s'ha mantingut, ja que els coneixements teòrics plantejats al principi sobre els fractals han sigut els mateixos fins al final, tot i que sí que és veritat que a mesura que el treball avançava es van haver de revisar i afegir altres conceptes (annexos) que es van necessitar per avançar el treball. En l'apartat pràctic a l'inici del treball es va plantejar la construcció d'una antena fractal, malauradament, algunes decisions mal preses i sobretot la dificultat per fer aquest experiment a escala pràctica i teòrica, van portar a considerar inviable la seva posada en pràctica. Per aquest motiu, la pràctica es va centrar en els paisatges 3D i les seves qüestions plantejades, les quals es van resoldre posteriorment.

Un cop acabat el treball s'ha observat que una manera d'avançar-lo o millorar-lo hauria estat considerar la possibilitat de realitzar més paisatges, però amb funcions de Mandelbrot i Julia diferents de les funcions polinòmiques quadràtiques vistes en el treball (veure equacions 3.3 i 3.2).

En general, el procés que s'ha seguit per a fer aquest treball i la seva execució es pot qualificar d'extens, i especialment complex, perquè va ser necessari aprendre de conceptes i fórmules matemàtiques desconegudes per a mi, algunes les quals, estic aviat a donar en aquest curs acadèmic. També em va portar a expandir els meus coneixements en programació i l'ús d'una nova eina (Python).

Com a reflexió personal, crec que els resultats obtinguts podrien extrapolar-se i usar-se en les tècniques d'efectes visuals (CGI, VFX, etc.) en els seus diferents camps d'ús (pel·lícules, sèries, videojocs, etc.). Això, com un complement per a l'elaboració d'aquests efectes, clar, des d'una visió subjectiva dels meus resultats obtinguts.

9. ANNEX

9.1 Rotació d'un segment en el pla

Es defineix com a rotació, una transformació on una recta \overline{AB} gira un angle θ al voltant del punt fix A , per obtenir el punt B' , tal que, \overline{AB} i $\overline{AB'}$ tinguin la mateixa longitud r .

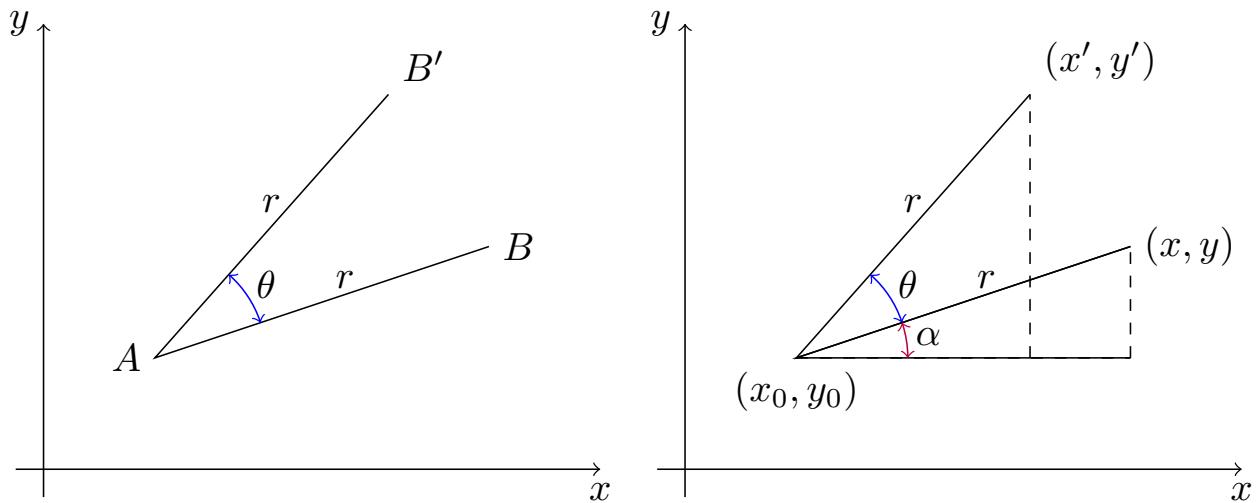


Figura 9.1: Rotació d'un segment \overline{AB} en el pla.

Siguin (x_0, y_0) , (x, y) i (x', y') les coordenades dels punts A , B i B' respectivament (veure Figura 9.1). Llavors:

$$r \cos(\alpha) = x - x_0 \quad \text{i} \quad r \sin(\alpha) = y - y_0. \quad (9.1)$$

$$\cos(\alpha + \theta) = \frac{x' - x_0}{r} \quad \text{i} \quad \sin(\alpha + \theta) = \frac{y' - y_0}{r}. \quad (9.2)$$

Resolem l'equació 9.2

$$\begin{aligned} r \cos \alpha \cos \theta - r \sin \alpha \sin \theta &= x' - x_0. \\ r \sin \alpha \cos \theta + r \cos \alpha \sin \theta &= y' - y_0. \end{aligned} \quad (9.3)$$

Substituem l'equació 9.1 en l'equació 9.3

$$\begin{aligned}(x - x_0) \cos \theta - (y - y_0) \sin \theta &= x' - x_0. \\ (y - y_0) \cos \theta + (x - x_0) \sin \theta &= y' - y_0.\end{aligned}\tag{9.4}$$

Per tant, de l'equació 9.4 obtenim:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} + \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x - x_0 \\ y - y_0 \end{pmatrix}$$

9.2 El conjunt dels nombres complexos

Un nombre complex és una expressió de la forma $z = a + ib$, on $a, b \in \mathbb{R}$ i $i = \sqrt{-1}$.

Definim el pla complex \mathbb{C} com:

$$\mathbb{C} = \{z = a + ib : a, b \in \mathbb{R}\}.$$

Un nombre complex també es pot representar com el parell ordenat (a, b) , on a i b són la part real i imaginària de z respectivament.

El **mòdul** d'un nombre complex z és la distància de l'origen cap al parell ordenat (a, b) i es representa per:

$$|z| = \sqrt{a^2 + b^2} = \sqrt{z \cdot z}.$$

9.3 Mètode de Newton-Raphson

El mètode de **Newton-Raphson** per a trobar les arrels d'una funció pren una estimació inicial d'una arrel, x_0 , i cerca successivament millors aproximacions de la mateixa de la manera següent:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, \quad n = 0, 1, \dots$$

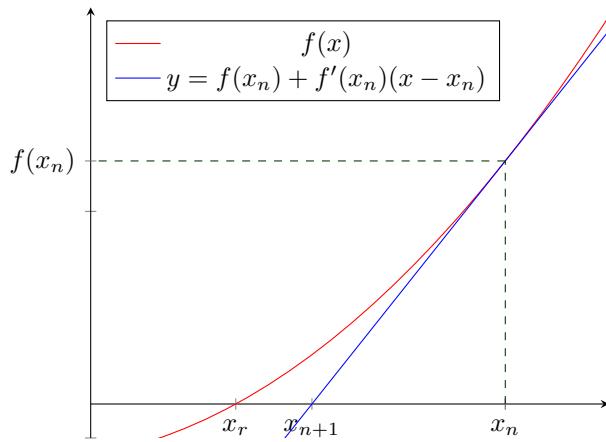


Figura 9.2: Mètode de Newton-Raphson. Per a trobar les arrels d'una funció f , s'inicia l'aproximació de l'arrel x_r de f amb un punt inicial x_n .

És a dir, en cada iteració, l'arrel s'aproxima a x_{n+1} , la intercepció en l'eix x de la tangent a la gràfica en $f(x_n)$ (veure Figura 9.2). Quan s'aplica a funcions de variable complexa z , el mètode pot utilitzar-se per a crear fractals interessants si es considera a quina arrel convergeix per a un conjunt de números en el pla complex.

BIBLIOGRAFIA

Llibres

- [1] Carlo Cattani, Anouar Ben Mabrouk i Sabrine Arfaoui. *Fractal Analysis: Basic Concepts And Applications*. World Scientific Publishing Co. Pte. Ltd., 2022.
- [2] Robert L. Devaney. *Chaos and Fractals New Frontiers of Science*. 2a ed. CRC Press, 2020.
- [3] Michael Frame i Nial Neger. *Kitchen Science Fractals: A Lab Manual for Fractal*. World Scientific Publishing Co. Pte. Ltd, 2021.
- [4] Michael Frame i Amelia Urry. *Fractal Worlds: Grown, Built, and Imagined*. Yale University Press, 2016.
- [5] Heinz-Otto Peitgen, Hartmut Jürgens i Dietmar Saupe. *Chaos and Fractals New Frontiers of Science*. 2a ed. Springer, 2004.

Pàgines web

- [6] Asociacionceat. *Peano*. Jul. de 2024. URL: <https://www.asociacionceat.org/aw/2/peano.htm>.
- [7] Complejidad.net. *Historia de los fractales y de su geometría*. Abr. de 2024. URL: <https://complejidad.net/2017/03/21/historia-de-los-fractales-y-de-su-geometria/>.
- [8] EFE. *La historia de la película de la catalana fractus cuya tecnología usa tu móvil*. Ag. de 2024. URL: <https://efe.com/cataluna/2023-01-09/la-historia-de-pelicula-de-la-catalana-fractus-cuya-tecnologia-usa-tu-movil-3/>.
- [9] Fractus. *FRACTUS / Inventors, Innovators, Partners*. Jul. de 2024. URL: <https://www.fractus.com/>.
- [10] IBM Inc. *Benoît Mandelbrot*. Gen. de 2024. URL: <https://www.ibm.com/history/benoit-mandelbrot>.

- [11] Institucional. *Fractales: bellos y sin embargo útiles*. Jul. de 2024. URL: <https://institucional.us.es/blogimus/2018/10/fractales-belllos-y-sin-embargo-utiles/>.
- [12] Aprendemos Matemáticas. *Conjuntos de Julia y Mandelbrot*. Maig de 2024. URL: <https://www3.gobiernodecanarias.org/medusa/ecoblog/mrodrperv/fractales/conjuntos-de-julia-y-mandelbrot/>.
- [13] Mathigon.org. *Fractal Course*. Jul. de 2024. URL: <https://es.mathigon.org/course/fractals/introduction>.
- [14] Fractal Antenna Systems. *Fractal Antenna Systems - Our technology*. Ag. de 2024. URL: <https://www.fractenna.com/our/our.html>.
- [15] Ministerio para la Transición Ecológica y el Reto Demográfico. *Exposiciones del ceneam*. Jul. de 2024. URL: https://www.miteco.gob.es/es/ceneam/exposiciones-del-ceneam/exposiciones-itinerantes/copia_de_emergencia_climatica-2020.html.
- [16] Enric Lopez Uestes. *Curva de Koch*. Jul. de 2024. URL: <https://www.enriclopezruestes.cat/es/curva-de-Koch/>.
- [17] Enric Lopez Uestes. *Curva del dragon*. Jul. de 2024. URL: <https://www.enriclopezruestes.cat/es/curva-del-drag%C3%B3n/>.
- [18] Viquipèdia. *Fractals per dimensió de Hausdorff*. Jul. de 2024. URL: https://ca.wikipedia.org/wiki/Fractals_per_dimensi%C3%B3_de_Hausdorff.
- [19] Wikipedia. *Benoît Mandelbrot*. Gen. de 2024. URL: https://es.wikipedia.org/wiki/Beno%C3%A9t_Mandelbrot.