Pràctica de Càlcul Numèric

Conques de convergència d'arrels de polinomis imaginaris i representació de fractals en gnuplot

Grau: Matemàtica Computacional i Analítica de Dades

Assignatura: Càlcul Numèric

NIU de l'estudiant: 1600959

Nom de l'estudiant: Pau Blasco Roca

Índex

1. Abstract

Aquesta pràctica estudia la generació i representació de fractals a partir d'aplicar el mètode de convergència de newton a un espai d'$\mathbb{R}^2$ que conté arrels de polinomis imaginaris. Després d'escriure vàries rutines en C per avaluar polinomis i les seves derivades, he escrit un codi que permet pintar de diferents colors les conques d'atracció a les arrels pertanyents a la regió d'$\mathbb{R}^2$ a estudiar.

Al llarg de la memòria estudiarem la implementació del codi juntament amb com hem anat avaluant i recalibrant el seu funcionament. Finalment presentarem un manual d'execució per a que l'usuari pugui replicar les experiències descrites; i avaluarem els resultats i els fractals obtinguts.

2. Objectius

Se'ns demana construir diferents rutines de càlcul per a estudiar alguns fractals de forma senzilla. Una empresa de disseny, amb els seus propis entorns d'execució i de representació de la sortida de les dades, ens ho ha encarregat. A partir del Mètode de Convergència de Newton, haurem d'estudiar una regió del pla d'$\mathbb{R}^2$ que rebrem desde la terminal (informació passada per argument) i mirar a on convergeix cada punt. Així generarem un fitxer de text que després representarem amb el gnuplot.

3. Introducció teòrica

3.1. Mètode de Newton

En càlcul numèric, el mètode de Newton, o mètode de Newton-Raphson, és un algorisme per tal de trobar aproximacions del zero d'una funció amb valors reals.[1] Cal notar, però, que el Mètode de Newton és un mètode obert, és a dir, que no ens assegura l'èxit. Podria passar que un punt en concret no tendís cap a cap arrel. Per això, voldrem limitar les iteracions màximes que fa el nostre programa per a cada punt. El mètode de Newton funciona de la següent manera:

Sigui f contínua i dues vegades diferenciable en $[a, b]$, i que una de les arrels de la funció està continguda en aquest interval. Direm que $\alpha \in [a, b]$ és solució si $f(\alpha) = 0$. Ara prenem un altre punt, $x_0 \in [a, b]$, tal que $f'(x_0) \neq 0$. Si considerem ara la expansió del Polinomi de Taylor en aquest punt, tenim que:

$$f(\alpha) = 0 \approx f(x_0) + (\alpha - x_0)f'(x_0)$$

Ja que hem ignorat tots els termes a partir del segon. Així doncs, aillant α de l'expressió anterior, obtenim:

$$\alpha \approx x_0 - \frac{f(x_0)}{f'(x_0)} = x_1$$

Així, x_1 hauria de ser una millor aproximació a α que x_0. Si iterem aquest mètode n vegades, arribarem gairebé a tocar de l'arrel que es trobava en l'interval.

3.2. Conques d'atracció

Definim les conques d'atracció com les regions de l'espai les quals els seus punts tendeixen, iterant el Mètode de Newton, a una arrel concreta. Així, pintarem del mateix color totes les conques d'atracció que convergeixin a α_0, d'un altre diferent les d'α_1, d'un altre les d'α_2... i així amb cada arrel del polinomi.

[1] https://ca.wikipedia.org/wiki/Mètode_de_Newton

3.3. Arrels de la unitat, interpretació analítica i geomètrica

Sobre el pla dels polars, podem representar els nombres imaginaris fent servir la següent notació:

$$\forall z = a + bi \in \mathbb{C}, \quad z = r_\alpha$$

$$\text{on} \quad r = \sqrt{a^2 + b^2}, \quad \alpha = \arctan \frac{b}{a}$$

Cal fer notar algunes propietats d'utilitzar aquest sistema de coordenades. En concret, observem les propietats del producte (en aquest cas, d'elevar un nombre a n):

$$z^n = (r_\alpha)^n = r^n_{\alpha \cdot n}$$

Tenint aquesta propietat en ment, suposem que volem resoldre una equació en $\mathbb{C}$ del tipus següent:

$$z^n - 1 = 0 \quad \forall z \in \mathbb{C}, \quad \forall n \in \mathbb{N}$$

Les solucions d'aquesta equació, o les arrels del polinomi $P(z) = z^n - 1$ es coneixen com les *arrels de la unitat*, i per la definició d'arrel d'un polinomi, tenen la propietat que verifiquen l'anterior igualtat. Ara bé, què passa si estudiem la mateixa eqüació fent servir notació polar?

$$r^n_{\alpha \cdot n} - 1_{0^o} = 0 \iff r^n_{\alpha \cdot n} = 1_{0^o}$$

En aquest cas hem vist que treballem amb la unitat real en coordenades polars, també. El seu argument serà zero, doncs està sobre l'eix real i té valor positiu. Notem que $r = 1$, per força. Ara només hem d'estudiar els α que verifiquin el següent:

$$\alpha \cdot n = 0^{\text{o}} \implies \alpha \cdot n = 360^{\text{o}} \implies \alpha = \frac{360^{\text{o}}}{n}$$

I així ja hem vist que les arrels del polinomi $P(z) = z^n - 1$ (arrels n-èssimes de la unitat) seran:

$$\alpha_0 = 0 \cdot \frac{360^{\text{o}}}{n}, \quad \alpha_1 = 1 \cdot \frac{360^{\text{o}}}{n}, \quad \alpha_2 = 2 \cdot \frac{360^{\text{o}}}{n}, \quad \ldots, \quad \alpha_{n-1} = (n-1) \cdot \frac{360^{\text{o}}}{n}$$

Les següents ja tornarien a ser les mateixes un altre cop, per tant estarien repetides.

Observem ara la part geomètrica. Aquestes arrels quedaran distribuides de forma uniforme sobre en el cercle unitari de manera que, per exemple, si unim els punts de les arrels cinquenes de la unitat obtindrem un pentàgon regular; si unim els de les arrels setenes un heptàgon regular...

4. Implementació de les rutines

4.1. avalp()

En aquesta primera funció havíem d'avaluar el polinomi sobre $\mathbb{C}$. La dificultat afegida (encara que més endevant, potser avantatge) ha estat que el polinomi se'ns donava fragmentat en monomis que contenien les seves arrels. Per exemple, $x^2 - 1 = 0 \implies (x-1)(x+1) = 0$. A més a més, no podíem treballar amb aritmètica complexa, si no que havíem de separar per components i anar desenvolupant binomis. La meva idea ha estat la següent:

$$P(z) = \prod_{k=0}^{n-1} (z - w_k) \tag{1}$$

On w_k seria la k-èssima arrel, i n el nombre d'arrels. En codi, això es tradueix a un petit bucle for que primer multiplica el primer pel segon element, després agafa el resultat i el multiplica pel tercer, i així successivament. Hem d'anar en compte, però, amb no sobreescriure cap variable mentre encara la necessitem. Per això he fet servir variables auxiliars durant el bucle.

4.2. avaldp()

En aquesta rutina havíem d'avaluar la derivada del polinomi en qüestió. Seguint la mateixa dinàmica que abans i amb les mateixes premises que ens impedeixen fer servir aritmètica complexa, ja se'ns proporcionava una fórmula matemàtica per a avaluar P:

$$P'(z) = \sum_{j=0}^{n-1} \prod_{\substack{k=0 \\ k \neq j}}^{n-1} (z - w_k) \tag{2}$$

De la mateixa manera que abans, he fet servir un for per al productori i a més a més un altre for per al sumatori, de manera que anava sumant el resultat de cada iteració sobre dues variables (part real i part imaginària).

4.3. cnvnwt()

En aquesta part del codi havia de fer servir les altres dues rutines per a replicar la fórmula descrita a **3.1**. He situat aquest fragment de codi dins d'un while, que faig parar quan la distància del punt a l'arrel és menor a la tolerància (variable passada per argument) o bé quan passa un cert nombre d'iteracions (també passat per argument). Aquesta rutina retorna un *int*, que equival al nombre de l'arrel a la que s'ha convergit.

5. Manual d'utilització

Primerament, hem de fer servir el fitxer *Makefile* en cas que els arxius no vinguin compilats, per a generar un executable que poguem cridar i executar desde, per exemple, interfícies com *mingw*.

Suposarem que estem treballant en una interfície Linux, com la que se'ns descriu a l'enunciat de la pràctica. Obrint la terminal i movent-nos al directori on tinguem els fitxers de codi, haurem d'executar la següent comanda:

make dibfr

Això ens generarà l'executable $dibfr$. Aquest l'haurem de cridar amb els següents arguments:

dibfr narr xmn xmx nx ymn ymx ny tolconv maxit >fractaln.txt

On $narr$ serà un int representant el nombre d'arrels, xmn i xmx seran el mínim i el màxim de l'eix horitzontal representat, nx les subdivisions de l'interval (la "qualitat de la foto"). El mateix per l'eix vertical amb ymn, ymx i ny. Finalment tenim $tolconv$, que és la diferència mínima entre l'aproximació i l'arrel per a considerar que hi ha convergència; i $maxit$, el nombre màxim d'iteracions que farà el nostre programa mentre no convergeixi (i així evitar bucles infinits). El nom del fitxer al qual redirigim la sortida és arbitrari, per tant podem posar el que nosaltres volguem.

Amb aquesta comanda, el programa ens deixarà introduir les arrels. Les haurem de posar a la terminal de la següent manera:

componentX componentY valorR valorG valorB

I en acabar la linia, premem intro. Haurem d'introduir tantes línies com el $narr$ que li haguem passat al programa. És imporant que les components RGB les donem normalitzades. Un cop fet això, el codi processarà i farà càlculs durant una estona. Quan acabi, tindrem un nou fitxer en el nostre directori que consistirà en una llista de coordenades i valors de color.

Anem a la terminal de **gnuplot**, ens movem al directori on tenim el fitxer txt i executem la següent comanda:

plot 'fractaln.txt' using 1:2:3:4:5 w rgbimage

I si tot va bé, ens hauria d'aparèixer el nostre fractal per pantalla!

6. Resultats finals, exemples gràfics

Per a *testejar* el codi i que funcionés de forma correcta, he fet algunes proves:

Figura 1: $z^3 - 1 = 0$

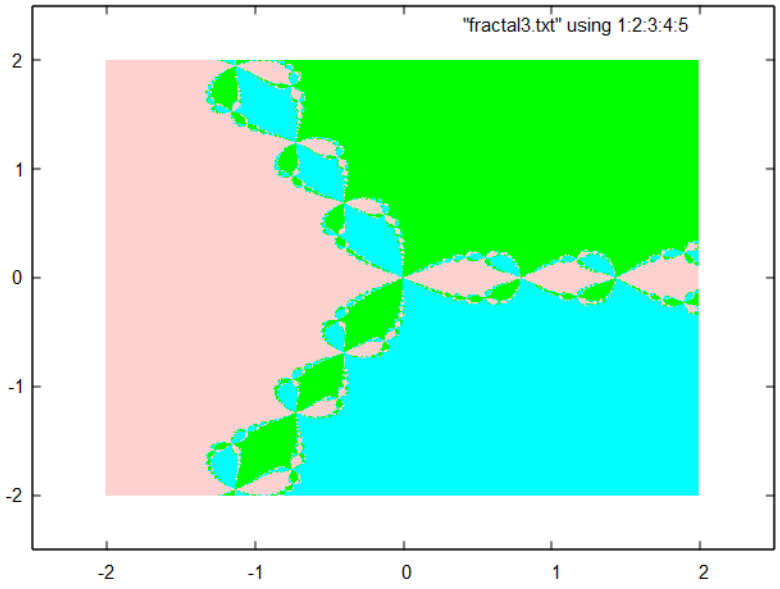


Figura 2: $z^4 - 1 = 0$

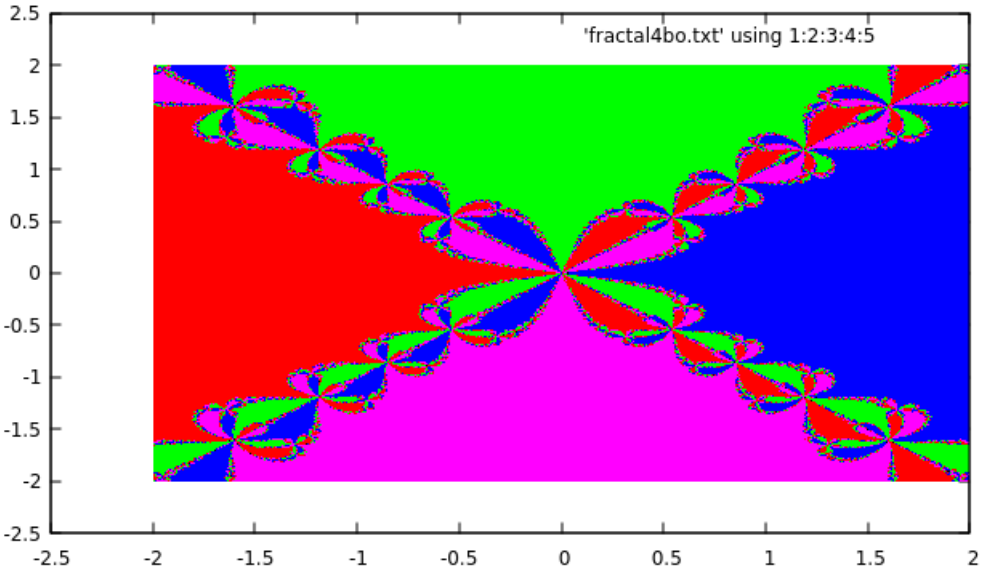


Figura 3: $z^5 - 1 = 0$

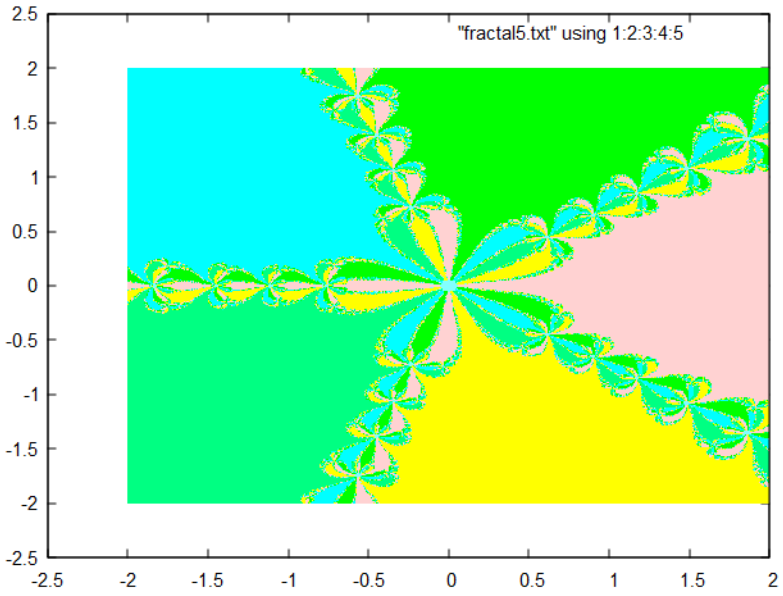


Figura 4: $z^6 - 1 = 0$

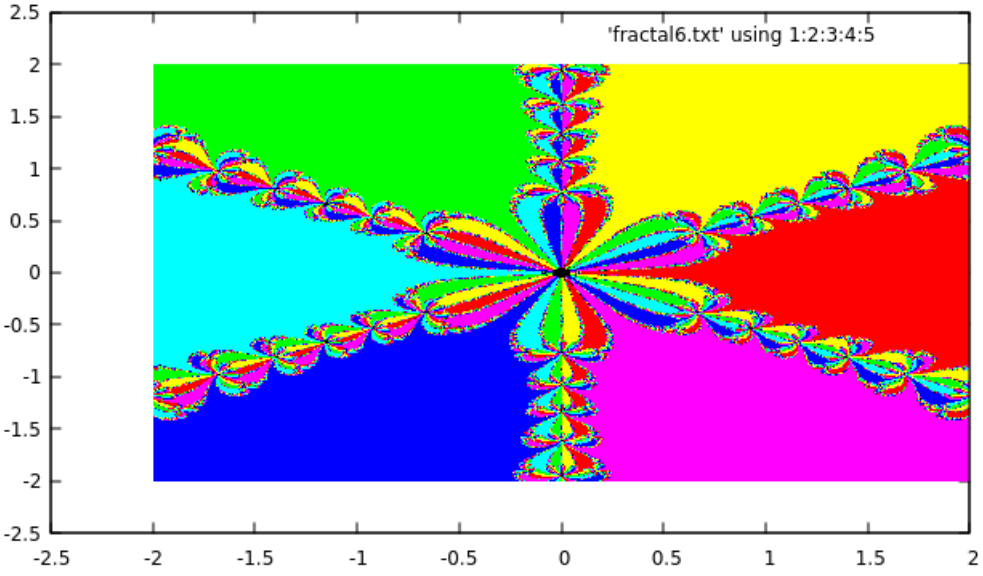


Figura 5: $\cosh z - \frac{1}{2} = 0$

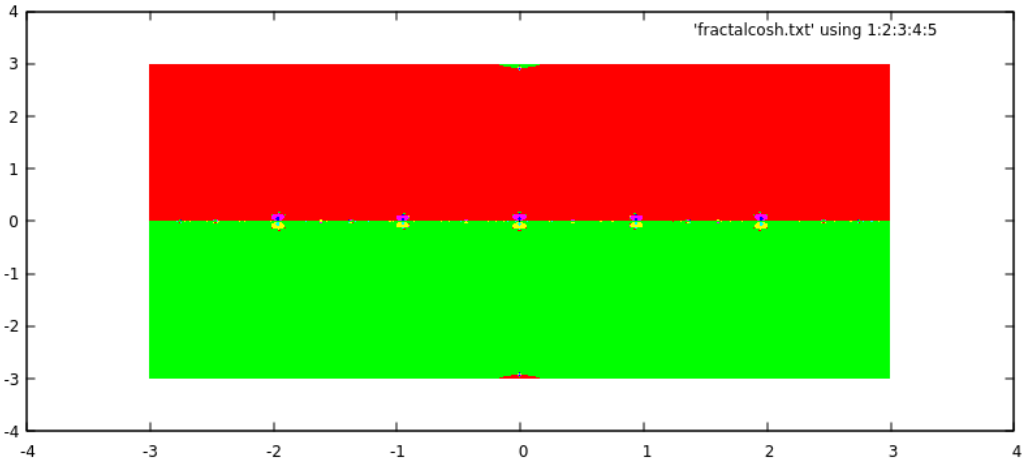
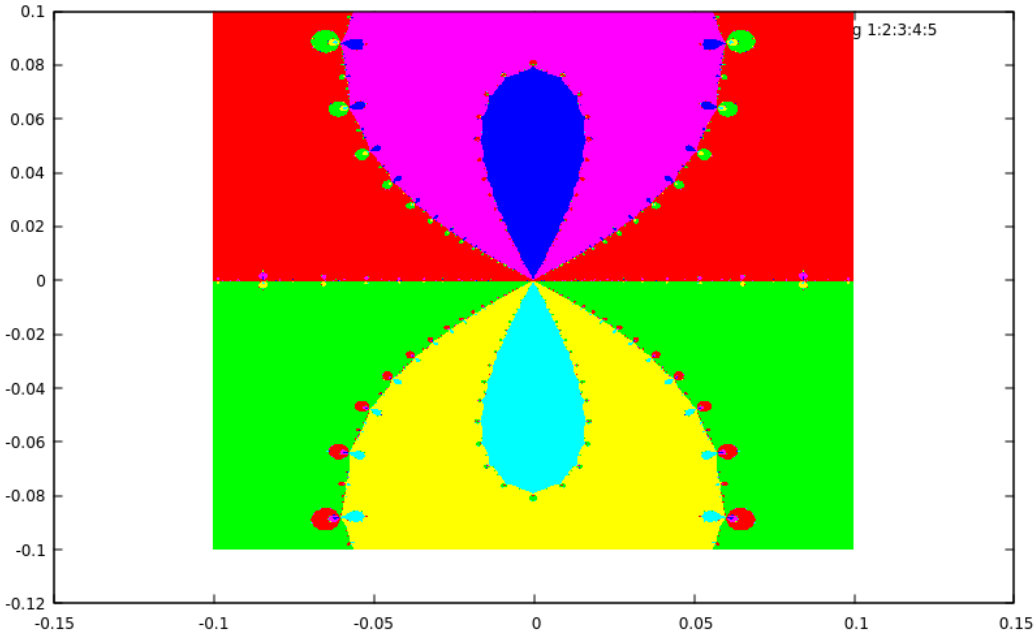


Figura 6: $\cosh z - \frac{1}{2} = 0$, ampliat per a veure què passa



7. Autoavaluació

Personalment, crec que el meu interès per la pràctica va començar bastant alt, però va decaure al trobar-me amb un problema bastant crític: el meu ordinador no pot fer servir gnuplot. Per sort, vaig poder trobar una alternativa i al veure els fractals vaig tornar a centrar-me i a interessar-me pel projecte. Per què no podia fer servir gnuplot? Perque, al crear el meu usuari, vaig deixar un espai al nom. Hi ha aplicacions que funcionen encara que hi hagi un espai al PATH, però gnuplot no és una d'elles; així com IntelliJ (que em funciona parcialment) o PyCharm (que no troba el PATH cap al python). He pogut treballar, per sort, amb una màquina virtual, on sí que tenia els path correctes i gnuplot sí que em trobava els fitxers.

Això no ha presentat cap problema per als resultats finals de la pràctica, però sí que ha estat un obstacle personal per mi que em va frustrar moltíssim fins que no vaig entendre perquè passava.

També he de dir que la idea de treballar amb C no em va semblar (ni continua fent-ho) massa atractiva, al ja haver treballat amb altres llenguatges més moderns com C#, C++ o Java / JavaScript. Però al veure els resultats i tot el potencial que tenen aquestes aplicacions més «antigues», m'he sentit satisfet amb els resultats.

8. Conclusió

Crec que els resultats obtinguts han estat els que s'esperaven. He pogut treballar amb comoditat amb el codi i adaptar-lo al meu propi estil de programació, de manera que quedi tot net i endreçat i no com un mix de peces de foros i pàgines diferents. A més a més, crec que he optimitzat al màxim possible l'espai ocupat pel programa, fent servir mallocs dinàmics i ajustats a les necessitats de cada exeució.

Pel que fa als resultats gràfics, seria genial aconseguir imatges en 4K o inclús molt més grans, però he comprovat de primera mà que el temps de computació s'allargaria molt més. Vaig provar a crear-ne de 540x720 i de 1080x1440; però aquestes últimes trigaven ja molt més (el quàdruple) a generar-se.

Personalment aquest projecte m'ha acabat motivant molt més del que em pensava un cop he pogut fer servir gnuplot des de la màquina virtual; fins al punt que no descarto tornar-hi en unes setmanes i intentar generar fractals amb més qualitat o més estrambòtics.