

# Pràctica 1: Fractals generats pel mètode de Newton

Gerard Lahuerta Martín

30 d'abril del 2021

# Índex

<b>1</b>	<b>Introducció al treball</b>	<b>3</b>
1.1	Motivació del treball . . . . .	3
1.2	Metodologia . . . . .	3
1.2.1	Explicació de la metodologia . . . . .	3
1.2.2	Comentaris . . . . .	4
1.2.3	Informació adicional d'interés . . . . .	4
<b>2</b>	<b>Explicació de la biblioteca</b>	<b>5</b>
2.1	Funcionament . . . . .	5
2.1.1	Càlcul dels fractals . . . . .	5
2.1.2	Organització . . . . .	6
2.1.3	Biblioteca nwtfr.c . . . . .	7
2.1.4	test_avalp i el funcionament correcte de la biblioteca . . . . .	11
2.1.5	Programa dibfr.c i executable dibfr . . . . .	12

# Capítol 1

## Introducció al treball

### 1.1 Motivació del treball

Aquest treball ha estat motivat per l'empresa de disseny al precisar un conjunt de rutines que generin fractals mitjançant el mètode de Newton. Per tal de donar alguns casos de referència també s'ha procedit a programar una utilitat mitjançant el programa gnuplot i la línia d'ordres d'Unix.

### 1.2 Metodologia

#### 1.2.1 Explicació de la metodologia

La metodologia utilitzada alhora de programar el codi es la següent:

1. Llenguatge de programació ràpid i eficient

Utilitzem el llenguatge C gràcies a la rapidesa en els càlculs i l'alta eficiència.

2. Fàcil d'executar

La programació ha estat orientada a la fàcil utilització d'aquest per tal de ser intuïtiva, a més, s'afegeix el manual d'ús del software per les consultes necessàries.

3. Fàcil modificació

El programa està forçament comentat i enfocat per a ser fàcilment intuïtiu per si calgués alguna modificació a posteriori i, així, facilitar la tasca.

### 1.2.2 Comentaris

El disseny de les rutines ha estat pensat i implementat per a evitar els càlculs i interaccions innecessàries. També, s'ha evitat la creació de rutines i variables redundants per a obtenir la millor optimització possible.

Tal com hem comentat anteriorment, per tal d'evitar confusions i ajudar a l'enteniment de les accions del programa s'han implementat línies de codi comentat, aquestes expliquen els conceptes més ambigus del programa i estan localitzades al costat de cada línia on s'ha cregut necessària una explicació.

### 1.2.3 Informació adicional d'interés

La biblioteca ha estat programada amb l'editor d'ús públic Notepad++, encara que pot ser modificat en altres editors com podrien ser l'editor de notes o d'altres més complexos com VisualStudio.

Durant la programació de les rutines s'ha ideat un programa per testejar el correcte funcionament de la biblioteca i així en cas de possibles errors poder comprovar el correcte funcionament d'aquesta.

L'ús d'aquest test que està anomenat com *test\_avalp* és fàcilment executable i és explicat més a fons a la Subsecció 2.1.4 *test\_avalp* i el funcionament correcte de la biblioteca

# Capítol 2

## Explicació de la biblioteca

En aquest capítol parlarem a fons sobre l'estructura, els materials que implementa, el funcionament, les estratègies utilitzades i el correcte ús i execució del programa.

### 2.1 Funcionament

La biblioteca està pensada per a ser implementada en els entorns d'execució de l'empresa de disseny i no per a substituir els mateixos.

Els mètodes que s'inclouen en la biblioteca (dels quals parlem a continuació) poden ser utilitzats independentment i no precisa cap altra eina externa més enllà de la llibreria *math.h* del conjunt de llibreries del llenguatge C.

#### 2.1.1 Càlcul dels fractals

El mètode matemàtic que hem utilitzat per a saber de manera aproximada les òrbites dels punts que generen els fractals és el mètode de Newton de manera iterativa.

A partir d'un punt  $z_0 = a + bi \in \mathbb{C}$ , amb  $a, b \in \mathbb{R}$  (coordenades de l'eix x, y; coordenades del píxel de la pantalla), calculem la seva òrbita després de  $K \in \mathbb{N}$  iteracions de la següent manera:

$$z_1 = z_0 - \frac{p(z_0)}{p'(z_0)} \longrightarrow z_2 = z_1 - \frac{p(z_1)}{p'(z_1)} \longrightarrow z_3 = z_2 - \frac{p(z_2)}{p'(z_2)} \longrightarrow \cdots \longrightarrow z_k = z_{k-1} - \frac{p(z_{k-1})}{p'(z_{k-1})}$$
$$z_j = z_{j-1} - \frac{p(z_{j-1})}{p'(z_{j-1})}, \forall j \in \{1, \dots, K\}$$

Essent  $p(z) \in \mathbb{C}_t[Z]$  expressat de forma factoritzada; on  $z \in \mathbb{C}$ ,  $t \in \mathbb{N}$  el nombre d'arrels.

### 2.1.2 Organització

La biblioteca *nwtfr.h* ha estat organitzada de manera que totes les funcions han estat programades en un sol fitxer per així facilitar la implementació de la biblioteca en el software de l'empresa.

Per a fer els càlculs mitjançant el mètode de Newton hem dividit el treball en tres funcions que depenen entre elles: la funció *avalp* (avalua el polinomi), la funció *avaldp* (avalua la derivada del polinomi) i la funció *cnvnwt* (calcula l'arrel del polinomi a la que s'aproxima  $z_0$  mitjançant el mètode de newton). Explicuem aquestes tres funcions amb més profunditat en els següents subapartats.

La dependència de les funcions *avalp*, *avaldp* i *cnvnwt* és piramidal, de manera que les funcions deleguin càlculs en altres funcions per a optimitzar i organitzar millor el codi.

La funció per a dur a terme tots els càlculs del fractal és l'anomenada *cnvnwt* i que crida a les funcions *avalp* i *avaldp* que analitzaran el polinomi que genera el fractal i la seva derivada.

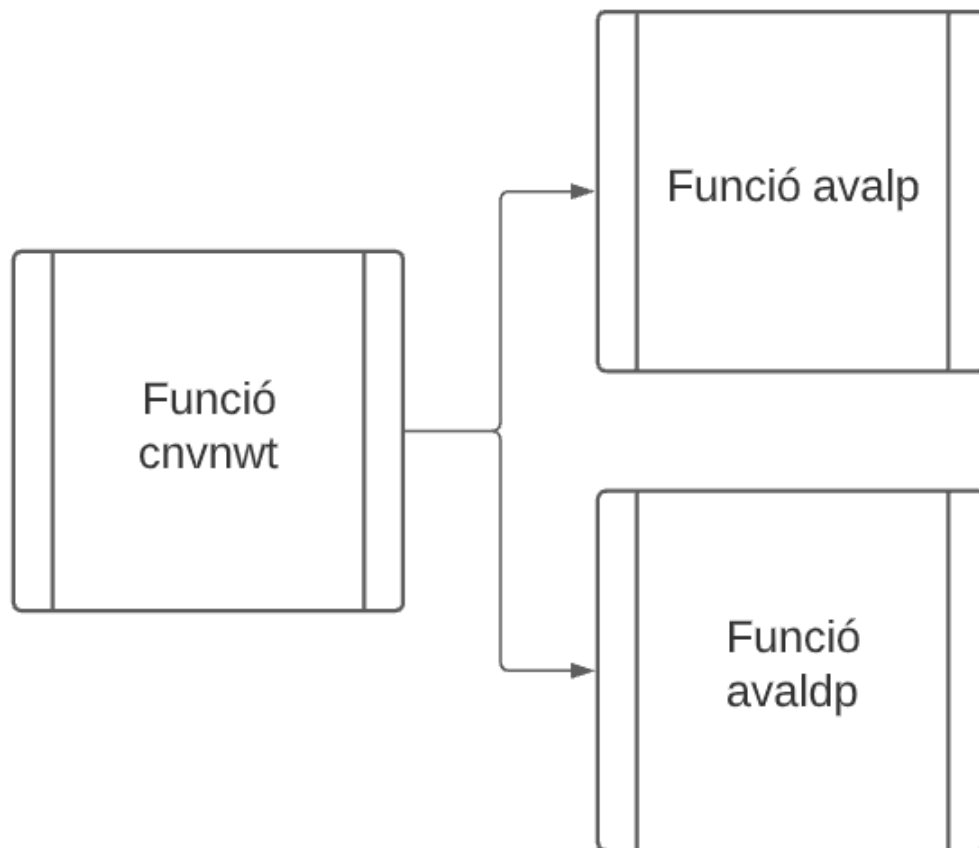


Diagrama de flux de les dependències entre les funcions de la biblioteca

### 2.1.3 Biblioteca nwtfr.c

Explicuem a continuació el funcionament i composició dels mètodes i com compilar-los.

#### Mètode avalp

El mètode avalp és l'encarregat de calcular el valor  $p(z_0)$  a partir dels valors  $a, b \in \mathbb{R}/z_0 = a + bi$  la llavor,  $\{w_1, \dots, w_n \mid w_q = \alpha + \beta i \ \forall q \in \{1, \dots, n\}, \alpha, \beta \in \mathbb{R}\}$  les arrels de  $p(z)$  essent  $n \in \mathbb{N}$  el nombre d'arrels del polinomi.

Per fer l'avaluació, els arguments que es passen al mètode de manera consecutiva són els següents:

- Variables double  $x$  i  $y$ : són els valors de  $z \in \mathbb{C}$  que s'introdueixen per a calcular el valor de  $p(z)$ ; és a dir,  $z = x + y \cdot i$  amb  $x, y \in \mathbb{R}$ .
- Apuntadors de tipus double  $px$  i  $py$ : són les adreces de memòria on es guardaran el valor de  $p(z)$ ; és a dir,  $p(z) = px + py \cdot i$ .
- Variable integer  $n$ : és el nombre d'arrels del polinomi  $p(z)$ ; indica, per tant, la llargada de les llistes que contenen els valors de les arrels del polinomi.
- Llistes del tipus double  $u$  i  $v$ : són les llistes que contenen totes les arrels del polinomi  $p(z)$  de manera que:

$$p(z) = \prod_{q=1}^n (z - w_q)$$

Amb  $w_q = \alpha_q + \beta_q i \ \forall q \in \{1, \dots, n\}$ ,  $\alpha_q, \beta_q \in \mathbb{R}$  de manera que els valors de  $\alpha_q$  és guardada a la posició  $q$ -èssima de la llista  $u$  i  $\beta_q$  és guardada a la posició  $q$ -èssima de la llista  $v$ .

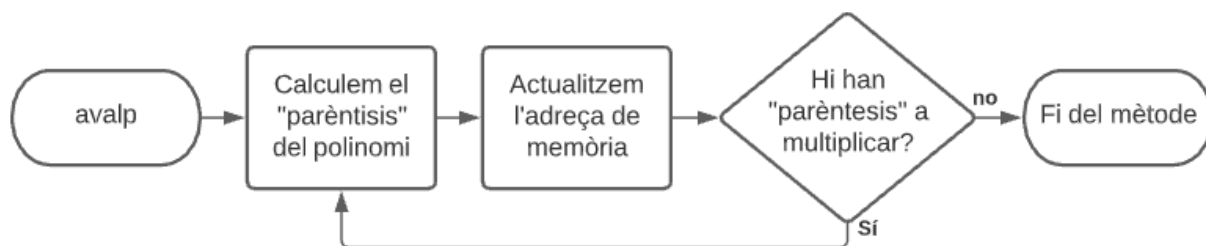


Diagrama de flux del mètode *avalp*

## Mètode avaldp

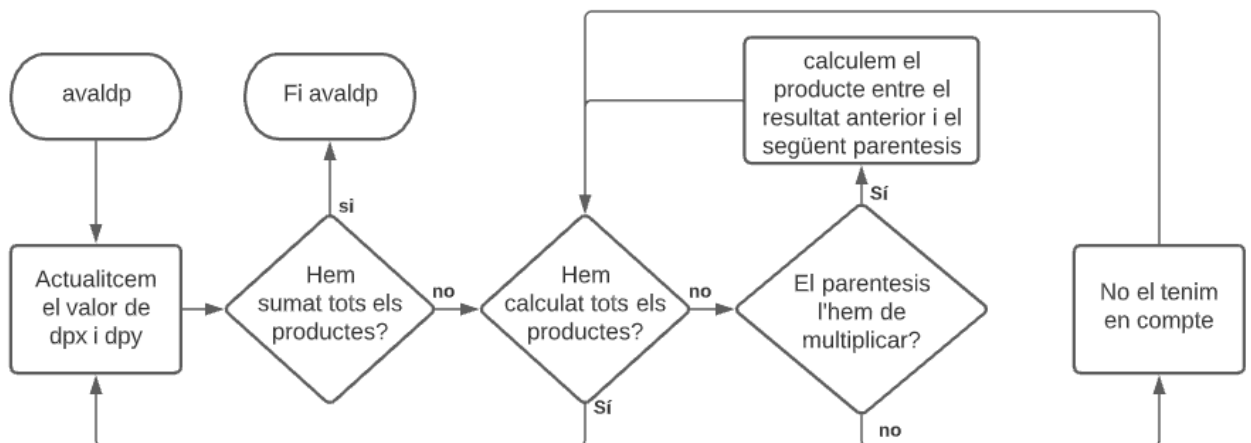
El mètode *aval<sub>dp</sub>* és l'encarregat de calcular el valor  $p'(z_0)$  a partir dels mateixos valors que l'hi són introduïts a la funció *aval<sub>p</sub>* pel que els donarem per definits (tornar al punt 2.1.3 Mètode *aval<sub>p</sub>* per veure els valors utilitzats).

Per fer l'avaluació, els arguments que es passen són similars al del mètode *avalp*, per tant de forma consecutiva són els següents:

- Variables double  $x$  i  $y$ : són els valors de  $z \in \mathbb{C}$  que s'introdueixen per a calcular el valor de  $p'(z)$ ; és a dir,  $z = x + y \cdot i$  amb  $x, y \in \mathbb{R}$ .
- Apuntadors de tipus double  $dpx$  i  $dpy$ : són les adreces de memòria on és guardaran el valor de  $p'(z)$ ; és a dir,  $p(z) = dpx + dpy \cdot i$ .
- Variable integer  $n$ : és el nombre d'arrels del polinomi  $p(z)$ ; indica, per tant, la llargada de les llistes que contenen els valors de les arrels del polinomi.
- Llistes del tipus double  $u$  i  $v$ : són les llistes que contenen totes les arrels del polinomi  $p(z)$  de manera que:

$$p'(z) = \sum_{k=0}^{n-1} \prod_{\substack{q=0 \\ q \neq k}}^{n-1} (z - w_q)$$

Amb  $w_q$  definida de la mateixa manera que a la funció *avalp*.

Diagrama de flux del mètode *aval* $\delta p$



## Mètode *cnvnwt*

El mètode *cnvnwt* és l'encarregat de calcular el valor de la posició de la llista que conté l'arrel del polinomi  $p(z)$  que la llavor  $z_0 = x + y \cdot i$  s'aproxima amb una certa tolerància, *tolcnv*, en un màxim d'iteracions establertes, *maxit*. Per això utilitza els mètodes abans explicats de *avalp* i *avaldp*. Els arguments necessaris de *cnvnwt*, consecutivament, són:

- Variables double  $x$  i  $y$ : són els valors de  $z_0 = x + y \cdot i \in \mathbb{C}$  que s'introdueixen per a calcular la seva òrbita, amb  $x, y \in \mathbb{R}$ .
- Variable double *tolcnv*: és la distància mínima que ha d'estar un punt de l'òrbita de  $z_0$  per ser considerada com a valor zero i retornar el valor de la posició d'aquesta arrel.
- Variable integer *maxit*: és el valor màxim d'iteracions que durà a terme el mètode *cnvnwt* per tal de calcular a quina arrel s'aproxima l'òrbita de  $z_0$ .
- Variable integer  $n$ : és el nombre d'arrels del polinomi  $p(z)$ ; indica, per tant, la llargada de les llistes que contenen els valors de les arrels del polinomi.
- Llistes del tipus double  $u$  i  $v$ : són les llistes que contenen totes les arrels del polinomi  $p(z)$  que s'utilitzaran per a calcular la distància entre els punts de l'òrbita i les arrels de  $p(z)$ , calcular el valor de  $p(z)$  amb el mètode *avalp* i calcular  $p'(z)$  amb *avaldp*.

Comentar també que en cas que algun  $z_0$  no s'aproximés a cap arrel, aquesta funció retornarà el valor  $-1$  per a indicar aquest fenomen.

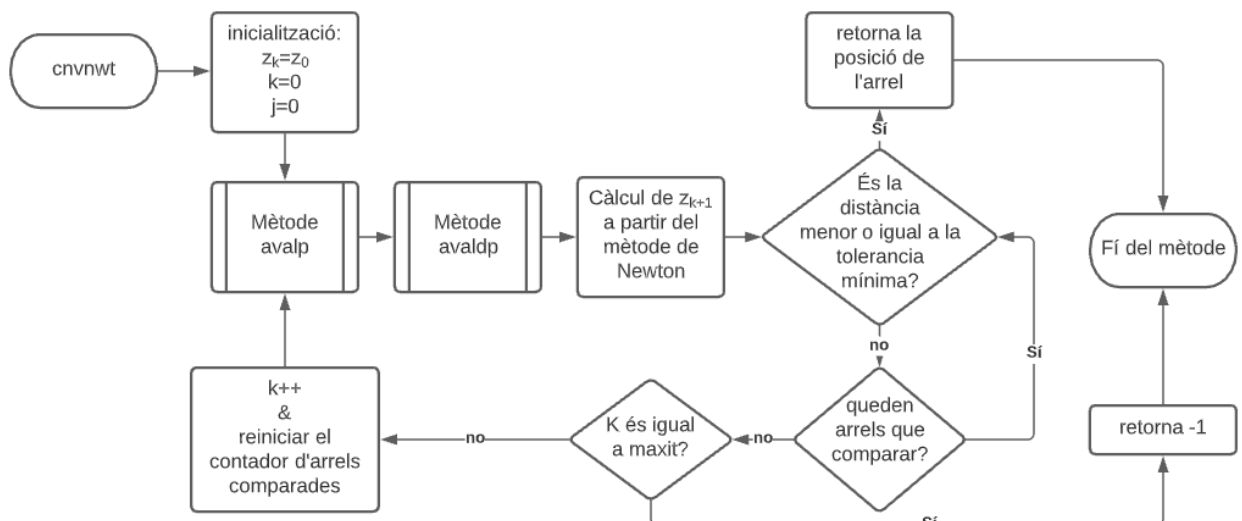


Diagrama de flux del mètode *cnvnwt*

## Compilació de la biblioteca i document Makefile

Per a poder organitzar de millor manera la compilació de la biblioteca, així com la seva implementació al programa *dibfr.c* (que és el programa encarregat de fer els càlculs necessaris per a representar els fractals i que en la següent secció expliquem en profunditat); hem creat un arxiu que s'encarrega de la seva compilació. Aquest arxiu porta el nom *Makefile* seguit d'una barra baixa i el nom del sistema operatiu pel qual ha estat pensat.

A l'empresa només l'interessa utilitzar el fitxer *Makefile\_Unix.txt*, però igualment s'ha volgut crear una còpia modificada del fitxer per si es precisa fer modificacions del programa amb un software o plataforma de Windows. Si és aquest el cas, utilitzar l'arxiu *Makefile\_Windows.txt*. L'execució dels arxius Makefile, encarregats de la compilació de l'executable encarregat de fer el fitxer amb la informació per després dibuixar els fractals amb el *gnuplot*, és senzilla.

- **Si es tracta del sistema Operatiu Unix**, en la consola/terminal de l'ordinador executar la comanda *make -f Makefile\_Unix.txt* per crear l'executable *dibfr*.

Anàlogament, executar la comanda *clean -f Makefile\_Unix.txt* per netejar d'arxius objecte innecessaris el directori de treball o *realclean -f Makefile\_Unix.txt* per esborrar tot el treball fet per l'arxiu Makefile.

- **Si es tracta del sistema Operatiu Windows**, en la consola/terminal de *mingw-w64* executar la comanda *mingw32-make -f Makefile\_Windows.txt* per crear l'executable *dibfr.exe*.

Anàlogament, executar la comanda *minw32-make clean -f Makefile\_Windows.txt* per netejar d'arxius objecte innecessaris el directori de treball o *minw32-make realclean -f Makefile\_Unix.txt* per esborrar tot el treball fet per l'arxiu Makefile.

Esmentar que sobre l'arxiu del codi del programa *dibfr.c* es comentarà més endavant a la Secció 2.1.5 Programa *dibfr.c* i executable *dibfr*, on es parlarà més a fons sobre el codi que s'utilitza, la implementació de la biblioteca en el codi i el seu ús en *gnuplot* en la representació de fractals.

### 2.1.4 test\_avalp i el funcionament correcte de la biblioteca

Explicuem a continuació el programa *test\_avalp* i la correcta implementació de la biblioteca en els codis de programació de l'empresa així com en el mateix codi.

#### Implementació i correcta utilització de la biblioteca *nwtfr.h*

Per a implementar a biblioteca *nwtfr.h* en qualsevol codi en C cal a l'inici del codi escriure la comanda següent:

***#include "nwtfr.h"***

Una vegada escrita la comanda el programa implementarà els mètodes de l'extensió al codi. Si no s'utilitza l'arxiu Makefile, explicat en el passat apartat, es pot compilar els arxius que continguin aquesta biblioteca amb la comanda següent al terminal/console:

***gcc -o nom\_del\_executable -Wall nom\_del\_programa.c nwtfr.c***

#### Programa i executable *test\_avalp*

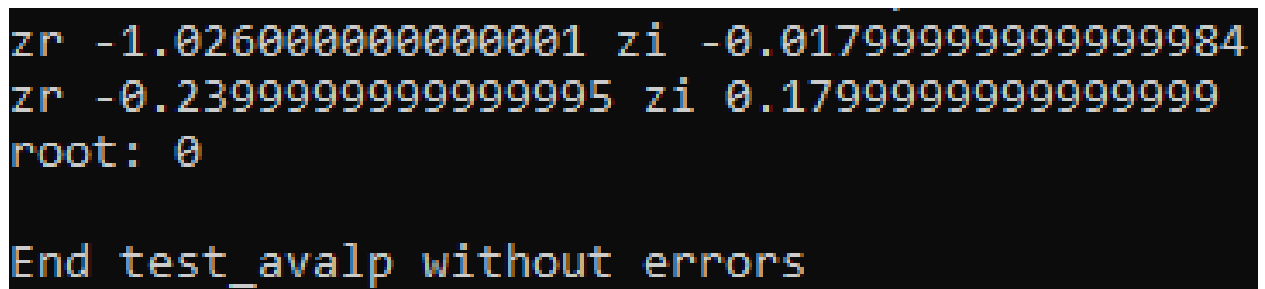
Per tal d'assegurar el funcionament correcte de la biblioteca hem posat a disposició de l'empresa el codi *test\_avalp* que té com a objectiu assegurar que els mètodes que actualment conté la biblioteca funcionin de manera correcta.

El programa *test\_avalp* és molt senzill, únicament crida les tres funcions que conté *nwtfr.h* per fer uns càlculs de prova i eventualment mostrar-los per pantalla.

Per obtenir l'executable utilitzem la comanda abans explicada,

***gcc -o test\_avalp -Wall test\_avalp.c nwtfr.c***

Una vegada obtingut l'executable *test\_avalp* (o *test\_avalp.exe* si es tracta d'un dispositiu Windows) cal executar-lo per consola.



```
zr -1.0260000000000001 zi -0.017999999999999984
zr -0.23999999999999995 zi 0.17999999999999999
root: 0

End test_avalp without errors
```

Missatge de sortida en executar *test\_avalp* si la biblioteca funciona correctament

## 2.1.5 Programa *dibfr.c* i executable *dibfr*

En aquesta secció explicarem el funcionament del programa *dibfr*, el seu funcionament intern i les seves dependències.

També comentarem com utilitzar la informació obtinguda per a representar els fractals per pantalla mitjançant el programa d'accés lliure i gratuït de gnuplot.

### Dependències i necessitats del programa

El programa *dibfr* depèn íntegrament de la biblioteca *nwtfr.h*. Concretament utilitza la funció *cnvnt* que calcula l'arrel a la qual l'òrbita d'un  $z_0$  l'anul·la.

El programa per ser executat requereix certs arguments que s'han d'acompanyar a l'hora d'executar; és a dir, s'ha d'introduir al terminal de l'Unix la comanda:

*./dibfr narr xmn xmx nx ymn ymx ny tolcnv maxit*

(En cas de ser utilitzada en un terminal de Windows, eliminar el *./* a l'inici de la comanda)

Aquestes variables que hem anomenat a posteriori de l'executable són:

- ▶ **variable *narr***: enter que determina el nombre d'arrels de  $p(z)$ .
- ▶ **variable *xmn***: valor mínim que pot tenir la part real de la llavor  $z_0$ .
- ▶ **variable *xmx***: valor màxim que pot tenir la part real de la llavor  $z_0$ .
- ▶ **variable *nx***: nombre d'interval·ls a dividir el segment  $\overline{xmn\ xmx}$ .
- ▶ **variable *ymn***: valor mínim que pot tenir la part imaginària de la llavor  $z_0$ .
- ▶ **variable *ymx***: valor màxim que pot tenir la part imaginària de la llavor  $z_0$ .
- ▶ **variable *ny***: nombre d'interval·ls a dividir el segment  $\overline{ymn\ ymx}$ .
- ▶ **variable *tolcnv***: tolerància màxima per aplicar al càlcul de la funció *cnvnt*.
- ▶ **variable *maxit***: nombre màxim d'iteracions que pot fer la funció *cnvnt*.

Aquesta comanda executarà el programa *dibfr* i mostrarà per pantalla tota la informació que calcula.

*Expliquem a la següent secció la informació que retorna, com utilitzar-la i el procés que segueix per a obtenir-la.*

## Programa dibfr.c

El programa *dibfr.c*, com hem estat comentant, s'encarrega de fer els càlculs per a representar els fractals, i recordar també que, l'executable del programa es pot obtenir mitjançant el fitxer Makefile explicat en l'apartat anterior.

El funcionament de la mateixa (explicat al final de la secció) és senzilla, una vegada a llegit i guardat els arguments introduïts per pantalla (en la comanda abans explicada) el programa preguntarà tants cops com arrels ha decidit l'usuari que tingui els següents valors (que han de ser introduir seguits amb un espai entre ells):

- ◇ **Valor real de l'arrel**, aquest ha de ser un valor double
- ◇ **Valor imaginari de l'arrel**, aquest ha de ser un valor double
- ◇ **Color vermell en l'arrel**, aquest ha de ser 0 o 1, indica si l'arrel és vermella
- ◇ **Color verd en l'arrel**, aquest ha de ser 0 o 1, indica si l'arrel és verda
- ◇ **Color blau en l'arrel**, aquest ha de ser 0 o 1, indica si l'arrel és blava

*En els valors dels colors, el valor 0 indicarà absència del mateix i 1 indicarà el color de l'arrel*

Un exemple d'introducció de la informació per pantalla una vegada executat *dibfr*, introduïm l'arrel  $z = 1 + 0 \cdot i$  i indiquem que serà de color vermell: 1 0 1 0 0.

Els valors, respectivament, representen: el valor real de l'arrel  $z$ , el valor imaginari de l'arrel  $z$ , conté el color vermell, no conté el color verd i no conté el color blau.

Una vegada executat amb els arguments explicats i introduïts per consola els valors abans anomenats, el programa mostrarà per pantalla tota la informació, per la qual cosa es recomana a l'hora d'executar el programa per pantalla (just després dels arguments) col·locar una redirecció d'entrada a un fitxer per a contenir tota la informació allà i poder treballar de manera més còmoda amb el gnuplot (que explicarem a continuació d'aquesta secció).

Eventualment, cal mencionar que el funcionament de *dibfr.c* es basa en obtenir la informació necessària per a poder fer les declaracions com a arguments (ja explicats), rebre la informació sobre els càlculs per pantalla (també explicada) i calcular els colors dels píxels de la pantalla tractant les coordenades X i Y de la pantalla com els components real i imaginari (respectivament) d'un nombre complex aplicant la funció *cnvnt* de la nostra biblioteca.

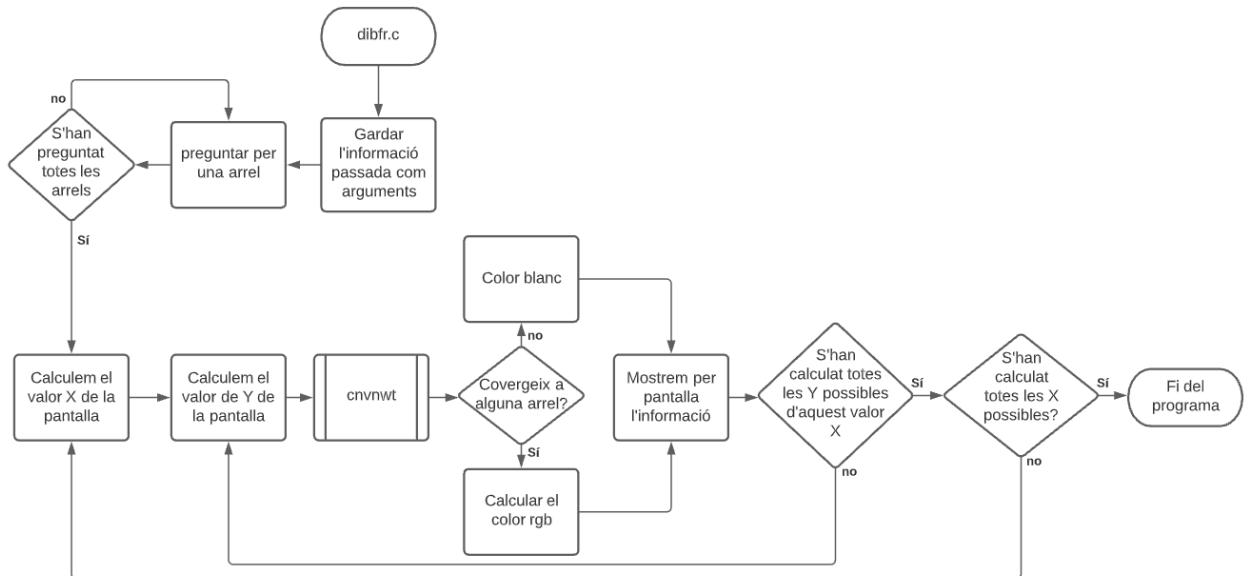


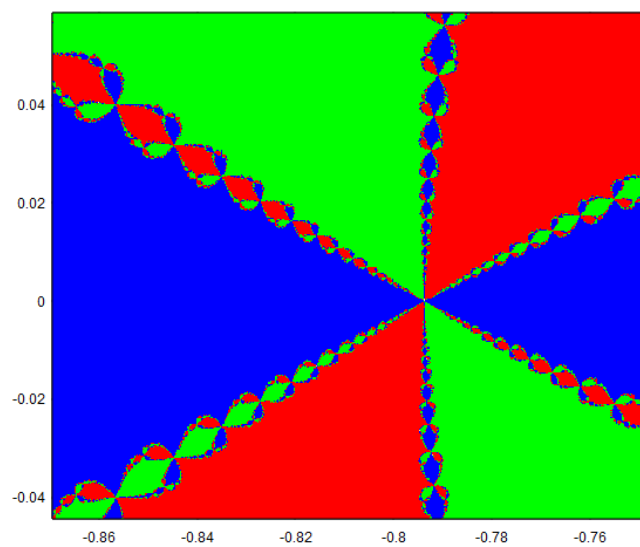
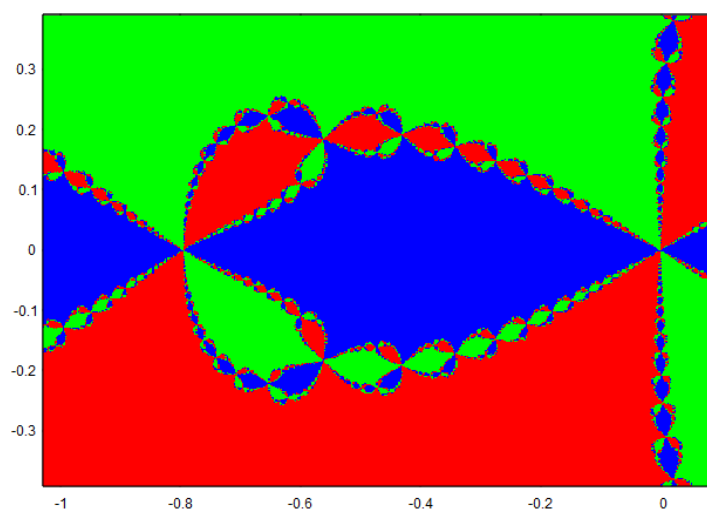
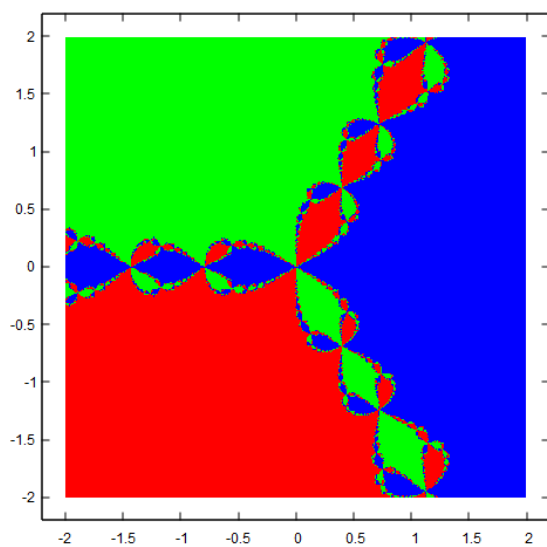
Diagrama de flux de *dibfr.c*

## Representació amb gnuplot

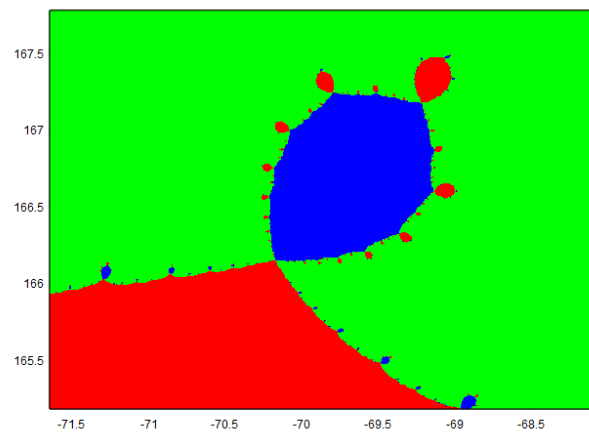
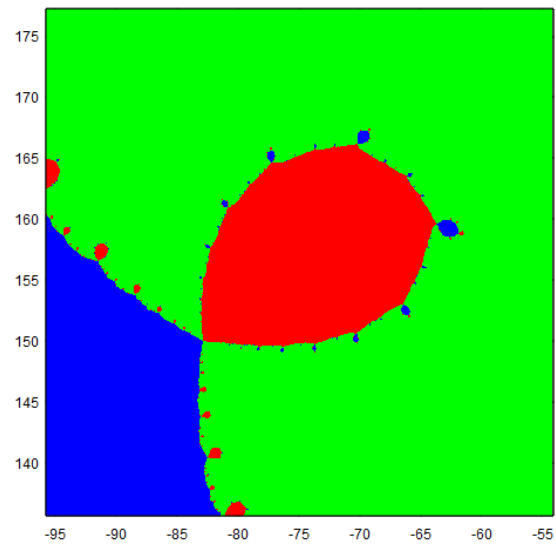
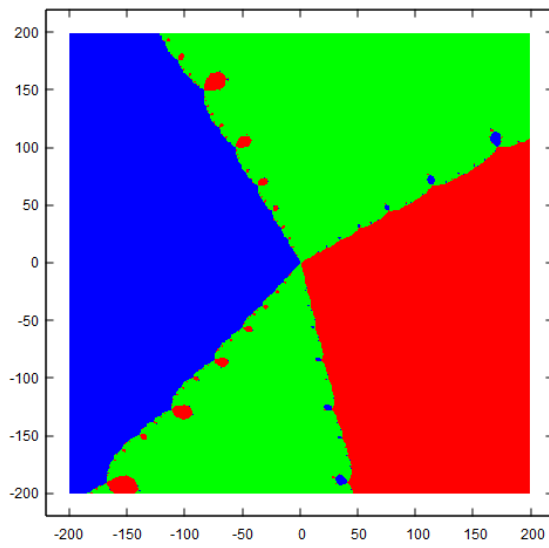
En aquesta secció expliquem com representar per mitjà del software d'accés lliure *gnuplot*. Per a poder representar fractals a partir del *gnuplot* ens situarem primerament en el directori on disposem de l'arxiu on hem recollit tota la informació obtinguda del programa *dibfr*. Una vegada en el directori, introduïm per pantalla les següents comandes:

- > unset key
- > plot 'nom\_del\_fitxer\_amb\_l'informació.txt' using 1:2:3:4:5 with rgbimage

Per finalitzar, mostrarem a continuació una recopilació d'imatges de les representacions que hem obtingut seguint aquesta metodologia.



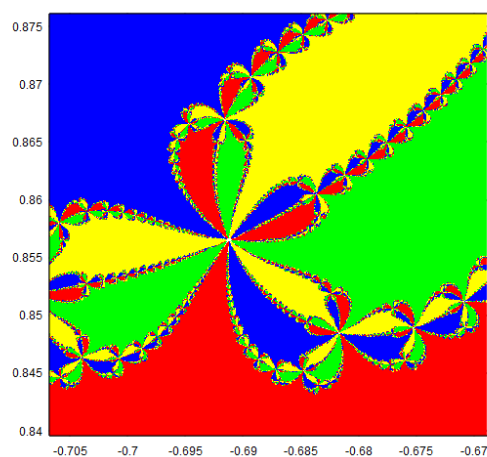
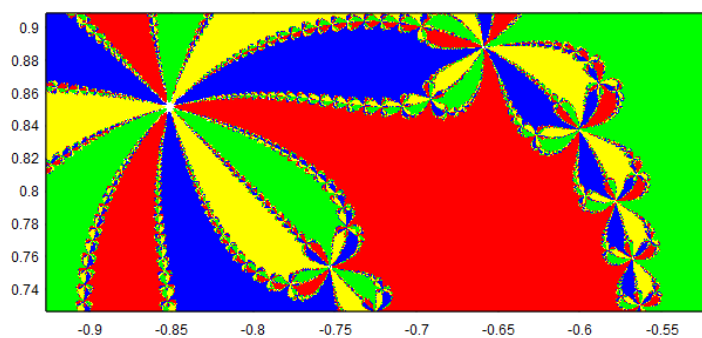
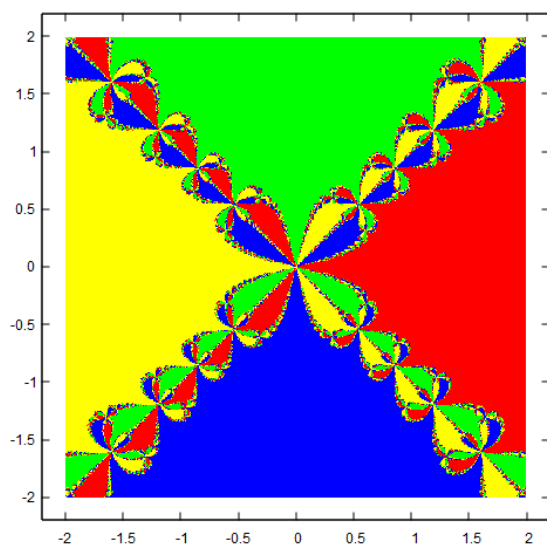
Fractal dibuxat pel polinomi  $p(z) = z^3 - 1$



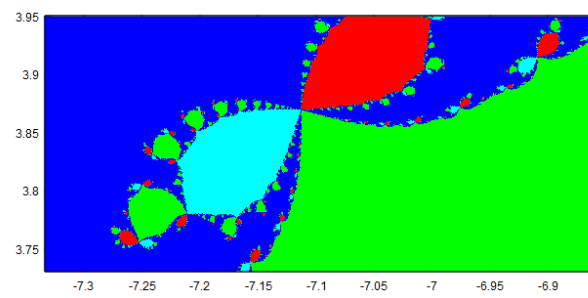
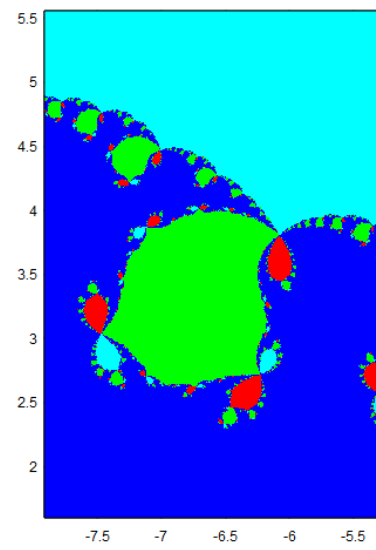
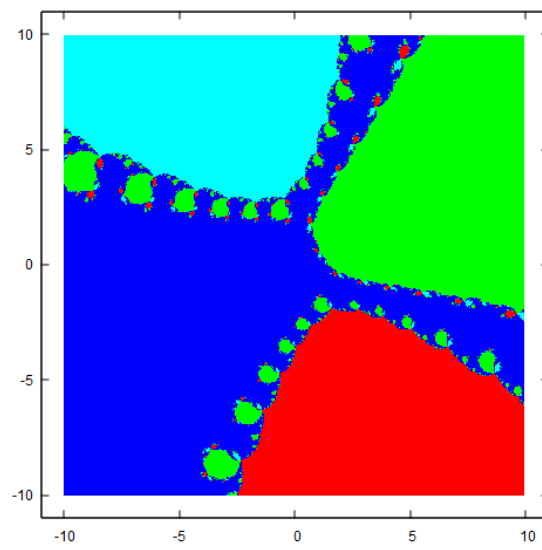
Fractal dibuxat pel polinomi

$$p(z) = (z - 1.7693)(z - 0.88465 - 0.58983i)(z + 0.88465 - 0.58983i)$$

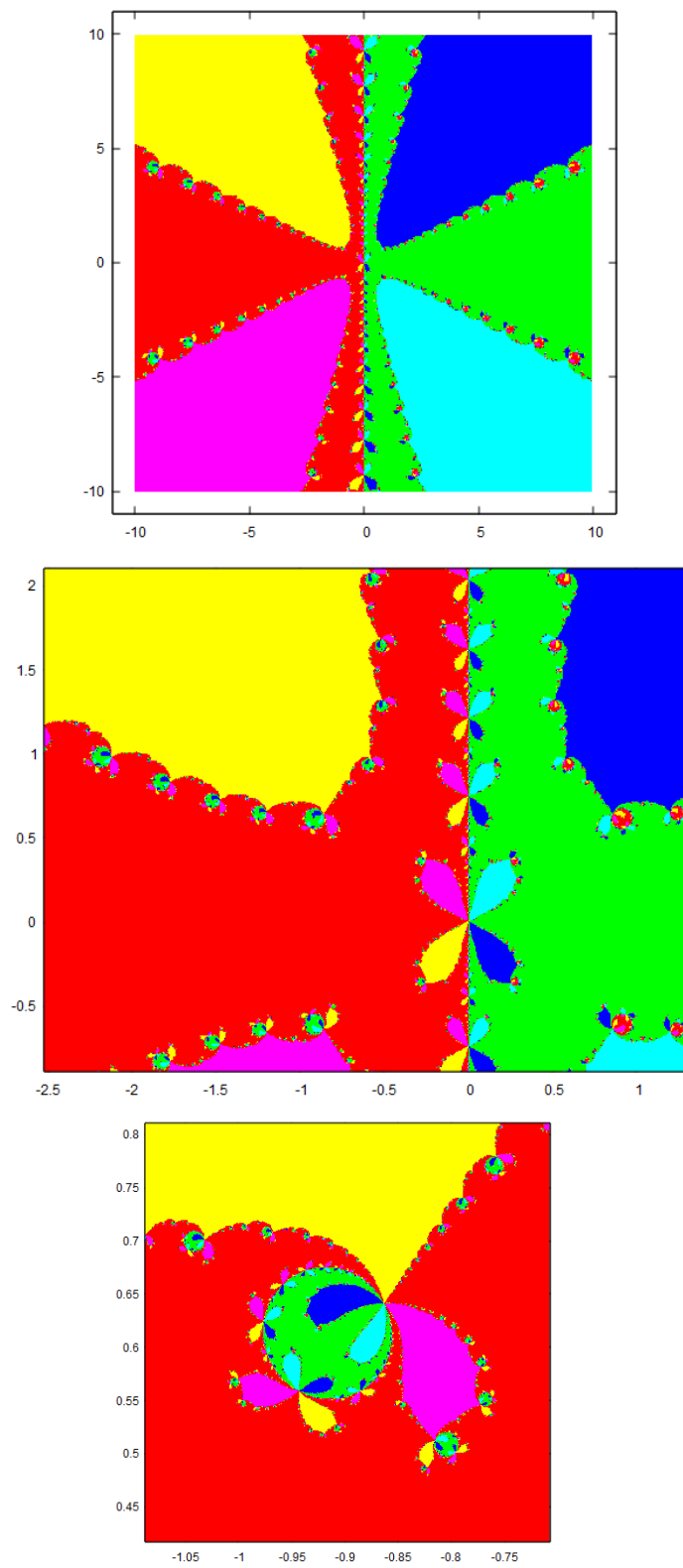




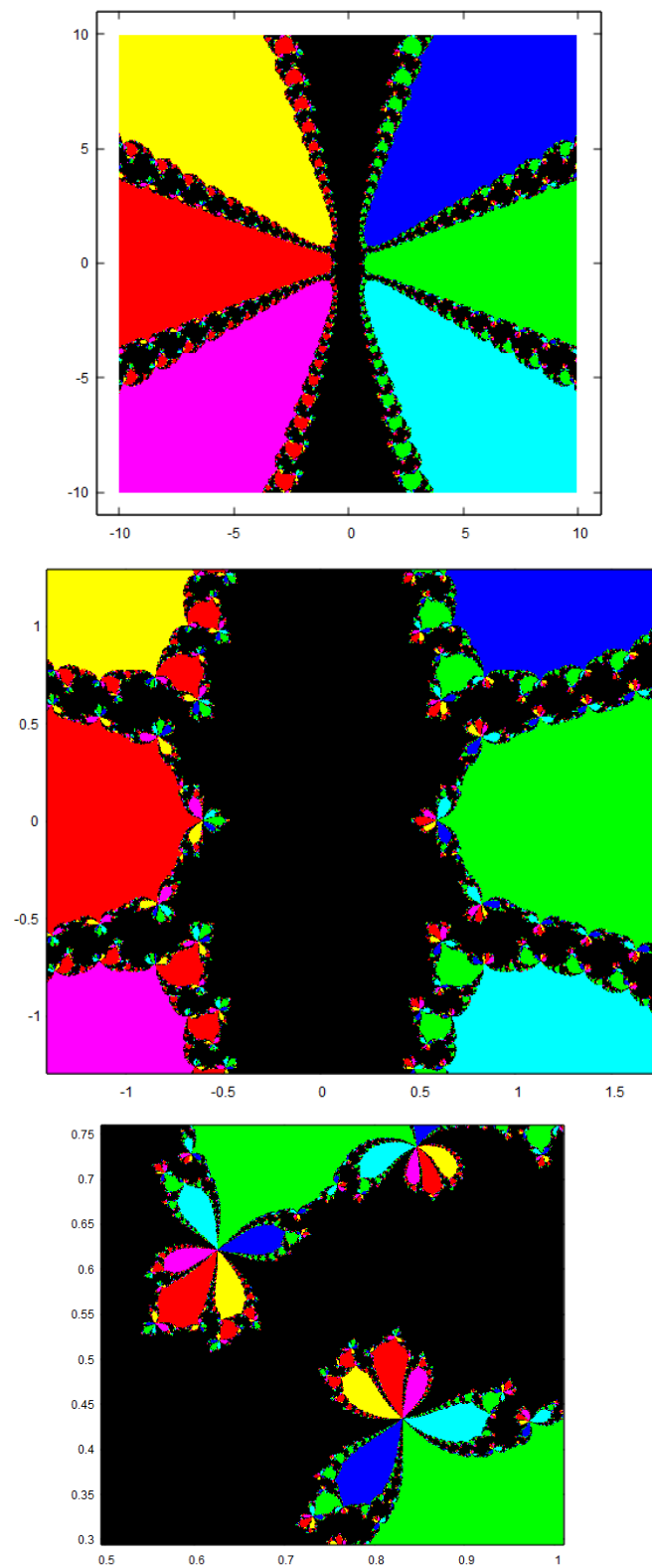
Fractal dibuxat pel polinomi  $p(z) = z^4 - 1$



Fractal dibuxat pel polinomi  $p(z) = z(z - 2 + 3i)(z + 1 - 4i)(z - 2 - i)$



Fractal dibuxat pel polinomi  $p(z) = (z+1)(z+1+i)(z+1-i)(z-1)(z-1+i)(z-1-i)$



Fractal dibuxat pel polinomi  $p(z) = z(z+1)(z+1+i)(z+1-i)(z-1)(z-1+i)(z-1-i)$

$$\sum_{k=0}^{\infty} e^{-\lambda} \cdot \lambda^{2K} / (2k)! = e^{-\lambda} \cdot (1/2) \cdot (e^{\lambda} + e^{-\lambda}) = (1 + e^{-2\lambda})/2$$