区間演算を用いた反復計算について

大墨 礼子 (関東学院大 理工学部 情報学系)

近藤 祐史 (香川高等専門学校)

藤村 雅代 (防衛大 数学教育室)

2022. December 19

何をやっているか

藤村先生からの依頼

Hedgehog(存在は既知だが，まだ誰も見たことがない)
を見たい！

⇓

その前に正確に Julia 集合の描画を行いたい
(平成 25 年 7 月 18 日 ふじむらメモ　より)

何をやっているか

藤村先生からの依頼

Hedgehog(存在は既知だが，まだ誰も見たことがない)
を見たい！

⇓

その前に正確に Julia 集合の描画を行いたい
(平成 25 年 7 月 18 日 ふじむらメモ　より)

目標

Julia 集合の正確な描画をしたい！

Level Set 法

描画アルゴリズムにはいろいろあるが，

Level Set 法

での描画を考える

Level Set 法

描画領域の各格子点 z に対して，p_c による軌道

$$z, p_c(z), p_c(p_c(z)), \ldots$$

を計算する．

十分大きな n で

$$|p_c^n(z)| > R(> 2)$$

となれば，$z \in A_0(\infty)$ と判定する

残った点が p_c の filled-in Julia 集合

Level Set 法

描画領域の各格子点 z に対して

$z \to z^2 + c \to (z^2 + c)^2 + c \to ((z^2 + c)^2 + c)^2 + c \to \ldots$

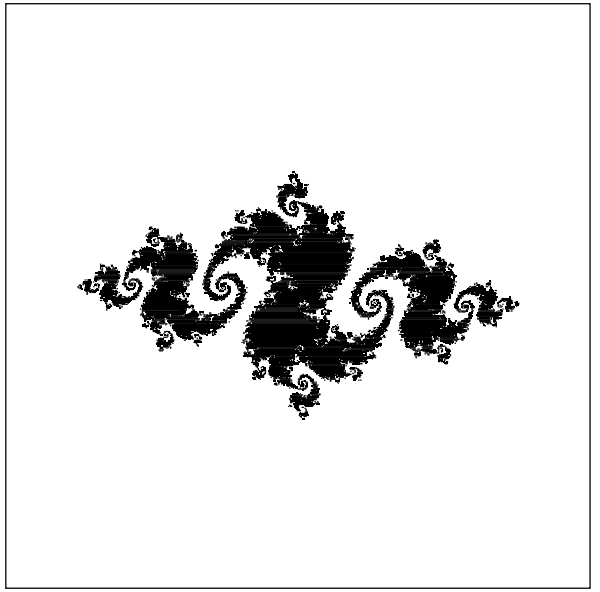
を計算する．

十分大きな繰り返し回数で
$|p_c^n(z)| < 10$ となるものを正確に描画したい

c の値は $c = -0.778264 + 0.1155285i$ のように与えられる

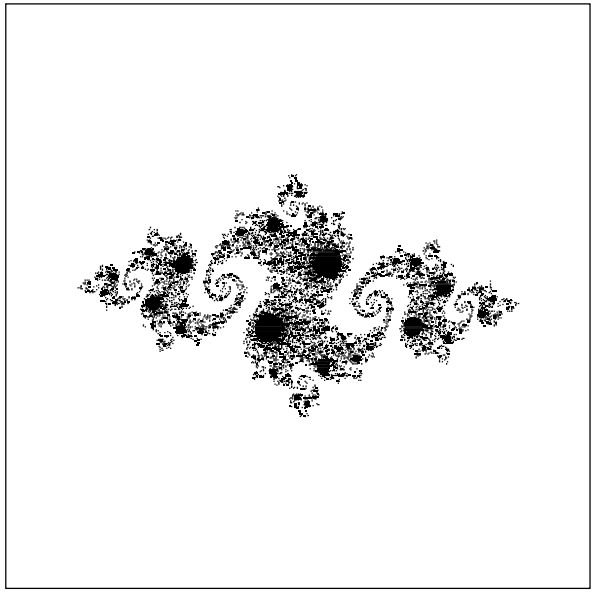
LevelSet 法 浮動小数点計算で

繰り返し回数 500 回 28079 点



LevelSet 法 浮動小数点計算で

繰り返し回数 10000 回 11862 点



浮動小数点での計算は

繰り返し回数を多くしても計算は可能，ただし

これらの絵は正確なのか？

正しい絵を何とか描きたい

ではどうする？

有理関数や多項式の **Julia** 集合を

できるだけ (数学的に)

正しく描きたい。

⇓

解像度？

精度？

そもそも計算機が描けるのは安定した集合のみでは？

⇓

誤差をどれだけ減らせるか？

ではどうする？

有理関数や多項式の **Julia** 集合を

できるだけ (数学的に)

正しく描きたい。

⇓

解像度？

精度？

そもそも計算機が描けるのは安定した集合のみでは？

⇓

誤差をどれだけ減らせるか？

安直な方法

正確に書きたいんだったら，全部有理数でやればよい

$z \to z^2 + c \to (z^2 + c)^2 + c \to ((z^2 + c)^2 + c)^2 + c \to \ldots$

$c = -\frac{778264}{1000000} + \frac{1155285}{10000000} i$

かなり計算しても終了しない

式はどうなっているか

巨大な有理数

- メモリを大量に消費しているわけではない
- 待ってれば出る？

とはいえそのまま単に実行して待つのは現実的ではない

もう少し気軽に，かつ精度を上げられないもんか？

区間演算を用いてみたらどうか？

Risa/Asir では区間演算の実装あり

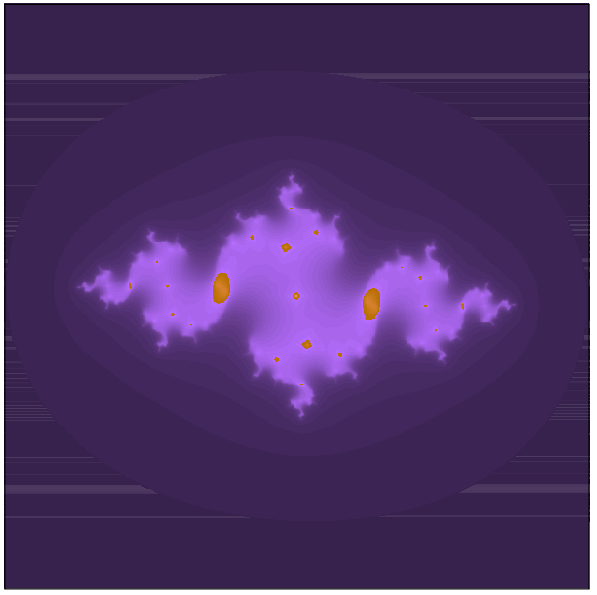
- $A = intval(a, b);$

とすることで，A を区間数とすることができる

(そのまま使えば a,b は double)

描画領域の各格子点 z を区間数として演算してみる

区間数では？



区間数では？

単に，格子点を区間数としてみるだけでは 1 点も検出できない

要は 1 点も条件を満たす点は存在しないと判断される

区間数→演算を繰り返すと膨らむ

ゼロを挟んだ区間の扱い

ゼロ書き換えの実行

ゼロ書き換え とは？

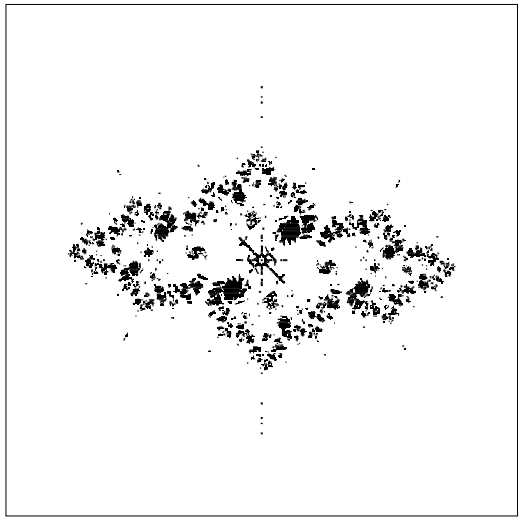
区間演算の結果に 0 が含まれる区間数を 0 に置き換える

ctrl("zerorewrite", 1);

とすることでゼロ書き換えモード有効

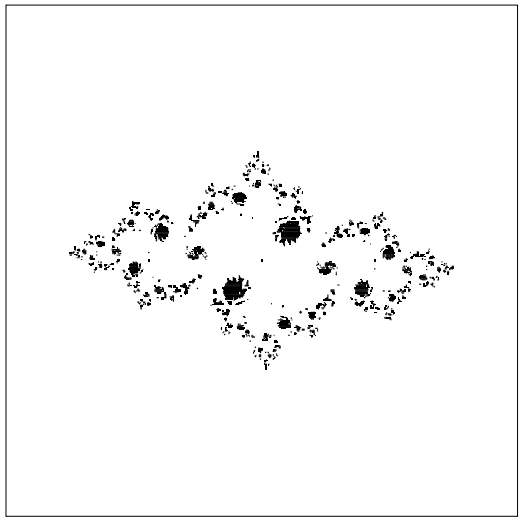
ゼロを挟んだ区間の扱い

double の区間数 で ゼロ書き換え実行



ゼロを挟んだ区間の扱い

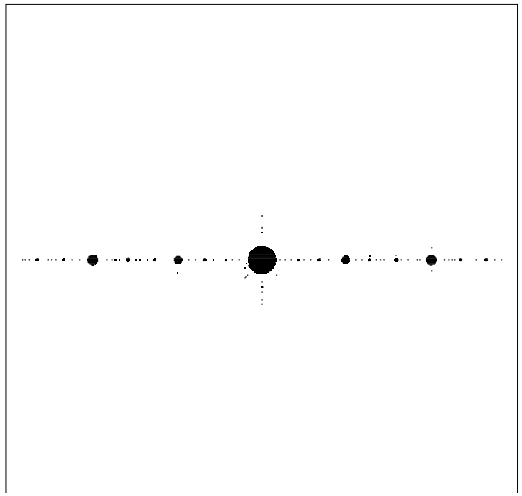
bigfloat の区間数 で ゼロ書き換え実行



ゼロを挟んだ区間の扱い

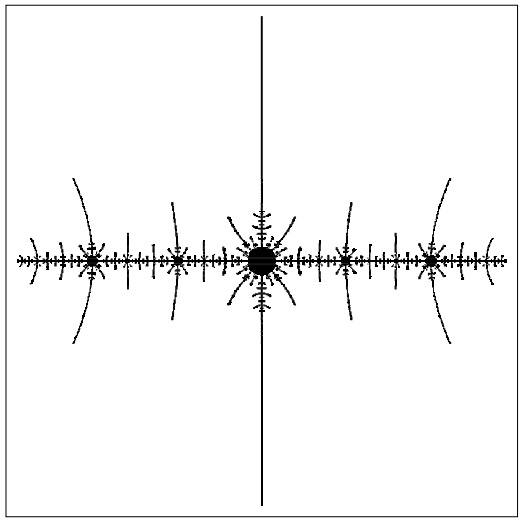
単純な LebelSet 法では誤差で実軸上のものが消えてしまうものは区間数（ゼロ書き換え）を使うと出てくる

$julia_l(0.001, 0.001, 2, -1.75487766624669276, 500);$



ゼロを挟んだ区間の扱い

$julia_l(0.001, 0.001, 2, -1.75487766624669276, 500);$



ゼロを挟んだ区間の扱い

格子点を区間として扱った場合
ゼロ書き換えの効果は実軸上の場合のみうまく効いている？ ように見える

- 浮動小数点数での LebelSet 法
  誤差の影響大→削りすぎる
  ただしうまく実軸や虚軸上でしか動かない軌道にハマるとその部分だけは描画可能
  （ちょっと軸をずらすとやっぱり描画できない）
- 格子点を区間数とした LebelSet 法
  1 つも描画されない
  演算を繰り返すたびに区間が膨らむ→結局削りすぎる
- ゼロ書き換えを適用した格子点を区間数とした LabelSet 法
  実軸方向はそこそこうまく描けるがそれ以外は×

2022 区間数 再検討

正確にやるなら有理数計算→繰り返し回数は多くできない

区間数もそう簡単にはうまくいかない (2019)

では式の展開を重ねてから，等を考えてみたがそれもなかなかうまくいかない (2021)

2022 区間数 再検討

区間数は本当に使えないのか？

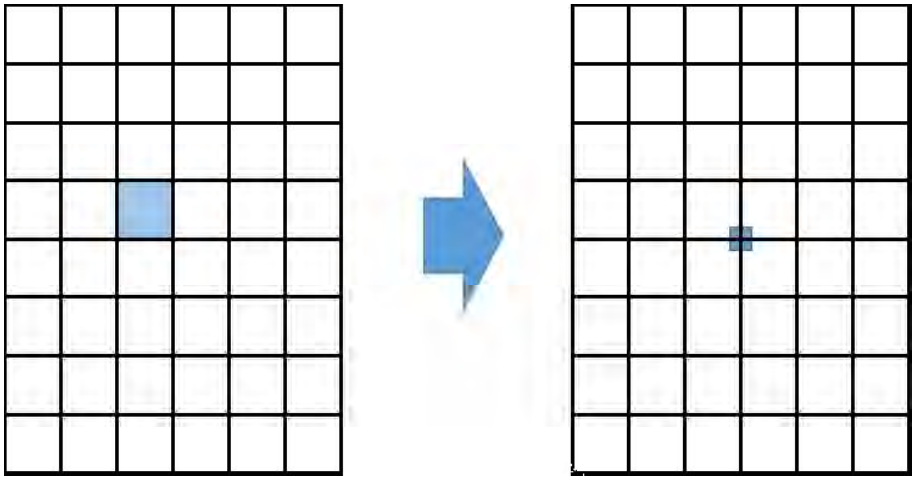
- ゼロ書き換えは効果があった

それ以外にも見直せるところは？

- 区間を格子点間としている

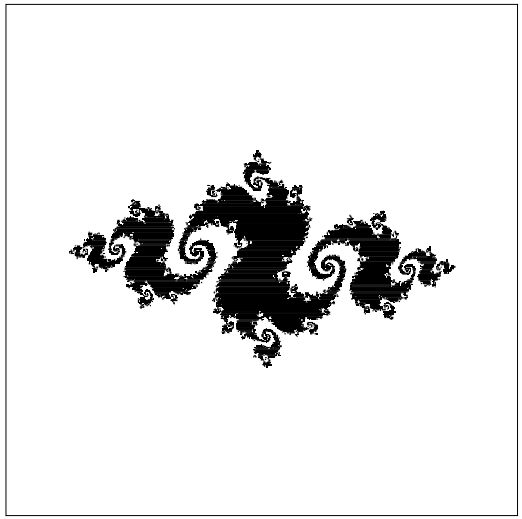
区間の取り方に工夫はできないだろうか？

2022 区間数 再検討



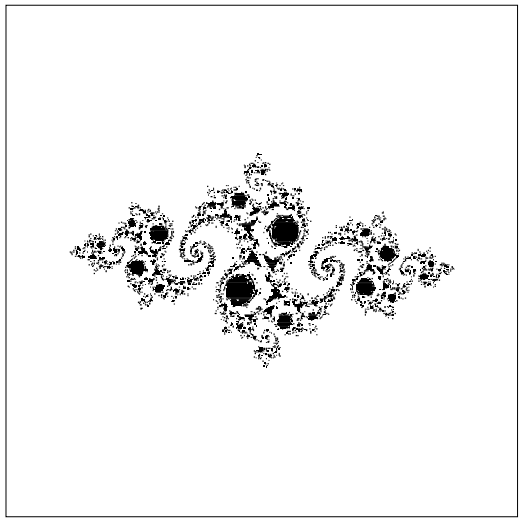
Cell として見る → 格子点として見る

2022 区間数 再検討



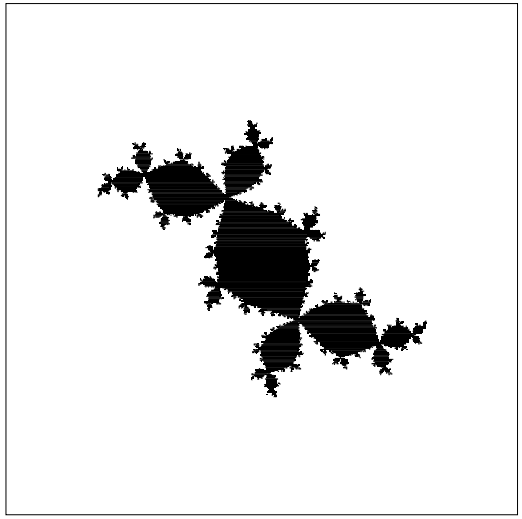
格子点および c の値を 土$\frac{5}{1000000000000000}$ 100 回繰り返し 30155 点

2022 区間数 再検討



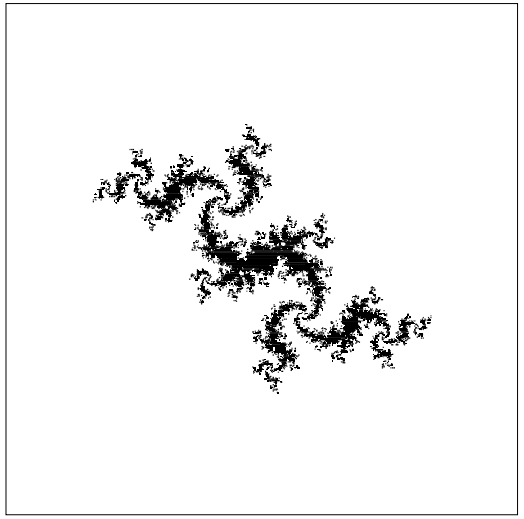
格子点および c の値を 土$\frac{5}{1000000000000000}$ 150 回繰り返し ９８６４点

2022 区間数 再検討



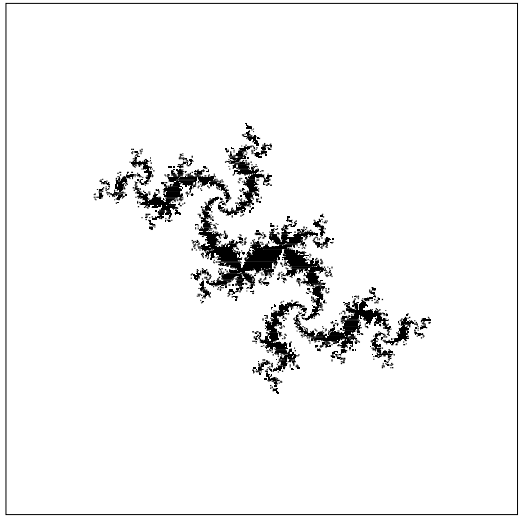
$c = -0.1226 + 0.7449i$,
格子点および c の値を $\pm\frac{5}{10000000}$ 100 回繰り返し 29833 点

2022 区間数 再検討



$c = -0.222 + 0.71838i$,
格子点および c の値を $\pm\frac{5}{10000000}$ 100 回繰り返し 12381 点

2022 区間数 再検討



$c = -0.222 + 0.71838i$,
格子点および c の値を $\pm\frac{5}{1000000000000000}$ 500 回繰り返し 11673 点

絵を見る限り

そこそこうまく描けているように見える（DEM 法の結果に近い）

数値計算なので，実行速度も速い

ただし，区間数であれ数値計算であるので，誤差の影響は必ずある

特に，区間数→必ず区間が膨らむ→この影響は必ず出ている
（削りすぎる点が絶対出てくる）

今後の検討

- 適切な区間の設定
- 繰り返し回数
- 有理数計算との併用

まとめ

- 正確さを求めるなら有理計算一択
- 区間演算の適切な利用

  レベルセット法は浮動小数点であれ，区間数であれ，
  削りすぎてしまう点は変わらない（頻度の差はあれども）

  うまく使えば区間数は存在する点に関しては保証ができる
  （と思われる）