第3章 数学との親和性を体感

Julia科学計算メカニズムの研究

寺崎 敏志

マンデルブロ集合によるベンチマーク

● マンデルブロ集合の定義

Juliaのコードはどれくらい性能があるのか確かめます．ここでは比較のため，マンデルブロー集合を計算するCのコードと，Juliaのコードを用意しました．マンデルブロ集合は，

$z_n = z_{n-1}^2 + c \ (n \geq 1)$
$z_0 = 0$

で定義される複素数列，$\{z_n\}_{n=0}^{\infty}$で絶対値$|z_n|$が発散しないようにする複素数cの集合です．$|z_n| > 2$となる要素があれば，発散することが知られています．各領域複素数領域$c \in \mathbb{C}$において定まる数列，

$z_1, z_2, \cdots, z_k, \cdots$

を逐次計算することになりますが，ここでは近似のため計算回数の最大値を255としておきます．

● JuliaとC言語を比べてみる

リスト1(a)にC言語のコードを示します．次にC言語の場合のコンパイル例と実行例を示します．

```
$ gcc -O2 mand.c && ./a.out
Elapsed time 1.724609 [sec]
```

リスト2(a)にJuliaのコードを示します．REPLで入力するかファイルに保存した後でincludeで読み込ませます．次に実行例を示します．

```
julia> main(2500,2500)
  1.601916 seconds (2 allocations:
          5.961 MiB, 0.65% gc time)
```

Juliaは，C言語と同等の性能を発揮できています．

● Juliaのもう少しスマートな書き方

上のJuliaの例では，C言語と比較するため，わざと冗長な書き方をしましたが，下記のようにもっと抽象化されたコードを記述できます[**リスト2**(b)]．

次に実行例を示します．

```
julia> main(2500,2500)
  1.648182 seconds (4 allocations:
          11.921 MiB, 0.11% gc time)
```

次の通り，絶対値の2乗が4.0よりも大きい場合，kを返すということを1行で書けます．

```
abs2(z) > 4.0 && return k
```

これは短絡評価(Short-Circuit Evaluation)を使ったテクニックで，Juliaのパッケージでもよく使われています．

```
xs = range(-2, 2, length = N)
```

は-2から2までの区間を，N等分して得られるrangeオブジェクトです．

```
cnt.(init_z, complex.(xs', ys))
```

はdot syntaxという関数の実行を，ベクトル化するJuliaの文法です．xs'はxsを列ベクトルから行ベクトルに転置する役割を果たします．

```
complex(xs', ys)
```

はxsの要素xは横側に，ysの要素はyは縦方向にループを走らせた2重ループ内で，

```
complex(x,y)
```

を実行していることと同じです．

```
cnt.(init_z, complex(xs', ys))
```

は2次元配列，

```
complex.(xs', ys)
```

の要素cに対して，

```
cnt(init_z, c)
```

を実行する意味になります．

イメージしやすいよう，**リスト3**のようなサンプルを作ってみました．

もともと2重ループで記述していた部分を，dot syntaxtにより，

```
grid = cnt.(init_z, complex.(xs',ys))
```

と簡潔に表記できました．関数ごとにドット“.”を付け加えることになります．

さらに楽にベクトル化するため，dot syntaxは“@.”というマクロで，次のように記述することもできます．

```
@. cnt(init_z, complex(xs', ys))
```

実行結果は次の通りです．

```
julia> main(2500,2500)
```