複合台形公式による留数の数値計算

2610130079
4年2組58番
比留間　勇人

担当教員　　桂田　祐史准教授

目次

1、はじめに 1

2、複合台形公式とは 2

3、留数とは 5

4、実験 6
4-1、手順 6
4-2、結果 7

5、結論 22

6、数値実験に使用したプログラム 23

7、参考文献 32

1、はじめに

今回、私は「複合台形公式による留数の数値計算」を卒業研究のテーマとした。

一般的に、解析的周期関数の1周期の積分において、複合台形公式を用いて数値計算を行うと、非常に精度の良い結果が得られるということが知られている。

今回の研究では、閉曲線のまわりの正則関数の積分を複合台形公式で数値計算を行い、精度の良い近似値が得られるのか、特に留数の計算において、複合台形公式での数値計算で精度の良い結果を得られるのかについて、実際にプログラムを作成し、数値実験を行い、検証していった。

2、複合台形公式について

複合台形公式とは、数値積分の１つである。以下がその説明である。

$\int_a^b f(x)dx$の近似値を求める場合、区間$[a,b]$をN等分して、各小区間 $[x_{j-1},x_j]\,(j=1,2,\cdots N)$で区間の端点での関数の値$f(x_{j-1})$、 $f(x_j)$で、1次関数補間して、それを積分したもの（台形の面積の和）を作る。

$$T_N \coloneqq h\left(\frac{1}{2}f(a)+\sum_{j=1}^{N-1}f(a+jh)+\frac{1}{2}f(b)\right),\ h\coloneqq\frac{b-a}{N}$$

数値積分には、複合台形公式以外にも、複合中点公式、複合Simpson公式がある。
複合中点公式、複合Simpson公式についても、説明しておく。

複合中点公式

$\int_a^b f(x)dx$の近似値を求める場合、区間$[a,b]$をN等分して、各小区間$[x_{j-1},x_j]$ $(j=1,2,\cdots N)$で、区間の中点での関数の値$f\left(x_{j-1}+\frac{h}{2}\right)$を用いて0次関数補間して、それを積分したもの（長方形の面積の和）を作る。

$$M_N \coloneqq h\sum_{j=1}^{N}f\left(a+\left(j-\frac{1}{2}\right)h\right),\ h\coloneqq\frac{b-a}{N}$$

複合Simpson公式

$\int_a^b f(x)dx$の近似値を求める場合、区間$[a,b]$をm等分して、各小区間$[x_{j-1},x_j]$ $(j=1,2,\cdots m)$をさらに2等分して、区間の2つの端点と中点の3点での関数値を使って、2次関数補間して、それを積分したもの（$\frac{h}{3}\left(f(x_{j-1})+4f\left(\frac{x_{j-1}+x_j}{2}\right)+f(x_j)\right)$の和）を作る。

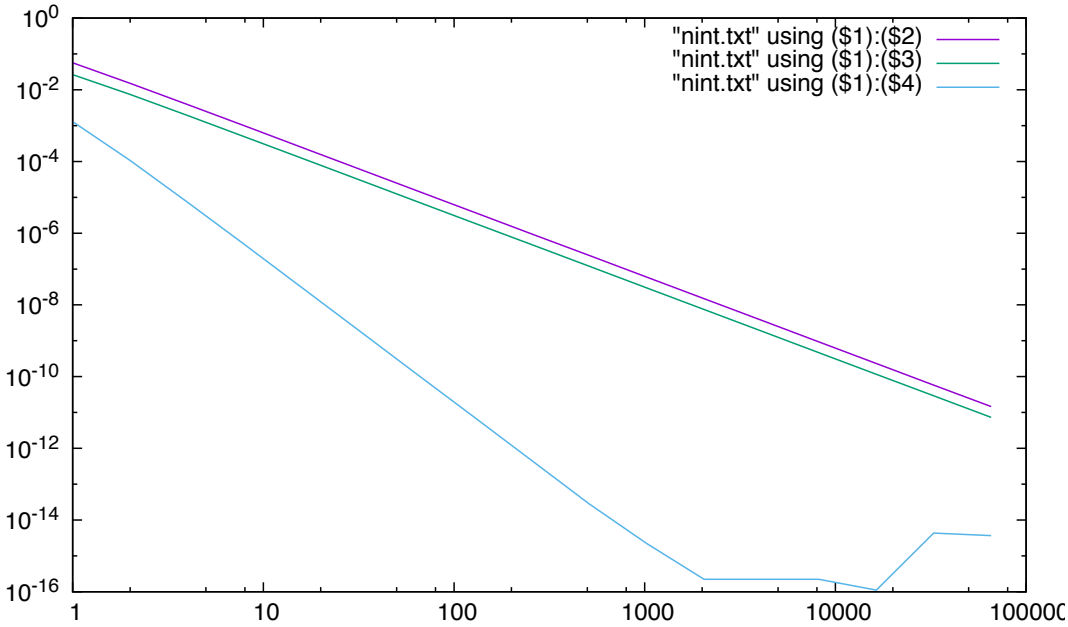
$$S_{2m} \coloneqq \frac{h}{3}\left(f(a)+2\sum_{j=1}^{m-1}f(a+2jh)+4\sum_{j=1}^{m}f(a+(2j-1)h)f(b)\right),\ h\coloneqq\frac{b-a}{2m}$$

この3つの公式のうち、通常、精度が一番良いとされているのは複合Simpson公式である。複合台形公式、複合中点公式では、精度はそれほど期待できない。

以下は $\int_1^2 \frac{dx}{x} = \log 2$ を3つの方法で計算した結果である。

N	複合台形則の誤差	複合中点則の誤差	複合Simpson則の誤差
1	5.685282e-02	2.648051e-02	1.297264e-03
2	1.518615e-02	7.432895e-03	1.067877e-04
4	3.876629e-03	1.927289e-03	7.350095e-06
8	9.746698e-04	4.866265e-04	4.722595e-07
16	2.440216e-04	1.219662e-04	2.972988e-08
32	6.102771e-05	3.051106e-05	1.861510e-09
64	1.525832e-05	7.628987e-06	1.163973e-10
128	3.814668e-06	1.907323e-06	7.275514e-12
256	9.536725e-07	4.768356e-07	4.551914e-13
512	2.384185e-07	1.192092e-07	2.797762e-14
1024	5.960464e-08	2.980232e-08	2.220446e-15
2048	1.490116e-08	7.450581e-09	2.220446e-16
4096	3.725290e-09	1.862645e-09	2.220446e-16
8192	9.313247e-10	4.656625e-10	2.220446e-16
16384	2.328292e-10	1.164144e-10	1.110223e-16
32768	5.820444e-11	2.910883e-11	4.329870e-15
65536	1.455958e-11	7.274403e-12	3.663736e-15

下図は上記データをグラフ化したものである。縦軸が真の解との誤差、横軸がNとなっている。また、紫色のグラフは台形公式、緑色のグラフは中点公式、水色のグラフはSimpson公式をそれぞれ表している。



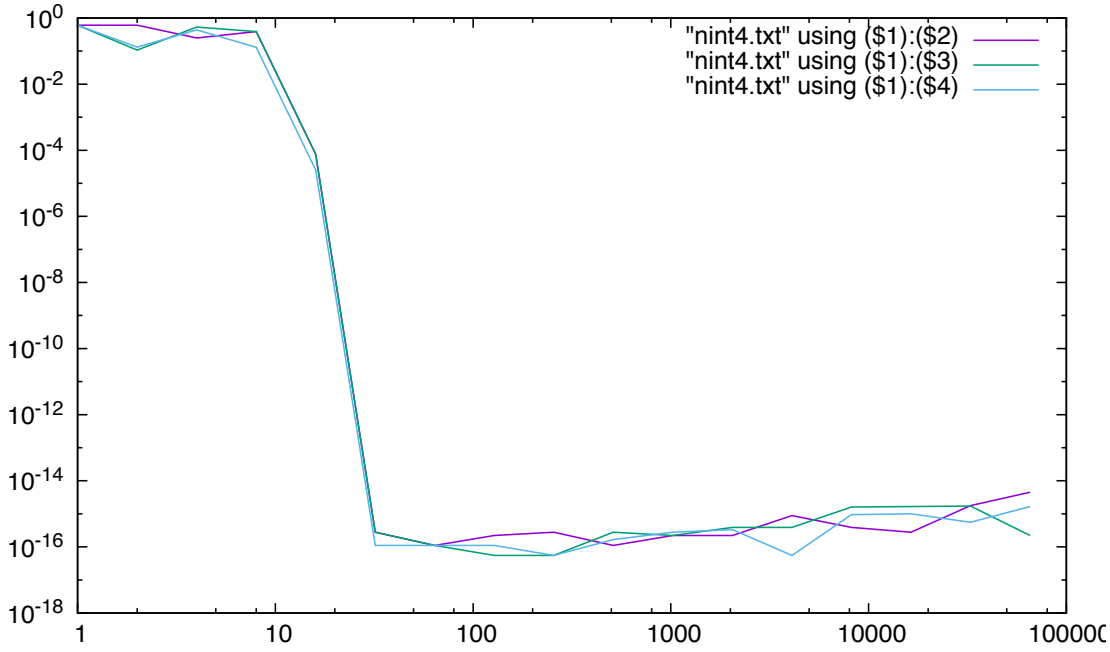
しかし、解析的周期関数の1周期の積分については、$T_N = h\sum_{j=0}^{N-1} f(a+jh)$ と表され、複合台形公式の精度が非常に良いということが知られている。

以下は、解析的周期関数の積分である $\frac{1}{2\pi i}\int_{-\pi}^{\pi}\cos(4t-5\sin t)dt$ を3つの方法で計算した結果である。

複合台形則 T_m, 複合中点則 M_m, 複合 Simpson 則 S_{2m} の誤差

m	I−T_m	I−M_m	I−S_{2m}
1	6.087676e−01	6.087676e−01	6.087676e−01
2	6.087676e−01	1.075702e−01	1.312091e−01
4	2.505987e−01	5.321711e−01	4.383136e−01
8	3.913849e−01	3.912324e−01	1.303599e−01
16	7.627816e−05	7.627816e−05	2.542605e−05
32	2.775558e−16	2.775558e−16	1.110223e−16
64	1.110223e−16	1.110223e−16	1.110223e−16
128	2.220446e−16	5.551115e−17	1.110223e−16
256	2.775558e−16	5.551115e−17	5.551115e−17
512	1.110223e−16	2.775558e−16	1.665335e−16
1024	2.220446e−16	2.220446e−16	2.775558e−16
2048	2.220446e−16	3.885781e−16	3.330669e−16
4096	8.881784e−16	3.885781e−16	5.551115e−17
8192	3.885781e−16	1.609823e−15	9.436896e−16
16384	2.775558e−16	1.665335e−15	9.992007e−16
32768	1.776357e−15	1.720846e−15	5.551115e−16
65536	4.496403e−15	2.220446e−16	1.665335e−15

グラフの見方は、前ページと同じである。



このように、複合台形公式は、解析的周期関数の1周期の積分において、非常に高精度な結果を得ることができる。

3、留数について

まずは、留数の定義から説明する。留数の定義とは以下のものである。

$z=c$を関数 $f(z)$ の孤立特異点とする。$0<|z-c|<R$ で $f(z)$ が1価正則であるとすると、

$$f(z)=\sum_{k=1}^{\infty}\frac{a_{-k}}{(z-c)^k}+\sum_{k=0}^{\infty}a_k(z-c)^k \qquad (0<|z-c|<R)$$

とローラン展開できる。この$\frac{1}{z-c}$の項の係数a_{-1}を$f(z)$の$z=c$ における「留数」と呼び、$\mathrm{Re}\,s(f;c)$と表す。

ローラン展開における係数a_{-k}は、$0<r<R$を満たす任意のrに対して

$$a_{-k}=\frac{1}{2\pi i}\oint_{|z-c|=r}f(z)(z-c)^{-(k-1)}dz$$

となる$(k\in\mathbb{Z})$。よって$k=1$のとき、

$$\mathrm{Re}\,s(f;c)=a_{-1}=\frac{1}{2\pi i}\oint_{|z-c|=r}f(z)dz$$

となる。留数は、定積分の計算、級数の和の計算、偏角の原理、Roucheの定理などに応用されている非常に重要なものである。

ここで留数定理についても触れておく。留数定理とは、以下のものである。

$f(z)$が単純閉曲線Cとその内部で、C内の有限個の孤立特異点$c_1,c_2,c_3,\cdots c_n$を除いて、1価正則な関数とする。このとき、

$$\oint_C f(z)dz=2\pi i\sum_{j=1}^{N}\mathrm{Re}\,s(f;c_j)$$

が成り立つ。

この留数定理を使うことにより、領域内に複数の特異点が存在する場合の1周線積分の計算が簡単にできる。

4、実験

4-1、手順

数値実験①

「閉曲線のまわりの正則関数の積分について複合台形公式で数値計算をし、近似値を求める」をテーマに実験を行う。

1，$\oint_{|z-1|=2} \frac{dz}{z-2} = 2\pi i$ の数値計算

2，$\oint_{|z+i|=1} \frac{e^z}{z+i} dz = 2\pi(\sin 1 + i\cos 1)$ の数値計算

数値実験②

「留数の計算について複合台形公式で数値計算をし、近似値を求める」をテーマに実験を行う。

1，$f(z) = \frac{1}{(z-1)(z-2)}$ の場合の $\mathrm{Re}\,s(f;1) = -1$ と $\mathrm{Re}\,s(f;2) = 1$ の数値計算

(i)　r=0.1についての留数計算
(ii)　r=0.5についての留数計算
(iii)　r=0.9についての留数計算
(iv)　r=0.99についての留数計算

この数値実験について、rの値をこのように与えていく理由は、$f(z)$ の特異点が1と2であるので、半径を、中心に取っていない特異点に近づけていった時に台形公式の精度がどのようになるのかを調べるためである。

2，$f(z) = e^{\frac{1}{z}}$ の場合の $\mathrm{Re}\,s(f;0) = 1$ の数値計算

(i)　r=1についての留数計算
(ii)　r=0.5についての留数計算
(iii)　r=0.1についての留数計算
(iv)　r=0.01についての留数計算

この数値実験について、rの値をこのように与えていく理由は、$e^{\frac{1}{z}} = 1 + \sum_{n=1}^{\infty} \frac{1}{n!} \frac{1}{z^n}$ より、$f(z)$ の特異点0は真性特異点であるということが分かるので、半径を真性特異点に近付けて取っていったときに台形公式の精度がどのようになるのかを調べるためである。

数値実験③

「留数定理を必要とする計算について複合台形公式で数値計算をし、近似値を求める」をテーマに実験を行う。

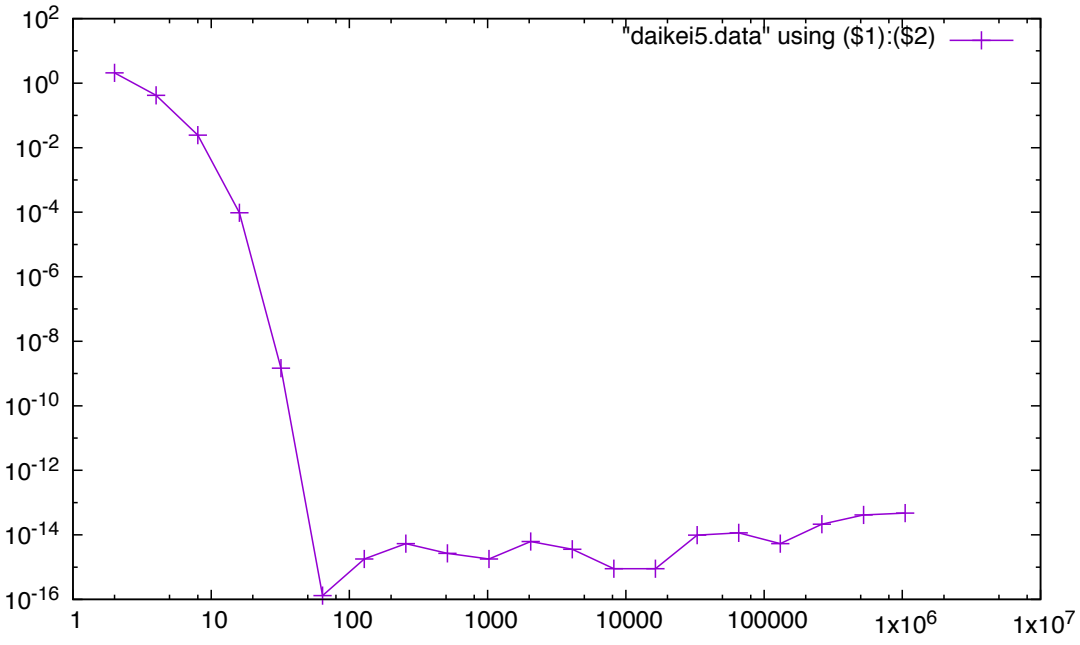
$\oint_{|z|=10} \frac{z}{1-\cos z} dz = 2\pi i(\mathrm{Re}\,s(f;0) + \mathrm{Re}\,s(f;2\pi i) + \mathrm{Re}\,s(f;-2\pi i)) = 12\pi i$ の数値計算

4-2、結果

それぞれの実験結果について、数値データとグラフを掲載した。尚、グラフの見方に関しては、3、4ページと同じく、縦軸が真の解との誤差、横軸がNである。

①-1 $\oint_{|z-1|=2} \frac{dz}{z-2} = 2\pi i$ の数値計算の結果

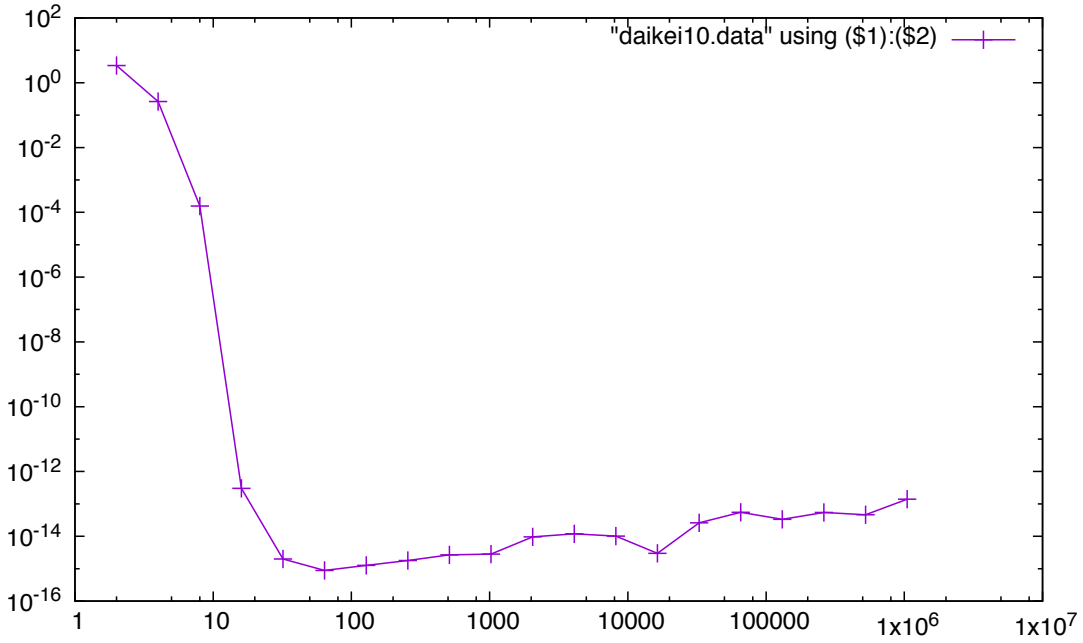
N	誤差
2	2.09
4	0.419
8	0.0246
16	9.59e-05
32	1.46e-09
64	1.31e-16
128	1.78e-15
256	5.33e-15
512	2.67e-15
1024	1.78e-15
2048	6.22e-15
4096	3.56e-15
8192	8.89e-16
16384	8.95e-16
32768	9.77e-15
65536	1.15e-14
131072	5.33e-15
262144	2.13e-14
524288	4.17e-14
1048576	4.71e-14



上記データから、N=64で誤差が10^{-16}程度の高精度な結果が得られた。

①-2　$\oint_{|z+i|=1} \frac{e^z}{z+i} dz = 2\pi(\sin 1 + i\cos 1)$ の数値計算の結果

刻み数	誤差
2	3.41
4	0.262
8	0.000156
16	3.01e-13
32	1.99e-15
64	8.88e-16
128	1.26e-15
256	1.78e-15
512	2.66e-15
1024	2.81e-15
2048	9.61e-15
4096	1.19e-14
8192	1.01e-14
16384	2.98e-15
32768	2.6e-14
65536	5.52e-14
131072	3.34e-14
262144	5.44e-14
524288	4.62e-14
1048576	1.4e-13



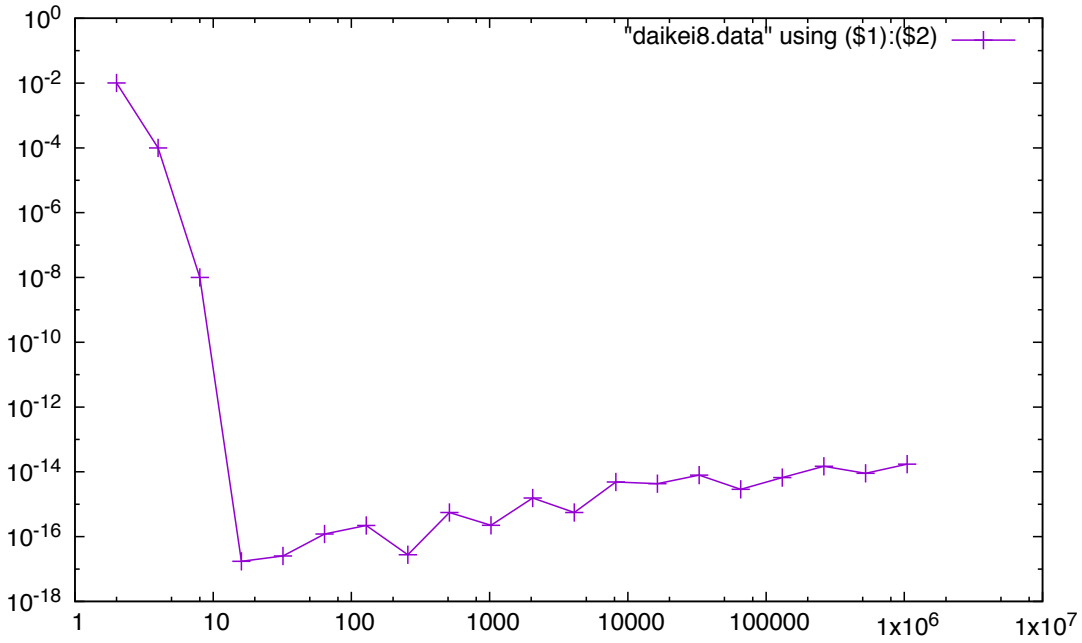
上記データから、N=64で誤差が10^{-16}程度の高精度な結果が得られた。

②-1　$f(z)=\dfrac{1}{(z-1)(z-2)}$ の場合の $\mathrm{Re}\,s(f;c)$ の数値計算の結果

$\mathrm{Re}\,s(f;1)=-1$ の計算

②-1-(i)　$r=0.1$

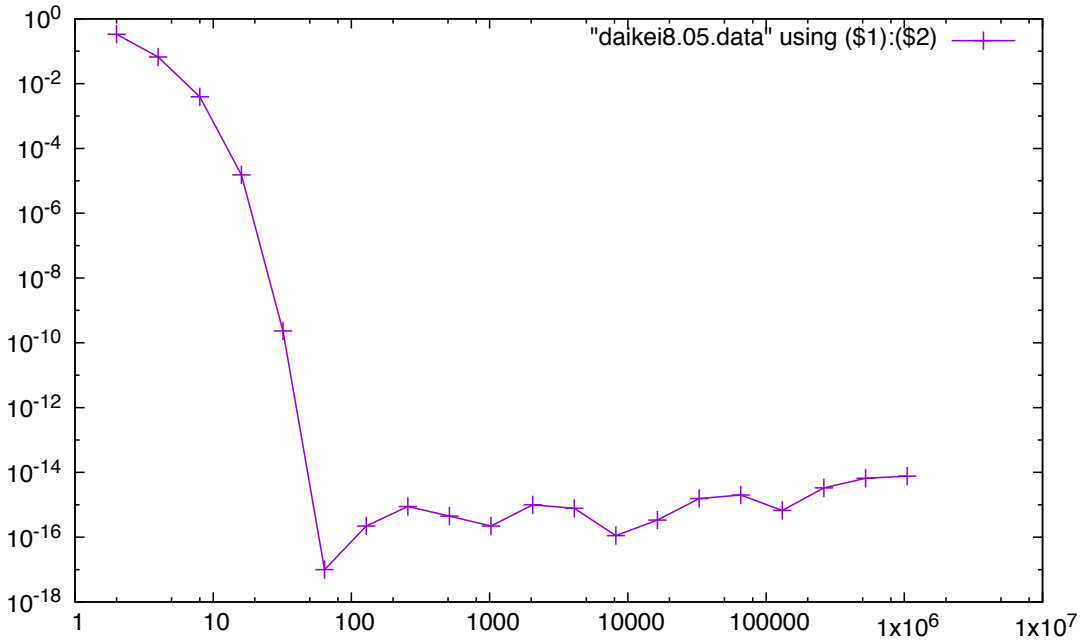
刻み数	誤差
2	0.0101
4	0.0001
8	1e-08
16	1.73e-17
32	2.52e-17
64	1.2e-16
128	2.22e-16
256	2.75e-17
512	5.56e-16
1024	2.24e-16
2048	1.55e-15
4096	5.56e-16
8192	4.88e-15
16384	4.33e-15
32768	7.88e-15
65536	2.89e-15
131072	6.66e-15
262144	1.49e-14
524288	8.99e-15
1048576	1.72e-14



上記データから、N=16で誤差が10^{-17}程度の高精度な結果が得られた。

②-1-(ii)　$r = 0.5$

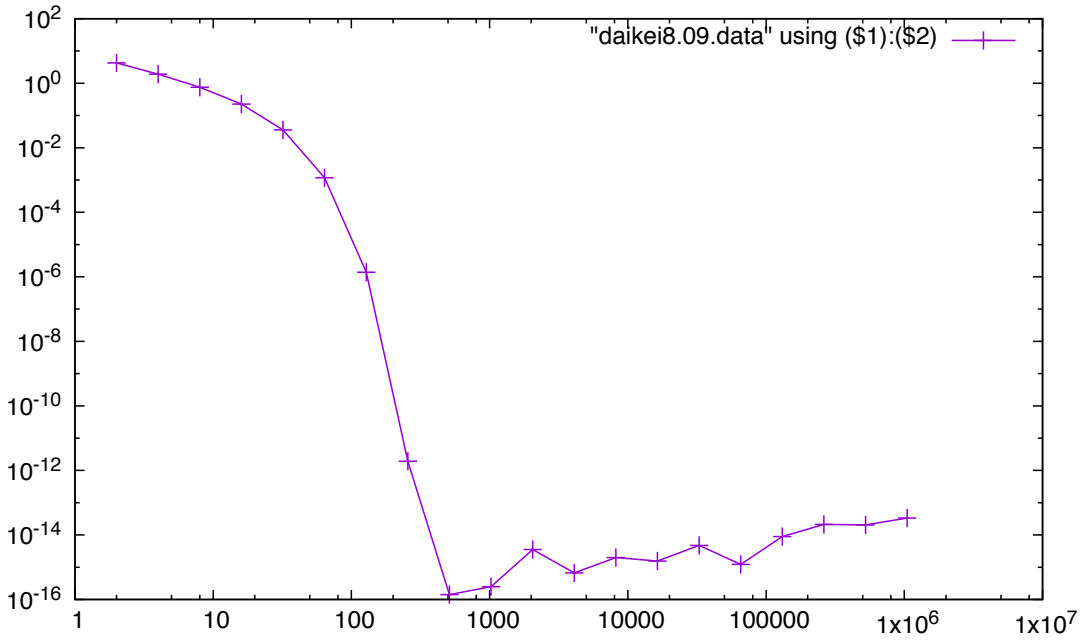
刻み数	誤差
2	0.333
4	0.0667
8	0.00392
16	1.53e-05
32	2.33e-10
64	9.97e-18
128	2.22e-16
256	8.89e-16
512	4.46e-16
1024	2.22e-16
2048	9.99e-16
4096	7.77e-16
8192	1.12e-16
16384	3.36e-16
32768	1.55e-15
65536	2e-15
131072	6.66e-16
262144	3.33e-15
524288	6.55e-15
1048576	7.66e-15



上記データから、N=64で、誤差が10^{-17}程度の高精度な結果が得られた。$r = 0.1$の場合の実験結果と比較すると、誤差の収束が遅くなっていることが確認できる。

②-1-(iii)　$r = 0.9$

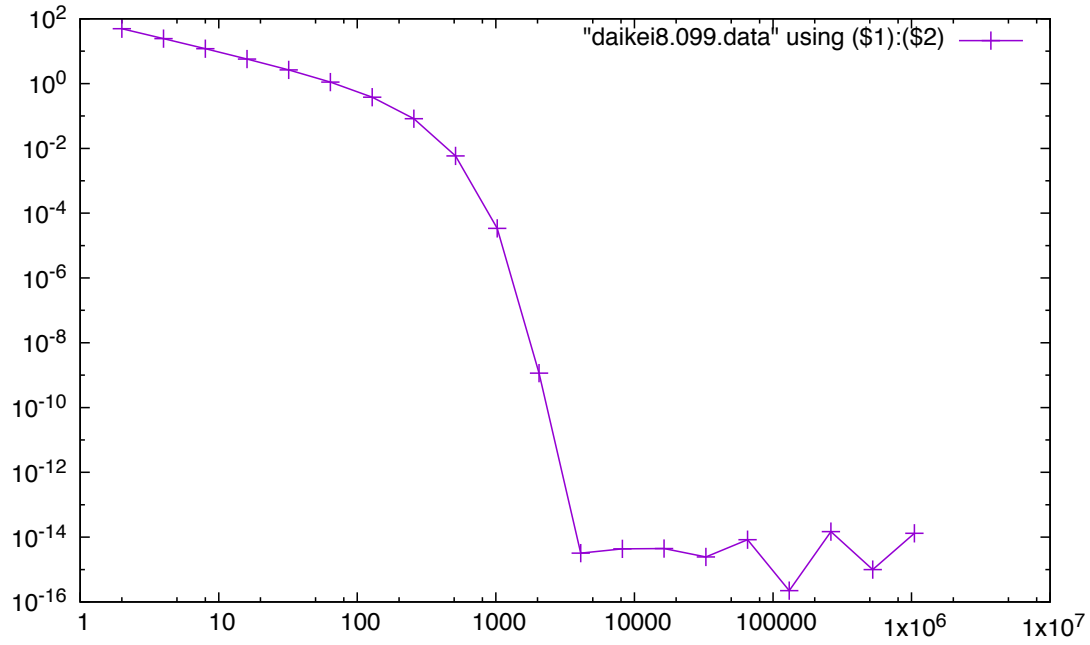
刻み数	誤差
2	4.26
4	1.91
8	0.756
16	0.227
32	0.0356
64	0.00118
128	1.39e-06
256	1.93e-12
512	1.41e-16
1024	2.48e-16
2048	3.55e-15
4096	6.68e-16
8192	2e-15
16384	1.55e-15
32768	4.77e-15
65536	1.23e-15
131072	8.88e-15
262144	2.11e-14
524288	2.05e-14
1048576	3.35e-14



上記データから、N=512で誤差が10^{-16}程度の高精度な結果が得られた。$r = 0.1$、$r = 0.5$の場合の実験結果と比較すると、誤差の収束が遅くなっていることが確認できる。

②-1-(iv)　$r = 0.99$

刻み数	誤差
2	49.3
4	24.4
8	11.9
16	5.73
32	2.64
64	1.11
128	0.382
256	0.0826
512	0.00586
1024	3.39e-05
2048	1.15e-09
4096	3.22e-15
8192	4.37e-15
16384	4.45e-15
32768	2.45e-15
65536	8.34e-15
131072	2.23e-16
262144	1.48e-14
524288	9.93e-16
1048576	1.31e-14

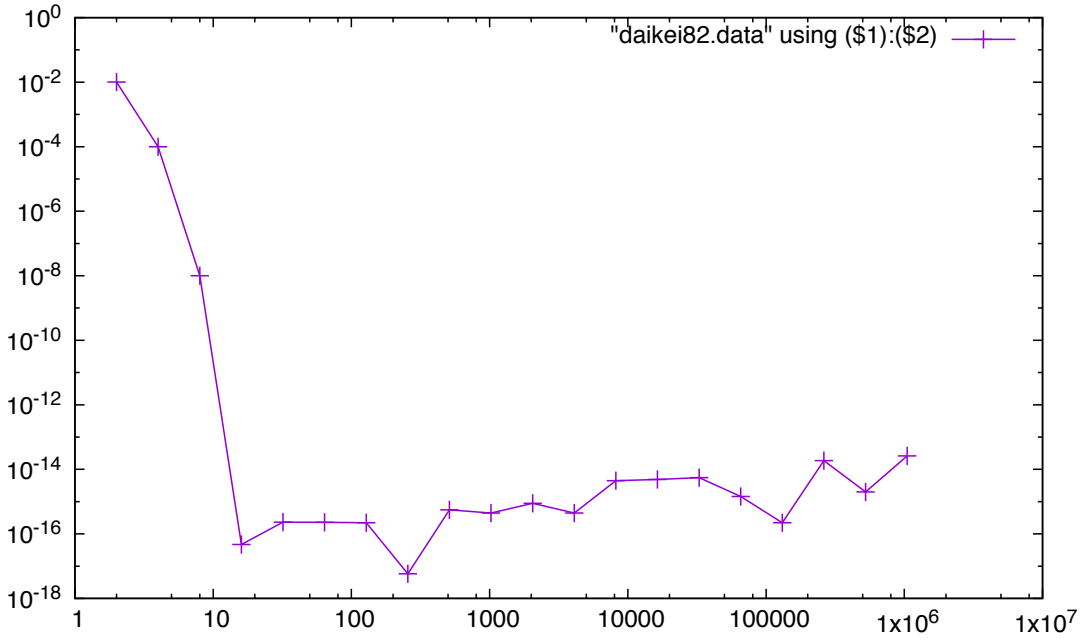


上記データから、N=131072で誤差が10^{-16}程度の高精度な結果が得られた。$r = 0.1$、$r = 0.5$、$r = 0.9$の場合の実験結果を比較すると、誤差の収束が遅くなっていることが確認できる。

$\mathrm{Re}\,s(f,2)=1$の計算

②-1-(i)　$r=0.1$

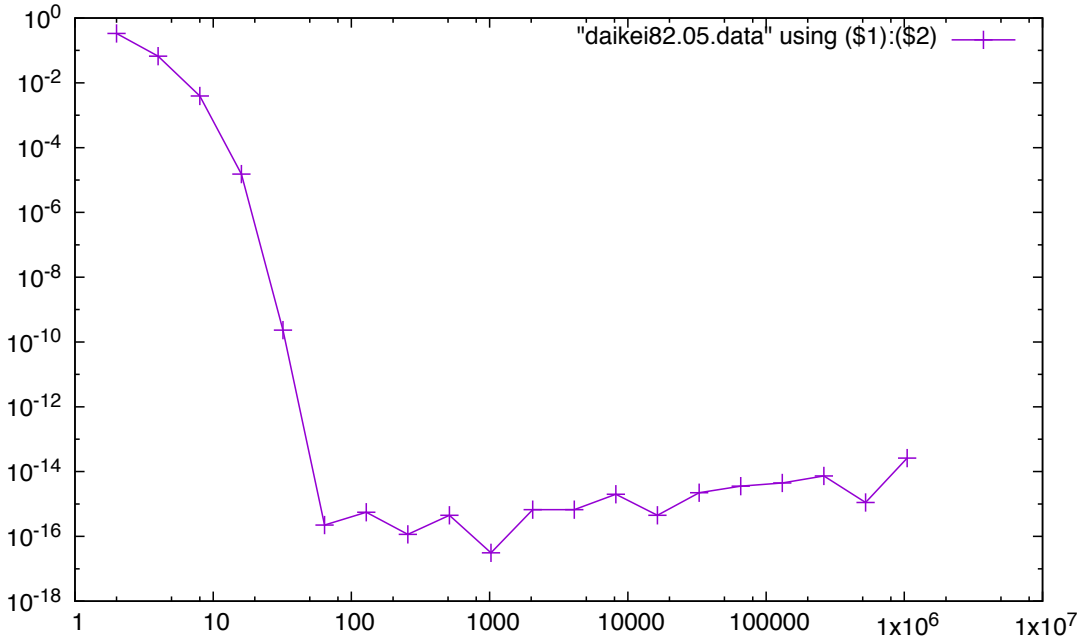
刻み数	誤差
2	0.0101
4	0.0001
8	1e-08
16	4.68e-17
32	2.3e-16
64	2.29e-16
128	2.22e-16
256	5.79e-18
512	5.55e-16
1024	4.45e-16
2048	8.88e-16
4096	4.45e-16
8192	4.44e-15
16384	4.88e-15
32768	5.55e-15
65536	1.44e-15
131072	2.22e-16
262144	1.87e-14
524288	2e-15
1048576	2.61e-14



上記データから、N=16で誤差が10^{-17}程度の高精度な結果が得られた。

②-1-(ii)　$r = 0.5$

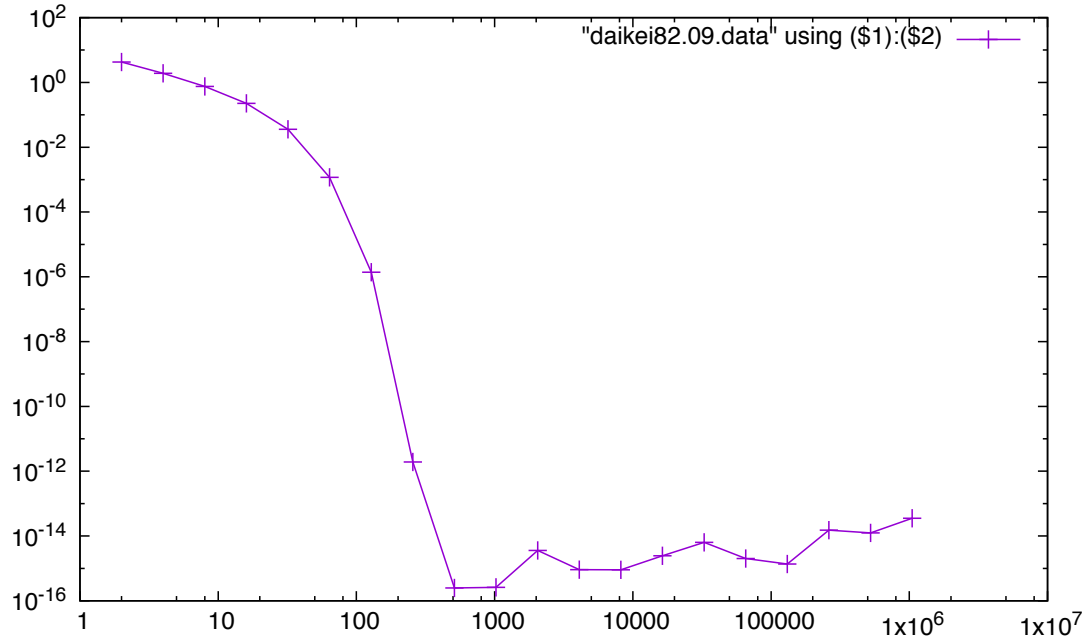
刻み数	誤差
2	0.333
4	0.0667
8	0.00392
16	1.53e-05
32	2.33e-10
64	2.23e-16
128	5.56e-16
256	1.16e-16
512	4.46e-16
1024	3.1e-17
2048	6.67e-16
4096	6.67e-16
8192	2e-15
16384	4.48e-16
32768	2.22e-15
65536	3.55e-15
131072	4.44e-15
262144	7.33e-15
524288	1.11e-15
1048576	2.62e-14



上記データから、N=64で誤差が10^{-16}程度の高精度な結果が得られた。$r = 0.1$の場合の実験結果と比較すると、誤差の収束が遅くなっていることが確認できる。

②-1(iii)　$r = 0.9$

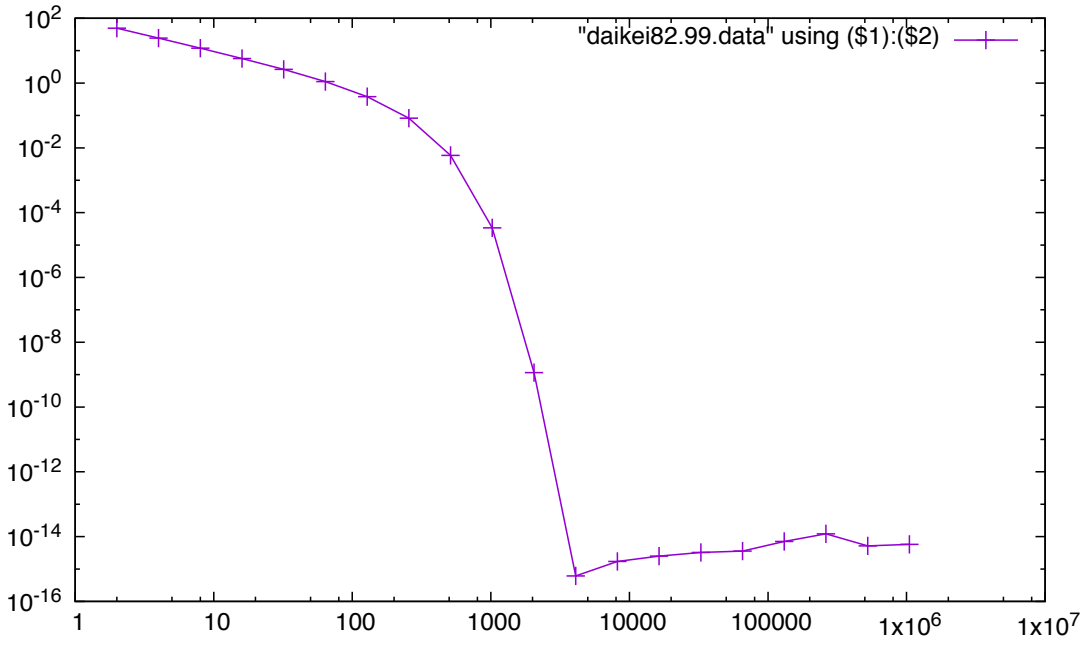
刻み数	誤差
2	4.26
4	1.91
8	0.756
16	0.227
32	0.0356
64	0.00118
128	1.39e-06
256	1.93e-12
512	2.48e-16
1024	2.61e-16
2048	3.58e-15
4096	9.09e-16
8192	9.02e-16
16384	2.45e-15
32768	6.33e-15
65536	2.01e-15
131072	1.36e-15
262144	1.51e-14
524288	1.24e-14
1048576	3.51e-14



上記データから、N=512で誤差が10^{-16}程度の高精度な結果が得られた。$r = 0.1$、$r = 0.5$の場合の実験結果と比較すると、誤差の収束が遅くなっていることが確認できる。

②-1-(iv)　$r = 0.99$

刻み数	誤差
2	49.3
4	24.4
8	11.9
16	5.73
32	2.64
64	1.11
128	0.382
256	0.0826
512	0.00586
1024	3.39e-05
2048	1.15e-09
4096	6.14e-16
8192	1.71e-15
16384	2.49e-15
32768	3.23e-15
65536	3.59e-15
131072	7.11e-15
262144	1.22e-14
524288	5.2e-15
1048576	5.75e-15

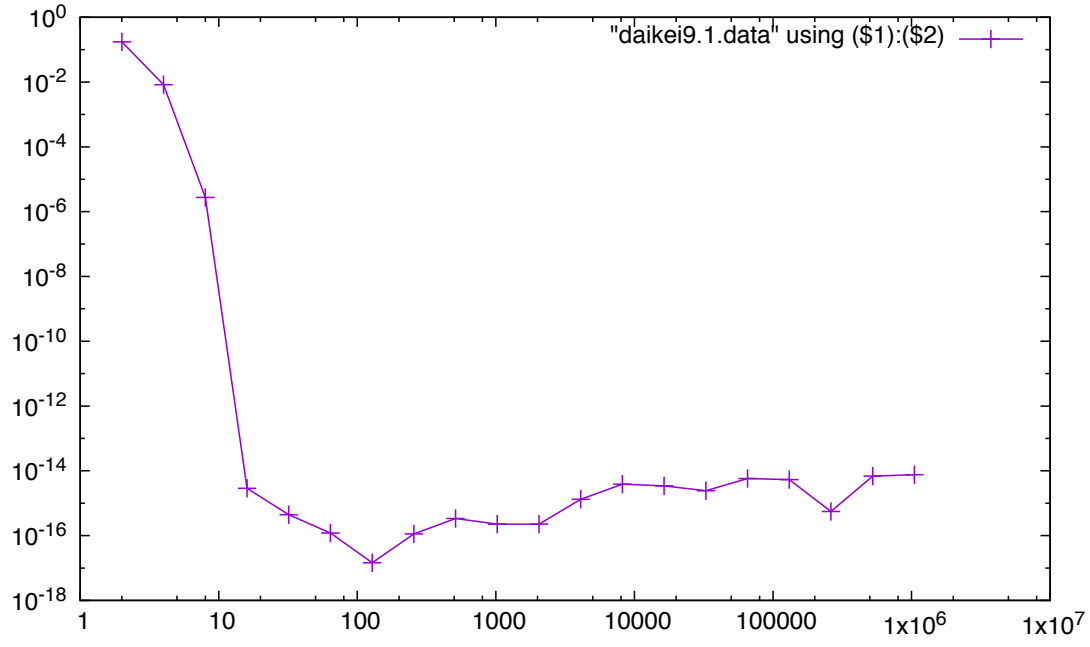


上記のデータから、N=4096で誤差が10^{-16}程度の高精度な結果が得られた。$r = 0.1$、$r = 0.5$、$r = 0.9$の場合の実験結果と比較すると、誤差の収束が遅くなっていることが確認できる。

②-2　$f(z)=e^{\frac{1}{z}}$ の場合の $\mathrm{Re}\,s(f;0)=1$ の数値計算の結果

②-2-(i)　$r=1$

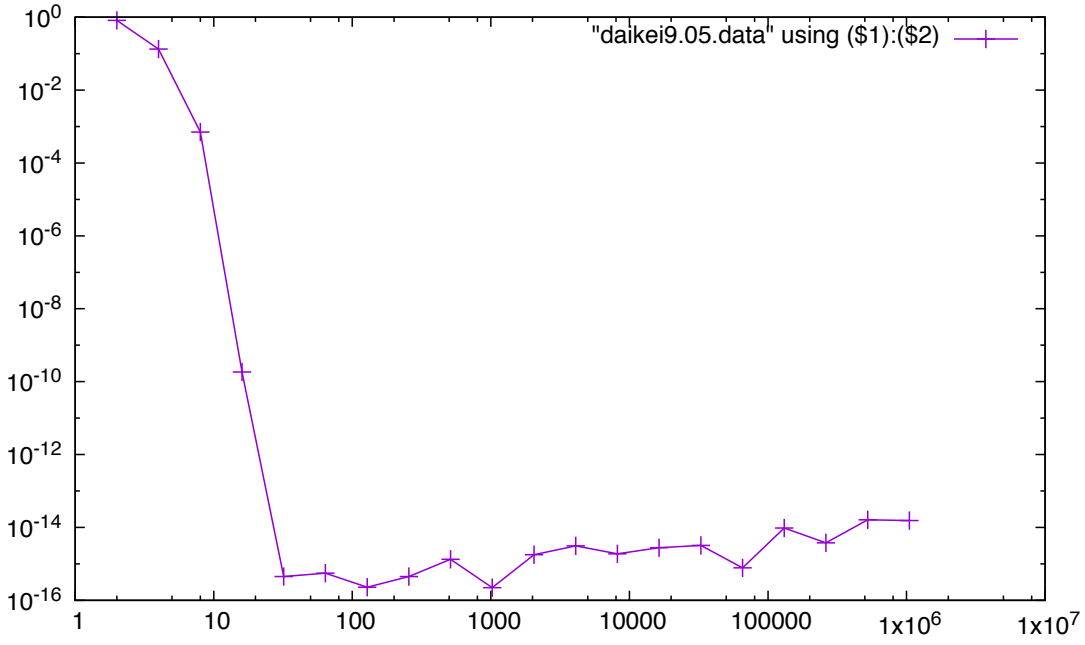
刻み数	誤差
2	0.175
4	0.00834
8	2.76e-06
16	2.89e-15
32	4.44e-16
64	1.2e-16
128	1.45e-17
256	1.13e-16
512	3.36e-16
1024	2.26e-16
2048	2.25e-16
4096	1.33e-15
8192	3.89e-15
16384	3.44e-15
32768	2.44e-15
65536	5.77e-15
131072	5.33e-15
262144	5.55e-16
524288	6.88e-15
1048576	7.55e-15



上記データから、N=32で誤差が10^{-16}程度の高精度な結果を得られた。

②-2-(ii)　$r = 0.5$

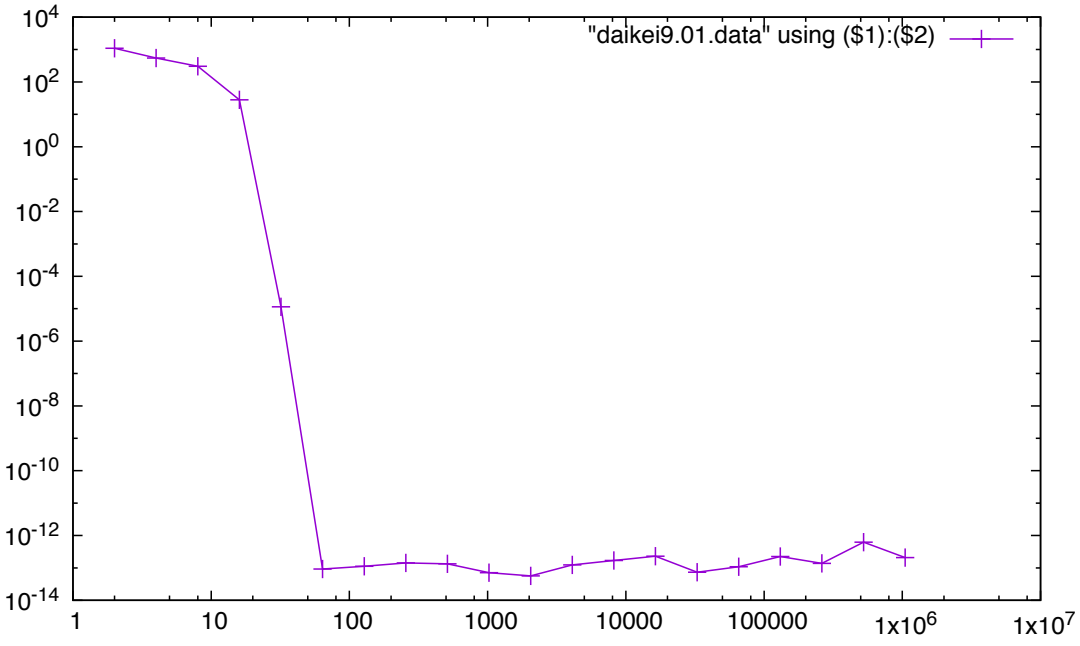
刻み数	誤差
2	0.813
4	0.134
8	0.000705
16	1.84e-10
32	4.48e-16
64	5.56e-16
128	2.29e-16
256	4.49e-16
512	1.33e-15
1024	2.22e-16
2048	1.78e-15
4096	3.11e-15
8192	1.89e-15
16384	2.78e-15
32768	3.22e-15
65536	7.78e-16
131072	9.66e-15
262144	3.78e-15
524288	1.62e-14
1048576	1.55e-14



上記データから、N=32で誤差が10^{-16}程度の高精度な結果が得られた。$r=1$の場合の実験結果と比較すると、N=16のとき、$r=1$では10^{-15}程度の精度なのに対し、　こちらは10^{-10}程度の精度のため、$r=1$の場合の方が、誤差の収束が速いことが確認できる。

②-2-(iii)　$r = 0.1$

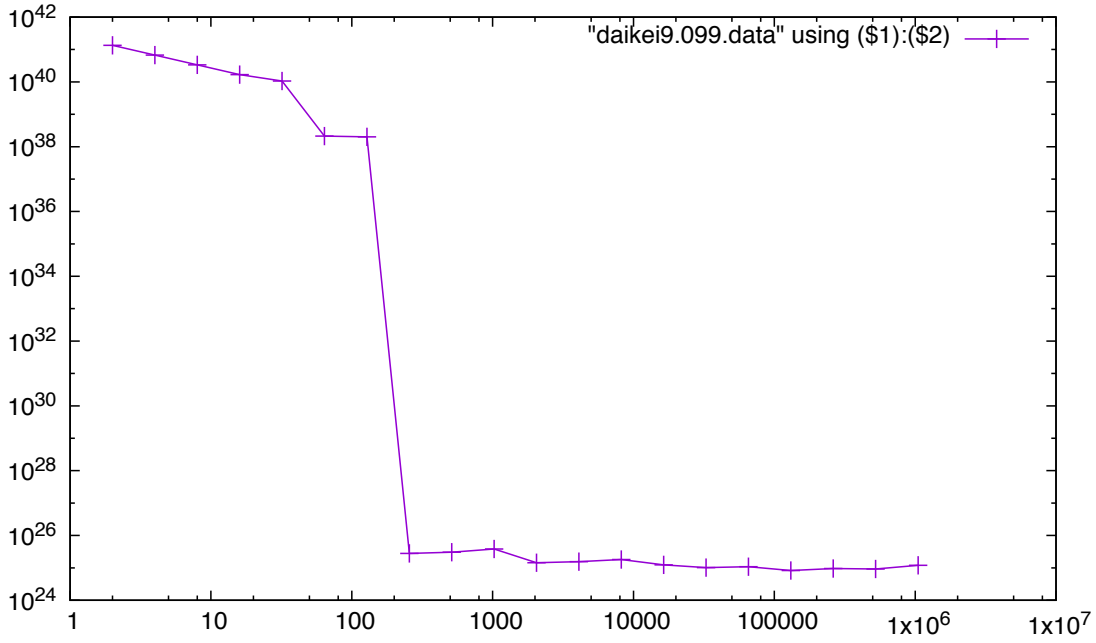
刻み数	誤差
2	1.1e+03
4	550
8	304
16	28.1
32	1.15e-05
64	9.26e-14
128	1.12e-13
256	1.42e-13
512	1.33e-13
1024	7.12e-14
2048	5.67e-14
4096	1.24e-13
8192	1.7e-13
16384	2.3e-13
32768	7.39e-14
65536	1.09e-13
131072	2.24e-13
262144	1.38e-13
524288	6.23e-13
1048576	2.09e-13



上記データから、N=64で誤差が10^{-14}程度の精度となったが、$r = 1$や$r = 0.5$の場合の実験で得られた10^{-16}程度の精度を得ることはできなかった。

②-2-(iv)　$r = 0.01$

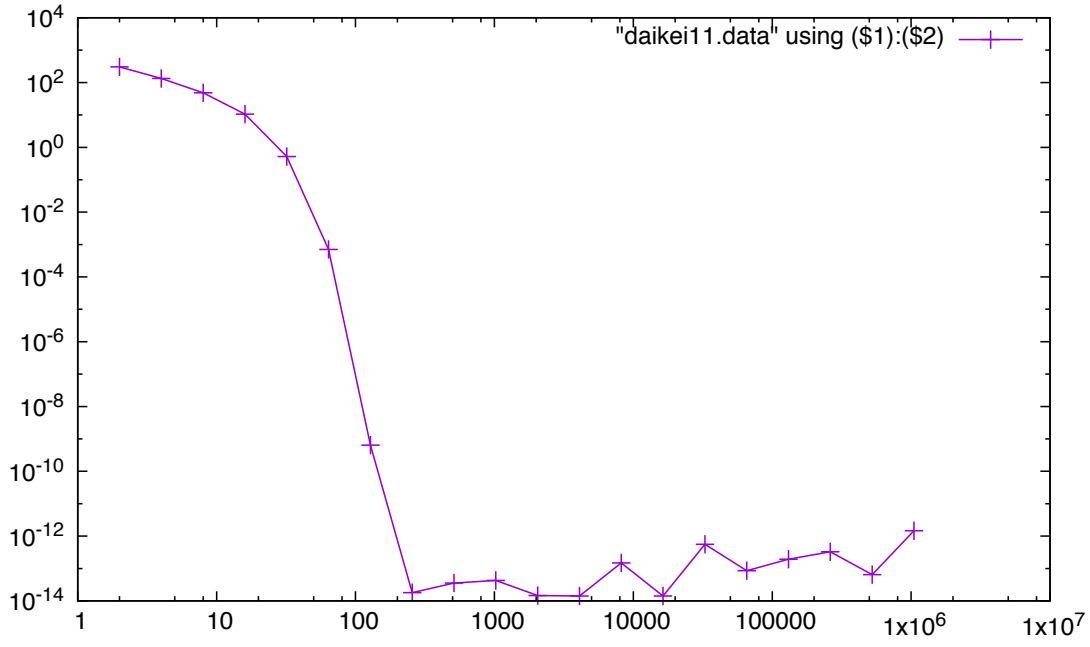
刻み数	誤差
2	1.34e+41
4	6.72e+40
8	3.36e+40
16	1.68e+40
32	1.06e+40
64	2.13e+38
128	2.01e+38
256	2.78e+25
512	3.04e+25
1024	3.83e+25
2048	1.44e+25
4096	1.55e+25
8192	1.81e+25
16384	1.24e+25
32768	1.02e+25
65536	1.09e+25
131072	8.31e+24
262144	9.55e+24
524288	9.25e+24
1048576	1.2e+25



上記データから、近似値は求まらないという結果が得られた。

]③ $\oint_{|z|=10} \frac{z}{1-\cos z} dz = 12\pi i$ の数値計算の結果

刻み数	誤差
2	304
4	133
8	48.1
16	10.6
32	0.522
64	0.000708
128	6.39e-10
256	1.79e-14
512	3.55e-14
1024	4.27e-14
2048	1.46e-14
4096	1.42e-14
8192	1.49e-13
16384	1.42e-14
32768	5.61e-13
65536	8.53e-14
131072	1.92e-13
262144	3.27e-13
524288	6.44e-14
1048576	1.8e-12



上記データから、N=256で10^{-14}程度の精度の結果が得られたが、10^{-16}程度の精度の結果までは得られなかった。

6、結論

まず、数値実験①の結果から、閉曲線のまわりの正則関数の積分について、複合台形公式による数値計算で誤差が10^{-16}程度の近似値が得られることが確認できた。
そして、数値実験②の結果から、留数についての計算にも複合台形公式による数値計算で誤差が10^{-16}の近似値が得られることが確認できた。ただ、②-1では、半径rを中心に取っていない特異点に近づけていくと、誤差の収束が遅くなった。また②-2では、半径rを真性特異点に近づけていくと、誤差の収束が遅くなり、$r=0.01$の場合では、近似値を得られなかった。留数について複合台形公式を用いて数値計算をするとき、特異点が複数存在する場合は、半径rを他の特異点に近付けずにとること、真性特異点のまわりでの留数を考える場合は、半径rを真性特異点に近付けずにとることが重要であるということが考察できる。
最後に、数値実験③の結果から、手で計算をする場合に留数定理を用いる積分計算についても複合台形公式による数値計算で近似値が求まることが確認できた。
今回の研究では、理論的な解析をすることはできなかったが、数値実験的に、閉曲線のまわりの正則関数の積分、特に留数の計算について、複合台形公式による数値計算は有用であるということが検証できた。

7、数値実験に使用したプログラム集（プログラムは全てC++を使用）

①-1 $\oint_{|z-1|=2} \frac{dz}{z-2} = 2\pi i$ の数値計算に使用したプログラム

```
#include <iostream>
#include <iomanip>
#include <complex>
using namespace std;

typedef complex<double> ddfunction(double);

double pi;
complex<double> i(0,1);


complex<double> trapezoidal(ddfunction F, double a, double b, int N)
{
    int j;
    double h;
    complex<double> T;
    h = (b - a) / N;
    T = (F(a) + F(b)) / 2.0;
    for (j = 1; j < N; j++) T += F(a + j * h);
    T *= h;
    return T;
}

complex<double> z(double t)
{
    complex<double> d(0,t);
    return 2.0 * exp(d) + 1.0;
}

complex<double> f(complex<double> z)
{
    return 1.0 / (z - 2.0);
}

complex<double> dzdt(double t)
{
    complex<double> d(0,t);
    return 2.0 * i * exp(d);
}

complex<double> F(double t)
{
    return f(z(t)) * dzdt(t);
}
```

```
int main(void)
{

    int m, N;
    double a, b, h;
    complex<double> I, IhN;



    pi = 4.0 * atan(1.0);
    a = 0;
    b = 2.0 * pi;


    printf("1 / z-2 の積分\n");

    I = complex<double>(0, 2 * pi);

    N = 2;

    for (m = 1; m <= 20; m++) {
        IhN = trapezoidal(F, a, b, N);
        cout << setprecision(3);
        cout << "" << N << " " << abs(IhN - I) << endl;
        N *= 2;
    }
    return 0;
}
```

---

①-2 $\oint_{|z+i|=1} \frac{e^z}{z+i} dz = 2\pi(\sin 1 + i\cos 1)$の数値計算に使用したプログラム

```
#include <iostream>
#include <iomanip>
#include <complex>
using namespace std;

typedef complex<double> ddfunction(double);


double pi = 4.0 * atan(1.0);
complex<double> i(0,1);


complex<double> trapezoidal(ddfunction F, double a, double b, int N)
{
    int j;
    double h;
    complex<double> T;
    h = (b - a) / N;
    T = (F(a) + F(b)) / 2.0;
```

```
    for (j = 1; j < N; j++) T += F(a + j * h);
    T *= h;
    return T;
}

complex<double> z(double t)
{
    return exp(i * t) - i;
}

complex<double> f(complex<double> z)
{
    return exp(z) / (z + i);
}

complex<double> dzdt(double t)
{
    return i * exp(i * t);
}

complex<double> F(double t)
{
    return f(z(t)) * dzdt(t);
}


int main(void)
{

    int m, N;
    double a, b, h;
    complex<double> I, IhN;



    a = 0;
    b = 2.0 * pi;
    I = complex<double>(2.0 * pi * sin(1.0), 2.0 * pi * cos(1.0));

    printf("exp(z)/(z+i)の積分\n");

    N = 2;

    for (m = 1; m <= 20; m++) {
        IhN = trapezoidal(F, a, b, N);
        cout << setprecision(3);
        cout << "" << N << " " << abs(IhN - I) << endl;
        N *= 2;
    }

    return 0;
}
```

---

②-1 $f(z)=\dfrac{1}{(z-1)(z-2)}$ の場合の $\mathrm{Re}\,s(f;1)=-1$ の数値計算に使用したプログラム

```
#include <iostream>
#include <iomanip>
#include <complex>
using namespace std;

typedef complex<double> cdfunction(double);
typedef complex<double> ccfunction(complex<double>);

double pi = 4.0 * atan(1.0);
complex<double> i(0,1);

complex<double> f(complex<double> z)
{
    return 1.0 / ((z - 1.0) * (z - 2.0));
}

complex<double> z(double t, complex<double> c, double r)
{
    return r * exp(i * t) + c;
}

complex<double> dzdt(double t, double r)
{
    return r * i * exp(i * t);
}

complex<double> trape_residue(ccfunction f, complex<double> c, double r,
int N)
{
    int j;
    double h;
    complex<double> T;
    h = (2.0 * pi) / N;
    T = f(z(0, c, r)) * dzdt(0, r);
    for (j = 1; j < N; j++) T += f(z(j * h, c, r)) * dzdt(j * h, r);
    T *= h / (2.0 * i * pi);
    return T;
}

int main(void)
{
    int m, N;
    double r, I;
    complex<double> IN, c;

    I = -1;
    c = 1;
    cout << "r=" ; cin >> r;
    N = 2;
```

```
    for (m = 1; m <= 20; m++) {
        IN = trape_residue(f, c, r, N);
        cout << setprecision(3);
        cout << "" << N << " " << abs(IN - I) << endl;
        N *= 2;
    }
    return 0;
}
```

---

②-1 $f(z)=\frac{1}{(z-1)(z-2)}$ の場合の $\mathrm{Re}\,s(f;2)=1$ の数値計算に使用したプログラム

```
#include <iostream>
#include <iomanip>
#include <complex>
using namespace std;

typedef complex<double> cdfunction(double);
typedef complex<double> ccfunction(complex<double>);

double pi = 4.0 * atan(1.0);
complex<double> i(0,1);

complex<double> f(complex<double> z)
{
    return 1.0 / ((z - 1.0) * (z - 2.0));
}

complex<double> z(double t, complex<double> c, double r)
{
    return r * exp(i * t) + c;
}

complex<double> dzdt(double t, double r)
{
    return r * i * exp(i * t);
}

complex<double> trape_residue(ccfunction f, complex<double> c, double r,
int N)
{
    int j;
    double h;
    complex<double> T;
    h = (2.0 * pi) / N;
    T = f(z(0, c, r)) * dzdt(0, r);
    for (j = 1; j < N; j++) T += f(z(j * h, c, r)) * dzdt(j * h, r);
    T *= h / (2.0 * i * pi);
    return T;
}
```

```
int main(void)
{

    int m, N;
    double r, I;
    complex<double> IN, c;
    I = 1.0;

    c = 2.0;
    cout << "r=" ; cin >> r;
    N = 2;

    printf("留数計算その１    1/(z-1)(z-2)");

    for (m = 1; m <= 20; m++) {
        IN = trape_residue(f, c, r, N);
        cout << setprecision(3);
        cout << "" << N << “ " << abs(IN - I) << endl;
        N *= 2;
    }
    return 0;
}
```

---

②-2　$f(z)=e^{\frac{1}{z}}$の場合の$\mathrm{Re}\,s(f,0)=1$の数値計算に使用したプログラム

```
#include <iostream>
#include <iomanip>
#include <complex>
using namespace std;

typedef complex<double> cdfunction(double);
typedef complex<double> ccfunction(complex<double>);

double pi = 4.0 * atan(1.0);
complex<double> i(0,1);

complex<double> f(complex<double> z)
{
    return exp(1.0 / z);
}

complex<double> z(double t, complex<double> c, double r)
{
    return r * exp(i * t) + c;
}

complex<double> dzdt(double t, double r)
{
    return r * i * exp(i * t);
}
```

```
complex<double> trape_residue(ccfunction f, complex<double> c, double r,
int N)
{
    int j;
    double h;
    complex<double> T;
    h = (2.0 * pi) / N;
    T = f(z(0, c, r)) * dzdt(0, r);
    for (j = 1; j < N; j++) T += f(z(j * h, c, r)) * dzdt(j * h, r);
    T *= h / (2.0 * i * pi);
    return T;
}

int main(void)
{

    int m, N;
    double r, I;
    complex<double> IN, c;
    I = 1.0;

    c = 0;
    cout << "r=" ; cin >> r;
    N = 2;

    printf("留数計算その2    exp(1/z)");

    for (m = 1; m <= 20; m++) {
        IN = trape_residue(f, c, r, N);
        cout << setprecision(3);
        cout << "" << N << " " << abs(IN - I) << endl;
        N *= 2;
    }
    return 0;
}
```

---

③ $\oint_{|z|=10} \frac{z}{1-\cos z} dz = 12\pi i$ の数値計算に使用したプログラム

```
#include <iostream>
#include <iomanip>
#include <complex>
using namespace std;

typedef complex<double> cdfunction(double);
typedef complex<double> ccfunction(complex<double>);

double pi = 4.0 * atan(1.0);
complex<double> i(0,1);
```

```
complex<double> f(complex<double> z)
{
    return z / (1.0 - cos(z));
}

complex<double> z(double t, double r, complex<double> c)
{
    return r * exp(i * t) + c;
}

complex<double> dzdt(double t, double r)
{
    return r * i * exp(i * t);
}

complex<double> trapezoidal(ccfunction f, complex<double> c, double r,
int N)
{
    int j;
    double h;
    complex<double> T;
    h = (2.0 * pi) / N;
    T = f(z(0, r, c)) * dzdt(0, r);
    for (j = 1; j < N; j++) T += f(z(j * h, r, c)) * dzdt(j * h, r);
    T *= h;
    return T;
}


int main(void)
{

    int m, N, r;
    complex<double> I, IhN, c;

    c = 2.0 * pi;
    r = 10;
    N = 2;
    I = 12 * pi * i;

    printf("z/(1-cosz)の積分\n");


    for (m = 1; m <= 20; m++) {
        IhN = trapezoidal(f, c, r, N);
        cout << setprecision(3);
        cout << "" << N << " " << abs(IhN - I) << endl;
        N *= 2;
    }

    return 0;
}
```

```
./a.out > ファイル名.data


$ gnuplot
set logscale
set format y “10^{%L}”
plot “ファイル名.data” using ($1):($2) with linespoints
set term pdf
set output “ファイル名.pdf”
replot
```

8、参考文献

[1] 桂田祐史『複素関数講義ノート(2016年度)』

[2] 桂田祐史『応用複素関数講義ノート(2016年度)』

[3] 馬場敬之『スバラシク実力がつく！複素関数キャンパスゼミ』（マセマ出版社）