プログラミングで数学の世界を冒険だ!

2026 年 1 月 24 日(土)

目次

1 はじめに

1.1 目標

プログラミング言語 SageMath を用いて数学の世界を一緒に冒険しましょう. 数学の中でも微分積分学の数列と無限級数を数値実験します. 数値実験をして計算式の意味を汲み取り、更にはその神秘性を感知して欲しいです. つまり実験結果を見てどうしてこんな不思議な性質があるのだろうか？と疑問を持って貰えたら幸いです.

1.2 プログラミング初心者の方へ

SageMath は数学の幅広い処理を扱うプログラミング言語で, Python とほぼ同じ文法です. SageMath の使い方は本稿でも説明しますが、詳しく勉強したい人は Python の入門書を読むと良いでしょう． 基本に的を絞った(滝澤成人, 2018) または (三谷純, 2021) があります.

SageMath を動かす環境は CoCalc というオンライン環境(https://cocalc.com)をお勧めします. ブラウザで動作するのでアプリをインストール必要がありません. そしてなんと無料で使えるので安心して下さい.

プログラミング初心者のためにスキルアップのコツを箇条書きします.

- まず教科書を写経して自分の手元でプログラムを動かす.
- 小さいプログラムを沢山書いて沢山動かす.
- 分からない部分は人工知能に質問する.
- 初心者のうちはバイブコーディングを使わない.

プログラムを動かすときは事前に出力を予想して下さい. 予想が外れた場合は不安にならずじっくり考察して下さい. 理屈で考えても分からないときは、プログラムの 1 部を変更したり不要なコードを除去してから再度動かしてみて下さい.

2 SageMath の電卓モード

2.1 チュートリアル

まずは SageMath を電卓として使いましょう. コマンドラインで

```
$ sage
```

と入力し 10 秒ほど待ちます. すると SageMath が電卓モードで起動します. 試しに$9+6=15$を計算しましょう.

```
sage: 9+6
15
```

次に電卓を終了させましょう.

```
sage: exit()
$
```

つづいて引算, 掛算, 割算をします.

```
sage: 9-6
3
sage: 9*6
54
sage: 9/6
3/2
```

ここで$9/6$が有理数$\frac{3}{2}$と認識されていることに注意して下さい. 浮動小数点の$1.5000...$として認識させたい場合は, 関数 n を使います.

```
sage: n(9/6)
1.50000000000000
```

更に関数 n は引数 digits で有効桁数を指定できます.

```
sage: n(100000 / 17, digits=20)
5882.3529411764705882
```

次は整数 17 を 3 で割ると余りが 2 になることを計算します.

```
sage: 17 % 3
2
```

次は$2^8 = 256$を計算します. 以下の 2 通りの記法法があります.

```
sage: 2^8
256
sage: 2**8
256
```

続いて$\sqrt{2} = 1.41421356237...$を計算します.

```
sage: n(sqrt(2))
1.41421356237310
```

sqrt は square root の略です.

2.2 電卓モードで遊ぼう

以下の演習問題を解くことで SageMath の計算を特訓します. 雑談は興味が無ければ読み飛ばしてもかまいません.

問題: $\frac{297}{210}$を数値計算せよ.
解答:

```
sage: n(297/210)
1.41428571428571
```

雑談: $\sqrt{2}$に近い値です. コピー用紙の A4 サイズは 210mm × 297mm と国際規格に基づいて定められています. 実は縦横比がほぼ$1 : \sqrt{2}$となるように設計されています.

問題: 0.841×1.189を数値計算せよ.
解答:

```
sage: 0.841 * 1.189
0.999949000000000
```

雑談: 1に近い値です. コピー用紙の A0 サイズは 841mm × 1189mm と定められています. その面積はほぼ 1 平方メートルとなるよう設計されています.

問題: $\frac{257}{182}$を数値計算せよ.
解答:

```
sage: n(257/182)
1.41208791208791
```

雑談: $\sqrt{2}$に近い値です. コピー用紙の B5 サイズは 182mm × 257mm と規格で定められています. 実は縦横比がほぼ$1 : \sqrt{2}$となるように設計されています.

問題: 1.030×1.456を数値計算せよ.
解答:

```
sage: 1.030 * 1.456
1.49968000000000
```

雑談: 1.5に近い値です. コピー用紙の B0 サイズは 1030mm × 1456mm と規格で定められています. その面積はほぼ 1.5 平方メートルになるよう設計されています.

問題: $1 + \frac{24}{60} + \frac{51}{60^2} + \frac{10}{60^3}$を数値計算せよ.
解答:

```
sage: n(1 + 24/60 + 51/60^2 + 10/60^3)
1.41421296296296
```

雑談: $\sqrt{2}$に近い値です. この値はバビロニアの粘土板 YBC7289 に刻印されています.

問題: $1+\frac{1}{3}+\frac{1}{3\times4}-\frac{1}{3\times4\times34}$を数値計算せよ.
解答:

```
sage: n(1 + 1/3 + 1/(3*4) - 1/(3*4*34))
1.41421568627451
```

雑談: $\sqrt{2}$に近い値です. 古代インドの数学者がこの値を算出しました.

問題: $3\frac{10}{71}$及び$3\frac{1}{7}$を数値計算せよ.
解答:

```
sage: n(3 + 10/71)
3.14084507042254
sage: n(3 + 1/7)
3.14285714285714
```

雑談: ともに円周率πに近い値です. アルキメデスは$3\frac{10}{71}<\pi<3\frac{1}{7}$を示しました.

問題: $\frac{355}{113}$を数値計算せよ.
解答:

```
sage: n(355/113)
3.14159292035398
```

雑談: 円周率πに近い値です. 祖沖之(そちゅうし)がこの値を算出しました.

問題: 142857 に 1,2,3,4,5,6 を順に掛けなさい.
解答:

```
sage: 142857 * 1
142857
sage: 142857 * 2
285714
sage: 142857 * 3
428571
sage: 142857 * 4
571428
sage: 142857 * 5
714285
sage: 142857 * 6
857142
```

雑談: 不思議なことに各積では 1,4,2,8,5,7 の 6 つの数字が巡回します.

問題: $\frac{1}{7}$を有効桁数 50 で数値計算せよ.
解答:

```
sage: n(1/7, digits=50)
0.14285714285714285714285714285714285714285714285714
```

雑談: 循環小数の中に前問の 142857 が出てきます.

問題: $1^2, 2^2, 3^2, 4^2, 5^2, 6^2$を$13$で割った余りを求めよ.
解答:

```
sage: 1^2 % 13
1
sage: 2^2 % 13
4
sage: 3^2 % 13
9
sage: 4^2 % 13
3
sage: 5^2 % 13
12
sage: 6^2 % 13
10
```

雑談: この 1,3,4,9,10,12 を次の問題で使います.

問題: 076923 に 1,3,4,9,10,12 を順に掛けなさい.
解答:

```
sage: 076923 * 1
76923
sage: 076923 * 3
230769
sage: 076923 * 4
307692
sage: 076923 * 9
692307
sage: 076923 * 10
769230
sage: 076923 * 12
923076
```

雑談: 不思議なことに各積では 0,7,6,9,2,3 の 6 つの数字が巡回します.

問題: $\frac{1}{13}$を有効桁数 50 で数値計算せよ.

```
sage: n(1/13, digits=50)
0.076923076923076923076923076923076923076923076923077
```

雑談: 循環小数の中に前問の 076923 が出てきます.

問題: 076923 に 2,5,6,7,8,11 を順に掛けなさい.

```
sage: 076923 * 2
153846
sage: 076923 * 5
384615
sage: 076923 * 6
```

```
461538
sage: 076923 * 7
538461
sage: 076923 * 8
615384
sage: 076923 * 11
846153
```

雑談:不思議なことに各積では 1,5,3,8,4,6 の 6 つの数字が巡回します.

3 SageMath でプログラミング

3.1 チュートリアル

3.1.1 プログラムの作成と起動

プログラムを作成して動作させる手順を説明します. まず hello.sage というファイル名で以下のテキスト文章を作成します.

```
# hello.sage
print("Hello World")
print('Welcome to SageMath')
```

1 行目の #(シャープ)はコメントを表し, プログラムの挙動には一切関係ありません. 2 行目及び 3 行目は print 関数で文字列を表示させます. 文字列は'(シングルクォーテーション)または"(ダブルクォーテーション)で囲います. 早速コマンドラインでこの hello.sage を起動しましょう.

```
$ sage hello.sage
```

と入力して下さい. すると以下の出力が得られます.
出力:

```
Hello World
Welcome to SageMath
```

3.1.2 変数

変数の使い方を説明します. 通常数学では変数を 1 文字で表しますが, SageMath では 2 文字以上使っても OK です.

```
# variable.sage
alpha = 5 # 変数アルファに5を代入する
beta = 7 # 変数ベータに7を代入する

# 変数αと変数βを足して表示する
Sum = alpha + beta # 変数Sumに和を代入する
print(Sum)

# 変数αと変数βを掛けて表示する
```

```
Product = alpha * beta # 変数Productに積を代入する
print(Product)
```

出力:

```
12
35
```

3.1.3 for ループ

for ループは同じ処理を繰り返したいときに使います. 以下の for.sage では同じ処理を 5 回繰り返します.

```
# for.sage
for i in range(0, 5):
    print(i, "回目のHello World")
```

出力:

```
0 回目のHello World
1 回目のHello World
2 回目のHello World
3 回目のHello World
4 回目のHello World
```

変数 i が 0 から始まり 4 で終わっていることに注意して下さい.

3.1.4 辞書型

dict は辞書型を表し, その名の通り英和辞書のような物です. 例題として色を表す英和辞書を作ります.

```
# dict.sage
d = dict() # 空の辞書を作成する
# 英和辞書を作成する
d["blue"] = "青色"
d["red"] = "赤色"
d["green"] = "緑色"
d["yellow"] = "黄色"

# 辞書を引く
print("英語greenは日本語で", d["green"])
```

出力:

```
英語greenは日本語で 緑色
```

3.1.5 f 文字列

f 文字列を用いて数字を文字列に埋め込みます. f 文字列であることを明示するため文字列の直前に f を付けます.

```
# f-string_A.sage
x = 123.456789
# 浮動小数点xを文字列msgの一部に埋め込む
msg = f"変数xの値は{x}です"
print(msg)
```

出力

変数xの値は123.456789000000です

次に数字の文字幅を指定します.

```
# f-string_B.sage
x = 53

# 空白文字を含めて5文字で表現する
print(f"変数xの値は{x:5}です")

# ゼロを含めて5文字で表現する
print(f"変数xの値は{x:05}です")
```

出力:

変数xの値は　53です
変数xの値は00053です

次は小数点以下の桁数を指定します.

```
# f-string_C.sage
x = 123.456789

# 小数点以下4桁を表示する
print(f"変数xの値は約{x:.4f}です")
```

出力:

変数xの値は約123.4568です

四捨五入されていることに注意して下さい.

3.1.6 フィボナッチ数

これまでやってきたことのまとめとしてフィボナッチ数を計算します. フィボナッチ数は以下で定義される数列F_nです.

$F_0 = 0, F_1 = 1,$
$F_n = F_{n-1} + F_{n-2}, \quad n \geq 2$

```
# Fibonacci.sage
# フィボナッチ数を計算して出力する
# [1] 準備
N = 20 # 第20項まで計算する
```

```
# [2] フィボナッチ数F[n]を計算する
F = dict() # 空の辞書を作成する
F[0] = 0
F[1] = 1

# 漸化式を使う
for n in range(2, N+1):
    F[n] = F[n-1] + F[n-2]

# [3] フィボナッチ数F[n]を出力する
print("n, F[n]")
for n in range(0, N+1):
    print(f"{n}, {F[n]}")
```

出力:

```
n, F[n]
0, 0
1, 1
2, 1
3, 2
4, 3
5, 5
6, 8
7, 13
8, 21
9, 34
10, 55
11, 89
12, 144
13, 233
14, 377
15, 610
16, 987
17, 1597
18, 2584
19, 4181
20, 6765
```

ウェブサイト https://oeis.org/A000045 の結果と一致しているので Fibonacci.sage に不具合はなさそうです.

3.1.7 高精度の浮動小数点

高精度の浮動小数点を扱いたい場合は RealField を用います.

```
# RealField.sage
# Rは500ビットの浮動小数点の型
R = RealField(500)

# sqrt(2)を小数点以下10桁まで表示する
x = R(sqrt(2))
print(f"(square root of 2) = {x:.10f}")
```

出力:

```
(square root of 2) = 1.4142135624
```

3.1.8 グラフへプロット

数列を辞書型で作成しグラフへプロットします.

```
# plot.sage
# 数列を辞書で表現しグラフへプロットする
# [1] 数列a[n]を辞書で表現する
a = dict() # 空の辞書を作成する
a[1] = 3
a[2] = 5
a[3] = 7
a[4] = 9
a[5] = 11

# [2] 数列a[n]をグラフへプロットする
plt = list_plot(a)

# [3] グラフをPDFファイルで出力する
plt.save("output.pdf")
```

出力:

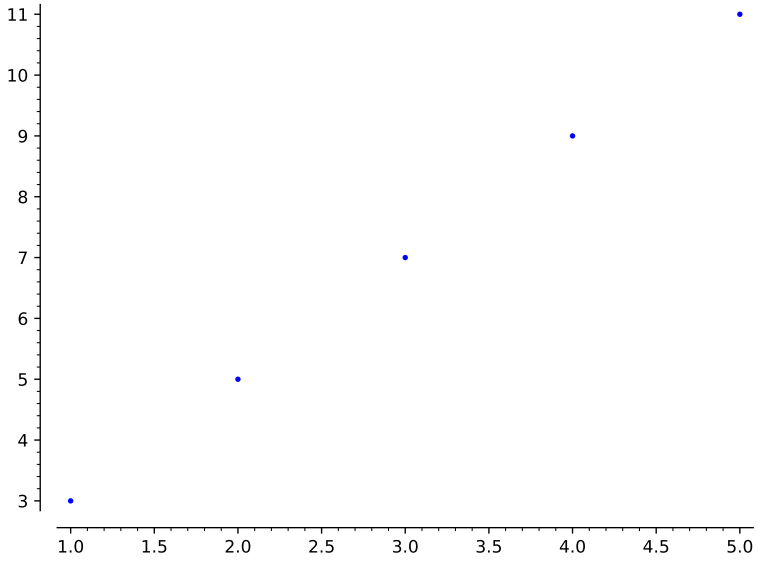

Figure 1: output.pdf

3.2 平方根$\sqrt{2} = 1.414213562373095048 80...$

方法 A,B の 2 通りの方法で$\sqrt{2}$を数値計算する.

3.2.1 方法 A

定理: 数列a_nを$a_0 = 1.5$

$$a_n = \frac{1}{2}\left(a_{n-1} + \frac{2}{a_{n-1}}\right), \quad n \geq 1$$

で定める. このとき, $\lim_{n\to\infty} a_n = \sqrt{2}$が成立する.

この定理を次の sqrt2_A.sage で数値計算する.

```
# sqrt2_A.sage
# sqrt(2)を計算して出力する
# [1] 準備
R = RealField(500) # 500ビットの浮動小数点を使う
N = 50 # 第50項まで計算する

# [2] 数列a[n]を計算する
a = dict() # 空の辞書を作成する
a[0] = R('1.5') # 初期値

# 漸化式を使う
for n in range(1, N+1):
    a[n] = R('0.5') * (a[n-1] + 2 / a[n-1])

# [3] 数列a[n]を出力する
print("n, a[n]")
for n in range(0, N+1):
    print(f"{n:02}, {a[n]:.50f}")
```

sqrt2_A.sage の出力

```
n, a[n]
00, 1.5000000000000000000000000000000000000000000000000
01, 1.4166666666666666666666666666666666666666666666667
02, 1.4142156862745098039215686274509803921568627450980
03, 1.4142135623746899106262955788901349101165596221157
04, 1.4142135623730950488016896235025302436149819257762
05, 1.4142135623730950488016887242096980785696718753772
06, 1.4142135623730950488016887242096980785696718753769
07, 1.4142135623730950488016887242096980785696718753769
08, 1.4142135623730950488016887242096980785696718753769
09, 1.4142135623730950488016887242096980785696718753769
10, 1.4142135623730950488016887242096980785696718753769
11, 1.4142135623730950488016887242096980785696718753769
12, 1.4142135623730950488016887242096980785696718753769
13, 1.4142135623730950488016887242096980785696718753769
14, 1.4142135623730950488016887242096980785696718753769
15, 1.4142135623730950488016887242096980785696718753769
16, 1.4142135623730950488016887242096980785696718753769
17, 1.4142135623730950488016887242096980785696718753769
18, 1.4142135623730950488016887242096980785696718753769
19, 1.4142135623730950488016887242096980785696718753769
20, 1.4142135623730950488016887242096980785696718753769
21, 1.4142135623730950488016887242096980785696718753769
22, 1.4142135623730950488016887242096980785696718753769
23, 1.4142135623730950488016887242096980785696718753769
24, 1.4142135623730950488016887242096980785696718753769
25, 1.4142135623730950488016887242096980785696718753769
26, 1.4142135623730950488016887242096980785696718753769
27, 1.4142135623730950488016887242096980785696718753769
28, 1.4142135623730950488016887242096980785696718753769
29, 1.4142135623730950488016887242096980785696718753769
30, 1.4142135623730950488016887242096980785696718753769
31, 1.4142135623730950488016887242096980785696718753769
32, 1.4142135623730950488016887242096980785696718753769
33, 1.4142135623730950488016887242096980785696718753769
34, 1.4142135623730950488016887242096980785696718753769
35, 1.4142135623730950488016887242096980785696718753769
36, 1.4142135623730950488016887242096980785696718753769
37, 1.4142135623730950488016887242096980785696718753769
38, 1.4142135623730950488016887242096980785696718753769
39, 1.4142135623730950488016887242096980785696718753769
40, 1.4142135623730950488016887242096980785696718753769
41, 1.4142135623730950488016887242096980785696718753769
42, 1.4142135623730950488016887242096980785696718753769
43, 1.4142135623730950488016887242096980785696718753769
44, 1.4142135623730950488016887242096980785696718753769
45, 1.4142135623730950488016887242096980785696718753769
46, 1.4142135623730950488016887242096980785696718753769
47, 1.4142135623730950488016887242096980785696718753769
48, 1.4142135623730950488016887242096980785696718753769
49, 1.4142135623730950488016887242096980785696718753769
50, 1.4142135623730950488016887242096980785696718753769
```

Figure 2: sqrt2_A.sage の出力

数列a_nが$\sqrt{2}$に収束している.

3.2.2 方法 B

定理: 数列p_n及びq_nを$p_0 = 1, p_1 = 3, q_0 = 1, q_1 = 2,$

$$p_n = 2p_{n-1} + p_{n-2}, \quad n \geq 2$$
$$q_n = 2q_{n-1} + q_{n-2}, \quad n \geq 2$$

で定める. このとき$\lim_{n\to\infty} \frac{p_n}{q_n} = \sqrt{2}$が成立する.

この定理を次の sqrt2_B.sage で数値計算をする.

```
# sqrt2_B.sage
# sqrt(2)を計算して出力する
# [1] 準備
R = RealField(500) # 500ビットの浮動小数点を使う
N = 50 # 第50項まで計算する

# [2] 数列p[n]及びq[n]を計算する
p = dict() # 空の辞書を作成する
p[0] = 1 # 初期値
p[1] = 3 # 初期値
q = dict() # 空の辞書を作成する
q[0] = 1 # 初期値
q[1] = 2 # 初期値

# 漸化式を使う
for n in range(2, N+1):
    p[n] = 2 * p[n-1] + p[n-2]
    q[n] = 2 * q[n-1] + q[n-2]

# [3] 数列p[n] / q[n] を出力する
print("n, p[n] / q[n]")
for n in range(0, N+1):
    print(f"{n:02}, {R(p[n] / q[n]):.50f}")
```

sqrt2_B.sage の出力

```
n, p[n] / q[n]
00, 1.0000000000000000000000000000000000000000000000000000000000
01, 1.5000000000000000000000000000000000000000000000000000000000
02, 1.4000000000000000000000000000000000000000000000000000000000
03, 1.4166666666666666666666666666666666666666666666666666666667
04, 1.4137931034482758620689655172413793103448275862069O
05, 1.4142857142857142857142857142857142857142857142857142857142
06, 1.4142011834319526627218934911242603550295857988165 7
07, 1.4142156862745098039215686274509803921568627450980 4
08, 1.4142131979695431472081218274111675126903553299492 4
09, 1.4142136248948696383515559293523969722455845248107 7
10, 1.4142135516460546943041282006619055913603901759275 4
11, 1.4142135642135642135642135642135642135642135642135642135642
12, 1.4142135620573204626281342458384387794746122351394 2
13, 1.4142135624272734024906538585328414745859226065212 5
14, 1.4142135623637995128829637225996667093962312524035 4
15, 1.4142135623746899106262955788901349101165596221157 4
16, 1.4142135623728214137728085694503949629142184009874 3
17, 1.4142135623731419971503638569934516673286665379107 3
18, 1.4142135623730869937385181134378317398249598610377 6
19, 1.4142135623730964308320372569747414537112549445344 3
20, 1.4142135623730948116827681384183157570065648528971 3
21, 1.4142135623730950894848637061937437797915388495107 6
22, 1.4142135623730950418215594180968295997433664867164 1
23, 1.4142135623730950499992895789028639393416700696941 5
24, 1.4142135623730950485962129021635714130473151050551 8
25, 1.4142135623730950488369428017932922115288997018134 9
26, 1.4142135623730950487956400807542599463542382401452 5
27, 1.4142135623730950488027265073587327389035632657441 1
28, 1.4142135623730950488015106687709282487817723993990 0
29, 1.4142135623730950488017192736932823969631777892360 6
30, 1.4142135623730950488016834827469619979965358813986 4
31, 1.4142135623730950488016896235025302436149819257762 0
32, 1.4142135623730950488016885699154411688709475669712 0
33, 1.4142135623730950488016887506824073717167076754125 5
34, 1.4142135623730950488016887196676992293861813835691 0
35, 1.4142135623730950488016887249889818805235790261884 2
36, 1.4142135623730950488016887240759941160297194623159 8
37, 1.4142135623730950488016887242326380518554792029313 1
38, 1.4142135623730950488016887242057622013947803231117 3
39, 1.4142135623730950488016887242103733683332138614139 1
40, 1.4142135623730950488016887242095822171633115114204 0
41, 1.4142135623730950488016887242097179572442920730792 6
42, 1.4142135623730950488016887242096946679283110531196 3
43, 1.4142135623730950488016887242096986637432166112185 5
44, 1.4142135623730950488016887242096979781697642825846 5
45, 1.4142135623730950488016887242096980957955726962891 4
46, 1.4142135623730950488016887242096980756141745426960 8
47, 1.4142135623730950488016887242096980790767550505499 7
48, 1.4142135623730950488016887242096980784826701570196 8
49, 1.4142135623730950488016887242096980785845990103475 3
50, 1.4142135623730950488016887242096980785671107839107 4
```

Figure 3: sqrt2_B.sage の出力

数列$\frac{p_n}{q_n}$が$\sqrt{2}$に収束している.

3.3 対数$\log 2 = 0.69314718055994530941\ldots$

方法 A,B の 2 通りの方法で$\log 2$を数値計算する.

3.3.1 方法 A

定理: メルカトル級数

$$\log 2 = \sum_{k=1}^{\infty} \frac{(-1)^{k-1}}{k} = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \frac{1}{6} + \frac{1}{7} - \ldots.$$

この定理を次の log2_A.sage で数値計算する. その際, 数列S_nを$n \geq 1$の範囲で

$$S_n = \sum_{k=1}^{n} \frac{(-1)^{k-1}}{k}$$

と定義する.

```
# log2_A.sage
# log2を計算して出力する
# [1] 準備
R = RealField(500) # 500ビットの浮動小数点を使う
N = 50 # 第50項まで計算する

# [2] 数列S[n]を計算する
S = dict() # 空の辞書を作成する
S[0] = R('0.0') # 初期値

for n in range(1, N+1):
    S[n] = S[n-1] + (-1)^(n-1) / n

# [3] 数列S[n]を出力する
print("n, S[n]")
for n in range(1, N+1):
    print(f"{n:02}, {S[n]:.50f}")
```

log2_A.sage の出力

```
n, S[n]
01, 1.00000000000000000000000000000000000000000000000000
02, 0.50000000000000000000000000000000000000000000000000
03, 0.83333333333333333333333333333333333333333333333333
04, 0.58333333333333333333333333333333333333333333333333
05, 0.78333333333333333333333333333333333333333333333333
06, 0.61666666666666666666666666666666666666666666666667
07, 0.75952380952380952380952380952380952380952380952381
08, 0.63452380952380952380952380952380952380952380952381
09, 0.74563492063492063492063492063492063492063492063492
10, 0.64563492063492063492063492063492063492063492063492
11, 0.73654401154401154401154401154401154401154401154401
12, 0.65321067821067821067821067821067821067821067821068
13, 0.73013375513375513375513375513375513375513375513376
14, 0.65870518370518370518370518370518370518370518370518
15, 0.72537185037185037185037185037185037185037185037185
16, 0.66287185037185037185037185037185037185037185037185
17, 0.72169537978361507773272479154832096008566596801891
18, 0.66613982422805952217716923599276540453011041246335
19, 0.71877140317542794322980081494013382558274199141072
20, 0.66877140317542794322980081494013382558274199141072
21, 0.71639045079447556227741986255918144463036103902977
22, 0.67093590533993010773196531710463599008490649357522
23, 0.71441416620949532512326966493072294660664562401001
24, 0.67274749954282865845660299826405627993997895734334
25, 0.71274749954282865845660299826405627993997895734334
26, 0.67428596108129019691814145980251781840151741888180
27, 0.71132299811832723395517849683955485543855445591884
28, 0.67560871240404151966946421112526914115284017020455
29, 0.71009147102473117484187800422871741701490913572179
30, 0.67675813769139784150854467089538408368157580238846
31, 0.70901620220752687376660918702441634174609193142072
32, 0.67776620220752687376660918702441634174609193142072
33, 0.70806923251055717679691221732744664477639496172375
34, 0.67865746780467482385573574673921135065874790290022
35, 0.70722889637610339528430717531063992208731933147165
36, 0.67945111859832561750652939753286214430954155369387
37, 0.70647814562535264453355642455988917133656858072090
38, 0.68016235615166843400724063508620496081025279124721
39, 0.70580338179269407503288166072723060183589381688824
40, 0.68080338179269407503288166072723060183589381688824
41, 0.70519362569513309942312556316625499207979625591263
42, 0.68138410188560928989931603935673118255598673210311
43, 0.70463991583909766199233929517068467092807975535892
44, 0.68191264311182493471961202244341194365535248263165
45, 0.70413486533404715694183424466563416587757470485387
46, 0.68239573489926454824618207075259068761670513963648
47, 0.70367233064394539931001185798663324080819450133861
48, 0.68283899731061206597667852465329990747486116800527
49, 0.70324716057591818842565811649003460135241218841344
50, 0.68324716057591818842565811649003460135241218841344
```

Figure 4: log2_A.sage の出力

数列S_nの収束が遅すぎる. 追加実験すると

$$S[100] = \underline{0.6}88172179310195203244645882693484065938370302741 00$$
$$S[1000] = \underline{0.692}64743055982030966723105896592648170186535531202$$
$$S[10000] = \underline{0.6930}97183059945296917232371458165943076275134273 88$$
$$S[10^5] = \underline{0.69314}21805849453094159821214584265680753938843603 3$$
$$S[10^6] = \underline{0.69314}66805601953094172319964581765683255001343591 9$$

となる. $\log 2$に収束していることが分かる.

3.3.2 方法 B

定理:

$$\log 2 = \sum_{k=1}^{\infty} \frac{1}{k \cdot 2^k}$$
$$= \frac{1}{1 \cdot 2^1} + \frac{1}{2 \cdot 2^2} + \frac{1}{3 \cdot 2^3} + \frac{1}{4 \cdot 2^4} + \frac{1}{5 \cdot 2^5} + \frac{1}{6 \cdot 2^6} + \frac{1}{7 \cdot 2^7} + \cdots.$$

この定理を log2_B.sage で数値計算をする. その際, 数列S_nを$n \geq 1$の範囲で

$$S_n = \sum_{k=1}^{n} \frac{1}{k \cdot 2^k}$$

と定義する.

```
# log2_B.sage
# log2を計算して出力する
# [1] 準備
R = RealField(500) # 500ビットの浮動小数点を使う
N = 50 # 第50項まで計算する

# [2] 数列S[n]を計算する
S = dict() # 空の辞書を作成する
S[0] = R('0.0') # 初期値

for n in range(1, N+1):
    S[n] = S[n-1] + 1 / (n * 2^n)

# [3] 数列S[n]を出力する
print("n, S[n]")
for n in range(1, N+1):
    print(f"{n:02}, {S[n]:.50f}")
```

log2_B.sage の出力

```
n, S[n]
01, 0.50000000000000000000000000000000000000000000000000
02, 0.62500000000000000000000000000000000000000000000000
03, 0.66666666666666666666666666666666666666666666666667
04, 0.68229166666666666666666666666666666666666666666667
05, 0.68854166666666666666666666666666666666666666666667
06, 0.69114583333333333333333333333333333333333333333333
07, 0.69226190476190476190476190476190476190476190476190
08, 0.69275018601190476190476190476190476190476190476190
09, 0.69296719990079365079365079365079365079365079365079
10, 0.69306485615079365079365079365079365079365079365079
11, 0.69310924535533910533910533910533910533910533910534
12, 0.69312959040742243867243867243867243867243867243867
13, 0.69313898043146090021090021090021090021090021090021
14, 0.69314334008547875735375735375735375735375735375735
15, 0.69314537459068709068709068709068709068709068709069
16, 0.69314632826500349693709068709068709068709068709069
17, 0.69314677705291709987826715767892238480473774591422
18, 0.69314698898054296793382271323447794036029330146977
19, 0.69314708936731311596013850270816215088660909094346
20, 0.69314713705102893627263850270816215088660909094346
21, 0.69314715975756027927859088366054310326756147189584
22, 0.69314717059476842025870452002417946690392510825947
23, 0.69314717577778100942310669393722294516479467347686
24, 0.69314717826130787506438273560388961183146134014353
25, 0.69314717945340077057219523560388961183146134014353
26, 0.69314718002652235495095124521927422721607672475892
27, 0.69314718030246978446664858318223719017903968772188
28, 0.69314718043551586655457408541438004732189683057902
29, 0.69314718049974500963150363821610418525293131333764
30, 0.69314718053078909545201958873693751858626464667098
31, 0.69314718054581042730065633898895364761852271118711
32, 0.69314718055308638491483976489227396011852271118711
33, 0.69314718055661412193989839563327774799731058997498
34, 0.69314718055832611196676508408111782152672235468087
35, 0.69314718055915764997981461847006871438386521182372
36, 0.69314718055956186984726925324247539841164298960150
37, 0.69314718055975851735035529178040297442515650311501
38, 0.69314718055985425363475244212123613643173545048344
39, 0.69314718055990089438868951536420613843494057868856
40, 0.69314718055992363175623383857015401441150307868856
41, 0.69314718055993472315503594745110419781470429820076
42, 0.69314718055994013681397507202394893019007632201028
43, 0.69314718055994278069392208635022193902456033363819
44, 0.69314718055994407258980528653237806834141047568364
45, 0.69314718055994470418334818439920995378520387846142
46, 0.69314718055994501311497242792102989775227673851577
47, 0.69314718055994516429427790879340987033190813811683
48, 0.69314718055994523830914621713717923190735267750485
49, 0.69314718055994527456132661306065810288308061516429
50, 0.69314718055994529232489500706316274966118730461742
```

Figure 5: log2_B.sage の出力

数列S_nが$\log 2$に収束している.

3.4 ネイピア数$e = 2.718281828459045235 36\ldots$

方法 A,B,C,D の 4 通りの方法でネイピア数eを数値計算する.

3.4.1 方法 A

定義 ネイピア数eを

$$e = \lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n$$

で定義する.

この定義式を e_A.sage で数値計算をする. その際, 数列a_nを$n \geq 1$の範囲で

$$a_n = \left(1+\frac{1}{n}\right)^n$$

と定義する.

```
# e_A.sage
# ネイピア数eを計算して出力する
# [1] 準備
R = RealField(500) # 500ビットの浮動小数点を使う
N = 50 # 第50項まで計算する

# [2] 数列a[n]を計算する
a = dict() # 空の辞書を作成する

for n in range(1, N+1):
    a[n] = R((1+1/n)^n)

# [3] 数列a[n]を出力する
print("n, a[n]")
for n in range(1, N+1):
    print(f"{n:02}, {a[n]:.50f}")
```

e_A.sage の出力

```
n, a[n]
01, 2.0000000000000000000000000000000000000000000000000000
02, 2.2500000000000000000000000000000000000000000000000000
03, 2.3703703703703703703703703703703703703703703703703703
04, 2.4414062500000000000000000000000000000000000000000000
05, 2.4883200000000000000000000000000000000000000000000000
06, 2.5216263717421124828532235939643347050754458161865 6
07, 2.5464996970407131139479055738437458639075312400202 5
08, 2.5657845139503479003906250000000000000000000000000000
09, 2.5811747917131971819900315081167532159095488622957 2
10, 2.5937424601000000000000000000000000000000000000000000
11, 2.6041990118975308781817744553909617778905515982184 4
12, 2.6130352902246781602995330443548766467020798169505 2
13, 2.6206008878857322107930994787374649880248003890004
14, 2.6271515563008693883842367121090552216121189649698 5
15, 2.6328787177279190470443497891511876131053048152030 0
16, 2.6379284973665998587631122129781857665875577367842 2
17, 2.6424143751831096202574857116586844468956880036062 0
18, 2.6464258210976854673490605270517860168008192400085 8
19, 2.6500343266404449072632676129300099758062349272839 9
20, 2.6532977051444201339454307651519775390625000000000 0
21, 2.6562632139261049855383983572429240899618522554923 0
22, 2.6589698585377882029213537001360108323939495206752 9
23, 2.6614501186387814544955469044088252834580795308609 8
24, 2.6637312580685940367405399559557309012795297472665 3
25, 2.6658363314874199930406400339519781528339147548262 4
26, 2.6677849665337408840230069613300133011485855579403 2
27, 2.6695939778125744088223082194170053019726603258113 6
28, 2.6712778534408396403577865220737372674282453604895 1
29, 2.6728491439808013627537148911336088635500295417439 4
30, 2.6743187758702945960644354448555678214384083442150 2
31, 2.6756963059146878836782940105535274033322846844497 2
32, 2.6769901293781826846423269494124023993338622591479 7
33, 2.6782076512537812248664132880112340692207934249514 8
34, 2.6793554280957766646523480031601610517222763555657 5
35, 2.6804392861534696616966222206430941789571724799514 0
36, 2.6814644203008677059197539700298222866925282019420 1
37, 2.6824354773085307791240298855530561735212547865555 0
38, 2.6833566262745706536518076992060337978521770350082 5
39, 2.6842316184671013278968530530273009277433300625718 7
40, 2.6850638383899727315187097490623929084272239566583 1
41, 2.6858563475377501329642549659917953508720027556996 1
42, 2.6866119220325803210411820675012350409548772823564 1
43, 2.6873330851182890660249566073120155486580187351015 4
44, 2.6880221353133066697814757091527093865619436432509 9
45, 2.6886811708843331817075307950223371685583663413134 1
46, 2.6893121111897703648946733445588455379272890010103 6
47, 2.6899167153502694307958504111309055935595301588661 4
48, 2.6904965986289358793430442173773145808301239966142 0
49, 2.6910532468424151587562240637084228679828966441497 0
50, 2.6915880290736053938940873551532594852288282764078 9
```

Figure 6: e_A.sage の出力

数列a_nの収束が遅すぎる. 追加実験すると

$$a[100] = \underline{2.7}04813829421526093267194710807530833677938382781 00$$
$$a[1000] = \underline{2.716}92393223589245738308812194757718896431501883657$$
$$a[10000] = \underline{2.7181}45926825224864037664674913146536113822649220 72$$
$$a[10^5] = \underline{2.71826}82371744896680350648244260464479744469326778 2$$
$$a[10^6] = \underline{2.718280}46931937688381979970845435639275164502668251$$
$$a[10^7] = \underline{2.7182816}92544966271198550225777813273153508271284 40$$
$$a[10^8] = \underline{2.71828181}48676362176529772430091766921532384267806 4$$
$$a[10^9] = \underline{2.718281827}09990432237664402386033286282501316408962$$
$$a[10^{10}] = \underline{2.7182818283}231311439497940012972294998851799338839 7$$
$$a[10^{11}] = \underline{2.71828182844}5453826218116833092997577392137855102 63$$
$$a[10^{12}] = \underline{2.718281828457}6860944460591946141537298947220026331 6$$
$$a[10^{13}] = \underline{2.7182818284589}0932126886453154968619715311420767644$$
$$a[10^{14}] = \underline{2.71828182845903}1643951145176251073613457595381093 18$$
$$a[10^{15}] = \underline{2.718281828459043}8762193732418312906967848884785790 0$$

となる. ネイピア数eに収束していることが分かる.

3.4.2 方法 B

定理:

$$e = \sum_{k=0}^{\infty} \frac{1}{k!} = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \frac{1}{5!} + \frac{1}{6!} + \frac{1}{7!} + \ldots.$$

この定理を次の e_B.sage で数値計算をする. その際, 数列f_n及びS_nを$n \geq 0$の範囲で

$$f_n = n!, \quad S_n = \sum_{k=0}^{n} \frac{1}{k!}$$

と定義する.

```
# e_B.sage
# ネイピア数eを計算して出力する
# [1] 準備
R = RealField(500) # 500ビットの浮動小数点を使う
N = 50 # 第50項まで計算する

# [2] 数列S[n]を計算する
f = dict() # 空の辞書を作成する
f[0] = R('1.0') # 初期値
S = dict() # 空の辞書を作成する
S[0] = R('1.0') # 初期値
```

```
for n in range(1, N+1):
    f[n] = n * f[n-1]
    S[n] = S[n-1] + 1 / f[n]

# [3] 数列S[n]を出力する
print("n, S[n]")
for n in range(1, N+1):
    print(f"{n:02}, {S[n]:.50f}")
```

e_B.sage の出力

```
n, S[n]
01, 2.0000000000000000000000000000000000000000000000000
02, 2.5000000000000000000000000000000000000000000000000
03, 2.6666666666666666666666666666666666666666666666667
04, 2.7083333333333333333333333333333333333333333333333
05, 2.7166666666666666666666666666666666666666666666667
06, 2.7180555555555555555555555555555555555555555555556
07, 2.7182539682539682539682539682539682539682539682539
08, 2.7182787698412698412698412698412698412698412698413
09, 2.7182815255731922398589065255731922398589065255731
10, 2.7182818011463844797178130511463844797178130511463
11, 2.7182818261984928651595318261984928651595318261984
12, 2.7182818282861685639463417241195018972796750574528
13, 2.7182818284467590023145578701134256689812245367800
14, 2.7182818284582297479122875948272773669599066424463
15, 2.7182818284589944642854695764748674801584854494907
16, 2.7182818284590422590587934503278418622333966249310
17, 2.7182818284590450705160477958486050611789796352510
18, 2.7182818284590452267081174817108696833426231358243
19, 2.7182818284590452349287527283351994002986043726966
20, 2.7182818284590452353397844906664158861464034345402
21, 2.7182818284590452353593574317298071473772510089137
22, 2.7182818284590452353602471108690522047059258986580
23, 2.7182818284590452353602857925707585115463030677773
24, 2.7182818284590452353602874043083296076646521164906
25, 2.7182818284590452353602874687778324515093860784391
26, 2.7182818284590452353602874712574287147341835385141
27, 2.7182818284590452353602874713492656133721389999837
28, 2.7182818284590452353602874713525455026092088379085
29, 2.7182818284590452353602874713526586022380733150778
30, 2.7182818284590452353602874713526623722257021309834
31, 2.7182818284590452353602874713526624938382062863352
32, 2.7182818284590452353602874713526624976385970411900
33, 2.7182818284590452353602874713526624977537603973977
34, 2.7182818284590452353602874713526624977571475549332
35, 2.7182818284590452353602874713526624977572443308628
36, 2.7182818284590452353602874713526624977572470190831
37, 2.7182818284590452353602874713526624977572470917377
38, 2.7182818284590452353602874713526624977572470936496
39, 2.7182818284590452353602874713526624977572470936987
40, 2.7182818284590452353602874713526624977572470936999
41, 2.7182818284590452353602874713526624977572470936999
42, 2.7182818284590452353602874713526624977572470936999
43, 2.7182818284590452353602874713526624977572470936999
44, 2.7182818284590452353602874713526624977572470936999
45, 2.7182818284590452353602874713526624977572470936999
46, 2.7182818284590452353602874713526624977572470936999
47, 2.7182818284590452353602874713526624977572470936999
48, 2.7182818284590452353602874713526624977572470936999
49, 2.7182818284590452353602874713526624977572470936999
50, 2.7182818284590452353602874713526624977572470936999
```

Figure 7: e_B.sage の出力

e_A.sage より収束が速い.

3.4.3 方法 C

定理:

$$e = \frac{1}{\sum_{k=0}^{\infty} \frac{(-1)^k}{k!}}.$$

この定理を次の e_C.sage で数値計算する. その際, 数列f_n, A_n及びS_nを$n \geq 0$の範囲で

$$f_n = n!, \quad A_n = \sum_{k=0}^{n} \frac{(-1)^k}{k!}, \quad S_n = \frac{1}{A_n}$$

と定義する.

```
# e_C.sage
# ネイピア数eを計算して出力する
# [1] 準備
R = RealField(500) # 500ビットの浮動小数点を使う
N = 50 # 第50項まで計算する

# [2] 数列S[n]を計算する
f = dict() # 空の辞書を作成する
f[0] = R('1.0') # 初期値
A = dict() # 空の辞書を作成する
A[0] = R('1.0') # 初期値
S = dict() # 空の辞書を作成する
S[0] = R('1.0') # 初期値

for n in range(1, N+1):
    f[n] = n * f[n-1]
    A[n] = A[n-1] + (-1)^n / f[n]
    S[n] = 1 / A[n]

# [3] 数列S[n]を出力する
print("n, S[n]")
for n in range(1, N+1):
    print(f"{n:02}, {S[n]:.50f}")
```

e_C.sage の出力
n, S[n]
01, Infinity
02, 2.0000000000000000000000000000000000000000000000000000
03, 3.0000000000000000000000000000000000000000000000000000
04, 2.6666666666666666666666666666666666666666666666666667
05, 2.7272727272727272727272727272727272727272727272727273
06, 2.7169811320754716981132075471698113207547169811320754
07, 2.7184466019417475728155339805825242718446601941747572
08, 2.7182633317602642756016989145823501651722510618216 1
09, 2.7182836938934499910109666207227182836938934499910 1
10, 2.7182816576664037376372792913051392512590255445664 7
11, 2.7182818427778273384920361985403726496587915070036 1
12, 2.7182818273518744088075986743931517096224602330742 4
13, 2.7182818285384861664135778815996451660083959085656 9
14, 2.7182818284537281836930799163720348764381457017808 9
15, 2.7182818284593787158742820055731316238240389442053 7
16, 2.7182818284590255576129561867650242693628330632632 9
17, 2.7182818284590463316283282912135453199838412160385 7
18, 2.7182818284590451775163631742914084928093139944691 1
19, 2.7182818284590452382590981804451807354912191288419 1
20, 2.7182818284590452352219614301374920588826491041137 9
21, 2.7182818284590452353665869896759534242815427602190 9
22, 2.7182818284590452353600131006060233622176228209698 3
23, 2.7182818284590452353602989218699333649160534615000 3
24, 2.7182818284590452353602870126506037814702855169445 5
25, 2.7182818284590452353602874890193769648081162347247 7
26, 2.7182818284590452353602874706975010731412765917332 2
27, 2.7182818284590452353602874713760890691289373192514 2
28, 2.7182818284590452353602874713518537835579494361257 7
29, 2.7182818284590452353602874713526894830603972941645 9
30, 2.7182818284590452353602874713526616264103156988966 3
31, 2.7182818284590452353602874713526625250119312342278 5
32, 2.7182818284590452353602874713526624969306307487487 5
33, 2.7182818284590452353602874713526624977815792483087 2
34, 2.7182818284590452353602874713526624977565513512628 4
35, 2.7182818284590452353602874713526624977572664340355 8
36, 2.7182818284590452353602874713526624977572465706252 3
37, 2.7182818284590452353602874713526624977572471074741 6
38, 2.7182818284590452353602874713526624977572470933465 5
39, 2.7182818284590452353602874713526624977572470937088 0
40, 2.7182818284590452353602874713526624977572470936997 4
41, 2.7182818284590452353602874713526624977572470936999 6
42, 2.7182818284590452353602874713526624977572470936999 6
43, 2.7182818284590452353602874713526624977572470936999 6
44, 2.7182818284590452353602874713526624977572470936999 6
45, 2.7182818284590452353602874713526624977572470936999 6
46, 2.7182818284590452353602874713526624977572470936999 6
47, 2.7182818284590452353602874713526624977572470936999 6
48, 2.7182818284590452353602874713526624977572470936999 6
49, 2.7182818284590452353602874713526624977572470936999 6
50, 2.7182818284590452353602874713526624977572470936999 6

Figure 8: e_C.sage の出力

3.4.4 方法 D

定理: 数列p_n及びq_nを$p_0 = 1, p_1 = 3, q_0 = 1, q_1 = 1$

$$p_n = 2(2n-1)p_{n-1} + p_{n-2}, \quad n \geq 2$$
$$q_n = 2(2n-1)q_{n-1} + q_{n-2}, \quad n \geq 2$$

で定める. このとき$\lim_{n\to\infty} \frac{p_n}{q_n} = e$が成立する.

この定理を次の e_D.sage で数値計算をする.

```
# e_D.sage
# ネイピア数eを計算して出力する
# [1] 準備
R = RealField(500) # 500ビットの浮動小数点を使う
N = 50 # 第50項まで計算する

# [2] 数列p[n]及びq[n]を計算する
p = dict() # 空の辞書を作成する
p[0] = 1 # 初期値
p[1] = 3 # 初期値
q = dict() # 空の辞書を作成する
q[0] = 1 # 初期値
q[1] = 1 # 初期値

# 漸化式を使う
for n in range(2, N+1):
    p[n] = 2 * (2*n - 1) * p[n-1] + p[n-2]
    q[n] = 2 * (2*n - 1) * q[n-1] + q[n-2]

# [3] 数列p[n] / q[n] を出力する
print("n, p[n] / q[n]")
for n in range(0, N+1):
    print(f"{n:02}, {R(p[n] / q[n]):.50f}")
```

e_D.sage の出力
n, p[n] / q[n]
00, 1.0000000000000000000000000000000000000000000000000
01, 3.0000000000000000000000000000000000000000000000000
02, 2.714285714285714285714285714285714285714285714285
03, 2.718309859154929577464788732394366197183098591549305
04, 2.718281718281718281718281718281718281718281718281718
05, 2.718281828735695726684725523798993863674056056166735
06, 2.718281828458563411277850606202642376785584483618625
07, 2.718281828459045851404621084949961134721768973083795
08, 2.718281828459045234757560631479773329703773190735885
09, 2.718281828459045235360753230188480692633383946700755
10, 2.718281828459045235360287179900086259351744270348095
11, 2.718281828459045235360287471503357984170958205123075
12, 2.718281828459045235360287471352597036092056850785325
13, 2.718281828459045235360287471352662521984043872659335
14, 2.718281828459045235360287471352662497749516857440015
15, 2.718281828459045235360287471352662497757249242165535
16, 2.718281828459045235360287471352662497757247093175185
17, 2.718281828459045235360287471352662497757247093700075
18, 2.718281828459045235360287471352662497757247093699965
19, 2.718281828459045235360287471352662497757247093699965
20, 2.718281828459045235360287471352662497757247093699965
21, 2.718281828459045235360287471352662497757247093699965
22, 2.718281828459045235360287471352662497757247093699965
23, 2.718281828459045235360287471352662497757247093699965
24, 2.718281828459045235360287471352662497757247093699965
25, 2.718281828459045235360287471352662497757247093699965
26, 2.718281828459045235360287471352662497757247093699965
27, 2.718281828459045235360287471352662497757247093699965
28, 2.718281828459045235360287471352662497757247093699965
29, 2.718281828459045235360287471352662497757247093699965
30, 2.718281828459045235360287471352662497757247093699965
31, 2.718281828459045235360287471352662497757247093699965
32, 2.718281828459045235360287471352662497757247093699965
33, 2.718281828459045235360287471352662497757247093699965
34, 2.718281828459045235360287471352662497757247093699965
35, 2.718281828459045235360287471352662497757247093699965
36, 2.718281828459045235360287471352662497757247093699965
37, 2.718281828459045235360287471352662497757247093699965
38, 2.718281828459045235360287471352662497757247093699965
39, 2.718281828459045235360287471352662497757247093699965
40, 2.718281828459045235360287471352662497757247093699965
41, 2.718281828459045235360287471352662497757247093699965
42, 2.718281828459045235360287471352662497757247093699965
43, 2.718281828459045235360287471352662497757247093699965
44, 2.718281828459045235360287471352662497757247093699965
45, 2.718281828459045235360287471352662497757247093699965
46, 2.718281828459045235360287471352662497757247093699965
47, 2.718281828459045235360287471352662497757247093699965
48, 2.718281828459045235360287471352662497757247093699965
49, 2.718281828459045235360287471352662497757247093699965
50, 2.718281828459045235360287471352662497757247093699965

Figure 9: e_D.sage の出力

3.5 円周率$\pi = 3.14159265358979323846...$

方法 A,B,C,D,E,F の 6 通りの方法で円周率πを数値計算する.

3.5.1 方法 A

定理: ライプニッツ級数

$$\pi = 4\sum_{k=0}^{\infty}\frac{(-1)^k}{2k+1} = 4\left(1-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\frac{1}{11}+\frac{1}{13}-...\right).$$

この定理を次の pi_A.sage で数値計算をする. その際, 数列A_n及びS_nを$n \geq 0$の範囲で

$$A_n = \sum_{k=0}^{n}\frac{(-1)^k}{2k+1}, \quad S_n = 4A_n$$

と定義する.

```
# pi_A.sage
# 円周率πを計算して出力する
# [1] 準備
R = RealField(500) # 500ビットの浮動小数点を使う
N = 50 # 第50項まで計算する

# [2] 数列S[n]を計算する
A = dict() # 空の辞書を作成する
A[0] = R('1.0') # 初期値
S = dict() # 空の辞書を作成する
S[0] = 4 * A[0] # 初期値

for n in range(1, N+1):
    A[n] = A[n-1] + (-1)^n / (2*n + 1)
    S[n] = 4 * A[n]

# [3] 数列S[n]を出力する
print("n, S[n]")
for n in range(0, N+1):
    print(f"{n:02}, {S[n]:.50f}")
```

pi_A.sage の出力
n, S[n]
00, 4.0000000000000000000000000000000000000000000000000000
01, 2.6666666666666666666666666666666666666666666666666667
02, 3.4666666666666666666666666666666666666666666666666667
03, 2.8952380952380952380952380952380952380952380952380952
04, 3.3396825396825396825396825396825396825396825396825396
05, 2.9760461760461760461760461760461760461760461760461760
06, 3.2837384837384837384837384837384837384837384837384837
07, 3.0170718170718170718170718170718170718170718170718170
08, 3.2523659347188758953464835817776994247582482876600 5
09, 3.0418396189294022111359572659882257405477219718705 8
10, 3.2323158094055926873264334564644162167381981623467 7
11, 3.0584027659273318177612160651600683906512416406076 4
12, 3.2184027659273318177612160651600683906512416406076 4
13, 3.0702546177791836696130679170119202425030934924594 9
14, 3.2081856522619422903027230894257133459513693545284 6
15, 3.0791533941974261612704650249095843136933048383994 2
16, 3.2003655154095473733916771461217055258145169596115 5
17, 3.0860798011238330876773914318359912401002312453258 3
18, 3.1941879092319411957854995399440993482083393534339 4
19, 3.0916238066678386316829354373799967841057752508698 4
20, 3.1891847822775947292439110471360943450813850069674 0
21, 3.0961615264636412408718180238802803915930129139441 4
22, 3.1850504153525301297607069127691692804819018028330 3
23, 3.0999440323738067255053877638329990677159443560245 2
24, 3.1815766854350312153013061311799378432261484376571 7
25, 3.1031453128860116074581688762779770589124229474610 9
26, 3.1786170109992191546279801970326940400449841915921
27, 3.1058897382719464273552529243054213127717711464319 3
28, 3.1760651768684376554254283629019125408419465850284 3
29, 3.1082685666989461300016995493425905069436415002826 6
30, 3.1738423371907494086902241395065249331731496970039 7
31, 3.1103502736986859166267320760144614411096576335119 1
32, 3.1718887352371474550882705375529229795711960950503 7
33, 3.1121872426998340222524496420305349198697035577369 4
34, 3.1701582571925876454408554391319841952320223983166 5
35, 3.1138202290235735609338131856108574346686421166265 1
36, 3.1686147495715187664132652404053779826138475960785 6
37, 3.1152814162381854330799319070720446492805142627452 3
38, 3.1672294681862373811318799590200965973324623146932 8
39, 3.1165965567938323178407407185137674834084116817818 9
40, 3.1659792728432150338901234345631501994577943978312 7
41, 3.1778650175887768449253307311736706692767391590356
42, 3.1648453252882894491984154260585435375159092100212 1
43, 3.1188683137940365756351970352539458363664839226648 9
44, 3.1638121340187556767587925408719233644563715631143 3
45, 3.1198560900627117207148364969158794084124155191582 8
46, 3.1628668427508837637255891850879224191651036912012 9
47, 3.1207615795929890268834839219300276823229984280434 0
48, 3.1619986929950508825535870147135328369621736857753 6
49, 3.1215946525910104785131829743094924329217696453713 2
50, 3.1611986129870500824735790139134528289613736057673 6

Figure 10: pi_A.sage の出力

数列S_nの収束が遅すぎる. 追加実験すると

$$S[100] = \underline{3.1}51493401070990575252687870117716536303551896844 39$$
$$S[1000] = \underline{3.142}59165433954305090112773725220456615353825631696$$
$$S[10000] = \underline{3.1416}92643590543213460768320877940222544825752138 71$$
$$S[10^5] = \underline{3.14160}26534897939884601433645294403940409178314727 6$$
$$S[10^6] = \underline{3.141593}65358879323921264313327931538413467038375009$$

となる. 円周率πに収束していることが分かる.

3.5.2 方法 B

定理:

$$\pi = 2\sqrt{3}\sum_{k=0}^{\infty}\frac{(-1)^k}{(2k+1)\cdot 3^k}$$
$$= 2\sqrt{3}\left(1 - \frac{1}{3\cdot 3^1} + \frac{1}{5\cdot 3^2} - \frac{1}{7\cdot 3^3} + \frac{1}{9\cdot 3^4} - \frac{1}{11\cdot 3^5} + \frac{1}{13\cdot 3^6} - \ldots\right).$$

この定理を次の pi_B.sage で数値計算する. その際, 数列A_n及びS_nを$n \geq 0$の範囲で

$$A_n = \sum_{k=0}^{n}\frac{(-1)^k}{(2k+1)\cdot 3^k}, \quad S_n = 2\sqrt{3}A_n$$

と定義する.

```
# pi_B.sage
# 円周率πを計算して出力する
# [1] 準備
R = RealField(500) # 500ビットの浮動小数点を使う
N = 50 # 第50項まで計算する

# [2] 数列S[n]を計算する
A = dict() # 空の辞書を作成する
A[0] = R('1.0') # 初期値
S = dict() # 空の辞書を作成する
S[0] = 2 * R(sqrt(3)) * A[0] # 初期値

for n in range(1, N+1):
    A[n] = A[n-1] + (-1)^n / ((2*n + 1) * 3^n)
    S[n] = 2 * R(sqrt(3)) * A[n]

# [3] 数列S[n]を出力する
print("n, S[n]")
for n in range(0, N+1):
    print(f"{n:02}, {S[n]:.50f}")
```

pi_B.sage の出力

```
n, S[n]
00, 3.464101615137754587054892683011744733885610507620 76
01, 3.079201435678004077382126829343773096787209340107 34
02, 3.156181471569954179316680000077367424206889573610 03
03, 3.137852891595680345522738768950321155773632375157 01
04, 3.142604745663084672802649458501777595737810167348 53
05, 3.141308785462883492635401088624107657565761678569 02
06, 3.141674312698837671656932680128065845255313816429 91
07, 3.141568715941784242161823553693589035478332087714 54
08, 3.141599773811505839072149767350788097177444360866 12
09, 3.141590510938080099642754229944255043688235437294 60
10, 3.141593304503081513121460820590669774105615906308 23
11, 3.141592454287646300323593597350456595282934893999 74
12, 3.141592715020379765581606212477455303455223737774 34
13, 3.141592634547313881242713430030850763895875329201 93
14, 3.141592659521713638451335328031521138241880007724 40
15, 3.141592651733997585128216671665720698929685000443 20
16, 3.141592654172575339199092210527739018310271315854 49
17, 3.141592653406165187919674184028247546504944188153 80
18, 3.141592653647826046431202390582141253830948237428 79
19, 3.141592653571403381773710564577918457497083709025 59
20, 3.141592653595634958372427485018281783163918803397 34
21, 3.141592653587933449530974820382197315316320052472 98
22, 3.141592653590386522717511595044061256927036691656 29
23, 3.141592653589603627019680709513679147902339891916 94
24, 3.141592653589853940610143645703665264393229344894 83
25, 3.141592653589773774819733947185303697673924879562 04
26, 3.141592653589799488375148378785532879451814991083 88
27, 3.141592653589791228869469803786671384698916955261 71
28, 3.141592653589793885435623731417884146169147317660 65
29, 3.141592653589793029931269076756985121288903641633 88
30, 3.141592653589793305749612927166783167561769089096 06
31, 3.141592653589793216728877610367853639399627330920 44
32, 3.141592653589793245489422866564430871575088514331 02
33, 3.141592653589793236188749027495885995498446838103 72
34, 3.141592653589793239199112057532564771813108636689 37
35, 3.141592653589793238223924033717866013288640730105 28
36, 3.141592653589793238540080881621261501212098361920 21
37, 3.141592653589793238437505548745937631796932108042 48
38, 3.141592653589793238470809228250912914074583489171 62
39, 3.141592653589793238459989045458157231646823335808 99
40, 3.141592653589793238463506718053332947333214496778 73
41, 3.141592653589793238462362414919962533796677613089 78
42, 3.141592653589793238462734874371216433104648441898 34
43, 3.141592653589793238462613575316210373943048746692 48
44, 3.141592653589793238462653099727392123557502579961 81
45, 3.141592653589793238462640214479790747309567081203 68
46, 3.141592653589793238462644417194958221282908050332 67
47, 3.141592653589793238462643045782640413986344155143 21
48, 3.141592653589793238462643493494565471351201784156 95
49, 3.141592653589793238462643347272152237127662423839 33
50, 3.141592653589793238462643395047792204745254492061 92
```

Figure 11: pi_B.sage の出力

数列S_nが円周率πに収束している.

3.5.3 方法 C

定理:

$$\pi = 4\sum_{k=0}^{\infty}\frac{(-1)^k}{2k+1}\left\{\left(\frac{1}{2}\right)^{2k+1} + \left(\frac{1}{3}\right)^{2k+1}\right\}.$$

この定理を次の pi_C.sage で数値計算する. その際数列A_n及びS_nを$n \geq 0$の範囲で

$$A_n = \sum_{k=0}^{n}\frac{(-1)^k}{2k+1}\left\{\left(\frac{1}{2}\right)^{2k+1} + \left(\frac{1}{3}\right)^{2k+1}\right\}, \quad S_n = 4A_n$$

と定義する.

```
# pi_C.sage
# 円周率πを計算して出力する
# [1] 準備
R = RealField(500) # 500ビットの浮動小数点を使う
N = 50 # 第50項まで計算する

# [2] 数列S[n]を計算する
A = dict() # 空の辞書を作成する
A[0] = R(1/2 + 1/3) # 初期値
S = dict() # 空の辞書を作成する
S[0] = 4 * A[0] # 初期値

for n in range(1, N+1):
    A[n] = A[n-1] + (-1)^n / (2*n + 1) * ((1/2)^(2*n+1) + (1/3)^(2*n+1))
    S[n] = 4 * A[n]

# [3] 数列S[n]を出力する
print("n, S[n]")
for n in range(0, N+1):
    print(f"{n:02}, {S[n]:.50f}")
```

pi_C.sage の出力
n, S[n]
00, 3.33333333333333333333333333333333333333333333333333
01, 3.11728395061728395061728395061728395061728395061728
02, 3.14557613168724279835390946502057613168724279835391
03, 3.14085056176105558821608204324253706969756352472402
04, 3.14174119743368905082058564759372563061025185701302
05, 3.14156158787759105927068127952139535739596705912816
06, 3.14159934096619856274973211754528965658705366405751
07, 3.14159118436090672908137907388372946481309717617542
08, 3.14159298133456687617304767293551641171904062269981
09, 3.14159257960635121096510401104147911290382987070616
10, 3.14159267045068592896219887652289026992758121614467
11, 3.14159264971678824735077862446922053902179001684898
12, 3.14159265448534866704397380579026233373660422833254
13, 3.14159265338153952123818597849385885691409247817991
14, 3.14159265363845810331242127336743316294554005386620
15, 3.14159265357837256701741192316104013472004908721780
16, 3.14159265359248353692207180917269226897927541706999
17, 3.14159265358915738258560053834523148983465805552992
18, 3.14159265358994397283803346025822708599099921965575
19, 3.14159265358975740979697666470182412695884929637183
20, 3.14159265358980177539485998261260696108589224219317
21, 3.14159265358979119987478853983499227427228756317851
22, 3.14159265358979372624899021914261234447574204144426
23, 3.14159265358979312153176509481901878092016528695972
24, 3.14159265358979326654048701964491037058757001795071
25, 3.14159265358979323170996072047776219785825372918138
26, 3.14159265358979324008900241965561888057619600781634
27, 3.14159265358979323807041510173843772018050555364854
28, 3.14159265358979323855735502486434323942946643764577
29, 3.14159265358979323843974665360688042118948666360494
30, 3.14159265358979323846818474337750763748860948523585
31, 3.14159265358979323846130092006008388252871284022487
32, 3.14159265358979323846296892340237524796371662661141
33, 3.14159265358979323846256437035293971428622711132910
34, 3.14159265358979323846266257707146202148422152425010
35, 3.14159265358979323846263871698844076608507864832435
36, 3.14159265358979323846264451858396990593924427236846
37, 3.14159265358979323846264310686239114868433166754360
38, 3.14159265358979323846264345062576259930272477690471
39, 3.14159265358979323846264336686063727747610977792138
40, 3.14159265358979323846264338728484993310653410389363
41, 3.14159265358979323846264338230183419483527097460009
42, 3.14159265358979323846264338351827627211913651254256
43, 3.14159265358979323846264338322115679921934194686725
44, 3.14159265358979323846264338329376745692238161074168
45, 3.14159265358979323846264338327601375215433070601367
46, 3.14159265358979323846264338328035672832070874968404
47, 3.14159265358979323846264338327929384204841096533690
48, 3.14159265358979323846264338327955408482126738160476
49, 3.14159265358979323846264338327949033848549194630712
50, 3.14159265358979323846264338327950595949351612475874

Figure 12: pi_C.sage の出力

数列S_nが円周率πに収束している.

3.5.4 方法 D

定理: オイラー

$$\pi = 2\sum_{k=0}^{\infty} \frac{2^k (k!)^2}{(2k+1)!}$$

$$= 2\left(1 + \frac{1}{3} + \frac{1\cdot 2}{3\cdot 5} + \frac{1\cdot 2\cdot 3}{3\cdot 5\cdot 7} + \frac{1\cdot 2\cdot 3\cdot 4}{3\cdot 5\cdot 7\cdot 9} + \frac{1\cdot 2\cdot 3\cdot 4\cdot 5}{3\cdot 5\cdot 7\cdot 9\cdot 11} + \frac{1\cdot 2\cdot 3\cdot 4\cdot 5\cdot 6}{3\cdot 5\cdot 7\cdot 9\cdot 11\cdot 13} + \ldots\right).$$

この定理を次の pi_D.sage で数値計算する. その際, 数列a_n, A_n及びS_nを$n \geq 0$の範囲で

$$a_n = \frac{2^n (n!)^2}{(2n+1)!}, \quad A_n = \sum_{k=0}^{n} a_n, \quad S_n = 2A_n$$

と定義する.

```
# pi_D.sage
# 円周率πを計算して出力する
# [1] 準備
R = RealField(500) # 500ビットの浮動小数点を使う
N = 50 # 第50項まで計算する

# [2] 数列S[n]を計算する
a = dict() # 空の辞書を作成する
a[0] = R('1.0') # 初期値
A = dict() # 空の辞書を作成する
A[0] = a[0] # 初期値
S = dict() # 空の辞書を作成する
S[0] = 2 * A[0] # 初期値

for n in range(1, N+1):
    a[n] = n / (2*n + 1) * a[n-1]
    A[n] = A[n-1] + a[n]
    S[n] = 2 * A[n]

# [3] 数列S[n]を出力する
print("n, S[n]")
for n in range(0, N+1):
    print(f"{n:02}, {S[n]:.50f}")
```

pi_D.sage の出力
n, S[n]
00, 2.000000000000000000000000000000000000000000000000
01, 2.666666666666666666666666666666666666666666666667
02, 2.933333333333333333333333333333333333333333333333
03, 3.047619047619047619047619047619047619047619047619
04, 3.098412698412698412698412698412698412698412698412
05, 3.121500721500721500721500721500721500721500721500
06, 3.132156732156732156732156732156732156732156732156
07, 3.137129537129537129537129537129537129537129537129
08, 3.139469680646151234386528504175562999092410857116 74
09, 3.140578169680336862999401699092101568881754640268 57
10, 3.141106021601377638529341315719024697352870727483 73
11, 3.141358472520136270304529828018857497926013203977 94
12, 3.141479648961140413556620313922777242201121592695 16
13, 3.141537993173475741789108325654294156111358965040 48
14, 3.141566159344947969211688745110888528343887351689 95
15, 3.141579788137595821190356690009240643940272054907 44
16, 3.141586396037061446392135087535714396956701001921 97
17, 3.141589605588230464347284594905715934136109347614 75
18, 3.141591166991501878487627598491122087358524218492 32
19, 3.141591927675146926402153677160935341492521206868 57
20, 3.141592298740339632701922496024258880094470957296 01
21, 3.141592479958224442755297965701695957086120835411 73
22, 3.141592568553634794336948195321776305837594109157 20
23, 3.141592611908835604685415328965645412673421455883 70
24, 3.141592633144036001590786986260601709899132809382 40
25, 3.141592643553447960858126033954207737950952100313 13
26, 3.141592648659951940876065944143523902655618167562 17
27, 3.141592651166781167430327354600097292601545146029 88
28, 3.141592652398206050649964538684028080645158398610 51
29, 3.141592653003482688164701459674434739175069997336 58
30, 3.141592653011597230080146995057822761569937344149 8
31, 3.141592653447635724280121214343429476894130811390 06
32, 3.141592653519746986444850575494271175718567526208 57
33, 3.141592653555264473779717275762596191557469191716 19
34, 3.141592653572765844350521157054234605159246823995 30
35, 3.141592653581393280547396309803633823131954107513 18
36, 3.141592653585647906617088165954022478570549480206 93
37, 3.141592653587746855478136148321547548586923197402 51
38, 3.141592653588782700370601386373053427296302434460 07
39, 3.141592653589294066836501946930125949697135222374 56
40, 3.141592653589546593486329384242260528660509438628 63
41, 3.141592653589671335566364624360302911040007545452 93
42, 3.141592653589732972829440860653923852921641904119 05
43, 3.141592653589763437223834862500196272472334748057 48
44, 3.141592653589778498272748751053409603486160423712 44
45, 3.141592653589785946044189684953350261679810483102 25
46, 3.141592653589789629888128211398482200141185781295 06
47, 3.141592653589791452421445166587126422327339876190 45
48, 3.141592653589792354293602010391816346708117160262 40
49, 3.141592653589792800674770549244642672916784704904 07
50, 3.141592653589793021655547053627229963119095370568 26

Figure 13: pi_D.sage の出力

数列S_nが円周率πに収束している.

3.5.5 方法 E

定理:BBP 公式の１種

$$\pi = \sum_{k=0}^{\infty} \frac{(-1)^k}{4^k}\left(\frac{2}{4k+1} + \frac{2}{4k+2} + \frac{1}{4k+3}\right).$$

この定理を次の pi_E.sage で数値計算する. その際, 数列S_nを$n \geq 0$の範囲で

$$S_n = \sum_{k=0}^{n} \frac{(-1)^k}{4^k}\left(\frac{2}{4k+1} + \frac{2}{4k+2} + \frac{1}{4k+3}\right)$$

と定義する.

```
# pi_E.sage
# 円周率πを計算して出力する
# [1] 準備
R = RealField(500) # 500ビットの浮動小数点を使う
N = 50 # 第50項まで計算する

# [2] 数列S[n]を計算する
S = dict() # 空の辞書を作成する
S[0] = R(10/3) # 初期値

for n in range(1, N+1):
    S[n] = S[n-1] + (-1)^n / 4^n * (2 / (4*n+1) + 2 / (4*n+2) + 1 / (4*n+3))

# [3] 数列S[n]を出力する
print("n, S[n]")
for n in range(0, N+1):
    print(f"{n:02}, {S[n]:.50f}")
```

pi_E.sage の出力
n, S[n]
00, 3.33333333333333333333333333333333333333333333333333
01, 3.11428571428571428571428571428571428571428571428571
02, 3.14635642135642135642135642135642135642135642135642
03, 3.14067876567876567876567876567876567876567876567877
04, 3.14177794438533602620289926791474779090878162085592
05, 3.14155370078473372994842910040110201639344188812489
06, 3.14160105432818102339572254769454930984073533541833
07, 3.14159080712063560255775443497671467876917434172163
08, 3.14159306543695929112409006013586924968845114335138
09, 3.14159256065081639848448689421691385704884797743242
10, 3.14159267476304992034083294937181756555437157087434
11, 3.14159264872790772304409124929986772487187593925752
12, 3.14159265471365468296087046098454159441969506690858
13, 3.14159265332852165082581610452688324996889778592262
14, 3.14159265365083239080983725620154770186427034941786
15, 3.14159265357547166272195889428146678389863530287324
16, 3.14159265359316624081045045109122263749777248409997
17, 3.14159265358899615999710059175916598599455025215759
18, 3.14159265358998216314864724563006787032751335224398
19, 3.14159265358974833832009801241987336414803881833907
20, 3.14159265358980393554449650554615942404509089294281
21, 3.14159265358979068432524618249153024224099144642804
22, 3.14159265358979384954548561188021681668137749803249
23, 3.14159265358979309198915145803958227836270841788101
24, 3.14159265358979327363133743432346329011613187392824
25, 3.14159265358979323000529137033878399369923664129848
26, 3.14159265358979324049941648777038012123258253760025
27, 3.14159265358979323797146913387631345899920982687102
28, 3.14159265358979323858124007410721105059360706922611
29, 3.14159265358979323843397408657442647448565502105518
30, 3.14159265358979323846958141055924334685997384747584
31, 3.14159265358979323846096264695256542714780501583176
32, 3.14159265358979323846305093304717055328729398685945
33, 3.14159265358979323846254447003156501438473868641809
34, 3.14159265358979323846266741021760984056855030263124
35, 3.14159265358979323846263754221704165642347619958527
36, 3.14159265358979323846264480435029584168350167586014
37, 3.14159265358979323846264303729828904318829827110895
38, 3.14159265358979323846264346757145601634554443437171
39, 3.14159265358979323846264336272998762091556204044979
40, 3.14159265358979323846264338829235719277006149417965
41, 3.14159265358979323846264338205594707663787627084419
42, 3.14159265358979323846264338357832019021016311554848
43, 3.14159265358979323846264338320648657867498851222919
44, 3.14159265358979323846264338329735360412295174236683
45, 3.14159265358979323846264338327513668347458927727031
46, 3.14159265358979323846264338328057133533811498562796
47, 3.14159265358979323846264338327924130684169581851440
48, 3.14159265358979323846264338327956695086726051125256
49, 3.14159265358979323846264338327948718623814722392560
50, 3.14159265358979323846264338327950673211871175179968

Figure 14: pi_E.sage の出力

数列S_nが円周率πに収束している.

3.5.6 方法 F

定理: アルキメデス
数列a_n及びb_nを$a_0 = 3, b_0 = 2\sqrt{3},$

$$b_n = \frac{2a_{n-1}b_{n-1}}{a_{n-1} + b_{n-1}}$$
$$a_n = \sqrt{a_{n-1}b_n}$$

で定める. このとき$\lim_{n\to\infty} a_n = \lim_{n\to\infty} b_n = \pi$が成立する.

この定理を次の pi_F.sage で数値計算をする.

```
# pi_F.sage
# 円周率πを計算して出力する
# [1] 準備
R = RealField(500) # 500ビットの浮動小数点を使う
N = 50 # 第50項まで計算する

# [2] 数列a[n]及びb[n]を計算する
a = dict() # 空の辞書を作成する
a[0] = R('3.0') # 初期値
b = dict() # 空の辞書を作成する
b[0] = 2 * R(sqrt(3)) # 初期値

for n in range(1, N+1):
    b[n] = (2 * a[n-1] * b[n-1]) / (a[n-1] + b[n-1])
    a[n] = sqrt(a[n-1] * b[n])

# [3] 数列a[n]及びb[n]を出力する
print("n, a[n]")
for n in range(0, N+1):
    print(f"{n:02}, {a[n]:.50f}")

print("n, b[n]")
for n in range(0, N+1):
    print(f"{n:02}, {b[n]:.50f}")
```

pi_F.sage の出力(1/2)

```
n, a[n]
00, 3.00000000000000000000000000000000000000000000000000
01, 3.10582854123024914818678605148857994018882681583917
02, 3.13262861328123819716174946949173624464977691548157
03, 3.13935020304686720713514682120842118915035058936259
04, 3.14103195089050963811135292645966010703641221616283
05, 3.14145247228546207545060930896122564524766230454968
06, 3.14155760791185764551646334512985954150437647958850
07, 3.14158389214831840866896960372115335052004491578109
08, 3.14159046322805009573845850593095172355428230867580
09, 3.14159210599927155054477664061011735312749725496628
10, 3.14159251669215744759287408476883190596771889236819
11, 3.14159261936538395518954931206531904222219272559484
12, 3.14159264503369089667214150891923841272262301124210
13, 3.14159265145076765170425364049221902044842747238542
14, 3.14159265305503684169112318041547420225727680588685
15, 3.14159265345610413926464315961507833135434148742941
16, 3.14159265355637096366282331655411336427491350418412
17, 3.14159265358143766976266836592257517887034950737165
18, 3.14159265358770434628764837889804718575024086724766
19, 3.14159265358927101541889455405649997380040637883158
20, 3.14159265358966268270170617109077471998597937915933
21, 3.14159265358976059952240907992713475335611514596288
22, 3.14159265358978507872758480742233672087513968308886
23, 3.14159265358979119852887873931401921020342658408646
24, 3.14159265358979272847920222228805745737603148388846
25, 3.14159265358979311096678309303163687072171603227401
26, 3.14159265358979320658867831071753608978017050208549
27, 3.14159265358979323049415211513901116740241120283305
28, 3.14159265358979323647052056624437995386157307072586
29, 3.14159265358979323796461267902072215154221364349319
30, 3.14159265358979323833813570721480770102898941829715
31, 3.14159265358979323843151646426332908840484683897390
32, 3.14159265358979323845486165352545943524907141145407
33, 3.14159265358979323846069795084099202196014381815605
34, 3.14159265358979323846215702516987516863791293630542
35, 3.14159265358979323846252179375209595530735527937238
36, 3.14159265358979323846261298589765115197471586910972
37, 3.14159265358979323846263578393403995114155601679221
38, 3.14159265358979323846264148344313715093326605372835
39, 3.14159265358979323846264290832041145088119356296335
40, 3.14159265358979323846264326453973002586817544027216
41, 3.14159265358979323846264335359455966961492090959937
42, 3.14159265358979323846264337585826708055160727693117
43, 3.14159265358979323846264338142419393328577886876412
44, 3.14159265358979323846264338281567564646932176672236
45, 3.14159265358979323846264338316354607476520749121192
46, 3.14159265358979323846264338325051368183917892233431
47, 3.14159265358979323846264338327225558360767178011491
48, 3.14159265358979323846264338327769105904979499456006
49, 3.14159265358979323846264338327904992791032579817134
50, 3.14159265358979323846264338327938964512545849907417
```

Figure 15: pi_F.sage の出力(1/2)

pi_F.sage の出力(2/2)

```
n, b[n]
00, 3.464101615137754587054892683011744733885610507620 76
01, 3.215390309173472477670643901929531596686336954275 43
02, 3.159659942097500483316634977833210486227538453357 70
03, 3.146086215131434971098098794237254156265359587343 05
04, 3.142714599645368298168859093772123871000969091511 16
05, 3.141873049979823871745487093690229416883881023485 84
06, 3.141662747056848526224490813893915884017907989528 60
07, 3.141610176604689538763470360659801096849072272904 23
08, 3.141597034321526151993218894814256842407581086613 03
09, 3.141593748771352027975981135631688075828638517278 34
10, 3.141592927385097033548008299052950505484542317041 78
11, 3.141592722038613818342804671499823029506680461451 78
12, 3.141592670701998047877018250500842265743795619708 52
13, 3.141592657867844419844008529336759834985004473030 91
14, 3.141592654659306032497220392249859399317220539240 73
15, 3.141592653857171436889364868302591484891994681854 54
16, 3.141592653656637788064203581586040576824440821976 40
17, 3.141592653606504375862713422046841158842617588148 53
18, 3.141592653593971022812640892295756928031028332332 43
19, 3.141592653590837684550141510491342619923516086062 32
20, 3.141592653590054349984517836954823832295023707169 27
21, 3.141592653589858516343111991815355684608872750675 54
22, 3.141592653589809557932760535108279994511826598833 05
23, 3.141592653589797318330172671217623031164067360525 02
24, 3.141592653589794258429525705262840787775658500542 66
25, 3.141592653589793493454363963775262851769089462957 15
26, 3.141592653589793302210573528403438219319980527040 47
27, 3.141592653589793254399625919560486426929736625777 09
28, 3.141592653589793242446889017349748751689802733755 96
29, 3.141592653589793239458704791797064349933420953456 59
30, 3.141592653589793238711658735408893250560175614175 87
31, 3.141592653589793238524897221311850475783479910967 82
32, 3.141592653589793238478206842787589782093469462141 57
33, 3.141592653589793238466534248156524608671227067245 99
34, 3.141592653589793238463616099498758315315682732104 03
35, 3.141592653589793238462886562334316741976797664792 41
36, 3.141592653589793238462704178043206348642076461494 12
37, 3.141592653589793238462658581970428750308396164640 15
38, 3.141592653589793238462647182952234350724976090674 82
39, 3.141592653589793238462644333197685750829121072199 00
40, 3.141592653589793238462643620759048600855157317581 02
41, 3.141592653589793238462643442649389313361666378926 58
42, 3.141592653589793238462643398121974491488293644262 97
43, 3.141592653589793238462643386990120786019950460597 07
44, 3.141592653589793238462643384207157359652864664680 60
45, 3.141592653589793238462643383511416503061093215701 48
46, 3.141592653589793238462643383337481288913150353456 70
47, 3.141592653589793238462643383293997485376164637895 50
48, 3.141592653589793238462643383283126534491918209005 21
49, 3.141592653589793238462643383280408796770856601782 63
50, 3.141592653589793238462643383279729362340591199976 99
```

Figure 16: pi_F.sage の出力(2/2)

数列a_n及びb_nが円周率πに収束している.

4 おわりに

数値実験をして計算式の神秘を体感出来たでしょうか？ 取り敢えずプログラムを動作させたなら、SageMath/Python の超基礎は出来たので自信を持って下さい. 今回は残念なことに神秘のカラクリ, つまり数学の理論は全く説明しませんでした. 興味があって勉強したい人は教科書を少しずつ読んでいきましょう. 何れも難しい内容なのですぐには理解できませんが、地道な取り組みでいつか分かるかもしれません. そしてページ数が多くなりましたが, 神秘の計算式を 14 個も紹介しました. 数学の世界は広く深く 14 個では終わりません. 実は高校生レベルですら他にも沢山の計算式があります. 自分で探して本稿を参考にしながらコーディングしたら楽しいかもしれません. 陰ながら応援します.

参考文献

三谷純. (2021). Python ゼロからはじめるプログラミング. 翔泳社.

滝澤成人. (2018). ステップ 30 Python[基礎編]ワークブック. カットシステム.