アルゴリズムとデータ構造

独立行政法人 国立高等専門学校機構長野工業高等専門学校
3年 電子情報工学科
渋谷圭亮

2021年1月2日

1　目的

カタラン定数 K を C 言語を用いた多倍長演算により、計算することを目的とする。

2　原理

まず、カタラン定数 K は式 1 の通りに定義される。

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)^2} = 0.9159..... \tag{1}$$

今回は式 1 の計算を多倍長演算にて実装して、計算する。また、この式の演算により出力された値が正しいのかを確かめるために同じカタラン定数を示す式 2 も実装する。[1]
そして、これらで計算した値を比較することでカタラン定数を多倍長演算で正確に計算できたかを検証する。

$$K = \frac{1}{64} \sum_{n=1}^{\infty} \frac{(-1)^{n-1} 2^{8n} (40n^2 - 24n + 3)\{(2n)!\}^3 (n!)^2}{n^3 (2n-1)\{(4n)!\}^2} \tag{2}$$

3　実験

実際に C 言語プログラムにて多倍長演算にて式 1、式 2 らを実装する。また今実験にて C 言語プログラムを実行した環境を表 1 に示す。

表 1: 実行環境

名称	型番
CPU	AMD Ryzen 7 3700X
M/B	Asrock X570 Taichi
RAM	Corsair CMW16GX4M2C3600C18
GPU	GIGABYTE RTX 2070 Super AORUS
OS	Ubuntu 18.04.5 LTS
Compiler	gcc Version 7.5.0

今回の実験で作成した C 言語プログラムをソースコード 1 に示す。

ソースコード 1: 今回作成した C 言語プログラム

```
1
2 #include <stdio.h>
3 #include <string.h>
4 #include <stdlib.h>
5 #include <limits.h>
6 #include <time.h>
7 #include <sys/timeb.h>
8
9 #define KETA 1000
```

```
10
11 typedef struct NUMBER
12 {
13   int n[KETA];//各桁の値
14   int sign;//符号
15 }Number;
16
17 int add(Number*, Number*, Number*);
18
19
20
21
22 //int sub(Number*, Number*, Number*); //宣言
23
24 int setSign(Number* a, int s)
25 //多倍長変数a の符号を
26 //s=1なら正に,s=-1なら負に設定して 0
27 //それ以外ならエラーとして-1
28 {
29   if (s == 1)
30   {
31     a->sign = 1;
32     return 0;
33   }
34   else if (s == -1)
35   {
36     a->sign = -1;
37     return 0;
38   }
39   else
40   {
41     return -1;
42   }
43 }
44
45 int getSign(Number* a)//a が 0 なら 1 を, 負なら-1
46 {
47   if (a->sign == 1)
48   {
49     return 1;
50   }
51   else
52   {
53     return -1;
54   }
55 }
56
57 void clearByZero(Number* a)//多倍長変数の値を全部ゼロにし、+の符号をつける
58 {
59   int i;
```

```
60
61   for (i = 0; i < KETA; i++) //すべての配列を 0にセット
62   {
63     a->n[i] = 0;
64   }
65
66   setSign(a, 1);
67 }
68
69 void dispNumber(Number* a)//a を表示
70 {
71   int i;
72
73   if (getSign(a) == 1)
74   {
75     printf("+"); //符号を先に出力
76   }
77   else
78   {
79     printf("-");
80   }
81
82   for (i = KETA-1; i >= 0; i--)
83   {
84     printf("%2d", a->n[i]); //間隔をあける
85   }
86 }
87
88 int zeroNumber(Number* a)
89 //多倍長変数の上位にある 0を数える関数
90 //
91 //戻り値
92 //多倍長変数の上位にある 0の数
93 {
94   int zeroNumber = 0;
95   int i;
96
97   for (i = KETA - 1; i >= 0; i--)
98   {
99     if (a->n[i] == 0) //0があったのでzeronumber を足す
100    {
101      zeroNumber++;
102    }
103    else
104    {
105      break;
106    }
107  }
108
109  return zeroNumber; //返す
```

```
110 }
111
112
113
114 int isZero(Number* a)
115 //a がゼロか判別
116 //
117 //0･･･a==0
118 //-1･･･a!=0
119 {
120   int i;
121
122   if (getSign(a) == -1) //マイナスなので
123   {
124     return -1;
125   }
126
127   for (i = 0; i < KETA; i++)
128   {
129     if (a->n[i] != 0) {
130       return -1;
131     }
132   }
133
134   return 0; //終了
135 }
136
137 void copyNumber(Number* a, Number* b)//a を b にコピー
138 {
139   int i;
140
141   setSign(b,getSign(a)); //符号も忘れずに
142
143   for (i = 0; i < KETA; i++)
144   {
145     b->n[i] = a->n[i];
146   }
147 }
148
149 void getAbs(Number* a, Number* b)//b=|a|
150 {
151   copyNumber(a, b);
152   setSign(b,1);
153 }
154
155 int mulBy10(Number* a, Number* b)
156 //a を 10 倍して b に返す
157 //
158 //戻り値
159 //0･･･正常終了
```

```
160 //-1…オーバーフロー
161 {
162   int i;
163 
164   clearByZero(b);
165 
166   if (a->n[KETA - 1] != 0)
167   {
168     return -1;
169   }
170 
171   int zero = zeroNumber(a);
172 
173   for (i = 0; i < KETA - zero; i++)
174   {
175     b->n[i + 1] = a->n[i];
176   }
177 
178   b->n[0] = 0;
179   setSign(b, getSign(a));
180 
181   return 0;
182 }
183 
184 int mul10E(Number* a,int i){ //10^i した値を引数のところにまんま返す
185   Number b;
186   clearByZero(&b);
187 
188     while(1){
189     mulBy10(a,&b);
190     copyNumber(&b,a);
191     if(i<=1){
192       break;
193     }
194     i--;
195   }
196 
197 
198   return 0;
199 }
200 
201 
202 
203 int divBy10(struct NUMBER *a , struct NUMBER *b){ //mulBy10 の割り算バージョン
204     int i;
205   clearByZero(b);
206 
207     b->n[KETA-1] = 0;
208     for(i=0;i<KETA-1;i++){
209         b->n[i] = a->n[i+1];
```

```
210     }
211     return a->n[0];
212 }
213
214 int div10E(Number* a,int i){ //mul10E の割り算バージョン
215   Number b;
216   clearByZero(&b);
217   while(1){
218
219     divBy10(a,&b);
220     copyNumber(&b,a);
221     if(i<=1){
222       break;
223     }
224     i--;
225
226   }
227
228
229   return 0;
230 }
231
232
233 int setInt(Number* a, int x)
234 //多倍長変数a に int 型変数 x の値を設定する
235 //
236 //0…正常終了
237 //-1…x の値が a に設定しきれなかった
238 {
239   int i;
240   int Length = KETA;
241
242   clearByZero(a); //ひとまずキレイにする
243
244   if (x < 0) //負の値か区別
245   {
246     setSign(a, -1);
247
248     for (i = 0; i < 10; i++) //10進数であることに留意
249     {
250       if (x == 0)
251       {
252         return 0;
253       }
254       else if (Length == 0)
255       {
256         clearByZero(a);
257         return -1;
258       }
259       a->n[i] = x % 10 * (-1);
```

```
260       Length--;
261       x = (x - x % 10) / 10;
262     }
263   }
264   else
265   {
266     for (i = 0; i < 10; i++)
267     {
268       if (x == 0)
269       {
270         return 0;
271       }
272       else if (Length == 0)
273       {
274         clearByZero(a);
275         return -1;
276       }
277       a->n[i] = x % 10;
278       Length--;
279       x = (x - x % 10) / 10;
280     }
281   }
282
283   if (x == 0)
284   {
285     return 0;
286   }
287   else
288   {
289     clearByZero(a);
290     return -1;
291   }
292 }
293
294
295 int numComp(Number* a, Number* b)
296 //2つの多倍長変数a,b の大小を比較
297 //
298 //0･･･a==b
299 //1･･･a>b
300 //-1･･･a<b
301 {
302   if (getSign(a) == 1 && getSign(b) == -1)
303   {
304     return 1; //a>b
305   }
306   else if (getSign(a) == -1 && getSign(b) == 1)
307   {
308     return -1; //a<b
309   }
```

```
310   else if (getSign(a) == 1 && getSign(b) == 1) //同じ符号なので (+ +)
311   {
312     int aZero = zeroNumber(a);
313     int bZero = zeroNumber(b);
314
315     if (aZero > bZero)
316     {
317       return -1; //上位の 0の数を比較することで大きさを判別
318     }
319     else if (aZero < bZero)
320     {
321       return 1;
322     }
323     else //判別できないので
324     {
325       int i;
326
327       for (i = KETA - 1 - aZero; i >= 0; i--) //1桁ずつ比較する
328       {
329         if (a->n[i] > b->n[i])
330         {
331           return 1;
332         }
333         else if (a->n[i] < b->n[i])
334         {
335           return -1;
336         }
337       }
338
339       return 0;
340     }
341   }
342   else
343   {
344     int aZero = zeroNumber(a);
345     int bZero = zeroNumber(b);
346
347     if (aZero > bZero)
348     {
349       return 1;
350     }
351     else if (aZero < bZero)
352     {
353       return -1;
354     }
355     else
356     {
357       int i;
358
359       for (i = KETA - 1 - aZero; i >= 0; i--)
```

```
360       {
361         if (a->n[i] > b->n[i])
362         {
363           return -1;
364         }
365         else if (a->n[i] < b->n[i])
366         {
367           return 1;
368         }
369       }
370 
371       return 0;
372     }
373   }
374 }
375 
376 
377 int sub(Number* a, Number* b, Number* c)
378 //c <- a-b
379 //
380 //0…正常終了
381 //-1…オーバーフロー
382 {
383   clearByZero(c);
384 
385   int i;
386   int h = 0;
387   int aSign = getSign(a);
388   int bSign = getSign(b);
389 
390   if (aSign == 1 && bSign == 1)
391   {
392     if (isZero(a) == 0)
393     {
394       copyNumber(b, c);
395       setSign(c, -1);
396       return 0;
397     }
398     else if (isZero(b) == 0)
399     {
400       copyNumber(a, c);
401       return 0;
402     }
403 
404 
405     if (numComp(a, b) == 1)
406     {
407       for (i = 0; i < KETA; i++)
408       {
409         if (a->n[i] < b->n[i] + h)
```

```
410         {
411           c->n[i] = 10 + a->n[i] - b->n[i] - h;
412           h = 1;
413         }
414         else
415         {
416           c->n[i] = a->n[i] - b->n[i] - h;
417           h = 0;
418         }
419       }
420     }
421     else if (numComp(a, b) == -1)
422     {
423       Number d;
424       sub(b, a, &d);
425       copyNumber(&d, c);
426       setSign(c, -1);
427     }
428 
429     return 0;
430   }
431   else if (aSign == 1 && bSign == -1)
432   {
433     Number d;
434     getAbs(b, &d);
435     int r = add(a, &d, c);
436     return r;
437   }
438   else if (aSign == -1 && bSign == 1)
439   {
440     Number d;
441     getAbs(a, &d);
442     int r = add(&d, b, c);
443     if (r == 0)
444     {
445       setSign(c, -1);
446     }
447     return r;
448   }
449   else
450   {
451     Number d, e;
452     getAbs(a, &d);
453     getAbs(b, &e);
454     int r = sub(&e, &d, c);
455     return r;
456   }
457 }
458 
459 
```

```
460 int add(Number* a, Number* b, Number* c)
461 //c <- a+b
462 //
463 //0…正常終了
464 //-1…オーバーフロー
465 {
466   int i, d;
467   int e = 0;
468
469   clearByZero(c);
470
471   int aSign = getSign(a);
472   int bSign = getSign(b);
473
474   if (aSign == 1 && bSign == 1)
475   {
476     if (isZero(a) == 0)
477     {
478       copyNumber(b, c);
479       return 0;
480     }
481     else if (isZero(b) == 0)
482     {
483       copyNumber(a, c);
484       return 0;
485     }
486
487
488     for (i = 0; i < KETA; i++)
489     {
490       d = a->n[i] + b->n[i] + e;
491       c->n[i] = d % 10;
492       e = d / 10;
493     }
494
495     if (e != 0)
496     {
497       clearByZero(c);
498       return -1;
499     }
500
501     return 0;
502   }
503   else if (aSign == 1 && bSign == -1)
504   {
505     Number d;
506     getAbs(b, &d);
507     int r = sub(a, &d, c);
508     return r;
509   }
```

```
510   else if (aSign == -1 && bSign == 1)
511   {
512     Number d;
513     getAbs(a, &d);
514     int r = sub(b, &d, c);
515     return r;
516   }
517   else
518   {
519     Number d, e;
520     getAbs(a, &d);
521     getAbs(b, &e);
522     int r = add(&d, &e, c);
523     if (r == 0)
524     {
525       setSign(c, -1);
526     }
527     return r;
528   }
529 }
530 
531 int increment(Number* a, Number* b)
532 //b <- a+1
533 {
534   Number one;
535   int r;
536 
537   setInt(&one, 1);
538   r = add(a, &one, b);
539 
540   return r;
541 }
542 
543 int inc(Number* a)
544 //a+1
545 {
546   Number one, b;
547   clearByZero(&b);
548 
549   int r;
550 
551   setInt(&one, 1);
552   r = add(a, &one, &b);
553 
554   if (r == 0)
555   {
556     copyNumber(&b, a);
557   }
558 
559   return r;
```

```
560 }
561
562 int decrement(Number* a, Number* b)
563 //b <- a-1
564 {
565   Number one;
566   int r;
567
568   setInt(&one, 1);
569   r = sub(a, &one, b);
570
571   return r;
572 }
573 int dec(Number* a)
574 //a-1
575 {
576   Number one, b;
577   clearByZero(&b);
578   int r;
579
580   setInt(&one, 1);
581   r = sub(a, &one, &b);
582
583   if (r == 0)
584   {
585     copyNumber(&b, a);
586   }
587
588   return r;
589 }
590
591 int multiple(Number* a, Number* b, Number* c)
592 //c <- a*b
593 //
594 //0…正常終了
595 //-1…オーバーフロー
596 {
597   int i, j, e, h, r;
598   Number d,tmpC;
599
600   int aSign = getSign(a);
601   int bSign = getSign(b);
602   int aZero = zeroNumber(a);
603   int bZero = zeroNumber(b);
604
605   clearByZero(c);
606
607   if (isZero(a) == 0 || isZero(b) == 0)
608   {
609     return 0;
```

```
610   }
611   else if (aSign == 1 && bSign == 1)
612   {
613     clearByZero(&tmpC);
614
615     for (i = 0; i < KETA - bZero + 1; i++)
616     {
617       h = 0;
618       clearByZero(&d);
619
620       for (j = 0; j < KETA - aZero + 1; j++)
621       {
622         e = a->n[j] * b->n[i] + h;
623         if (i + j > KETA - 1 && e != 0)// a->n[] * b->[i] がオーバーフロー
624         {
625           clearByZero(c);
626           return -1;
627         }
628
629         d.n[i + j] |= e % 10;
630         h = (e - e % 10) / 10;
631         if (h != 0 && i + j >= KETA - 1) //最上位の桁まで計算してもなお繰上り
                があある
632         {
633           clearByZero(c);
634           return -1;
635         }
636       }
637
638       r = add(&tmpC, &d, c);
639
640       if (r == -1)//加算でオーバーフロー
641       {
642         return r;
643       }
644
645       copyNumber(c, &tmpC);
646     }
647
648     return 0;
649   }
650   else if (aSign == 1 && bSign == -1)
651   {
652     getAbs(b, &d);
653     r = multiple(a, &d, c);
654     if (r == 0)
655     {
656       setSign(c, -1);
657     }
658     return r;
```

```
659   }
660   else if (aSign == -1 && bSign == 1)
661   {
662     getAbs(a, &d);
663     r = multiple(&d, b, c);
664     if (r == 0)
665     {
666       setSign(c, -1);
667     }
668     return r;
669   }
670   else
671   {
672     Number f;
673     getAbs(a, &d);
674     getAbs(b, &f);
675     r = multiple(&d, &f, c);
676     return r;
677   }
678 }
679
680 int divide(Number* a, Number* b, Number* c, Number* d)
681 //
682 //c <- 商
683 //d <- あまり
684 //0...正常終了
685 //-1...割る数が0
686 {
687   Number n, m;
688
689   clearByZero(c);
690   clearByZero(d);
691
692   if (isZero(b) == 0)
693   {
694     return -1;
695   }
696
697   int aSign = getSign(a);
698   int bSign = getSign(b);
699
700   if (aSign == 1 && bSign == 1)
701   {
702     copyNumber(a, &n);
703
704     while (1)
705     {
706       if (numComp(&n, b) == -1)
707       {
708         copyNumber(&n, d);
```

```
709         return 0;
710       }
711       else
712       {
713         increment(c, &m);
714         copyNumber(&m, c);
715         sub(&n, b, &m);
716         copyNumber(&m, &n);
717       }
718     }
719   }
720   else if (aSign == 1 && bSign == -1)
721   {
722     Number p;
723     getAbs(b, &p);
724     divide(a, &p, c, d);
725     setSign(c, -1);
726   }
727   else if (aSign == -1 && bSign == 1)
728   {
729     Number p;
730     getAbs(a, &p);
731     divide(&p, b, c, d);
732     setSign(c, -1);
733     setSign(d, -1);
734   }
735   else
736   {
737     Number p, q;
738     getAbs(a, &p);
739     getAbs(b, &q);
740     divide(&p, &q, c, d);
741     setSign(d, -1);
742   }
743 }
744
745 int power(Number* a, Number* b, Number* c)
746
747
748 //
749 //c <- a^b
750 //0...正常終了
751 //-1...オーバーフローまたはアンダーフロー
752 //-1...b < 0
753 {
754   clearByZero(c);
755   Number one,two,d,gm,tmp,tmp1;
756   int r;
757
758   int aZero = isZero(a);
```

```
759   int bZero = isZero(b);
760
761   if (numComp(b, c) == -1)
762   {
763     return -2;
764   }
765
766   increment(c, &one);
767
768   if (bZero == 0)
769   {
770     setInt(c, 1);
771     return 0;
772   }
773   if (aZero == 0)
774   {
775     clearByZero(c);
776     return 0;
777   }
778
779   if (numComp(a, &one) == 0)
780   {
781     setInt(c, 1);
782     return 0;
783   }
784   if (numComp(b, &one) == 0)
785   {
786     copyNumber(a, c);
787     return 0;
788   }
789
790   increment(&one, &two);
791
792   if(b->n[0] % 2 == 0)
793   {
794     r = multiple(a, a, &tmp);
795     if (r == -1)
796     {
797       clearByZero(c);
798       return -1;
799     }
800
801     divide(b, &two, &d, &gm);
802     r = power(&tmp, &d, c);
803     if (r == -1)
804     {
805       clearByZero(c);
806       return -1;
807     }
808
```

```
809     return 0;
810   }
811   else
812   {
813     decrement(b, &tmp1);
814     r = power(a, &tmp1, &tmp);
815     if (r == -1)
816     {
817       clearByZero(c);
818       return -1;
819     }
820
821     r = multiple(a, &tmp, c);
822     if (r == -1)
823     {
824       clearByZero(c);
825       return -1;
826     }
827
828     return 0;
829   }
830 }
831
832
833 int inverseNumber(Number* a,Number* b,int p){ //a の逆数を Number* b に返す
      p は精度 この逆数ルーチンは 2次収束
834   Number eps,x,y,g,x1,pow1,pow2,pow3,tei1,tei2,tei0,h,j,a1;
835   int i,c1,c0,c2;
836   int n = KETA - zeroNumber(a); //n = N+1 =log_10(a)
837       //ずらす分の 10^p
838     c2=1;
839
840   c0=isZero(a); //最初にa について判定して実行できるか確かめる
841   if(c0==0){
842     printf("異常終了");
843     return -1;
844   }
845   c0=getSign(a);
846   if(c0==-1){
847     setSign(a,1); //いったん+にセットする
848     c2=-1; //あとで-の符号をつけるために
849   }
850
851   //初期値セット
852   setInt(&eps,1);
853   //mul10E(&eps,9); //機械イプシロンのセット  ε =1
854
855   setInt(&tei2,2);
856   copyNumber(&tei2,&x);
857
```

```
858   c1=p-n-1; //ずらす
859
860   mul10E(&x,c1); //x=2.0*10^{p-n-1}
861   setInt(&tei0,2);
862   mul10E(&tei0,p); //x=y*(2.0-a*y)の 2.0も 10^p 倍する
863
864
865   while(1){
866     copyNumber(&x,&y); //１つ前のx
867
868     multiple(a,&y,&tei2); //te2=a*y
869
870     sub(&tei0,&tei2,&h); //2.0-a*y=h
871
872     multiple(&y,&h,&x); //x=y*(2.0-a*y)
873
874     div10E(&x,p); //ずれてんのでその分直す
875
876     sub(&x,&y,&j); //j=x-y
877
878     getAbs(&j,&g); //直す
879
880     if(numComp(&g,&eps)==-1||numComp(&g,&eps)==0){
881       break; //g<eps と比較することで十分に正確な値を求め切ったかを確認
882     }
883   }
884
885   if(c2<0){
886     setSign(&x,(int)-1); //-にセットする
887   }
888
889   copyNumber(&x,b); //逆数を返すところにx（答え）を入れる
890
891   return 0; //正常終
892 }
893
894 int ultimatedivide(Number* a, Number* b, Number* c)
895 //NewtonRapson 法を応用した除算
896 //c <- a/b
897 //0...正常終了
898 //-1...割る数が 0
899 {
900   Number m,d,e;
901
902   int n=KETA-zeroNumber(a)+5; //精度
903
904   clearByZero(c);
905   clearByZero(&e);
906
907   if (isZero(b) == 0)
```

```
908   {
909     return -1;
910   }
911
912   int aSign = getSign(a); //被除数の符号取得
913   int bSign = getSign(b); //除数の符号取得
914
915   if (aSign == 1 && bSign == 1) //+の時
916   {
917     inverseNumber(b,&d,n); //除数の逆数をとる
918     multiple(a,&d,&e); //Q=NX(被除数×除数の逆数)
919     div10E(&e,n); //ずらす
920     copyNumber(&e,c); //答えをc に返す
921
922
923     }
924   else if (aSign == 1 && bSign == -1) //被除数 (+) 除数 (-)
925   {
926     Number p;
927     getAbs(b, &p);
928     ultimatedivide(a, &p, c);
929     setSign(c, -1);
930   }
931   else if (aSign == -1 && bSign == 1) //被除数 (-) 除数 (+)
932   {
933     Number p;
934     getAbs(a, &p);
935     ultimatedivide(&p, b, c);
936     setSign(c, -1);
937   }
938   else //両方とも (-)
939   {
940     Number p, q;
941     getAbs(a, &p);
942     getAbs(b, &q);
943     ultimatedivide(&p, &q, c);
944   }
945 }
946
947 Number kaijo(Number* a){ //階乗を行う
948
949     Number a1,b,one,i;
950
951     setInt(&a1,1);
952     setInt(&i,1);
953
954     while(1){
955         multiple(&a1,&i,&b); //b=a*i
956         copyNumber(&b,&a1); //a==b a*=i
957         inc(&i);
```

```
958         if(numComp(&i,a)==1){ //i>a なのでやめようね
959             break;
960         }
961     }
962   copyNumber(&a1,a); //引数のやつにも答えを入れる
963     return a1; //答えを返すお
964 }
965
966 Number kensan2(int keta){
967     Number Keta,bunbo,bunshi,one,eps,two,forty,twe4,three,four,eight,six4,n,n8,
          c1,c2,c3,c0,c4,c5,h,h1,value,twon,tmp,before;
968   int flag=0;
969   keta+=1;
970
971   clearByZero(&bunbo);
972   clearByZero(&bunshi);
973   clearByZero(&value);
974   clearByZero(&before);
975   setInt(&one,1);
976   setInt(&Keta,1);
977   setInt(&eps,1);
978   setInt(&two,2);
979   setInt(&forty,40);
980   setInt(&twe4,24);
981   setInt(&three,3);
982   setInt(&four,4);
983   setInt(&eight,8);
984   setInt(&six4,64);
985   copyNumber(&one,&n); //n=1
986
987
988   mul10E(&Keta,keta); //Keta=10^{keta}
989
990
991   while(1){
992
993     copyNumber(&value,&before);
994
995     multiple(&eight,&n,&n8); //n8=8*n
996
997     power(&two,&n8,&c1); //c1=2^(8*n)
998
999     power(&n,&two,&n8); //n8=n^2
1000
1001      multiple(&forty,&n8,&c0); //c0=40*n^2
1002    multiple(&twe4,&n,&c2); //c2=24*n
1003    sub(&c0,&c2,&c3); //c0-c2=c3
1004    add(&c3,&three,&c2); //c2=c3+3 → 40*n^2-24*n+3 =c2
1005
1006    multiple(&two,&n,&n8); //n8=2*n
```

```
1007      copyNumber(&n8,&twon); //twon=2*n
1008
1009      kaijo(&n8); //n8=(2*n)!
1010      power(&n8,&three,&c3); //c3=((2*n!))^3
1011
1012        copyNumber(&n,&c4); //c4=n
1013      kaijo(&c4); //c4=n!
1014      power(&c4,&two,&c0); //c0=(n!)^2
1015
1016
1017      multiple(&c1,&c2,&c4); //c4=c1*c2
1018      multiple(&c3,&c0,&c5); //c5=c0*c3
1019
1020      //c4*c5=分子
1021
1022
1023      multiple(&c4,&c5,&bunshi);
1024
1025      multiple(&bunshi,&Keta,&c4); //正確に計算
1026      copyNumber(&c4,&bunshi);
1027      //ここまでで分子
1028
1029      power(&n,&three,&c0); //c0=n^3
1030      sub(&twon,&one,&c1); //c1=2*n-1
1031      multiple(&two,&twon,&c3); //c3=(2*n)*2=4*n
1032      kaijo(&c3); //(4*n)!
1033          power(&c3,&two,&c4); //c4=(4*n!)^2
1034
1035      multiple(&c0,&c1,&c5); //c5=c0*c1=(n^3)*(2*n-1)
1036      multiple(&c5,&c4,&bunbo); //bunbo=c5*c4=c5*((4*n!)^2)
1037      multiple(&six4,&bunbo,&c5);
1038      copyNumber(&c5,&bunbo);
1039
1040      //ここまでで分母求めた 64*n^3*(2*n-1)*[(4*n)!]^2
1041
1042      ultimatedivide(&bunshi,&bunbo,&h); //h=Σのところ
1043
1044      if (isZero(&h)==0) //たばいちょうで計算できる桁数超えそうになったらおしまい
1045      {
1046
1047
1048        break;
1049      }
1050
1051
1052
1053      if(n.n[0]%2==0){ //+もしくは-かをつける
1054
1055        sub(&value, &h, &h1); //h1=value-h
1056        copyNumber(&h1, &value); //value-=h
```

```
1057
1058      }
1059      else{
1060        add(&value, &h, &h1); //h1=value+h
1061        copyNumber(&h1, &value); //value+=h
1062      }
1063
1064
1065      inc(&n);
1066      flag++;
1067
1068
1069    }
1070
1071    printf("%d 回ループ\n",flag);
1072
1073    div10E(&value,1);
1074
1075    return value;
1076  }
1077
1078  Number catalan2(int keta) //カタラン定数を定義により求める
1079  {
1080    Number value, a, two, loop, tmp, tmp1, tmp2, Keta,eps;
1081    int i=0;
1082    keta+=2;
1083
1084    setInt(&two, 2);
1085    clearByZero(&loop);
1086    setInt(&Keta,1);
1087    clearByZero(&value);
1088    clearByZero(&tmp);
1089    clearByZero(&tmp2);
1090    setInt(&eps,1);
1091
1092
1093    mul10E(&Keta,keta);
1094
1095
1096    while (1)
1097    {
1098      multiple(&two, &loop, &tmp); //tmp=2*n
1099      inc(&tmp); //インクリメント 2*n+1
1100      power(&tmp, &two, &tmp1); //(2*n+1)^2=tmp1
1101
1102      if (numComp(&Keta,&tmp1)==-1) //たばいちょうで計算できる桁数超えそうになったらおしまい
1103      {
1104        break;
1105      }
```

```
1106
1107
1108
1109
1110      ultimatedivide(&Keta, &tmp1, &a); //a<=Keta/tmp1
1111      if (loop.n[0] % 2 == 0) //奇数偶数で計算パターンを変更
1112      {
1113        add(&value, &a, &tmp2); //tmp2=value+a
1114        copyNumber(&tmp2, &value); //value=tmp2 すなわちvalue+=a
1115      }
1116      else
1117      {
1118        sub(&value, &a, &tmp2); //tmp2=value-a
1119        copyNumber(&tmp2, &value); //value-=a
1120      }
1121      inc(&loop); //loop++(n++)
1122      i++;
1123
1124    }
1125    printf("%d 回ループ\n",i);
1126    div10E(&value,2);
1127    return value;
1128  }
1129
1130  int main(){
1131
1132    clock_t start,end;
1133    start=clock();
1134    int a=10;
1135    Number C,B,D;
1136    clearByZero(&C);
1137    C=kensan2(a);
1138    dispNumber(&C);
1139    printf("\n");
1140    clearByZero(&D);
1141    D=catalan2(a);
1142    dispNumber(&D);
1143    if(numComp(&C,&D)==0){
1144      printf("\n");
1145      printf("定義式で計算した値と検算用の式で計算した値は一致した。");
1146    }
1147
1148    end = clock();
1149    printf("\n");
1150    printf("%.6f[s]\n",(double)(end-start)/CLOCKS_PER_SEC);
1151
1152
1153
1154  }
```

カタラン定数の計算で使用する式 1 は非常に収束が遅いため、もとめる桁数は 10 桁とする。収束が遅いことにより、式 1 のとおり計算を行う catalan2 関数の計算量は桁数が増えるにつれて膨大になっていく。その様子を図 1 に示す。 縦軸の loop（回）とは while 文が回る回数を示す。

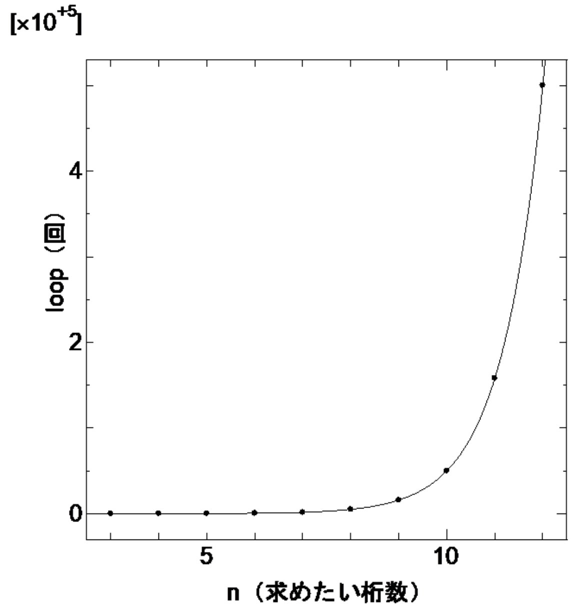

図 1: catalan2 関数の計算量

今回作成したプログラムについておおまかに説明する。kensan2 関数では引数 (int 型) 分の桁数ほどカタラン定数を式 2 の通り計算する。また、catalan2 関数では引数 (int 型) 分の桁数ほどカタラン定数を式 1 の通り計算する。これらの計算に必要な級数表現は while 文を用いて再現し、四則演算または階乗、累乗はそれぞれの処理を行う関数を作成することで可能にした。そして、これらの関数は精度向上のために、引数分の桁数より少し多い桁数で計算をしているため、while のループを抜けたあとに計算値を返す前に余計に多い桁数だけ 10 で割っている。main 関数では指定した桁数だけ catalan2 関数もしくは kensan2 関数で計算させた後に numcomp 関数という引数となる値 2 つを比較する関数に渡すことによって、式 1 によって計算したカタラン定数の検算を行っている。

4　実行結果

実行結果をソースコード 2 に示す。

ソースコード 2: 実行結果

```
18回ループ
+ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 9 1 5 9 6 5 5 9 4 1
500000回ループ
+ 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 9 1 5 9 6 5 5 9 4 1
定義式で計算した値と検算用の式で計算した値は一致した。
1333.416000[s]
```

作成したプログラムと照らし合わせると、C と D の値が完全に一致している時のみ表示される文字が出力されているといえるため、式 1 のとおり計算したカタラン定数の値は正しいといえる。

参考文献

[1] Catalan's Constant https://mathworld.wolfram.com/CatalansConstant.html