Padé Approximant Using ISCZ Method

Haruka Mishima and Hiroshi Kai
{mishima.haruka, kai}@hpc.cs.ehime-u.ac.jp
Graduate School of Science and Engineering,
Ehime University
790-8577
Japan

Abstract

Padé approximant is a rational approximation constructed from the coefficients of a power series of a given input function. Padé approximant may be obtained by the extended Euclidean algorithm, but the algorithm is unstable when the computations are performed in floating-point arithmetic. In this paper, we utilize the ISCZ method to stabilize the numerical computations of Padé approximants using the extended Euclidean algorithm. Experimental results show that the proposed method can guarantee the correct solutions with less CPU time and smaller memory usage compared to those of a normal extended Euclidean algorithm in the case of the power series with coefficients over $\mathbb{Q}(e)$.

1 Introduction

Padé approximant is a rational function approximation of a series expansion of $f(z)$. It is widely applied in engineering to solve the mathematical problems, such as signal processing, model order reduction. The coefficients of Padé approximant can be computed by solving the linear equations obtained from the definition of Padé approximant [1][4]. Thus, it is possible to derive the approximate solutions by numerical methods fast and accurately.

However, if computations are done in floating-point arithmetic, and the Padé approximant is not normal, then pathological features will be observed. Due to the rounding errors, a zero of the numerator polynomial may arise which is very close to the zero of the denominator polynomial. The couples of zeros of the numerator polynomial and the denominator polynomial are called Froissart doublets [6][7] that would affect the computation accuracy of the numerical methods.

While it is known that symbolic computation is a good way to derive the exact results, it still has the problem in terms of the long computation time and large memory usage compared with numerical methods. It has been reported that the extended Euclidean algorithm can compute the Padé approximants with less complexity such as $O(n^2)$ [2][3][5][8], or fast algorithms with $O(n \log^2 n)$[3], and the extended Euclidean algorithm for polynomials may be computed with floating-point coefficients, however, unstable results remain a big concern.

In this paper, we propose an efficient method to compute the Padé approximant based on the extended Euclidean algorithm. In order to address the Froissart doublets issue that appear in the solutions obtained by a naive implementation of the extended Euclidean algorithm in floating-point arithmetic, we improved the extended Euclidean algorithm by introducing the ISCZ method [9]. Experimental results show that the proposed method can derive Padé approximants faster and with smaller memory usage than the method using the extended Euclidean algorithm in the case of the power series with coefficients over $\mathbb{Q}(e)$.

2 Padé approximant and Froissart doublets

Suppose the Taylor series expansion for a given function $f(z)$ at a certain point has the form $f(z) = \sum_{i=0}^{\infty} c_i z^i$. If a rational function

$$r_{m,n}(z) = \frac{p_m(z)}{q_n(z)} = \frac{a_0 + a_1 z + a_2 z^2 + \cdots + a_m z^m}{1 + b_1 z + b_2 z^2 + \cdots + b_n z^n}$$

satisfies $f q_n(z) - p_m(z) = O(z^{m+n+1})$, then the rational function $r_{m,n}(z)$ is called $[m/n]$ Padé approximant of $f(z)$. Based on the definition, it is well known that a Padé approximant can be obtained by solving linear equations.

If a Padé approximant appears only once in the Padé table, then it is called to be "normal" [4]. For example, $f(z) = e^z$ is normal at an expansion point $z = 0$. However, $f(z) = 1 + sin(z)$ is not normal, because [2/1] Padé approximant is $1 + z$, and also [1/0], [2/0], [1/1] Padé approximants have the same form $1 + z$.

In a naive method to calculate the linear equations in floating-point arithmetic, if the Padé approximant is not normal, unnecessary poles may be generated due to rounding errors. In addition, zeros also occur in the numerator polynomial in order to cancel the poles approximately. The pairs of poles and zeros are called Froissart doublets.

To address the issue of Froissart doublets, Ibryaeva et al. proposed a method to compute the linear equations and numerical rank of matrices [7] and Gonnet et al. proposed other method using SVD [6].

In this paper, we examine the Padé approximant algorithm using the extended Euclidean algorithm. The algorithm is as follows:

Algorithm 1 (Padé approximant using the extended Euclidean algorithm [5])
Input : power series s, m, n
Output : $[m/n]$ *Padé approximant*
Method :

```
1 $N := m + n,\ a(z) := z^{N+1}$

2 $b(z) := s(z) \bmod z^{N+1}$

3 if $n = 0$ then return $b$

4 $c := 0,\ d := 1,\ i := 0,\ t := b$

5 while $b \neq 0$ and $i < n$
```

```
    $q := \mathrm{quo}(a,b),\ r := \mathrm{rem}(a,b)$
    while $j = 2, \cdots, \deg(q)$
        $i := i + 1$
    if $i \geq n$ then skip to step6.
    $r1 := c - q \times d$
    $i := i + 1$
    if $r \neq 0$ then $t := \frac{r}{r1}$.
    $a := b,\ b := r,\ c := d,\ d := r1$.
6 return $t$
```

The extended Euclidean algorithm is numerically unstable in floating-point arithmetic. It does not converge to the actual solution even if the precision of the coefficients of polynomials is increased, where the precision denotes the number of digits of the floating-point numbers. Since the zero equivalence test in step 5 does not be performed exactly, the while loop in step 5 can not stop correctly.

Table1 shows the experimental results of the algorithm implemented in Maple 2016. For a Taylor series of $f(z) = (-2.01 + z)/((0.1 + z)(2.01 + z))$ at $z = 0$, computations to obtain [2/3] Padé approximant are repeated by increasing the digits of arbitrary precision arithmetic. Because $f(z)$ is a rational function, results of the [2/3] Padé approximant should converge to the [1/2] Padé approximant. However, even if digits are increased, the degree of the results does not converge.

Table 1: [2/3] Padé approximation of $f(z) = \frac{-2.01+z}{(0.1+z)(2.01+z)}$

Digits	Degree of results	Digits	Degree of results
100	[2/3]	600	[2/3]
200	[2/3]	700	[2/3]
300	[2/3]	800	[4/1]
400	[2/3]	900	[2/3]
500	[3/3]	1000	[2/3]

For example, the following result is obtained in 100 digits computation.

$$r_{2,3}(z) = \frac{-10.000 + 46.546z - 20.682z^2}{1.0000 + 6.3404z - 38.664z^2 - 20.682z^3}$$

Zeros of the numerator polynomial are computed numerically as

$$2.0100 \text{ and } 0.24055.$$

Those of the denominator polynomial are

$$0.24055,\ -0.10000,\ \text{and } -2.0100.$$

The approximant $r_{2,3}(z)$ does not coincide with $f(z)$ because of the appearance of Froissart doublets at 0.24055. On the other hand, for 800 digits, we obtained another approximant $r_{4,1}(z)$ in the form of

$$r_{4,1}(z) = \frac{-10.000 + 9.9502z - 4.9498z^2 + 2.4568z^3 - 1.1643z^4}{1.0000 + 10.000z}.$$

The result is the [4/1] Padé approximant of $f(z)$, but it is not our desired output in this case. In this paper, we apply the ISCZ method [11] to solve this issue.

3 Padé approximant using ISCZ method

The ISCZ method (Interval-Symbol method with Correct Zero rewriting), proposed by Shirayanagi and Sekigawa [9], is based on the stabilization theory [11]. The motivation of ISCZ is to handle the problem that the computations by symbolic algorithms waste memory space by an intermediate swell of coefficients. If the symbolic algorithm is modified into a numerical algorithm carefully, the computations may be more accurate and quick.

In the stabilization theory, the coefficients are described by interval numbers. The computations are executed by increasing the precision of the inputs, and the results thus can converge to the true output. If an interval number contains zero, it will be rewritten to zero. Such process is called Zero Rewriting.

When implementing a program using the stabilization theory, it is required to detect the termination of the stabilized algorithm. The ISCZ method can guarantee a correct zero-rewriting for each processes, and the result is therefore always correct when the ISCZ algorithm terminates. The following theorem is proved in [9].

Theorem 2 *Let $\mathcal{A}$ be an algebraic algorithm with discontinuity at zero. Suppose that $\mathcal{A}$ terminated with an input $\mathcal{I}$. Then, the ISCZ method for $\mathcal{A}$ always terminates in a finite number of steps and gives correct result, i.e. the same result as the output of $\mathcal{A}(\mathcal{I})$.*

Here we stabilize the Padé approximant using the ISCZ method by the following procedures:

R-to-IS Each input coefficient of series is transformed into the pair $[[\tilde{a}, \alpha], Symbol_a]$ named IS, where $[\tilde{a}, \alpha]$ is the interval number of a with a precision α, and $Symbol_a$ is a formal symbol representing of a.

IS Arithmetic Perform arithmetic between IS, where $\dot{+}, \dot{-}, \dot{\times}, \dot{\div}$ denote the formal IS operations of addition, subtraction, multiplication and division:

$$\begin{aligned}
[[A, \alpha], s] + [[B, \beta], t] &= [[A, \alpha] + [B, \beta], s \dot{+} t] \\
[[A, \alpha], s] - [[B, \beta], t] &= [[A, \alpha] - [B, \beta], s \dot{-} t] \\
[[A, \alpha], s] \times [[B, \beta], t] &= [[A, \alpha] \times [B, \beta], s \dot{\times} t] \\
[[A, \alpha], s] \div [[B, \beta], t] &= [[A, \alpha] \div [B, \beta], s \dot{\div} t]
\end{aligned}$$

Correct Zero Rewriting For any IS, say $[[E, \epsilon], s]$, if $|E| \le \epsilon$, then evaluate the symbol s. The evaluated result is denoted as $r(s)$. If $r(s) = 0$, the IS is rewritten to $[[0, 0], 0]$ (Zero Rewriting), then proceed to the next step; otherwise, increase the precision and return to **R-to-IS**.

IS-to-R Substitute the original input coefficient values for the respective symbols to evaluate the accurate output.

Padé approximant using the ISCZ method is summarized as follows:

Algorithm 3 (Padé approximant using ISCZ method)
Input : $f(z)$, m, n
Output : $[m/n]$ *Padé approximant of* $f(z)$
Method :

Step 1 *Set the initial precision for floating-point arithmetic to 10 digits.*

Step 2 *Compute the Taylor expansion of* $f(z)$ *with the precision value.*

Step 3 *Perform the Algorithm 1 using the ISCZ method. If* $r(s) \neq 0$ *in the Correct Zero Rewriting, then increase the precision by 10 digits and return to Step 2.*

Step 4 *output the result* t *obtained in the Algorithm 1.*

Example 4 *An example of Algorithm 3: for the input* $f(z) = (3 + 5z + z^2)/(1 - 4z + z^2)$, $m = 5$ *and* $n = 5$. *The Taylor series of* $f(z)$ *at point* $z = 0$ *is obtained in the initial precision* 10*:*

$$f(z) = 3.000000000 + 17.00000000z + \cdots + 2.491282000 \times 10^6 z^{10} + O(z^{11}).$$

While in the computation of the Step 5 of the Algorithm 1, we have an IS $[[-7.000000001 \times 10^{-11}, 2.600000001 \times 10^{-10}], s]$. *The evaluated result of the symbol* s *in the IS is not equal to zero that is* $r(s) = 157/1551621500881 \neq 0$. *Thus, we increase the precision, and we perform the Algorithm 1 in the precision of* 20. *Then, we obtain the solution as follows:*

$$r_{2,2}(z) = \frac{[2.9999999, 3.0000001] + [4.9999999, 5.0000000]z + [0.99999997, 1.0000000]z^2}{[0.99999998, 1.0000000] + [-4.0000001, -3.9999999]z + [0.99999998, 1.0000000]z^2}.$$

Some experimental results of the proposed method are shown in the next section.

4 Experiments of the proposed method

Computation time of a symbolic implementation of Algorithm 1 and the proposed method are compared. Computations are executed by Maple 2016 on a PC with a Intel(R) Core(TM) i7-3770 CPU (3.40 GHz), 8GB RAM and Windows 8.1 Enterprise 64bit. The interval arithmetic is performed by intpakX v1.0. In this paper, a naive algorithm for polynomial division is used both in the symbolic method and the proposed method.

Table 2 shows the CPU time and the memory usage for $f(z) = \frac{5-z+2z^2+3z^3}{3+3z^2+4z^3}$, where ZR refers the number of Zero Rewriting. For all inputs m and n, we obtained the same result $r_{3,3}(z)$ by the symbolic method and the proposed method.

For the results of CPU time, if the coefficients of the Taylor series are over $\mathbb{Q}$, the symbolic implementation of Algorithm 1 is faster than the proposed method, because the coefficients of the Padé approximant are not complicated and the proposed method iterates the Algorithm 1 several times.

Table 2: Experimental results for $f(z) = \frac{5-z+2z^2+3z^3}{3+3z^2+4z^3}$

		CPU Time		Memory Usage	
(m,n)	ZR	Symbolic	ISCZ	Symbolic	ISCZ
$(10,10)$	21	15[ms]	43[ms]	1.38[MiB]	3.67[MiB]
$(20,20)$	41	14[ms]	126[ms]	1.71[MiB]	11.34[MiB]
$(30,30)$	61	81[ms]	221[ms]	8.38[MiB]	15.87[MiB]
$(40,40)$	81	99[ms]	233[ms]	13.92[MiB]	21.45[MiB]

Table 3 shows the CPU time and the memory usage for an input $f(z) = \frac{-3+ez+z^2}{2-5z+z^2}$, where e is Euler's number. For all inputs m and n, we obtained the same result $r_{2,2}(z)$ by the symbolic method and the proposed method.

From the results of CPU time, it can be seen that the proposed method is faster than the symbolic computation when $(m,n) \geq (20,20)$. In addition, the symbolic computations use large memory space since the coefficients are expressed by rational functions with respect to e.

The results confirm the advantage of the proposed algorithm when the coefficients of Taylor series are over $\mathbb{Q}(e)$.

Table 3: Experimental results for $f(z) = \frac{-3+ez+z^2}{2-5z+z^2}$

		CPU Time		Memory Usage	
(m,n)	ZR	Symbolic	ISCZ	Symbolic	ISCZ
$(10,10)$	20	10[ms]	50[ms]	39.18[MiB]	4.49[MiB]
$(20,20)$	40	4.26[s]	296[ms]	0.53[GiB]	27.8[MiB]
$(30,30)$	60	8.69[s]	433[ms]	1.01[GiB]	46.19[MiB]
$(40,40)$	80	18.95[s]	440[ms]	2.24[GiB]	40.04[MiB]

4.1 An improvement of the proposed method

The process of the Correct Zero Rewriting in the ISCZ method requires symbolic computations for zero equivalence testing of symbols in ISs. If the symbols in ISs are large, then the computation time will increase. For example, it will take more than one week to achieve the computation results we perform the proposed method for the input $f(z) = \frac{5+2z^2+4z^4+7z^5+7z^7+ez^{16}}{9-z^2+8z^3-2z^4+11z^7+z^{16}}$, $m = 20$, $n = 20$. This is because that the symbols expressed by e in the rational functions become complicated forms that cause extreme long time to perform the simplification processes for zero equivalence testing.

In order to further reduce the CPU time for the proposed method, we improved the method by using the probabilistic technique of the zero equivalence testing [12].

Let $P(e)$ be a polynomial with degree D and x be a random number such that $0 \leq x < B$ and B is the upper bound of x. The probability that $P(x) = 0$ even though $P(e)$ is not identically zero is estimated by D/B.

The solution can be derived by setting the upper bound B of the random number large enough. By evaluating the symbols at certain random points, we examine if the symbols are identically zero or not, with high probability.

Table 4 shows the experimental results, where we use `rand()` in Maple 2016 to generate random numbers x and set $B = 10^{12} - 1$. Through the probabilistic technique, we can obtain the results of Padé approximants rapidly as shown in the Table 4.

Table 4: Experimental results for $f(z) = \frac{5+2z^2+4z^4+7z^5+7z^7+ez^{16}}{9-z^2+8z^3-2z^4+11z^7+z^{16}}$

		Probabilistic ISCZ Method		Symbolic Method	
(m,n)	ZR	CPU Time	Memory Usage	CPU Time	Memory Usage
$(20,20)$	71	655.00[ms]	61.52[Mib]	$>$ 1 week	-
$(40,40)$	116	1.31[s]	134.00[MiB]	-	-
$(60,60)$	156	2.68[s]	269.22[MiB]	-	-
$(80,80)$	196	4.28[s]	435.37[MiB]	-	-

5 Conclusion

In this paper, we discussed the problems of the computation of Padé approximant by extended Euclidean algorithm. The computation is unstable in floating-point arithmetic, and Froissart doublets may appear in the solutions of Padé approximant. To address the issues, we proposed a new method using ISCZ method which can derive accurate results. Experimental results show that the proposed method is faster than the symbolic method using extended Euclidean algorithm when the coefficients of Taylor series have transcendental numbers.

We also improved the proposed method by introducing the probabilistic technique to ISCZ method for zero equivalence testing, and confirmed its effectiveness.

Acknowledgements

We would like to thank the referees for their helpful comments. We would like to thank Dr. Senling Wang and Prof. Hiroshi Takahashi for their advice and insightful comments.

References

[1] G. A. Baker, Jr. and P. Graves-Morris, Padé Approximants, second edition, Cambridge U.P., 1996.

[2] R.P. Brent, F.G. Gustavson, and D.Y.Y. Yun, Fast Solution of Toeplitz Systems of Equations and Computation of Padé Approximants, Journal of Algorithms 1, 259-233, 1980.

[3] S. Cabay and D.K. Choi, Algebraic Computations of Scaled Padé Fractions, SIAM J. Comput., Vol.15, No.1, pp.243-270, 1986.

[4] A. Cuyt, L. Wuytack, Nonlinear Methods in Numerical Analysis, North-Holland Publishing Co., 1986.

[5] S. R. Czapor, K.O. Geddes, A Comparison of Algorithm for the Symbolic Computation of Padé Approximants, EUROSAM 84, Lecture Notes in Computer Science, Vol.174, pp. 248-259, 1984.

[6] P. Gonnet, S. Guttel, L.N. Trefethen, Robust Padé Approximation via SVD, SIAM Review, Vol.55, No.1, pp. 101-117, 2013.

[7] O.L. Ibryaeva, V.M. Adukov, An algorithm for computing a Padé approximant with minimal degree denominator, Journal of Computational and Applied Mathematics, Vol. 237, pp. 529-541, 2013.

[8] R.J. McEliece and J.B. Shearer, A property of Euclid's algorithm and an application to Padé approximation, SIAM J. Appl. Math. 34, 4, pp. 611-617, 1978.

[9] K. Shirayanagi and H. Sekigawa, Reducing Exact Computations to Obtain Exact Results Based on Stabilization Techniques, Proc. International Workshop on Symbolic-Numeric Computation 2009 (SNC2009), pp. 191-197, 2009.

[10] K. Shirayanagi and H. Sekigawa, Interval-symbol method with correct zero rewriting: Reducing exact computations to obtain exact results, Proc. 18th Asian Technology Conference in Mathematics (ATCM2013) , pp. 226-235, 2013.

[11] K.Shirayanagi and M. Sweedler, A Theory of Stabilizing Algebraic Algorithms, Technical Report 95-28, Mathematical Sciences Institute, Cornell University, pp. 1-92, 1995.

[12] R. Zippel, Effective Polynomial Computation, Kluwer Academic Publishers, 1993.