RintC: fast and accuracy-aware decomposition of distributions of RNA secondary structures with extended logsumexp

Hiroki Takizawa¹, Junichi Iwakiri¹, and Kiyoshi Asai¹,²

¹Graduate School of Frontier Sciences, The University of Tokyo, Japan.
²Artificial Intelligence Research Center (AIRC), National Institute of Advanced Industrial Science and Technology (AIST), Japan.

The analysis of secondary structures is essential to understanding the function of RNAs. Because RNA molecules thermally fluctuate, it is necessary to analyze the probability distribution of secondary structures. Existing methods, however, are not applicable to long RNAs owing to their high computational complexity. Additionally, previous research has suffered from two numerical difficulties: overflow and significant numerical error. In this research, we reduced the computational complexity in calculating the landscape of the probability distribution of secondary structures by introducing a maximum-span constraint. In addition, we resolved numerical computation problems through two techniques: extended logsumexp and accuracy-guaranteed numerical computation. We analyzed the stability of the secondary structures of 16S ribosomal RNAs at various temperatures without overflow. The results obtained are consistent with in vivo assay results reported in previous research. Furthermore, we quantitatively assessed numerical stability using our method. These results demonstrate that the proposed method is applicable to long RNAs. Source code is available on https://github.com/eukaryo/rintc.

RNA Secondary Structure | Dynamic Programming | Accuracy-Guaranteed Numerical Computation | Interval Arithmetic | Ribosomal RNA

Introduction

Functional noncoding RNAs (ncRNAs) play essential roles in a wide range of biological phenomena. Secondary structures are often crucial for the function of RNAs. There are a number of studies and software tools that can predict a single secondary structure for a given RNA. According to detailed analyses of free energy, however, some RNAs do not always form a single stable structure. Therefore, quantitative evaluations of the fluctuation of RNA secondary structures have recently attracted attention. Recent studies have provided methods to analyze the distribution of RNA secondary structures in more detail using the marginal probability of the Hamming distance(1), (2), (3).

The computational costs of basic algorithms used by previous methods are high, but the Fourier transform has been shown to reduce the time complexity of analysis. (4), (5), (6) Those previous methods, however, cannot be used to analyze long ncRNAs (e.g., > 1000 nucleotides) because the computational complexities are still too high. In this paper, we show that further reduction of computational complexity is possible by introducing maximum-span constraint on base-pairs (7).

While implementing and experimenting with our proposed method, we encountered two numerical computation problems. The first problem consists of numerical errors caused by Fourier transform. Specifically, the magnitude of numerical errors is uniform across large and small marginal probabilities, resulting in the small marginal probabilities being unreliable. This phenomenon occurs because the Fourier transform distributes numerical errors in its dynamic post-processing. A previous research (8) mentioned this type of numerical instability, but they have not shown detailed analysis. Accuracy-guaranteed numerical computation may provide quantitative evaluation of numerical instability, but this was unexamined by previous studies. In the context of kinetic analyses, for example, meta-stable structures are particularly interesting (9). Meta-stable structures may have considerably high free energy compared to the minimum free energy structure. In such a case, the Boltzmann probability of the meta-stable region can be very small. For reliable evaluation of meta-stable regions using distance-wise decomposition like RintW, quantitative assessment of the numerical error introduced by the Fourier transform method is necessary.

Interval arithmetic is a method in which arithmetic operations are defined along intervals expressing numerical values between the upper/lower edges. The approximate calculation of pi by Archimedes in the 3rd century BC is known as the oldest example of interval arithmetic. Around the 1950s, interval arithmetic came to be used for estimating the upper bounds on the numerical error caused by floating-point arithmetic in computers. For example, Sunaga (10) published one of the first studies in English on comprehensive algorithms for interval arithmetic for computers. Numerical calculations, however, have been used in only limited research fields, and few early studies were published.

The other numerical problem is that overflow occurs for long sequences. In stochastic models such as hidden Markov models common throughout bioinformatics, logsumexp (logarithm of summation of exponentials) is the standard solution to prevent overflow or underflow in numerical calculation. There is a limitation, however, in that it cannot handle zero nor negative values. This limitation is a problem when processing complex numbers with rectangular coordinates in...
Fourier transform. One solution is to apply logsumexp only to radii using polar coordinates, but simple application of polar coordinates causes problems when combined with interval arithmetic for accuracy-guaranteed numerical computation. Complicated conditions occur when the angular interval crosses zero or the radius interval contains zero. In this paper, in addition to a radius of polar coordinates, normalized orthogonal coordinates, rather than angles, are combined. Consequently, we can realize the advantages of logsumexp and interval arithmetic while preserving the simplicity of implementation.

We introduced maximum-span constraint to RintW in order to reduce the computational costs. While Turner energy parameters have been determined by experiments using short RNA, the computational analysis tends to be unstable when RNA can form long range base-pairs (11). Our method lost little by excluding long range base-pairs.

Methods

A. RintW + maximum-span. At first, we introduced maximum-span constraint in base pairs to the baseline algorithm of RintW (6). Detailed descriptions of RintW and the proposed method are in the accompanying supplementary file. The inputs of the algorithm are an RNA sequence and a reference secondary structure, and the outputs are the existence probability and the base pairing probability matrix for each Hamming distance from the reference secondary structure.

A.1. the representation of RNA secondary structure. As a computer-efficient expression, the RNA secondary structure was represented by a binary upper triangular matrix $\sigma$ where each element was $\{0, 1\}$. Each element of $\sigma$ was decided as follows.

$$
\sigma_{i,j} = \begin{cases} 
1 \quad (i < j \text{ and } (i,j) \text{ forms a base pair}) \\
0 \quad (\text{otherwise}) 
\end{cases}
$$

The distance between two RNA secondary structures $\sigma_1, \sigma_2$ were determined by the number of elements with different values, namely, Hamming distance. It was also possible to use distance other than Hamming distance (5), but this time Hamming distance was used. The reason is that when the RNA secondary structure changes with time, it can be considered to be limited to the change of the Hamming distance 1, that is, whether one base pair is generated or extinguished. Hamming distance is a natural distance in that sense. Also, many previous studies (1), (4), (6) used Hamming distance, which was considered to be a standard method. For example, 5’-3’ distance (12) is known as a distance other than the Hamming distance, but introducing this was out of scope of this work.

Only secondary structures that satisfy the following constraints were considered. $N = \text{sequence length}$.

1. Watson-Crick base pairs (A-T, C-G) or wobble base pairs (G-U) only exist.

2. (prohibition of pseudo-knot) for all $1 \leq i < j < k < l \leq N$, $(i, k)$ and $(j, l)$ do not form base pairs at the same time.

3. (max loop constraint) for all $1 \leq i < j < k < l \leq N$, if $(i, l)$ and $(j, k)$ form base pairs and no paired base exists between $i+1$ and $j-1$ nor between $k+1$ and $l-1$, then $j-i+l-k \leq C+2$. $C$ is a max-loop parameter, and We used $C = 30$ following many previous works.

4. (max span constraint) for all $1 \leq i < j \leq N$, if $(i, j)$ forms a base pair, then $j-i \leq W$.

1. and 2. were the standard constraints used in the previous methods (1), (4), (6). 3. is called max loop constraint. This constraint was adopted by many of RNA secondary structure analysis methods using the energy model (13) described in the next section. This constraint reduces the time complexity. It is empirically known that this constraint has little effect on the calculation result. 4. is a constraint being studied in other previous works (14), (15), (7), (16), (11), but it was not used in the research by RintW(6) and we introduced it. This constraint is considered to be suitable for examining local structural motifs. (11)

A.2. energy model. Each structure was considered to have energy, and this Boltzmann probability was analyzed. An energy model (13) that can be analyzed by dynamic programming was adopted. In this model, the energy of the secondary structure was expressed by the sum of the following functions.

1. $f_{h}(i, j) = \text{the energy of base pair } (i, j) \text{ forming a hairpin loop}$.

2. $f_{l}(i, j, k, l) = \text{the energy of base pair } (i, l) \text{ and } (j, k) \text{ making a 2-Loop when } i < j < k < l$.

3. $f_{mc} = \text{the energy of having one multi loop}$.

4. $f_{mi} = \text{the energy of having one internal branch of multi loop}$.

5. $f_{d}(i, j) = \text{the energy of base pair } (i, j) \text{ forming a multi loop or being an outermost base pair}$.

A.3. polynomial approach. In previous researches (4), (5), (6), the polynomial approach was used as a method to reduce the time complexity of dynamic programming. For details of the approach, see the description of the previous articles. Briefly, a naïve dynamic programming method requires a convolution operation. This operation is regarded as computation in the spatial domain and is expressed by calculation in the frequency domain. Convolution operation can be converted to an inner product, and the computational complexity is reduced. After completing the dynamic programming computation, shift to the spatial domain by performing the Fourier transform. The same method was used in this study.
**A.4. Preprocessing.** As in the RintW algorithm, we calculate the following $g_0^Z(i, j)$ functions in $O(N^2)$ time as preprocessing, to obtain the gains of the Hamming distance $g_1^Z$ to $g_5^Z$ and $g_1^W$ to $g_5^W$

for $(1 \leq i \leq N)\quad g_0^Z(i, i) = g_0^Z(i, i-1) = 0$

for $(1 \leq i < j \leq N)\quad g_0^Z(i, j) = \sum_{p=i}^{j-1} \sum_{q=p+1}^{j} \sigma_{pq} \quad = \sigma_{ij} + g_0^Z(i+1, j) + g_0^Z(i, j-1) - g_0^Z(i+1, j-1)$

where $\sigma$ is a binary matrix representation of the reference secondary structure. The maximum Hamming distance of the secondary structure from the representative secondary structure ($H_{max}$) is also computed at this time (5).

**A.5. Definitions of Function $g_s$.**

$b(i, j) = 1 - 2\sigma_{ij}\quad g_1^Z(i, j) = g_0^Z(i, j)\quad g_2^Z(i, j, k) = g_0^Z(i, j) - g_0^Z(i, k) - g_0^Z(k+1, j)\quad g_3^Z(i, j, k) = g_0^Z(i, j) - g_0^Z(i, k)\quad g_4^Z(i, j) = g_0^Z(i, j) + b(i, j)\quad g_5^Z(i, j, k, l) = g_0^Z(i, j) - g_0^Z(k, l) + b(i, j)\quad g_6^Z(i, j, k) = g_0^Z(i, j) - g_0^Z(i, j-1) - g_0^Z(k, j-1) + b(i, j)\quad g_7^Z(i, j, k) = g_0^Z(i, j) - g_0^Z(i, k) - g_0^Z(k, j)\quad g_8^Z(i, j, k) = g_0^Z(i, j) + g_0^Z(i, k) - g_0^Z(k, j)\quad g_9^Z(i, j, k) = g_0^Z(i, j, k, l) - b(i, j, k)\quad g_1^W(i, j, h, l) = g_0^Z(1, N) - g_0^Z(i, j) - g_0^Z(1, i-1) - g_0^Z(j+1, N)\quad g_2^W(i, j, h, l) = g_0^Z(h, l) - g_0^Z(i, j) + b(h, l)\quad g_3^W(i, j, h, l) = g_2^W(i, j, h, l) - g_0^Z(h+1, i-1)\quad g_4^W(i, j, h, l) = g_2^W(i, j, h, l) - g_0^Z(j+1, l-1)\quad g_5^W(i, j, h, l) = g_2^W(i, j, h, l) - g_0^Z(h+1, l-1) - g_0^Z(j+1, l-1)$

These functions calculate the Hamming distance of a substructure from the reference substructure. These functions were also used in previous researches. (5), (6) According to Mori et al. research (5), by changing this function, one can decompose the structures by another distance indicator. (i.e. other than the Hamming distance) This indicates a further potential of this concept, but it was out of the scope of this study to examine it.

**A.6. Dynamic programming of the partition function.** In the following equations, $x$ is the $(H_{max}+1)$-th root of unity. If Cooley–Tukey fast Fourier transform (FFT) is used instead of discrete Fourier transform (DFT) in post-processing, $x$ is the smallest power of 2 that is equal to or greater than $(H_{max}+1)$. There are $(H_{max}+1)$ kinds of $(H_{max}+1)$-th roots of unity calculated independently. Therefore, parallel computation is possible.

In order to avoid overflow, the proposed extended logsumexp is used. In the following equations, $g_s^*$ and $-\frac{f}{2\pi}$ are real numbers, but $x^{g_s^*}$ and $e^{-\frac{f}{2\pi}}$ are converted into complex logsumexp type. Consequently, All DP-variable $Z_{*,*}^*, W_{*,*}^*$, and $Q_{*,*}^*$ are also complex logsumexp type.

Initialization:

for $(1 \leq i \leq N)$

\[
Z_{1,i} = Z_{i,1} = 1 \\
Z_{1,i} = Z_{i,1} = Z_{i,i-1}^m = Z_{i,i-1}^m = 0 \\
W_{1,N}^b = \begin{cases} 
1 & ((1, N) \text{ forms a base pair}) \\
0 & (\text{otherwise}) 
\end{cases}
\]
Recursion:
for \((1 \leq i < j \leq N)\) s.t. \((j - i \leq W)\)

\[
Z_{i,j} = x g_{i,j} + \sum_{h=1}^{j-1} Z_{i,h-1} Z_{h,j} x g_{h,j}
\]

\[
Z_{i,N} = Z_{i+1,N} x g_{i,N} + \min(N,i+W) \sum_{h=i+1}^{N} Z_{i,h} e^{-\frac{f_d(i,h)}{kT}} Z_{h+1,N} x g_{h,j}
\]

\[
Z_{i,j}^1 = \sum_{h=i+1}^{j-1} Z_{h,i} e^{-\frac{f_d(i,h)}{kT}} Z_{h+1}^2
\]

\[
Z_{i,j}^b = e^{-\frac{f_d(i,j)}{kT}} x g_{i,j}
\]

\[
Z_{i,j}^m = \sum_{h=i}^{j-1} \left( x g_{i,j} + \sum_{l=max(h+1,j-h-C-2)}^{j-1} Z_{h,l} x g_{l,j} \right) Z_{i,h}
\]

\[
Z_{i,j}^m^1 = \sum_{h=i}^{j-1} Z_{b,h} e^{-\frac{f_d(i,j)+f_m}{kT}} x g_{i,j}
\]

\[
W_{i,j}^b = Z_{1,i-1} Z_{j+1,N} e^{-\frac{f_d(i,j)}{kT}} x g_{i,j}^W + \sum_{h=max(1,i-C-1,i-W)}^{i-1} \sum_{l=j+1}^{min(N,h+W)} W_{h,l} e^{-\frac{f_d(i,h)+f_m}{kT}} x g_{h,j}^W
\]

\[
Q_{i,j}^b = Z_{i,j}^b W_{i,j}
\]

The \(Z_{*,*}\) functions are the inside partition functions, which represent the sums of all the Boltzmann factors in the corresponding sub-sequences. \(Z_{1,*}^b, Z_{*,*}^b, Z_{*,*}^m, \) and \(Z_{*,*}^m^1\) are the specified partition functions defined in the McCaskill algorithm (17). \(W_{i,j}^b\) is the outside partition function, which represents the outside of the base-pair \((i,j)\). The \(Q_{i,j}^b\) is the conditional partition function, the sum of all the Boltzmann factors when \((i,j)\) forms a base pair.

The values \(g_{i,j}^Z\) to \(g_{i,j}^W\) and \(g_{i,j}^W\) to \(g_{i,j}^W\), which are computed using the pre-computed function \(g_{i,j}^Z\), are the gains of the Hamming distance for the transitions represented by the recursions of partition functions. The significant difference from RintW is that the recursions of \(Z_{1,n}\) and \(Z_{i,j}\), and \(W_{i,j}^b\) include the maximum-span constraint \(W\) of base-pairs in their range of the sum. A small improvement is that only edges are considered in the sum for \(Z_{*,*}\) \((Z_{1,j}^1\) and \(Z_{i,N}\)). Regarding the maximum-span constraint of base pairs, the algorithmic concept is equivalent to the calculation of dynamic programming (DP) variables \(\alpha_{Outer}\) and \(\beta_{Outer}\) in Rfold (7) and Parasor (11), but the notation of RintW is followed in the above recursions.
A.7. Fourier transform and post-processing. For all \((i,j)\), such as \((1 \leq i \leq j \leq N)\) and \((j-i \leq W)\), by considering \(Q_{i,j}^b\) and \(Z_{1,N}\) of \((H_{\text{max}} + 1)\) ways as a complex number sequence of \((H_{\text{max}} + 1)\) elements and performing Fourier transformation, the conditional partition functions on each Hamming distance are obtained. Let \(Z(d)_{1,N}\) and \(Q(d)_{1,N}^b\) be the conditional partition functions on Hamming distance \(d\) of \(Z_{1,N}\) and \(Q_{i,j}^b\), respectively. Then, the existence probability on Hamming distance \(d\) is written as

\[
\frac{Z(d)_{1,N}}{\sum_{d=0}^{H_{\text{max}}} Z(d)_{1,N}},
\]

and the base-pairing probability on Hamming distance \(d\) is written as

\[
\frac{Q(d)_{i,j}^b}{Z(d)_{1,N}}.
\]

A.8. Computational complexity. In the following description, \(N\) is the length of the sequence, \(H_{\text{max}}\) is the maximum Hamming distance from the reference structure, and \(U\) is the degree of parallelism. Here \(U \leq H_{\text{max}} + 1\) is assumed. In the original RintW algorithm, the computational complexity of pre-processing is \(O(N^2)\) in both time and space. In the partition function calculation, the time complexity is \(O(N^2 H_{\text{max}}/U)\) and the space complexity is \(O(N^2 H_{\text{max}} U)\). The original RintW uses DFT for post-processing; the time complexity of post-processing part is \(O(N^2 H_{\text{max}}^2/U)\), and its space complexity is \(O(H_{\text{max}} U)\). Since \(H_{\text{max}} \leq N\) holds, the computational complexity of the post-processing can be ignored in total complexity in both time and space. Finally, the time and space complexity of the original RintW algorithm as a whole are \(O(N^4 H_{\text{max}}/U)\) and \(O(N^2 H_{\text{max}} U)\), respectively.

When the maximum-span constraint is introduced, the computational complexity of pre-processing remains \(O(N^2)\) in both time and space. In the distribution function calculation, the time complexity is \(O(N W^3 H_{\text{max}}/U)\), and the space complexity is \(O(N W H_{\text{max}}^2 U)\) for the maximum-span of base pair \(W\). When DFT is used for post-processing, the complexity of the post-processing is \(O(N W H_{\text{max}}^2/U)\) in time and \(O(H_{\text{max}} U)\) in space. Because the \(H_{\text{max}}\) value may be close to \(N\), the computational complexity of the post-processing cannot be ignored. By using FFT instead of DFT, we can reduce the time complexity of the post-processing component to \(O(N W H_{\text{max}} \log(H_{\text{max}})/U)\). Then, the total computational complexity is \(O(N(W + W H_{\text{max}}(W^2 + \log(H_{\text{max}})))/U)\) in time and \(O(N(W + W H_{\text{max}} U))\) in space.

The summary of computational complexities is shown, with the notation simplified by using \(H_{\text{max}} \leq N\), in Table 1.

B. Interval arithmetic and accuracy assurance. In this subsection we briefly explain the rounding mode control function of IEEE 754, and the accuracy assurance arithmetic. Representing real numbers by floating-point numbers can cause deviations from actual values. Therefore, numerical values can conceivably be held as an interval including the actual value. We define arithmetic operations between intervals to obtain an interval necessarily containing the results of arithmetic operations on actual values. Then, the upper bound of the numerical error is obtained as the width of the interval of the calculation result. Most modern computers use the IEEE 754 method for floating-point arithmetic. This method has a rounding mode control function, and we can specify truncation and rounding-up. By using this function, the accuracy assurance calculation described above can be executed efficiently.

Our accuracy assurance calculation used the kv library (18) implemented in C++. The kv library is open source software and requires only C++ Boost for its backend.

C. Logsumexp on complex numbers with interval arithmetic. A method to perform logsumexp on whole complex numbers has been developed. Details of the calculation algorithm are provided in the following subsections. There are different parts of algorithms for scalar and interval types, but those for scalar types are described in the supplementary file. In this subsection, only methods for interval types are described. If only the scalar type is considered, the complex number defined in polar coordinates and logsumexp defined only in terms of a radius are sufficient. Extensions to interval arithmetic, however, are complicated.

The Vienna RNA Package (19) prevents overflow by scaling. Their scaling factor construction is sophisticated, and under some assumptions, their scaling is equivalent to a kind of logsumexp. The original RintW (6) also utilized the same scaling technique as Vienna. However, with Vienna’s method, the deviation between the scaling factor and the value of the actual distribution function can increase exponentially, so overflowing cannot be completely avoided. Unlike them, logsumexp does not need scaling factors, and overflows are completely avoided.

C.1. Notation and representation. In this subsection, a bracketing character like \([x]\) indicates an interval type variable. A pair of values in a bracket (e.g., \([0,1]\)) indicates a closed interval. When two variables are enclosed (e.g., \([x,y]\)), each variable \(x\) and \(y\) is a scalar type (or floating-point type), not an interval type. It is possible to convert one scalar \(x\) into an interval type while guaranteeing accuracy. Such an interval variable is expressed as \([x,x]\) (i.e., \([x]\) is an interval that includes the real value \(x\)). Finally, a function \(f_{\text{upper}}([x]) = u\) for obtaining the maximum value of the interval type variable \([x]\) is \([l,u]\), a function \(f_{\text{lower}}([x]) = l\) for obtaining the minimum value, and a function \(f_{\text{mid}}([x]) = \frac{l + u}{2}\) for obtaining the median value are used. It is assumed that they are not necessarily accuracy-guaranteed. In this subsection, the base of \(\log\) is \(e\).

To represent the complex number \([a, b]\), \((r), (c), (d)\) is held for

\[
[a, b] = e^{r} (c, d).
\]
However, as a normalization condition, 

\[ f_{upper}([c]^2 + [d]^2) = \begin{cases} 0 & (\text{[a]} = [0,0] \text{ and } [b] = [0,0]) \\ 1 & \text{(otherwise)} \end{cases} \]

must be satisfied. It is assumed that 1 is numerically almost 1. The difference from 1 accumulates by multiplication, but it is reset by addition. For convenience, \([r] = [0,0]\) must be satisfied when \((\text{[a]} = [0,0] \text{ and } [b] = [0,0])\).

The normalization protocol and conversion protocol between this and usual representation are described in the supplementary file. Multiplication and addition protocols are described below.

### C.2. Multiplication

The multiplication of the two values \(([r_1],[c_1],[d_1])\) and \(([r_2],[c_2],[d_2])\) can be described as

\[
([r_1],[c_1],[d_1])([r_2],[c_2],[d_2]) = e^{[r_1]}([c_1] + [d_1]i)e^{[r_2]}([c_2] + [d_2]i) = e^{[r_1]+[r_2]}([c_1] + [d_1]i)([c_2] + [d_2]i) = e^{[r_1]+[r_2]}([[c_1][c_2] - [d_1][d_2]] + ([c_1][d_2] + [d_1][c_2])i),
\]

and \(([r_1] + [r_2],[c_1][c_2] - [d_1][d_2],[c_1][d_2] + [d_1][c_2])\) is obtained as a solution. In normalization post-processing, if \([c_1][c_2] - [d_1][d_2] = [c_1][d_2] + [d_1][c_2] = [0,0], [r] = [0,0]\) is substituted. Otherwise, because the product of the complex numbers with absolute value 1 is absolute value 1, it is naturally normalized.

### Algorithm 1 Multiplication

**Input:** \(([([r_1],[c_1],[d_1]),([r_2],[c_2],[d_2]))\)\)  

\(((\text{Interval},\text{Interval},\text{Interval})),(\text{Interval},\text{Interval},\text{Interval}))\) 

where 

- IsNormalized\(([r_1],[c_1],[d_1])\) = True 
- IsNormalized\(([r_2],[c_2],[d_2])\) = True 

**Output:** \(((r],[c],[d]))\) where 

\[ e^{[r]}([c] + [d]i) \geq e^{[r_1]}([c_1] + [d_1]i)e^{[r_2]}([c_2] + [d_2]i) \text{ and } \text{IsNormalized}((r],[c],[d]) = \text{True} \]

1. \([r] \leftarrow [r_1] + [r_2] \]
2. \([c] \leftarrow [c_1][c_2] - [d_1][d_2] \]
3. \([d] \leftarrow [c_1][d_2] + [d_1][c_2] \]
4. \text{if } ([c],[d]) = ([0,0],[0,0]) \text{ then}
5. \([r] \leftarrow [0,0] \]
6. \text{end if}
7. \text{return } ((r],[c],[d])

### C.3. Addition

Consider the sum of the two values \(([r_1],[c_1],[d_1])\) and \(([r_2],[c_2],[d_2])\). Since addition is commutative, assuming \(f_{mid}(r_1) \geq f_{mid}(r_2)\) does not decrease generality. Then, it can be formulated as

\[ p = f_{upper}(r_1) - f_{mid}(r_1) + f_{upper}(r_2) - f_{mid}(r_2) \quad (p \geq 0) \]

and

\[
([r_1],[c_1],[d_1]) + ([r_2],[c_2],[d_2]) = e^{[r_1]}([c_1] + [d_1]i) + e^{[r_2]}([c_2] + [d_2]i) = e^{[r_1]}([c_1] + [d_1]i) + e^{[r_1]}(e^{[r_2]}(e^{[r_2]}([c_2] + [d_2]i)) = e^{[r_1]}([c_1] + e^{[r_2]}([c_2]) + ([d_1] + e^{[r_2]}(e^{[r_2]}([d_2])i) = e^{[r_1]} + [p,p]([c_1] + e^{[r_2]}([c_2] + [p,p][d_2] + e^{[r_2]}(-[p,p][d_2])\].

Thus, \(f_{upper}(e^{-[p,p]}) \leq 1\) follows from the assumption of \(p \geq 0\). Additionally, \(f_{upper}(e^{[r_2]}(-[p,p]) \leq 1\) follows.

| RintW, time | RintC (proposed), time |
|-------------|------------------------|
| preprocessing | \(O(N^2)\) | \(O(N^2)\) |
| main calculation | \(O(N^3/U)\) | \(O(N^2W^3/U)\) |
| postprocessing (DFT) | \(O(N^3/U)\) | \(O(N^2W/U)\) |
| postprocessing (FFT) | \(O(N^3\log N/U)\) | \(O(N^2W\log N/U)\) |
| total (DFT) | \(O(N^3/U)\) | \(O(N^2W(W^2 + N)/U)\) |
| total (FFT) | \(O(N^3/U)\) | \(O(N^2W(W^2 + \log N)/U)\) |

| RintW, space | RintC (proposed), space |
|-------------|------------------------|
| preprocessing | \(O(N^2)\) | \(O(N^2)\) |
| main calculation | \(O(N^3U)\) | \(O(N^2WU)\) |
| postprocessing (DFT) | \(O(NU)\) | \(O(NU)\) |
| postprocessing (FFT) | \(O(NU)\) | \(O(NU)\) |
| total (DFT) | \(O(N^3U)\) | \(O(N^2WU)\) |
| total (FFT) | \(O(N^3U)\) | \(O(N^2WU)\) |
from the assumption that $f_{mid}([r_1]) \geq f_{mid}([r_2])$ (the proof is provided in the Supplementary materials). Therefore, $e^{-[p,p]}$ and $e^{[r_2]-[r_1]-[p,p]}$ can be directly calculated without overflow occurring. Therefore,

$$
[r'] = [r_1] + [p,p]
$$

$$
[c'] = (e^{-[p,p]}[c_1] + e^{[r_2]-[r_1]-[p,p]}[c_2])
$$

$$
[d'] = (e^{-[p,p]}[d_1] + e^{[r_2]-[r_1]-[p,p]}[d_2])
$$

can be calculated, and $([r'], [c'], [d'])$ satisfies

$$
([r_1],[c_1],[d_1]) + ([r_2],[c_2],[d_2]) = ([r'],[c'],[d'])
$$
as the summation. Finally, since this is not normalized, normalization processing is required.

**Algorithm 2 Addition**

**Input:** $((r_1], [c_1],[d_1]), ([r_2],[c_2],[d_2]))::((Interval,Interval,Interval),(Interval,Interval,Interval))$

where

- $IsNormalized([r_1],[c_1],[d_1]) = True$
- $IsNormalized([r_2],[c_2],[d_2]) = True$

**Output:** $([r'],[c'],[d':(Interval,Interval,Interval))$

where $e^{[r]}([c] + [d]) \geq e^{[r_1]}([c_1] + [d_1]) + e^{[r_2]}([c_2] + [d_2])$

and

- $IsNormalized([r],[c],[d]) = True$

1: if $f_{mid}([r_1]) < f_{mid}([r_2])$ then
2: return Addition$(([r_2],[c_2],[d_2]),([r_1],[c_1],[d_1]))$
3: end if
4: $p \leftarrow f_{upper}([r_1]) - f_{mid}([r_1]) + f_{upper}([r_2]) - f_{mid}([r_2])$
5: $[k_1] \leftarrow e^{-[p,p]}$
6: $[k_2] \leftarrow e^{[r_2]-[r_1]-[p,p]}$
7: $[r] \leftarrow [r_1] + [p,p]$
8: $[c] \leftarrow [k_1][c_1] + [k_2][c_2]$
9: $[d] \leftarrow [k_1][d_1] + [k_2][d_2]$
10: return Normalize$([r],[c],[d])$

In the classic logsumexp, it is known that one can reduce the numerical error of summation by using summation-specific technique than recursively using the two-oprand addition function. For the summation-specific technique, in three or more operands, one choice the maximum number as the scaling factor and scale the others. On the other hand, we developed only the normal two-oprand addition function. Following experiment showed that our method brings sufficient numerical accuracy. Nevertheless, further improvement may be still possible.

**C.4. Requirements for an accuracy assurance calculation library.** The functions that the accuracy assurance calculation library must perform in this method are as follows:

1. The conversion from scalar to interval type guarantees accuracy.

### Results and Discussion

**D. Experimental procedure.** The S151 Rfam Dataset ‘with all pseudoknots removed’ (20) was used for evaluation of time complexity and numerical accuracy in interval operation.

For the application of our proposed method to RNA molecules longer than those in the S151 Rfam dataset (20), the primary sequences and the corresponding native secondary structures of 16S rRNAs were obtained from three-dimensional structures of *E. coli* and *T. thermophilus* 70S ribosomes from the Nucleic Acid Database (NDB, (21), (22)). The NDB IDs of the *E. coli* and *T. thermophilus* ribosome structures were 4V9D (chainID: AA) (23) and 4V51 (chainID: AA) (24), respectively. As the secondary structures of these 16S rRNAs, base pairs were selected according to the "base pair hydrogen bonding classification" provided by NDB. Specifically, base-pairs were classified as 1 in the Leontis–Westhof classification (25) and either 19, 20, or 28 in the Saenger classification (26). Base-to-base correspondence between the primary sequence and its secondary structure (derived from the three-dimensional structure in which several residues are missing) was estimated using Needleman–Wunsch alignment (27).

The energy parameter rna_turner2004.par included in the Vienna RNA package (19) version 2.4.9 was used. However, the source code itself of Vienna was not used. The algorithms were implemented by the authors, except for parameter file reading, which is based on ParasOR’s implementation. (11)

**E. Computation time.** To demonstrate computational efficiency, the computation time of the proposed method using the S151 Rfam Dataset (20) was measured. In the proposed method, the reference structures were obtained by Centroid-Fold (28)(γ = 1.0). All cores of the Intel Core i7 4770 CPU were used as a computational resource in this measurement.
We measured the computation time in the case where the maximum-span constraint \( W = 100 \) is introduced and in the case where no restriction is applied (equivalent to RintW \( (6) \) (Figure 1). This result shows the computation time of the methods using the constraint \( W = 100 \) and no-restriction roughly, which follow the theoretical complexities of the square and fifth power, respectively, confirming that computational complexity can be reduced by introducing the maximum-span constraint into the proposed method.

We also examined how the calculation time changes when the value of the maximum-span constraint \( W \) is changed (Figure 2). 32 Cores of the Intel Xeon Gold 6130 were used as a computational resource in this experiment. As a result, when \( N \leq W \), the same calculation time is required regardless of the value of \( W \). When \( W < N \), the calculation time is reduced by the effect of the maximum-span.

F. Thermal stability of ribosomal RNA. As an application of the proposed method, we analyzed the thermal stability of the secondary structures of 16S rRNAs derived from \( E. coli \) and \( T. thermophilus \) using Credibility Limit \( (29) \) as a metric. Credibility Limit is a distance in which a certain percentage of structures is distributed. The larger the Credibility Limit value, the more intense the thermal fluctuation of the molecule. The Credibility Limit \( (0.5) \) was obtained from the Hamming distances of the secondary structures under the proposed method with temperatures ranging from 37 to 55 degrees Celsius (Figure 3). For the 16S rRNAs, three types of reference structure were prepared. (i) The "initial reference" was obtained using CentroidFold. (ii) The "refined reference" was obtained by the following steps: RintC was performed with the initial reference; a region whose existence probability is locally high was chosen; BPPM of the region was calculated; and CentroidFold was used to determine a "refined reference" from the BPPM. (iii) The "natural reference" was the reference structure derived from the three-dimensional structure in NDB. This result shows that the rRNA of thermophilic bacteria \( T. thermophilus \) had a lower Credibility Limit than that of \( E. coli \), which suggests that the thermal stability of \( T. thermophilus \) rRNA is higher than that of \( E. coli \) rRNA. According to an in vivo assay \( (30) \), thermophilic bacteria have reduced dynamics of intracellular macromolecules compared with mesophilic bacteria. Our experimental result is consistent with their results. The use of the natural secondary structure as a representative structure exhibited a relatively higher Credibility Limit, compared with the "initial" and "refined" references. This implies the DP calculation with the Turner model is compatible for the representative structure derived from the Turner model, such as the "initial" and "refined" references. Note that this result would not indicate the advantage of "initial" and "refined" references over the "natural" reference.

G. Numerical error evaluation. For a quantitative evaluation of RintC numerical error, accuracy-guaranteed numerical computations with interval arithmetic were applied to the calculation process of RintC with the RF00008B sequence in the S151 Rfam dataset \( (20) \). The length of the RF00008B sequence was short enough for the evaluation of time-consuming calculation without any type of Fourier transform. The numerical errors of three types of calculation (DFT, FFT, and non-FFT) are shown in Figure 4. For the DFT and FFT methods, in this result, the numerical error (i.e., interval width) is almost equal to 1, when the calculated existence probability is quite small. Interval width \( \leq 1 \) indicates that the probability is within \( [0,1] \), provides no meaningful information owing to numerical error. In contrast, the numerical error remains low when the existence probability is moderate or high. DFT-based results are slightly more accurate than FFT-based results. In further numerical error comparisons between non-Fourier transform results and DFT or FFT results, the numerical error of the non-Fourier result is smaller than those of the DFT or FFT results. This implies that the problematic numerical error is indeed caused by Fourier transform.

The numerical error evaluation was also conducted for all sequences in the S151 Rfam dataset and \( E. coli \) 16S rRNA (Fig-
Since RNA secondary structures have large thermal fluctuations, prediction of the most stable secondary structure is insufficient for representing native structural behavior of an RNA molecule. Marginal probabilities on Hamming distances from reference structures, which represent the landscape of all the possible RNA secondary structures, can be efficiently computed by combining Fourier transform with dynamic programming, but the computational costs are still too high for long RNAs. In this research, we have implemented a maximum-span constraint of base pairs to reduce computational complexity. For long RNAs, however, there remains another problem: numerical overflow. Since the standard method for avoiding overflow in stochastic models, logsumexp (logarithm of sum of exponentials), is not directly applicable to Fourier transform, we have developed an extended logsumexp method for whole complex numbers. We have shown that reduced computational time enables us to analyze the thermal-stability of long RNAs, such as 16S ribosomal RNAs.

We have also adopted accuracy-guaranteed numerical computation with interval arithmetic to evaluate numerical errors. We have shown that numerical errors for small probabilities are substantial when FFT or DFT is used. Quantitative assessment of the observed numerical instabilities, however, reveal that our method achieves sufficient numerical accuracy for thermodynamic stability analysis of RNA secondary structures. These results demonstrate that our method is a powerful tool for understanding long RNAs.

**Acknowledgements**

The authors deeply thank the developers of the RinT, ParaCorF, and ViennaRNA Package implementations, which were necessary for the developed software tool. The authors also deeply thank Dr. Shun Sakuraba, who suggested the possible application of maximum-span constraint and numerical problems in previous research. The authors also thank members of the Artificial Intelligence Research Center, AIST, and Hisanori Kiryu’s laboratory for useful discussions. Computations were partially performed on the NIG supercomputer at the ROIS National Institute of Genetics.

This work was supported by MEXT/SPS KAKENHI Grant Numbers JP16H02484 and JP16H02789 to K.A. and JP16K16143 to J.J. JST CREST Grant Number JP-MJCR18S1 to K.A.

**Bibliography**

1. E. Freyhult, V. Moulton, and P. Clote. RNAbor: a web server for RNA structural neighbors. Nucleic Acids Research, 35(Web Server):W305–W309, may 2007. ISSN 0305-1048. doi: 10.1093/nar/gkm255.
2. Ronny Lorenz, Christoph Flamm, and Ivo L. Hofacker. 2D projections of RNA folding landscapes. In: German Conference on Bioinformatics, pages 11–20, 2009.
3. Lee A. Newberg and Charles E. Lawrence. Exact Calculation of Distributions on Integers, with Application to Sequence Alignment. Journal of Computational Biology, 16(1):11–18, Jan 2009. ISSN 1066-5277. doi: 10.1089/cmb.2008.0137.
4. Evan Senter, Saad Sheikh, Ivan Dutu, Yann Potty, and Peter Clote. Using the Fast Fourier Transform to Accelerate the Computational Search for RNA Conformational Switches. PLoS ONE, 7(12):e50506, Dec 2012. ISSN 1932-6203. doi: 10.1371/journal.pone.005056.
5. Ryota Mori, Michiaki Hamada, and Kyoshi Asai. Efficient calculation of exact probability distributions of integer features on RNA secondary structures. BMC Genomics, 15(Suppl 10):S6, Dec 2014. ISSN 1471-2164. doi: 10.1186/s12864-014-3166-3.
6. Taichi Hago, Shun Sakuraba, Junichi Ishikawa, Ryota Mori, and Kyoshi Asai. Capturing alternative secondary structures of RNA by decomposition of base-pairing probabilities. BMC Bioinformatics, 19(S1):S8, Feb 2018. ISSN 1471-2105. doi: 10.1186/s12859-018-2104-x.
7. Hisanori Kiryu, Taishin Kin, and Kyoshi Asai. Rfold: an exact algorithm for computing local base pairing probabilities. Bioinformatics, 24(3):367–373, Feb 2008. ISSN 1460-2059. doi: 10.1093/bioinformatics/btm351.
8. Evan Senter, Ivan Dotu, and Peter Cotte. RNA folding pathways and kinetics using 2D energy landscapes. *Journal of Mathematical Biology*, 70(2):173–196, jan 2015. ISSN 0303-6812. doi: 10.1007/s00285-014-0760-4.

9. Juraj Michál, Hélène Touzet, and Yann Penty. Efficient approximations of RNA kinetics landscape using non-redundant sampling. *Bioinformatics*, 33(14):2893–2892, jul 2017. ISSN 1367-4803. doi: 10.1093/bioinformatics/btx269.

10. Teruo Sunaga. Theory of an interval algebra and its application to numerical analysis. *Japan Journal of Industrial and Applied Mathematics*, 26(2-3):125–143, oct 1998. ISSN 0916-7005. doi: 10.1007/BF03165539.

11. Risa Kawaguchi and Hisanori Kiryu. Parallel computation of genome-scale RNA secondary structure to detect structural constraints on human genome. *BMC Bioinformatics*, 17(1):203, dec 2016. ISSN 1471-2105. doi: 10.1186/s12859-016-1067-9.

12. Hilary S.W. Han and Christian M. Reidys. The 5-3 Distance of RNA Secondary Structures. *Journal of Computational Biology*, 19(7):867–878, jul 2012. ISSN 1066-5277. doi: 10.1089/cmb.2011.0301.

13. David H Mattheis, Matthew D Disney, Jessica L Childs, Susan J Schroeder, Michael Zuker, and Douglas H Turner. Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure. *Proceedings of the National Academy of Sciences of the United States of America*, 101(19):7897–98, may 2004. ISSN 0027-8424. doi: 10.1073/pnas.0401799101.

14. I. L. Hofacker, B. Prewer, and P. F. Stadler. Prediction of locally stable RNA secondary structures for genome-wide surveys. *Bioinformatics*, 20(2):186–190, jan 2004. ISSN 1367-4803. doi: 10.1093/bioinformatics/btg388.

15. S. H. Berhart, I. L. Hofacker, and P. F. Stadler. Local RNA base pairing probabilities in large sequences. *Bioinformatics*, 22(5):614–615, mar 2006. ISSN 1367-4803. doi: 10.1093/bioinformatics/btl014.

16. Sila J. Lange, Daniel Maticzka, Mathias Möhl, Joshua N. Gagnon, Chris M. Brown, and Rolf Backstein. Global or local? Predicting secondary structure and accessibility in mRNA. *Nucleic Acids Research*, 40(12):5215–5226, jul 2012. ISSN 1362-4642. doi: 10.1093/nar/gks181.

17. J. S. McCaskill. The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, 29(6-7):1105–1119, may 1990. ISSN 0006-3525. doi: 10.1002/bip.360290621.

18. Masahide Kashigawa. -v a C++ Library for Verified Numerical Computation, 2018.

19. Ronny Lorenz, Stephan H Berhart, Christian Hörer zu Siederdissen, Hakim Tafer, Christoph Flamm, Peter F Stadler, and Ivo L Hofacker. ViennaRNA Package 2.0. Algorithms for Molecular Biology, 6(1):26, nov 2011. ISSN 1748-7188. doi: 10.1186/1748-7188-6-26.

20. C. B. Do, D. A. Woods, and S. Batzoglou. CONTRAfold: RNA secondary structure prediction without physics-based models. *Bioinformatics*, 22(14):e90–e98, jul 2006. ISSN 1367-4803. doi: 10.1093/bioinformatics/btl246.

21. Buvaneswari/Combiatore Narayanan, John Westbrook, Saheli Ghosh, Anton I. Petrov, Blake Sweeney, Craig L. Zirbel, Neocles B. Leontis, and Helen M. Berman. The Nucleic Acid Database: new features and capabilities. *Nucleic Acids Research*, 42(D1):D114–D122, jan 2014. ISSN 0305-1048. doi: 10.1093/nar/gkt1090.

22. H. M. Berman, W. K. Olson, D. L. Beveridge, J. Westbrook, A. Gelbin, T. Demeny, S. Hsieh, A. R. Srinivasan, and B. Schneider. The nucleic acid database. A comprehensive relational database of three-dimensional structures of nucleic acids. *Biophysical Journal*, 63(3):751–759, sep 1992. ISSN 0006-3495. doi: 10.1016/S0006-3495(92)81449-1.

23. Jack A Dunkle, Ley Wang, Michael B Feldman, Arto Pulk, Vincent B Chen, Gary J Kapral, Jonas Neeske, Jane S Richardson, Scott C Blanchard, and Jamie H Doudna Cate. Structures of the bacterial ribosome in classical and hybrid states of tRNA binding. *Science (New York, N.Y.)*, 332(6032):981–4, may 2011. ISSN 1095-9203. doi: 10.1126/science.1202902.

24. Maria Selmer, Christine M Dunham, Frank V Murphy, Albert Weixbaumer, Sabine Petry, Ann C Kelley, John R Weir, and V Ramakrishnan. Structure of the 70S ribosome complex with mRNA and RNA. *Science (New York, N.Y.)*, 313(5795):1935–42, sep 2006. ISSN 1095-9203. doi: 10.1126/science.1131127.

25. N B Leontis and E Westhof. Geometric nomenclature and classification of RNA base pairs. *RNA (New York, N.Y.)*, 7(4):499–512, apr 2001. ISSN 1355-8382.

26. Wolfram Saenger. Principles of Nucleic Acid Structure. Springer Advanced Texts in Chemistry. Springer New York, New York, NY, 1984. ISBN 978-0-387-90761-1. doi: 10.1007/978-1-4612-5190-3.

27. Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similitudes in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, mar 1970. ISSN 0022-2836. doi: 10.1016/0022-2836(70)90057-4.

28. Michiaki Hamada, Hisanori Kiryu, Kengo Sato, Toutai Mituyama, and Kiyoshi Asai. Prediction of RNA secondary structure using generalized centroid estimators. *Bioinformatics*, 25(4):465–473, feb 2009. ISSN 1460-2059. doi: 10.1093/bioinformatics/btp091.

29. Bobbie-Jo M. Webb-Robertson, Lee Ann McCue, and Charles E. Lawrence. Measuring Global Credibility with Application to Local Sequence Alignment. *PLoS Computational Biology*, 4(5):e1000077, may 2008. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1000077.

30. Moeava Tehei, Bruno Franzetti, Dominique Madern, Margaret Ginzburg, Ben Z Ginzburg, Marie-Thérèse Giudici-Orton, Mireille Bruschi, and Giuseppe Zaccaria. Adaptation to extreme environments: macromolecular dynamics in bacteria compared in vivo by neutron scattering. *EMBO reports*, 5(1):66–70, jan 2004. ISSN 1469-221X. doi: 10.1038/sj.embo.7400049.
Supplementary Note 1: All Functions for Logsumexp on Complex Number but not Interval

Here we explain how to perform logsumexp on the whole complex number. In this subsection, we describe methods for the normal scalar type (or floating point number type) which is not an interval type. In this section, the base of log is e.

A. representation. First, as a representation of the complex number \(a + bi\), hold \((r, c, d)\) of

\[a + bi = e^r(c + di)\]

However, as a normalization condition,

\[c^2 + d^2 = \begin{cases} 0 & \text{if } a + bi = 0 \\ 1 & \text{otherwise} \end{cases}\]

must be satisfied. For convenience, it must be satisfied that \(r = 0\) when \(a + bi = 0\). The algorithm for checking whether \((r, c, d)\) is normalized can be written as follows.

\[\text{Algorithm 3 IsNormalized}\]

Input: \((r, c, d)\)::(Real,Real,Real)
Output: Bool
1: if \((c, d) = (0, 0)\) then
2: if \(r = 0\) then
3: return True
4: end if
5: return False
6: end if
7: if \(c^2 + d^2 = 1\) then
8: return True
9: end if
10: return False

B. normalization. When a number \((r', c', d')\) which is not normalized is given, a method of obtaining the normalized number \((r, c, d) = (r', c', d')\) is as follows.

\[\text{Algorithm 4 Normalize}\]

Input: \((r', c', d')\)::(Real,Real,Real)
Output: \((r, c, d)\)::(Real,Real,Real) where
\[e^r(c + di) = e^{r'}(c' + d'i) \text{ and } \text{IsNormalized}(r, c, d) = True\]
1: \(s \leftarrow (c')^2 + (d')^2\)
2: if \(s = 0\) then
3: return \((0, 0, 0)\)
4: end if
5: \(t \leftarrow \frac{1}{\sqrt{s}}\)
6: return \((r' - \log(t), tc', td')\)

Description:
First, compute

\[s = (c')^2 + (d')^2\]
\[t = \frac{1}{\sqrt{s}}\]

\(t\) is the reciprocal of the absolute value of the input. At this time

\[(r, c, d) = \begin{cases} (0, 0, 0) & \text{if } s = 0 \\ (r' - \log(t), tc', td') & \text{otherwise} \end{cases}\]

is a normalized solution.
C. conversion. To obtain \((r, c, d)\) when \((a, b)\) such as \(a + bi\) is given, normalize \((0, a, b)\).

**Algorithm 5 ConvertUsualIntoWide**

*Input:* \((a, b)\): (Real, Real)

*Output:* \((r, c, d)\): (Real, Real, Real) where \(e^r(c + di) = (a + bi)\) and IsNormalized\((r, c, d)\) = True

1: return Normalize\((0, a, b)\)

For inverse transformation, calculate normally.

**Algorithm 6 ConvertWideIntoUsual**

*Input:* \((r, c, d)\): (Real, Real, Real)

*Output:* \((a, b)\): (Real, Real) where \((a + bi) = e^r(c + di)\)

1: return \((e^rc, e^rd)\)

**Algorithm 7 Multiplication**

*Input:* \(((r_1, c_1, d_1), (r_2, c_2, d_2))\): ((Real, Real, Real), (Real, Real, Real)) where IsNormalized\((r_1, c_1, d_1)\) = True and IsNormalized\((r_2, c_2, d_2)\) = True

*Output:* \((r, c, d)\): (Real, Real, Real) where \(e^r(c + di) = e^{r_1}(c_1 + d_1i)e^{r_2}(c_2 + d_2i)\) and IsNormalized\((r, c, d)\) = True

1: \(r \leftarrow r_1 + r_2\)
2: \(c \leftarrow c_1c_2 - d_1d_2\)
3: \(d \leftarrow c_1d_2 + d_1c_2\)
4: if \((c, d) = (0, 0)\) then
   5: \(r \leftarrow 0\)
   6: end if
7: return \((r, c, d)\)

D. multiplication. Description:

The multiplication of the two values \((r_1, c_1, d_1)\) and \((r_2, c_2, d_2)\) can be described as

\[
(r_1, c_1, d_1)(r_2, c_2, d_2) = e^{r_1}(c_1 + d_1i)e^{r_2}(c_2 + d_2i) = e^{r_1+r_2}(c_1 + d_1i)(c_2 + d_2i)
\]

and \((r_1 + r_2, c_1c_2 - d_1d_2, c_1d_2 + d_1c_2)\) is obtained as a solution. As a post-processing for normalization, if \(c_1c_2 - d_1d_2 = c_1d_2 + d_1c_2 = 0\), substitute \(r = 0\). Otherwise, since the product of the complex numbers with absolute value 1 is absolute value 1, it is naturally normalized.
Algorithm 8 Addition

Input: \(((r_1, c_1, d_1), (r_2, c_2, d_2))\):
\((\text{Real,Real,Real}, \text{Real,Real,Real})\) where
- IsNormalized\((r_1, c_1, d_1)\) = True and
- IsNormalized\((r_2, c_2, d_2)\) = True

Output: \((r, c, d)\)\((\text{Real,Real,Real})\) where
- IsNormalized\((r, c, d)\) = True

1: if \(r_1 < r_2\) then
2: \quad return Addition\(((r_2, c_2, d_2), (r_1, c_1, d_1))\)
3: end if
4: \(k \leftarrow e^{r_2 - r_1}\)
5: \(r \leftarrow r_1\)
6: \(c \leftarrow c_1 + kc_2\)
7: \(d \leftarrow d_1 + kd_2\)
8: return Normalize\((r, c, d)\)

E. addition. Description:
Consider adding the two values \((r_1, c_1, d_1)\) and \((r_2, c_2, d_2)\). Since addition is commutative, assuming \(r_1 \geq r_2\) does not lose generality. Then, it can be formulated as

\[
(r_1, c_1, d_1) + (r_2, c_2, d_2) = e^{r_1} (c_1 + d_1 i) + e^{r_2} (c_2 + d_2 i)
\]

\[
= e^{r_1} (c_1 + d_1 i) + e^{r_2} (c_2 + d_2 i)
\]

\[
= e^{r_1} (c_1 + d_1 i) + e^{r_1} (e^{r_2 - r_1} c_2 + e^{r_2 - r_1} d_2 i)
\]

\[
= e^{r_1} ((c_1 + e^{r_2 - r_1} c_2) + (d_1 + e^{r_2 - r_1} d_2 i)
\]

Since it is \(e^{r_2 - r_1} \leq 1\) from the assumption of \(r_1 \geq r_2\), \(e^{r_2 - r_1}\) can be directly calculated without overflowing. Therefore,

\[
e' = (c_1 + e^{r_2 - r_1} c_2)
\]

\[
d' = (d_1 + e^{r_2 - r_1} d_2)
\]

can be calculated and \((r_1, c_1', d_1')\) satisfies

\[
(r_1, c_1, d_1) + (r_2, c_2, d_2) = (r_1, c_1', d_1')
\]

as an answer of addition. Finally, since this is not normalized, it needs normalization processing.
Supplementary Note 2: Auxiliary Functions for Logsumexp on Complex Number and Interval

The algorithm for checking whether $([r],[c],[d])$ is normalized can be written as follows.

Algorithm 9 IsNormalized

Input: $([r],[c],[d]) : (\text{Interval},\text{Interval},\text{Interval})$

Output: Bool

1: if $(c, d) = ([0,0],[0,0])$ then
2: if $r = [0,0]$ then
3: return True
4: end if
5: return False
6: end if
7: if $f_{\text{upper}}(c^2 + d^2) \approx 1$ then
8: return True
9: end if
10: return False

A. normalization. When a number $([r'],[c'],[d'])$ which is not normalized is given, a method of obtaining the normalized number with accuracy assurance $([r],[c],[d]) \succeq ([r'],[c'],[d'])$ is as follows.

Algorithm 10 Normalize

Input: $([r'],[c'],[d']) : (\text{Interval},\text{Interval},\text{Interval})$

Output: $([r],[c],[d]) : (\text{Interval},\text{Interval},\text{Interval})$ where $e^{[r']([c] + [d]i)} \succeq e^{[r']([c'] + [d']i)}$ and IsNormalized$([r],[c],[d]) = True$

1: $s \Leftarrow f_{\text{upper}}([c']^2 + [d']^2)$
2: if $s = 0$ then
3: return $([0,0],[0,0],[0,0])$
4: end if
5: $t \Leftarrow \frac{1}{\sqrt{s}}$
6: return $([r'] - \text{log}([t,t]),[t,t][c'],[t,t][d'])$

Description:
First, compute

\begin{align*}
    s &= f_{\text{upper}}([c']^2 + [d']^2) \\
    t &= \frac{1}{\sqrt{s}}
\end{align*}

$t$ is the reciprocal of maximum value of the absolute value of the input. At this time

\begin{align*}
    ([r],[c],[d]) &= \begin{cases} 
        ([0,0],[0,0],[0,0]) & (s = 0) \\
        ([r'] - \text{log}([t,t]),[t,t][c'],[t,t][d']) & (\text{otherwise})
    \end{cases}
\end{align*}

is a normalized solution.

B. conversion. To obtain $([r],[c],[d])$ when $|a| + |b|i$ is given, normalize $([0,0],[a],[b])$. For inverse transformation, calculate normally.

Algorithm 11 ConvertUsualIntoWide

Input: $([a],[b]) : (\text{Interval},\text{Interval})$

Output: $([r],[c],[d]) : (\text{Interval},\text{Interval},\text{Interval})$ where $e^{[r']([c] + [d]i)} \succeq ([a] + [b]i)$ and IsNormalized$([r],[c],[d]) = True$

1: return Normalize$([0,0],[a],[b])$
Algorithm 12 ConvertWideIntoUsual

Input: \((r, c, d)::(\text{Interval, Interval, Interval})\)

Output: \((a, b)::(\text{Interval, Interval})\) where \((a + b) \supseteq e^r (c + d)\)

1: return \((e^r c, e^r d)\)
Supplementary Note 3: Proof of Proposition 1

Proposition 1: $f_{upper}(e^{[r_2]-[r_1]-[p,p]}) \leq 1$

Proof:

Since they are assumptions with $f_{mid}([r_1]) \geq f_{mid}([r_2])$ and

$$p = f_{upper}([r_1]) - f_{mid}([r_1])$$
$$+ f_{upper}([r_2]) - f_{mid}([r_2]) \ (p \geq 0)$$

, it holds that

$$f_{upper}([r_2] - [r_1])$$
$$= f_{upper}([r_2]) - f_{lower}([r_1])$$
$$= (f_{upper}([r_2]) - f_{mid}([r_2]) + f_{mid}([r_2]))$$
$$- (f_{lower}([r_1]) - f_{mid}([r_1]) + f_{mid}([r_1]))$$
$$= (f_{upper}([r_2]) - f_{mid}([r_2]) + f_{mid}([r_2]))$$
$$- (f_{upper}([r_1]) + f_{mid}([r_1]) + f_{mid}([r_1]))$$
$$= (f_{upper}([r_1]) - f_{mid}([r_1]) + f_{upper}([r_2])$$
$$- f_{mid}([r_2]) + f_{mid}([r_2]) - f_{mid}([r_1])$$
$$= p + f_{mid}([r_2]) - f_{mid}([r_1])$$
$$\leq p.$$

Therefore,

$$f_{upper}([r_2] - [r_1] - [p,p]) \leq 0$$

holds. Since,

$$f_{upper}(e^{[r_2]-[r_1]-[p,p]}) \leq 1.$$

QED.