Synthetic Biology, 2017, 2(1): ysx002

doi: 10.1093/synbio/ysx002
Research Article

Chemical reaction networks for computing logarithm

Chun Tung Chou*

School of Computer Science and Engineering, University of New South Wales, Sydney, NSW, Australia

*Corresponding author: E-mail: ctchou@cse.unsw.edu.au

Abstract

Living cells constantly process information from their living environment. It has recently been shown that a number of cell signaling mechanisms (e.g. G protein-coupled receptor and epidermal growth factor) can be interpreted as computing the logarithm of the ligand concentration. This suggests that logarithm is a fundamental computation primitive in cells. There is also an increasing interest in the synthetic biology community to implement analog computation and computing the logarithm is one such example. The aim of this article is to study how the computation of logarithm can be realized using chemical reaction networks (CRNs). CRNs cannot compute logarithm exactly. A standard method is to use power series or rational function approximation to compute logarithm approximately. Although CRNs can realize these polynomial or rational function computations in a straightforward manner, the issue is that in order to be able to compute logarithm accurately over a large input range, it is necessary to use high-order approximation that results in CRNs with a large number of reactions. This article proposes a novel method to compute logarithm accurately in CRNs while keeping the number of reactions in CRNs low. The proposed method can create CRNs that can compute logarithm to different levels of accuracy by adjusting two design parameters. In this article, we present the chemical reactions required to realize the CRNs for computing logarithm. The key contribution of this article is a novel method to create CRNs that can compute logarithm accurately over a wide input range using only a small number of chemical reactions.

1. Introduction

Cells constantly process information from their living environment in order to adjust their behavior accordingly (1,2). From a computational point of view, cells use molecules and chemical reactions to realize communication, information processing and decision making. Inspired by this viewpoint, a research problem in synthetic biology is to design molecular circuits that can carry out different types of computation (3). Potential applications of these circuits include metabolic engineering, therapeutics and diagnosis (4,5). One method of realizing these novel circuits is chemical reaction tetworks (CRNs) (6–8). Theoretically, CRNs have been proven to be Turing universal, which intuitively means that they are powerful computational engines (9). CRNs have been used to realize a few different classes of computation. Salehi *et al.* show how CRNs can be used to realize polynomial computation using a new type of representation of the polynomial variable (10). Oishi and Klavins show that linear dynamical systems that are specified by linear ordinary differential equations can be realized using CRNs that are based on three classes of chemical reactions, namely catalysis, degradation and annihilation (11). This work is further extended by Foo *et al.* to non-linear operators (12). More recently, Zechner *et al.* show how *de novo* molecular circuits can be used to implement optimal filters approximately (13).

There is a growing interest in the synthetic biology community to use chemical reactions to realize analog computation due to the energy efficiency of analog computation (14–17). A pioneering work in analog biochemical computation is by Daniel *et al.* (18), where the authors present a number of synthetic gene circuits to compute the natural logarithm function $\log(1+x)$ for non-negative x. Furthermore, they show that the natural logarithm function can be used as a primitive to realize other computation, e.g. computing the ratio of two numbers. A reason why computing logarithm is important is because it is related to the concept of fold-change detection (19,20). Recently, Olsman and Goentoro show that logarithmic sensing can be found in many cellular signal transduction mechanisms (21). Moreover, theoretical analyses

Submitted: 29 December 2016; **Received (in revised form):** 20 March 2017. **Accepted:** 23 March 2017

of cell signaling show that optimal decoding of cellular signals require the computation of logarithm (22–25).

The aim of this article is to address the computation of logarithm using CRNs. Although the article (18) presents three gene circuits to compute logarithm, the problem of how chemical reactions can be systematically designed to compute logarithm has not been addressed. In this article, we present a novel method to derive CRNs for computing the logarithm function. The method has two tuning parameters that can be used to derive CRNs which compute the logarithm to different levels of accuracy. The advantage of the proposed method is that it can accurately compute $\log(1+x)$ without significantly increasing the number of chemical reactions in the CRNs.

2. Materials and methods

Our proposed method of deriving CRNs consists of a number of steps. In one of the steps, we make use of Pade approximation to compute logarithm. We will first describe Pade approximation. We will then describe our proposed method. After that, we will present how the proposed computation method can be realized by CRNs. Finally, we discuss our proposed method.

2.1 Pade approximation

Pade approximation is a well-known technique to derive rational function approximation of the logarithm function. The key idea is to approximate the logarithm function $\log(1+x)$ by a rational function $Q_{M,N}(x)$ which is the ratio of two polynomials $p_M(x)$ and $p_N(x)$:

$$\log(1+x) \approx Q_{M,N}(x) = \frac{p_M(x)}{p_N(x)}$$

where $p_M(x)$ and $p_N(x)$ are polynomials of degrees M and N, respectively, in the variable x. The coefficients of the polynomials $p_M(x)$ and $p_N(x)$ are chosen such that the first $(M+N)$ terms in the Taylor series expansion of $\log(1+x)$ are identical to the first $(M+N)$ terms in the polynomial expansion of $Q_{M,N}(x)$. This implies that, the difference $\log(1+x) - Q_{M,N}(x)$ is of the order $Q(x^{M+N+1})$. Therefore, the level of approximation can be tuned by adjusting the degrees M and N. We will refer to $Q_{M,N}(x)$ as the Pade (M, N) approximation.

There are standard algorithms to derive the coefficients of the polynomials $p_M(x)$ and $p_N(x)$ (26). It can be shown that the Pade (1,1) and (2,2) approximations of $\log(1+x)$ are:

$$Q_{1,1}(x) = \frac{x}{1+0.5x} \tag{1}$$

$$Q_{2,2}(x) = \frac{x+\frac{1}{2}x^2}{1+x+\frac{1}{6}x^2} \tag{2}$$

Since rational functions can be implemented by CRNs (11), this leads to a way to systematically approximate the logarithm function using CRNs. Figure 1 plots $\log(1+x)$, $Q_{1,1}(x)$ and $Q_{2,2}(x)$ for a range of x. It shows that Pade approximation gives good approximation for small values of x, but the approximation error increases with x. For example, for the Pade (1,1) approximation, the approximation error is 1.5% when $x = 0.5$, but the error increases to 20% when $x = 5$. The figure also shows that $Q_{2,2}(x)$ gives better approximation for a larger range of x. Although the approximation error can be decreased using a higher order Pade approximation (i.e. larger values of M and N), we will show shortly that there are more efficient methods to reduce approximation error than using larger M and N values.

2.2 New method to compute $\log(1+x)$

In order to reduce the approximation error for computing $\log(1+x)$ for large x, this article proposes a new method to compute $\log(1+x)$ approximately. This method requires a technique to approximately compute logarithm as an intermediate step. For illustration, we will use Pade (1,1) approximation,

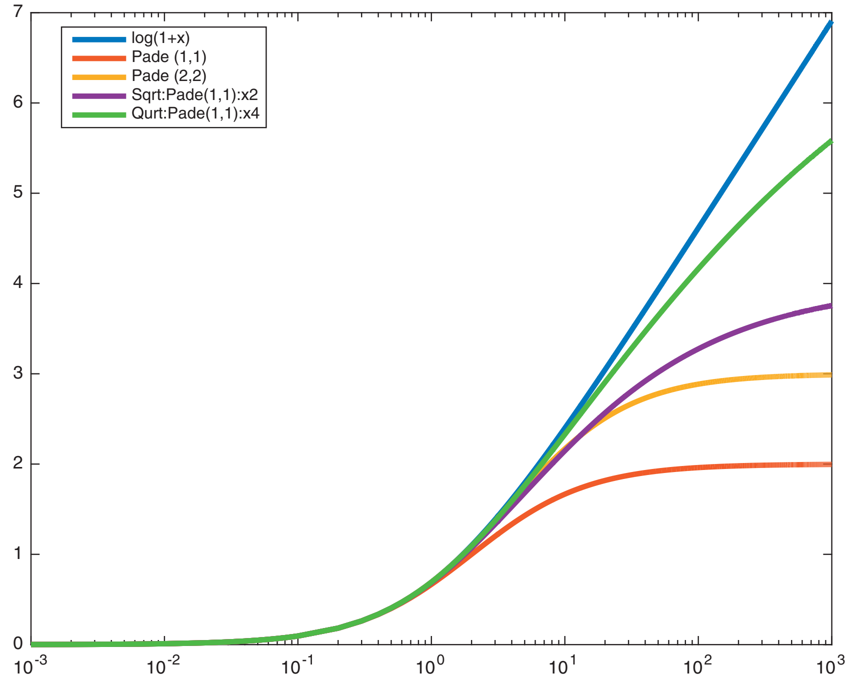

Figure 1. This figure plots $\log(1+x)$ and four different approximations of this function. The approximations are Pade (1,1), Pade (2,2), Sqrt:Pade(1,1):×2 and Qurt:Pade(1,1):×4.

i.e. $Q_{1,1}(x)$, in the explanation, but in general any method for computing logarithm can be used. The proposed method consists of three steps:

(i) compute $x_1 = \sqrt{1+x}$;
(ii) compute $x_2 = Q_{1,1}(x_1 - 1)$ ($\approx \log(x_1)$); and
(iii) compute $x_3 = 2 \times x_2$.

It can readily be shown that x_3 is an approximation of $\log(1+x)$, because $\log(1+x) = 2\log\sqrt{1+x}$. In order to understand how this method will give a better approximation, we first point out that both square root (Step (i)) and multiplication by two (Step (iii)) can be implemented by CRNs exactly, see Section 2.3 for the CRN implementation. Therefore, we will assume that Steps (i) and (iii) can be carried out with high precision and the only approximation is in the computation of $\log\sqrt{1+x}$ in Step (ii). The intuition is that, for any $x > 0$, we have $\sqrt{1+x} < (1+x)$, therefore the proposed method computes the logarithm of a smaller number, which can be more accurately calculated. This improvement in accuracy is demonstrated in Figure 1, where the line labeled as Sqrt:Pade(1,1):×2 (short for: square root—Pade (1,1)—multiplication by 2) shows that the proposed method is more accurate than using Pade (2,2) approximation.

The proposed method is modular. It is possible to replace Steps (i) and (iii) by quartic root (or 4th root) and multiplication by 4, respectively. Figure 1 shows that this revised computation scheme, labeled as Qurt:Pade(1,1):×4 (short for: quartic root—Pade (1,1)—multiplication by 4) shows much better approximation. Moreover, the computation of the quartic root can be implemented in CRN as a cascade of two consecutive square root calculations. Similarly, multiplication by 4 can be implemented as two consecutive multiplication-by-2 modules.

In general, our method is based on the equality $\log(1+x) = n\log((1+x)^{\frac{1}{n}})$, which holds for any non-zero n. Since our goal is to compute $\log(1+x)$ accurately using a small number of chemical reactions in CRNs, we have two additional requirements: (i) We need $(1+x)^{\frac{1}{n}} < (1+x)$ for $x > 0$ and (ii) the computation of $(1+x)^{\frac{1}{n}}$ can be implemented accurately using only a small number of chemical reactions in CRNs. The first requirement means we need $n > 1$. The second requirement means that we need to choose n to be a power of 2, i.e. $n = 2, 4, 8, \ldots$ (where $n = 2$ corresponds to square root and $n = 4$ is the quartic root) because we can compute these roots by repeatedly computing square roots. Note that there are CRNs to compute the nth root, where n is an integer >1, see (27) for their implementation. However, the computation of roots where n is a power is 2 can be done more efficiently compared with other values of n.

For illustration, we used Pade (1,1) approximation in Step (ii) above. However, we can replace it by other Pade approximations. In order to demonstrate the efficiency of the proposed method, Figure 2 compares the function $\log(1+x)$, Qurt:Pade(2,2):×4 (short for: quartic root—Pade (2,2)—multiplication by 4) and Pade (100,100). It can be seen that Qurt:Pade(2,2):×4 is more accurate than Pade (100,100). Moreover, the implementation complexity of CRNs for Pade (100,100) is far more complicated than that of Qurt:Pade(2,2):×4. A CRN implementing Pade (100,100) will require about 100 reactions, but we will see subsequently that a CRN implementing Qurt:Pade(2,2):×4 requires only 24 reactions. This figure therefore shows that there are more efficient methods to reduce approximation error than simply increasing the order of Pade approximation.

In summary, a key contribution of this article is to propose a modular method to compute the logarithm function $\log(1+x)$ to different levels of accuracy. The proposed method can be tuned by two mechanisms. First, computation of $(1+x)^{\frac{1}{n}}$ at the beginning and multiplication by n at the end, where n is a parameter that can be tuned. Second, the order of approximation in Pade approximation can be chosen to adjust the level of accuracy.

2.3 CRN realization

In this section, we show how the computation modules of our proposed method can be implemented in CRNs. We will present four different modules, for computing square root, $Q_{1,1}(x_1 - 1)$, $Q_{2,2}(x_1 - 1)$ and multiplication by 2.

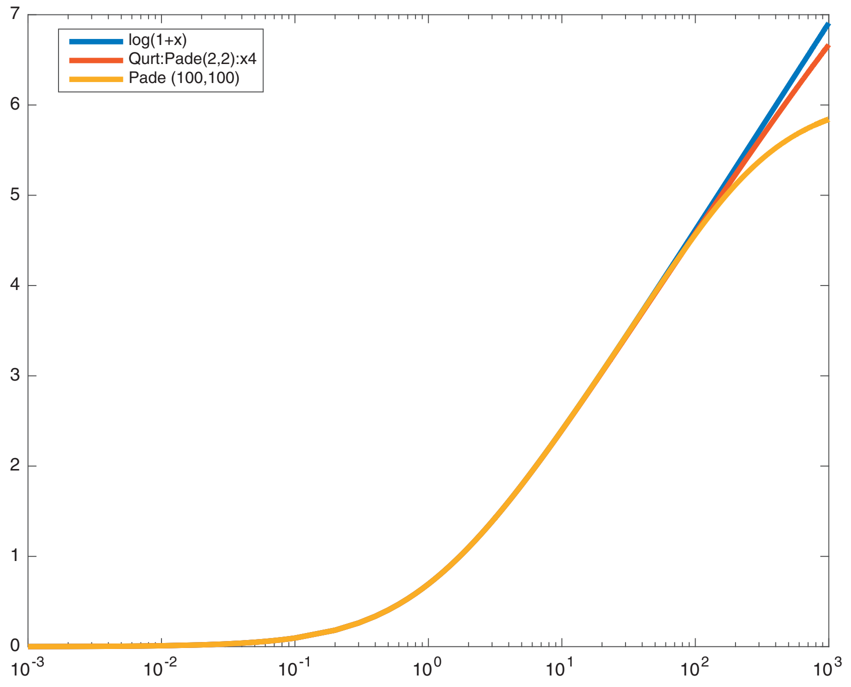

Figure 2. This figure plots $\log(1+x)$ and two different approximations of this function. The approximations are Qurt:Pade(2,2):×4 and Pade (100,100).

The following notation will be used. We will use uppercase letters, sometimes with a subscript, e.g. X and Y_1, to denote chemical species. We will use ϕ to denote a species whose concentration we are not keeping track of. The concentrations of the species at time t will be denoted by the notations $[X](t)$ and $[Y_1](t)$ for the concentrations X and Y_1 at time t, respectively. The letter k is used to denote a base reaction rate constant, and we will express all reaction rates as a multiple of base rate constant k.

The proposed algorithm to approximately compute $\log(1+x)$ takes x as the input and computes x_1, x_2 and x_3 in three steps. Both x_1 and x_2 are intermediate results and x_3 is the approximate value of $\log(1+x)$ computed by the algorithm. In the CRN description below, the concentrations of the chemicals X, X_1, X_2 and X_3 correspond to the numerical values of x, x_1, x_2 and x_3, respectively.

We assume that the number of molecules in the CRNs is large, and we can model the behavior of the CRNs using deterministic reaction rate equations.

Computing $\sqrt{1+x}$.

For a given x, the computation of $\sqrt{1+x}$ can be realized by the following CRN:

$$\begin{gathered}
C \xrightarrow{k} C + Y_1 \\
X \xrightarrow{k} X + Y_1 \\
Y_1 \xrightarrow{k} \phi \\
Y_1 \xrightarrow{k} Y_1 + 2X_1 \\
X_1 + X_1 \xrightarrow{k} \phi
\end{gathered}$$

where C, X, Y_1 and X_1 are chemical species. The chemical species C has a constant concentration of 1 unit and the initial concentration of X is x. The first three reactions are used to compute $(1+x)$, which is the steady-state concentration of the species Y_1. This is the standard method to implement addition in a CRN. The key idea is to use both chemical species C and X to produce the same chemical species Y_1. In order for the concentration of Y_1 to reach a steady state, we need to add the degradation of Y_1 which is the third reaction above. At steady state, when the rate of production of Y_1 balances its rate of degradation, the concentration of Y_1 equals to the concentration of C plus the concentration of X.

The last two reactions are used to compute $\sqrt{1+x}$, which is given by the steady-state concentration of X_1. This method of computing square root is similar to that found in (27). The chemical X_1 is produced at a rate of $2k[Y_1](t)$ (the fourth reaction; note that the multiplication factor 2 comes from the fact a molecule of Y_1 produces two molecules of X_1) and consumed at a rate of $2k[X_1](t)^2$ (the fifth reaction; note that two molecules of X_1 are consumed per reaction). Thus, at steady state, when the rate of production equals the rate of consumption, we have $[Y_1](\infty) = [X_1](\infty)^2$, i.e. the steady concentration of X_1 is the square root of the concentration of Y_1.

We can argue that the above reactions provide the correct result by writing down the mass-action kinetics. We have:

$$\begin{aligned}
\frac{d[X](t)}{dt} &= 0 \\
\frac{d[Y_1](t)}{dt} &= k[C] + k[X](t) - k[Y_1](t) \\
\frac{d[X_1](t)}{dt} &= 2k[Y_1](t) - 2k[X_1](t)^2
\end{aligned}$$

Using the fact that $[C] = 1$ and $[X](0) = x$, it can be shown that the steady-state solution is as claimed.

Computing $Q_{1,1}(x_1 - 1)$.

The aim of this subsection is to show how $Q_{1,1}(x_1 - 1)$ can be computed. From Equation (1), we have

$$Q_{1,1}(x_1 - 1) = \frac{x_1 - 1}{0.5x_1 + 0.5}$$

Note that the numerator of $Q_{1,1}(x_1 - 1)$ contains a minus 1 term. Since the concentration of a chemical species is always positive, the subtraction can in principle be handled using the framework proposed in (11) or (16). In this article, we propose a method to compute $Q_{1,1}(x_1 - 1)$ without using subtraction.

Our proposed method is to compute the numerator $x_1 - 1$ making use of the fact that $x_1 = \sqrt{1+x}$ or $x_1 = (1+x)^{\frac{1}{4}}$. If $x_1 = \sqrt{1+x}$, then it can be shown that

$$x_1 - 1 = \sqrt{1+x} - 1 = \frac{x}{\sqrt{1+x}+1} = \frac{x}{x_1 + 1}. \tag{3}$$

This means that we can compute $x_1 - 1$ by computing the ratio of x and $x_1 + 1$, and this calculation does not involve any negative number. If $x_1 = (1+x)^{\frac{1}{4}}$, then it can be shown that:

$$x_1 - 1 = \frac{x}{1 + x_1 + x_1^2 + x_1^3} \tag{4}$$

which again means that we can compute $x_1 - 1$ without using any negative number. This method can be generalized to other integral roots of $(1+x)$.

We will show how we can use a CRN to compute $x_1 - 1$ using Equation (3). The CRN is:

$$\begin{gathered}
X \xrightarrow{k} X + Y_2 \\
Y_2 \xrightarrow{k} \phi \\
X_1 + Y_2 \xrightarrow{k} X_1
\end{gathered}$$

where X and X_1 are the same chemical species that we have used earlier. Note that although X and X_1 are involved in some reactions in the above CRN, the chemical species X and X_1 appear on both sides of the reaction equations, which means their quantities are not changed by these reactions. We will use this CRN to illustrate how it can compute $y_2 = \frac{x}{x_1+1}$, where y_2 is the concentration of Y_2. In terms of concentration, the calculation that we want to perform is:

$$[Y_2] = \frac{[X]}{[X_1] + 1} \tag{5}$$

This calculation is not in the form that can be implemented by a CRN directly because it uses division. We know from mass kinetic that the rate of a chemical reaction is proportional to the product of the concentration of the reactants. Therefore, we need to rewrite this calculation in a form that has only products. We can rewrite the above calculation as:

$$0 = k[X] - k[Y_2] - k[X_1][Y_2]. \tag{6}$$

We can implement each of the terms on the right-hand side (RHS) using a chemical reaction. The first term is positive and corresponds to the production of Y_2 from X at a rate of $k[X]$; this is the first chemical reaction shown above. The second term is negative and corresponds to the degradation of Y_2; this is the second chemical reaction. The third term is negative and corresponds to X_1 and Y_2 reacting together to degrade Y_2; this is the third chemical reaction. Since the fundamental idea of this article is to use rational function to approximate the computation of logarithm, our proposed CRN will need to perform division. The above strategy for deriving a CRN that carries out division will be used throughout this article.

Formally, from the chemical reactions given above, we can write down the mass-action kinetics of Y_2 as follows:

$$\frac{d\,[Y_2](t)}{dt} = k\,[X](t) - k\,[Y_2](t) - k\,[X_1](t)\,[Y_2](t).$$

At steady state, i.e. when $t = \infty$, it can be shown that:

$$[Y_2](\infty) = \frac{[X](\infty)}{1 + [X_1](\infty)}$$

which means the steady-state concentration of Y_2 is $\frac{x}{x_1+1}$. The implementation of Equation (4) by CRN will be similar and will not be presented here.

Let $y_2 = \frac{x}{x_1+1}$, then $Q_{1,1}(x_1 - 1)$ (for the case where $x_1 = \sqrt{1+x}$) is:

$$Q_{1,1}(x_1 - 1) = \frac{y_2}{0.5x_1 + 0.5}.$$

Since this is the output of Step (ii) of our algorithm, we use x_2 to denote $\frac{y_2}{0.5x_1+0.5}$. In terms of concentration, the relation that we want to realize is $[X_2] = \frac{[Y_2]}{0.5+0.5[X_1]}$. We then follow our strategy of implementing division and rewrite this as $0 = [Y_2] - 0.5[X_2] - 0.5[X_1][X_2]$. We then implement each term by a chemical reaction. The resulting CRN is:

$$Y_2 \xrightarrow{k} Y_2 + X_2$$
$$X_2 \xrightarrow{\frac{k}{2}} \phi$$
$$X_1 + X_2 \xrightarrow{\frac{k}{2}} X_1$$

where Y_2 and X_1 are the same chemical species as before. Note that the concentrations of Y_2 and X_1 are not changed by these reactions. By writing down the mass-action kinetics of X_2, we have:

$$\frac{d\,[X_2](t)}{dt} = k\,[Y_2](t) - \frac{k}{2}\,[X_2](t) - \frac{k}{2}\,[X_1](t)\,[X_2](t).$$

At steady state, we have:

$$[X_2](\infty) = \frac{[Y_2](\infty)}{0.5 + 0.5\,[X_1](\infty)}$$

which means that the steady-state concentration of $[X_2](\infty)$ is $x_2 = Q_{1,1}(x_1 - 1)$, which is the value that we need at Step (ii) of our algorithm.

Computing $Q_{2,2}(x_1 - 1)$.
From Equation (2), we have:

$$Q_{2,2}(x_1 - 1) = \frac{(x_1 - 1)(0.5x_1 + 0.5)}{x_1 + \frac{1}{6}(x_1 - 1)^2}$$

where we have purposely written $Q_{2,2}(x_1 - 1)$ as a rational function in terms of x_1 and $x_1 - 1$. Again, we want to avoid using negative numbers and we can make use of the above method to calculate $x_1 - 1$. Let $y_2 = x_1 - 1$. We can write $Q_{2,2}$ as:

$$Q_{2,2} = \frac{y_2\,(0.5x_1 + 0.5)}{x_1 + \frac{1}{6}y_2^2} \tag{7}$$

Let Y_2 be the chemical whose concentration corresponds to the number $y_2 = x_1 - 1$. (Note: this is the same notation as above.) We again use x_2 to denote the calculation in Equation (7), because this is the output of Step (ii) of our algorithm. In terms of concentration, the relation that we want to realize is:

$$[X_2] = \frac{[Y_2]\,(0.5 + 0.5\,[X_1])}{[X_1] + \frac{1}{6}[Y_2]^2}$$

We then follow our strategy of implementing division and rewrite this as:

$$0 = 0.5\,k[Y_2] + 0.5\,k[X_1][Y_2] - k[X_1][X_2] - \frac{k}{6}[Y_2]^2[X_2]. \tag{8}$$

Although all the terms are in the form of a product of concentrations, the last term contains the product of three concentrations. Since chemical reactions generally involve only two chemical species at a time, we first compute $[Y_2]^2$ and let this be $[Y_3]$. We can now rewrite Equation (8) as:

$$0 = 0.5\,k[Y_2] + 0.5\,k[X_1][Y_2] - k[X_1][X_2] - \frac{k}{6}[Y_3][X_2]. \tag{9}$$

Each of the terms on the RHS can now be implemented by a chemical reaction, and four chemical reactions are required for these calculations. In addition, we also need to compute $[Y_3] = [Y_2]^2$, and this can be done with two chemical reactions. The CRN for implementing the computation of $Q_{2,2}$ is:

$$Y_2 \xrightarrow{\frac{k}{2}} Y_2 + X_2$$
$$X_1 + Y_2 \xrightarrow{\frac{k}{2}} X_1 + Y_2 + X_2$$
$$Y_2 + Y_2 \xrightarrow{k} Y_2 + Y_2 + Y_3$$
$$Y_3 \xrightarrow{k} \phi$$
$$X_1 + X_2 \xrightarrow{k} X_1$$
$$Y_3 + X_2 \xrightarrow{\frac{k}{6}} X_2$$

where Y_3 is an intermediate chemical species. By writing down the mass-action kinetics for the species Y_3 and X_2, we have:

$$\frac{d\,[Y_3](t)}{dt} = k\,[Y_2](t)^2 - k\,[Y_3](t)$$
$$\frac{d\,[X_2](t)}{dt} = \frac{k}{2}\,[Y_2](t) + \frac{k}{2}\Big[X_1](t)[Y_2](t) - k[X_1](t)[X_2](t) - \frac{k}{6}[Y_3](t)[X_2](t).$$

The steady-state concentrations of Y_3 and X_2 are:

$$[Y_3](\infty) = [Y_2](\infty)^2$$
$$[X_2](\infty) = \frac{[Y_2](\infty)(0.5 + 0.5\,[X_1](\infty))}{[X_1](\infty) + \frac{1}{6}[Y_3](\infty)}$$

This means that the above CRN implements the calculation in Equation (7).

Multiplication by two.
We can implement the multiplication-by-two module using the following CRN:

$$X_2 \xrightarrow{k} X_2 + 2X_3$$
$$X_3 \xrightarrow{k} \phi$$

By writing down the mass-action kinetics, we can readily show that at steady state, the concentration of X_3 is twice that of X_2. Details are omitted for brevity.

3. Results

3.1 Implementation using DNA strand displacement reactions

Using the modules presented earlier, we can realize four different CRNs:

1. Sqrt:Pade(1,1):×2
2. Sqrt:Pade(2,2):×2
3. Qurt:Pade(1,1):×4
4. Qurt:Pade(2,2):×4

These CRNs can be implemented using DNA strand displacement reactions (SDRs). SDR is chosen, because it is shown that SDR can be used to emulate both unimolecular and bimolecular reactions (28). For example, the unimolecular chemical reaction $Z_1 \rightarrow Z_2 + Z_3$ in a CRN can be implemented using the following two reactions in DNA SDR:

$$Z_1 + G_1 \rightarrow \text{ waste} + O_1 \qquad (10)$$

$$O_1 + T_1 \rightarrow \text{ waste} + Z_2 + Z_3 \qquad (11)$$

where G_1, O_1 and T_1 are auxiliary chemical species. A special case of this unimolecular reaction is $Z_1 \rightarrow^k \phi$, which can be implemented by:

$$Z_1 + G_1 \rightarrow \text{ waste} + O_1 \qquad (12)$$

where O_1 is a chemical species for which we will not keep track. Note that both Z_1 and O_1 are single-stranded DNA, while both G_1 and waste are double-stranded DNA. In the reaction, a strand in the double-stranded G_1 is replaced by Z_1 to produce the waste and O_1.

DNA SDR can also implement bimolecular chemical reactions in a CRN. In this case, a bimolecular reaction $Z_1 + Z_2 \rightarrow Z_3$ can be implemented by four DNA SDRs with five auxiliary chemical species, see (28) for implementation details. For our CRNs, we will also need bimolecular reactions of the form $Z_1 + Z_2 \rightarrow Z_1 + Z_2 + Z_3$, which can also be implemented as DNA SDRs, see (10) for implementation details.

Since both single molecular and bimolecular reactions can be implemented by DNA SDRs, this allows general CRNs to be implemented. We use the methods in (28) to turn the CRNs into their DNA implementation. In our implementation, we impose the limits mentioned in (28), namely the maximum SDR reaction rate is 10^6 M^{-1} s^{-1} and the initial concentration of the auxiliary chemical species is set to its maximum value of 10^{-5} M.

We turn the CRNs and their DNA SDR implementations into ordinary differential equations. We solve these ordinary differential equations to steady state using Matlab. (Alternatively, note that public domain tools for performing these calculations can also be found in (29,30).) We will show the results for Qurt:Pade(2,2):×4. We consider x in the range of $[0.1, 100]$. As reference, we compute the true $\log(1+x)$ and true Qurt:Pade(2,2):×4 values, which are given by the solid lines in Figure 3. For the CRN and DNA SDR implementations, we use 12 discrete values of x and record the steady-state value of the simulation output. The 12 values of x used are 0.1, 0.5 and 10 values equally spaced in the log scale in $[1, 100]$. The CRN outputs are marked by crosses in Figure 3. It can be seen that the CRN accurately approximates the Qurt:Pade(2,2):×4 computation scheme. The DNA SDR outputs are marked by circles in Figure 3. It can be seen that the DNA SDR implementation gives accurate approximation for Qurt:Pade(2,2):×4. The error is due to the fact that the DNA SDR implementation is exact only if the concentration of the auxiliary species is infinity.

3.2 Number of chemical species and number of reactions

The aim of this section is to determine the number of chemical species and chemical reactions needed to realize our proposed CRNs. We consider 9 different CRNs that realize Pade(N,N), Sqrt:Pade(N,N):×2 and Qurt:Pade(N,N):×4 with $N = 1$, 2 and 3. For our implementation, the quartic root is realized by concatenating two square root modules and multiplication by four is implemented by two multiplication-by-two modules. In Columns 2 and 3 of Table 1, we summarize the number of chemical species and chemical reactions used by these CRNs. The corresponding results for DNA SDR implementation can be found in Columns 4 and 5 of Table 1. We will focus our discussion on the number of chemical species and chemical reactions required by the CRNs.

From Table 1, we observe linear increase in the number of chemical species for all three approximation schemes. The number of reactions for Pade(N,N) increases linearly with N. However, the number of chemical reactions needed for Sqrt:Pade(N,N):×2 and Qurt:Pade(N,N):×4 do not appear to increase linearly. We will investigate this further by deriving how the number of chemical species and chemical reactions needed to implement the CRN for Pade(N,N), Sqrt:Pade(N,N):×2 and Qurt:Pade(N,N):×4 vary with N in general.

The Pade(N,N) approximation is a rational function, where both the numerator and the denominator are polynomials of degree N. For this discussion, we assume $N \leq 13$. This is because if $N \leq 13$, all the coefficients of the Pade(N,N) approximation for $\log(1+x)$ are non-negative. For $N > 13$, negative coefficients begin to appear in the Pade(N,N) approximation for $\log(1+x)$. Since CRNs handle negative numbers differently (11,16), the CRN implementation for the $N > 13$ cases will be slightly different. In addition, the $N > 13$ cases may not be practical, so we will not consider them.

The Pade(N,N) case: For any CRN implementation for Pade(N,N) approximation, we need a chemical species to represent the input value x and another for the output value; in addition, we also need chemical species to represent these powers of x: $x^2, \ldots, x^N$. Therefore, we need $(N+1)$ chemical species. Note that the coefficient for the constant term in the numerator is always zero for the Pade(N,N) approximation for $\log(1+x)$, we therefore do not require a chemical species to represent a numerical constant.

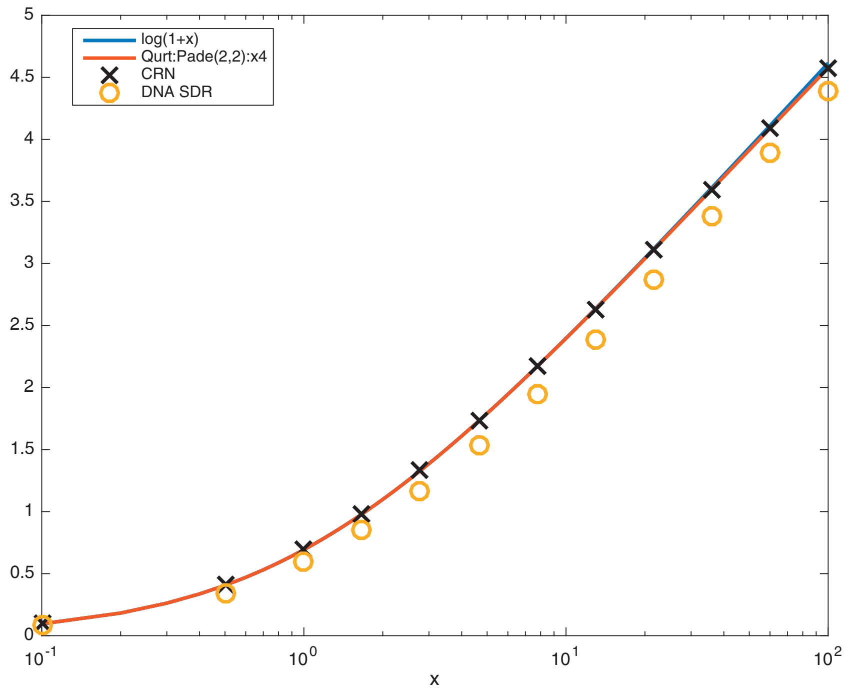

Figure 3. This figure plots log $(1+x)$, the approximation Qurt:Pade(2,2):×4, CRN implementation of Qurt:Pade(2,2):×4 (crosses) and DNA SDR implementation of Qurt:Pade(2,2):×4 (circles).

Table 1. This table shows the number of chemical species and chemical reactions required by the CRN as well as its DNA SDR implementation

Approximation method	CRN		DNA SDR	
	#Species	#Reactions	#Species	#Reactions
Pade (1,1)	2	3	14	9
Pade (2,2)	3	7	32	22
Pade (3,3)	4	11	50	35
Sqrt:Pade (1,1):×2	7	13	62	40
Sqrt:Pade (2,2):×2	8	16	81	54
Sqrt:Pade (3,3):×2	9	21	98	66
Qurt:Pade (1,1):×4	10	21	102	68
Qurt:Pade (2,2):×4	11	24	121	82
Qurt:Pade (3,3):×4	12	29	138	94

The CRN for Pade(N,N) approximation consists of two classes of reactions. The first class of reactions is to compute these powers of x: $x^2, \ldots, x^N$. This requires $2(N-1)$ reactions. The second class of reactions is to calculate the output value. Each non-zero term in the numerator and denominator polynomials requires one chemical reaction. Since there are $2(N-1)$ non-zero terms, we need $2(N-1)$ reactions. The total number of chemical reactions needed is therefore $2(N-1)+(2N+1)=4N-1$.

The number of chemical species and reactions for general N are summarized in Table 2. Note that these formulas are consistent with the results for $N=1$, 2, 3 in Table 1.

The Sqrt:Pade(N,N):×2 case: We can break down the computations of Sqrt:Pade(N,N):×2 into the following steps:

Step 1: Computation of $(1+x)$ from x with three chemical reactions.
Step 2: Computation of $x_1=\sqrt{1+x}$ from $(1+x)$ with two chemical reactions.

Table 2. This table shows the number of chemical species and chemical reactions required by the CRN assuming Pade(N,N) is used

Approximation method	CRN	
	#Species	#Reactions
Pade (N,N)	$N+1$	$4N-1$
Sqrt:Pade(N,N):×2	$N+6$	$4N+9$
Qurt:Pade(N,N):×4	$N+9$	$4N+17$

Step 3: Computation of $y_2=x_1-1$ from x_1 and x with 3 chemical reactions
Step 4: Computation of Pade(N,N) with up to $(4N-1)$ chemical reactions.
Step 5: Multiplication by two using two chemical reactions.

The number of reactions required is therefore $3+2+3+(4N-1)+2=4N+9$. The number of chemical species required is $N+6$. These results are summarized in Table 2. Note that the number of reactions presented here is upper bound rather than the exact number. For $N=2$, the general expression says that $4N+9=17$ reactions are required, but Table 1 says that Sqrt:Pade(2,2):×2 requires 16 reactions. This is because the Pade(2,2) expression in Sqrt:Pade(2,2):×2 can be written in a way with one fewer non-zero term than the standard expression; therefore, our implementation of Sqrt:Pade(2,2):×2 requires one chemical reaction fewer than the upper bound.

The Qurt:Pade(N,N):×4 case: The computation steps in Qurt:Pade(N,N):×4 are analogous to those in Sqrt:Pade(N,N):×2, except that we need to replace Steps 2, 3 and 5 above by:

Step 2: Computation of $x_1=(1+x)^{\frac{1}{4}}$ from $(1+x)$ with four chemical reactions.
Step 3: Computation of $y_2=x_1-1$ from x_1 and x with seven chemical reactions.
Step 5: Multiplication by four using four chemical reactions.

The number of reactions required is therefore $3+4+7+(4N-1)+4=4N+17$. The number of chemical species required is $N+9$. Again the number of reactions presented here is upper bound, and the Qurt:Pade(2,2):×4 in Table 1 needs one fewer chemical reaction than the upper bound because of the same reason discussed before.

4. Discussion

4.1 Comparing Pade approximation to other approximations

A justification for using Pade approximation is that it is a more accurate approximation compared to other well-known approximations of the logarithm, such as Taylor series and the inverse hyperbolic tangent series approximation (31). The Taylor series expansion of $\log(1+x)$ up to degree L is given by:

$$\log(1+x) \approx x - \frac{1}{2}x^2 + \frac{1}{3}x^3 - \ldots + \frac{1}{L}(-1)^{L+1}x^L$$

As mentioned earlier, the coefficients of the Pade (M, N) approximation $Q_{M,N}(x)$ are chosen such that its first $(M+N)$ coefficients match the first $(M+N)$ coefficients of the Taylor series expansion. A fair comparison of Pade and Taylor series is to compare $Q_{M,N}(x)$ against Taylor series of degree $L=M+N$. We use absolute value of the approximation error as the criterion, e.g. for Pade approximation, we consider $|\log(1+x) - Q_{M,N}(x)|$. The top graph of Figure 4 compares the absolute approximation error of $Q_{2,2}(x)$ against Taylor series of degree $L=4$. It can be seen that Pade approximation gives a better approximation over a wider range of x. Note that the graph compares the approximation error for x in the interval $(-1, 0)$ too. Our CRN will not be operating in this range of x, but we have included this range of x for a discussion point later.

The inverse hyperbolic tangent series approximation for $\log(1+x)$ is:

$$\log(1+x) = 2\tanh^{-1}\left(\frac{x}{x+2}\right)$$
$$\approx 2\left(\frac{x}{x+2} + \frac{1}{3}\left(\frac{x}{x+2}\right)^3 + \frac{1}{5}\left(\frac{x}{x+2}\right)^5 + \ldots + \frac{1}{H}\left(\frac{x}{x+2}\right)^H\right)$$

The inverse hyperbolic tangent series is known to be more accurate for large x and there is a CRN implementation (32). Note that the inverse hyperbolic tangent series approximation for $H=1$ is identical to the Pade (1, 1) approximation. A fair comparison is to compare $Q_{N,N}(x)$ approximation with an inverse hyperbolic tangent series with $H=N$. This is because we will be comparing two rational function approximations with both numerator and denominator being polynomials in x of degree N. The bottom graph of Figure 4 compares the absolute approximation error of Pade (3,3) against inverse hyperbolic tangent series approximation for $H=3$. It can be seen that Pade approximation is more accurate.

4.2 Complexity and accuracy

In the discussion under Section 4.1, we used the same order of approximation and compared the accuracy of different approximation methods. This is the standard method used in mathematics. However, the order of approximation may not truly reflect the complexity of a CRN. Here, we will use the number of reactions as a measure of complexity and compare the accuracy of Pade, Taylor series and inverse hyperbolic tangent series for a given number of reactions. For a given number of chemical reactions, if an approximation method has a better accuracy, then it is a better approximation method.

For comparison, we will vary the order of each approximation method. We know from Section 3.2 that Pade(N,N) requires $4N-1$ chemical reactions. For fair comparison, we will realize the CRNs for Taylor series and inverse hyperbolic tangent series in exactly the same manner as that of Pade(N,N).

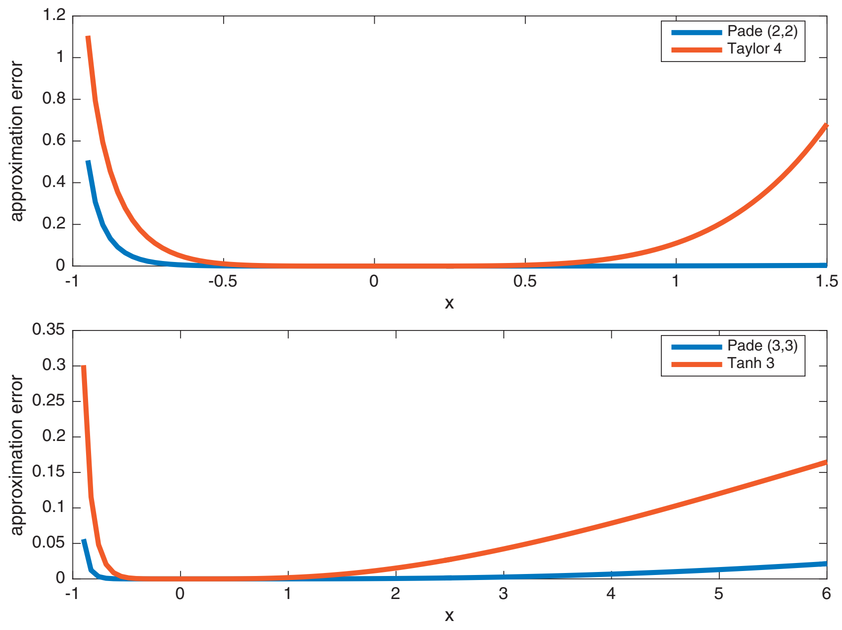

Figure 4. This figure compares the absolute value of the approximation error of Pade, Taylor series and inverse hyperbolic tangent series of $\log(1+x)$. The top graph compares Pade (2,2) approximation against a Taylor series expansion of degree 4. The bottom graph compares Pade (3,3) approximation against inverse hyperbolic tangent series of up to degree 3.

For a Taylor series of degree N (with $N \geq 1$), its CRN can be realized by $3N+1$ reactions. These reactions come from the following: (i) the computation of $x^2, \ldots, x^N$ requires $2(N-1)$ reactions; (ii) we need $N+2$ reactions to obtain the positive and negative terms; and (iii) one reaction is needed to subtract the negative term from the positive one.

For the inverse hyperbolic tangent series with N terms (with $N \geq 1$), its CRN uses $(3N+4)$ reactions. These reactions come from the following: (i) The computation of $\frac{x}{x+2}$ requires three chemical reactions. (ii) Let $z = \frac{x}{x+2}$. We need to compute $z^3, z^5, \ldots, z^{2N-1}$. An efficient method is to first compute z^2 and then compute z^3 from $z \times z^2$, z^5 from $z^3 \times z^2$, etc. These computations require $2N$ reactions. (iii) To sum the N terms up, we need $N+1$ reactions.

We first consider an input value of $x=0.1$. For each of Pade, Taylor and inverse hyperbolic tangent approximation, we vary their order to compute different approximations of $\log(1+x)$. In Figure 5, we plot the approximation error (in log scale) versus the number of chemical reactions. It can be seen that when the number of reactions increases (by increasing the order), the approximation error decreases. The figure shows that for a given number of chemical reactions, the Pade approximation gives the best accuracy. Since Taylor series gives a lot worse results, we will not consider it any more.

We next consider an input value of $x=5$. We compare only Pade and inverse hyperbolic tangent series. In Figure 6, we plot the approximation error (in linear scale) versus the number of chemical reactions. It can be seen that for a given number of chemical reactions, the Pade approximation again gives the best accuracy.

4.3 Error analysis

In order to understand why the proposed method works better, here we analyze the error in using Sqrt:Pade(1,1):×2 for approximating $\log(1+x)$. Define

$$\Delta(x) = \log(1+x) - Q_{1,1}(x)$$

which is the approximation error in using Pade (1,1) approximation. If Sqrt:Pade(1,1):×2 is used, the approximation error $E(x)$ is:

$$E(x) = \log(1+x) - 2Q_{1,1}(\sqrt{1+x}-1)$$

Using the definition of $\Delta(x)$, we have

$$Q_{1,1}(\sqrt{1+x}-1) = \log(\sqrt{1+x}) - \Delta(\sqrt{1+x}-1)$$

Therefore, the approximation error $E(x)$ in using Sqrt:Pade(1,1):×2 can be written as:

$$E(x) = 2\Delta(\sqrt{1+x}-1)$$

which is equal to twice the error of using Pade (1,1) approximation to calculate $\log(\sqrt{1+x})$. The reason why Sqrt:Pade(1,1):×2 works better is because the approximation error in calculating $\log(\sqrt{1+x})$ using Pade (1,1) approximation is less than half of that calculating $\log(1+x)$ using Pade (1,1) approximation. Although our analysis is based on Pade(1,1) approximation, the argument can also be applied to other approximations of $\log(1+x)$.

4.4 Does scaling down the number work?

A key contribution of this article is to compute $\log(\sqrt{1+x})$ instead of $\log(1+x)$ because $\log(\sqrt{1+x})$ can be computed more accurately. Another way to obtain a smaller number from $(1+x)$ is to compute $\frac{1+x}{b}$, where $b>1$. This will allow us to compute $\log(1+x)$ via:

$$\log(1+x) = \log\left(\frac{1+x}{b}\right) + \log(b).$$

For this method, one first computes $\log(\frac{1+x}{b})$, which is the logarithm of a smaller number and adds the result to $\log(b)$ to

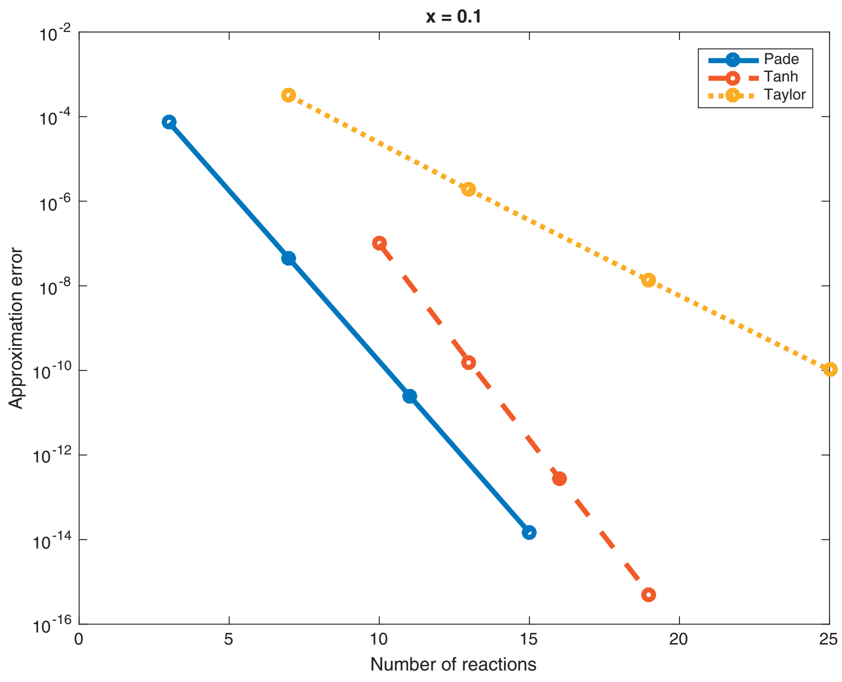

Figure 5. The plot compares the CRN implementation for Pade, Taylor series and inverse hyperbolic tangent series for computing $\log(1+x)$ with $x=0.1$. Vertical axis is the absolute approximation error. The horizontal axis is the number of reactions.

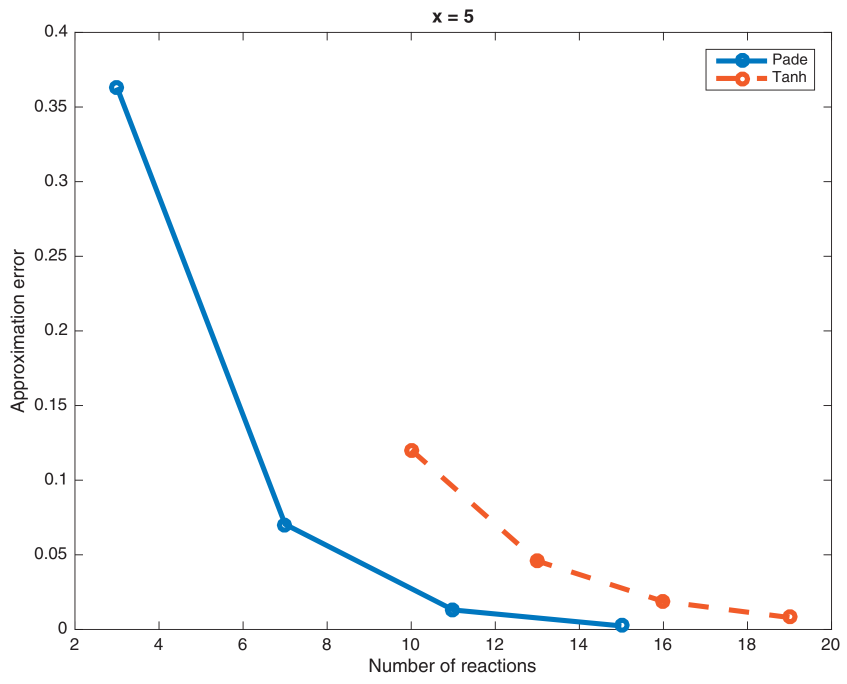

Figure 6. The plot compares the CRN implementation for Pade and inverse hyperbolic tangent series for computing $\log(1+x)$ with $x = 5$. Vertical axis is the absolute approximation error. The horizontal axis is the number of reactions.

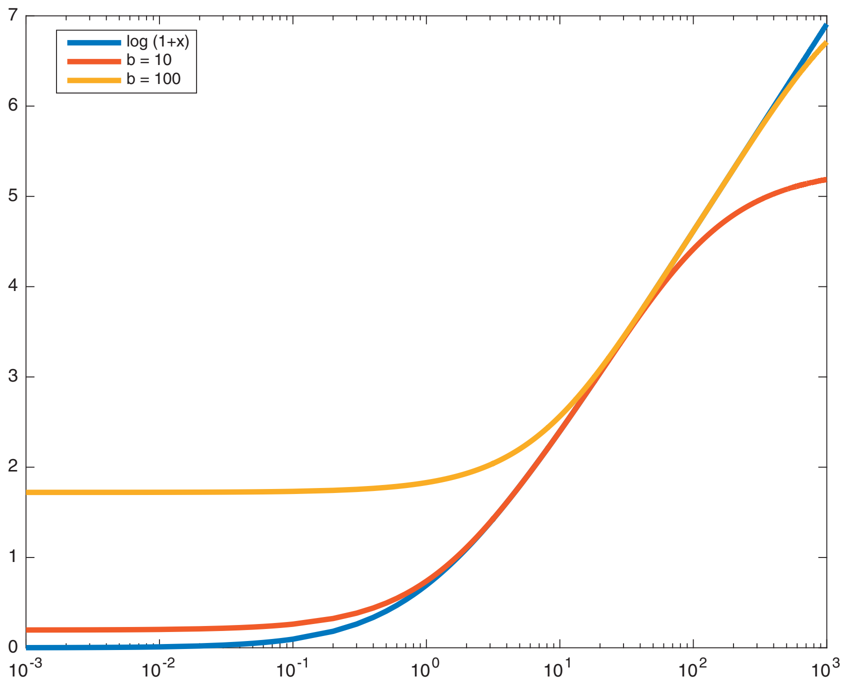

Figure 7. This figure plots $\log(1+x)$ against an approximation method based on computing $\log\left(\frac{1+x}{b}\right)$ for $b = 10$ and $b = 100$.

obtain $\log(1+x)$. Figure 7 shows the values of $\log(1+x)$ calculated by this method with $\log\left(\frac{1+x}{b}\right)$ computed using Pade (2,2) approximation and for $b = 10$ and $b = 100$. It can be seen from Figure 7 that this method does not work well for small values of x.

We will use the approximation error graph in Figure 4 to explain why this method does not work for small x. As an example, consider $x = 9$ and $b = 100$. This method requires us to compute $\log\left(\frac{1+x}{b}\right)$ which is $\log(0.1)$. Since our function approximates $\log(1+x)$, we therefore have to compute $Q_{2,2}(-0.9)$ which approximates $\log(0.1) = \log(1-0.9)$. It can be seen from Figure 4 that Pade approximation gives poorer approximation for x near −1, and this explains why this scaling method does not work well for small x. It can also be seen from Figure 4 that this method will also face the same problem if other approximation methods are used.

4.5 Pade approximation and logarithm sensing in biochemical systems

Recently, (21) shows that many allosteric proteins act as logarithm sensors. For example, consider a G-coupled protein receptor, which measures an input concentration c and uses the fraction of Guanosine triphosphate (GTP)-bound α-unit, denoted by $\hat{\alpha}_{\text{GTP}}$, as the output. It is shown in Ref. (21) that, at steady state, $\hat{\alpha}_{\text{GTP}}$ and c are related by an input–output response function of the form:

$$\hat{\alpha}_{\text{GTP}} = \frac{w_1 c}{1 + w_2 c} \tag{13}$$

where w_1 and w_2 are coefficients. The authors in (21) argue that the RHS of Equation (13) computes logarithm approximately when the concentration of c is around $\frac{1}{w_2}$. In fact, (21) has made use of Pade approximation without mentioning it. Their argument is based on the approximation:

$$\frac{L}{1+L} \approx \frac{1}{4}\log\,(L) + \frac{1}{2} \tag{14}$$

which can be derived from equations [S8] and [S9] in the supplementary information of (21). If we substitute $L = 1 + x$ in Equation (14), then after some manipulations, we get $\log\,(x) \approx \frac{x}{1+0.5x}$, which is the Pade(1,1) approximation in Equation (1). In order to see how Equation (13) computes the logarithm, we first rewrite c as $c = \frac{1}{w_2} + \Delta c$, where Δc is the deviation of the concentration from $\frac{1}{w_2}$. It can be shown that:

$$\begin{aligned}\frac{w_1 c}{1 + w_2 c} &= \frac{w_1}{4w_2}\frac{w_2\,\Delta c}{1 + \frac{1}{2}w_2\,\Delta c} + \frac{w_1}{2w_2} \\ &\approx \frac{w_1}{4w_2}\log\,(1 + w_2\,\Delta c) + \frac{w_1}{2w_2}\end{aligned}$$

where Equation (14) or Pade(1,1) approximation is used to obtain the result in the second line. This shows that, when the input concentration c is around $\frac{1}{w_2}$, the output is a logarithm function of the change of input concentration Δc.

Interestingly, we want to point out that Equation (13) also computes logarithm in another range of input concentration. First, note that the Pade (1,1) approximation in Equation (1) holds when x is small; e.g., if we limit the relative approximation error to 10%, then the approximation holds for x in the interval $[0,\,0.41]$. By comparing the RHS of Equation (13) and the Pade (1,1) approximation, it can be shown that:

$$\frac{w_1 c}{1 + w_2 c} \approx \frac{w_1}{2w_2}\log\,(1 + 2w_2 c)$$

Therefore, if the input concentration c is in the interval $\left[0,\,\frac{0.41}{2w_2}\right]$, then it computes logarithm to a 90% accuracy.

An interesting question is what biochemical reactions can compute logarithm. In fact, the article (21) finds many allosteric protein that can compute logarithm in some range of concentration. We have seen above that if the input–output response is of the form on the RHS side of Equation (13), then there are two ranges of concentration that this response computes logarithm, though two different logarithm functions. It is also shown in (21) that logarithm computation also holds for certain input concentration range for some Monod–Wyman–Changeux (MWC) or Hill models. Therefore, one may ask when an input–output response function computes logarithm. One answer to this question lies with Pade approximation because Pade approximation tells us which rational functions can be used to compute algorithm. Since the input–output response of Equation (13), MWC and Hill models all has the form of a rational function, we can therefore start with the rational input–output response of a set of biochemical reactions and use Pade approximation to check whether this response may compute logarithm.

Finally, we remark that rational function response with higher order polynomials can also be found in gene regulatory circuits (33). This opens the possibility of designing synthetic gene expression circuits that implement higher order Pade approximations.

4.6 Square roots and quartic roots as primitive computation elements in CRNs

Until now, we have presented our method as an algorithm with three computation steps. Here, we present a closed-form expression for our proposed method of approximating $\log\,(1+x)$ in terms of x. For example, for Sqrt:Pade(5,5):×2, our approximation is:

$$\log\,(1+x) \approx \frac{2x}{1+x_1}\frac{0.018x_1^5 + 0.306x_1^4 + 1.306x_1^3 + 2x_1^2 + x_1}{0.004x_1^5 + 0.119x_1^4 + 0.833x_1^3 + 2.222x_1^2 + 2.5x_1 + 1},$$

where $x_1 = \sqrt{1+x}$. Assuming that $x \gg 1$ such that $\sqrt{1+x} \approx x^{\frac{1}{2}}$, then this approximation has the form:

$$\log\,(1+x) \approx \frac{\text{Numerator polynomial in } x^{\frac{1}{2}}}{\text{Denominator polynomial in } x^{\frac{1}{2}}}.$$

An important point to note is that our approximation uses non-integral powers (or exponents) of the input x. To the best of our knowledge, previous work on CRNs has so far been focusing on integral powers of the input, e.g. polynomials with integral powers (10) and rational functions with integral powers (11,12). Therefore, a novelty of our approach is the use of non-integral exponents.

We argue that certain non-integral exponents—namely square roots, quartic roots and so on—may have a special role to play in CRN. This is because these roots can be easily calculated by CRNs. For example, the square root of a number can be calculated using a CRN with only two chemical reactions. Interestingly, this is the same number of chemical reactions required to compute the square of a number. Hence, if we measure the complexity of a computation in CRN using the number of chemical reactions required to implement that computation, then the complexity of calculating the square root is the same as that of calculating the square. However, this is not true for digital computation where the complexity of calculating the square root is much higher. This may explain why our proposed method has not been considered in digital computation. More importantly, the implication for this discussion on computing using chemical is that it is worthwhile incorporating square root, quartic root calculations into algorithms designed for CRN because these calculations have low complexity. We have seen earlier in Figure 2 that the improvement can be drastic: Qurt:Pade(2,2):×4 requires only 24 reactions and is more accurate than Pade (100,100), which requires about 100 reactions.

4.7 Relation to (18): non-integral Hill coefficients and extending the range of approximation

In the gene expression circuit for computing $\log\,(1+x)$ in (18), one of the blocks is based on the cooperative binding of an inducer to a transcription factor to form an inducer–transcription factor complex (or complex for short). This block quantitatively relates the total inducer concentration I_T, the total transcription concentration T_T and the concentration of the complex T_C. It is shown in equation (16) in the supplementary information of (18) that the relationship between I_T, T_T and T_C has the form:

$$T_C \approx \frac{w_1 I_T^{h_1} T_T}{1 + w_2 * I_T^{h_1} + w_3 I_T^{h_2} T_T} \tag{15}$$

where w_1, w_2 and w_3 are coefficients, and h_1 and h_2 are Hill coefficients. In supplementary table S2 of (18), $h_1 = 1.4$ and $h_2 = 1.05$ for one circuit and $h_1 = 3$ and $h_2 = 2.5$ for another circuit. Note that this computation block calculates the ratio of two polynomials with non-integral power of the input I_T.

This discussion shows that it may be possible to implement our proposed method as a rational function for non-integral powers of the input using gene expression circuits. For our proposed method, the non-integral power comes from taking the square root of the input in order to 'compress' its range. There is a possibility that the non-integral Hill coefficients in Equation (15) have the same purpose. Furthermore, an interesting future research question is to explore the role of fractional Hill coefficient in computation of cells.

We would also like to contrast how this article and that of (18) extend the range of approximation. For this work, we use square root, quartic root, etc. to extend the range of approximation. A larger range of approximation is achieved using a higher order root. Our method is open loop because feedback is not used. In contrast, the work (18) uses positive feedback to extend the range of approximation. The circuit in (18) computes $\log(1 + I_T)$, where I_T is the total concentration of the inducer as defined above. An issue that limits the range of the circuit in (18) is the saturation of the transcription factor; this is because the validity of Equation (15) requires that the transcription factor is not saturated. Instead of using a fixed amount of transcription factor T_T, the circuit in (18) uses a positive feedback circuit to dynamically adjust the total amount of transcription factor. If the input—which is the amount of inducer I_T—is small, the circuit needs only a small amount of transcription factor T_T. However, if the input is large, the positive feedback in the circuit increases the total amount of transcription factor to prevent saturation.

In principle, it may be possible to modify our proposed method so that the range of approximation is dynamically adjusted according to the input x using positive feedback. Recall that a key step in our work is to compute $(1 + x)^{\frac{1}{n}}$. The range of approximation in our method is determined by the parameter n. The idea is to make n bigger when the input x is bigger, and *vice versa*. However, the implementation of this idea can be difficult. This is because different sets of chemical reactions need to be used for different values of n.

4.8 Direct implementation of CRN

An open research problem is whether we can implement the proposed CRN directly. We see that it may be possible to implement the proposed method using either allosteric protein or gene expression circuits. We will discuss two possible methods here. For the first possible method, we can view our proposed CRN as a two-stage computation process, where the first stage computes the square root (or a higher order root) and the second stage computes a rational function. The article (18) presents a gene expression circuit which, for a given input y, can compute $y^{0.7}$. It may be possible to design a similar circuit to compute $y^{0.5}$ (square root) or $y^{0.25}$ (quartic root). Alternatively, since $y^{0.5} \approx (y^{0.7})^{0.7}$, it may be possible to compute the square root by concatenating two circuits that compute $y^{0.7}$, assuming that the two circuits can operate independently. For the second stage, there are examples of gene expression circuits that can compute rational function and these may be used. For the second possible method, we can explore the fact that the overall input–output response function of our proposed method can be expressed as rational function whose polynomials have non-integral exponents. This type of input–output response functions have been observed in allosteric protein or gene expression circuits, where the Hill coefficient is not an integer. It is an interesting open research problem to explore these possibilities further.

4.9 Using only one reaction rate constant

The chemical reactions in the CRN proposed in Section 2.3 uses multiple reaction rate constants. However, for the implementation of CRNs using SDRs, it may be desirable to have all chemical reactions in the CRN using the same reaction rate constant. This is because SDR can only realize different reaction rate constants with limited precision. In this discussion, we explain how we can modify our design so that all reactions use the same reaction rate constant of the base value k.

The place where our design uses multiple reaction rate constants is in the computation of Pade approximations $Q_{1,1}(x_1{-}1)$ and $Q_{2,2}(x_1{-}1)$. The general problem is the computation of a term of the form $a y_1^m$, where a is the coefficient and m is the exponent. Since the computation of y_1^m can be computed with a CRN with only one reaction rate constant, the remaining problem is to multiply the y_1^m by a constant a using only one reaction rate constant. We consider the generic problem of computing $a z_1$, where z_1 has already been computed by another CRN. The following CRN computes $z_2 = a z_1$ with only one reaction rate constant.

$$D_1 + Z_1 \xrightarrow{k} D_1 + Z_1 + Z_2$$
$$D_2 + Z_2 \xrightarrow{k} D_2$$

where D_1 and D_2 are two chemical species with constant concentrations of d_1 and d_2 respectively. It can be shown that at steady state, we have

$$z_2 = \frac{d_1}{d_2} z_1.$$

Therefore, by choosing $\frac{d_1}{d_2}$ to be a, we can realize the desired computation.

5. Conclusions

This article proposes a novel method to compute the logarithm $\log(1 + x)$ using CRNs. Two key steps in the method are the computation of $(1 + x)^{\frac{1}{n}}$ (where n is an integer) and the computation of $\log((1 + x)^{\frac{1}{n}})$ using Pade approximation. The accuracy of the proposed method can be increased using a larger n or higher order Pade approximation. We have presented the chemical reactions needed to realize the CRNs. We have also presented an implementation of the CRN using DNA SDRs. We have demonstrated that our proposed CRN can accurately compute logarithm over a wide range of *x* with a small number of chemical reactions.

In this article, we have evaluated the quality of our proposed CRNs to compute logarithm using two criteria: wide input range and small number of chemical reactions. We have also assumed the number of molecules in the CRNs is large, which means we can use the deterministic reaction rate equations to determine the CRN output. However, in reality, chemical reactions are noisy. Also, we have also assumed that the input concentration is deterministic and constant, while real input concentration can be noisy and time varying. It will be interesting to study how the accuracy of our proposed CRNs is affected by noisy chemical reactions and noisy inputs. It will also be interesting to see the dynamic response of the proposed CRNs when the input is time varying. These are open problems to be studied.

Conflict of interest statement. None declared.

References

1. Alberts, B., Johnsosn, A., Lewis, J., Raff, M., Roberts, K. and Walter, P. (2007) *Molecular Biology of the Cell*, 5 edn. Garland Science, New York, NY.
2. Lim, W., Mayer, B. and Pawson, T. (2014) *Cell Signaling*. Garland Science, New York, NY.
3. Ma, K.C., Perli, S.D. and Lu, T.K. (2016) Foundations and emerging paradigms for computing in living cells. *J. Mol. Biol.*, 428, 893–915.
4. Riccione, K.A., Smith, R.P., Lee, A.J. and You, L. (2012) A synthetic biology approach to understanding cellular information processing. *ACS Synth. Biol.*, 1, 389–402.
5. Hennig, S., Rödel, G. and Ostermann, K. (2015) Artificial cell-cell communication as an emerging tool in synthetic biology applications. *J. Biol. Eng.*, 9, 289.
6. Érdi, P. and Tóth, J. (1989) *Mathematical Models of Chemical Reactions: Theory and Applications of Deterministic and Stochastic Models*. Manchester University Press, Manchester, UK.
7. Horn, F.and Jackson, R. (1972) General mass action kinetics. *Arch. Ration. Mech. Anal.*, 47, 81–116.
8. Soloveichik, D., Cook, M, Winfree, E. and Bruck, J. (2008) Computation with finite stochastic chemical reaction networks. *Nat. Comput.*, 7, 615–633.
9. Magnasco, M.O. (1997) Chemical kinetics is Turing universal. *Phys. Rev. Lett.*, 78, 1190–1193.
10. Salehi, S.A., Parhi, K.K. and Riedel, M.D. (2017) Chemical reaction networks for computing polynomials. *ACS Synth. Biol.*, 6, 76–83.
11. Oishi, K. and Klavins, E. (2011) Biomolecular implementation of linear I/O systems. *Syst. Biol. IET*, 5, 252–260.
12. Foo, M., Sawlekar, R., Kim, J., Stan, G., Bates, D. and Kulkarni, V. (2016) On the biomolecular implementation of nonlinear system theoretic operators. In: *Proceedings of the European Control Conference*, 29 June–1 July 2016. Aalborg, Denmark. pp. 1824–1831.
13. Zechner, C., Seelig, G., Rullan, M. and Khammash, M. (2016) Molecular circuits for dynamic noise filtering. *Proc. Natl Acad. Sci. U. S. A.*, 113, 4729–4734.
14. Teo, J.J.Y., Woo, S.S. and Sarpeshkar, R. (2015) Synthetic biology: a unifying view and review using analog circuits. *IEEE Trans. Biomed. Circuits Syst.*, 9, 453–474.
15. Sarpeshkar, R. (2014) Analog synthetic biology. *Philos. Trans. A Math. Phys Eng. Sci.*, 372, 20130110.
16. Song, T., Garg, S., Mokhtar, R., Bui, H. and Reif, J. (2016) Analog computation by DNA strand displacement circuits. *ACS Synth. Biol.*, 5, 898–912.
17. Perli, S.D.and Lu, T.K. (2016) Analog synthetic gene networks. In: *Proceedings of the 3rd ACM International Conference on Nanoscale Computing and Communication (ACM NanoCom 2016)*, 28 September–30 September 2016. New York, NY, USA. Article number 10.
18. Daniel, R., Rubens, J.R., Sarpeshkar, R. and Lu, T.K. (2013) Synthetic analog computation in living cells. *Nature*, 497:619–623.
19. Adler, M., Mayo, A. and Alon, U. (2014) Logarithmic and power law input-output relations in sensory systems with fold-change detection. *PLoS Comput. Biol.*, 10:e1003781.
20. Goentoro, L., Shoval, O, Kirschner, M.W. and Alon, U. (2009) The incoherent feedforward loop can provide fold-change detection in gene regulation. *Mol. Cell*, 36:894–899.
21. Olsman, N. and Goentoro, L. (2016) Allosteric proteins as logarithmic sensors. *Proc. Natl Acad. Sci. U. S. A.*, 113, E4423–E4430.
22. Siggia, E.D. and Vergassola, M. (2013) Decisions on the fly in cellular sensory systems. *Proc. Natl Acad. Sci. U. S. A.*, 110, E3704–E3712.
23. Chou, C.T. (2015) Maximum a-posteriori decoding for diffusion-based molecular communication using analog filters. *IEEE Trans. Nanotechnol.*, 14, 1054–1067.
24. Awan, H. and Chou, C.T. (2017) Generalized solution for the demodulation of reaction shift keying signals in molecular communication networks. *IEEE Trans. Commun.*, 65, 715–727.
25. Chou, C.T. (2015) A Markovian approach to the optimal demodulation of diffusion-based molecular communication networks. *IEEE Trans. Commun.*, 63, 3728–3743.
26. Baker, G.A.J. and Graves-Morris, P.R. (1996) *Pade Approximants*. Cambridge University Press, Cambridge, UK.
27. Buisman, H.J., ten Eikelder, H.M.M., Hilbers, P.A.J. and Liekens, A.M.L. (2009) Computing algebraic functions with biochemical reaction networks. *Artif. Life*, 15, 5–19.
28. Soloveichik, D., Seelig, G. and Winfree, E. (2010) DNA as a universal substrate for chemical kinetics. *Proc. Natl Acad. Sci. U. S. A.*, 107, 5393–5398.
29. *CRNSimulator Mathematica Package*. http://users.ece.utexas.edu/∼soloveichik/crnsimulator.html (20 March 2017, date last accessed).
30. *Visual DSD: A Design and Analysis Tool for DNA Strand Displacement Systems*. https://www.microsoft.com/en-us/research/project/programming-dna-circuits/ (20 March 2017, date last accessed).
31. Abramowitz, M. and Stegun, I. (eds). (1964) *Handbook of Mathematical Functions*. Dover, New York, UK.
32. Foo, M., Sawlekar, R. and Bates, D.G. (2016) Exploiting the dynamic properties of covalent modification cycle for the design of synthetic analog biomolecular circuitry. *J. Biol. Eng.*, 10, 15.
33. Bintu, L., Buchler, N.E., Garcia, H.G., Gerland, U., Hwa, T., Kondev, J. and Phillips, R. (2005) Transcriptional regulation by the numbers: models. *Curr. Opin. Genet. Dev.*, 15, 116–124.