Bounded Analog Complexity

Ho-Lin Chen
Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan

Xiang Huang
Department of Computer Science, University of Illinois Springfield, IL, USA

Abstract

Current analog complexity theory, built on the General-Purpose Analog Computer (GPAC) model and polynomial ODEs, allows unbounded state variables – an assumption that is physically unrealistic for chemical reaction networks and other laboratory-scale analog computers. We develop a *bounded analog complexity theory* in which all state variables remain in compact intervals and physical time is the only diverging resource.

Our main technical contribution is *bounded surrogate compilation*, a compilation framework that transforms unbounded polynomial ODE systems into bounded ones while preserving computational limits and time-to-precision guarantees. We prove that on compact domains, physical time and trajectory length differ by at most constant factors; combined with the Bournez–Graça–Pouly characterization, this yields: bounded-GPAC polynomial time equals **P** over the reals.

We exhibit concrete constructions demonstrating fine-grained bounded time complexity – a tunable polynomial-degree family, a Lambert-W–based system achieving $\Theta(r \log r)$ time-to-precision (where r is the desired precision parameter, in nats: $|x(t) - \alpha| < e^{-r}$), and an iterated-logarithm tower realizing arbitrarily high complexity classes – all for the task of computing the constant 1. We show that bounded GPACs are closed under exponentiation (α^β) with time complexity equal to the harder input, and that the full GPAC-to-CRN compilation pipeline preserves time complexity class via a low-pass filter analysis of readout modules.

2012 ACM Subject Classification Theory of computation → Models of computation; Theory of computation → Computational complexity and cryptography

Keywords and phrases Analog computation, GPAC, bounded complexity, chemical reaction networks, polynomial ODE

Digital Object Identifier 10.4230/LIPIcs.DNA.32.14

Supplementary Material *Software*: `https://colab.research.google.com/github/zinan-huang/notebooks/blob/main/bounded_gpac_exponentiation.ipynb`

Funding *Ho-Lin Chen*: Supported in part by NSTC (Taiwan) grant 113-2221-E-002-204-MY3.
Xiang Huang: Supported in part by Department of Energy EXPRESS grant DE-SC0024278.

Acknowledgements We thank the anonymous DNA32 reviewers for their detailed and constructive feedback.

1 Introduction

This paper develops a time complexity theory for chemical reaction network (CRN) computation in which all species concentrations remain bounded – as they must in any physical implementation. Our main technical contribution is a compilation method – *bounded surrogate compilation* – that transforms any polynomial ODE system into a bounded one while preserving all computed limits. In a bounded system, physical time – the time parameter t of the ODE, representing the actual elapsed time of the system – is a faithful complexity resource: it agrees with trajectory length up to constant factors.

We work primarily with GPACs (polynomial ODEs), which offer a larger design space and greater algebraic flexibility than CRNs. However, our ultimate target is CRN implementation: every bounded GPAC construction in this paper can be translated into a CRN via dual-rail encoding and a readout subtraction module (Section 7). We prove that the subtraction module – the only step that could degrade precision – preserves time complexity.

We begin by recalling the line of work that leads to the present paper.

1.1 From computable numbers to analog complexity

(i) Turing [19] introduced the notion of *computable numbers*: real numbers whose digits can be produced by a mechanical procedure.

(ii) Shannon [17] introduced the General-Purpose Analog Computer (GPAC) as a mathematical abstraction of Bush's differential analyzer. After refinements by Pour-El [16], Lipshitz–Rubel [15], and Graça–Costa [10], GPACs are now characterized by polynomial initial value problems (PIVPs): systems of the form $y' = p(y)$, $y(0) = y_0$, where p is a vector of polynomials with rational coefficients.

(iii) Bournez, Graça, and Pouly [4] proved that GPACs, measured by *trajectory length* $L(t) = \int_0^t \|y'(s)\|\, ds$, are equivalent to Turing machines in both computability and complexity. In particular, polynomial trajectory length characterizes exactly the class **P** of polynomial-time computable functions.

(iv) Huang et al. [13] defined CRN-computable numbers: real numbers computed as limits of chemical reaction network (CRN) trajectories. Notably, in their framework *all species concentrations are required to be bounded.* This ensures that the time parameter t of the ODE aligns with the notion of physical time, preventing Zeno-type speedups caused by diverging variables. Under this condition, the class of real-time CRN-computable numbers was shown to be a field containing transcendental numbers such as e and π. Our complexity theory can be viewed as extending this framework from a single convergence rate (real-time) to a full hierarchy of rates.

(v) Huang and Huls [12] extended these ideas to large-population protocols (LPPs), showing that LPPs compute the same class of real numbers as GPACs and CRNs. Population protocols are inherently bounded: species live on a probability simplex.

A common thread runs through items (iv) and (v): the computations are bounded by design, as dictated by chemistry. No test tube holds infinite molecules; no population exceeds its total count. In these settings, the notion of time is unambiguous – physical time *is* the complexity resource, because no cost can be hidden in diverging auxiliary variables.

In contrast, the Bournez–Graça–Pouly framework in (iii) permits unbounded state variables – mathematically convenient but physically unrealistic for CRN implementations. This is why trajectory length, rather than physical time, serves as their complexity measure. Our goal is to bridge this gap: develop a complexity theory that respects the bounded nature of chemical systems while retaining the full power of the GPAC/PIVP framework as a design tool.

1.2 Why trajectory length, and why we depart from it

In the Bournez–Graça–Pouly framework, the complexity resource is not physical time t but the *trajectory length* (arc length) $L(t) = \int_0^t \|y'(s)\|\, ds$. This is the total distance traveled by the state vector in $\mathbb{R}^d$, measured by integrating the *speed* $\|y'(t)\|$ of the solution curve. When variables are unbounded, the speed $\|y'(t)\| = \|p(y(t))\|$ often grows without bound

as well, since p is a polynomial evaluated at a growing argument. In such cases, $L(t)$ can diverge even over a finite time interval, and arc length and physical time decouple: L may be infinite while t is finite.

A concrete example makes this vivid. Consider $x'(t) = (1-x)(1+v)^2$, $v'(t) = (1+v)^2$, $x(0) = v(0) = 0$. The output $x(t) = 1 - e^{1-1/(1-t)}$ converges to 1 at the finite time $t = 1$ – a naïve reading suggests this system "computes 1 in under one second." But the auxiliary variable $v(t) = 1/(1-t) - 1$ diverges, and the trajectory speed $\|y'(t)\| \geq (1+v)^2 = 1/(1-t)^2$ explodes. The arc length $L(t) \geq t/(1-t) \to \infty$: Bournez et al.'s "time" is infinite, even though the clock reads less than one second. This illustrates the robustness of arc length as a complexity measure: it is invariant under time reparametrization, so no finite-time speedup via function composition can reduce the true cost. Despite the apparent speedup, the arc-length cost for r bits of precision is $\Theta(r)$ – the explosion hides computational work in the diverging variable, not in genuine progress toward the answer.

A dual scenario shows that arc length is also unreasonable as a resource on the convergent side.

▶ **Example 1.1** (BFK protocol for 1/3)**.** The Bournez–Fraigniaud–Koegler large-population protocol for 1/3 [3] runs over three states $Q = \{1, 2, 3\}$ with cyclic transition rules

$$i\ j \ \to\ (i{+}1)\ (j{+}1) \pmod 3 \qquad \text{for every } (i,j) \in Q^2.$$

The mean-field ODE on the 3-simplex $\{(x_1, x_2, x_3) : x_i \geq 0,\ x_1 + x_2 + x_3 = 1\}$ is

$$\dot{x}_i \ =\ 2\,(x_{i-1} - x_i), \qquad i \in \{1,2,3\} \pmod 3,$$

with unique stable equilibrium $(1/3, 1/3, 1/3)$. Starting from the corner $(1, 0, 0)$, the trajectory spirals toward this equilibrium with envelope decaying as e^{-3t} (Figure 1). The arc length is uniformly bounded ($L(t) \leq \int_0^\infty \|y'(s)\|\, ds < \infty$), so by the BGP count the cost is finite. But the limit is approached only as $t \to \infty$, and the rate of approach – which is the natural and substantive complexity question for such systems – is not visible in the arc length.

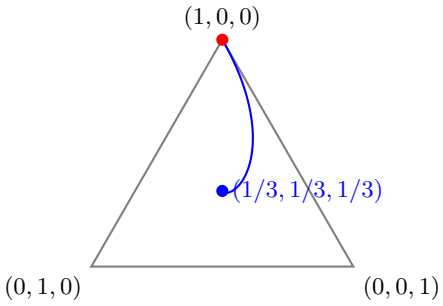


Figure 1 Trajectory of the BFK protocol for 1/3 from $(1, 0, 0)$ to the equilibrium $(1/3, 1/3, 1/3)$ on the 2-simplex. The curve has finite total arc length but reaches the equilibrium only as $t \to \infty$.

When all state variables are bounded, the speed $\|y'(t)\| = \|p(y(t))\|$ is bounded on the compact state space, so $L(t) \leq v_{\max}\, t$: the trajectory length cannot grow faster than t. We adopt the time variable t itself as the complexity resource, and define the corresponding complexity theory on this basis.

But not every system of interest has bounded variables. The resolution is *bounded surrogate compilation* (Section 3): any polynomial ODE system can be compiled into a bounded one that computes the same output, with at most polynomial overhead in dimension.

The key idea is to slow the internal clock in proportion to the growth of unbounded variables, replacing diverging quantities with bounded surrogates in $[0, 1]$. Boundedness therefore does not restrict the class of computable objects – it reinterprets all analog computations in a setting where physical time is a faithful complexity resource.

1.3 Complexity in the bounded setting

Once physical time is faithful, we define time complexity in the style of computable analysis [14, 20]: the *time modulus* $\mu(r)$ is the physical time needed to achieve r bits of precision,[1] and its growth rate classifies the computation. This is finer than the Anderson–Joshi speed [1], which captures exponential convergence rates but does not distinguish sub-exponential profiles, and it is the natural analog of the framework in which CRN-computable numbers [13] were originally studied.

1.4 Results

We prove the following results.

1. **Bounded surrogate compilation** (Section 3). Any unbounded PIVP can be compiled into a bounded one preserving all computed limits, using bounded surrogate variables $U_{n,m} = f^m/(1 + f^n) \in [0, 1]$. Unbounded quantities – including the original time variable t and diverging state variables – are eliminated from the compiled system via what we call the *pass-through technique*: they influence the dynamics but are never represented as state variables (Remark 6.5).
2. **Bounded analog complexity for limit-readout computation** (Section 4). The complexity developed here is for computing real numbers under the limit-readout convention – a bounded PIVP computes α if a designated output variable converges to α at a quantified rate – not for encoding Turing-machine decision problems via GPAC or CRN. Bounded concentrations are strictly stronger than bounded derivatives, but the latter reduces to the former via compilation (Remark 4.2). The main result is that any (possibly unbounded) PIVP whose trajectory length and physical time are both polynomial in the BGP sense [4] compiles to a bounded PIVP with polynomial time modulus (Theorem 4.5). We do not claim a converse: BGP arc length is uninformative on convergent systems (Example 1.1).
3. **Constructible complexity classes** (Section 5). We construct bounded PIVPs realizing prescribed time complexities for the task of computing 1: polynomial of any degree n, $\Theta(r \log r)$ via the Lambert W function, and arbitrarily high classes via iterated-logarithm towers.
4. **Closure under exponentiation** (Section 6). If $\alpha > 0$ and β are bounded-GPAC computable, so is α^β, with time modulus $\mu_{\alpha^\beta}(r) = \max(\mu_\alpha(r), \mu_\beta(r)) + O(1)$. The output variable $z(t) = e^{R(t)}$ is strictly positive and does not require dual-railing; by the selective dual-rail technique of [11], only the intermediate variables need dual-rail encoding.
5. **CRN complexity preservation** (Section 7). The full GPAC-to-CRN pipeline – bounded surrogate compilation, dual-rail encoding, and readout – preserves time complexity class. The key step is the readout subtraction module, which acts as a low-pass filter: it is transparent to all sub-exponential inputs and thus preserves every constructible class of Section 5.

[1] Our formal modulus uses $|x - \alpha| < e^{-r}$, so r is in nats; polynomial classifications are invariant under the bits/nats rescaling.

1.5 Related work

The GPAC model originates with Shannon [17], with refinements by Pour-El [16], Lipshitz–Rubel [15], and Graça–Costa [10]. The computability and complexity theory of polynomial ODEs is due to Bournez, Graça, and Pouly [4, 5]. CRN computation has been studied under both stochastic [18] and deterministic [6, 7] semantics. CRN-computable numbers [13] and large-population protocols [12] connect chemical systems to computable analysis, with the notable feature that all species concentrations are bounded. Anderson and Joshi [1] develop modular CRN arithmetic with input-independent convergence rates, providing the key tool for our readout analysis (Section 7). Population protocols [2] provide another bounded setting; their connection to continuous models is established by Kurtz's theorem.

Since many existing frameworks – CRN-computable numbers [13], population protocols [12], and the dual-rail compilation of [8, 11] – already require bounded species, our bounded surrogate compilation can serve as a *front end* that brings unbounded GPAC designs into their scope. In particular, combined with the balancing dilation technique of [12], our bounded compilations can be further translated into population protocols.

2 Preliminaries

We recall the basic definitions of polynomial initial value problems, the GPAC model, and the Bournez–Graça–Pouly complexity framework.

2.1 Polynomial Initial Value Problems

▶ **Definition 2.1** (PIVP). *A* polynomial initial value problem *(PIVP) is a system*

$$y'(t) = p(y(t)), \qquad y(0) = y_0 \in \mathbb{Q}^d,$$

where $p : \mathbb{R}^d \to \mathbb{R}^d$ *is a vector of polynomials with rational coefficients. We write* $y(t) \in \mathbb{R}^d$ *for the (unique, maximal) solution.*

A PIVP is *bounded* if there exists $M > 0$ such that $\|y(t)\| \le M$ for all $t \ge 0$. Equivalently, the trajectory remains in a compact set.

▶ **Proposition 2.2** (Bounded concentrations imply bounded derivatives). *If* $y' = p(y)$ *with* $\|y(t)\| \le M$ *for all* $t \ge 0$*, then* $\|y'(t)\| \le C(p, M)$*, where* C *depends only on* p *and* M*.*

Proof. p is continuous on the compact set $\|y\| \le M$. ◀

2.2 The GPAC Model

Shannon's General-Purpose Analog Computer (GPAC) generates functions by composing elementary units: constant multipliers, adders, integrators, and multipliers. It is well known that the class of functions generated by a GPAC coincides with the solutions of PIVPs [17]. Conversely, any PIVP can be realized by a GPAC circuit. We therefore use "GPAC" and "PIVP" interchangeably throughout.

2.3 Computability and Complexity via Trajectory Length

Bournez, Graça, and Pouly [4] define analog computation using a convergence semantics: a PIVP computes a real number α if a designated output variable $x(t) \to \alpha$ as $t \to \infty$. They measure complexity by the *trajectory length*

$$L(T) = \int_0^T \|y'(t)\|\, dt,$$

and show that the class of functions computable by PIVPs of polynomial trajectory length coincides with the polynomial-time computable functions of computable analysis.

This framework is elegant but permits unbounded state variables. Our goal is to enforce boundedness while preserving the complexity characterization.

2.4 Bounded-Time Computability

▶ **Definition 2.3** (Bounded-time computability). *Let $x(t)$ be a designated output variable of a bounded PIVP. The system* computes *a real number α if there exists a* time modulus *$\mu : \mathbb{N} \to [0, \infty)$ such that*

$$|x(t) - \alpha| < e^{-r} \qquad \text{whenever } t > \mu(r).$$

The time complexity *of the computation is the asymptotic growth of $\mu(r)$.*

This aligns with computable analysis [14, 20]: the precision parameter r is in nats (one nat equals $1/\log 2 \approx 1.44$ bits), and the polynomial classification we work with is invariant under this constant rescaling. Throughout, log denotes the natural logarithm.

3 Bounded Compilation Framework

We present *bounded surrogate compilation*, a method that eliminates unbounded variables from polynomial ODE systems while preserving all computed limits.

The key idea is a time reparametrization. Composing the solution with a new time $t = s(\tau)$ produces the reparametrized variable $\bar{f}(\tau) := f(s(\tau))$ and introduces a factor $s'(\tau)$ into every ODE via the chain rule. If f^n is the highest power of the unbounded variable f in the system, setting $s'(\tau) = 1/(1 + \bar{f}(\tau)^n)$ converts every monomial $f^m \cdot s'(\tau)$ into the bounded quantity $U_{n,m} = \bar{f}^m/(1 + \bar{f}^n) \in [0, 1]$ – a *surrogate*. The surrogates satisfy a closed polynomial ODE: differentiating $U_{n,m}$ by the quotient rule introduces $d\bar{f}/d\tau$, but this quantity itself decomposes as $\sum_k h_k U_{n,k}$ – the rate of change of the unbounded variable is captured by its own surrogates. The system thus closes over itself, with no reference to f.

▶ **Construction 3.1** (Bounded surrogate compilation). *Let $Z = (z_1, \ldots, z_{d-1}, f)$ satisfy $\dot{Z} = p(Z)$ with $p \in \mathbb{Q}[Z]^d$, where f is the unbounded variable to eliminate. Let n be the highest degree of f in any component of p. The construction proceeds in four steps.*

Step 1: Time change. *Define $\bar{f}(\tau) = f(s(\tau))$ and $\bar{z}_j(\tau) = z_j(s(\tau))$, where $t = s(\tau)$ satisfies $s'(\tau) = 1/(1 + \bar{f}(\tau)^n)$. Define the bounded surrogates*

$$U_{n,m}(\tau) = \frac{\bar{f}(\tau)^m}{1 + \bar{f}(\tau)^n} \in [0, 1], \quad m = 0, \ldots, n, \qquad U_{n,0} + U_{n,n} = 1.$$

Step 2: Rewrite $\bar{z}_j$ equations. *Order each p_j by powers of $\bar{f}$: $p_j(\bar{Z}) = \sum_{m=0}^{n} g_{j,m}(\bar{z})\, \bar{f}^m$, where $g_{j,m}$ is a polynomial in variables other than f. The chain rule and $s'(\tau) = 1/(1 + \bar{f}^n)$ give:*

$$\frac{d\bar{z}_j}{d\tau} = \sum_{m=0}^{n} g_{j,m}(\bar{z}) \cdot \underbrace{\frac{\bar{f}^m}{1 + \bar{f}^n}}_{U_{n,m}}.$$

Step 3: Derive surrogates' ODE. *By the quotient rule,*

$$\frac{dU_{n,m}}{d\tau} = \big(m\,U_{n,m-1} - n\,U_{n,n-1}\,U_{n,m}\big)\cdot\frac{d\bar{f}}{d\tau}.$$

To express $d\bar{f}/d\tau$*: order* $p_d(\bar{Z}) = \sum_m h_m(\bar{z})\,\bar{f}^m$ *and apply* $s'(\tau) = U_{n,0}$*:*

$$\frac{d\bar{f}}{d\tau} = p_d(\bar{Z})\cdot s'(\tau) = \sum_m h_m(\bar{z})\cdot\underbrace{\frac{\bar{f}^m}{1+\bar{f}^n}}_{U_{n,m}} = \sum_m h_m\cdot U_{n,m}.$$

Substituting:

$$\boxed{\frac{dU_{n,m}}{d\tau} = \big(m\,U_{n,m-1} - n\,U_{n,n-1}\,U_{n,m}\big)\cdot\sum_k h_k(\bar{z})\,U_{n,k}.} \tag{1}$$

The compiled system is a polynomial ODE in the variables $(\bar{z}_1,\ldots,\bar{z}_{d-1},U_{n,0},\ldots,U_{n,n})$. The unbounded variable $\bar{f}$ no longer appears: it has been fully absorbed by the bounded surrogates. The original time $t = s(\tau)$ is also absent from the compiled system. If any of $\bar{z}_1,\ldots,\bar{z}_{d-1}$ remain unbounded, apply the construction again to eliminate the next unbounded variable, and repeat until all variables are bounded.

▶ **Proposition 3.2** (Limit preservation)**.** *Let* $T^* \le \infty$ *be the maximal existence time of the original system. If* $\lim_{t\to T^{*-}} z_j(t) = L$*, then* $\lim_{\tau\to\infty}\bar{z}_j(\tau) = L$*.*

Proof. Since $s'(\tau) = 1/(1+\bar{f}^n) > 0$, the function $t = s(\tau)$ is strictly increasing. If $T^* = \infty$ (no blowup), then $s(\tau) \le \tau \to \infty$. If $T^* < \infty$ (finite-time blowup), then $\bar{f}(\tau) \to \infty$ as $s(\tau) \to T^{*-}$, forcing $s'(\tau) \to 0$; thus $s(\tau) \to T^*$ but never reaches it, and the approach takes $\tau \to \infty$. In both cases, $\bar{z}_j(\tau) = z_j(s(\tau)) \to L$. ◀

Applying Construction 3.1 iteratively to each unbounded variable and invoking Proposition 3.2 at each step, we obtain:

▶ **Theorem 3.3** (Bounded GPACs compute the same real numbers)**.** *Every real number computable by a (possibly unbounded) GPAC is also computable by a bounded GPAC.*

In other words, unboundedness does not enlarge the class of GPAC-computable real numbers.

3.1 Worked example: finite-time explosion

Consider $x' = (1-x)(1+v)^2$, $v' = (1+v)^2$, $x(0) = v(0) = 0$. The auxiliary variable $v(t) = 1/(1-t) - 1$ explodes at $t = 1$, while the output $x(t) = 1 - e^{1-1/(1-t)}$ converges to 1 as $t \to 1^-$. Since $1 + v(t) = 1/(1-t)$, the arc length is $L(T) \ge T/(1-T) \to \infty$; the cost for r bits of precision is $\Theta(r)$ – no better than $z' = 1 - z$. Applying Construction 3.1 with $n = 2$, the surrogates $U_{0,2} = 1/(1+\bar{v}^2)$ and $U_{1,2} = \bar{v}/(1+\bar{v}^2)$ yield the bounded system:

$$\boxed{\begin{aligned}\bar{x}' &= (1-\bar{x})(1+2U_{1,2}),\\ U_{1,2}' &= (U_{0,2}+U_{1,2})^2(2U_{0,2}-1),\\ U_{0,2}' &= -2(U_{0,2}+U_{1,2})^2U_{1,2},\end{aligned}} \qquad \bar{x}(0) = 0,\ U_{0,2}(0) = 1,\ U_{1,2}(0) = 0, \tag{2}$$

with all variables in $[0,1]$ for all $\tau \ge 0$. The variable v has been eliminated; the finite-time blowup is a disguise, not a speedup.

3.2 Application: computing π via pole compilation

Bounded surrogate compilation reveals a duality: *finite-time singularities in unbounded systems become infinite-time limits in bounded systems.* We illustrate by computing $\pi/2$ from the pole of $\tan(t)$.

The tangent function satisfies $x' = 1 + x^2$, $x(0) = 0$, with pole at $t = \pi/2$. Applying bounded surrogate compilation with $\phi(x) = 1/(1 + x^2)$ and introducing surrogates $U_0 = 1/(1 + \tau^2)$, $U_1 = \tau/(1 + \tau^2)$:

$$\boxed{U_0' = -2\,U_0\,U_1, \quad U_1' = U_0(2U_0 - 1).} \qquad U_0(0) = 1,\ U_1(0) = 0. \tag{3}$$

All variables bounded. The physical time $t(\tau) = \arctan(\tau) \to \pi/2$: the pole location becomes the computed output, readable from the surrogates (Figure 2).

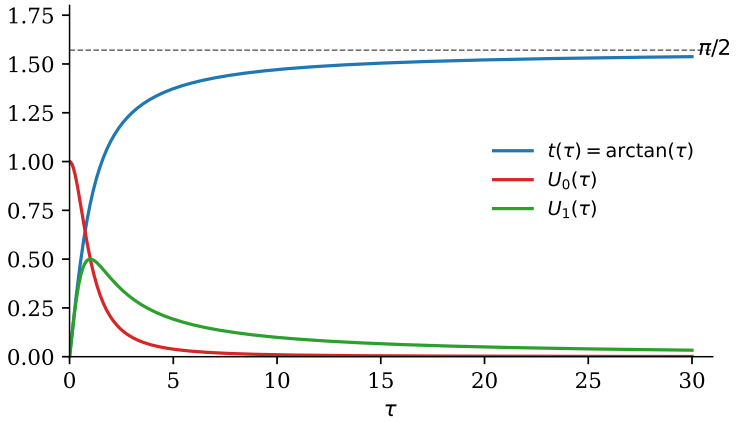


Figure 2 Time evolution of the compiled bounded system (3) in the new time variable τ. The physical time $t(\tau) = \arctan(\tau)$ converges to $\pi/2$ as $\tau \to \infty$ (dashed line). The surrogates U_0, U_1 remain in $[0, 1]$ throughout: the pole of $\tan(t)$ at $t = \pi/2$ has been compactified into an infinite-time limit in τ.

The error $\pi/2 - t(\tau) \sim 1/\tau$ gives time modulus $\mu(r) = \Theta(e^r)$. This is not optimal for computing π – specialized constructions achieve $\mu(r) = \Theta(r)$ [13] – but the method is fully generic, as the following theorem shows.

▶ **Theorem 3.4** (Poles of GPAC-implementable functions are GPAC-computable). *Let $y' = p(y)$, $y(0) = y_0 \in \mathbb{Q}^d$ be a PIVP with finite maximal existence time $T^* < \infty$. Then T^* is computable by a bounded PIVP.*

Proof. See Appendix A.1. ◀

4 Bounded Analog Complexity

The bounded surrogate compilation of Section 3 shows that every GPAC or CRN computation can be transformed into one whose state variables remain bounded, without loss of generality. Physical time t is then the only diverging resource, and we can use it to measure computational cost directly. With the time modulus of Definition 2.3 in hand, we now develop a complexity theory for bounded analog systems.

The complexity studied here is for analog computation of *real numbers* under the limit-readout convention of Definition 2.3; we do not encode Turing-machine computation in bounded CRNs, nor do we discuss decision problems under such setting in this work.

Terminological convention. A standard but important inversion relative to dynamical-systems terminology: *exponential convergence* ($O(e^{-kt})$) corresponds to *linear time* ($\mu(r) = \Theta(r)$), while *polynomial convergence* ($O(t^{-k})$) corresponds to *exponential time* ($\mu(r) = \Theta(e^{r/k})$). Throughout, "polynomial time" refers to the computable-analysis notion: $\mu(r) = O(r^k)$.

4.1 One-Sided Trajectory–Time Bound on Compact Domains

▶ **Lemma 4.1** (One-sided bound). *Let $y'(t) = p(y(t))$ be a PIVP whose trajectory remains in a compact domain D. Then the polynomial vector field p satisfies $\|p(y)\| \le v_{\max}$ on D for some $v_{\max} < \infty$, hence*

$$L(t) \;=\; \int_0^t \|y'(\tau)\|\, d\tau \;\le\; v_{\max}\, t.$$

Proof. By compactness, $\|p\|$ attains a maximum $v_{\max}$ on D. The bound on $L(t)$ follows directly. ◀

The converse one-sided inequality $L(t) \ge v_{\min}\, t$ requires a non-degeneracy hypothesis $\|p(y(t))\| \ge v_{\min} > 0$ which is incompatible with convergence to a fixed point and therefore fails for the very systems described by Definition 2.3. We do not assume it; only the upper bound of Lemma 4.1 is used below.

▶ Remark 4.2 (Bounded concentrations vs. bounded derivatives). Bounded concentrations imply bounded derivatives (compactness of the state space), but not conversely: $x' = 1$ has bounded derivative yet $x(t) \to \infty$. A system with bounded derivatives admits the one-sided bound $L(t) \le Mt$ but is not automatically bounded; Construction 3.1 compiles it into a bounded-concentration system without changing the computed object.

4.2 Iterative Compilation Preserves Polynomial Arc Length

The proof of bounded compilation is by induction on the number of unbounded variables eliminated. The tracked resource through each compilation step is the trajectory arc length L. The base case is the BGP polynomial trajectory length of the original system. The inductive step (Lemma 4.4) shows that one compilation iteration sends a polynomial-arc-length system to another polynomial-arc-length system. When the iteration terminates with all variables bounded, Lemma 4.1 converts polynomial arc length to polynomial physical time directly. The converse direction is immediate: any bounded PIVP whose physical time is polynomial has polynomial arc length by Lemma 4.1.

▶ **Lemma 4.3** (Polynomial state bound from polynomial arc length). *Let $y'(t) = p(y(t))$ be a PIVP for which there is a polynomial q such that $L(t_r) \le q(r)$ for every precision r, where t_r is the physical time at which the output reaches precision r (Definition 2.3). Then for every component y_i and every $s \in [0, t_r]$,*

$$|y_i(s)| \;\le\; |y_i(0)| + q(r).$$

Proof. $|y_i(s) - y_i(0)| \le \int_0^s |y_i'(u)|\, du \le \int_0^s \|p(y(u))\|\, du = L(s) \le L(t_r) \le q(r)$. ◀

▶ **Lemma 4.4** (One compilation step preserves polynomial arc length). *Let S_{pre} be a PIVP whose trajectory length and physical time both satisfy $L_{\mathrm{pre}}(t_r), t_r \le q(r)$ for some polynomial q and every precision r. Apply bounded surrogate compilation (Construction 3.1) to eliminate one unbounded variable f of maximal degree n in p. Then the post-compilation system S_{post}, with new time variable τ, satisfies*

$$\tau_r \;\le\; Q_t(r) \qquad \text{and} \qquad L_{\mathrm{post}}(\tau_r) \;\le\; Q_L(r)$$

for some polynomials Q_t, Q_L determined by q and n.

Proof. By Lemma 4.3, $|f(s)| \le |f(0)| + q(r) =: M_f(r) = \mathrm{poly}(r)$ for $s \in [0, t_r]$.

For the new time variable: from $d\tau/dt = 1 + \bar{f}^n$ (Construction 3.1, Step 1) and $|f(u)| \le M_f(r)$,

$$\tau_r \;=\; \int_0^{t_r} \big(1 + f(u)^n\big)\, du \;\le\; t_r\big(1 + M_f(r)^n\big) \;\le\; q(r)\big(1 + M_f(r)^n\big) \;=\; \mathrm{poly}(r) \;=:\; Q_t(r).$$

For the new arc length: the post-compilation vector field p_{post} acts on the state $(\bar{z}_1, \dots, \bar{z}_{d-1}, U_{n,0}, \dots, U_{n,n})$, where each $U_{n,m} \in [0,1]$ (Construction 3.1, Step 1). By Steps 2 and 3 of the construction, $d\bar{z}_j/d\tau = \sum_m g_{j,m}(\bar{z}) U_{n,m}$ and $dU_{n,m}/d\tau$ is polynomial in $(\bar{z}, U_{n,\cdot})$. Since the $U_{n,m}$ lie in $[0,1]$ and $|\bar{z}_j(s)| \le |\bar{z}_j(0)| + q(r) = M_z(r) = \mathrm{poly}(r)$ on the precision-r prefix by Lemma 4.3, every component of p_{post} is bounded by a polynomial in $M_z(r) + M_f(r)$, hence $\|p_{\mathrm{post}}\| \le M_p(r) = \mathrm{poly}(r)$. Therefore

$$L_{\mathrm{post}}(\tau_r) \;=\; \int_0^{\tau_r} \|p_{\mathrm{post}}\|\, d\sigma \;\le\; M_p(r)\,\tau_r \;\le\; \mathrm{poly}(r) \;=:\; Q_L(r). \qquad \blacktriangleleft$$

▶ **Theorem 4.5** (Iterative bounded compilation). *Let $y'(t) = p(y(t))$ be a (possibly unbounded) PIVP whose trajectory length and physical time both satisfy $L(t_r), t_r \le q(r)$ for some polynomial q and every precision r (BGP normalization). Iterated bounded surrogate compilation (Construction 3.1) produces a* fully bounded *PIVP computing the same real number (or function), whose trajectory length and physical time at precision r are both polynomial in r. The dimensional overhead is polynomial in the maximal monomial degree of p.*

Proof. By induction on the number of remaining unbounded variables.

Base case. $S_0 := y$ has $L_0(t_r), t_r \le q(r)$ by hypothesis.

Inductive step. Suppose after i compilation iterations the system S_i satisfies $L_i(\tau_r^{(i)}), \tau_r^{(i)} \le Q^{(i)}(r)$ for some polynomial $Q^{(i)}$, and still contains at least one unbounded variable f_{i+1}. Apply Lemma 4.4 to S_i with input bound polynomial $Q^{(i)}$, eliminating f_{i+1}. The output system S_{i+1} satisfies $L_{i+1}, \tau_r^{(i+1)} \le Q^{(i+1)}(r)$ for the polynomial $Q^{(i+1)} := \max(Q_t, Q_L)$.

Termination. The original system has finitely many unbounded variables $f_1, \dots, f_k$, each one fully absorbed by its compilation step (Construction 3.1). The induction therefore terminates after k iterations with S_k fully bounded, having $L_k, \tau_r^{(k)} \le Q^{(k)}(r) = \mathrm{poly}(r)$. Limit preservation throughout by Proposition 3.2. Each step adds at most $n+1$ surrogate variables, so the total dimensional overhead is at most $k(n+1)$. ◀

▶ **Theorem 4.6** (From BGP poly-length to bounded poly-time). *If a real number (or function on a compact domain) is computable by a (possibly unbounded) PIVP in polynomial trajectory length* and *polynomial physical time in the sense of Bournez–Graça–Pouly [4], then it is computable by a bounded PIVP in polynomial physical time (Definition 2.3).*

Proof. Theorem 4.5. ◀

▶ Remark 4.7 (The converse direction is not pursued). We do not claim a converse. For a PIVP whose vector field has exponentially decaying norm along the trajectory, the arc length $L(t)$ stays finite as $t \to \infty$, while the time required to reach precision e^{-r} can range from linear to exponential in r. Example 1.1 is the canonical instance: BGP arc length is bounded by a constant; the time-to-precision modulus is $\Theta(r)$. Two such systems whose times-to-precision differ by exponentials in r can share the same finite arc length, so the BGP resource does not distinguish between them. The BGP framework records the cost of explosive (state-diverging) computation faithfully, but on convergent systems it loses the rate-of-approach information that constitutes the natural complexity question.

5 Constructible Bounded Time Complexity

Analyzing the convergence rate of a general ODE is notoriously difficult. Rather than analyzing arbitrary systems, we take the opposite approach: we start from known functions with well-understood asymptotics and *construct* bounded polynomial ODE systems that realize prescribed time complexity classes. All constructions compute the same value: the real number 1.

Throughout this section, $r \in \mathbb{N}$ denotes the *precision parameter*: the system is required to produce r *nats* of accuracy at the output, i.e., $|x_1(t) - 1| < e^{-r}$ (Definition 2.3; equivalently, r nats are $r/\log 2$ bits in the standard computable-analysis sense). The *time modulus* $\mu(r)$ is the time the bounded PIVP needs to reach this precision.

The design recipe is as follows. Given a target time modulus $\mu(r)$, we:

1. **Choose a precision clock.** Pick a monotone function $g(t) \to \infty$ with $g(\mu(r)) \geq r$, intuitively the number of nats of precision delivered by time t. The clock g is a design abstraction, not a state variable of the PIVP: it is generally unbounded and therefore not directly realizable in a bounded system.
2. **Apply the Möbius surrogate.** Substitute the fractional-linear (Möbius) compactification $u(t) = g(t)/(1+g(t)) \in [0, 1)$, so $u \to 1$ as $g \to \infty$. The rational inverse $g = u/(1-u)$ turns polynomial dependence on g into a rational function of u with denominators that are powers of $(1-u)$; residual denominators are cleared by the time reparametrization $s'(t) = (1-u)^{-k}$ familiar from Section 3. The surrogate u, together with any auxiliary bounded states, is what the PIVP actually instantiates: g never appears as a state, in the sense of Remark 6.5. (Throughout this section the direct Möbius closure happens to be already polynomial, so no reparametrization is needed.)
3. **Read out via exponential approach.** Drive the output by $x_1'(t) = (1 - x_1(t)) \cdot h(\text{bounded states})$ so that $1 - x_1(t) = e^{-g(t)}$ (or a constant multiple). Then $|x_1(t) - 1| < e^{-r}$ exactly when $g(t) > r$, i.e., when $t > \mu(r)$.

5.1 Linear and Polynomial Time

The simplest instance is $x_1' = 1 - x_1$, $x_1(0) = 0$, giving $x_1(t) = 1 - e^{-t}$ and $\mu(r) = r$. This is the degenerate case of the recipe with $g(t) = t$. Since $g'(t) \equiv 1$ is already constant, no Möbius surrogate is needed: step 2 of the recipe is trivial and the bounded PIVP closes on the single output variable x_1. We generalize this to a tunable polynomial family in Section 5.2 below.

5.2 Tunable Polynomial Time: $\mu(r) = \Theta(r^n)$

▶ **Theorem 5.1** (Polynomial-time constructibility). *For every $n \in \mathbb{N}$, there exists a bounded PIVP that computes the constant 1 with time modulus $\mu_n(r) = \Theta(r^n)$.*

Proof. We instantiate the recipe with precision clock $g(t) = (1+t)^{1/n} - 1$ and Möbius surrogate

$$u(t) = \frac{g(t)}{1+g(t)} = 1 - (1+t)^{-1/n} \in [0,1).$$

The bounded system on the two state variables $(u, x) \in [0,1]^2$ is

$$\begin{aligned} u'(t) &= \tfrac{1}{n}\,(1-u(t))^{\,n+1}, \\ x'(t) &= (1-x(t)) \cdot \tfrac{1}{n}\,(1-u(t))^{\,n-1}, \end{aligned} \qquad u(0) = x(0) = 0, \tag{4}$$

a polynomial PIVP of degree $n+1$. Closure follows from $1-u = (1+t)^{-1/n}$, hence $1+t = (1-u)^{-n}$, which gives

$$u' = \tfrac{1}{n}(1+t)^{-1/n-1} = \tfrac{1}{n}(1-u)(1+t)^{-1} = \tfrac{1}{n}(1-u)\cdot(1-u)^{\,n} = \tfrac{1}{n}(1-u)^{\,n+1}.$$

For the output, $g'(t) = \tfrac{1}{n}(1+t)^{(1-n)/n} = \tfrac{1}{n}(1-u)^{\,n-1}$, so $x' = (1-x)\,g' = (1-x)\tfrac{1}{n}(1-u)^{\,n-1}$. Integration yields $1 - x(t) = \exp(-g(t)) = \exp(-(1+t)^{1/n}+1)$. Therefore $|x(t)-1| < e^{-r}$ iff $(1+t)^{1/n} - 1 > r$, i.e., $t > (r+1)^n - 1 = \Theta(r^n)$. ◀

▶ Remark 5.2 (Smooth representative of the asymptotic class). The clock $g(t) = (1+t)^{1/n} - 1$ is the smooth representative of the asymptotic class $t^{1/n}$: the shift $t \mapsto 1+t$ moves the branch point of $t^{1/n}$ off the operating region $t \geq 0$. Using $g_0(t) = t^{1/n}$ directly yields the closure $u'(t) = \tfrac{1}{n}(1-u)^{\,n+1}/u^{\,n-1}$, which is singular at the initial state $u = 0$ for $n \geq 2$; the "+1" shift is the minimal regularization that restores polynomial closure with finite initial slope $u'(0) = 1/n$. The same regularization appears implicitly in the Lambert-W construction of Section 5.2 and the iterated-logarithm tower of Appendix A.2.

Thus every polynomial degree is constructible: linear for $n = 1$, quadratic for $n = 2$, cubic for $n = 3$, etc.

$\Theta(r \log r)$ Time via Lambert W

▶ **Proposition 5.3** ($\Theta(r \log r)$ constructibility). *There exists a bounded PIVP that computes the constant 1 with time modulus $\mu(r) = \Theta(r \log r)$.*

Step 1 (precision clock and target readout). Let W denote the Lambert function satisfying $W(t)\,e^{W(t)} = t$. We take the precision clock

$$g(t) \;=\; e^{W(t)} - 1,$$

which is monotone with $g(0) = 0$ and $g(t) \to \infty$, and design the readout to satisfy

$$1 - x_1(t) \;=\; e^{-g(t)} \;=\; e^{\,1 - e^{W(t)}}.$$

The precision condition $|1 - x_1(t)| < e^{-r}$ is then equivalent to $g(t) \geq r$, i.e., $e^{W(t)} \geq r+1$, equivalently $W(t) \geq \log(r+1)$. The Lambert identity $t = We^W$ gives $t \geq (r+1)\log(r+1) = \Theta(r \log r)$, so the readout has the desired modulus $\mu(r) = \Theta(r \log r)$.

Step 2 (Möbius surrogates). The Lambert evolution

$$W'(t) \;=\; \frac{1}{(1+W)\,e^{W}}$$

involves *two* unbounded quantities, $1+W$ and e^W. A bounded polynomial encoding therefore needs to carry information about both. We introduce two Möbius surrogates:

$$v(t) := \frac{W(t)}{1+W(t)} \in [0,1) \qquad \text{(Möbius surrogate of } W\text{)},$$
$$u(t) := \frac{g(t)}{1+g(t)} \;=\; 1-e^{-W(t)} \in [0,1) \qquad \text{(Möbius surrogate of } g\text{)}.$$

With $u(0) = v(0) = 0$ and $u, v \to 1$, The rational inversions $1+W = 1/(1-v)$ and $e^W = 1/(1-u)$ together polynomialize the Lambert dynamics: from $W' = 1/((1+W)e^W) = (1-u)(1-v)$,

$$u' = e^{-W}W' = (1-u)^2(1-v), \qquad v' = \frac{W'}{(1+W)^2} = (1-v)^3(1-u).$$

Neither surrogate alone would close in polynomial form: v' requires e^W (read off u), and u' requires W' which in turn requires $1+W$ (read off v). The pair (u,v) is the minimal bounded encoding of the Lambert pair (W, e^W).

Step 3 (readout dynamics). The readout x_1 is the designated output variable of the bounded PIVP. Following the recipe, it satisfies the exponential-approach equation $x_1'(t) = (1-x_1(t))\,g'(t)$, which with $x_1(0) = 0$ integrates to $1-x_1(t) = e^{-g(t)}$ exactly, as targeted in Step 1. The chain rule gives

$$g'(t) = e^{W(t)}\,W'(t) = \frac{e^W}{(1+W)\,e^W} = \frac{1}{1+W} = 1-v,$$

so the readout equation is $x_1' = (1-x_1)(1-v)$. The surrogates u, v are auxiliary states that exist only to polynomialize g'; they are not themselves the output.

Bounded polynomial system. Collecting steps 1–3 yields the polynomial PIVP on $(u, v, x_1) \in [0,1]^3$:

$$\boxed{\begin{aligned} u' &= (1-u)^2(1-v), \\ v' &= (1-v)^3(1-u), \\ x_1' &= (1-x_1)(1-v), \end{aligned}} \qquad u(0) = v(0) = x_1(0) = 0. \tag{5}$$

Proof of Proposition 5.3. The bounded polynomial system (5) arises from the recipe instantiation in Steps 1–3 above. Integrating $x_1'/(1-x_1) = 1-v = 1/(1+W)$ from $x_1(0) = 0$ yields $1-x_1(t) = e^{-g(t)} = e^{1-e^{W(t)}}$ exactly, so $|1-x_1(t)| < e^{-r}$ holds iff $g(t) \geq r$, i.e., iff $t \geq (r+1)\log(r+1) = \Theta(r \log r)$. ◀

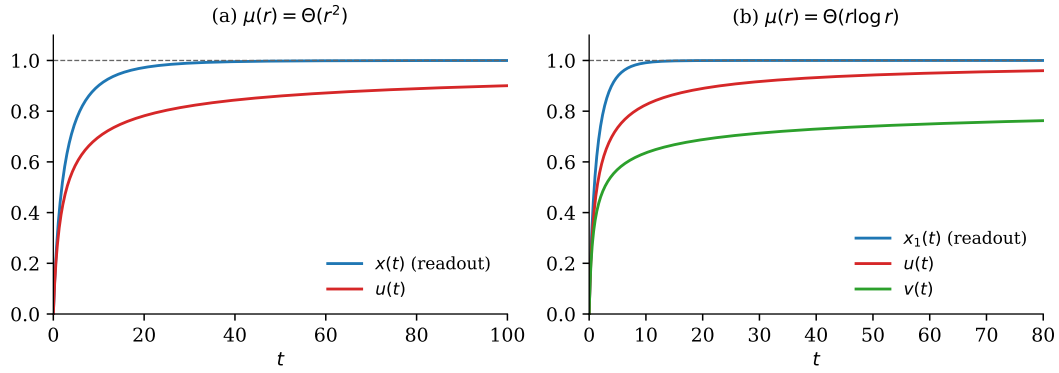


Figure 3 Time evolution of the constructible families. (a) Polynomial $\Theta(r^2)$: $u(t) = 1-(1+t)^{-1/2}$, $x(t) = 1 - e^{-(\sqrt{1+t}-1)}$. (b) Lambert-$W$ $\Theta(r \log r)$: $u(t) = 1 - e^{-W(t)}$, $v(t) = W(t)/(1 + W(t))$, $x_1(t) = 1 - e \cdot e^{-e^{W(t)}}$. All states remain in $[0, 1)$; the output convergence rate determines the time modulus.

An iterated-logarithm tower construction extends the linear–$\Theta(r \log r)$–$\Theta(r^n)$ hierarchy to every level of the elementary exponential hierarchy: a depth-k construction yields time modulus $\mu(r) = \exp^{(k+1)}(r + O(1))$. The construction, analysis, and tabulation are recorded in Appendix A.2.

▶ Remark 5.4. All constructions in this section – the linear, polynomial, Lambert W, and iterated-logarithm families – share the common design pattern of the recipe in Section 5: choose a precision clock with known asymptotics, encode it via a Möbius surrogate, and read off the time modulus. Every construction computes the same trivial value 1; the complexity is intrinsic to the dynamics, not to the target.

6 Closure and Richness of Bounded Computation

The preceding sections established the foundations of bounded analog complexity: the compilation framework the core complexity definitions, and constructible time complexity classes. We now demonstrate the computational richness of bounded GPACs by showing closure under exponentiation.

6.1 Closure Under Exponentiation

Beyond compilation, we demonstrate the computational richness of bounded GPACs by showing closure under exponentiation.

▶ **Theorem 6.1** (Bounded GPAC exponentiation). *If $\alpha > 0$ and $\beta \in \mathbb{R}$ are computable by bounded GPACs, then α^β is computable by a bounded GPAC.*

The key identity is $\alpha^\beta = e^{\beta \ln \alpha}$. Direct computation of $\ln(x(t))$ is problematic when $x(0) = 0$. The solution is to introduce a shifted variable $x_1(t) \to \alpha - 1$ with $x_1(0) = 0$, so that $\ln(1 + x_1)$ starts at $\ln(1) = 0$, avoiding the singularity.

▶ **Construction 6.2** (Computing α^β). *Given upstream GPACs computing $x(t) \to \alpha$ and $y(t) \to \beta$, define:*

$$\boxed{\begin{aligned} x_1' &= (x-1) - x_1, \\ u' &= (1-v)\, x_1', \\ v' &= (1-v)^2\, x_1', \\ z' &= z\,\big(y'\,u + y\,(1-v)\,x_1'\big), \end{aligned}} \tag{6}$$

with $x_1(0) = u(0) = v(0) = 0$ and $z(0) = 1$. Here $u = \ln(1+x_1)$ and $v = x_1/(1+x_1)$.

Proof of Theorem 6.1. See Appendix A.4. ◀

▶ **Corollary 6.3** (CRN exponentiation). *The construction of Construction 6.2 can be implemented as a CRN. For $\alpha < 1$, the intermediate variables x_1, u, v may become negative and require dual-rail encoding. However, every term in the ODE for z contains z as a factor ($z' = z \cdot (\ldots)$), so no negative contribution to z' can arise independently of z itself. By the selective dual-railing criterion of [11], z is not infected and does not require dual-rail encoding: it is represented directly as a single species concentration.*

▶ **Theorem 6.4** (Complexity of α^β). *Let $\alpha > 0$ and $\beta \in \mathbb{R}$ be computable by bounded GPACs with time moduli $\mu_\alpha(r)$ and $\mu_\beta(r)$. Then α^β is computable by a bounded GPAC with time modulus*

$$\mu_{\alpha^\beta}(r) = \max\big(\mu_\alpha(r+C),\ \mu_\beta(r+C)\big) + O(1),$$

where $C = C(\alpha, \beta)$ is a constant depending only on α and β. In particular, exponentiation preserves the time complexity class: the cost is determined by the slower input.

Proof. See Appendix A.5. ◀

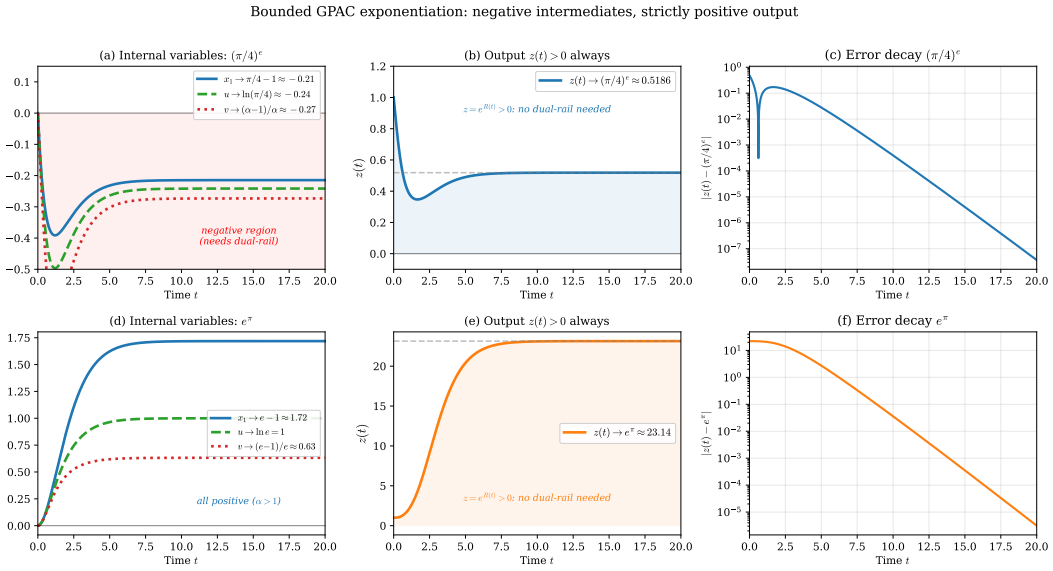
Exponentiation preserves the complexity class: the time modulus of α^β is determined by the *slower* input. For example, both π and e are computable in linear time ($\mu(r) = \Theta(r)$) by known GPAC constructions [13]; therefore $(\pi/4)^e$ and e^π are also computable in linear time.

▶ Remark 6.5 (The pass-through technique). The quantity $R(t) = y(t) \cdot \ln(1 + x_1(t))$ appears in the exponent of z but is never tracked as an explicit variable – only its derivative R' enters the ODE for z. This is a general design technique for bounded GPACs: intermediate quantities that are transcendental or unbounded can *pass through* the system as long as their derivatives close over the existing bounded state variables. Bounded surrogate compilation (Construction 3.1) is another instance: both the physical time t and the unbounded variable f pass through the compiled system without appearing in the final ODE.

7 Bounded CRN Time Complexity

The GPAC-to-CRN pipeline has two steps: *dual-rail encoding* converts a signed polynomial ODE into mass-action kinetics, and a *readout module* extracts the algebraic difference $X_1^+ - X_1^-$ as a single species concentration. Dual-rail encoding, following [8] (see also [13]), is exact: $X_1^+ - X_1^-$ satisfies the original ODE, and all dual-rail species remain bounded.

The question is whether the readout step – subtraction – also preserves time complexity. Subtraction is itself a dynamical system with its own convergence rate; if it is too slow, it could degrade the complexity class inherited from the GPAC. We show that subtraction acts as a low-pass filter: it is transparent to all sub-exponential inputs, and thus preserves the time complexity class for all constructible classes of Section 5.



Interactive notebook: `https://colab.research.google.com/github/zinan-huang/notebooks/blob/main/bounded_gpac_exponentiation.ipynb`

Figure 4 Computing $(\pi/4)^e$ (top) and e^π (bottom) via the bounded GPAC construction (Construction 6.2), using the ODE constructions of [13] for $\pi/4$, e, and π. **(a)** When $\alpha < 1$, the intermediate variables x_1, u, v take negative values (red shading), requiring selective dual-rail encoding for CRN implementation [11]. **(b, e)** The output $z(t) = e^{R(t)}$ is strictly positive for all $t \geq 0$, regardless of the sign of the intermediates: no dual-railing is needed for the output species. **(d)** When $\alpha > 1$ (as in e^π), all intermediate variables remain positive. **(c, f)** Both computations converge exponentially, consistent with linear time modulus $\mu(r) = \Theta(r)$.

7.1 Readout subtraction modules

After dual-rail encoding, $x_1(t) = X_1^+(t) - X_1^-(t)$ satisfies the original ODE but is not a single species concentration. A readout species Z must compute $Z(t) \to \alpha$ via mass-action reactions. Anderson and Joshi [1] provide the *absolute-difference module* ($A + Z \to A + 2Z$, $B + Z \to B$, $2Z \to Z$):

$$\dot{Z} = x_1(t)\, Z - Z^2, \qquad Z^* = \alpha. \tag{7}$$

An alternative is *double inversion* [13], computing $1/x_1$ then $1/(1/x_1) = x_1$ via:

$$\dot{Y} = 1 - x_1(t)\, Y, \qquad \dot{Z} = 1 - Y\, Z, \tag{8}$$

with $Y^* = 1/\alpha$ and $Z^* = \alpha$.

Anderson and Joshi [1] define the *speed* $\rho := -\limsup_{t\to\infty} \frac{\ln|x(t)-x^*|}{t}$ and show each arithmetic module has speed ≥ 1. Positive speed ($\rho > 0$) is equivalent to exponential convergence, i.e., linear time $\mu(r) = \Theta(r)$ – the real-time class studied in [13]. Our time modulus (Definition 2.3) refines and generalizes this notion: it distinguishes the sub-exponential profiles ($\rho = 0$) that the speed framework treats uniformly, with $\rho = \lim_{r\to\infty} r/\mu(r)$ when the limit exists.

7.2 Low-pass filter analysis

Write $x_1(t) = \alpha + \epsilon(t)$ and $Z(t) = \alpha + \delta(t)$. Linearizing (7) around $Z^* = \alpha > 0$:

$$\dot{\delta} + \alpha\,\delta = \alpha\,\epsilon(t), \tag{9}$$

with solution

$$\delta(t) = e^{-\alpha t}\,\delta(0) + \alpha \int_0^t e^{-\alpha(t-s)}\,\epsilon(s)\,\mathrm{d}s. \tag{10}$$

This is a first-order low-pass filter with cutoff rate α. The convolution kernel $e^{-\alpha(t-s)}$ exponentially forgets the past: if $\epsilon(s)$ varies slowly relative to the forgetting rate, it can be pulled out of the integral, giving $\delta(t) \approx \epsilon(t)$ – the module is transparent. Conversely, if $\epsilon(s) = Ce^{-\beta s}$ with $\beta > \alpha$, the integral evaluates to $\frac{\alpha C}{\beta-\alpha}e^{-\alpha t}$: the fast input component vanishes and the output decays at the module's own rate α. In short: *a fast module cannot speed up a slow input; it can only slow down a fast one.*

For the double-inversion subtraction of [13], the cascade has effective speed $\min(\alpha, 1/\alpha)$; the analysis is analogous.

7.3 Convergence preservation

The convolution (10) determines how the readout error $\delta(t)$ relates to the input error $\epsilon(t)$. The key dichotomy:

▶ **Lemma 7.1** (Input-limited regime). *If $\epsilon(t)$ decays slower than $e^{-\alpha t}$ (i.e., $e^{\alpha t}\,|\epsilon(t)| \to \infty$), then*

$$\delta(t) \sim \epsilon(t) \qquad (t \to \infty).$$

The readout error tracks the input error asymptotically.

Proof. The homogeneous part $e^{-\alpha t}\,\delta(0) = o(\epsilon(t))$. For the convolution, define $I(t) = \alpha \int_0^t e^{\alpha s}\,\epsilon(s)\,\mathrm{d}s$. Since $e^{\alpha t}\,|\epsilon(t)| \to \infty$, both $|I(t)| \to \infty$ and $e^{\alpha t} \to \infty$. By L'Hôpital's rule:

$$\lim_{t\to\infty} \frac{I(t)}{e^{\alpha t}} = \lim_{t\to\infty} \frac{\alpha\,e^{\alpha t}\,\epsilon(t)}{\alpha\,e^{\alpha t}} = \lim_{t\to\infty} \epsilon(t) = 0.$$

More precisely, $e^{-\alpha t}\,I(t) - \epsilon(t) \to 0$, giving $\delta(t) \sim \epsilon(t)$. ◀

▶ **Lemma 7.2** (Module-limited regime). *If $\epsilon(t) = C_\epsilon\,e^{-\beta t}$ with $\beta > \alpha$, then*

$$\delta(t) = \frac{\alpha\,C_\epsilon}{\beta - \alpha}\,e^{-\alpha t} + O(e^{-\beta t}).$$

The readout is throttled to the module's intrinsic speed α.

Proof. Direct evaluation of the convolution integral with $\epsilon(s) = C_\epsilon\,e^{-\beta s}$. ◀

▶ **Theorem 7.3** (CRN readout preserves time complexity). *Let a bounded GPAC compute $\alpha > 0$ with time modulus $\mu(r)$. Let Z be the readout species implemented via either the Anderson–Joshi absolute-difference module or the double-inversion method. Then:*

1. *If $\mu(r) = \omega(r/\alpha)$ (input convergence slower than the module's intrinsic rate $e^{-\alpha t}$), then $\mu_Z(r) = \mu(r) + O(1)$. The time complexity class is **preserved exactly**.*
2. *If $\mu(r) = O(r/\alpha)$ (input convergence at least as fast as $e^{-\alpha t}$), then $\mu_Z(r) = r/\alpha + O(1)$. The readout is **throttled** to linear time with rate α.*

In either case, the readout speed in the Anderson–Joshi sense is $\rho_Z = \min(\rho_{\text{in}}, \alpha)$*, consistent with their Composition Theorem.*

Proof. See Appendix A.3. ◀

▶ **Corollary 7.4** (Constructible classes are CRN-realizable)**.** *All constructible bounded time complexity classes of Section 5 (computing* $\alpha = 1$*) are realizable by CRNs with the same time modulus:*

Class	***GPAC modulus*** $\mu(r)$	***CRN readout modulus*** $\mu_Z(r)$
Linear	$\Theta(r)$	$\Theta(r)$
Lambert W	$\Theta(r \log r)$	$\Theta(r \log r)$
Polynomial degree n	$\Theta(r^n)$	$\Theta(r^n)$

Proof. With $\alpha = 1$, the module speed is 1 (exponential decay e^{-t}). The Lambert W and polynomial constructions converge sub-exponentially, hence slower than e^{-t}. By Theorem 7.3(1), the time modulus is preserved. The linear case ($\mu(r) = \Theta(r)$) is the boundary: $\epsilon(t) = e^{-t}$ matches the module's intrinsic rate. L'Hôpital gives a polynomial prefactor ($\delta(t) \sim t\, e^{-t}$), changing the modulus by at most $O(\log r)$, which is absorbed into $\Theta(r)$. ◀

8 Conclusion

We have introduced bounded analog complexity theory, in which all state variables remain in compact intervals and the time variable t is the sole diverging resource. Bounded surrogate compilation transforms unbounded PIVPs into bounded ones preserving all computed limits, and a polynomial change of time scale sends BGP polynomial-trajectory-length-and-time computation to bounded polynomial-t computation. Constructive examples realize fine-grained time moduli from $\Theta(r)$ through $\Theta(r \log r)$ and $\Theta(r^n)$ to every level of the elementary exponential hierarchy. Closure under exponentiation demonstrates the computational richness of the bounded setting, with the output variable remaining strictly positive and thus directly realizable as a CRN species without dual-railing. The full GPAC-to-CRN pipeline preserves time complexity, via a low-pass filter analysis of the readout module.

The complexity developed here concerns computation of real numbers via the limit readout, and our compilation absorbs unbounded variables only insofar as they affect the asymptotic limit. Existing polynomial-ODE simulations of Turing machines in the BGP framework [9, 4, 8] encode the tape positionally in state variables that grow as 10^n in tape length n. Whether Turing-complete simulation can be carried out in entirely bounded polynomial state, in the rational-coefficient PIVP and mass-action CRN form of the present paper, is left open.

References

1 David F. Anderson and Badal Joshi. Chemical mass-action systems as analog computers: implementing arithmetic computations at specified speed. *arXiv preprint*, 2024. arXiv:2404.04396.

2 Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, 2007. doi:10.1007/S00446-007-0040-2.

3 Olivier Bournez, Pierre Fraigniaud, and Xavier Koegler. Computing with large populations using interactions. In *Mathematical Foundations of Computer Science (MFCS)*, volume 7464 of *Lecture Notes in Computer Science*, pages 234–246. Springer, 2012. doi:10.1007/978-3-642-32589-2_23.

4 Olivier Bournez, Daniel S. Graça, and Amaury Pouly. Polynomial time corresponds to solutions of polynomial ordinary differential equations of polynomial length. *Journal of the ACM*, 64(6):38:1–38:76, 2017. doi:10.1145/3127496.

5 Olivier Bournez and Amaury Pouly. A universal ordinary differential equation. In *Proc. 45th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 107 of *LIPIcs*, pages 116:1–116:14, 2018. doi:10.4230/LIPIcs.ICALP.2017.116.

6 Ho-Lin Chen, David Doty, and David Soloveichik. Deterministic function computation with chemical reaction networks. *Natural Computing*, 13(4):517–534, 2014. doi:10.1007/S11047-013-9393-6.

7 Ho-Lin Chen, David Doty, and David Soloveichik. Rate-independent computation in continuous chemical reaction networks. *Journal of the ACM*, 70(3):1–61, 2023. arXiv:2107.13681.

8 François Fages, Guillaume Le Guludec, Olivier Bournez, and Amaury Pouly. Strong Turing completeness of continuous chemical reaction networks and compilation of mixed analog-digital programs. In *Computational Methods in Systems Biology (CMSB 2017)*, volume 10545 of *Lecture Notes in Computer Science*, pages 108–127. Springer, 2017. doi:10.1007/978-3-319-67471-1_7.

9 Daniel S. Graça, Manuel L. Campagnolo, and Jorge Buescu. Computability with polynomial differential equations. *Advances in Applied Mathematics*, 40(3):330–349, 2008. doi:10.1016/J.AAM.2007.02.003.

10 Daniel S. Graça and José Félix Costa. Analog computers and recursive functions over the reals. *Journal of Complexity*, 19(5):644–664, 2003. doi:10.1016/S0885-064X(03)00034-7.

11 Nicholas Haisler, Xiang Huang, Andrei N. Migunov, Khalid Mohammed, and Garrett Provence. A selective dual-railing technique for general-purpose analog computers. In *Unconventional Computation and Natural Computation (UCNC 2025)*, volume 16364 of *Lecture Notes in Computer Science*. Springer, 2025. doi:10.1007/978-3-032-15641-9_26.

12 Xiang Huang and Rachel N. Huls. Computing real numbers with large-population protocols having a continuum of equilibria. In *28th International Conference on DNA Computing and Molecular Programming (DNA 28)*, volume 238 of *LIPIcs*, pages 6:1–6:17. Schloss Dagstuhl, 2022. doi:10.4230/LIPIcs.DNA.28.7.

13 Xiang Huang, Titus H. Klinge, and James I. Lathrop. Real-time equivalence of chemical reaction networks and analog computers. In *25th International Conference on DNA Computing and Molecular Programming (DNA 25)*, volume 11648 of *Lecture Notes in Computer Science*, pages 37–53. Springer, 2019. doi:10.1007/978-3-030-26807-7_3.

14 Ker-I Ko. *Complexity Theory of Real Functions*. Birkhäuser, 1991.

15 Leonard Lipshitz and Lee A. Rubel. A differentially algebraic replacement theorem, and analog computability. *Proceedings of the American Mathematical Society*, 99(2):367–372, 1987.

16 Marian B. Pour-El. Abstract computability and its relation to the general purpose analog computer. *Transactions of the American Mathematical Society*, 199:1–28, 1974.

17 Claude E. Shannon. Mathematical theory of the differential analyzer. *Journal of Mathematics and Physics*, 20(1–4):337–354, 1941.

18 David Soloveichik, Matthew Cook, Erik Winfree, and Jehoshua Bruck. Computation with finite stochastic chemical reaction networks. *Natural Computing*, 7(4):615–633, 2008. doi:10.1007/S11047-008-9067-Y.

19 Alan M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42(1):230–265, 1936. doi:10.1112/PLMS/S2-42.1.230.

20 Klaus Weihrauch. *Computable analysis: an introduction*. Springer Science & Business Media, 2000.

A Deferred proofs

A.1 Proof of Theorem 3.4

Proof. Apply bounded surrogate compilation (Construction 3.1) to eliminate all unbounded variables, obtaining a bounded system in surrogate variables with time parameter τ. The physical time satisfies $s'(\tau) = U_{n,0}(\tau)$, $s(0) = 0$, where $U_{n,0} = 1/(1 + \bar{f}^n) \in (0, 1]$.

Adjoin s as an additional variable to the compiled system. This adds one equation $s' = U_{n,0}$ to a bounded polynomial ODE, and $s(\tau) < T^*$ for all τ, so the augmented system remains bounded.

We claim $s(\tau) \to T^*$ as $\tau \to \infty$. Since s is increasing and bounded above by T^*, the limit $L = \lim_{\tau\to\infty} s(\tau)$ exists. If $L < T^*$, then $\bar{f}(\tau) = f(s(\tau)) \to f(L)$, which is finite since the solution exists on $[0, T^*)$. But then $s'(\tau) = 1/(1 + \bar{f}(\tau)^n) \to 1/(1 + f(L)^n) > 0$, contradicting the convergence of $s(\tau)$ to a finite limit. Hence $L = T^*$.

Therefore $s(\tau) \to T^*$ in a bounded PIVP, and T^* is GPAC-computable. ◀

A.2 Iterated-logarithm tower constructions

This appendix records the iterated-logarithm tower construction deferred from Section 5.

▶ **Construction A.1** (Iterated-logarithm tower). *For $k \geq 0$, define inductively*

$$w_k' = (1 - w_k)^2 \prod_{j=0}^{k-1} (1 - w_j), \qquad w_k(0) = 0, \tag{11}$$

with the convention that the empty product ($k = 0$) equals 1. The readout variable is

$$x_1' = (1 - x_1) \prod_{j=0}^{k} (1 - w_j), \qquad x_1(0) = 0. \tag{12}$$

▶ **Proposition A.2** (Tower solutions). *Define $L_0(t) = t$ and $L_k(t) = \log(1 + L_{k-1}(t))$ for $k \geq 1$. Then for each $k \geq 0$:*

$$1 - w_k(t) = \frac{1}{1 + L_k(t)}.$$

In particular, as $t \to \infty$, $L_k(t) = \log^{(k)}(t) + O(1)$ (the k-fold iterated logarithm).

Proof. By induction on k. Base case $k = 0$: $w_0' = (1 - w_0)^2$ has solution $w_0(t) = t/(1 + t)$, giving $1 - w_0 = 1/(1 + t) = 1/(1 + L_0(t))$. Inductive step: let $u_k = 1 - w_k$. Then $u_k' = -u_k^2 \prod_{j=0}^{k-1}(1 + L_j(t))^{-1}$. The chain rule gives $L_k'(t) = L_{k-1}'(t)/(1 + L_{k-1}(t)) = \prod_{j=0}^{k-1}(1 + L_j(t))^{-1}$, so $du_k/u_k^2 = -L_k'(t)\,dt$. Integrating: $1/u_k(t) = 1 + L_k(t)$. ◀

▶ **Theorem A.3** (Exponential-tower modulus at depth k). *For each $k \geq 0$, the depth-k tower construction computes $x_1(t) \to 1$ with $1 - x_1(t) = \exp(-L_{k+1}(t))$, and time modulus $\mu(r) = \exp^{(k+1)}(r + O(1))$, the $(k+1)$-fold iterated exponential.*

Proof. By Proposition A.2, $\prod_{j=0}^{k}(1 - w_j) = L_{k+1}'(t)$, hence $x_1'/(1 - x_1) = L_{k+1}'(t)$ and $-\log(1 - x_1) = L_{k+1}(t)$. Precision e^{-r} then requires $L_{k+1}(t) \geq r$, i.e., $t \geq \exp^{(k+1)}(r - O(1))$. ◀

Depth k	Convergence profile $1 - x_1(t)$	Time modulus $\mu(r)$
-1 (no tower)	e^{-t}	$\Theta(r)$
0	$e^{-\log(1+t)} = (1+t)^{-1}$	$\Theta(e^r)$
1	$e^{-\log\log(1+t)}$	$\Theta(e^{e^r})$
k	$e^{-\log^{(k+1)}(t)}$	$\Theta(\exp^{(k+1)}(r))$

A.3 Proof of Theorem 7.3

Proof. Case (1) follows from Lemma 7.1: since $\delta(t) \sim \epsilon(t)$, the time to reach precision e^{-r} is determined by ϵ, not by the module.

Case (2) follows from Lemma 7.2: the dominant term is $C\,e^{-\alpha t}$, giving $\mu_Z(r) = r/\alpha + O(1)$.

For the double-inversion method, the same argument applies with α replaced by $\min(\alpha, 1/\alpha)$. ◀

A.4 Proof of Theorem 6.1

Proof.

ODE derivation. Set $x_1 = x - 1$ (shifted to have $x_1(0) = 0$); the tracking ODE $x_1' = (x-1) - x_1$ is a low-pass filter that drives $x_1 \to \alpha - 1$. Define $u = \ln(1 + x_1)$. By the chain rule, $u' = x_1'/(1+x_1)$. Setting $v = x_1/(1+x_1)$ (equivalently, $v = 1 - 1/(1+x_1)$), we obtain $u' = (1-v)\,x_1'$, since $1/(1+x_1) = 1 - v$. Differentiating v by the quotient rule gives $v' = x_1'/(1+x_1)^2 = (1-v)^2\,x_1'$. Finally, $z = e^{y\cdot u}$, so $z' = z\,(y'\,u + y\,u') = z\,(y'\,u + y\,(1-v)\,x_1')$. All right-hand sides are polynomial in the state variables (x_1, u, v, z) and the upstream variables (x, y).

Convergence. As $t \to \infty$: $x_1 \to \alpha - 1$, $u \to \ln(\alpha)$, $v \to (\alpha-1)/\alpha$, and the exponent $y \cdot u \to \beta \ln \alpha$, giving $z \to e^{\beta \ln \alpha} = \alpha^\beta$.

Well-definedness. The trajectory satisfies $1 + x_1(t) > 0$ for all $t \geq 0$: since $x_1(0) = 0$ and x_1 converges to $\alpha - 1 > -1$ (for $\alpha > 0$), the denominator in u and v never vanishes. ◀

A.5 Proof of Theorem 6.4

Proof. Since the construction operates entirely within the GPAC model (no dual-railing), the analysis is standard computable-analysis error propagation [14, 20] through four steps. Let $\varepsilon_\alpha(t) = |x(t) - \alpha|$ and $\varepsilon_\beta(t) = |y(t) - \beta|$.

Step 1: x_1 tracking error. Define $e_1(t) = x_1(t) - (\alpha - 1)$. From (6): $e_1' = (x(t) - \alpha) - e_1$, $e_1(0) = -(\alpha - 1)$. This is a first-order linear ODE with rate 1, giving

$$|e_1(t)| \leq e^{-t}\,|\alpha - 1| + \int_0^t e^{-(t-s)}\,\varepsilon_\alpha(s)\,ds.$$

This is the low-pass filter of Section 7.2 with cutoff 1. By Lemma 7.1, $|e_1(t)|$ inherits the decay profile of $\varepsilon_\alpha(t)$.

Step 2: u and v errors. By the mean value theorem, $|u(t) - \ln\alpha| \leq |e_1(t)|/\alpha_{\min}$ and $|v(t) - (\alpha-1)/\alpha| \leq |e_1(t)|/\alpha_{\min}^2$, where $\alpha_{\min} = \min_t(1 + x_1(t)) > 0$. These are proportional to $|e_1|$ with constant factors.

Step 3: Exponent error. $R(t) = y(t)\,u(t)$ targets $R^* = \beta \ln \alpha$. By the triangle inequality,

$$|R(t) - R^*| \le |\ln \alpha|\, \varepsilon_\beta(t) + \frac{|\beta|}{\alpha_{\min}}\, |e_1(t)| + o(1).$$

Step 4: Output error. Since $z = e^R$ and $\alpha^\beta = e^{R^*}$, the mean value theorem gives $|z - \alpha^\beta| \le e^{\max(R,R^*)} |R - R^*|$. For large t, there exists $C = C(\alpha, \beta) > 0$ such that $|z - \alpha^\beta| < e^{-r}$ whenever both $\varepsilon_\beta(t) < e^{-(r+C)}$ and $|e_1(t)| < e^{-(r+C)}$. Hence $\mu_{\alpha^\beta}(r) = \max\big(\mu_\alpha(r + C), \mu_\beta(r + C)\big) + O(1)$. ◀