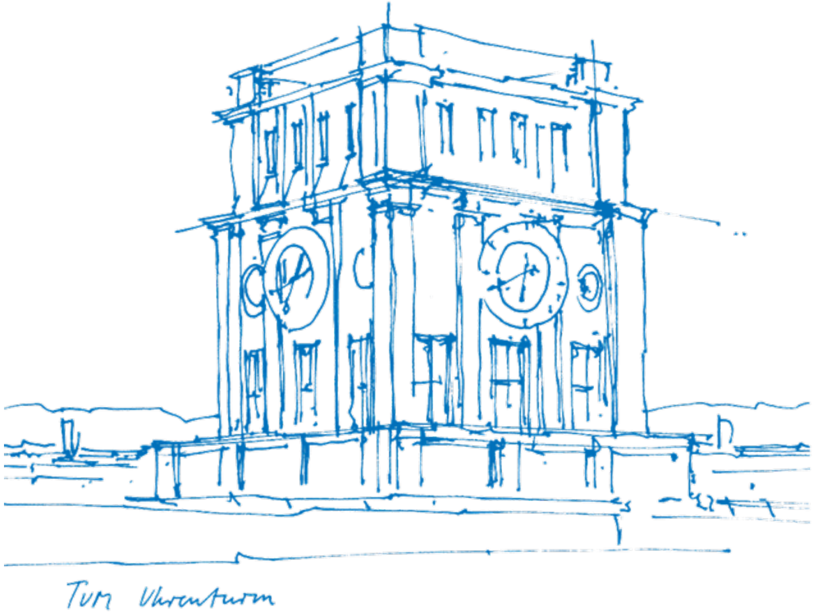
Complexity Blowup if Continuous-Time LTI Systems are Implemented on Digital Hardware

Holger Boche Volker Pohl

Technical University of Munich
Department of Electrical and Computer Engineering
Chair of Theoretical Information Technology

60th IEEE Conference on Decision and Control (CDC)
Dec. 17, 2021



Outline of the Talk

What is the computational complexity of simulating causal, time-invariant linear systems on digital computers?

Having low-complexity input signal. What is the complexity for calculating the output signal?

Outline

1. Linear Time-Invariant (LTI) Systems
2. Computability and Complexity
3. Examples of Complexity Blowup
4. Summary

Linear Time-Invariant (LTI) Systems

First-Order Linear Time-Invariant (LTI) Systems

- We consider causal, linear, time-invariant systems of first order with input $x(t)$ and output $y(t)$.
- Input-Output relation is described by linear differential equation with constant coefficients

$$\begin{aligned} & y'(t) + \alpha_0\, y(t) = \beta_1 x'(t) + \beta_0 x(t)\,, \quad t > 0 \\ & \text{with initial condition} \quad y(0) = y_0 \quad \text{and} \quad x(0) = x_0\,, \end{aligned} \tag{1}$$

  with coefficients $\alpha_0, \beta_0, \beta_1, x_0, y_0 \in \mathbb{R}$, which are assumed to be polynomial-time computable.
- The unique solution of (1) is given for $t > 0$ by the closed form expression

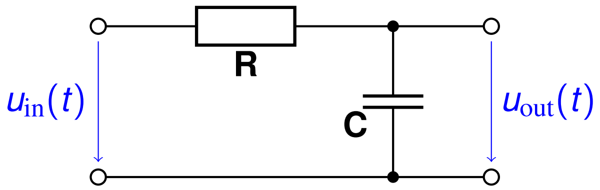
$$y(t) = (\mathrm{S}x)(t) = y_0\, \mathrm{e}^{-\alpha_0 t} + a\left[x(t) - x_0\, \mathrm{e}^{-\alpha_0 t}\right] + b \int_0^t x(\tau)\, \mathrm{e}^{-\alpha_0(t-\tau)}\, \mathrm{d}\tau \tag{2}$$

  with the constants $a := \beta_1$ and $b := \beta_0 - \beta_1 \alpha_0$.

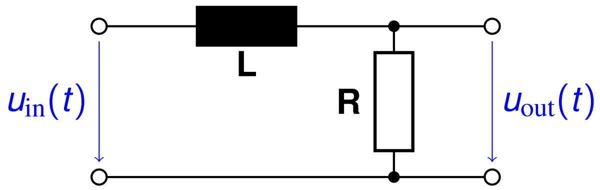
- Goal: Simulate the output signal $y(t)$, for $t \in [0,1]$ on a *digital computer*, for feasible (continuously differentiable, computable) signals $x(t)$.
- Question: Assume low-complexity input $x(t)$. What is the complexity for computing $y(t)$?

Examples of First Order LTI systems

Two examples for a general LTI system $\mathbb{S}$ with input $x(t) = u_{\mathrm{in}}(t)$ and output $y(t) = u_{\mathrm{out}}(t)$.



RC low pass with resistor R, capacitor C, and cutoff frequency $\omega_0 = (RC)^{-1}$



LR low pass with inductor L, resistor R, and cutoff frequency $\omega_0 = R/L$

Computability

Computable Rational Numbers

Definition: A sequence $\{r_n\}_{n\in\mathbb{N}} \subset \mathbb{Q}$ of rational numbers is said to be computable if there exist recursive functions $a, b, s : \mathbb{N} \to \mathbb{N}$ with $b(n) \neq 0$ and such that

$$r_n = (-1)^{s(n)} \frac{a(n)}{b(n)}\,, \qquad n \in \mathbb{N}\,.$$

A recursive function $a : \mathbb{N} \to \mathbb{N}$ is a mapping that is build form elementary computable functions and recursion and can be calculated on a *Turing machine*.

Turing machine

- can simulate any given algorithm and therewith provide a simple but very powerful model of computation.
- is a theoretical model describing the fundamental limits of any realizable digital computer.
- Most powerful programming languages are called Turing-complete (such as C, C++, Java, etc.).

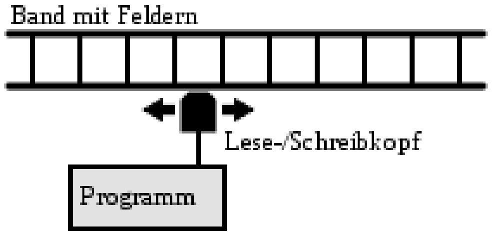


Figure taken from *Wikipedia*

A. M. Turing, "On computable numbers, with an application to the Entscheidungsproblem," *Proc. London Math. Soc.*, vol. s2-42, no. 1, 1937.

Computable Real Numbers

- ▷ Any real number $t \in \mathbb{R}$ is the limit of a sequence of rational numbers.
- ▷ For $t \in \mathbb{R}$ to be computable, the convergence has to be effective.

Definition (Computable number): A number $t \in \mathbb{R}$ is said to be *computable* if there exist a recursive function $\gamma : \mathbb{N} \to \mathbb{Q}$ such that

$$|t - \gamma(n)| \leq 2^{-n}, \qquad \text{for all } n \in \mathbb{N}.$$

In this case, we say that γ *binary converges* to t.

$\Rightarrow$ $x \in \mathbb{R}$ is computable if a Turing machine can approximate it with exponentially vanishing error.

- $\mathbb{R}_c$ stand for the set of all *computable real numbers*.
- Note that the set of computable numbers $\mathbb{R}_c \subsetneq \mathbb{R}$ is only countable.

Computable Functions

- We consider function-oracle Turing machines: Ordinary Turing machine TM with an additional *function-oracle* γ
- The function oracle is able to calculate the function value γ in a single operation.
- We neglect the computational complexity for determine t in calculating $x(t)$.

Definition: A function $x : [a,b] \to \mathbb{R}$ is said to be *computable* on the interval $[a,b] \subseteq \mathbb{R}$ if there exists a function-oracle Turing machine TM so that for each $t \in [a,b]$ and each γ that binary converges to t, the function $\widetilde{x}(n) = \mathrm{TM}_\gamma(n)$ computed by TM with oracle γ binary converges to $x(t)$, i.e. if

$$\left| x(t) - \mathrm{TM}_\gamma(n) \right| \leq 2^{-n}, \qquad \text{for all } n \in \mathbb{N}\,.$$

Remark:
If $x : [a,b] \to \mathbb{R}$ is a computable function on $[a,b]$ then $x \in \mathscr{C}([a,b])$, i.e. x is a continuous function on $[a,b]$.

Computational Complexity

Definition: Let $x : [a,b] \to \mathbb{R}$ be a computable function. We say that the complexity of x is bounded by a function $q : \mathbb{N} \to \mathbb{N}$ if there exists a function–oracle Turing machine TM_γ, which computes x so that for all γ that binary converge to $t \in [a,b]$ and for all $n \in \mathbb{N}$, $\mathrm{TM}_\gamma(n)$ satisfies

$$\left| x(t) - \mathrm{TM}_\gamma(n) \right| < 2^{-n}$$

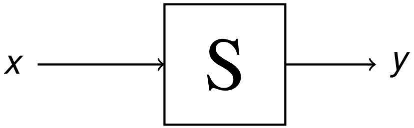
after a computation time of at most $q(n)$.
The function $x : [a,b] \to \mathbb{R}$ is said to be polynomial-time computable if its complexity is bounded by a polynomial q.

Complexity Classes

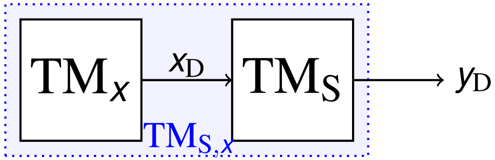
- *FP* The class of functions, which can be computed by an function-oracle Turing machine in polynomial time.
- $\#P$ The class of functions that enumerate the number of accepting computations of polynomial-time function-oracle Turing machines.

Assumption: $FP \subsetneq \#P$ – Note that $FP = \#P$ would imply $P = NP$.

Simulation of LTI Systems on Digital Computers



A general LTI system S with input signal x and output signal y.



A Turing machine implementation $\mathrm{TM}_{\mathrm{S},x}$ for simulating S for input x consisting of a signal generator TM_x and a Turing machine TM_{S} which simulates the behavior of S.

- ▷ Digital computers can calculate exactly only with rational numbers
- ▷ Signal generator TM_x prepares the input signal x up to n significant digits

$$|x(t) - x_D(t)| < 2^{-n}$$

- ▷ Assume low complexity input signal, i.e generation of x_D needs time $p_x(n)$ with a certain polynomial p_x.
- ▷ Question: Is the output y_{D} again a low complexity signal?
  Can $y_D(t)$ with $|(\mathrm{S}x)(t) - y_D(t)| < 2^{-m}$ be computed in time $p_y(m)$ with a certain polynomial p_y?

Complexity Blowup

Complexity Blowup

We consider causal continuous-time linear systems S mapping functions on $\mathbb{R}_+$ onto functions on $\mathbb{R}_+$.

$$(\mathrm{S}x)(t) = ax(t) + b\int_0^t x(\tau)\,h(t-\tau)\,\mathrm{d}\tau\,, \quad t > 0\,. \tag{3}$$

Definition: Let $\mathrm{S} : L^\infty([0,1]) \to L^\infty([0,1])$ be an LTI system with input-output relation (3). We say that S shows complexity blowup, if there exists a polynomial-time computable input signal $x_* : [0,1] \to \mathbb{R}$ with the property that the corresponding output signal $y_* = \mathrm{S}x_*$ is not polynomial-time computable.
In this case, we say that S shows complexity blowup for the input signal x_*.

Complexity Blowup – Results

We consider causal continuous-time linear systems S mapping functions on $\mathbb{R}_+$ onto functions on $\mathbb{R}_+$.

$$(\mathrm{S}x)(t) = ax(t) + b\int_0^t x(\tau)\,h(t-\tau)\,\mathrm{d}\tau\,, \quad t > 0\,.$$

Theorem: Assume $FP \neq \#P$ and let S be a first-order LTI system described by (3) with polynomial-time computable coefficients. If $b = \beta_0 - \beta_1\alpha_0 \neq 0$ then S shows complexity blowup, i.e. there exists a polynomial-time computable signal $x_* : [0,1] \to \mathbb{R}$ so that the output $y_* = \mathrm{S}x_*$ is not polynomial-time computable on $[0,1]$.

Corollary: A first-order LTI system (1) with polynomial-time computable coefficients shows no complexity blowup if and only if $b = \beta_0 - \beta_1\alpha_0 = 0$.

- Complete characterization of all first-order systems showing complexity blowup.
- Only trivial cases ($b = 0$) show no complexity blowup
- Extension to higher-order systems and non-causal systems possible
- Proof is based on a result of *Friedman* and *Ko*.

Discussion – Practical Computability

- ▷ Applied science, applications, cryptogrophy $\Rightarrow$ (practically) "computability"versus "non-computable"
- ▷ (practically) "non-computable" if the calculation time, using the best possible algorithm, grows faster than any polynomial in the number of the parameters
- ▷ (practically) "computable" if the calculation time grows at post polynomially in the number of the parameters

Our results: For any first order, causal LTI system S (with $b \neq 0$) there exit input signals x_* which are practically computable but such that the output $y_* = \mathrm{S}x_*$ is practically non-computable.

Further Examples of Complexity Blowup

Calculation of Fourier Series Approximation

- $\mathscr{C}_{2\pi}$: be the set of all computable, continuous, and 2π-periodic functions
- $\mathscr{C}_{2\pi}^{\mathrm{pol}}$: be the set of all $x \in \mathscr{C}_{2\pi}$ which are polynomial-time computable
- For $x \in \mathscr{C}_{2\pi}$, we consider the problem of calculating for $N \in \mathbb{N}$ the partial Fourier series

$$(\mathrm{F}_N x)(t) = \frac{a_0}{2} + \sum_{k=1}^{N} [a_k \cos(kt) + b_k \sin(kt)]$$

with the *Fourier coefficients* of x given by

$$a_k = \frac{1}{\pi}\int_{-\pi}^{\pi} x(\tau)\cos(k\tau)\,\mathrm{d}\tau \qquad \text{and} \qquad b_k = \frac{1}{\pi}\int_{-\pi}^{\pi} x(\tau)\sin(k\tau)\,\mathrm{d}\tau$$

Theorem: If $FP_1 \neq \#P_1$ then there exists an $x_* \in \mathscr{C}_{2\pi}^{\mathrm{pol}}$ such that there is an $N \in \mathbb{N}$ so that $(\mathrm{F}_N x_*)(t)$ is not polynomial-time computable for $t \in [-\pi, \pi)$.

Calculation of the Fourier Transform

- $\mathscr{C}(0,\pi)$: set of all computable continuous functions on $[0,\pi] \subset \mathbb{R}$
- $\mathscr{C}^{\text{pol}}(0,\pi)$: the set of all $x \in \mathscr{C}(0,\pi)$, which are polynomial-time computable
- For $x \in \mathscr{C}(0,\pi)$, we consider the problem of calculating the Fourier transform

$$\widehat{x}(\omega) = (\mathscr{F}x)(\omega) = \int_0^{\pi} x(\tau)\mathrm{e}^{\mathrm{i}\omega\tau}\,\mathrm{d}\tau\,, \qquad \omega \in \mathbb{R}\,.$$

pointwise for some $\omega \in \mathbb{R}$.

Theorem: For any $\omega \in \mathbb{R}$ there exists an $x_* \in \mathscr{C}^{\text{pol}}(0,\pi)$ such that if $FP_1 \neq \#P_1$ then $\widehat{x}_*(\omega)$ is not polynomial-time computable.

Summary

Summary

- ▷ For every causal LTI system $\mathbb{S}$ there exist low complexity input signals x so that the corresponding output $y = \mathbb{S}x$ is high complexity.
- ▷ Other examples of complexity blowup: Calculation of Fourier series approximation and Fourier transform
- ▷ Extension to non-causal and higher-order LTI systems possible
- ▷ Notion of "practical non-computability": foundation of modern cryptography
  - polynomial-time computable $\Rightarrow$ is practically computable
  - not polynomial-time computable $\Rightarrow$ is practically non-computable
- ▷ For any (first-order) LTI system $\mathbb{S}$ (with $b \neq 0$) there exist practically computable input signals x_* so that the corresponding output signal $y_* = \mathbb{S}x_*$ is practically non-computable

H. Boche and V. Pohl, "Complexity Blowup in Simulating Analog Linear Time-Invariant Systems on Digital Computer," *IEEE Trans. Signal Processing*, vol. 69 (Aug. 2021), 5005–5020.