Computability, Complexity and Programming with Ordinary Differential Equations

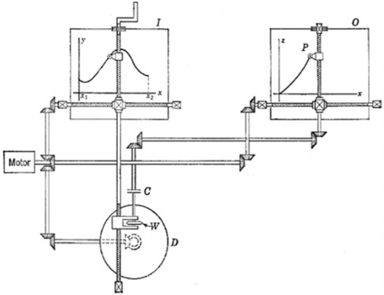
Olivier Bournez

Laboratoire d'Informatique de l'X
CNRS, Ecole Polytechnique
Institut Polytechnique de Paris

Créteil
October 18, 2021

Menu

How to compute an integral?



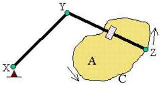
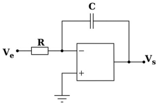
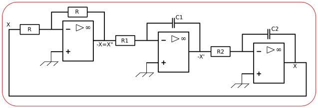
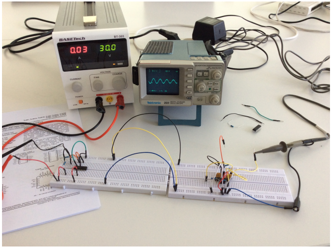


Figure 1. A simple planimeter.







Our actual motivation

- Understand how analog models compare to classical digital models of computation.
  - At **computability** level
  - At **complexity** level.
- Continuous time analog models correspond to various classes of ordinary differential equations.
- Discussing hardness of solving IVP according to various classes of dynamics is basically discussing the **computational power** of various classes of analog models.

Take home message

- **Turing machines $\sim$ polynomial Ordinary Differential Equations**
  I.e.

$$\begin{array}{rcl} \mathbf{y}' & = & \mathbf{p}(t, \mathbf{y}) \\ \mathbf{y}(t_0) & = & \mathbf{y}_0 \end{array}$$

  where $\mathbf{p}$ is a (vector of) polynomials.

  in a very very strong sense.

- Programming with/Solving ODEs is **simple** and **fun**.

Take home message

- **Turing machines $\sim$ polynomial Ordinary Differential Equations**
  I.e.

$$\begin{aligned} \mathbf{y}' &= \mathbf{p}(t, \mathbf{y}) \\ \mathbf{y}(t_0) &= \mathbf{y}_0 \end{aligned}$$

  where **p** is a (vector of) polynomials.

  in a very very strong sense.

- Programming with/Solving ODEs is **simple** and **fun**.

- Analog's world: Many concepts from **computer science** can be defined using polynomial ODEs
  - Computable functions.
  - Polynomial Time Computable Functions
  - *NP*, *PSPACE*, ...?
  - Revisiting computation theory with pODEs ...
  - Bioinformatics (proteins) computations $\geq$ Turing machines $=$ Classical computers.
  - ...

Some basics concepts/remarks

- $f : \mathbb{R}^d \to \mathbb{R}^d$
- f can be continuous, derivable, $\mathcal{C}^\infty$, ..., $\mathcal{C}^\infty$, analytic, generable, ...
- Dynamical system:
  - Discrete time: $\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t)$
  - Continuous time: $\mathbf{x}'(t) = \mathbf{f}(\mathbf{x}(t))$
- Computability $\neq$ Complexity

Some basics concepts/remarks

- $f : \mathbb{R}^d \to \mathbb{R}^d$
- f can be continuous, derivable, $\mathcal{C}^\infty$, ..., $\mathcal{C}^\infty$, analytic, generable, ...
- Dynamical system:
  - Discrete time: $\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t)$
    - AKA: $\frac{\delta \mathbf{x}}{\delta t}(t) = \frac{\mathbf{x}_{t+1} - \mathbf{x}_t}{1} = \overline{\mathbf{f}}(\mathbf{x}_t)$, for $\overline{\mathbf{f}}(\mathbf{x}) = \mathbf{f}(\mathbf{x}) - \mathbf{x}$
  - Continuous time: $\mathbf{x}'(t) = \mathbf{f}(\mathbf{x}(t))$
- Computability $\neq$ Complexity

Menu

Analog Computations: Our actual motivation

How to Compute with Iterations (dODEs)

How to Compute with Ordinary Differential Equations

Some computability results based on these ideas

Conclusion

Sub-menu

Turing Machines

- Let M be some one tape Turing machine, with m states and 10 symbols.
- If

$$\ldots B\ B\ B\ a_{-k}\ a_{-k+1}\ldots\ a_{-1}\ a_0\ a_1\ldots\ a_n\ B\ B\ B\ldots$$

  is the tape content of M, it can be seen as

$$\begin{aligned} y_1 &= 0.a_0a_1\ldots a_n \\ y_2 &= 0.a_{-1}a_{-2}\ldots a_{-k} \end{aligned} \qquad (1)$$

- The configuration of M is then given by three values: its internal state s, y_1 and y_2.

Alternative View of a Turing Machine

$$
\begin{aligned}
y_1 &= a_0 10^{-1} + a_1 10^{-2} + \ldots + a_n 10^{-n-1} \\
y_2 &= a_{-1} 10^{-1} + a_{-2} 10^{-2} + \ldots + a_{-k} 10^{-k}.
\end{aligned} \tag{2}
$$

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t))$$

Turing Machine	
State Space $\{q_0, q_1, \cdots, q_{m-1}\} \times \Sigma^*$	
State $(q_i, a_{-m} \ldots a_{-1}, a_0 \ldots a_n)$	

Alternative View of a Turing Machine

$$
\begin{aligned}
y_1 &= a_0 10^{-1} + a_1 10^{-2} + \ldots + a_n 10^{-n-1} \\
y_2 &= a_{-1} 10^{-1} + a_{-2} 10^{-2} + \ldots + a_{-k} 10^{-k}.
\end{aligned} \tag{2}
$$

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t))$$

Turing Machine	
State Space $\{q_0, q_1, \cdots, q_{m-1}\} \times \Sigma^*$	State Space $[0, m] \times [0, 1]$
State $(q_i, a_{-m} \ldots a_{-1}, a_0 \ldots a_n)$	State $\mathbf{x} = (x = s + y_2, y = y_1)$

Alternative View of a Turing Machine

$$\begin{aligned} y_1 &= a_0 10^{-1} + a_1 10^{-2} + ... + a_n 10^{-n-1} \\ y_2 &= a_{-1} 10^{-1} + a_{-2} 10^{-2} + ... + a_{-k} 10^{-k}. \end{aligned} \quad (2)$$

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t))$$

Turing Machine	
State Space $\{q_0, q_1, \cdots, q_{m-1}\} \times \Sigma^*$	State Space $[0, m] \times [0, 1]$
State $(q_i, a_{-m}...a_{-1}, a_0...a_n)$	State $\mathbf{x} = (x = s + y_2, y = y_1)$
q_1: if 2 is read, then write 4; goto q_2	

Alternative View of a Turing Machine

$$\begin{aligned} y_1 &= a_0 10^{-1} + a_1 10^{-2} + \ldots + a_n 10^{-n-1} \\ y_2 &= a_{-1} 10^{-1} + a_{-2} 10^{-2} + \ldots + a_{-k} 10^{-k}. \end{aligned} \tag{2}$$

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t))$$

Turing Machine	
State Space $\{q_0, q_1, \cdots, q_{m-1}\} \times \Sigma^*$	State Space $[0, m] \times [0, 1]$
State $(q_i, a_{-m} \ldots a_{-1}, a_0 \ldots a_n)$	State $\mathbf{x} = (x = s + y_2, y = y_1)$
q_1: if 2 is read, then write 4; goto q_2	$\begin{cases} x := x + 1 \\ y := y + \frac{2}{10} \end{cases}$ if $\begin{cases} 1 \leq x < 2 \\ \frac{2}{10} \leq y < \frac{3}{10} \end{cases}$

Alternative View of a Turing Machine

$$
\begin{aligned}
y_1 &= a_0 10^{-1} + a_1 10^{-2} + \ldots + a_n 10^{-n-1} \\
y_2 &= a_{-1} 10^{-1} + a_{-2} 10^{-2} + \ldots + a_{-k} 10^{-k}.
\end{aligned} \quad (2)
$$

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t))$$

Turing Machine	
State Space $\{q_0, q_1, \cdots, q_{m-1}\} \times \Sigma^*$	State Space $[0, m] \times [0, 1]$
State $(q_i, a_{-m} \ldots a_{-1}, a_0 \ldots a_n)$	State $\mathbf{x} = (x = s + y_2, y = y_1)$
q_1: if 2 is read, then write 4; goto q_2	$\begin{cases} x := x + 1 \\ y := y + \frac{2}{10} \end{cases}$ if $\begin{cases} 1 \leq x < 2 \\ \frac{2}{10} \leq y < \frac{3}{10} \end{cases}$
q_5: if 3 is read, then move right; goto q_1	

Alternative View of a Turing Machine

$$
\begin{aligned}
y_1 &= a_0 10^{-1} + a_1 10^{-2} + \ldots + a_n 10^{-n-1} \\
y_2 &= a_{-1} 10^{-1} + a_{-2} 10^{-2} + \ldots + a_{-k} 10^{-k}.
\end{aligned} \tag{2}
$$

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t))$$

Turing Machine	
State Space $\{q_0, q_1, \cdots, q_{m-1}\} \times \Sigma^*$	State Space $[0, m] \times [0, 1]$
State $(q_i, a_{-m} \ldots a_{-1}, a_0 \ldots a_n)$	State $\mathbf{x} = (x = s + y_2, y = y_1)$
q_1: if 2 is read, then write 4; goto q_2	$\begin{cases} x := x + 1 \\ y := y + \frac{2}{10} \end{cases}$ if $\begin{cases} 1 \leq x < 2 \\ \frac{2}{10} \leq y < \frac{3}{10} \end{cases}$
q_5: if 3 is read, then move right; goto q_1	$\begin{cases} x := \frac{x-5}{10} + \frac{3}{10} - 4 \\ y := 10y - 3 \end{cases}$ if $\begin{cases} 5 \leq x < 6 \\ \frac{3}{10} \leq y < \frac{4}{10} \end{cases}$

Alternative View of a Turing Machine

$$
\begin{aligned}
y_1 &= a_0 10^{-1} + a_1 10^{-2} + \ldots + a_n 10^{-n-1} \\
y_2 &= a_{-1} 10^{-1} + a_{-2} 10^{-2} + \ldots + a_{-k} 10^{-k}.
\end{aligned} \qquad (2)
$$

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t))$$

Turing Machine	
State Space $\{q_0, q_1, \cdots, q_{m-1}\} \times \Sigma^*$	State Space $[0, m] \times [0, 1]$
State $(q_i, a_{-m} \ldots a_{-1}, a_0 \ldots a_n)$	State $\mathbf{x} = (x = s + y_2, y = y_1)$
q_1: if 2 is read, then write 4; goto q_2	$\begin{cases} x := x + 1 \\ y := y + \frac{2}{10} \end{cases}$ if $\begin{cases} 1 \leq x < 2 \\ \frac{2}{10} \leq y < \frac{3}{10} \end{cases}$
q_5: if 3 is read, then move right; goto q_1	$\begin{cases} x := \frac{x-5}{10} + \frac{3}{10} - 4 \\ y := 10y - 3 \end{cases}$ if $\begin{cases} 5 \leq x < 6 \\ \frac{3}{10} \leq y < \frac{4}{10} \end{cases}$
q_3: if 5 is read, then move left; goto q_7	

Alternative View of a Turing Machine

$$\begin{aligned} y_1 &= a_0 10^{-1} + a_1 10^{-2} + \ldots + a_n 10^{-n-1} \\ y_2 &= a_{-1} 10^{-1} + a_{-2} 10^{-2} + \ldots + a_{-k} 10^{-k}. \end{aligned} \tag{2}$$

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t))$$

Turing Machine	
State Space $\{q_0, q_1, \cdots, q_{m-1}\} \times \Sigma^*$	State Space $[0, m] \times [0, 1]$
State $(q_i, a_{-m} \ldots a_{-1}, a_0 \ldots a_n)$	State $\mathbf{x} = (x = s + y_2, y = y_1)$
q_1: if 2 is read, then write 4; goto q_2	$\begin{cases} x := x + 1 \\ y := y + \frac{2}{10} \end{cases}$ if $\begin{cases} 1 \leq x < 2 \\ \frac{2}{10} \leq y < \frac{3}{10} \end{cases}$
q_5: if 3 is read, then move right; goto q_1	$\begin{cases} x := \frac{x-5}{10} + \frac{3}{10} - 4 \\ y := 10y - 3 \end{cases}$ if $\begin{cases} 5 \leq x < 6 \\ \frac{3}{10} \leq y < \frac{4}{10} \end{cases}$
q_3: if 5 is read, then move left; goto q_7	$\begin{cases} x := 10(x-3) - j + 4 \\ y := \frac{y}{10} + \frac{j}{10} \end{cases}$ if $\begin{cases} 3 + \frac{j}{10} \leq x < 3 + \frac{j+1}{10} \\ \frac{5}{10} \leq y < \frac{6}{10} \end{cases}$

Alternative View of a Turing Machine

$$\begin{aligned} y_1 &= a_0 10^{-1} + a_1 10^{-2} + \ldots + a_n 10^{-n-1} \\ y_2 &= a_{-1} 10^{-1} + a_{-2} 10^{-2} + \ldots + a_{-k} 10^{-k}. \end{aligned} \tag{2}$$

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t))$$

Turing Machine	
State Space $\{q_0, q_1, \cdots, q_{m-1}\} \times \Sigma^*$	State Space $[0, m] \times [0, 1]$
State $(q_i, a_{-m} \ldots a_{-1}, a_0 \ldots a_n)$	State $\mathbf{x} = (x = s + y_2, y = y_1)$
q_1: if 2 is read, then write 4; goto q_2	$\begin{cases} x := x + 1 \\ y := y + \frac{2}{10} \end{cases}$ if $\begin{cases} 1 \leq x < 2 \\ \frac{2}{10} \leq y < \frac{3}{10} \end{cases}$
q_5: if 3 is read, then move right; goto q_1	$\begin{cases} x := \frac{x-5}{10} + \frac{3}{10} - 4 \\ y := 10y - 3 \end{cases}$ if $\begin{cases} 5 \leq x < 6 \\ \frac{3}{10} \leq y < \frac{4}{10} \end{cases}$
q_3: if 5 is read, then move left; goto q_7	$\begin{cases} x := 10(x - 3) - j + 4 \\ y := \frac{y}{10} + \frac{j}{10} \end{cases}$ if $\begin{cases} 3 + \frac{j}{10} \leq x < 3 + \frac{j+1}{10} \\ \frac{5}{10} \leq y < \frac{6}{10} \end{cases}$ for $j \in \{0, 1, \ldots, 9\}$.

Alternative View of a Turing Machine

$$
\begin{aligned}
y_1 &= a_0 10^{-1} + a_1 10^{-2} + \ldots + a_n 10^{-n-1} \\
y_2 &= a_{-1} 10^{-1} + a_{-2} 10^{-2} + \ldots + a_{-k} 10^{-k}.
\end{aligned} \tag{2}
$$

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t))$$

Turing Machine	PAM
State Space $\{q_0, q_1, \cdots, q_{m-1}\} \times \Sigma^*$	State Space $[0, m] \times [0, 1]$
State $(q_i, a_{-m} \ldots a_{-1}, a_0 \ldots a_n)$	State $\mathbf{x} = (x = s + y_2, y = y_1)$
q_1: if 2 is read, then write 4; goto q_2	$\begin{cases} x := x + 1 \\ y := y + \frac{2}{10} \end{cases}$ if $\begin{cases} 1 \le x < 2 \\ \frac{2}{10} \le y < \frac{3}{10} \end{cases}$
q_5: if 3 is read, then move right; goto q_1	$\begin{cases} x := \frac{x-5}{10} + \frac{3}{10} - 4 \\ y := 10y - 3 \end{cases}$ if $\begin{cases} 5 \le x < 6 \\ \frac{3}{10} \le y < \frac{4}{10} \end{cases}$
q_3: if 5 is read, then move left; goto q_7	$\begin{cases} x := 10(x-3) - j + 4 \\ y := \frac{y}{10} + \frac{j}{10} \end{cases}$ if $\begin{cases} 3 + \frac{j}{10} \le x < 3 + \frac{j+1}{10} \\ \frac{5}{10} \le y < \frac{6}{10} \end{cases}$ for $j \in \{0, 1, \ldots, 9\}$.

Key remark: f (and $\overline{f}$) is piecewise affine

Morality

- If you prefer, a Turing Machine can be seen as a **piecewise affine function**
- It remains to simulate

$$\mathbf{x}(t+1) := \mathbf{f}(\mathbf{x}(t))$$

  for $t = 1, 2, \ldots$.

- ... to compute ...

Morality

- If you prefer, a Turing Machine can be seen as a **piecewise affine function**
- It remains to simulate

$$\mathbf{x}(t+1) := \mathbf{f}(\mathbf{x}(t))$$

for $t = 1, 2, \ldots$.

- . . . to compute . . .
  - Improvement:
    - We don't care about $\mathbf{f}$ on points not encoding a configuration.
    - Hence, with a slight modification, we can even assume $\mathbf{f}$ continuous, and even $\mathcal{C}^\infty$.

Sub-menu

Turing Machines

- Let M be some one tape Turing machine, with m states and 10 symbols.
- If

$$\ldots B\ B\ B\ a_{-k}\ a_{-k+1}\ldots\ a_{-1}\ a_0\ a_1\ldots\ a_n\ B\ B\ B\ldots$$

  is the tape content of M, it can be seen as

$$\begin{aligned} y_1 &= a_n \ldots a_1 a_0 \\ y_2 &= a_{-k} \ldots a_{-2} a_{-1} \end{aligned} \quad (3)$$

- The configuration of M is then given by three values: its internal state s, y_1 and y_2.

Alternative View of a Turing Machine

$$\begin{aligned} y_1 &= a_0 10^0 + a_1 10^1 + \ldots + a_n 10^n \\ y_2 &= a_{-1} 10^0 + a_{-2} 10^1 + \ldots + a_{-k} 10^{k-1} \end{aligned} \quad (4)$$

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t))$$

Turing Machine	
State Space $\{q_0, q_1, \cdots, q_{m-1}\} \times \Sigma^*$	
State $(q_i, a_{-m} \ldots a_{-1}, a_0 \ldots a_n)$	

Key remark: f (and $\overline{f}$) is (KM-)elementary

Alternative View of a Turing Machine

$$
\begin{aligned}
y_1 &= a_0 10^0 + a_1 10^1 + \ldots + a_n 10^n \\
y_2 &= a_{-1} 10^0 + a_{-2} 10^1 + \ldots + a_{-k} 10^{k-1}
\end{aligned} \tag{4}
$$

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t))$$

Turing Machine	
State Space $\{q_0, q_1, \cdots, q_{m-1}\} \times \Sigma^*$	State Space $\mathbb{N}^2$
State $(q_i, a_{-m}\ldots a_{-1}, a_0 \ldots a_n)$	State $\mathbf{x} = (x = s + my_2, y = y_1)$

Key remark: f **(and** $\overline{f}$**) is (KM-)elementary**

Alternative View of a Turing Machine

$$
\begin{aligned}
y_1 &= a_0 10^0 + a_1 10^1 + \ldots + a_n 10^n \\
y_2 &= a_{-1} 10^0 + a_{-2} 10^1 + \ldots + a_{-k} 10^{k-1}
\end{aligned} \tag{4}
$$

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t))$$

Turing Machine	
State Space $\{q_0, q_1, \cdots, q_{m-1}\} \times \Sigma^*$	State Space $\mathbb{N}^2$
State $(q_i, a_{-m}\ldots a_{-1}, a_0\ldots a_n)$	State $\mathbf{x} = (x = s + my_2, y = y_1)$
q_1: if 2 is read, then write 4; goto q_2	

Key remark: f **(and** $\overline{f}$**) is (KM-)elementary**

Alternative View of a Turing Machine

$$\begin{aligned} y_1 &= a_0 10^0 + a_1 10^1 + \ldots + a_n 10^n \\ y_2 &= a_{-1} 10^0 + a_{-2} 10^1 + \ldots + a_{-k} 10^{k-1} \end{aligned} \quad (4)$$

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t))$$

Turing Machine	
State Space $\{q_0, q_1, \cdots, q_{m-1}\} \times \Sigma^*$	State Space $\mathbb{N}^2$
State $(q_i, a_{-m} \ldots a_{-1}, a_0 \ldots a_n)$	State $\mathbf{x} = (x = s + my_2, y = y_1)$
q_1: if 2 is read, then write 4; goto q_2	$\begin{cases} x := x+1 \\ y := y+2 \end{cases}$ if $\begin{cases} mod_m(x) = 1 \\ mod_{10}(y) = 2 \end{cases}$

Key remark: f **(and** $\overline{f}$**) is (KM-)elementary**

Alternative View of a Turing Machine

$$y_1 = a_0 10^0 + a_1 10^1 + \ldots + a_n 10^n$$
$$y_2 = a_{-1} 10^0 + a_{-2} 10^1 + \ldots + a_{-k} 10^{k-1} \quad (4)$$

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t))$$

Turing Machine	
State Space $\{q_0, q_1, \cdots, q_{m-1}\} \times \Sigma^*$	State Space $\mathbb{N}^2$
State $(q_i, a_{-m} \ldots a_{-1}, a_0 \ldots a_n)$	State $\mathbf{x} = (x = s + my_2, y = y_1)$
q_1: if 2 is read, then write 4; goto q_2	$\begin{cases} x := x+1 \\ y := y+2 \end{cases}$ if $\begin{cases} mod_m(x) = 1 \\ mod_{10}(y) = 2 \end{cases}$
q_5: if 3 is read, then move right; goto q_1	

Key remark: f (and $\overline{f}$) is (KM-)elementary

Alternative View of a Turing Machine

$$
\begin{aligned}
y_1 &= a_0 10^0 + a_1 10^1 + \ldots + a_n 10^n \\
y_2 &= a_{-1} 10^0 + a_{-2} 10^1 + \ldots + a_{-k} 10^{k-1}
\end{aligned} \qquad (4)
$$

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t))$$

Turing Machine	
State Space $\{q_0, q_1, \cdots, q_{m-1}\} \times \Sigma^*$	State Space $\mathbb{N}^2$
State $(q_i, a_{-m} \ldots a_{-1}, a_0 \ldots a_n)$	State $\mathbf{x} = (x = s + my_2, y = y_1)$
q_1: if 2 is read, then write 4; goto q_2	$\begin{cases} x := x+1 \\ y := y+2 \end{cases}$ if $\begin{cases} mod_m(x) = 1 \\ mod_{10}(y) = 2 \end{cases}$
q_5: if 3 is read, then move right; goto q_1	$\begin{cases} x := 10(x-5) + 3 * m - 4 \\ y := (y-3)/10 \end{cases}$ if $\begin{cases} mod_m(x) = 5 \\ mod_{10}(y) = 3 \end{cases}$

Key remark: f **(and** $\overline{f}$**) is (KM-)elementary**

Alternative View of a Turing Machine

$$
\begin{aligned}
y_1 &= a_0 10^0 + a_1 10^1 + \dots + a_n 10^n \\
y_2 &= a_{-1} 10^0 + a_{-2} 10^1 + \dots + a_{-k} 10^{k-1}
\end{aligned} \quad (4)
$$

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t))$$

Turing Machine	
State Space $\{q_0, q_1, \cdots, q_{m-1}\} \times \Sigma^*$	State Space $\mathbb{N}^2$
State $(q_i, a_{-m} \dots a_{-1}, a_0 \dots a_n)$	State $\mathbf{x} = (x = s + m y_2, y = y_1)$
q_1: if 2 is read, then write 4; goto q_2	$\begin{cases} x := x+1 \\ y := y+2 \end{cases}$ if $\begin{cases} mod_m(x) = 1 \\ mod_{10}(y) = 2 \end{cases}$
q_5: if 3 is read, then move right; goto q_1	$\begin{cases} x := 10(x-5) + 3 * m - 4 \\ y := (y-3)/10 \end{cases}$ if $\begin{cases} mod_m(x) = 5 \\ mod_{10}(y) = 3 \end{cases}$
q_3: if 5 is read, then move left; goto q_7	

Key remark: f (and $\overline{f}$) is (KM-)elementary

Alternative View of a Turing Machine

$$\begin{aligned} y_1 &= a_0 10^0 + a_1 10^1 + \ldots + a_n 10^n \\ y_2 &= a_{-1} 10^0 + a_{-2} 10^1 + \ldots + a_{-k} 10^{k-1} \end{aligned} \qquad (4)$$

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t))$$

Turing Machine	
State Space $\{q_0, q_1, \cdots, q_{m-1}\} \times \Sigma^*$	State Space $\mathbb{N}^2$
State $(q_i, a_{-m} \ldots a_{-1}, a_0 \ldots a_n)$	State $\mathbf{x} = (x = s + my_2, y = y_1)$
q_1: if 2 is read, then write 4; goto q_2	$\begin{cases} x := x+1 \\ y := y+2 \end{cases}$ if $\begin{cases} mod_m(x) = 1 \\ mod_{10}(y) = 2 \end{cases}$
q_5: if 3 is read, then move right; goto q_1	$\begin{cases} x := 10(x-5) + 3*m - 4 \\ y := (y-3)/10 \end{cases}$ if $\begin{cases} mod_m(x) = 5 \\ mod_{10}(y) = 3 \end{cases}$
q_3: if 5 is read, then move left; goto q_7	$\begin{cases} x := (x - j*m - 3)/10 + 4 \\ y := 10y + j \end{cases}$ if $\begin{cases} mod_m(x) = 3 \\ mod_{10}(\frac{x-3}{10}) = j \\ mod_{10}(y) = 5 \end{cases}$

Key remark: f **(and** $\overline{f}$**) is (KM-)elementary**

Alternative View of a Turing Machine

$$\begin{aligned} y_1 &= a_0 10^0 + a_1 10^1 + \ldots + a_n 10^n \\ y_2 &= a_{-1} 10^0 + a_{-2} 10^1 + \ldots + a_{-k} 10^{k-1} \end{aligned} \quad (4)$$

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t))$$

Turing Machine	
State Space $\{q_0, q_1, \cdots, q_{m-1}\} \times \Sigma^*$	State Space $\mathbb{N}^2$
State $(q_i, a_{-m} \ldots a_{-1}, a_0 \ldots a_n)$	State $\mathbf{x} = (x = s + my_2, y = y_1)$
q_1: if 2 is read, then write 4; goto q_2	$\begin{cases} x := x+1 \\ y := y+2 \end{cases}$ if $\begin{cases} mod_m(x) = 1 \\ mod_{10}(y) = 2 \end{cases}$
q_5: if 3 is read, then move right; goto q_1	$\begin{cases} x := 10(x-5) + 3*m - 4 \\ y := (y-3)/10 \end{cases}$ if $\begin{cases} mod_m(x) = 5 \\ mod_{10}(y) = 3 \end{cases}$
q_3: if 5 is read, then move left; goto q_7	$\begin{cases} x := (x - j*m - 3)/10 + 4 \\ y := 10y + j \end{cases}$ if $\begin{cases} mod_m(x) = 3 \\ mod_{10}(\frac{x-3}{10}) = j \\ mod_{10}(y) = 5 \end{cases}$ for $j \in \{0, 1, \ldots, 9\}$.

Key remark: f (and $\overline{f}$) is (KM-)elementary

Alternative View of a Turing Machine

$$
\begin{aligned}
y_1 &= a_0 10^0 + a_1 10^1 + \ldots + a_n 10^n \\
y_2 &= a_{-1} 10^0 + a_{-2} 10^1 + \ldots + a_{-k} 10^{k-1}
\end{aligned} \tag{4}
$$

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t))$$

Turing Machine	(KM-)Elementary
State Space $\{q_0, q_1, \cdots, q_{m-1}\} \times \Sigma^*$	State Space $\mathbb{N}^2$
State $(q_i, a_{-m} \ldots a_{-1}, a_0 \ldots a_n)$	State $\mathbf{x} = (x = s + m y_2, y = y_1)$
q_1: if 2 is read, then write 4; goto q_2	$\begin{cases} x := x+1 \\ y := y+2 \end{cases}$ if $\begin{cases} mod_m(x) = 1 \\ mod_{10}(y) = 2 \end{cases}$
q_5: if 3 is read, then move right; goto q_1	$\begin{cases} x := 10(x-5) + 3 * m - 4 \\ y := (y-3)/10 \end{cases}$ if $\begin{cases} mod_m(x) = 5 \\ mod_{10}(y) = 3 \end{cases}$
q_3: if 5 is read, then move left; goto q_7	$\begin{cases} x := (x - j * m - 3)/10 + 4 \\ y := 10y + j \end{cases}$ if $\begin{cases} mod_m(x) = 3 \\ mod_{10}(\frac{x-3}{10}) = j \\ mod_{10}(y) = 5 \end{cases}$ for $j \in \{0, 1, \ldots, 9\}$.

Key remark: f (and $\overline{f}$) is (KM-)elementary

Morality

- If you prefer, a Turing Machine can be seen as a **(KM-)elementary map**
- It remains to simulate

$$\mathbf{x}(t+1) := f(\mathbf{x}(t))$$

for $t = 1, 2, \ldots$.

- ... to compute ...

Koiran-Moore's 99 result

- **Theorem [KM99]:** For any Turing machine M and any input w, there is an elementary function $\mathbf{f}$ on two variables and constants a and b such that M halts on input w after t steps if and only if $f^{[t]}(a + bw, 0) = (0, 0)$.

Koiran-Moore's 99 result

- **Theorem [KM99]:** For any Turing machine M and any input w, there is an elementary function **f** on two variables and constants a and b such that M halts on input w after t steps if and only if $f^{[t]}(a + bw, 0) = (0, 0)$.

  - Using the trick that $\mod_2(x)$ is basically $\sin(\pi x)^2$, etc.

Koiran-Moore's 99 result

- **Theorem [KM99]:** For any Turing machine M and any input w, there is an elementary function $\mathbf{f}$ on two variables and constants a and b such that M halts on input w after t steps if and only if $f^{[t]}(a+bw,0)=(0,0)$.
  - $f:\mathbb{R}^n \to \mathbb{R}^n$ is (KM-)elementary if its n components are in U_n, where U_n is the smallest class of functions $f:\mathbb{R}^n \to \mathbb{R}$ containing rational constants, π, the n projections $x \mapsto x_i$ and satisfying the following closure properties:
    - if $f, g \in U_n$ then $f \oplus g \in U_n$, where $\oplus \in \{+,-,\times\}$
    - if $f \in U_n$ then $\sin(f) \in U_n$
  - Using the trick that $\bmod_2(x)$ is basically $\sin(\pi x)^2$, etc.

Graça-Campagnolo-Buescu'2005

- Remarks:
  - Key point: With this encoding, integers are sent to integers...
  - $\sigma(x) = x - 0.2 \sin(2\pi x)$ is basically a contraction on the vicinity of integers.

(static error correction:)

- **Theorem [GCB'05]:** Let $0 < \delta < \epsilon < 1/2$. The transition function θ of a TM admits an analytic extension $f_M : \mathbb{R}^3 \to \mathbb{R}^3$, robust to perturbations.

Graça-Campagnolo-Buescu'2005

- Remarks:
  - Key point: With this encoding, integers are sent to integers...
  - $\sigma(x) = x - 0.2\sin(2\pi x)$ is basically a contraction on the vicinity of integers.

(static error correction:)

- **Theorem [GCB'05]:** Let $0 < \delta < \epsilon < 1/2$. The transition function θ of a TM admits an analytic extension $f_M : \mathbb{R}^3 \to \mathbb{R}^3$, robust to perturbations.
  - Ie: for all f such that $\|f - f_M\| \leq \delta$, and for all $\overline{x_0} \in \mathbb{R}^3$ satisfying $\|\overline{x_0} - x_0\| \leq \epsilon$, where $x_0 \in \mathbb{N}^3$ represents an initial configuration,

$$\left\| f^{[j]}(\overline{x_0}) - \theta^{[j]}(x_0) \right\| \leq \epsilon \text{ for all } j \in \mathbb{N}.$$

Graça-Campagnolo-Buescu'2005

- Remarks:
  - Key point: With this encoding, integers are sent to integers...
  - $\sigma(x) = x - 0.2\sin(2\pi x)$ is basically a contraction on the vicinity of integers.

    (static error correction:)

  - For
    $$l_2(x, y) = \frac{1}{\pi}\arctan(4y(x - 1/2)) + \frac{1}{2},$$
    we have $|a - l_2(\bar{a}, y)| < 1/y$ for $\bar{a}$ close to $a \in \{0, 1\}$, $y > 0$

    (dynamic error correction:)

- **Theorem [GCB'05]:** Let $0 < \delta < \epsilon < 1/2$. The transition function θ of a TM admits an analytic extension $f_M : \mathbb{R}^3 \to \mathbb{R}^3$, robust to perturbations.
  - Ie: for all f such that $\|f - f_M\| \leq \delta$, and for all $\overline{x_0} \in \mathbb{R}^3$ satisfying $\|\overline{x_0} - x_0\| \leq \epsilon$, where $x_0 \in \mathbb{N}^3$ represents an initial configuration,
    $$\left\| f^{[j]}(\overline{x_0}) - \theta^{[j]}(x_0) \right\| \leq \epsilon \text{ for all } j \in \mathbb{N}.$$

Menu

Sub-menu

Some ODEs...

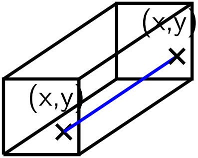


Figure: A linear path.

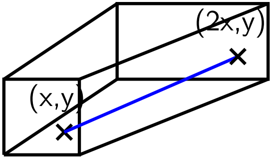


Figure: A dilation (acting on x of factor 2).

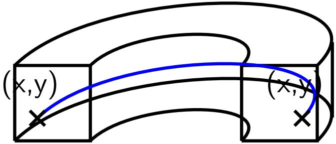


Figure: A U-turn.

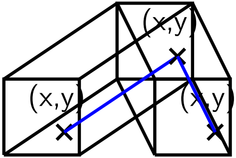
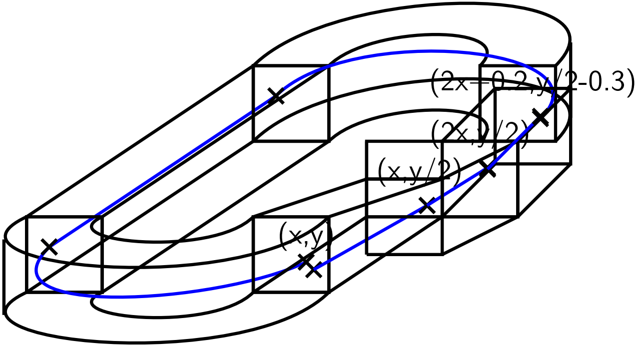
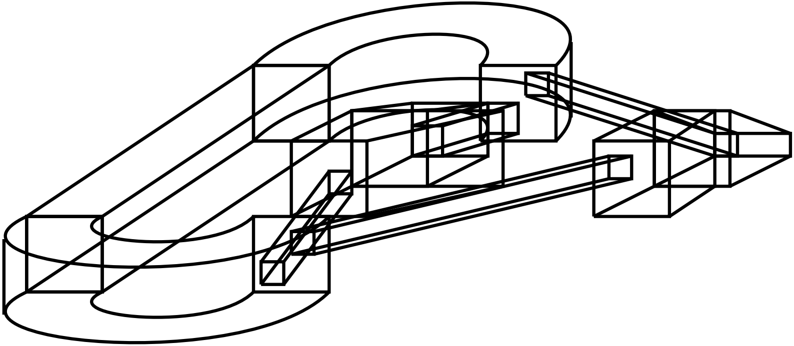


Figure: A merge (symbolic view: this can exist only in dimension 4) .

Some ODEs. . .



Some ODEs. . .



Sub-menu

Branicky's clock (1995): with non-analytic functions

- We basically need to do $\mathbf{x} := \mathbf{f}(\mathbf{x})$ repeatedly

Branicky's clock (1995): with non-analytic functions

- Doing $\mathbf{x}_2 := \mathbf{f}(\mathbf{x}_1)$; $\mathbf{x}_1 := \mathbf{x}_2$ repeatedly is fine.

Branicky's clock (1995): with non-analytic functions

- Doing $\mathbf{x}_2 := \mathbf{f}(\mathbf{x}_1)$; $\mathbf{x}_1 := \mathbf{x}_2$ repeatedly is fine.

- Key observation: the solution of

$$y' = c(g - y)^3 \varphi(t)$$

converges at $t = 1/2$ close to the goal g with some arbitrary precision, independently from initial condition at $t = 0$

for any function φ of positive integral if c is sufficiently big.

Branicky's clock (1995): with non-analytic functions

- Doing $\mathbf{x}_2 := \mathbf{f}(\mathbf{x}_1)$; $\mathbf{x}_1 := \mathbf{x}_2$ repeatedly is fine.

- Key observation: the solution of

$$y' = c(g - y)^3 \varphi(t)$$

converges at $t = 1/2$ close to the goal g with some arbitrary precision, independently from initial condition at $t = 0$

for any function φ of positive integral if c is sufficiently big.

  - **If you prefer, this roughly does $y(1/2) := g$.**

Branicky's clock (1995): with non-analytic functions

- Doing $\mathbf{x}_2 := \mathbf{f}(\mathbf{x}_1)$; $\mathbf{x}_1 := \mathbf{x}_2$ repeatedly is fine.

- Key observation: the solution of

$$y' = c(g - y)^3 \varphi(t)$$

converges at $t = 1/2$ close to the goal g with some arbitrary precision, independently from initial condition at $t = 0$

for any function φ of positive integral if c is sufficiently big.

  - **If you prefer, this roughly does $y(1/2) := g$.**

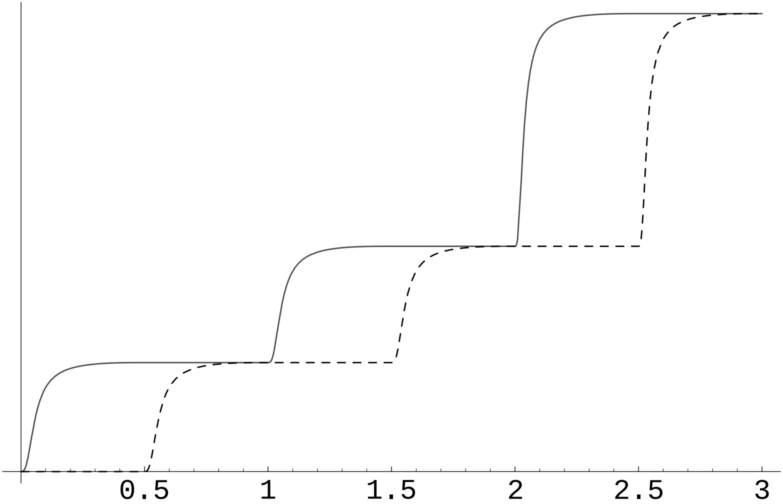
- The following system is then a solution

$$\begin{cases} \mathbf{x}_1' = c_1(\mathbf{r}(\mathbf{x}_2) - \mathbf{x}_1)^3 \theta(-\sin(2\pi t)) \\ \mathbf{x}_2' = c_2(\mathbf{f}(\mathbf{r}(\mathbf{x}_1)) - \mathbf{x}_2)^3 \theta(\sin(2\pi t)) \end{cases} \quad \begin{cases} \mathbf{x}_1(0) = \mathbf{x}_0 \\ \mathbf{x}_2(0) = \mathbf{x}_0 \end{cases}$$

considering functions:

  - θ such that $\theta(x) = 0$ if $x \leq 0$, $\theta(x) = x^2$ if $x \geq 0$.
  - $r(x) = j$ whenever $x \in [j - 1/4, j + 1/4]$, for $j \in \mathbb{Z}$.

Example: $y(t+1) := 2 * y(t)$



Simulation of iterations of $h(n) = 2^n$ by ODEs.

Sub-menu

If analytic functions are forbidden...

- We want polynomial ODEs
  - non analytics functions (e.g. θ, r) are forbidden.
- Requires to program with ODEs.
  - and to deal with errors ...

Errors...

- We would dream to do:

$$y' = c(g - y)^3 \varphi(t)$$

- We do at best something like

$$z' = c(\bar{g}(t) - z)^3 \varphi(t) + E(t)$$

where $|\bar{g}(t) - g| \leq \rho$ and $|E(t)| \leq \delta$.

  - not so bad if ρ, δ small, and c big enough.

What do we get at the end?

- We would dream to do:

$$\left\{ \begin{array}{lcl} \mathbf{x}_1' & = & c_1(\mathbf{r}(\mathbf{x}_2) - \mathbf{x}_1)^3 \theta(-\sin(2\pi t)) \\ \mathbf{x}_2' & = & c_2(\mathbf{f}(\mathbf{r}(\mathbf{x}_1)) - \mathbf{x}_2)^3 \theta(\sin(2\pi t)) \end{array} \right.$$

- We do something like

$$\mathbf{x}_1' = c_1 \left(\sigma^{[n]}(\mathbf{x}_2) - \mathbf{x}_1 \right)^3 \zeta_{\epsilon_1}(t)$$
$$\mathbf{x}_2' = c_2 \left(\mathbf{f} \circ \sigma^{[m]}(\mathbf{x}_1) - \mathbf{x}_2 \right)^3 \zeta_{\epsilon_2}(-t)$$

Considering

$$\zeta_\epsilon(t) = l_2(\vartheta(t), 1/\epsilon),$$
$$\vartheta(t) = \frac{1}{2} \left(\sin^2(2\pi t) + \sin(2\pi t) \right)$$
$$l_2(x, y) = \frac{1}{\pi} \arctan(4y(x - 1/2)) + \frac{1}{2}.$$
$$\sigma(x) = x - 0.2 \sin(2\pi x).$$

$+$ dynamic error control on $\mathbf{f}$

Dynamic error control on $\mathbf{f}$

ω of period 10, $\bar{y} = \omega(\bar{y}_1)$

$$f_M(\overline{y}_1, \overline{y}_2, \bar{q}) = (\overline{y}_1^{next}, \overline{y}_2^{next}, \overline{q}_{next})$$

$$\overline{y}_1^{next} = \overline{P}_1 \frac{1}{2}(1-H)(2-H) + \overline{P}_2 H(2-H) + \overline{P}_3(-\frac{1}{2})H(1-H), \qquad (5)$$

with (move left, don't move, move right:)

$$\begin{aligned}
\overline{P}_1 &= 10(\sigma^{[j]}(\overline{y}_1) + \sigma^{[j]}(\overline{s}_{next}) - \sigma^{[j]}(\overline{y}) + \sigma^{[j]} \circ \omega \circ \sigma^{[j]}(\overline{y}_2) \\
\overline{P}_2 &= \sigma^{[j]}(\overline{y}_1) + \sigma^{[j]}(\overline{s}_{next}) - \sigma^{[j]}(\overline{y}) \\
\overline{P}_3 &= \frac{\sigma^{[j]}(\overline{y}_1) - \sigma^{[j]}(\overline{y})}{10},
\end{aligned}$$

$$H = l_3(\overline{h}_3, 10000(\overline{y}_1 + 1/2) + 2).$$

$$q_{next} = \sum_{i=0}^{9} \sum_{j=1}^{m} \left(\prod_{r=0, r\neq i}^{9} \frac{(\sigma^{[n]}(\overline{y}) - r)}{(i-r)} \right) \left(\prod_{s=1, s\neq j}^{m} \frac{(\sigma^{[n]}(\overline{q}) - s)}{(j-s)} \right) q_{i,j}, \qquad (6)$$

$s =$ interpolation of same type

Some philosophical comments

- We are basically considering ODEs which are obtained by simulating dODEs (iterations)

- Basically,

  - we usually have in mind some discrete time/space model/reasonning

    (errorless)

  - That we simulate with some discete time/real space model

    (if we want analycity, should take care of errors)

  - That we simulate in turn continuous time models

    (introduces even more errors)

Menu

Analog Computations: Our actual motivation

How to Compute with Iterations (dODEs)

How to Compute with Ordinary Differential Equations

Some computability results based on these ideas

Conclusion

Sub-menu

- Some computability results based on these ideas
  - Doing stuffs with polynomial ODEs
  - Computable Analysis with GPACs

Discrete time simulation with a GPAC

Proposition ([**?**, **?**])

There is a computable polynomial p *and some computable value* $\alpha \in \mathbb{R}^n$

$$z' = p(z,t), \qquad z(0) = (\tilde{x}_0, \alpha)$$

such that for all $\tilde{x}_0 \in \mathbb{R}^{2l+1}$ *satisfying* $\|\tilde{x}_0 - x_0\|_\infty \leq \varepsilon$*, one has*

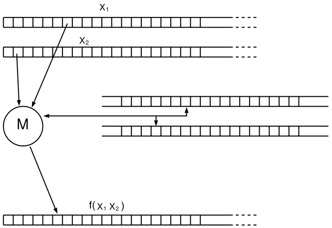
$$\left\| z_1(t) - \psi_M^{[j]}(x_0) \right\|_\infty \leq \delta.$$

for all $j \in \mathbb{N}$ *and for all* $t \in [j, j+1/2]$*.*

Sub-menu

Computable Analysis

Due to Turing, Grzegorczyk, Lacombe. Here presentation from Weihrauch.



A tape represents a real number

Each real number x is represented via an infinite sequence $(x_n)_n \in \mathbb{Q}$,

$$||x_n - x|| \leq 2^{-n}$$

M behaves like a Turing Machine

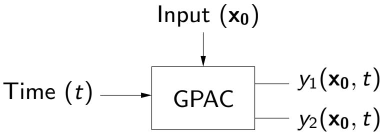
Read-only one-way input tapes
Write-only one-way output tape.
M outputs a representation of $f(x_1, x_2)$ from representations of x_1, x_2.

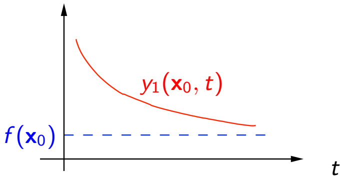
GPAC Computability vs GPAC Generation

Definition

A function $f : [a, b] \to \mathbb{R}$ is GPAC-computable iff there exist some computable polynomials $p : \mathbb{R}^{n+1} \to \mathbb{R}^n$, $p_0 : \mathbb{R} \to \mathbb{R}$, and $n-1$ computable real values $\alpha_1, \ldots, \alpha_{n-1}$ such that:

1. $(y_1, \ldots, y_n)$ is the solution of the Cauchy problem $y' = p(y, t)$ with initial condition $(\alpha_1, \ldots, \alpha_{n-1}, p_0(x))$ set at time $t_0 = 0$
2. $\lim_{t\to\infty} y_2(t) = 0$
3. $|f(x) - y_1(t)| \leq y_2(t)$ for all $x \in [a, b]$ and all $t \in [0, +\infty)$.





Simulating Type-2 machines with a GPAC

Theorem ([?])

Let $f : [a, b] \to \mathbb{R}$ *be a computable function. Then there exists a GPAC and some index* i *such that if we set the initial conditions* $(x, \bar{n}) \in [a, b] \times \mathbb{R}$*, where* $|\bar{n} - n| \leq \varepsilon < 1/2$*, with* $n \in \mathbb{N}$*, there exists some* $T \geq 0$ *such that the output* y_i *of the GPAC satisfies* $|y_i(t) - f(x)| \leq 2^{-n}$ *for all* $t \geq T$*.*

Theorem (Bournez-Campagnolo-Graça-Hainry's Theorem [?])

Let a *and* b *be computable reals. A function* $f : [a, b] \to \mathbb{R}$ *is computable iff it is GPAC-computable.*

Idea of the construction

Proposition (e.g. Ko 91)

A real function $f : [a, b] \to \mathbb{R}$ *is computable iff there exist three computable functions* $m : \mathbb{N} \to \mathbb{N}$, $sgn, abs : \mathbb{N}^4 \to \mathbb{N}$ *such that:*

1. m *is a modulus of continuity for* f, *i.e. for all* $n \in \mathbb{N}$ *and all* $x, y \in [a, b]$, *one has*

$$|x - y| \leq 2^{-m(n)} \quad \Longrightarrow \quad |f(x) - f(y)| \leq 2^{-n}$$

2. *For all* $(i, j, k) \in \mathbb{N}^3$ *such that* $(-1)^i j/2^k \in [a, b]$, *and all* $n \in \mathbb{N}$,

$$\left|(-1)^{sgn(i,j,k,n)} \frac{abs(i, j, k, n)}{2^n} - f\left((-1)^i \frac{j}{2^k}\right)\right| \leq 2^{-n}.$$

General Idea

Idea

- From $n \in \mathbb{N}$, $x \in \mathbb{R}$,
  - compute integers i, j, k such that that

$$|(-1)^i j/2^k - x| \leq 2^{-m(n)}$$

  - compute values $sgn(i,j,k,n)$ and $abs(i,j,k,n)$.
  - output

$$(-1)^{sgn(i,j,k,n)} \frac{abs(i,j,k,n)}{2^n},$$

    guaranteed to be at 2^{-n} of $f(x)$

- Then increase n, and restart.

General Idea

Idea

- From $n \in \mathbb{N}$, $x \in \mathbb{R}$,
  - compute integers i, j, k such that that

$$|(-1)^i j/2^k - x| \leq 2^{-m(n)}$$

  - compute values $sgn(i, j, k, n)$ and $abs(i, j, k, n)$.
  - output

$$(-1)^{sgn(i,j,k,n)} \frac{abs(i, j, k, n)}{2^n},$$

  guaranteed to be at 2^{-n} of $f(x)$

- Then increase n, and restart.

Problems / Solutions.

General Idea

Idea

- From $n \in \mathbb{N}$, $x \in \mathbb{R}$,
  - compute integers i, j, k such that that

$$|(-1)^i j/2^k - x| \leq 2^{-m(n)}$$

  - compute values $sgn(i,j,k,n)$ and $abs(i,j,k,n)$.
  - output

$$(-1)^{sgn(i,j,k,n)} \frac{abs(i,j,k,n)}{2^n},$$

    guaranteed to be at 2^{-n} of $f(x)$

- Then increase n, and restart.

Problems / Solutions.

1. Since we are on a compact, w.l.o.g. we can assume $x \geq 0$, and take $i = 1$ constant.

General Idea

Idea

- From $n \in \mathbb{N}$, $x \in \mathbb{R}$,
  - compute integers i, j, k such that that
  $$|(-1)^i j/2^k - x| \leq 2^{-m(n)}$$
  - compute values $sgn(i, j, k, n)$ and $abs(i, j, k, n)$.
  - output
  $$(-1)^{sgn(i,j,k,n)} \frac{abs(i, j, k, n)}{2^n},$$
  guaranteed to be at 2^{-n} of $f(x)$
- Then increase n, and restart.

Problems / Solutions.

1. One can take $k = m(n)$.

General Idea

Idea

- From $n \in \mathbb{N}$, $x \in \mathbb{R}$,
  - compute integers i, j, k such that that
  $$|(-1)^i j/2^k - x| \leq 2^{-m(n)}$$
  - compute values $sgn(i,j,k,n)$ and $abs(i,j,k,n)$.
  - output
  $$(-1)^{sgn(i,j,k,n)} \frac{abs(i,j,k,n)}{2^n},$$
  guaranteed to be at 2^{-n} of $f(x)$
- Then increase n, and restart.

Problems / Solutions.

1. Taking $j = \lceil x 2^{m(n)} \rceil$ would be great, but integer part is not analytic.

General Idea

Idea

- From $n \in \mathbb{N}$, $x \in \mathbb{R}$,
  - compute integers i, j, k such that that

$$|(-1)^i j/2^k - x| \leq 2^{-m(n)}$$

  - compute values $sgn(i, j, k, n)$ and $abs(i, j, k, n)$.
  - output

$$(-1)^{sgn(i,j,k,n)} \frac{abs(i, j, k, n)}{2^n},$$

  guaranteed to be at 2^{-n} of $f(x)$

- Then increase n, and restart.

Problems / Solutions.

1. Taking $j = x2^{m(n)}$ yields valid output y_1 only when $x2^{m(n)}$ is close to an integer.

General Idea

Idea

- From $n \in \mathbb{N}$, $x \in \mathbb{R}$,
  - compute integers i, j, k such that that

$$|(-1)^i j/2^k - x| \leq 2^{-m(n)}$$

  - compute values $sgn(i,j,k,n)$ and $abs(i,j,k,n)$.
  - output

$$(-1)^{sgn(i,j,k,n)} \frac{abs(i,j,k,n)}{2^n},$$

  guaranteed to be at 2^{-n} of $f(x)$

- Then increase n, and restart.

Problems / Solutions.

1. Taking $j = x2^{m(n)} + 1/2$ yields valid output y_1' only when $x2^{m(n)} - 1/2$ is close to an integer.

Getting a Valid Output For All x

Actually, according to the value of $\overline{k}_1 = x2^{m(n)}$ either y_1 or y_1' is valid.
We consider

$$\bar{y} = \frac{\omega_1(\overline{k}_1)y_1 + \omega_2(\overline{k}_1)y_1'}{\omega_1(\overline{k}_1) + \omega_2(\overline{k}_1)}. \tag{7}$$

where ω_1 (respectively ω_2) is ≥ 0 and close to 0 iff y_2 is valid (resp. y_2' is valid).

Last Ingredients (no details)

- We need to know when a computation is terminated.
- We use error-free external clocks
- We need to be able to switch dynamics
- We need to be able to reset machines

Menu

Take home message

- **Turing machines $\sim$ polynomial Ordinary Differential Equations**
  I.e.

$$\begin{aligned} \mathbf{y}' &= \mathbf{p}(t, \mathbf{y}) \\ \mathbf{y}(t_0) &= \mathbf{y}_0 \end{aligned}$$

where $\mathbf{p}$ is a (vector of) polynomials.

in a very very strong sense.

- Programming with/Solving ODEs is **simple** and **fun**.

Take home message

- **Turing machines $\sim$ polynomial Ordinary Differential Equations**
  I.e.
  $$\begin{aligned} \mathbf{y}' &= \mathbf{p}(t, \mathbf{y}) \\ \mathbf{y}(t_0) &= \mathbf{y}_0 \end{aligned}$$
  where $\mathbf{p}$ is a (vector of) polynomials.

  in a very very strong sense.

- Programming with/Solving ODEs is **simple** and **fun**.

- Analog's world: Many concepts from **computer science** can be defined using polynomial ODEs
  - Computable functions.
  - Polynomial Time Computable Functions
  - *NP*, *PSPACE*, ...?
  - Revisiting computation theory with pODEs ...
  - Bioinformatics (proteins) computations $\geq$ Turing machines $=$ Classical computers.
  - ...