Numerical Non-standard Calculus: Applications and Software Implementation

Lorenzo Fiaschi

July 10, 2023

Department of Information Engineering
University of Pisa
lorenzo.fiaschi@ing.unipi.it

The vision

The goal of this research

- Make NSA numerical and use it in Engineering applications
- The steps:
    1. Propose a numerical encoding for non-standard numbers
    2. Implement a software library to execute non-standard computations
    3. Identify and tackle real-world Engineering applications
    4. Design a hardware accelerator for non-standard computations (co-processor)
- Five applications:
    - × Linear Programming
    - × Game Theory
    - ✓ Quadratic Programming
    - × Evolutionary Optimization
    - ✓ Reinforcement Learning

Non-standard model

Axiom (Existence)
Every sequence $\varphi(n)$ has a unique α-limit denoted by $\lim_{n\uparrow\alpha}\varphi(n)$.

Axiom (Alpha Number)
The α-limit of the identity sequence $i(n)=n$ is a number denoted by α, that is $\lim_{n\uparrow\alpha} n=\alpha\notin\mathbb{N}$.

Axiom (Field Axiom)
The set of all α-limits of real sequences

$$^{*}\mathbb{R}=\left\{\lim_{n\uparrow\alpha}\varphi(n)\,\middle|\,\varphi\colon\mathbb{N}\to\mathbb{R}\right\}$$

is a field, called the hyperreal field, where:

- $\lim_{n\uparrow\alpha}\varphi(n)+\lim_{n\uparrow\alpha}\psi(n)=\lim_{n\uparrow\alpha}(\varphi(n)+\psi(n))$
- $\lim_{n\uparrow\alpha}\varphi(n)\cdot\lim_{n\uparrow\alpha}\psi(n)=\lim_{n\uparrow\alpha}(\varphi(n)\cdot\psi(n))$

From theory to computations: the bounded algorithmic numbers and the BANs library

Algorithmic numbers

Definition (monosemium)
$\xi \in {}^*\mathbb{R}$ is called monosemium if $\exists r \in \mathbb{R}$ and $p \in \mathbb{Q}$ such that

$$\xi = r\alpha^p.$$

Definition (Algorithmic number)
A number $\xi \in {}^*\mathbb{R}$ is called algorithmic if it can be represented as a finite sum of monosemia, namely

$$\xi = \sum_{k=1}^{\ell} r_k \alpha^{s_k}; \quad r_k \in \mathbb{R},\ s_k \in \mathbb{Q};\ s_k > s_{k+1}.$$

Proposition (AN normal form)
Any AN can be represented in the following "normal form":

$$\xi = \alpha^p P\left(\eta^{\frac{1}{m}}\right),$$

where $p \in \mathbb{Q}$, $m \in \mathbb{N}$, and $P(x)$ is a polynomial with real coefficients such that $P(0) \neq 0$.

Algorithmic numbers

ANs still require infinite memory

- Not closed w.r.t. division ($\eta := \alpha^{-1}$)

$$\frac{1}{\alpha + 1} = \eta - \eta^2 + \eta^3 - \ldots = \sum_{i=1}^{\infty} (-1)^{i-1} \eta^i$$

- Requires exact arithmetic for representing rational powers

$$\alpha^{\frac{1}{6}} \cdot \alpha^2 = \alpha^{\frac{2}{6}} = \alpha^{\frac{1}{3}}$$

Bounded algorithmic numbers

Definition (Truncation function)
Given a polynomial $P(x) = p_0 x^{z_0} + \ldots + p_m x^{z_m}$, $z_{i-1} < z_i$, $i = 1, \ldots, m$, *the truncation function* $\mathfrak{tr}$ *with truncation parameter* n *is defined as follows:*

$$\mathfrak{tr}_n \left[P\left(x \right) \right] = \begin{cases} P(x) & n \geq m \\ p_0 x^{z_0} + \ldots + p_n x^{z_n} & n < m \end{cases}$$

Definition (Bounded algorithmic number)
A BAN is any AN who admits the following normal form:

$$\xi = \alpha^p P(\eta),$$

where $p \in \mathbb{Z}$ *and* $P(0) \neq 0$.

BANs library

```julia
abstract type AbstractAlgNum <: Number end

const SIZE = 3;

# Ban declaration
mutable struct Ban <: AbstractAlgNum
    # Members
    p::Int
    coef::Array{T,1} where T<:Real

    # Constructor
    Ban(p::Int,coef::Array{T,1}, check::Bool) where T <: Real = new(p,copy(coef))
    Ban(p::Int,coef::Array{T,1}) where T <: Real =
                                (_constraints_satisfaction(p,coef) && new(p,copy(coef)))
    Ban(a::Ban) = new(a.p,copy(a.coef))
    Ban(x::Bool) = one(Ban)
    Ban(x::T) where T<:Real = ifelse(isinf(x), Ban(0, ones(SIZE).*x), one(Ban)*x)
end

# α constant
const α = Ban(1, [one(Int64); zeros(Int64, SIZE-1)], false);
# η constant
const η = Ban(-1, [one(Int64); zeros(Int64, SIZE-1)], false);
```

Lexicographic multi-objective optimization

Definition (lexicographic multi-objective program)
Let $\mathbb{V}$ and $\mathbb{F}$ be a vectorial space and a number field, respectively. Let also $f_1, \ldots, f_n$ be a finite sequence of scalar functions such that $f_i : \mathbb{V} \to \mathbb{F}$, $i = 1, \ldots, n$. Then, a lexicographic multi-objective optimization problem consists of the following programs in cascade:

$$\begin{aligned} \min \quad & f_1(x) \\ \text{s.t.} \quad & x \in \Omega \end{aligned} \qquad \begin{aligned} \min \quad & f_i(x) \\ \text{s.t.} \quad & x \in \Omega, \\ & f_j(x) = \bar{f}_j \quad j = 1, \ldots, i-1 \end{aligned}$$

where $\Omega \subseteq \mathbb{V}$ is the problem domain and $\bar{f}_j$ is the optimal value of the j-th program, $j = 1, \ldots, n-1$.

A result

Theorem
Consider an n-objective LMOP, where $f_i \colon \Omega \to \mathbb{R}$, $i = 1, \dots, n$, $\Omega \subseteq \mathbb{R}^m$, $m \in \mathbb{N}$, and the priority is induced by the natural order. Then, $\exists F \colon \Omega \to {}^*\mathbb{R}$ such that the following is an equivalent scalar program:

$$\begin{aligned} \min \quad & F(x) \\ \text{s.t.} \quad & x \in \Omega \end{aligned}$$

In particular

$$F(x) = \beta_1 f_1(x) + \ldots + \beta_n f_n(x),$$

$$\beta_i \in {}^*\mathbb{R}^+ \quad \forall i = 1, \ldots, n,$$

and

$$\frac{\beta_{i+1}}{\beta_i} \approx 0 \quad \forall i = 1, \ldots, n-1.$$

Preemptive

$$\begin{aligned} \min \quad & f_1(x) \\ \text{s.t.} \quad & x \in \Omega \end{aligned}$$

$$\begin{aligned} \min \quad & f_i(x) \\ \text{s.t.} \quad & x \in \Omega, \\ & f_j(x) = \bar{f}_j \quad j = 1, \ldots, i-1 \end{aligned}$$

- Direct application of the definition
- Inefficient
- Different optimizers could be needed
- Equivalent to the original problem

Standard approaches to LMOP

Scalarization

$$\begin{aligned} \min \quad & w_1 f_1(x) + \ldots + w_n f_n(x) \\ \text{s.t.} \quad & x \in \Omega \end{aligned}$$

- $w_i \in \mathbb{R}^+, \; i = 1, \, \ldots, \; n$
- $\frac{w_{i+1}}{w_i} \ll 1, \; i = 1, \ldots, \; n - 1$
- Efficient optimization
- Reuse of existing algorithms
- Lack of guarantee to be equivalent to the original problem

Application to lexicographic quadratic programming

Reviewing quadratic programming

Definition (Quadratic program)
A quadratic program is an optimization problem having the following form:

$$\begin{aligned} \min \quad & \frac{1}{2}x^T Q x + c^T x \\ \text{s.t.} \quad & Ax = b, \\ & x \geq 0 \end{aligned} \qquad\qquad \begin{aligned} \max_{x, \lambda} \quad & -\frac{1}{2}x^T Q x + b^T \lambda \\ \text{s.t.} \quad & A^T \lambda - Qx + s = c, \\ & x, s \geq 0 \end{aligned}$$

where $Q \in \mathbb{R}^{n \times n}$, $Q \succeq 0$, and $c \in \mathbb{R}^n$ constitute the objective function, $A \in \mathbb{R}^{m \times n}$ is the constraint matrix, $n > m$, $b \in \mathbb{R}^m$ is the constant term vector, and $x \in \mathbb{R}^n$ is the unknown.

In the NS case: $Q \in {}^*\mathbb{R}^{n \times n}$, and $c \in {}^*\mathbb{R}^n$

Reviewing quadratic programming

Solving algorithm: Interior Point Method

First order conditions

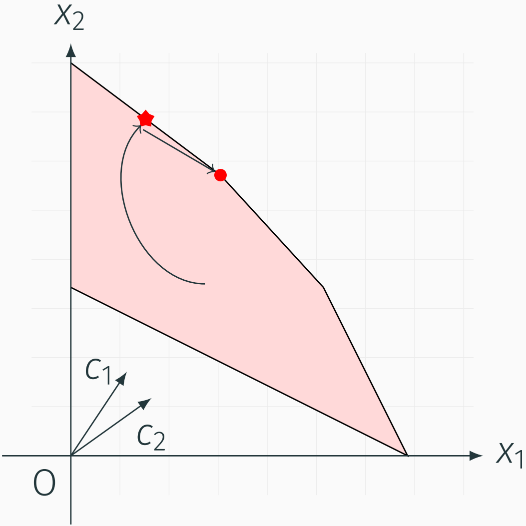
$$\begin{bmatrix} -Q & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} x \\ \lambda \\ s \end{bmatrix} = \begin{bmatrix} c \\ b \\ 0 \end{bmatrix}$$

Iterative scheme

$$\begin{bmatrix} -Q & A^T & I \\ A & 0 & 0 \\ S & 0 & X \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta \lambda \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r_c \\ -r_b \\ \sigma\mu\mathbf{1} - XS\mathbf{1} \end{bmatrix}$$

In the NS case: (full) NS Interior Point Method

NS-IPM at work



NS-IPM at work

iter	$\mu \in \mathbb{R}$	$x \in \mathbb{R}^3$	$f(x) \in {}^*\mathbb{R}$
0	0.53	$[1.46 \quad 1.46 \quad 1.46]$	$-4.38 - 3.64\eta - 6.08\eta^2$
1	0.21	$[1.32 \quad 1.32 \quad 0.74]$	$-3.37 - 10.27\eta - 5.08\eta^2$
2	0.02	$[1.30 \quad 1.30 \quad 0.40]$	$-3.01 - 11.83\eta - 4.44\eta^2$
3	$1.60e\text{-}4$	$[1.30 \quad 1.30 \quad 0.40]$	$-3.00 - 11.84\eta - 4.44\eta^2$
4	$1.60e\text{-}6$	$[1.30 \quad 1.30 \quad 0.40]$	$-3.00 - 11.84\eta - 4.44\eta^2$
5	**1.61e-8**	$\mathbf{[1.30 \quad 1.30 \quad 0.40]}$	$\boldsymbol{-3.00 - 11.84\eta - 4.44\eta^2}$
6	$\boldsymbol{0.06\eta}$	$\mathbf{[1.38 \quad 1.38 \quad 0.25]}$	$\boldsymbol{-3.00 - 12.13\eta - 3.92\eta^2}$
7	$2.21e\text{-}3\eta$	$[1.41 \quad 1.41 \quad 0.17]$	$-3.00 - 12.17\eta - 3.66\eta^2$
8	$2.46e\text{-}5\eta$	$[1.42 \quad 1.42 \quad 0.17]$	$-3.00 - 12.17\eta - 3.64\eta^2$
9	$2.48e\text{-}7\eta$	$[1.42 \quad 1.42 \quad 0.17]$	$-3.00 - 12.17\eta - 3.64\eta^2$
10	$\boldsymbol{1.62e\text{-}9\eta}$	$\mathbf{[1.42 \quad 1.42 \quad 0.17]}$	$\boldsymbol{-3.00 - 12.17\eta - 3.64\eta^2}$
11	$\boldsymbol{0.14\eta^2}$	$\mathbf{[1.54 \quad 1.29 \quad 0.17]}$	$\boldsymbol{-3.00 - 12.17\eta - 3.82\eta^2}$
12	$0.01\eta^2$	$[1.65 \quad 1.19 \quad 0.17]$	$-3.00 - 12.17\eta - 3.89\eta^2$
13	$1.63e\text{-}4\eta^2$	$[1.67 \quad 1.17 \quad 0.17]$	$-3.00 - 12.17\eta - 3.89\eta^2$
14	$1.78e\text{-}6\eta^2$	$[1.67 \quad 1.17 \quad 0.17]$	$-3.00 - 12.17\eta - 3.89\eta^2$
15	$1.59e\text{-}8\eta^2$	$[1.67 \quad 1.17 \quad 0.17]$	$-3.00 - 12.17\eta - 3.89\eta^2$

Table 1: NS-IPM iterations to solve a 3-objective program.

Application to reinforcement learning

Reviewing reinforcement learning

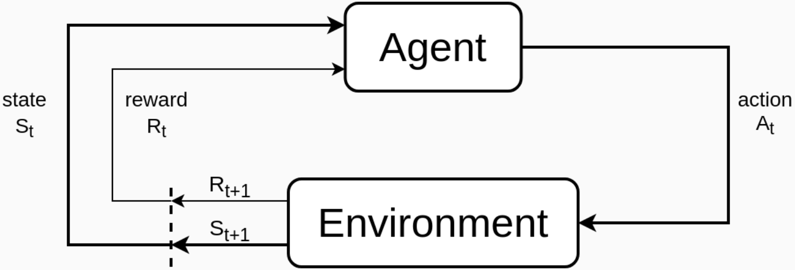


Figure 1: Schema of an RL problem: the agent takes action in the environment considering the current state; the environment reacts by changing its state and giving a reward as feedback.

Multi-objective reinforcement learning

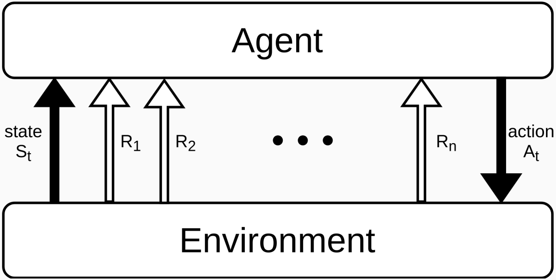
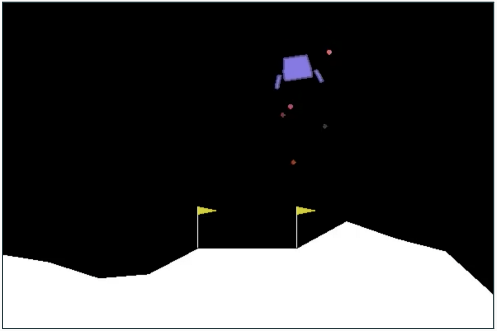


Figure 2: Schema of a Multi-Objective RL problem: after each action is taken, the environment returns an *n*-tuple of rewards.

- Very few approaches
- Single-policy vs Multiple-policy

- Tabular approaches
- Only scalarization allows for deep

The Lunar Lander environment



8-dimensional state space

- horiz/vert coordinates
- horiz/vert acceleration
- rotation angle
- angular velocity
- legs touching the ground

4-dimensional action space

- right engine
- left engine
- main engine
- do nothing

The Lunar Lander environment

5-dimensional reward

1. Distance from the pad
2. Module of the velocity
3. Body rotation angle
4. Contact with the ground
5. Fuel consumption

Standard approach

- Weighted scalarization
- Optimality 200 points avg reward in last 100 episodes
- Trial&Error weights tuning

Latent priority structure

1. Controlled flight: 1-3
2. Correct landing: 4
3. Efficient trajectory: 5

$$r = \mathit{fly} + \mathit{lan} \cdot \eta + \mathit{eff} \cdot \eta^2$$

- Complex policy gradient
- Rarely agent learns how to land
- Never reported the number of correct landings

Non-standard DQN

Implementation of the first NS DNN

- Integration of the BANs library with the Julia library *Flux*
- Custom rules for NS gradient calculation (*ChainRules*)
- Three types of NS-DQNs (fully connected)
  - Naive
  - Gradient-Clipping
  - Hybrid

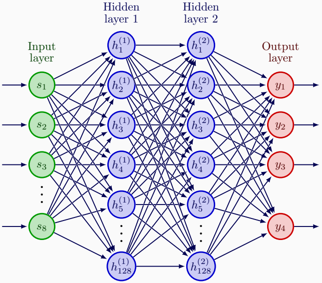


Figure 3: DQN for lunar lander

Results for lunar lander

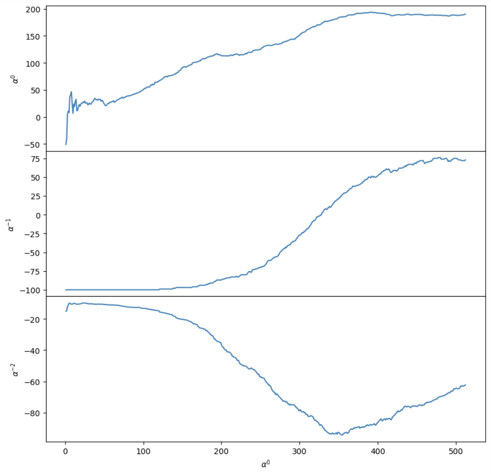


Figure 4: Average reward over 100 episodes obtained by a GC-NS-DQL agent during successful training on the Lunar Lander environment.

Results for lunar lander

Agent	Param	Avg Training Episodes	Landings (%)	StdDev	Pad Landings (%)	StdDev
Standard	$\wp_1$	532.3	75.0	26.057	68.8	26.894
Standard	$\wp_2$	788.5	66.9	19.121	56.8	24.195
GC-NS-DQL	$\wp_1$	598.8	79.7	13.787	73.8	12.689
GC-NS-DQL	$\wp_2$	659.3	87.5	5.146	83.7	6.532
GC-H-NS-DQL	$\wp_1$	616.7	84.0	17.365	78.0	18.342
GC-H-NS-DQL	$\wp_2$	664.2	77.8	12.304	70.8	14.622

Table 2: Agents performance comparison on Lunar Lander environment (in green the best performing agent, in red the worse one).

Algorithm	Param.	Avg. time per step (ms)	Exp. overall time (h)
Standard	$\wp_1$	3.73	0.55
GC-NS-DQL	$\wp_2$	19.58	3.59
GC-H-NS-DQL	$\wp_1$	14.47	2.48

Table 3: Average time, expressed in milliseconds, required for each training step of the agents.

A short resume

A short resume

- Introduced NSA reference framework
- Proposed the BAN encoding for NS numbers
- Implemented the BAN Julia library
- Discussed two engineering applications
- Numerical validation of the study