Neural Systems with Numerically-Matched Input-Output Statistic: Variate Generation

Simone Fiori

Affiliation:
Faculty of Engineering, University of Perugia
Polo Didattico e Scientifico del Ternano
Loc. Pentima bassa, 21, I-05100 Terni (Italy)

E-mail: fiori@unipg.it

Pages: 31. **Figures and Tables:** 15. **References:** 15.

Manuscript accepted for publication on:
Neural Processing Letters (NEPL)

October 12, 2005

Neural Systems with Numerically-Matched Input-Output Statistic: Variate Generation

Simone Fiori

Abstract

The aim of this paper is to present a neural system trained to exhibit matched input-output statistic for random samples generation. The learning procedure is based on a cardinal equation from statistics that suggests how to warp an available samples set of known probability density function into a samples set with desired probability distribution. The warping structure is realized by a fully-tunable neural system implemented as a *look-up table*. Learnability theorems are proven and discussed and the numerical features of the proposed methods are illustrated through computer-based experiments.

Keywords: Probability density function; pseudo-random samples generation; look-up-table based neural systems; fixed-point iteration.

1 Introduction

Standard programming environments are endowed with basic pseudo-random signal generators such as the uniform and the Gaussian ones. The uniform random variable is the prototype for the compact-support distributions, while the Gaussian random variable is the prototype for the unlimited-support distributions. Pseudo-random signal generators are useful for testing and simulation in signal processing, but usually distributions more complex than uniform/Gaussian are of use.

On the basis of the available variate generators, the two main methods known in the literature to obtain samples batches endowed with arbitrary distributions are the *transformation method* and the *rejection method*:

- **Transformation method**: A non-linear function transforms the values of a random variable, distributed in a known way, into values distributed in a different way. By properly selecting the said non-linearity, it is thus possible to generate random samples drawn from the wanted distribution.
- **Rejection method**: It is based on the generation of a bidimensional uniform distribution in the plane of the wanted probability density function.

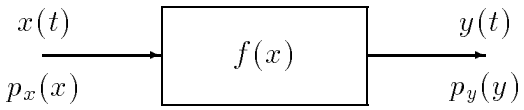

Figure 1: The considered neural system and relevant related quantities.

This may be achieved through the help of a suitably-chosen auxiliary probability density function, termed *comparison function*, that the transformation method may be easily applied to. Then the generated pair-samples whose coordinates lie in between the two curves are rejected, otherwise they are accepted as valid samples. The ratio of rejected/accepted samples depends on the choice of the comparison function.

In the present paper, we focus on the transformation method, which may be well-implemented through a tunable neural system, because the availability of a random signal source (of known statistics) and of a tunable non-linear system, along with a proper learning procedure, would allow obtaining a wide class of pseudo-random signal generators. We focus on the transformation method even because it provides at the same time a more general tool for *statistical modeling* [10, 11], which, however, exceeds the scope of the present contribution.

A neural system is able to implement a non-linear tunable input-output transference on the basis of the features of incoming signals and on external cues. A well-known effect of non-linear neural systems is to warp the statistical distribution of the input signal. In particular, we suppose that the system under consideration has the structure shown in the Figure 1, where $f(x)$ denotes the neural system transference function and $x(t) \in \mathcal{X} \subseteq \mathbb{R}$ denotes the stationary input signal, having probability density function $p_x(x)$, and $y(t) \in \mathcal{Y} \subseteq \mathbb{R}$ denotes the output signal, having probability density function $p_y(y)$. In the hypothesis that the system transference is strictly monotonic, for instance $f'(x) > 0$, $\forall x \in \mathcal{X}$, the relationship between the input distribution, the output distribution and the system transfer function is known to be [13]:

$$f'(x)p_y(f(x)) = p_x(x) \ , \tag{1}$$

which essentially expresses the invariance of probability under measure changes.

Usually, equation (1) is interpreted as an analysis formula, which allows for computing the output distribution when the input distribution and the system transference function are known. However, the cardinal equation (1) may also

be interpreted as a formula that allows designing the non-linear system when $p_x(x)$ is known and it is desired that the system responds according to the pre-fixed distribution $p_y(y)$. In general, this design operation is rather difficult, because the equation (1) in the unknown $f(x)$ involves the solution of a non-linear differential equation, provided that consistent initial conditions are specified.

Here we present an effective numerical representation of the involved quantities, based on the 'look-up table'-like implementation of neural activation functions as well as an efficient numerical algorithm to solve the cardinal equation, and evaluate through experiments the effectiveness of the proposed approach.

It deserves to note that the proposed approach, based on a non-linear adaptive system, is closely related to adaptive-activation function neurons (for a recent review of the theory and applications see e.g. [5, 6, 7]). There are two important distinctions to consider, however, between the approach proposed here and the mentioned theory: First, the adaptive-activation-function neurons were developed in order to approximate/estimate probability density functions only. Second, here we consider systems that learn in a supervised way, in fact both input and target distributions are known, while adaptive activation function neurons were devoted to unsupervised learning for blind signal processing applications. As a side note, we underline that the classical theory of neural networks prescribes the transfer function of the neurons to be bounded and saturating: As it will be apparent within this paper, there is no reason whatsoever to consider this condition of use in the present context and, as a matter of fact, the theoretical neural transfer functions in some cases will result necessarily unbounded.

2 Formal and Numerical Solutions of the Cardinal Design Equation

The present section formalizes the learning problem at hand, states two learnability results and illustrates a fixed-point-based numerical algorithm for the solution of the learning cardinal equation.

As a technical hypothesis, we may suppose that both the distributions $p_x(x)$ and $p_y(y)$ are even functions (that is the case, e.g., for the Gaussian or the double-sided Laplacian distributions). Alternatively, we may hypothesize that

both $p_x(x)$ and $p_y(y)$ are single-sided, e.g., different from zero only for $x > 0$ and $y > 0$ (this is the case, e.g., for the one-side Laplacian and for the Rayleigh distribution) and therefore "completely skewed to the right". In both cases, the non-linearity $f(x)$ may be required to have a zero in the origin, namely that $f(0) = 0$. In fact, it is readily seen from equation (1) that if $p_x(x)$ is an even function the product $f'(x)p_y(f(x))$ should not change its sign when x changes its sign; that may happen only if $f(x)$ is an even function (which implies $f'(x)$ to be an odd function). In the other case, if $p_y(y) = 0$ for $y < 0$ and $p_x(x) = 0$ for $x < 0$, in order to obtain a bounded, smooth monotonic non-linear system, it is sufficient that $f(x) = 0$ for $x = 0$.

These are not very restrictive hypotheses: If the probability density functions that are really dealt with are not centered, e.g. they are symmetric with respect to a central value different from zero, it is sufficient to generate a zero-mean random variable because it is easily seen that changing the mean value of the output results in a lifting of the system transference, that means replacing $f(x)$ with $f(x) + \mu$, with the constant μ chosen as the wanted non-zero mean value.

With this hypothesis on the non-linear neural transfer function, the cardinal design equation (1) particularizes into:

$$f'(x) = \frac{p_x(x)}{p_y(f(x))} \ , \ f(0) = 0 \ . \tag{2}$$

It is important to note that this equation is well-posed, in that as the probability density functions are non-negative then $f'(x)$ stays non-negative as required in order for the transference of the neural system to be monotonically increasing[1].

As mentioned, in general a closed-form solution to equation (2) may not be envisaged, thus we should resort to an iterative learning algorithm to search for a solution. Formally, this means designing an algorithm that generates a succession of functions $f_n(x)$, $n \in \mathbb{N}$, whose limit coincides to the solution of (2).

A way to generate such succession is to employ a fixed-point algorithm. It consists in hypothesizing an initial candidate solution $f_0(x)$ to start the iteration; replacing the putative solution $f_n(x)$ into the right-hand side of equation (2) provides the putative derivative $f'_{n+1}(x)$, whose integration yields the new

[1] With a slight modification of the equations, it is easy to cast the problem for a neural system endowed with monotonically decreasing transference. Such modification, which might be profitable in neural statistical modeling, is not necessary in the economy of the present paper and will not be considered hereafter.

attempted solution $f_{n+1}(x)$ (the boundary condition is to be employed in the integration). Repeating the process a sufficiently large number of times leads to more refined solutions of the cardinal differential equation.

Formally, the fixed-point algorithm writes:

$$\begin{cases} f_0(x) = \text{Initial guess} \ , \\ f'_{n+1}(x) = p_x(x)/p_y(f_n(x)) \ , \ n \geq 0 \ , \\ f_{n+1}(x) = \int_0^x f'_{n+1}(\xi) d\xi \ . \ (\text{This implies } f(0) = 0.) \end{cases} \tag{3}$$

As a figure-of-convergence of learning progress it is useful to consider the difference between two successive solutions, namely:

$$\Delta f_n \stackrel{\text{def}}{=} \| f_n - f_{n-1} \| \ , \ n = 1, \ 2, \ 3, \ ... \tag{4}$$

where $\| \cdot \|$ denotes some functional norm. Normally, we are not interested in an approximation of the non-linearity $f(x)$ in the whole range $\mathcal{X}$, especially when the distributions of interest possess unbounded supports, so it is acceptable to let the variable x range in a subset $\bar{\mathcal{X}} \subset \mathcal{X}$. In this spirit, a sensible choice for the norm in (4) is the following p_x-weighted L_1 norm:

$$\| h_1 - h_2 \| \stackrel{\text{def}}{=} \int_{\bar{\mathcal{X}}} |h_1(x) - h_2(x)| p_x(x) dx \ , \tag{5}$$

for absolutely integrable functions h_1, h_2 on $\bar{\mathcal{X}}$.

2.1 On solution learnability and on convergence of the fixed-point learning rule

The fixed-point method described by the algorithm (3) provides a way to solve the Cauchy problem:

$$f'(x) = g(x, f(x)) \ , \ f(0) = 0 \ , \ x \in \mathcal{X} \ , \tag{6}$$

where the function $g(\cdot, \cdot)$ describes the learning task and the solution $f(x)$ describes the non-linear neural transference function. In the present case, the Cauchy kernel $g(\cdot, \cdot)$ has the special structure:

$$g(x, y) \stackrel{\text{def}}{=} p_x(x) q_y(y), \ \text{with } q_y(y) \stackrel{\text{def}}{=} \frac{1}{p_y(y)} \ . \tag{7}$$

About the existence (and possibly uniqueness) of a solution of a Cauchy problem and about the convergence of the iterative algorithm (3) for solving the initial-value problem (6), some general-purpose results are available from calculus, as

the Cauchy-Lipschitz and Peano-Picard theorems [4]. However, due to its broad scope, the standard theorems might restrict the applicability of the iteration (3) to a very tight interval $\bar{\mathcal{X}}$, so that, from a practical point of view, it would seem to be impossible to obtain the variate $y(t)$ in a sufficiently large range.

In order to overcome this practical limitation, in what follows we prove two learnability results that are tailored to the problem at hand. It deserves to mention that, as we have granted the technical hypothesis that all the distributions of interest in the present paper are either symmetrical around the origin or one-sided, the claims and the proofs of the following results are stated for intervals of interest of the form $[0, \bar{x}]$, which clearly encompass both cases properly.

Theorem 1 (Existence and uniqueness of a solution.) *Let us consider the Cauchy problem:*

$$f'(x) = p_x(x)/p_y(f(x)) \ , \ f(0) = 0 \ , \ x \in [0, \overline{x}] \ , \tag{8}$$

where $p_x(x)$ and $p_y(y)$ are bounded probability density functions over $\mathbb{R}$, and let the following quantities be defined:

$$P_x(x) \stackrel{\text{def}}{=} \int_0^x p_x(t)dt \ , \ x \in [0, \overline{x}] \ , \ \overline{y} \stackrel{\text{def}}{=} P_x(\overline{x}) \ . \tag{9}$$

If $p_x(x)$ is continuous in $[0, \overline{x}]$ and $p_y(y)$ is continuous and non-null in $[0, \overline{y}]$, then the problem (8) has a unique continuous solution $f(x)$ in $[0, \overline{x}]$.

Proof. The differential equation (8) has separable variables. Let us define the function:

$$P_y(y) \stackrel{\text{def}}{=} \int_0^y p_y(t)dt \ , \tag{10}$$

with $y \in [0, \overline{y}]$. As $f'(x)p_y(f(x)) = \frac{d}{dx}[P_y(f(x)) + \kappa_1]$, where $\kappa_1 \in \mathbb{R}$ is an arbitrary constant, the following equivalences hold in $[0, \overline{x}]$:

$$\frac{d}{dx}[P_y(f(x))+\kappa_1] = p_x(x) \ \Leftrightarrow \ P_y(f(x)) = P_x(x)+\kappa \ \Leftrightarrow \ f(x) = P_y^{-1}(P_x(x)+\kappa) \ ,$$

where $\kappa \in \mathbb{R}$ denotes an arbitrary constant of integration and $P_y^{-1}(\cdot)$ denotes the inverse of function $P_y(\cdot)$. Such inverse exists and is unique because its derivative differs from zero everywhere in the interval of interest by hypothesis. By the boundary condition we gather that the unique solution of the differential equation (8) is $f(x) = P_y^{-1}(P_x(x))$. As $P_x(x)$ is continuous in $[0, \overline{x}]$ and $P_y^{-1}(y)$ is continuous in $[0, \overline{y}]$, then $f(x)$ is continuous in $[0, \overline{x}]$. □

It is worth noting that, by definition of $P_x(\cdot)$ as the integral of a probability density function over the real line, it surely holds that $P_x(x) \leq \frac{1}{2}$, if $p_x(x)$ is symmetric, and $P_x(x) \leq 1$, if $p_x(x) \neq 0$ only for $x \geq 0$, for all $x \in [0, \overline{x}]$.

Theorem 2 (Properties of the learning iteration.) *Let us consider the functional iteration for the Cauchy problem (6)+(7):*

$$\begin{cases} f_0(x) = 0 \ , \\ f_n'(x) = p_x(x) q_y(f_{n-1}(x)) \ , \ f_n(0) = 0 \\ x \in [0, \overline{x}] \ , \ n = 1, \ 2, \ ... \end{cases} \tag{11}$$

Let the quantities $f(\cdot)$ and $\overline{y}$ be defined as in Theorem 1. Under the hypotheses:

- *the probability density function $p_x(x)$ is continuous and strictly positive in $[0, \overline{x}]$,*
- *the reciprocal $q_y(y)$ of a probability density function is continuous, strictly positive and monotonically increasing in $[0, \overline{y}]$,*
- *the function $q_y(y)$ is Lipschitz in the interval $[0, \overline{y}]$ with Lipschitz constant $L \in (0, +\infty)$, namely:*

$$\forall y_1, y_2 \in [0, \overline{y}] : |q_y(y_1) - q_y(y_2)| \leq L|y_1 - y_2| \ ,$$

then the following properties prove to hold:

1. *every function $f_n(x)$ is continuous and monotonically increasing $\forall n \geq 1$,*
2. *all the functions $f_n(x)$ are bounded by the solution $f(x)$ of the Cauchy problem (6)+(7), namely $f_n(x) < f(x)$, $\forall n \geq 0$, $\forall x \in (0, \overline{x}]$,*
3. *the solutions are monotonic with n, namely $f_n(x) > f_{n-1}(x)$, $\forall n \geq 1$, $\forall x \in (0, \overline{x}]$,*
4. *the succession $f_n(x)$ converges uniformly to $f(x)$, $\forall x \in [0, \overline{x}]$.*

Proof. Under the considered hypotheses, Theorem 1 holds and we may make use of solution $f(x)$ of the Cauchy problem (6)+(7) found therein as well as of function $P_x(\cdot)$ defined therein.

The claim 1) may be proven by induction. The function $f_1(x)$ satisfies the claim, in fact, by the iteration equations (11) we get:

$$f_1(x) = \int_0^x p_x(t) q_y(f_0(t)) dt = q_y(0) P_x(x) \ , \tag{12}$$

which, by definition of $P_x(x)$, is continuous and monotonically increasing. For $n > 1$, let us suppose function $f_{n-1}(x)$ satisfies the claim 1), then we have:

$$f_n(x) = \int_0^x p_x(t) q_y(f_{n-1}(t)) dt \ ,$$

which is continuous and monotonically increasing like the integrand.

In order to prove the claim 2), it is worth observing that the exact solution $f(x)$ satisfies:

$$f(x) = \int_0^x p_x(t) q_y(f(t)) dt \ . \tag{13}$$

For $n = 0$ the claim 2) clearly holds. For $n \geq 1$, let us hypothesize that $f_{n-1}(t) - f(t) < 0$, $\forall x \in (0, \overline{x}]$. Then we have:

$$\begin{aligned} f_n(x) - f(x) &= \int_0^x p_x(t) q_y(f_{n-1}(t)) dt - \int_0^x p_x(t) q_y(f(t)) dt \\ &= \int_0^x p_x(t) [q_y(f_{n-1}(t)) - q_y(f(t))] dt < 0 \ , \end{aligned}$$

because the function $q_y(\cdot)$ is monotonically increasing by hypothesis. The claim is thus proven by induction. This key result ensures that the graph of each function $f_n(x)$ is contained in the compact set $[0, \overline{x}] \times [0, \overline{y}]$. It is worth noting that the inequality does not hold only for $x = 0$ just because $f_n(0) = f(0) = 0$, $\forall n \geq 0$.

The claim 3) may be proven in a similar way by induction. First, let us observe that $f_1(x) > f_0(x)$, $\forall x \in (0, \overline{x}]$. It is now necessary to show that $f_n(x) > f_{n-1}(x)$ implies $f_{n+1}(x) > f_n(x)$, $\forall n \geq 1$. This may be accomplished by noting that:

$$\begin{aligned} f_{n+1}(x) - f_n(x) &= \int_0^x p_x(t) q_y(f_n(t)) dt - \int_0^x p_x(t) q_y(f_{n-1}(t)) dt \\ &= \int_0^x p_x(t) [q_y(f_n(t)) - q_y(f_{n-1}(t))] dt > 0 \ , \end{aligned}$$

again thanks to the monotonicity of the function $q_y(\cdot)$. Even in this case it is worth noting that the inequality does not hold only for $x = 0$.

The proof of the claim 4) is equivalent to proving that the series:

$$S_n(x) \stackrel{\text{def}}{=} [f_1(x) - f_0(x)] + [f_2(x) - f_1(x)] + \ldots + [f_n(x) - f_{n-1}(x)] \ , \tag{14}$$

converges uniformly in $x \in (0, \overline{x}]$ as $n \to +\infty$. The key step is to prove that the following inequality holds:

$$f_n(x) - f_{n-1}(x) \leq \frac{q_y(0) L^{n-1} P_x^n(x)}{n!} \ , \forall x \in [0, \overline{x}] \ . \tag{15}$$

(It is worth remarking that this result is non-trivial because, from claim 3), we know that $f_n(x) - f_{n-1}(x) > 0$ everywhere but for the origin.) This result may be shown by induction. For $n = 1$ we know from equation (12) that the property (15) holds true (with equality sign). For $n > 1$, we suppose that:

$$f_{n-1}(x) - f_{n-2}(x) \leq \frac{q_y(0)L^{n-2}P_x^{n-1}(x)}{(n-1)!} \ , \forall x \in [0, \overline{x}] \tag{16}$$

and show that inequality (16) implies inequality (15). From iteration formulas (11) we get:

$$\begin{aligned} f_n(x) - f_{n-1}(x) &= \int_0^x p_x(t)[q_y(f_{n-1}(t)) - q_y(f_{n-2}(t))]dt \\ &\leq \int_0^x p_x(t)L[f_{n-1}(t) - f_{n-2}(t)]dt \\ &\leq L\int_0^x p_x(t)\frac{q_y(0)L^{n-2}P_x^{n-1}(t)}{(n-1)!}dt \\ &= \frac{q_y(0)L^{n-1}}{(n-1)!}\int_0^x p_x(t)P_x^{n-1}(t)dt = \frac{q_y(0)L^{n-1}P_x^n(x)}{n!} \ . \end{aligned}$$

In the above passages we have exploited the Lipschitzianity of the function $q_y(\cdot)$, the inequality (16) and the fact that $P_x(x)$ is a primitive of $p_x(x)$. By recalling that surely $P_x(x) \leq 1$, it is immediate to conclude that the series $S_n(x)$ then converges totally (hence uniformly) because it is dominated by the convergent numerical series $\sum_{n=1}^{\infty} \frac{q_y(0)L^{n-1}}{n!}$, for all $x \in [0, \overline{x}]$. □

As section closure, some comments on the conclusions of Theorem 2 are in order:

- The property 2) ensures that the fixed point operator describing the algorithm (11) insists on functions whose graph lies on the rectangle $[0, \overline{x}] \times [0, \overline{y}]$, where $\overline{y}$ is a function of $\overline{x}$. This is an important intrinsic property of the fixed-point iteration (11) without which we should have been forced to require the function $q_y(\cdot)$ to be Lipschitz over the whole real line, which might be not the case, or to progressively restrict the domain of $f_n(\cdot)$ at every iteration to, say, intervals $[0, \overline{x}_n]$, in such a way that the image $f_n([0, \overline{x}_n])$ is a sub-interval of $[0, \overline{y}]$. The latter option would have made the feasible interval $[0, \overline{x}]$ so tight that the practical profitability of the iteration scheme would have easily resulted to be compromised.

- The property 4) ensures that the fixed-point iteration converges asymptotically to the desired solution. However, it does not imply that the figure of

convergence Δf_n defined in (4) decreases *monotonically* to zero. However, we can observe that the convergence is dominated by terms like $\frac{L^n}{2^n n!}$ or $\frac{L^n}{n!}$ which, for n sufficiently large, become surely smaller than 1.

- We remark that if the interval $[0, \overline{x}]$ is chosen such that $p_x(\cdot)$ is not continuous nor different from zero therein, then the candidate solutions $f_n(\cdot)$ may loose some properties such as monotonicity. However, these properties still hold in the largest subinterval of $[0, \overline{x}]$ where $p_x(\cdot)$ meets the conditions of continuity and positivity.

2.2 Numerical implementation of the learning procedure

From an implementation viewpoint, the algorithm (3) is a functional fixed-point iteration and a function representation needs thus to be defined in order to make the algorithm be numerically realizable on a computer.

In the present contribution, we choose to represent $f_n(x)$ by a numerical vector: In practice, we suppose there exists a compact sub-interval $\bar{\mathcal{X}} = [\underline{x}, \overline{x}] \subset \mathcal{X}$ of particular interest and we also suppose this interval is partitioned into $N - 1 \geq 1$ discrete bins. This gives rise to the vector-type representation $\mathbf{x} \in \mathbb{R}^N$ for the domain/support of the input sequence probability density function, where $\mathbf{x}$ contains N values regularly spaced in $\bar{\mathcal{X}}$ with spacing-width denoted as Δ_x. Then $f_n(x)$ may be represented by a numerical vector $\mathbf{f}_n \in \mathbb{R}^N$ and the neural input-output transference is now represented by the discrete relationship $(\mathbf{x}, \mathbf{f}) \in \mathbb{R}^{2N}$, namely, a numerical *look-up table*. The entries of a vector/table $\mathbf{f}_n$ may be denoted by a further footer, i.e., by $f_{n,k}$.

In order to translate the algorithm (3) into a version suitable to numerical representation, we should consider the inherent limitations of numerical integration of differential equations, as the fact that the term $p_x(x)/p_y(f_n(x))$ does not compute properly when $p_y(y)$ has a quasi-null value (that is, a value below the machine precision) and the fact that by arbitrarily fixing a $\bar{\mathcal{X}} \neq \mathcal{X}$, we intrinsically make the learnt non-linearity a (possibly coarse) approximation of the true one. According to these considerations, the following remedies may be considered:

- **Denominator shifting.** In order to get rid of division-by-zero-type errors, we choose to add a small constant γ to the denominator of the fraction $p_x(x)/p_y(f_n(x))$ that prevents the denominator from being too

close to zero. The effects due to the presence of such constant should be carefully taken into account in the numerical experiments.

- **One-moment matching.** In order to mitigate the consequences of the finite-domain-approximation assumption, we may incorporate in the adapting algorithm some information that is easily accessible. For instance, as a simple information that is easily profitable, we suggest to consider the knowledge of output-signal moments of order m:

$$\mu_y^{(m)} \stackrel{\text{def}}{=} \int_{\mathcal{Y}} y^m p_y(y) dy = \int_{\mathcal{X}} f^m(x) p_x(x) dx \ . \tag{17}$$

  Note that the quantity $\mu_y^{(m)}$ depends only on $p_y(y)$, therefore it is to be considered a known quantity.

- **Range scaling.** When the output distribution has finite support, denoted by the compact interval $\bar{\mathcal{Y}} = [\underline{y}, \overline{y}]$, it might be advantageous to employ a more strict normalization, that linearly scales the entries of the putative solution $\mathbf{f}_n$ so that $f_{n,1} = \underline{y}$ and $f_{n,N} = \overline{y}$.

In practice, the considered numerical, regularized version of (3) reads:

$$\begin{cases} (\mathbf{A0}) & \mathbf{f}_0 = \mathbf{0} \ , \\ (\mathbf{A1}) & \mathbf{f}'_{n+1} = \frac{\mathbf{p}_x}{p_y(\mathbf{f}_n)+\gamma} \ , \\ (\mathbf{A2}) & \mathbf{f}_{n+1} = \text{cumsum}\{\mathbf{f}'_{n+1}\} \ , \end{cases} \tag{18}$$

plus the possible normalization operations:

$$\begin{cases} (\mathbf{A3}) & \mathbf{f}_{n+1} = \text{normalize}\{\mathbf{f}_{n+1}; \mu_y^{(m)}\} \ , \\ (\mathbf{A4}) & \mathbf{f}_{n+1} = \text{linscale}\{\mathbf{f}_{n+1}; \underline{y}, \overline{y}\} \ . \end{cases} \tag{19}$$

The integral has been replaced by the numerical cumulative sum ('cumsum'), and $\mathbf{p}_x$ denotes the vector of N elements containing the values of $p_x(\cdot)$ corresponding to the values in $\mathbf{x}$. The moment-matching operation has been denoted as 'normalize', while the linear scaling operator has been denoted with 'linscale'. The mentioned operators are defined as follows for a look-up table $(\mathbf{x}, \mathbf{g})$:

Let $\mathbf{x}$ denotes a generic vector with entries x_k, with limit values $\underline{x}, \overline{x}$, computed as $x_k = \underline{x} + k \cdot \Delta_x$.

Let us denote by k_0 the index such that $x_{k_0} = 0$, then:

$$\text{cumsum}(\mathbf{g})_k \stackrel{\text{def}}{=} \begin{cases} -\sum_{i=k}^{k_0} g_i \Delta_x & \text{for } 2 \le k \le k_0 \ . \\ 0 & \text{for } k = k_0 \ , \\ \sum_{i=k_0}^{k} g_i \Delta_x & \text{for } k_0 \le k \le N-1 \ . \end{cases}$$

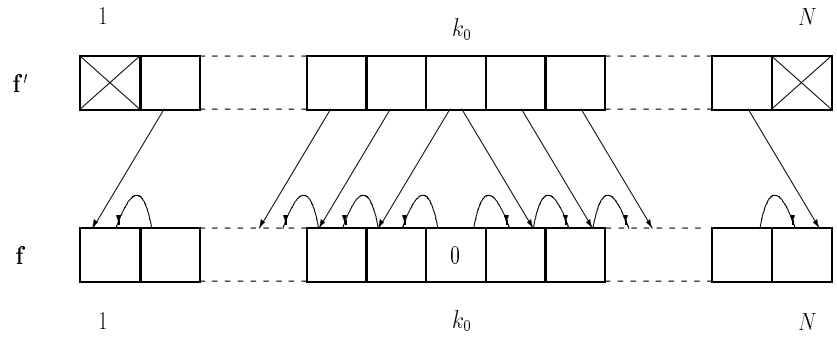

Figure 2: Visual representation of vectors $\mathbf{f}'$ and $\mathbf{f}$ and their relationship through 'cumsum' numerical-integration operator.

$$\text{Let } \hat{\mu}_y^{(m)}(\mathbf{g}) \stackrel{\text{def}}{=} \sum_{k=1}^{N} g_k^m p_{xk} \Delta_x, \text{ then:}$$

$$\text{normalize}\{\mathbf{g}; \mu_y^{(m)}\}_k \stackrel{\text{def}}{=} g_k \sqrt[m]{\frac{\mu_y^{(m)}}{\hat{\mu}_y^{(m)}(\mathbf{g})}} \ ,$$

$$\text{linscale}\{\mathbf{g}; a, b\}_k \stackrel{\text{def}}{=} a + \frac{(g_k - \min\{\mathbf{g}\})(b-a)}{\max\{\mathbf{g}\} - \min\{\mathbf{g}\}} \ .$$

The understating of the numerical integration leading to 'cumsum' operator is facilitated by the Figure 2, from which it is readily seen that the first and last elements of vector $\mathbf{f}'$ are lost in the integration process.

The interval Δ_x relates to the integer N and may be defined in two different ways:

1. If we want to arbitrarily fix N, then $\Delta_x = (\overline{x} - \underline{x})/N$.

2. If we want to arbitrarily fix Δ_x, then $N = \lfloor (\overline{x} - \underline{x})/\Delta_x \rfloor$, where the operator $\lfloor \cdot \rfloor$ ('floor') rounds the argument to the nearest integer towards minus infinity. This choice has the drawback that the values in x may always result smaller than $\overline{x}$.

In the present manuscript, we choose the first option, which allows us to directly select the number N of necessary points of the numerical model $\mathbf{f}$. It is worth remarking that the previous parameters should be selected in order to ensure that there exist a zero element of $\mathbf{x}$.

3 Application to Pseudo-Random Variate Generation with Desired Statistic

Randomness, random numbers and random signals are currently used for a variety of purposes such as games, for the generation of cryptographic keys and for some classes of scientific experiments. Randomness may be introduced into artificial systems in the form of pseudo-random numbers/signals generators. The following papers survey many of the basic concepts and results of random number generation: [3, 9, 12, 15].

In the following, we consider the generation of random samples of prescribed probability density function on the basis of two basic random generators having uniform or Gaussian distributions. A uniform distribution within the interval $[a, b]$ is denoted by $\mathcal{U}_{[a,b]}(u)$, $u \in \mathbb{R}$, while a zero-mean Gaussian distribution with variance σ is hereafter denoted by $G_\sigma(u) \stackrel{\text{def}}{=} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{u^2}{2\sigma^2}\right)$, $u \in \mathbb{R}$.

The numerical results arising from neural system training given below are compared to the closed-form expression of the exact generators, whenever this is possible.

3.1 Closed-form solution of some special design cases

Hand-by-hand integration of the cardinal equation, with the proper integration limits, yield the closed-form solution of the non-linear system design problem given by Theorem 1. Of course, such closed-form solution is profitable only in the case that the integrals and the necessary functional inverse functions can be computed in closed form.

In the following, four cases are illustrated in which the closed-form solution may be accessed.

Gaussian from Gaussian. With this toy example, we aim at verifying an intuitive result: If $p_y(y)$ and $p_x(x)$ are both Gaussian with variances σ_y and σ_x, respectively, the warping system is a simple scaler. This comes directly from the Cauchy problem (2), where $p_x(x) = G_{\sigma_x}(x)$ and $p_y(y) = G_{\sigma_y}(y)$. The Cauchy problem, in this case, writes:

$$f'(x) = \frac{\sigma_y}{\sigma_x} \exp\left[\frac{1}{2}\left(\frac{f^2(x)}{\sigma_y^2} - \frac{x^2}{\sigma_x^2}\right)\right] \ , \ f(0) = 0 \ .$$

Such initial-value problem admits the solution $f(x) = \frac{\sigma_y}{\sigma_x} x$, that confirms the informal intuition.

Gaussian from Uniform. Let us suppose $p_x(x) = \mathcal{U}_{[-A,+A]}(x)$ and $p_y(y) = G_\sigma(y)$. In this case $\mathcal{Y} = \mathcal{X} = \mathbb{R}$ and the cardinal equation casts into:

$$\frac{1}{2}\text{erf}\left[\frac{f(x)}{\sqrt{2}\sigma}\right] = \begin{cases} -\frac{1}{2}, & x < -A \ , \\ \frac{x}{2A}, & |x| \leq A \ , \\ \frac{1}{2}, & x > A \ , \end{cases}$$

In this case, the solution may be found in explicit form, namely:

$$f(x) = \sqrt{2}\sigma\text{erf}^{-1}\left(\frac{x}{A}\right) \ , \ |x| \leq A \ , \tag{20}$$

while for $|x| > A$ the function is undefined; in the above formula $\text{erf}^{-1}(\cdot)$ denotes the inverse error function.

Laplacian from Gaussian. A double-sided Laplacian distribution is defined by:

$$S_\lambda(u) \stackrel{\text{def}}{=} \frac{\lambda}{2}e^{-\lambda|u|} \ , \tag{21}$$

where $\lambda > 0$ is the Laplacian dispersion parameter. Let us suppose $p_x(x) = G_\sigma(x)$ and $p_y(y) = S_\lambda(y)$. In this case $\mathcal{Y} = \mathcal{X} = \mathbb{R}$ and the solution may be written explicitly as:

$$f(x) = \begin{cases} \frac{1}{\lambda}\ln\left[1 + \text{erf}\left(\frac{x}{\sqrt{2}\sigma}\right)\right], & x \leq 0 \ , \\ -\frac{1}{\lambda}\ln\left[1 - \text{erf}\left(\frac{x}{\sqrt{2}\sigma}\right)\right], & x > 0 \ . \end{cases} \tag{22}$$

Note that functions (20) and (22) are null and continuous in the origin. It is also interesting to note that both functions are unbounded and, in particular, tend to grow indefinitely as x grows.

U-shaped from Uniform. Let us consider the generation of a bounded-support distribution on the basis of a bounded-support distribution. This is the case when x possesses a uniform distribution and y is a unitary-amplitude sinusoid sampled at random times with uniform distribution within its period. In this case, the probability density function of the sinusoid's values is described by:

$$U(u) \stackrel{\text{def}}{=} \frac{1}{\pi\sqrt{1-u^2}} \ , \tag{23}$$

which is defined for $u \in (-1, 1)$ only and exhibits a U-type shape. So, we assume $p_x(x) = \mathcal{U}_{[-A,+A]}(x)$ and $p_y(y) = U(y)$ in $\mathcal{Y} = (-1, +1)$. The solution of the cardinal design equation in this case writes:

$$f(x) = \begin{cases} -1, & x < -A \ , \\ \sin\left(\frac{\pi x}{2A}\right), & |x| \leq A \ , \\ +1, & x > A \ . \end{cases} \tag{24}$$

Unlike the previous cases considered, the found transference function is bounded.

Ricean from Uniform. To end with, it might be interesting to consider the generation of a Rice random variable [14] from a uniformly-distributed random variable. The Ricean probability density function is defined as:

$$P_{\kappa,\sigma}(u) \stackrel{\text{def}}{=} \frac{u}{\sigma^2} \exp\left(-\frac{u^2+\kappa^2}{2\sigma^2}\right) I_0\left(\frac{\kappa u}{\sigma^2}\right) \ , \tag{25}$$

where κ and σ are positive shape-parameters and $x \geq 0$. (The meaning of $I_0(u)$ is explained in the Appendix).

The Ricean distribution is inherently one-sided, thus completely skewed to the right, therefore it is worth considering the generation of a Rice random variable through a one-sided uniform variable. In the general case, however, the cardinal equation cannot be exactly solved for a Ricean distribution. In the particular case in which $\kappa = 0$, the Rice distribution reduces to the Rayleigh distribution:

$$R_\sigma(u) \stackrel{\text{def}}{=} \frac{u}{\sigma^2} \exp\left(-\frac{u^2}{2\sigma^2}\right) \ , \tag{26}$$

In this special case, assuming $p_y(y) = R_\sigma(u)$ and $p_x(x) = \mathcal{U}_{[0,A]}(x)$, the cardinal design equation reads:

$$1 - \exp\left[-\frac{f^2(x)}{2\sigma^2}\right] = \frac{x}{A} \ ,$$

for $f(x) \geq 0$. Note that for $x \notin [0, A)$, the non-linearity $f(x)$ is undefined. Within the interval $[0, A)$, the non-linearity $f(x)$ writes:

$$f(x) = \sigma\sqrt{2\log\left(\frac{A}{A-x}\right)} \ . \tag{27}$$

It deserves to note that $f(0) = 0$.

Square-root Lévy distribution. The case of square-root Lévy distribution is only briefly touched here because its tackling does not differ much from the case of a Gaussian distribution already considered in an above point. Nevertheless, its is perhaps worthwhile mentioning.

A probability density function structure that has received growing attention over the past years in the signal processing community is the one referred to as α-stable. α-stable distributions are being proposed as alternative models of signals encountered in audio environments because they can better model the intrinsic impulsiveness of the related signals. α-stable distributions are primarily defined via their characteristic functions and only a few closed-form expressions of the corresponding density function are known. Gaussian and Cauchy's distributions are special cases of so-called symmetric α-stable distributions, while the

Lévy distribution is one among the few known closed-form asymmetric α-stable distributions. With the purpose of generating multivariate jointly-α-stable distributions, in the manuscript [8] a straightforward method was proposed, which is essentially based on the generation of multivariate Gaussian distributions and scalar square-root Lévy distributions. By this, it is meant that if $z(t)$ denotes a Lévy sequence, the required sequence takes on the form $y(t) \stackrel{\text{def}}{=} \sqrt{z(t)}$. The sequence $y(t)$ is, of course, well defined because the Lévy sequence $z(t)$ is completely skewed to the right. The probability density function of the variable y reads [8]:

$$V(y) = \frac{1}{\sqrt{\pi}} y^{-2} \exp\left(-\frac{1}{4y^2}\right) \ , y > 0 \ ,$$

and is null elsewhere. If we assume $p_y(y) = V(y)$ as a random variable to be generated and manage to compute the associated cumulative distribution function $P_y(y)$, we easily find that $P_y(y) = \text{erfc}\left(\frac{1}{2y}\right)$ for $y \geq 0$ and null elsewhere, with $\text{erfc}(\cdot)$ denoting the complementary error function. As anticipated, in spite of the potential importance of the Lévy distribution in applied fields, this case does not seem worthwhile of further investigation in the present manuscript.

3.2 Random number generation after non-linearity learning

When a suitable non-linearity $f(\cdot)$ has been learnt by the neural system, it may be used to generate random samples drawn from the desired statistical distribution by employing the model f as a look-up table, as explained by the following procedure:

- First, it is necessary to produce a realization $\{x_k\}$ drawn from the available-generator distribution $p_x(x)$ (having, e.g., uniform or Gaussian probability density function) ranging in $\mathcal{X}$.

- Then, it is necessary to address the proper values in the look-up table $\mathbf{f}$ corresponding to the values of $\{x_k\}$, by computing indices/pointers:

$$r_k \stackrel{\text{def}}{=} \lfloor \frac{x_k - \underline{x}}{\overline{x} - \underline{x}} \rfloor + 1 \ . \tag{28}$$

  It is important to note that if $p_x(x)$ has an unbounded support, it is possible that a generated $x_k \notin \bar{\mathcal{X}}$: In this case, it happens that $r_k < 1$ or $r_k > N$, therefore the obtained value of the pointer r_k must be discarded. Clearly, every occurrence of the above-mentioned 'overflow' prob-

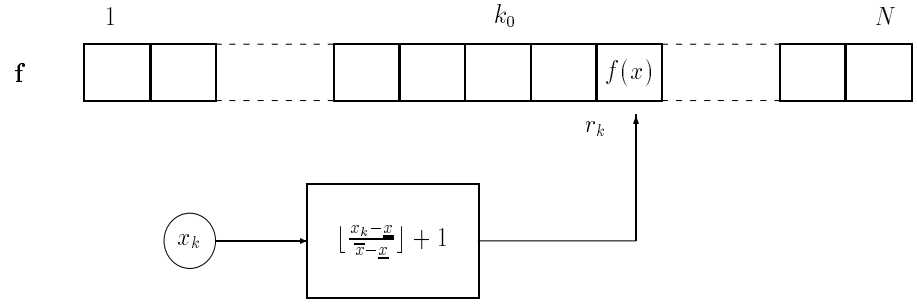

Figure 3: Usage of the numerical vector **f** through 'look-up-table' entries addressing, where the meaning of the unexplained quantities is the same as of Figure 2.

lem causes the number of available generated random samples to decrease of one unity, therefore the final number of available samples might be slightly smaller than the cardinality of $\{x_k\}$.

- To end with, the desired set $\{y_k\}$ of samples, distributed according to the probability density function $p_y(y)$, may be obtained by setting:

$$y_k = f_{r_k} \ . \tag{29}$$

The understating of the numerical usage of the 'look-up-table' **f** is facilitated by the Figure 3.

3.3 Computer-based numerical experiments

In the experiments, the index Δf_n is computed as the numerical approximation of the p_x-weighted L_1 norm (5) given by:

$$\Delta f_n \approx \sum_{k=1}^{N} |f_{n+1,k} - f_{n,k}| p_{xk} \Delta_x \ . \tag{30}$$

In the following, we consider three numerical examples in which the closed-form solution is available, so that it is easy to verify the quality of the numerical result. We further consider two numerical cases in which the closed-form solutions are not available. All the probability density functions considered in the following sections, either for the available generators and as wanted distributions, match the hypotheses made in the learnability Theorems 1 and 2 over properly-chosen intervals $[\underline{x}, \overline{x}]$.

In order to quantitatively measure the quality of the generated random variable, we may compare the moments of the generated variables – computed numerically – with the exact moments, up to order six. The empirical moments of a sample-set $\{(y_s)_{D,p_1,p_2}\}_{s=1,...,Q}$ are denoted as $\hat{\mu}_D^{(m)}(p_1,p_2)$, where $m = 1,\ 2,\ 3,\ ...$ is the moment order, D denotes the kind of distribution ('S' for Laplacian, 'U' for U-shaped, 'R' for Rayleigh, 'B' for Gamma and 'P' for Ricean), p_1 and p_2 are the parameters that the distribution (hence the moments) may depend upon and Q denotes the number of available samples in the generated distribution. The empirical moments are computed as:

$$\hat{\mu}_D^{(m)}(p_1,p_2) \stackrel{\text{def}}{=} \frac{1}{Q}\sum_{s=1}^{Q}(y_s)_{D,p_1,p_2}^m \ ,$$

The exact expressions of the moments are given in the Appendix for the distributions of interest.

3.3.1 Experiments and comparisons with closed-form solutions

The first example concerns the generation of a Laplacian distribution from a Gaussian one. In this case $\sigma_x = 0.3$, for which it is appropriate to take $\bar{\mathcal{X}} = [-1,+1]$, and $\lambda = 3$. The results provided by the algorithm (18) plus (**A3**) are shown in the Figure 4, which pertain to parameters $N = 501$, $Q = 50,000$, $\gamma = 0.001$, $m = 2$. In this case the found non-linearity is curly as required to warp a Gaussian distribution into a Laplacian one and the tuned non-linear transfer function looks in good accordance with the theoretical one. In Figure 5 we may observe the histogram estimates (with 50 bins) of the generated Gaussian data, the Laplacian output obtained with the learnt non-linearity, and, for comparison purposes, the Laplacian output obtained with the exact non-linearity. As anticipated in section 3.2, while a total of $50,000$ samples were generated from the prototype distribution $p_x(x)$, a total of $49,966$ samples were obtained for the target distribution $p_y(y)$, due to the practical limitation intrinsic in $\bar{\mathcal{X}} \neq \mathcal{X}$.

The second example concerns the generation of a U-shaped distribution from a uniform $\mathcal{U}_{[-1.5,+1.5]}$. In this case it is appropriate to take $\bar{\mathcal{X}} = [-2,+2]$ and $\bar{\mathcal{Y}} = [-0.9999,+0.9999]$. The results provided by the algorithm are shown in the Figure 6 and pertain to values $N = 501$, $Q = 50,000$, $\gamma = 0$, in the numerical algorithm (18) plus (**A4**). Note that, in this case, $p_y(y)$ is much larger than zero for every value of y, therefore there is no need to add a constant

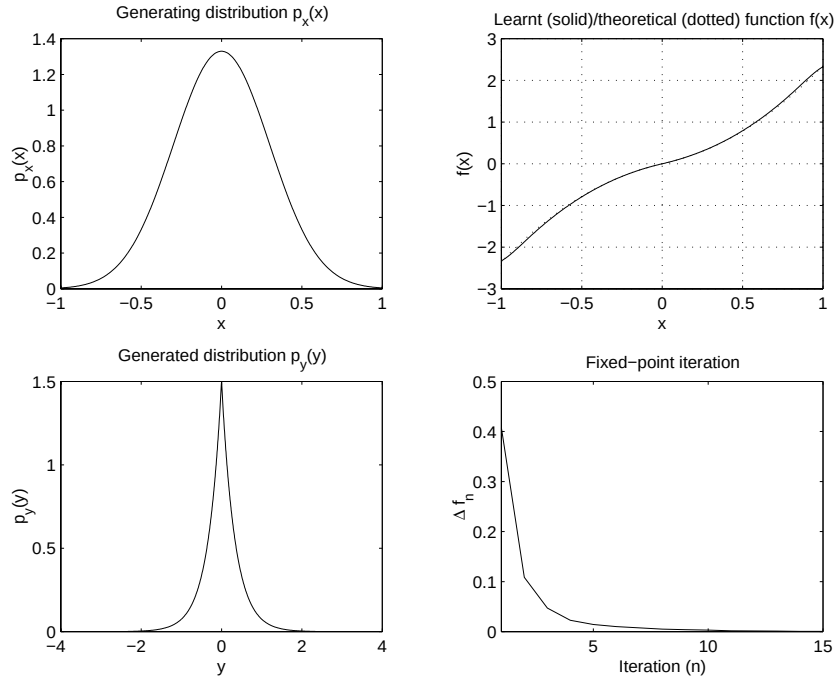

Figure 4: Result of warping-system adaptation with Gaussian input and Laplacian output.

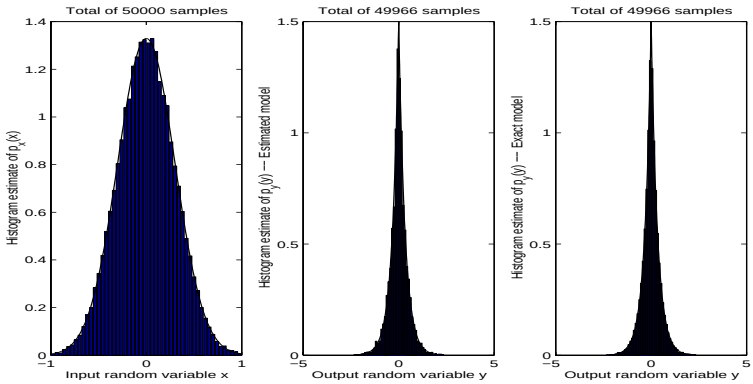

Figure 5: Histogram estimates of the generated Gaussian data (left-hand), Laplacian output obtained with the learnt non-linearity (center-panel) and, Laplacian output obtained with the exact non-linearity (right-hand).

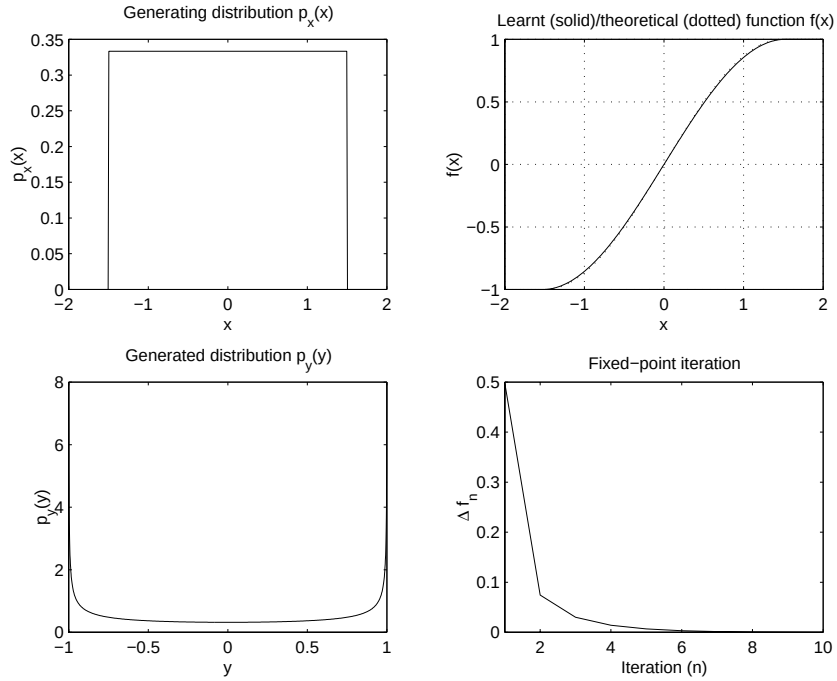

Figure 6: Result of warping-system adaptation with uniform input and U-shaped output.

to the denominator in (**A1**). In this case the found non-linearity is bounded (saturating) as predicted by the closed-form solution and the tuned non-linear transfer function looks in good accordance with the theoretical one. The values of the index Δf_n shows that the fixed-point algorithm may be stopped after 10 iterations. In Figure 5 we may observe the histogram estimates (with 50 bins) of the generated Gaussian data, the U-shaped output obtained with the learnt non-linearity, and, for comparison purposes, the U-shaped output obtained with the exact non-linearity.

The third example concerns the generation of a Rayleigh distribution, with $\sigma = 0.5$, from a skewd-to-the-right uniform distribution $\mathcal{U}_{[0,2]}$. In this case and it is appropriate to take $\bar{\mathcal{X}} = [0, 1.999]$. The results provided by the algorithm are shown in the Figure 8. These results pertain to values $N = 2,001$, $Q = 50,000$, $\gamma = 0.01$, $m = 6$ within algorithm (18) plus (**A3**). In this case the found non-linearity is unbounded as predicted by the closed-form solution and the tuned non-linear transfer function looks in good accordance with the theoretical one. The values of the learning index Δf_n shows that the fixed-point

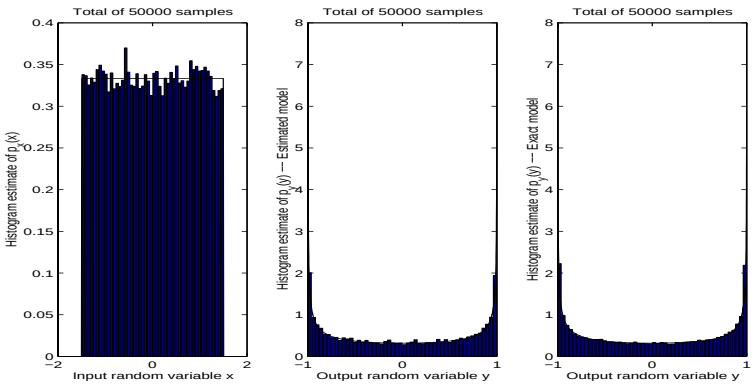

Figure 7: Histogram estimates of the generated uniform data (left-hand), U-shaped output obtained with the learnt non-linearity (center-panel) and, U-shaped output obtained with the exact non-linearity (right-hand).

algorithm may be stopped after 10 iterations. In Figure 9 we may observe the histogram estimates (with 50 bins) of the generated uniform data, the Rayleigh output obtained with the learnt non-linearity, and, for comparison purposes, the Rayleigh output obtained with the exact non-linearity.

The Table 1 reports the comparison of the theoretical moments, the empirical moments pertaining to the random numbers obtained through the learnt non-linearity and the empirical moments pertaining to the random numbers obtained through the exact non-linearity. It is important to note that, in order to generate the samples through the non-linearity (both exact and learnt), only available samples of the generators *within* $\mathcal{X}$ have been used, for fair comparison.

For the Laplacian-from-Gaussian experiment, it is interesting to note that the moments obtained with the learnt non-linearity are closer to the theoretical ones than the moments corresponding to the exact non-linearity. This might be explained by noting that, from Figures 4 and 5, the learnt model slightly overestimates the domain of the output variable y, therefore the moments are slightly overestimated.

For the U-shaped-from-uniform experiment, both probability density function shape and moments comparison show that the output distribution is quite close to the wanted distribution $p_y(y)$, even if the generated input uniform distribution does not match the true probability density function $p_x(x)$ in a utterly accurate way.

For the Rayleigh-from-uniform experiment, the higher-order moments ob-

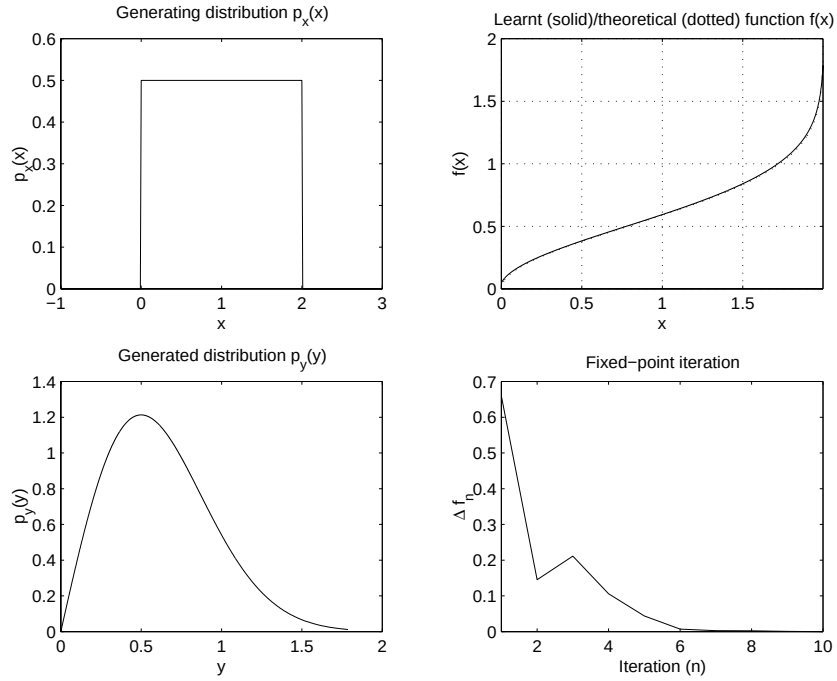

Figure 8: Result of warping-system adaptation with uniform input and Rayleight output.

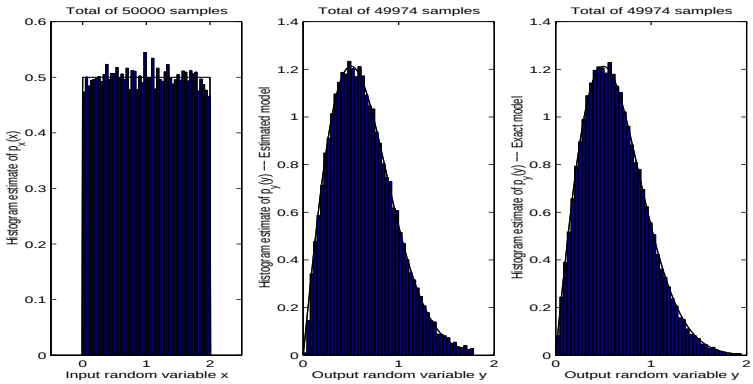

Figure 9: Histogram estimates of the generated uniform data (left-hand), Rayleigh-type output obtained with the learnt non-linearity (center-panel) and, Rayleigh-type output obtained with the exact non-linearity (right-hand).

MOMENTS	$m=1$	$m=2$	$m=3$	$m=4$	$m=5$	$m=6$
$\mu_S^{(m)}(3)$	0	0.2222	0	0.2963	0	0.9877
$\hat{\mu}_S^{(m)}(3)$	–	0.2273	–	0.2836	–	0.6972
$\hat{\mu}_{S(em)}^{(m)}(3)$	–	0.2167	–	0.2487	–	0.5610
$\mu_U^{(m)}$	0	0.5000	0	0.3750	0	0.3125
$\hat{\mu}_U^{(m)}$	–	0.4854	–	0.3571	–	0.2933
$\hat{\mu}_{U(em)}^{(m)}$	–	0.5011	–	0.3752	–	0.3120
$\mu_R^{(m)}(0.5)$	0.6267	0.5000	0.4700	0.5000	0.5875	0.7500
$\hat{\mu}_R^{(m)}(0.5)$	0.6316	0.5048	0.4739	0.5011	0.5807	0.7234
$\hat{\mu}_{R(em)}^{(m)}(0.5)$	0.6243	0.4939	0.4585	0.4797	0.5514	0.6840

Table 1: Comparison of empirical moments $\hat{\mu}_D^{(m)}(p_1, p_2)$ obtained through the learnt non-linearity, empirical moments $\hat{\mu}_D^{(m)}(em)(p_1, p_2)$ obtained through the exact model and exact moments $\mu_D^{(m)}(p_1, p_2)$, pertaining to generated Laplacian, U-shaped and Rayleigh random variables. (Known-to-be-zero empirical moments were not computed.)

tained with the learnt non-linearity are closer to the theoretical ones than the moments corresponding to the exact non-linearity. Again, this is because the learnt model slightly overestimates the domain of the output variable y.

3.3.2 Experiments on samples generation without closed-form solutions

The first example of generation of a random variable for which the closed-form solution is not available concerns a Gamma distribution defined by:

$$B_{a,b}(u) \stackrel{\text{def}}{=} \frac{ab^{\frac{1}{a}}}{2\Gamma\left(\frac{1}{a}\right)} \exp(-b|u|^a) \ . \tag{31}$$

with $a = 0.8$ and $b = 4$, from a Gaussian with $\sigma_x = 0.3$. In this case we took $\bar{\mathcal{X}} = [-1.5, +1.5]$, $N = 10,001$, $Q = 50,000$, $\gamma = 0.01$, $m = 6$ within algorithm (18) plus (**A3**). The results provided by the algorithm are shown in the Figure 10. In this case the found non-linearity is curly as required to warp a Gaussian distribution into a Gamma one. The values of the index Δf_n shows that the fixed-point algorithm may be stopped after 15 iterations. In Figure 11 we may observe the histogram estimates (with 50 bins) of the

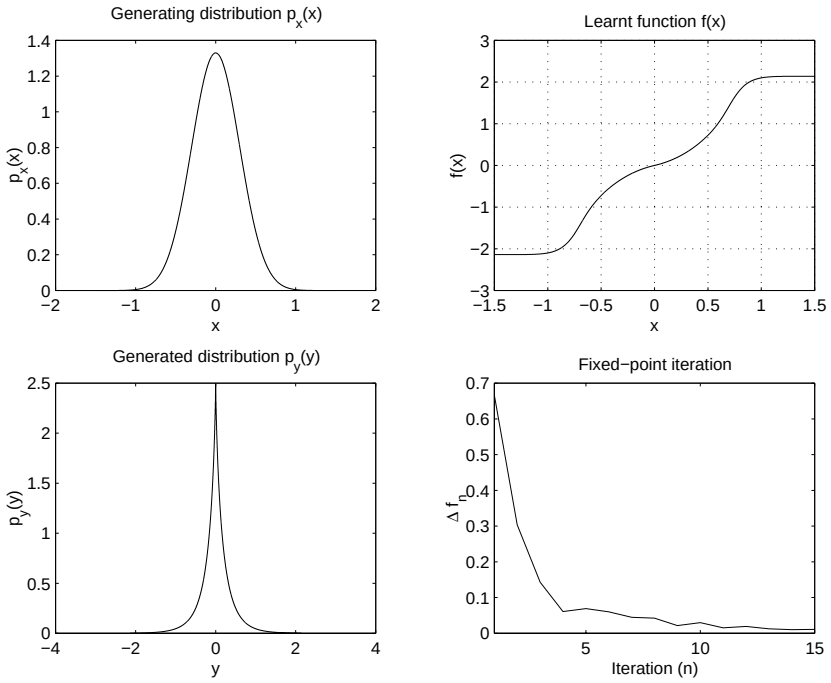

Figure 10: Result of warping-system adaptation with Gaussian input and Gamma output.

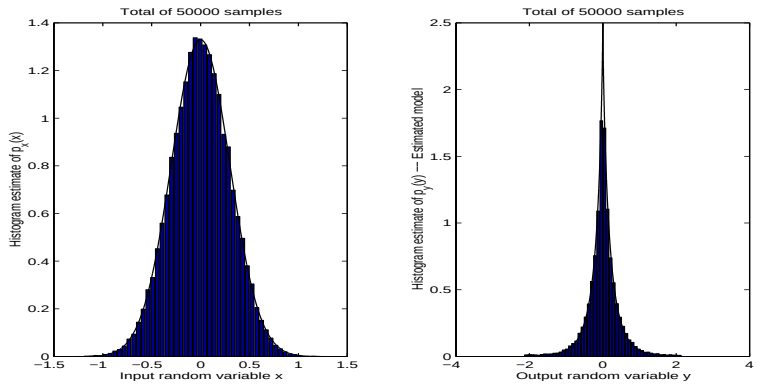

Figure 11: Histogram estimates of the generated Gaussian data (left-hand) and Gamma output obtained with the learnt non-linearity (right-hand).

generated Gaussian data and the Gamma output obtained with the learnt non-linearity.

As the above experiment pertains to the largest number of bins in the partition of $\bar{\mathcal{X}}$, we report the CPU-time necessary to run the code on a 450Mhz-256MB platform, which is less than 3 seconds.

The second example of generation in absence of closed-form solution concerns a Ricean random variable, with $\sigma = 0.5$ and $\kappa = 0.5$, from a uniform distribution of the type $\mathcal{U}_{[0,2]}$. In this case we took $\bar{\mathcal{X}} = [0, 1.9999]$, $N = 2,001$, $Q = 50,000$, $\gamma = 0.01$, $m = 6$ within algorithm (18) plus (**A3**). The results provided by the algorithm are shown in the Figure 12. The values of the index Δf_n shows that the fixed-point algorithm, in the present case, may be stopped after 10 iterations. In Figure 13 we may observe the histogram estimates (with 50 bins) of the generated uniform data and the Ricean output obtained with the learnt non-linearity.

The Table 2 reports the comparison of the exact and empirical moments. In the Ricean case, the agreement between the true and estimated moments of the obtained random samples is very good. A slight disagreement in the moment of order 4 is observed for the Gamma distribution.

4 Conclusion

We presented a pseudo-random samples generator based on a non-linear monotonic neural system. The neural system is tuned on the basis of the cardinal

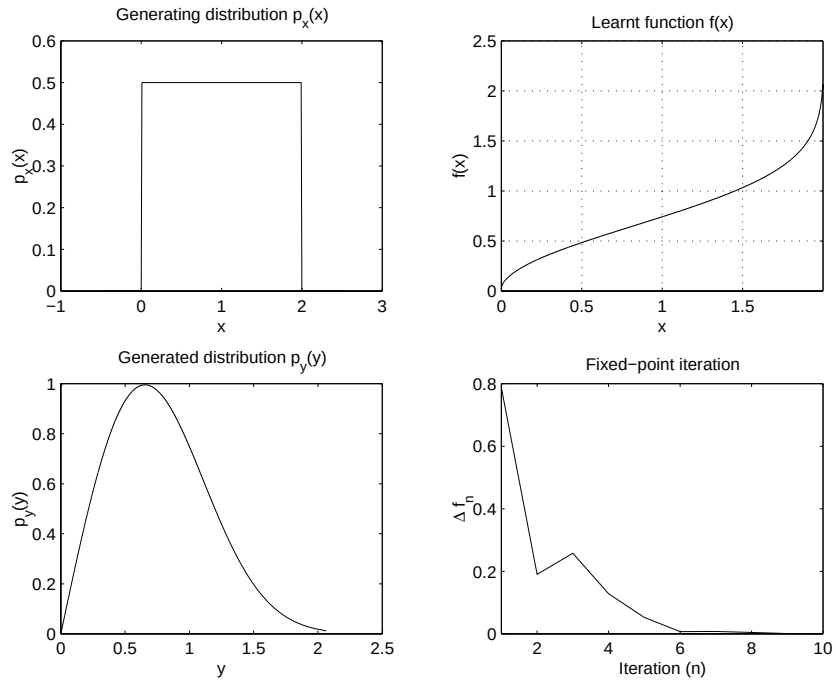

Figure 12: Result of warping-system adaptation with uniform input and Ricean output.

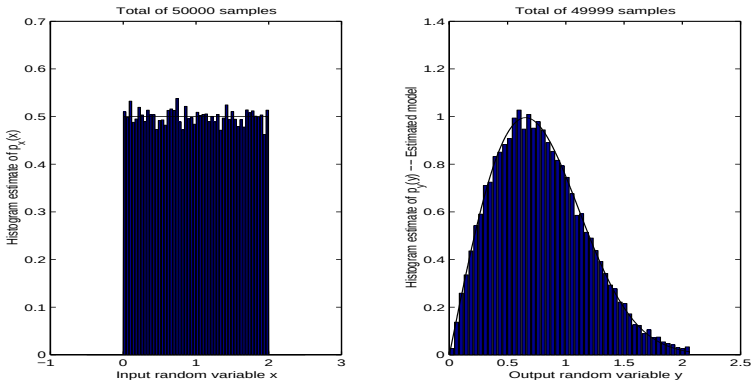

Figure 13: Histogram estimates of the generated uniform data (left-hand) and Ricean output obtained with the learnt non-linearity (right-hand).

MOMENTS	$m=1$	$m=2$	$m=3$	$m=4$	$m=5$	$m=6$
$\mu_B^{(m)}(0.8,4)$	0	0.1524	0	0.1992	0	0.7979
$\hat{\mu}_B^{(m)}(0.8,4)$	–	0.2026	–	0.2977	–	0.7926
$\mu_P^{(m)}(0.5,0.5)$	0.7743	0.7500	0.8439	1.0625	1.4633	2.1719
$\hat{\mu}_P^{(m)}(0.5,0.5)$	0.7784	0.7582	0.8570	1.0798	1.4786	2.1618

Table 2: Comparison of empirical moments $\hat{\mu}_D^{(m)}(p_1,p_2)$ and exact moments $\mu_D^{(m)}(p_1,p_2)$ pertaining to obtained Gamma and Ricean. (Known-to-be-zero empirical moments were not computed.)

probability density function transformation equation of statistics. The cardinal equation may be interpreted as a design equation that shows as a non-linear monotonic system warps an available signal of known statistics into another random signal with desired statistics.

The key points of the proposed methods are:

- In order to obtain a fully-tunable neural transference function, a look-up-table representation has been chosen. It guarantees high flexibility in the shape of the neural transference as well as easiness of representation and handling of the involved quantities. Also, the look-up-table based representation overcomes known limitations of e.g. polynomial-based or kernel-based representations due to sticking in local minima during parameters adaptation and the problem of model order selection.

- The developed fixed-point learning algorithm exhibits fast convergence over other possible methods such as the gradient-based one, unlike these it does not require the computation of derivatives of the involved functions, and is ensured to converge by proven learnability theorems.

- The learnability results, proven in section 2.1, look to be applicable to a wide range of random generation problems.

Cases-study illustrated the behavior of the proposed algorithm, both in cases in which the closed form-solution is available, so that a direct comparison may be effected between the learnt neural transference and the exact one, and in cases where the exact solution is not available, whose results were evaluated indirectly through output-probability-density-function-shape as well as output high-order

moments comparison. The obtained results confirm that the generated samples statistic are in good agreement with the desired statistic.

5 Acknowledgments

I would like to gratefully thank Prof. Domenico Candeloro for the helpful discussion on the proofs of Theorems 1 and 2.

References

[1] G.E. Andrews, R. Askey, and R. Roy, *Laguerre polynomials*, Section 6.2 in *Special Functions*. Cambridge, England: Cambridge University Press, pp. 282 – 293, 1999

[2] M. Abramowitz and C.A. Stegun, (Eds.), *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables*, 9$^{\text{th}}$ printing, New York: Dover, 1972

[3] P. L'Ecuyer, *Random Number Generation*. In J. Banks (Ed.), The Handbook of Simulation, pp. 93 – 137. Wiley, New York, 1998

[4] L.C. Evans, *Partial Differential Equations*, Graduate Studies in Mathematics, 19, American Mathematical Society, 1998

[5] S. Fiori, *Blind signal processing by the adaptive activation function neurons*, Neural Networks, Vol. 13, No. 6, pp. 597 – 611, Aug. 2000

[6] S. Fiori, *Probability density function learning by unsupervised neurons*, International Journal of Neural Systems, Vol. 11, No. 5, 399 – 417, Oct. 2001

[7] S. Fiori, *Non-symmetric PDF estimation by artificial neurons: Application to statistical characterization of reinforced composites*, IEEE Trans. on Neural Networks, Vol. 14, No. 4, pp. 959 – 962, July 2003

[8] P.G. Georgiou and C. Kyriakakis, *An alternative model for sound signals encountered in reverberant environments; robust maximum likelihood localization and parameter estimation based on a sub-Gaussian model*, 113$^{\text{th}}$ Audio Engineering Society Convention, October 2002, Los Angeles (California, USA)

[9] G. Marsaglia, *A current view of random number generators*. In L. Billard, (Ed.), Computer Science and Statistics: The Interface, pp. 3 – 10. Elsevier Science Publishers B.V., Amsterdam, 1985.

[10] F. Morrison, *The Art of Modeling Dynamic Systems: Forecasting for Chaos, Randomness, and Determinism*, Wiley-Interscience, January 1991

[11] H. Ney, *Stochastic modeling: From pattern classification to speech recognition and translation*, IEEE International Conference on Pattern Recognition (ICPR'00), Vol. 3, p. 3025, 2000

[12] H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods*, SIAM, Philadelphia, 1992

[13] A. Papoulis, *Probability and Statistics*, Prentice Hall, 1996

[14] A. Papoulis, *The Fourier Integral and Its Applications*, New York: McGraw-Hill, 1962

[15] B.D. Ripley, *Thoughts on pseudorandom number generators*, Journal of Computer and Applied Mathematics, Vol. 31, pp. 153 – 163, 1990

A Appendix: Moments of some distributions

In this Appendix, details are given about the computation of the even- and odd-order (non centered) moments of some of the distributions considered within the paper. Such moments are denoted as $\mu_D^{(m)}(p_1, p_2)$ where $m = 1,\ 2,\ 3,\ ...$ is the order, D denotes the kind of distribution and p_1 and p_2 are the parameters that the distribution (hence the moments) may depend upon.

Within the text and in the following, we make use of some special functions (which also helps expressing the moments in closed form):

- $I_\nu(u)$ denotes the modified Bessel function of the first kind [2].
- $L_\beta(u)$ are referred to as Laguerre polynomials [1]
- $\Gamma(u)$ and is referred to as Gamma function [2]. In the case that u is an integer value, let us say $u = m + 1$, with $m \in \mathbb{N}$, then the identity $\Gamma(m+1) = m!$ holds, where we denoted the factorial of an integer number m by $m!$.

- $m!!$ denotes the product of every positive odd integer number up to m if m is a positive odd integer as well as the product of every positive even integer number up to m if m is a positive even integer.

Moments of the Gaussian probability density function. It is necessary to compute the following integral:

$$\mu_G^{(m)}(\sigma) \stackrel{\text{def}}{=} \int_{-\infty}^{+\infty} u^m G_\sigma(u) du \ .$$

For odd values of the order m, we have $\mu_G^{(m)} = 0$, while for even values of the order, the moment of the Gaussian distribution expresses as:

$$\mu_G^{(m)}(\sigma) = \frac{(\sqrt{2}\sigma)^m}{\sqrt{\pi}} \Gamma\left(\frac{m+1}{2}\right) = (m-1)!!\sigma^m \ . \tag{32}$$

The first three even-order moments of the Gaussian distribution are:

$$\mu_G^{(2)}(\sigma) = \sigma^2 \ , \ \mu_G^{(4)}(\sigma) = 3\sigma^4 \ , \ \mu_G^{(6)}(\sigma) = 15\sigma^6 \ .$$

Moments of the Gamma probability density function. It is necessary to compute the following integral:

$$\mu_B^{(m)}(a,b) \stackrel{\text{def}}{=} \int_{-\infty}^{+\infty} u^m B_{a,b}(u) du \ .$$

For odd values of m, we have $\mu_B^{(m)} = 0$, while for even values of the order, the moment of the Gamma distribution expresses as:

$$\mu_B^{(m)}(a,b) = \frac{\Gamma\left(\frac{1+m}{a}\right)}{b^{\frac{m}{a}}\Gamma\left(\frac{1}{a}\right)} \ . \tag{33}$$

Moments of the Laplacian probability density function. In this case, it is necessary to compute the following integral:

$$\mu_S^{(m)}(\lambda) \stackrel{\text{def}}{=} \int_{-\infty}^{+\infty} u^m S_\lambda(u) du \ .$$

For odd values of the order m, we have $\mu_S^{(m)} = 0$, while for even values of the order we have:

$$\mu_S^{(m)}(\lambda) = \frac{m!}{\lambda^m} \ . \tag{34}$$

The first three even-order moments of the Laplacian distribution are:

$$\mu_S^{(2)}(\lambda) = \frac{2}{\lambda^2} \ , \ \mu_S^{(4)}(\lambda) = \frac{24}{\lambda^4} \ , \ \mu_S^{(6)}(\lambda) = \frac{720}{\lambda^6} \ .$$

Moments of the U-shaped probability density function. It is necessary to compute the following integral:

$$\mu_U^{(m)} \stackrel{\text{def}}{=} \int_{-1}^{+1} u^m U(u) du \ .$$

For odd values of m, we have $\mu_U^{(m)} = 0$, while for even values of the order we have:

$$\mu_U^{(m)} = \frac{\Gamma\left(\frac{m+1}{2}\right)}{\sqrt{\pi}\left(\frac{m}{2}\right)!} \ . \tag{35}$$

The first three even-order moments of the U-shaped distribution are:

$$\mu_U^{(2)} = \frac{1}{2} \ , \ \mu_U^{(4)} = \frac{3}{8} \ , \ \mu_U^{(6)} = \frac{5}{16} \ .$$

Moments of the Rayleigh probability density function. In this case, it is necessary to compute the following integral:

$$\mu_R^{(m)}(\sigma) \stackrel{\text{def}}{=} \int_0^{+\infty} u^m R_\sigma(u) du \ .$$

The result of integral computation, which is valid for every order m, is:

$$\mu_R^{(m)}(\sigma) = (\sqrt{2}\sigma)^m \Gamma\left(1 + \frac{m}{2}\right) \ . \tag{36}$$

The first six moments of the Rayleigh distribution are:

$$\mu_R^{(1)}(\sigma) = \sqrt{\frac{\pi}{2}}\sigma \ , \ \mu_R^{(2)}(\sigma) = 2\sigma^2 \ , \ \mu_R^{(3)}(\sigma) = 3\sqrt{\frac{\pi}{2}}\sigma^3 \ ,$$
$$\mu_R^{(4)}(\sigma) = 8\sigma^4 \ , \ \mu_R^{(5)}(\sigma) = 15\sqrt{\frac{\pi}{2}}\sigma^5 \ , \ \mu_R^{(6)}(\sigma) = 48\sigma^6 \ .$$

Moments of Rice's probability density function. It is necessary to compute the following integral:

$$\mu_P^{(m)}(\kappa, \sigma) \stackrel{\text{def}}{=} \int_0^{+\infty} u^m P_{\kappa,\sigma}(u) du \ .$$

The result of integral computation, which is valid for every order m, is:

$$\mu_P^{(m)}(\kappa, \sigma) = (\sqrt{2}\sigma)^m \Gamma\left(1 + \frac{m}{2}\right) L_{m/2}\left(-\frac{\kappa^2}{2\sigma^2}\right) \ . \tag{37}$$

The first six moments of the Ricean distribution are:

$$\begin{aligned}
\mu_P^{(1)}(\kappa, \sigma) &= \sigma\sqrt{\frac{\pi}{2}} L_{1/2}\left(-\frac{\kappa^2}{2\sigma^2}\right) \ , \\
\mu_P^{(2)}(\kappa, \sigma) &= \kappa^2 + 2\sigma^2 \ , \\
\mu_P^{(3)}(\kappa, \sigma) &= 3\sigma^3\sqrt{\frac{\pi}{2}} L_{3/2}\left(-\frac{\kappa^2}{2\sigma^2}\right) \ ,
\end{aligned}$$

$$
\begin{aligned}
\mu_P^{(4)}(\kappa,\sigma) &= \kappa^4 + 8\kappa^2\sigma^2 + 8\sigma^4 \ , \\
\mu_P^{(5)}(\kappa,\sigma) &= 15\sigma^5\sqrt{\frac{\pi}{2}}L_{5/2}\left(-\frac{\kappa^2}{2\sigma^2}\right) \ , \\
\mu_P^{(6)}(\kappa,\sigma) &= \kappa^6 + 18\kappa^4\sigma^2 + 72\kappa^2\sigma^4 + 48\sigma^6 \ .
\end{aligned}
$$