Christian Knoll, `christian.knoll@tugraz.at`

Nonlinear systems with fading memory

Representation of nonlinear systems using Volterra series

Volterra representation of an arbitrary nonlinear systems is given by:

$$y(t) = \int_0^\infty h_1(\tau)x(t-\tau)d\tau + \int_0^\infty \int_0^\infty h_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)d\tau_1 d\tau_2 + \dots$$
$$\dots + \int_0^\infty \int_0^\infty \dots \int_0^\infty h_n(\tau_1,\tau_2,\dots,\tau_n)x(t-\tau_1)x(t-\tau_2)\dots x(t-\tau_n)d\tau_1 d\tau_2 \dots d\tau_n + \dots$$

The discrete-time representation is:

$$y(t) = \sum_{i=0}^{\infty} h_1[i]x[n-i] + \sum_{i=0}^{\infty}\sum_{j=0}^{\infty} h_2[i,j]x[n-i]x[n-j] + \dots$$
$$\dots + \sum_{i=0}^{\infty}\sum_{j=0}^{\infty}\dots\sum_{k=0}^{\infty} h_n[i,j,\dots,k]x[n-i]x[n-j]\dots x[n-k] + \dots$$

In practice, both series are truncated in time as well as in order to keep the number of nonlinear terms $h(\cdot)$ finite.

Problem 3.1

For given input sequences $x[n]$ and nonlinear systems defined in `nlsystems1.m`, generate corresponding output signals. Use `vkernels.m` to identify the parameters of the Volterra model. Compare the estimated parameters with the true ones (see help on `nlsystem1.m`). Which input sequence produces the most accurate estimates? Why?

(a) $x[n] = n - 512$; $n = [0 : 1023]$

(b) $x[n] = \sin(0.5\pi n)$; $n = [0 : 1023]$

(c) $x[n] = \mathcal{N}(0,\sigma^2)$, i.e., random samples drawn from a normal distribution with zero mean and variance $\sigma^2 = 1$; $n = [0 : 1023]$

Problem 3.2

Let the input signal be $x[n] = \cos(0.1\pi n)$. Using the nonlinearities defined in `nlsystem1.m`, generate the corresponding output sequences $y_{\text{sys}}[n]$ of the nonlinear system. Now, using white noise input, identify the Volterra kernel of these nonlinear systems and, using the identified parameters, compute the output of the corresponding approximation $y_{\text{model}}[n]$ (use `vkernels_o.m` to compute the outputs). Compare both signals with each other. As a quality measure, use

$$Q_2 = 10\log_{10}\frac{\sum_n |y_{\text{model}}[n] - y_{\text{sys}}[n]|^2}{\sum_n |y_{\text{sys}}[n]|^2}$$

Problem 3.3

Let us assume there is a nonlinear process $s[n]$ that we wish to model using a Volterra series. To do that we partition $s[n]$ into two sequences: a training sequence $s_1[n]$ and a validation sequence $s_2[n]$. The first sequence $s_1[n]$ is used to estimate the parameters of the Volterra model (training stage), and the second sequence $s_2[n]$ is used to check the modeling performance (validation stage). Output $y[n]$ and input $x[n]$ for the Volterra system are then generated from $s_1[n]$ and $s_2[n]$ such that $y_1[n] = x_1[n+1]$ and $y_2[n] = x_2[n+1]$, respectively.

Using $x_1[n]$ as the input and $y_1[n]$ as the output signal, identify the parameters of the Volterra model with `vkernels.m`. Try different memory lengths and nonlinearity orders. Compare the performance of the estimated models on the validation signals $x_2[n]$ and $y_2[n]$ using Q_2. Perform this analysis for:

(a) a vowel sound (use `vowel.m` to load data into Matlab's workspace).

(b) A received ultra-wideband radio signal from an indoor scenario including many reflections and signal scattering (load `radiochannel.mat`).

Higher-order statistics and spectral analysis

The joint non-central moments of order r of random variables $X_1, \ldots, X_n$ are defined:

$$M_r(X_1^{k_1}, \ldots, X_n^{k_n}) = E\{X_1^{k_1} X_2^{k_2} \ldots X_n^{k_n}\} = (-j)^r \left. \frac{\partial^r \Phi(w_1, \ldots, w_n)}{\partial w_1^{k_1} \ldots \partial w_n^{k_n}} \right|_{w_1 = \ldots = w_n = 0}$$

where $\sum_{i=1}^n k_i = r$, and $\Phi(\cdot)$ is the joint characteristic function.
The joint cumulants are defined as

$$C_r(X_1^{k_1}, \ldots, X_n^{k_n}) = (-j)^r \left. \frac{\partial^r \ln \Phi(w_1, \ldots, w_n)}{\partial w_1^{k_1} \ldots \partial w_n^{k_n}} \right|_{w_1 = \ldots = w_n = 0}$$

For a stationary discrete-time random process $x[n]$ the non-central moments of order $r > 1$ are defined as

$$m_r^x(\tau_1, \tau_2, \ldots, \tau_{r-1}) = E\{x[n]x[n+\tau_1]x[n+\tau_2] \cdots x[n+\tau_{r-1}]\},$$

where $\tau_r \in \mathbb{Z}$.

The r-th order *cumulant* is a function of the moments of order up to r

$$c_1^x = m_1^x = E\{x[n]\} \text{ (first order cumulants)}$$
$$c_2^x(\tau_1) = m_2^x(\tau_1) - (m_1^x)^2 = E\{x[n]x[n+\tau_1]\} - E\{x[n]\}^2 \text{ (second order cumulants)}$$
$$c_3^x(\tau_1, \tau_2) = m_3^x(\tau_1, \tau_2) - m_1^x \cdot \Big(m_2^x(\tau_1) + m_2^x(\tau_2) + m_2^x(\tau_1 - \tau_2) \Big) + 2(m_1^x)^3 \text{ (third order cumulants)}$$
$$\vdots$$

Spectra of the cumulants and moments of the random variable $X = \{x[n], n \in \mathbb{Z}\}$ are defined as the Fourier transforms of the corresponding functions (assuming that the latter are absolutely summable):

$$\mathcal{M}_r^x(w_1, \ldots, w_{r-1}) = \mathcal{F}\{m_r^x(\tau_1, \ldots, \tau_{r-1})\}$$
$$\mathcal{C}_r^x(w_1, \ldots, w_{r-1}) = \mathcal{F}\{c_r^x(\tau_1, \ldots, \tau_{r-1})\}$$

where $\mathcal{F}\{\cdot\}$ is a Fourier transform operator.

Some important properties of cumulants are:

- If $x[n]$ is a Gaussian process, then $c_r^x(\tau_1, \ldots, \tau_{r-1}) = 0$ for $r > 2$
- if $x[n]$ is an i.i.d. process, then $c_r^x(\tau_1, \ldots, \tau_{r-1}) = a \cdot \delta(\tau_1, \ldots, \tau_{r-1})$, where a is some constant and $\delta(\cdot)$ is a discrete delta function.
- If $x[n]$ is symmetrically distributed around zero, then $c_r^x(\tau_1, \ldots, \tau_{r-1}) = 0$ for $r = 0, 3, 5, 7, \ldots$
- If $z[n] = x[n] + y[n]$, where $x[n]$ and $y[n]$ are jointly stationary and statistically independent random processes, then $c_r^z(\cdot){=}c_r^x(\cdot){+}c_r^y(\cdot)$. This property does not hold for moments.

Problem 3.4

Using the `cumest.m`, estimate the second and third order cumulants of the given stochastic processes $x[n]$. Use `rpiid.m` to generate the (uncorrelated) signals and `cumest.m` to estimate the cumulants. Note that the estimation of the third order cumulants with `cumest.m` is done only for a single slice row of the 2D correlation function. Thus, you have to iterate through the required rows manually, for example, using the following Matlab code

```
for k = -MaxLag : MaxLag
    c3(:, k+MaxLag+1) = cumest(x, 3, MaxLag, 256, 0, 'unbiased', k);
end
```

Have a look at `help cumest` to be able understand this code. Once done, use `viscumul3.m` to visualize the cumulants for the following processes $x[n]$:

(a) exponentially distributed, i.i.d.

(b) normally distributed, i.i.d

(c) An exponentially distributed i.i.d. process run through an LTI system with impulse response $h[n] = (0.8)^n$. Try $n = 0 \ldots 1$, $n = 0 \ldots 5$, and $n = 0 \ldots 10$.

(d) An i.i.d. Gaussian process run through the LTI system in **(c)**.

(e) An normally distributed i.i.d. process run through a nonlinear system. The Volterra representation of the system can be obtained using `quadnl.m`. Use `vkernels_o.m` to compute the corresponding output.

Problem 3.5

For the signals used in the Problem 3.4, compute the bispectra as a two dimensional DFT of the corresponding cumulants. Use `fft2()` function to compute the 2D FFT. Note that it is essential to pre-window the data before computing the Fourier transform. The `gabrrao.m` script computes the window of the required size. You can use `visbispec3.m` function to plot the computed bispectra.