随机变量、随机序列的计算机产生方法原理 --计算机仿真

在工程设计和科研中，越来越多地使用计算机仿真方法。 在计算机仿真实验中，需要用计算机产生各种不同分布的随机数、产生随机序列。

计算机随机仿真方法具有广泛而重要的应用价值。

1. 均匀分布随机数的计算机模拟产生方法

在计算机高级语言中，例如C语言， 已有专门用于产生均匀随机数的函数 **rand( )**，这种随机数是怎么产生的？ 用的是什么算法？

- ♦ 混合同余法:

选定参数 **a**、**c**，**M**， 再选定一个初值 $y(0)$， 按下式产生随机数:

$$y(n+1) = a\cdot y(n) + c \quad (\mathrm{mod}\ \mathrm{M}) \qquad (\,n = 1,2,\ldots\,)$$

$$x(n+1)=y(n+1)/M$$

其中 **M** 为足够大的整数. 在 **C** 语言中， $M = 2^{16}$，

a、**c** 和 $y(0)$ 均为 **0** 至 **M** 中的常数。

按上面算法公式，则 $y(1),\ldots,y(\mathrm{N})$ 为（**0**，**M**）之间的均匀随机数;

$x(1),\ldots,x(\mathrm{N})$ 为（**0**，**1**）之间的均匀随机数。

- ♦ 乘同余法: 在混合同余法算法中， 令 **c=0**, 则为乘同余法。

2. 任意分布随机数的计算机模拟产生方法

用 [0，1] 区间均匀分布的随机数 X， 产生概率密度为 $p_Y(y)$ 的随机数 Y。

♦ 原理、方法:

首先根据 Y 的分布函数 $F_Y(z)=\int_{-\infty}^{z} p_Y(\lambda)\,d\lambda$ 求出其逆函数 $F_Y^{-1}(z)$,

于是， 先产生一个 [0，1] 区间均匀分布的随机数 X，然后令 $Y=F_Y^{-1}(X)$,

则 Y 即为概率密度为 $p_Y(y)$ 的随机数。

为什么？

证明:

前面已学过随机变量变换公式: 设由随机数 X 变换为 Y, $Y=g(X)$,

则 Y 的概率密度为

$$f_Y(y)=p_X(g^{-1}(y))\left|\frac{dg^{-1}(y)}{dy}\right|$$

当 X 是 [0，1] 区间均匀随机变量，$p_X(x)=1$

$$=\left|\frac{dg^{-1}(y)}{dy}\right| = \left|\frac{dF_Y(y)}{dy}\right|$$

$$Y=g(X)=F_Y^{-1}(X) \rightarrow g^{-1}(Y)=F_Y(Y)$$

$$=p_Y(y)$$ --要产生的概率密度！

例 1　试写出用计算机产生指数分布的随机序列 **Y(n)** 的算法。

方法：

指数分布概率密度为　$p_Y(y)=\alpha e^{-\alpha y}\quad (y\geq 0),$

$$F_Y(z)=\int_{-\infty}^{z}p_Y(\lambda)d\lambda=\int_0^z \alpha e^{-\alpha\lambda}d\lambda=1-e^{-\alpha z},\qquad (z\geq 0)$$

求反函数　$Y=F_Y^{-1}(X)=g(X)=-\dfrac{1}{\alpha}\ln(1-X)$

根据前面的原理和方法，于是有下面算法：

（1）　$n\leftarrow 0;$

（2）　产生一个 **(0，1)** 区间均匀随机数 ***X*(n)**；

（3）　令　$Y(n)=F_Y^{-1}(X(n))=-\dfrac{1}{\alpha}\ln(1-X(n)),$　**Y(n)** 即为所求的指数分布随机数。

（4）　$n\leftarrow n+1,\quad goto\,(2).$

例 2

利用中心极限定理的原理产生高斯随机变量 **Z,** 且均值和方差分别为 m_Z, σ_Z^2.

方法：

（1） 产生 **N** 个 **(0，1)** 区间均匀随机数： $X_i,\ i=1,\ldots,N;$

（2） 令 $Y=\sum_{i=1}^{N}(X_i-\frac{1}{2})/\sqrt{N/12}\ ;$

（3） 令 $Z=\sigma_Z Y+m_Z$

则 **Z** 即近似为 $N(m_Z,\sigma_Z^2)$ 分布的随机数.

反复应用上面算法， 即可得均值和方差分别为 m_Z、σ_Z^2 的高斯随机序列 **Z(n).**

实用中，通常取 **N=12,** 这时 $Y=\sum_{i=1}^{12}X_i-6.$

例 3　产生高斯白噪声随机序列的另一种方法.

（1）产生两个 [0，1] 区间的均匀分布随机数 X_1、X_2，(设相互独立）；

（2）令 $Y_1=\sqrt{-2\ln X_1}\cdot\cos(2\pi X_2)$，$Y_2=\sqrt{-2\ln X_1}\cdot\sin(2\pi X_2)$;

（3）再令 $Z_1=\sigma_Z Y_1+m_Z$，$Z_2=\sigma_Z Y_2+m_Z$;

则 Z_1、Z_2 即为相互独立的且均值为 m_Z、方差为 σ_Z^2 的随机数。

反复应用上面算法，即可得所要的高斯白噪声序列 *Z*(n).

证明：令 $Y_1=\sqrt{-2\ln X_1}\cdot\cos(2\pi X_2)=g_1(X_1,X_2)$, $Y_2=\sqrt{-2\ln X_1}\cdot\sin(2\pi X_2)=g_2(X_1,X_2)$

则反函数为
$$X_1=f_1(Y_1,Y_2)=e^{-\frac{1}{2}(Y_1^2+Y_2^2)},\quad X_2=f_2(Y_1,Y_2)=\frac{1}{2\pi}tg^{-1}(Y_2/Y_1)$$

$$P_{Y_1Y_2}(y_1,y_2)=P_{X_1X_2}(f_1(y_1,y_2),f_2(y_1,y_2))\cdot|J|$$

$$J=\begin{vmatrix}\frac{\partial f_1}{\partial y_1} & \frac{\partial f_1}{\partial y_2}\\ \frac{\partial f_2}{\partial y_1} & \frac{\partial f_2}{\partial y_2}\end{vmatrix}=\begin{vmatrix}-y_1\exp\{\frac{-1}{2}(y_1^2+y_2^2)\} & -y_2\exp\{\frac{-1}{2}(y_1^2+y_2^2)\}\\ \frac{1}{2\pi}\frac{-y_2/y_1^2}{1+(y_2/y_1)^2} & \frac{1}{2\pi}\frac{1/y_1}{1+(y_2/y_1)^2}\end{vmatrix}=-\frac{1}{2\pi}e^{-\frac{1}{2}(y_1^2+y_2^2)}$$

$$\therefore\quad P_{Y_1Y_2}(y_1,y_2)=\frac{1}{2\pi}e^{-\frac{1}{2}(y_1^2+y_2^2)}.$$

于是 $Y_1\sim N(0,1)$, $Y_2\sim N(0,1)$, 且 **Y1** 与 **Y2** 相互独立.

Z_1、Z_2 为相互独立的且均值和方差分别为 m_Z、σ_Z^2 的随机数。

例 4　产生相关函数为 $R_X(m)=a^{|m|},\ (|a|<1)$ 的高斯随机序列 **X(n).**

解:

（1）先产生均值为 **0** 方差为 **1** 的高斯白噪声序列 $W(n)$;　**(可按例2或例3的算法)**

（2）按下面步骤产生 **X(n):**

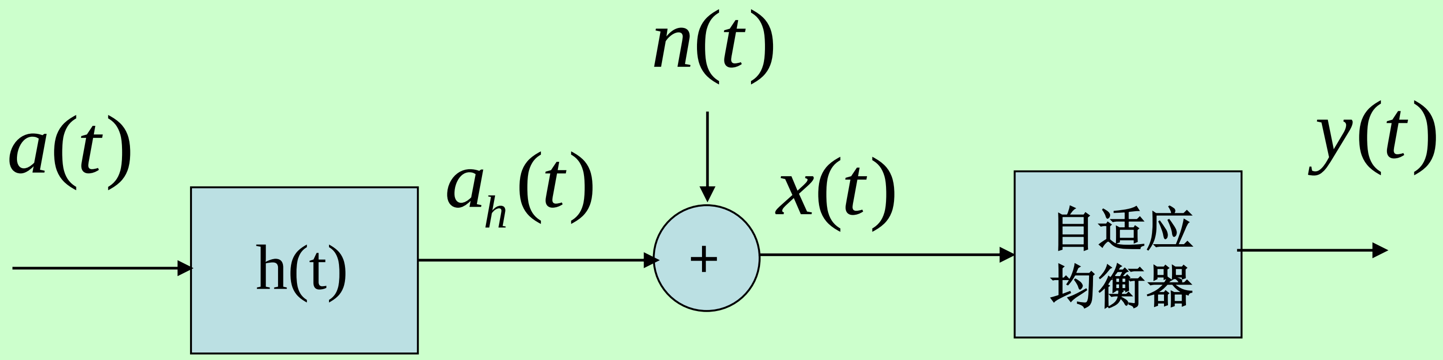
(i) 初始化: $X(0)=W(0)$;

(ii) 递推产生 $X(n+1)=aX(n)+W(n),\qquad n=0,1,2,\cdots\cdots$

则可证明这样产生的 **X(n)** 即为满足本题要求的高斯随机序列.

例 5　　计算机随机仿真在自适应信道均衡研究中的应用举例.

信道均衡问题:



$a(t)$：原信号；　$h(t)$：信道冲激响应；　$a_h(t)$：$h(t)$的输出；

$n(t)$：噪声；　　$x(t)$：信道的输出．也是自适应均衡器的输入.

自适应均衡器：自适应滤波器.

－ 设计准则：　使　　$y(t) \to a(t)$

计算机仿真研究 ：

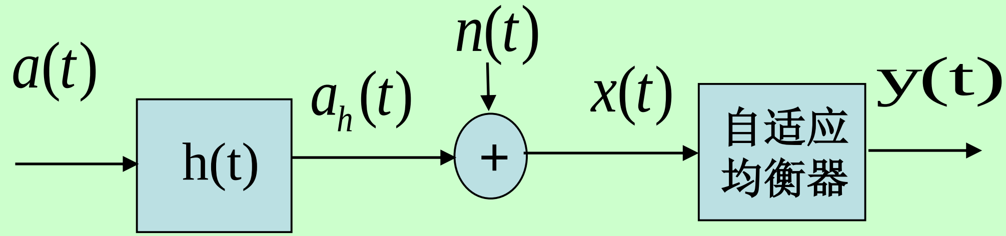
产生原信号 $a(t)$：2 进制数字序列，+1 或 −1，　以随机方式产生 +1 或 −1.

信道 $h(t)$：　　$h(k) = \{ -.557,\ -.05,\ 1.0,\ .033,\ .35 \}$

$$h(0) = -0.557, \ldots, h(4) = 0.35$$

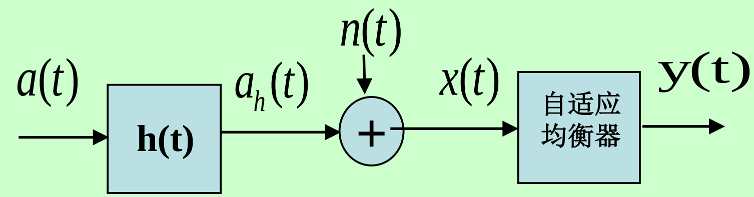
$$a_h(t) = a(t) * h(t) \quad = \sum_k a(k) h(t-k);$$

$$x(t) = a_h(t) + n(t)$$



如何产生噪声?

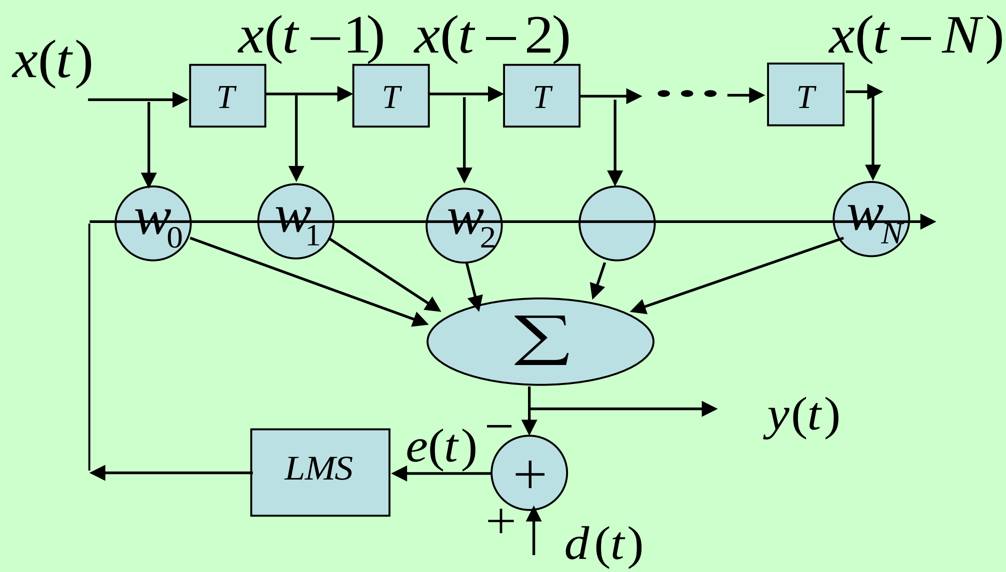
-- 产生高斯白噪声



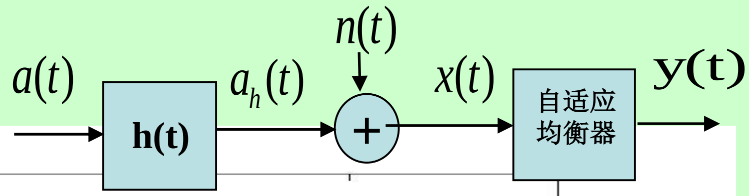
♦ **先产生正态随机数** **X(t) ~ N(0,1)** ；

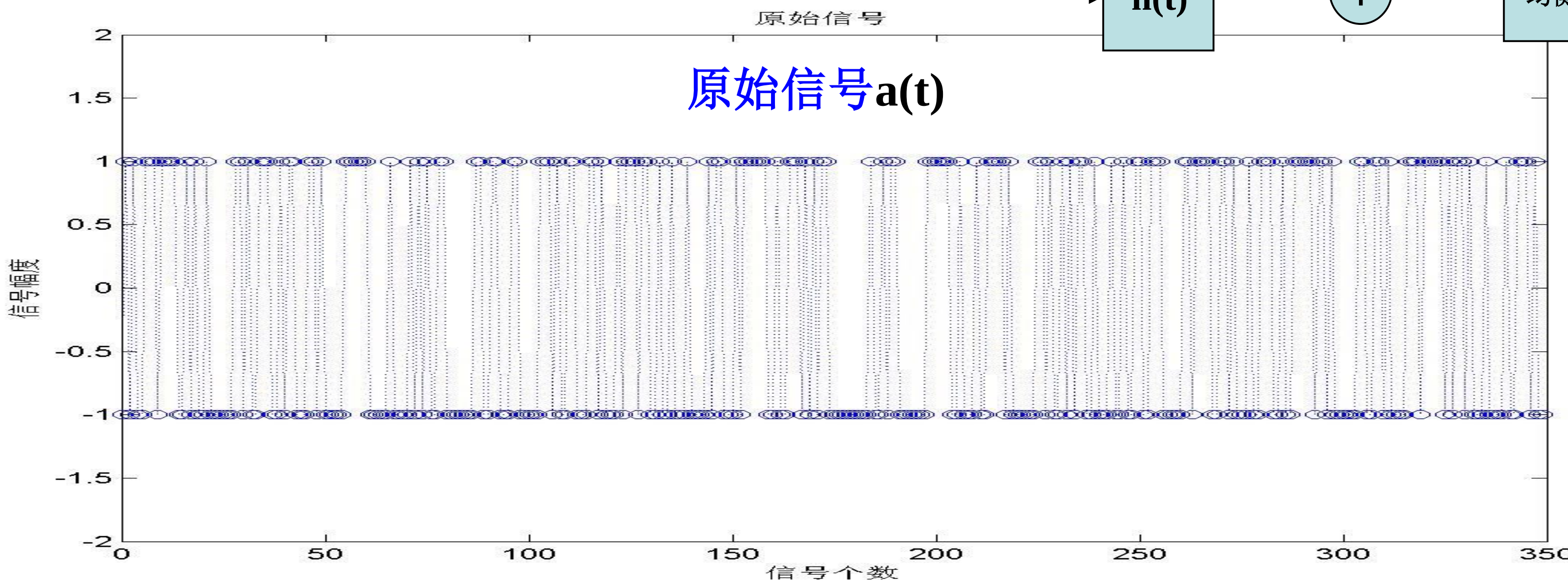
然后 $n(t) = \sigma_X X(t)$

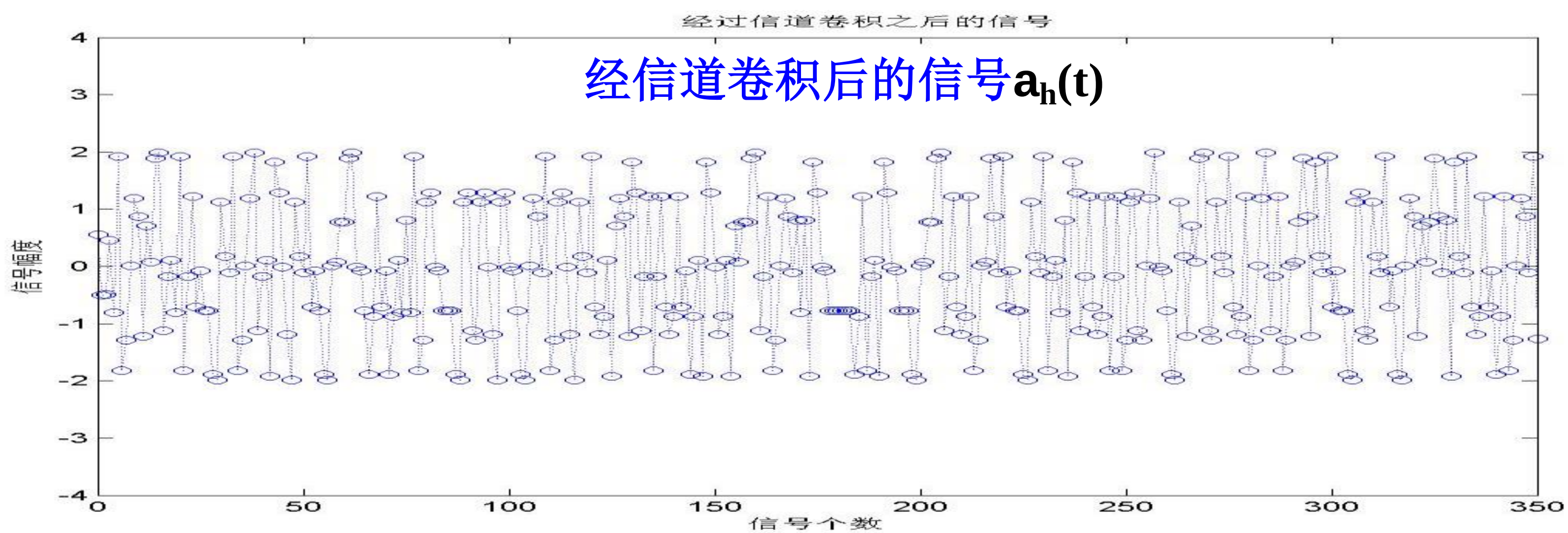
自适应信道均衡器设计举例:



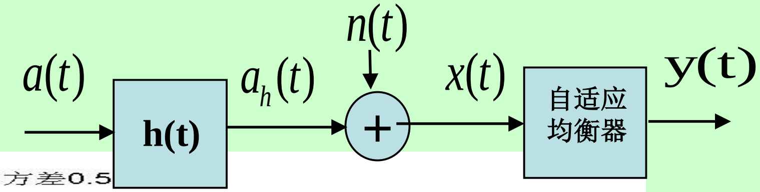
自适应信道均衡结果举例:

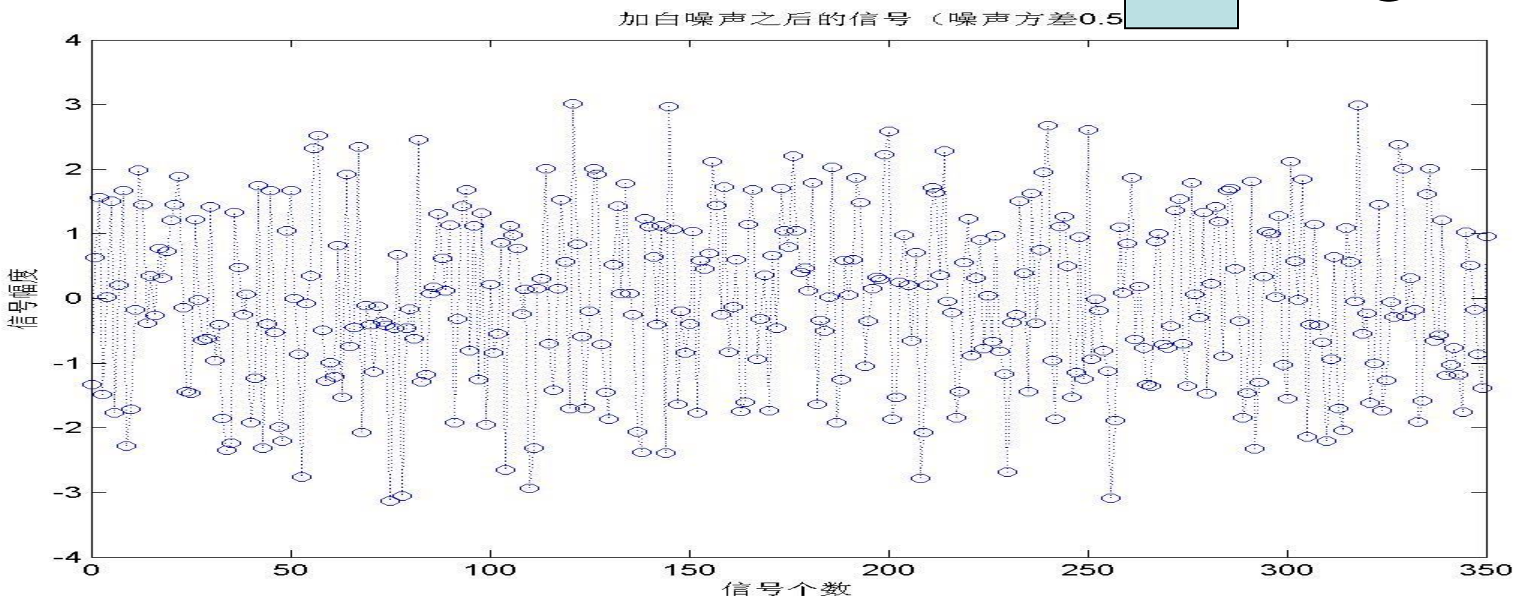




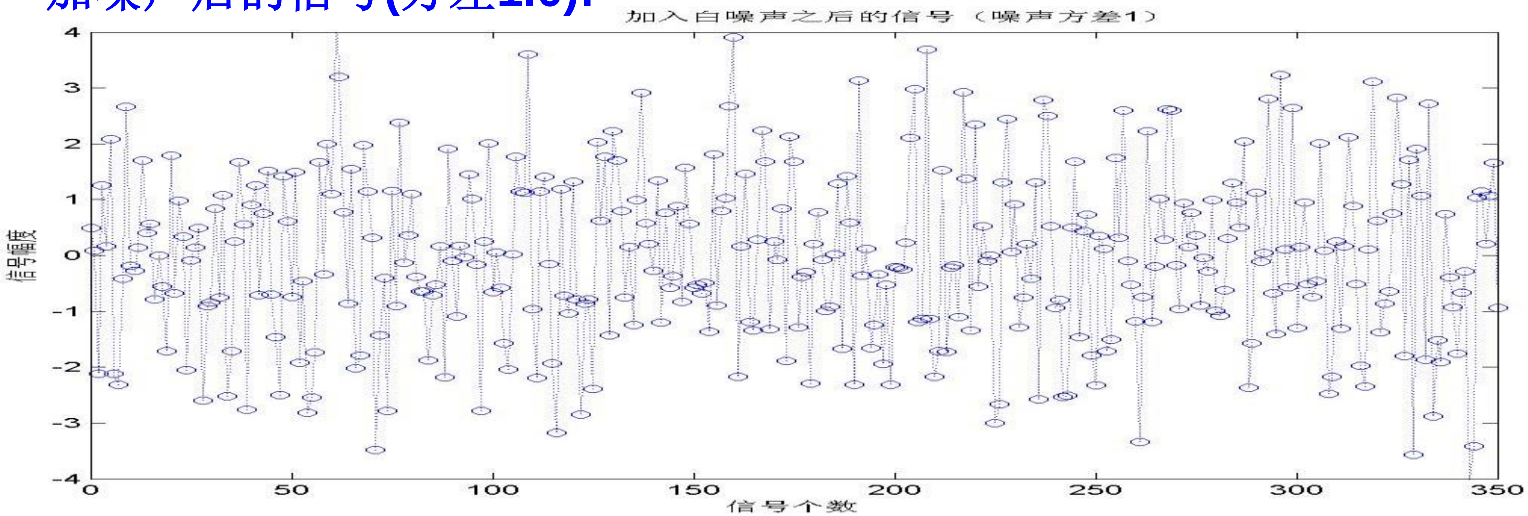


加噪声后的信号(方差0.5):

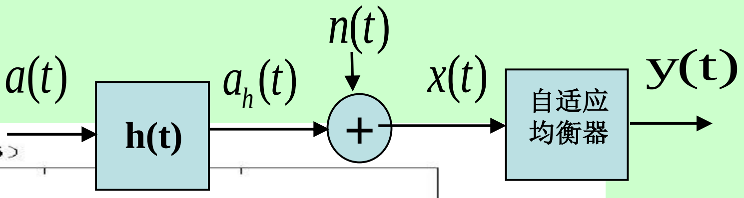


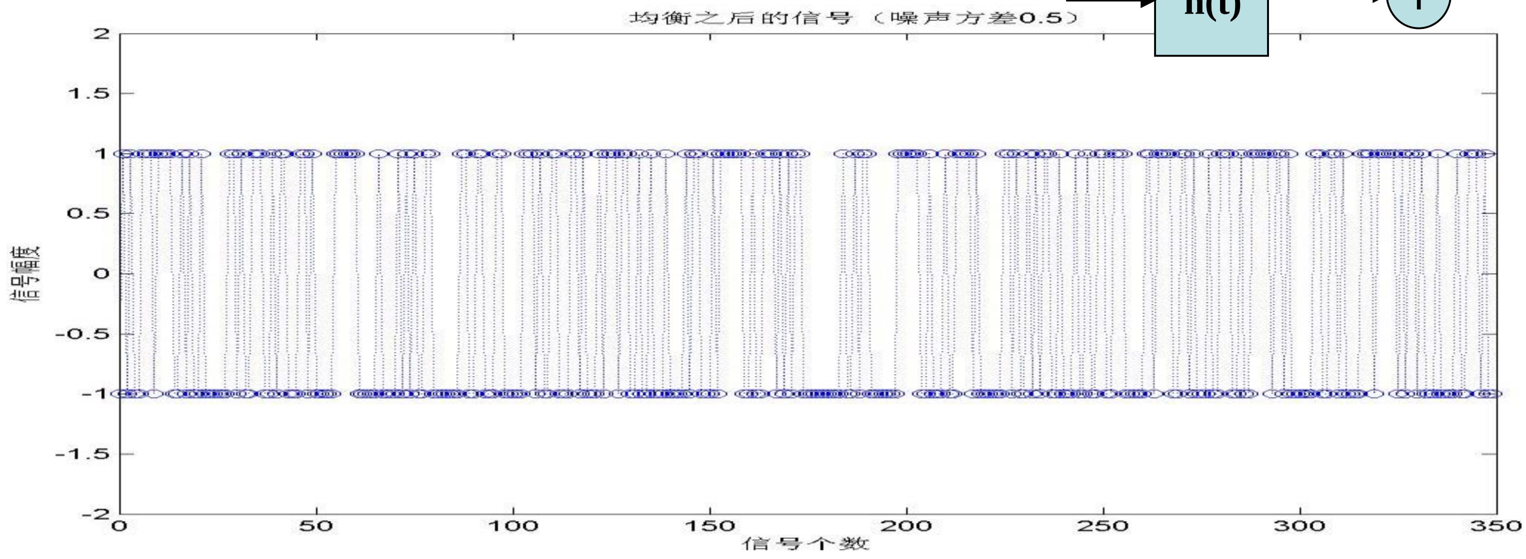


加噪声后的信号(方差1.0):



经自适应均衡后的结果 y(t) (方差0.5):





经自适应均衡后的结果 y(t) (方差1.0):