这段代码是一个BPSK（二进制相移键控）通信系统的完整仿真，包含信号生成、调制、加噪、滤波、解调、判决等环节。下面用通俗易懂的语言分步骤解释：

**一、系统框架**

整个流程可以理解为：发送端（信号生成→调制）→ 信道（加噪声）→ 接收端（滤波→解调→判决）

**二、代码逐段解析**

**1. 参数设置与基带信号生成**

python

|  |  |
| --- | --- |
|  | size = 10 # 发送10个二进制符号 |
|  | sampling\_t = 0.01 # 每个符号持续0.01秒 |
|  | t = np.arange(0, size, sampling\_t) # 生成时间轴（0到1秒，步长0.01秒） |
|  |  |
|  | a = np.random.randint(0, 2, size) # 随机生成0/1序列（如[1,0,1,1,0...]） |
|  | m = np.zeros(len(t)) # 创建连续信号容器 |
|  |  |
|  | # 将离散符号扩展为连续信号（每个符号持续0.01秒） |
|  | for i in range(len(t)): |
|  | m[i] = a[math.floor(t[i])] # 例如：t=0.005时取a[0]，t=0.015时取a[1] |

**作用**：生成基带信号，将离散的二进制序列转换为连续时间信号。

**2. BPSK调制**

python

|  |  |
| --- | --- |
|  | fc = 4000 # 载波频率4kHz |
|  | fs = 20\*fc # 采样率80kHz（满足奈奎斯特定理） |
|  | ts = np.arange(0, (100\*size)/fs, 1/fs) # 高精度时间轴（用于精确采样） |
|  |  |
|  | coherent\_carrier = np.cos(2\*pi\*fc\*ts) # 生成相干载波cos(2πfct) |
|  |  |
|  | # BPSK调制：0→相位+π/4，1→相位+π/4+π（实际是DQPSK的变种） |
|  | bpsk = np.cos(2\*pi\*fc\*ts + pi\*(m-1) + pi/4) |

**原理**：用相位变化表示0/1。这里有个特殊设计：

* 传统BPSK：0→0相位，1→π相位
* 此代码：0→π/4相位，1→5π/4相位（通过+pi\*(m-1)+pi/4实现）

**目的**：将数字信号加载到高频载波上，便于无线传输。

**3. 添加高斯白噪声**

python

|  |  |
| --- | --- |
|  | def awgn(y, snr): |
|  | snr = 10\*\*(snr/10) # SNR线性值转换 |
|  | xpower = np.sum(y\*\*2)/len(y) # 计算信号功率 |
|  | npower = xpower / snr # 计算噪声功率 |
|  | return np.random.randn(len(y))\*np.sqrt(npower) + y # 生成噪声并叠加 |
|  |  |
|  | noise\_bpsk = awgn(bpsk, 5) # 添加5dB信噪比的噪声 |

**作用**：模拟实际信道中的噪声干扰，SNR=5dB表示信号功率是噪声的3.16倍。

**4. 带通滤波器设计**

python

|  |  |
| --- | --- |
|  | [b11,a11] = signal.ellip(5, 0.5, 60, |
|  | [2000\*2/80000, 6000\*2/80000], |
|  | btype='bandpass') |

**参数解析**：

* 5阶椭圆滤波器
* 0.5dB通带波纹
* 60dB阻带衰减
* 通带2kHz-6kHz（归一化频率：2000*2/80000=0.05，6000*2/80000=0.15）

**作用**：滤除带外噪声，保留有效信号频段（BPSK信号能量集中在载波频率附近）。

**5. 相干解调**

python

|  |  |
| --- | --- |
|  | bandpass\_out = signal.filtfilt(b11, a11, noise\_bpsk) # 先通过带通滤波 |
|  | coherent\_demod = bandpass\_out \* (coherent\_carrier \* 2) # 与载波相乘下变频 |

**原理**：

1. 滤波去除带外噪声
2. 乘以相同频率的载波实现下变频（将高频信号搬移到基带）
3. \*2是为了补偿信号幅度衰减

**6. 低通滤波器设计**

python

|  |  |
| --- | --- |
|  | [b12,a12] = signal.ellip(5, 0.5, 60, |
|  | 2000\*2/80000, |
|  | btype='lowpass') |
|  | lowpass\_out = signal.filtfilt(b12, a12, coherent\_demod) # 提取基带信号 |

**作用**：滤除下变频后的高频分量，保留基带信号。

**7. 抽样判决**

python

|  |  |
| --- | --- |
|  | # 每个符号取100个采样点（0.01秒/80kHz采样率=800点，但代码用100点可能有误） |
|  | for i in range(size): |
|  | tempF = sum(lowpass\_out[i\*100:(i+1)\*100]) # 积分求和 |
|  | flag[i] = 1 if tempF>0 else 0 # 判决：大于0判为1，否则判为0 |
|  |  |
|  | # 重建判决后的连续信号 |
|  | detection\_bpsk = np.repeat(flag, 100) # 将判决结果扩展为连续信号 |

**原理**：在每个符号周期内对信号积分，根据积分结果的正负恢复原始二进制数据。

**三、可视化部分**

代码绘制了三个关键波形：

1. 原始基带信号（0/1方波）
2. BPSK调制信号（高频载波相位变化）
3. 加噪后的调制信号
4. 解调后的基带信号（判决结果）

**四、潜在问题说明**

1. **符号持续时间矛盾**：理论每个符号应有800个采样点（0.01秒×80kHz），但代码用100点，可能是参数设置错误。
2. **相位偏移设计**：传统BPSK不需要π/4偏移，这可能是为了实现差分编码（DQPSK），但代码未完全实现差分逻辑。
3. **判决门限**：简单使用0作为阈值，实际可能需要更复杂的门限调整机制。

**五、通俗比喻**

想象你正在用手电筒发送摩尔斯电码：

1. **基带信号**：你要发送的"SOS"（···---···）
2. **调制**：用手电筒快速开关表示点，长时间亮表示划
3. **加噪**：发送过程中有雾（噪声）干扰
4. **滤波**：用滤光镜（带通滤波器）减少雾气干扰
5. **解调**：接收方用手电筒同步闪烁（相干载波）解码
6. **判决**：根据接收到的光信号持续时间判断是点还是划

这个仿真完整展示了数字通信系统从发送到接收的全过程，是理解现代无线通信原理的基础模型。