深蓝学院
shenlanxueyuan.com

第五章作业讲评 – GCC & NLMS

主讲人 巫健

题目

- GCC（-PHAT）

输入信号和参考信号存在时延。本实验中，我们将利用输入信号和参考信号的起始段 16384 个采样点（约 1s）的数据进行时延估计（这部分数据只用来做时延估计，不会用于 AEC 的处理）。请大家实现基于 GCC 的时延估计算法，估计出输入信号相对于参考信号的时延(单位是 samples)，返回该时延的数值。TDE 函数位于文件 src/dios_ssp_tde.c 中：

```
int dios_ssp_tde(short* inputdata, short* refdata, long int inputdata_length);
```

- NLMS

NLMS 函数位于 srv/dios_ssp_aec_firfilter.c 中：

```
void nlms_complex(int ch, objFirFilter *srv, int i_ref);
```

算法流程

- GCC（-PHAT）
  - Real FFT + 向量点乘 + Real iFFT

$$\mathbf{r}_{xy} = \mathcal{F}^{-1}(\mathbf{x} \odot \mathbf{y}^*)$$

  - 准确结果是3213个点
- NLMS

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \frac{2\mu \cdot \mathbf{x}(n) \cdot e(n)^*}{\mathbf{x}(n)^H \mathbf{x}(n)}$$

算法流程

- Real FFT
  - 输入：实信号序列（假设长度为N）
  - 输出：前N/2+1个FFT变换的结果（共轭）
  - 对应关系

$$[y(0),\cdots,y(N-1)] \in \mathbb{R}^{N\times 1} \rightarrow \begin{bmatrix} y(0) & 0 \\ y(1) & -y(N-1)) \\ \cdots & \cdots \\ y(i) & -y(N-i) \\ \cdots & \cdots \\ y(N-1) & 0 \end{bmatrix} \in \mathbb{C}^{N/2+1\times 1}$$

算法流程

- Real iFFT
  - 输入：前N/2+1个FFT变换的结果（共轭）
  - 输出：长度为N的实信号序列
- 向量乘

$$\left(\begin{bmatrix} y(0) & 0 \\ y(1) & -y(N-1)) \\ \cdots & \cdots \\ y(i) & -y(N-i) \\ \cdots & \cdots \\ y(N-1) & 0 \end{bmatrix} \odot \begin{bmatrix} r(0) & 0 \\ r(1) & -r(N-1)) \\ \cdots & \cdots \\ r(i) & -r(N-i) \\ \cdots & \cdots \\ r(N-1) & 0 \end{bmatrix}^*\right)^* = \begin{bmatrix} y(0) & 0 \\ y(1) & y(N-1)) \\ \cdots & \cdots \\ y(i) & y(N-i) \\ \cdots & \cdots \\ y(N-1) & 0 \end{bmatrix} \odot \begin{bmatrix} r(0) & 0 \\ r(1) & -r(N-1)) \\ \cdots & \cdots \\ r(i) & -r(N-i) \\ \cdots & \cdots \\ r(N-1) & 0 \end{bmatrix}$$

作业问题

深蓝学院
shenlanxueyuan.com

- 同学们基本都提交了正确的结果
- 个别同学GCC中的向量乘法写的存在问题，没有和rfft的格式对应上
- 做圆周相关的话，fft size取N（原信号长度）即可
- NLMS的分母项结果是个实数
- 作业提供的AEC结果，NLMS权重更新没有乘2，所以和大家交的结果不一样

参考代码

深蓝学院 shenlanxueyuan.com

- GCC（-PHAT）

```
long int N = inputdata_length, delay = 0;

RFFT_PARAM *rfft_handle = (RFFT_PARAM*)dios_ssp_share_rfft_init(N);
float *inp = (float*)calloc(N, sizeof(float));
float *ref = (float*)calloc(N, sizeof(float));
float *inp_rfft = (float*)calloc(N, sizeof(float));
float *ref_rfft = (float*)calloc(N, sizeof(float));

for (int i = 0; i < N; i++) {
  inp[i] = inputdata[i];
  ref[i] = refdata[i];
}

dios_ssp_share_rfft_process(rfft_handle, inp, inp_rfft);
dios_ssp_share_rfft_process(rfft_handle, ref, ref_rfft);

long int h = N / 2;
float abs = 0;
float *gcc = (float*)calloc(N, sizeof(float));

gcc[0] = ref_rfft[0] * inp_rfft[0]; // 1
gcc[h] = ref_rfft[h] * inp_rfft[h]; // 1

for (int i = 1; i < h; i++) {
  gcc[i] = inp_rfft[i] * ref_rfft[i] + inp_rfft[N - i] * ref_rfft[N - i];
  gcc[N - i] = -inp_rfft[i] * ref_rfft[N - i] + inp_rfft[N - i] * ref_rfft[i];
  // abs = sqrt(gcc[i] * gcc[i] + gcc[N - i] * gcc[N - i]);
  // gcc[i] / abs;
  // gcc[N - i] / abs;
}
```

```
float *delta = (float*)calloc(N, sizeof(float));
dios_ssp_share_irfft_process(rfft_handle, gcc, delta);

float max_delta = delta[0];
for (int i = 1; i < N; i++) {
  if (delta[i] > max_delta) {
    delay = i;
    max_delta = delta[i];
  }
}
// end TDE

dios_ssp_share_rfft_uninit(rfft_handle);
free(ref);
free(inp);
free(ref_rfft);
free(inp_rfft);
free(gcc);
free(delta);

printf("delay = %d\n", delay);
return delay;
```

参考代码

- NLMS

```
xcomplex ***x = srv->stack_sigIn_adf;
int order = srv->num_main_subband_adf[ch];

// 1)
float norm = 0;
for (int i = 0; i < order; i++)
    norm += complex_abs2(x[i_ref][ch][i]);
// 2)
// float norm = srv->power_in_ntaps_smooth[i_ref][ch]

float scale = 2 * srv->weight[ch] / norm;
xcomplex alpha = complex_real_complex_mul(scale, complex_conjg(srv->err_adf[ch]));
for (int i = 0; i < order; i++) {
    srv->adf_coef[i_ref][ch][i] = complex_add(complex_mul(alpha,
        x[i_ref][ch][i]), srv->adf_coef[i_ref][ch][i]);
}
```

深蓝学院
shenlanxueyuan.com

感谢各位聆听

Thanks for Listening