在 C++ 中实现 Python 的 LogMelFilterBank（对数 Mel 滤波器组），我们需要将计算过程分为几个部分：

1. **Mel 滤波器组**
2. **计算功率谱**
3. **应用对数变换**

以下是详细的分步实现：

Python 的 LogMelFilterBank 工作原理

1. **输入**：
   - 信号的 STFT 或频谱功率
   - 滤波器组配置（数量、采样率等）
2. **输出**：
   - 使用 Mel 滤波器加权后的对数 Mel 能量谱

关键步骤：

- 创建 Mel 滤波器组（如 librosa.filters.mel）。
- 将频谱能量分布映射到 Mel 频率空间。
- 取对数。

C++ 实现

1. Mel 滤波器组

我们使用三角滤波器来构造 Mel 滤波器组，与前述实现一致。
以下函数生成 Mel 滤波器组矩阵：

```
#include <vector>
#include <cmath>

// 线性频率到 Mel 频率
double hz_to_mel(double hz) {
    return 2595.0 * std::log10(1.0 + hz / 700.0);
}

// Mel 频率到线性频率
double mel_to_hz(double mel) {
    return 700.0 * (std::pow(10.0, mel / 2595.0) - 1.0);
}

// 生成 Mel 滤波器
std::vector<std::vector<double>> mel_filterbank(int n_filters, int n_fft, double sample_rate, double f_min, double f_max) {
    int fft_bins = n_fft / 2 + 1;
```

```
    double bin_freq_step = sample_rate
/ n_fft;

    // Mel 频率点
    std::vector<double>
mel_points(n_filters + 2);
    double mel_min = hz_to_mel(f_min);
    double mel_max = hz_to_mel(f_max);

    for (int i = 0; i < n_filters + 2;
i++) {
        mel_points[i] =
mel_to_hz(mel_min + (mel_max -
mel_min) * i / (n_filters + 1));
    }

    // 滤波器组矩阵
    std::vector<std::vector<double>>
filterbank(n_filters,
std::vector<double>(fft_bins, 0.0));

    for (int i = 1; i <= n_filters; i+
+) {
        double left = mel_points[i -
1];
        double center = mel_points[i];
        double right = mel_points[i +
1];
```

```
            for (int j = 0; j < fft_bins;
j++) {
                double freq = j *
bin_freq_step;

                if (freq >= left && freq
<= center) {
                    filterbank[i - 1][j] =
(freq - left) / (center - left);
                } else if (freq > center
&& freq <= right) {
                    filterbank[i - 1][j] =
(right - freq) / (right - center);
                }
            }
        }

        return filterbank;
}
```

2. 功率谱计算

输入信号通常是 STFT 的结果（频率-时间矩阵）。我们需要计算功率谱。

```
#include <complex>

// 计算功率谱
std::vector<double>
compute_power_spectrum(const
```

```
std::vector<std::complex<double>>&
stft_frame) {
    std::vector<double>
power_spectrum(stft_frame.size());
    for (size_t i = 0; i <
stft_frame.size(); i++) {
        power_spectrum[i] =
std::norm(stft_frame[i]); // |x|^2
    }
    return power_spectrum;
}
```

3. 应用 Mel 滤波器组

将功率谱通过 Mel 滤波器组加权映射。

```
std::vector<double>
apply_mel_filterbank(const
std::vector<std::vector<double>>&
filterbank, const std::vector<double>&
power_spectrum) {
    std::vector<double>
mel_energies(filterbank.size(), 0.0);

    for (size_t i = 0; i <
filterbank.size(); i++) {
        for (size_t j = 0; j <
power_spectrum.size(); j++) {
```

```
            mel_energies[i] +=
filterbank[i][j] * power_spectrum[j];
        }
    }

    return mel_energies;
}
```

4. 对数变换

对 Mel 滤波器组输出取对数。

```
#include <algorithm>

std::vector<double>
log_mel_energies(const
std::vector<double>& mel_energies,
double epsilon = 1e-6) {
    std::vector<double>
log_mel(mel_energies.size());
    for (size_t i = 0; i <
mel_energies.size(); i++) {
        log_mel[i] =
std::log(std::max(mel_energies[i],
epsilon)); // 避免 log(0)
    }
    return log_mel;
}
```

5. 组合成 LogMelFilterBank

完整函数将各部分组合在一起：

```
#include <vector>
#include <complex>

std::vector<double>
log_mel_filterbank(const
std::vector<std::complex<double>>&
stft_frame,

int n_filters,

int n_fft,

double sample_rate,

double f_min = 0.0,

double f_max = -1.0) {
    if (f_max < 0.0) {
        f_max = sample_rate / 2.0;
    }

    // 1. 创建 Mel 滤波器组
    std::vector<std::vector<double>>
filterbank = mel_filterbank(n_filters,
n_fft, sample_rate, f_min, f_max);

    // 2. 计算功率谱
```

```
    std::vector<double> power_spectrum
= compute_power_spectrum(stft_frame);

    // 3. 应用 Mel 滤波器组
    std::vector<double> mel_energies =
apply_mel_filterbank(filterbank,
power_spectrum);

    // 4. 取对数
    return
log_mel_energies(mel_energies);
}
```

使用示例

```
#include <iostream>
#include <complex>

int main() {
    // 模拟一个 STFT 帧
    std::vector<std::complex<double>>
stft_frame = {{1.0, 0.5}, {0.8, 0.3},
{0.6, 0.1}, {0.3, 0.0}};

    // 参数
    int n_filters = 10;
    int n_fft = 8;
    double sample_rate = 16000.0;

    // 计算 Log Mel 滤波器组
```

```
    std::vector<double> log_mel =
log_mel_filterbank(stft_frame,
n_filters, n_fft, sample_rate);

    // 输出结果
    for (double value : log_mel) {
        std::cout << value << " ";
    }
    return 0;
}
```

扩展和优化

1. **优化性能**：
   - 使用 SIMD 加速滤波器计算。
   - 合并计算流程避免多次遍历。
2. **FFT 支持**：
   - 若需要从时域信号计算 STFT，可使用 FFT 库（如 FFTW）。
3. **动态参数支持**：
   - 添加灵活接口以支持动态采样率、频率范围。

通过这些步骤，您即可在 C++ 中高效实现类似 Python 的 LogMelFilterBank 功能！