基于 N 点 IDFT（IFFT）计算 2N 点 IMDCT 的方法

起草：季文骢，版本：16

第一部分：概述

N 点实序列 $\hat{X}(k)$（$0 \le k < N$）的 2N 点 IMDCT（Inverse Modified Discrete Cosine Transform）的定义式如下：

$$\hat{x}(n) = \frac{1}{N}\sum_{k=0}^{N-1} \hat{X}(k) \cos\left(\frac{\pi}{N}\cdot\left(n+\frac{1}{2}+\frac{N}{2}\right)\cdot\left(k+\frac{1}{2}\right)\right) \quad (0 \le n < 2N)$$

在实际应用中，一种常见的做法是采用基于 2N 点 IDFT（IFFT）的算法来计算 2N 点 IMDCT[1]。下面给出一种时间复杂度更低的基于 N 点 IDFT（IFFT）的计算 IMDCT 输出序列 $\hat{x}(n)$ 的方法及其正确性证明。

另外，文中所有的定义和推导过程都默认各种序列的序号是整数，因此对于序列的序号，本文仅给出其范围而不再强调其为整数这一事实。本文虚部单位用符号 j 表示，对于任意复数 $x = a + bj$，取实部函数 $\mathrm{Re}\{x\}$ 定义为 $\mathrm{Re}\{x\} = a$，取虚部函数 $\mathrm{Im}\{x\}$ 定义为 $\mathrm{Im}\{x\} = b$。

第二部分：算法流程及正确性证明

定义记号 $\gamma(k) = \cos\left(\frac{\pi}{N}\cdot\left(n+\frac{1}{2}+\frac{N}{2}\right)\cdot\left(k+\frac{1}{2}\right)\right)$，先证明序列 $\gamma(k)$ 对于 $0 \le k < 2N$，有 $\gamma(k) = (-1)^{N+1}\cdot\gamma(2N-1-k)$。

证明：

$$\begin{aligned}\gamma(k) &= \cos\left(\frac{\pi}{N}\cdot\left(n+\frac{1}{2}+\frac{N}{2}\right)\cdot\left(k+\frac{1}{2}\right)\right)\\ &= \mathrm{Re}\left\{e^{\pm j\frac{\pi}{N}\cdot\left(n+\frac{1}{2}+\frac{N}{2}\right)\cdot\left(k+\frac{1}{2}\right)}\right\}\end{aligned}$$

将 k、$2N-1-k$ 依次代入上式，自然底数 e 的指数部分的正负号依次取 − 和 +，则有：

$$\gamma(k) = \mathrm{Re}\left\{e^{j\frac{\pi}{N}\cdot\left(n+\frac{1}{2}+\frac{N}{2}\right)\cdot\left(-k-\frac{1}{2}\right)}\right\}$$

$$\begin{aligned}\gamma(2N-1-k) &= \mathrm{Re}\left\{e^{j\frac{\pi}{N}\cdot\left(n+\frac{1}{2}+\frac{N}{2}\right)\cdot\left(2N-k-\frac{1}{2}\right)}\right\}\\ &= \mathrm{Re}\left\{e^{j\frac{\pi}{N}\cdot\left(n+\frac{1}{2}+\frac{N}{2}\right)\cdot\left(-k-\frac{1}{2}\right)}e^{j\frac{\pi}{N}\cdot\left(n+\frac{1}{2}+\frac{N}{2}\right)\cdot 2N}\right\}\end{aligned}$$

比较二式，不难发现二者的区别在于 $e^{j\frac{\pi}{N}\cdot\left(n+\frac{1}{2}+\frac{N}{2}\right)\cdot 2N}$ 这一部分，那么单独考察该部分：

$$\begin{aligned}e^{j\frac{\pi}{N}\cdot\left(n+\frac{1}{2}+\frac{N}{2}\right)\cdot 2N} &= e^{j2\pi n}e^{j\pi(N+1)}\\ &= \left(e^{j2\pi}\right)^n\left(e^{j\pi}\right)^{N+1}\\ &= (1)^n(-1)^{N+1}\\ &= (-1)^{N+1}\end{aligned}$$

由此即可推出 $\gamma(k)$、$\gamma(2N-1-k)$ 之间的关系：

$$\gamma(2N-1-k) = \mathrm{Re}\left\{e^{j\frac{\pi}{N}\cdot\left(n+\frac{1}{2}+\frac{N}{2}\right)\cdot\left(-k-\frac{1}{2}\right)}\cdot(-1)^{N+1}\right\}$$

$$= (-1)^{N+1} \cdot \mathrm{Re}\left\{e^{j\frac{\pi}{N}\left(n+\frac{1}{2}+\frac{N}{2}\right)\cdot\left(-k-\frac{1}{2}\right)}\right\}$$
$$= (-1)^{N+1} \cdot \gamma(k)$$

证毕。□

构造 N 点实序列 $\hat{X}(k)$ 的 $2N$ 点延扩序列 $\widehat{X_p}(k)$（其中 $0 \le k < 2N$，显然该序列也是实序列）：

$$\widehat{X_p}(k) = \begin{cases} \hat{X}(k) & (0 \le k < N) \\ (-1)^{N+1} \cdot \hat{X}(2N-1-k) & (N \le k < 2N) \end{cases}$$

那么便有：

$$\frac{1}{N}\sum_{k=N}^{2N-1} \widehat{X_p}(k)\gamma(k) = \frac{1}{N}\sum_{k=N}^{2N-1} (-1)^{N+1} \cdot \hat{X}(2N-1-k) \cdot (-1)^{N+1} \cdot \gamma(2N-1-k)$$
$$= \frac{1}{N}\sum_{k=N}^{2N-1} \hat{X}(2N-1-k)\gamma(2N-1-k)$$
$$= \frac{1}{N}\sum_{k=0}^{N-1} \hat{X}(k)\gamma(k)$$

修改上式的求和范围便有：

$$\frac{1}{N}\sum_{k=0}^{2N-1} \widehat{X_p}(k)\gamma(k) = \frac{1}{N}\sum_{k=0}^{N-1} \widehat{X_p}(k)\gamma(k) + \frac{1}{N}\sum_{k=N}^{2N-1} \widehat{X_p}(k)\gamma(k)$$
$$= \frac{1}{N}\sum_{k=0}^{N-1} \hat{X}(k)\gamma(k) + \frac{1}{N}\sum_{k=0}^{N-1} \hat{X}(k)\gamma(k)$$
$$= 2\hat{x}(n)$$

由此便得到了基于延扩序列 $\widehat{X_p}(k)$ 的 $\hat{x}(n)$ 表达式：

$$\hat{x}(n) = \frac{1}{2N}\sum_{k=0}^{2N-1} \widehat{X_p}(k)\gamma(k)$$
$$= \frac{1}{2N}\sum_{k=0}^{2N-1} \widehat{X_p}(k)\cos\left(\frac{\pi}{N}\cdot\left(n+\frac{1}{2}+\frac{N}{2}\right)\cdot\left(k+\frac{1}{2}\right)\right)$$

将 $\hat{x}(n)$ 展开：

$$\hat{x}(n) = \frac{1}{2N}\sum_{k=0}^{2N-1} \widehat{X_p}(k)\cos\left(\frac{\pi}{N}\left(n+\frac{1}{2}+\frac{N}{2}\right)k + \frac{\pi}{2N}\left(n+\frac{1}{2}+\frac{N}{2}\right)\right)$$

定义 $\alpha(n,k) = \frac{\pi}{N}\left(n+\frac{1}{2}+\frac{N}{2}\right)k$，$w(n) = \frac{\pi}{2N}\left(n+\frac{1}{2}+\frac{N}{2}\right)$，那么：

$$\hat{x}(n) = \frac{1}{2N}\sum_{k=0}^{2N-1} \widehat{X_p}(k)\cos\big(\alpha(n,k) + w(n)\big)$$

$$= \frac{1}{2N}\sum_{k=0}^{2N-1} \widehat{X_p}(k)\big(\cos\big(\alpha(n,k)\big)\cos\big(w(n)\big) - \sin\big(\alpha(n,k)\big)\sin\big(w(n)\big)\big)$$

$$= \cos\big(w(n)\big)\cdot \overbrace{\frac{1}{2N}\sum_{k=0}^{2N-1} \widehat{X_p}(k)\cos\big(\alpha(n,k)\big)}^{T_1(n)} - \sin\big(w(n)\big)\cdot \overbrace{\frac{1}{2N}\sum_{k=0}^{2N-1} \widehat{X_p}(k)\sin\big(\alpha(n,k)\big)}^{T_2(n)}$$

定义序列 $T_1(n)$、$T_2(n)$（$0 \le n < 2N$）：

$$T_1(n) = \frac{1}{2N}\sum_{k=0}^{2N-1} \widehat{X_p}(k)\cos\big(\alpha(n,k)\big)$$

$$T_2(n) = \frac{1}{2N}\sum_{k=0}^{2N-1} \widehat{X_p}(k)\sin\big(\alpha(n,k)\big)$$

显然有 $\hat{x}(n) = \cos\big(w(n)\big)T_1(n) - \sin\big(w(n)\big)T_2(n)$，继续展开 $T_1(n)$、$T_2(n)$：

$$e^{j\alpha(n,k)} = \cos\big(\alpha(n,k)\big) + j\cdot\sin\big(\alpha(n,k)\big)$$
$$e^{-j\alpha(n,k)} = \cos\big(\alpha(n,k)\big) - j\cdot\sin\big(\alpha(n,k)\big)$$

$$T_1(n) = \frac{1}{2N}\sum_{k=0}^{2N-1} \widehat{X_p}(k)\frac{e^{j\alpha(n,k)} + e^{-j\alpha(n,k)}}{2}$$

$$= \frac{1}{2}\left(\frac{1}{2N}\sum_{k=0}^{2N-1} \widehat{X_p}(k)e^{j\alpha(n,k)} + \frac{1}{2N}\sum_{k=0}^{2N-1} \widehat{X_p}(k)e^{-j\alpha(n,k)}\right)$$

$$T_2(n) = \frac{1}{2N}\sum_{k=0}^{2N-1} \widehat{X_p}(k)\frac{e^{j\alpha(n,k)} - e^{-j\alpha(n,k)}}{2j}$$

$$= \frac{1}{2j}\left(\frac{1}{2N}\sum_{k=0}^{2N-1} \widehat{X_p}(k)e^{j\alpha(n,k)} - \frac{1}{2N}\sum_{k=0}^{2N-1} \widehat{X_p}(k)e^{-j\alpha(n,k)}\right)$$

定义序列 $A(n)$ 及其共轭序列 $A^*(n)$（$0 \le n < 2N$）：

$$A(n) = \frac{1}{2N}\sum_{k=0}^{2N-1} \widehat{X_p}(k)e^{j\alpha(n,k)}$$

$$A^*(n) = \big(A(n)\big)^*$$

$$= \frac{1}{2N}\sum_{k=0}^{2N-1} \left(\widehat{X_p}(k)\right)^* \left(e^{j\alpha(n,k)}\right)^*$$

$$= \frac{1}{2N}\sum_{k=0}^{2N-1} \widehat{X_p}(k)e^{-j\alpha(n,k)}$$

于是可用 $T_1(n)$、$T_2(n)$ 就可以用序列 $A(n)$ 及共轭序列 $A^*(n)$ 来表示：

$$T_1(n) = \frac{1}{2}\big(A(n) + A^*(n)\big)$$

$$T_2(n) = \frac{1}{2j}\left(A(n) - A^*(n)\right)$$

定义序列 $\widehat{X_m}(k)$（其中 $0 \le k < 2N$）：

$$\widehat{X_m}(k) = \widehat{X_p}(k)e^{j\pi\left|\frac{k}{2}\right|}$$

$$= (-1)^{\left|\frac{k}{2}\right|} \cdot \widehat{X_p}(k)$$

将序列 $\widehat{X_m}(k)$ 按自变量 k 的奇偶性进行分解（下二式中 $0 \le k < N$）：

$$\widehat{X_m}(2k) = \widehat{X_p}(2k)e^{j\pi k}$$
$$= (-1)^k \cdot \widehat{X_p}(2k)$$
$$\widehat{X_m}(2k+1) = \widehat{X_p}(2k+1)e^{j\pi k}$$
$$= (-1)^k \cdot \widehat{X_p}(2k+1)$$

定义序列 $Z(k)$ 及其共轭序列 $Z^*(k)$（其中 $0 \le k < N$）：

$$Z(k) = \frac{1}{4}\left(\widehat{X_m}(2k) + j \cdot \widehat{X_m}(2k+1)\right)e^{j\frac{\pi}{N}k}$$

$$Z^*(k) = \left(Z(k)\right)^*$$

$$= \frac{1}{4}\left(\widehat{X_m}(2k) - j \cdot \widehat{X_m}(2k+1)\right)e^{-j\frac{\pi}{N}k}$$

于是则有：

$$Z(k) + Z^*(k)e^{j\frac{2\pi}{N}k} = \frac{1}{2}\widehat{X_m}(2k)e^{j\frac{\pi}{N}k}$$

$$Z(k) - Z^*(k)e^{j\frac{2\pi}{N}k} = j \cdot \frac{1}{2}\widehat{X_m}(2k+1)e^{j\frac{\pi}{N}k}$$

对序列 $Z(k)$ 进行 N 点 IDFT，得到序列 $z(n)$ 及其共轭序列 $z^*(n)$（其中 $0 \le n < N$）：

$$z(n) = \frac{1}{N}\sum_{k=0}^{N-1} Z(k)e^{j\frac{2\pi}{N}kn}$$

$$z^*(n) = \left(z(n)\right)^*$$

$$= \frac{1}{N}\sum_{k=0}^{N-1} Z^*(k)e^{-j\frac{2\pi}{N}kn}$$

将 $N-n-1$ 作为自变量代入 $z^*(n)$：

$$z^*(N-n-1) = \frac{1}{N}\sum_{k=0}^{N-1} Z^*(k)e^{-j\frac{2\pi}{N}k(N-n-1)}$$

$$= \frac{1}{N}\sum_{k=0}^{N-1} Z^*(k)e^{j\frac{2\pi}{N}k(n+1)}$$

$$= \frac{1}{N}\sum_{k=0}^{N-1} \left(Z^*(k)e^{j\frac{2\pi}{N}k}\right)e^{j\frac{2\pi}{N}kn}$$

那么就有：

$$z(n)+z^*(N-n-1)=\frac{1}{N}\sum_{k=0}^{N-1}Z(k)e^{j\frac{2\pi}{N}kn}+\frac{1}{N}\sum_{k=0}^{N-1}\left(Z^*(k)e^{j\frac{2\pi}{N}k}\right)e^{j\frac{2\pi}{N}kn}$$

$$=\frac{1}{N}\sum_{k=0}^{N-1}\left(Z(k)+Z^*(k)e^{j\frac{2\pi}{N}k}\right)e^{j\frac{2\pi}{N}kn}$$

$$=\frac{1}{2N}\sum_{k=0}^{N-1}\left(\widehat{X_m}(2k)e^{j\frac{\pi}{N}k}\right)e^{j\frac{2\pi}{N}kn}$$

$$z(n)-z^*(N-n-1)=\frac{1}{N}\sum_{k=0}^{N-1}Z(k)e^{j\frac{2\pi}{N}kn}-\frac{1}{N}\sum_{k=0}^{N-1}\left(Z^*(k)e^{j\frac{2\pi}{N}k}\right)e^{j\frac{2\pi}{N}kn}$$

$$=\frac{1}{N}\sum_{k=0}^{N-1}\left(Z(k)-Z^*(k)e^{j\frac{2\pi}{N}k}\right)e^{j\frac{2\pi}{N}kn}$$

$$=j\cdot\frac{1}{2N}\sum_{k=0}^{N-1}\left(\widehat{X_m}(2k+1)e^{j\frac{\pi}{N}k}\right)e^{j\frac{2\pi}{N}kn}$$

考虑当 $0\le n<N$ 时 $A(n)$ 的值：

$$A(n)=\frac{1}{2N}\sum_{k=0}^{2N-1}\widehat{X_p}(k)e^{j\frac{\pi}{N}\left(n+\frac{1}{2}+\frac{N}{2}\right)k}$$

$$=\overbrace{\frac{1}{2N}\sum_{k=0}^{N-1}\widehat{X_p}(2k)e^{j\frac{\pi}{N}\left(n+\frac{1}{2}+\frac{N}{2}\right)\cdot 2k}}^{A_{even}(n)}+\overbrace{\frac{1}{2N}\sum_{k=0}^{N-1}\widehat{X_p}(2k+1)e^{j\frac{\pi}{N}\left(n+\frac{1}{2}+\frac{N}{2}\right)(2k+1)}}^{A_{odd}(n)}$$

$$=A_{even}(n)+A_{odd}(n)$$

分别对 $A_{even}(n)$ 和 $A_{odd}(n)$ 进行展开，有：

$$A_{even}(n)=\frac{1}{2N}\sum_{k=0}^{N-1}\widehat{X_p}(2k)e^{j\frac{\pi}{N}\left(n+\frac{1}{2}+\frac{N}{2}\right)\cdot 2k}$$

$$=\frac{1}{2N}\sum_{k=0}^{N-1}\left(\left(\widehat{X_p}(2k)e^{j\pi k}\right)e^{j\frac{\pi}{N}k}\right)e^{j\frac{2\pi}{N}kn}$$

$$=\frac{1}{2N}\sum_{k=0}^{N-1}\left(\widehat{X_m}(2k)e^{j\frac{\pi}{N}k}\right)e^{j\frac{2\pi}{N}kn}$$

$$=z(n)+z^*(N-n-1)$$

$$A_{odd}(n)=\frac{1}{2N}\sum_{k=0}^{N-1}\widehat{X_p}(2k+1)e^{j\frac{\pi}{N}\left(n+\frac{1}{2}+\frac{N}{2}\right)(2k+1)}$$

$$=\frac{1}{2N}e^{j\frac{\pi}{N}\left(n+\frac{1}{2}+\frac{N}{2}\right)}\sum_{k=0}^{N-1}\left(\left(\widehat{X_p}(2k+1)e^{j\pi k}\right)e^{j\frac{\pi}{N}k}\right)e^{j\frac{2\pi}{N}kn}$$

$$= \frac{e^{j\frac{\pi}{N}\left(n+\frac{1}{2}+\frac{N}{2}\right)}}{j}\Big(z(n) - z^*(N-n-1)\Big)$$

$$= e^{j\frac{\pi}{N}\left(n+\frac{1}{2}\right)}\Big(z(n) - z^*(N-n-1)\Big)$$

再考虑当 $0 \le n < N$ 时 $A(n+N)$ 的值：

$$A(n+N) = \frac{1}{2N}\sum_{k=0}^{2N-1} \widehat{X_p}(k) e^{j\frac{\pi}{N}\left(n+N+\frac{1}{2}+\frac{N}{2}\right)k}$$

$$= \frac{1}{2N}\sum_{k=0}^{2N-1} \widehat{X_p}(k) e^{j\frac{\pi}{N}\left(n+\frac{1}{2}+\frac{N}{2}\right)k} e^{j\pi k}$$

$$= \frac{1}{2N}\sum_{k=0}^{2N-1} \widehat{X_p}(k) e^{j\frac{\pi}{N}\left(n+\frac{1}{2}+\frac{N}{2}\right)k} (-1)^k$$

$$= \overbrace{\frac{1}{2N}\sum_{k=0}^{N-1} \widehat{X_p}(2k) e^{j\frac{\pi}{N}\left(n+\frac{1}{2}+\frac{N}{2}\right)\cdot 2k}}^{A_{even}(n)} - \overbrace{\frac{1}{2N}\sum_{k=0}^{N-1} \widehat{X_p}(2k+1) e^{j\frac{\pi}{N}\left(n+\frac{1}{2}+\frac{N}{2}\right)(2k+1)}}^{A_{odd}(n)}$$

$$= A_{even}(n) - A_{odd}(n)$$

到此为止，对于 $0 \le n < 2N$，所有的 $A(n)$ 都已被求出，由此便可推出 $T_1(n)$、$T_2(n)$ 序列并得到最终结果 $\hat{x}(n)$：

$$T_1(n) = \frac{1}{2}\Big(A(n) + A^*(n)\Big)$$
$$= \mathrm{Re}\{A(n)\}$$
$$T_2(n) = \frac{1}{2j}\Big(A(n) - A^*(n)\Big)$$
$$= \mathrm{Im}\{A(n)\}$$
$$\hat{x}(n) = \cos\big(w(n)\big)\,T_1(n) - \sin\big(w(n)\big)\,T_2(n)$$
$$= \cos\big(w(n)\big)\,\mathrm{Re}\{A(n)\} - \sin\big(w(n)\big)\,\mathrm{Im}\{A(n)\}$$

<u>第三部分：总结</u>

最后总结基于 N 点 IDFT 计算序列 $\hat{x}(n)$ 的值的流程（IDFT 过程可采用 IFFT 来实现）：

第一步：构造延扩序列 $\widehat{X_p}(k)$ 及 $\widehat{X_m}(k)$（$0 \le k < 2N$）；

第二步：构造序列 $Z(k)$（$0 \le k < N$）并计算其 N 点 IDFT，得到序列 $z(n)$（$0 \le n < N$）；

第三步：计算 $0 \le n < N$ 时 $A(n)$ 及 $A(n+N)$ 的值，由此构造序列 $A(n)$（$0 \le n < 2N$）；

第四步：利用序列 $A(n)$ 构造 IMDCT 输出序列 $\hat{x}(n)$。

算法伪代码如下（输入为数组 X[]，输出为数组 x[]）：

```
//  Xp[0...2N - 1]:
if (N 是偶数) {
   Xp_factor = -1;
} else {
   Xp_factor = 1;
}
for (k = 0; k < N; ++k) {
   Xp[k] = X[k];
   Xp[2N - 1 - k] = Xp_factor * X[k];
```

```
}

//  Xm[0...2N - 1]:
Xm_factor = 1;
for (k = 0; k < N; ++k) {
   Xm[2k] = Xm_factor * Xp[2k];
   Xm[2k + 1] = Xm_factor * Xp[2k + 1];
   Xm_factor = -Xm_factor;
}

//  Z[0...N - 1], z[0...N - 1]:
for (k = 0; k < N; ++k) {
   Z[k] = 0.25 * (Xm[2k] + 1j * Xm[2k + 1]) * pow(E, 1j * k * PI / N);
}
z = IDFT(Z);

//  A[0...2N - 1]:
for (n = 0; n < N; ++n) {
   A_even = z[n] + z[N - n - 1].conjugate();
   A_odd = pow(E, 1j * (n + 0.5) * PI / N) * (z[n] - z[N - n - 1].conjugate());
   A[n] = A_even + A_odd;
   A[n + N] = A_even - A_odd;
}

//  x[0...2N]:
for (n = 0; n < 2N; ++n) {
   w = (n + 0.5 + 0.5N) * PI / 2N;
   x[n] = cos(w) * re(A[n]) - sin(w) * im(A[n]);
}
```

不难注意到 x[n] 的值仅依赖于 A[n] 的值，因此数组 A[] 实际上是不必需的，完全可以把上面的伪代码的后半部分简化为（这样也省去了一次长度为 2N 的遍历）：

```
...
//  A[0...2N - 1], x[0...2N]:
for (n = 0; n < N; ++n) {
   A_even = z[n] + z[N - n - 1].conjugate();
   A_odd = pow(E, 1j * (n + 0.5) * PI / N) * (z[n] - z[N - n - 1].conjugate());
   u = A_even + A_odd;
   w = (n + 0.5 + 0.5N) * PI / 2N;
   x[n] = cos(w) * re(u) - sin(w) * im(u);
   u = A_even - A_odd;
   w = ((n + N) + 0.5 + 0.5N) * PI / 2N;
   x[n + N] = cos(w) * re(u) - sin(w) * im(u);
}
```

<u>第四部分：性能分析</u>

本部分对三种不同的计算 IMDCT 的算法的性能进行了分析和评测，第一种是直接按公式计算，第二种是基于 2N 点 IFFT 的算法[1]，最后一种是本文提出的基于 N 点 IFFT 的算法。

首先进行理论分析，直接按公式进行计算的算法时间复杂度为 $O(N^2)$，基于 N 点和 2N 点 IFFT 的算法时间复杂度都为 $O(N \cdot \log_2 N)$，但注意到：

$$2N \cdot \log_2 2N = (2N + 1) \log_2 N > N \cdot \log_2 N$$

因此基于 N 点 IFFT 的算法相比于基于 2N 点的算法来说迭代次数要少很多。

下表列出了不同数据规模下各种算法的计算时间，测试程序均采用纯 JavaScript 语言编写，表中列出的时间为重复计算同一数据 10000 次的耗时。对于较大的数据规模，直接按 IMDCT 定义式计算所需的时间过长，因此不予列出。

测试程序针对 N 是 2 的整次幂的情况采用 Cooley-Tukey 算法计算 IFFT，否则采用 Bluestein 算法[2]进行计算。测试程序运行的硬件环境采用 Intel(R) Core(TM) i7-6700 CPU、DDR4 32GB 2133MT/s 内存，操作系统为 Ubuntu Linux 18.04.2 桌面版（内核版本 5.4.0），JavaScript 运行时为 Node.JS v14.16.0。

N=100（非 2 的整次幂，IFFT 采用 Bluestein 算法实现）					
直接计算	3903 ms	3851 ms	3737 ms	3944 ms	3823 ms
2N 点 IFFT	265 ms	265 ms	268 ms	264 ms	267 ms
N 点 IFFT	150 ms	147 ms	147 ms	150 ms	146 ms
N=300（非 2 的整次幂，IFFT 采用 Bluestein 算法实现）					
2N 点 IFFT	1151 ms	1114 ms	1113 ms	1103 ms	1102 ms
N 点 IFFT	602 ms	588 ms	580 ms	586 ms	586 ms
N=512					
2N 点 IFFT	279 ms	281 ms	282 ms	279 ms	275 ms
N 点 IFFT	200 ms	196 ms	193 ms	196 ms	197 ms
N=1024					
2N 点 IFFT	593 ms	565 ms	568 ms	576 ms	569 ms
N 点 IFFT	387 ms	381 ms	399 ms	390 ms	382 ms
N=2048					
2N 点 IFFT	1231 ms	1192 ms	1187 ms	1183 ms	1188 ms
N 点 IFFT	802 ms	799 ms	788 ms	795 ms	792 ms
N=4096					
2N 点 IFFT	2617 ms	2526 ms	2480 ms	2581 ms	2565 ms
N 点 IFFT	1637 ms	1630 ms	1624 ms	1642 ms	1637 ms

在本文所述的测试环境下，基于 N 点 IFFT 的算法的运行耗时较 2N 点的算法少约 30%~40%。考虑到 MDCT/IMDCT 的主要应用场景是音视频的编解码，而 IMDCT 一般是用于解码器中，因此采用本文所述的算法大概率可以有效提高解码器部分的工作效率。

附录 1：基于 DFT（FFT）的算法变形

在很多的实现中，IFFT 的过程往往是通过 FFT 来进行的[3]，而这样的 IFFT 实现方式往往需要时间复杂度为 $O(N)$ 的数据预处理和后处理，相比于直接进行 FFT 操作来说会带来不少的性能损耗。因此此处给出一种基于 DFT（FFT）的变种算法。

首先考虑到有 $e^{-j\pi k} = e^{j\pi k} = (-1)^k$，因此可以断言在不修改序列 $\widehat{X_m}(k)$ 的定义的情况下，下二式对序列 $\widehat{X_m}(k)$ 也成立（下二式中 $0 \le k < N$）：

$$\widehat{X_m}(2k) = \widehat{X_p}(2k)e^{-j\pi k}$$

$$\widehat{X_m}(2k+1) = \widehat{X_p}(2k+1)e^{-j\pi k}$$

定义序列 $m(k)$ 及其共轭序列 $m^*(k)$（$0 \le k < N$）：

$$m(k) = \frac{1}{4N}\left(\widehat{X_m}(2k) + j \cdot \widehat{X_m}(2k+1)\right)e^{-j\frac{\pi}{N}k}$$

$$m^*(k) = \left(m(k)\right)^*$$

$$= \frac{1}{4N}\left(\widehat{X_m}(2k) - j \cdot \widehat{X_m}(2k+1)\right)e^{j\frac{\pi}{N}k}$$

于是则有：

$$m(k) + m^*(k)e^{-j\frac{2\pi}{N}k} = \frac{1}{2N}\widehat{X_m}(2k)e^{-j\frac{\pi}{N}k}$$

$$m(k)-m^*(k)e^{-j\frac{2\pi}{N}k}=j\cdot\frac{1}{2N}\widehat{X_m}(2k+1)e^{-j\frac{\pi}{N}k}$$

对序列 $m(k)$ 进行 N 点 DFT，得到序列 $M(n)$ 及其共轭序列 $M^*(n)$（其中 $0\le n<N$）：

$$M(n)=\sum_{k=0}^{N-1}m(k)e^{-j\frac{2\pi}{N}kn}$$

$$M^*(n)=\big(M(n)\big)^*$$

$$=\sum_{k=0}^{N-1}m^*(k)e^{j\frac{2\pi}{N}kn}$$

将 $N-n-1$ 作为自变量代入 $M^*(n)$：

$$M^*(N-n-1)=\sum_{k=0}^{N-1}m^*(k)e^{j\frac{2\pi}{N}k(N-n-1)}$$

$$=\sum_{k=0}^{N-1}\left(m^*(k)e^{-j\frac{2\pi}{N}k}\right)e^{-j\frac{2\pi}{N}kn}$$

于是则有：

$$M(n)+M^*(N-n-1)=\sum_{k=0}^{N-1}\left(m(k)+m^*(k)e^{-j\frac{2\pi}{N}k}\right)e^{-j\frac{2\pi}{N}kn}$$

$$=\frac{1}{2N}\sum_{k=0}^{N-1}\left(\widehat{X_m}(2k)e^{-j\frac{\pi}{N}k}\right)e^{-j\frac{2\pi}{N}kn}$$

$$M(n)-M^*(N-n-1)=\sum_{k=0}^{N-1}\left(m(k)-m^*(k)e^{-j\frac{2\pi}{N}k}\right)e^{-j\frac{2\pi}{N}kn}$$

$$=j\cdot\frac{1}{2N}\sum_{k=0}^{N-1}\left(\widehat{X_m}(2k+1)e^{-j\frac{\pi}{N}k}\right)e^{-j\frac{2\pi}{N}kn}$$

考虑当 $0\le n<N$ 时 $A^*(n)$ 的值：

$$A^*(n)=\frac{1}{2N}\sum_{k=0}^{2N-1}\widehat{X_p}(k)e^{-j\frac{\pi}{N}\left(n+\frac{1}{2}+\frac{N}{2}\right)k}$$

$$=\overbrace{\frac{1}{2N}\sum_{k=0}^{N-1}\widehat{X_p}(2k)e^{-j\frac{\pi}{N}\left(n+\frac{1}{2}+\frac{N}{2}\right)2k}}^{A^*_{even}(n)}+\overbrace{\frac{1}{2N}\sum_{k=0}^{N-1}\widehat{X_p}(2k+1)e^{-j\frac{\pi}{N}\left(n+\frac{1}{2}+\frac{N}{2}\right)(2k+1)}}^{A^*_{odd}(n)}$$

$$=A^*_{even}(n)+A^*_{odd}(n)$$

分别对 $A^*_{even}(n)$ 和 $A^*_{odd}(n)$ 进行展开，有：

$$A^*_{even}(n)=\frac{1}{2N}\sum_{k=0}^{N-1}\widehat{X_p}(2k)e^{-j\frac{\pi}{N}\left(n+\frac{1}{2}+\frac{N}{2}\right)2k}$$

$$= \frac{1}{2N}\sum_{k=0}^{N-1}\left(\left(\widehat{X_p}(2k)e^{-j\pi k}\right)e^{-j\frac{\pi}{N}k}\right)e^{-j\frac{2\pi}{N}kn}$$

$$= \frac{1}{2N}\sum_{k=0}^{N-1}\left(\widehat{X_m}(2k)e^{-j\frac{\pi}{N}k}\right)e^{-j\frac{2\pi}{N}kn}$$

$$= M(n) + M^*(N-n-1)$$

$$A_{odd}^*(n) = \frac{1}{2N}\sum_{k=0}^{N-1}\widehat{X_p}(2k+1)e^{-j\frac{\pi}{N}\left(n+\frac{1}{2}+\frac{N}{2}\right)(2k+1)}$$

$$= \frac{1}{2N}e^{-j\frac{\pi}{N}\left(n+\frac{1}{2}+\frac{N}{2}\right)}\sum_{k=0}^{N-1}\left(\widehat{X_m}(2k+1)e^{-j\frac{\pi}{N}k}\right)e^{-j\frac{2\pi}{N}kn}$$

$$= \frac{1}{2N}e^{-j\frac{\pi}{N}\left(n+\frac{1}{2}+\frac{N}{2}\right)}\sum_{k=0}^{N-1}\left(\left(\widehat{X_p}(2k+1)e^{-j\pi k}\right)e^{-j\frac{\pi}{N}k}\right)e^{-j\frac{2\pi}{N}kn}$$

$$= \frac{e^{-j\frac{\pi}{N}\left(n+\frac{1}{2}+\frac{N}{2}\right)}}{j}\left(M(n) - M^*(N-n-1)\right)$$

$$= e^{-j\frac{\pi}{N}\left(n+\frac{1}{2}\right)}\left(M^*(N-n-1) - M(n)\right)$$

再考虑当 $0 \le n < N$ 时 $A^*(n+N)$ 的值：

$$A^*(n+N) = \frac{1}{2N}\sum_{k=0}^{2N-1}\widehat{X_p}(k)e^{-j\frac{\pi}{N}\left(n+N+\frac{1}{2}+\frac{N}{2}\right)k}$$

$$= \frac{1}{2N}\sum_{k=0}^{2N-1}\widehat{X_p}(k)e^{-j\frac{\pi}{N}\left(n+\frac{1}{2}+\frac{N}{2}\right)k}e^{-j\pi k}$$

$$= \frac{1}{2N}\sum_{k=0}^{2N-1}\widehat{X_p}(k)e^{-j\frac{\pi}{N}\left(n+\frac{1}{2}+\frac{N}{2}\right)k}(-1)^k$$

$$= \overbrace{\frac{1}{2N}\sum_{k=0}^{N-1}\widehat{X_p}(2k)e^{-j\frac{\pi}{N}\left(n+\frac{1}{2}+\frac{N}{2}\right)2k}}^{A_{even}^*(n)} - \overbrace{\frac{1}{2N}\sum_{k=0}^{N-1}\widehat{X_p}(2k+1)e^{-j\frac{\pi}{N}\left(n+\frac{1}{2}+\frac{N}{2}\right)(2k+1)}}^{A_{odd}^*(n)}$$

$$= A_{even}^*(n) - A_{odd}^*(n)$$

到此为止，对于 $0 \le n < 2N$，所有的 $A^*(n)$ 都已被求出，由此便可推出 $T_1(n)$、$T_2(n)$ 序列并得到最终结果 $\hat{x}(n)$：

$$T_1(n) = \frac{1}{2}\left(A(n) + A^*(n)\right)$$

$$= \mathrm{Re}\{A^*(n)\}$$

$$T_2(n) = \frac{1}{2j}\left(A(n) - A^*(n)\right)$$

$$= -\mathrm{Im}\{A^*(n)\}$$

$$\begin{aligned}\hat{x}(n) &= \cos\big(w(n)\big)\,T_1(n) - \sin\big(w(n)\big)\,T_2(n)\\ &= \cos\big(w(n)\big)\,\mathrm{Re}\{A^*(n)\} + \sin\big(w(n)\big)\,\mathrm{Im}\{A^*(n)\}\end{aligned}$$

总结基于 N 点 DFT 计算序列 $\hat{x}(n)$ 的值的流程（DFT 过程可采用 FFT 来实现）：
第一步：构造延扩序列 $\widehat{X_p}(k)$ 及 $\widehat{X_m}(k)$（$0 \le k < 2N$）；
第二步：构造序列 $m(k)$（$0 \le k < N$）并计算其 N 点 DFT，得到序列 $M(n)$（$0 \le n < N$）；
第三步：计算 $0 \le n < N$ 时 $A^*(n)$ 及 $A^*(n+N)$ 的值，由此构造序列 $A^*(n)$（$0 \le n < 2N$）；
第四步：利用序列 $A^*(n)$ 构造 IMDCT 输出序列 $\hat{x}(n)$。

算法伪代码如下：

```
//  Xp[0...2N - 1]:
...(略，见上文)...

//  Xm[0...2N - 1]:
...(略，见上文)...

//  m[0...N - 1], M[0...N - 1]:
for (k = 0; k < N; ++k) {
   m[k] = (Xm[2k] + 1j * Xm[2k + 1]) * pow(E, -1j * k * PI / N) / 4N;
}
M = DFT(m);

//  A_conj[0...2N-1] = conjugate{A[0...2N - 1]}:
for (n = 0; n < N; ++n) {
   A_conj_even = M[n] + (M[N - n - 1]).conjugate();
   A_conj_odd = pow(E, -1j * (n + 0.5) * PI / N) * (M[N - n - 1].conjugate() - M[n]);
   A_conj[n] = A_conj_even + A_conj_odd;
   A_conj[n + N] = A_conj_even - A_conj_odd;
}

//  x[0...2N]:
for (n = 0; n < 2N; ++n) {
   w = (n + 0.5 + 0.5N) * PI / 2N;
   x[n] = cos(w) * re(A_conj[n]) + sin(w) * im(A_conj[n]);
}
```

同样不难注意到 x[n] 的值仅依赖于 A_conj[n] 的值，因此数组 A_conj[] 实际上是不必需的，完全可以把上面的伪代码的后半部分简化为（这样也省去了一次长度为 2N 的遍历）：

```
...
//  A_conj[0...2N-1] = conjugate{A[0...2N - 1]}, x[0...2N]:
for (n = 0; n < N; ++n) {
   A_conj_even = M[n] + (M[N - n - 1]).conjugate();
   A_conj_odd = pow(E, -1j * (n + 0.5) * PI / N) * (M[N - n - 1].conjugate() - M[n]);
   u = A_conj_even + A_conj_odd;
   w = (n + 0.5 + 0.5N) * PI / 2N;
   x[n] = cos(w) * re(u) + sin(w) * im(u);
   u = A_conj_even - A_conj_odd;
   w = ((n + N) + 0.5 + 0.5N) * PI / 2N;
   x[n + N] = cos(w) * re(u) + sin(w) * im(u);
}
```

在相同的测试环境下对基于 N 点 FFT 的算法进行性能测试，结果如下：

N=100（非 2 的整次幂，FFT 采用 Bluestein 算法实现）					
N 点 FFT	146 ms	140 ms	140 ms	140 ms	140 ms
N=300（非 2 的整次幂，FFT 采用 Bluestein 算法实现）					
N 点 FFT	584 ms	581 ms	580 ms	576 ms	578 ms
N=512					
N 点 FFT	177 ms	177 ms	178 ms	175 ms	178 ms
N=1024					

N 点 FFT	360 ms	365 ms	358 ms	353 ms	363 ms
N=2048					
N 点 FFT	729 ms	730 ms	722 ms	724 ms	723 ms
N=4096					
N 点 FFT	1523 ms	1490 ms	1519 ms	1520 ms	1525 ms

可以看到在本文所述的测试环境下，基于 N 点 FFT 的算法的性能略好于基于 N 点 IFFT 的算法，这是由于本文所述的测试环境的 IFFT 是采用 FFT 来实现的，并不说明在任何场景下基于 FFT 的算法都要比基于 IFFT 的算法的性能要好。在实际应用中，必须根据实际情况选择恰当的算法。

附录 2：IMDCT 结果乘以某一实常数的情况

本文所述的所有推导过程均基于 IMDCT 的定义式：

$$\hat{x}(n)=\frac{1}{N}\sum_{k=0}^{N-1}\hat{X}(k)\cos\left(\frac{\pi}{N}\cdot\left(n+\frac{1}{2}+\frac{N}{2}\right)\cdot\left(k+\frac{1}{2}\right)\right)$$

但在很多的音视频标准中，IMDCT 过程的计算公式并不是上述的定义式，而是下面的这种形式（其中 c 是一个实常数，$0\le n<2N$）：

$$\widehat{x_G}(n)=c\cdot\frac{1}{N}\sum_{k=0}^{N-1}\hat{X}(k)\cos\left(\frac{\pi}{N}\cdot\left(n+\frac{1}{2}+\frac{N}{2}\right)\cdot\left(k+\frac{1}{2}\right)\right)$$

例如在 LC3[4] 音频编码中，IMDCT 过程的计算公式为：

$$\widehat{x_G}(n)=\sqrt{\frac{2}{N}}\sum_{k=0}^{N-1}\hat{X}(k)\cos\left(\frac{\pi}{N}\cdot\left(n+\frac{1}{2}+\frac{N}{2}\right)\cdot\left(k+\frac{1}{2}\right)\right)$$

那么显然对于 LC3 来说有 $c=\sqrt{2N}$。

一种最直观的对 $\widehat{x_G}(n)$ 的变形如下：

$$\widehat{x_G}(n)=\frac{1}{N}\sum_{k=0}^{N-1}\left(c\cdot\hat{X}(k)\right)\cos\left(\frac{\pi}{N}\cdot\left(n+\frac{1}{2}+\frac{N}{2}\right)\cdot\left(k+\frac{1}{2}\right)\right)$$

这也就意味着只需要在数据预处理阶段对所有的 $\hat{X}(k)$（$0\le k<N$）都乘以 c 再按照计算 IMDCT 定义式的过程来进行计算就可以得到 $\widehat{x_G}(n)$ 序列。但这种方法有一个问题，每处理一组数据就需要进行 N 次额外的乘法运算，而这实际上是不需要的。

下面给出推导过程。首先显然有：

$$\begin{aligned}\widehat{x_G}(n)&=c\cdot\hat{x}(n)\\&=\cos\big(w(n)\big)\,\mathrm{Re}\{c\cdot A(n)\}-\sin\big(w(n)\big)\,\mathrm{Im}\{c\cdot A(n)\}\end{aligned}$$

于是对于 $0\le n<N$ 就有：

$$\begin{aligned}c\cdot A(n)&=c\cdot A_{even}(n)+c\cdot A_{odd}(n)\\c\cdot A(n+N)&=c\cdot A_{even}(n)-c\cdot A_{odd}(n)\\c\cdot A_{even}(n)&=c\cdot z(n)+c\cdot z^{*}(N-n-1)\\&=c\cdot z(n)+\big(c\cdot z(N-n-1)\big)^{*}\end{aligned}$$

$$c\cdot A_{odd}(n)=e^{j\frac{\pi}{N}\left(n+\frac{1}{2}\right)}\left(c\cdot z(n)-c\cdot z^{*}(N-n-1)\right)$$

$$=e^{j\frac{\pi}{N}\left(n+\frac{1}{2}\right)}\left(c\cdot z(n)-\left(c\cdot z(N-n-1)\right)^{*}\right)$$

$$c\cdot z(n)=\frac{1}{N}\sum_{k=0}^{N-1}c\cdot Z(k)e^{j\frac{2\pi}{N}kn}$$

$$c\cdot Z(k)=\frac{1}{4}\left(\widehat{X_m}(2k)+j\cdot\widehat{X_m}(2k+1)\right)\cdot c\cdot e^{j\frac{\pi}{N}k}$$

注意到 $c\cdot e^{j\frac{\pi}{N}k}$ 这一项只与变量 k 有关，因此只需提前计算出所有的 $c\cdot e^{j\frac{\pi}{N}k}$（$0\le k<N$）的值，在处理每一组输入数据的时候就不再需要进行额外的 N 次乘法运算。

同理，对于本文所述的基于 N 点 DFT（FFT）的算法，有：

$$\widehat{x_G}(n)=c\cdot\hat{x}(n)$$
$$=\cos\left(w(n)\right)\mathrm{Re}\left\{c\cdot A^{*}(n)\right\}+\sin\left(w(n)\right)\mathrm{Im}\left\{c\cdot A^{*}(n)\right\}$$

于是对于 $0\le n<N$ 就有：

$$c\cdot A^{*}(n)=c\cdot A^{*}_{even}(n)+c\cdot A^{*}_{odd}(n)$$
$$c\cdot A^{*}(n+N)=c\cdot A^{*}_{even}(n)-c\cdot A^{*}_{odd}(n)$$
$$c\cdot A^{*}_{even}(n)=c\cdot M(n)+c\cdot M^{*}(N-n-1)$$
$$=c\cdot M(n)+\left(c\cdot M(N-n-1)\right)^{*}$$

$$c\cdot A^{*}_{odd}(n)=e^{-j\frac{\pi}{N}\left(n+\frac{1}{2}\right)}\left(c\cdot M^{*}(N-n-1)-c\cdot M(n)\right)$$

$$=e^{-j\frac{\pi}{N}\left(n+\frac{1}{2}\right)}\left(\left(c\cdot M(N-n-1)\right)^{*}-c\cdot M(n)\right)$$

$$c\cdot M(n)=\sum_{k=0}^{N-1}\left(c\cdot m(k)\right)e^{-j\frac{2\pi}{N}kn}$$

$$c\cdot m(k)=\frac{1}{4N}\left(\widehat{X_m}(2k)+j\cdot\widehat{X_m}(2k+1)\right)\cdot c\cdot e^{-j\frac{\pi}{N}k}$$

注意到 $c\cdot e^{-j\frac{\pi}{N}k}$ 这一项只与变量 k 有关，因此提前计算出所有的 $c\cdot e^{-j\frac{\pi}{N}k}$（$0\le k<N$）的值即可规避处理每一组输入数据时的额外的 N 次乘法运算。

<u>附录 3：DFT/IDFT 的定义</u>

由于在不同的文献对于 DFT/IDFT 的定义可能有所不同，为避免混淆，特在此列出在本文中所用的 DFT/IDFT 定义。

N 点复序列 $f(n)$（$0\le n<N$）的 N 点 DFT 序列 $F(k)$（$0\le k<N$）的定义为：

$$F(k)=\mathrm{DFT}\left\{\{f(0),f(1),\dots,f(N-1)\}\right\}(k)$$
$$=\sum_{n=0}^{N-1}f(n)e^{-j\frac{2\pi}{N}kn}$$

N 点复序列 $F(k)$（$0 \le k < N$）的 N 点 IDFT 序列 $f(n)$（$0 \le n < N$）的定义为：

$$f(n) = \mathrm{IDFT}\{\{F(0), F(1), \dots, F(N-1)\}\}(n)$$

$$= \frac{1}{N}\sum_{k=0}^{N-1} F(k) e^{j\frac{2\pi}{N}kn}$$

参考文献

编号	引用文献	DOI/URL
1	Bosi, M. and Goldberg, R.E. 2003. Introduction to Digital Audio Coding and Standards. Springer US.	10.1007/978-1-4615-0327-9
2	L. Bluestein, "A linear filtering approach to the computation of discrete Fourier transform," in IEEE Transactions on Audio and Electroacoustics, vol. 18, no. 4, pp. 451-455, December 1970, doi: 10.1109/TAU.1970.1162132.	10.1109/TAU.1970.1162132
3	Alan V. Oppenheim and Ronald W. Schafer. 2009. Discrete-Time Signal Processing (3rd. ed.). Prentice Hall Press, USA.	10.5555/77000
4	Bluetooth Low Complexity Communication Codec 1.0 Specification	https://www.bluetooth.com/specifications/specs/low-complexity-communication-codec-1-0/