The Discrete Fourier Transform

Gagarine Yaikhom

I. Introduction

We implement the *Discrete Fourier Transform (DFT)*. Please note, this will be slow as our aim is only to understand how the DFT formula is translated into a program.

⟨*Data structures*⟩≡

```
typedef struct {
    float real, imag;
} complex_t;
```

⟨*Functions*⟩≡

```
int dft_real_formula(complex_t *freq,
    int num_samples, const float *signal)
{
    ⟨Validate input and output buffers⟩;
    float rads_per_sample = (2 * M_PI) / num_samples;
    for (int i = 0; i < num_samples; ++i) {
        freq[i].real = 0.0; freq[i].imag = 0.0;
        for (int j = 0; j < num_samples; ++j) {
            float rads = rads_per_sample * i * j;
            freq[i].real += signal[j] * cosf(rads);
            freq[i].imag += signal[j] * sinf(rads);
        }
        freq[i].imag = -freq[i].imag;
    }
    return 0;
}
```

⟨*Validate input and output buffers*⟩≡

```
if (signal == NULL || freq == NULL) return -1;
```

⟨*Functions*⟩+≡

```
int dft_real_lookup(complex_t *freq,
    int num_samples, const float *signal)
{
    ⟨Validate input and output buffers⟩;
    ⟨Allocate lookup table for sines and cosines⟩;
    ⟨Fill lookup table with values⟩;
    for (int i = 0; i < num_samples; ++i) {
        freq[i].real = 0; freq[i].imag = 0;
        for (int j = 0; j < num_samples; ++j) {
            int k = (i * j) % num_samples;
            freq[i].real += signal[j] * cosines[k];
            freq[i].imag += signal[j] * sines[k];
        }
        freq[i].imag = -freq[i].imag;
    }
    free(ltab);
    return 0;
}
```

⟨*Allocate lookup table for sines and cosines*⟩≡

```
float *ltab = NULL, *sines, *cosines;
ltab = (float *) calloc(2 * num_samples, sizeof(float));
if (ltab == NULL) return -2;
sines = ltab;
cosines = sines + num_samples;
```

⟨*Fill lookup table with values*⟩≡

```
float rads_per_sample = (2 * M_PI) / num_samples;
for (int i = 0; i < num_samples; ++i) {
    float radians = rads_per_sample * i;
    sines[i] = sinf(radians);
    cosines[i] = cosf(radians);
}
```

⟨*Functions*⟩+≡

```
int dft_real(complex_t *freq,
    int num_samples, const float *signal)
{
    ⟨Validate input and output buffers⟩;
    ⟨Allocate lookup table for sines and cosines⟩;
    ⟨Fill lookup table with values⟩;
    for (int i = 0; i < num_samples; ++i) {
        freq[i].real = 0; freq[i].imag = 0;
    }
    for (int i = 0; i < num_samples; ++i) {
        int k = (i * i) % num_samples;
        freq[i].real += signal[i] * cosines[k];
        freq[i].imag += signal[i] * sines[k];
        for (int j = i + 1; j < num_samples; ++j) {
            k = (i * j) % num_samples;
            freq[i].real += signal[j] * cosines[k];
            freq[i].imag += signal[j] * sines[k];
            freq[j].real += signal[i] * cosines[k];
            freq[j].imag += signal[i] * sines[k];
        }
        freq[i].imag = -freq[i].imag;
    }
    free(ltab);
    return 0;
}
```

⟨*Functions*⟩+≡

```
int dft_complex(complex_t *freq,
    int num_samples, const complex_t *signal)
{
    ⟨Validate input and output buffers⟩;
    ⟨Allocate lookup table for sines and cosines⟩;
    ⟨Fill lookup table with values⟩;
    for (int i = 0; i < num_samples; ++i) {
        freq[i].real = 0; freq[i].imag = 0;
    }
    for (int i = 0; i < num_samples; ++i) {
        int k = (i * i) % num_samples;
        float c = cosines[k], s = sines[k];
        float a = signal[i].real, b = signal[i].imag;
        freq[i].real += a * c + b * s;
        freq[i].imag += b * c - a * s;
        for (int j = i + 1; j < num_samples; ++j) {
            k = (i * j) % num_samples;
            c = cosines[k]; s = sines[k];
            a = signal[j].real; b = signal[j].imag;
            freq[i].real += a * c + b * s;
            freq[i].imag += b * c - a * s;
            a = signal[i].real; b = signal[i].imag;
            freq[j].real += a * c + b * s;
            freq[j].imag += b * c - a * s;
        }
    }
    free(ltab);
    return 0;
}
```

⟨*Functions*⟩+≡

```
int idft(complex_t *signal, int num_samples,
    const complex_t *freq)
{
    ⟨Validate input and output buffers⟩;
    ⟨Allocate lookup table for sines and cosines⟩;
    ⟨Fill lookup table with values⟩;
    for (int i = 0; i < num_samples; ++i) {
        signal[i].real = 0; signal[i].imag = 0;
    }
    for (int i = 0; i < num_samples; ++i) {
        int k = (i * i) % num_samples;
        float c = cosines[k], s = sines[k];
        float a = freq[i].real, b = freq[i].imag;
        signal[i].real += a * c - b * s;
        signal[i].imag += b * c + a * s;
        for (int j = i + 1; j < num_samples; ++j) {
            k = (i * j) % num_samples;
            c = cosines[k]; s = sines[k];
            a = freq[j].real; b = freq[j].imag;
            signal[i].real += a * c - b * s;
            signal[i].imag += b * c + a * s;
            a = freq[i].real; b = freq[i].imag;
            signal[j].real += a * c - b * s;
            signal[j].imag += b * c + a * s;
        }
        signal[i].real /= num_samples;
        signal[i].imag /= num_samples;
    }
    free(ltab);
    return 0;
}
```

⟨*Test functions*⟩≡

```
void print_dft_real(int num_samples,
    const float *signal, const complex_t *freq)
{
    printf("%10s %21s\n", "Signal", "DFT");
    for (int i = 0; i < num_samples; ++i) {
        printf("%10.5f %10.5f%+10.5fi\n", signal[i],
               freq[i].real, freq[i].imag);
    }
}

void print_dft_complex(int num_samples,
    const complex_t *signal, const complex_t *freq)
{
    printf("%21s %21s\n", "Signal", "DFT");
    for (int i = 0; i < num_samples; ++i) {
        printf("%10.5f%+10.5fi %10.5f%+10.5fi\n",
               signal[i].real, signal[i].imag,
               freq[i].real, freq[i].imag);
    }
}

void print_idft(int num_samples,
    const complex_t *freq, const complex_t *signal)
{
    printf("%21s %21s\n", "DFT", "Signal");
    for (int i = 0; i < num_samples; ++i) {
        printf("%10.5f%+10.5fi %10.5f%+10.5fi\n",
               freq[i].real, freq[i].imag,
               signal[i].real, signal[i].imag);
    }
}

void test_dft()
{
    int num_samples = 16;
    float series[16] = {
        3, 2, 1, 1, 2, 3, 3, 3, 3, 2, 1, 1, 2, 3, 3, 3
    };
    complex_t signal1[16] = {
        {3, 0}, {2, 0}, {1, 0}, {1, 0},
        {2, 0}, {3, 0}, {3, 0}, {3, 0},
        {3, 0}, {2, 0}, {1, 0}, {1, 0},
        {2, 0}, {3, 0}, {3, 0}, {3, 0}
    };
    complex_t signal2[16] = {
        {1, 2}, {3, 4}, {5, 6}, {7, 8},
        {9, 10}, {11, 12}, {13, 14}, {15, 16},
        {1, 2}, {3, 4}, {5, 6}, {7, 8},
        {9, 10}, {11, 12}, {13, 14}, {15, 16}
    };
    complex_t freq[16];

    printf("DFT real series: Using formula\n");
    dft_real_formula(freq, num_samples, series);
    print_dft_real(num_samples, series, freq);

    printf("\nDFT real series: Lookup table\n");
```

```
    dft_real_lookup(freq, num_samples, series);
    print_dft_real(num_samples, series, freq);

    printf("\nDFT real series: Symmetric\n");
    dft_real(freq, num_samples, series);
    print_dft_real(num_samples, series, freq);

    printf("\nComplex DFT\n");
    dft_complex(freq, num_samples, signal1);
    print_dft_complex(num_samples, signal1, freq);

    printf("\nInverse DFT\n");
    idft(signal1, num_samples, freq);
    print_idft(num_samples, freq, signal1);

    printf("\nComplex DFT\n");
    dft_complex(freq, num_samples, signal2);
    print_dft_complex(num_samples, signal2, freq);

    printf("\nInverse DFT\n");
    idft(signal2, num_samples, freq);
    print_idft(num_samples, freq, signal2);

}
```

II. Main program

Main program for testing the functions.

⟨*Main*⟩≡

```
int main(void) {
    test_dft();
    return 0;
}
```

⟨*Standard libraries*⟩≡

```
#include <math.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
```

⟨*algorithm.c*⟩≡

⟨*Standard libraries*⟩
⟨*Data structures*⟩
⟨*Functions*⟩
⟨*Test functions*⟩
⟨*Main*⟩

III. Running the program

The program does not take any imputs. The test values are already hard-coded. Running the program should produce the following:

```
./algorithm
DFT real series: Using formula
    Signal                  DFT
   3.00000   36.00000  -0.00000i
   2.00000    0.00000  -0.00000i
   1.00000    3.41422  +8.24264i
   1.00000    0.00000  +0.00000i
   2.00000    2.00000  -2.00000i
   3.00000    0.00000  -0.00000i
   3.00000    0.58578  +0.24264i
   3.00000    0.00000  +0.00000i
   3.00000    0.00000  +0.00000i
   2.00000   -0.00001  +0.00001i
   1.00000    0.58579  -0.24263i
   1.00000   -0.00001  +0.00002i
   2.00000    1.99998  +2.00001i
   3.00000   -0.00000  +0.00002i
   3.00000    3.41418  -8.24264i
   3.00000   -0.00001  +0.00000i

DFT real series: Lookup table
    Signal                  DFT
   3.00000   36.00000  -0.00000i
   2.00000    0.00000  -0.00000i
   1.00000    3.41422  +8.24264i
   1.00000    0.00000  +0.00000i
   2.00000    2.00000  -2.00000i
   3.00000    0.00000  +0.00000i
   3.00000    0.58579  +0.24264i
   3.00000    0.00000  +0.00000i
   3.00000    0.00000  +0.00000i
   2.00000    0.00000  +0.00000i
   1.00000    0.58579  -0.24264i
   1.00000    0.00000  +0.00000i
   2.00000    2.00000  +2.00000i
   3.00000    0.00000  +0.00000i
   3.00000    3.41422  -8.24264i
   3.00000    0.00000  +0.00000i

DFT real series: Symmetric
    Signal                  DFT
   3.00000   36.00000  -0.00000i
   2.00000    0.00000  -0.00000i
   1.00000    3.41422  +8.24264i
   1.00000    0.00000  +0.00000i
   2.00000    2.00000  -2.00000i
   3.00000    0.00000  +0.00000i
   3.00000    0.58579  +0.24264i
   3.00000    0.00000  +0.00000i
   3.00000    0.00000  +0.00000i
   2.00000    0.00000  +0.00000i
   1.00000    0.58579  -0.24264i
   1.00000    0.00000  +0.00000i
```

```
   2.00000    2.00000  +2.00000i
   3.00000    0.00000  +0.00000i
   3.00000    3.41422  -8.24264i
   3.00000    0.00000  +0.00000i
```

```
Complex DFT
               Signal                    DFT
   3.00000  +0.00000i   36.00000  +0.00000i
   2.00000  +0.00000i    0.00000  -0.00000i
   1.00000  +0.00000i    3.41422  +8.24264i
   1.00000  +0.00000i    0.00000  +0.00000i
   2.00000  +0.00000i    2.00000  -2.00000i
   3.00000  +0.00000i    0.00000  +0.00000i
   3.00000  +0.00000i    0.58579  +0.24264i
   3.00000  +0.00000i    0.00000  +0.00000i
   3.00000  +0.00000i    0.00000  +0.00000i
   2.00000  +0.00000i    0.00000  +0.00000i
   1.00000  +0.00000i    0.58579  -0.24264i
   1.00000  +0.00000i    0.00000  +0.00000i
   2.00000  +0.00000i    2.00000  +2.00000i
   3.00000  +0.00000i    0.00000  +0.00000i
   3.00000  +0.00000i    3.41422  -8.24264i
   3.00000  +0.00000i    0.00000  +0.00000i
```

```
Inverse DFT
                  DFT               Signal
  36.00000  +0.00000i    3.00000  +0.00000i
   0.00000  -0.00000i    2.00000  -0.00000i
   3.41422  +8.24264i    1.00000  -0.00000i
   0.00000  +0.00000i    1.00000  -0.00000i
   2.00000  -2.00000i    2.00000  +0.00000i
   0.00000  +0.00000i    3.00000  +0.00000i
   0.58579  +0.24264i    3.00000  -0.00000i
   0.00000  +0.00000i    3.00000  +0.00000i
   0.00000  +0.00000i    3.00000  -0.00000i
   0.00000  +0.00000i    2.00000  -0.00000i
   0.58579  -0.24264i    1.00000  +0.00000i
   0.00000  +0.00000i    1.00000  -0.00000i
   2.00000  +2.00000i    2.00000  +0.00000i
   0.00000  +0.00000i    3.00000  -0.00000i
   3.41422  -8.24264i    3.00000  +0.00000i
   0.00000  +0.00000i    3.00000  -0.00000i
```

```
Complex DFT
               Signal                    DFT
   1.00000  +2.00000i  128.00000+144.00000i
   3.00000  +4.00000i    0.00001  +0.00000i
   5.00000  +6.00000i  -54.62740 +22.62742i
   7.00000  +8.00000i    0.00000  +0.00001i
   9.00000 +10.00000i  -32.00000  +0.00000i
  11.00000 +12.00000i    0.00001  +0.00001i
  13.00000 +14.00000i  -22.62741  -9.37257i
  15.00000 +16.00000i    0.00001  +0.00001i
   1.00000  +2.00000i  -16.00001 -15.99999i
   3.00000  +4.00000i    0.00001  +0.00000i
   5.00000  +6.00000i   -9.37258 -22.62742i
   7.00000  +8.00000i    0.00000  +0.00001i
   9.00000 +10.00000i   -0.00000 -32.00000i
  11.00000 +12.00000i    0.00001  +0.00001i
  13.00000 +14.00000i   22.62742 -54.62741i
  15.00000 +16.00000i    0.00000  +0.00001i
```

```
Inverse DFT
                  DFT               Signal
 128.00000+144.00000i    1.00000  +2.00001i
   0.00001  +0.00000i    3.00000  +4.00000i
 -54.62740 +22.62742i    5.00000  +6.00000i
   0.00000  +0.00001i    7.00000  +8.00000i
 -32.00000  +0.00000i    9.00000 +10.00000i
   0.00001  +0.00001i   11.00000 +12.00000i
 -22.62741  -9.37257i   13.00000 +14.00000i
   0.00001  +0.00001i   15.00000 +16.00000i
 -16.00001 -15.99999i    1.00000  +2.00000i
   0.00001  +0.00000i    3.00000  +4.00000i
  -9.37258 -22.62742i    5.00000  +6.00000i
   0.00000  +0.00001i    7.00000  +8.00000i
  -0.00000 -32.00000i    9.00000 +10.00000i
   0.00001  +0.00001i   11.00000 +12.00000i
  22.62742 -54.62741i   13.00000 +14.00000i
   0.00000  +0.00001i   15.00000 +16.00000i
```