

Fast Fourier Transform

Last Updated: February 2nd, 2026

1 Introduction

Like Strassen's algorithm, the Fast Fourier Transform (FFT) is considered one of the more surprising and interesting known divide-and-conquer algorithms. It finds important use in the field of signal and image processing but is perhaps best understood as a means for efficiently multiplying two polynomials which we present in this lecture.

2 Polynomial Multiplication and the Fast Fourier Transform

Given two polynomials

$$A(x) = a_0 + a_1x + \cdots + a_dx^d$$

and

$$B(x) = b_0 + b_1x + \cdots + b_dx^d,$$

our goal is to compute the product $C(x) = A(x)B(x)$ where $C(x)$ is a degree- $2d$ polynomial whose k th term c_k , $k = 0, 1, \dots, 2d$, is computed as

$$c_k = \sum_{i=0}^k a_i b_{k-i}.$$

Thus, using the above formula we see that computing the first $d + 1$ coefficients of $C(x)$ requires

$$1 + 2 + 3 + 4 + \cdots + d + (d + 1) = \Theta(d^2)$$

steps.

The following algorithm provides an alternative way to compute $C(x)$.

Alternative Polynomial Multiplication Algorithm

Input: Coefficients of polynomials $A(x)$ and $B(x)$.

Output: Coefficients of $C(x) = A(x)B(x)$.

Pick points: $x_0, x_1, \dots, x_{n-1}$, for some $n \geq 2d + 1$.

Evaluate A and B : compute $A(x_0), \dots, A(x_{n-1})$ and $B(x_0), \dots, B(x_{n-1})$.

Evaluate C : compute $C(x_0) = A(x_0)B(x_0), \dots, C(x_{n-1}) = A(x_{n-1})B(x_{n-1})$.

Interpolate: determine the unique coefficients $c_0, c_1, \dots, c_{2d}$ for which, for all $i = 0, 1, \dots, n-1$,

$$C(x_i) = c_0 + c_1x_i + \dots + c_{2d}x_i^{2d}.$$

Return $c_0, c_1, \dots, c_{2d}$.

Notes:

1. On the surface, it appears that this method will also require $O(d^2)$ steps, since evaluating a d -degree polynomial on some input x_i generally requires $\Theta(d)$ steps via Horner's algorithm.
2. Moreover, interpolation also requires $O(d^2)$ steps since, as we'll see, it involves inverting a $2d \times 2d$ Vandermonde matrix.
3. However, by choosing to *simultaneously* evaluate A (and B) with the n inputs via a divide-and-conquer approach, the total number of evaluation and interpolation steps may be reduced to $O(n \log n)$.

2.1 A Divide and Conquer approach to polynomial evaluation

In what follows we assume that n is a power of two. Consider the polynomial

$$A(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1}.$$

Then $A(x)$ may be written as

$$A(x) = A_e(x^2) + xA_o(x^2),$$

where $A_e(y)$ and $A_o(y)$ are the polynomials

$$A_e(y) = a_0 + a_2y + a_4y^2 + \cdots + a_{n-2}y^{\frac{n-2}{2}},$$

and

$$A_o(y) = a_1 + a_3y + \cdots + a_{n-1}y^{\frac{n-1}{2}}.$$

Thus, we may evaluate $(n-1)$ -degree polynomial $A(x)$ by evaluating two $(\frac{n-2}{2})$ -degree polynomials at x^2 . In other words, we've taken the problem and divided it into two subproblems, each of which is one-half the size. This yields the recurrence

$$T(n) = 2T(n/2) + n,$$

where $f(n) = n$ accounts for the fact that there are n inputs to $A(x)$, each of which may be computed in $O(1)$ steps using the equation

$$A(x) = A_e(x^2) + xA_o(x^2).$$

Therefore, assuming an appropriate choice for the n inputs to $A(x)$, the algorithm requires $O(n \log n)$ steps.

Example 2.1. For

$$A(x) = -2 + 5x + 3x^2 - 4x^3 + 9x^4 + x^5 - 8x^6 - 2x^7,$$

compute both $A_e(y)$ and $A_o(y)$.

Upon examining the formula

$$A(x) = A_e(x^2) + xA_o(x^2),$$

in order for the divide-and-conquer strategy to work, the following two properties must be true about the n numbers

$$x_1, x_2, \dots, x_{\frac{n}{2}}, x_{\frac{n}{2}+1}, x_{\frac{n}{2}+2}, \dots, x_n$$

that we use to evaluate $A(x)$.

1. Without loss of generality, we should have $x_{\frac{n}{2}+i} = -x_i$, i.e. if x_i is in the list, then so is its additive inverse. Why? Because the answers to $A_e(x^2)$ and $A_o(x^2)$ may be used by both x_i and $-x_i$, since $x_i^2 = (-x_i)^2$. Thus we only need to evaluate A_e and A_o a number of times that is one-half the number of inputs. In other words, if the problem is to evaluate a degree- $(n-1)$ polynomial at n inputs, then a problem instance that has a polynomial of degree $(n-2)/2$ should require the evaluation of $\frac{n}{2}$ inputs.
2. The first property should recursively also hold for $x_1^2, x_2^2, \dots, x_{\frac{n}{2}}^2$. Why? Because these numbers will serve as the inputs to the $n/2$ -sized subproblems A_e and A_o which are the two subproblems of our divide-and-conquer algorithm. In general, the first property must hold for *any* subproblem of any size that occurs as one of the problem instances of the divide-and-conquer algorithm.

The problem is that, in the case of a real number, its square is nonnegative, and so Property 1 will not hold when moving down to level 1 of the recursion (here we assume that level 0 is the root level). However, Property 1 *does* hold for the n th roots of unity which are complex numbers that we now define below.

3 Roots of Unity

Euler's Identity. For any real number θ , $e^{i\theta} = \cos \theta + i \sin \theta$

Example 3.1. Evaluate $e^{i\theta}$ for selected values of θ .

Complex n th roots of unity. For each $j = 0, \dots, n-1$, $e^{\frac{2\pi ij}{n}}$ is a **complex n th root of unity**, meaning that

$$e^{(\frac{2\pi ij}{n})n} = e^{2\pi ij} = \cos(2\pi j) + i \sin(2\pi j) = 1.$$

Example 3.2. Determine the complex 4th roots of unity.

Solution.

The next proposition shows that $e^{\frac{2\pi ij}{n}}$, $j = 0, \dots, n-1$, are the only unique powers of $e^{\frac{2\pi i}{n}}$.

Proposition 3.3. If integers j and k satisfy $j \equiv k \pmod{n}$, then

$$e^{\frac{2\pi ij}{n}} = e^{\frac{2\pi ik}{n}}.$$

Proof of Proposition. Assume $j \equiv k \pmod{n}$. Then $k = nq + j$, for some integer q . Then

$$e^{\frac{2\pi ik}{n}} = e^{\frac{2\pi i(j+nq)}{n}} = e^{\frac{2\pi ij}{n}} e^{\frac{2\pi inq}{n}} = e^{\frac{2\pi ij}{n}} e^{2\pi iq} = e^{\frac{2\pi ij}{n}} \cdot 1 = e^{\frac{2\pi ij}{n}}.$$

Definition 3.4. For $j = 0, \dots, n-1$, ω_n^j denotes the j th root of unity $e^{\frac{2\pi ij}{n}}$.

Proposition 3.5. The n th roots of unity form an **abelian group** under multiplication. In other words, the following properties hold for all integers i, j and k . Note: all exponents and exponent arithmetic is assumed as mod n arithmetic.

Associativity $(\omega_n^i \cdot \omega_n^j) \cdot \omega_n^k = \omega_n^i \cdot (\omega_n^j \cdot \omega_n^k) = \omega_n^{i+j+k}$.

Commutative $\omega_n^i \cdot \omega_n^j = \omega_n^j \cdot \omega_n^i = \omega_n^{i+j}$.

Existence of Unit $\omega_n^0 = 1$ and $1 \cdot \omega_n^i = \omega_n^i$.

Existence of Multiplicative Inverse $(\omega_n^i)^{-1} = \omega_n^{-i}$, since $\omega_n^i \cdot \omega_n^{-i} = \omega_n^{i-i} = \omega_n^0 = 1$.

Example 3.6. For the 6th roots of unity, determine the multiplicative inverse of $\frac{\sqrt{3}}{2} + \frac{1}{2}i$. Verify your answer by writing the inverse in standard form and multiplying with $\frac{1}{2} + \frac{\sqrt{3}}{2}i$.

Proposition 3.7. When $n \geq 2$ is even then the n th roots of unity have the following properties.

1. ω_n^j and $-\omega_n^j$ are both roots of unity. In other words, roots of unity come in additive-inverse pairs. Furthermore, if $0 \leq j < n/2$, then $\omega_n^{j+n/2} = -\omega_n^j$.
2. the squares of the n th roots of unity yield the $n/2$ roots of unity.

Proof of Proposition.

1. By the sum-of-angle formulas for cosine and sine, we have

$$e^{(\theta+\pi)i} = \cos(\theta + \pi) + i \sin(\theta + \pi) = -\cos \theta - \sin \theta i = -e^{\theta i}.$$

Therefore,

$$-\omega_n^j = e^{(\frac{2\pi i j}{n} + \pi i)} = e^{(\frac{2\pi i j}{n} + \frac{2\pi i (n/2)}{n})} = e^{\frac{2\pi i (j+n/2)}{n}} = \omega_n^{j+(n/2)}$$

which is a root of unity.

2. For $0 \leq j < n/2$, we have

$$(\omega_n^j)^2 = \omega_n^{2j} = e^{\frac{2\pi i (2j)}{n}} = e^{\frac{2\pi i j}{n/2}},$$

which is an $n/2$ root of unity. Note also that, for $n/2 \leq j < n$, $e^{\frac{2\pi i j}{n}}$ is just the negative of ω_n^j , and thus its square yields the same $n/2$ root of unity as its additive-inverse counterpart.

□

The above divide-and-conquer algorithm leads us to the following definition.

Definition 3.8. Given complex coefficients $c_0, \dots, c_{n-1}$, let $p(x)$ be the polynomial

$$p(x) = \sum_{k=0}^{n-1} c_k x^k.$$

Then the **n th order discrete Fourier transform** is the function

$$\text{DFT}_n(c_0, \dots, c_{n-1}) = (y_0, \dots, y_{n-1}),$$

where $y_j = p(\omega_n^j)$, $j = 0, \dots, n-1$.

In words the n th order discrete Fourier transform, takes as input the complex coefficients of a degree $n-1$ polynomial p , and returns the n -dimensional vector whose components are the evaluation of p at each of the n th roots of unity. Another way to write $\text{DFT}_n(c_0, \dots, c_{n-1})$ is $\text{DFT}_n(p)$, where p is the polynomial of degree $n-1$ whose coefficients are $c_0, \dots, c_{n-1}$.

Example 3.9. Compute $\text{DFT}_4(0, 1, 2, 3)$.

3.1 Fast Fourier Transform

We may now write our divide-and-conquer algorithm in terms of DFT_n . In what follows we define

$$(u_1, \dots, u_n) \odot (v_1, \dots, v_n) = (u_1 v_1, \dots, u_n v_n),$$

which we call the **scaling of v with u** .

Fast Fourier Transform

Input: polynomial $A(x) = a_0 + a_1 x + a_2 x^2 + \dots + a_{n-1} x^{n-1}$, where n is a power of two.

Output: $\text{DFT}_n(A)$.

If $n = 1$, then return (a_0) .

$Y_0 = \text{DFT}_{\frac{n}{2}}(A_e)$.

$Y_0 = Y_0 \circ Y_0$. //Concatenate vector Y_0 with itself.

$Y_1 = \text{DFT}_{\frac{n}{2}}(A_o)$.

$Y_1 = Y_1 \circ Y_1$. //Concatenate vector Y_1 with itself.

$Y_1 = \vec{\omega}_n \odot Y_1$. //Scale Y_1 with the length- n vector of n th roots of unity.

Return $Y_0 + Y_1$. //Return the vector sum of Y_0 with Y_1 .

We see that the running time for FFT is $\Theta(n \log n)$, since its running time satisfies

$$T(n) = 2T(n/2) + n.$$

Thus, we have found a way to evaluate a polynomial at n points using only a log-linear number of steps!

Example 3.10. Compute $\text{DFT}_4(0, 1, 2, 3)$ using the FFT algorithm.

Solution.

4 Solving Interpolation with the Inverse DFT

Returning to the alternative polynomial multiplication algorithm, the FFT algorithm allows us to compute $C(\omega_n^j)$, for each $j = 0, 1, \dots, n-1$. To finish the algorithm, we must find coefficients $c_0, c_1, \dots, c_{n-1}$ for which, for each $j = 0, 1, \dots, n-1$,

$$C(\omega_n^j) = c_0 + c_1\omega_n^j + \dots + c_{n-1}\omega_n^{j(n-1)}.$$

Furthermore, we can write these n equations in matrix form as follows.

$$\begin{pmatrix} C(\omega_n^0) \\ C(\omega_n^1) \\ \vdots \\ C(\omega_n^{n-1}) \end{pmatrix} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & \omega_n^1 & \dots & \omega_n^{1(n-1)} \\ \vdots & \vdots & \dots & \vdots \\ 1 & \omega_n^{n-1} & \dots & \omega_n^{(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{pmatrix}$$

Letting F_n denote the $n \times n$ matrix in the above equation, we leave it as an exercise to show that its inverse is

$$F_n^{-1} = \frac{1}{n} \begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & \omega_n^{-1} & \dots & \omega_n^{-1(n-1)} \\ \vdots & \vdots & \dots & \vdots \\ 1 & \omega_n^{-(n-1)} & \dots & \omega_n^{-(n-1)(n-1)} \end{pmatrix}.$$

Thus, we may compute the coefficients of $C(x)$ as

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \end{pmatrix} = \frac{1}{n} \begin{pmatrix} 1 & 1 & \dots & 1 \\ 1 & \omega_n^{-1} & \dots & \omega_n^{-1(n-1)} \\ \vdots & \vdots & \dots & \vdots \\ 1 & \omega_n^{-(n-1)} & \dots & \omega_n^{-(n-1)(n-1)} \end{pmatrix} \begin{pmatrix} C(\omega_n^0) \\ C(\omega_n^1) \\ \vdots \\ C(\omega_n^{n-1}) \end{pmatrix}.$$

Thus, for all $j = 0, 1, \dots, n-1$, we have

$$c_j = \frac{1}{n} (C(\omega_n^0) + C(\omega_n^1)\omega_n^{-j} + \dots + C(\omega_n^{n-1})\omega_n^{-j(n-1)}).$$

Notice that this equation is essentially the evaluation of polynomial

$$\frac{1}{n} (C(\omega_n^0) + C(\omega_n^1)x + \dots + C(\omega_n^{n-1})x^{n-1})$$

on input $x = \omega_n^{-j}$. This suggests the following definition.

Definition 4.1. Given complex coefficients $y_0, \dots, y_{n-1}$, let $p(x)$ be the polynomial

$$p(x) = \sum_{k=0}^{n-1} y_k x^k.$$

Then the **n th order inverse discrete Fourier transform** is the function

$$\text{DFT}_n^{-1}(y_0, \dots, y_{n-1}) = (c_0, \dots, c_{n-1}),$$

where $c_j = \frac{1}{n}p(\omega_n^{-j})$, $j = 0, \dots, n-1$.

In words the n th order inverse discrete Fourier transform, takes as input the complex coefficients of a degree $n-1$ polynomial p , and returns the n -dimensional vector whose components are the evaluation of $\frac{1}{n}p(x)$ at each of the inverses of the n th roots of unity.

4.1 The Inverse Fast Fourier Transform

We may provide a similar divide-and-conquer algorithm for computing DFT_n^{-1} which we call the **Inverse Fast Fourier Transform (IFFT)**.

Inverse Fast Fourier Transform

Input: polynomial $A(x) = a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1}$, where n is a power of two.

Output: $\text{DFT}_n^{-1}(A)$.

If $n = 1$, then return (a_0) .

$Y_0 = \text{DFT}_{\frac{n}{2}}^{-1}(A_e)$.

$Y_0 = Y_0 \circ Y_0$. //Concatenate vector Y_0 with itself.

$Y_1 = \text{DFT}_{\frac{n}{2}}^{-1}(A_o)$.

$Y_1 = Y_1 \circ Y_1$. //Concatenate vector Y_1 with itself.

$Y_1 = \omega_n^{-1} \odot Y_1$. //Scale Y_1 with the respective inverses of the n th roots of unity.

Return $\frac{1}{2}(Y_0 + Y_1)$. //Return the vector sum of Y_0 with Y_1 .

Notice that in the final line we must scale the vector by $1/2$. This is because both $\text{DFT}_{\frac{n}{2}}^{-1}(A_e)$ and $\text{DFT}_{\frac{n}{2}}^{-1}(A_o)$ give the polynomial evaluations divided by $n/2$. However, we want both to be divided by n . So we must multiply by $n/2$ to undo the division by $n/2$, and then divide by n , which has the net effect of multiplying by $1/2$.

Example 4.2. Compute $\text{DFT}_4^{-1}(0, 1, -1, 2)$ by a) using the definition of $\text{DFT}_4^{-1}(0, 1, -1, 2)$, and b) using the IFFT algorithm on $\text{DFT}_4^{-1}(0, 1, -1, 2)$.

4.2 Summary

$DFT_n(p)$ The *discrete Fourier transform* that evaluates an $(n - 1)$ -degree polynomial p at each of the n th roots of unity and returns a vector of these evaluations.

FFT An *algorithm* for computing $DFT_n(p)$ in $O(n \log n)$ steps when n is assumed a power of 2.

$DFT_n^{-1}(p)$ The *inverse discrete Fourier transform* that evaluates an $(n - 1)$ -degree polynomial p at each multiplicative inverse of each n th root of unity, and returns a vector of these evaluations scaled by $\frac{1}{n}$. Moreover if the coefficients of p are the values $q(\omega_n^0), q(\omega_n^1), \dots, q(\omega_n^{n-1})$, for some $(n - 1)$ -degree polynomial q , then $DFT_n^{-1}(p)$ outputs the coefficients of q , meaning that it solves the problem of *polynomial interpolation* with respect to q

IFFT An *algorithm* for computing $DFT_n^{-1}(p)$ in $O(n \log n)$ steps when n is assumed a power of 2.

FFT Core Exercises

1. Compute $\text{DFT}_4(1, -1, 2, 4)$ using the definition.
2. Compute $\text{DFT}_4(-1, 3, 4, 10)$ using the definition.
3. Compute $\text{DFT}_4^{-1}(0, 0, -4, 0)$ using the definition.
4. Compute $\text{DFT}_4^{-1}(2, 1 - i, 0, 1 + i)$ using the definition.
5. Use the FFT algorithm to compute $\text{DFT}_4(1, -1, 2, 4)$.
6. Use the FFT algorithm to compute $\text{DFT}_4(-1, 3, 4, 10)$.
7. Compute $\text{DFT}_4^{-1}(0, 0, -4, 0)$ using the definition.
8. Compute $\text{DFT}_4^{-1}(2, 1 - i, 0, 1 + i)$ using the definition.
9. Use the IFFT algorithm to compute $\text{DFT}_4^{-1}(0, 0, -4, 0)$.
10. Use the IFFT algorithm to compute $\text{DFT}_4^{-1}(2, 1 - i, 0, 1 + i)$.

Solutions to FFT Core Exercises

1. $\text{DFT}_4(1, -1, 2, 4) = (6, -1 - 5i, 0, -1 + 5i)$
2. $\text{DFT}_4(-1, 3, 4, 10) = (16, -5 - 7i, -10, -5 + 7i)$
3. $\text{DFT}_4^{-1}(0, 0, -4, 0) = (-1, 1, -1, 1)$
4. $\text{DFT}_4^{-1}(2, 1 - i, 0, 1 + i) = (1, 0, 0, 1)$
5. $p_0(x) = 1 + 2x$, $\text{DFT}_2(1 + 2x) = (3, -1)$. Thus,

$$Y_0 = (3, -1, 3, -1).$$

Also, $p_1(x) = -1 + 4x$, and $\text{DFT}_2(-1 + 4x) = (3, -5)$. Thus,

$$Y_1 = (3, -5, 3, -5).$$

Furthermore, $Y_{1j} \leftarrow \omega_4^j Y_{1j}$ gives

$$Y_1 = (3, -5i, -3, 5i).$$

Finally, $\text{DFT}_4(1, -1, 2, 4) = Y_0 + Y_1 = (6, -1 - 5i, 0, -1 + 5i)$.

6. $p_0(x) = -1 + 4x$, $\text{DFT}_2(-1 + 4x) = (3, -5)$. Thus,

$$Y_0 = (3, -5, 3, -5).$$

Also, $p_1(x) = 3 + 10x$, and $\text{DFT}_2(3 + 10x) = (13, -7)$. Thus,

$$Y_1 = (13, -7, 13, -7).$$

Furthermore, $Y_{1j} \leftarrow \omega_4^j Y_{1j}$ gives

$$Y_1 = (13, -7i, -13, 7i).$$

Finally, $\text{DFT}_4(-1, 3, 4, 10) = Y_0 + Y_1 = (16, -5 - 7i, -10, -5 + 7i)$.

7. Input $(0, 0, -4, 0)$ corresponds with polynomial $p(x) = -4x^2$. Moreover,

$$p(\omega_4^{(-1)(0)}) = p(1) = -4,$$

$$p(\omega_4^{-1}) = p(-i) = 4,$$

$$p(\omega_4^{-2}) = p(-1) = -4,$$

and

$$p(\omega_4^{-3}) = p(i) = 4.$$

Thus,

$$\text{DFT}_4^{-1}(0, 0, -4, 0) = \frac{1}{4}(-4, 4, -4, 4) = (-1, 1, -1, 1),$$

and so $\text{DFT}_4^{-1}(0, 0, -4, 0) = (-1, 1, -1, 1)$, which corresponds with polynomial $-1 + x - x^2 + x^3$.

8. Input $(2, 1 - i, 0, 1 + i)$ corresponds with polynomial $p(x) = 2 + (1 - i)x + (1 + i)x^3$. Moreover,

$$p(\omega_4^{(-1)(0)}) = p(1) = 4,$$

$$p(\omega_4^{-1}) = p(-i) = 0,$$

$$p(\omega_4^{-2}) = p(-1) = 0,$$

and

$$p(\omega_4^{-3}) = p(i) = 4.$$

Thus, $\text{DFT}_4^{-1}(2, 1 - i, 0, 1 + i) = (1, 0, 0, 1)$, which corresponds with polynomial $1 + x^3$.

9. $p_0(x) = -4x$, $\text{DFT}_2^{-1}(-4x) = \frac{1}{2}(-4, 4) = (-2, 2)$. Thus,

$$C_0 = (-2, 2, -2, 2).$$

Also, $p_1(x) = 0$, and $\text{DFT}_2^{-1}(0) = (0, 0)$. Thus,

$$C_1 = (0, 0, 0, 0).$$

Furthermore, $C_{1j} \leftarrow \omega_4^{-j} C_{1j}$ gives

$$C_1 = (0, 0, 0, 0).$$

Finally, $\text{DFT}_4^{-1}(0, 0, -4, 0) = \frac{1}{2}(C_0 + C_1) = \frac{1}{2}(-2, 2, -2, 2) = (-1, 1, -1, 1)$, which corresponds with polynomial $-1 + x - x^2 + x^3$.

10. $p_0(x) = 2$, $\text{DFT}_2^{-1}(2) = \frac{1}{2}(2, 2) = (1, 1)$. Thus,

$$C_0 = (1, 1, 1, 1).$$

Also, $p_1(x) = (1 - i) + (1 + i)x$, and $\text{DFT}_2^{-1}((1 - i) + (1 + i)x) = \frac{1}{2}(2, -2i) = (1, -i)$. Thus,

$$C_1 = (1, -i, 1, -i).$$

Furthermore, $C_{1j} \leftarrow \omega_4^{-j} C_{1j}$ gives

$$C_1 = (1, -1, -1, 1).$$

Finally, $\text{DFT}_4^{-1}(2, 1 - i, 0, 1 + i) = \frac{1}{2}(C_0 + C_1) = (1, 0, 0, 1)$, which corresponds with polynomial $1 + x^3$.

Additional Exercises

- A. Prove that for any two complex numbers c and d , $\overline{cd} = \overline{c}\overline{d}$.
- B. Write the standard form for each of the complex cube roots of unity.
- C. Write the standard form for each of the 6th roots of unity. Use the standard forms to verify both parts of Proposition 3.7.
- D. For the 6th roots of unity, determine the multiplicative inverse of each root, and verify that $(a + bi)(a + bi)^{-1} = 1$ through direct multiplication of the corresponding standard forms.
- E. Write the standard form for each of the 8th roots of unity. Use the standard forms to verify both parts of Proposition 3.7.
- F. Let $n \geq 1$, $d > 0$, and k be integers. Prove that $\omega_{dn}^{dk} = \omega_n^k$. This is called the **cancellation rule**.
- G. Let n be an even positive integer. Prove that the square of each of the n th roots of unity yields the $n/2$ roots of unity. Moreover, each $n/2$ root of unity is associated with two different squares of n th roots of unity.
- H. Show that $\omega_n^{n/2} = -1$, for all even $n \geq 2$.
- I. For positive integer n and for integer j not divisible by n , prove that

$$\sum_{k=0}^{n-1} \omega_n^{jk} = 0.$$

Hint: use the geometric series formula

$$\sum_{k=0}^{n-1} a^k = \frac{a^n - 1}{a - 1},$$

which is valid when a is a complex number.

- J. Show the sequence of polynomials that are evaluated when evaluating $p(x) = x^3 - 3x^2 + 5x - 6$ using Horner's algorithm. Use the algorithm to evaluate $p(-2)$.
- K. Show the sequence of polynomials that are evaluated when evaluating $p(x) = 2x^4 - x^3 + 2x^2 + 3x - 5$ using Horner's algorithm. Use the algorithm to evaluate $p(5)$.
- L. Find the equation of the quadratic polynomial whose graph passes through the points $(2, 13)$, $(-1, 10)$, and $(3, 26)$.
- M. Find the equation of the cubic polynomial whose graph passes through the points $(0, -1)$, $(1, 0)$, $(-1, -4)$, and $(2, 5)$.

Solutions to Additional Exercises

A. Let $c = a + bi$, and $d = e + fi$. Then

$$\overline{cd} = \overline{(ae - bf) + i(af + be)} = (ae - bf) - i(af + be).$$

On the other hand,

$$\overline{cd} = (a - bi)(e - fi) = (ae - bf) + i(-af - be) = (ae - bf) - i(af + be),$$

which proves the claim.

B. For $j = 0$,

$$e^{\frac{(2\pi)(0)i}{3}} = 1.$$

For $j = 1$,

$$e^{\frac{2\pi i}{3}} = -1/2 + \frac{\sqrt{3}i}{2}.$$

For $j = 2$,

$$e^{\frac{4\pi i}{3}} = -1/2 - \frac{\sqrt{3}i}{2}.$$

C. For $j = 0$,

$$e^{\frac{(2\pi)(0)i}{6}} = 1.$$

For $j = 1$,

$$e^{\frac{2\pi i}{6}} = \frac{1}{2} + \frac{\sqrt{3}i}{2}.$$

For $j = 2$,

$$e^{\frac{4\pi i}{6}} = \frac{-1}{2} + \frac{\sqrt{3}i}{2}.$$

For $j = 3$,

$$e^{\frac{6\pi i}{6}} = e^{\pi i} = -1.$$

For $j = 4$,

$$e^{\frac{8\pi i}{6}} = \frac{-1}{2} - \frac{\sqrt{3}i}{2}.$$

For $j = 5$,

$$e^{\frac{10\pi i}{6}} = \frac{1}{2} - \frac{\sqrt{3}i}{2}.$$

Notice that, for $j = 0, 1, 2$, $\omega_6^j = -\omega_6^{3+j}$. For example,

$$\omega_6^1 = \frac{1}{2} + \frac{\sqrt{3}i}{2} = -\left(-\frac{1}{2} - \frac{\sqrt{3}i}{2}\right) = \omega_6^4.$$

Finally, computing the squares of each sixth root of unity yields the numbers 1 , $\frac{-1}{2} + \frac{\sqrt{3}i}{2}$, $\frac{-1}{2} - \frac{\sqrt{3}i}{2}$ which are the third roots of unity.

D. We have

$$\omega_6^0 \cdot \omega_6^0 = (1)(1) = 1.$$

$$\omega_6^1 \cdot \omega_6^5 = \left(\frac{1}{2} + \frac{\sqrt{3}i}{2}\right)\left(\frac{1}{2} - \frac{\sqrt{3}i}{2}\right) = 1,$$

$$\omega_6^2 \cdot \omega_6^4 = \left(-\frac{1}{2} + \frac{\sqrt{3}i}{2}\right)\left(-\frac{1}{2} - \frac{\sqrt{3}i}{2}\right) = 1,$$

and

$$\omega_6^3 \cdot \omega_6^3 = (-1)(-1) = 1.$$

E. For $j = 0$,

$$e^{\frac{(2\pi)(0)i}{3}} = 1.$$

For $j = 1$,

$$e^{\frac{\pi i}{4}} = \frac{\sqrt{2}}{2} + \frac{\sqrt{2}i}{2}.$$

For $j = 2$,

$$e^{\frac{\pi i}{2}} = i.$$

For $j = 3$,

$$e^{\frac{3\pi i}{4}} = \frac{-\sqrt{2}}{2} + \frac{\sqrt{2}i}{2}.$$

For $j = 4$,

$$e^{\pi i} = -1.$$

For $j = 5$,

$$e^{\frac{5\pi i}{4}} = \frac{-\sqrt{2}}{2} - \frac{\sqrt{2}i}{2}.$$

For $j = 6$,

$$e^{\frac{3\pi i}{2}} = -i.$$

For $j = 7$,

$$e^{\frac{7\pi i}{4}} = \frac{\sqrt{2}}{2} - \frac{\sqrt{2}i}{2}.$$

Notice that, for $j = 0, 1, 2, 3$, $\omega_8^j = -\omega_8^{4+j}$. For example,

$$\omega_8^2 = i = -(-i) = \omega_8^6.$$

Finally, computing the squares of each eighth root of unity yields the numbers 1, i , -1 , and $-i$ which are exactly the fourth roots of unity.

F. By definition,

$$\omega_{dn}^{dk} = e^{\frac{2\pi i dk}{dn}} = e^{\frac{2\pi i k}{n}} = \omega_n^k.$$

G. For $j = 0, \dots, n-1$,

$$(\omega_n^j)^2 = \omega_n^j \omega_n^j = \omega_n^{2j} = \omega_{n/2}^j,$$

where the last equality is due to the cancellation rule from Exercise 6. Thus the square of an n th root of unity is indeed an $n/2$ root of unity. Moreover, notice that j ranges from 0 to $n-1$. By definition, when j ranges from 0 to $n/2-1$, we obtain each $n/2$ root of unity. Then, due to the cyclic nature of the roots of unity, when j ranges from $n/2$ to $n-1$, we once again obtain each $n/2$ root of unity. Therefore, each $n/2$ root of unity $\omega_{n/2}^j$ is the square of exactly two different n th-roots of unity, namely $(\omega_{n/2}^j)^2$ and $(\omega_{n/2}^{j+n/2})^2$.

H. We have, for even $n \geq 2$,

$$\omega_n^{n/2} = e^{(2\pi i/n)n/2} = e^{\pi i} = \cos \pi + i \sin \pi = -1.$$

I. Using the geometric series formula

$$\sum_{k=0}^{n-1} a^k = \frac{a^n - 1}{a - 1},$$

we have

$$\begin{aligned} \sum_{k=0}^{n-1} (\omega_n^j)^k &= \sum_{k=0}^{n-1} \omega_n^{jk} = \\ \frac{\omega_n^{jn} - 1}{\omega_n^j - 1} &= \frac{\omega_1^j - 1}{\omega_n^j - 1} = \frac{1 - 1}{\omega_n^j - 1} = 0, \end{aligned}$$

where the first equality is due to the cancellation rule, and the 2nd to last equality is due to the fact that $\omega_1^1 = 1$. Notice also that the denominator is not equal to zero, since we assumed j is not divisible by n ; i.e. $j \not\equiv 0 \pmod n$.

J. $p_0(x) = 1$, $p_1(x) = xp_0(x) - 3 = x - 3$, $p_2(x) = xp_1(x) + 5 = x^2 - 3x + 5$, $p_3(x) = xp_2(x) - 6 = x^3 - 3x^2 + 5x - 6$. $p_0(-2) = 1$, $p_1(-2) = -2(1) - 3 = -5$, $p_2(-2) = -2(-5) + 5 = 15$, $p_3(-2) = -2(15) - 6 = -36$.

K. $p_0(x) = 2$, $p_1(x) = xp_0(x) - 1 = 2x - 1$, $p_2(x) = xp_1(x) + 2 = 2x^2 - x + 2$, $p_3(x) = xp_2(x) + 3 = 2x^3 - x^2 + 2x + 3$, $p_4(x) = xp_3(x) - 5 = 2x^4 - x^3 + 2x^2 + 3x - 5$. $p_0(5) = 2$, $p_1(5) = 5(2) - 1 = 9$, $p_2(5) = 5(9) + 2 = 47$, $p_3(5) = 5(47) + 3 = 238$, $p_4(5) = 5(238) - 5 = 1185$.

L. We desire a polynomial of the form $c_0 + c_1x + c_2x^2$. The three points imply the following system of equations.

$$c_0 + 2c_1 + 4c_2 = 13$$

$$c_0 - c_1 + c_2 = 10$$

$$c_0 + 3c_1 + 9c_2 = 26$$

Solving this system gives the polynomial $5 - 2x + 3x^2$.

M. We desire a polynomial of the form $c_0 + c_1x + c_2x^2 + c_3x^3$. The four points imply the following system of equations.

$$c_0 = -1$$

$$c_0 + c_1 + c_2 + c_3 = 0$$

$$c_0 - c_1 + c_2 - c_3 = -4$$

$$c_0 + 2c_1 + 4c_2 + 8c_3 = 5$$

Solving this system gives the polynomial $-1 + x - x^2 + x^3$.