Faster Polynomial Multiplication? (in Coefficient Form)

\[ A(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1} \]
\[ B(x) = b_0 + b_1 x + \cdots + b_{n-1} x^{n-1} \]

\[ C(x) = c_0 + c_1 x + \cdots + c_{2n-1} x^{2n-1} \]

**Time \( \Theta(n^2) \)**

- **Forward FFT**
  - Time \( \Theta(n \log n) \)
  - \[ A(\omega_{2n}^0), B(\omega_{2n}^0) \]
  - \[ A(\omega_{2n}^1), B(\omega_{2n}^1) \]
  - \[ \vdots \]
  - \[ A(\omega_{2n}^{2n-1}), B(\omega_{2n}^{2n-1}) \]

- **Pointwise Multiplication**
  - Time \( \Theta(n) \)
  - \[ C(\omega_{2n}^0) \]
  - \[ C(\omega_{2n}^1) \]
  - \[ \vdots \]
  - \[ C(\omega_{2n}^{2n-1}) \]

- **Interpolation**
  - Time?

**Ordinary Multiplication**

**Time**

\[ \Theta(n^2) \]
Point-Value Form ⇒ Coefficient Form

Given:

\[
\begin{bmatrix}
1 & 1 & 1 & \cdots & 1 \\
1 & \omega_n & (\omega_n)^2 & \cdots & (\omega_n)^{n-1} \\
1 & \omega_n^2 & (\omega_n^2)^2 & \cdots & (\omega_n^2)^{n-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \omega_n^{n-1} & (\omega_n^{n-1})^2 & \cdots & (\omega_n^{n-1})^{n-1}
\end{bmatrix}
\]

\(V(\omega_n)\)

\[
V(\omega_n) \cdot \vec{a} = \vec{y}
\]

We want to solve: \(\vec{a} = [V(\omega_n)]^{-1} \cdot \vec{y}\)

It turns out that: \([V(\omega_n)]^{-1} = \frac{1}{n} V\left(\frac{1}{\omega_n}\right)\)

That means \([V(\omega_n)]^{-1}\) looks almost similar to \(V(\omega_n)\)!
Point-Value Form $\Rightarrow$ Coefficient Form

Show that: $[V(\omega_n)]^{-1} = \frac{1}{n} V \left( \frac{1}{\omega_n} \right)$

Let $U(\omega_n) = \frac{1}{n} V \left( \frac{1}{\omega_n} \right)$

We want to show that $U(\omega_n)V(\omega_n) = I_n$, where $I_n$ is the $n \times n$ identity matrix.

Observe that for $0 \leq j, k \leq n - 1$, the $(j, k)^{th}$ entries are:

$[V(\omega_n)]_{jk} = \omega_{n}^{jk}$ and $[U(\omega_n)]_{jk} = \frac{1}{n} \omega_{n}^{-jk}$

Then entry $(p, q)$ of $U(\omega_n)V(\omega_n)$,

$[U(\omega_n)V(\omega_n)]_{pq} = \sum_{k=0}^{n-1} [U(\omega_n)]_{pk} [V(\omega_n)]_{kq} = \frac{1}{n} \sum_{k=0}^{n-1} \omega_{n}^{k(q-p)}$
Point-Value Form ⇒ Coefficient Form

\[ [U(\omega_n)V(\omega_n)]_{pq} = \frac{1}{n} \sum_{k=0}^{n-1} \omega_n^k(p-q) \]

CASE \( p = q \):

\[ [U(\omega_n)V(\omega_n)]_{pq} = \frac{1}{n} \sum_{k=0}^{n-1} \omega_n^0 = \frac{1}{n} \sum_{k=0}^{n-1} 1 = \frac{1}{n} \times n = 1 \]

CASE \( p \neq q \):

\[ [U(\omega_n)V(\omega_n)]_{pq} = \frac{1}{n} \sum_{k=0}^{n-1} (\omega_n^{q-p})^k = \frac{1}{n} \times \frac{(\omega_n^{q-p})^n - 1}{\omega_n^{q-p} - 1} = \frac{1}{n} \times \frac{(1)^{q-p} - 1}{\omega_n^{q-p} - 1} = 0 \]

Hence \( U(\omega_n)V(\omega_n) = I_n \)
We need to compute the following matrix-vector product:

\[
\begin{bmatrix}
\frac{1}{\omega_0} & \frac{1}{\omega_1} & \frac{1}{\omega_2} & \cdots & \frac{1}{\omega_{n-1}} \\
1 & \frac{1}{\omega_0} & \frac{1}{\omega_1} & \cdots & \frac{1}{\omega_{n-1}} \\
\frac{1}{\omega_0} & \frac{1}{\omega_1} & \frac{1}{\omega_2} & \cdots & \frac{1}{\omega_{n-1}} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\frac{1}{\omega_0} & \frac{1}{\omega_1} & \frac{1}{\omega_2} & \cdots & \frac{1}{\omega_{n-1}} \\
\end{bmatrix} \cdot 
\begin{bmatrix}
1 \\
\frac{1}{\omega_0} \\
\frac{1}{\omega_1} \\
\vdots \\
\frac{1}{\omega_{n-1}} \\
\end{bmatrix} = \frac{1}{n} \cdot 
\begin{bmatrix}
y_0 \\
y_1 \\
y_2 \\
\vdots \\
y_{n-1} \\
\end{bmatrix}
\]

This inverse problem is almost similar to the forward problem, and can be solved in $\Theta(n \log n)$ time using the same algorithm as the forward FFT with only minor modifications!
Faster Polynomial Multiplication? (in Coefficient Form)

Two polynomials of degree bound $n$ given in the coefficient form can be multiplied in $\Theta(n \log n)$ time!
Some Applications of Fourier Transform and FFT

- Signal processing
- Image processing
- Noise reduction
- Data compression
- Solving partial differential equation
- Multiplication of large integers
- Polynomial multiplication
- Molecular docking
Some Applications of Fourier Transform and FFT

Jean Baptiste Joseph Fourier

Any periodic signal can be represented as a sum of a series of sinusoidal (sine & cosine) waves. [1807]
Spatial (Time) Domain ⇔ Frequency Domain

Source: The Scientist and Engineer’s Guide to Digital Signal Processing by Steven W. Smith
Spatial (Time) Domain $\iff$ Frequency Domain (Fourier Transforms)

Let $s(t)$ be a signal specified in the time domain.

The strength of $s(t)$ at frequency $f$ is given by:

$$ S(f) = \int_{-\infty}^{\infty} s(t) \cdot e^{-2\pi i ft} \, dt $$

Evaluating this integral for all values of $f$ gives the frequency domain function.

Now $s(t)$ can be retrieved by summing up the signal strengths at all possible frequencies:

$$ s(t) = \int_{-\infty}^{\infty} S(f) \cdot e^{2\pi i ft} \, df $$
Why do the Transforms Work?

Let’s try to get a little intuition behind why the transforms work. We will look at a very simple example.

Suppose: \( s(t) = \cos(2\pi h \cdot t) \)

\[
\frac{1}{T} \int_{-T}^{T} s(t) \cdot e^{-2\pi if t} \, dt = \begin{cases} 
1 + \frac{\sin(4\pi fT)}{4\pi fT}, & \text{if } f = h, \\
\frac{\sin(2\pi (h-f)T)}{2\pi (h-f)T} + \frac{\sin(2\pi (h+f)T)}{2\pi (h+f)T}, & \text{otherwise}.
\end{cases}
\]

\[
\Rightarrow \lim_{T \to \infty} \left( \frac{1}{T} \int_{-T}^{T} s(t) \cdot e^{-2\pi if t} \, dt \right) = \begin{cases} 
1, & \text{if } f = h, \\
0, & \text{otherwise}.
\end{cases}
\]

So, the transform can detect if \( f = h \)!
Noise Reduction

Data Compression

− Discrete Cosine Transforms (DCT) are used for lossy data compression (e.g., MP3, JPEG, MPEG)

− DCT is a Fourier-related transform similar to DFT (Discrete Fourier Transform) but uses only real data (uses cosine waves only instead of both cosine and sine waves)

− Forward DCT transforms data from spatial to frequency domain

− Each frequency component is represented using a fewer number of bits (i.e., truncated/quantized)

− Low amplitude high frequency components are also removed

− Inverse DCT then transforms the data back to spatial domain

− The resulting image compresses better
Data Compression

Transformation to frequency domain using cosine transforms work in the same way as the Fourier transform. Suppose: \( s(t) = \cos(2\pi h \cdot t) \)

\[
\frac{1}{T} \int_{-T}^{T} s(t) \cdot \cos(2\pi ft) \, dt = \begin{cases} 
1 + \frac{\sin(4\pi fT)}{4\pi fT}, & \text{if } f = h, \\
\frac{\sin(2\pi (h - f)T)}{2\pi (h - f)T} + \frac{\sin(2\pi (h + f)T)}{2\pi (h + f)T}, & \text{otherwise.}
\end{cases}
\]

\[
\Rightarrow \lim_{T \to \infty} \left( \frac{1}{T} \int_{-T}^{T} s(t) \cdot \cos(2\pi ft) \, dt \right) = \begin{cases} 
1, & \text{if } f = h, \\
0, & \text{otherwise.}
\end{cases}
\]

So, this transform can also detect if \( f = h \).
Protein-Protein Docking

- Knowledge of complexes is used in
  - Drug design
  - Studying molecular assemblies
  - Structure function analysis
  - Protein interactions

Protein-Protein Docking: Given two proteins, find the best relative transformation and conformations to obtain a stable complex.

Docking is a hard problem
- Search space is huge (6D for rigid proteins)
- Protein flexibility adds to the difficulty
To maximize skin-skin overlaps and minimize core-core overlaps

- assign positive real weights to skin atoms
- assign positive imaginary weights to core atoms

Let $A'$ denote molecule $A$ with the pseudo skin atoms.

For $P \in \{A', B\}$ with $M_P$ atoms, affinity function:

$$f_P(x) = \sum_{k=1}^{M_P} w_k \cdot g_k(x)$$

Here $g_k(x)$ is a Gaussian representation of atom $k$, and $w_k$ its weight.
Let $A'$ denote molecule $A$ with the pseudo skin atoms.

For $P \in \{A', B\}$ with $M_P$ atoms, affinity function:

$$f_P(x) = \sum_{k=1}^{M_P} w_k \cdot g_k(x)$$

For rotation $r$ and translation $t$ of molecule $B$ (i.e., $B_{t,r}$),

the interaction score, $F_{A,B}(t, r) = \int_x f_{A'}(x)f_{B_{t,r}}(x) \, dx$
For rotation $r$ and translation $t$ of molecule $B$ (i.e., $B_{t,r}$),

the interaction score, $F_{A,B}(t,r) = \int_x f_A'(x)f_{B_{t,r}}(x) \, dx$

\[ \text{Re} \left( F_{A,B}(t,r) \right) = \text{skin-skin overlap score} - \text{core-core overlap score} \]

\[ \text{Im} \left( F_{A,B}(t,r) \right) = \text{skin-core overlap score} \]
Docking: Rotational & Translational Search
Translational Search using FFT

$\forall z \in \Omega = [-n, n]^3, \quad h(z) = \int_{x \in \Omega} f_{A'}(x)f_{B'}(z - x)dx$