# The Discrete Fourier Transform

Rutger Noot

IRMA
Université de Strasbourg et CNRS

21 January 2017 — Strasbourg

## Polynomials

### The basics

A polynomial is an expression

$$P(X) = \sum_{i=0}^{d} a_i X^i$$

for some integer $d > 0$.
The degree of $P$ is $\leq d$.

### A word on the $a_i$

The $a_i$ belong to a field $K$,
for the time being we can think of $K = \mathbb{R}$ or $K = \mathbb{C}$.

## Evaluation

For $P$ a polynomial and $x \in K$, we can evualuate $P$ at $x$:

$$P(x) = \sum_{i=0}^{d} a_i x^i$$

(addition and multiplication in $K$).

## Computing $P(x)$

### Computational complexity

The number of operations needed to compute $P(x)$:

- $d - 1$ multiplications to compute the $x^i$ and
- another $d$ multiplications and $d$ additions to obtain $P(x)$.

A total of $2d - 1$ multiplications and $d$ additions.

In practice, a multiplication is much more 'expensive' than an addition.

## Another way of computing $P(x)$

We can do better!
In fact,

$$P(x) = a_0 + x(a_1 + x(a_2 + \cdots + x(a_{d-1} + xa_d))),$$

so starting from the innermost parenthesis we only need

$d$ multiplications and $d$ additions.

An improvement by a factor 2!

## Addition ...

For two polynomials

$$P(X) = \sum_{i=0}^{d} a_i X^i \qquad \text{and} \qquad Q(X) = \sum_{i=0}^{d} b_i X^i,$$

the sum is

$$(P+Q)(X) = \sum_{i=0}^{d} (a_i + b_i) X^i.$$

It can be computed in just $d$ additions.

The evaluation function associated to $P + Q$ is the sum of the functions defined by $P$ and $Q$.

## ... and multiplication

The product $PQ$ is

$$PQ(X) = \sum_{i=0}^{2d} c_i X^i,$$

where

$$c_i = \sum_{j=\max(0,i-d)}^{\min(i,d)} a_j b_{i-j}.$$

- This is a convolution product.
- The function associated to $PQ$ is the product of the functions defined by $P$ and $Q$.
- The degree of $PQ$ may be $> d$, but is $\leq 2d$.

## Computing $PQ$

Using the formula, $c_i$ can be computed in $i+1$ multiplications and $i$ additions (for $i \leq d$) so the computation of $PQ$ takes

$(d+1)^2$ multiplcations and $d^2$ additions.

NB. To compute $(PQ)(x)$ for $x \in K$, take $P(x)$ and $Q(x)$ and then multiply instead of first computing the polynomial $PQ$ and then taking $(PQ)(x)$.

## A silly idea?

Why not define a product

$$P * Q(X) = \sum_{i=0}^{d} (a_i b_i) X^i ?$$

- Much easier to compute: $d+1$ multiplications,
- but has no meaning in terms of functions.

## Roots of unity

Assume that $d+1 \neq 0$ (in $K$) and let $\zeta \in K$ be a primitive $(d+1)$th root of unity:

- $\zeta^{d+1} = 1$ but
- $\zeta^i \neq 1$ for $1 \leq i \leq d$.

We have:

- For $i = 0, \cdots d$, the $\zeta^i$ are the distinct $x \in K$ with $x^{d+1} = 1$.
- For $i = 1, \ldots, d$:
$$\sum_{j=0}^{d} \zeta^{ij} = 0$$

## The Discrete Fourier Transform

**Definition**
The Discrete Fourier Transform (DFT) of $P$ (of degree $\leq d$) is

$$\mathrm{FD}_d(P) = \sum_{j=0}^{d} y_j X^{d-j}$$

where $y_j = P(\zeta^j)$.

## The inverse of the DFT

**Theorem**

$$\mathrm{FD}_d(P)(\zeta^k) = (d+1) a_k \zeta^{-k}$$

*so*

$$\mathrm{FD}_d(\mathrm{FD}_d(P)) = (d+1) X^d P\left(\frac{1}{\zeta X}\right).$$

(Which is indeed a polynomial!)

## More properties of the DFT

- $\mathrm{FD}_d \circ \mathrm{FD}_d$ is bijective.
- Hence $\mathrm{FD}_d$ is bijective (on polynomials of degree $\leq d$).
- $(PQ)(\zeta^i) = P(\zeta^i)Q(\zeta^i)$ so $\mathrm{FD}_d(PQ) = \mathrm{FD}_d(P) * \mathrm{FD}_d(Q)$.
- If $P = \mathrm{FD}_d(\widetilde{P})$ and $Q = \mathrm{FD}_d(\widetilde{Q})$ for $\widetilde{P}, \widetilde{Q}$ of degree $\leq d/2$ then

$$\mathrm{FD}_d(P * Q) = \mathrm{FD}_d(\mathrm{FD}_d(\widetilde{P}) * \mathrm{FD}_d(\widetilde{Q})) = \mathrm{FD}_d(\mathrm{FD}_d(\widetilde{P}\widetilde{Q})) =$$
$$(d+1)X^d \left(\widetilde{P}\widetilde{Q}\right)\left(\frac{1}{\zeta X}\right) = \frac{1}{(d+1)X^d}\mathrm{FD}_d(P)\mathrm{FD}_d(Q)$$

## Conclusion

### A formula for $PQ$

For polynomials $P, Q$ of degree $\leq d/2$

$$(PQ)(X) = \frac{X^d}{\zeta(d+1)}\mathrm{FD}_d(\mathrm{FD}_d(P) * \mathrm{FD}_d(Q))\left(\frac{1}{\zeta X}\right).$$

### A useful algorithm?

- It takes $d$ multiplications to compute $\mathrm{FD}_d(P) * \mathrm{FD}_d(Q)$,
- so everything depends on the complexity of $\mathrm{FD}_d$,
- but the obvious algorithm evaluates $P$ for $d+1$ values so it takes $\frac{3}{2}(d^2+d)$ multiplications...

## A special case

From now on: $d + 1 = 2^e$ is a power of 2.

Write $d' = (d-1)/2$ and

$$P^{(\text{even})}(X) = \sum_{i=0}^{d'} a_{2i}X^i \qquad \text{and} \qquad P^{(\text{odd})}(X) = \sum_{i=0}^{d'} a_{2i+1}X^i$$

so

$$P(X) = P^{(\text{even})}(X^2) + XP^{(\text{odd})}(X^2).$$

## Fast Fourier Transform

### Theorem

$$\mathrm{FD}_d(P)(X) =$$
$$X^{d'+1}\left(\mathrm{FD}_{d'}(P^{(\text{even})})(X) - \zeta^{-1}\mathrm{FD}_{d'}(P^{(\text{odd})})(\zeta^{-1}X)\right) +$$
$$\left(\mathrm{FD}_{d'}(P^{(\text{even})})(X) + \zeta^{-1}\mathrm{FD}_{d'}(P^{(\text{odd})})(\zeta^{-1}X)\right).$$

So $\mathrm{FD}_d$ can be computed by induction!

## Computing the FFT

- $M(d)$ = number of multiplications to compute $\mathrm{FD}_d(P)$ (for $P$ of degree $d$).
- By the theorem $M(d) = 2M(d') + d + 1$.
- Hence
$$M(d) = Cd \log d + \text{lower order terms}$$
for some constant $C > 0$.
- Notation: $M(d) = \Theta(d \log d)$.

## A useful algorithm!

Theorem

*Using the Fast Fourier Transform, the formula*

$$(PQ)(X) = \frac{X^d}{\zeta(d+1)} \mathrm{FD}_d(\mathrm{FD}_d(P) * \mathrm{FD}_d(Q)) \left(\frac{1}{\zeta X}\right).$$

*computes the product $PQ$ in*

$\Theta(d \log d)$ *multiplications.*

## But in practice?

- For $K = \mathbb{C}$ we need floating point arithmetic, that's not good.
- For $p$ a prime number, $\mathbb{F}_p = \mathbb{Z}/p\mathbb{Z}$ is a field.
- $\mathbb{F}_p$ contains a primitive $(p-1)$th root of unity.
- Choose $p$ such that $p - 1$ is divisible by a large power of 2.
- For example

$$12289 = 3 \cdot 2^{12} + 1, \quad 40961 = 5 \cdot 2^{13} + 1, \quad 61441 = 15 \cdot 2^{12} + 1.$$

## FFT with coefficients in $\mathbb{F}_p$

- Take $p$ prime with $p - 1$ divisible by $2^e$.
- The multiplication algorithm applies to polynomials of degree $< 2^{e-1}$ with coefficients in $\mathbb{F}_p$.

## Applications

- Error correcting codes
- Cryptography
- Integer arithmetic

## Multiplication of integers

- Write integers in base $R$,

  for example $R = 2^{64}$ on a 64 bit processor.
- $N > 0$ is expressed as

$$N = a_0 + a_1 R + \cdots + a_d R^d = P_N(R)$$

  with $a_i \in \{0, \ldots, R-1\}$.
- Multiplying integers amounts to multiplying polynomials.
- Want to take $p > R$ and work with coefficients in $\mathbb{F}_p$.

## Avoid pitfalls

- The coefficients of $P_N P_M$ may be $\geq R$ so:
  - Take $p \gg R$ such that the coefficients stay $< p$.
  - The coefficients are determined by their reduction modulo $p$.
  - Treat carries to obtain a valid representation in base $R$.

    (Computationally easy.)
- Choose $R$ close to the word-size of the computer.
- Best to have $p$ within the word-size as well.
- But that is too small to get the necessary precision.

## A final trick

- Use several primes $p_1, p_2, \ldots, p_k$ and start by computing $PQ$ modulo $p_1$, modulo $p_2$,...
- By the Chinese Remainder Theorem this determines $PQ$ modulo the product $p_1 p_2 \cdots p_k$.
- For $p_1 p_2 \cdots p_k$ large enough this determines the coefficients in $\mathbb{Z}$.
- Can treat integers up to $R^{2^{e-1}-1}$,

  think of $R = 2^{64}$ and $e = 12$!

This is Pollard's method for integer multiplication.

# References


- T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein.
  *Introduction to algorithms*.
  MIT Press, Cambridge, MA, third edition, 2009.
- M. Demazure.
  *Cours d'algèbre*, Nouvelle Bibliothèque Mathématique 1.
  Cassini, Paris, 1997.