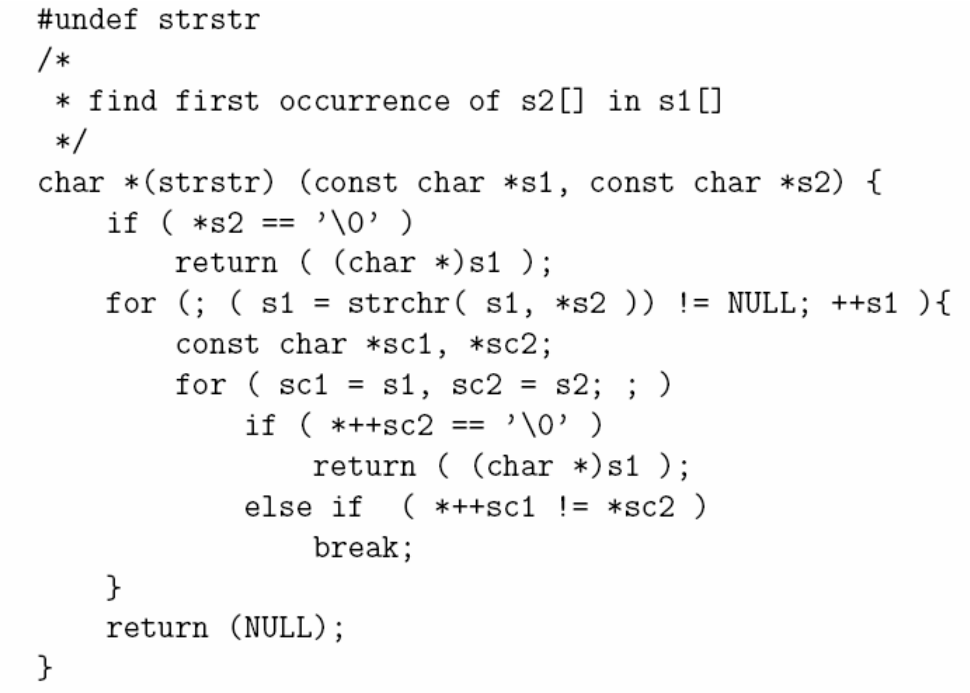
# FFT analysis of DNA sequences

Harvey Lab Group Meeting

March 1, 2004

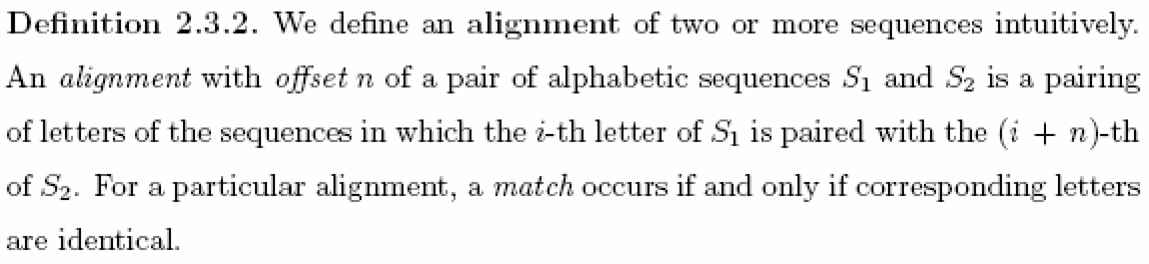
Russell Hanson

# Naïve string search: implementation

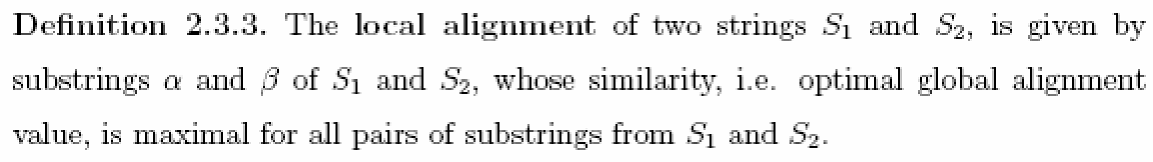


```
#undef strstr
/*
 * find first occurrence of s2[] in s1[]
 */
char *(strstr) (const char *s1, const char *s2) {
    if ( *s2 == ’\0’ )
        return ( (char *)s1 );
    for (; ( s1 = strchr( s1, *s2 )) != NULL; ++s1 ){
        const char *sc1, *sc2;
        for ( sc1 = s1, sc2 = s2; ; )
            if ( *++sc2 == ’\0’ )
                return ( (char *)s1 );
            else if  ( *++sc1 != *sc2 )
                break;
    }
    return (NULL);
}
```

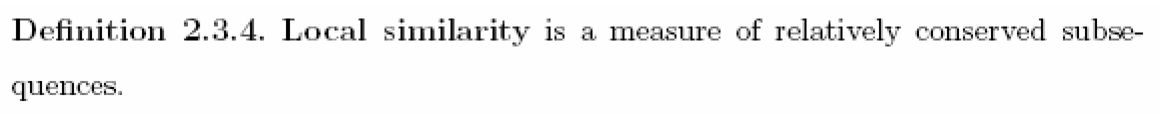
# Alignments local and global



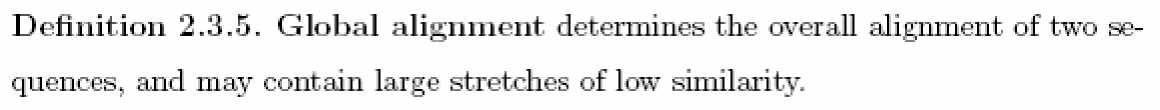
**Definition 2.3.2.** We define an **alignment** of two or more sequences intuitively. An *alignment* with *offset* $n$ of a pair of alphabetic sequences $S_1$ and $S_2$ is a pairing of letters of the sequences in which the $i$-th letter of $S_1$ is paired with the $(i + n)$-th of $S_2$. For a particular alignment, a *match* occurs if and only if corresponding letters are identical.



**Definition 2.3.3.** The **local alignment** of two strings $S_1$ and $S_2$, is given by substrings $\alpha$ and $\beta$ of $S_1$ and $S_2$, whose similarity, i.e. optimal global alignment value, is maximal for all pairs of substrings from $S_1$ and $S_2$.

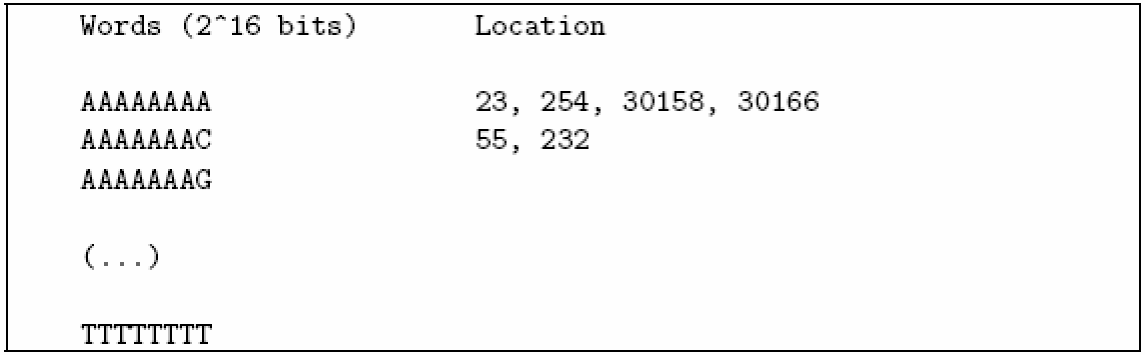


**Definition 2.3.4.** **Local similarity** is a measure of relatively conserved subsequences.



**Definition 2.3.5.** **Global alignment** determines the overall alignment of two sequences, and may contain large stretches of low similarity.

# BLAST



| Words (2^16 bits) | Location |
|---|---|
| AAAAAAAA | 23, 254, 30158, 30166 |
| AAAAAAAC | 55, 232 |
| AAAAAAAG | |
| (...) | |
| TTTTTTTT | |

Figure 2.1: BLAST brute-force table lookup

- Two steps: 1) Hash

  2) Lookup table
- It encodes all the 8-mers as numbers, then it encodes the search string (i.e. chromosome sequence fragment), then it shifts the 8-mers along the search string, building a lookup table of 8-mers and their locations in the search string. For any subsequent search, therefore, you need only compare the hash values of the query string with the lookup table (avoiding working with the search string ever again).

# Fourier transforms

The Fourier transform of a function $f(x)$ is the new function $F(x)$:

$$F(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(u) e^{ixu} du.$$

The $k$-th element $X_k$ of the transformed complex vector $X_0, \ldots, X_{N-1}$ is:

$$X_k = \sum_{j=0}^{N-1} x_j \, e^{-2\pi ijk/N}$$

The inverse Fourier transform reverses the process; it maps $N$ complex numbers (the $X_j$'s) into $N$ complex numbers (the $x_j$'s), i.e. $X \mapsto x$:

$$x_j = \frac{1}{N} \sum_{k=0}^{N-1} X_j \, e^{+2\pi ijk/N}. \qquad (1.11)$$

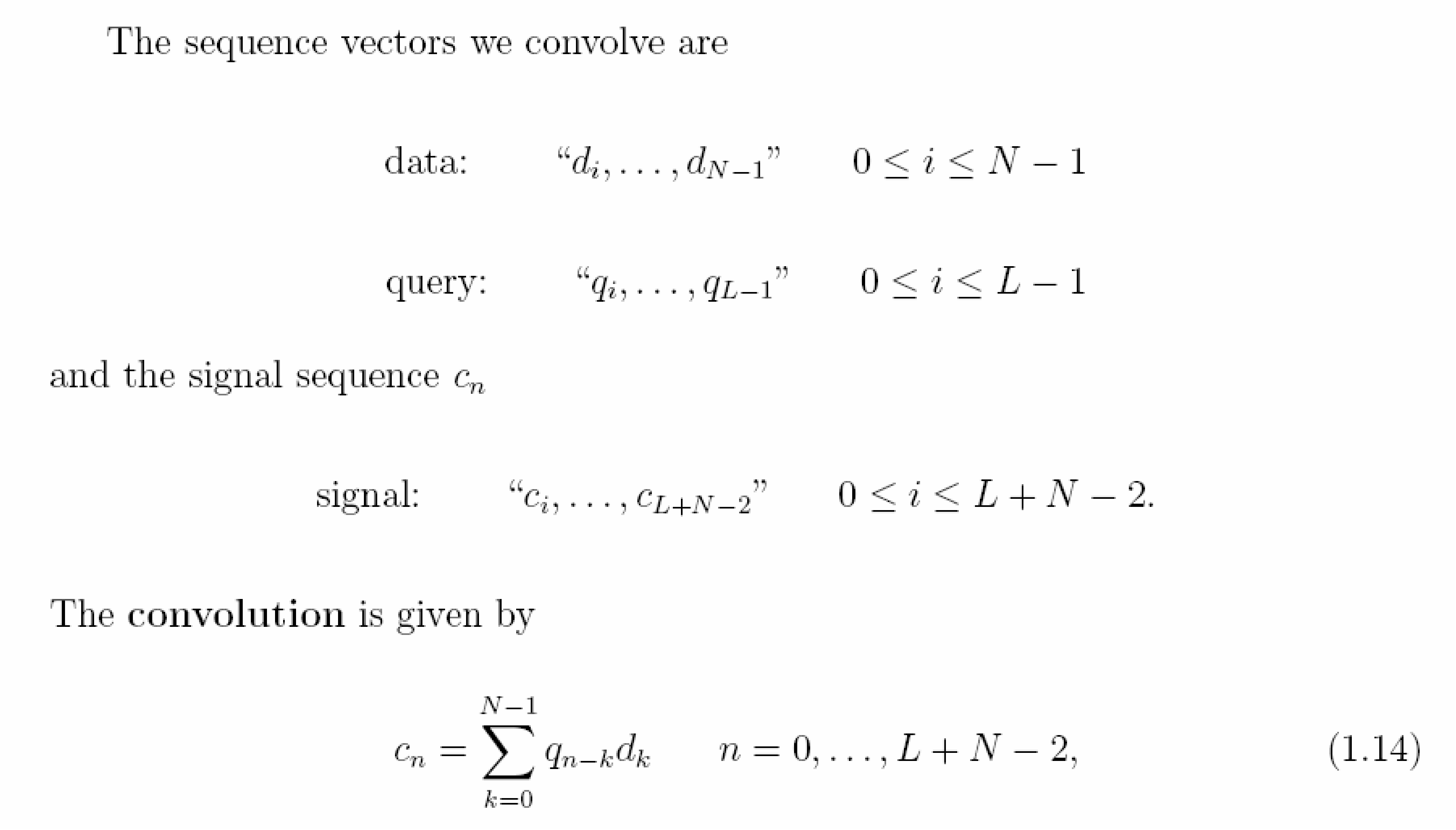
# Fourier transforms II

$$X_k = x_j F_{jk}$$

$$F_{jk} = exp(-2\pi i/n)^{jk} = w_n^{jk}$$

- This is a matrix vector multiplication, which takes O(n) operations.

# Convolution & correlation



The sequence vectors we convolve are

$$\text{data:} \qquad \text{“}d_i, \ldots, d_{N-1}\text{”} \qquad 0 \leq i \leq N-1$$

$$\text{query:} \qquad \text{“}q_i, \ldots, q_{L-1}\text{”} \qquad 0 \leq i \leq L-1$$

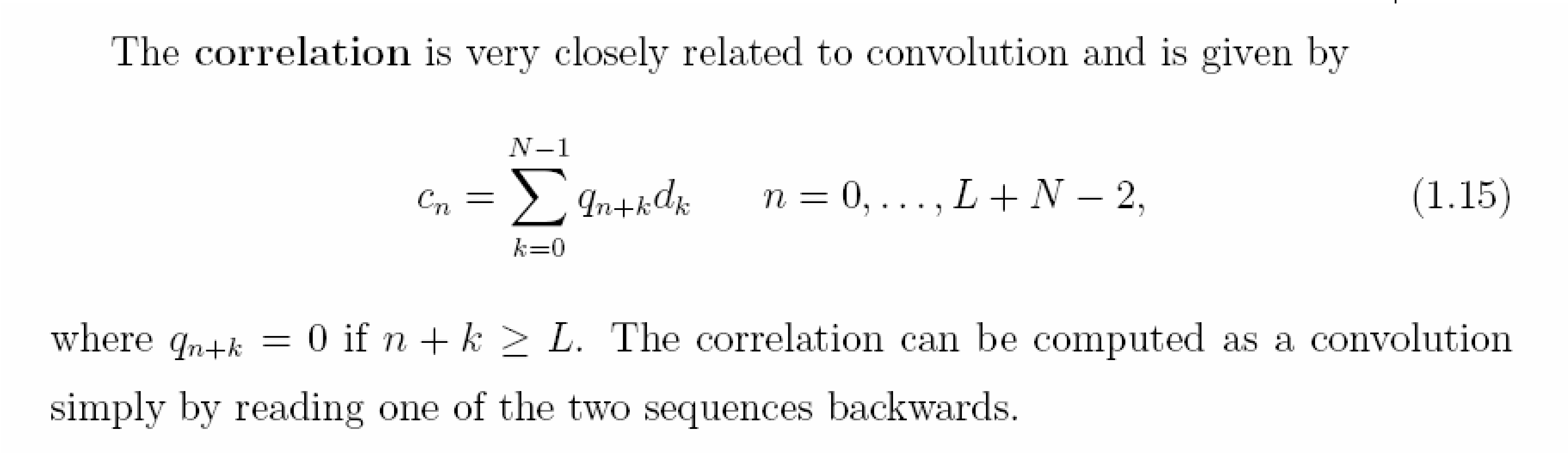
and the signal sequence $c_n$

$$\text{signal:} \qquad \text{“}c_i, \ldots, c_{L+N-2}\text{”} \qquad 0 \leq i \leq L+N-2.$$

The **convolution** is given by

$$c_n = \sum_{k=0}^{N-1} q_{n-k} d_k \qquad n = 0, \ldots, L+N-2, \qquad (1.14)$$

# Convolution & correlation II

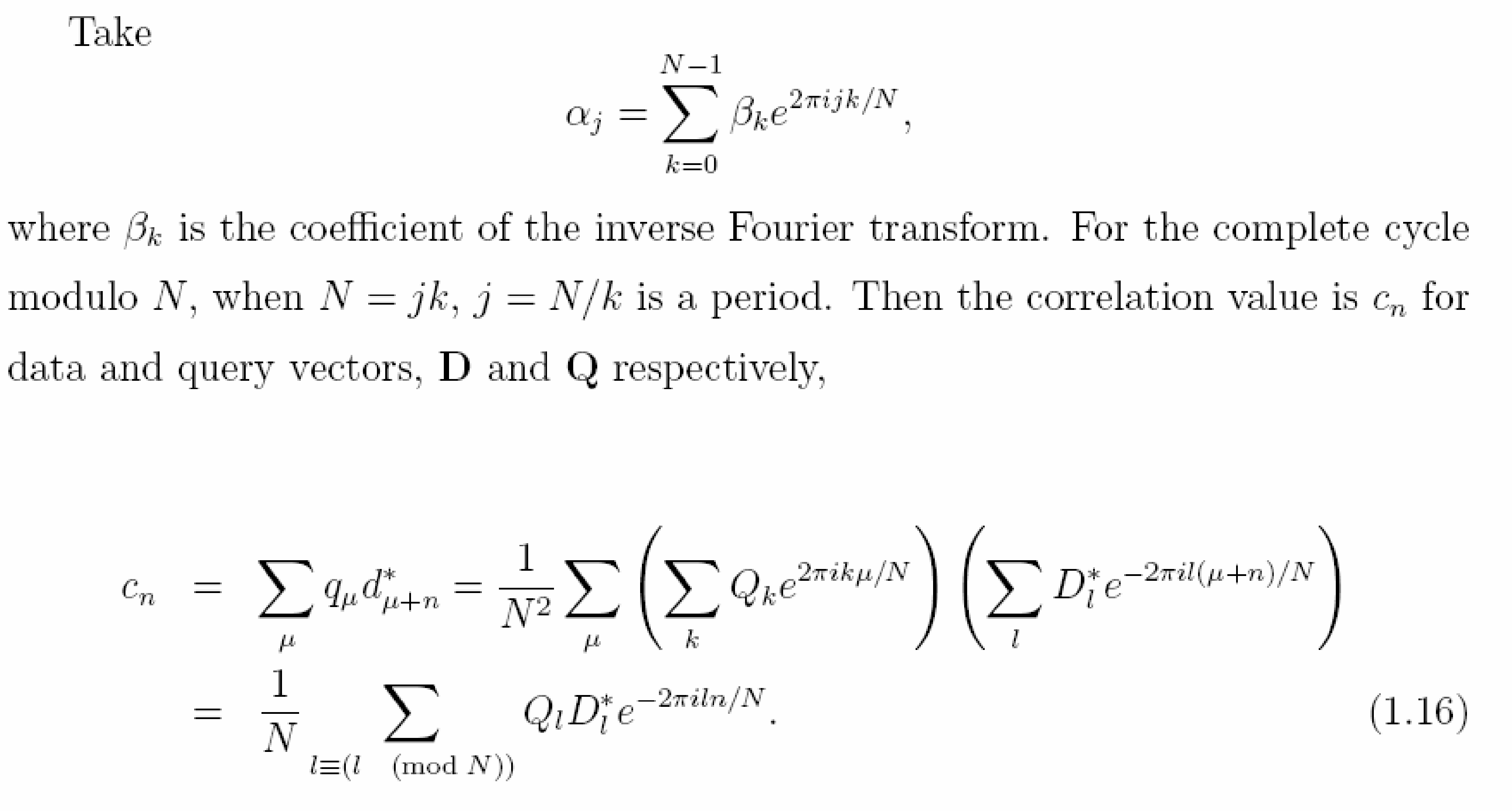


The **correlation** is very closely related to convolution and is given by

$$c_n = \sum_{k=0}^{N-1} q_{n+k} d_k \qquad n = 0, \ldots, L+N-2, \tag{1.15}$$

where $q_{n+k} = 0$ if $n + k \geq L$. The correlation can be computed as a convolution simply by reading one of the two sequences backwards.

# FFT Convolution equation



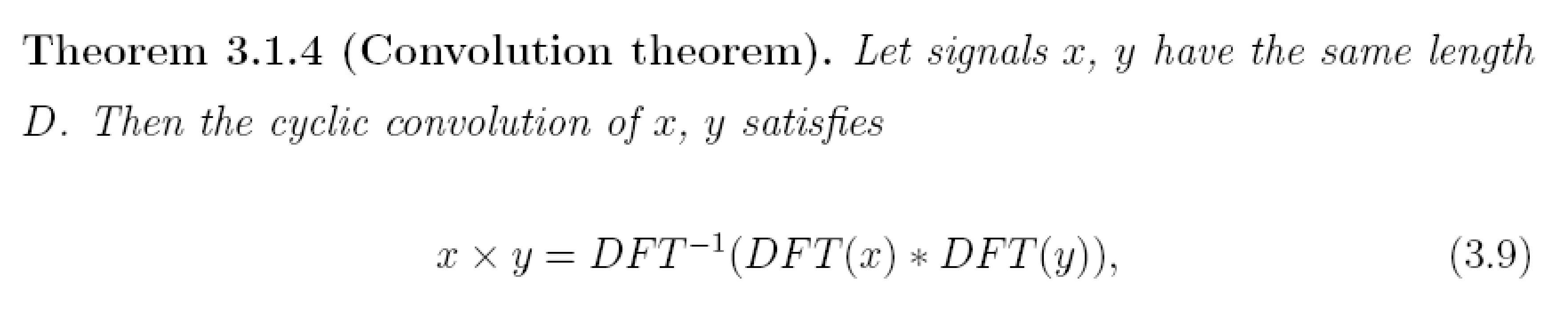
Take

$$\alpha_j = \sum_{k=0}^{N-1} \beta_k e^{2\pi ijk/N},$$

where $\beta_k$ is the coefficient of the inverse Fourier transform. For the complete cycle modulo $N$, when $N = jk$, $j = N/k$ is a period. Then the correlation value is $c_n$ for data and query vectors, $\mathbf{D}$ and $\mathbf{Q}$ respectively,

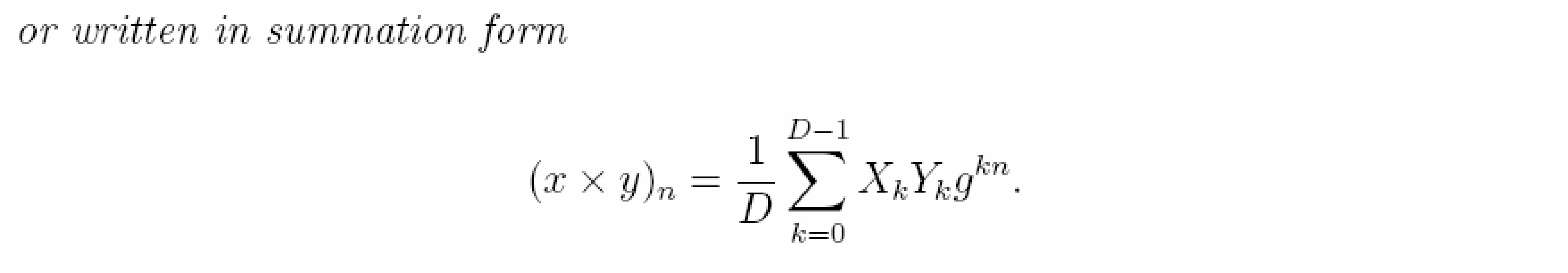
$$\begin{aligned} c_n &= \sum_{\mu} q_\mu d^*_{\mu+n} = \frac{1}{N^2} \sum_{\mu} \left( \sum_{k} Q_k e^{2\pi ik\mu/N} \right) \left( \sum_{l} D^*_l e^{-2\pi il(\mu+n)/N} \right) \\ &= \frac{1}{N} \sum_{l \equiv (l \pmod{N})} Q_l D^*_l e^{-2\pi iln/N}. \qquad (1.16) \end{aligned}$$

# Convolution theorem



**Theorem 3.1.4 (Convolution theorem).** *Let signals $x$, $y$ have the same length $D$. Then the cyclic convolution of $x$, $y$ satisfies*

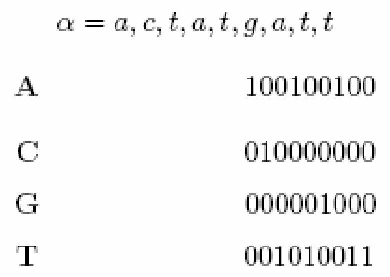
$$x \times y = DFT^{-1}(DFT(x) * DFT(y)), \tag{3.9}$$



*or written in summation form*

$$(x \times y)_n = \frac{1}{D} \sum_{k=0}^{D-1} X_k Y_k g^{kn}.$$

# Base vector encodings



$\alpha = a, c, t, a, t, g, a, t, t$

| | |
|---|---|
| A | 100100100 |
| C | 010000000 |
| G | 000001000 |
| T | 001010011 |

Figure 3.3: 4-Vector complex-plane base encoding

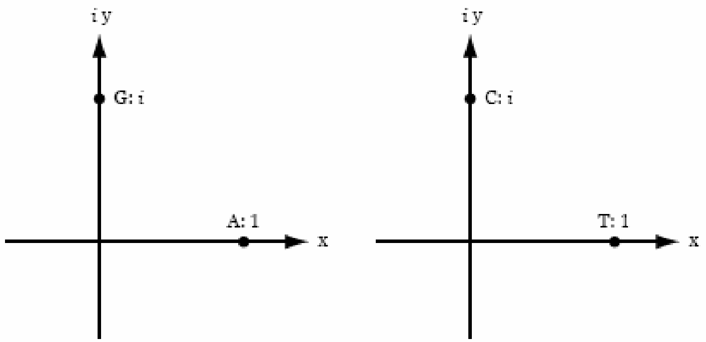


Figure 3.4: 2-Vector complex-plane base encoding

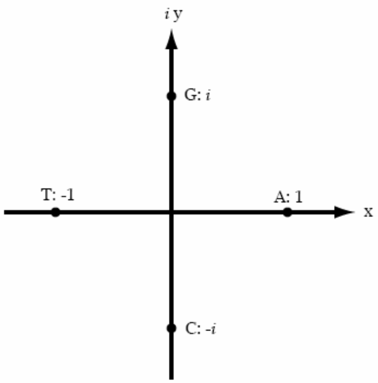
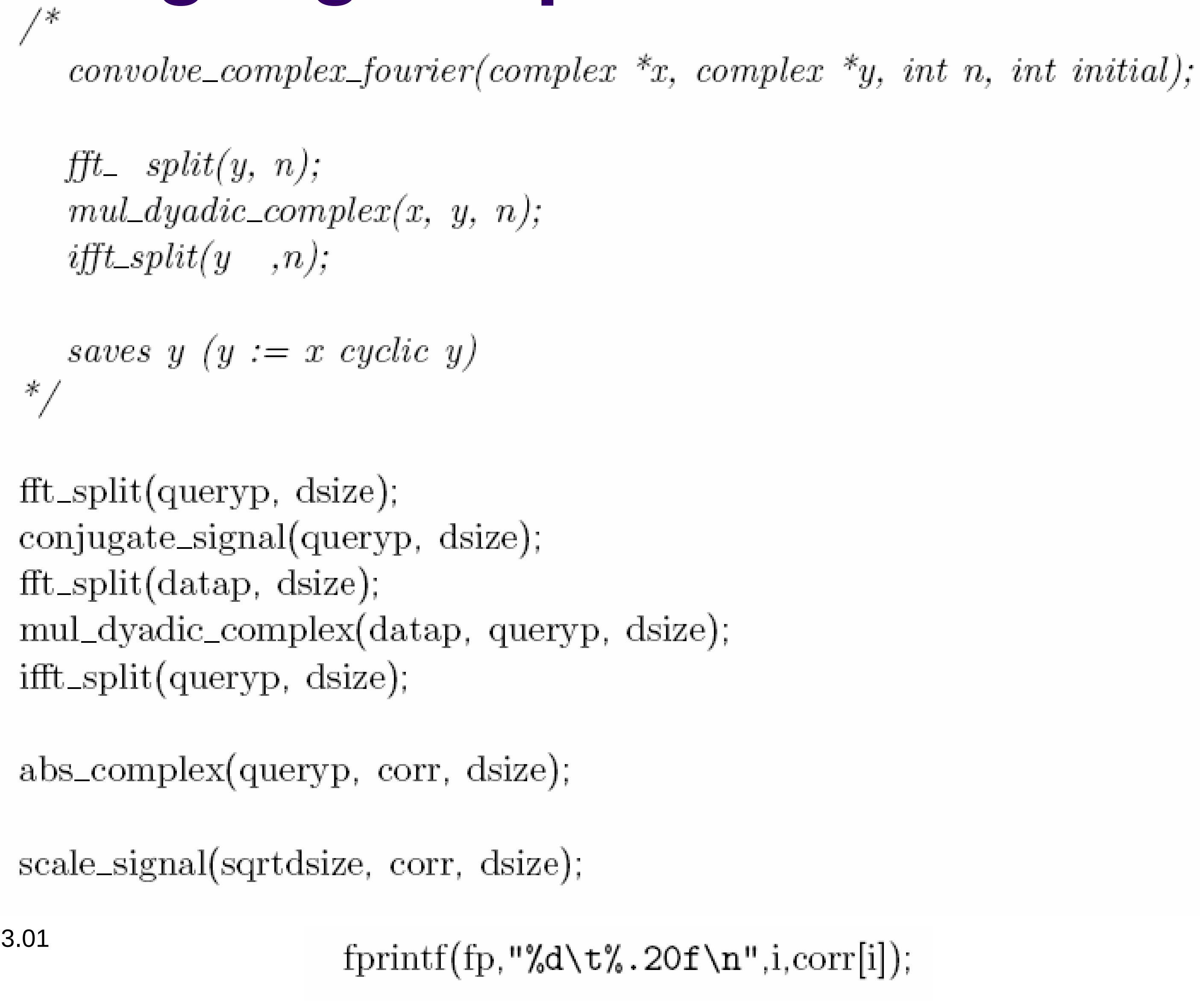


Figure 3.5: 1-Vector complex-plane base encoding

# C Language implementation



```
/*
   convolve_complex_fourier(complex *x, complex *y, int n, int initial);

   fft_  split(y, n);
   mul_dyadic_complex(x, y, n);
   ifft_split(y   ,n);

   saves y (y := x cyclic y)
*/

fft_split(queryp, dsize);
conjugate_signal(queryp, dsize);
fft_split(datap, dsize);
mul_dyadic_complex(datap, queryp, dsize);
ifft_split(queryp, dsize);

abs_complex(queryp, corr, dsize);

scale_signal(sqrtdsize, corr, dsize);

fprintf(fp,"%d\t%.20f\n",i,corr[i]);
```

# Results

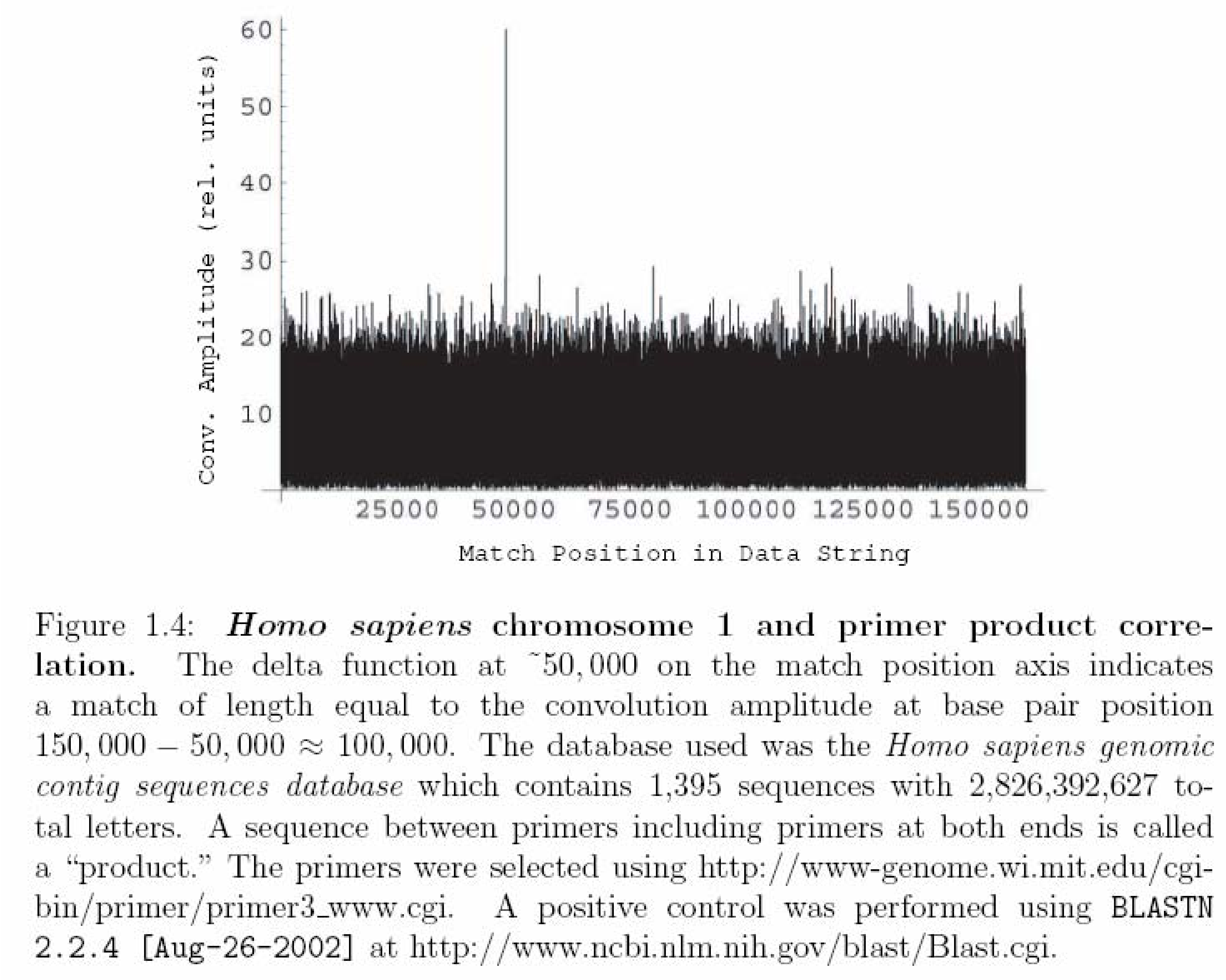
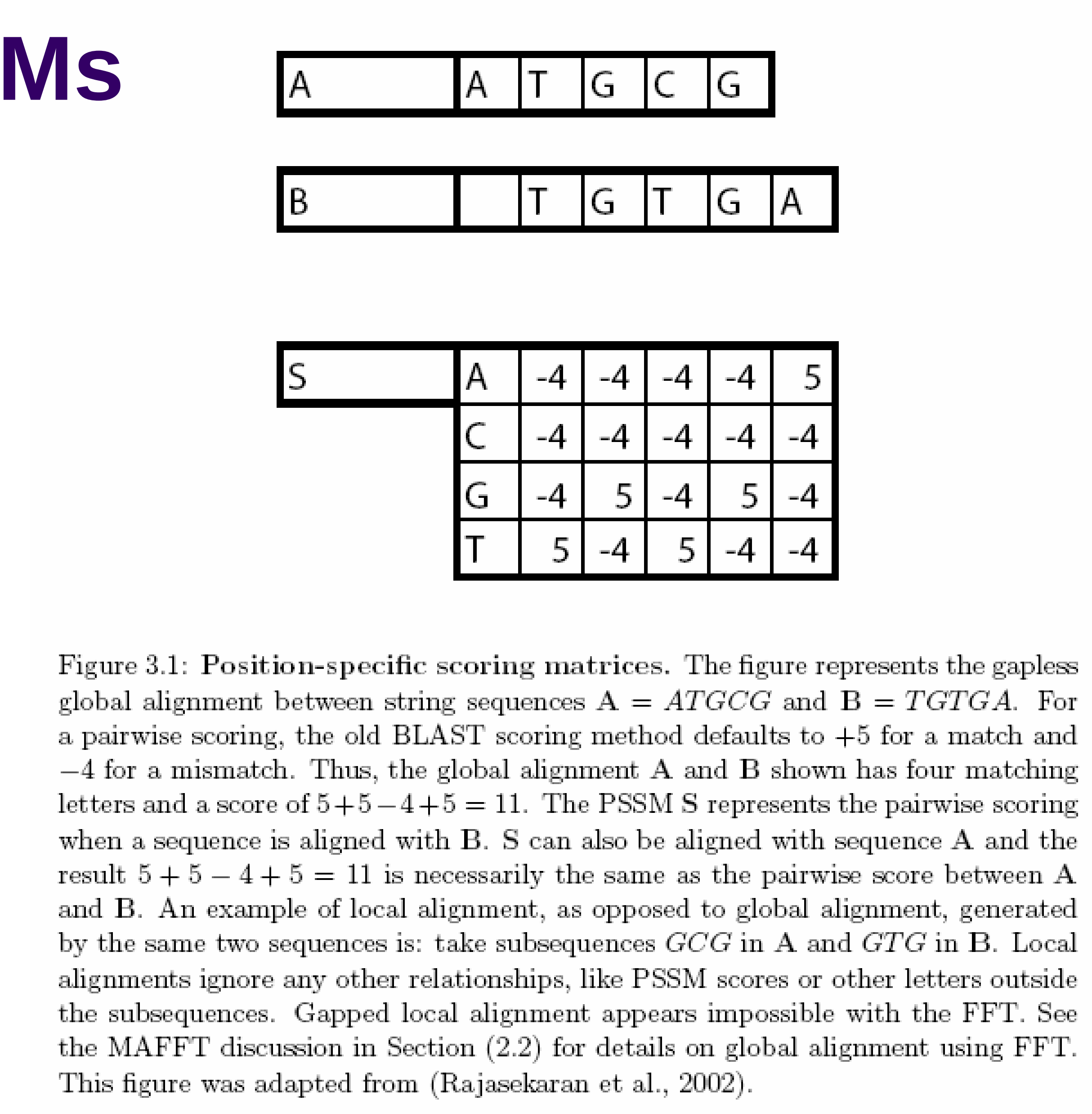


Figure 1.4: ***Homo sapiens* chromosome 1 and primer product correlation.** The delta function at ˜50,000 on the match position axis indicates a match of length equal to the convolution amplitude at base pair position $150,000 - 50,000 \approx 100,000$. The database used was the *Homo sapiens genomic contig sequences database* which contains 1,395 sequences with 2,826,392,627 total letters. A sequence between primers including primers at both ends is called a "product." The primers were selected using http://www-genome.wi.mit.edu/cgi-bin/primer/primer3_www.cgi. A positive control was performed using BLASTN 2.2.4 [Aug-26-2002] at http://www.ncbi.nlm.nih.gov/blast/Blast.cgi.

# PSSMs



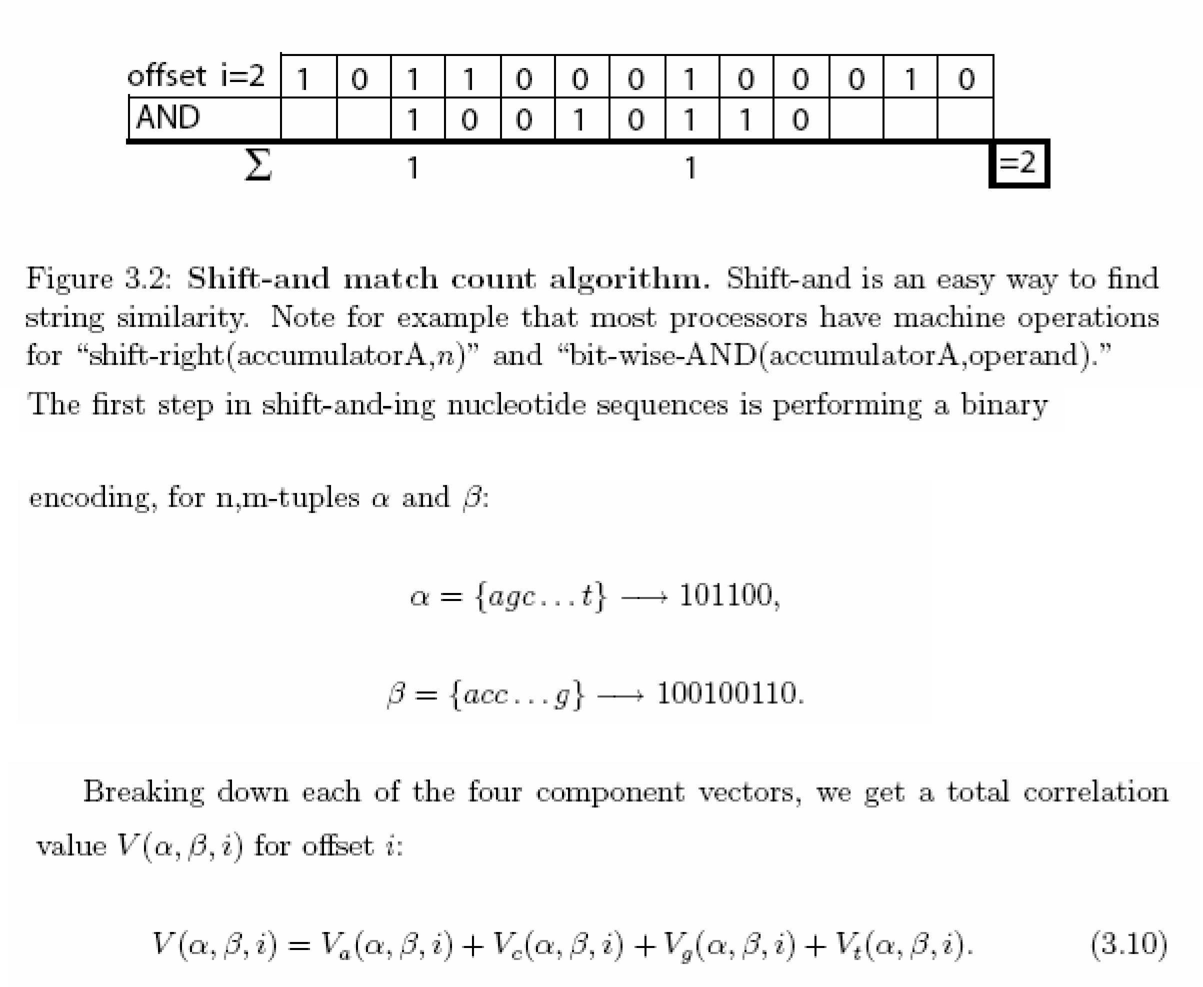
| A | A | T | G | C | G |
|---|---|---|---|---|---|

| B | | T | G | T | G | A |
|---|---|---|---|---|---|---|

| S | | | | | |
|---|---|---|---|---|---|
| | A | -4 | -4 | -4 | -4 | 5 |
| | C | -4 | -4 | -4 | -4 | -4 |
| | G | -4 | 5 | -4 | 5 | -4 |
| | T | 5 | -4 | 5 | -4 | -4 |

Figure 3.1: **Position-specific scoring matrices.** The figure represents the gapless global alignment between string sequences A = *ATGCG* and B = *TGTGA*. For a pairwise scoring, the old BLAST scoring method defaults to +5 for a match and $-4$ for a mismatch. Thus, the global alignment A and B shown has four matching letters and a score of $5+5-4+5 = 11$. The PSSM S represents the pairwise scoring when a sequence is aligned with B. S can also be aligned with sequence A and the result $5 + 5 - 4 + 5 = 11$ is necessarily the same as the pairwise score between A and B. An example of local alignment, as opposed to global alignment, generated by the same two sequences is: take subsequences *GCG* in A and *GTG* in B. Local alignments ignore any other relationships, like PSSM scores or other letters outside the subsequences. Gapped local alignment appears impossible with the FFT. See the MAFFT discussion in Section (2.2) for details on global alignment using FFT. This figure was adapted from (Rajasekaran et al., 2002).

# Shift-and algorithm



| offset i=2 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| AND | | | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | | | | |
| Σ | | | 1 | | | | | 1 | | | | | | =2 |

Figure 3.2: **Shift-and match count algorithm.** Shift-and is an easy way to find string similarity. Note for example that most processors have machine operations for "shift-right(accumulatorA,$n$)" and "bit-wise-AND(accumulatorA,operand)."

The first step in shift-and-ing nucleotide sequences is performing a binary

encoding, for n,m-tuples $\alpha$ and $\beta$:

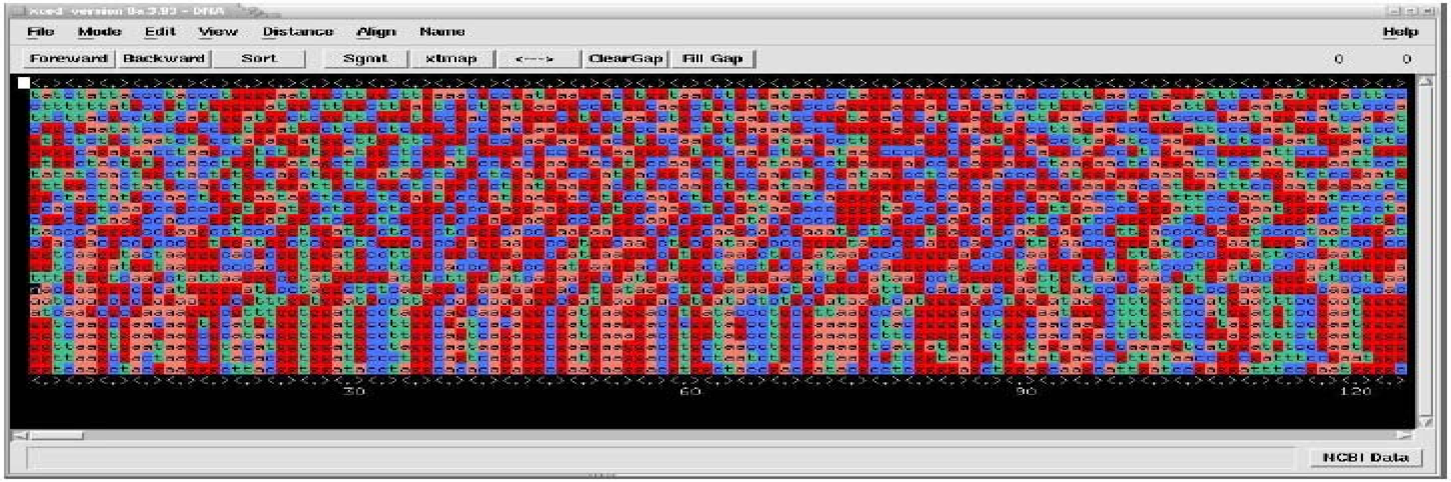
$$\alpha = \{agc \dots t\} \longrightarrow 101100,$$

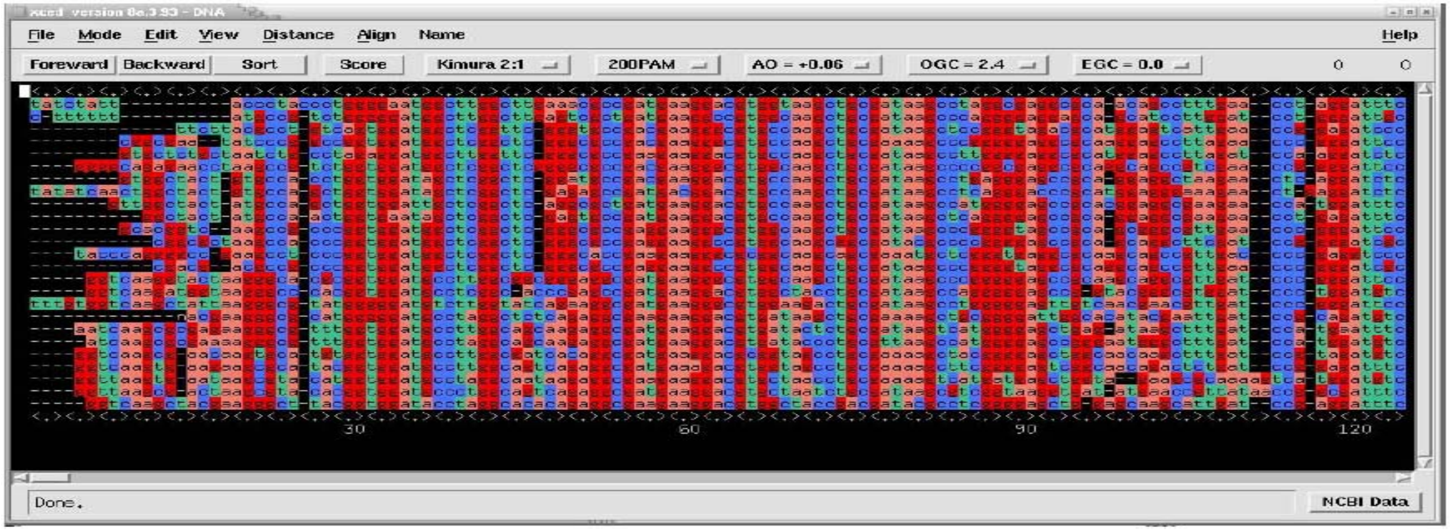
$$\beta = \{acc \dots g\} \longrightarrow 100100110.$$

Breaking down each of the four component vectors, we get a total correlation value $V(\alpha, \beta, i)$ for offset $i$:

$$V(\alpha, \beta, i) = V_a(\alpha, \beta, i) + V_c(\alpha, \beta, i) + V_g(\alpha, \beta, i) + V_t(\alpha, \beta, i). \qquad (3.10)$$

# MAFFT – Multi-Alignment FFT





Kimura 2:1

200PAM

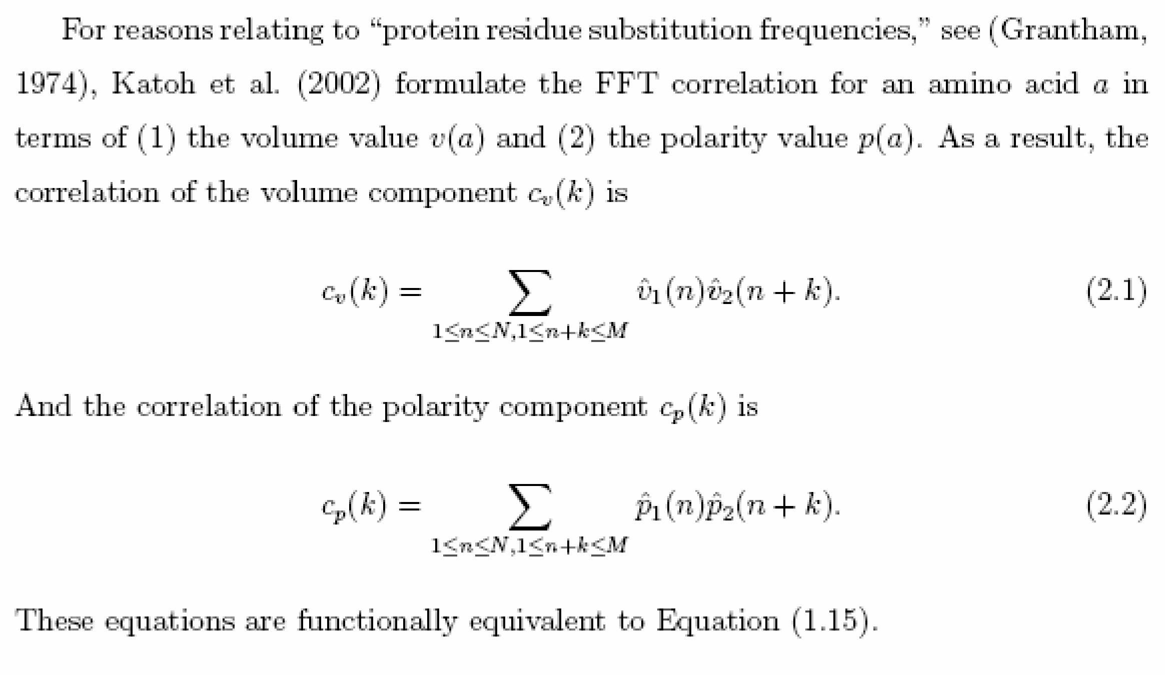
AO = +0.06

OGC = 2.4

EGC = 0.0

Done.

# FFT for global alignment



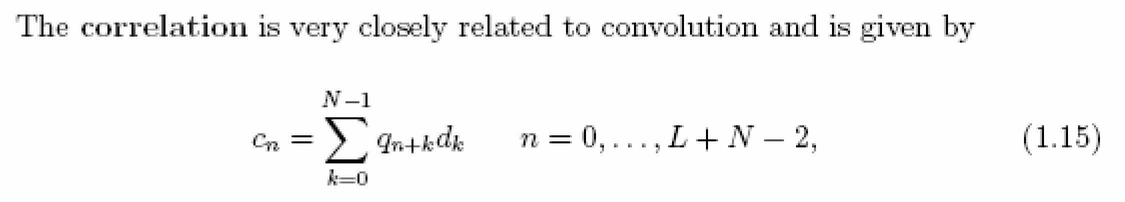
For reasons relating to "protein residue substitution frequencies," see (Grantham, 1974), Katoh et al. (2002) formulate the FFT correlation for an amino acid $a$ in terms of (1) the volume value $v(a)$ and (2) the polarity value $p(a)$. As a result, the correlation of the volume component $c_v(k)$ is

$$c_v(k) = \sum_{1 \le n \le N, 1 \le n+k \le M} \hat{v}_1(n)\hat{v}_2(n+k). \qquad (2.1)$$

And the correlation of the polarity component $c_p(k)$ is

$$c_p(k) = \sum_{1 \le n \le N, 1 \le n+k \le M} \hat{p}_1(n)\hat{p}_2(n+k). \qquad (2.2)$$

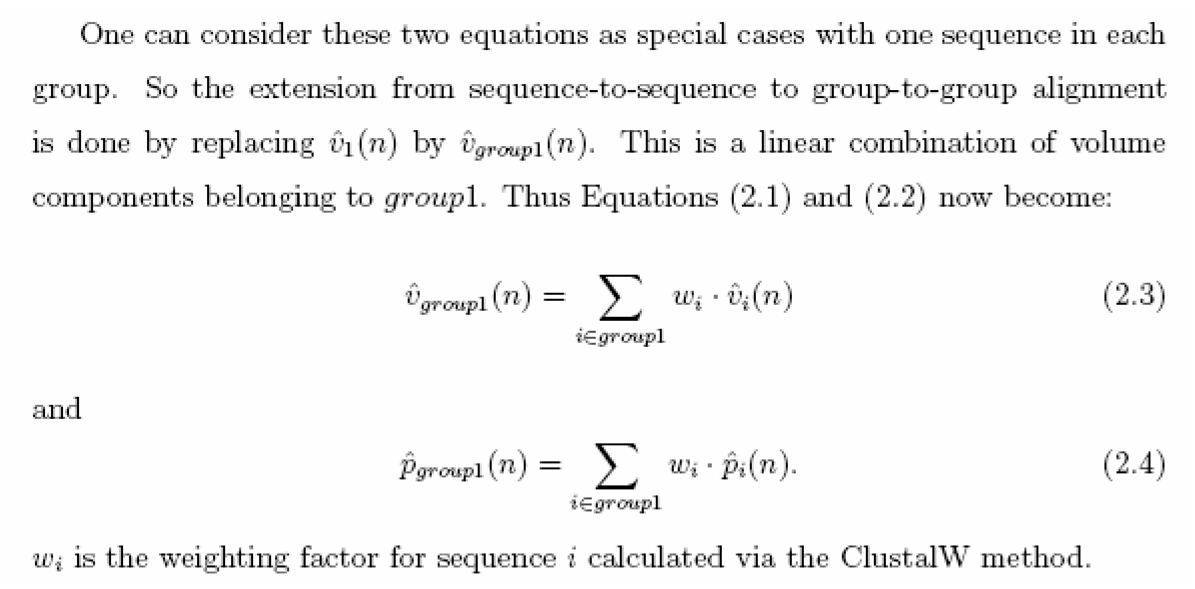
These equations are functionally equivalent to Equation (1.15).



The correlation is very closely related to convolution and is given by

$$c_n = \sum_{k=0}^{N-1} q_{n+k} d_k \qquad n = 0, \ldots, L+N-2, \qquad (1.15)$$

# Group-group alignment



One can consider these two equations as special cases with one sequence in each group. So the extension from sequence-to-sequence to group-to-group alignment is done by replacing $\hat{v}_1(n)$ by $\hat{v}_{group1}(n)$. This is a linear combination of volume components belonging to $group1$. Thus Equations (2.1) and (2.2) now become:

$$\hat{v}_{group1}(n) = \sum_{i \in group1} w_i \cdot \hat{v}_i(n) \tag{2.3}$$

and

$$\hat{p}_{group1}(n) = \sum_{i \in group1} w_i \cdot \hat{p}_i(n). \tag{2.4}$$

$w_i$ is the weighting factor for sequence $i$ calculated via the ClustalW method.