Process for Forward Fast Fourier Transform (FFT) Numerical Computation

Description

Forward FFT takes time domain data points and converts them to frequency domain coefficients.

Process Overview

Solve for complex numbers (C_k), then convert to a, b coefficients for use in trigonometric interpolation equation.

Prerequisites

- N = Number of data points. N must be a power of 2, can zero-pad otherwise. Data points must be evenly spaced on $[-\pi, \pi]$.
- m = Degree. Higher solves for higher frequencies in the approximation. $m \leq N/2$

Step 1) Generate Data Points

For data points (x_j, y_j) on $[-\pi, \pi]$:

$$x_j = -\pi + \frac{2\pi j}{N}, \quad j = 0, 1, \ldots, N-1$$

$$y_j = f(x_j), \quad j = 0, 1, \ldots, N-1$$

Step 2) Pre-compute setup values:

a) Pre-computed "Roots of Unity":

$$W_k = e^{-2\pi ik/N} = \cos(2\pi k/N) - i\sin(2\pi k/N), \quad k = 0, 1, \ldots, (N/2) - 1$$

Note: Only need to compute each W_k once.

Note: W values "loop" around a unit circle for higher values.

ex: $N = 8$, with $\log_2(8) = 3$:

$$W_0 = e^{-2\pi i(0)/8} = 1.0000 + 0.0000i \qquad (0^\circ)$$
$$W_1 = e^{-2\pi i(1)/8} = 0.7071 - 0.7071i \qquad (45^\circ)$$
$$W_2 = e^{-2\pi i(2)/8} = 0.0000 - 1.0000i \qquad (90^\circ)$$
$$W_3 = e^{-2\pi i(3)/8} = -0.7071 - 0.7071i \qquad (135^\circ)$$

$$W_4 = -W_0 = -1.0000 + 0.0000i \qquad (180^\circ)$$
$$W_5 = -W_1 = -0.7071 + 0.7071i \qquad (225^\circ)$$

...

Loop rules:

- $W_{k+4} = -W_k$ (opposite side of circle)
- $W_{k+8} = W_k$ (full rotation)

b) Initial complex values array:

i) Set initial complex values to y values:

$$C_j = y_j, \quad j = 0, 1, \ldots, N-1$$

ii) Bit-reverse the complex values index positions so butterfly pairing is correct. For array size N, each index i (for 0 to $N-1$) is found by:

(a) Convert to binary in $\log_2(N)$ bits

(b) Reverse the bits

(c) Convert back to decimal

(d) Re-order complex pairs with new index

Ex: $N = 8$, with $\log_2(8) = 3$ bits:

i	binary	reversed	new i
0	000	000	0
1	001	100	4
2	010	010	2
3	011	110	6
4	100	001	1
5	101	101	5
6	110	011	3
7	111	111	7

$$C_0 \to C_0, C_1 \to C_4, C_2 \to C_2, C_3 \to C_6, \text{ etc.}$$

Note: Only need to compute bit-reversal index table once.

Step 3) Butterfly Pairs

Combine complex values in pairs over stages, updating values one step at a time, until the final complex values are fully computed. Pairs of C_k are found by increasing the distance when grouping the arrays:

- Stage Number: $s = 1, 2, \ldots, \log_2(N)$
- Distance between pairs in stage: $d = 2^{s-1}$

For example, with $N = 8$ and $\log_2(8) = 3$ stages:

$$\begin{aligned}
&\text{Stage 1 (distance = 1):} && [0], [1], [2], [3], [4], [5], [6], [7] \to \{(0,1), (2,3), (4,5), (6,7)\} \\
&\text{Stage 2 (distance = 2):} && [0,1], [2,3], [4,5], [6,7] \to \{(0,2), (1,3), (4,6), (5,7)\} \\
&\text{Stage 3 (distance = 4):} && [0,1,2,3], [4,5,6,7] \to \{(0,4), (1,5), (2,6), (3,7)\}
\end{aligned}$$

For each pair, perform the following (N = total # of points, d = distance between pairs):

$$\begin{aligned}
\eta &= C_{k+d} \times W_{(k \times N/2d)} \\
C_k &= C_k + \eta \\
C_{k+d} &= C_k - \eta
\end{aligned}$$

Step 4) Coefficient Extraction

Convert complex values to a_k and b_k values:

$$a_0 = \frac{1}{N} Re(C_0)$$

$$-a_k = \frac{2}{N} Re(C_k) \qquad b_k = \frac{2}{N} Im(C_k) \quad \text{, for odd } k = 1, 3, \ldots, (N/2) - 1$$

$$a_k = \frac{2}{N} Re(C_k) \qquad -b_k = \frac{2}{N} Im(C_k) \quad \text{, for even } k = 2, 4, \ldots, (N/2) - 1$$

$$a_{N/2} = \frac{1}{N} Re(C_{N/2}), \quad \text{for } k = N/2$$

Step 5) Construct Fourier series approximation

Interpolating Polynomial Equation for FFT:

$$S_m(x) = \frac{a_0 + a_m \cos(mx)}{2} + \sum_{k=1}^{m-1} (a_k \cos(kx) + b_k \sin(kx))$$