Fast Fourier Transform.

Author: Arkadi Kagan.
arkadi_kagan@hotmail.com

This text do not contain generalized prove of what is FFT, what are requirements and strict limitations for FFT to be a valid approach. What I do describe here is Fast Fourier Transform for Complex and Real Vectors, based on set of Pairwise Orthogonal Functions of the form $e^{j 2 \pi \frac{nk}{N}}$, $\cos(2 \pi \frac{nk}{N})$ and $\sin(2 \pi \frac{nk}{N})$. Else I restrict FFT to the only its "Radix 2" variant.

Lets start from reminding Discrete Fourier Transform formulas that can be found in a lot of sources. The forward DFT, given a vector of complex values, compute Fourier coefficients:

$$F(k) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} f(n) e^{-j 2 \pi \frac{nk}{N}}$$

and corresponding inverse transform:

$$f(n) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} F(k) e^{j 2 \pi \frac{nk}{N}} .$$

From the above formulas we get the simplest algorithm to calculate DFT:

```
DFT(N, f)
    for k = 0 to N-1
        S = 0
        for n = 0 to N-1
            S += f(n) e^(-j 2 π nk/N)
        F(k) = 1/√N S
    return F
```

The same algorithm can be written for the inverse transform.
This algorithm, sometimes refered as Direct DFT is very simple. However, cosmetic changes will not change a fact that Direct DFT involve N^2 operations loop.

Equations needed for Fast Transforms.

Before getting close to the algorithm itself I want to prove a set of trivial trigonometric equations that will have a use in the farther computations.

$$e^{-j2\pi\frac{x(k+N)}{N}} = e^{-j2\pi\frac{xk}{N}} \cdot e^{-j2\pi x}$$

$e^{-j2\pi x} = \cos(2\pi x) - j \cdot \sin(2\pi x)$ by Euler formula.

Remind that $\cos(2\pi x) = 1$ and $\sin(2\pi x) = 0$ for integer x.

Conclusion:

$e^{-j2\pi\frac{x(k+N)}{N}} = e^{-j2\pi\frac{xk}{N}}$ for any integer x.

$\cos(2\pi\frac{x(k+N)}{N})$ = ($\cos(\alpha+\beta) = \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta)$)

$= \cos(2\pi\frac{xk}{N})\cos(2\pi x) - \sin(2\pi\frac{xk}{N})\sin(2\pi x)$ =

$(\begin{matrix}\cos(2\pi x) = 1 \\ \sin(2\pi x) = 0\end{matrix}\}$ for any integer x) = $\cos(2\pi\frac{xk}{N})$

Conclusion:

$\cos(2\pi\frac{x(k+N)}{N}) = \cos(2\pi\frac{xk}{N})$ for any integer x.

$\sin(2\pi\frac{x(k+N)}{N})$ = ($\sin(\alpha+\beta) = \sin(\alpha)\cos(\beta) + \cos(\alpha)\sin(\beta)$)

$= \sin(2\pi\frac{xk}{N})\cos(2\pi x) - \cos(2\pi\frac{xk}{N})\sin(2\pi x)$ =

$(\begin{matrix}\cos(2\pi x) = 1 \\ \sin(2\pi x) = 0\end{matrix}\}$ for any integer x) = $\sin(2\pi\frac{xk}{N})$

Conclusion:

$\sin(2\pi\frac{x(k+N)}{N}) = \sin(2\pi\frac{xk}{N})$ for any integer x.

$$e^{-j2\pi\frac{x(k+N/2)}{N}} = e^{-j2\pi\frac{xk}{N}} \cdot e^{-j\pi x}$$

By Euler formula $e^{-j\pi x} = \cos(\pi x) - j \cdot \sin(\pi x) = \cos(\pi x)$ since $\sin(\pi x) = 0$ for any integer x. This mean:

$$e^{-j\pi x} = \begin{cases} -1 & \textit{for x odd integer} \\ 1 & \textit{for x even integer} \end{cases}$$

Conclusion:

$$e^{-j2\pi\frac{x(k+N/2)}{N}} = \begin{cases} -e^{-j2\pi\frac{xk}{N}} & \textit{for x odd integer} \\ +e^{-j2\pi\frac{xk}{N}} & \textit{for x even integer} \end{cases}$$

$$\cos\left(2\pi\frac{x(k+N/2)}{N}\right) = \left(\ \cos(\alpha+\beta) = \cos(\alpha)\cos(\beta) - \sin(\alpha)\sin(\beta)\ \right)$$

$$= \cos\left(2\pi\frac{xk}{N}\right)\cos(\pi x) - \sin\left(2\pi\frac{xk}{N}\right)\sin(\pi x) =$$

$\sin(\pi x) = 0$ for any integer x

$$\left(\ \cos(\pi x) = \begin{cases} -1 & \text{for odd integer x} \\ 1 & \text{for even integer x} \end{cases}\ \right) = \cos\left(2\pi\frac{xk}{N}\right)\cos(\pi x)$$

Conclusion:

$$\cos\left(2\pi\frac{x(k+N/2)}{N}\right) = \begin{cases} -\cos\left(2\pi\frac{xk}{N}\right) & \text{for odd integer x} \\ +\cos\left(2\pi\frac{xk}{N}\right) & \text{for even integer x} \end{cases}$$

$$\sin\left(2\pi\frac{x(k+N/2)}{N}\right) = \left(\ \sin(\alpha+\beta) = \sin(\alpha)\cos(\beta) + \cos(\alpha)\sin(\beta)\ \right)$$

$$= \sin\left(2\pi\frac{xk}{N}\right)\cos(\pi x) + \cos\left(2\pi\frac{xk}{N}\right)\sin(\pi x) =$$

$$\left(\text{ since } \sin(\pi x) = 0\ \right) = \sin\left(2\pi\frac{xk}{N}\right)\cos(\pi x)$$

Conclusion:

$$\sin\left(2\pi\frac{x(k+N/2)}{N}\right) = \begin{cases} -\sin\left(2\pi\frac{xk}{N}\right) & \text{for odd integer x} \\ +\sin\left(2\pi\frac{xk}{N}\right) & \text{for even integer x} \end{cases}$$

Fast Transform.

Fast Transform algorithm can be of two kinds: Decimation In Time (DIT) and Decimation In Frequency (DIF). Decimation In Time FFT is a bit simpler for understanding and that for I will start from this variant.

Recall Forward DFT formula:

$$F(k)=\frac{1}{\sqrt{N}}\sum_{n=0}^{N-1} f(n)e^{-j2\pi\frac{nk}{N}}$$

The form of DFT formula is a sum of N elements. Split this summation to the odd and even elements:

$$F(k)=\frac{1}{\sqrt{N}}\left(\sum_{n=0}^{\frac{N}{2}-1} f(2n)e^{-j2\pi\frac{(2n)k}{N}}+\sum_{n=0}^{\frac{N}{2}-1} f(2n+1)e^{-j2\pi\frac{(2n+1)k}{N}}\right) .$$

$$\sum_{n=0}^{\frac{N}{2}-1} f(2n)e^{-j2\pi\frac{(2n)k}{N}}=\sum_{n=0}^{\frac{N}{2}-1} f(2n)e^{-j2\pi\frac{nk}{(N/2)}} .$$

Notice that last summation is factored DFT. Introduce new variables:

$$f_e(n)=f(2n)$$

$$F_e(k)=\frac{1}{\sqrt{N/2}}\sum_{n=0}^{\frac{N}{2}-1} f_e(n)e^{-j2\pi\frac{nk}{(N/2)}}$$

Here F_e is DFT of even members of the vector f, marked f_e. There is exactly $\frac{N}{2}$ even elements of vector f, if N is an even number. It is common to restrict FFT to work with values of N that are power of 2. In this case there is no problem of odd N.

$$\sum_{n=0}^{\frac{N}{2}-1} f(2n+1)e^{-j2\pi\frac{(2n+1)k}{N}}=\sum_{n=0}^{\frac{N}{2}-1} f(2n+1)e^{-j2\pi\frac{2nk}{N}}e^{-j2\pi\frac{k}{N}}=$$

$$\left(\sum_{n=0}^{\frac{N}{2}-1} f(2n+1)e^{-j2\pi\frac{nk}{N/2}}\right)e^{-j2\pi\frac{k}{N}}$$

Define variables for odd elements of vector f:

$$f_o(n)=f(2n+1)$$

$$F_o(k)=\frac{1}{\sqrt{N/2}}\sum_{n=0}^{\frac{N}{2}-1} f_o(n)e^{-j2\pi\frac{nk}{N/2}}$$

Finally we get

$$F(k)=\frac{1}{\sqrt{2}}F_e(k)+\frac{1}{\sqrt{2}}F_o(k)e^{-j2\pi\frac{k}{N}}$$

This equation allow to write the basic loop of DIT FFT algorithm:

for k = 0 to N-1

$$F(k)=\frac{1}{\sqrt{2}}F_e(k)+\frac{1}{\sqrt{2}}F_o(k)e^{-j2\pi\frac{k}{N}}$$

This loop is computing right F(k), but unfortunately only for $k<\frac{N}{2}$. Lets compute $F(k+\frac{N}{2})$ to get the reminded values.

$$F(k+\frac{N}{2})=\frac{1}{\sqrt{N}}(\sum_{n=0}^{N/2-1}f(2n)e^{-j2\pi\frac{(2n)(k+N/2)}{N}}+\sum_{n=0}^{N/2-1}f(2n+1)e^{-j2\pi\frac{(2n+1)(k+N/2)}{N}})$$

Remind that: for even $x=2n$ $e^{-j2\pi\frac{x(k+N/2)}{N}}=e^{-j2\pi\frac{xk}{N}}=e^{-j2\pi\frac{(2n)k}{N}}$

and for odd $x=2n+1$ $e^{-j2\pi\frac{x(k+N/2)}{N}}=-e^{-j2\pi\frac{xk}{N}}=-e^{-j2\pi\frac{(2n+1)k}{N}}$.

By using notation of F_e and F_o we can rewrite:

$$F(k+N/2)=\frac{1}{\sqrt{2}}F_e(k)-\frac{1}{\sqrt{2}}F_o(k)e^{-j2\pi\frac{k}{N}}$$

The remainder of the algorithm is to supply particles to the main loop:

$DIT(N, f)$
if N is equal to 1
then return f
for $n = 0$ to $\frac{N}{2} - 1$
$f_e(n) = f(2n)$
$f_o(n) = f(2n+1)$
$F_e = DIT(\frac{N}{2}, f_e)$
$F_o = DIT(\frac{N}{2}, f_o)$
for $k = 0$ to $\frac{N}{2} - 1$
$F(k) = \frac{1}{\sqrt{2}} F_e(k) + \frac{1}{\sqrt{2}} F_o(k) e^{-j2\pi \frac{k}{N}}$
$F(k + N/2) = \frac{1}{\sqrt{2}} F_e(k) - \frac{1}{\sqrt{2}} F_o(k) e^{-j2\pi \frac{k}{N}}$
return F

This is the basic DIT-FFT algorithm.
It can be optimized in a lot of ways and there are a lot of variants exist already.
Reverse FFT is symmetric to the forward FFT. Recall IDFT formula:

$$f(n) = \sum_{k=0}^{N-1} F(k) e^{j2\pi \frac{nk}{N}}$$

By using the same semantics we can write the main loop of the reverse DIT FFT like this:

for $n = 0$ to $\frac{N}{2} - 1$
$f(n) = \frac{1}{\sqrt{2}} f_e(n) + \frac{1}{\sqrt{2}} f_o(n) e^{j2\pi \frac{n}{N}}$
$f(n + N/2) = \frac{1}{\sqrt{2}} f_e(n) - \frac{1}{\sqrt{2}} f_o(n) e^{j2\pi \frac{n}{N}}$

The whole DIT IFFT algorithm can be rewritten this way:

$IDIT\,(N\,,F\,)$
if N is equal to 1
then return F
for $k=0$ to $\frac{N}{2}-1$
$F_e(k)=F(2k)$
$F_o(k)=F(2k+1)$
$f_e=IDIT\,(\frac{N}{2}\,,F_e)$
$f_o=IDIT\,(\frac{N}{2}\,,F_o)$
for $n=0$ to $\frac{N}{2}-1$
$f(n)=\frac{1}{\sqrt{2}}f_e(n)+\frac{1}{\sqrt{2}}f_o(n)e^{j2\pi\frac{n}{N}}$
$f(n+N/2)=\frac{1}{\sqrt{2}}f_e(n)-\frac{1}{\sqrt{2}}f_o(n)e^{j2\pi\frac{n}{N}}$
return f

For this algorithm we can apply the same optimizations that can be used for the forward transform.

Decimation In Frequency.

The second variant of FFT algorithm is Decimation In Frequency (DIF).
Recall forward DFT formula:

$$F(k)=\frac{1}{\sqrt{N}}\sum_{n=0}^{N-1}f(n)e^{-j2\pi\frac{nk}{N}}$$

Instead of dividing the sum to the odd and even addendums, the sum can be divided simple by its summation from 0 to $\frac{N}{2}-1$ and from $\frac{N}{2}$ to $N-1$:

$$F(k)=\frac{1}{\sqrt{N}}\sum_{n=0}^{N-1} f(n)e^{-j2\pi\frac{nk}{N}}=$$

$$=\frac{1}{\sqrt{N}}\left(\sum_{n=0}^{\frac{N}{2}-1} f(n)e^{-j2\pi\frac{nk}{N}}+\sum_{n=\frac{N}{2}}^{N-1} f(n)e^{-j2\pi\frac{nk}{N}}\right)=$$

$$=\frac{1}{\sqrt{N}}\left(\sum_{n=0}^{\frac{N}{2}-1} f(n)e^{-j2\pi\frac{nk}{N}}+\sum_{n=0}^{\frac{N}{2}-1} f(n+\frac{N}{2})e^{-j2\pi\frac{(n+\frac{N}{2})k}{N}}\right)=$$

$$=\frac{1}{\sqrt{N}}\sum_{n=0}^{\frac{N}{2}-1}\left(f(n)+f(n+\frac{N}{2})e^{-j\pi k}\right)e^{-j2\pi\frac{nk}{N}}$$

The last summation is looking like a kind of Fourier Transform, however it is not, since $N\neq\frac{N}{2}$.

It was found that there are exist two cases and both afford to the recursive processing:

$$F(2k)=\frac{1}{\sqrt{2}}\frac{1}{\sqrt{N/2}}\sum_{n=0}^{\frac{N}{2}-1}\left(f(n)+f(n+\frac{N}{2})e^{-j2\pi k}\right)e^{-j2\pi\frac{nk}{N/2}}=$$

$$\left(e^{-j2\pi k}=(cos2\pi-jsin2\pi)^k=1\right)$$

$$=\frac{1}{\sqrt{2}}\frac{1}{\sqrt{N/2}}\sum_{n=0}^{\frac{N}{2}-1}\left(f(n)+f(n+\frac{N}{2})\right)e^{-j2\pi\frac{nk}{N/2}}$$

$$F(2k+1)=\frac{1}{\sqrt{2}}\frac{1}{\sqrt{N/2}}\sum_{n=0}^{\frac{N}{2}-1}\left(f(n)+f(n+\frac{N}{2})e^{-j2\pi k}e^{-j\pi}\right)e^{-j2\pi\frac{n(2k+1)}{N}}=$$

$$\left(e^{-j\pi}=\cos\pi-jsin\,\pi=-1\right)$$

$$=\frac{1}{\sqrt{2}}\frac{1}{\sqrt{N/2}}\sum_{n=0}^{\frac{N}{2}-1}\left(f(n)-f(n+\frac{N}{2})\right)e^{-j2\pi\frac{nk}{N/2}}e^{-j2\pi\frac{n}{N}}$$

Here is the DIF FFT algorithm:

$DIF(N, f)$
  if N is equal to 1
    then return f
  for $n = 0$ to $\frac{N}{2} - 1$

$$f_e(n) = (f(n) + f(n + \frac{N}{2})) \frac{1}{\sqrt{2}}$$

$$f_o(n) = (f(n) - f(n + \frac{N}{2})) \frac{1}{\sqrt{2}} e^{-j\pi \frac{n}{N}}$$

$$F_e = DIF(\frac{N}{2}, f_e)$$

$$F_o = DIF(\frac{N}{2}, f_o)$$

$return\ Merge(F_e, F_o)$ // merge odd and even cooffecients

This is the basic algorithm and there are a lot of possibilities to improve its performance. Else, as known, it is always possible to replace recursion by loops.
The inverse DIF is absolutely symmetric to the forward transform.
Recall inverse DFT:

$$f(n) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} F(k) e^{j2\pi \frac{nk}{N}}$$

Or after splitting to the odd/even parts:

$$f(2n) = \frac{1}{\sqrt{2}} \frac{1}{\sqrt{N/2}} \sum_{k=0}^{\frac{N}{2}-1} (F(k) + F(k + \frac{N}{2})) e^{j2\pi \frac{nk}{N/2}}$$

$$f(2n+1) = \frac{1}{\sqrt{2}} \frac{1}{\sqrt{N/2}} \sum_{k=0}^{\frac{N}{2}-1} (F(k) - F(k + \frac{N}{2})) e^{j2\pi \frac{nk}{N/2}} e^{j2\pi \frac{k}{N}}$$

From here we get the basic DIF IFFT:

$IDIF(N, F)$

if N is equal to 1

return F

for $k = 0$ to $\frac{N}{2} - 1$

$$F_e(k) = (F(k) + F(k + \frac{N}{2})) \frac{1}{\sqrt{2}}$$

$$F_o(k) = (F(k) - F(k + \frac{N}{2})) \frac{1}{\sqrt{2}} e^{j\pi \frac{k}{N}}$$

$$f_e = IDIF(\frac{N}{2}, F_e)$$

$$f_o = IDIF(\frac{N}{2}, F_o)$$

$return\ Merge(f_e, f_o)$ // merge odd and even values

Fast Cosine/Sine Transform.

Remind formulas of Discrete Cosine/Sine Transforms:

Cosine Transform:

$$f[m]=\frac{F[0]}{2}+\sum_{n=1}^{N-1}F[n]\cos(2\pi\frac{mn}{N})$$

$$F[m]=\frac{2}{N}\sum_{n=0}^{N-1}f[x]\cos(2\pi\frac{mn}{N})$$

Sine Transform:

$$f[m]=\sum_{n=1}^{N-1}F[n]\sin(2\pi\frac{mn}{N})$$ (remind that $\sin(0)\equiv 0$)

$$F[m]=\frac{2}{N}\sum_{n=0}^{N-1}f[x]\sin(2\pi\frac{mn}{N})$$

Given this formulas we can try to build DIT ("Decimation In Time") algorithm the same way we did for Discrete Fourier Transform.

$$\sum_{n=0}^{N-1}F[n]\cos(2\pi\frac{mn}{N})=$$

$$=\sum_{n=0}^{\frac{N}{2}-1}F[2n]\cos(2\pi\frac{mn}{N/2})+\sum_{n=0}^{\frac{N}{2}-1}F[2n+1]\cos(2\pi\frac{m(2n+1)}{N})=$$

$$(\cos(\alpha+\beta)=\cos(\alpha)\cos(\beta)-\sin(\alpha)\sin(\beta))$$

$$=\sum_{n=0}^{\frac{N}{2}-1}F[2n]\cos(2\pi\frac{mn}{N/2})+\cos(2\pi\frac{m}{N})\sum_{n=0}^{\frac{N}{2}-1}F[2n+1]\cos(2\pi\frac{mn}{N/2})-$$

$$-\sin(2\pi\frac{m}{N})\sum_{n=0}^{\frac{N}{2}-1}F[2n+1]\sin(2\pi\frac{mn}{N/2})$$

For large m the formula maignt be quit similar. Lets replace m by $m+N/2$.

Remind that $\cos(2\pi\frac{(m+N/2)n}{N})=\begin{cases}-\cos(2\pi\frac{mn}{N}) & \text{for odd integer x}\\ +\cos(2\pi\frac{mn}{N}) & \text{for even integer x}\end{cases}$

and $\sin(2\pi\frac{(m+N/2)n}{N}) = \begin{cases} -\sin(2\pi\frac{mn}{N}) & \text{for odd integer x} \\ +\sin(2\pi\frac{mn}{N}) & \text{for even integer x} \end{cases}$.

Therefore

$$\sum_{n=0}^{N-1} F[n]\cos(2\pi\frac{(m+N/2)n}{N}) =$$

$$= \sum_{n=0}^{\frac{N}{2}-1} F[2n]\cos(2\pi\frac{mn}{N/2}) - \cos(2\pi\frac{m}{N})\sum_{n=0}^{\frac{N}{2}-1} F[2n+1]\cos(2\pi\frac{mn}{N/2}) +$$

$$+\sin(2\pi\frac{m}{N})\sum_{n=0}^{\frac{N}{2}-1} F[2n+1]\sin(2\pi\frac{mn}{N/2})$$

Symetrically we can compute sum of $F[n]\sin(2\pi\frac{mn}{N})$:

$$\sum_{n=0}^{N-1} F[n]\sin(2\pi\frac{mn}{N}) =$$

$$= \sum_{n=0}^{\frac{N}{2}-1} F[2n]\sin(2\pi\frac{mn}{N/2}) + \sum_{n=0}^{\frac{N}{2}-1} F[2n+1]\sin(2\pi\frac{m(2n+1)}{N}) =$$

$$(\sin(\alpha+\beta) = \sin(\alpha)\cos(\beta) + \cos(\alpha)\sin(\beta))$$

$$= \sum_{n=0}^{\frac{N}{2}-1} F[2n]\sin(2\pi\frac{mn}{N/2}) + \cos(2\pi\frac{m}{N})\sum_{n=0}^{\frac{N}{2}-1} F[2n+1]\sin(2\pi\frac{mn}{N/2}) +$$

$$+\sin(2\pi\frac{m}{N})\sum_{n=0}^{\frac{N}{2}-1} F[2n+1]\cos(2\pi\frac{mn}{N/2})$$

This equations can be used to compute Fast Cosine/Sine Transform based on DIT approach.

DIT-COS(N, f)

if N is equal 1

then return f

for n = 0 to $\frac{N}{2}-1$

$$f_e[n]=f[2n]$$

$$f_o[n]=f[2n+1]$$

$$F_e=\text{DIT-COS}(\frac{N}{2}, f_e)$$

$$F_{o1}=\text{DIT-COS}(\frac{N}{2}, f_o)$$

$$F_{o2}=\text{DIT-SIN}(\frac{N}{2}, f_o)$$

for k = 0 to $\frac{N}{2}-1$

$$F[k]=F_e[k]+\cos(2\pi\frac{k}{N})F_{o1}[k]-\sin(2\pi\frac{k}{N})F_{o2}[k]$$

$$F[k+N/2]=F_e[k]-\cos(2\pi\frac{k}{N})F_{o1}[k]+\sin(2\pi\frac{k}{N})F_{o2}[k]$$

return F

DIT-SIN(N, f)

if N is equal 1

then return {0}

for n = 0 to $\frac{N}{2}-1$

$$f_e[n]=f[2n]$$

$$f_o[n]=f[2n+1]$$

$$F_e=\text{DIT-SIN}(\frac{N}{2}, f_e)$$

$$F_{o1}=\text{DIT-SIN}(\frac{N}{2}, f_o)$$

$$F_{o2}=\text{DIT-COS}(\frac{N}{2}, f_o)$$

for k = 0 to $\frac{N}{2}-1$

$$F[k]=F_e[k]+\cos(2\pi\frac{k}{N})F_{o1}[k]+\sin(2\pi\frac{k}{N})F_{o2}[k]$$

$$F[k+N/2]=F_e[k]-\cos(2\pi\frac{k}{N})F_{o1}[k]-\sin(2\pi\frac{k}{N})F_{o2}[k]$$

return F

This two functions provide single recursion for fast computation of Fast Cosine/Sine Transform summation. Notice that computed sums are shifted Cosine/Sine Transforms.
To compute real DCT some adjustment is needed:

FCT-DIT-COS(N, f)	forward FCT

F = DIT-COS(N, f)

for n = 0 to N-1

$$F[n]=\frac{2}{N}F[n]$$

return F

IFCT-DIT-COS(N, F) — inverse FCT

f = DIT-COS(N, F)

for n = 0 to N – 1

$$f[n] = f[n] - \frac{F[0]}{2}$$

return f

The Sine Transform (DST) is quit similar:

FST-DIT-SIN(N, f) — forward FST

F = DIT-SIN(N, f)

for n = 0 to N-1

$$F[n] = \frac{2}{N} F[n]$$

return F

Calling IFST-DIT-SIN(N, F) is equivalent to DIT-SIN(N, F).

The resulting algorithm FCT or FST have one advanced loop that is absent in the original Complex FFT – instead of $2N\log_2 N$ summations we have $N + 6N\log_3 N$ summations. The possibilities to optimize FFT or FCT/FST are desired for separate discussion and this is out of scope.

TODO: Compute FCT/FST using DIF ("Decimation In Frequency") approach.
I will update this article when more free time will be available.

References:

1. The FFT Demystified.
   Engineering Productivity Tools Ltd. http://www.eptools.com/tn/T0001/INDEX.HTM
2. Fourier and the Frequency Domain
   The University of Strathclyde http://www.spd.eee.strath.ac.uk/~interact/FFT/fourier.html
3. Eric W. Weisstein. "Discrete Fourier Transform.".
   From *MathWorld*--A Wolfram Web Resource.
   http://mathworld.wolfram.com/DiscreteFourierTransform.html
4. Fast Fourier transform. From Wikipedia, the free encyclopedia.
   http://en.wikipedia.org/wiki/Fast_Fourier_transform