DFT interpolation

Suppose we have measured data at \( n \) evenly spaced points on \([c, d]\).
Interpolation with DFT

Suppose $y = F_n x \Rightarrow x = F_n^{-1} y$.

$$x_j = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} y_k (\omega^{-k})^j = \sum_{k=0}^{n-1} y_k \frac{i2\pi k(t_j-c)}{d-c} \frac{1}{\sqrt{n}}$$

The $y_j$ are the coefficients of the (complex) trigonometric polynomial interpolating $(t_j, x_j)$.

The Fourier transform $F_n$ transforms data $\{x_j\}$ into interpolating coefficients $\{y_j\}$. 
DFT interpolation theorem

Given $x_0, x_1, \ldots, x_{n-1}$, we think of the points $x_j$ as occurring at evenly spaced points on $[c,d]$, $t_j = c + j(d - c)/n$, $j = 0, 1, \ldots, n - 1$. Then

\[
Q(t) = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} F_n x_k e^{i2\pi k(t-c)/(d-c)}
\]

satisfies $Q(t_j) = x_j$ for $j = 0, \ldots, n - 1$.

If the $x_j$ are real and $F_n x_j = a_j + ib_j$, then $P(t_j) = x_j$ for

\[
P(t) = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} \left( a_k \cos \frac{2\pi k(t-c)}{d-c} - b_k \sin \frac{2\pi k(t-c)}{d-c} \right)
\]
Order \( n \) trigonometric function

Applying some trigonometric formulas we can show that for even \( n \), \( t_j = c + j(d - c)/n \) for \( j = 0, \ldots, n - 1 \) and \( F_n x = a + ib \),

\[
P_n(t) = \frac{a_0}{\sqrt{n}} + \frac{2}{\sqrt{n}} \sum_{k=1}^{n/2-1} \left( a_k \cos \frac{2\pi k(t - c)}{d - c} - b_k \sin \frac{2\pi k(t - c)}{d - c} \right) + \frac{a_{n/2}}{\sqrt{n}} \cos \frac{n\pi(t - c)}{d - c}
\]

satisfies \( P_n(t_j) = x_j, \ j = 0, \ldots, n - 1 \).
Expansion of $n$ data points to $p > n$ points

We can rewrite $P_n(t)$ as a $p$ order function:

$$P_p(t) = \frac{\sqrt{\frac{p}{n}} a_0}{\sqrt{p}} + \frac{2}{\sqrt{p}} \sum_{k=1}^{p/2-1} \left( \sqrt{\frac{p}{n}} a_k \cos \frac{2\pi k(t-c)}{d-c} \right)$$

$$- \sqrt{\frac{p}{n}} b_k \sin \frac{2\pi k(t-c)}{d-c} + \sqrt{\frac{p}{n}} a_{n/2} \cos \frac{n\pi(t-c)}{d-c}$$

where $a_k = b_k = 0$ for $k = n/2 + 1, \ldots, p/2$.

To produce $p$ points lying on the curve $P_n(t)$ we must multiply the $F_n x_k$ by $\sqrt{p/n}$ and invert the DFT.
Evaluation of trigonometric functions

To plot the interpolating trigonometric function, we can invert the expanded DFT. The steps are the following:

1. Calculate the DFT of the evenly spaced data points: \( x \rightarrow F_n x \)

2. Multiply by \( \sqrt{p/n} \): \( F_n x \rightarrow \sqrt{p/n} F_n x \)

3. Expand the \( n \) points to \( p \) points: add zeros in positions \( n/2 + 1 \) to \( p - n/2 \)

4. Invert: \( \sqrt{p/n} F_n x \rightarrow F_p^{-1} \sqrt{p/n} F_n x \).
Example

With $n = 8$ and $p = 10$:

$$
\begin{pmatrix}
  y_0 \\
  y_1 \\
  y_2 \\
  y_3 \\
  y_4 \\
  0 \\
  0 \\
  \bar{y}_3 \\
  \bar{y}_2 \\
  \bar{y}_1 \\
\end{pmatrix}
\begin{array}{l}
\{ \begin{array}{l}
  n/2 + 1 \\
  p - n \\
  n/2 - 1 \\
\end{array} \} \\
\end{array}
$$
In MATLAB

We use the commands `fft` and `ifft` to compute

\[
F_n = \frac{1}{\sqrt{n}} \cdot \text{fft} \quad \text{and} \quad F_p^{-1} = \sqrt{p} \cdot \text{ifft}
\]

so the evaluation corresponds to:

\[
F_p^{-1} \sqrt{\frac{p}{n}} F_n x = \sqrt{p} \cdot \text{ifft} \sqrt{\frac{p}{n}} \frac{1}{\sqrt{n}} \cdot \text{fft} = \frac{p}{n} \cdot \text{ifft}_{[p]} \cdot \text{fft}_{[n]}
\]
Example of FFT in MATLAB

```matlab
>> x=[-2.2 -2.8 -6.1 -3.9 0 1.1 -0.6 -1.1];
>> y=fft(x)'/sqrt(8)
```

```
y =
   -5.5154
   -1.0528 - 3.6195i
    1.5910 + 1.1667i
   -0.5028 + 0.2695i
   -0.7778
   -0.5028 - 0.2695i
    1.5910 - 1.1667i
   -1.0528 + 3.6195i
```
\[ y = a + bi = F_n x \]

\(a_k\) and \(b_k\) are the coefficients of the interpolating trigonometric polynomial.

Applying formula

\[
P(t) = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} \left( a_k \cos \frac{2\pi k(t-c)}{d-c} - b_k \sin \frac{2\pi k(t-c)}{d-c} \right)
\]

with \(n = 8, c = 0, d = 1\), we get

\[
F(t) = -1.95 - 0.7445 \cos 2\pi t - 2.5594 \sin 2\pi t + 1.125 \cos 4\pi t \\
+ 0.825 \sin 4\pi t - 0.3555 \cos 6\pi t + 0.1906 \sin 6\pi t - 0.2750 \cos 8\pi t
\]
Fourier interpolation in MATLAB

%Interpolate n data points on [c,d] with trig function P(t)
% and plot interpolant at p (>= n) evenly spaced points.
%Input: interval [c,d], data points x,
% even number of data points n, even number p>=n
%Output: data points of interpolant xp
function xp=dftinterp(inter,x,n,p)
c=inter(1);d=inter(2);
t=c+(d-c)*(0:n-1)/n; % n evenly-spaced time points
tp=c+(d-c)*(0:p-1)/p; % p evenly-spaced time points
y=fft(x); % apply DFT
yp=zeros(p,1); % yp will hold coefficients for ifft
yp(1:n/2+1)=y(1:n/2+1); % move n frequencies from n to p
yp(p-n/2+2:p)=y(n/2+2:n); % same for upper tier
xp=real(ifft(yp))*(p/n); % invert fft to recover data
plot(t,x,’o’,tp,xp) % plot data points and interpolant