Outline

• Recap from Wednesday
• Fourier Analysis
  – Scalings and Symmetries
  – Sampling Theorem
• Discrete Fourier Transforms
  – Naïve approach
  – Danielson-Lanczos lemma
  – Cooley-Tukey algorithm
• Examples
Recap from Wednesday

• Schrodinger’s equation
• Infinite & Finite Quantum Wells
• Kronig-Penney model
• Numerical solutions:
  – Real space
  – Fourier space
Fourier Analysis

- Fourier transformation is a linear operation that maps time-series data into the Fourier-domain:

\[ \tilde{f}(\omega) = \int_{-\infty}^{\infty} dt \ f(t)e^{i\omega t} \]

- Independent variable interpreted as frequency

- Inverse Fourier transform:

\[ f(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} d\omega \ \tilde{f}(\omega)e^{-i\omega t} \]
## Fourier Symmetries

<table>
<thead>
<tr>
<th>Time-domain property of $f(t)$</th>
<th>Frequency-domain property of $\tilde{f}(\omega)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(t) = \text{Re}{f(t)}$</td>
<td>$\tilde{f}(-\omega) = \tilde{f}^*(\omega)$</td>
</tr>
<tr>
<td>$f(t) = \text{Im}{f(t)}$</td>
<td>$\tilde{f}(-\omega) = -\tilde{f}^*(\omega)$</td>
</tr>
<tr>
<td>$f(t) = f(-t)$</td>
<td>$\tilde{f}(-\omega) = \tilde{f}(\omega)$</td>
</tr>
<tr>
<td>$f(t) = -f(-t)$</td>
<td>$-\tilde{f}(-\omega) = \tilde{f}(\omega)$</td>
</tr>
<tr>
<td>$f(t) = f(-t) = \text{Re}{f(t)}$</td>
<td>$\tilde{f}(-\omega) = \tilde{f}(\omega) = \text{Re}{\tilde{f}(\omega)}$</td>
</tr>
<tr>
<td>$f(t) = -f(-t) = \text{Re}{f(t)}$</td>
<td>$-\tilde{f}(-\omega) = \tilde{f}(\omega) = \text{Im}{\tilde{f}(\omega)}$</td>
</tr>
<tr>
<td>$f(t) = f(-t) = \text{Im}{f(t)}$</td>
<td>$\tilde{f}(-\omega) = \tilde{f}(\omega) = \text{Im}{\tilde{f}(\omega)}$</td>
</tr>
<tr>
<td>$f(t) = -f(-t) = \text{Im}{f(t)}$</td>
<td>$-\tilde{f}(-\omega) = \tilde{f}(\omega) = \text{Re}{\tilde{f}(\omega)}$</td>
</tr>
</tbody>
</table>
## Fourier Scalings

<table>
<thead>
<tr>
<th>Time Domain Expression</th>
<th>Frequency-Domain Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(t)$</td>
<td>$\tilde{f}(\omega)$</td>
</tr>
<tr>
<td>$\alpha f(t) + \beta g(t)$</td>
<td>$\alpha \tilde{f}(\omega) + \beta \tilde{g}(\omega)$</td>
</tr>
<tr>
<td>$f(at)$</td>
<td>$\frac{1}{</td>
</tr>
<tr>
<td>$f(t - t_0)$</td>
<td>$\tilde{f}(\omega) e^{i\omega t_0}$</td>
</tr>
<tr>
<td>$f(t)e^{-i\omega t_0}$</td>
<td>$\tilde{f}(\omega - \omega_0)$</td>
</tr>
<tr>
<td>$t^n f(t)$</td>
<td>$i^n \frac{d^n \tilde{f}(\omega)}{d\omega^n}$</td>
</tr>
<tr>
<td>$\frac{d^n f(t)}{dt^n}$</td>
<td>$(i\omega)^n \tilde{f}(\omega)$</td>
</tr>
</tbody>
</table>
Sampling Theorem

• At finite sampling rates, need at least two data points

• For a sample time interval $\Delta$, max measurable frequency is the Nyquist frequency $f_c = 1/2\Delta$

• Sampling theorem says that $f_c$ bandwidth-limited spectrum completely determined by data sampled at intervals of $\Delta$
Discrete Fourier Transforms

• Accounts for finite time spacing of data points
• Gives rise to finite frequency spacing and maximum frequency (from sampling theorem)
• DFT defined by:
  \[ F(n) = \sum_{k=1}^{N} f(x_k) e^{-2\pi j (x_k n / x_N)} \]
• For uniform time spacing \( \Delta \):
  \[ F(n) = \sum_{k=1}^{N} f_k e^{-2\pi j k n / N} \]
Discrete Fourier Transforms: Naïve Algorithm

• Rewrite DFT as:

\[ F(n) = \sum_{k=1}^{N} f_k W^{nk} \]

• Where \( W = e^{-2\pi j/N} \)

• Just perform sum (\( N \) operations) for each frequency (\( N \) times)

• Overall time scales as \( N^2 \)
Discrete Fourier Transforms: Naïve Algorithm

- Obtain same number of points in DFT as original series
- Symmetry properties same as for continuous FT
- For inverse Fourier transform – just use the inverse of $W$:

$$f_k = \frac{1}{N} \sum_{k=1}^{N} F(n) \left( \frac{1}{W} \right)^{nk}$$
Fast Fourier Transforms: Danielson-Lanczos Lemma

• Based on Danielson-Lanczos lemma:

\[
F(n) = \sum_{k=1}^{N} f_k W^{nk}
\]

\[
= \sum_{k=1}^{N/2} f_{2k} W^{n2k} + \sum_{k=1}^{N/2} f_{2k+1} W^{n(2k+1)}
\]

\[
= F^e(n) + W^n F^o(n)
\]

• Can be applied recursively
Discrete Fourier Transforms: Cooley-Tukey Algorithm

- If one applies D-L lemma recursively to a data set with $N = 2^m$, reduce to FT of single point!
- Key is to order everything to keep track of where everything should go – then work backwards

2/1/2013 ECE 595, Prof. Bermel
Discrete Fourier Transforms: Cooley-Tukey Algorithm

• Algorithm devised by Cooley and Tukey:
  – Sort data into bit reversed order
  – Perform FTs on lengths 1, 2, 4, 8, etc. with D-L lemma

• Operations for first step go as $N$

• Operations for second step go as $N$ per cycle, with $\log_2 N$ cycles

• Overall time is $N \log_2 N$—considerably better than naïve approach
Fast Fourier Transform Data

- Input data is uniformly spaced
- Output consists of rising positive frequencies, followed by negative frequencies decreasing in magnitude
Cooley-Tukey Algorithm
Next Class

- Is on Wednesday, Feb. 6
- Will discuss numerical tools for Fast Fourier Transforms
- Recommended reading: Numerical Recipes, Chapters 12-13
- Please email me your HW #2 today (by 4:30 pm)