Advanced Digital Signal Processing

Prof. Dr. Nizamettin AYDIN

naydin@yildiz.edu.tr
naydin@ieee.org
http://www.yildiz.edu.tr/~naydin

Digital Signal Processing (DSP)

Basics:
What is DSP?

CONVERGING FIELDS

EE
CmpE
Math
Physics
Computer Science
Applications

BIO

Digital Signal Processing (DSP)
Dictionary definitions of the words in DSP:
• Digital
  – operating by the use of discrete signals to represent data in the form of numbers
• Signal
  – a variable parameter by which information is conveyed through an electronic circuit
• Processing
  – to perform operations on data according to programmed instructions
• So a simple definition of DSP could be:
  – changing or analyzing information which is measured as discrete sequences of numbers
• Unique features of DSP as opposed to ordinary digital processing:
  – signals come from the real world
  – this intimate connection with the real world leads to many unique needs such as the need to react in real time and a need to measure signals and convert them to digital numbers
  – signals are discrete
  – which means the information in between discrete samples is lost

WHY USE DSP ?

• Versatility:
  – digital systems can be reprogrammed for other applications
  – digital systems can be ported to different hardware
• Repeatability:
  – digital systems can be easily duplicated
  – digital systems do not depend on strict component tolerances
  – digital system responses do not drift with temperature
• Simplicity:
  – some things can be done more easily digitally than with analogue systems

DSP is used in a very wide variety of applications

• Radar, sonar, telephony, audio, multimedia, communications, ultrasound, process control, digital camera, digital tv, Telecommunications, Sound & Music, Fourier Optics, X-ray Crystallography, Protein Structure & DNA, Computerized Tomography, Nuclear Magnetic Resonance: MRI, Radioastronomy
• All these applications share some common features:
  – they use a lot of maths (multiplying and adding signals)
  – they deal with signals that come from the real world
  – they require a response in a certain time
• Where general purpose DSP processors are concerned, most applications deal with signal frequencies that are in the audio range
Fundamental concepts in DSP

- DSP applications deal with analogue signals
  - the analogue signal has to be converted to digital form

A typical biomedical measurement system

Transducers

- A “transducer” is a device that converts energy from one form to another.
- In signal processing applications, the purpose of energy conversion is to transfer information, not to transform energy.
- In physiological measurement systems, transducers may be
  - input transducers (or sensors)
    - they convert a non-electrical energy into an electrical signal
    - for example, a microphone.
  - output transducers (or actuators)
    - they convert an electrical signal into a non-electrical energy.
    - For example, a speaker.

Signal Encoding: Analog-to Digital Conversion

- The analogue signal
  - a continuous variable defined with infinite precision
  - is converted to a discrete sequence of measured values which are represented digitally
- Information is lost in converting from analogue to digital, due to:
  - inaccuracies in the measurement
  - uncertainty in timing
  - limits on the duration of the measurement
- These effects are called quantisation errors
Analog-to Digital Conversion

- ADC consists of four steps to digitize an analog signal:
  1. Filtering
  2. Sampling
  3. Quantization
  4. Binary encoding

- Before we sample, we have to filter the signal to limit the maximum frequency of the signal as it affects the sampling rate.
- Filtering should ensure that we do not distort the signal, i.e., remove high frequency components that affect the signal shape.

Sampling

- The sampling results in a discrete set of digital numbers that represent measurements of the signal
  - usually taken at equal intervals of time
- Sampling takes place after the hold
  - The hold circuit must be fast enough that the signal is not changing during the time the circuit is acquiring the signal value
- We don't know what we don't measure
- In the process of measuring the signal, some information is lost

Sampling

- Analog signal is sampled every \( T_s \) secs.
- \( T_s \) is referred to as the sampling interval.
- \( f_s = 1/T_s \) is called the sampling rate or sampling frequency.
- There are 3 sampling methods:
  - Ideal - an impulse at each sampling instant
  - Natural - a pulse of short width with varying amplitude
  - Flattop - sample and hold, like natural but with single amplitude value
- The process is referred to as pulse amplitude modulation PAM and the outcome is a signal with analog (non-integer) values
According to the Nyquist theorem, the sampling rate must be at least 2 times the highest frequency contained in the signal.

\[ F_s \geq 2f_m \]
Quantization

- Sampling results in a series of pulses of varying amplitude values ranging between two limits: a minimum and a maximum.
- The amplitude values are infinite between the two limits.
- We need to map the infinite amplitude values onto a finite set of known values.
- This is achieved by dividing the distance between minimum and maximum into $L$ zones, each of height $\Delta$.

$$\Delta = \frac{\text{max} - \text{min}}{L}$$

Quantization Levels

- The midpoint of each zone is assigned a value from 0 to $L-1$ (resulting in $L$ values).
- Each sample falling in a zone is then approximated to the value of the midpoint.

Quantization Zones

- Assume we have a voltage signal with amplitudes $V_{\text{min}}=-20\text{V}$ and $V_{\text{max}}=+20\text{V}$.
- We want to use $L=8$ quantization levels.
- Zone width $\Delta = (20 - (-20))/8 = 5$
- The 8 zones are: -20 to -15, -15 to -10, -10 to -5, -5 to 0, 0 to +5, +5 to +10, +10 to +15, +15 to +20
- The midpoints are: -17.5, -12.5, -7.5, -2.5, 2.5, 7.5, 12.5, 17.5

Assigning Codes to Zones

- Each zone is then assigned a binary code.
- The number of bits required to encode the zones, or the number of bits per sample as it is commonly referred to, is obtained as follows:
  $$n_b = \log_2 L$$
- Given our example, $n_b = 3$
- The 8 zone (or level) codes are therefore: 000, 001, 010, 011, 100, 101, 110, and 111
- Assigning codes to zones:
  - 000 will refer to zone -20 to -15
  - 001 to zone -15 to -10, etc.

Quantization and encoding of a sampled signal

- When a signal is quantized, we introduce an error - the coded signal is an approximation of the actual amplitude value.
- The difference between actual and coded value (midpoint) is referred to as the quantization error.
- The more zones, the smaller $\Delta$ which results in smaller errors.
- BUT, the more zones the more bits required to encode the samples -> higher bit rate
Analog-to-digital Conversion

Example An 12-bit analog-to-digital converter (ADC) advertises an accuracy of ± the least significant bit (LSB). If the input range of the ADC is 0 to 10 volts, what is the accuracy of the ADC in analog volts?

Solution: If the input range is 10 volts then the analog voltage represented by the LSB would be:

\[ V_{\text{LSB}} = \frac{V_{\text{max}}}{2^{12}} = \frac{10}{4096} = 0.0024 \text{ volts} \]

Hence the accuracy would be ± 0.0024 volts.

Sampling related concepts

- Over/exact/under sampling
- Regular/irregular sampling
- Linear/Logarithmic sampling
- Aliasing
- Anti-aliasing filter
- Image
- Anti-image filter

Steps for digitization/reconstruction of a signal

- Band limiting (LPF)
- Sampling / Holding
- Quantization
- Coding

These are basic steps for A/D conversion

Digital data: end product of A/D conversion and related concepts

- Bit: least digital information, binary 1 or 0
- Nibble: 4 bits
- Byte: 8 bits, 2 nibbles
- Word: 16 bits, 2 bytes, 4 nibbles

Some jargon:
- integer, signed integer, long integer, 2s complement, hexadecimal, octal, floating point, etc.

Data types

- Our first requirement is to find a way to represent information (data) in a form that is mutually comprehensible by human and machine.
  - Ultimately, we will have to develop schemes for representing all conceivable types of information – language, images, actions, etc.
  - We will start by examining different ways of representing integers, and look for a form that suits the computer.
  - Specifically, the devices that make up a computer are switches that can be on or off, i.e. at high or low voltage. Thus they naturally provide us with two symbols to work with: we can call them on & off, or (more usefully) 0 and 1.
Signal

- An information variable represented by physical quantity.
- For digital systems, the variable takes on discrete values.
- Two level, or binary values are the most prevalent values in digital systems.
- Binary values are represented abstractly by:
  - digits 0 and 1
  - words (symbols) False (F) and True (T)
  - and words On and Off.
- Binary values are represented by values or ranges of values of physical quantities.

Number Systems – Representation

- Positive radix, positional number systems
- A number with radish \( r \) is represented by a string of digits:
  \[ A_n \cdot r^A_{n-2} \cdots A_1 r \cdot A_0 \cdot A_{-1} r^{-1} A_{-2} \cdots A_{-m} r^{-m} \]
  in which \( 0 \leq A_i < r \) and \( \cdot \) is the radix point.
- The string of digits represents the power series:
  \[ (\text{Number})_r = \left( \sum_{i=0}^{n-1} A_i \cdot r^i \right) + \left( \sum_{j=0}^{m-1} A_j \cdot r^{-j} \right) \]
  (Integer Portion) + (Fraction Portion)

Decimal Numbers

- “decimal” means that we have ten digits to use in our representation (the symbols 0 through 9)
- What is 3546?
  - it is three thousands plus five hundreds plus four tens plus six ones
  - i.e. 3546 = 3 \cdot 10^3 + 5 \cdot 10^2 + 4 \cdot 10^1 + 6 \cdot 10^0
- How about negative numbers?
  - we use two more symbols to distinguish positive and negative:
    + and -

Unsigned Binary Integers

\[ Y = \text{"abc"} = a \cdot 2^2 + b \cdot 2^1 + c \cdot 2^0 \]

(where the digits \( a, b, c \) can each take on the values of 0 or 1 only)

<table>
<thead>
<tr>
<th>N = number of bits</th>
<th>3-bits</th>
<th>5-bits</th>
<th>8-bits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 0 0 0</td>
<td>0 0 0 0 0</td>
<td>0 0 0 0 0 0</td>
</tr>
<tr>
<td>Range is: ( 0 \leq \frac{N}{2} + 1 )</td>
<td>1 0 0 1</td>
<td>0 0 0 0 0 1</td>
<td>0 0 0 0 0 0 0 1</td>
</tr>
</tbody>
</table>

Problem:
- How do we represent negative numbers?

4 0 1 0 0 0 1 0 0 0 0 0 0 1 0 0

Two’s Complement

- Transformation
  - To transform a into -a, invert all bits in a and add 1 to the result

<table>
<thead>
<tr>
<th>-16</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>-3</td>
<td>11101</td>
</tr>
<tr>
<td>-2</td>
<td>11110</td>
</tr>
<tr>
<td>-1</td>
<td>11111</td>
</tr>
<tr>
<td>0</td>
<td>00000</td>
</tr>
<tr>
<td>+1</td>
<td>00001</td>
</tr>
<tr>
<td>+2</td>
<td>00010</td>
</tr>
<tr>
<td>+3</td>
<td>00011</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>+15</td>
<td>01111</td>
</tr>
</tbody>
</table>

Advantages:
- Operations need not check the sign
- Only one representation for zero
- Efficient use of all the bits

Limitations of integer representations

- Most numbers are not integer!
  - Even with integers, there are two other considerations:
  - Range:
    - The magnitude of the numbers we can represent is determined by how many bits we use:
      \( \text{e.g. with 32 bits the largest number we can represent is about } \pm 2 \text{ billion, far too small for many purposes.} \)
  - Precision:
    - The exactness with which we can specify a number:
      \( \text{e.g. a 32 bit number gives us 31 bits of precision, or roughly 9 } \) 
      - We need another data type!
### Real numbers

- Our decimal system handles non-integer real numbers by adding yet another symbol - the decimal point (.) to make a fixed point notation:
  - e.g. \(3.45678 = 3 \times 10^1 + 4 \times 10^{-1} + 5 \times 10^{-2} + 6 \times 10^{-3} + 7 \times 10^{-4} + 8 \times 10^{-5}\)

- The floating point, or scientific, notation allows us to represent very large and very small numbers (integer or real), with as much or as little precision as needed:
  - Unit of electric charge \(e = 1.602176462 \times 10^{-19}\) Coulomb
  - Volume of universe = \(1 \times 10^{85}\) cm\(^3\)

### Real numbers in binary

- We mimic the decimal floating point notation to create a “hybrid” binary floating point number:
  - We first use a “binary point” to separate whole numbers from fractional numbers to make a fixed point notation:
    - e.g. \(00011001.110 = 1.1001110 \times 2^4\)
    - In order to handle both +ve and -ve exponents, we add 127 to the actual exponent to create a “biased exponent”:
      - \(2^0\) => biased exponent = 0000 0000 (bias = 0)
      - \(2^7\) => biased exponent = 0111 1111 (bias = 127)
      - \(2^{127}\) => biased exponent = 1111 1110 (bias = 254)

### IEEE-754 fp numbers

- **32 bits**: 8 bits biased exp., 23 bits fraction
- \(N = (-1)^s \times 1.fraction \times 2^{biased \ exp. – 127}\)
- **Sign**: 1 bit
- **Mantissa**: 23 bits
  - We “normalize” the mantissa by dropping the leading 1 and recording only its fractional part (why?)
- **Exponent**: 8 bits
  - In order to handle both +ve and -ve exponents, we add 127 to the actual exponent to create a “biased exponent”:
    - \(2^0\) => biased exponent = 0000 0000 (bias = 0)
    - \(2^7\) => biased exponent = 0111 1111 (bias = 127)
    - \(2^{127}\) => biased exponent = 1111 1110 (bias = 254)

### Binary Numbers and Binary Coding

- **Flexibility of representation**
  - Within constraints below, can assign any binary combination (called a code word) to any data as long as data is uniquely encoded.
- **Information Types**
  - **Numeric**
    - Must represent range of data needed
    - Very desirable to represent data such that simple, straightforward computation for common arithmetic operations permitted
    - Tight relation to binary numbers
  - **Non-numeric**
    - Greater flexibility since arithmetic operations not applied.
    - Not tied to binary numbers

### Non-numeric Binary Codes

- **Given \(n\) binary digits (called bits), a binary code is a mapping from a set of represented elements to a subset of the \(2^n\) binary numbers.**
- **Example**: A binary code for the seven colors of the rainbow
  - **Code 100 is not used**

<table>
<thead>
<tr>
<th>Color</th>
<th>Binary Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red</td>
<td>000</td>
</tr>
<tr>
<td>Orange</td>
<td>001</td>
</tr>
<tr>
<td>Yellow</td>
<td>010</td>
</tr>
<tr>
<td>Green</td>
<td>011</td>
</tr>
<tr>
<td>Blue</td>
<td>101</td>
</tr>
<tr>
<td>Indigo</td>
<td>110</td>
</tr>
<tr>
<td>Violet</td>
<td>111</td>
</tr>
</tbody>
</table>
Given M elements to be represented by a binary code, the minimum number of bits, $n$, needed, satisfies the following relationships:

$$2^n > M > 2^{(n - 1)}$$

$$n = \lceil \log_2 M \rceil$$

where $\lceil x \rceil$, called the ceiling function, is the integer greater than or equal to $x$.

Example: How many bits are required to represent decimal digits with a binary code?

- 4 bits are required ($n = \lceil \log_2 9 \rceil = 4$)