# Notes: Miscellanea

Michael Van Wickle

Spring 2016

# Contents

# 1 AC Power

- Real/Active Power, $P$ (Watts, W)
  - power delivered to a purely resistive load
- Reactive Power, $Q$ (reactive volt-amperes, var)
  - exists in an AC circuit when voltage and current are not in phase
  - indicative of a capacitance and/or inductance in the circuit
  - $Q = V_{rms} I_{rms} sin(\phi)$
- Complex Power, $S$ (volt-ampere, VA)
  - $S = P + jQ$
- Apparent Power, $|S|$ (volt-ampere, VA)
  - the magnitude of the complex power
- Phase of voltage relative to current, $\phi$
  - if $\phi$ is in Quadrant I: current lagging voltage
  - if $\phi$ is in Quadrant IV: current leading voltage
- The unit for all kinds of power is the Watt, W
  - this is generally reserved to refer to active power
- On the imaginary plane:
  - Active power is on the real axis
  - Reactive power is on the imaginary axis, as it does no work
  - Complex power is the vector sum of the active and reactive power
  - Apparent power is the magnitude of complex power
  - The angle formed by the active and complex power vectors is $\phi$
- Capacitive Loads
  - "source" reactive power
  - cause current to lead voltage
- Inductive Loads
  - "sink" reactive power
  - cause current to lag behind voltage
- Power Factor, $PF$
  - the ratio of real power to apparent power
  - $PF = cos\phi = \dfrac{P}{|S|}$

## 1.1 Three Phase Power

- Three-Phase Power
  - three conductors carry AC power of the same frequency and voltage amplitude
    - each is phase shifted one third of a period (120 degrees)

## 2 Algorithms

- In 2000, IEEE published a list of the 10 most important, influential algorithms of the $20^{th}$ century
- They are, in no particular order:
  - Metropolis Algorithm for Monte Carlo
  - Simplex Method for Linear Programming
  - Krylov Subspace Iteration Methods
  - The Decompositional Approach to Matrix Computations
  - The Fortran Optimizing Compiler
  - QR Algorithm for Computing Eigenvalues
  - Quicksort Algorithm for Sorting
  - Fast Fourier Transform
  - Integer Relation Detection
  - Fast Multipole Method

### 2.1 Graham Scan

The Graham Scan takes a finite set of points as an input and finds a convex hull from the set. It has a time complexity of $O(nlog(n))$.

Given a set containing $n$ points

1. find the point with the lowest y-coordinate, call this point $P$
   - if more than one point share this y-coordinate, choose the point with the lowest x-coordinate
2. order the set in increasing order of the angle they and $P$ make with the x-axis
3. consider each point in the sorted sequence
   - for each point it is determined whether traveling from the two points immediately preceding this point is a left or right turn
     * if it is a right turn, the second-to-last point is not part of the convex hull, it lies inside the hull
     * the same determination is made for the set of the latest point and the two points preceding the point found inside the hu
     * this is repeated until a left turn is found, after which the algorithm moves on to the next set of points in the array, skipp

### 2.2 MD5

A cryptographic hashing function, defined in `RFC 1321`, it produces a 128-bit hash value, or digest. This it typically expressed as a 32 digit hexadecimal number. Commonly used to verify data integrity.

#### Definitions

- Word: 32-bits
- Byte: 8-bits
- Big Endian: Most significant byte at lowest address, least significant byte at highest address
- Little Endian: Most significant byte at highest address, least significant byte at lowest address

#### Algorithm

- Input message is an arbitrary length sequence of bits
- This is broken up into 512-bit blocks, i.e., 16 32-bit words
- The input is padded to make it divisible by 32
  1. A single 1 is appended
  2. Zeroes are added until the length is 64 bits less than a multiple of 512
  3. The remaining 64 bits are the binary representation of the original length of the input
- The main algorithm operates on a 128-bit state, divided into 4 32-bit words: $A, B, C, D$
- The words are initialized to constants
  - Values are: Little Endian, Hex
  - $A: 67\ \ 45\ \ 23\ \ 01$
  - $B: EF\ \ CD\ \ AB\ \ 89$
  - $C: 98\ \ BA\ \ DC\ \ FE$
  - $D: 10\ \ 32\ \ 54\ \ 76$
- They are then operated on by each 512 bit block by four different functions
  - $F(B, C, D) = (B \cdot C) + (B' \cdot D)$
  - $G(B, C, D) = (B \cdot D) + (C \cdot D')$
  - $H(B, C, D) = B \oplus C \oplus D$
  - $I(B, C, D) = C \oplus (B + D')$
- For the actual algorithm, a table, `T` is constructed, consisting of 64 elements
  - for `i` from `1` to `64`:

    `T[i]=int(`$2^{32}$`·abs(sin(`$i+1$`))`

    * where `i` is in radians and `int()` takes the integer portion of a number
- Each *chunk*(512-bits) of the input is broken into 16 32-bit words: $M[j], 0 \leq j < 16$

## 2.3 RSA

A public-key encryption system, used for secure data transmission.

- Four steps to the algorithm:
  1. Key Generation
  2. Key Distribution
  3. Encryption
  4. Decryption

### 2.3.1 Key Generation

1. Choose two distinct primes, $p$ and $q$
   - $p$ and $q$ should be chosen at random and should be similar in magnitude, but differ in length by a few digits
2. Compute: $n = p \cdot q$

   - $n$ is used as the *modulus* for both the public and private keys
   - its length, in bits, is the key length
3. Compute $\phi(n) = \phi(p)\phi(q) = (p-1)(q-1) = n - (p+q-1)$
   - $\phi(n)$ is Euler's Totient Function
     * counts the number of positive integers coprime with $n$
   - this value is kept private
4. Choose an integer $e$, such that: $1 < e < \phi(n)$ and $gcd(e, \phi(n)) = 1$
   - i.e., $e$ and $\phi(n)$ are coprime
   - $e$ is released as the *public key exponent*
5. Determine $d$ such that: $d \equiv e^{-1}\ (mod\ \phi(n))$
   - $d$ is the modular multiplicative inverse of $e$ (*modulo* $\phi(n)$
   - i.e., solve for $d$, given $d \cdot e \equiv 1\ (mod\ \phi(n))$
   - $d$ is kept as the private key exponent

- The *public key* consists of the modulus $n$, and the public (or encryption) exponent $e$
- The *private key* consists of the modulus $n$, and the private (or decryption) exponent $d$
- $d$, $p$, $q$, and $\phi(n)$ must all be kept secret
  - $p$, $q$, and $\phi(n)$ can be used to calculate $d$

### 2.3.2 Key Distribution

The public key, $(n, e)$ is then transmitted to parties wishing to send encrypted messages. The private key, $d$, is kept secret and never revealed.

### 2.3.3 Encryption

- A message, $M$, is to be sent.
- $M$ is turned into an integer, $m$, such that: $0 \leq m < n$ and $gcd(m, n) = 1$
  - $m$ is computed using a padding scheme
    * the padding scheme is a mutually agreed upon, reversible protocol
    * e.g., Optimal Asymmetric Encryption Padding (RFC 2437)
- The cyphertext, $c$ is computed, using the public key, $e$
  - $c \equiv m^e\ (mod\ n)$
- $c$ is then transmitted

### 2.3.4 Decryption

- $m$ can be recovered from $c$ using the private key $d$
  - $c^d \equiv (m^e)^d \equiv m\ mod\ n$
- with $m$, $M$ can be recovered by reversing the padding scheme

### 2.3.5 Diffie-Hellman Key Exchange

- A secure way of sending encryption keys across parties

## 2.4 Advanced Encryption Standard (AES), Rijndael

A Federal Information Processing Standard (FIPS) approved cryptographic algorithm, used to transmit secure, government data. A symmetric block cipher that can encrypt (encipher) and decrypt (decipher) information. Processes data blocks of 128-bits.

#### Parameters, Symbols, and Functions

- `AddRoundKey()`
  - a transformation in the cipher and inverse cipher in which a Round Key is added to the state using an XOR operation
- `InvMixColumns()`
  - the inverse of the `MixColumns()` transformation
- `InvShiftRows()`
  - the inverse of the `ShiftRows()` transformation
- `InvSubBytes()`
  - the inverse of the `SubBytes()` transformation
- `K`
  - the cipher key
- `MixColumns()`
  - a transformation in the cipher that takes all of the columns of the state and mixes their date (independently) to produce new columns
- `Nb`
  - number of columns (32-bit words) in the state, `Nb` is 4 for this standard
- `Nk`
  - number of 32-bit words comprising the cipher key, for this standard, `Nk` is 4, 6, or 8
- `Nr`
  - number of rounds, a function of `Nb` and `Nk`, which is fixed, for this standard, `Nr` is 10, 12, or 14
- `Rcon[]`
  - the round constant word array
- `RotWord()`
  - function used in Key Expansion that takes a four byte word and performs a cyclic permutation
- `ShiftRows()`
  - transformation in the cipher which processes the state by cyclically shifting the last tree rows of the state by different offsets
- `SubBytes()`
  - transformation in the cipher which processes the state using a non-linear substitution table (S-box) that operates

on each of the state bytes independently

- `SubWord()`
  - function used in Key Expansion which takes a four byte input word and applies an S-box to each of the four bytes to produce an output word
- `XOR`
  - the exclusive-OR operation
- $\oplus$
  - the exclusive-OR operation
- $\otimes$
  - multiplication of two polynomials (each with degree <4) modulo $x^4 + 1$
- $\bullet$
  - finite-field multiplication

## Notation and Convention

- Input and Output
  - the input and output of the algorithm are sequences of 128-bits, also referred to as blocks with the number of bits they contain referred to as their length
- Cipher Key
  - the cipher key for the algorithm is a sequence of 128, 192, or 256 bits
- other input, output, or cipher key lengths are not used in this standard
- the bits within these sequences are numbered starting at zero and ending at one less than the sequence length
- the number $i$ attached to a bit is called its index

## Bytes

- the basic unit of processing in the algorithm
- the input, output, and cipher key bit sequences are proccesed as arrays of bytes that are formed by dividing the sequences into groups of eight bits
- for an input, output, or cipher key, $a$, the bytes will be referred to as $a_n$ or $a[n]$, where $n$ will be:
    * for a key length of 128 bits: $0 \leq n < 16$
    * for a key length of 192 bits: $0 \leq n < 24$
    * for a key length of 256 bits: $0 \leq n < 32$
    * in a block (128-bits): $0 \leq n < 16$
- in the AES algorithm, byte values will be represented as the concatenation of its individual bit values:

i.e., $\{b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0\}$

  - these bytes are interpreted as finite field elements using a polynomial representation:

$$\text{i.e., } b_7a^7 + b_6a^6 + b_5a^5 + b_4a^4 + b_3a^3 + b_2a^2 + b_1a^1 + b_0 = \sum_{i=0}^{7} b_i x^i$$

  - for example, $\{01100011\}$ identifies the finite-field element: $x^6 + x^5 + x + 1$

## Arrays of Bytes

- arrays of bytes are represented in the following form:
    - $a_0 a_1 a_2 \dots a_{15}$
- the bytes and bit ordering within bytes are derived from the 128-bit input sequence
    - $input_0\ input_1\ input_2\ \dots\ input_{126}\ input_{127}$
- like so,
    - $a_0 = \{input_0,\ input_1,\ \dots\ , input_7\}$
    - $a_1 = \{input_8,\ input_9,\ \dots\ , input_{15}\}$

      $\vdots$
    - – $a_{15} = \{input_{120},\ input_{121},\ \dots\ , input_{127}\}$
- this can be generalized for longer sequences as:
    - $a_n = \{input_{8n},\ input_{8n+1},\ \dots\ , input_{8n+7}\}$

## The State

- internally, the algorithm's operations are performed on a two dimensional array of bytes called the State
- this consists of four rows of bytes, each containing $Nb$ bytes
    - $Nb$ being the block length divided by 32
- in the State array, $s$, each individual byte has two indicies
    - its row number, $r$, in the range $0 \le r < 4$
    - its column number, $c$, in the range $0 \le c < Nb$
- at the start of the cipher and inverse cipher, the input, an array of bytes $(in_0,\ in_1,\ \dots\ ,\ in_{15})$ is copied into the State array
    - this copy takes place by converting the index of the input into a pair, $r, c$
        - this conversion takes the form: $s[r, c] = in[r + 4c]$, where $0 \le r < 4$ and $0 \le c < Nb$
    - at the end of the cipher and inverse cipher, the State is copied into the output array, $out$
        - this conversion takes the form: $out[r + 4c] = s[r, c]$, where $0 \le r < 4$ and $0 \le c < Nb$
- the four bytes in the State form 32-bit words, with the row number, $r$, providing an index for the four bytes within
each word
- the State can therefore be seen as a one-dimensional array of 32-bit words (columns), $w_0 \dots w_3$, where each column number, $c$
provides an index into the array

## Specification

- the length of the input block, the output block, and the State is 128 bits
    - this is represented by $Nb = 4$, which is the number of 32-bit words in the State
- the length of the cipher key, $K$, is 128, 192, or 256 bits
    - this is represented by $Nk = 4$, 6, or 8 which is the number of 32-bit words in the key

## 2.5 Huffman Coding

A technique for lossless data compression. Generates a series of prefix codes which are sent in lieu of the actual characters.

### 2.5.1 Compression

- In this method, a binary tree of nodes is created
- There are two different types of nodes, *leaf nodes* and *internal nodes*
- Leaf Nodes contain:
  - the symbol itself
  - the weight (frequency of appearance) of the symbol
  - optionally, a link to a parent node
- Internal Nodes contain:
  - symbol weight
    - generally the sum of the weights of its children
  - links to two child nodes
    - commonly, bit 0 represents the left child and bit 1 represents the right child
  - optionally, a link to a parent node
- The process starts with the leaf nodes representing all of the symbols and their probabilities
- A new, internal, node is added, whose children are the two nodes with the smallest probability
- This new node's probability is equal to the sum of its children's probability
- The two nodes whose parent was just added are no longer considered, with their parent now being considered
- This process is repeated until an entire tree is formed

There are two general methods to constructing the tree, using a priority queue and using two queues. The priority queue method runs in logarithmic time, $O(nlog(n))$, while the method using two queues can run in linear time, $O(n)$. However, the two queue method also requires a sort prior to commencing, which is generally $O(nlog(n))$. The two methods end up being similarly complex in a time sense. In addition, the $n$ is normally fairly small, rendering time complexity of little importance.

### 2.5.2 Using a Priority Queue

- In the priority queue, the lowest probability is given the highest priority
- runs in logarithmic time, $O(nlog(n))$

1. Create a leaf node for each symbol and add it to the queue
2. While there is more than one node in the queue
   1. Remove the two nodes of highest priority (lowest probability) from the queue
   2. Create a new internal node with these two nodes as children and probability equal two their sum
   3. Add the new node to the queue
3. The remaining node is the root node, tree is complete

### 2.5.3 Using Two Queues

• If the symbols are sorted via probability, this runs in linear time

1. Start with as many leaves as there are symbols
2. Add all nodes into the first queue, with the least likely (lowest probability) node at the front of the queue
3. While there is more than one node in the queues
    1. Remove the two nodes with the lowest weights from the queues
    2. Create an internal node with the two nodes as children and weight being equal to their sum
    3. Add the new node to the back of the second queue
4. The remaining node is the root node, tree complete

• Break ties by choosing the node in the first queue

### 2.5.4 Decompression

Transmit a series of prefix codes. Assuming the receiver has knowledge of the structure of the tree, which can be sent beforehand or with the codes, it traverses the tree and finds the character at each leaf.

## 2.6 LZ77

A technique for lossless data compression. A dictionary coder, it maintains a sliding window during compression. Generates output similar to run-length encoding.

## 2.7 DEFLATE

A technique for lossless data compression, defined in `RFC 1951`. It uses a combination of Huffman coding and LZ77.

## 2.8 Discrete Cosine Transform (DCT)

## 2.9 Metropolis Algorithm for Monte Carlo

Also known as the Metropolis-Hastings algorithm, it is a Markov chain Monte Carlo (MCMC) method for generating a sequence of random samples from a probability distribution in which direct sampling is difficult. Used to approximate a distribuion.

## 2.10 Simplex Method for Linear Programming

A linear programming problem is generally defined as the problem of maximizing or minimizing a linear function, subject to linear constraints.

## 2.11 Krylov Subspace Iteration Methods

## 2.12 The Decompositional Approach to Matrix Computations

## 2.13 The Fortran Optimizing Compiler

The first optimizing compiler to be programmed. It has influenced most, if not all, modern compilers.

## 2.14 QR Algorithm for Computing Eigenvalues

An algorithm used to find the eigenvalues and eigenvectors of a matrix.

### 2.14.1 QR Decomposition

A matrix decomposition of a matrix $A$ into $A = QR$, where $Q$ is an orthogonal matrix and $R$ is an upper triangular matrix.

- For a square matrix
  - a real matrix, $A$ may be decomposed as $A = QR$
  - if $A$ is invertible, the it has a unique factorization if the diagonal elements of $R$ need to be positive
  - if $A$ is complex, then there is a decomposition $A = QR$, where $Q$ is a unitary matrix ($Q^*Q = I$)
  - if $A$ has $n$ linearly independent columns:
    - the first $n$ columns of $Q$ form an orthonormal basis for the column space of $A$
      - generally: the first $k$ columns of $Q$ form an orthonormal basis for the span of the first $k$ columns of $A$ for any $1 \leq k \leq n$
      - the fact that any column $k$ of $A$ only depends on the first $k$ columns of $Q$ is responsible for the triangular form of $R$
- For a rectangular matrix
  - for a complex matrix, $A$, of dimensions, $m \times n$, and $m \geq n$, $A$ can be factored as the product of an $m \times m$ unitary matrix $Q$ and an $m \times n$ upper rectangular matrix $R$
  - as the bottom $(m - n)$ rows of an $m \times n$ upper rectangular matrix consists of zeroes, $R$ or both $R$ and $Q$ be partitioned as:

$$A = QR = \begin{bmatrix} R_1 \\ 0 \end{bmatrix} = \begin{bmatrix} Q_1, Q_2 \end{bmatrix} \begin{bmatrix} R_1 \\ 0 \end{bmatrix} = Q_1 R_1$$

where $R_1$ is an $nxn$ upper triangular matrix, 0 is an $(m - n) \times n$ zero matrix, $Q_1$ is $m \times n$, $Q_2$ is $m \times (m - n)$, and $Q_1$ and $Q_2$ both have orthogonal columns

## 2.15 Quicksort Algorithm for Sorting

Generally the most efficient method of sorting, with an average complexity of $O(nlog(n))$. A comparison sort, it is generally not stable, and can be performed in place.

## 2.16 Fast Fourier Transform (FFT)

A method for computing the Discrete Fourier Transform (DFT) of a sequence, or its inverse.

## 2.17 Integer Relation Detection

An integer relation between a set of real numbers, $x_1, x_2, \ldots, x_n$ is the set of integers, $a_1, a_2, \ldots, a_n$, not all zero, such that: $a_1x_1 + a_2x_2 + \ldots + a_nx_n = 0$. Integer relation detection algorithms find these relations, or find that, given a upper bound of magnitude, no integer relation exists.

## 2.18 Fast Multipole Method

A method developed to speed up the calculation of long-ranged forces in the n-body problem.

# 3 Matrix Decompositions

## 3.1 Cholesky

## 3.2 LU

### 3.2.1 LU with Full Pivoting

## 3.3 Block LU

## 3.4 QR

## 3.5 Spectral

## 3.6 Schur

## 3.7 Singular Value

# 4 Math Notes

## 4.1 Binary to Hexadecimal

| Bit Pattern | Character |
|---|---|
| 0000 | 0 |
| 0001 | 1 |
| 0010 | 2 |
| 0011 | 3 |
| 0100 | 4 |
| 0101 | 5 |
| 0110 | 6 |
| 0111 | 7 |
| 1000 | 8 |
| 1001 | 9 |
| 1010 | a |
| 1011 | b |
| 1100 | c |
| 1101 | d |
| 1110 | e |
| 1111 | f |

The above tables show how groups of four bits are converted into hexadecimal characters. A binary number, e.g., $\{01101101\}$ can thus be represented as: $\{6d\}$. In the AES spcecification, there are mainly eight bit numbers used. However, occasionally a nine bit number will appear, in this case the leading bit is represented as $\{01\}$ which is added to the front. An example of this: $\{01\}\{1b\}$ represents $\{100011011\}$.

## 4.2 Finite Field Mathematics

This deals with the finite field used in AES. Addition and multiplication can be used in this field, but the operations differ from the ones normally used.

### Addition

- the addition of two elements is interpreted by performing the XOR operation on the coefficients for corresponding powers of the polynomial representation of the elements
- subtraction of elements is identical to the addition of elements
- another way of interpreting addition is the modulo 2 addition of the corresponding bits in the byte
  - for two bytes, $\{a_7a_6a_5a_4a_3a_2a_1a_0\}$ and $\{b_7b_6b_5b_4b_3b_2b_1b_0\}$, the sum is $\{c_7c_6c_5c_4c_3c_2c_1c_0\}$, where $c_i = a_i \oplus b_i$
  - the following are equivalent:
    - $(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2$
    - $\{01010111\} \oplus \{10000011\} = \{11010100\}$
    - $\{57\} \oplus \{83\} = \{d4\}$

### Multiplication

- multiplication, denoted by $\bullet$ corresponds with the multiplication of polynomials modulo an irreducible polynomial of degree 8.
  - a polynomial is irreducible if its only divisors are one and itself
- in the AES algorithm, the irreducible polynomial is: $m(x) = x^8 + x^4 + x^3 + x + 1$, or $\{01\}\{1b\}$
- Example of multiplication: $\{57\} \bullet \{83\} = \{c1\}$

  $(x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1) =$

$$= x^{13} + x^{11} + x^9 + x^8 + x^7 + x^7 + x^5 + x^3 + x^2 + x + x^6 + x^4 + x^2 + x + 1$$
$$= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$$
$$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \text{ modulo } (x^8 + x^4 + x^3 + x + 1)$$
$$= x^7 + x^6 + 1$$

• this reduction by a polynomial ensures that the result of multiplication will be a binary polynomial of a degree less than eight, i.e., representable using a byte

• this multiplication is associative, and has an identity element of $\{01\}$

• for a non-zero binary polynomial $b(x)$ of degree less than eight, the multiplicative inverse of $b(x)$, denoted $b^{-1}(x)$ can be found like so:

– the extended Euclidean algorithm is used to calculate polynomials $a(x)$ and $c(x)$ such that $b(x)a(x) + m(x)c(x) = 1$

– from this: $a(x) \bullet b(x) \bmod m(x) = 1$ which gives: $b^{-1}(x) = a(x) \bmod m(x)$

– for any $a(x)$, $b(x)$, and $c(x)$ in the field: $a(x) \bullet (b(x) + c(x)) = a(x) \bullet b(x) + a(x) \bullet c(x)$

**Multiplication by x**

• multiplying $b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0$ with the polynomial $x$ gives:

$$b_7x^8 + b_6x^7 + b_5x^6 + b_4x^5 + b_3x^4 + b_2x^3 + b_1x^2 + b_0x$$

– the result of $x \bullet b(x)$ is found by reducing the above modulo $m(x)$

– if $b_7$ is zero, then it is already in reduced form

– if $b_7$ is one, then the reduction is performed by subtracting (XOR-ing) $m(x)$ from the result

• multiplication at the byte level by $x$ (i.e., $\{00000010\}$ or $\{02\}$) can be implemented as a left shift and a conditional bitwise XOR by $\{1b\}$

– the XOR is only performed if the most significant bit of the original number, i.e., $b_7$, was a one

– this operation, in the AES specification, is denoted using `xtime()`

– multiplication by higher powers of $x$ can be implemented using repeated applications of `xtime()`

– from the addition of intermediate results, multiplication by any constant can be implemented

$$\{57\} \bullet \{13\} = \{f3\} \text{ because:}$$
$$\{57\} \bullet \{02\} = xtime(\{57\}) = \{ae\}$$
$$\{57\} \bullet \{04\} = xtime(\{ae\}) = \{47\}$$
$$\{57\} \bullet \{08\} = xtime(\{47\}) = \{8e\}$$
$$\{57\} \bullet \{10\} = xtime(\{8e\}) = \{07\}$$

– from this:

$$\begin{aligned}\{57\} \bullet \{13\} &= \{57\} \bullet (\{01\} \oplus \{02\} \oplus \{10\}) \\ &= \{57\} \oplus \{ae\} \oplus \{01\} \\ &= \{fe\}\end{aligned}$$

– a C macro for `xtime()`: `#define xtime(n) ((n<<1)`$\oplus$`(((n>>7) & 1)* 0x11b))`

∗ the right side of this macro, tests whether the MSB of `n` is 0 or 1, it then multiplies the result by $m(x)$ or $\{01\}\{1b\}$

∗ if the right side comes to 0, then `n<<1` is returned, as a number XOR 0 is the number

∗ if the right side comes to 1, then `n<<1` is returned XOR `0x11b` ($\{01\}\{1b\}$)

## Polynomials with Coefficients in GF($2^8$)

- four-term polynomials, with coefficients that are finite field elements, are defined:

$$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$$

- these are denoted as a word in the form: $[a_0, a_1, a_2, a_3]$
- these polynomials behave differently than the ones used in the definition of finite field elements
  - the coefficients themselves are finite field elements, i.e., bytes instead of bits
- let $b(x) = b_3x^3 + b_2x^2 + b_1x + b_0$ be a four-term polynomial
  - addition is performed by adding the finite field elements of like powers of $x$
  - i.e., XOR the corresponding bytes

$$a(x) + b(x) = (a_3 \oplus b_3)x^3 + (a_2 \oplus b_2)x^2 + (a_1 \oplus b_1) + (a_0 \oplus b_0)$$

- multiplication is performed in two steps, the first of which is expanding the polynomial product: $c(x) = a(x) \bullet b(x)$

$$c(x) = c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0$$

  - where:

$$\begin{aligned}
c_0 &= a_0 \bullet b_0 \\
c_1 &= a_1 \bullet b_0 \oplus a_0 \bullet b_1 \\
c_2 &= a_2 \bullet b_0 \oplus a_1 \bullet b_1 \oplus a_0 \bullet b_2 \\
c_3 &= a_3 \bullet b_0 \oplus a_2 \bullet b_1 \oplus a_1 \bullet b_2 \oplus a_0 \bullet b_3 \\
c_4 &= a_3 \bullet b_1 \oplus a_2 \bullet b_2 \oplus a_1 \bullet b_3 \\
c_5 &= a_3 \bullet b_2 \oplus a_2 \bullet b_3 \\
c_6 &= a_3 \bullet b_3
\end{aligned}$$

  - the result does not represent a four-byte word, thus, it must be reduced modulo a polynomial of degree four
  - for the AES algorithm the polynomial is $x^4 + 1$, so that:

$$x^i \; mod(x^4 + 1) = x^{i \; mod \; 4}$$

  - the modular product of $a(x)$ and $b(x)$, denoted by $a(x) \otimes b(x)$ is given by the four term polynomial, $d(x)$

$$d(x) = d_3x^3 + d_2x^2 + d_1x + d_0$$

  - where:

$$\begin{aligned}
d_0 &= (a_0 \bullet b_0) \oplus (a_3 \bullet b_1) \oplus (a_2 \bullet b_2) \oplus (a_1 \bullet b_3) \\
d_1 &= (a_1 \bullet b_0) \oplus (a_0 \bullet b_1) \oplus (a_3 \bullet b_2) \oplus (a_2 \bullet b_3) \\
d_2 &= (a_2 \bullet b_0) \oplus (a_1 \bullet b_1) \oplus (a_0 \bullet b_2) \oplus (a_3 \bullet b_3) \\
d_3 &= (a_3 \bullet b_0) \oplus (a_2 \bullet b_1) \oplus (a_1 \bullet b_2) \oplus (a_0 \bullet b_3)
\end{aligned}$$

  - when $a(x)$ is a fixed polynomial, this can be written in matrix form:

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

  - because $x^4 + 1$ is not irreducible over GF($2^8$), multiplication by a fixed four-term polynomial is not necessarily invertible
  - the AES algorithm specifies a fixed four-term polynomial that does have an inverse:

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$$

$$a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$$

– AES uses another polynomial which has: $a_0 = a_1 = a_2 = \{00\}$ and $a_3 = \{01\}$, this is equivalent to: $x^3$

– this polynomial forms the output word by rotating the input word, i.e., $[b_0, b_1, b_2, b_3]$ is transformed into: $[b_1, b_2, b_3, b_0]$

## 4.3 Elementary Number Theory

• the integer $n$ divides the integer $a$ if and only if there exists an integer $d$ such that $a = nd$

– thus, $a$ is divisible by $n$, $n$ is a divisor or factor of $a$, and $a$ is a multiple of $n$

– the notation $n|a$ means that $n$ divides $a$, and that $n|a - b$ means that $n$ divides $(a - b)$

• if $mn|a$, then $m|a$ and $n|a$ for any integers $m$ and $n$

• a prime number is an integer greater than one, which only has positive integer divisors of one and itself

– a non-prime number greate than one is called a composite number

• The Fundamental Theorem of Arithmetic: every integer $n > 1$ can be represented in exactly one way as a product of primes except for the order of the factors

– i.e., $n$ can be uniquely written in the form: $n = p_1^{k_1} \cdot p_2^{k_2} \cdot \ldots \cdot p_r^{k_r}$ for primes $p_1 < p_2 < \cdot < p_r$ and positive integers $k_1, \ldots, k_r$

• two numbers $a$ and $b$ which have no common factors other than one are said to be coprime or relatively prime

• the greatest common divisor of two integers $a$ and $b$ is the largest integer that divides both numbers

• $a$ and $b$ are coprime if and only if $gcd(a, b) = 1$

• *mod* as a binary operation: $a = b \ mod \ n$, defines an operation by which $a$ is equal to the remainder of dividing $b$ by $n$, $0 \leq a < n$

• *mod* as a congruence relation: $a \equiv b \ (mod \ n)$ means that $a$ and $b$ have the same remainder when dividing by $n$

– or, $a$ is congruent to $b$ modulo $n$

• every integer is congruent modulo $n$ to exactly one of the integers in the set of least positive residues: $\{0, 1, 2, \ldots, n - 1\}$

• properties of congruence: for a fixed positive integer $n$ and any integers $a$, $b$, $c$, $d$

– reflexive: $a \equiv a \ (mod \ n)$

– symmetric: if $a \equiv b \ (mod \ n)$ then $b \equiv a \ (mod \ n)$

– transitive: if $a \equiv b \ (mod \ n)$ and $b \equiv c \ (mod \ n)$ then $a \equiv c \ (mod \ n)$

– addition and multiplication rules: if $a \equiv b \ (mod \ n)$ and $c \equiv d \ (mod \ n)$ then $a \pm c \equiv b \pm d \ (mod \ n)$

– cancellation rule: if $ac \equiv bc \ (mod \ n)$ and $c \neq 0$ then $a \equiv b \ (mod \ \frac{n}{gcd(c,n)})$, if $gcd(c, n) = 1$ then $a \equiv b \ (mod \ n)$

– associative and distributive: $(a + b) + c \equiv a + (b + c) \ (mod \ n)$, $(ab)c \equiv a(bc) \ (mod \ n)$, and $a(b + c) \equiv ab + bc \ (mod \ n)$

– if $a \equiv b \ (mod \ n)$ then $a^r \equiv b^r \ (mod \ n)$, for any integer $r \geq 1$

– $a \equiv 0 \ (mod \ n)$ if and only if $n|a$

• if $m$ and $n$ are coprime and $a \equiv b \ (mod \ m)$ and $a \equiv b \ (mod \ n)$, then $a \equiv b \ (mod \ mn)$

• the Euler Totient Function of a positive integer $n$, denoted $\phi(n)$, is the number of positive integer not exceeding $n$ which are relatively prime to $n$

• for any prime $p$, $\phi(p) = p - 1$, since all numbers less than $p$ are coprime with it

• if $m$ and $n$ are coprime, then $\phi(m)\phi(n) = \phi(mn)$

- Fermat's Little Theorem: If $p$ is a prime and $a$ is any integer, then $a^p \equiv a \ (mod\ p)$
  - if $gcd(a, p) = 1$ then $a^{p-1} \equiv 1 \ (mod\ p)$
- Euler-Fermat Theorem: if $n$ is a positive integer and $a$ is any integer with $gcd(a, n) = 1$, then $a^{\phi(n)} \equiv 1 \ (mod\ n)$
- the Carmichael function of a positive integer $n$, denoted $\lambda(n)$, if defined as the smallest positive integer $m$ such that $a^m \equiv 1 \ (mod\ n)$ for all integers relatively prime to $n$
  - it can be shown that $\lambda(n)$ divides $\phi(n)$
- the least common multiple of the non-zero integers $a$ and $b$, denoted $lcm(a, b)$, is the smallest positive integer that is a multiple of both $a$ and $b$
- if $n$ is the product of two distinct primes $p$ and $q$, $n = pq$, then $\lambda(pq) = lcm(p - 1, q - 1)$
- the modular multiplicative inverse or modular inverse of an integer $a$ modulo $n$ is an integer $x$ such that $ax \equiv 1 \ (mod\ n)$
  - write $x = a^{-1} \ mod\ n$ or $x = (\frac{1}{a}) \ mod\ n$
  - a modular inverse exists if and only if $gcd(a, n) = 1$

# 5 Linear Programming: An Intro

In general, a linear programmming problem is one that involves the maximization or minimization of a linear function subject to linear constraints. The constraints may be equalities or inequalities.

## 5.1 Standard Problems

There are two kinds of standard problems, the standard maximum problem and the standard minimum problem. All linear programming problems can be converted into standard form.

### The Standard Maximum Problem

Given, an $m$-vector, $\mathbf{b} = (b_1 \dots b_m)^T$,and $n$-vector, $\mathbf{c} = (c_1 \dots c_n)^T$, and an $m \times n$ matrix,

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}$$

of real numbers

The standard maximum problem is: find an $n$-vector, $\mathbf{x} = (x_1, \dots, x_n)^T$, to maximise:

$$\mathbf{c}^T\mathbf{x} = c_1x_1 + \dots + c_nx_n$$

subject to the constraints:

$$\begin{aligned} &a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n \leq b_1 \\ &a_{21}x_1 + a_{22}x_2 + \dots + a_{2n}x_n \leq b_2 \\ &\vdots \qquad\qquad\qquad\qquad\qquad\quad \vdots \qquad \text{(or } \mathbf{Ax} \leq \mathbf{b}\text{)} \\ &a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n \leq b_m \end{aligned}$$

and

$$x_1 \geq 0,\ x_2 \geq 0,\ \dots,\ x_m \geq 0 \qquad \text{(or } \mathbf{x} \geq \mathbf{0}\text{)}$$

### The Standard Minimum Problem

Find an $m$-vector, $\mathbf{y} = (y_1, \dots, y_m)$, to minimize

$$\mathbf{y}^T\mathbf{b} = y_1b_1 + \dots + y_mb_m$$

subject to the constraints:

$$\begin{aligned} &y_1a_{11} + y_2a_{21} + \dots + y_ma_{m1} \geq c_1 \\ &y_1a_{21} + y_2a_{22} + \dots + y_ma_{m2} \geq c_2 \\ &\vdots \qquad\qquad\qquad\qquad\qquad\quad \vdots \qquad \text{(or } \mathbf{y}^T\mathbf{A} \geq \mathbf{c}^T\text{)} \\ &y_1a_{1n} + y_2a_{2n} + \dots + y_ma_{mn} \geq c_n \end{aligned}$$

and

$$y_1 \geq 0,\ y_2 \geq 0,\ \dots,\ y_m \geq 0 \qquad \text{(or } \mathbf{y} \geq 0\text{)}$$

### Converting to Standard Form

- All linear programming problems can be converted to standard form
- a minimum problem can be converted to a maximum by multiplying the objective function by $-1$
  - constraints of the form: $\sum_{j=1}^{n} a_{ij}x_j \geq b_i$ can be changed into the form: $\sum_{j=1}^{n}(-a_{ij}x_j \geq -b_i$
  - from this conversion, two problems arise:
    1. Some constraints may be equalities: an equality constraint $\sum_{j=1}^{n} a_{ij}x_j = b_i$ may be removed, by solving this constraint for some $x_j$ for which $a_{ij} \neq 0$ and substituting this solution into the other constraints and into the objetive function wherever $x_j$ appears, this removes one constraint and one variable from the problem
    2. Some variable may not be restricted to be non-negative: an unrestricted variable, $x_j$ may be replaced by the difference of two non-negative variables, $x_j = u_j - v_j$, where $u_j \geq 0$ and $v_j \geq 0$, this adds one variable and two non-negative constraints to the problem

## 5.2 Duality

To every linear program, there is a dual linear program for which it is deeply connected.

### Standard Program Duality

Standard program: $\mathbf{c}$ and $\mathbf{x}$ are $n$-vectors, $\mathbf{b}$ and $\mathbf{y}$ are $m$-vectors, and $\mathbf{A}$ is an $m \times n$ matrix. Assume $m \geq 1$ and $n \geq 1$

***Definition***: The dual of the standard maximum problem,

maximize $\mathbf{c}^T\mathbf{x}$

subject to the constraints $\mathbf{Ax} \leq \mathbf{b}$ and $\mathbf{x} \geq 0$

is defined to be the standard minimum problem

minimize $\mathbf{y}^T\mathbf{b}$

subject to the constraints $\mathbf{y}^T\mathbf{A} \geq \mathbf{c}^T$ and $\mathbf{y} \geq 0$

If the standard minimum problem that is the dual of the standard maximum problem is transformed into a standard maximum probmem (by multiplying $\mathbf{A}$, $\mathbf{b}$, and $\mathbf{c}$ by $-1$), its dual, by definition, is a standard minimum problem which, when transformed into a standard maximum problem, by the same process, is the original standard maximum problem. Thus, the dual of the standard minimum problem is the standard maximum problem.
From this it can also be shown that:

***Theorem 1***: if $\mathbf{x}$ is feasible for the standard maximum problem and if $\mathbf{y}$ is feasible for its dual, then:

$$\mathbf{c}^T\mathbf{t} \leq \mathbf{y}^T\mathbf{b}$$

***Proof***:

$$\mathbf{c}^T\mathbf{x} \leq \mathbf{y}^{\mathbf{Ax}} \leq \mathbf{t}^T\mathbf{b}$$

The first inequality comes from $\mathbf{x} \geq \mathbf{0}$ and $\mathbf{c}^T \leq \mathbf{y}^T\mathbf{A}$

The second inequality comes from $\mathbf{y} \geq \mathbf{0}$ and $\mathbf{A}\mathbf{x} \leq \mathbf{b}$

***Corollary 1***: if a standard problem and its dual are both feasible, then both are bounded feasible

***Proof***: if $\mathbf{y}$ is feasible for the minimum problem, then Theorem 1 shows that $\mathbf{t}^T\mathbf{b}$ is an upper bound for the values of $\mathbf{c}^T\mathbf{x}$ for $\mathbf{x}$ feasible for the maximum problem. Similarly for the converse

***Corollary 2***: if there exists feasible $\mathbf{x}^*$ and $\mathbf{y}^*$ for a standard maximum problem and its dual such that $\mathbf{c}^T\mathbf{x}^* = \mathbf{y}^{*T}\mathbf{b}$, then both are optimal for their respective problems

***Proof***: if $\mathbf{x}$ is any feasible vector for the standard maximum problem, then $\mathbf{c}^T\mathbf{x} \leq \mathbf{y}^{*T}\mathbf{b} = \mathbf{c}^T\mathbf{x}^*$, which shows that $\mathbf{x}^*$ is optimal, a symmetric argument works for $\mathbf{y}^*$

From the theorem and its corollaries, a fundamental theorem is derived.

***The Duality Theorem***: if a standard linear programming problem is bounded feasible, then so is its dual, their values are equal, and there exists optimal vectors for both problems

There are three possibilities for a linear program. It may be feasible bounded (f.b.), feasible unbounded (f.u.), or infeasible (i). For a program and its dual, there are nine possibilites, three of which are precluded by corollary 1. If a problem and its dual are both feasible, then both must be bounded feasible. Two other possibilites are removed by the Duality Theorem. If a program is feasible bounded, its dual cannot be infeasible.

There are four possibilites for a standard maximum problem and its dual

- both feasible bounded
- standard maximum problem infeasible, its dual feasible unbounded
- standard maximum problem feasible unbounded, its dual infeasible
- both infeasible

From the Duality Theorem, a corollary is found:

***The Equilibrium Theorem***: let $\mathbf{x}^*$ and $\mathbf{y}^*$ be feasible vectors for a standard maximum problem and its dual, respectively, then $\mathbf{x}^*$ and $\mathbf{y}^*$ are optimal if, and only if:

$$y_i^* = 0 \text{ for all } i \text{ for which } \sum_{j=1}^{n} a_{ij}x_j^* < b_i$$

and

$$x_j^* = 0 \text{ for all } j \text{ for which } \sum_{i=1}^{m} y_i^* a_{ij} > c_j$$

***Proof***: the first equation in the theorem implies that $y^* = 0$ unless there is equality in $\sum_j a_{ij}x_j^* \leq b_i$

therefore:

$$\sum_{i=1}^{m} y_i^* b_i = \sum_{i=1}^{m} y_i^* \sum_{j=1}^{n} a_{ij}x_j^* = \sum_{i=1}^{m}\sum_{j=1}^{n} y_i^* a_{ij} x_j^*$$

similarly, the following equation implies:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} y_i^* a_{ij} x_j^* = \sum_{j=1}^{n} c_J x_j^*$$

# 6 Regression

## 6.1 Least Squares

#### Polynomials

## 6.2 Mixed-Models

- a mixed model is a statistical model that contains both fixed and random effects
- a Generalized Linear Mixed Model has the form:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\boldsymbol{\mu} + \boldsymbol{\epsilon}$$

where

$\mathbf{y}$ is a $N \times 1$ column vector, the outcome variable

$\mathbf{X}$ is a $N \times p$ matrix of the $p$ predictor variables

$\boldsymbol{\beta}$ is a $p \times 1$ column vector of the fixed-effect regression coefficients

$\mathbf{Z}$ is the $N \times q$ design matrix for $q$ random effects (the random complement to the fixed $\mathbf{X}$)

$\boldsymbol{\mu}$ is a $q \times 1$ vector of the random effects (the random complement to the fixed $\beta$)

$\boldsymbol{\epsilon}$ is an $N \times 1$ column vector of the residuals

- example: doctors ($q = 407$) see $n_j$ patients
  - the grouping variable is the doctor
  - not every doctor sees the same number of patients
    * ranges from 2 to 40, with an average of 21
  - the total number of patients is the average seen by each doctor:

$$N = \sum_{j}^{q} n_j = 8525$$

- the outcome, $\mathbf{y}$ is a continuous variable, mobility scores
- there are 6 fixed-effect predictors:
  1. Age, in years
  2. Married (0 = no, 1 = yes)
  3. Sex (0 = female, 1 = male)
  4. Red Blood Cell(RBC) count
  5. White Blood Cell(WBC) count
  6. a fixed intercept and random intercept for each doctor
- only the random intercept will be considered in this example
- all other effects will be fixed
- putting this into the model: $q = 407$, $p = 6$, $N = 8525$

$$\mathbf{y} = \begin{bmatrix} mobility \\ 2 \\ 2 \\ \dots \\ 3 \end{bmatrix} \quad \mathbf{X} = \begin{bmatrix} Intercept & Age & Married & Sex & WBC & RBC \\ 1 & 64.97 & 0 & 1 & 6087 & 4.87 \\ 1 & 53.92 & 0 & 0 & 6700 & 4.68 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 1 & 56.07 & 0 & 1 & 6430 & 4.73 \end{bmatrix} \quad \boldsymbol{\beta} = \begin{bmatrix} 4.782 \\ 0.025 \\ 0.011 \\ 0.012 \\ 0 \\ -0.009 \end{bmatrix}$$

• **Z** is quite large, so it will not be shown here, since only random intercepts are being modeled, it is a special matrix that only codes which doctor a patient belongs to

– in this case, it is all 0s and 1s

– each column represents one doctor and each row represents one patient

– if the patient belongs to the doctor in that column, the cell will have a 1, 0 otherwise

• if estimated, $\boldsymbol{\mu}$ would be a column vector, similar to $\boldsymbol{\beta}$

• however, is almost always assumed that:

$$\boldsymbol{\mu} \sim \mathcal{N}(\mathbf{0}, \mathbf{G})$$

read, $\boldsymbol{\mu}$ is distributed as normal with mean zero and variance **G**

## 7 Probability

# 8 Batted Ball Visualizer

- the base batted ball attributes:
  - `v_tot`: $v_o$ is the exit velocity of the ball
  - `theta`: $\theta$ is the angle from horizontal of the flight path
    - up is positive, down is negative
  - `phi`: $\phi$ is the azimuthal angle of the flight path
    - i.e., the angle between the x-axis and the ball's x-y component vector
    - 0 degrees (or radians) is the 1st base line, 90 degrees or $\frac{\pi}{2}$ radians is the 3rd base line
  - `g`: is the gravitational constant, defaulting to -9.80665 meters per second squared
  - `freq`: the sampling frequency, defaulting to 60 hertz

## 8.1 Velocity of a Projectile with Air Resistance

- the force on a projectile due to air resistance can be modeled as: $F_D = -kv^2$
  - $v$ is velocity
  - $k$ is the drag coefficient: $k = \frac{1}{2}\rho A C_D$
  - $\rho$ is the density of air
  - $A$ is the frontal area of the projectile
    - for a baseball: $A = \pi r^2$
  - $C_D$ is the coefficient of drag
    - for a baseball: $C_D = 0.3$

#### Horizontal Motion

- starting from angle with horizon: $\theta$, at time: $t$, with mass: $m$

$$a_x = -\frac{kv_x^2}{m} = \frac{dv_x}{dt}$$
$$-\int \frac{k}{m}dt = \int v_x^{-2}dv_x \Rightarrow -\frac{kt}{m} = \frac{-1}{v_x} + C$$
$$v_x(0) = v_{x0} \rightarrow C = \frac{1}{v_{x0}} \Rightarrow \frac{kt}{m} = \frac{1}{v_x} - \frac{1}{v_{x0}}$$
$$v_x(t) = \frac{1}{\frac{1}{v_{x0}} + \frac{kt}{m}} = (v_{x0}^{-1} + \frac{kt}{m})^{-1}$$

- finding the horizontal distance: $s_x(t)$

$$\int v_x dt = \int (\frac{1}{\frac{1}{v_{x0}} + \frac{kt}{m}})dt$$
$$s_x(t) = \frac{m}{k}\ln(ktv_{x0} + m) + C$$
$$s_x(0) = 0 \Rightarrow C = -\frac{m}{k}\ln(m)$$
$$s_x(t) = \frac{m}{k}(\ln(\frac{ktv_{x0} + m}{m}))$$

#### Vertical Motion

- for vertical motion, there are two solutions, one for upwards motion and one for downwards motion

• for upward motion:

$$a_z = -\frac{kv_z^2}{m} - g = \frac{dv_z}{dt} \Rightarrow\ v_z(t) = \sqrt{\frac{mg}{k}}\tan(C - \sqrt{\frac{gk}{m}}t)$$

$$v_z(0) = v_{z0} \Rightarrow\ C = \arctan(\sqrt{\frac{k}{mg}}v_{z0}) \Rightarrow\ v_z(t) = \sqrt{\frac{mg}{k}}\tan(\arctan(\sqrt{\frac{k}{mg}}v_{z0}) - \sqrt{\frac{gk}{m}}t)$$

– vertical height: $s_z(t)$

$$\int ds_z = \int (\sqrt{\frac{mg}{k}}\tan(\arctan(\sqrt{\frac{k}{mg}}v_{z0}) - \sqrt{\frac{gk}{m}}t))dt$$

$$s_z(t) = \frac{m}{k}\ln(\cos(\sqrt{\frac{mg}{k}}t - \arctan(\sqrt{\frac{k}{mg}}v_{z0}))) + C$$

$$s_z(0) = 0 \Rightarrow\ c = -\frac{m}{k}\ln(\cos(\arctan(\sqrt{\frac{k}{mg}}v_{z0})))$$

$$s_z(t) = \frac{m}{k}\ln(\cos(\sqrt{\frac{gk}{m}}t - \arctan(\sqrt{\frac{k}{mg}}v_{z0}))) - \frac{m}{k}\ln(\cos(\arctan(\sqrt{\frac{k}{mg}}v_{z0})))$$

– $h$: maximum height, $t_a$: time to reach $h$

$$t_a = \sqrt{\frac{m}{gk}}\arctan(\sqrt{\frac{k}{mg}}v_{z0})$$

$$h = -\frac{m}{k}\ln(\cos(\arctan(\sqrt{\frac{k}{mg}}v_{z0})))$$

•for downward motion: $t$ is the decending time

$$a_z = \frac{kv_z^2}{m} - g = \frac{dv_z}{dt} \Rightarrow\ v_z(t) = -\sqrt{\frac{mg}{k}}\tanh(C + \sqrt{\frac{gk}{m}}t)$$

$$v_z(0) = 0 \Rightarrow C = 0 \Rightarrow v_z(t) = -\sqrt{\frac{mg}{k}}\tanh(\sqrt{\frac{gk}{m}}t)$$

– vertical height after $t_a$

$$s_z(t) = -\frac{m}{k}\ln(\cosh(\sqrt{\frac{gk}{m}}t)) + C$$

$$s_z(t) = h \Rightarrow\ C = h \Rightarrow\ s_z(t) = h - \frac{m}{k}\ln(\cosh(\sqrt{\frac{gk}{m}}t))$$

– $t_d$: time required to hit the ground

$$t_d = \sqrt{\frac{m}{gk}}arccosh(e^{\frac{hk}{m}}) = \sqrt{\frac{m}{gk}}arccosh(\sqrt{1 + \frac{kv_{z0}^2}{gn}})$$

• $v(t)$: time at any instant, $T$: total flight time, $R$: range, $v_{hit}$: velocity when hitting the ground

$$v(t) = \sqrt{v_x^2(t) + v_z^2(t)}$$

$$T = t_a + t_d$$

$$R = \frac{m}{k}(\ln(\frac{kTv_{x0} + m}{m}))$$

$$v_{hit} = \sqrt{(v_{x0}^{-1} + \frac{kT}{m})^{-2} + \frac{mg}{k}(\tanh(arcosh(e^{\frac{hk}{m}})))^2}$$

## Calculating Density

# 9 Glossary

- **Affine Transformation**
  - a transformation consisting of multplication by a matrix followed by addition of a vector
- **basis**
  - a set of vectors in a vector space, $V$, is called a basis if the vectors are linearly independent and every vector in $V$ is a linear combination of this set
- **big endian**
  - the most significant byte is stored at an address, and subsquent bytes are stored in lower addresses
- **bit**
  - a binary digit, a 1 or a 0
- **block**
  - in AES, a sequence of binary bits that comprise the input, output, State, and Round Key, the length of a sequence is the number of bits it contains, also interpreted as arrays of bits
- **block cipher**
  - a determistic algorithm which operates on a fixed-length group of bits, called blocks, with an unvarying transformation specified by a symmetric key
- **bounded**
  - for a standard maximum or minimum problem, not unbounded
- **byte**
  - a unit of digital information; typically (read: always) 8 bits
  - historically the number of bits used to encode a single character
- **cipher**
  - series of transformations that converts plain text to cipher text using the cipher key
- **cipher key**
  - secret, cryptographic key that is used by the Key Expansion routine (AES) to generate a series of Round Keys
    - can be pictured as a rectangular array of bytes having four rows and having a number of columns corresponding to the number of words (32-bit) in the key
- **ciphertext**
  - data output from the cipher, or input to the inverse cipher
- **column space**
  - the set of all possible linear combinations of a matrix's column vectors
- **conjugate transpose**
  - the conjugate transpose of $mxn$ matrix $A$ is the matrix, $A^*$ obtained from $A$ by taking the transpose and the taking the complex conjugate of each entry
- **constraint set**
  - the set of feasible vectors
- **coprime**
  - two integers, $a$ and $b$, whose only common factor is one

  - can also be stated as: their greatest common divisor is one, $gcd(a, b) = 1$
  - also called relatively prime, or written as $(a, b) = 1$, or $a \perp b$
- **deterministic algorithm**
  - an algorithm which, given the same inputs, will always return the same output
- **dictionary coder**
- **digest**
  - the output of a hash function, also called a hash
- **discrete Fourier transform**
  - an algorithm which takes a finite sequence of evenly spaced samples of a function and converts them into the list of coefficients of a finite combination of complex sinusoids, ordered by their frequencies
  - the Fourier analysis of finite-domain (or periodic) discrete-time functions
- **eigenvalue**
  - a scalar value, $\lambda$, such that: $T\mathbf{v} = \lambda\mathbf{v}$ is non-zero and holds for some linear transformation $T$ on a vector space $V$ for some non-zero vector $\mathbf{v} \in V$
- **eigenvector**
  - given a linear transformation, $T$, on a vector space, $V$, and that $T\mathbf{v} = \lambda\mathbf{v}$ is true and non-zero for some scalar $\lambda$(eigenvalue), $\mathbf{v} \in V$ is the eigenvector associated with $\lambda$
- **Euler's Totient function**
  - $\phi(n)$, a function which counts the positive integers less than or equal to $n$ that are coprime with $n$
- **feasible**
  - a vector $\mathbf{x}$, for the standard maximum problem, or $\mathbf{y}$ for the standard minimum problem, is said to be feasible if it satisfies the constraints
  - a linear programming problem is said to be feasible if the constraint set is not empty
- **Fourier analysis**
  - the study of the way functions may be represented or approximated by sums of trigonometric functions
- **Fourier transform**
  - an algorithm which decomposes a function of time into the frequencies that compose it
  - the result is a complex valued function of frequency
    - the absolute value (magnitude) represents the amount of that frequency in the original function
    - the complex argument is the phase shift of the basic sinusoid in that frequency
- **hash**
  - generally, a numerical sum representing some data
- **hash function**
  - a function that maps arbitrarily sized data to data of a fixed size, a hash
- **hexadecimal**
  - the base-16 number system
  - digits are: `0 1 2 3 4 5 6 7 8 9 A B C D E F`
- **identity matrix**

  - the simplest non-trivial diagonal matrix, $I$ such that: $I(X) = X$
- **infeasible**
  - a linear programming problem is said to be infeasible if the constraint set it empty
- **inner product space**
  - a vector space with an additional structure: the inner product which associates each pair of vectors to a scalar, known as the inner product of the vectors
- **integer relation**
- **inverse cipher**
  - a series of transformations converting ciphertext to plaintext using the cipher key
- **key expansion**
  - in AES, a routine used to generate a series of Round Keys from the cipher key
- **Krylov subspace**
- **linear programming**
- **linear time**
  - a time complexity, the time to run an algorithm is directly proportional to the length of the input
  - represented in big-O notation as: $O(n)$
- **little endian**
  - the most significant byte is stored at an address, and subsequent bytes are stored in higher addresses
- **logarithmic time**
  - a time complexity, the time to run an algorithm is proportional to the length of the input times the log of the length of the input
  - represented in big-O notation as: $O(nlog(n))$
- **lossless compression**
  - data compression in which the original data can be exactly reconstructed from the compressed data
- **lossy compression**
  - data compression that uses inexact approximations to represent the original data
- **main diagonal**
  - for a matrix, $A$, composed of elements $a_{i,j}$, the main diagonal is the collection of entries such that: $i = j$
- **matrix decomposition**
  - the factorization of a matrix into the product of matrices
- **matrix multiplication**
  - given $\mathbf{A}$, a $n\times$, matrix and $\mathbf{B}$, a $m \times p$, their product, $\mathbf{AB}$, is an $n \times p$ matrix
  - each entry, of $\mathbf{AB}$ is defined as $(\mathbf{AB})_{ij} = \sum_{k=1}^{m} \mathbf{A}_{ik}\mathbf{B}_{kj}$
  - $\mathbf{AB}_{ij}$ = the $ith$ row of $\mathbf{A}$ dotted with the $jth$ column of $\mathbf{B}$
- **Markov chain**
- **Monte Carlo methods**
- **multipole**

- **n-body problem**
- **objective function**
  - in linear programming, the function to be maximized or minmized
- **octet**
  - a sequence of eight bits, i.e., a byte
- **orthogonal matrix**
  - a matrix, $A$, is orthogonal if: $A^T A = I$
    * $A^T$ is the transpose of $A$
    * $I$ is the identity matrix
  - an orthogonal matrix is always invertible and $A^{-1} = A^T$
- **orthonormal basis**
  - a basis for a finite dimensional vector space, $V$, whose vectors are orthonormal, i.e., they are all orthogonal and unit vectors
- **prefix code**
- **priority queue**
  - a data structure which orders its elements by priority
  - elements with the highest priority are at the front of the queue
- **public-key encryption**
  - cryptographic algorithms which use different keys for encryption and decryption
  - one key is kept private, the other may be published freely
- **queue**
  - a data structure that follow a first-in first-out (FIFO) principle
- **Round key**
  - in AES, keys derived from the cipher key using the key expansion routine, applied to the State in the cipher and inverse cipher
- **run-length encoding**
- **S-box**
  - a non-linear substitution table used in several byte substitution transformations and the Key Expansion to perform to perform a one-for-one substitution of a byte value
- **simplex**
- **sliding window**
- **span**
  - the span of a set of vectors in a vector space is the intersection of all subspaces containing the set
- **state**
  - an intermediate result of encryption or hashing algorithms
- **symmetric-key encryption**
  - cryptographic algorithms which use the same key for encryption and decryption
- **time complexity**

- **transpose**
  - the transpose of matrix, $A$ is the matrix $A^T$, it can be created several ways
    - reflecting $A$ over its main diagonal
    - writing the rows of $A$ as the columns of $A^T$
    - writing the columsn of $A$ as the rows of $A^T$
    - more fomally, the $ith$ row, $jth$ column element of $A^T$ is the $jth$ row, $ith$ column element of $A$
      - $[A^T]_{i,j} = [A]_{j,i}$
    - if $A$ is an $m\ x\ n$ matrix, $A^T$ is an $nxm$ matrix
- **unbounded**
  - for maximum problems:
    - the problem is said to be unbounded if the objective function can assume arbitrarily large positive values at feasible vectors
  - for minimum problems:
    - the problem is said to be unbounded if the objective function can assume arbitrarily large negative values at feasible vectors
- **unitary matrix**
  - a matrix, $A$, whose conjugate transpose $A^*$ is also its inverse, i.e., $A^*A = I$, where $I$ is the identity matrix
- **upper triangular matrix**
  - a matrix in which all the entries below the main diagonal are zero
- **value**
  - for maximum problems:
    - the maximum value of the objective function as the variables range over the constraint set
  - for minimum problems:
    - the minimum value of the objective function as the variables range over the constraint set
- **word**
  - a unit of data for a particular processor

## 10 Thoughts on Film

An aspect of writer/directors like Quentin Tarantino or Aaron Sorkin that I enjoy is that it there seems to be an awareness that it is a movie. In Sorkin scripts, there are witty retorts akin to what one would dream up in the shower the next day. The idealized world where people think up the smartest line on the spot is something that is particularly enticing. Tarantino seems to have a similar vision but it extends to the entire movie world.