Informal Notes about PROOFS

- **What is a PROOF?**

A proof is a finite logical sequence of TRUE statements culminating in the statement for which a proof was required.
(Thus the conclusion is also TRUE)

- **What types of Statements are acceptable (admissible) in a PROOF?**

In this case

- A statement is a sentence that could be either TRUE or FALSE.
  Sentences like 'command' are not acceptable.
- Statements which are acceptable as TRUE are:

(1) Assumptions or Hypotheses.

(2) Definitions

(3) Previously known results

(4) Statements which logically follow from earlier statements in the sequence.

Note: Most likely errors in proofs occur in (4).

Typical Propositional Forms

(1) Implication : This is the most common form.

Symbolic Notation : $p \Rightarrow q$

Language Expression : If p, then q

OR p implies q.

(2) The Biconditional :

Symbolic Notation : $p \Leftrightarrow q$

Language Expression : p if and only if q

OR : p is equivalent to q

Note: This is a way to combine two implications. It means $p \Rightarrow q$ and $q \Rightarrow p$ (The converse of $p \Rightarrow q$)

Types of Proofs (for $p \Rightarrow q$):

(1) Direct Proof :

$p \rightarrow$ True (Assumption) : Then a logical sequence of true statement follows.

$$\left.\begin{array}{l} p_1 \\ \vdots \\ p_n \end{array}\right\} \text{(All True)}$$

$q \rightarrow$ The end : True (Conclusion)

(2) Contrapositive Proof:

- The contrapositive of $p \Rightarrow q$ is

$\sim q \Rightarrow \sim p$ (The symbol '$\sim$' stands for negation or not.)

Now $p \Rightarrow q$ and $\sim q \Rightarrow \sim p$ are equivalent.
(By a Theorem of Logic)

This provides another type of proof

We start with : $\sim q \rightarrow$ True (Assumption)

$$\left.\begin{array}{l} p_1 \\ \vdots \\ p_n \end{array}\right\} \text{(All True)}$$

$\sim p \rightarrow$ True (The desired conclusion)

Conclusion: Proving $\sim q \Rightarrow \sim p$ yields a proof of $p \Rightarrow q$

(3) Proof BY WAY OF CONTRADICTION (BWOC):

Here we prove p and $\sim q \Longrightarrow r$
(Notation: $p \wedge \sim q$ ($\wedge \equiv$ and))

where r is some proposition known to be false (NOT TRUE).

Thus we start with:

$p \longrightarrow$ True (By the given assumption)

$\sim q \longrightarrow$ New assumption .... ⊛
(Introduced by the Prover)

⋮

$r \longrightarrow$ follows from the proof

$\sim r \longrightarrow$ True (Previously known)

So, r and $\sim r$ is true which is not possible.
Thus there is something wrong in the proof.
The only possible place for error is the assumption ⊛. Thus $\sim q$ is false.
So, q is true and hence $p \Rightarrow q$ is true which is what we wanted.

SOME TIPS FOR WRITING PROOFS

(1) Be clear about
- what is given (assumptions/hypotheses)

and
- What is the desired conclusion (to be proved)

You can write:

GIVEN

RTP (Required to be proved)

at the start of the proof

- This is particularly necessary for 'if and only if' propositions.

(2) <u>Pay attention to notation:</u>

- Write down what each symbol stands for.
- Do not introduce a new symbol without stating what it stands for.
- Do not use the same symbol for two different objects.
- Clearly distinguish vectors and scalars.

(3) Check that each statement (step) in the proof is legitimate.

<u>Note:</u> A common error is to write a statement which is <u>not a known result</u>. It may be false or just as hard to prove as the desired conclusion.

(4) Use short simple statements as steps in the proof: with explanation if necessary (put in brackets if appropriate).

(5) Use precise mathematical language and symbols/equations as far as possible.

(6) <u>Objectives of the proof:</u>

Every proof is written for a certain class of readers and it has to be appropriate for them.

<u>e.g.</u> (1) Proofs in journal articles: By

experts for experts : Brief with many steps left out.

(2) Text books and Lecture notes: For learners:

— Usually steps are not left out and there are extra explanations

(3) By a student in a test: Your aim is to convince the examiner that you have understood the logic. Do not leave gaps and cite used results explicitly.