



**UNIVERSITY COLLEGE LONDON**

**Department of Physics and Astronomy**

**LABORATORY I:**

**STUDENT'S HANDBOOK**

**Evening Course 1B70 – PRACTICAL SKILLS**

**Course Supervisor: Dr Malcolm Coupland**

**SESSION 2002/2003**

## CONTENTS

<b>1</b>	<b>INTRODUCTION</b> .....	<b>4</b>
<b>2</b>	<b>SAFETY IN THE LABORATORY</b> .....	<b>4</b>
<b>3</b>	<b>OBJECTIVES</b> .....	<b>5</b>
<b>4</b>	<b>THE COMPONENTS OF THE COURSE</b> .....	<b>5</b>
<b>5</b>	<b>ORGANISATION OF COURSE</b> .....	<b>6</b>
5.1	TIMETABLE AND ATTENDANCE .....	6
5.2	LECTURES .....	6
5.3	COMPUTER PROGRAMMING AND WORD-PROCESSING .....	7
5.4	PHYSICS LABORATORY STAFF.....	7
5.5	THE PHYSICS LABORATORY EXPERIMENTS .....	8
5.6	LABORATORY NOTEBOOKS .....	8
5.7	DEADLINES AND PENALTIES .....	8
<b>6</b>	<b>LABORATORY PROCEDURES</b> .....	<b>9</b>
6.1	THE STUDENT'S RESPONSIBILITIES .....	9
6.2	PERFORMING THE EXPERIMENTS .....	10
6.3	INSTRUCTION SHEETS ( <i>SCRIPTS</i> ).....	11
6.4	APPARATUS .....	11
<b>7</b>	<b>THE LABORATORY NOTEBOOK</b> .....	<b>12</b>
7.1	THE AIM.....	12
7.2	THE CONTENTS.....	12
	THE <i>CONCLUSIONS</i> SECTION.....	14
7.3	CHECKING BY THE SUPERVISOR.....	15
7.4	ASSESSMENT OF LABORATORY NOTEBOOKS.....	15
<b>8</b>	<b>FORMAL REPORTS</b> .....	<b>16</b>
8.1	PURPOSE.....	16
8.2	CONTENT.....	16
8.3	STYLE.....	17
8.4	ASSESSMENT OF FORMAL REPORTS .....	17
<b>9</b>	<b>COMPUTING</b> .....	<b>18</b>
9.1	MISUSE OF LABORATORY COMPUTERS .....	18
9.2	NECESSARY MATERIALS.....	18
9.3	BOOKS .....	18
9.4	THE COMPUTERS .....	19
9.5	REGISTRATION .....	19
9.6	USING THE COLLEGE PCS.....	19
9.7	USING THE PHYSICS DEPARTMENT PCS .....	20
9.8	USING YOUR OWN COMPUTER.....	20
9.9	DISKS .....	20
9.10	PRINTING.....	21
9.11	ASSESSMENT OF WORD-PROCESSING .....	22
9.12	ASSESSMENT OF COMPUTER PROGRAMMING.....	22
<b>10</b>	<b>COURSE COMPLETION AND ASSESSMENT</b> .....	<b>22</b>
<b>11</b>	<b>STUDENT EVALUATION OF THE COURSE</b> .....	<b>23</b>
<b>12</b>	<b>DOCUMENTATION REQUIRED FOR THE TREATMENT OF EXPERIMENTAL DATA AND THE COMPUTING COMPONENTS OF THE COURSE</b> .....	<b>23</b>

<b>13</b>	<b>TREATMENT OF EXPERIMENTAL DATA.....</b>	<b>25</b>
13.1	INTRODUCTION.....	25
13.2	THE AVERAGING METHOD.....	27
13.3	THE GRAPHICAL METHOD.....	35
13.4	PROPAGATION OF UNCERTAINTIES.....	39
13.5	MISCELLANEOUS.....	44
13.6	EXERCISES.....	48
<b>14</b>	<b>WORD-PROCESSING.....</b>	<b>50</b>
14.1	INTRODUCTION: DOCUMENTATION, ASSESSMENT, DEADLINES.....	50
14.2	IF YOU ARE ALREADY EXPERIENCED IN WORD-PROCESSING.....	51
14.3	GUIDE TO THE TUTORIAL.....	51
14.4	IMPORTANT TOPICS NOT COVERED BY THE TUTORIAL.....	54
14.5	TYPOGRAPHY.....	56
14.6	ADVANCED EXERCISE.....	56
14.7	SUMMARY OF WORD FEATURES.....	57
14.8	SPECIAL SYMBOLS AVAILABLE USING THE <i>SYMBOL</i> AND <i>MT EXTRA</i> FONTS.....	59
<b>15</b>	<b>COMPUTER PROGRAMMING IN VISUAL BASIC.....</b>	<b>60</b>
15.1	INTRODUCTION.....	60
15.2	BOOKS.....	60
15.3	THE COURSE-WORK.....	60
15.4	STARTING VISUAL BASIC.....	60
15.5	THE KEYBOARD.....	61
15.6	WHAT <i>IS</i> A COMPUTER PROGRAM?.....	61
15.7	VISUAL BASIC'S WINDOWS AND MENUS.....	63
15.8	USE OF DISKS.....	66
15.9	USE OF THE PRINTER.....	67
15.10	USING VISUAL BASIC 5 CCE.....	68
<b>16</b>	<b>VISUAL BASIC COURSE-WORK PROBLEMS.....</b>	<b>69</b>
	UNIT 1.....	69
	UNIT 2.....	75
	UNIT 3.....	81
	UNIT 4.....	84
	UNIT 5.....	87
<b>APPENDIX A</b>		
<b>AIMS AND OBJECTIVES OF THE PRACTICAL SKILLS COURSE PROGRAMME</b>		
.....		<b>91</b>
<b>APPENDIX B</b>		
<b>OBJECTIVES OF THE TREATMENT OF EXPERIMENTAL DATA COURSE</b>		
<b>COMPONENT.....</b>		<b>92</b>
<b>APPENDIX C</b>		
<b>OBJECTIVES OF THE COMPUTING COURSE COMPONENT.....</b>		<b>93</b>

*While every effort has been made to ensure the accuracy of the information in this document, the Department cannot accept responsibility for any errors or omissions contained herein.*

*A copy of this document can be found on the Department web site: [www.phys.ucl.ac.uk](http://www.phys.ucl.ac.uk)*

# 1 Introduction

Welcome to the First Year Practical Skills course. This course is one component of an integrated practical skills programme extending through the four years of your degree. Generally speaking the *aim* of practical skills training is to equip you with those skills relating to laboratory physics which an employer would be likely to expect to find in a graduate in physics whether you are employed as a scientist or in a related field. To fulfil this aim the department sets a series of *objectives* for the programme which are stated in Appendix A. The aim of the first year practical skills course is to fulfil some of these objectives as detailed below.

In fulfilling these objectives we intend that you should find the course interesting as well as being educational and informative. If you do not, we will consider this a failure on our part. In a later section you will learn how we ask for your help in giving us feedback on your experiences in the course. This is invaluable to the department in making the course a more useful and relevant experience for you. We ask for your full cooperation in this process.

# 2 Safety in the Laboratory

- Safety is a paramount consideration in the design of the courses which run in Laboratory I. In general the experiments carry no greater level of hazard than you would encounter when using a domestic appliance except for a few cases where low level hazards particular to physics experimentation are involved (low powered lasers, weak radio-active sources etc). In these cases the scripts for the experiments will carry information and warnings as to the particular hazards and the safety regime to be followed to ensure safe practice at all times. You are expected to follow such safety instructions to the letter and both the supervisor and the lab technicians will check to ensure that you do so.
- All equipment used in the laboratory is regularly maintained and in particular mains powered electrical equipment is subject to testing to make sure that it conforms to required safety standards. Testing is carried out in the long vacation preceding the academic year. All electrical equipment used in the laboratory will carry a sticker indicating that this has been done and the date when. Visual checks are carried out during the term by the technical staff to spot equipment which has deteriorated in use. Nevertheless be vigilant in ensuring your own safety. If you notice a mains plug which is coming loose or mains lead which is cracking etc, please draw these to the attention of the lab technician who is instructed to deal promptly with such matters.
- In spite of the courses being designed with safety in mind, a laboratory can be an unsafe place if the occupants behave irresponsibly in it. Any such behaviour will be viewed seriously and can in extreme cases lead to disciplinary action.
- Note that maintaining a disciplined atmosphere and tidy environment in the laboratory is a prerequisite of safe practice. For this and other reasons College regulations prohibit eating, smoking and drinking in the laboratory. Students are reminded that College regulations also prohibit the use of mobile phones.
- Apart from hazards intrinsic to experimentation, the only general hazard which may affect you in the laboratory, as in other parts of the College, is that of fire. Should fire break out in the laboratory or the alarms sound to warn of fire elsewhere in the building then your *only* duty is to cooperate in the speedy and orderly evacuation of the laboratory to the assembly point. The laboratory technician is the fire warden for the area and will, with help from the supervising staff, direct you through the fire exits at each end of the laboratory, down the stairs to the ground floor and out of the building to an assembly point. Cooperate at all times. Do not attempt to fight fires yourself and do not attempt to use a lift in leaving the building.

- Should an accident occur which requires medical attention, inform the supervisor or the laboratory technician who will take the initiative in summoning a first aid trained person or other medical assistance.

### 3 Objectives

At the end of the course you should —

- have acquired increased skill and confidence in the acquisition of experimental data through the performance of experiments at the introductory level;
- have become familiar with some of the basic items of equipment used in a physics laboratory;
- have improved your ability to record your work concisely and precisely as you perform it through repeated practice in recording experiments in your laboratory notebook, and with frequent feedback from teachers;
- have increased understanding and ability to apply the principles of data and uncertainty analysis to experiments performed, as introduced in lectures;
- have increased ability to condense the information in your personal notebook record of an experiment into a concise but precise and complete formal report of the experiment in the appropriate style and format;
- have become conversant with and be able to use the Visual Basic programming language to a level sufficient for basic calculations, input of data, and output of results as typically encountered in the analysis of experimental data at the introductory level.
- have become conversant with and be able to use the *Microsoft Word* word-processing program to a level sufficient for the preparation of a well formatted text-based formal report.

### 4 The Components of the Course

**1B70** is a half unit laboratory-based practical course for first year students taking the part-time evening Physics B.Sc. degree.

The course is taken in **Terms 1 and 2** and has the following mandatory components:

- **Treatment of Experimental Data:** A course of about 6 lectures on the numerical assessment of experimental data. This course provides the basic mathematical and statistical tools needed when analysing physics experiments. There is one study aid problem sheet associated with the course. (See Appendix B for detailed objectives.)
- **Computing:** A course of about 18 lectures and practical sessions providing an introduction to the basic principles of word-processing and computer programming. There are 5 assessed computer programming tasks and one assessed word-processing task associated with the course. (See Appendix C for detailed objectives.)

- **Physics Laboratory Experiments:** This laboratory course of 14 three hour sessions covers the basic techniques of laboratory physics as well as illustrating some aspects of the physics topics taught in the lecture courses. It is presented via a number of experiments which fall roughly into three groups: introductory, basic, and extended. It is possible to complete the course by performing a sufficient number of experiments from the *introductory* and *basic* categories. The more able students will progress to undertake some experiments from the *extended* category which are generally of a more challenging nature. Any experiment from this latter group not completed in the first year will normally be undertaken as part of the second year course. Students must complete a reasonable number of these experiments (see Section 10) which must be recorded and reported as described below (Sections 7 & 8).
- **Formal Experiment Reports:** Full formal reports are to be produced for two of the Physics experiments that you do.

Marks are awarded by continuous assessment of each of the above components and count toward the final degree on the same footing as for any other half-unit course (see Section 10).

## 5 Organisation of Course

### 5.1 Timetable and Attendance

The course begins in Term 1 for eight weeks with the lectures on *Treatment of Experimental Data* and the computer programming and word-processing. This is held on Thursday evenings with the TED lecture in the first hour and the computing work in the second two hours. The TED lectures take place in a lecture room (to be assigned) and the computing takes place in the “Cluster Room” D105 (east end of the building on the first floor).

The last three weeks of Term 1 and all of Term 2 are devoted to Physics Laboratory work which takes place in Lab I (west end of building, first floor). Each student will attend this section of the course for one three-hour evening each week on either Tuesday or Thursday. You will be informed about half way through the term which evening you are to attend. You may only attend on the assigned evening except under special circumstances and by prior arrangement with the Course Supervisor and subject to the availability of apparatus and space.

The Physics Laboratory will be closed during vacations and throughout the third term.

### 5.2 Lectures

The lectures on *Treatment of Experimental Data* cover the most basic techniques of analysis that you will apply during the physics laboratory course. Although the problem sheet for the lectures is not part of the course assessment, a proper understanding of the methods of analysis covered in these lectures is essential if you are to obtain a satisfactory level of attainment in the physics laboratory. Attendance of the lectures should therefore be regarded as mandatory.

### 5.3 Computer Programming and Word-processing

In the first eight weeks of the first term you will be learning elementary word-processing and computer programming (see also Sections 9, 10 and Appendix C). The classes will be conducted in the “computer cluster room” D105 which is at the far end of the long corridor at the east of the Physics Building on the first floor. The first hour of each evening will be devoted to the course on *The Treatment of Experimental Data* (in a lecture room to be assigned) so the computing part will generally begin at 7:15. The cluster room is available for your use at other times whenever the building is open provided that there is not a class taking place. There are other cluster rooms around the College which you may also use, for example in the basement of the Lewis's building on the corner of Gower Street and Gower Place and in the Science Library.

Over the first three weeks of the course the word-processing application *Microsoft Word 2002* will be learnt by following a tutorial document, each student working at their own pace with the teacher available to provide additional assistance. Your word-processing skills will be assessed towards the end of the second term or immediately after the Easter vacation when you will have prepared one of your laboratory “formal reports” using the word-processor (see Section 8).

During weeks 4 to 8 of the first term you will learn the basic elements of the Visual Basic programming language. This part of the course is divided into five *units* each composed of a small amount of learning material and a programming task to be completed. Each unit has its own deadline for completing the work (see Section 10). The teacher will run through the new material at the beginning of each period and answer any queries that arise and then the student will work on the programming task at the computer.

A good student who has some previous experience with computers will easily be able to complete the computer programming tasks during the time-tabled sessions. However many students have little or no previous experience of programming computers and for them some additional work outside the classes will be necessary. This can be done on the College computers in the cluster rooms (say, on the Tuesday or Thursday when you are not attending the computing or physics laboratory class), on a suitable computer at home, or on the department computers in the laboratories. Normally you should not enter a laboratory to use the computers when a class is in progress. However it is quite all right to attend Laboratory I on the Tuesday or Thursday evening when you are not doing your physics laboratory work to work with the computers since there is a more than adequate number of machines. You can also get advice from the laboratory supervisor whenever he is not occupied with the students of the laboratory class.

### 5.4 Physics Laboratory Staff

In the laboratory your work will be overseen by the laboratory class supervisor. Also present or on call will be the laboratory technician. The supervisor is there to provide advice and assistance to the students as well as allocating experiments and marking students' notebooks. At the start of each class the supervisor will go round to every student to ensure that they understand the experiment they are doing. Thereafter he will keep an eye on each student's progress, occasionally checking the notebook to see that the measurements are being made correctly and recorded and analysed properly. If you are uncertain about anything at any time you should immediately let the supervisor know so that he can come to your assistance at the earliest opportunity.

It is perfectly all right to contact the technician directly about any matter relating to a particular piece of apparatus or the general use of the computers, but you should not expect technical staff to be able to assist with the experiment as such or to give detailed advice on using particular computer programs.

## 5.5 The Physics Laboratory Experiments

Each of the experiments has a title and an identification code. At the start of each laboratory session be sure to write *both* – and the date – on a fresh page in your notebook before doing anything else.

Here is the list of the experiments currently available.

### INTRODUCTORY

- P0 Introductory Experiment
- P53 The Simple Pendulum
- E52 Simple Circuits
- H52 Thermal Conductivity of Asbestos
- L60 Reflection Diffraction Grating
- T50 Semiconductor Diode

### BASIC

- E50 Series and Parallel Resistors
- E51 Ohm's Law
- L18 Newton's Rings
- L52 Refraction through a Prism
- L53 Measurements with a Thin Lens
- P13 Young's Modulus of a Wire
- P52 Sliding Surfaces

### EXTENDED

- A5 Gamma-Ray Absorption
- E12 Wheatstone Bridge
- E55 Electromagnetic Induction
- H3 Ratio of the Principal Heat Capacities of Air

## 5.6 Laboratory Notebooks

During a Physics Laboratory class everything that you write down must be written directly in your laboratory notebook — nothing whatever should be written in any other notebook or on any separate sheet of paper unless specifically required for a particular experiment. This notebook must be of the correct type having alternate pages of lined paper and of millimetre or two millimetre grid graph paper. Your first notebook should be obtained from your laboratory class supervisor (at cost price): it contains some special pages inserted at the beginning which are needed for the P0 Introductory Experiment. Subsequent notebooks in either soft-backed or hard-backed versions may be purchased cheaply from the technician in Lab I. Section 7 of these notes deals with the manner in which these laboratory notebooks are to be used.

## 5.7 Deadlines and Penalties

The deadline for handing in each Unit of computer programming course-work is approximately four weeks after the material is covered in the course and is stated in each Unit handout. Work handed in late will be marked out of a maximum which declines by 10% per working day. Each Unit will only be accepted for marking when all previous Units have been handed in.

The **final deadline** for all course-work is the Thursday of the first week of the third term. This relates specifically to the physics experiment formal reports and physics experiment work in the laboratory notebooks. To be sure that your outstanding coursework is taken in for marking you should aim to hand it in well in advance of this final deadline since **no work will be accepted after this day.**



Coursework of all kinds must be handed directly and in person to *either* the technician in Lab I *or* to your laboratory course supervisor. The work will be date stamped immediately it is received. The course supervisor will be available in his room to receive course-work each evening during the first week of the third term up until 6:30 pm.

## 6 Laboratory Procedures

### 6.1 The Student's Responsibilities

*A laboratory is potentially a dangerous place:*

- Do make sure that you follow carefully any specific safety instructions issued with any experiment.
- Report anything which you consider might be dangerous to the lab technician.
- Familiarise yourself with the notes on laboratory safety (Section 2).

*College regulations prohibit smoking, eating or drinking in the laboratory.*

At your first physics laboratory class please bring with you

- this Handbook containing the lectures on *Treatment of Experimental Data*.
- a calculator of the “scientific” kind, preferably not “programmable” or “graphical”.
- £2 to pay for an experimental notebook.
- pen (blue or black ink) and transparent ruler.

At the first laboratory session all students will perform the same introductory experiment: P0. Thereafter your laboratory supervisor will allocated an experiment for you to do each week based on your progress. This is indicated on the booking chart which is located on the notice board to the left of the entrance doors. There you will find your name and a code (“Tu” or “Th”) to indicate whether you are attending on Tuesdays or Thursdays. Each column of the chart corresponds to one week of the course, headed by the dates of the corresponding Tuesday and Thursday. During the evening your supervisor will write the code number of the experiment you are to perform the following week in the relevant column opposite your name. At the end of the session check on the chart which experiment you will be doing the following week and take a copy of the *script* for that experiment from the grey metal drawers located on the right side at the same end of the laboratory (see Section 6.3). Do try to read through the script at least once before your next session.

- Do not alter the booking chart.

When you first enter the laboratory collect your notebook and make your way to the bench where your experiment is set out. This can usually be identified by the master script folder which is stood upright next to the apparatus. The supervisor or technician will direct you to the correct location if you cannot find it yourself. Experiments whose code begins with L are optics experiments and are normally located in the dark room at the far right end of the laboratory. Read through your work of the previous week to see what comments have been made by your supervisor. Try to progressively improve the standard of your work based on this feed-back.

- Do not move apparatus from one bench to another without permission.

Read through the script once more and resolve any difficulties of understanding by discussion with the supervisor. Then begin the experiment as instructed.

*Never turn on an electrical power supply or water supply or any light source until the supervisor has spoken to you.*

## **6.2 Performing the Experiments**

The experiments are designed to be completed to an adequate standard by an average student in one three-hour session. This involves:

- recording in very brief note form the procedure and apparatus used;
- obtaining and correctly recording sufficient data of a reasonable quality;
- performing an adequate analyses on the data, including the drawing of graphs where appropriate and always including a quantitative estimation of uncertainties;
- writing a “Conclusion”.

(See Section 7 for more details on keeping your notebook.)

Keep a check on the time and seek assistance if you experience any difficulties or feel you might run out of time. The supervisor is there to *help* you. You do not need to disassemble the apparatus when you have finished. At the end of the session your notebook will be collected by the supervisor. Your work will be marked during the week and the notebook returned to you at the next session.

For about the first half of the course you will usually perform a new experiment every week even if you do not manage to quite complete the one of the previous week. This is because you need to gain experience from a wide range of experiments in order to quickly build up your skills and knowledge of the techniques. In the later half of the course there will be time for you to return to the analysis of experiments that were not completed provided that you are regularly attending the class. However it is important that you always maintain the aim of acquiring an adequate quantity of data and carrying out a significant amount of analysis within one session, otherwise your work cannot be assessed and you will not learn. To simply make the measurements and then leave the laboratory in the hope of performing the analysis at some later date is a sure-fire recipe for failure. It is imperative, if you are to progress satisfactorily, that you attend and work for the full three hours of the laboratory session.

Becoming a competent experimental scientist is a very long-term process — even professionals of many years standing are learning all the time. Be aware that you will encounter something new with each experiment so that you can steadily acquire and practice experimental and analysis skills. Do not worry therefore if your notebook is returned to you covered in red corrections; only try to understand where you have made mistakes and avoid repeating the same mistakes next time. Students who do this invariably end up with a good mark at the end of the course even though they may have begun it feeling unsure of themselves.

You will be required to prepare formal reports for two of the experiments which you have completed. The experiments allocated for the formal reports will be indicated on the booking chart by yellow highlight pen. The first will be allocated when you have satisfactorily performed five experiments and the second when you have completed eight. These formal reports will be mostly prepared outside the lab using the notes already recorded in your notebook. The formal reports must be handed in by the end of the first week of the third term, however it is recommended that you complete your first formal report well before the end of term two so that you can get some feedback to help you write the second one. (See Section 8 for more information on formal reports). All notebooks and all computer programming work must also be handed in by the end of the first week of the third term for final marking and/or second marking.

### 6.3 Instruction Sheets (*Scripts*)

Instruction sheets (the “scripts”) are provided for all experiments and are obtained from the grey metal drawers near the technician's room. The scripts are intended to define the purpose of the experiment and to give guidance on the experimental procedure and the methods of analysis to be employed. They do not *necessarily* contain all the information required for successful completion of the exercise; they must not be treated as a recipe to be followed exactly and without thought. Study all of the script carefully and thoughtfully *before* starting an experiment and seek advice from your supervisor if in doubt about anything. In this way the time spent working with the apparatus will be used most efficiently. The script should be kept by you either attached to the pages of the notebook or in a separate ring file.

When you are recording your experiment in your laboratory notebook do not copy out sections of the script; there is nothing to be gained by this. Do not regard the script as an example of a laboratory notebook or formal report to be imitated. The script fulfils quite a different purpose to either of these. Rather follow the directions given in the Sections 7 and 8 which explain in detail what is required.

### 6.4 Apparatus

You will encounter apparatus of a wide range of sophistication during the course, though in the first year we have tried to keep it as basic as possible and avoid mysterious “black boxes” with complex operating procedures. Many items have identification marks and you should note these down where relevant so that if you want to repeat some measurement at a later date you can do so under the same conditions. This is more important for the item being measured than for the measuring instruments themselves.

Many experiments use apparatus that must be connected to the mains electrical supply. Sometimes this involves plugging in to a nearby wall socket but often the outlets mounted along the side of the bench must be used. These are connected by long cables to wall sockets, so ensure that the appropriate wall socket is switched on. As a safety precaution switch off all apparatus and the wall socket as soon as the experiment is completed. Do not switch on the power to an electrical circuit until it has been checked by the supervisor.

Light sources can be hazardous and some have special switch-on procedures so do not switch them on until you have spoken to the supervisor. Caution is also necessary where liquids are concerned so again do not proceed without having first consulted the supervisor.

Breakages and defects should be reported to the technician or the supervisor. If in any doubt on the use of any item of apparatus consult your supervisor. If apparatus appears to be faulty or if items are missing, inform the technician.

## 7 The Laboratory Notebook

### 7.1 The Aim

You should regard your laboratory notebook as a sort of diary in which is recorded your progress sequentially through an experiment. A laboratory record of any experiment must contain sufficient information to enable the experimenter to retrieve the work done on the experiment at a later date, either in order to write it up (e.g. as a formal report) without having to appeal to memory, or to allow someone else to repeat the same experiment to extend or confirm the observations made previously. A professional report should therefore contain a brief description of what has been done and the techniques and apparatus used and should identify any difficulties encountered together with the ways in which these difficulties have been overcome, etc. However because your time in the laboratory is very limited, you are not required to duplicate material that is already contained in the experiment script. Do however make sure that it is clear from your record which part of the experiment you are carrying out at each stage and note down anything that deviates from or is additional to what the script describes.

All numerical readings obtained must be recorded even if subsequently discovered to be erroneous and should never be erased. The deductions made from these observations, together with details of the way in which these deductions were made, should also be recorded. You should strive to acquire the attitude of a professional scientist who, at some later stage, will have to use this laboratory record to write up the results of the experiment, including possible re-analysis of the data, perhaps for publication.

The record must be entered *directly* into your notebook as the experiment proceeds. Do not keep notes and experimental data on loose pieces of paper or rely on your memory for subsequent writing up. Write down instrument readings *exactly* as they are taken from the instrument without any arithmetical manipulation. Clearly the notebook might not be as neat and tidy as you would wish. It may contain discarded data and erroneous calculations, but this will not be held against you provided such items are clearly labelled as being incorrect and the reason for identifying them as such is stated. Acquiring the ability to keep a clear and logical record of what is being done, whilst it is being done, is an essential part of a training in experimental physics. Your notebook will be used to form a judgement of your ability at experimental physics and is therefore a very important component of the course assessment.

Check points:

- Write **only** in the notebook.
- Use only black or blue ink pens, not red or green, and **never pencil**.
- Never erase anything you write down: if something proves to be incorrect draw a single line through it and make a note of where in your notebook the corrected version is to be found.
- Number the pages of the notebook.
- Start each new experiment on a fresh lined page and leave space enough for your supervisor to add comments.
- Normally the graph paper pages should not be used for text.
- Do, of course, always aim for 100% legibility.

### 7.2 The Contents

A laboratory record should contain at least the following:

- **Identification:** Experiment code (e.g. P53), title of experiment (e.g. “Simple Pendulum”), the date.

- **Objectives:** A very brief statement of the objective of each stage of the experiment and the technique you will use. In most cases the experiment will be testing a hypothesis, in which case the relevant formulae should be quoted. Theoretical derivations of formulae given in the script need not be copied out (although a reference to where the derivation may be found would be useful). Nor do you need to give a lengthy description of the experimental procedure — just give the minimum necessary to enable another physicist, unfamiliar with the particular apparatus used but with the script available, to repeat what you have done.
- **Diagrams:** Sketches or diagrams of the apparatus and/or circuit diagrams are necessary for most experiments. The drawings in the script are designed to explain the theoretical background and give an idea of the purpose of the experiment. *Your* diagrams should express what you have done and how. Elaborate draughtsmanship is not necessary.
- **Measurements:** The observations made and readings taken should be recorded in a table with appropriate headings directly in the notebook. The formatting of the recorded values (number of significant figures) should always correctly reflect the precision of the experimental measurements. If there are calculations to be made upon each observation (and there frequently are) then design your table carefully to allow recording these computed values alongside the raw measurements. Avoid ever having to copy data from one table into another: this is a frequent source of error. The heading of each column of the table should consist of the name of the quantity, its mathematical symbol, and the units used. If a sequence of repeated measurements is made (e.g. micrometer readings) they must all be tabulated — not just their mean, for example. The credibility of your uncertainty analysis can be assessed, or the data re-analysed, only if the total raw data are presented.
- **Calculation of results:** You should always carry out at least a specimen set of intermediate calculations *whilst* the readings are being taken to ensure that the data is acceptable. Units must always be stated for the raw data and for any calculated quantity: do not think to yourself “it's obvious”. Whenever you calculate a mean, always find the standard deviation and standard error at the same time and write them down. Calculate the final result as soon as you have sufficient data to do so, so that any problems can be identified as early as possible. Underline and emphasise your key results: those that directly relate to the purpose of the experiment.
- **Uncertainties:** An assessment of the statistical uncertainties of the raw data must always be made at the time of making the measurement. In the first year it is not normally required of you to consider systematic uncertainty although this will become more important in later years so it is a good idea to get a little practice at this. The raw data uncertainties should then be appropriately combined and/or propagated through all intermediate calculations to arrive at an estimate of the accuracy of the final results.

- **Graphs:** Graphs usually help to highlight and analyse results. Choose convenient scales that are easily interpolated and that allow sufficient accuracy. Where appropriate, plot the independent variable (the one directly under the control of the experimenter) as abscissa: e.g. fringe number, mass added to scale pan. Label axes with name, number and units of the quantity plotted. Mark data points clearly with for example a dot enclosed by a circle, a cross or other symbol. In the first year course there are only a few instances where it might be appropriate to indicate the uncertainties of individual measurement on the graph using error bars on each plotted point; seek advice about this if in doubt. In general, experimental points on a plot should not be directly joined up by line segments. A line appearing on a plot should represent some hypothesis that is being tested by the data, usually a single straight line that is the closest fit to the points. There are no hard and fast rules when presenting graphical data; the guiding principles should be clarity and ease of use. Graphs must be drawn on the proper pages at the relevant place in your notebook — loose pages are only acceptable when using special types of graph paper and should then be fixed into the notebook. *Always plot graphs as you proceed* — do not wait until you think that you have completed the data-taking to plot your data or results. In this way will you be able to judge as you proceed whether, for example, your intervals between data points are suitable or whether something has suddenly gone wrong.
- **Analysis:** Here the data is compared with some hypothesis. This can take various forms. If it is a simple matter of comparing a quantity calculated from a formula with a “standard” value then this can be done within the *Conclusions* section of your report. Often though you will have measured the same quantity more than once using different conditions or different methods, so you will need some calculations to illustrate how much variation there is and whether it is consistent with your estimated uncertainties. Another common situation involves analysing the relationship between two quantities as the conditions are varied progressively by the experimenter and may require a straight line or other function to be fitted to a set of points already plotted on a graph. In the first year course all graph fitting of this kind is done using straight lines drawn by eye using a ruler. Once you have achieved a good standard of using hand-drawn straight line graphs to find results and uncertainties you may with your supervisor's permission progress to employing the “least-squares fit” computer program to find gradients, intercepts and uncertainties. However you should always also draw the graph, including the straight line calculated by the program.

## The *Conclusions* Section

The following information is always placed in the *Conclusions* section at the end of your report. “Conclusions” is the accepted title for this section because it *concludes* the report, i.e. it comes last. Do not think of the term as meaning “deductions”.

**The *Conclusions* section must be written in such a manner that it makes sense when read alone, without the script or the main body of your report. Hence all physical quantities must be referred to by name, not symbol (“the refractive index of a glass prism...” not “ $n = \dots$ ”). It must be written as proper sentences, not in note form.**

- **Introductory sentence:** The experiment should be very briefly described in just one or two sentences, e.g. *The refractive index of a glass prism was measured using the minimum deviation method.*
- **Statement of main results:** This is a quantitative statement of the main results, i.e. as numerical values with uncertainties and units. Regardless of where else they may appear in your experimental record, a clear summary of the final results and their uncertainties must be given near the beginning of this section.

- **Comparison of results:** Some experiments will be measuring quantities whose values are listed in standard data tables (e.g. *Kaye and Laby*, available for consultation in the laboratory). When this is the case the standard values should be given and compared to your results. This comparison is, of course, only meaningful when you are able to compare the discrepancy between your numerical result and an “accepted value” as against your estimate of the uncertainty in your experiment. Also, you may have measured the same quantity in more than one way: again you should compare these different results in terms of discrepancies and uncertainties.
- **Commentary:** This is a critical assessment of the experiment. In the first year course, especially during the first half, you should limit your comments to a simple statement of the extent to which the results of the experiment agree with accepted values or support a hypothesis or theory — unless something radically unexpected occurred during the performance of the experiment, and even then you need only comment on this very briefly. Towards the end of the first year when you have acquired sufficient experience to make measured judgements you may like to consider including further brief comments along these lines:
  - Assumptions or approximations made.
  - Internal consistency of the readings and/or the results.
  - Comparison of the scatter of the results with the final random uncertainties assigned.
  - Unexpected features in the data and their interpretation if possible.
  - Possible sources of systematic error (this and the three above are interrelated).
  - The apparent limitations of the apparatus used and/or the script and suggestions for improvement.
  - Possibilities for improving the accuracy.

No laboratory record can be passed as satisfactory unless it has an appropriate *Conclusions* section.

### 7.3 Checking by the Supervisor

When an experiment has been completed the notebook in which it is recorded should be shown promptly to your supervisor so that any obvious errors can be corrected before the end of the session. As explained above, the notebook will then be retained for detailed marking and returned the following week.

*All laboratory notebooks MUST be handed in at the end of the second term or the beginning of the third term when the existing marks recorded in them will be reviewed and any new work marked. Work which is not handed in promptly by the start of the third term (immediately after the Easter vacation) may not be passed on to the Examiners.*

### 7.4 Assessment of Laboratory Notebooks

Each experiment recorded in your laboratory notebook will be assessed according to these criteria:

- data of sufficient quantity and quality;
- data analysis correct and complete, producing the result required;
- uncertainties estimated for raw data and correctly calculated for results;
- quality of presentation and the writing of a properly headed *Conclusions* section.

The above criteria will be approximately equally weighted, though this will vary according to the experiment. You should note that the data can only be assessed when sufficient analysis has been carried out.

## 8 Formal Reports

You will be required to write formal reports for two of the experiments which you have completed as indicated on the booking chart by yellow highlight pen. One of these should be prepared using the word-processor and it will be assessed twice: for word-processing skills and as a report.

### 8.1 Purpose

The writing of a formal report can be considered as an exercise in the preparation of a scientific paper for publication. Whereas the essential feature of the laboratory notebook is the recording of all procedures and all results in such a way that the experimenter or an informed colleague could repeat the experiment or re-analyse its results at a later date, and which gives your teachers an insight into how you go about your work, the essence of the formal report is the communication of the experimental method and its key results to a wider audience. To emphasise the point, the lab notebook is a record of what you do, compiled as you make your way through an investigation and as such, particularly when you come to do longer experiments or projects in later years, may contain things you try which do not work out, thoughts which you jot down which in retrospect lead nowhere and the like. The formal report is a digest of this from which you have excluded irrelevant material in an effort to convey the essence of your investigation clearly, fully and without ambiguity. In similar vein a scientist preparing a report may have to add information not contained in his laboratory notebook in order to make the work more comprehensible or informative to a third party. Such items might be a potted version of the theory behind the technique used or references to allied work.

### 8.2 Content

The report should be complete and self-contained and written in good formal English with neatly drawn diagrams and graphs. In general the contents should include those items listed in Section 7.2 but with more detail. In addition, descriptions of the apparatus and procedures should be given in your own words. On no account repeat the script although you might make use of it: the script is not written as a report, being more akin to a series of instructions and is an unsuitable model. The report should be divided into suitable sections, each section having a clear heading. There are no strict rules about how this should be done but a typical sequence of sections might be: *Introduction, Experimental Method, Experimental Data, Calculation of Results, Uncertainty Analysis, Conclusions*. If you start with such a structure in mind and vary it accordingly as you see that some element of your experiment does not fit into this pattern then you will soon become schooled into a habit of ordered well-structured presentation.

One further element needs to be included to form a complete report. No report or scientific paper for publication is complete without an *Abstract* which is a short paragraph following the title and preceding the body of the report. The purpose of the abstract is to inform a browser or anyone conducting a search through the literature that the paper contains something which makes it worth reading. It is important that it arrest the attention of the reader. A good abstract is written in concise language and conveys three pieces of information: the purpose of the investigation, a statement of the techniques employed and a statement of the principal results. Some examples of well written abstracts from student projects are available for your guidance.



### 8.3 Style

The report should preferably be written in the past tense, passive form (“The heater voltage was then set to about 10V...”) although in your first year it is acceptable to use the first person past tense (“I then set the heater voltage to about 10V...”). Be sure to be consistent in your style of language and also in the style of page layout. Pay attention to how you lay out tables and equations and note how Roman characters used as symbols for physical quantities are conventionally printed in italic (e.g. current *I*) in order to distinguish them from ordinary language. In the laboratory there are examples available for consultation of well written reports and you are recommended to inspect these at some time during the course.

One of the formal reports for course 1B70 must be word-processed and presented as numbered sheets stapled or otherwise bound together. In the first year course you will not be penalised significantly if you insert drawings and formulae by hand although you will gain by familiarising yourself with the relevant word-processor features as soon as possible since in later years your reports will be required to be produced entirely on the computer. All lab computers and the College computer system have the *Word* program which contains an *equation editor* for preparing mathematical formulae and an elementary drawing facility.

### 8.4 Assessment of Formal Reports

The formal report is regarded as primarily an exercise in scientific report-writing and members of staff try not to mark the notebook and formal report twice for the same thing, although some replication is probably unavoidable. You should certainly ensure that any errors in the notebook that have been pointed out in the marking are corrected before writing the formal report. The broad criteria for assessing a formal report are the following:

**QUALITY OF THE ABSTRACT:** all the elements of a good abstract should be present in concise form.

**STRUCTURAL COHERENCE:** the report should follow an ordered progression of ideas through the appropriate use of sections with informative headings. Pages, graphs, equations and tables of data should all be numbered and referred to by their numbers.

**ENGLISH EXPRESSION:** the report should be written clearly in concise grammatical English using correct syntax and punctuation.

**PRESENTATION OF INFORMATION:** the diagrams, graphs and tables should have titles and be neat and clear.

## 9 Computing

See also sections 14 and 15 and Appendix C.

### 9.1 Misuse of Laboratory Computers

The laboratory computers are there primarily to support the practical work which takes place in the laboratory but also for students to use when preparing reports or solving problems associated with other parts of the degree. It would be an act of great discourtesy to fellow students and very expensive of technical staff time to repair the situation if you interfered with the programs or the system set-up in any way. The laboratory staff and supervisors maintain vigilance over the computers to try to ensure that this does not happen. Any interference which does take place will be viewed seriously and culprits identified will be subject to College disciplinary procedures.

On a lesser level, but still causing inconvenience, are sloppy or impatient habits in using the computers. You should not for instance create files on the hard disk. You will be issued with a floppy disk for the purpose of keeping your personal files. Similarly the response time of the computers, in particular in printout cycles, can be frustrating. Usually four computers share one printer. In these circumstances it is of no use becoming impatient if your job does not print out as quickly as you would like. In particular, sending repeated requests to print will merely cause the system to run even more slowly or even hang up completely.

### 9.2 Necessary Materials

Both the word-processing and the programming parts of the course make reference to information files written on a “floppy disk” which will be provided for you. Bring this disk to every class and keep it safe: do not carry it loose in a bag or pocket; special plastic wallets can be purchased from the College shop or Union shop. For the word-processing element you will be provided with a tutorial document and you will also need the supplementary notes given in Section 14 of this Handbook. For the computer programming you will need the introductory notes and the individual notes for each of the five problem units given in Section 16. Be sure to bring this Handbook to every class. You will also be provided with a CD-ROM containing a version of the Visual Basic application so that you can work on your computer programming tasks when away from the College.

### 9.3 Books

Computing books are generally rather expensive. For word-processing you probably will not need to have a book but if you decide you do then two very suitable and inexpensive ones are *Microsoft Word 2002 explained* by Oliver and Kantaris, published by babinibooks, currently priced £6-99 or *Word 2002 in Easy Steps* by Scott Basham, published by Computer Step, currently priced £9-99. There is a vast choice of more expensive books if you wish to have more detailed coverage, but beware: many expensive flashy looking books lack the information that you need. Be certain that any book you buy relates to *Word 2002 for Windows* or *Word XP for Windows* and that it has a reference section and a good index that lists the features you need, such as equation editing, subscripts and toolbar customising.

For the computer programming part of the course it is more necessary to have a reference book on the Visual Basic programming language. Unfortunately nearly all books on Visual Basic focus on producing fancy looking programs with elaborate user interfaces rather than quickly writing programs to perform simple computations. However a moderately priced book that is quite adequate is *Visual Basic 6 in Easy Steps* by Tim Anderson (ISBN 1840780290), published by Computer Step, currently priced at £9.99. Make sure any book that you buy is for Visual Basic version 5 or version 6.

A copy of the above books will be available in the laboratories for consultation.

## 9.4 The Computers

The desk-top personal computers used on this course are IBM PCs or “compatibles” running the *Windows NT* or *98* operating system.

In room D105 are about 30 computers which we will be using for class work. They have been reserved for our use on Thursday evenings (a block booking). These computers are connected through the College network just like those in other cluster rooms. To use them you must first *log in*, which requires you to have been issued with a unique personal user identification code (Username) and a password (see *Registration* below). You can use these or other networked computers in the College at any other time. There is a booking system and it is advisable to book a computer in advance since a user who has booked has priority over a casual user.

In the Physics Department laboratories on the first and second floors are various computers which are available for your use at any time during College teaching hours provided that there is not a practical class running at the time. These are the same computers that you will eventually be using to analyse the data that you acquire in your physics practical class.

## 9.5 Registration

You should register with the College as a computer user as soon as possible. This is necessary before you can use the networked computers. You will then have your own storage area on the network disks (see *Disks* below) and your own “e-mail” account which will enable you to send and receive messages using the computers. If you enrolled on the course a reasonable time before the start of term your computer account will have been arranged by the Course Tutor.

If there are any problems with your registration you should go to the Information Systems Division (ISD) Help Desk in the basement of the Lewis's building on the corner of Gower Street and Gower Place. They close at 6:30. If you can arrive 30 minutes before your lectures start on one evening you could call in there on your way to the Physics Department. Otherwise you will need to come in on an evening when you do not have lectures. Be sure to inform them that you are an “intercollegiate student” following the Physics BSc degree at UCL.

## 9.6 Using the College PCs

To book a networked computer in advance so as to reserve it for your use go to the special booking system computer in any cluster room. Log in (as described below) and then follow the instructions.

To use a computer in D105 or any other cluster room you must “log in” to that computer at the beginning of each session. If the previous user of the computer has left it in the correct state you should find it with the *WTS Start Panel* displayed.

To log in click on *Cluster WTS* when a dialogue box will appear requesting your User ID and Password. Note that as you type your password you will not see the characters appear on the screen; it is therefore important to do it slowly and carefully. After a short delay you will see a page of messages about the computer system with the instruction to “Click to Continue”. Click inside this area when, after another short delay, you will finally find yourself with the *Windows NT* desktop screen. There you should find icons for the *Word* program and other applications. For any applications that do not have a desktop icon (e.g. Visual Basic) click on the Start button at bottom left and navigate through the menus Programs > Software >... until you find the program you are looking for.

If you are using a floppy disk insert it after you have logged in. Remember to take it out again when you leave (see *Disks* below).

When you have finished using the computer, exit from *Word* or *Visual Basic* and then from *Windows NT* using the Start > Log Off button. Do not switch the computer off.

**Note.** To switch on a computer you generally have to switch on two things: the monitor or display unit and the system unit (box with slots for disks).

## 9.7 Using the Physics Department PCs

Here you do not log in. To do word-processing double-click on the *Word* icon. To do computer programming double-click on the Visual Basic icon if there is one or otherwise insert your CD-ROM, wait a few seconds, then double-click on the *My Computer* icon. Double-click the CD-ROM disk drive (usually D), then the folder called *Visual Basic 5 CCE*, and finally and double click the file called “vb5cce” or “vb5cce.exe”.

**Note.** You cannot access the network from the laboratory computers.

## 9.8 Using your own computer

If you have access to a suitable computer outside the College that has the appropriate software installed on it then of course you may use that machine to do your course-work. The word-processing must be done using *Microsoft Word* and the computer programming must be done using Visual Basic. This is because this course-work is assessed as part of the examination for your degree and the same standards and criteria must be applied to all students. (See also the sections of this Handbook on word-processing and computer programming).

You must not copy any programs from the College or Department computers since they are commercial software and are therefore covered by copyright law. Microsoft applications may be bought through the College at a considerable discount. Contact Information Systems Division for more details.

## 9.9 Disks

When you finish using a program like *Word* or Visual Basic all the work that you have done is lost unless you explicitly ask the program to **save** it onto a *magnetic storage disk* of some sort. The computers each have their own internal disk called the “hard disk” (designated as “C:”) but on this course you should use your own personal “floppy disk” (called “A:”) which is inserted into a slot on the front of the computer and removed again to be taken away with you when you have finished your work. The information or data that you have created is recorded magnetically onto these disks in a manner similar to sound or video recording onto magnetic tape.

You will be provided with a floppy disk free of charge for use on this course. This is where the work you will do using the computer will be stored. It has some files that you will need already recorded onto it so it is very important that you look after this disk and bring it to every computing class.

#### CARE OF FLOPPY DISKS

Floppy disks look solid but the plastic case is very thin and they are in fact quite fragile. Never write on the label using a ball-point pen; to do so will certainly damage the disk inside. The metal shutter is also prone to damage and if it gets deformed even slightly the disk may become stuck inside the computer. Always carry a disk around in one of the specially made plastic wallets which can be obtained from the College Shop.

#### INSERTING THE DISK

The floppy disk has a metal sliding cover on one edge: this is the edge that goes first into the slot at the front of the computer. If the slot is horizontal then the disk should be inserted with its label and brand name uppermost. If the slot is vertical then usually the label and brand name should face away from the eject button. Be sure to push the disk in far enough until you sense that it has been seized by the mechanism. To eject the disk from the computer you press the button found close to the slot.

**DO NOT HAVE THE DISK INSERTED BEFORE SWITCHING ON OR RESETTNG THE COMPUTER:** the computer will not start up properly if you do. Wait until you see the *Windows Desktop* screen.

**CAUTION:** The *Word* and Visual Basic programs, the DOS and *Windows* operating systems, and the many other programs and data files necessary for the operation of the computer are all stored on the computer's internal "hard disk" (disk drive C). Please do not try to access this disk in any way. In particular, do not alter any of the system configuration files, do not copy any programs from the hard disk, and do not place any of your own files or programs on the disk. If you do you will cause considerable trouble and inconvenience for staff and other users of the computers.

One floppy disk will be entirely adequate for everything you have to do on this course but if you want to have more than one you can buy them: they cost less than 50p each. Make sure you buy disks that are ready formatted for DOS or for Windows (i.e. for IBM compatible computers). You can buy these disks from the UCL Union Shop in the basement of the Bloomsbury Theatre in Gordon Street or the ULU Shop in Malet Street (opposite Waterstones book shop).

On the networked computers in the College cluster rooms you can use the part of the network disks allocated to you (called "R:") once you have registered with the computing centre and have your own personal Username. However it is advisable to also save or copy your work to your floppy disk in case you wish to continue your work on a non-network computer at a later date.

## 9.10 Printing

Printing your work on paper is essential as well as saving it onto disk. This is so that you can study it more carefully than you can on the computer screen and also in order to hand it in for marking (in this Year One course you do not hand in your floppy disk for marking). The details of how to print your work are covered in the sections of this Handbook on word-processing and computer programming but you should also know that different computers have different printers attached to them.

In D105 all the computers are connected to one high quality laser printer at the side of the room. The printer takes material from the various computers in batches, so if someone else is printing, your work may take a while to come through. Be careful to check that you take away all your work and none of anybody else's. Anything you print like this should have your name on it so that you can easily identify it as yours.

In the Physics Teaching Laboratories the computers on each table share the use of one or two ink jet printers. These are controlled by an automatic routing switch. Again, make sure that anything you print has your name on it and be careful not to accidentally take someone else's material.

## 9.11 Assessment of Word-processing

The word-processed formal report(s) will be assessed twice: first as a physics experiment formal report and then as a piece of word-processing. The criteria used for assessing your word-processing skills in year one are:

- sensible and consistent layout of pages and paragraphs;
- clear formatting of headings and sub-headings;
- page numbering;
- appropriate use of tables for data;
- use of proper special characters, such as  $\pm$ ,  $-$ ,  $\times$  (not  $+/-$ ,  $-$ ,  $x$ , and not written in by hand);
- appropriate use of italics or Greek characters for variables;
- appropriate use of sub-scripts and super-scripts;
- correct spelling;
- correct spacing of punctuation.

Small bonus marks will also be awarded for the use of the equation editor or computer-generated drawings.

## 9.12 Assessment of Computer Programming

Except where the documentation for the Unit states otherwise (Unit 1) the computer programming tasks will be assessed according to these criteria:

- appropriate explanatory comments through the use of Rem statements;
- appropriate instructions for the user through the use of Print statements;
- a correctly functioning program;
- logical and simple program structure;
- appropriate choice of variable names;
- well-formatted output of results to screen and/or printer;
- thorough testing of the program with sufficient samples of output.

# 10 Course Completion and Assessment

IMPORTANT:

All returned course-work for this module should be kept safely by you and be made available for re-submission for second marking in the third term immediately following the Easter vacation. **Work that is not re-submitted at this time cannot be counted.**

Course assessment will be based on marks from all components listed in Section 4 of these notes. For the Laboratory Physics a mark will be formed from the best 10 of the experiments which the student has performed. It is therefore advantageous for students to complete more than this number of experiments. Also two formal reports must have been submitted on time, at least one of which must be word-processed.

If no assessable work is received for any significant component, irrespective of the marks gained for the other components, the course will be deemed to be *incomplete*. Until work for the missing component has been handed in no course mark can be awarded.

Illness and other reasonable causes for non-achievement of the required work may be taken into account in the final assessment but students who believe they have grounds for special treatment must inform the relevant Tutor and the Course Supervisor as soon as possible.

It is very important that you observe the deadlines which will be strictly adhered to. Work which is handed in late may well be refused.

The total mark for the course is obtained from the sum of the marks for each element — 10 experiments, 2 formal reports, 5 programming units, 1 word-processing task — each of which is equally weighted.

Any student who feels that they are falling behind in their lab work should discuss the problem with the supervisor. Do not be tempted in such circumstances to copy results or analysis from other students. Such an offence is easily spotted and students are warned that they risk losing all credit for the work in question; this applies both to the student who copies and to the one who has provided the material for copying. Serious cases will be referred to the College Tutor to Students for action under the College disciplinary procedures.

## **11 Student Evaluation of the Course**

As in other courses, towards the end of the course you will be asked to fill in a “student assessment” form by which you may give your opinion of the course by assessing key indicators. The forms will be handed out in a lab period to be completed in the lab and passed on to a student volunteer who will collate the replies.

You will also find in the laboratory on the supervisor's table a supply of experiment assessment forms. These may be filled in on a voluntary basis if you wish to express any opinions about a particular experiment. The completed form should be posted into the box provided which is located on the supervisor's table. Both kinds of feedback are invaluable to the Department in trying to make sure that its laboratory courses run well and meet your needs and we ask for your cooperation in both exercises.

## **12 Documentation Required for the Treatment of Experimental Data and the Computing Components of the Course**

The following documents are required for the lectures on *Treatment of Experimental Data*, the word-processing course, and the computer programming course. The notes for the *Treatment of Experimental Data* lectures include practice exercises which will be done in the class (this is not part of the course assessment). For word-processing you will need to refer to both the separate Tutorial document and the supplementary guide given here.

For computer programming there is an introductory document plus five programming Units, each associated with a particular programming problem. Each Unit provides the essential new information required for that task as well as describing what the program should do. In Unit 1 you are not actually writing a complete program, just trying out small segments, and the Unit tells you precisely what to do. In Unit 2 you progress to writing a simple complete program and to help you with this an abbreviated version of the program (“example program”) is provided on the floppy disk as a guide. From Unit 3 onwards you will write each program from scratch. Note that Unit 5 actually involves the writing of two separate programs. Be sure to read through all of each Unit before you begin; do not treat the document as a step-by-step set of instructions.



## 13 Treatment of Experimental Data

### 13.1 Introduction

#### TWO TECHNIQUES

No two physics experiments require precisely the same procedure for analysing the results. However the two techniques that you will be using in your level-1 practical course form the basis of virtually all the others.

#### THE THEORETICAL FORMULA

To obtain the result for an experiment we invariably have to use a formula which relates to some **theory** about the phenomenon that we are investigating. It is the fact that the result relates to some theory that makes what we are doing a scientific experiment rather than simply a measurement (which is what engineers do).

#### AN EXAMPLE

We can illustrate this using a particular example which is the first proper experiment that you will be carrying out in the laboratory: **to find the acceleration due to gravity using a simple pendulum.**

A *simple pendulum* is made from a small weight hung on the end of a long thread. The pendulum is made to swing to and fro. This kind of motion is called *oscillation*. The time for a single complete swing of the pendulum is called the *period*. We can use Newton's laws of motion to find a theoretical relationship between the period,  $T$ , the length of the pendulum,  $L$ , and the acceleration due to gravity,  $g$ . Note that the value of the mass of the weight does not enter into this relationship so it is not essential to measure it.

$$T = 2\pi \sqrt{\frac{L}{g}}$$

#### REARRANGING A FORMULA

There are three unknowns in this formula, so if we measure two of them,  $T$  and  $L$ , we can calculate the third,  $g$ . To make this calculation more straightforward we first *rearrange the formula* so as to make  $g$  the subject of the equation:

$$g = 4\pi^2 \left( \frac{L}{T^2} \right)$$

Rearranging a formula like this, so that the quantity we wish to find is expressed directly in terms of the quantities we are able to measure, is a very important skill in experimental physics.

[Technical note: According to Newton's laws, the formula quoted above is only true when the size of the weight is very small compared to the length of the thread and when the amplitude of the oscillations is small.]

#### WHAT MAKES A SCIENTIFIC EXPERIMENT?

Once we have constructed a suitable pendulum, the problem of finding  $g$  might seem to be very easy: use a ruler to measure  $L$ , use a clock to measure  $T$ , and then calculate  $g$ . However there are three **serious inadequacies** with this approach:

- 1 When doing science we try not to take anything as true without **verifying** it. We should not simply believe the formula: we want our experiment to demonstrate that it is correct, or incorrect.
- 2 We must allow that we might have made a serious **error** in one of our measurements, or in our arithmetic. We need to check that this is not so.

- 3 The **accuracy** of our measurements is not perfect so the value of  $g$  that we calculate will also have an imperfect accuracy. The result is of no use to anyone unless we can provide some estimate of its accuracy. Related to this is the possibility that the accuracy is not as good as we wish it to be and we require some way of improving the accuracy.

These requirements can be met by some form of **repeating the experiment** several times. Here we come to the two basic approaches.

#### THE TWO METHODS

##### A *Averaging Method*

Repeat the measurement of each quantity several times ( $T$  and  $L$ ) so as to show we are consistent and to estimate its accuracy. Take an average value for each quantity we measure and use these averages to calculate the result ( $g$ ).

##### B *Graphical Method*

Adjust the apparatus in some way so that the next set of observations will be different. Then adjust the apparatus again in the same way and measure again. Then adjust and measure again, and so on. In the case of our example we could obviously adjust the length of the thread between each set of measurements. Where there are only two quantities being measured as here ( $T$  and  $L$ ), we can then plot a graph to show how one of them varies in relationship to the other. The shape of this graph could then be used to find the result we want ( $g$ ).

#### WHICH METHOD IS BETTER?

Method (A) satisfies requirements (2) and (3) but it does not in any way confirm that the formula we are using to find our result is correct or appropriate. In using this approach we are therefore not really doing an experiment, just a more sophisticated measurement.

Method (B) satisfies all three requirements since the shape of the graph we should obtain is predicted by the formula. If the experimental graph does not conform to this theoretical shape then we know that the theory is wrong. This would then be a truly scientific experiment of interest to other scientists.

Quite obviously method (B)—the graphical method—is much to be preferred, but equally obviously it is very laborious compared to (A)—the averaging method.

If we had an unlimited amount of time to perform the Simple Pendulum experiment we would measure  $L$  and  $T$  just once each time we adjusted the length of the thread, doing this 10 to 100 times altogether.

#### A COMPROMISE

Usually we do not have unlimited time; certainly not on a part-time degree course. So we sometimes use a combination of methods (A) and (B):

We do measure each quantity several times and take an average for each configuration of the apparatus, but we *also* change the apparatus a few times and *plot our graph using the averages* of the measurements for each configuration.

We might for instance choose to configure the pendulum with just 5 different lengths and for each length measure  $T$  and  $L$  10 times and find the averages. Then we would have a graph with five points on it which would probably be sufficient to show that the shape is or is not the same as that predicted by the theory. The usefulness of this result would be nearly as good as that obtained from 50 completely independent measurements using 50 different lengths which would give a graph with 50 points on it.

## 13.2 The Averaging Method

If we measured some quantity very carefully several times we would not expect to get precisely the same value each time, even if we are using the same instrument and keeping the conditions as nearly constant as possible. There is always some error which is beyond our control. This error is random and is usually called *statistical error*.

If we make our measurements **coarsely**, however, then this statistical measurement error could be masked or overwhelmed by the *rounding error*, which means the error we introduce simply by not quoting the result as a sufficiently precise number. You must always make your measurements as carefully as you can and record the result as precisely as possible. Do not guess the accuracy and then artificially reduce the number of digits you use to record the result according to your guess: you can never recover this lost precision but it is always possible to throw away superfluous digits later on when you are sure what the accuracy really is.

### THE WRONG WAY TO DO IT

For example, say we have a brass cylinder whose length has been measured very accurately to be 38.12 mm. We pass the cylinder round to everyone in the class and each person uses a metre ruler to measure the length four times. However we all measure the length relative to the zero mark on the ruler and we quote the value only to the nearest centimetre (one centimetre is about the width of a finger). It is very likely that we will all say that every time we measured the cylinder we obtained the value 4 cm (i.e. 40 mm).

Does the fact that everyone got precisely the same value for every measurement mean that the length of the cylinder is precisely 40 mm? Clearly, NO! The consistency of our results is purely due to the rounding error introduced by our very poor experimental technique. *DO NOT DO THIS IN THE LABORATORY!*

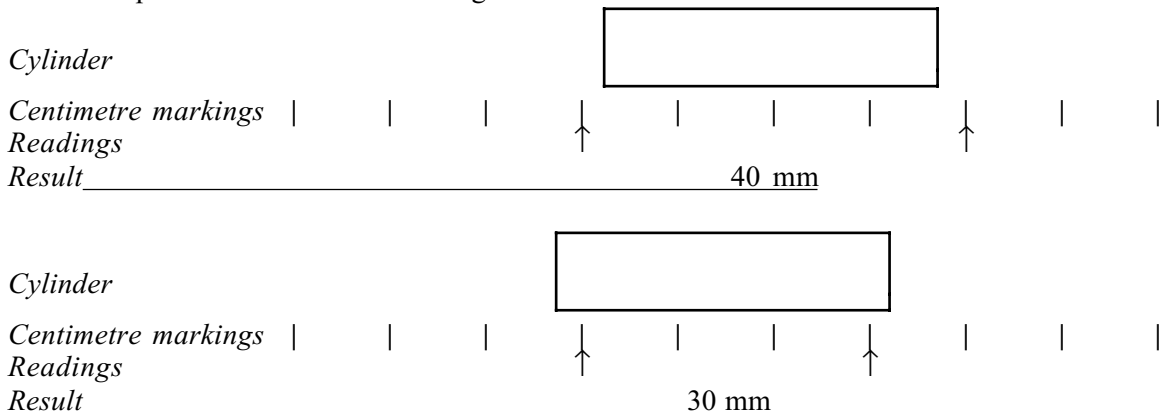
Notice that there are actually *two* factors that contribute to this rounding error:

- we took readings from the ruler at the centimetre markings, ignoring the millimetre marks;
- we **placed** one end of the cylinder against the zero mark of the ruler and only **recorded** the reading for the other end.

### FIRST IMPROVEMENT: RANDOMISE THE READINGS

We could obtain a much better result if we placed the cylinder **randomly** against the ruler and recorded the readings for **both ends**, still only taking the nearest centimetre mark. We would then use the difference between these two readings to obtain the length of the cylinder.

Most of the time we would still come up with a length of 4 cm (40 mm) but occasionally we would get a value of 3 cm (30 mm). This would depend on just how the cylinder lay in relationship to the centimetre markings:



A simple statistical calculation shows that we can expect to get the value 40 mm in 81% of cases and the value 30 mm in 19% of cases. If we made a large enough number of measurements the *average* of all our results should come very close to the “true” value of 38.12 mm, although it would require more than a million measurements to give an average that was correct to this number of decimal places!

#### AVOID OBSERVER BIAS

Here we have discovered the counter-intuitive result that seeming to be “less careful” about our measurements has improved the accuracy provided we repeated the procedure many times. When we were doing it all wrong – by carefully placing one end of the cylinder against the zero mark of the ruler – we were introducing an artificial ingredient to the measurement: in effect we were attempting to *set* the reading at one end to be a particular chosen value (0). This is an example of *observer bias*. We should always attempt to make our measurements as independent of ourselves as possible and as nearly as possible dependent purely on the measuring instrument.

The first golden rule is

**always READ the instrument,  
never *adjust* things to give a *predetermined* value.**

#### SECOND IMPROVEMENT: REDUCE ROUNDING ERROR TO THE MINIMUM

Repeating and finding the average improves the accuracy, but we can also make a big improvement by making each individual measurement more accurate, by reducing the rounding error.

If each of us recorded the locations of the two ends of the cylinder against the ruler by finding the nearest millimetre mark instead of the nearest centimetre mark this would obviously lead to a big improvement in accuracy and we would need to take fewer measurements now for the average value to achieve the desired accuracy.

We might now obtain a series of values something like this:

37, 38, 38, 39, 37, 38, 39, 38, 38, 40, 38, ...

There would be more values of 38 mm than any other, a few of 39 mm, fewer of 37 mm, and fewer still of 40 mm.

Can we do even better? Yes, but only by improving the apparatus a little. If each of us is provided with a magnifier and a lamp we would be able to estimate the location of the ends of the cylinder to better than one millimetre. We could estimate the values to a precision of **one tenth of a millimetre**, even though the marks on the ruler occur at only millimetre intervals.

This would be the best we can do to maximise the accuracy of each measurement short of discarding the rulers and using some much more sophisticated instrument such as a travelling microscope.

The second golden rule is

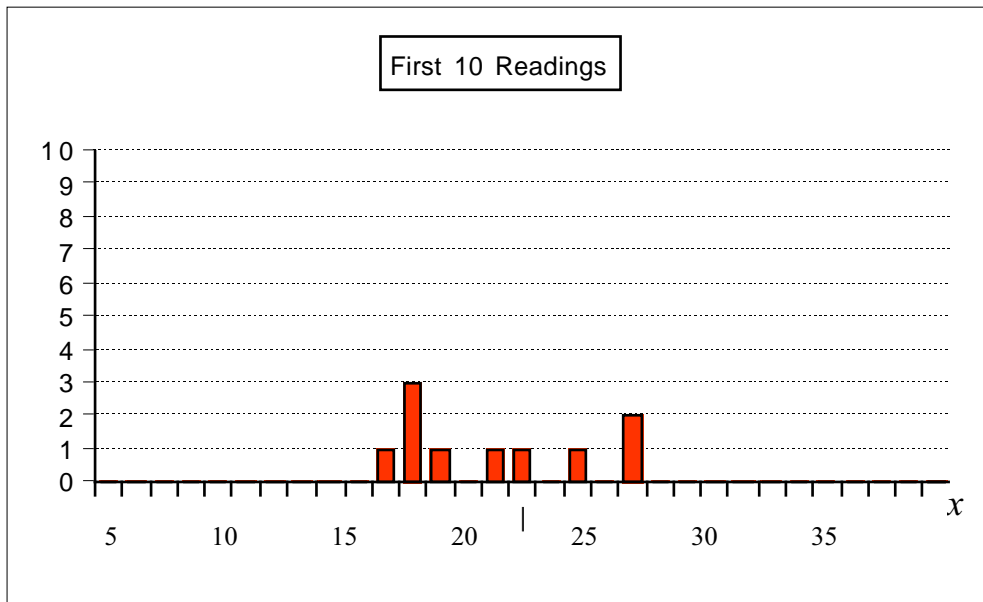
**always read the instrument as PRECISELY as you can,  
to better than its scale markings if possible.**

#### THE HISTOGRAM OF RESULTS

We can represent how the results of repeated measurements are **distributed** using a type of graph called a *histogram*.

The values along the horizontal axis represent the values of the quantity that is being measured. The graph is divided into vertical columns (called *bins*) with one column for each possible measurement value. So if we are measuring to a precision of 0.1 mm there would be a column every 0.1 mm. The height of each column corresponds to the number of times we have obtained that particular result (called its *frequency* in statistics).

For example, say we are measuring some quantity  $x$  whose true value is 20.00 mm and we are recording our data to a precision of 1 mm. Then after we have made 10 measurements our histogram might look something like this:



$$\text{mean} = 18.90 \quad \sigma = 3.45$$

Check that the columns do add up to 10. The mean value of the 10 readings has been calculated and found to be 18.90. Notice that this is close to, but not equal to, the true value, 20.00. The formula for the mean can be expressed like this:

$$\text{mean of } N \text{ values } x_i \equiv \bar{x}_N = \frac{1}{N} \sum_i x_i = \frac{\sum_r f_r x_r}{\sum_r f_r}$$

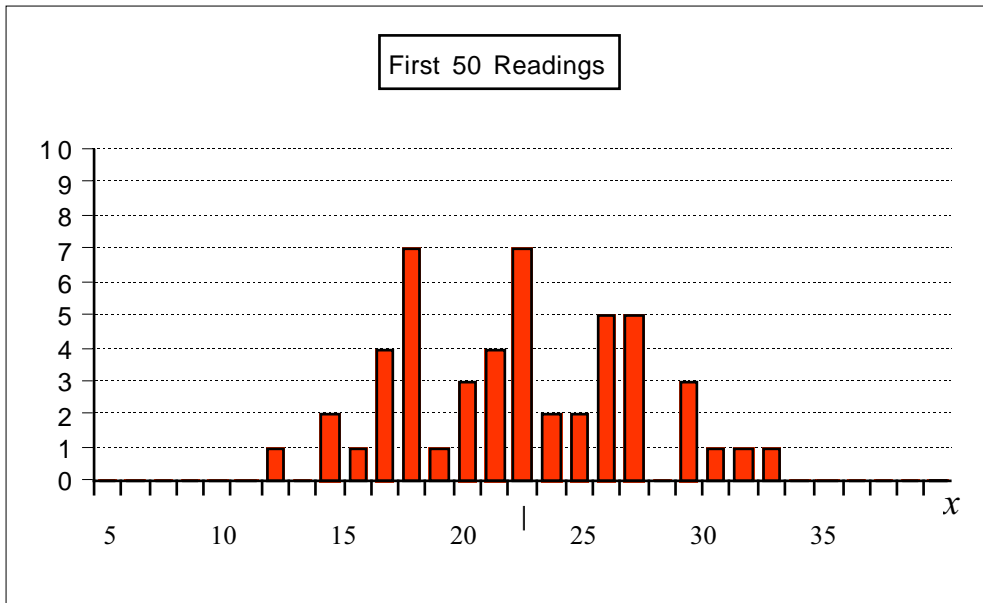
where the  $x_i$  are all of the measured values,  $i=1,N$ , and the  $x_r$  are the different values obtained for  $x$ , each with a *frequency*  $f_r$ . Here the subscript  $r$  labels each bin of the histogram. So for the above histogram, the 12<sup>th</sup> bin, which is the second of the occupied bins, has  $x_{12} = 16$  and  $f_{12} = 3$ . Obviously the sum of all of the frequencies must be equal to the total number of values:

$$N = \sum_r f_r$$

The symbol  $\sigma$  stands for the *standard deviation* of the distribution and it is the usual measure of how **wide** the distribution is. We will come to its exact mathematical definition later.

#### DEPENDENCE OF THE MEAN AND STANDARD DEVIATION ON THE NUMBER OF MEASUREMENTS

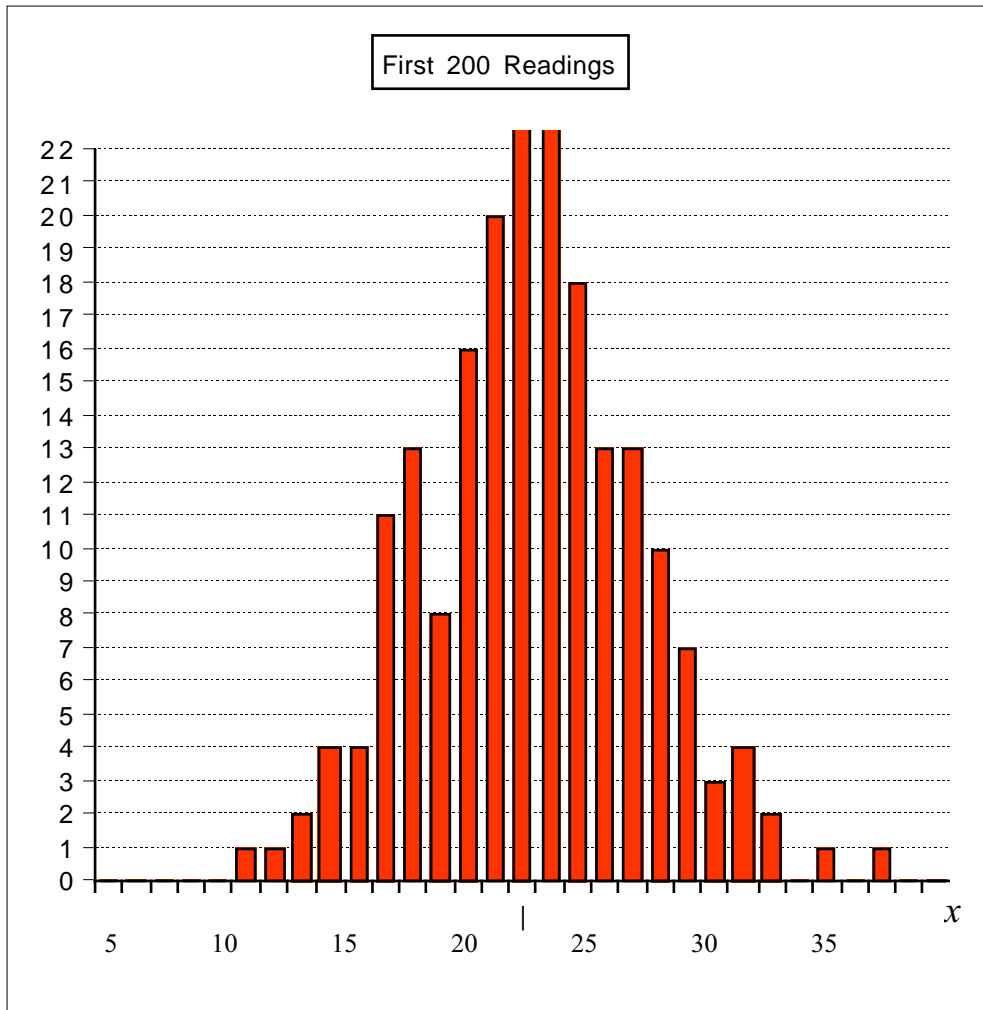
If we keep on measuring and entering the results into the histogram we might obtain the following by the time we had made 50 measurements:



$mean = 19.86 \quad \sigma = 4.26$

Notice that now the **mean is closer to the true value**. The standard deviation, however, has not altered much.

Continuing up to 200 measurements, we find:



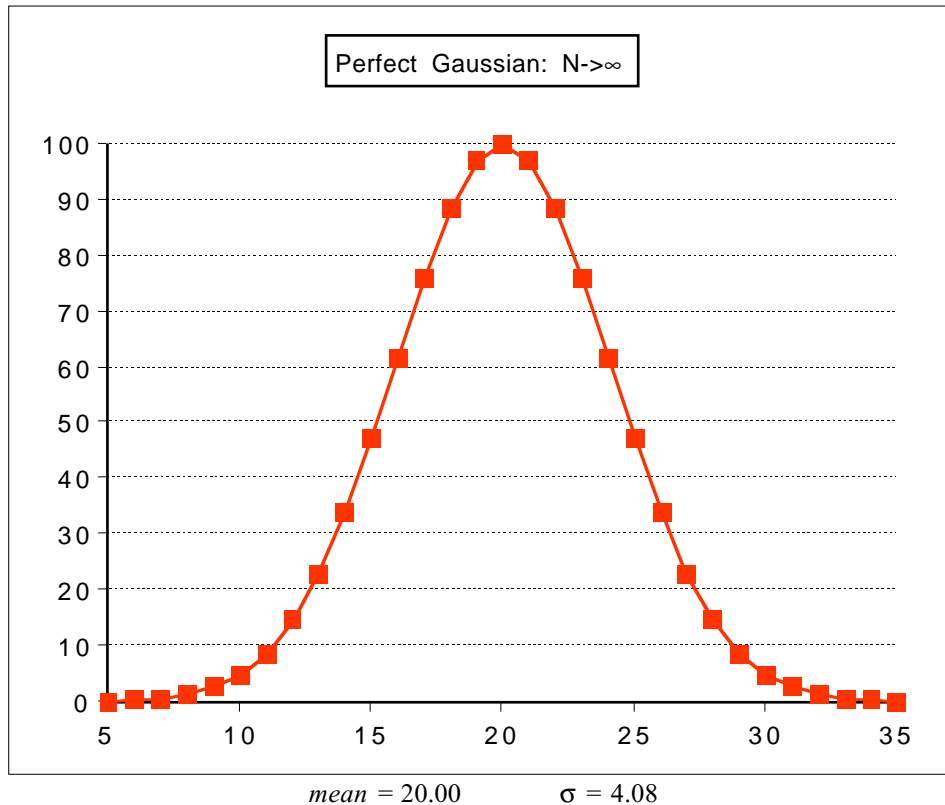
$$mean = 20.33 \quad \sigma = 3.91$$

The mean is again close to the true value, and the standard deviation has not changed much.

#### THE LIMIT AS THE NUMBER OF MEASUREMENTS BECOMES VERY LARGE

As we go on making more measurements we find that the **mean** fluctuates up and down but the fluctuations get smaller and it shows a trend of getting **closer and closer to the true value**. The standard deviation also fluctuates up and down with the fluctuations getting smaller and it also approaches some limit.

We also see that the **shape** of our distribution gets more regular as we increase the number of entries and in fact is approaching the smooth function called the *gaussian curve*:



[Here, the numbers on the vertical scale are included for convenience, but of course they do not correspond to the actual frequencies since  $N$  is approaching infinity.]

#### SHAPE OF THE DISTRIBUTION

You might be expecting that the shape of the distribution of values would depend on the technique used to make the measurements: the type of measuring instrument and the skill of the experimenter in using it. Surprisingly, regardless of such factors, **the shape always approaches this universal curve, the gaussian function, provided each measurement is statistically independent of the previous ones.**

#### A FEW MEASUREMENTS CAN BE ENOUGH TO ESTIMATE THE TRUE VALUE AND THE UNCERTAINTY

Notice that when the number of readings is small, even though the shape is hardly recognisable as a gaussian, the values of the mean and the standard deviation quite soon approach the values corresponding to the gaussian limit. We therefore say that

**the mean is a good estimate of the true value of the quantity being measured (i.e. the mean of the limiting distribution),**

and

**the standard deviation is a good measure of the width of the limiting distribution.**



Let us look at those first 10 measurements in more detail. We make a table of the values measured and how far they differ from the true value:

measurement value	difference from 20
19	-1
16	-4
24	+4
16	-4
15	-5
22	+2
16	-4
24	+4
20	0
17	-3

Notice that the magnitudes of the differences from the mean are typically similar to the standard deviation of the whole distribution, whether we take the value given by just these 10 readings (3.45) or the limiting value (4.08). This is the physical significance of the standard deviation:

**the standard deviation is an estimate of the typical random error in each of the measurements in a distribution.**

An estimate of the error in a quantity is called its *uncertainty*. So we see that repeated measurement allows us to estimate the uncertainty of each reading: it is just the standard deviation of the distribution of values.

#### THE MEAN

By repeating the measurement we get more than just an estimate of the uncertainty of each reading: we can find an estimate of the true value of the quantity being measured which is **more accurate than any of the individual readings**: the mean of the distribution.

So what is the uncertainty of this mean? Remember that there is no point in using the mean as the declared result of our experiment unless we are able to estimate its accuracy, or in other words, its uncertainty.

The uncertainty in the mean of a distribution is called the *standard error* of the mean, and there is a very simple formula that relates it to the standard deviation of that distribution:

standard error of mean of  $N$  values of  $x$ ,  $\sigma_{\bar{x}} = \frac{\sigma_x}{\sqrt{N}}$

Notice the rather clumsy notation:  $\sigma_{\bar{x}}$  for standard error to distinguish it from the standard deviation,  $\sigma_x$ . I will usually refer to the standard error as simply “s.e.” and the standard deviation as just “ $\sigma$ ” or “s.d.”.

**the standard error is an estimate of the random error in the mean of the measurements in a distribution.**

## STANDARD DEVIATION, $\sigma$

It's now time to see how the standard deviation of a distribution is calculated. This is the formula:

$$\sigma = \sqrt{\frac{\sum_{i=1}^N (x_i - \bar{x})^2}{N-1}}$$

where  $\bar{x}$  is the mean of the distribution.

[Technical note: In books on statistics the formula is usually slightly different. The definition of the standard deviation involves the true value,  $X$ , in the numerator, instead of the mean,  $\bar{x}$ , and in the denominator there is  $N$  instead of  $N-1$ . In practical science we never actually know the “true” value of a quantity so we have to make do with our best estimate, which is the mean  $\bar{x}$ . The substitution of  $N-1$  for  $N$  in the denominator is a way of compensating for the fact that we are using  $\bar{x}$  instead of  $X$ .]

This formula looks a bit daunting at first but if we look at an example you will see that it makes sense.

Extend the table of our 10 measured values. First we need to know the mean, which is just the sum of all the readings divided by 10:

$$\bar{x} = 18.90$$

Now make two additional columns, one for  $(x - \bar{x})$ , and another for  $(x - \bar{x})^2$ :

	$x$	$x - \bar{x}$	$(x - \bar{x})^2$
	19	0.10	0.01
	16	-2.90	8.41
	24	5.10	26.01
	16	-2.90	8.41
	15	-3.90	15.21
	22	3.10	9.61
	16	-2.90	8.41
	24	5.10	26.01
	20	1.10	1.21
	17	-1.90	3.61
SUM:	189	SUM:	106.90
MEAN ( $\div N$ ):	18.90	$\div(N-1)$ :	11.87
		$\sqrt{\phantom{x}}$ :	3.446

Notice that now we are calculating using 2 or 3 decimal places even though the original data values are rounded to whole numbers. You should always use two more significant figures than the original data when making calculations so as to avoid accidentally introducing rounding errors.

## USING A CALCULATOR OR A COMPUTER

Because the process of finding the mean and standard deviation is such a common requirement in science, all hand-held calculators of the “scientific” kind have a means of doing this calculation automatically. You just type in your data values, press a couple of buttons, and the calculator finds the mean and standard deviation for you. A scientific calculator is therefore a vital piece of equipment for practical physics.

Note that many calculators have two buttons for finding the standard deviation. One uses the mathematical definition as its formula (but assuming that  $\bar{x}$  is the true value), the other uses the practical formula given above. The latter is sometimes known as the *sample standard deviation*. If in doubt, try both, and use the one that gives the larger number.

In Unit 4 of the Computer Programming part of this course you will create a program which contains nearly all that is required to find standard deviations. You might like to add a few more lines to allow it to do this and also to find the standard error (which most calculators do not do automatically).

## 13.3 The Graphical Method

The most straightforward way to experimentally test a theory is to investigate how one quantity changes as we vary another, where this relationship is predicted by the theory. In the case of the simple pendulum, Newton's Laws give us the relationship

$$T = 2\pi\sqrt{\frac{L}{g}}$$

If we were to measure the period of the pendulum,  $T$ , for various lengths  $L$ , and plot these values on a graph, then the points should lie on a curve corresponding to this equation. The value of the constant  $g$  which gives a curve with the best match to the data points will then be our result for the measurement of the acceleration due to gravity.

There are two obvious questions which arise with this method:

- how do we quickly find the value of  $g$  which gives the best fit?
- how do we estimate the uncertainty in this value of  $g$ ?

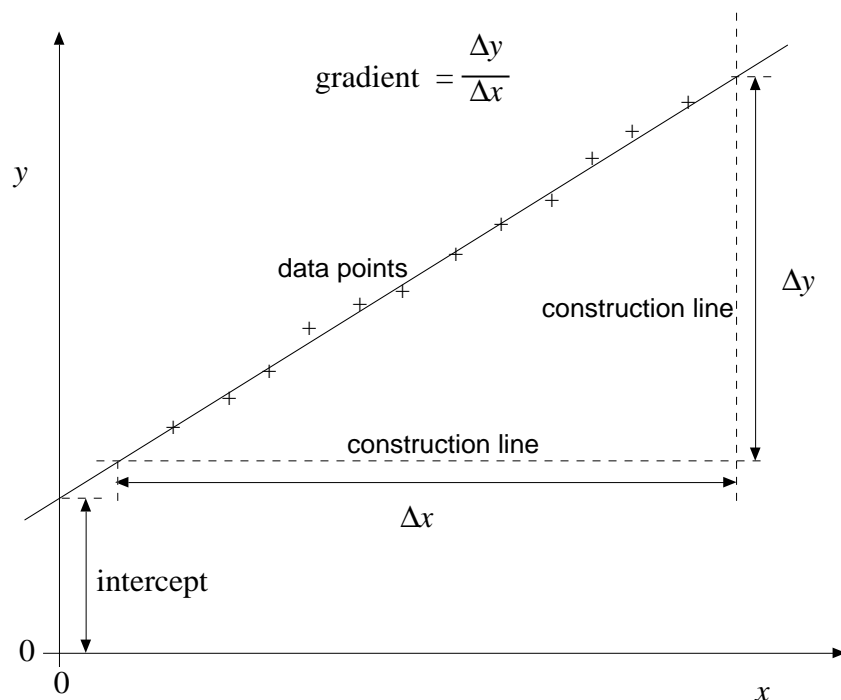
### FITTING DATA TO A STRAIGHT LINE

There is a general mathematical method for achieving these requirements called *the least squares method*, but in most cases we do not need to use any clever mathematics because we can re-cast the equation in the form of a **straight line**.

The general equation of a straight line is

$$y = Ax + B$$

where  $x$  and  $y$  are the two variable quantities that have a straight line relationship and  $A$  and  $B$  are constants. If  $x$  is plotted using the horizontal axis (the *abscissa*) and  $y$  is plotted using the vertical axis (the *ordinate*) then  $A$  will be the *gradient* or *slope* of the line and  $B$  will be the *vertical intercept* of the line. These can be found by simply laying a ruler against the points on the graph, drawing a straight line, and measuring the line against the scales on the axes.



For the **construction lines** use the centimetre lines of the graph paper: do not draw your own (but *do* indicate on the graph which lines are being used for the calculation). Choose lines that lie outside the range of the data points, as shown. This will ensure that you achieve the greatest accuracy in finding the gradient.

[Note that we are finding the gradient of the fitted line, so we do not use the coordinates of any of the data points to do this.]

#### RE-ARRANGING THE EQUATION

To be able to apply this method we need to look at how to re-arrange an equation so that it has the form of a straight line. There is always more than one way of doing this. We usually try to do it so that the gradient or the intercept will equal the quantity we are trying to find ( $g$  in our case), or will have some very simple relationship to what we want to find.

We require there to be one variable quantity on the left hand side, the other variable quantity on the right hand side multiplied by a constant (the gradient), and a further constant added to the right hand side (the intercept). So we might re-write the simple pendulum formula as follows:

$$T = 2\pi\sqrt{\frac{L}{g}}$$

$$T = \frac{2\pi}{\sqrt{g}}\sqrt{L} + B$$

where now our “vertical” coordinate is  $T$ , our “horizontal” coordinate is  $\sqrt{L}$ , our gradient will be  $2\pi/\sqrt{g}$ , and the intercept will be  $B$  (which we expect to come out to be zero).

We can do better than this, however, by giving a little thought to the problem. Can we arrange that the gradient is directly equal to  $g$ , or a simple multiple of  $g$ ? Yes, we can.

$$T = 2\pi\sqrt{\frac{L}{g}}$$

$$T^2 = 4\pi^2\left(\frac{L}{g}\right)$$

$$L = \left(\frac{g}{4\pi^2}\right)T^2 + B$$

where, again,  $B$  is theoretically 0. Now if we make  $L$  our “vertical” coordinate and  $T^2$  our “horizontal” coordinate then the gradient will be  $g/(4\pi^2)$ . So if we determine from our graph that the gradient has the value  $A$ , then we can calculate  $g$  very easily:

$$A = \frac{g}{4\pi^2}$$

$$g = 4\pi^2 A$$

Finding that the intercept  $B$  is close to zero gives a further test of the theory behind the equation.

If we can estimate the uncertainty in the gradient  $A$ , then we can very easily calculate the corresponding uncertainty in  $g$ : it is just the uncertainty in  $A$  multiplied by  $4\pi^2$ . We will look later at how in general we find the uncertainty of a function of a measured quantity.

Note: In the above description the general terminology “ $x$ ” and “ $y$ ” is sometimes used for “horizontal” and “vertical” variables. **Do not do this in the laboratory.** Always use the correct symbols for the quantities you are plotting on the graph, i.e. “ $T^2$ ” and “ $L$ ” in this case.

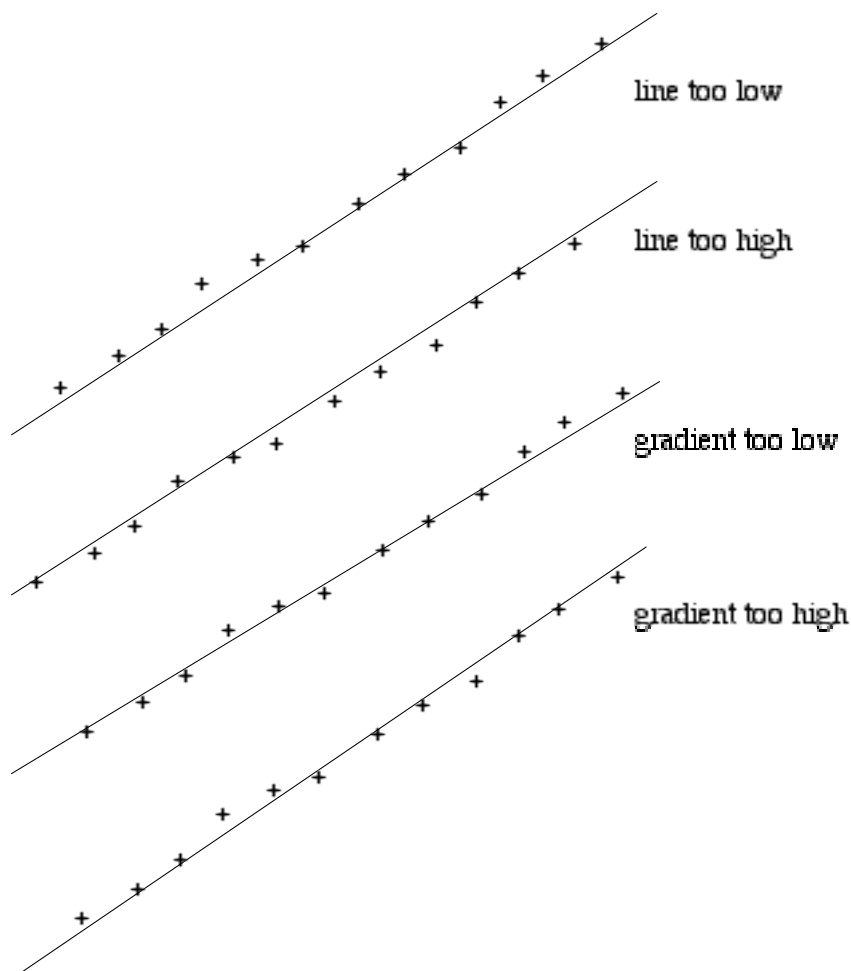
#### RULER TECHNIQUE

When we have plotted the data points on the graph using suitable axes and we bring the edge of a ruler up to lie along these points, we will usually find that they do not fall exactly along the straight edge but are scattered about some imaginary straight line. Our task is to find the line which lies as close to the data points as possible. A transparent ruler (or set square) is required for this since it is impossible to judge how far the points are from the edge when some of them are being obscured by the ruler itself.

Finding “the best line” is a matter of judgement. The method is to keep varying the position of the ruler until there are roughly equal numbers of points above and below the line at both ends.

Check these four things:

- is the line too low; are there more points above it than below it?
- is the line too high; are there more points below it than above it?
- is the gradient of the line too low; are there too many points above the right end or below the left end?
- is the gradient of the line too high; are there too many points above the left end or below the right end?



### UNCERTAINTIES OF THE LINE

We also use the ruler to estimate the uncertainties in the gradient  $A$  and in the intercept  $B$ .

For the uncertainty in the **gradient**,  $\delta A$ , we imagine two additional lines. One, the **MAX LINE**, has a gradient slightly greater than the **BEST LINE** so that it just becomes obvious that its gradient is too great. The other is the **MIN LINE**, which has a gradient slightly less than the **BEST LINE** so that it is just obvious that its gradient is too low. These lines are drawn to have the largest plausible deviation in gradient from the **BEST LINE** whilst still having the least distance from the data points:

The **MAX LINE** will be higher than the **BEST LINE** on the right and lower than the **BEST LINE** on the left.

The **MIN LINE** will be lower than the **BEST LINE** on the right and higher than the **BEST LINE** on the left.

The three lines will therefore **cross over each other** somewhere near the middle of the range of the data.

We obtain the gradients of these two lines in the same way as for the **BEST LINE**. The uncertainty in the **BEST LINE** gradient is then found as

$$\delta A = \frac{1}{2}(A_{MAX} - A_{MIN})$$

i.e. half the difference between the **MAX LINE** gradient and the **MIN LINE** gradient.

Note that these two additional lines are solely for the purpose of estimating the uncertainty and are not used in finding the best gradient.

A similar technique is used to find the uncertainty in the intercept,  $B$ , only this time the choice of MIN and MAX lines depends on where the ordinate axis is relative to the data points.

If the ordinate axis passes near the middle of the range of the data then the two additional lines are **parallel** to the BEST LINE. If the ordinate axis is far away from the range of the data then the lines of MIN and MAX gradients are to be used. Use whichever pair of lines gives the largest range of intercept.

$$\delta B = \frac{1}{2}(B_{MAX} - B_{MIN})$$

where  $B_{MAX}$  is the intercept of the line higher than the BEST LINE and  $B_{MIN}$  is the intercept of the line lower than the BEST LINE.

You need only draw in the two ends of each of the MIN and MAX lines if you think it would make the graph too untidy to show their full lengths.

### 13.4 Propagation of Uncertainties

In the example of the simple pendulum graph we had this relationship between  $g$  and the gradient of the graph,  $A$ , and between the uncertainty in the gradient and the corresponding uncertainty in  $g$ :

$$g = 4\pi^2 A$$

$$\delta g = 4\pi^2 \delta A$$

The second equation states that because we are uncertain of the true value of the gradient of the line then we are correspondingly uncertain of the value of  $g$  by the amount  $\delta g$ . To see that the formula for  $\delta g$  is correct, think of a small change in  $A$  (its uncertainty, i.e. its estimated error) and ask yourself how much the calculated value of  $g$  would change as a consequence. This kind of relationship, between the uncertainty in one quantity and the uncertainty in another which is a function of the first, is called *propagation of uncertainties*.

#### TWO GENERAL METHODS FOR PROPAGATING UNCERTAINTIES

The general relationship can be expressed using the notation of differential calculus. If the quantity  $z$  is some function  $f$  of a quantity  $x$ , and we know the uncertainty in  $x$ , then the uncertainty in  $z$  is found by

$$z = f(x)$$

$$\delta z = \left| \frac{df}{dx} \right| \delta x$$

or put more simply

$$\delta z = \left| \frac{dz}{dx} \right| \delta x$$

Notice that the modulus is required because it may happen that  $z$  decreases when  $x$  increases ( $dz/dx < 0$ ) and uncertainties are always given as positive quantities.

Using this formula is equivalent to imagining that  $x$  were increased by its uncertainty and asking how much  $z$  would change by, so an alternative formula to use is:

$$\delta z = |f(x + \delta x) - f(x)|$$

It doesn't matter which of these two approaches is used to propagate the uncertainties: use whichever seems the easier.

### STANDARD RESULTS FOR PROPAGATING UNCERTAINTIES

We can use the differential calculus approach to find rules-of-thumb for some simple commonly found cases. (In the following,  $k$  is a constant.)

$$(1) \quad \boxed{\begin{array}{l} z = k \pm x \\ \left| \frac{dz}{dx} \right| = 1 \\ \delta z = \delta x \end{array}}$$

Note: independent of constant  $k$ .

$$(2) \quad \boxed{\begin{array}{l} z = kx \\ \frac{dz}{dx} = k \\ \delta z = |k|\delta x \\ \text{or alternatively ...} \\ \delta z = |k|\delta x = \frac{|z|\delta x}{|x|} \\ \frac{\delta z}{|z|} = \frac{\delta x}{|x|} \end{array}}$$

Note: independent of  $k$  when written in the second form using *fractional uncertainties*.

The first of these is the form we used for the simple pendulum.

$$(3) \quad \boxed{\begin{array}{l} z = \frac{k}{x} \\ \frac{dz}{dx} = -\frac{k}{x^2} \\ \delta z = \frac{|k|\delta x}{x^2} = \left| \left( \frac{k}{x} \right) \left( \frac{\delta x}{x} \right) \right| = \left| z \left( \frac{\delta x}{x} \right) \right| \\ \frac{\delta z}{|z|} = \frac{\delta x}{|x|} \end{array}}$$

Note: again independent of the proportionality constant  $k$  when written as *fractional uncertainties*.



$$(4) \quad \begin{array}{l} z = kx^n \\ \frac{dz}{dx} = knx^{n-1} = \frac{nz}{x} \\ \frac{\delta z}{|z|} = |n| \frac{\delta x}{|x|} \end{array}$$

This is the general form for cases (2) and (3). Note again, this is independent of the proportionality constant when written as *fractional uncertainties*.

#### PROPAGATION OF UNCERTAINTIES WITH THE GRAPHICAL METHOD

In the simple pendulum experiment our measurements of the length and of the period are both subject to random error, yet using the graphical method we do not need to estimate these. This is because the errors in these measurements automatically propagate into the uncertainty in the gradient of the graph through the *scatter* of the data points.

Measurement error in either  $L$  or  $T$  will cause the data points on the graph to be randomly scattered away from the ideal straight line and it is the extent of this scatter that determines our estimated uncertainty in the gradient.

So here we have a further advantage of the graphical method: the uncertainties in several different quantities are automatically accounted for by the scatter of the data points.

To tell whether a particular quantity's uncertainty is being accounted for in this way, ask yourself whether an error in one of its measurements will cause one of the points on the graph to move but not any of the others. If so, then this means that the scatter of the points is sensitive to the random errors in that quantity.

An error which affects all the data points simultaneously is not properly accounted for in this way. For example, if the stop watch was running slow, all of our  $T$  measurements would be too small, so all of the data points would be shifted to the left on the graph. This kind of “systematic” error does not therefore affect the scatter significantly and so is not being accounted for by the standard graphical analysis.

#### PROPAGATING UNCERTAINTIES IN MORE THAN ONE QUANTITY

It often happens that the quantity we are calculating depends on several measured quantities and that not all of the uncertainties in these quantities are accounted for by the graphical method.

For instance, say the manufacturer states that the speed of the stop watch has an uncertainty of 0.1% (not a very good stop watch!). This is a *fractional uncertainty*, so according to the formula given in (4) above, the resulting fractional uncertainty in  $T^2$  will be twice this: 0.2%. The gradient is found by dividing an interval of  $L$  by an interval of  $T^2$  so according to the formula in (3) the fractional uncertainty in the gradient will also have a contribution of 0.2%.

This is in addition to the uncertainty due to the random errors in each individual  $T$  measurement which is accounted for by the scatter of the points. So we have two contributions to the uncertainty in the gradient. How do we combine them into one net uncertainty?

We do not simply add them together because they are each random and independent of each other. One random error might be positive and the other negative so that they would tend to cancel each other, or they could be of the same sign and reinforcing each other: we can't know – they are random!. The net uncertainty is therefore less than the simple sum of the two contributions but greater than either of the two alone.

**We combine them “in quadrature”...**

**COMBINING UNCERTAINTIES IN QUADRATURE**

If there are several independent contributions to the uncertainty in the quantity  $z$ , say  $\delta z_1, \delta z_2, \delta z_3, \delta z_4, \dots$ , then the net uncertainty is found by squaring, adding, and then taking the square root:

$$\delta z = \sqrt{\delta z_1^2 + \delta z_2^2 + \delta z_3^2 + \delta z_4^2 + \dots}$$

i.e.

$$\delta z^2 = \delta z_1^2 + \delta z_2^2 + \delta z_3^2 + \delta z_4^2 + \dots$$

Each of the individual contributions is found using the standard methods, treating just one of the measured quantities at a time as uncertain (i.e. *variable*) and the others as fixed (i.e. *constant*).

**STANDARD RESULTS FOR PROPAGATING UNCERTAINTIES IN SEVERAL QUANTITIES**

Again we can use the rules found for propagating the uncertainty in a single quantity together with the above quadrature formula to find rules-of-thumb for some simple commonly found cases:

$z = k \pm u \pm v \pm w \pm x$ $\delta z_u = \delta u \text{ (treating } v, w, x \text{ as constants)}$ $\delta z_v = \delta v \text{ (treating } u, w, x \text{ as constants)}$ $\delta z_w = \delta w \text{ (treating } u, v, x \text{ as constants)}$ $\delta z_x = \delta x \text{ (treating } u, v, w \text{ as constants)}$ $\delta z = \sqrt{\delta z_u^2 + \delta z_v^2 + \delta z_w^2 + \delta z_x^2}$ $= \sqrt{\delta u^2 + \delta v^2 + \delta w^2 + \delta x^2}$ $\delta z^2 = \delta u^2 + \delta v^2 + \delta w^2 + \delta x^2$
---

(B)

$$\begin{aligned}z &= \frac{kvx^n}{w} \\ \delta z_v &= |z| \frac{\delta v}{|v|} \\ \delta z_w &= |z| \frac{\delta w}{|w|} \\ \delta z_x &= |z| \frac{n\delta x}{|x|} \\ \delta z &= \sqrt{\delta z_v^2 + \delta z_w^2 + \delta z_x^2} \\ &= |z| \sqrt{\left(\frac{\delta v}{v}\right)^2 + \left(\frac{\delta w}{w}\right)^2 + \left(\frac{n\delta x}{x}\right)^2} \\ \left(\frac{\delta z}{z}\right)^2 &= \left(\frac{\delta v}{v}\right)^2 + \left(\frac{\delta w}{w}\right)^2 + \left(\frac{n\delta x}{x}\right)^2\end{aligned}$$

Notice that this formula is similar to (A) except that fractional uncertainties are combined in quadrature instead of absolute uncertainties.

These two formulae can only be used when we have a function composed purely of addition and subtraction or purely of multiplication and division. When there is a mixture we must break down the process into several steps, treating one standard part of the formula at a time.

e.g.

$$\begin{aligned}z &= kw(x - y) \\ \text{let } u &= x - y \\ \delta u^2 &= \delta x^2 + \delta y^2 \\ z &= kwu \\ \left(\frac{\delta z}{z}\right)^2 &= \left(\frac{\delta w}{w}\right)^2 + \left(\frac{\delta u}{u}\right)^2 \\ &= \left(\frac{\delta w}{w}\right)^2 + \frac{\delta x^2 + \delta y^2}{(x - y)^2}\end{aligned}$$

## EXAMPLES OF PROPAGATING UNCERTAINTIES

(1)

$$z = \sin(\theta + \phi)$$

$$\frac{dz}{d\theta} = \frac{dz}{d\phi} = \cos(\theta + \phi)$$

$$\delta z_{\theta} = |\cos(\theta + \phi)| \delta\theta$$

$$\delta z_{\phi} = |\cos(\theta + \phi)| \delta\phi$$

$$\delta z = |\cos(\theta + \phi)| \sqrt{\delta\theta^2 + \delta\phi^2}$$

Note that the differential derivative is correct only if the angles  $\theta$  and  $\phi$  are in radians.

(2)

$$n = \frac{\sin\left(\frac{A+D}{2}\right)}{\sin\left(\frac{A}{2}\right)}$$

Here is a case where the formulae for the differential derivative are rather complicated and it is probably best to employ the “direct” method of propagating the uncertainties:

$$\delta n_A = \left| \frac{\sin\left(\frac{A+\delta A+D}{2}\right)}{\sin\left(\frac{A+\delta A}{2}\right)} - n \right|$$

$$\delta n_D = \left| \frac{\sin\left(\frac{A+D+\delta D}{2}\right)}{\sin\left(\frac{A}{2}\right)} - n \right|$$

$$\delta n = \sqrt{\delta n_A^2 + \delta n_D^2}$$

(3)

$$z = \ln(N)$$

$$\frac{dz}{dN} = \frac{1}{N}$$

$$\delta z = \frac{\delta N}{N}$$

## 13.5 Miscellaneous

### SYSTEMATIC ERROR

The case cited in section (4) where we imagined using a stop watch that runs slow is an example of a *systematic error*. This term means any error which is unchanging throughout the experiment and does not show up as random variations in the results through repeated measurements.

It is in the nature of systematic errors, therefore, that we have no direct way of finding out their magnitude. Usually we have to make a reasonable guess about them based on our understanding of the way the apparatus works. In principle we should do this for every individual piece of equipment that we use. The total uncertainty is then the quadrature combination of the statistical uncertainty and the systematic uncertainty.

Sometimes the manufacturer states the “tolerance” or “calibration accuracy” of the equipment. If not, then the finest division of the measurement scale is usually taken as a reasonable guess for the systematic uncertainty.

You may find the term *systematic error* useful if you find you wish to express a suspicion that there is indeed some hidden error due to imperfections in the apparatus which has clearly distorted your results. This could arise, for example, if you expect to find a straight line graph but your data shows some curvature. But do not immediately leap to the conclusion that the *apparatus* is “wrong”. It is very much more likely that you have made some error in performing the experiment or in analysing the results.

#### FORMAT FOR RECORDING RESULTS

All results should be recorded using the following format:

$$x \pm \delta x \text{ units}$$

where  $x$  is the best estimate of the value of the measured quantity,  $\delta x$  is the estimated uncertainty in this value, and *units* are the most appropriate units for both  $x$  and  $\delta x$ .

The final net uncertainty should be rounded to two significant figures (“2 s.f.”). This means you must first calculate the uncertainties to three or more significant figures.

When this has been done, the value of the measured quantity should be rounded to the same number of decimal places, in the same units, as the uncertainty. The value of the quantity should therefore also be first calculated to a greater precision than two significant figures of the uncertainty. Here “units” means *including any factor of a power of 10*.

#### **Correct examples:**

$$9.81 \pm 0.13 \text{ ms}^{-2}$$

$$0.0369 \pm 0.0050 \text{ }\mu\text{F}$$

$$(2.134 \pm 0.022) \times 10^{-8} \text{ C}$$

$$1004 \pm 12 \text{ mm.}$$

Wherever possible use a standard S.I. unit with one of the standard order-of-magnitude prefixes:

$$\text{p} \equiv 10^{-12} \quad (\text{pico...})$$

$$\text{n} \equiv 10^{-9} \quad (\text{nano...})$$

$$\mu \equiv 10^{-6} \quad (\text{micro...})$$

$$\text{m} \equiv 10^{-3} \quad (\text{milli...})$$

$$\text{k} \equiv 10^3 \quad (\text{kilo...})$$

$$\text{M} \equiv 10^6 \quad (\text{mega...})$$

$$\text{G} \equiv 10^9 \quad (\text{giga...})$$

$$\text{T} \equiv 10^{12} \quad (\text{tera...})$$

**Incorrect examples:**

- $9.81 \pm 0.08 \text{ ms}^{-2}$  [uncertainty given to only one significant figure]
- $0.037 \pm 0.0050 \text{ }\mu\text{F}$  [precision of value of quantity does not match precision of uncertainty]
- $2.134 \times 10^{-8} \pm 2.2 \times 10^{-10} \text{ C}$  [value and uncertainty do not have the same units]
- $10040 \pm 100 \text{ mm}$  [precision of value and uncertainty are ambiguous owing to trailing zeroes]

**COMPARISONS BETWEEN RESULTS OR WITH ACCEPTED VALUE**

An experimental result is of greatest significance if it can be compared with another independent result for the same quantity. This might be some “accepted” value found in a reference book of physical tables such as the one by Kaye and Laby.

When we make such a comparison we should say whether the two values are *consistent* or *inconsistent*. To do this we find the difference between the two values and compare this with their combined uncertainty.

If the difference is significantly less than twice the combined uncertainty we normally declare the two values to be consistent.

If the difference is significantly greater than twice the combined uncertainty we normally declare the two values to be inconsistent.

If the difference is of the order of twice the combined uncertainty we normally declare that the consistency of the values is undecided.

e.g.

first value	uncertainty	second value	uncertainty	difference	combined uncertainty	comment
1763	22	1732	15	31	27	consistent
1712	12	1778	15	66	19	inconsistent
1779	15	1732	15	47	21	undecided

**GRAPHS**

When your experiment is to be analysed using a graph, make your measurements so that the values cover **as wide a range as possible**. At least four points are needed for a meaningful graph and you should normally aim for ten to twenty depending on the available time.

When plotting the data on the graph use a **clear symbol** such as + or ×, not just a dot.

Do **not** force the fitted line to pass through the origin: always determine the intercept from the data.

If there is more than one graph for the experiment, each should have a **number**: Graph 1; Graph 2; etc.

All graphs should have a **title** which describes what is being plotted in words and refers these quantities to the symbols used in the axis labels. It should not be necessary for the reader to read through your report or the experiment script in order to get an idea of the purpose of the graph.

### GRAPHS: CHOICE OF SCALES

Use only scales that make plotting and reading the graph as easy as possible. This means that each centimetre division of the graph paper should correspond to a factor of 1, 2 or 5 units times some power of 10.

Choose scales that will make the plotted data span as large a region of the paper as possible consistent with the above restriction. This will often mean that there will be a “false origin”, i.e. the true origin is not shown.

Enumerate every centimetre of the scale.

Label the scale with the quantity being plotted and its correct units in brackets:  
e.g.  $T^2$  (s<sup>2</sup>).

### GRAPHS: FINDING THE INTERCEPT WHEN THE DATA IS FAR FROM THE ORIGIN

It often happens that we are plotting the data on a graph with a “false origin” for the abscissa (horizontal quantity) which means that we cannot find the intercept with the ordinate axis by drawing the fitted line to intersect it. In this case we have to **calculate** the intercept using the gradient of the line and some point that lies on the line.

If the equation of the line is

$$y = Ax + B$$

and  $(X, Y)$  is a point that lies on the line and we have found the gradient  $A$ , then

$$Y = AX + B$$

$$\therefore B = Y - AX$$

### ALTERNATIVE FORMULA FOR STANDARD DEVIATION

The formula for the standard deviation given in section (2) is

$$\sigma = \sqrt{\frac{\sum_{i=1}^N (x_i - \bar{x})^2}{N - 1}}$$

There is a practical complication in using this formula since the mean  $\bar{x}$  must be calculated before the squares of differences  $(x_i - \bar{x})^2$  can be computed and summed. This means that, when using a calculator or computer, the full data set must be stored. This makes the program more complex and for a large data set means that a very large amount of computer memory could be needed.

By developing the equation, however, we can find an expression that enables the standard deviation and the mean to be computed “as we go” using only the latest data value without memorising any of the previous ones.

$$\begin{aligned}\sigma^2 &= \frac{\sum_{i=1}^N (x_i - \bar{x})^2}{N - 1} = \frac{\sum (x_i^2 - 2\bar{x}x_i + \bar{x}^2)}{N - 1} \\ &= \frac{\sum x_i^2 - 2\bar{x} \sum x_i + N\bar{x}^2}{N - 1} = \frac{\sum x_i^2 - 2\bar{x}(N\bar{x}) + N\bar{x}^2}{N - 1} \\ &= \frac{\sum x_i^2 - N\bar{x}^2}{N - 1} = \frac{\sum x_i^2 - (\sum x_i)^2 / N}{N - 1}\end{aligned}$$

where we have used the definition of the mean:

$$\bar{x} = \frac{\sum x_i}{N}$$

Using the final expression for  $\sigma^2$  we only need to memorise the number of data values so far,  $N$ , the sum of all data values so far,  $\sum x_i$ , and the sum of the squares of all data values so far,  $\sum x_i^2$ . Including further data values in the calculation simply requires updating these three quantities to then re-calculate the mean and standard deviation.

### 13.6 Exercises

- For the following data set  
5.52, 4.74, 3.60, 4.92, 4.02  
calculate the following quantities:  
mean;  
standard deviation using the formula in section (2);  
sum of squares;  
standard deviation using the formula in section (5);  
standard deviation using a scientific calculator function, if you have one;  
standard error.
- Plot the following pendulum data on a graph, choosing appropriate scales, labelled correctly:

$T^2$ (s <sup>2</sup> )	$L$ (mm)
0.80	227
1.66	390
1.78	462
2.29	544
2.39	616
3.09	737
3.32	851
4.07	987

Find the best gradient and estimate its uncertainty. Quote the result in the standard format with correct units.

Calculate the acceleration due to gravity and its uncertainty. Quote the result in the standard format with correct units.

Compare the result with the accepted value of  $9.81 \text{ ms}^{-2}$ .

Calculate the intercept using the point on the line at  $L = 500 \text{ mm}$ .

- Two temperatures are measured as follows:  
 $T_1 = 48.70 \pm 0.50 \text{ }^\circ\text{C}$   
 $T_2 = 18.5 \pm 1.0 \text{ }^\circ\text{C}$   
Find the temperature difference  $T_1 - T_2$  and its uncertainty.



- 4 In a certain experiment the formula used to find the thermal conductivity of a sample is

$$\kappa = \frac{VIL}{A(\Delta T)}$$

where the measured values are

$$V = 10.00 \pm 0.20 \text{ V}$$

$$I = 2.60 \pm 0.10 \text{ A}$$

$$L = 3.25 \pm 0.10 \text{ mm}$$

$$A = 27.45 \pm 0.50 \text{ cm}^2$$

$$\Delta T = 16.2 \pm 1.2 \text{ }^\circ\text{C}$$

Put these measurements into a consistent set of units and then calculate the thermal conductivity,  $\kappa$ .

Set out each stage of your calculation clearly so that it is easy to check for mistakes.

Write down the formula to find the uncertainty in  $\kappa$ .

Calculate the uncertainty in  $\kappa$ .

- 5 In an experiment to measure the refractive index  $n$  of glass using the minimum deviation of light through a prism, the formula used to find  $n$  is

$$n = \frac{\sin\left(\frac{A+D}{2}\right)}{\sin\left(\frac{A}{2}\right)}$$

The measured angles are

$$A = 59.96 \pm 0.10 \text{ }^\circ$$

$$D = 36.23 \pm 0.60 \text{ }^\circ.$$

Find the refractive index and its uncertainty.

Set out each stage of your calculation clearly so that it is easy to check for mistakes. Always first write down any formula using purely symbols and only then substitute the numerical values to find the result.

## 14 Word-processing

### 14.1 Introduction: documentation, assessment, deadlines

The word-processing program *Microsoft Word* is available on College PCs in room D105 and also on the PCs in the Physics Department Teaching Laboratory (version 2002/XP and version 97 respectively). *Word* runs under the *Windows NT* or *98* operating system as does *Visual Basic* which we will use in the computer programming part of this course.

Using *Word* for basic word-processing tasks is described in the tutorial document *Word 2002 (XP) Part 1* (document 349) published by the College Information Systems Division (ISD) and provided for you. If you wish to buy a **book** then see section 9.3 of this Handbook. Note that some books about this version of *Word* will call it “Word 2002”, others will call it “Word XP”: they are the same thing!

The sixth and final unit of assessment of this course (following the five units of computer programming) is to use *Word* to produce one of the two “formal reports” on your physics laboratory work. The two experiments for which full reports are to be produced will be notified to you during term 2. The word-processed report will be assessed twice — first as a physics practical report, then as an exercise in word-processing. The **deadline** for the submission of this work is the first Thursday of the Summer Term, i.e. 1st May 2003.

#### What is a word-processor?

In principle an advanced word-processor can be used to produce a complete document: text, graphs, equations and illustrations (sometimes in combination with other application programs) of a quality almost up to that of a printed book. However in the short time available we will only be able to cover the most basic aspects of text preparation, therefore the **graphs, drawings** and some of the **equations** in your report will have to be done by hand, with spaces left in the printed *Word* document where these elements are to be written or drawn later. All graphs should be drawn on the correct type of millimetre graph paper.

Perhaps the first thing to note about the operation of a word-processor is that, unlike a typewriter, it does not require you to signal the end of each line of text: **the computer automatically continues onto the next line where necessary**. The **Enter key** (or Return key) is used to signal the end of a paragraph when using a word-processor, not the end of a line. The line lengths are adjusted automatically by the word-processor to fit between the margins of the text, flowing on to the following line as necessary. This feature is called *word wrapping*. At a more sophisticated level word-processors have many formatting features to enable you to affect the appearance of your text such as **boldening**, *italicising*, and underlining, as well as changes in **text** size, <sup>superscript</sup> and subscripts, and **different** typefaces (also known as fonts).

In this course we will mainly be using the ISD tutorial notes (accompanied by these guidance notes) but to help you find something quickly, at the end there is a summary of the *Word* features covered by the notes and a few others that you may find useful.

## 14.2 If You Are Already Experienced In Word-processing

This course is designed primarily for people who have no previous experience with a word-processor. If you are already familiar with the basic operations then all you need is to get to know the *Windows* environment implemented on the College computers, perhaps the menus and toolbars of *Word 2002* and the conventions and tools used in scientific word-processing. You should therefore work rapidly through the tutorial, skipping anything that you already know, and then work carefully on the advanced exercise described in section 14.5.

## 14.3 Guide to the Tutorial

Please note that these notes were prepared using a draft version of the Tutorial so it is possible that some of the section or page references may not be accurate.

Throughout the Tutorial there are *Tasks* to perform and at the end there are *Practice Exercises*. I suggest that you do not do the exercises at first, only the *tasks*. Try the exercises if you wish when you have finished the tutorial as a self-assessment.

**Read the introductory section on page iii in the ISD tutorial document *Word 2002 Part 1*.**

Now proceed through the Tutorial sections and tasks in order, checking with these notes before starting each one to see if it is required or if there is any additional information you need to know. About five minutes before the end of the class carry out the operations in **Section 3.9** which tells you how to finish off. Also refer to section 9.6 of this Handbook to see how to log off the computer before leaving.

**Follow the operations described in Chapter 1: *The Word Environment*.**

If you are new to computing or to the *Windows* operating system then follow what I do and let me know immediately if you are having difficulty with anything. Take it slowly to begin with.

**SKIP Chapter 2 for now.**

**Change the Auto-formatting settings as follows:**

When you first start *Word* all of the “Auto” features will be turned *on*. These are intended to correct accidental typing errors and to anticipate what you are trying to do. However they can be extremely irritating if you are attempting to do something slightly unconventional, such as typing in a simple equation like

$$y = mx + c$$

The standard Auto-correction feature thinks you have mistakenly begun a new sentence with a lower case letter and will change the “y” into a “Y”. There are many other automated operations within *Word* that can get you very confused if you don’t know what’s happening. It is therefore best to begin with to turn most of them *off*.

To do this go to the **Tools** menu and choose **Auto-correction Options**. Click the **Auto Correct** tab and then clear all of the ticks by clicking in the boxes. Then click “OK”.

**Read Sections 3.1 and 3.2 then carry out Task 3 part 1 on p.14:  
*Typing / Correcting Text.***

Note that the shape of the characters that appear as you type the document will probably be different to the characters as they appear in the Tutorial notes. The number of words that fit on one line will also probably be different. Don't worry about this.

**Read Section 3.3 to 3.6, *Saving...*, and carry out parts 2 and 3 of Task 3.**

If you are new to computing this is another point where you will probably need to carry out the instructions under my guidance. Please let me know when you have reached this point.

In Task 3 you are required to save your work on the network disk called “r:”. You should only do so if you have been allocated your own unique Username (“user ID”) and password.

If you have not yet received your own ID and are using one of the “guest” IDs then you should save your work on your floppy disk, called disk “a:”. Even if you do have your own ID, at the end of the session before you exit from the *Word* program you should also save your work on the floppy disk.

Note, if you cannot find the specified folder `training.dir\wrdp1` then just save the file directly to drive r. Always make a note of where you have saved your work for future reference.

If at any stage in using the computer you get the message

Floppy disk A not ready / Abort, Retry

or something similar, this means that the computer is waiting for a floppy disk. Insert a disk and click on “Retry”. If you don't have a disk with you, click Abort and then use the Network disc instead of a floppy disc. However it is much the best thing to avoid this situation by always putting your floppy disk in before selecting the program you are going to use. The only time when it is important not to have a disk in the computer is when first switching on the machine or restarting Windows from the *Start* menu.

**Read Sections 3.7 and 3.8 and carry out part 4 of Task 3.**

Note: There is a copy of the file `bentham.doc` provided on your floppy disk if you cannot locate it in `training.dir\wrdp1` on the network disk.

**Read Section 4.1 *Moving Around a Document* and then carry out Task 4 on p 17.**

This assumes you have already loaded the file `r:\training.dir\wrdp1\bentham.doc` as instructed in Task 3 part 4. You can also load the file from your floppy disk where it is called just `bentham.doc`, so select disk drive “a”, not “r”. You need a long document like this to properly try out some of these techniques and loading this ready-made one saves you having to do a lot of typing.

**Read Sections 4.2 – 4.9 and then carry out Task 5 on p 21: *Selecting, Copying, Moving, Deleting, etc.***

Note: if you type anything while some text is selected — deliberately or accidentally — what you type will replace all the text that was previously selected. If this is not what you intended you can recover the lost text by clicking on the UNDO button in the standard tool bar.

**Read Section 4.10, *Find and Replace*, and then carry out Task 7 on p 25.**

**Read Chapter 5, *Spelling and Grammar Checking*, and then carry out Task 8 on p 29.**

Note that the *Word* program is often encountered set up to use American English. Whenever you open a document or start a new document you should first select all of the text using the **Edit/SelectAll** menu and then choose the **Tools/Language/SetLanguage** menu to set the language to UK English. The spell checker will then use the appropriate dictionary.

**Read Chapter 6 and then carry out Task 9 on p 33: *Formatting Text*.**

In scientific writing we frequently need to include a character as a **superscript**, e.g. to raise a symbol to a power:  $x^2$ ,  $e^{-x}$ . We also need **subscripted** characters, as for indexed variables:  $x_i$ . As well as being raised or lowered, these characters are also usually in a smaller size than the main text. This is achieved using the **Format/Font** menu and clicking in the box next to the appropriate “effect”. You might like to try out some of the other effects listed here as well.

Unfortunately there are no icons for these features on the standard format toolbar, but you can obtain them if you wish. Here's how.

From the **Tools** menu select **Customize**. Click on the tab **Commands** and select **Format** in the **Categories** list. Now in the **Commands** box on the right you will get a long alphabetical list of all the formatting commands available in *Word*. Scroll down until you find **Subscript** and **Superscript**. Drag these out of the dialogue box one at a time onto the **Format Tool Bar** and you will get new icons for subscript and superscript. The toolbar may now be too long to fit on the screen so if you want you can reduce its length by dragging *off* the toolbar any icons that you never use (whilst still in the **Customize** dialogue).

**Read the introduction to Chapter 7, *Formatting Paragraphs*, and section 7.2, *To Align Text*. Carry out parts 1–4 of Task 10 on p 37.**

**Skip Sections 7.3 and 7.4 and Task 11.**

**Read Chapter 8, *Tabs*, but skip section 8.1.1, and carry out Task 12 on p 44.**

**Read Chapter 10, *Page Layout*, and then carry out Task 15 on p 56.**

Your documents must always have page numbers. I recommend that you place these in a **header** or **footer** rather than using the **Insert/PageNumbers** command (see sections 10.3.1) since it is much easier to control the layout that way and having a header/footer and separate page numbers at the same time can cause much confusion.

**Read Chapter 12, *Printing*, and then carry out Task 17 part 1 on p 61.**

Note, whenever you print a document make sure that it contains your name at the top of the first page so that you can identify your own material. This is particularly important on a course like this where everyone is printing almost identical documents on the same printer! Using a Header is a good way to achieve this.

**Read Chapter 2, *Getting Help*, and then carry out Task 2 on p8.**

## 14.4 Important Topics Not Covered by the Tutorial

### SPECIAL SYMBOLS

There are many symbols that we need in physics and maths that do not appear on the computer keyboard such as Greek letters,  $\alpha$ ,  $\beta$ ,  $\chi$ ,  $\Gamma$ , and the multiplication sign  $\times$  (never use an x for this).

There are two “fonts” that are particularly useful in mathematical or scientific texts, called *Symbol* and *MT Extra*, selectable directly from the format tool-bar. The **Symbol font** contains the Greek characters plus most of the other special symbols that you will need. The keys A, B, C, ... X, Y, Z give the characters  $\alpha$ ,  $\beta$ ,  $\chi$ , ...  $\xi$ ,  $\psi$ ,  $\zeta$ . **MT Extra** contains rather few characters (not all the keys on the keyboard do anything) but some particularly useful ones are “h-bar” ( $\hbar$ , obtained by pressing the H key) and “curly el” ( $\ell$ , obtained by pressing the L key) which are used in quantum mechanics and atomic and nuclear physics. All the symbols available are listed at the end of this chapter.

A quick way to get a single character in **Symbol font** is to first press the key combination Ctrl/Shift/Q and then the next key to be pressed will be in Symbol font. To quickly get a sequence of characters in Symbol font you can first type them in the normal font, then *select* them and then press Ctrl/Shift/Q.

Apart from the Greek characters, **other special symbols** are best obtained by using the **Insert/Symbol** menu. This produces a dialogue box with a restricted choice of special fonts and a table of symbols for each. You can click on any symbol to see it magnified and then click the Insert button if you decide you want it in your document.

There is a rather serious shortcoming of this feature in *Word* version 2002 which is that the table of characters does not tell you which key to press to get the symbol directly without using the **Insert/Symbol** dialogue box, which you can do by first selecting the corresponding font in the normal format tool-bar and then pressing a key. With the dialogue box open you can find out by trial and error what key combination (with or without the Shift key) produces which character by just pressing them at random! Alternatively, *select* the symbols in your document and then put them into your usual font to find out which Roman letters they correspond to. Make a note of this and then use the UNDO button to restore the symbols to their proper form.

### EXERCISES

Try selecting some text and then turning it into Symbol font using the Font box or Ctrl/Shift/Q. You will see that it all looks Greek. Click the UNDO button to put it back as it was, or select “Default Paragraph Font” from the Styles box. Now try typing in the following:

The value of the fine structure constant,  $\alpha$ , is 1/137.

You will need to select Symbol font before pressing the **A** key to get the *alpha* and then revert to your normal font to type the remainder of the sentence (this happens automatically if you use the Ctrl/Shift/Q method).

Now try something a bit more difficult using the Symbol font, subscript and superscript, and tabs:

We can also write 
$$\alpha = e^2/(4\pi\epsilon_0\hbar c) \tag{1}$$




Notice that in scientific writing we employ the *italic* style for letters of the Roman alphabet when they are used as mathematical symbols.

Try inserting a few special characters into the document and view the different characters available with the various fonts.

## TABLES

Scientific data is invariably presented as tables of numerical values as well as graphs. Such tables can be constructed using tabs but there is a special tables feature in *Word* that provides greater flexibility.

Before inserting a table give careful consideration to how many rows and column it should have, allowing for headings. Then select the **Table/Insert** menu and enter the required number of rows and column in the dialogue box. You can change this later as well as adjusting the width and height of columns and rows and adding borders. Here is the result of inserting a 4×7 table with auto-formatting:


Enter your text and data by clicking in each cell or move around by using the TAB key. Then you can set styles and alignment in the same way as for normal text. To select one cell, position the cursor towards the bottom left of the cell so that you get an arrow symbol  and click. To select a whole row, position the cursor to the left of the row and click, or drag up and down to select several rows. To select a whole column, position the cursor above the column so that you get the symbol  and click, or drag left and right to select several columns. To adjust the column width, position the cursor over the column border to get a symbol like  and drag left or right. Try the effect of holding down the left Shift key while you do this: it changes the effect you have on the *other* columns To insert an extra row or column select the row or column below or to the right of where you want to insert and select the **Table/InsertRow** or **Table/InsertColumn** menu. Deleting rows and columns is done in a similar way. To change the style of borders, select the whole table and then choose the **Format/Borders and Shading** menu.

## EQUATIONS

*Word* does contain a special facility for producing **equations** (the Equation Editor) obtainable using the **Insert/Object** menu. This takes you to a dialogue box that displays a list of different object types among which is *Microsoft Equation*. Select that, make sure the *Float over text* box is not checked, and press OK. Now you are taken to a special equation design window that has its own menus and formatting icons. When you have finished designing your equation simply click in the main part of the document and the equation will be inserted. To re-edit an existing equation simply double-click on it.

The best way to become familiar with the equation editor is by experimentation. It is quite intuitive to use but its operation is too complex to be covered in this short course. When you come to write your practical physics report using *Word* it is perfectly acceptable for you to leave blank spaces in the printed text and then write in the equations afterwards with a pen. Mastering equation editing can be rather time consuming so I recommend that you do not attempt it until you have got through the first year of your course unless you are a skilled computer user.

## 14.5 Typography

### MAIN TEXT

The main text should all be in the same font at size 11 or 12 point. A common choice is Times Roman or Times New Roman (as here). Symbols for physical quantities should be in italics unless they are Greek characters or unless they are vectors when they should be in bold type (not underlined). Functions such as sin or log or  $\frac{d}{dt}$  should not be italicised.

A special case is when you want to represent a physical quantity with a lower case vee italicised, e.g. for speed. In most fonts this comes out (*v*) looking too much like the Greek letter nu ( $\nu$ ). The Palatino font will give you a recognisably distinct italic *v*: *v*.

### PARAGRAPHS

For greatest clarity leave a blank line between paragraphs. You do not then need to indent the first line.

### HEADINGS

In printed text underlining is very rarely used for headings and you should avoid it. Use bold text of an increased size, possibly in a different font, e.g. Arial, as in the Tutorial document. The title of the document can be centre aligned but section and sub-section headings should be aligned at the left. Sub-headings should be of the same size as the main text and can be bold or a different font or italic or small capitals (as here). Do not apply more than one effect at the same time. Make sure that your heading styles are consistent throughout your document; the format painter is very useful for achieving this.

### EMPHASIS

In printed documents underlining is rarely used for emphasis and you should avoid it. Use bold or italic text but not both at the same time.

### PUNCTUATION

Remember that punctuation marks never have a space before them and always have a space after (except for opening quotation marks).

## 14.6 Advanced Exercise

Do not worry if you do not have time to attempt this exercise: you can return to it at the end of your first year since a high level of competence will not be required until your second year.

Copy out as accurately as you can the first two pages of the *Notes On the Treatment of Experimental Data* in chapter 13 of this Handbook. Pay special attention to the styles of chapter and section headings and sub-headings, spacing between paragraphs and headings, use of text formatting for emphasis and for mathematical symbols, indenting of paragraphs, and setting out equations using the Equation Editor.



## 14.7 Summary of *Word* features

**To start *Word*:** Double click the *Word 2002* icon on the *Windows* desk-top or go to the Start menu and look under Programs

**To exit from *Word*:** use the File/Exit menu.

**To set the insertion point:** click at the point in the text where you wish to insert.

**To force a new line:** press the Enter key.

**To delete single characters:** at the insertion point: use the Backspace key.

**To delete sections of text:** select the text to be deleted, then press Backspace.

**To select a word in the text:** double click on the word.

**To select one line of text:** click to the left of the line (mouse pointer is an arrow).

**To select a range of text:** drag over the text, or click at beginning then hold down the Shift key and click at end.

**To de-select text:** click anywhere in the text.

**To replace text:** select the old text then type the new text.

**To set bold, italic or underline styles:** select the text then click the **B** *I* or U buttons in the formatting tool bar, or press Ctrl/B, Ctrl/I, Ctrl/U.

**To recover from a mistaken action:** click the UNDO button in the standard tool bar.

**To search for text:** use the Edit/Find... menu, or press Ctrl/F.

**To save document to disc:** click the SAVE button in standard tool bar or use File/Save menu.

**To save document under a new name:** use File/SaveAs... menu.

**To close a document:** use File/Close menu.

**To open a new document:** click on NEW WINDOW button in standard tool bar or use File/New menu.

**To open an existing document:** click on OPEN DOCUMENT button in standard tool bar or use File/Open menu.

**To move through a long document:** click scroll bar, click scroll bar arrows, or drag scroll box.

**To move a section of text within the document:** select the text and then either drag it to the new location, or use Cut and Paste in the Edit menu or the CUT and PASTE buttons in the standard tool bar.

**To copy a section of text within the document:** select the text and then either drag it to the new location while holding down the Ctrl key, or use Copy and Paste in the Edit menu or the COPY and PASTE buttons in the standard tool bar.

**To change the font (typeface):** select the text to be changed and then choose new font name and size from the 2nd and 3rd selection boxes of the format tool bar, or use the Format/Font menu, or the Font menu.

**To indent a paragraph:** click the INCREASE INDENT button in the formatting tool bar, or drag the margin square in the ruler.

**To set left margin for paragraph:** Drag the square and triangles at the left of the ruler, or use the Format/Paragraph... menu.

**To set line spacing in a paragraph:** use the Format/Paragraph... menu.

**To display paragraph, tab, and space markers:** click the paragraph marker ¶ in the standard tool bar.

**To restore default formatting of paragraph:** Ctrl/Q.

**To select Symbol font:** Ctrl/Shift/Q

**To select normal font:** choose Default Paragraph font from Style box (1st box in format tool bar) or use Ctrl/Space.

**To centre paragraph text:** click on CENTRE TEXT button in the formatting tool bar.

**To include special characters:** use the Insert/Symbol... menu, combined with font selection.

**To have page numbers printed:** use the View/Head and Footer... menu.

**To preview the printed form on screen:** click the PRINT PREVIEW button in the standard tool bar.

**To force a page break:** Ctrl/Enter.

**To print a whole document:** click the PRINT button in the standard tool bar.

**To set tab markers:** click the box at the extreme left of the ruler until the required type of tab is displayed (left, right, centre, or decimal), then click within the ruler where you want the tab to appear.

**To move existing tab markers:** drag the tab marker within the ruler.

**To remove a tab marker:** drag the tab marker out of the ruler area.

**To enter subscript or superscript:** Use the Format/Font... menu.

## 14.8 Special symbols available using the *Symbol* and *MT Extra* fonts

key	Symbol	MT Extra	key	Symbol	MT Extra	key	Symbol	MT Extra
a	$\alpha$	$\mapsto$	A	A		1	1	$\lrcorner$
b	$\beta$	$\updownarrow$	B	B		2	2	$\rceil$
c	$\chi$	$\updownarrow$	C	X	$\amalg$	3	3	$\lrcorner$
d	$\delta$		D	$\Delta$	$\lambda$	4	4	$\_$
e	$\epsilon$		E	E		5	5	
f	$\phi$	$\succ$	F	$\Phi$		6	6	$\lrcorner$
g	$\gamma$		G	$\Gamma$		7	7	$\rceil$
h	$\eta$	$\hbar$	H	H		8	8	$\lrcorner$
i	$\iota$		I	I	$\cap$	9	9	
j	$\phi$		J	$\vartheta$		0	0	
k	$\kappa$		K	K	$\dots$	-	-	
l	$\lambda$	$\ell$	L	$\Lambda$	$\dots$	=	=	
m	$\mu$	$\mp$	M	M	$\vdots$	\	$\therefore$	
n	$\nu$		N	N	$\dot{\cdot}$	,	,	
o	$\omicron$	$\circ$	O	O	$\ddot{\cdot}$	.	.	
p	$\pi$	$\prec$	P	$\Pi$		/	/	
q	$\theta$		Q	$\Theta$	$\ddot{\cdot}$	[	[	
r	$\rho$	$\_$	R	P		]	]	
s	$\sigma$	$\_$	S	$\Sigma$		;	;	
t	$\tau$	$\_$	T	T		'	$\eth$	
u	$\upsilon$	$\_$	U	$\Upsilon$	U	#	#	$\text{'}\prime$
v	$\varpi$	$\_$	V	$\zeta$				
w	$\omega$	$\_$	W	$\Omega$		<	<	$\triangleleft$
x	$\xi$		X	$\Xi$		>	>	$\triangleright$
y	$\psi$		Y	$\Psi$		?	?	
z	$\zeta$		Z	Z		{	{	$\}$
!	!		&	&	$\cdot$	}	}	$\}$
"	$\forall$		*	*		:	:	
£	$\leq$		(	(	$\smile$	@	$\equiv$	
\$	$\exists$	$\wedge$	)	)	$\frown$	$\sim$	$\sim$	
%	%	$\sim$	$\_$	$\_$		$\text{'}\prime$		$\text{'}\prime$
^	$\perp$		+	+		$\neg$	$\leftarrow$	

## 15 Computer Programming In Visual Basic

### 15.1 Introduction

The *Basic* programming language is not a standardised language: it varies considerably in its syntax from one kind of computer to another and even between different versions for the same computer. For this reason it is not really appropriate to learn it from a general text book: you need documentation specifically for the version of Basic that you will be using. It does however have the advantage over all other languages that it has until recently been available as standard on virtually every computer, from the cheapest home micro to the largest supercomputer. It is also a fairly simple language, which makes it relatively easy to learn.

In this course you will learn the version of Basic called **Visual Basic 6.0** that runs under the Windows operating system. Visual Basic is not provided as a standard part of the installed system of PCs and is expensive to buy. However you are provided with a CD-ROM containing a version called *Visual Basic 5 CCE* that is freely available and which you can use if you wish to do some programming away from College. It is very similar to Visual Basic 6.0 but lacks a few features that you will not need and also lacks the Help files.

Visual Basic is quite a complex application to use and not easy to get used to. (Unfortunately there is no longer a really simple version of Basic that runs under the Windows operating system.) It is designed to provide the programmer with the full range of all the user interface tools that you are familiar with from using Word and other Windows applications: menus, dialogue boxes, list boxes, action buttons, multiple text windows, etc. However in this course we will be focusing just on the essentials of writing programs for performing simple computations.

### 15.2 Books

You may not need to buy a book if you find these notes and the Visual Basic Help facility adequate, but if you wish to, see section 9.3 of this Handbook for some suggestions.

### 15.3 The Course-work

There are five items of course-work for the computer programming part of 1B70, called Unit 1 to Unit 5. The **deadline** for handing in each Unit of computer programming course-work is approximately four weeks after the material is covered in the course and is stated at the top of the Unit documentation (see later in this Handbook). Work handed in late will be marked out of a maximum that decreases by 10% for each working day. Each Unit will only be accepted for marking when all previous Units have been handed in.

Apart from Unit 1, your course-work should be handed in in the form of a computer print-out of your computer program code accompanied by computer printed examples of the output produced by the program. Usually several examples of output are expected. Both of these should be produced directly while using Visual Basic, not typed or hand written later.

### 15.4 Starting Visual Basic

Visual Basic 6.0 is a program designed to run under the Windows NT operating system. On the networked computers in Cluster Rooms therefore you should log in and proceed to the WTS service, as you did for word-processing, where you will find it listed in the Start menu under Programs > Visual Studio.

You can also use Visual Basic on the Physics Laboratory computers.

See section 15.7 for further details of running Visual Basic.

**CAUTION:** The Visual Basic program, Windows operating system and the many other programs and data files necessary for the operation of the computer are all stored on the computer's internal "hard disc" (disc drive C). Please do not try to access this disc other than to run Visual Basic. In particular do not alter any of the system configuration settings and do not copy any programs from the hard disc. If you place any of your own files or programs on this disc they may disappear without notice since the hard disc contents are checked and cleaned up regularly.

## 15.5 The Keyboard

There are a few more special keys on the keyboard that you will need to know about in addition to those you have used for word-processing:

The **Break** key at the extreme right of the top row of keys. This is always used in combination with the **Control** key (bottom left). It forces your program to halt prematurely.

The **End** and **Home** keys. These are very useful for quickly jumping to the end or beginning of a line of program code.

The **Escape** key (top left) is handy for getting yourself out of situations where you don't know what's going on!

The **F1** function key (to the right of the **Escape** key) is good for quickly getting help on a particular topic without having to navigate through the complicated Help hierarchy.

The **underline** character "\_" is obtained by pressing hyphen "-" between "0" and "=" in the main block of the keyboard whilst holding down the Shift key. It is used at the end of a line of program code, preceded by a space, to continue a long statement onto the following line and can also be incorporated into variable names.

## 15.6 What Is a Computer Program?

The traditional answer to this question is that a computer program is a list of instructions to the computer, called *statements*, arranged in the form of lines of text. In Visual Basic a *program line* and a *program statement* are generally the same thing. When typing in a program line there is no "word-wrapping" as there is in a word processor. You terminate each line by pressing the **Enter** key.

[It is possible to have more than one statement on the same line, separated by colons, but this is not recommended (see Unit 5 for an example). It is also possible to spread one long statement over more than one line by using the underline character, but avoid doing this where possible.]

The computer generally carries out these instructions in the order that they are listed, beginning with the first line (although this can be overridden when required by special "branching" instructions).

Each program line has to be composed according to the rules of the programming language (*Visual Basic* in our case) and contains certain elements which are:

- Statement keywords
- Constants
- Variables
- Operators
- Labels

... separated from each other using particular characters, such as space, comma, semicolon, etc., called *separators*.

[Note: Visual Basic does not use line numbers as many older versions of Basic do.]

It is essential to accurately abide by the syntax rules of the language in order that Visual Basic interprets the program correctly, and in particular so that it can identify these different elements within the line.

Here is an example of a program line:

```
Print "The result of multiplying that number by 4 is " ; a * 4
```

Here `Print` is a *statement keyword*: it tells Visual Basic what kind of action is to be performed, in this case the placing of some text in the results display area (the “form”).

`"The result of multiplying that number by 4 is "` is a *constant*: it tells Visual Basic explicitly what is to be placed on the display – this text lying within the quotation marks.

`;` is a *separator*: it simply separates one item in the program line from the next item.

`a` is a *variable*: a symbol which has a value that is not directly expressed within the program line (its value will have been assigned by an earlier statement in the program).

`*` is an operator: this one means “multiplied by”.

`4` is another constant: explicitly the number four.

Notice that Visual Basic automatically gives anything it interprets as a statement keyword an initial capital letter and anything it interprets as the name of a variable it leaves as you typed it. You should always type your program using a lower case letter for the first character of any word and let Visual Basic set the initial letter of a keyword to upper case. This way you will know which words are being interpreted as keywords.

So far our definition of a Basic program would apply to pretty much any implementation of the language going back as far as the early 80s. However Visual Basic provides additional means to design your program which do not necessarily need lines of program code. This part of the design process is done within the Form Layout window using the Form Properties window and the Object Toolbox described below. It enables you to set up the way your program will look to the user when it is run by adding any type of “window” or “dialogue box” equipped with buttons, menus and list boxes and so on, all of which are termed *objects*. Some of these objects are invisible to the user, such as timers which can cause things to happen after a predetermined delay or repeat with a specified period. You can even create your own objects (but that does involve writing more program code: it’s called “object oriented programming”).

This aspect of program design might seem to be more fun than simply writing lines of code and for this reason most books on Visual Basic concentrate on these *objects*. However actually writing a program that uses and controls the objects properly is a very involved business and you can expend a lot of time getting nowhere when all you really want is to write a program to do some simple numerical computations. In this course therefore we will not worry too much about the *look* of our programs but just get things working in the most straightforward way possible.

## 15.7 Visual Basic's Windows and Menus

### BEGINNING A NEW PROGRAM

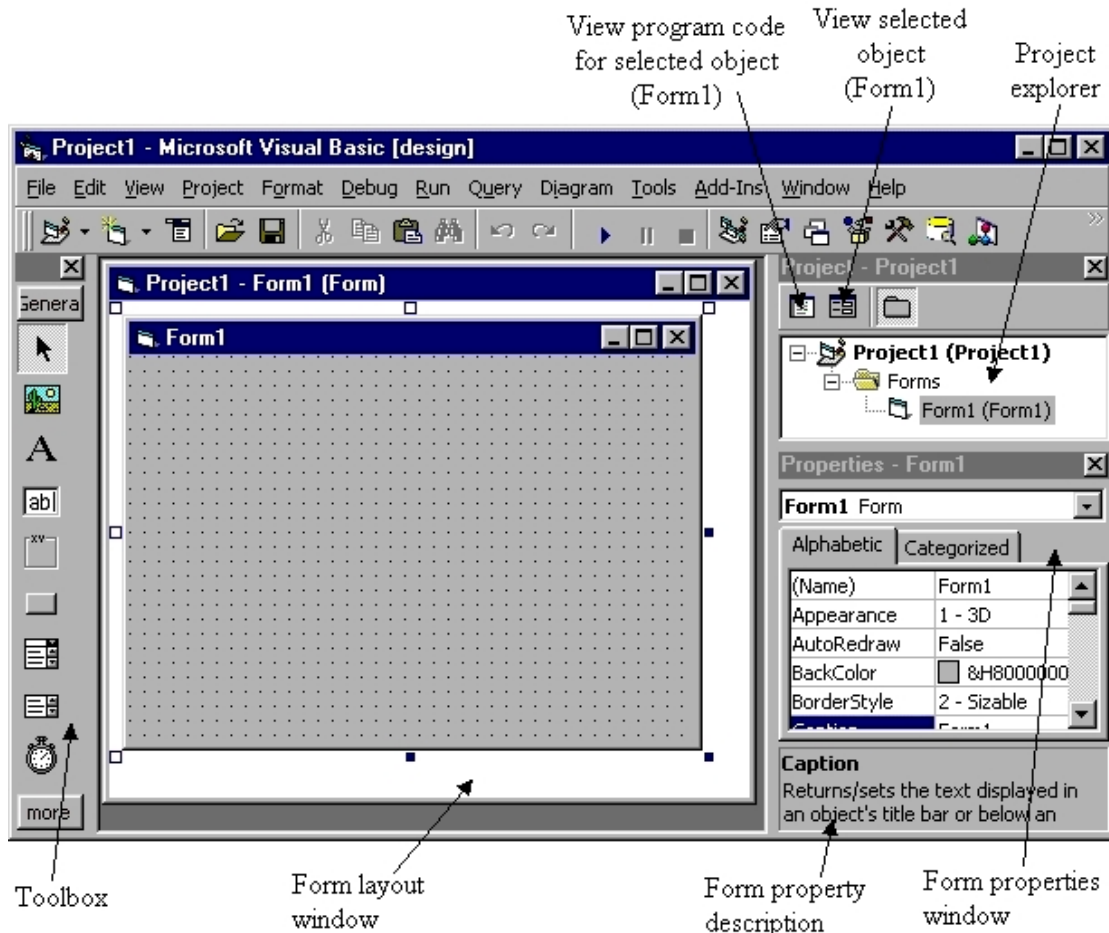
When Visual Basic is first started you see a *Dialogue Box* providing you with a choice of how to begin. There are three “tabs” to choose from: New, Existing and Recent. On the *New* tab you can choose to begin a new program of various types: select *Standard.EXE*. On the *Existing* tab you are presented with the usual Windows dialogue box for opening a file: this is only needed if you want to work on something that you haven’t worked on for some time or a program written by someone else or if you have changed your floppy disc. To continue working on a program you have worked on recently, insert your floppy disc and select the *Recent* tab when you will see a list of these programs with their names and the folders where they reside. Select the one you want and then click Open.


Note that within the Visual Basic programming environment the program that you are creating is referred to as a *project*. A project can have many sub-components or “modules”, all contributing to the final program. We will work with the minimum of modules. Unfortunately, this flexibility in Visual Basic does make the task of keeping track of your program files rather tedious. There is not just one file for each project but a minimum of three. Moreover, it is quite tricky to keep control over the naming of these files. For this reason you are strongly recommended to follow these two rules:

- A Always give each project/program a unique name;
- B Always save the files for that program in its own folder.

We will look at this in more detail shortly.

Once you have opted to start a new Standard.EXE project you will see the normal Visual Basic working window. Below is a picture showing roughly what this looks like.



You might also see an Immediate Window just below the Form Layout Window. This can be closed to make more room for other things by clicking the usual Close Box  since we will not be using it very often.

#### OBJECT TOOLBOX

This is for adding various user interface objects to your program such as buttons, pictures, text boxes, etc. We will not be using the Toolbox so you can close this also to make more space on the screen.

#### FORM LAYOUT WINDOW

This shows the “object” that will appear when your program is run. It is called a *Form*. You can have more than one form within a project but we will stick to just one which is always called “Form1”. You can change the shape and size of Form1 by dragging on the small boxes round its edges. Try this now.

Note: Form1 is one of the files that will be saved as part of your project. That *file* can have any name you chose, but within the project it is always called “Form1”.

#### PROJECT EXPLORER

This lists all of the modules within the project. You begin with just one – Form1 – and we will stick to just this one. Whenever you wish to manipulate the shape of Form1 or change its properties (such as colour) or write some program code for it, make sure that Form1 is *selected* in the Project Explorer window as shown above: it should have a grey background after you click on it.

Try now clicking on “Project1(Project1)” and then on “Form1(Form1)” and note the effect on the Form Properties Window below and on the View Program Code and View Object buttons to the left.

Notice the way that Form1 is named: “Form1(Form1)”. The first part of this name is the name of the form *within the project*. The part of the name in brackets is the name of the *file* that you have saved it as which will change when you save you project for the first time and choose that name. If you are lazy and let all of your Form1 files for different projects have the same file name – “Form1” – then it is extremely likely that you will accidentally use the form from one project as the form in another project and get very confused. So don’t be lazy or you’ll pay for it!

#### FORM PROPERTIES WINDOW

This shows the settings for all of the properties of the Form. There are a lot of them, so don’t worry about them to begin with. Only change something if you have a good reason for wanting it changed.

#### FORM PROPERTY DESCRIPTION

When you select one of the properties in the Form Properties Window a brief description of the purpose of that property appears in the space below the property list. The property selected in the illustration above is “Caption” and you can see that this is the text that appears in the blue bar along the top of Form1. You may wish to set this to something more meaningful such as a name for your program.

Try now clicking on a few of the properties listed in the left column and read the property descriptions that pop up below.



## VIEW PROGRAM CODE / VIEW OBJECT BUTTONS

These two buttons determine what is displayed in the Form Layout Window. If you click the “View Program Code” button then a window appears into which you can type the *code* for your program, i.e. the program instructions.

Try now clicking on each of the two buttons in turn.

## PROGRAM CODE WINDOW

Typing in a program is a bit like entering text into a word processor except that you enter a program *one line at a time*, pressing the **Enter** key to signal the end of each line. When you do this, or when you modify a line and move the cursor off that line, Visual Basic checks the syntax of what you have typed and alerts you to anything which is obviously incorrect.

The code that you type in is associated with the Form1 object and there are several *standard procedures* associated with a Form where code can be placed (see next paragraph). The code for any given standard procedure will be carried out depending on whether particular events have happened. This is called “event-driven programming”.

You will see two drop-down list boxes along the top of the code window, the left one initially showing “(General)” and the right one showing “(Declarations)”. Click to the right of the left box and you will see “Form” listed. Click on Form and the “Private Sub Form\_Load()” procedure will appear in the code window. This is one of many standard procedures that are created automatically. Now click to the right of the right list box (currently showing “Load”) and you will see a long list of all the other standard procedures for a Form.

You can create your own procedures and you will do this extensively in your Level 2 course but this year we will work mainly with this standard Form\_Load procedure.

This procedure is executed for Form1 whenever your program is run (this is its “event”) so it’s a good place to put your program code. Note that any procedure begins with a line something like

```
Private Sub xxx()
```

and ends with a line like

```
End Sub
```

where “xxx” is the *name* of the procedure.

The program code for that procedure is typed in between these two lines.

[“Private Sub” is one of several different types of procedure!]

When you have typed in a complete program you can tell Visual Basic to “execute” the program using the **Run** menu. Visual Basic will then carry out the instructions of your program in the order in which they are listed on the screen (which may not be the same as the order in which you entered the lines). We will be writing programs that produce results as text or data that appear on Form1.

We will leave actually typing in program code until we get to the Unit 1 exercise.

## MENUS

Visual Basic has many menus and they are very long ones! You therefore need to focus only on the menu items that you need and ignore the rest.

## FILE MENU

The items that you will need are

“Save Project As” to save a new project for the first time and give it a unique name;

“Print” to print out the program code you have written or the content of Form1 after the program has run..

## VIEW MENU

Use this to restore any of the windows that you had previously closed. Also when/if you become very proficient you may like to take a look at the Object Browser as a way of discovering more about the possibilities within Visual Basic.

## RUN MENU

Select “Start” to execute your program. Also if you make your program prematurely halt using the Break key or if it halts because of a programming error you can select “End” to reset Visual Basic to its normal state.

## HELP MENU

Obviously this is to get help with the Visual Basic programming language. Unfortunately the Help facility is set up in a way that is shared with several other Microsoft programming applications so it’s rather laborious finding what you want. Probably the most useful route through the hierarchy is

Contents > Visual Basic Documentation [click the plus sign] > Reference > Language Reference > Keywords

then click on the keyword that you want to know about.

“Search” sounds as though it would be helpful but it usually lists so many references that you can’t possibly look at them all.

“Index” is more useful. You can switch between Contents, Search and Index while in the Help facility using its own View menu.

*Read the following two sections but do not bother yet about trying out the things they describe. Start the Units and return to these sections when you need to know about using discs or the printer.*

## 15.8 Use Of Disks

You will be recording your programs on the floppy disc provided. Note that it already contains a folder containing files that are required for Unit 2 of the programming course.

### OPERATIONS WITH PROGRAM FILES ON DISC

To **place a copy of your program onto the floppy disc** use the **Save Project As** command from the **File** menu. The usual Windows *dialogue box* appears. Perform these steps:

- A. Navigate to “Floppy disc A” using the “Save in” drop down list at the top of the dialogue box.
- B. Create a new folder by clicking on the middle folder symbol to the right of the “Save in” box. Type the name of the folder where it says “New Folder”, then double click the folder to open it. The name you choose should preferably contain your initials and a suitable name for this program./project.
- C. Delete what appears in the “File name” box towards the bottom of the dialogue (“Form1.frm”) and type in the chosen program name. Click Save.
- D. Delete again what appears in the “File name” box (“Project1.vbp”) and again type in the chosen program name. Click Save again.

You are now returned to the main Visual Basic window. Check in the Project Explorer window that both Project1 and Form1 have the correct file names that you chose.

Note: In the top right-hand corner of the floppy disc is the **write-protect tab**. This should normally be closed. However if you wish to prevent any modification to the contents of the disc you can open the tab by sliding it across on the under side of the disc, when the computer will be prevented from writing to the disc. The files are still readable though; i.e. Open will work, but Save will not.

To **save your program once you have done some work on it** so as to make sure the copy on disc is up-to-date simply click on the usual Save icon in the main Visual Basic Toolbar (*not* the *Toolbox!*). This is usually located roughly underneath the Format menu.

To **save a second copy of your program with a different name** you must save both the project and Form1. First use **Save Project As** in the **File** menu to save the project under a different name in a new folder, as described above. Then select **Save form As** in the **File** menu and save Form1 with the same new name in the same folder that you just saved the project files. [Here *form* in the **Save form As** command will be the current file name for Form1.] It is very important that you perform these steps in this order. If you were to save Form1 first and then save the project you would end up with two different projects sharing the *same* Form1; in other words they would be the same program!

## 15.9 Use Of The Printer

Anything which you send to the printer should have your name on it somewhere so that it can be separated from the printing being done by other users of the same printer. All printers in the college are shared amongst several computers.

### TO PRINT YOUR PROGRAM ON THE PRINTER

Use the **Print** command from the **File** menu. “Range” should be set to “Current Module” and “Print What” should be set to “Code”.

### TO PRINT OUT THE CONTENT OF FORM1 UNDER PROGRAM CONTROL

Use the **PrintForm** statement in your program. This should be followed by a **Cls** statement to clear Form1 before writing further output onto it. PrintForm will only work properly if the AutoRedraw form property is set to “True”. Use PrintForm sparingly, especially when you are working in a cluster room.

## TO PRINT OUT THE CONTENT OF FORM1 BY INTERRUPTING THE PROGRAM

If you have not yet included any **PrintForm** statements in your program or if you wish to print out Form1 at a stage in the program where it would not normally be printed, you can break into the execution of the program, print the form, and then resume the running of the program from where it left off.

Make sure that the Immediate Window is displayed (use the **View** menu). To interrupt the execution of the program press the Ctrl/Break key combination. Then click in the Immediate Window and type

**PrintForm**

and press the Enter key. The current content of Form1 will then be sent to the printer.

If Form1 does not somewhere have your name on it (for identification purposes) then it is a good idea to type a **Print** statement into the Immediate Window to print your name on the form before using **PrintForm**.

It is also a good idea to now type **Cls** into the Immediate Window to clear Form1 before resuming execution of the program so that there is room for more information to be written there.

To resume the program select **Continue** in the **Run** menu.

## 15.10 Using Visual Basic 5 CCE

Visual Basic 5 CCE is a slightly earlier version of Visual Basic which is freely available and provided for you on the CD. It lacks Help files and you cannot create an executable binary file, but otherwise it is very similar to the full version 6.0. You can run it directly from the CD or install it on your own computer.

To run it from the CD, open the folder called *Visual Basic 5 CCE* and double click the file called “vb5cce” or “vb5cce.exe”.

To install the program on your computer, open the folder called *VB5CCE installer* and double click the file called “vb5ccein” or “vb5ccein.exe”.

Remember when you begin a new program/project to select “Standard.EXE”. This is not the default selection with Visual Basic 5 CCE as it is with version 6.0.

# 16 Visual Basic Course-work Problems

## Unit 1

**Deadline for completion and handing in: Thursday 21 November**

**In this first Unit you will not be using the printer nor saving your work to floppy disc.**

**Write down your answers on a piece of paper and hand this in.**

You will be learning about using Visual Basic's windows and menus, entering and running a program, editing, numerical and string constants and variables, the **assignment (Let)** statement and the **Show, Print** and **Cls** statements.

Read the general notes on Basic programming in sections 15.1 – 15.7 before attempting this Unit, in particular section 15.6 *What Is A Computer Program?*

Some of the concepts mentioned there will only become clear once you start to use them in writing programs. However for reference purposes they are each briefly described here before going on to try some examples of writing short programs.

### **HOW DATA IS HANDLED WITHIN A PROGRAM**

Data is in one of two basic forms. A *constant* is a value that is explicitly typed into a program and forms part of it. A *variable* is a symbol which does not have any fixed intrinsic value but it can be given a value. A particular variable can be repeatedly used throughout the program, so allowing values of data to be carried from one part of the program to another. This is quite like the use of variables in algebra. However it is important to be aware that the value of a variable will change as the program proceeds whenever a program line is encountered that affects it. This is different to a mathematical development where an “unknown” is taken to have the same value throughout.

### **Numerical and String Constants**

Within a program (or when typing a command into the Immediate Window) a *numerical constant* is represented in much the same way as you do when using a calculator using the symbols “-” “.” and “E”. e.g.:

324  
-1.345  
68.4E-6

A sequence of characters which you might want to appear on the Form is called a character string or *string constant*, and it is enclosed in double quotation marks:

**"These characters form a string, excluding the quotation marks!"**

[Note: these quotation marks are produced by a single press of the “2” key towards the top left of the keyboard while holding down the Shift key.]

The quotation marks distinguish a character string from the other forms of text which go to make up a program, such as *numerical constants* (see above), *variable names* (see following section), and *statement keywords* – e.g. Clr, Print – which serve to tell the computer what to do.

## Variables

A number or a character string may be stored in the computer's memory at some point in a program for later use at some other point in the program by means of assigning its value to a *variable*. A variable is signified by a sequence of letters or figures (without any spaces between, and beginning with a letter, and not enclosed in quotation marks). This is called the variable's *name* which you can choose to be whatever you like within these syntax rules. The underline character may also be used in a variable name, e.g. `next_one`, but not as the first character.

[In Visual Basic you do not need to use different styles of variable names for different types of values (numerical, string, etc) as you do in some versions of Basic.]

To indicate the operation of **assigning** some value to a variable an **equals sign** is used:

```
xd =14.1  
animal = "Cow"
```

The variable called *xd* now has the numerical value 14.1, and the variable *animal* has the value "Cow" as a string of characters: C-o-w. Further assignment statements at other points in the program may change the values of these variables to something different.

## BASIC STATEMENTS USED IN THIS UNIT

*Statements* are composed of one or more keywords. The syntax rules for these keywords are the same as the rules for naming variables. For this reason you cannot have a variable with the same name as a statement keyword, e.g. **Name** is a keyword and so must not be used as a variable name. In order not to have to be constantly referring to a book or the help facility to see whether your chosen variable names clash with keywords, always type variables in lower case and check that Visual Basic leaves them in lower case, which indicates that you have used an acceptable name (keywords have their first letter automatically put into capital letters). Make sure that the keyboard has **caps lock** turned off.

### The Show statement

**Show** is a particularly simple statement because it is composed of only a single keyword and does not require any other elements to make a complete program line. Its effect is to make the text or images written to Form1 visible to the user when the program is run. Without it the form will appear blank. It is usually the first statement in the program.

### The Cls statement

The "clear screen" instruction, **Cls**, is also composed of only a single keyword and does not require any other elements to make a complete program line. Its effect is to clear Form1 of all content, i.e. to return it to its initial blank state.

### The Print statement

When followed on the same line by a list of constants or variables separated by commas or semicolons the keyword **Print** causes the values of those constants or variables to be written on Form1, all on one line and in the order of the list.

The choice of a comma or a semicolon to separate the items in the list affects the spacing between their values as they appear on the form (see details later).

The location on the form where the computer is placing the current character is called the *cursor position*.. After a **Print** statement has been executed the computer moves the output cursor on by one line, so if you enter the **Print** statement without a list of items the effect is simply to move the cursor down the form by one line, hence creating a blank line. If, on the other hand, your **Print** statement ends with a comma or a semicolon then the cursor is not moved down to the next line and any subsequent **Print** statement will cause the new characters to be appended to the end of this line.

[Note: the word “print” is used in Basic to signify placing output on a form as well as printing on paper using a printer.]

### ***THE ASSIGNMENT FOR THIS UNIT***

In this unit you will not be writing any serious programs, just trying things out, so you will not need to save your work on floppy disc. Start Visual Basic in the way described in section 15.4.

The following notes tell you exactly what to do. In later Units you will be given less specific instructions because it will be assumed that you have understood the material contained in the previous Units. It is therefore very important that you study this Unit carefully so that you do understand everything and do not just write things down without thinking.

### ***NOTE:***

**For this Unit you are required NOT TO USE THE PRINTER:  
write down everything by hand**

### **CARRY OUT THE FOLLOWING ACTIONS labelled A to F.**

You should write down your results and observations by hand: do **not** use the File/Print menu to produce your material for handing in for this Unit.

- A.** Within the Form\_Load procedure in the code window type the following lines, pressing the Enter key at the end of each:

```
show  
print “hello there!”
```

You have now created a program. You should see that as you pressed the Enter key Visual Basic turned the word *show* into Show and the word *print* into Print. This signifies that they were interpreted as statement keywords. The characters “hello there!” however remained as you typed them since they were interpreted as forming a string constant.

This program line will cause the value of the string constant “hello there!” to be displayed on Form1 when you run the program.

Run the program by selecting the top item in the **Run** menu (**Start**).

Even though both of the statements in your program have been executed, the program has not yet finished: it is still displaying Form1. To make it finish, click on the close box of the form in the usual way and this will return you to the code window.

Write down briefly what you have done, *precisely* what the result was, and why.

Now change the characters of the string constant between the quotation marks in some way. You might make the phrase begin with a capital letter or change the word “*there*” to your own name. Run the program again.

Write down briefly what you have done, *precisely* what the result was, and why.

- B.** Now we will modify the program to use a variable which we will call *test* and give it a specific numerical value. At the beginning of your program insert the line

```
test=57000
```

Also change your Print statement to have the form

```
Print test
```

Note there are no quotation marks this time.

Run this program and observe the differences to what happened in **A**.

Write down briefly what you have done, what the result was, and why.

- C.** Now we will try out the effect of the Cls statement. Insert a line at the beginning of your program containing just the Cls keyword. Run the program again. Now move the Cls statement forward by one line and run the program again.

[To quickly move this line, select it by clicking in the white space just to the left of it, then place the mouse pointer over the selected word and drag it down to the beginning of the line where you want it to move to.]

Move Cls forward one more line and run the program. Repeat until you have tried the program with Cls at every possible position in the program.

Write down briefly what you have done, what the results were, and why.

Delete the Cls statement.

- D.** Insert two more lines at the beginning of your program:

```
b=3
```

```
say="The value of A is "
```

and modify the Print statement to look as follows:

```
Print test, b, say
```

Run the program.



Write out your complete program as it now stands and carefully copy down what appears on Form1.

Explain which elements of your program have produced each of the groups of characters on Form1.

Measure (or estimate) the distance from the left edge of the form to the beginning of each group of characters on Form1 (not the distance *between* groups) and indicate this in your report.

[You can do this holding a piece of lined paper sideways to the screen and marking the edge of the form and the start of each group of characters.]

- E. Now modify the Print statement again to look as follows:

```
Print test, say; b; "B="
```

Notice the use of semicolons in some places in the Print list instead of commas.

Run the program.

Write out the Print statement as you currently have it in your program and carefully copy down what appears on Form1.

Measure the distance from the left edge of the form to the beginning of each group of characters and the distance *between* each group and indicate this in your report.

Compare these distances with those found in D..

Describe the difference between using commas and semicolons as separators in a Print list.

State whether each element in the Print list is a variable, a numerical constant, or a string constant.

If you are unsure what the difference is between using a comma and a semicolon, experiment with changing them over at different parts of the Print statement. You could also try consulting the Help facility. To go straight to help about the Print statement, place the cursor within the Print keyword and press the F1 key. You will at least learn how arcane the Visual Basic Help facility is!

- F. This part tests whether you have properly understood everything described so far. Based on what you have learnt so far:

rearrange the items in the Print list using commas or semicolons as separators where necessary so that when the program is Run the following output is produced:

```
The value of A is 57000    B = 3
```

N.B. Do not introduce any new elements into the Print statement: use only the ones already encountered in previous parts of the exercise.

(Write down your answer!)

If you are not able to do this part then ask for guidance or go through the previous parts again.

### **Finishing off**

When you have finished the above exercises and you are happy that you have accurately written down what is required and understood the roles played by the various elements of the programs you have written, leave Visual Basic by selecting **Exit** from the **File** menu.

A dialogue box will appear asking if you wish to save the current program files onto a disc. In this Unit we are not using any disc, so click **No**.

You should now find that you are back at the Windows desktop. On the Network machines log out before leaving (never switch the computer off). On the Physics Laboratory computers you do not need to log out. Leave the computer switched on during normal hours or “Shut Down” and switch it off at the end of the day.

## Unit 2

### Deadline for completion and handing in: Thursday 28 November

Be sure to have read the general notes on Basic programming in sections 15.8 – 15.9 before studying this Unit. Read ALL of this Unit before attempting the problem (outlined in a box).

In this and future Units we will be writing text onto Form1 in a more controlled way and also sending the form to the printer. For these reasons we need to change four of the Form1 properties from their “default” values in order that things will work as we require. These four properties are:

AutoRedraw = True	needed in order that PrintForm should work properly,
BackColor = white	needed in order that the form is printed clearly without any shading,
Font = ???	not vital, but you may like to try the effect of changing font size,
ScaleMode = Character	needed to make controlling the print cursor more straightforward.

To find out more about each of these form properties select the property in the Object Property window and press the F1 key.

These properties, except for Font, are set up correctly in the example program which is provided on the floppy disc in the folder U2TEMPL for use in this Unit (see end of the notes for this Unit). The reason that Font is not set in any special way is that the available fonts vary from one computer to another.

If you are not going to use the example program then you will need to set these properties. Click in the column to the right of the property name in the Form1 Property Window when you will either be presented with a drop down list of choices or a dialogue box for more complex selections. When setting the background colour (BackColor) you can either choose one of the standard window colours or select from a “palette” of colours.

### NEW BASIC STATEMENTS USED IN THIS UNIT

#### **The Rem statement has the general form:**

*Rem any text you wish to include as a comment*

OR

*' any text you wish to include as a comment*

[When describing the general form of a program statement, *italic text* is used to describe the nature of an item which does not have a single fixed form.]

Use this to put comments in your program. Such comments have no effect on the way the program works, they only manifest themselves when you view or print the program. They are for the benefit of yourself and anyone else who needs to inspect your program and try to understand it—such as myself. They are NOT for the benefit of the *user* of the program who should not be required to study the program before being able to use it. Any information for the user should be given in Print statements.

In Visual Basic the **Rem** keyword may be abbreviated by an apostrophe '. Such comments should include, at the beginning of your program, your name, the date you finished the program, the name of the program (Unit 2) and a brief description of what the program does. Usually there is no need to duplicate in a **Rem** statement information that is already in the program as part of a **Print** statement. Think carefully about the difference between the **Rem** and **Print** statements and use them appropriately: they are very different.

### **The InputBox function:**

The **InputBox** function is used to give a value to a single string variable whilst the program is running, this value being chosen at that time by the *user* of the program who types the characters for the string variable in at the keyboard. This is very different to the assignment statement used in Unit 1 which gives a value to a variable dependent on constants and other variables which are already within the program.

It is used in an assignment statement with a single string variable as follows:

```
string variable = InputBox ("a message")
```

Because it is used in the form of an assignment statement **InputBox** is known as a *function*. Notice the need for brackets to enclose the parameters of a function.

When this statement is executed in a program a dialogue box appears containing the *message* followed by an empty text box where something should now be typed in from the keyboard. There are also two buttons in the dialogue: "OK" and "Cancel". Execution of the program pauses until this is done and one of the buttons is clicked. Whatever is typed at the keyboard — including commas, quotation marks and other punctuation marks — is interpreted as the new value for the string variable. Execution of the following lines of the program then resumes.

The *message* is called a "prompt". Usually the message should tell the user what to type and in what format and then what to do, e.g. "Please type in your full name and click OK: ".

Note: With the **InputBox** function the user does not need to enclose the typed-in text within quotation marks, even though it is being interpreted as a string.

### **The MsgBox statement.**

This statement is similar to the **InputBox** function except that nothing is typed in by the user. It is used simply to convey information and wait for a user to respond by clicking a button.

```
MsgBox "Click OK to continue"
```

### **The CurrentX and CurrentY properties and the Tab() function**

Statements using these keywords are used to affect where on Form1 text will be placed when the next Print statement is executed. The choice of method is a matter of aesthetics and convenience.

To make one blank line it is often simplest to use a **Print** statement with no list:

```
Print
```

To cause the text to be printed away from the left edge of the form, leading commas are a tidy way provided it is not critical exactly where the text appears. Each comma moves the output cursor to the next multiple of 14 standard character widths from the left edge:

```
Print , , , "Results"
```

will cause *Results* to appear beginning at column 43 (=3×14+1).

Note that “standard characters” will probably not have the same size as the characters your program is printing. This will depend on the font used which is set in the Font property for Form1.

More accurate control of the output location is achieved using **Tab()** which works only when included in the list of a Print statement:

```
Print Tab(12); “ ABC”
```

will cause *ABC* to be displayed 12 standard character widths from the left edge of the form. Note that it is usually separated from the next item in the list by a semicolon rather than a comma, as shown above, in order to function as desired. You cannot make the cursor go backwards using the **Tab** function.

The two “properties” **CurrentX** and **CurrentY** allow even more accurate control of the Print cursor. They are if you like standard system variables and are assigned values in the same way you assign a value to any variable. However since they relate to specific properties of a particular object (Form1) they can only be assigned values from a particular allowed set, otherwise an error is caused..

**CurrentX** and **CurrentY** specify the coordinates of the lower left point where the text will begin, measured from the left and top edges of the form respectively. They can be given non-integer values. The units in which those coordinates are to be measured is set be the property ScaleMode. ScaleMode can be set using a program assignment statement but it is usually more convenient to set it in the form property window. To find out the allowed values for ScaleMode and what those values correspond to, select ScaleMode in the property window and press the F1 key.

#### **The PrintForm statement**

This is a one word statement requiring no parameters and it simply causes the current content of Form1 to be sent to the printer. It will only work properly if the AutoRedraw form property is set to “True”. Only include this in your program once it is working properly, otherwise you will generate a lot of waste paper and waste a lot of time.

***ON THE FLOPPY DISC PROVIDED*** there is an example program in the folder **U2TEMPL** (see end of the notes for this Unit).

**Read section 15.8 if you have not already done so and then Open this program in Visual Basic and study it with reference to these notes.**

**Save it with some different name (being careful to give new names to both the Project and to Form1) and then Run it to see the effect of the various statements.**

There is a small “bug” in this program, i.e. it does not do quite what it says it is supposed to do. Try to find the error and correct it.

Now create from scratch *or* modify the example program in order to ...

**WRITE A PROGRAM** using the Rem, Print and Cls statements and the InputBox function (and optionally MsgBox, the Tab function and the Form1 properties CurrentX and CurrentY) which

- requests from the keyboard any address (in unspecified format) of up to 8 lines, and then

- clears the screen and
- displays the address on the form

laid out in the manner used to address an envelope, i.e. spaced away from the top and left edges to be roughly in the middle of the form.

NOTE: This means any address that the user wishes (which can be different each time the program is run without having to modify the program itself), not just the examples given below and not just *your* address or the user's address any other specific address. If you find that you are typing a specific address into your program as you write it then you have not understood what is required here: [ask for help](#).

Use the Rem statement to **title the program** and to **include your name** and **which Unit** the program relates to. Also use Rem to explain how the various parts of your program are supposed to function. You will not get good marks for your work unless you do this.

**Use only those elements of the Visual Basic language that have so far been covered in the course. This rule applies to all of the Units.**

**If you are using the example program be sure to delete all of the Rem statements that begin '\*\*\* when you have finished and check that you have edited all of the other Rem statements to be appropriate to YOUR program.**

One tricky aspect of this programming task is to cater for addresses of differing numbers of lines.

e.g. can your program cope with each of the following:

Finance Office  
University College London.

*and*

John Smith esq.  
Flat 123  
Posh Mansions  
94 High Road  
Bethnal Green  
London  
E2 3PQ  
UNITED KINGDOM

Some people like to use commas in addresses and put names of buildings in quotation marks: e.g. the second example above might be written

John Smith, esq.  
:  
"Posh Mansions"  
:  
:

Find out how your program copes with such cases and demonstrate your findings by handing in various example printouts. However do not go to great lengths to try to make your program cater perfectly for all conceivable cases but do see if you can make your program **usable** despite any such shortcomings by providing sufficiently detailed instructions to the user using Print statements.

**SAVE** your program on a disc **BEFORE YOU RUN IT**. The reason is that it is always possible for a program to become locked into a condition which can only be escaped from by using the special key combination **Control/Break**, and if you can't make this work the only way to stop the program is by switching the computer off which will cause your program to be lost. In general it is a good idea always to save any program you have just written or modified before you run it.

**In all programming problems a significant amount of credit is given for clear but concise comments using Rem statements and for helpful information and instructions to the user using Print statements. Pay careful attention to these aspects as well as just getting your program to “work”. (Read section 9.12 and Appendix C for more information.)**

**In general you should hand in a computer-generated listing of your programs and also samples of the output that it produces. This needs the use of the printer. Use the Print command in the File menu to make a printed copy of your program (see section 15.9).**

### **Hints**

If you are not sure how to go about modifying the example program you might take it in stages as follows.

- 1 Try the effect of enabling and disabling the Cls statement and decide what works best.
- 2 Change the CurrentY value and the Tab() value until the output is at the required place on the screen.
- 3 Change the Rem and Print statements and the InputBox prompt message to suit your program, which up to this point deals with only a single line of input text.
- 4 Add one more InputBox statement, one more Print statement, and possibly one more CurrentY assignment, so as to make you program deal with two lines of input. Note: each InputBox statement will need its own string variable.
- 5 Complete the program so that it can deal with eight lines of input.
- 6 Consider how you should improve the Print statements that are giving information to the user.

*Here is what the example program looks like:*

```
Private Sub Form_Load()
' Unit 2 -- example program by Malcolm Coupland. 8/9/2002
'*** Change the above lines to contain your own name
'*** and a brief description of YOUR program
'
'*** DELETE ALL LINES BEGINING WITH '*** WHEN YOU HAVE FINISHED
'*** YOUR PROGRAM
'
Show
'
'*** Put your name on Form1 for identification purposes:
Print "MALCOLM COUPLAND"
Print
'
' Tell user what the program does ...
'*** Modify the following lines to suit your program.
'*** Insert extra Print statements if needed.
Print "This is the example program for Unit 2:"
Print "You can enter one line of text, then the screen is cleared"
Print "and the line is displayed 4 spaces in from the top and left"
Print "edges of the form."
'
' Send form to printer
'*** Once your program is working properly you can print out the form
'*** by removing the apostrophe from the following line.
'PrintForm
'
' The next line cause a pause until a button is clicked
MsgBox "Click OK to continue"
'
' Give instructions to user, and accept data from keyboard ...
'*** The following line allows just one line of text to be entered.
'*** You will need several lines similar to this in order to allow
'*** several lines of address to be entered
line1 = InputBox("Type in the line of text, then click OK")
'
'*** You can enable the following line if you wish by
'*** deleting the apostrophe at the beginning of the line:
' Cls
'
'*** Put your name on Form1 for identification purposes:
Print "MALCOLM COUPLAND"
Print
'
' Position cursor and display entered text ...
CurrentY = 4
'
'*** The following line displays the one line of text that has
'*** been entered. You will need several lines like this and
'*** possibly the one above to display all the lines of the address:
Print Tab(4); line1
'
'*** Once your program is working properly you can print out the form
'*** by removing the apostrophe from the following line.
'PrintForm
End Sub
```



## Unit 3

### Deadline for completion and handing in: Thursday 5 December

Make sure that the Immediate Window is open before starting this Unit. Use the **View** menu.

#### NEW BASIC STATEMENTS USED IN THIS UNIT

##### **The assignment statement (Let statement) and arithmetic expressions**

The general form is

LET *variable* = *arithmetic expression*

or simply

*variable* = *arithmetic expression*

The *arithmetic expression* is evaluated and the *variable* on the left is assigned the value of the result.

The simplest arithmetic expression is a single variable or a single constant:

w = x                    ...w takes the value that x has

v = 5.1                ...v takes the value 5.1

More complex arithmetic expressions can be constructed by the use of arithmetic operators:

+ - \* / ^ and brackets ( )

e.g.

w = 5.1 + (x - y^3) / (4 \* z)    ...w takes the numerical value of  $\left(5.1 + \frac{x - y^3}{4z}\right)$ .

##### **The Val() function:**

The **Val()** function provides one of several ways of properly entering numerical data into a program. The **InputBox** function we used in Unit 2 passes the input data to the program as a string, so if we type in a set of figures such as “1234” **InputBox** treats this simply as text, not as the number ‘one thousand two hundred and thirty four’. However Visual Basic is clever enough to automatically convert such a string into numerical form if it is used as a number, say in an arithmetical expression. That’s fine provided that the user doesn’t accidentally enter something that can’t be interpreted as a number: say the letter el “l” is used instead of the figure one “1”. Then when the program tried to apply an arithmetic operation to this “number” an error would occur and the program would stop. We need a way of forcing the conversion from a string into a number.

entry = “65”  
quantity = Val(entry)

Here the variable *entry* has a string value: the characters “6” and “5”. But the variable *quantity* has a numerical value: sixty-five.

entry = “K65”  
quantity = Val(entry)

Here the variable *entry* has a string value: the characters “K”, “6” and “5”. But the now variable *quantity* has the numerical value of zero since “K65” cannot be interpreted as a number.

This is not the ideal way to handle this situation because we would really like a way to *detect* that the user has not entered a valid number and so be able to ask them to enter it again. We will see how to do this in Unit 4. For now though we will simply use the **Val** function to make sure that no program execution errors occur.

## Labels and the Goto statement

*Goto label*

causes program execution to jump to the line specified by the *label* following the keyword **Goto**.

e.g.

**Goto Middle**

would cause the Visual Basic to jump to the line labelled Middle.

To set this label at some point in your program you would include at that point the line

**Middle:**

Note the colon which distinguishes the label name **Middle** from a variable name. The rules for label names are the same as those for variable names.

We will use **Goto** initially to make the computer go back to the beginning of a piece of program and so repeat that piece indefinitely.

**WRITE A PROGRAM** using the Rem, Print, assignment (Let), and Goto statements and the Val and InputBox functions which, when Run, accepts an arbitrary list of numbers from the keyboard one at a time and each time a number is entered evaluates and displays the following quantities up to that point:

- the value just entered
- the number of entries so far
- the sum of all entries so far

It should then request the next number in the list, and so on, indefinitely.

[Note that *the number of entries so far* and *the sum of entries so far* are running totals: the former increases by one each time a value is entered, the latter increases by the value just entered.]

### POINTS TO NOTE:

Remember to set the Form1 properties correctly as explained in Unit 2. In order to avoid having to do this every time you begin a new program you could create a blank Project that just contains a Form1 of the kind you need with all the properties set properly. Then to begin each new project first open this blank one and then save it under a new name as described in section 15.8.

Make sure that you test out the program as thoroughly as possible by entering a wide range of values: large and small, positive and negative, integer and non-integer, and zero.

**SAVE** your program on a disc **BEFORE YOU RUN IT**. The reason is that to break out of the loop of entering numbers you need to use the special key combination **Control/Break** and if you can't make this work the only way to stop the program is by switching the computer off, which will cause your program to be lost. In general it is a good idea always to save your program before you run it.

In order to send the results of this Unit to the printer you will need to interrupt the program using Cntrl/Break and type PrintForm in the Immediate Window as described in section 15.9.

Remember to observe the general points about the use of Rem and Print statements described at the end of Unit 2, as in all the Units.

**Hints**

If you have difficulty in working out how to write this program then build it up in stages.

- Begin with a program that simply asks for a number using the `InputBox` function, converts it to a numerical value using the `Val` function and Prints it on the form.
- Then make this program repeat this action in a never-ending loop.
- Then try to put in the part to keep a running total of the number of entries (this should produce the results 1 then 2 then 3 then 4 ... etc., increasing by one each time a new value is entered and printed out).
- Now include the piece to make the running total of the values entered.
- Now add appropriate `Rem` and `Print` statements.

## Unit 4

### Deadline for completion and handing in: Thursday 12 December

In this Unit we learn how to control the flow of a program—the order in which statements are executed—, how to test the value and type of a variable, and try more complex arithmetic expressions.

#### NEW BASIC STATEMENTS USED IN THIS UNIT

##### The **If . . . Then** statement

The general form of the **If . . . Then** statement is

```
If logical expression Then
    ⋮
    conditional Basic statements
    ⋮
End If
next line . . .
```

If the *logical expression* is **true** then the Basic statements following the keyword **Then** and up to the keyword **End If** are executed, and then the next line.

If the *logical expression* is **false** then the statements between **Then** and **End If** are ignored and the next line is executed.

##### Example:

```
      logical expression
      ┌───┐
If x < 0 Then
  y = x^2
  Print "x is negative..."; x, y
End If
Print "y has been calculated only if x is negative"
```

The general form of a **logical expression** is

*arithmetic expression relational operator arithmetic expression*

In the example quoted above, **x** and **0** are the two arithmetic expressions, and **<** is the relational operator. The available relational operators are

- > greater than
- < less than
- = equal to (not to be confused with the use of the same symbol in assignment statements)
- >= greater than or equal to
- <= less than or equal to
- <> not equal to.

For this unit we will confine ourselves to arithmetic expressions that are simple constants or single variables, as in the examples above.

An extension of this is the use of string constants or variables with string values in place of both arithmetic expressions; e.g.

```
If answer = "YES" Then
```

Here you must be careful to bear in mind that inside a string constant, upper case and lower case characters are distinct. So when `answer = "YES"` is **true**, `answer = "Yes"` will **not** be true. The order of significance of string characters can be found in most computing books in a table of “ASCII Codes” which gives the numerical code for each character used by the computer. The common punctuation marks come first, then the numerals, then the upper case letters, then the lower case letters.

### The Ucase function

The fact that a user might type in text in either lower or upper case, or a mixture, can be a nuisance when using an If statement to test the value of what has been typed. You can use the Ucase function to covert a string to all upper case.

e.g.

```
answer = "Yes"
a.caps = Ucase(answer)
⋮
```

The variable `a.caps` now has the value “YES”.

### The IsNumeric() function

This is a better way to check for numerical input than the Val function we used in Unit 3 but it requires an If statement, which is why we didn’t use it in that Unit.

**IsNumeric(variable)** evaluates as a logical expression. In other words its value is either True or False. It is True if the *variable* can be interpreted as a number and false otherwise.

```
entry = "65"
If IsNumeric(entry) Then
    Print "The data is numeric"
End If
```

The message “The data is numeric” will be printed.

```
entry = "K65"
If IsNumeric(entry) Then
    Print "The data is numeric"
End If
```

The message “The data is numeric” will *not* be printed.

### The End statement

So far our programs have not terminated completely, they have just gone into suspended animation when Visual Basic runs out of program lines. It has been up to the user to end the program by clicking in the Form1 close box. But you can force the program to end at any point by using the **End** statement.

### Examples of numerical and string logical expressions

TRUE:

<code>3 &gt; 2.6</code>	<code>-2 &gt; (-3.4)</code>	<code>"BAT" &gt; "BALL"</code>	<code>0 &gt; (-0.1)</code>
<code>2.6 &lt; 3</code>	<code>-3.4 &lt; (-2)</code>	<code>"BALL" &lt; "ball"</code>	<code>-0.1 &lt; 0</code>
<code>3 &gt;= 3</code>	<code>3 &gt;= 2.6</code>	<code>"Bat" &gt;= "Ball"</code>	<code>0 &gt;= (-0.1)</code>
<code>3 &lt;&gt; 2</code>	<code>-2 &lt;&gt; 2</code>	<code>"BAT" &lt;&gt; "Bat"</code>	<code>1 &lt;&gt; 1.000001</code>

FALSE:

<code>3 &lt; 2.6</code>	<code>-2 &lt; (-3.4)</code>	<code>"BAT" &lt; "BALL"</code>	<code>0 &lt; (-0.1)</code>
<code>2.6 &gt; 3</code>	<code>-3.4 &gt; (-2)</code>	<code>"BALL" &gt; "ball"</code>	<code>-0.1 &gt; 0</code>
<code>3 &lt;= 2.9</code>	<code>-3 &lt;= (-3.6)</code>	<code>"BAT1" &lt;= "BAT!"</code>	<code>0 &lt;= (-0.1)</code>
<code>3 &lt;&gt; 3</code>	<code>-2 = 2</code>	<code>"BAT" = "Bat"</code>	<code>1 = 1.000001</code>

### The If . . . Then . . . Else statement.

The general form of the If . . . Then . . . Else statement is

```
If logical expression Then
  ⋮
  true condition Basic statements
  ⋮
Else
  ⋮
  false condition Basic statements
  ⋮
End If
next line . . .
```

Here, if the *logical expression* is **true** then the Basic statements following the keyword **Then** and up to the keyword **Else** are executed, and then the lines following **End If**. If the *logical expression* is **false** then the statements following the **Else** keyword are executed. We thus provide for two exclusive alternatives.

Be careful that you do not use the more complex If...Then...Else statement when a simple If...Then statement will do. There are always several logically equivalent ways of writing an If statement. It is good programming practice to think of several ways of doing it and then use the simplest one that will do the job.

**MODIFY YOUR PROGRAM OF UNIT 3** so that it also finds and prints out:

- the arithmetic mean of all numbers entered so far (i.e. the simple average);
- the geometric mean of only the positive numbers entered so far\*;
- the arithmetic (ordinary) mean of the squares of each of all the numbers entered so far\*;
- the highest value of all numbers entered so far;
- the lowest value of all numbers entered so far.

Make provision for the user to send the results to the printer and to make the program end whenever desired in a “user-friendly” way. Your method for doing this should not preclude any particular number being entered for computation,

```
e.g.  If x = 0 Then
      End
      End If
```

*will not do.*

[\* The **geometric mean** of  $n$  numbers is the  $n^{\text{th}}$  root of the product of the numbers. So the geometric mean of the numbers 12, 6, 3 is  $(12 \times 6 \times 3)^{1/3} = 6$ . The geometric mean of a set of numbers containing negative ones is complex. This is why we are restricting this mean to only the positive number entered (but we are *not* restricting the arithmetic mean!).

To find the **mean of the squares** of the numbers you will obviously need the **sum of squares**. Be sure you understand the difference between “mean of squares” (which is what you want) and “square of the mean” (which is not what you want).

The **highest number** means the “the number nearest to  $+\infty$ ” which is found by using the **>** operator in an If statement, and the lowest number means the “the number nearest to  $-\infty$ ” which is found by using the **<** operator.]

## Unit 5

### Deadline for completion and handing in: Thursday 16 January

There are TWO parts to this Unit.

#### FUNCTIONS

Certain “in-line” or standard functions are provided within the Visual Basic language, such as Abs, Atn, Cos, Exp, Log, Sgn, Sin, Sqr, Tan, etc. There are also special functions for operating on strings, such as Ucase. Note that there are in-line functions for sin, cos, tan, and arctan ( $\tan^{-1}$ ), but not for  $\sin^{-1}$  or  $\cos^{-1}$ . However these can be derived from the others. Nearly all the arithmetic functions take a single arithmetic argument in brackets (which should be in radians when an angle), e.g. Sin(0.5\*theta). The most notable exception is the Mod (modulo) function, which is actually used as an arithmetic operator:  $n \text{ Mod } 2$  gives the remainder when  $n$  is divided by 2.

#### Example of deriving inverse trigonometric function:

The principle here is to find an expression in terms of  $\tan^{-1}$  so that the Visual Basic function Atn can be applied to obtain the result, which is an angle in radians. Note that Atn applied to a negative argument gives an angle in the range  $(-\frac{\pi}{2}, 0)$ , not  $(\pi, \frac{3\pi}{2})$ .

TO FIND A FUNCTION FOR  $\sin^{-1}x$ .

Let  $x = \sin\vartheta$ , then  $\vartheta = \sin^{-1}x$  is the required result.

To link  $\sin\vartheta$  to  $\tan\vartheta$ :

$$\begin{aligned}\frac{1}{\tan^2\vartheta} &= \frac{1}{\sin^2\vartheta} - 1 \\ \therefore \tan\vartheta &= \frac{1}{\sqrt{\frac{1}{\sin^2\vartheta} - 1}} \\ \therefore \vartheta &= \tan^{-1}\left(\frac{1}{\sqrt{\frac{1}{\sin^2\vartheta} - 1}}\right) = \tan^{-1}\left(\frac{\sin\vartheta}{\sqrt{1 - \sin^2\vartheta}}\right) \\ \text{i.e. } \vartheta &= \sin^{-1}x = \tan^{-1}\left(\frac{x}{\sqrt{1 - x^2}}\right)\end{aligned}$$

Notice how the correct sign for  $\vartheta$  is preserved by this formula for all values of  $x$ , and that it does not produce a division by zero when  $x$  is zero, as would the alternative form  $\tan^{-1}\left(\frac{1}{\sqrt{1/x^2 - 1}}\right)$ .

[It does however produce a division by zero for  $x = \pm 1$ , which is inevitable, since  $\tan(\pi/2) = \infty$ ].

This translates to the Basic statement:

$$\text{th} = \text{Atn}(x / \text{Sqr}(1 - x^2))$$

Applying the same approach to finding a function for  $\cos^{-1}x$  would lead to the form

$$\text{th} = \text{Atn}(\text{Sqr}(1 - x^2) / x)$$

which unfortunately does not give the correct value for th when  $x$  is negative.

The solution here is to make use of the relation

$$\cos\vartheta = \sin\left(\frac{\pi}{2} - \vartheta\right)$$

hence

$$x = \sin\left(\frac{\pi}{2} - \cos^{-1}x\right)$$

$$\sin^{-1}x = \frac{\pi}{2} - \cos^{-1}x$$

$$\cos^{-1}x = \frac{\pi}{2} - \sin^{-1}x.$$

### User-defined function procedures

As mentioned in section 15.7 you can define your own procedures. There are several types of procedure but the simplest to understand is the function procedure which in its most basic form can be thought of in a similar way to a function definition in algebra: as an expression using dummy arguments.

To set up such a function go to the Tools menu and click “Add Procedure”. This takes you to a dialogue box where you can specify the type of procedure you want – “Function” – and the name you wish to have. Just type in the name, click on the Function button and then on OK. You will immediately see the skeleton of your new function in the code window. If you had chosen the name xyTOz then what you would see would be

```
Public Function xyTOz()
```

```
End Function
```

Now inside the brackets you must type the dummy arguments that you want the function to be defined with: you can choose any variable names you like within the rules; it doesn't matter if you've already used those names in any other procedure. Say we chose  $x$  and  $y$ . Then we would edit the first line to give

```
Public Function xyTOz(x, y)
```

Now we just type in the definition of our function after this line. Say we want to define a function for  $\frac{xy}{x+y}$ , this would look like

```
Public Function xyTOz(x, y)
```

```
xyTOz = (x * y) / (x + y)
```

```
End Function
```

Now you can use this function in any other procedure, such as Form\_Load(), by reference to its name and placing inside the brackets the actual values or variables that you want the function to apply to, just as in algebra:

```
Print xyTOz (3, 4)
```

would cause the value of  $3 \times 4 / (3+4)$  to be printed on the form.

A function may not be defined in terms of itself (i.e. it may not be *recursive*),

e.g.  $\text{fna}(v) = 2 + \text{fna}(v)^2$  is **not allowed**.

However one function may be defined in terms of another one, so if arcsine has been defined as above, then arccos can be defined by

$$\arccos(x) = 0.5 * 3.14159 - \arcsin(x).$$



**Part A. WRITE A SHORT PROGRAM** that contains the expressions for  $\sin^{-1}$  and  $\cos^{-1}$  as USER-DEFINED FUNCTION PROCEDURES and which employs these user-defined functions to find the angles  $\sin^{-1}x$  and  $\cos^{-1}x$  for a range of values for  $x$  (**positive and negative**). It should also calculate the Sin and Cos of these angles respectively (using the in-line functions) and so allow you to compare the results with the value of  $x$  to verify that the user-defined functions are working correctly.

[Note:  $x$  itself is NOT an *angle*.]

Use your program to find out the effect of  $x$  being exactly  $-1$ ,  $0$  and  $+1$ , as well as other positive and negative numbers. [You do not have to make the program “work” for these values, just find out what it does and describe what happens by writing on your program listing.]

Make sure you have fully understood what is required in this exercise before you begin to write your program. You **MUST** use the user-defined function procedures to define your  $\cos^{-1}$  and  $\sin^{-1}$  functions and **NOT** place your function expressions directly into the main body of the program. This program does **NOT** need the use of any If statements, so if you are tempted to use an If statement think about what you are doing.

[Of course, If statements *may* be used to make the program repeat, as in Unit 4, if you wish.]

#### **A FURTHER EXTENSION OF FUNCTION PROCEDURES**

In the above I described user defined function procedures as a way of *defining* a function because this is familiar from algebra. But in fact there is much more to them than that. You can put into a function procedure anything that you can put into a Sub procedure like Form\_Load(), so long as somewhere in the function procedure the name of the function is assigned a value. So any kind of complicated computation can be made on the “dummy arguments”, running to many lines of code if required.

To understand how this is possible it is helpful to briefly explain what happens when Visual Basic encounters a reference to a function procedure. So in the line we met before

Print xyTOz (3, 4)

Visual Basic recognises that xyTOz is a function procedure and copies the argument values “3” and “4” into the dummy arguments  $x$  and  $y$  of the function xyTOz, treating now  $x$  and  $y$  as variables within that procedure having these values. Execution now passes to the function xyTOz and once the code there is exhausted the value that has been computed for xyTOz is passed back into the original statement that generated the reference to xyTOz: the Print statement in this case.

*CONTINUED*

**Part B. THE BINOMIAL COEFFICIENTS** are defined for a pair of positive integers,  $n$  and  $r$ , as follows:

$$C_{n,r} = \frac{n!}{r!(n-r)!}$$

where  $n \geq r$ . [0! = 1 by definition].

Below is a function procedure called *Factx* which finds  $Factx = x!$  Create a function procedure of this name and type in the code as it is given. Remember to give the function the dummy argument  $x$ .

**Write a program** using this function that can calculate the value of any required binomial coefficient specified by the values of  $n$  and  $r$  which the user types in.

```
' Function to find x!  
Factx=1  
If x<2 Then Goto finish  
k=1  
Loop1:  
k=k+1 : Factx=Factx*k  
If k<x Then Goto Loop1  
finish:
```

[Because you are using the given function your program should not contain any loops other than the one in that function (and possibly an overall repeat loop, as in Units 3 and 4). If you find that your program needs to contain more than this one loop for finding the factorials then you probably have not yet properly understood the way a function is used: [ask for help](#)]

Notice in the above function the compact single-line variation of the If...Then statement and the use a colon for putting two statements on a single line. Use these forms of the If statement and this type of colon very sparingly and only when it improves the clarity of the program, never simply to save typing more lines.

## APPENDIX A

### Aims and Objectives of the Practical Skills Course Programme

#### Aims

To equip the student with those practical skills and the flexibility of approach to the solution of practical problems which employers expect to find in Physics graduates employed as scientists in research and development or in other contexts.

#### Objectives

At the end of the full practical skills programme the student should:

- Be practised at working in an environment where best practice in matters of safety is required.
- Have developed observational, investigative and data analysis skills by the performance of a wide variety of experimental investigations.
- Have developed confidence and facility in the use of a wide range of technical equipment. In an era of rapid technological change this does not imply detailed training in the use of any particular equipment or technique. The student should rather have developed a receptiveness to the use of whatever equipment and techniques are appropriate for the solution of particular problems.
- Have increased conceptual understanding of topics in the theoretical part of the degree through the performance of related experiments.
- Have developed enterprise, initiative and originality of approach in problem solving by taking part in project work.
- have acquired competence in oral and written scientific communication skills.
- Have developed skills in the use of computers within a variety of applications from standard packages used in various contexts and specialised programs used for the solution of particular classes of technical problem to self-motivated computer programming.

**Note:** The full extent of these objectives is met in a set of courses extending through all four years of the BSc (part-time regulations) Physics degree.

## Appendix B

### Objectives of the Treatment of Experimental Data Course Component

AT THE END OF THE LECTURES ON THE TREATMENT OF EXPERIMENTAL DATA THE STUDENT SHOULD:

- Understand the principles of minimising observer bias and maximising accuracy in physical measurements.
- Be able to find the mean, standard deviation and standard error of a set of repeated measurements and understand the physical significance of each of these.
- Be able to analyse linearly correlated pairs of data values by plotting a graph, drawing a straight line fit, and finding the gradient and intercept, and to estimate the uncertainties in these quantities.
- To propagate the uncertainty of a physical quantity into another quantity which is a function of the first by the methods of differentiation, spot calculation, or rules-of-thumb for a range of standard functions.
- Be able to combine the uncertainties for a function of more than one measured quantity in quadrature both from first principles and using rules-of-thumb for a range of standard functions.
- Understand the distinction between random error and systematic error and know how to estimate these.
- Be able to quote numerical results of measurements using the correct format with regard to number of significant digits, uncertainty, and units.
- Know how to use the uncertainty to assess the significance of the discrepancy between different results for measurements of the same quantity.
- Be able to construct graphs with appropriate choices of scale and properly labelled axes.

## Appendix C

### Objectives of the Computing Course Component

AT THE END OF THE COURSE ON WORD-PROCESSING THE STUDENT SHOULD:

- Be familiar with the computer keyboard and use of the mouse for single clicking, double clicking, and dragging.
- Know how to use the College network system with respect to logging in, running *Windows*, running *Word*, exiting *Word* and *Windows*, and logging out.
- Be familiar with the environment of the Word word-processing application in respect of its menus, tool bars, and window controls.
- Know how to type a document consisting of paragraphs and headings.
- Know how to save a document onto the network disc or floppy disc and how to load an existing document from disc.
- Know how to edit a document using the most basic techniques for moving the insertion point, deleting text and moving text.
- Know how to change the appearance of text in terms of character font, size, and style.
- Know how to search a document and replace a given string of text.
- Know how to enter text as subscript or superscript and how to add icons for these operations to the tool bar.
- Know how to use the spell checking facility.
- Know how to include special Greek and mathematical symbols in a document using the Symbol and MT Extra fonts.
- Know how to access the Equation Editor.
- Know how to align the text within a paragraph and set left and right margins and indents.
- Know how to set and employ tabs.
- Know how to create simple tables for data presentation.
- Know how to set up the global formatting of a document with respect to margins and page numbering.
- Know how to print a document.
- Know how to access the Help facility.
- Be aware the basic format conventions for producing a scientific report.

AT THE END OF THE COURSE ON VISUAL BASIC COMPUTER PROGRAMMING THE STUDENT SHOULD:

- Be familiar with the Windows environment for running and using the Visual Basic program, including the use of menus and dialogue boxes.
- Be familiar with the code and immediate windows and the output form of Visual Basic and the ways to switch between them.
- Be able to write and edit, using the cut and paste operations, Visual Basic programs not employing procedures or function modules.
- Be able to use the Print statement with comma and semicolon separators to produce roughly formatted screen output of numerical and string constants and numerical and string variables.
- Be able to use the MsgBox statement and the InputBox function to enter numerical and string data to a program with appropriate user prompts.
- Be able to use the CurrentX and CurrentY statement and/or the Tab function to control the spacing of screen output.

- Be able to use the PrintForm statement to send output to a printer.
- Be able to use labels and the Goto statement to produce a program loop.
- Be able to write arithmetic expressions using all of the arithmetical operators.
- Know how to correctly initialise and increment variables used for counting, totalising and factoring.
- Be able to write basic If statements using relational operators (but not logical operators) for controlling program flow and finding minimum and maximum values.
- Be able to use Function procedures to define one-line functions.
- Be able to use a restricted number of in-line functions: Sin, Cos, Atn, Sqr, Ucase.
- Be able to write and employ Function Procedures as sub-programs within the main module.
- Know how to properly structure a program with appropriate use of internal comments (Rem statements), instructions and information to the user (Print statements), and well-chosen variable names.
- Know how to assess and demonstrate the performance of a program using test runs and sample printer output.
- Know how to print a program and to save and load a program on disc.