

Computation and its limits

Cockshott, Mackenzie, and Michaelson

March 23, 2010

Contents

1	Introduction	5
1.0.1	Acknowledgements	8
2	What is computation	9
2.1	The apparent mystery of maths	9
2.2	Counting sheep	14
2.3	Counting materialised in our own bodily movements	19
2.3.1	Fingers	19
2.3.2	Voice	21
2.4	From aides memoir to the first digital calculating devices	28
3	Mechanical computers and their limits	35
3.1	Antikythera	35
3.1.1	Was the Antikythera device an analogue computer?	42
3.2	Late mechanical computers	44
3.2.1	Estimating rates of change	46
3.3	Analogue mechanical multiply/accumulate	47
3.4	Mechanising the abacus	51
3.4.1	Digital multiplication	54
4	History of computability debate Greg	57
5	Heat, information and geometry Paul and Lewis	59
5.1	Triumph of digital computation	59
5.2	Analogue computing with real numbers	60
5.3	What memories are made of	62
5.4	Power consumption as a limit	67
5.4.1	CMOS	67
5.5	Entropy Lewis	71
5.6	Shannon's Information theory	71
5.7	Landauer's limit	74
5.7.1	Thermal noise	75
5.8	Non entropic computation Paul& Lewis	77
5.8.1	Adiabatic CMOS gates	82

6	Quantum computers	85
6.1	Quantum limits to real number representations	85
6.2	Error rates in classical and quantum gates	90
7	Hyper computing proposals	93
7.1	Infinite Turing machines	93
7.2	Infinitely precise analogue computers	95
7.2.1	Newtonian computing	96
7.2.2	Bournez and Cosnard's analogue super-Turing computer	100
7.2.3	Optical prime factorisation	103
7.3	Conclusions	106

Chapter 1

Introduction

Although we are entirely unaware of it, *computation* is central to all aspects of our existences, that is applying *rules* to *information* to produce new information, usually to make a decision about what to do next¹. Every day, we solve, or try to solve, a myriad of problems, from the utterly trivial to the bafflingly complex. To do this we deploy some processes of reasoning, often unarticulated, and often aided by large doses of what we might all “common sense” or “gut feeling” or “intuition”. And often, our reasoning is not as definitive as we might like, leading us to conclusions that are wrong or tentative or contrary to what we might prefer.

It is striking that our reasoning is most accurate where the information and rules are most precise, in particular where we use *numbers* and rules deriving from *mathematics*, such as those for arithmetic and geometry and logic. Again, we carry out most of this sort of computation quite unconsciously, except when we get it wrong. However, we do so using *standard* representations and rules that we learnt as children, and that we expect other people to share. One big strength of this standardisation of numeric reasoning is that it can easily be made explicit, and so repeated and checked, by oneself or other people, until it is right. Another big strength is that we can build *machines*, like calculators and computers, to carry out computations for us, and we can agree that these machines do actually embody the rules and information we hold in common.

Now, we actually perform very sophisticated computations all the time. In the course of a day, we might, say:

- check whether we have the right change for an automatic ticket machine;
- decide how much food to buy to cook for visiting friends as well as the household;
- make sure that the shop bill has accounted for everything we bought;

¹Here we are using “information” in the general sense of facts or knowledge about some circumstance, rather than the precisely defined technical conception of information discussed in the rest of this book.

- decide what order to cook the food in, to make sure all of each course is ready at the appropriate times;
- work out how many days we can afford to spend on our spring vacation, if we've already committed some to a summer holiday and need to leave some over for for a family celebration;
- decide if it's cheaper to take the train but spend a night in a hotel on the way or fly directly to our destination.

These sorts of computations seem trivial; certainly they are small and simple enough for us to perform in our heads. Nonetheless, they actually require considerable competence in mental arithmetic and sequencing of activities, as becomes clear if we try to explain how to do them blow by blow to someone else.

It is important to notice that we do not generally need fine precision for these sorts of daily computations. In the above examples, we want *enough* ticket machine change or food or time for cooking or holiday days left or travel cost saving, where "enough" allows a reasonable margin of error. So, we accept that we may make mistakes but are happy with "ball park" figures.

For more complex, or more costly, computations however, such as planning a new kitchen or an extensive holiday, we require greater assurance and precision. Here, we readily turn first to pencil and paper, next to a calculator, and, increasingly, to standard software on a personal computer, such as a spreadsheet. We do so, perhaps, because we believe that these offer a hierarchy of increasing checkability and reliability as mechanisation increases and human computation decreases. Indeed, these tools enable us to focus on making *models* of problems, at which human beings are supremely accomplished, rather than performing complex computations, where we are prone to inaccuracies and mistakes.

Now, sometimes things still go wrong when we use apparently reliable tools. Almost always, the mistakes are our own fault, for example if we enter the wrong data or faulty equations into a spreadsheet.

Nonetheless, we are plainly aware of limitations to our tools. Thus, it's surprisingly easy to try to carry out a calculation to more places than a pocket calculator can support. More to the point, we constantly notice our personal computers going "too slowly" or running out of memory as we demand more and more of them. However, these limitations do not seem insurmountable because we have got used to the continuous cheapening of computers and memory, accompanied by increases in their speed and capacity, either using new technologies or by making better use of existing ones.

For example, just from looking at advertisements, the speed of a single computer processor seems to have stayed at around 3 GHz since the turn of the century. However, we are now offered personal computers made up of *multiple cores*, first two, next four and eight, with no apparent limit to the number. Thus, the limited speed of one processor can be compensated by using lots of them.

For example, digital cameras and personal music players now use solid state memory where previously they were based on magnetic tape, magnetic disks, or CDs or DVDs. And laptops and “web books” no longer necessarily have hard disks, let alone CD or DVD writers.

We have also got used to a very different style of accessing and storing information, on remote machines run by other people, rather than on some tangibly physical device on a desk or in a pocket. Just as CDs killed cassette tapes and LPs for music, and DVDs displaced video for films, now direct downloading to media players of intangible codes is replacing personal ownership of physical copies².

Contrariwise, we trust more and more material which might once have been considered our personal responsibility to external repositories. Thus, we not only implicitly accept remote maintenance of our email folders by our service provider, but also explicitly upload what we still regard as “private” photographs and documents to unknown physical locations.

Finally, we routinely take advantage of complex computations performed remotely on our behalf. For example, we increasingly optimise flights for multi-location holidays, check weather forecasts or find routes for road journeys, all using on-line services rather than consulting time tables, barometers and maps.

Thus, even if there are restrictions on the machines we can own ourselves, there seem to be no bounds to the capacities of the shadowy world of services and servers at the other end of the Internet. And even if we constantly run up against the limits of current technology, or what we can afford, then all our experience suggests that the former will eventually be overcome, if not the latter.

Alas, as we explore in the rest of this book, there are deep reasons why such optimism in unlimited technological progress is absolutely unfounded, even if relatively appropriate. First of all, when we try to give a precise characterisation to the notion of *abstract computation*, we discover that certainty in our reasoning is undermined by troubling paradoxes at the core of mathematics. And secondly, when we look at the physical characteristics of our universe, we find that there are fundamental limits to the properties of the *concrete computers* we might construct.

Now, we are all familiar with the idea that some things are just not physically feasible: at least not within physics as we currently understand it. For example, we are generally accepting that time travel, perpetual motion, and faster than light speeds are only possible in science fiction worlds. Nonetheless, every year people come up with new attempts to transcend these physical barriers. And every year, scientists and engineers explore and ultimately refute such attempts.

Similarly, there have been many attempts to develop abstract *hyper-computational* systems or concrete *hyper-computers*, which seek to transcend the physical and mathematical limits we have just alluded to. Thus, as well as exploring lim-

²And books may yet succumb to eReaders.

its to computations and computers, we will also discuss attempts to overcome them, and consider why they are ultimately ill-founded.

The rest of this book is organised as follows:

- list of chapter summaries

1.0.1 Acknowledgements

Chapter 2

What is computation

2.1 The apparent mystery of maths

In an influential paper [81] Wigner asked why it was that maths was so 'unreasonably effective' in describing physical reality. The question he asked are similar to those we are dealing with in this book. Our topic is, on the surface, slightly narrower. We are dealing with computing rather than maths in general. But if we look at what Wigner has to say, we find that he too is concerned primarily with the practical application of maths, and that practical application to physics always involves doing real computations. So Wigners concerns make a good starting point.

He starts with a perhaps apocryphal tale:

There is a story about two friends, who were classmates in high school, talking about their jobs. One of them became a statistician and was working on population trends. He showed a reprint to his former classmate. The reprint started, as usual, with the Gaussian distribution and the statistician explained to his former classmate the meaning of the symbols for the actual population, for the average population, and so on. His classmate was a bit incredulous and was not quite sure whether the statistician was pulling his leg. "How can you know that?" was his query. "And what is this symbol here?" "Oh," said the statistician, "this is pi." "What is that?" "The ratio of the circumference of the circle to its diameter." "Well, now you are pushing your joke too far," said the classmate, "surely the population has nothing to do with the circumference of the circle."

Posed in this way, the fact that mathematics proves so useful to science and to understanding the world in general does seem quite remarkable. But look at this example. Is it really mathematics, or is it computation that is proving useful?

The Gaussian distribution, π etc are being used in order to perform particular calculations, by means of which the statistician is intending to make predic-

tions about future population. Most times that we come across examples where maths proves useful, it is not abstract maths that is directly useful, but applied mathematics. Applied maths are used to perform calculations, and these, either guide us in our practical engineering or allow us to make predictions of experimental science. So we can shift Wigner's question and ask instead about the remarkable usefulness of applied mathematics.

Prior to the universalisation of automatic computers, there did not exist the contemporary distinction between computation and mathematics. The one was seen as just a necessary component part of the other. Both were intellectual activities, both were performed by mathematicians, albeit with different specialisms. Now, when computing is done by machines, we tend to think of them as distinct things: maths an intellectual discipline, computation a rote automatic process. Of course everyone concedes that the actual programming of computers is a tricky intellectual task which may involve mathematical skill. But it is on computing that science and engineering now depend for practical predictions. Whether they are working at Boeing it is in the design and control of an airliner or in the Met Office working on long term climate forecasts. So we can now focus in a bit on Wigner's question and ask why is it that computers can be so unreasonably effective in the practice of science?

So long as the question was posed in its original form it seemed an insuperable enigma. True enough, Wigner did give an answer. He attributed the effectiveness of maths to the spatial and temporal invariance of the laws of physics. That no doubt plays its part, but the answer remains one that would appeal mainly to physicists whose vision of the practical application of mathematics was very much in their own field. It does really answer the query about π in his initial anecdote.

One can allow that the maths required to do physics would be far more complicated were the laws of physics not spatially invariant, without gaining much insight into why maths should work in other domains. Why does maths work for predictions about populations, or in the analysis of genetic inheritance?

Hamming, following up on Wigner asked:

Furthermore, the simplicity of mathematics has long been held to be the key to applications in physics. Einstein is the most famous exponent of this belief. But even in mathematics itself the simplicity is remarkable, at least to me; the simplest algebraic equations, linear and quadratic, correspond to the simplest geometric entities, straight lines, circles, and conics. This makes analytic geometry possible in a practical way. How can it be that simple mathematics, being after all a product of the human mind, can be so remarkably useful in so many widely different situations? [38]

We can visualise what Hamming is talking about in Figure 2.1. There we have Newton using the ideas of geometry to elaborate his "Mathematical Principles of Natural Philosophy". These are pure thoughts, but, remarkably, the mathematics mirrors what is happening in the solar system. There seems an

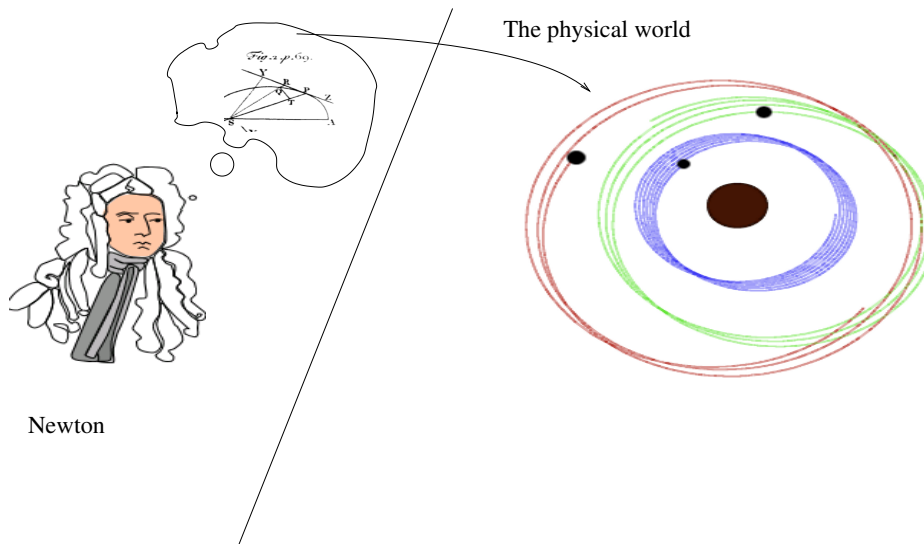


Figure 2.1: Newton thinks

uncanny correspondence between some simple principles of geometry in one man's mind and the movements of planets millions of miles away.

To explain his ideas to others, Newton had to resort to pictures, diagrams and arguments on the printed pages of his great book. The book itself was material, and copies survive to this day. While the thinking which went into writing the book was fleeting and perished along with the man, the maths seems eternal, independent of the man and of his paper book. It is as able to predict heavenly motions now as it was in the 17th century.

If we think of things in this way, as a correspondence between two quite different domains – that of thought and that of reality – the whole process seems so remarkable as to tempt one to ascribe some mystical properties to mathematics.

Thought \rightarrow Math \iff Reality
Mystic connection

But now look at Figure 2.2. It depicts the early days of space exploration. A boffin sits at his old valve computer and gets it to work out the course that will take a probe to Mars.

At one level this shows the same process as Figure 2.1, but the very physicality of that big grey metal box in front of the boffin hints at something different. The similarity is that Newton's laws, and their associated mathematics are being used in each case. But the fact that the calculations are now taking place in a machine makes it harder to see the process as being one of a correspondence between mathematical thought and reality.

To do his calculations the NASA scientist would have had to have fed the computer with a whole lot of data obtained by astronomers. He will have

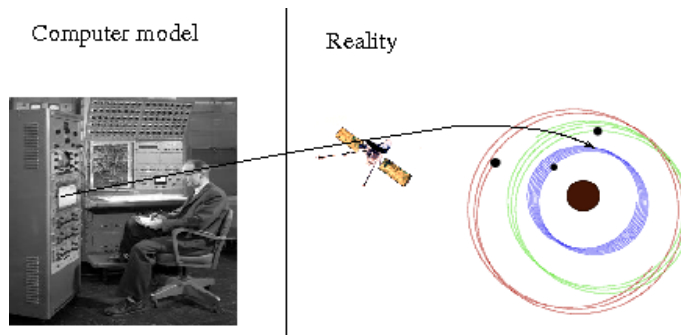
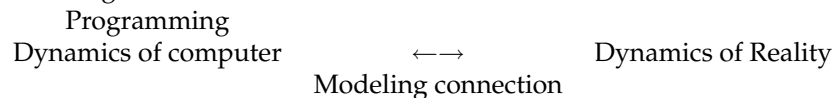


Figure 2.2: NASA computes

had to develop programs to represent the dynamics in question. And he then set it off working. We say he will have had to develop programs, but, that is not strictly necessary. The computer in the picture is actually an analogue one, which was programmed by rewiring it using the patch panel behind the operator.

So the correspondence here is actually one between the dynamics one physical system – the analogue computer, and the dynamics of another – the rocket that the designer wishes to control.



The diagram above is representative of a lot of what we now call 'computing'. In this book will be examining just how it is that we can set up a physical system to emulate another system. Our approach is that of practical computer scientists concerned with real computing systems, ones that are either buildable now, or could in principle be built. We think that this computer science perspective also helps with other problems raised by Wigner.

The great mathematician fully, almost ruthlessly, exploits the domain of permissible reasoning and skirts the impermissible. That his recklessness does not lead him into a morass of contradictions is a miracle in itself: certainly it is hard to believe that our reasoning power was brought, by Darwin's process of natural selection, to the perfection which it seems to possess. [81]

Does it demand a very high level of evolved reasoning power to do what a great mathematician does?

Does this make unreasonable demands on evolution?

The objection that we never had much selective pressure on us to master calculus during the paleolithic is fair enough. But need we imagine a specific selective pressure in the past for every behavioural trait that we observe in the present?

In one sense the answer has to be no. The diversity of human cultural behaviour is so great that nobody seriously contends that it is all in the genes. The social behaviour of the bee, yes, but the social behaviour of humanity no. The calculus is clearly a cultural product. It is something that arose at a specific time with Newton and Leibniz, and something that is learned by individuals rather than being an innate skill. This, however, still leaves open the question as to why we are able to do the reasoning required by calculus. The specific details of calculus had to be developed by Newton, but why did he or any other person have the reasoning ability to deal with something so abstract?

One possibility is to consider the implications of a very simple computer, the Turing Machine, described in Chapter 4. This simple device has a universal mathematical ability.

The key point is that one does not need anything more than a Turing Machine equivalent process on the part of the mathematician. A mathematician does not master calculus just by innate mental power. They use a set of learned rules plus external aids: chalk and blackboards, etc. Our dependence on external computing aids, from simple paper and pencils to digital computers, is another theme we explore.

The computational power of the Turing Machine is universal. Similarly, a person, with a sufficient starting set of mathematical rules plus external notational aids, also has a universal mathematical ability. One does not need to assume a series of specific evolved abilities in the brain to handle the whole diversity of maths. All one needs is a certain threshold of ability. Provided a certain threshold, the Turing Universality Threshold is passed, then the computational system composed of mathematician, innate reasoning primitives and external notations will also be universal.

It is true, of course, that physics chooses certain mathematical concepts for the formulation of the laws of nature, and surely only a fraction of all mathematical concepts is used in physics.

.....

A possible explanation of the physicist's use of mathematics to formulate his laws of nature is that he is a somewhat irresponsible person. As a result, when he finds a connection between two quantities which resembles a connection well-known from mathematics, he will jump at the conclusion that the connection is that discussed in mathematics simply because he does not know of any other similar connection. It is not the intention of the present discussion to refute the charge that the physicist is a somewhat irresponsible person. Perhaps he is. However, it is important to point out that the mathematical formulation of the physicist's often crude experience leads in an uncanny number of cases to an amazingly accurate description of a large class of phenomena. This shows that the mathematical language has more to commend it than being the only language which we can speak; it shows that it is, in a very real sense, the correct language. [81]

A point made by Wolfram [84] is that there may be multiple different ways of approaching this with different mathematical models. It is a matter of what mathematical tools were available to the modeler that determines how they represent it. Consider the problem of feedbacks occurring in cellular biology. This can be modeled using differential equations, it can be modeled using process algebra, or it can be modeled using boolean algebra. Because the modeling is now done using a computer, then all of these come down to the use of different software to model a real system.

We are no longer surprised to find multiple software packages available for some task. The mathematical techniques that software packages use are one way these packages may differ. Wigner agrees that only a fraction of all mathematical concepts have turned out to be of use in physics, similarly, only a tiny fraction of all the possible programs are suitable for a given computing task. What is interesting is not that some maths and some programs are useful, but the process by which useful programs and mathematical techniques are developed. Hamming observes that:

The idea that theorems follow from the postulates does not correspond to simple observation. If the Pythagorean theorem were found to not follow from the postulates, we would again search for a way to alter the postulates until it was true. Euclid's postulates came from the Pythagorean theorem, not the other way.[38]

This is a very good point and fits in with the view that the development of mathematics should be seen as a form of software development. Chapter 4 will look at the relationship between issues in software development and those that arise in mathematical proof.

One answer to Wigner's problem is that the simple maths and the simple geometric entities described by him, are systems with a low information content, and can be generated by processes with a low information content. This argument can be developed on the basis of material in Chapter 5.

2.2 Counting sheep

I have tried, with little success, to get some of my friends to understand my amazement that the abstraction of integers for counting is both possible and useful. Is it not remarkable that 6 sheep plus 7 sheep make 13 sheep; that 6 stones plus 7 stones make 13 stones? Is it not a miracle that the universe is so constructed that such a simple abstraction as a number is possible? To me this is one of the strongest examples of the unreasonable effectiveness of mathematics. Indeed, I find it both strange and unexplainable. [38]

In the above quotation from his paper *The unreasonable effectiveness of mathematics* Hamming points to how computing got started. People learned to count

out of practical necessity. Once animals had been domesticated and were being herded, the shepherds needed to know whether they had lost any sheep. Suppose you set off with a herd in the morning and then return to the pen at sunset. How, if you can't count, can you tell if you have lost any?

If sheep were more placid you line up your sheep and put stones in front of them, one per sheep. More practically, set up a pile of pebbles outside their pen. As the sheep come out take pebbles from the pile and put them in a bag. Allocate 6 sheep to one shepherd and 7 to another, get them to lead the sheep up to pasture and back. When the sheep come back, line the sheep up (or let them into a pen one at a time) and pull one stone out of the bag for each sheep. If they do not correspond, then you know a sheep has gone missing (or another sheep has been picked up along the way). At this stage of development the society need not have any specific language to refer to numbers, the stones substitute for words. Menninger[58] cites the Wedda of Ceylon as still being at this level in the 1960s.

This can be extended to deal with two shepherd's two sons each of whom takes part of the flock to different pastures. The elder son is given the greater part of the flock and a jar of his stones is set aside, one stone to each sheep he led away. It is 7 stones in Hamming's example. The younger son takes the rest, putting a stone in his own jar for each sheep he takes. When they come back the father can determine if either son has lost a sheep by comparing the stones with the sheep as they are penned for the night.

Hamming identifies in this simple procedure a real marvel. *'Is it not a miracle that the universe is so constructed that such a simple abstraction as a number is possible?'* he writes. But is it really such a marvel?

What is required for it to work?

Suppose, instead of mere pebbles, the shepherds went one better and made little clay models of their sheep, each in the likeness of a particular sheep. Given that people can become sufficiently familiar with their sheep to recognise them as individuals, this is not impossible. Now when Snowflake steps out of the pen he puts the clay Snowflake into the jar, as Misty trips along, in goes clay Misty etc. In the evening clay Snowflake comes out of the jar as the real Snowflake comes home. When, at dusk, clay Misty is left in the jar, the shepherd fears poor Misty met Mr Wolf.

This system is more complicated since individual sheep must be recognised. But it brings out the essential properties of sheep and tokens that are exploited in other token based counting mechanisms.

Is there anything remarkable about this?

Not really, all it requires is that stones and clay tokens don't evaporate, and that the bag or jar you kept them in has no holes. There is nothing mysterious about pebbles. The mystery only arises if you think that there is an abstract domain of numbers quite separate from counting technologies. If we focus instead on the actual historical development of enumeration systems, each step is pragmatic and understandable. From the use of tokens by herders stored in perishable bags, a second step developed. Suppose we have a more advanced form of society with a temple-state which administers economic resources. A

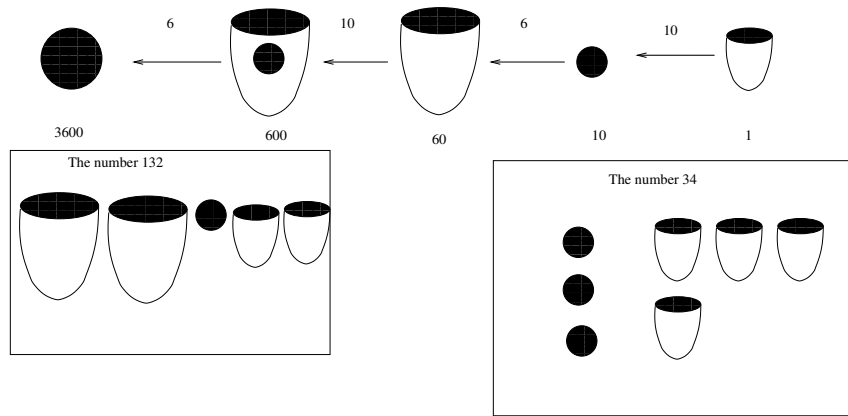


Figure 2.3: The proto-Elamite numeral system using alternate base 10 and 6 for its numbers.

herder is dispatched from one settlement to another with a herd of sheep. He is given a bag recording, with tokens, the number of sheep he set out with. At the other end the officials compare the sheep he delivered with the tokens. The problem is that if he simply takes a bag of sheep tokens there is nothing to stop him meeting his brother along the way and diverting a couple of sheep and throwing away a couple of tokens. Everything still tallies at the end of the journey.

To get round this bags were replaced by sealed clay containers inside which the tokens were placed [61]. These were marked with the temple seal to verify who put the tokens in the container. This provided a tamper proof way of sending and keeping a count of stock.

Next it was realised that it was possible to mark the outside of the container with impressions showing how many tokens were inside. If there were 4 sheep token inside, then a sheep token would be pressed into the outside of the container 4 times whilst the clay was still soft. It was but a small step then to dispense with the tokens inside and just use solid clay tablets with indicators on the front of what would previously have gone in the container. Several systems were initially used to write down numbers of tokens. Base 60 was used for inanimate objects, as in Figure 2.3. A base 10 notation was used for animals, an alternating base 60 and base 120 system for rations[27].

At this point you have the development of a written number system, and with the symbols for the numbers, the idea of the numbers existing independently of the things they count took hold of peoples imaginations.

If you look at the number system in Figure 2.3 it uses a number of distinct signs for ones, tens, sixties etc. Within one of these scales, a number is denoted just by repeating the symbol. So 3 is three indentations using the stem a small stylus. 30 is three indentations using the end of a small stylus, etc. There is a limit to the number of units that we can take in with a glance. Our ability to

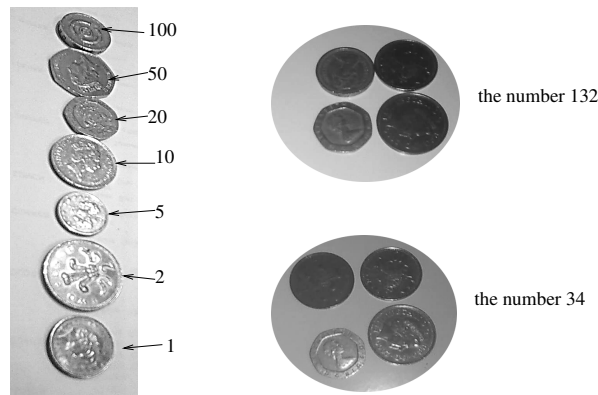


Figure 2.4: A token based computing system used in part of Northern Europe

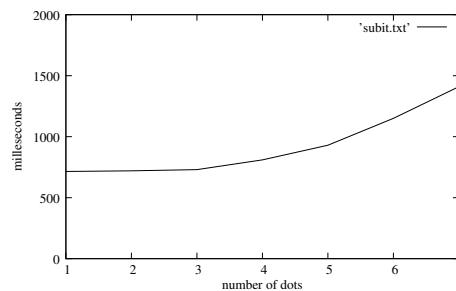


Figure 2.5: The time taken to recognise how many dots are present in a group is almost constant up to 4 and above that recognition time slows down.

take in a small number at a glance is called subitizing. Up to 4 dots, arranged randomly can be recognised in a quick glance [57], but above 4 our time to recognise the number rises, and above dots people have to count. If the dots are arranged in regular patterns however, this aids our ability to quickly judge how many there are. A number system like the proto-Elamite one builds on our inherent subitizing ability to recognise the number of units and tens in a written number. If a people lack both the language to describe numbers or a technology to represent them their ability to handle numerosity is limited to what is provided by our inherited subitizing faculty[36]. The subitizing faculty seems to be primitive in primates rather than being a derived human characteristic. Chimpanzees share the ability to recognise small numbers and can even be taught to associate these with written numerals[73].

Subitizing is limited to very small numbers and seems to be a purely visual process. We have overcome these initial limits by the use of our language ability and by use of external aids.

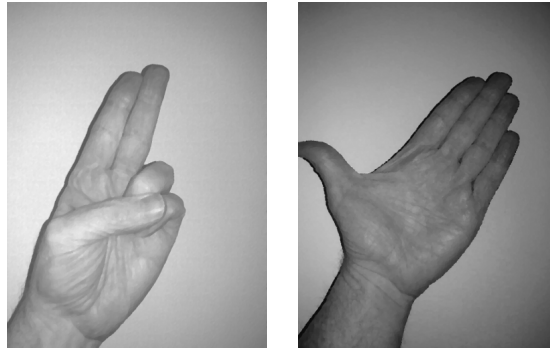


Figure 2.6: A primitive digital computer displaying the numbers 2 and 5.

Using multiple symbols, with a circle representing 10 is more complicated than simply putting stones in bags, or model sheep in jars. An intermediate step in the development was the use of special tokens representing 10 sheep, or 10 goats etcetera[61]. We are so used to doing sums with Arabic numbers that computational systems based on tokens or pictures of tokens seem a bit cumbersome until we realise that much of our daily computations are still done using these primitive computational aids. Figure 2.4 shows a token computing system that is still in wide use. If you compare it to Figure 2.3 you can see that the same basic principles are present in both systems. Different token types or symbols represent different scales of quantity, and accumulation of tokens model accumulations of the things they represent.

Let us consider how these could be used for our original example of counting sheep. Suppose there are what we would call 32 sheep in the pen. The farmer has a bag of unit sheep tokens and another bag of tokens each of which represents 10 sheep. As the beasts are led out, the farmer puts unit sheep tokens into a jar one by one. When all the sheep are out, he piles all the tokens in his jar on the ground. He then takes groups of 10 of the units sheep tokens from his pile and puts them back in his unit bag. Each time he does this he puts 10-sheep token into his jar. At the end he is left with 2 unit tokens in his original pile, and 3 of the 10-sheep tokens in his jar. To record the number of sheep being sent with the shepherd he now makes 2 cup shaped marks and 3 dots.

The invention of tokens of different value overcame the limit to calculation posed by the weight of a mass of individual tokens.

So the process of recording a number in the clay tablet would have required some prior computational aids - tokens in our assumption. But we have skipped over how the farmer got the piles of 10 tokens. Well the obvious computational aide for this is shown in Figure 2.6. We are all born equipped to count to 10. That explains why 10 occurs so often in notations for numbers. Think of the Roman or European number system in Figure 2.7. The first two






			
I	II	III	IIII
<hr/>			
			
V			
X	XX	XXX	XXXX
L			
C	CC	CCC	CCCC
D			

Figure 2.7: Roman numbers and their origins in finger counting according to Ifrah [43].

rows of the table are show the close relationship between the symbols for the numbers up to 5 and the corresponding hand-signs. The rows below that for the 10s and to 50 and 100s to 500 show a recursive application of the principle derived from the 1s and 5.

Another technology that may have fed into the Roman number system is that of tallying, the carving of notches on sticks to count things[43]. These markings remain in common use, using paper and pencil. They rely on our subitizing ability to recognise patterns of strokes up to 4.

2.3 Counting materialised in our own bodily movements

2.3.1 Fingers

Counting on the fingers is both a motor process and a visual process, with the visual process acting as a reminder for the motor sequence. If the use of the fingers is as shown in Figure 2.7 then visual reminders are not needed, the thumb holds down the fingers until they released. This allows one handed counting whilst the visual attention and right hand are used for another task. The thumb imposes a sequence on the release of the fingers, diminishing the

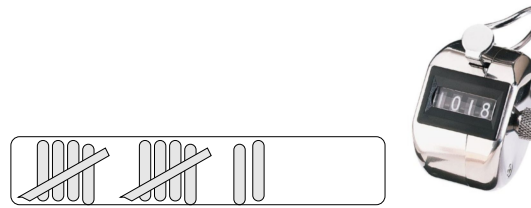


Figure 2.8: Notched tally sticks are another way of counting in multiples of 5. A tally for 12 is shown alongside a modern mechanical tally.

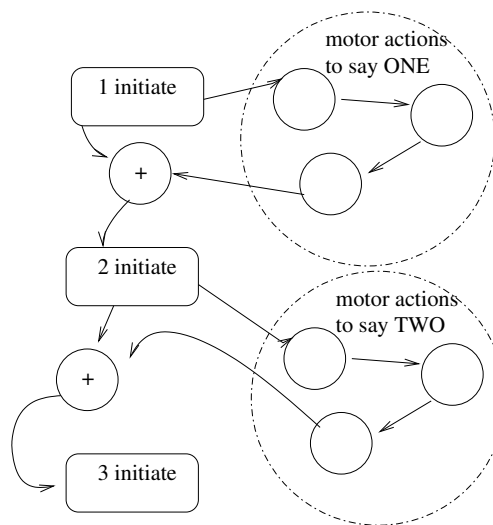


Figure 2.9: Sequencing for counting

mental effort involved.

Lakoff and Nunez[48] argue that much of our mathematical ability is grounded in primitive conceptual and motor schema primitive to the nervous system. Motor control processes, they argue, have a basic schema of components, some of which may be omitted in particular actions:

1. Readiness to act,
2. Starting up,
3. The main process
4. Possible interruption or resumption
5. Iteration

6. Checking for goal achievement
7. Perform completion actions
8. Finished state

A motor schema may include nested subschema that have a similar structure to the main schema. In the case of finger counting this schema would map onto the steps of (1) place thumb over index finger, tense index finger, watch for event; (2) observe an event e.g. a sheep leaving a pen; (3) move thumb causing finger to fly up; (5) iterate going back to step 1; (6) are all 4 fingers released?; (2) observe an event; (7) complete process by raising the thumb.

This is a comparatively simple iterative process, with the nervous sequencing proceeding independently of the count - the count itself being held on the hand as a mechanical register.

2.3.2 Voice

What computational capacity must our brains have to allow us to count aloud?

Counting aloud is a motor process, a more complex one than finger counting since it is one in which the sequencing has to be all internally driven by the nervous system. The fingers are no longer there outside the brain to act as mnemonics. It presupposes a process something like what is shown in Figure 2.9. This involves a type of sequencing that electrical engineers studied under the rubric of finite state automata. A finite state automaton is a physical system that has a finite number of states, transitions between those states, and actions. They were examined in the context of electronic control circuits by Moore[59] and extended to linguistics by Chomsky[13]. They have since become a foundation block for the analysis of computer languages and the design of computer hardware.

Chomsky had been investigating the possibility of formally specifying natural languages. He had classified grammars into classes. These classes of grammars are now referred to as Chomsky class 0, class 1, class 2 and class 3 grammars. It turned out that Chomsky class 2 and class 3 grammars are most suitable to describe programming languages, and the concepts involved in these are also relevant to understanding what goes on when a person counts out loud. To grasp what these different classes of grammars are we need to go into a little formal notation.

The syntax or grammar of a language can be thought of as being made up of a 4 tuple $(T; N; S; P)$ where:

T stands for what are called the terminal symbols of the language. In a human language these terminal symbols are the words or lexicon of the language. In a computer language they are things like identifiers, reserved words and punctuation symbols.

N stands for what are called the non-terminal symbols of the language. In a human language a non-terminal would be grammatical constructs like a

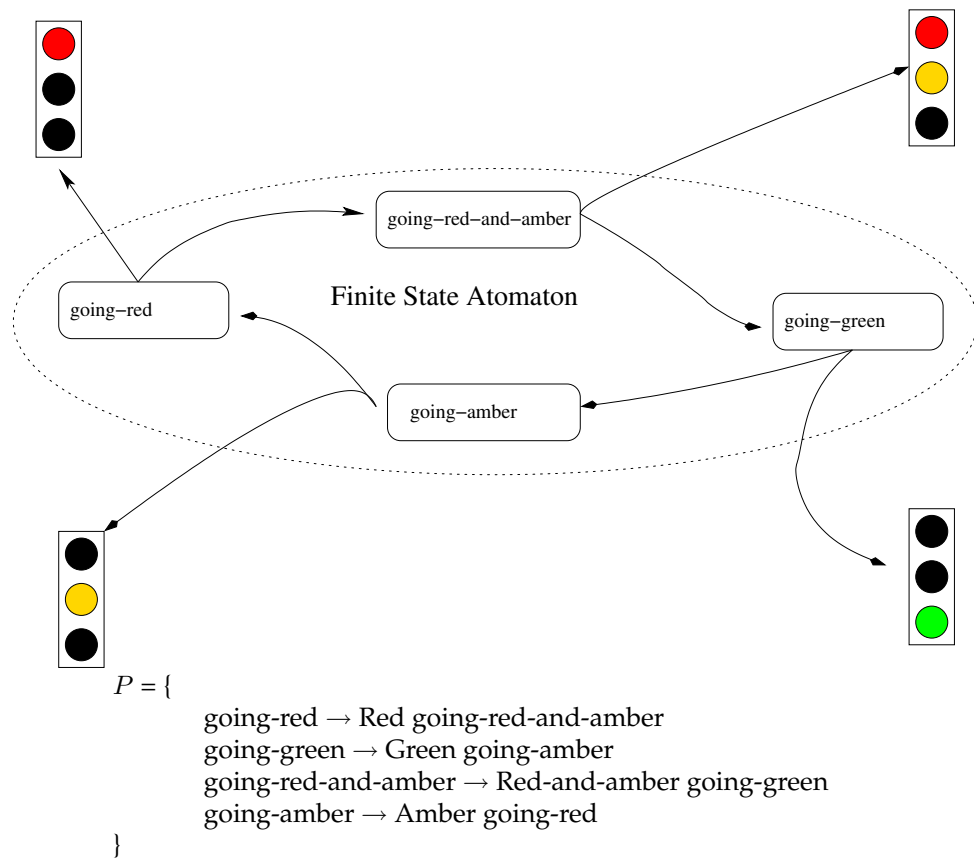


Figure 2.10: Production rules and behaviour of a traffic light.

sentence, a noun clause or a phrase. A computer language is likely to have a large number of non-terminals with names like clause, statement, expression.

S is the start symbol of the grammar. It is one of the non terminals. Its meaning will become clear shortly.

P is a set of productions or rewrite rules. These tell you how to expand a non-terminal in terms of other terminals and non-terminals.

This sounds a bit dry, but it will be clearer if we give an example. Suppose we wish to define a grammar that describes the 'speech' of a traffic light. A traffic light has a very limited vocabulary. It can say red or amber or green or red-and-amber. These are the terminal symbols of its language.

$T = \{ \text{Red, Green, Amber, Red-and-amber} \}$

At any moment in time the traffic light is in a current state and after some interval it goes into a new state that becomes its current state. Each state is described by one of the colours of T . This can be expressed as a set of non-terminal symbols which we will call:

$N = \{ \text{going-red, going-green, going-amber, going-red-and-amber} \}$

We will assume that when the power is applied for the first time the light enters state going-red. Thus

$S = \text{going-red}$

A traffic light has to go through a fixed sequence of colours. These are the syntax of the traffic light language. Which sequence it goes through is defined by the productions of the traffic light language. If the light is in going-red then it must output a red light and go into going-red-and-amber. We can write this down as:

$\text{going-red} \rightarrow \text{Red going-red-and-amber}$

This is an individual production in the traffic light language. The whole set of productions is given in Figure 2.10.

This combination of $(T; N; S; P)$ ensures that the only sequence of colours allowed by the traffic light are the cycles shown in the diagram.

The traffic light grammar is what Chomsky classified as type 3 and what computer scientists now call a regular grammar. Regular grammars can be produced by finite state machines. Finite state machines are popular among electrical engineers constructing simple control devices since they can be constructed with very few components. If we look back at Figure 2.9 you can see that the motor control structure shown there, is similar to the finite state machine in Figure 2.10. The difference is that the traffic light control process goes in a cycle, whereas counting is potentially unlimited. What sort of grammar is required to express say counting up to 100 in English?

Clearly one could do it with a class 0 grammar but that would require the brain to hold a finite automaton with 100 states for the purpose. This is obviously doable, since people can learn poems much longer than that, but it does not seem plausible that it is done that way since to count to 1000 we would have to learn a finite state machine with a 1000 states. Instead we make use of patterns to save on learning so many things by heart. We make use of the repeating pattern from 1 to 9 in counting aloud from 21..29 or 31..39. The technical issue is how this sequence is made use of. If we counted as follows:

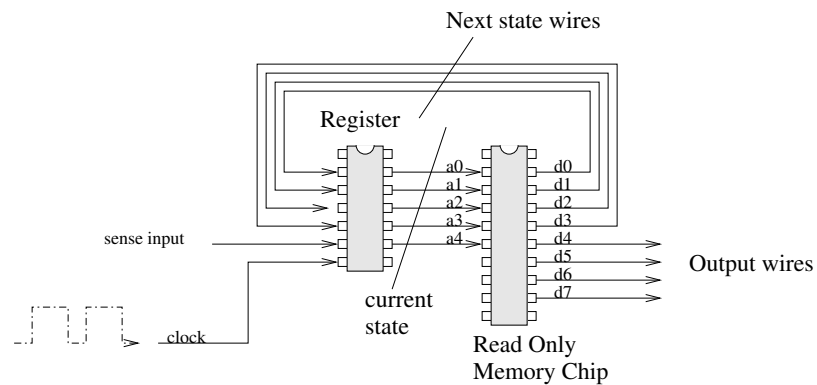


Figure 2.11: Finite state machine suitable for simple sequencing can be built out of a register chip and a read only memory (ROM) chip. In the picture the register latches a new address for the ROM each clock cycle. The ROM outputs a next state and a set of wires which go to control things like the lights in Figure 2.10. Thus each cycle it jumps to a new state. The behaviour of the machine is then defined by the table loaded into the ROM. One or more sense inputs can also be provided which provide additional address inputs. The sense inputs could be fed by things like the pedestrian crossing button. The presence of the sense inputs allows the next state transitions to be *conditional* on the sense input. Since the sense inputs are addresses, each sense input used means a doubling of the table size to handle both the true and false variants of the sense input.

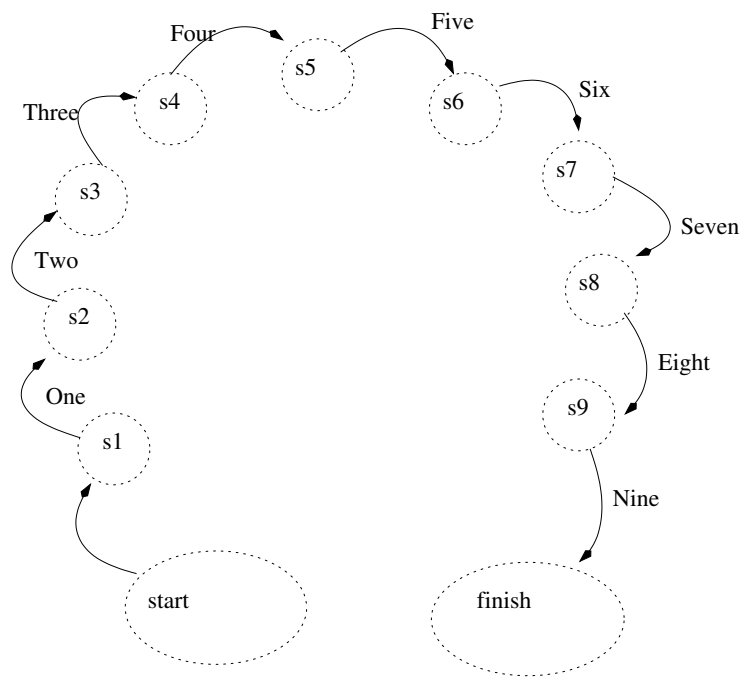


Figure 2.12: The repeated pattern invoked when counting aloud in English

twenty, one, two, three, four, five, six, seven, eight, nine, thirty, one, two, three,...

then what Chomsky called a class 2 or *context free* grammar would suffice for counting. Class 2 grammars have productions of the form:

$$a \rightarrow b$$

where a is a non-terminal symbol and b is some combination of terminals and non terminals. Class 2 grammars are more powerful than class 3 because they allow the use of nested patterns. We can for example write down context free grammar that defines how we say the numbers from 1 to 999999 in English. The grammar below is an extension and clarification of one given by Longuet-Higgins[64].

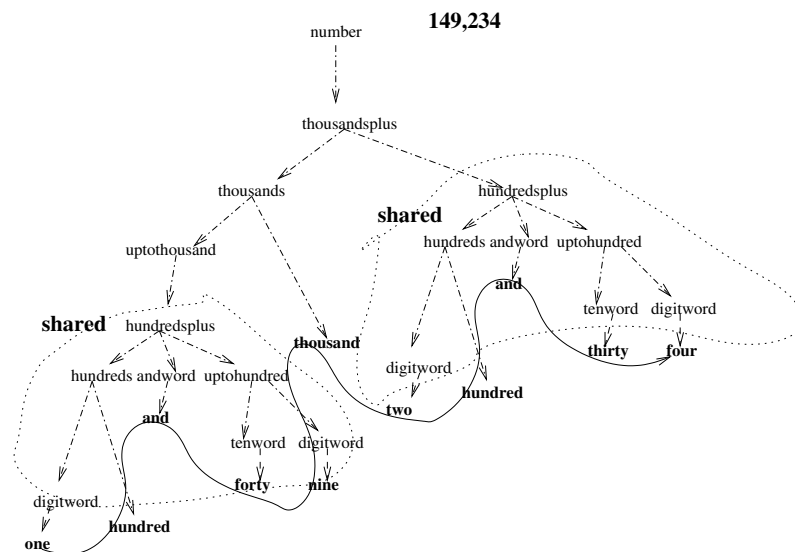
Lexicon of English Numbers

word	number	role
one	1	digitword
two	2	digitword
ten	10	firstten
nine	9	digitword
...		
eleven	11	teenword
...		
nineteen	19	teenword
twenty	20	tenword
...		
ninety	90	tenword
hundred	100	hundredword
thousand	1000	thousandword

Production rules

$\text{number} \rightarrow \text{uptothousand} | \text{thousands} | \text{thousandsplus}$
 $\text{andword} \rightarrow \text{and}$
 $\text{numberword} \rightarrow \text{tenword} | \text{teenword} | \text{digitword} | \text{firstten}$
 $\text{uptohundred} \rightarrow \text{numberword} | \text{tenword digitword}$
 $\text{hundreds} \rightarrow \text{digitword hundredword}$
 $\text{hundredsplus} \rightarrow \text{hundreds andword uptohundred}$
 $\text{uptothousand} \rightarrow \text{uptohundred} | \text{hundreds} | \text{hundredsplus}$
 $\text{thousands} \rightarrow \text{uptothousand thousandword}$
 $\text{thousandsplus} \rightarrow \text{thousands andword uptohundred} | \text{thousands hundred-}$
 plus

In the rules above we use the standard convention that $|$ marks an alternative production. Consider the number 149,234 which we pronounce one hundred and forty nine thousand two hundred and thirty four. We can draw a parse tree of how this can be produced by the grammar above:



The same grammatical derivation allows us to say the hundreds of thousands as allows us to say the hundreds. This grammar is certainly no more complex than we use in day to day speech, so saying an individual large number just involves learning a few additional rules of the type we are already familiar with. It is a well established principle of grammar theory[42] that context free languages can be produced by a stack automaton. A stack machine is the composition of an FSM and a stack onto which the state word can be pushed or from which the state word can be popped. This allows the automaton to perform nested operations. In the case above, the nested operation is the saying of a number in the range 1...999. Is this enough for counting?

No, because a classical stack machine can only look at the top of the stack. Suppose we are counting as follows:

one hundred and forty nine thousand two hundred and thirty four
one hundred and forty nine thousand two hundred and thirty five

...
 one hundred and forty nine thousand two hundred and thirty nine
 one hundred and forty nine thousand two hundred and forty
 one hundred and forty nine thousand two hundred and forty one

The italic letters indicate the words that we have to ‘stack’ in our mind to go onto the next step. The transitions between the final digits could be carried out using the simple automaton in Figure 2.12. This may look like a simple stacking operation, but it is not.

In order to say each number in the correct order we have to have access to all the stacked words so that we can repeat them whereas a classical stack machine can only see the top word of the stack. This means that the grammar required for counting, as opposed to saying one number, is *context sensitive*. But since natural languages do contain context sensitive components to their grammars, it is plausible that the process of counting rests upon the same linguistic

computing ability that is used to produce normal speech.

If we relied on the primitive ability of the motor control system to perform sequences of actions that correspond to regular grammars, then the ability to count would have been greater than that allowed by subitizing, but still small. By using our more advanced grammatical abilities we can overcome these limits and count much higher.

2.4 From aides memoir to the first digital calculating devices

The process of counting aloud is error prone. We can easily lose track of the numbers we are reciting. So the fallibility of our short term memory poses another limit, this time to unaided mental computation. Hence the reliance on tallies and counters for practical tasks. Tallies can be simple marks as in Figure 2.8 or can be mechanical devices. But these just count, the most basic computing operation. How did we get beyond counting to adding?

Well if you are counting with pebbles in jars, then all you need to do is pour the contents of one jar into the other and you have done an addition. This operation is reflected still in language. When baking a cake you add the sugar to the flour in the bowl. Physical addition as pouring is the primitive operation, arithmetic addition the derivative operation. If what you are pouring are counting pebbles, then pouring has the same effect as addition. Take two flocks of sheep into a single pen one after the other, and, supposing that you have pebble jars counting each flock on its own, then pouring one jar into the other gives you the total number of sheep in the pen. There is no mystery to this, just our reliance on sheep and pebbles not mysteriously vanishing.

Next assume that you have tokens in your jars, tokens which can represent different numbers of sheep as introduced on page 18. Again you simply add the contents of one jar to the other and you will have an accurate record of the number of sheep in the two flocks. Why does this work?

Because we have a procedure for translating the high valued tokens back into unit tokens.

We could either take both jars individually and whenever we find a 10 sheep token in the jar replace it with 10 individual sheep tokens, or we can wait till after we have put all the mixed tokens into a single jar before we do the translation. Whichever order we do it in, we end up with the same number of unit tokens in the final jar. Nowadays we teach children that $10(b + c) = 10b + 10c$, which is just an abstract way of talking about the same process.

The old practices persist. Who when collecting the coins from both pockets into one bothers to count the coins first and add their numbers?

No need. The old addition as pouring still works well enough. For business or state accountants though, more formal ways of dealing with collections of coins had to be developed, and for this purpose people reverted to using beads or stones, the value of which depended on their position on a reckoning board

or counting table, *abakion* in ancient Greek, an early one of which was found in Salamis, illustrated in Figure 2.13. It appears to have been designed to do additions in units of the then Greek monetary system of Talents, Drachma and Obols. Unlike counting with coins, where the numeric value is marked on the coin, with the *abakion*, the value assigned to a counter depended the row in which it was placed.

If two numbers are to be added tokens are arranged above and below the line as shown. To perform the actual addition, counters on the lower half are slid to the upper half, sliding the corresponding rows of counters in the upper half up to make space. This step has already performed the physical addition. The top half of the columns now contains the correct value, but it may be denormalised. That is to say some columns may have too many tokens in them – more than 9 for the drachmas, but more than 1 for the $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$ obol columns. Subsequently the drachma representation is renormalised by removing 10 counters from any column with more than 9 and adding one to the column to the left. The obol representation is renormalised by adding 1 drachma and taking away 6 obols if there are more than 5 obols in total.

The step from the Greek *abakion* to the more recent abacus is a small one. The main shift is that the Salamis *abakion* was a two register device, whereas most modern abacuses are single register. According to Menninger ([58] pp. 305-315), the Roman hand abacus, Figure 2.14, derived from the *abakion* spread to Asia where modified variants of it are still used. In the 20th century a hand operated device that was, in essence, an abacus was mass produced as a pocket calculator as shown in Figure 2.14(b). One of the authors remembers using these in school.

The Romans continued to use large reckoning tables like the Salamis one, on which they ‘calculated’ using *calculi* or small pebbles. The pebbles that first served as counters were later replaced by disks like those used in the modern game of draughts.

Multiplication

We have been arguing that computing techniques were ways that people predicted or modeled the world. This is clear enough with counting and adding which originate with the need to model movable possessions. The requirement to multiply arises as society becomes more complex. Architecture requires multiplication to determine quantities of bricks needed to construct walls, pyramids, tile floors etc. The organisation of work teams requires calculation total quantities of rations that people will eat. Extensive trade involves calculating the price of a cargo given the price of a unit. Suppose we know the sides of a rectangular floor to be tiled as 12 by 17. The 12 and the 17 can be represented in some way as counters, tokens, or in a written notation the answer likewise. A simple technique is to lay out pebbles in a regular rectangular pattern, 12 pebbles by 17, the resulting group of pebbles can then be carried to the tile maker, or counted and the result taken to the tile maker. This is direct physical modeling of the floor to be tiled with the pebbles, but at the same time it performs

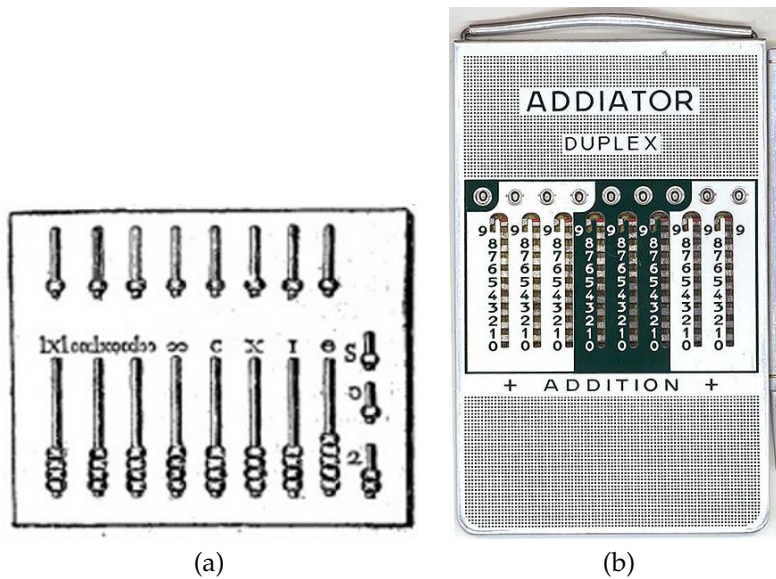


Figure 2.14: Pocket abacuses two millenia appart. (a) Diagram of a Roman hand abacus. Note that it holds two rows one to hold the units the other the fives; successive columns indicate powers of ten. The body was a flat bronze plate with movable studs which slid in slots. (b) The Addiator, an early 20th century pocket abacus that used sliders rather than beads and which was operated with a stylus. To add one inserted the stylus and moved the appropriate slider down, unless a red colour showed in which case one moved it up and round to perform a carry.

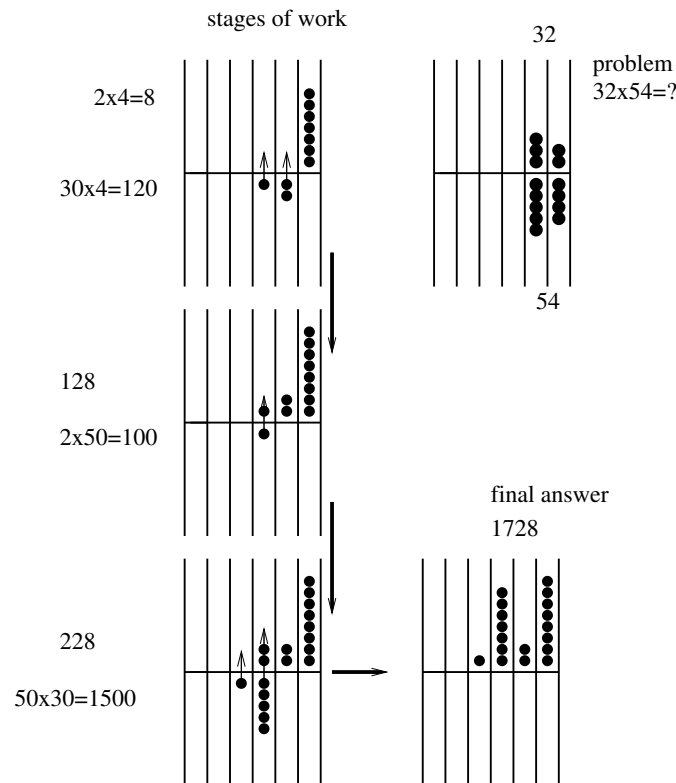


Figure 2.15: Multiplication on a 4 register reckoning table. The left pair of registers is used for working, the right to hold the two numbers to be multiplied. The working registers are shown in the successive phases of the calculation.

what we would now call multiplication, though we would scarcely recognise it as such.

Doing multiplication this way is slow. We tend to think of multiplication as being some form of short hand procedure which can achieve the same result. Menninger describes how reckoning tables were used to compute multiplications by a much quicker method. Figure 2.15 shows a 4 register reckoning table whilst 54 is being multiplied by 32.

Doing the multiplications requires that the operator learn by heart the multiplication tables up to 10 - just as children have to do now. Here we encounter another of the great abiding accelerators of computation - the lookup table. Tables can be memorised, or they can be written down - a large part of the Rhind papyrus, one of the first maths texts known consists of tables. Their great merit is that they remove the cost of repeated calculation, substituting for it the much easier task of looking something up. Here a temporal limit to

2.4. FROM AIDES MEMOIR TO THE FIRST DIGITAL CALCULATING DEVICES33

calculation is avoided by the use of a less complex process.

We can summarise the technologies for computation described so far in the following table:

Technique	Skill	Operations	Range
subitizing	inbuilt	counting	only to 4
counting on fingers	learned	counting adding	to 10
counting aloud	rote learning	count	to hundreds
unary tokens	simple learned	count,add	to tens
scaled tokens	recursive,learned	count,add	to thousands
base 60	recursive,learned	count, record	millions
reckoning tables	skilled,recursive	+, -, x	millions

Chapter 3

Mechanical computers and their limits

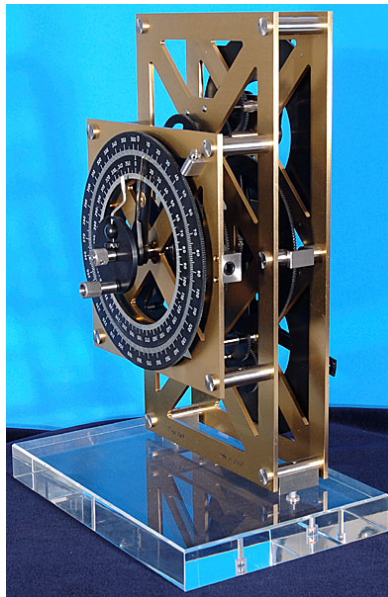
With large enough reckoning tables, papyrus for notes and enough time, very large and complex calculations could in principle be done. But the process of reckoning is slow and, when dealing with extended calculations, human error creeps in. To overcome these limits, mechanism was needed.

Mechanical computing started surprisingly early. The earliest known mechanical computer has been dated to between 150BC to 100BC[32], but given its sophistication it may be supposed that it had earlier predecessors.

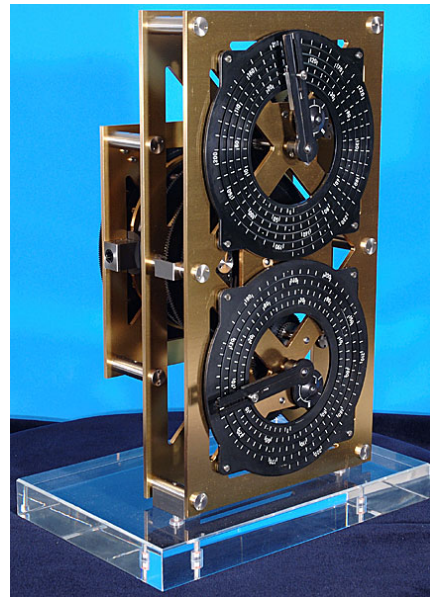
3.1 Antikythera

In 1900 a group of sponge divers sheltering from a storm anchored off the island of Antikythera. Diving from there they spotted an ancient shipwreck with bronze and marble statuary visible. A number of valuable artifacts were recovered and taken to the National Museum in Athens. Further diving in 1902 revealed what appeared to be gearwheels embedded in rock. On recovery these were found to be parts of a complicated mechanism, initially assumed to be a clock. Starting in the 1950s and going on to the 1970s the work of Price[21][22] established that it was not a clock but some form of calendrical computer. A key to this clarification was the use first of gamma ray photography and then of computer tomography[32] to reveal details of mechanism beneath the accretions of millennia of marine corrosion. A number of reconstructions have been made both physical and in software. Figure 3.1 shows one of these. It is now clear what the machine did, though why it was built remains a matter for speculation. Our description of its function follows [72] and [22].

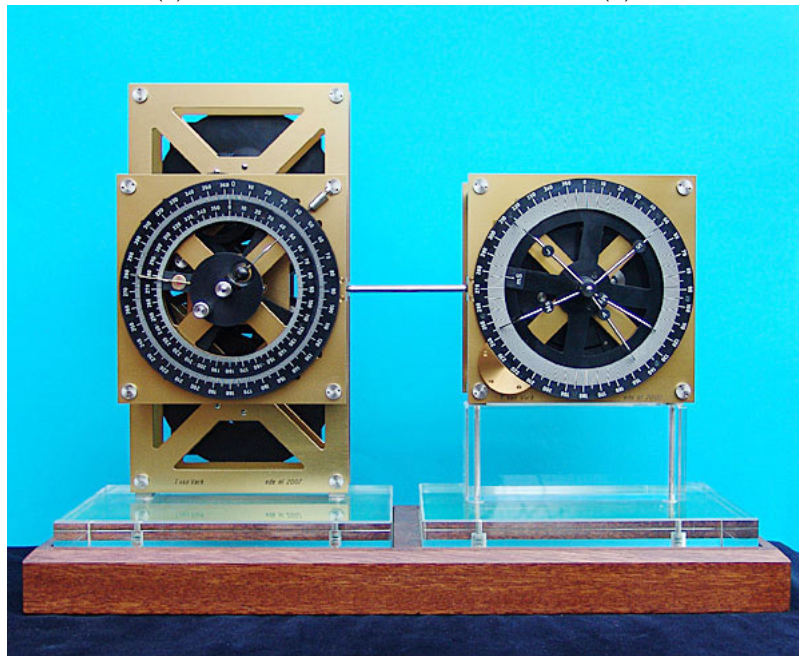
The machine had two faces: the front with a central dial, the back with two major and two subsidiary dials (Figure 3.1.(a)). The front dial had two pointers, one of which bore a solar, the other a lunar globe. The pointers showed the positions of the sun and moon both in the zodiac and relative to the calendar.



(a)



(b)



(c)

Figure 3.1: A modern physical reconstruction by T. van Vark, reprinted by permission of the builder. (a) The front dial showing the lunar and solar pointers against zodiac and calendar rings. (b) The rear face showing the Metonic and Saros cycle dials with spiral scales. (c) The mechanism linked to and driving the hypothesised planetary dials.

The calendar was denoted by a scale ring with 365 days. The scale could be advanced by one day every 4 years using a retaining pin which could be fitted into one of 365 holes in the frame behind the movable scale.

The back dials are more complex. The upper back dial shows what is called the Metonic cycle of 235 lunar months¹. This is the period in which the the lunar and solar calendars come into synchrony. 19 solar years correspond to 235 lunar months, so after the end of this cycle the phases of the moon will again be the same on the corresponding calendar day. In order to gain greater precision on the dial it is arranged in 5 deep spiral with a pin in a moving slider on the pointer going into the spiral. As the pointer turns the slider moves gradually outwards. The spiral arrangement increases the angular resolution of the dial.

The lower back dial uses the same spiral dial mechanism but in this case it displays the Saros cycle of 223 lunar months. The Saros cycle is relevant for predicting lunar and solar eclipses. If a lunar or solar eclipse occurs then another similar one will occur 223 lunar months later. The Saros cycle is $6585\frac{1}{3}$ days. Since it is not an integer number of days the geographical position from which eclipses are seen shifts by 120° each cycle. The 54 year Exeligemos cycle is needed to bring the geographical position of from which lunar eclipses can be seen back into alignment. For solar eclipse paths of visibility, the calculation is more complex. A small auxiliary dial shows the Exeligemos cycle. Another auxiliary dial shows the Callipic cycle which is four Metonic cycles less one day. The Callipic cycle improves the accuracy of reconciliation of the lunar and solar calendars.

The interpretation of the dials is based both on the fragmentary captions which have survived on their surfaces and on a mutually consistent reconstruction of the internal workings of the machine. The sophistication of the mechanism, when uncovered by Price, was astonishing, given what had previously been known about ancient Greek computing technology. It involves a large number of gear wheels with complex interactions. In order to understand these we need to grasp certain preliminary principles of how gearwheel computing works.

Addition The first point is to understand that wheels can implement addition by addition of angles. In Figure 3.2 we see a pair of concentric wheels with digits round the edge being used for addition. Assume that the two wheels are free to rotate both absolutely and relative to one another. To add 3 to 4 we first rotate the inner wheel until its zero is lined up with 3 on the other wheel, then read off the number on the outer wheel corresponding to 4 on the inner wheel. This basic mechanism was used in many ready reckoners and circular slide rules.

¹A lunar month is the period between corresponding phases of the moon in successive lunar cycles.

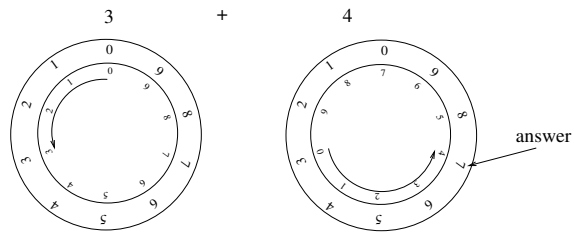


Figure 3.2: Wheels can do modular addition by addition of angles.

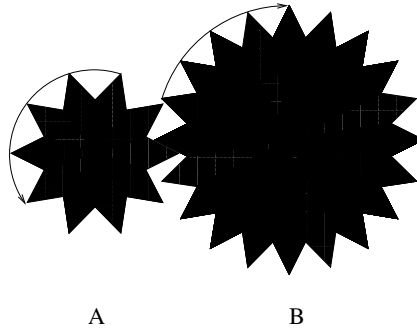


Figure 3.3: Multiplication by a constant. Rotation of wheel A by -144° will result in rotation of wheel B by 72° .

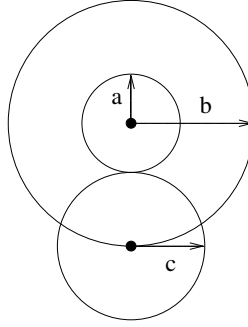


Figure 3.4: Differential gearing.

Multiplication by a constant. If two wheels contact tangentially and A has diameter a and B has diameter b then the wheel A will rotate at a rate of $\frac{-b}{a}$ times the rate of rotation of wheel B. If the wheels are cogwheels with matching sizes of teeth, then the ratio of their diameters will be the ratio of their tooth numbers. This is shown in Figure 3.3.

Note that this implies multiplication by rational numbers rather than arbitrary real numbers.

Differential gears If gear C is mounted on gear B, and if B and A rotate at the same rate, then so will C.

$$\Delta A = \Delta B \Rightarrow \Delta A = \Delta C$$

In the frame of reference of B the rotation of C will be $\frac{-c}{a}$ of the rate of rotation of A also in the frame of reference of B. But the rotation of A in the frame of reference of B is just $\Delta A - \Delta B$. So the full equation for the rotation rate of C is

$$\Delta C = \Delta B + (\Delta A - \Delta B) \frac{-c}{b}$$

or

$$\Delta C = \frac{c+b}{b} \Delta B - \frac{c}{b} \Delta A$$

This mechanism allows the computation of linear functions of differences between rates.

Non linear functions The 3 principles above would be sufficient to construct a mechanical model of the positions of the moon, sun and planets if these all followed uniform circular orbits. A problem arises however with the elliptical orbit of the moon. The angular speed of the moon is not constant. Kepler's law that equal areas are swept out in equal periods by its orbit, means that close to perigee its angular velocity against the fixed stars is greater than at apogee. This is termed the anomalous motion of the moon.

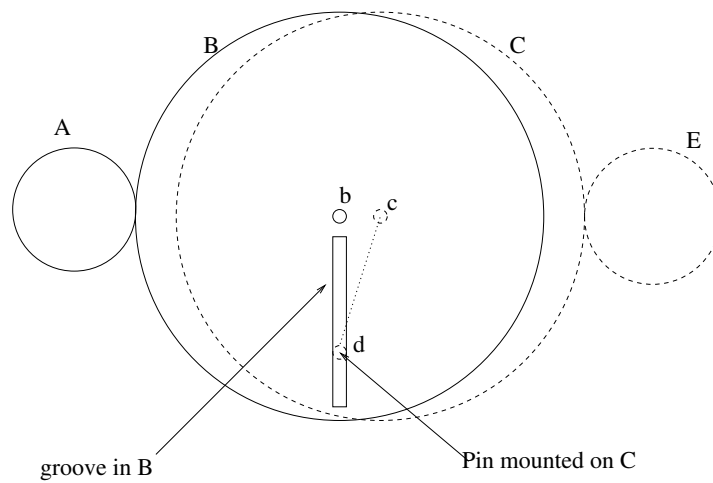


Figure 3.5: Variable speed coupling using eccentric axes. Wheel *B* has a different axis *b* than that of wheel *C*. A slot in *B* is engaged by a pin *d* mounted on *C*. *C* will complete one revolution for every revolution of *B*, but its rotation speed will change as *B* goes round, being fastest relative to that of *B* when the pin is horizontally to the right of *b* and slowest when it is horizontally to the left. When simulating lunar orbital motion, the more rapid orbital motion at perigee would be simulated by having the pin *d* horizontally to the right at perigee.

Apollonius of Perga (circa 262 BC to circa 190 BC) proposed to model the lunar motion by either of two models. In the one shown in Figure 3.6 involves the assumption that the moon rotated in a circular orbit around a point some distance from the center of the earth[60]. The other is the theory that we are more familiar with from Ptolemy, that the moon moved on an epicycle on an otherwise circular orbit. Apollonius proved the equivalence of the epicycle and eccentric models.

The major contribution to the anomalistic motion of the moon is the fact that the elliptical orbit is off center with respect to the earth. The deviation from circularity of the ellipse, is, compared to this relatively small. Thus an eccentric circular model gives a very good approximation of the lunar position. A computing machine like the Antikythera, which uses the mechanism in Figure 3.5 will give predictions of lunar position that are as accurate as the naked eye astronomy of the ancient world could observe².

Our current records of ancient astronomy do not indicate knowledge of Kepler's law at the time the Antikythera device was built. Despite this the device contains a mechanism to approximate the effects that we now attribute to Kepler's law. The technique used is to have two wheels with slightly different axes, coupled via a pin and slot mechanism like that illustrated in Figure 3.5.

The mechanical approximation mechanism used in the Antikythera device corresponds closely with the eccentric model of the lunar orbit. The line of sight from Earth to Moon in Apollonius' model (Figure 3.6(a)) is physically modelled by the groove in wheel B (Figure 3.5). The Moon in Apollonius model is represented by the pin d and the center of the lunar orbit by the axis c (both in Figure 3.5). If we compare the two figures we can see that the mechanism in Figure 3.5 directly implements the Apollonius' model of the lunar orbit. The mechanism is a simplified representation of a component actually used in the Antikythera device.

It is easier to construct an eccentric coupling of the sort shown in Figure 3.5 than it is to make a direct mechanical implementation of Kepler's law. The choice of model proposed by the Greek astronomers to explain the lunar orbit may thus have been influenced by what they could feasibly build into a computer. Apollonius of Perga who proposed the eccentric model for lunar orbits also developed the theory of ellipses in his work on conic sections so he would have been well equipped to propose ellipses as a basis for orbits. The preference of the Greek astronomers for circular motions might have been influenced by a concern for mechanisation.

We talk about scientists having a 'model' for some physical process that they seek to explain. By this we tend to mean a conceptual model. But a conceptual model does not produce numerical results until the conceptual model is implemented in a physical model. The Antikythera device is a beautiful illustration of the interaction between conceptual and physical models. There is a one to one correspondence between geometrical mechanisms proposed by the ancient astronomers, and the physical geometry of one of the computing

²For a heterodox view of the optical technology available to Greek astronomers see[74].

machines they used. Today, the correspondence is less evident since we use general purpose computers. But the correspondences will still be there. The conceptual model will now be expressed algebraically, and that algebra will be translated into expressions in Fortran, Matlab or some other computer notation for explicit calculation. The difference is that we can now have explicit representations of algebra rather than explicit representations of geometry in our machines.

With a device like the Antikythera mechanism, predicting eclipses became just a matter of turning a handle, something far simpler and less error prone than the numerical solution of kinematic equations using reckoning tables.

3.1.1 Was the Antikythera device an analogue computer?

In the web literature there are references to the Antikythera device as an analogue computer. Is this correct?

It depends on how we understand the term analogue. One sense of analogue, is to bear analogy to. The Antikythera device is built in such a way as to be analogous to operations of the cosmos, as understood by its builders. So in this sense it is an analogue device.

Another interpretation of analogue computer is one that works by means of continuous rather than discrete quantities. Again, in this sense, the antikythera device is analogue, since the dials on the front give continuous readings of lunar and solar angles.

But if we look at it from a different aspect, the device is digital, that is to say it performs mathematics in discrete operations. Look back at Figure 3.3. The ratio that this pair of gearwheels computes is always going to be a rational number, a ratio of the integral numbers of teeth on the two wheels. The multiplications performed by the machine were thus digital.

It is also arguable that its mode of operation was semi-digital, in that it seems to have been advanced by one 'clock cycle' per day. Each day the handle on its main driving wheel was rotated once, and the date pointer on the face would move on by an amount equal to one whole day. Because the internal multiplications were by rational numbers, the machine would not have drifted over time. Having completed one Metonic cycle, it would have gone on to compute the next with equal accuracy. Insofar as the machine drifted with respect to the actual motions of the moon, it would be because the gearing ratios were not specified to a sufficient number of digits. This drift, of the calculated position of the Moon from its observed position over multiple cycles, is characteristic of digital calculations. Digital calculations must always express ratios as ratios of whole numbers, and to get more and more accuracy when simulating some natural process we are forced to use larger and larger numbers in the numerator and denominator.

In modern terms, the Antikythera device did digital arithmetic, but used analogue input and output devices. The analogue scales used for the answer, although in principle continuous, would in practice usually have been read of as discrete numbers of days. Had one wished for more precision, an additional

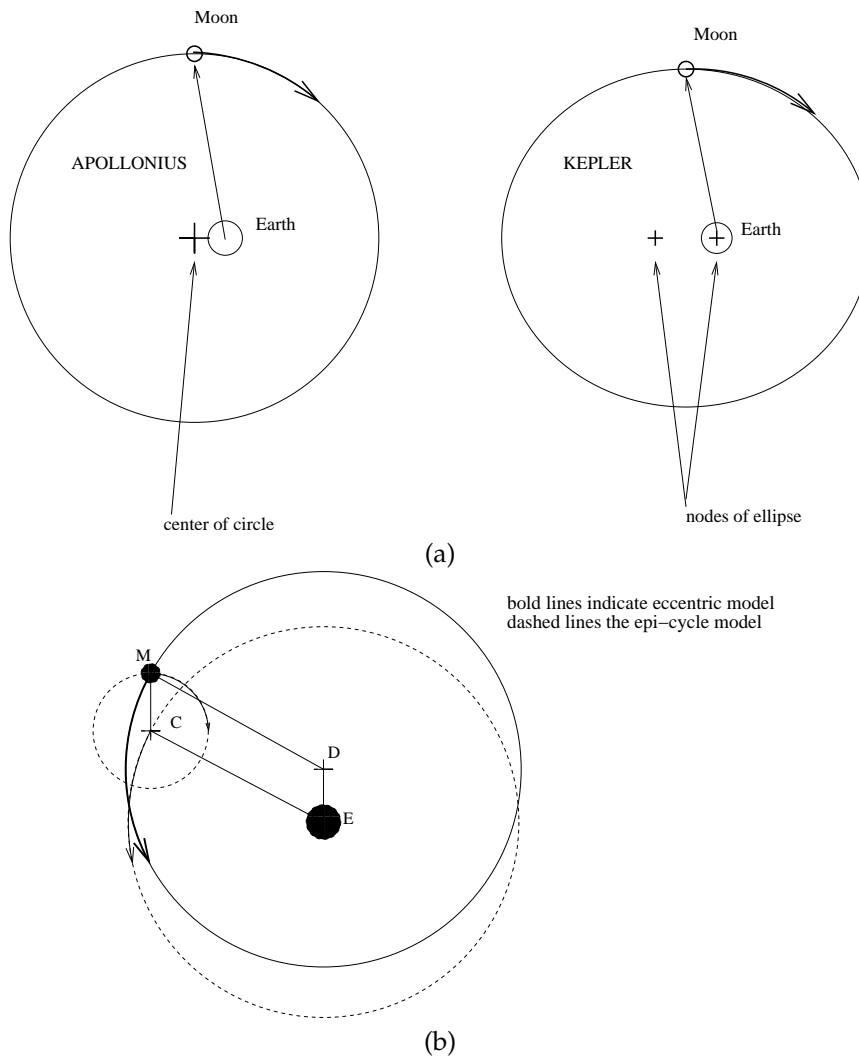


Figure 3.6: (a) The contrasting models of lunar orbits proposed by Apollonius and Kepler. (b) Apollonius proof of equivalence of the epicyclic and eccentric models. E is earth, M moon, D the center of orbit of the eccentric model, C the center of the epicycle. If the rotation of the epicycle is exactly the same magnitude but in the reverse direction to the orbital rotation, then CM must be parallel to ED, EDMC must be a parallelogram, and so the path of the moon is in each case a circle centered on D.

dial, which rotated faster could have been used. This is the principle used in clocks. A clock has a solar pointer (the hour hand) that completes two rotations every apparent rotation of the Sun. To enable us to be able to tell the time to greater accuracy we then have minute and second hands that rotate 12 times and 720 times faster. By appropriate multiplication of scales, a cog-wheel computational mechanism can add least significant digits to its output. Clocking down the main output, can add leading digits to the output : Antikythera does this with its Callipic cycle dial.

Limitations The techniques used in the Antikythera device were limited to the performance of the operations of angular addition and subtraction along with multiplication by negative constants. Final readout accuracy was limited by the mechanical precision of dials. Arithmetic is modular.

3.2 Late mechanical computers

The main use of computers in the first half of the 20th century was for warfare, particularly for naval warfare. It was here that the greatest computational demands lay. The development of long range guns and steam power meant that by the start of the 20th century it was no longer possible for navies to simply aim their guns using open or telescopic sights. The ranges at which successful firing between warships occurred rose from 7km at the Battle of Tsushima in 1905, to 25km at the Battle of Calabria in 1940. By 1914 the speeds of ships had risen so much that their relative velocities could exceed 100kmph.

At ranges above a few kilometers, shot from naval guns came down in a plunging fashion. To hit another ship it was not sufficient to aim directly at the ship, gunners had to know the range of the target ship and aim their guns high so that the shot would fall on the target. This problem could be broken down into a number of sub-problems.

1. The bearing of the target had to be found.
2. The range had to be estimated. Initially this was done using stereo matching (See Figure 3.7(c)). Later radar was used to get a more accurate estimate.
3. The relative velocity of the two ships then had to be estimated. The initial shell velocity might be around 800m/s. A shell might be in flight for 20 or 30 seconds. A ship moving with a relative velocity of 60kmph to the firing ship could have moved some 500meters during the time of flight.
4. The range and bearing that the target would be at the end of flight had to be estimated.
5. The up to date range and bearing had to be translated into an appropriate elevation and deflection of the gun before firing.

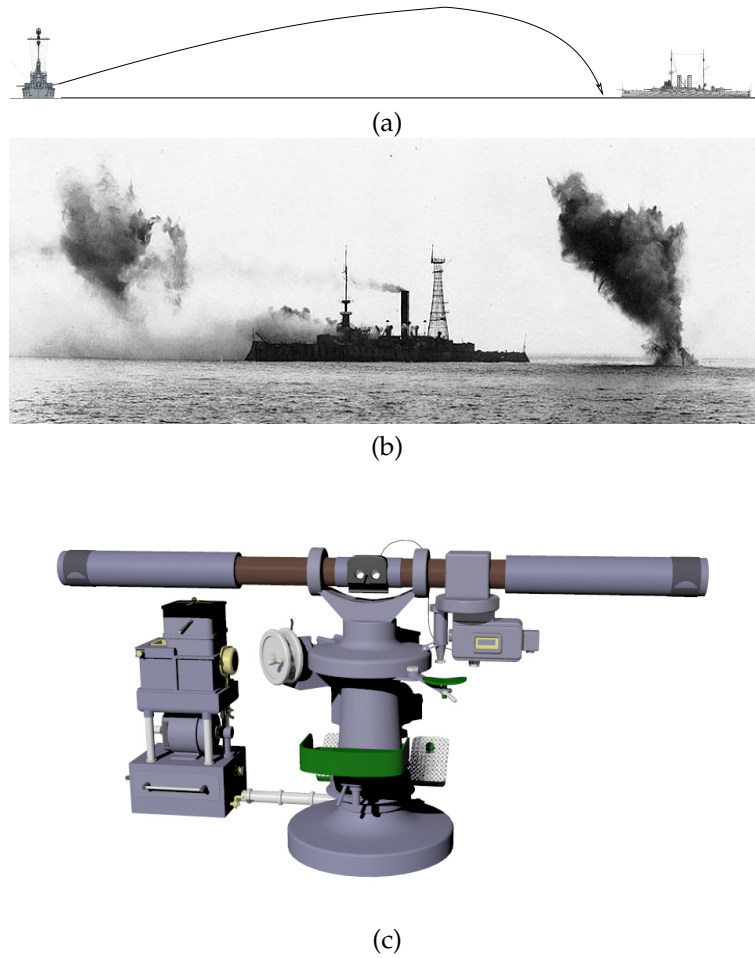


Figure 3.7: Gunners had to have an accurate estimate of range to prevent shot from falling short or far. (a) Schematic representation of fall of shell. (b) A salvo straddling USS Iowa in 1923 during ranging practice. (c) A model by Rob Brassington of the Argo Pollen gyro stabilised stereoscopic rangefinder used by the Royal Navy.

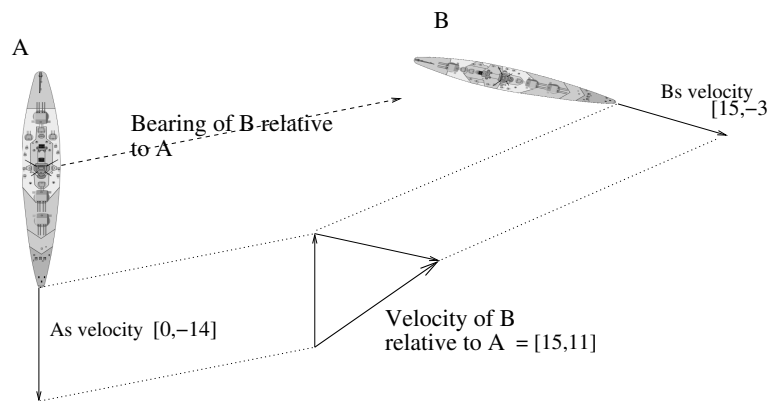


Figure 3.8: Vector addition of the relative motion of two ships.

All these calculations had to be done in real time, with the time between shots being of the order of 30 seconds. Early in the century it was realised that it was quite impractical to do the calculations by hand, both because that would have been beyond the trigonometrical ability of most gunnery officers, and because it would have been far too slow. The well funded admiralties of the day could command the attention of skilled engineers, scientists and instrument makers, so it was not long before mechanical solutions were devised.

The techniques that were arrived at, 2000 years after the Antikythera device, show a strong family resemblance to that earlier real-time calculator. They have a similar mode of display: based on dials; similar construction: brass frames and gearing; many of the same mechanical implementations of arithmetic techniques. We will focus on a couple of the devices invented for the purpose, for a more comprehensive account readers should consult Brooks[9].

3.2.1 Estimating rates of change

A moving ship has a course bearing and a speed. Together these can be represented as a vector. In modern computing we tend to think of vectors as a list of numbers, so a ship steering a course due South at 10 meters per second could be described by the pair $[0, -10]$ these being the components of its velocity in a cartesian coordinate system whose x axis was aligned with the equator. A ship sailing West at the same speed would have a velocity vector $[-10, 0]$ and a ship sailing North East a velocity vector $[7.07, 7.07]$ since it would be moving at just over 7 meters per second North and a similar amount East. Now let us look at the concrete example in Figure 3.8. Ship A is moving with velocity vector $[0, -14]$ and ship B is moving with a velocity vector $[15, -3]$. We want to know the combined velocity of B relative to A. To do this we subtract the velocity of A from that of B: $[15, -3] - [0, -14] = [15, 11]$.

This gives us the velocity of B relative to A in world coordinates. In order to

aim a gun we need to break this velocity down into two components, one along the line of sight, and one perpendicular to the line. If we were doing that now we would use multiplication by a rotation matrix to get this effect. Back in the early 1900's that would have involved a prohibitive amount of equipment but a naval officer Lieutenant Dumaresq[24] came up with a device, that, although at first sight a bit baffling, actually uses some beautifully simple principles.

Look at Figure 3.9(a). It shows the first principle: vector addition by direct geometric implementation. The Dumaresq consisted of a circular frame graduated its upper edge from 0° to 180° , Red and Green relative to the Fore and Aft Bar which is mounted on the frame. The frame itself was fixed with this bar parallel to the Fore and Aft line of the ship.

Under this bar a slider was mounted which traveled along the bar carrying an index cursor which could be moved aft from the centre along a scale graduated from 0 to 35 knots. This scale measured the ships own speed. On the slider was a rotating ruler, also marked in a scale of knots. The ruler had to be aligned with the enemy course relative your own ship's course, that is to say the angle between the bar and the ruler had to be the same as the angle between your course and the enemy course. Since the ruler is scaled in knots, a line from the Dumaresq center to the point on the ruler corresponding to the enemy speed will be the enemy's relative velocity vector.

The Dumaresq modeled geometry with geometry and avoided the detour into pairs of numbers that we would do on a modern computer.

There remained the problem of how to transform a relative velocity vector in the frame of reference of the ship into one in the frame of reference of the gun. We said that the modern technique would be to multiply the vector by a rotation matrix. The 1905 solution was to use a rotating disk on which was marked a system of cartesian coordinates giving velocity parallel with and normal to the line of sight. The disk was rotated until the arrow on it was aligned with the target, after which the range rate and deflection normal to the line of sight could be read off the scale. The task of the modern 'rotation matrix' was achieved using a matrix of lines that physically rotated.

3.3 Analogue mechanical multiply/accumulate

We now come to another new technique used in the gunnery computing devices, an integrator. Up until now we have discussed mechanical multiplication by a constant, achieved via gearing ratios. If we consider the problem faced by those dealing with ranging we can see that this would not have been enough.

Range estimates via the stereo rangefinders came through intermittently, and the gunners had to have intermediate estimates of range. Suppose we know range r_0 at time t_0 and using the Dumaresq we have an estimate r' of the rate at which the enemy ship is approaching along the line of sight. We need

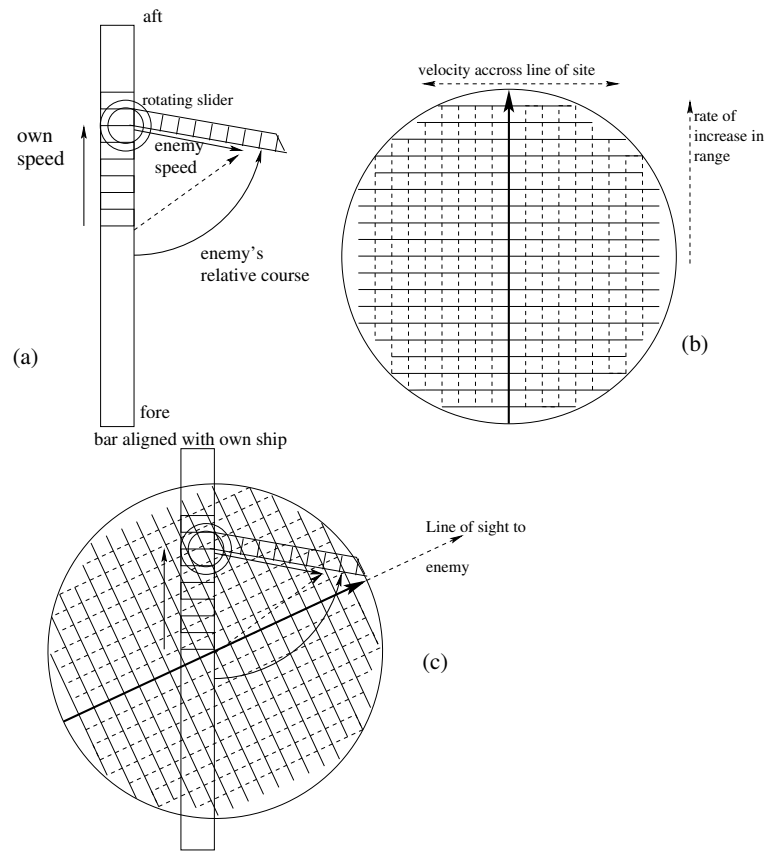


Figure 3.9: Operating principle of the Dumaresq.

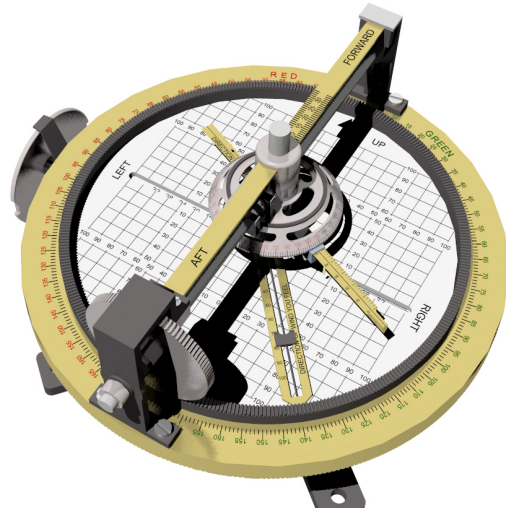


Figure 3.10: A Dumaresq analogue mechanical computer used in naval gunnery computations. The one illustrated is a Wind Dumaresq which in addition to giving corrections for relative motion of the ships also provides an additional compensation term due to the wind. Image from a model by Rob Brassington, by his permission.

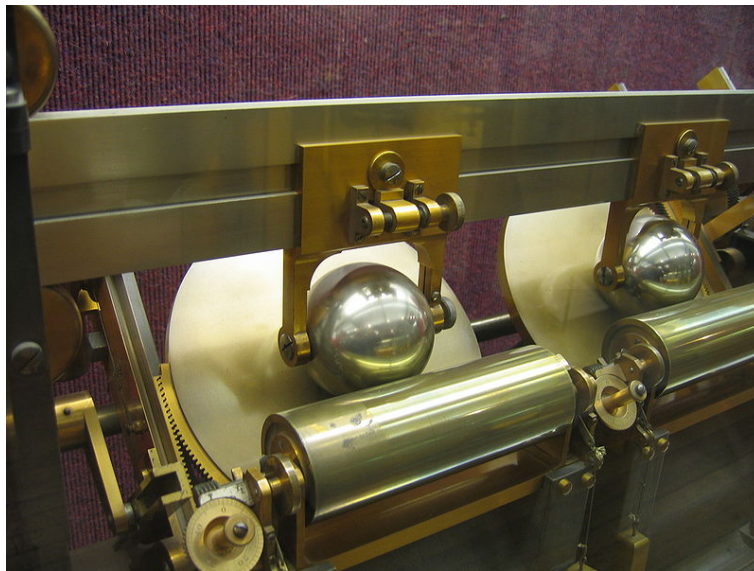


Figure 3.11: The original integrator developed by Thomson.

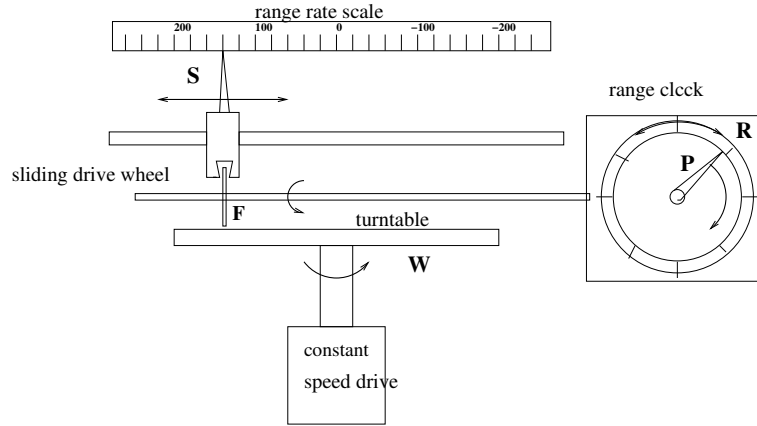


Figure 3.12: A mechanical integrator as used in range clocks and later in differential analysers.

to calculate

$$r_t = r_0 + \int_0^t r' dt$$

A mechanical integrator had been developed by William Thomson, later Lord Kelvin and his brother James Thomson in 1875 [75, 76], an illustration of his design is shown in Figure 3.11. His one was built into a more complex apparatus able to extract Fourier components of a signal. We will describe the simpler version used in the Vickers Range Clock[20]. Look at Figure 3.12. We have a flat horizontal wheel or turntable driven by a clock such that it rotates once every minute, and that we have a slider **S** we can use to record r' say in 100meters per minute. The slider carries a friction roller **F** that bears on the surface of **W** and is driven round by it. If we move the slider, then the friction roller moves with it. As it is moved closer to the center of the turntable the rotational velocity of **F** will fall, as the relative gearing rate between **F** and **W** alters. If **F** is moved past the axis of **W** to the other side, it will rotate in the reverse direction. The effect is to provide a variable multiplication of the rotation of the time clock by the range rate. This was then transferred to the range clock dial which moved at a rate corresponding to the approach or recession of the enemy.

Whenever an actual range observation, the rotating dial of the range clock **R** was moved round in order to bring the observed range into alignment with the range clock pointer **P**. The range clock would then point to the current range and would give estimates of the likely range as it changed.

The representation in Figure 3.12 is very schematic, actual integrators had a more compact arrangement often with the range clock dial above and coaxial with the turntable.

The basic technology of Dumaesq and Range clock had been developed by 1906. In the next decade, the pressure of the arms race meant that mechanical computer technology advanced rapidly. Whereas originally, the transfers of information between Dumaesque and range clock were manual, they became increasingly automated with information being transmitted by what amounted to asynchronous digital serial transmission between different parts of the ship. By the middle of the Great War, highly integrated machines like the Dreyer Table and the Argo Plotter Clock were in use (see Figure 3.13). The naval integrators derived from an earlier one built in 1876 by James Thomson, brother of Lord Kelvin who acted as a consultant to Arthur Pollen in the development of the Argo clock. Thompson's original mechanism had been used by Kelvin to analyse tidal motions. The integrator mechanism used in the range clock was later applied by Hartree[39] to more general scientific calculations during the 1930s.

The machines developed during the Great War provided the basis for a later generation of mechanical computers used by leading navies during the war of 1939 to 1945. Dumaesqs remained in use on small ships, but vessels above the size of a destroyer were fitted with machines descended from the Argo clock. These typically combined mechanical computations with digital electrical input and output. These mechanical gunnery computers remained in use in US Navy battleships until the first Gulf War.

Limitations These were all essentially mechanical analogue machines and were limited in that:

- Their accuracy would be limited to around 3 decimal digits of precision. Given that the margins of error of the original input data were much worse than this, the limited precision of computation was not significant.
- They were dedicated to performing a single suite of tasks.

Their most significant advance was the ability to multiply variables and to integrate.

In the rest of this chapter we will consider how the limitations in accuracy were addressed by the developers of mechanical computers. In the next chapter we will look at how the concept of a general purpose machine was arrived at.

3.4 Mechanising the abacus

There is a line of development that stretches from the Antikythera device, via clocks, down to the fire control computers of the 20th century. It is family of computing machines that developed to meet the needs of astronomy and physics. There was a distinct line of development that took off from the abacus and was initially dedicated to commercial calculations. An abacus performs exact integer arithmetic and can, by adding more rows to the device, be built

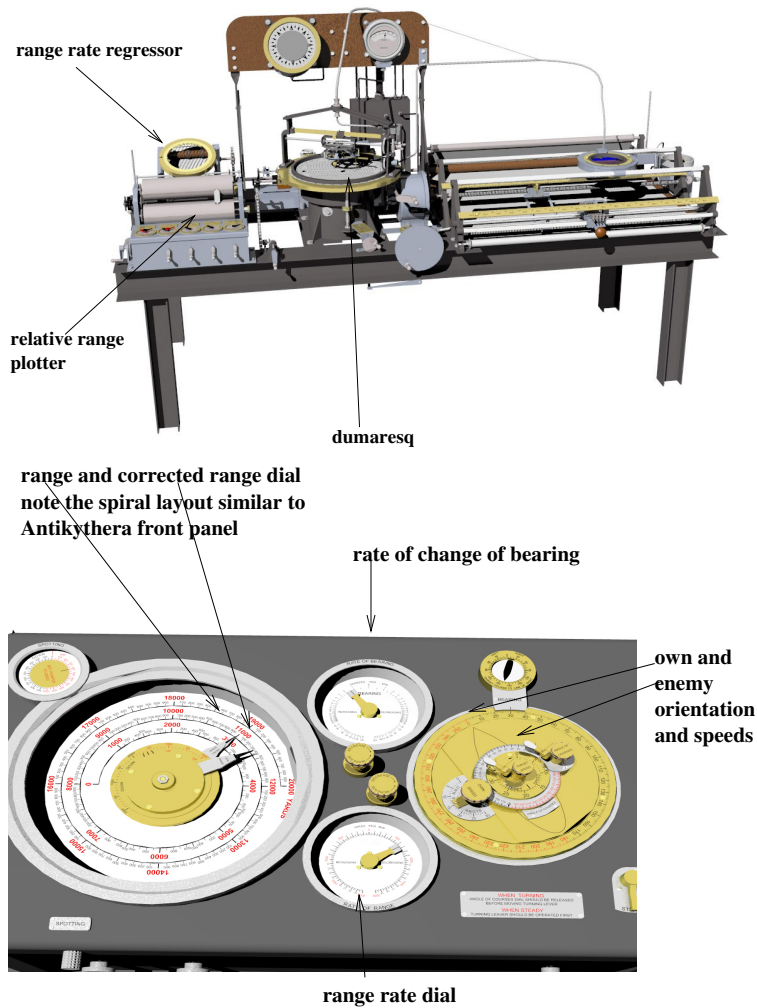


Figure 3.13: Mechanical computers used during the Great War. **Top** the Dreyer Table. In the middle is an automatic Dumaresq which controls a range integrator whose turntable is built in to the circular support of the Dumaresq. On either side are graph plotters which produce traces of the relative courses of the enemy ship. Scatter plots on these record the fall of shot, and a regression of these could be estimated using a rotating disk with scribed parallel lines. **Bottom** the control panel of the Argo Pollen range corrector. Both images are renders of digital models by R. Brassington.

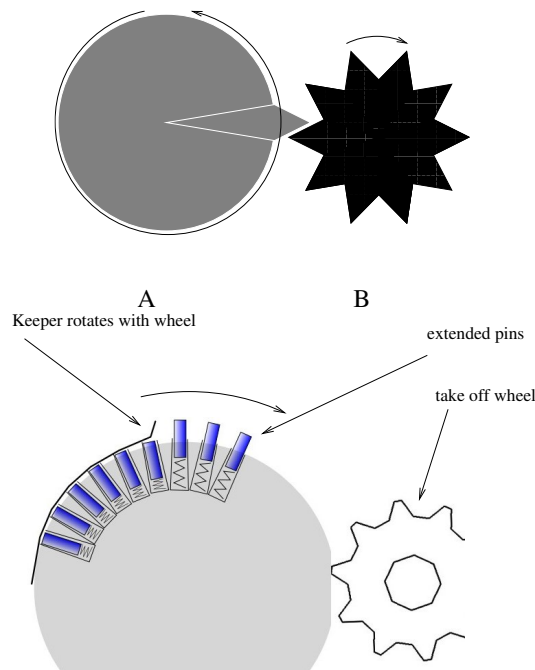


Figure 3.14: Top: use of a single toothed wheel as a carry mechanism. 1 rotation of A generates $\frac{1}{10}$ or a rotation of B. This was later generalised by Odhner who in 1874 invented the pinwheel. Bottom: use of a pinwheel to add a variable amount to the take off wheel.

to calculate to any desired number of significant figures. Such precision was essential to commerce. Suppose you are a 17th century banker lending 100,000 of Thaller at an interest rate of 2.5%. You would expect to get back 2,500 Thaller. If one did calculations using a machine accurate to only 3 significant figures, then all it would tell you is that you ought to charge between 2,000 and 3,000 Thaller, which makes a huge difference to your profit.

A chain of gears linked in successive 1:10 ratios, will allow numbers to be recorded accurately to a high degree of precision, but they only allow addition using the least significant digit wheel. If you want to repeatedly add numbers less than 10 to a running total, that is fine. But suppose you have a sequence of 6 such geared wheels and you want to add 20,000 to the total. You try and move wheel 5 on by two positions. What happens?

You find that the wheel is stuck since in order for wheel 5 to move round by $\frac{1}{10}$ of a rotation, the least significant wheel must rotate 1000 times. Friction will make this impossible. A means of overcoming this was invented by Schickard who in 1623 designed the first digital mechanical calculator using a gearing mechanism based on the principle shown in Figure 3.14. This allowed a carry

to take place between successive digit wheels such that as the units wheel made one rotation it would advance the 10s wheel by one place. On the other hand a rotation of the 10s wheel by one place would, in general, not move the units wheel since they would be decoupled. In 1645 Pascal developed an alternative carry mechanism which involved the units wheel raising a weighted arm which then fell on the tens wheel when the transition from 9 to 0 occurred. This further reduced the mechanical force required to propagate a carry[65, 45].

Limitations These early digital calculators were limited to performing addition. If you wanted to perform subtraction one had to use complement arithmetic. Suppose that you have a 3 digit Schickard or Pascal type machine and want to subtract 7 from 208, you first subtract 7 from 1000 in your head to give 993 and then do the sum $208 + 993 = 1201$ on the machine. But since the machine is only 3 digit, the thousands digit does not appear and we get the answer 201.

3.4.1 Digital multiplication

Multiplication on an abacus or reckoning table as described on page 32 required the user to memorise multiplication tables. It also involved a nested loop algorithm. Given 4 reckoning table registers, A, B, C, D with A, B containing the numbers to be multiplied, D the total and C a temporary. Registers D and C contain no beads initially. One proceeded as follows:

```
for i = 1 to number of digits in A do
  for j = 1 to number of digits in B do
    in your head set temp = A[i] * B[j] this yields a 2 digit number
    place this number in positions C[i+j] and C[i+j-1]
    Add the contents of register C to register D using standard method
```

If one wanted to do this mechanically, a major objective would be to obviate the need to memorise multiplications. An algorithm involving no mental arithmetic would be to use repeated addition to gain the effect of multiplication:

```
for i = 1 to number of digits in A do
  for j = 1 to number of digits in B do
    for k= 1 to A[i] do
      C[i+j-1] = B[j]
      Add the contents of register C to register D using standard method
```

This could have been done on a Pascal machine but it now involves 3 nested loops. We can reduce this to 2 loops if we have a more sophisticated addition mechanism. Let n be the number of digits in B.

```
for i = 1 to number of digits in A do
  for k= 1 to A[i] do
    C[i..i+n-1] = B[1..n]
    Add the contents of register C to register D using standard method
```

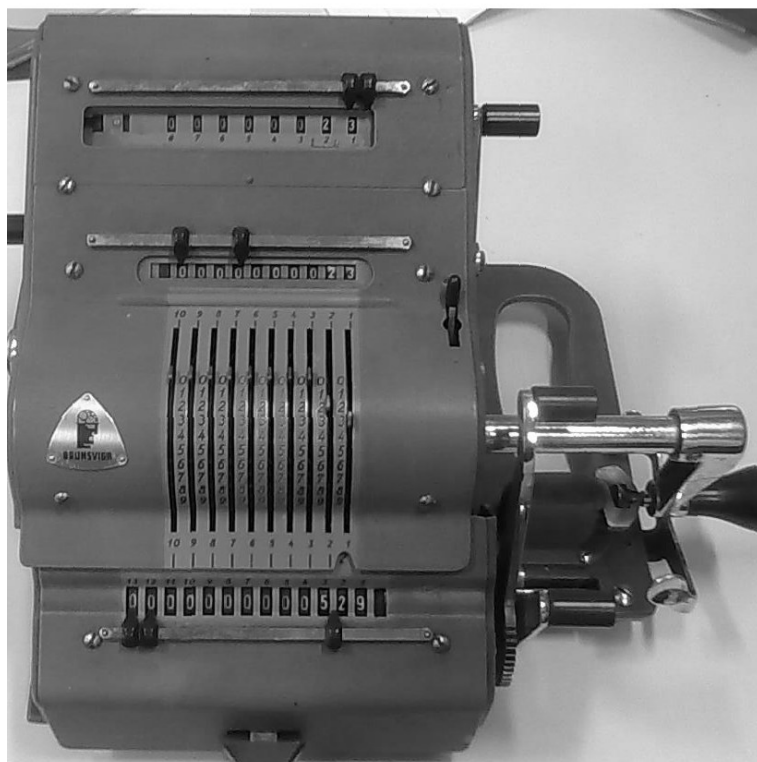


Figure 3.15: Brunsvega Model 13 pinwheel calculator showing the result of multiplying 23 by 23 using the algorithm of Leibniz. The register we have termed D is at the bottom and can slide sideways with respect to the barrel of pinwheels in the center. The barrel has 23 set on it. The barrel is the register B in our description of the Leibniz algorithm. A rotation of the large handle on the right will add the contents of the barrel to the appropriately shifted register D. The register at the top, A in our description, holds a count of the number of additions performed in each position. A reverse rotation of the handle performs subtraction. Produced 1952 - 1964.

The key step here is to transfer the whole of B into an appropriately shifted position in C and add it to D in a single step. It was Leibniz achievement to see a way to building a machine to implement this algorithm[65]. Assuming that all numbers are held as gear positions, he had to make two innovations:

1. The sliding carriage allowed number B to be placed in a mechanical register which could be moved left and right with respect to D.
2. The stepped gearwheel allowed number B to be held in a form which allowed a single rotation of register B would add register B to register D. The stepped wheel solved the problem of having a variable number of cogs by using a wheel in which the number of cogs varied along its length. A normal gear wheel moved along the stepped one's length would engage a different number of cogs depending on its position.

An improvement on the stepped wheel was the invention of the pinwheel in Figure 3.14 which allowed a more compact device.

Variable toothed wheels of different designs along with sliding carriages were incorporated in mass produced commercial calculators from the 19th century until they were replaced by electronic ones in the 1970s. Figure 3.15 shows a widely used model of pinwheel calculator that remained in production until the 1960s.

Chapter 4

History of computability debate *Greg*

Chapter 5

Heat, information and geometry

Paul and Lewis

5.1 Triumph of digital computation

Although Turing had advanced the idea of the universal digital computer during the 1930s, there was right up to the end of the 1960s an active tradition of analogue computing which was only gradually replaced by digital techniques. The tradition of analogue gun control computers continued, as we have mentioned¹, well beyond the 1940s. Through the 1950s and 1960s analogue electronic computers were serious rivals to digital ones for many tasks. The analogue approach ultimately lost out because of the twin problems of accuracy and programmability.

Analogue machines were significantly less accurate than digital ones. Wass[80] stated that

“Errors as small as 0.1% are hard to achieve; errors of 1% are not unusual, and they are sometimes as large as 5%-10%. This is in striking contrast to digital machines,” (page 3)

This meant that analogue computers had to be applied in areas where high accuracy was not as important as speed of operation. They were used in aerodynamic control applications such as missile guidance where their errors could be tolerated. In these applications the basic aerodynamic coefficients of the equations being solved were known to only about 10% accuracy, whereas the real-time response needed could not be attained by then existing digital machines.

Analogue machines were programmed by plug boards. Units to carry out operations of adding, subtracting, multiplying, differentiating etc were interconnected by removable wires in the same way as in a manual telephone exchange. Output was typically in the form of real time CRT traces of the graph of

¹Page 51.

the function being computed. Storage of analogue quantities remained a constant problem. MacKay[56] describes the use of modified Williams[82] tubes as analogue stores. Bergman[3] found that over a process of 8 read/write cycles with such the cumulative error amounted to 10%.

Sources of error in such machines were

1. Systematic scaling errors due to uncertainty in the parameters of the resistors and other passive components being used.
2. Distortions due to non-linearities in the amplifiers used.
3. Random fluctuations due to environmental factors. These include thermal noise in the amplifiers, and in the case of Williams tubes the effects of stray magnetic fields on the trajectories of the electrons.

These problems are representative of the difficulties that plague any attempt to carry out accurate analogue computation.

5.2 Analogue computing with real numbers

In principle an analogue device can be thought of as computing with real numbers. These real numbers can be divided into two classes, the parameters supplied as inputs to the calculation and the variables that take on time varying values as the computation proceeds. Whilst in a digital machine the parameters and the variables are uniformly represented by strings of bits in some digital store, in an analogue machine they are typically of different types. The variables in an electronic analogue computer for example are instantaneous voltages, whilst the parameters are provided by the physical setting of resistors, capacitors etc. These components are subject to manufacturing errors and the provision of higher specification components becomes exponentially expensive.

“That the machine is digital however has a more subtle significance. It means firstly that numbers can be represented by strings of digits that can be as long as one wishes. One can therefore work to any desired degree of accuracy. This accuracy is not obtained by more careful machining of parts, control of temperature variations, and such means, but by a slight increase in the amount of equipment in the machine. To double the number of significant figures, would involve increasing the amount of the equipment by a factor definitely less than two, and would also have some effect in increasing the time taken over each job. This is in sharp contrast with analogue machines, and continuous variable machines such as the differential analyser, where each additional decimal digit required necessitates a complete redesign of the machine, and an increase in the cost by as much as a factor of 10.” [79]

The parameters and variables are, in the mathematical abstraction, transformed by operators representing addition/multiplication etc. In an actual analogue computer these operators have to be implemented by some apparatus whose effect is analogous to that of the mathematical operator. The analogy is never exact. Multipliers turn out to be only approximately linear, adders show some losses etc. All of these mean that even were the variables perfect representations of the abstract concept of real numbers, the entire apparatus would only perform to bounded accuracy. But no physically measurable attribute of an apparatus will be perfect representation of a real number.

Voltage for example, is subject to thermal and ultimately quantum noise. We can never set up an apparatus that is completely isolated from the environment. The stray magnetic fields that troubled Bergman will never totally vanish. It may be objected that what we call digital computers are built out of transistors working with continuously varying currents. Since digital machines seem to stand on analogue foundations, what then privileges the digital over the analogue?

The analogue character of, for instance, voltage is itself a simplification. The charge on the gate of a transistor is, at a deeper level, derived from an integral number of electrons. The arrival of these electrons on the gate will be subject shot noise. The noise will follow a Poisson distribution whose standard deviation $\approx \sqrt{n}$, with n the mean number of electrons on the gate. It is clear that by raising n , making the device larger, we control the signal to noise level. For large enough transistors this noise can be reduced to such an extent that the probability of switching errors becomes negligible during the normal operation of the computer. It is only if we make devices too small that we have to bear the cost of lower reliability. We look at this in more detail in section 5.7.1.

Suppose we have a device, either electronic or photonic, that measures in the range 0..1 and that we treat any value above 0.6 as a boolean TRUE and any value below 0.4 as a boolean FALSE and say that in between the results are undefined. Similar coding schemes in terms of voltage are used by all logic families. For simplicity we will assume our measurements are in the range 0 volt to 1 volt.

Now suppose that our switching device is designed to be fed with a mean of 100 quanta when we input a TRUE to it. Following a poisson distribution we have $\sigma = 10$, so we need to know the probability that the reading will be indeterminate, below 0.6 volt, or how likely is it that only 60 quanta will arrive given shot noise; i.e. a deviation of 4σ from the mean. Using tabulations of the normal distribution we find that this probability is 0.0000317.

Consider a computer with a million gates each using 100 electrons. Then 31 of the gates would yield indeterminate results each clock cycle. This is unacceptable.

Assuming the million gates, and a 1 Ghz clock and that we will tolerate only one indeterminate calculation a day we want to push this down to a failure probability per gate per cycle of about 10^{-19} or 9σ . This implies $\sigma = 0.044$ volts which can be achieved when our capacitor is sufficiently large that about 500 electrons will generate a swing of 1.0 volt. The figures are merely illustrative,

and posed at a level of abstraction that would apply to both electronic and optical computers, but they illustrate the way reliability and number of quanta used to represent TRUE and FALSE trade off against one another.

Our digital computers are reliable because they use a large number of degrees of freedom, for example a large number of quanta of charge, to represent one bit. The number can be chosen so that the probability of a read error is very low, and then following a read, the signal can be regenerated. This prevents the propagation of errors that are the bane of analogue computing.

5.3 What memories are made of

Turing's initial idea of a universal computer was, as described in the last chapter, one that used a tape memory. During the Second World War he had experience with the code breaking computer's code named Colossus[41, 33] which was built to break into the German 'Fish' code. These machines used very high speed paper tape readers as a read only store. Most were destroyed after the war, but a few were retained for the generation of one time pads on punched tape for use in British diplomatic ciphers. But the use of tape was not an essential part of Turing's conception of a digital computer. His Automatic Computing Engine, designed in 1945 replaced tapes with mercury delay lines[79]. The essential point was that the memory could be used to hold either instructions or data. This had been implicit in his original paper[77]. Based on his experience since then he was by 1950 giving a very general definition of a digital computer:

The idea behind digital computers may be explained by saying that these machines are intended to carry out any operations which could be done by a human computer. The human computer is supposed to be following fixed rules; he has no authority to deviate from them in any detail. We may suppose that these rules are supplied in a book, which is altered whenever he is put on to a new job. A digital computer can usually be regarded as consisting of three parts:

- (i) Store.
- (ii) Executive unit.
- (iii) Control.

The store is a store of information, and corresponds to the human computer's paper, whether this is the paper on which he does his calculations or that on which his book of rules is printed. (*Turing*[78], page 436)

His mature conception of the memory is that it addressed randomly rather than sequentially

The information in the store is usually broken up into packets of moderately small size. In one machine, for instance, a packet might

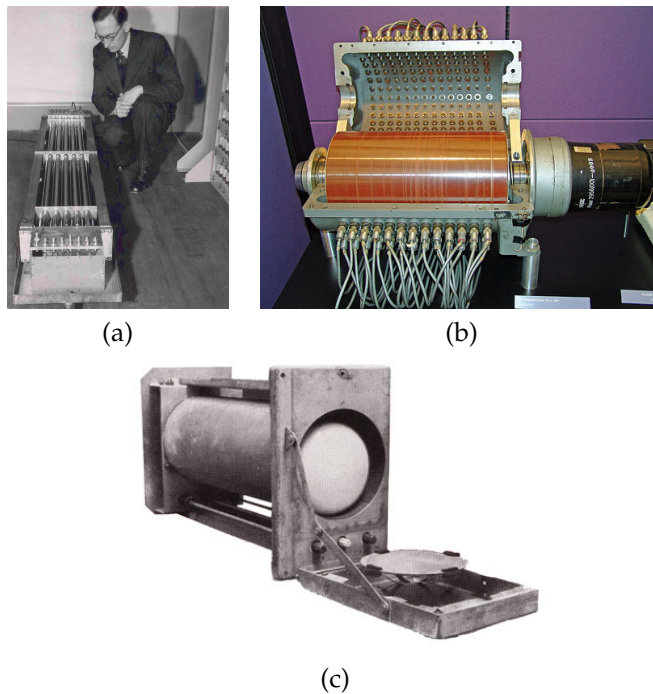


Figure 5.1: Memories available to Turing in the 1940s. (a) Mercury delay lines, shown with Maurice Wilkes designer of EDSAC. (b) Magnetic drum. (c) Williams Tube. Pictures (a)&(c) Wikimedia, (b) photo by Nial Kennedy.

consist of ten decimal digits. Numbers are assigned to the parts of the store in which the various packets of information are stored, in some systematic manner. A typical instruction might say:

'Add the number stored in position 6809 to that in 4302 and put the result back into the latter storage position.'

Needless to say it would not occur in the machine expressed in English. It would more likely be coded in a form such as 6809430217. (*Turing*[78], page 437)

At the time that Turing was designing ACE or working at Manchester, designing a suitable random access store was still a considerable problem. Although the conceptual design was for a store that could be accessed as a series of numbered slots, the actual implementations available were all cyclical.

In the Williams Tube[82, 53] a cathode ray tube was raster scanned to read out bits deposited as electric charge on the surface, an inherently cyclical/serial process.

In the drum store, the data was written to a set of tracks on the drum and read by fixed heads at slightly offset positions. Data from any head could be

randomly selected but one had to wait for a particular word to come back round as the drum rotates.

In a delay line the sound waves passed down a mercury filled tube and were picked up at the other end to be regenerated[55]. In both cases the limitation on the computers performance was the cycle time at which data came back round. On average one would expect memory access for a random word to take approximately $\frac{1}{2}$ the memory cycle time.

The first genuinely random access memory did not come until the invention of magnetic core planes such as the one shown in Figure 5.2. With these the access time could be much faster than in delay lines or drums. Time to access a core plane of n bits could be modeled as:

$$t_m = a \log(n) + b\sqrt{n} + c$$

where the logarithmic term $a \log(n)$ derives from the decode logic that selects row and column lines, the $b\sqrt{n}$ term derives from the time it takes for the signal to propagate along the row and column lines, and the constant term c is the time taken for a single core itself to respond to being selected. Why does it take this form?

Well in the photo the plane has 32 rows and 32 columns. When an access is performed one of the rows is selected. To select one row thus takes 5 bits since $32 = 2^5$. So the signal that selects the row must have had to pass through 5 AND gates. Activation of a row will cause all 32 bits on that row to be read out along the column lines. The bit that is wanted must be selected from these 32 using another 5 address lines. So the total number of AND gates involved in the critical path will be 10. There will be 5 to select the row and 5 for the column. There are 1024 bits in the memory, and $\log_2(1024) = 10$.

Next consider the time to propagate along the rows and down the columns. For a randomly chosen memory cell, the row signal has to propagate half way across and the column signal has to propagate half way down the column. So the mean propagation distance in our example will be $\frac{32}{2} + \frac{32}{2} = 32$ cells. So in Figure 5.2 what we have is

$$t_m = 10a + 32b + c$$

with a being the delay through an AND gate, and b the propagation time across a distance of 1 cell, and c the time for the selected core to switch.

Clearly, for large arrays, the square root term will dominate. In consequence there will be an optimal size of plane. Suppose that with the technology of 1961 the optimal size was 1024 bits. Rather than creating a plane of 4096 bits whose propagation time would have been twice as great, it would have been better to fit the machine with 4 planes of 1024 bits at a cost of only two more AND gate delays. The optimal size depended on the relative sizes of the constants a and b both of which would change over time as AND gates improved, and as physically smaller cores meant that the signal propagation distances fell.

During the 1970s the preferred technology for computer memories changed from induction to capacitance. Capacitive memories became commercially vi-

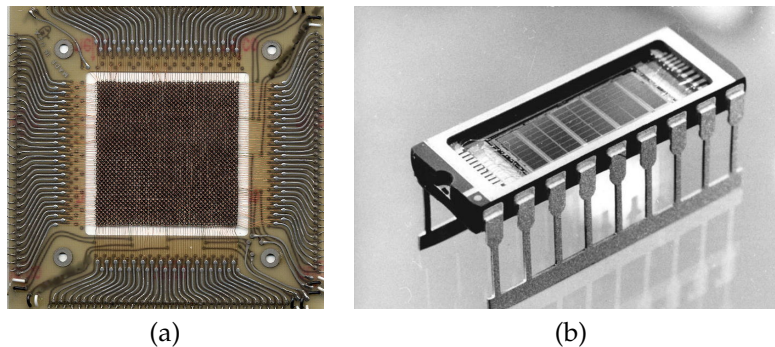


Figure 5.2: (a) Magnetic core memory storing 1024 bits of information, dating from 1961, made by Control Data Corporation. (b) First German 1 megabit Dynamic Random Access Memory chip, 1989, made by VEB Carl Zeiss Jena.

able with the Mostek 4096 bit dynamic RAM chip in 1973. These quickly displaced core memory because they were faster, cheaper and more compact. If you look at the DRAM shown in Figure 5.2(b) you can see that it was made of 64 smaller arrays, illustrating our previous argument that there is a sweet point for a given two dimensional memory array technology. An array size that minimises both decode and propagation delays. For larger memories one creates multiple of these optimally sized arrays. It is not accident that both the core memory and the DRAM were organised as two dimensional arrays. A combination of logic, economy, and thermodynamics have forced this approach onto computer designers.

It is a logical approach because the organisation of memory as a grid allows the decode logic to be shared across many memory cells. In the old 1024bit core memory, each row or column wire would have been fed by an output of a five level decode tree, a deeper version of Figure 5.3. A p level decode tree requires $2^p - 1$ demultiplexers, so our core matrix will have required 31 demux's for the rows and 31 for the columns, in general this approximates to $2\sqrt{n}$ components for an array of n bits. If each memory bit was fully decoded then n demultiplexers would be used, so the grid arrangement uses much less resource in gates.

There is another reason why two dimensional arrays are advantageous. We should remember that computers are labour saving devices. They are only worth using if the total work set free by using the machine exceeds the total work that goes into making it. If we look at the calculator shown in Figure 3.15, its cost in the 1950s was around two month's average earnings. It would only be worth while for somebody to buy one if they were an accountant or had some other job that involved them in a great deal of calculation over a long period.

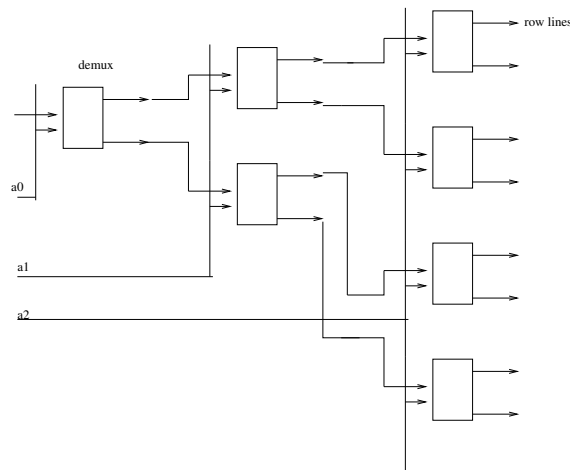


Figure 5.3: Three level decode tree.

By 1972 when the first mass produced electronic calculator, the Sinclair executive, was launched in Britain, its price was the equivalent of two weeks wages. Today a basic four function electronic calculator costs the equivalent of about 20 minutes average wages. At today's price, the time saved on even occasional calculations, will make buying a calculator worth while for most families. The fall in price which is most obvious with calculators, but has affected all computing equipment, is a consequence of the adoption of what is basically a printing technique to produce electronic logic. The chip layout is transferred to the chip by a parallel photographic process. This allows all transistors on the chip to be simultaneously manufactured. Successive generations of chips have small and smaller features on them. Improvements in imaging technology - the ability to use and focus ever shorter waves of electromagnetic radiation - have allowed repeated halvings of feature sizes. These halvings of feature sizes translates into an exponential growth in the number of transistors per hour that a state of the art plant can make. Despite the growth in capital costs of chip plants[63] this has still led to an exponential cheapening of transistors.

But for this to work it has been essential that there exists an axis normal to the circuit along which its constitutive information can be transmitted in parallel. In a three dimensional universe, this constrains us to make two dimensional printed products.

Prior to the adoption of printing techniques, some three dimensional core arrays were built. But core memories were manufactured sequentially, threading wires through the cores one at a time. Indeed, although machines were built to automate construction, it was in the end found cheaper to use oriental cheap labour to make them by hand[66]. For cores there was not the same

imperative to adopt a purely planar organisation.

Suppose that we did try to mass produce chips with a 3D structure. Since the manufacturing process is basically a printing process, at least one printing step would be required for each layer of our 3D structure. If we want to improve our gate count on a 2D chip by a factor of 4 we can do so provided we can print each feature at half the size. Using the third dimension, we could also increase the number of gates by a factor of 4 by having 4 times as many manufacturing steps to lay down 4 times as many layers. If we grow gates by shrinking feature sizes the number of manufacturing steps remains constant. If we do it by adding layers the cost rises at least in proportion to the additional layers. In fact the cost metric will be worse than this for a multi-layer chip since manufacturing errors will grow exponentially with the number of layers used. For these reasons 3D chip manufacture has never been economically competitive. *do*

Another factor is the problem of heat dissipation. A three dimensional structure can only conduct or radiate away heat from its surface. If heat is to be removed from the interior it has to be made porous and liquid or gas blown through to cool it. This was feasible with cores since they were porous, but it is much more of a problem with silicon chips. These have to dissipate their heat from their surfaces, and as circuit technology has advanced, the problem of heat dissipation has become more severe.

5.4 Power consumption as a limit

The first generations of electronic computers used vacuum tubes or 'valves' as they were called to perform binary switching. In a valve the cathode has to be heated by a red hot element, in consequence a valve computer used huge amounts of power. Lavington recounts how amazed people were by the amount of power used by the Manchester Mk1 in 1948[53]. Power consumption was substantially reduced in the second and third generations of mainframe computers which used discrete transistors or small scale integrated circuits[1, 66]. The key to the inexpensive mass production of electronic computers was the development of the microprocessor, initially as a device for cheapening the production of desktop calculators[12].

5.4.1 CMOS

The first generation of microprocessors like the Intel 4004 shown in Figure 5.4(a), used what are called PMOS transistors. These were followed shortly after by NMOS chips like the 8080 and then from the mid 1980s CMOS chips have dominated. The aim of successive changes in circuit technology has been to combine circuit speed with power economy.

To understand what is involved look at the NMOS gate shown in Figure 5.5. The diagram shows a cross section through a gate with air above and silicon below. The silicon is doped in two ways N-type doping which imparts

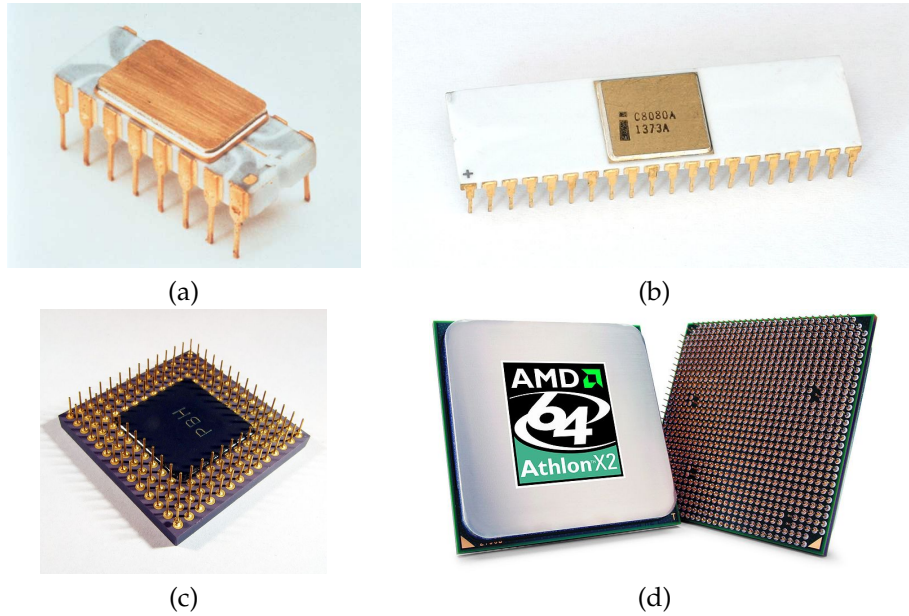


Figure 5.4: Intel and AMD microprocessors. (a) 4004 PMOS 4 bit chip, 1971. This was the first microprocessor and was designed to be used in Busicom desk calculators. (b) 8080 NMOS 8 bit chip, 1974. This was used in the first generation personal computers like the Altair 8800 or IMSAI 8080. (c) Intel 80386 CMOS 32 bit chip. (d) AMD Athlon-64 dual core CMOS 64 bit chip, 2005.

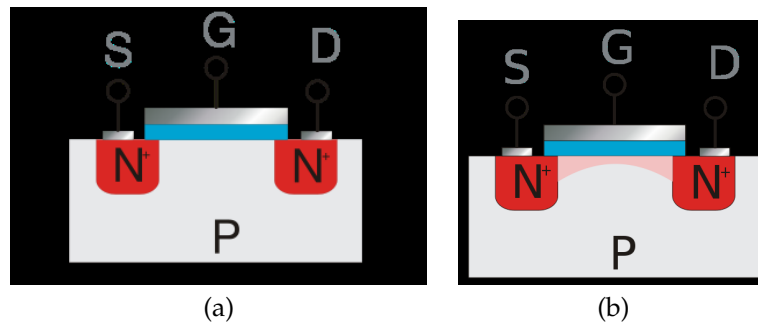


Figure 5.5: NMOS gate (a) in the off state, (b) in the on state.

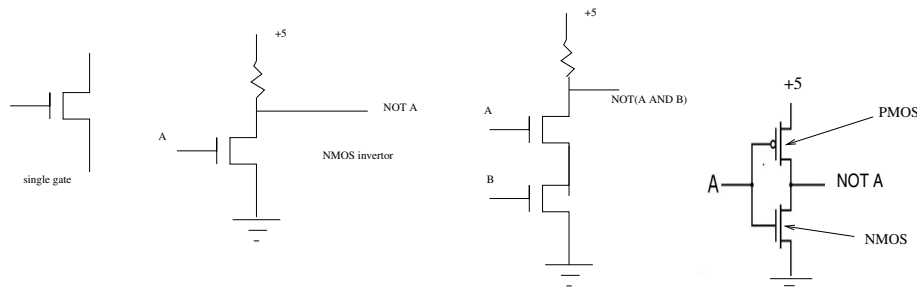


Figure 5.6: Left to right: Basic NMOS transistor symbol, NMOS NOT gate, NMOS NAND gate, CMOS NOT gate. Note the pull-up resistors on the NMOS NOT gate and. When the output of the NMOS NOT gate is low, a current will flow through this resistor to ground through the transistor. In the CMOS case, the output is connected either to high or to ground with no leakage current.

a negative charge, and P-type doping which imparts a positive charge. The two N-type regions are labeled S and D for Source and Drain. In the middle, labeled G, is the gate, a capacitor plate which is isolated from the silicon by a layer of insulating silicon dioxide. The gate acts as a switch that can be opened or closed by applying a voltage to it.

In the off situation shown in Figure 5.5(a), current can not flow from the source to the drain as it is opposed by the bias voltage between the N-type silicon of the source and the P-type silicon under the gate. If a positive charge is applied to the plate G, however, it induces a N-type depletion zone immediately under the gate. Consequently there is a continuous N-type region from the source to the drain and current flows. In a PMOS gate the source and drain are built of P-type silicon and in the off mode the channel is N. In this case a negative charge on the gate induces a P-type depletion zone between the source and drain.

CMOS chips combine the use of both NMOS and PMOS gates, the motivation being to reduce power consumption. If we look at the NMOS gates in Figure 5.5 we see that when the output is low a leakage current will flow through the pull-up resistor to ground. The CMOS gate avoids this leakage current by connection the output either to high or to ground. Thus there is no static current flow for a CMOS gate. The only current flow occurs during switching in order to overcome the capacitance of the circuit attached to the output of the gate. This led to a considerable economy in the use of power when the initial transfer from NMOS to CMOS technology occurred.

As clock speeds rose however the currents required to overcome capacitance rose so much that power consumption has again become a pressing problem. Let us construct an ultra-simple model of the power consumption of a CMOS processor. We can approximate a CMOS chip by a collection of capacitors a portion of which charge and discharge on each clock cycle. This

charging and discharging dissipates power. For our simple model we will consider only the capacitance of the gates themselves not of the wires between them. The capacitance of a parallel plate capacitor is given by

$$C = \kappa 8.854 \times 10^{-12} \frac{A}{d}$$

where C is capacitance in farads, A area in square meters, κ is the dielectric constant of the material between the plates and d their separation in meters. For the silicon dioxide insulator used in most chips κ is 3.9. Let us initially consider only the capacitance of the CMOS gates themselves, ignoring the wires for now. Since the capacitance of the gates is inversely proportional to gate insulation thickness, as feature sizes are made smaller, the capacitance tends to rise as the thickness of gate insulation falls. Since the insulation under gates is thinner than under wires, the gates contribute a disproportionate amount to the total capacitance.

Consider a gate that is 65nm square and has an oxide thickness of 1.2nm (realistic figures for about 2008) :

gate dimensions	65×10^{-9} meters
gate area	4.2×10^{-15} sq meters
distance between plates	1.2×10^{-9} meters
Capacitance	1.2×10^{-16} farad

The charge on the gate will be the product of the working voltage and the capacitance $=vC$, and the power used will be $P = fv^2C$ where f is the clock frequency. If we operated the gate at 3.3v and had a clock speed of 1 GHz then each gate would use a power of about 1.3×10^{-6} watts. This does not seem much until you take into account how many gates there are in a modern chip. An AMD K10 for example has of the order of 5×10^8 gates. The power used will depend on the fraction of the gates which switch each cycle, if 10% switch each cycle the total power consumption of the chip would be about 65 watts. Clearly the model above is very approximate. The fraction of gates switching each cycle are a guess, and we have made the simplifying assumption that gates are squares the of edge the minimum feature size. We have ignored capacitance on wires since the thickness of insulation here is much larger, and also neglected resistive losses.

Despite the simplifications, our estimated power consumption agrees reasonably well with the figures given by AMD for the Agena version of the K10 which consumes between 65 and 140 watts for clock speeds between 1.8GHz to 2.6GHz.

As clock speeds rise so does power consumption. As feature sizes shrink, and thus the insulator thickness falls, this too increases power consumption per sq cm of chip surface. The result is shown in Figure 5.7. The processor becomes physically dominated by the equipment required to remove excess heat. The rate of heat flow away from the surface of the chip is already more than can be handled by simple conduction. To augment this heat pipes[25] have to be used, Figure 5.7(c). We are approaching the limit of heat dissipation that can be practically handled by low cost techniques. Mainframe computers

in the past used liquid cooling[16], but this is expensive and not suitable for mass produced machines.

Our account of computing has led from primitive arithmetical aids to devices more like little steam engine boilers. This seems strange.

But there turns out to be a deep relationship between computing and thermodynamics, and it is thermodynamics that, on the basis of current understanding, fundamentally limits the power of our computing.

5.5 Entropy Lewis

Revision of this with respect to classical thermodynamic

5.6 Shannon's Information theory

The establishment of information theory as a science occurred in the middle of the last century and is closely associated with the name of Claude Shannon. If anyone was father to the information revolution it was him. Shannon's revolution came from asking new questions, and asking them in a very practical engineering context. Shannon was a telephone engineer working for Bell Laboratories and he was concerned with determining the capacity of a telephone or telegraph line to transmit information. He [70] formalized the concept of information through trying to measure the efficiency of communications equipment.

To measure the transmission of information over a telephone line, some definite unit of measurement is needed, otherwise the capacity of lines of different quality cannot be meaningfully compared. We need to quantify information. According to Shannon the information content of a message is a function of how surprised we are by it. The less probable a message the more information it contains.

Suppose that each morning the news told us

There has been no major earthquake in the last 24 hours.

We would soon get fed up. It conveys almost no information. If instead we hear:

We have just heard that Tokyo has been devastated by a force 8 earthquake.

This is real news. It is surprising. It is unusual. It is information.

A daily bulletin telling us whether or not a quake had happened would usually tell us nothing, then one day would give us some useful information. Leaving aside the details, if an announcement were to be made each morning, there would two possible messages

0 'No big earthquake'

1 'There has been a big quake'

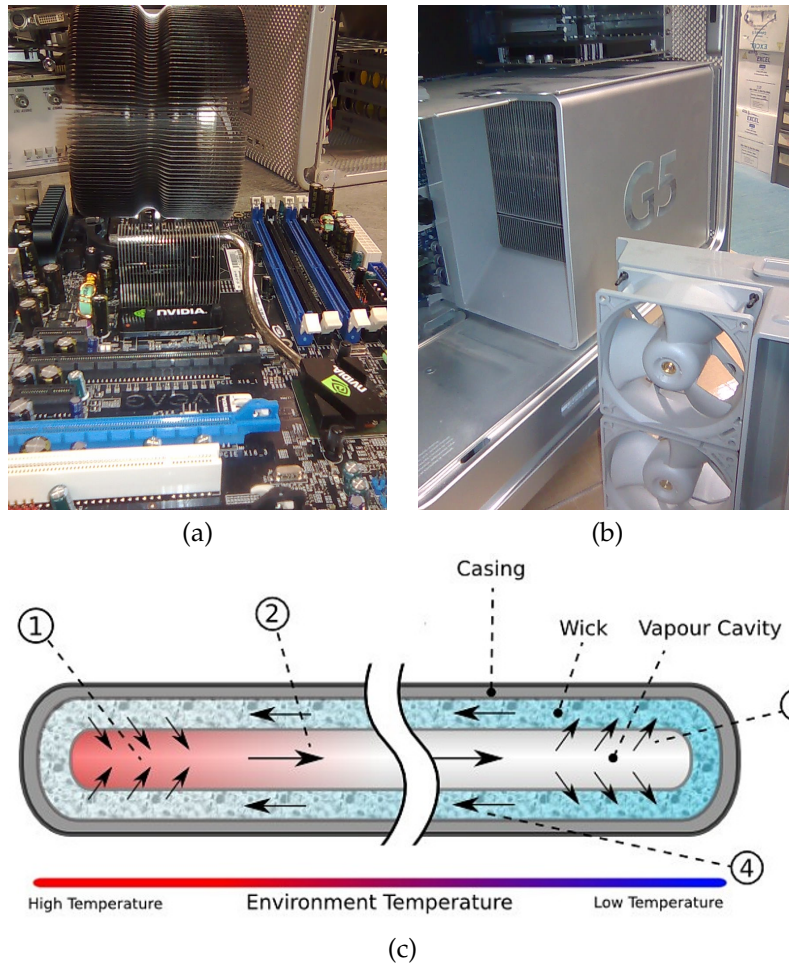


Figure 5.7: Heat exchangers used in modern desktop computers. (a) The heat exchanger used for a 4 core Intel processor. The large finned heat exchanger is cooled by an axial fan and heat is led up to it from the processor and graphics chips by heat pipes. (b) The heat exchanger system used for the G5 processor on an Apple computer. (c) The principle of the heat pipe. 1. Vapour evaporates absorbing heat. 2. Vapour flows to cold end. 3. Vapour condenses releasing heat. 4. Wick conducts liquid back.

Binary Code	Length	Meaning	Probability
0	1	False, False	$\frac{4}{9}$
10	2	False, True	$\frac{2}{9}$
110	3	True, False	$\frac{2}{9}$
111	3	True, True	$\frac{1}{9}$

Table 5.1: A possible code for transmitting messages that are true $\frac{1}{3}$ of the time

If such messages were being sent by wire, one could encode them as the presence or absence of a short electrical pulse, as a binary digit or 'bit' in the widely understood sense of the word. We normally assume that a bit is the information required to discriminate between *two possible alternatives*.

Shannon defines a bit more subtly as the amount of information required for the receiver of the message to decide between *two equally probable alternatives*.

For example, a sequence of tosses of a fair do contain one bit per toss, and can be efficiently encoded so that heads are 1 and tails 0.

Shannon's theorem says is that if we are sending a stream of 0 or 1 messages affirming or denying some proposition, then unless the truth and falsity of the proposition are equally likely these 0s and 1s contain less than one bit of information each. In which case there will be a more economical way of sending the messages. The trick is to use a system where the more probable message-contents gets a shorter codes.

For example, suppose the messages are the answer to a question which we know a priori will be true one time in every three messages. Since the two possibilities are not equally likely Shannon says there will be a more efficient way of encoding the stream of messages than simply sending a 0 if the answer is false and a 1 if the answer is true. Consider the code shown in Table 5.1. Instead of sending each message individually we package the messages into pairs, and use between one and three binary digits to encode the 4 possible pairs of messages.

The shortest code goes to the most probable message, the sequence *False False* with probability $\frac{2}{3} \times \frac{2}{3} = \frac{4}{9}$. The codes are such that they can be uniquely decoded at the receiving end.

For instance, suppose the sequence '110100' is received: checking the Table, we can see that this can only be parsed as 110, 10, 0, or True, False, False, True, False, False.

To find the mean number of digits required to encode two messages we multiply the length of the codes for the message-pairs by their respective probabilities:

$$\frac{4}{9} + 2 \times \frac{2}{9} + 3 \times \frac{2}{9} + 3 \times \frac{1}{9} = 1\frac{8}{9} \approx 1.889 \quad (5.1)$$

which is less than two bits.

Shannon came up with a formula which gives the shortest possible encoding for a stream of distinct messages, given the probabilities of their individual occurrences.

$$H = - \sum_{i=1}^n p_i \log_2 p_i \quad (5.2)$$

The mean information content in a collection of messages comes by multiplying the log of the probability of each message by the probability of that message. He showed that no encoding of messages in 1s and 0s could be shorter than this.

The formula gave him an irreducible minimum of the number of bits needed to transmit a message stream: the real information content of the stream.

In his 1948 article Shannon notes:

Quantities of the form $H = - \sum_{i=1}^n p_i \log p_i$ play a central role in information theory as measures of information, choice and uncertainty. The form of H will be recognized as that of entropy as defined in certain formulations of statistical mechanics where p_i is the probability of a system being in cell i of its phase space. H is then, for example the H in Boltzmann's famous H theorem. We shall call $H = - \sum p_i \log p_i$ the entropy of the set of probabilities p_1, \dots, p_n .

So here we get a critical result: information and entropy are the same.

5.7 Landauer's limit

Information is not a disembodied abstract entity; it is always tied to a physical representation. It is represented by engraving on a stone tablet, a spin, a charge, a hole in a punched card, a mark on paper, or some other equivalent. This ties the handling of information to all the possibilities and restrictions of our real physical world, its laws of physics and its storehouse of available parts.[49]

We discussed earlier how difficult it was to get rid of the heat generated by current CMOS circuits. That argument was based on currently existing techniques for building logic gates. Human ingenuity being what it is, we should expect that novel designs of gates will arise in the future that allow components to be still smaller, and to use less energy.

The decisive factor in cooling is the number of watts per sq cm of heat released. Provided that the shrinkage in area goes as fast as or faster than the shrinkage in power consumption we will be OK. The most obvious step is to reduce the voltage at which the chip operates - and this has indeed been done over time. But there is an interaction between gate voltage and reliability. As you shrink gate voltages, the reliability of the gate in the face of thermal noise declines.

5.7.1 Thermal noise

The thermal fluctuations of voltage on a capacitor in a CMOS chip give rise to a thermal noise voltage [44] which has the form

$$U_n = \sqrt{\frac{kT}{C}} \quad (5.3)$$

As the capacitance of the gate falls the thermal noise rises. Note the similarity of this to the distribution discussed on page 61. The discrete nature of charge lies at the base of the thermal noise as it does with shot noise. If we assume that gate oxide layers have hit an effective lower limit at about 1 nm, then capacitance is going to scale proportionately to λ^2 where λ is the minimum feature size on a chip. If we reduce the λ from 40nm to 20nm the area of a gate capacitor will fall from 160 nm² to 40 nm². Thus by equation 5.3, if we halve the feature size on a chip, we double the thermal noise voltage. To provide an adequate protection against the occurrence of errors it is necessary for the operating voltage level of the chip to be scaled with the thermal noise such that the voltage difference between a 1 and 0 is about 11 or 12 times U_n . From this it follows that if we keep shrinking our transistors, we will have to start increasing the operating voltage of the chips to provide a sufficient noise threshold.

This obviously has serious implications for power consumption per unit area since power consumption is proportional to the square of the voltage. It means that continual shrinkage of CMOS gates will hit a noise limited floor to power consumption per sq cm that will scale inversely with the square of the feature size. A halving of feature size will lead to a quadrupling of power consumption at a given clock speed. In effect this sets a floor to the amount of energy that a CMOS gate can use and still perform reliably².

But this seems tied to the particular properties of CMOS and raises a more general question : is there an ultimate lower limit to the amount of power that a computer must use?

This was asked by Landauer[50, 51, 52] and the answer he gave was yes. His argument was based on an ingenious combination of the theorems of Boltzmann and Shannon.

He first observed that two of the three basic gates from which we build our digital logic (AND, OR, NOT) are information destroying (Figure 5.8). Whenever data goes into an AND gate, two bits go in but only one bit comes out. Since information and entropy are equivalent this amounts to an entropy reduction operation. But entropy reduction is prohibited in any closed system by the second law of thermodynamics.

What happens to the entropy of the missing bit?

It can not just vanish, it has to be turned into heat. If the entropy of the logic goes down, the entropy of the environment must rise to compensate. Using

²See also Equation 6.16.

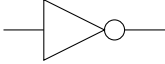
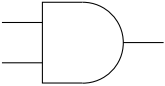

		INPUTS	OUTPUTS	BITS LOST
NOT GATE		1	1	0
AND GATE		2	1	1
OR GATE		2	1	1

Figure 5.8: The basic gates used in modern computers. All but the NOT gate destroy information.

Boltzmann's constant again he derives the minimum energy that is wasted by discarding a bit as $kT \ln(2)$ Joules³.

We can fit this into our previous calculation of the power used by K10 processor and find out what would be the minimum power that it could use. We make the same assumption about the number of active gates as before, and assume that it is working at room temperature.

k = Boltzmann's const	1.38×10^{-23}
T = Temperature	300 degrees Kelvin
$\ln(s)$	0.693
$e = kT \ln(2)$ = energy per gate	2.87×10^{-21} Joules
n = number of active gates	50,000,000
f = frequency	1 GHz
$p = fne$ = power	1.43×10^{-4} Watts

This is a very small amount of power. The actual K10 uses about a million times as much power (page 70). So there is plenty of room for improvement before we hit the ultimate limit of thermal efficiency for our computers!

So there is a limit. Why worry? Be happy! It is far away!

There are several reasons to worry.

One has to consider first that Landauer's limit is very much a lower bound for a gate. A gate dissipating energy at this rate so close to the thermal noise level would be too unreliable for practical use. But if we ignore that for now, and pretend that you really could have a computer using only the Landauer energy, the rate of improvement in computer technology has been such that we would soon reach even that limit. The first computers used triode valves which used about 5 watts for each gate. The Manchester Mk I on which Turing worked as a programmer was the first programmable general purpose electronic computer. It had about 4000 valves and used 25 Kilowatt[53, 54, 55]. It

³The $\ln(2)$ term arises because Boltzmann's constant is defined in terms of natural logs and information is defined in terms of \log_2 .

operated at a clock speed of 85KHz so that to perform one logical operation with a triode was using about 6×10^{-5} Joules, the calculations on page 70 indicate that the gates on the K10 use about 2.6×10^{-15} Joules per logic operation. So over a 60 year period gate efficiency has improved by a factor of about 10^{10} .

Efficiency has been improving by tenfold every six years. If this trend continues, then around 2045 gate efficiency will hit its ultimate thermodynamic limit.

But that looks only at the efficiency question. We have to consider the growth in processor power too. This can be bounded by the product of the clock speed and the number of logic operations performed each clock cycle. In practice, due increasing complexity of logic to achieve a given instruction mix, this upper bound is not met, but let us use the optimistic measure for now. Since between the 1949 Mk I and the 2007 K10 the gate count of a machine has risen from 4×10^3 to 5×10^8 , a factor of 10^5 and the clock speed rose by a factor of about 10^4 . The number of logic operations per second have been growing by a factor of 10 every 6.5 years. If this growth continued, how long would it be before even a machine working at maximal Landauer efficiency would be generating too much heat for us to keep it cool?

The K10 generates about 200 Watts per sq cm. It can be cooled, but the limit of what can be done by conductive plus air cooling is probably of the order of 1Kw per sq cm. Suppose we envisage a future computer chip 1 cm by 1cm operating at maximum Landauer efficiency and putting out 1Kw. Let us call this the Last Chip. How many logic operations per second could it do?

Clearly the answer is $\frac{10^3 \text{ Joules}}{2.87 \times 10^{-21} \text{ Joules}} = 3.48 \times 10^{23}$. Since we have estimated that our K10 is doing around 5×10^{16} logic operations per second, this allows our Last Chip to be about 10 million times more powerful than anything we have today. At current growth in processing power this level of performance could be attained in about 40 years - say around 2050.

These very crude calculations imply that, if historical trends continued, we would reach 100% Landauer efficiency about 2045 and that about 5 years later, even chips running at this efficiency level will become thermally limited. So although the Landauer limit is some years away, it can be expected within the working careers of students reading this book.

5.8 Non entropic computation Paul& Lewis

The discovery of the Landauer limit prompted research into whether it was in principle possible to build computers that would use less energy. One approach was that put forward by Fredkin and Tofoli[31] who invented a new class of logic gates which they called conservative logic. They made the ambitious claim that:

The central result of conservative logic is that it is ideally possible to build sequential circuits with zero internal power dissipation. ([31], p.3)

The key idea was to get round what they saw as the limit in conventional logic technology that it was not reversible. We have seen how the AND gate takes in two bits and outputs only one. If one could devise a new family of logic gates which met the conditions that:

- they had the same number of outputs as inputs;
- their logic functions were reversible, i.e., from the output of the gate you can determine what the inputs must have been;

then there would be no loss of information as the gates operated. If there was no loss of internal information or internal entropy, then there would be no need to shunt entropy into the external environment as heat. Hence given a suitable reversible gate family, one could in principle build zero power computers. The set out to answer four questions:

Question 1. Are there reversible systems capable of general-purpose computation?

Question 2. Are there any specific physical effects (rather than mere mathematical constructs) on which reversible computation can in principle be based?

Question 3. In Section 4, we have achieved reversibility of computation at the cost of keeping garbage signals within the system's mechanical modes. In a complex computation, won't garbage become unmanageable if we cannot dissipate it? And won't the need to dissipate garbage write off any energy savings that one may have achieved by organizing the computation in a reversible way?

Question 4. Finally, without damping and signal regeneration, won't the slightest residual noise either in the initial conditions or in the running environment be amplified by an enormous factor during a computation, and render the results meaningless?

([31], p.13)

The came up with adequate responses to the first three questions but, in our opinion, failed on the last one. Whether or not one considers their proposal to be plausible, it certainly remains instructive.

A number of such gates would be possible, but they gave as an example the so called Fredkin gate in Figure 5.9. They showed that if one fed in 0 or 1 into appropriate inputs one could emulate the classical AND and NOT gates with Fredkin gates. The fact that no bits were being thrown away did mean that you would have to keep hold of a series of garbage bits generated in the calculation. This might seem to offer no advantage over old style logic gates, since one has merely postponed discarding entropy by encoding it in the garbage bits. What happens to these at the end of the calculation?

If you just put them into a bit sink, you generate heat and have gained nothing by using Fredkin gates. Their ingenious solution was to do the calculation in 3 stages:

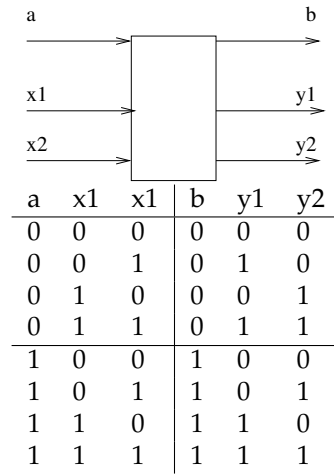


Figure 5.9: The the Fredkin gate with its input output table.

1. Put the input variables V and the required constants C into the gates and compute the boolean result R plus the garbage G .
2. Perform a copy of the result into a final register F . This operation is destructive and liberates $l_F kT$ of energy where l_F is the length in bits of the result register.
3. Send the information in $\langle R, G \rangle$ back through the logic in the reverse direction, restoring the original V and C .

Since the same C is used every time you do the calculation, the only cost of the whole operation is pre-setting V and setting F so the bit cost in terms of entropy is just the sum of the input variables plus the output variables. The energy cost no longer depends upon the number of logic steps performed, only on the size of the question and the size of the answer.

They have now answered their first and third question, what about physical realisability?

For this they proposed billiard ball logic as shown in Figure 5.10. They argue that spheres of radius $\frac{1}{\sqrt{2}}$ moving with unit velocity along a grid and then undergoing elastic collisions can be thought of as performing logic operations. By inserting reflectors at appropriate positions it is possible to get the switch gate shown in Figure 5.10(b). By appropriately routing 4 of these switch gates in sequence, they showed that one can logically realise the Fredkin gate.

Fredkin and Tofoli argued that they have good precedents for the model they proposed:

In the past century, a satisfactory explanation for the macroscopic behavior of perfect gases was arrived at by applying statistical mechanical arguments to kinematic models of a gas. The simplest

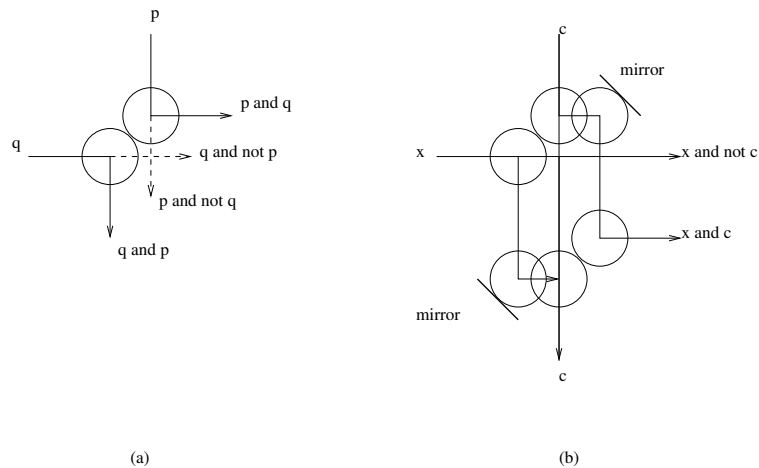


Figure 5.10: Billiard ball interaction gates. (a) Elementary collision. (b) The switch gate. The descending bit C switches the bit X between two possible paths and then continues on its way. Note that the logical output bit C may not be the original ball.

such model represents molecules as spheres of finite diameter. Both molecules and container walls are perfectly hard and elastic, and collisions are instantaneous. ([31], p.18)

This is true historically. Boltzmann in his Lectures on Gas Theory[7], did indeed assume molecules were spheres of the sort used by Fredkin and Tofoli, but he also paid considerable attention to the scattering properties of collisions between molecules ([7] Chapter 1.4). If we look at Fredkin and Tofoli's proposals from this standpoint we have to conclude that their gate model is unviable even within the theoretical framework of rigid perfectly elastic spheres. The problem lies with the exponential instability of collision paths.

Consider Figure 5.11. In this diagram the two balls should meet in the middle on vertical ingoing and outgoing paths shown as dashed lines. In the Fredkin model the balls should approach at 90 degrees rather than 180 degrees, but by choosing our frame of reference appropriately we can ignore all but the vertical component of the intended approach. The ingoing path of ball B is shown as being perturbed to the right resulting in an off center collision. The consequence of which is for the outgoing path to be perturbed by a greater amount than the ingoing one. The perturbation shown is quite large, but even for a small initial perturbation a similar growth in perturbation occurs with each collision. A system of colliding spheres is chaotic, the behaviour of the whole system quickly becomes randomised.

Perturbations are unavoidable. Collisions with the reflectors will give a thermal component to the energies of the balls equivalent to kT which will

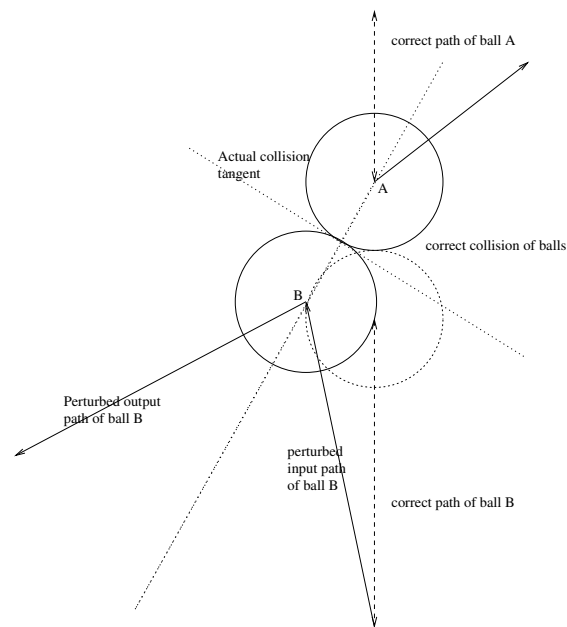


Figure 5.11: In the event of a perturbed collision between two balls, the deviation of the outgoing path from the correct path is greater than the deviation of the incoming paths.

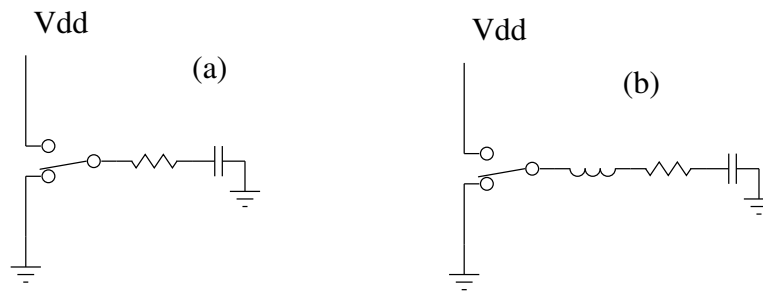


Figure 5.12: (a) A CMOS gate uses energy charging and discharging its capacitor through the resistance of the circuit that connects it alternately to power and ground. (b) If we place an inductance in the path we can reduce the potential across the resistance and thus reduce resistive losses at the cost of a slower operation.

become amplified with each collision. After a few layers of logic, the output of the gates will become completely non-deterministic and you will be left with what one should have expected from the kinetic theory of gases, ‘molecules’ bouncing about in a maximally entropic fashion.

For the billiard ball logic to work one has to postulate the complete isolation of the information encoding degrees of freedom from all thermal noise. For a mechanism operating classically this is impossible. Fredkin’s model did however prepare the way for Feynman’s proposal to use quantum mechanics to implement reversible gates[29, 30].

5.8.1 Adiabatic CMOS gates

We said when discussing power consumption in CMOS (page 69) that CMOS circuits used power because they were charging and discharging capacitances every cycle. Let us look at this slightly more closely. Figure 5.12 shows a conceptual model of a CMOS gate which is alternatively charged and discharged through the resistance of the path leading between the gate and either the power supply or ground. In practice the resistance to ground and to high would not be the same, but that is not important for what follows. Earlier, we estimated the energy loss used by the gate by considering the energy dissipated as the charge on the gate is allowed to fall in potential from Vdd (the supply voltage) to ground. What actually happens is that the potential energy on the capacitor is turned into heat overcoming the resistance of the resistance on the path to ground.

The power consumed by a resistor is the product of the voltage across the resistor and the current flowing. Initially, when the switch connecting the capacitor to ground is thrown, the voltage across the resistance is Vdd, but as the capacitor discharges, the voltage falls so the rate of power dissipation peaks

then falls off. Suppose that we could reduce the voltage across the resistor to less than the full power supply drop from the start. In principle this can be done by placing an inductance in the path. The initial voltage drop will now be split between the inductor and the resistor, and in consequence the resistor will heat up less leading to a more efficient overall operation. This mode of operation of CMOS gates is called adiabatic.

It is in principle possible to integrate inductors onto chips[11] but the cost in area of doing this is relatively high. Researchers in the area have instead adopted the approach of globally ramping the power supply up and down using external inductors in the power supply path to the chip. The Charge Recovery Logic family designed by Knight and Younis[46] works that way, and also requires a considerable growth in the number of transistors used for each logic gate in order to accommodate the fact that the operations are now done using AC rather than DC power. Younis[85] gave a design for a NAND gate which involved 32 transistors. A normal CMOS NAND requires 4 transistors. There is thus an eightfold increase in components required.

The other downside, is that the speed at which we can discharge or charge the gates goes down, so we have to operate the circuit at a slower rate. This would seem to defeat the purpose of the exercise since heat generation is only a problem in high speed circuits. But it turns out that if we slow the discharge down significantly then the power losses through the resistor become proportional to the square of the frequency at which we operate the circuit. Suppose that as a result of using using adiabatic operation we reduced the clock speed of a 100 watt processor chip from 1GHz to 100Mhz. Using conventional circuit the power would drop to 10 watts. But if we used an adiabatic circuit the power loss on each transistor would have fallen to only 1% of what it originally was. Allowing for having 8 times as many transistors we could expect the adiabatic circuit to be using 8 watts at 100MHz. But of course if we had simply reduced the clock speed of the original chip to 100MHz, power would also have fallen to 10 watts without the extra manufacturing cost of using 8 times as many gates.

But the number of transistors you can fit on a chip is a function of your manufacturing process. At any given time the maximum transistor budget is fixed. An adiabatic computer using 8 times as many transistors for each primitive logic gate would have less logic gates and thus would need to be of a simpler architectural design. It might have to be a 32 bit processor instead of a 64 bit one, it might have less internal parallelism, no dual instruction issue etc. This means that our 100MHz adiabatic chip would have less computing power than the conventional CMOS chip slowed down to 100MHz.

If we essay more drastic cuts in frequency, the adiabatic technology begins to show some advantages. At 25MHz the adiabatic chip is using 0.5 watts. To cut the conventional chip's power down this low we would have to run it at 5 Mhz. It is plausible that the adiabatic chip would now be doing more computing work for the same energy as the conventional one. So for some applications, for example where battery usage has to be kept to an absolute minimum, and where computing requirements are modest, adiabatic circuits

have arguable advantages.

It is harder to see them competing with mainstream processors. In principle, by throwing in enough slow adiabatic processors running in parallel one might be able to do the computing work that can be done with a current 3GHz processor chip and still use less electrical power. But the cost of buying 50 CPUs instead of one would be prohibitive, to say nothing of the impossibility of getting the 50 CPUs to run software that is currently designed to run on one processor. It is true that levels of parallelism are expected to rise anyway, but this is parallelism accompanied by an increase in performance. It is unlikely that people would be willing to spend a lot money and effort on parallelism at the same level of performance just to save modest amounts of electricity.

Chapter 6

Quantum computers

Lewis to do this , but should cover at least the issues of

1. computability limits of quantum computers, and key quantum algorithms
2. implementation techniques being investigated for them
3. the coherence problem
4. error correction

6.1 Quantum limits to real number representations

However, all other considerations aside, the idea of being able to represent real numbers physically to any desired degree of accuracy turns out to be in direct contradiction with quantum mechanics. To understand the significance of this statement, it is important to recognise that quantum theory and its extension, quantum field theory, obsolete and replace classical physics, and that this has been so comprehensively demonstrated empirically that it is beyond reasonable doubt. In its fundamental (micro-) nature, the world is not classical, and classical models can never be more than approximations applicable in a limited domain. Recent developments in quantum theory suggest that the classical appearance of the world is, in fact, an artefact of decoherence, whereby the state vector of a quantum system becomes entangled with the environment [62] and that consequently quantum mechanics can indeed be seen as a universal theory governing macroscopic as well as microscopic phenomena. In what follows, we use only the basic axioms of standard quantum theory [23], in particular those governing the deterministic time-evolution of an undisturbed quantum system according to Schrodinger's equation and the discontinuous change of state vector experienced when a system is measured. From these principles it is easy to demonstrate that there are fundamental limits on the accuracy with which a physical system can approximate real numbers. Suppose we wish to

build an analogue memory based on setting, and later retrieving, some property of a material body, A . To be suitable for analogue representation of real values to arbitrary accuracy, any such property must be continuously variable and so, according to quantum theory, can be represented as a Hermitian operator with a continuous spectrum of real eigenvalues. The most obvious approach to storing a real number in analogue fashion would be to use a positional coordinate of A , with some appropriate degree of precision, as the measurable property. Since only one coordinate is required, in what follows we will assume that A moves in only one dimension and that the value stored is given by its x coordinate, an eigenvalue of the x -position operator \mathbf{X} .

It is clear that any such system will have limits imposed by quantum mechanics: the aim here is to establish just how those limits would constrain any conceivable technology. As a first approximation, assume that A is a free body and is not therefore placed in a potential field. A natural scheme might store a real value, say, x_V , at time t_0 , by placing A at a point, at distance $x = x_V \pm \Delta x$ from some origin. Δx is the acceptable limitation on the precision of the analogue memory, determined by the physical length of the device and the number of values it is required to distinguish. If 10^R real values ($R \times \log_2 10$ bits) are allowed and the maximum value is L (the length of the device), then

$$\Delta x = \frac{L}{2 \times 10^R} \quad (6.1)$$

We will denote the interval $[x_V - \Delta x, x_V + \Delta x]$ by I_V .

In quantum mechanical terms, just prior to t_0 , A is described by a state vector $|\psi_0\rangle$ (Dirac's formalism). Placing A at the chosen point involves "collapsing" $|\psi_0\rangle$ into a new state confined to a positional subspace spanned by the eigenkets $|x\rangle$ with $x \in I_0 = [x_V - \Delta x_0, x_V + \Delta x_0]$. Δx_0 represents the accuracy of the measuring device and it is essential that $\Delta x_0 < \Delta x$ so that $I_0 \subset I_V$. Define $K > 1$, by $K = \frac{\Delta x}{\Delta x_0}$.

This process of "state preparation" is entirely equivalent to performing a measurement on A which leaves it somewhere in the required subspace. Unfortunately, any actual measurement has a non-zero probability of failing to yield a result in I_0 . In fact, the probability of success is given by:

$$P(x_V - \Delta x_0 < x < x_V + \Delta x_0) = \int_{x_V - \Delta x_0}^{x_V + \Delta x_0} |\langle x | \psi_0 \rangle|^2 dx^2 \quad (6.2)$$

It is reasonably easy to circumvent this problem however. Since the store operation, being a measurement, returns a positional value, it is easy to tell at once if it has failed (if the value lies outside I_0) and we can assume that any number of further attempts can be made until success is achieved. For the sake of simplicity suppose the store operation succeeds on the first attempt at time, t_0 , whereupon the new state vector of A is given by:

$$|\psi_s\rangle = \frac{1}{N} \int_{x_V - \Delta x_0}^{x_V + \Delta x_0} |x\rangle \langle x | \psi_V \rangle dx \quad (6.3)$$

where N is a normalisation constant. In wave mechanical terms, this describes a wave packet confined to the region I_0 . From the postulates of quantum mechanics, at a time immediately following t_0 , a second measurement on A will also retrieve a value within I_0 ; however, if left undisturbed, $|\psi_s\rangle$ will evolve deterministically, according to the Schrödinger equation and a later measurement will have no such guarantee. The Schrodinger equation can be solved analytically for certain shapes of wave packet, such as the Gaussian but a more general argument is presented below, applicable to a packet of any form. The conclusions are universal and the argument can be extended easily to a packet in a locally constant potential field (e.g. generated by neighbouring atoms). The key point is that, as soon as the wave packet is formed, it begins to spread (dispersion) outside I_0 . So long as it remains inside the interval I_V a retrieve operation (measurement) will yield a result in I_V , but once components outside I_V develop, the probability of a read error becomes non-zero and then grows rapidly. Let Δt be the maximum time interval after t_0 during which a measurement is still safe. The analogue memory must be refreshed by performing a new measurement before or an erroneous result may be generated. Note that any real measurement on an interval of length x_V will take at least $\frac{2x_V}{c}$, where c is the speed of light in vacuum, since a light beam must travel to and from A to perform the measurement. It follows that if $\Delta t < \frac{2x_V}{c}$ then the memory will not be feasible.

Since momentum and position are conjugate observables, their x -dimensional operators \mathbf{X} and \mathbf{P}_x obey the commutation relation

$$[\mathbf{X}, \mathbf{P}_x] = i\hbar \quad (6.4)$$

where \hbar is Planck's constant divided by 2π . From this it follows that the uncertainties (root mean square deviations over a quantum ensemble of identical systems) in the initial (time t_0) values of x and the p_x satisfy the so-called 'Uncertainty Relation'

$$\Delta p_x \cdot \Delta x_0 \geq \frac{\hbar}{2} \quad (6.5)$$

Since the mass of A written m_A , after the 'store' measurement at t_0 , A is moving with an expected velocity of $\frac{\Delta p_x}{m_A}$ away from the prepared position. Naively, therefore, one might expect that at time Δt , A will have travelled a distance of $\frac{\Delta p_x \cdot \Delta t}{m_A}$ away from its prepared position. This however is a quasi-classical argument and underestimates the problem, as one might suspect from noting that the Uncertainty Relation is an inequality and $\frac{\hbar}{2}$ a lower bound. The actual positional uncertainty, Δx , after the critical time Δt (at which the spreading wave packet exceeds its safe bounds), can be obtained via an application of Ehrenfest's Theorem from which it can be concluded ([14] p342) that:

$$\Delta x^2 = \left(\frac{\Delta p_x \cdot \Delta t}{m_A} \right)^2 + (\Delta x_0)^2 \quad (6.6)$$

In order to determine the theoretical limitations of this system we can now place an upper bound on Δt . Since

1. $\Delta x = K\Delta x_0$,
2. $2 \times 10^R \Delta x = L$ and ,
3. $\Delta p_x \geq \frac{\hbar}{2\Delta x_0}$,

it follows that the maximum value Δt after which the memory becomes unsafe is given by:

$$\Delta t \leq \frac{\sqrt{K^2 - 1}}{K^2} \times \frac{L^2}{4 \times 10^{2R}} \times \frac{2m_A}{\hbar} \quad (6.7)$$

It is easy to verify that

$$\max \left(\frac{\sqrt{K^2 - 1}}{K^2} \right) = \frac{1}{2} \quad (6.8)$$

Approximating $\hbar \simeq 10^{-34}$ gives,

$$\Delta t \leq 0.25 \times 10^{34-2R} \times L^2 m_A \quad (6.9)$$

For viability, $\Delta t \geq \frac{2x_V}{c}$ so, since $c \approx 3 \times 10^8 (ms^{-1})$:

$$m_A > \frac{8 \times 10^{2R-42}}{3L} \times \frac{x_V}{L} \quad (6.10)$$

This depends on the value being stored. However, the memory must work for all values of x_V so we can set $x_V = L$. We now have a final result:

$$m_A \times L > 2.6 \times 10^{2R-42} \quad (6.11)$$

or alternatively

$$m_A \times \Delta x > 1.3 \times 10^{R-42} \quad (6.12)$$

The limitation is applicable to any scheme which relies on the physical position of an object to store a real number. To increase the range of a positional analogue representation by 1 bit of precision requires the mass \times length product to increase by approximately a factor of 4. It is very easy to see that a system of this type cannot scale unless allowed to grow arbitrarily in physical size, with consequent implications not only for the feasibility of construction but also for the speed at which stored data might be retrieved.

In contrast to a digital machine where the resources used to perform higher accuracy computation grow linearly with the number of bits of accuracy, the resources needed by the analogue machine grow exponentially with the accuracy of the calculation.

The following table gives approximate values for the parameters required for one-dimensional analogue positional memories at several values of precision (in bits), where possible for $L=10\text{m}$. The 32-bit example requires masses of the order of a few atoms of uranium placed to an accuracy of 2.5nm, technically not an outrageous scenario.

Precision ($R \log_2 10$)	Length	Approx Mass	$2\Delta x$
32	10m	$5 \times 10^{-24}\text{kg}$	$2.3 \times 10^{-9}\text{m}$
64	10m	10^{-4}kg	$5.4 \times 10^{-19}\text{m}$
128	5km	$6 \times 10^{31}\text{kg}$	$1.4 \times 10^{-35}\text{m}$

However, above 32-bit precision the systems become increasingly implausible even with exotic technologies. For the 128 bit example, L is chosen to ensure that Δx exceeds the Planck Length, L_p ($1.6 \times 10^{-35}\text{m}$), which is the minimum precision possible in any physical measurement of position. However even accommodating this fundamental constraint, m_A is of the order of 50 solar masses and must occupy a space at least equal to its Schwarzschild radius, S_A . As this is given by:

$$S_A = m_A \times 1.48 \times 10^{-27} \quad (6.13)$$

A would be at least several km across even if it was a black hole. When dealing with objects with sufficient gravitational fields, the Uncertainty Principle has to be modified to take account of gravitational forces and, while the detail is still not fully settled, in most current theories of quantum gravity it is believed[68] to take a form such as:

$$\Delta x \geq \frac{\hbar}{2\Delta p} \times k L_p^2 \frac{\Delta p}{\hbar} \quad (6.14)$$

where k is a constant. This inequality (the so-called Generalised Uncertainty Principle) is interesting because it predicts that when dealing with massive objects that the Δx associated with a given Δp may significantly higher than in the non-gravitational case. The GUP also confirms that the uncertainty in position can never be less than L_p regardless of how imprecise the momentum knowledge is: this is in agreement with the claim that L_p is the smallest measurable quantum of length. We conclude:

1. Any physically build-able analogue memory can only approximate the reals and there are very definite limits as to the accuracy achievable.
2. Analogue storage of reals, will for high precision work, always be outperformed in terms of device economy by digital storage.
3. Physically build-able analogue computers can not rely upon the availability of exact stored real numbers to outperform digital computers.

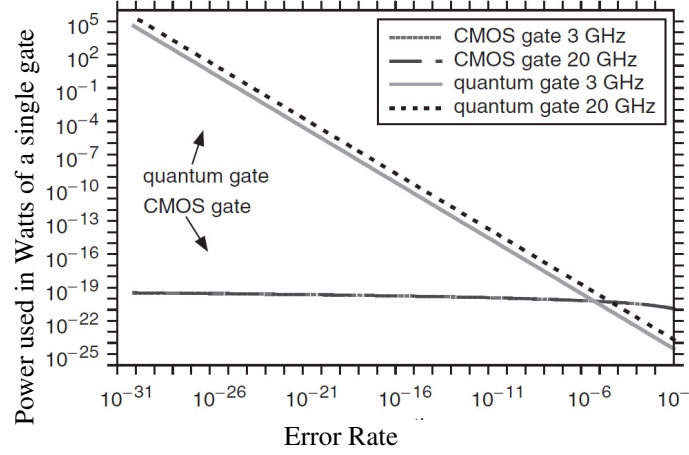


Figure 6.1: Comparative power used in CMOS and quantum gates as a function of error rate. Derived from [44].

6.2 Error rates in classical and quantum gates

We will discuss quantum computing in greater depth in Chapter 6, but this is an appropriate place to look at the power-consumption limits of quantum computing.

Both classical and quantum computer gates are subject to stochastic errors. In the classical case these are thermodynamic in character, in the quantum gates they are due to vacuum or zero point noise.

Banacloche[34, 35] shows that for a quantum logic gate the error rate ϵ_q is given by

$$\epsilon_q > \frac{fh}{E} \quad (6.15)$$

where f is the frequency of operation, h Planck's constant, and E the energy used to operate the gate.

For a classical gate the error rate is given by

$$\epsilon_c > \frac{2}{\sqrt{3}} e^{\frac{-E}{kT}} \quad (6.16)$$

where T is the gate temperature and E is the minimum energy dissipated in the classical circuit in the course of the operation. The exponential increase in the error rate with rising temperature is why cooling is so important.

If we consider clock frequencies in the GHz range, we can look at the minimal power dissipated by CMOS and quantum gates as a function of their error rates. These are shown in in Figure 6.1. At any error rate below 10^{-6} the

CMOS system has a lower energy requirement. Current CMOS technology can achieve error rates in the order of 10^{-25} . To achieve this error rate the quantum gate would require around 100 Joules for each switching operation. A single quantum gate operating in the GHz range would be using of the order of 100 Megawatts of power[44].

Whilst quantum computing does hold promise as a means of reducing the complexity of algorithms it seems unlikely that it will allow us to escape from the power consumption limits posed by classical computing.

Chapter 7

Hyper computing proposals

– Lewis and Paul

There have been many attempts to articulate new systems for computability that are claimed to transcend the limitations of Church-Turing systems, termed for example *hypercomputing* or *super-Turing*. Copeland[15] provides a thorough summary. Our view is that none of these proposals have yet made a convincing case for dropping the fundamental theoretical basis on which computer science has rested until now. In this final chapter we will review some of the alternative models of computation that have been put forward recently. We will look at:

1. Proposals to use infinities in Turing machines.
2. Proposals to use infinitely precise analogue computers.
3. Proposals to use black holes for computing.
4. Proposals to use new calculi.

7.1 Infinite Turing machines

Universal Computers proposed by Turing are material apparatuses which operate by finite means. Turing assumes that the computable numbers are those that are computable by finite machines, and initially justifies this only by saying that the memory of a human computer is necessarily limited. By itself this is not entirely germane, since the human mathematician has paper as an aide memoir and the tape of the TM is explicitly introduced as an analogy with the squared paper of the mathematician.

Turing is careful to construct his machine descriptions in such a way as to ensure that the machine operates entirely by finite means and uses no techniques that are physically implausible. His basic proposition remained that: “computable numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means.”.

Turing thus rules out any computation by infinite means. If infinite computation were to be allowed, then the limitations introduced by the TM would not apply.

Copeland[15] proposes the idea of accelerating TMs whose operation rate increases exponentially so that if the first operation were performed in a microsecond, the next would be done in $\frac{1}{2}\mu s$, the third in $\frac{1}{4}\mu s$, etc. The result would be that within a finite interval it would be able to perform an infinite number of steps.

The machine could for example compute π exactly:

Since a Turing machine can be programmed to compute π , an accelerating Turing machine can execute each act of writing that is called for by this program before two moments of operating time have elapsed. That is to say, for every n , the accelerating Turing machine writes down the n th digit of the decimal representation of π within two moments of operating time ([15] p. 284)

Another supposed advantage is that they make the halting problem readily solvable. The accelerating machine simulates the specified TM for an infinite number of steps within a finite period and gives a definite answer as to whether or not it will halt.

This obviously evades Turing's stipulation that computations must be by finite means, and, in the process, evades all possibility of physical realisation. A computing machine must transfer information between its component parts in order to perform an operation. If the time for each operation is repeatedly halved, then one soon reaches the point at which signals travelling at the speed of light have insufficient time to propagate from one part to another within an operation step. Beyond this speed the machine could not function.

In a hypothetical Newtonian universe without the constraint of a finite speed of light, this particular obstacle would be eliminated, but one would immediately face another. In order for the Newtonian machine to compute infinitely fast, its (now presumably mechanical) components would have to move infinitely fast and thus require infinite energy. Given that the machine would be dissipative [50] the heat released would raise its temperature to an infinite extent causing it to disintegrate.

Hamkins[37] discusses what could be computed on Turing machines if they were allowed to operate for an infinite time, but Turing ruled this out with obvious good reason.

Etesi and Nemeti[28] extend the idea of accelerating Turing Machines by proposing a gravitational mechanism by which the acceleration can be done. They suggest the use of a pair of computers with one (A) orbiting and another (B) falling towards the event horizon of a Kerr black hole. As the computer (B) approaches the event horizon its time is slowed down relative to the passage of time for (A). The closer it gets to the event horizon the slower its passage of time gets relative to that of (A). They propose that computer (A) be made to work through the infinitely many instances of some recursively enumerable set. An instance they cite is running through the infinitely many theorems of

ZFC set theory to see if any are false. If among the infinite set one is found to be false, a light signal is sent to (B) indicating this. Because of the slowdown of time for (B), things can be so arranged as to ensure that the light signal arrives before the event horizon is crossed.

Esti and Nemeti show remarkable ingenuity in working out the details of this scheme. They have a plausible response to the most obvious objection: that the light signal from (A) would be blue shifted to such an extent that (B) could not detect it. They suggest that (A) computes the degree of blue shift that the signal would experience and selects a suitably long wavelength and modulation system to compensate. This is not, however, an adequate response. There remain two serious objections:

1. Computer (B) is assumed, like any other TM, to operate with clock ticks. The clock cycle is the smallest time within which the machine can respond and carry out any action. There will, within (B)'s frame of reference, be a finite number of clock ticks before the event horizon is crossed. Consider the last clock tick before the horizon is crossed, i.e. the clock cycle that is in progress as the horizon is crossed. Prior to the start of this cycle, machine (A) will have only searched a finite number of the theorems of ZFC set theory. During the final clock cycle of (B) the entire infinite residual of the set of theorems are checked by (A). But any message sent from (A) whilst processing the infinite residual must occupy less than a clock cycle from (B)'s perspective. As such it will be too brief to register at (B).
Any signal that (B) can respond to will correspond to (A) only having searched a finite part of the infinite set of theorems.
2. If we consider things from the standpoint of (A), what we are demanding is that it continues to operate reliably, searching through theorems, not just for millions of years, but for an *infinite* number of years. The assumptions of infinite reliability and an infinite energy source to keep (A) operating, are clearly impossible.
3. For A to orbit the black hole for an infinite amount of time, the black hole would have to exist from now till infinity. If Hawking[40] is right black holes have very long but still finite existences.

7.2 Infinitely precise analogue computers

We have already discussed the limits that quantum measurement poses on the possibility of constructing computing machines that would store information in an analogue mechanical fashion. Our example was given in terms of a mass whose position encoded a number. The idea of encoding numbers in mechanical positions is not absurd, machines described in Chapter 3 used this sort of encoding. But we demonstrated in Section 6.1 that there is an inherent limit to the accuracy of such encoding.

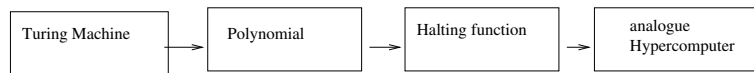


Figure 7.1: Da Costa's proposal.

A general motivation for the construction of such machines is given by Da Costa in a paper ambitiously titled *How to build a hypercomputer*[18]. The greater part of this account is given over to a mathematical justification of why the construction of a hypercomputer is feasible. Da Costa relies on the concept of an universal Diophantine polynomial. He cites Martin Davis [19] as describing an algorithmic procedure out of which, given a Turing machine with input a $M_m(a)$, we obtain a polynomial $p_m(a, x_1, \dots)$ so that it has roots if and only if the Turing Machine converges (outputs some result). From this he then defines a real valued and real parametered halting function such that the function will be less than 1 if the Turing Machine halts and greater than one if it goes into an infinite loop. This function it should be noted initially requires an integral over an infinite range, though he later proposes to compactify this.

Given this mathematical solution it is he says, just a matter of engineering to build the machine.

If we could have this halting function, surely we could just get a conventional computer to evaluate it?

But we know can not do this because were we able to do that because of Turing's proof. The problem is that we cannot compute on a TM the required infinite integrals - these integrals substitute for the potentially infinite time required to determine if a TM will halt. Da Costa writes that building the hypercomputing machine from the mathematical specification is just a matter of engineering.

Well in that case it is a matter of engineering more suited to Olympians than us mere mortals since it requires, among other things, encoding π in the machine to an infinite precision. Da Costa gives us no clue as to how he proposes to achieve this degree of precision. Others have rashly come forward with concrete proposals. It is rather easy to come up with ideas for computing devices which have a superficial plausibility because they are expressed in terms of idealised physical models in which some real world constraints are absent. In this section we will look at a few such proposals. In every case we shall see that their apparent plausibility rests on a judicious choice to ignore key features of the physical world as understood today.

7.2.1 Newtonian computing

Newtonian mechanics is a particularly beguiling area for those exploring the possibilities of infinitary computation. What we call Newtonian mechanics is both that form of abstract maths originating in the calculus of Newton, and a set of laws that allow us to use this mathematical apparatus to make predictive models of reality. The abstract maths was initially controversial with its

fluxions and infinitesimals[83]. But its predictive powers proved to be so great that philosophical doubts about the idea of infinite divisibility were put into abeyance.

When Newton's laws are used in a practical sense what they do is stipulate a set of algorithms for making finite predictions about things like observed planetary positions. They specify at a relatively abstract level what a mathematician should do to compute the position of for example Saturn on a particular day. The calculations may be done by hand or they may be done on a computer.

Suppose they are done on a computer. There are then a large number of possible programmes, in various programming languages that are valid applications of Newton's laws to the problem. According to the care with which the codes have been written, the numerical precision of the floating point calculation etc, these will give results of greater or lesser accuracy. But whatever we do, the predictions are always given as computable numbers – to some chosen finite number of digits.

The fact that calculations are always to a finite precision is not a fundamental problem. We can always choose a number of digits sufficient for a given practical purpose – whether it is pointing a telescope or navigating a probe to Jupiter. We can choose an algorithm in which the numerical errors will be a lot less than our uncertainties in observation and measurement of the actual angular position of Saturn or the actual orbit round Jupiter that our probe takes up.

Because we can use Newton's laws to write algorithms that compute movements to an arbitrary degree of numerical precision, a temptation arises to believe that in the reality physical bodies do move with an infinite accuracy.

It has been known since the early 20th century that, whilst Newtonian mechanics makes excellent predictions of a wide range of physical systems, at extremes of velocity, density and very small scales its success falters. Classical physics seems to falter at just the places where a relentless pursuit of its logic would lead us to infinities. Quantum theory was introduced in Einstein's paper on the photo-electric effect[26] in order to deal with the paradox created for classical electrodynamics by one such infinity - the so-called ultraviolet catastrophe. Einstein opened his paper by pointing to the inadequacies of a continuum approach, in this case the continuum presupposed by Maxwell's theory:

A profound formal distinction exists between the theoretical concepts which physicists have formed regarding gases and other ponderable bodies and the Maxwellian theory of electromagnetic processes in so-called empty space. While we consider the state of a body to be completely determined by the positions and velocities of a very large, yet finite, number of atoms and electrons, we make use of continuous spatial functions to describe the electromagnetic state of a given volume, and a finite number of parameters cannot be regarded as sufficient for the complete determination of such a state.

According to the Maxwellian theory, energy is to be considered a continuous spatial function in the case of all purely electromagnetic phenomena including light, while the energy of a ponderable object should, according to the present conceptions of physicists, be represented as a sum carried over the atoms and electrons. The energy of a ponderable body cannot be subdivided into arbitrarily many or arbitrarily small parts, while the energy of a beam of light from a point source (according to the Maxwellian theory of light or, more generally, according to any wave theory) is continuously spread an ever increasing volume.[26]

His response was to propose that light was quantised in the form of photons, and from this eventually followed the rest of the quantum theory.

If a modern paper suggests that some form of classical mechanics allows certain forms of infinitary calculation, what does this mean?

1. that one can logically deduce that certain equations will produce infinities in their solutions?
2. that certain real physical attributes can take on infinite values?

In the light of quantum theory we have to answer no to the last question even if we say yes to the first. If the maths you are using to represent the material world gives infinities in your equations, then that tells you more about the errors in your mathematical model than it does about reality.

Smith [71] gives as an example of uncomputability in Newtonian physics certain N -body problems involving point masses interacting under gravity. Because these can approach arbitrarily close to one another, at which point their mutual gravitational attraction becomes arbitrarily high, he suggests that one could so configure a set of initial conditions that the particles would move through an infinite number of configurations in a finite time interval. He argues that no Turing machine could simulate this motion in a finite time. This could be interpreted either:

1. as a limitation on the ability of computers to simulate the world;
2. or as a means by which, with suitable encoding, a physical system could be used to determine algorithmically uncomputable questions.

But let us consider the very properties that would allow this infinitary process - infinite velocities, point masses with position and no magnitude. These are the very points where Newtonian mechanics breaks down and has to be replaced with relativistic or quantum mechanics. The ability of a theory to produce infinities points to a weakness in conceptualisation. Smith, concurs, he goes on to show that once relativistic constraints on either velocity or density are introduced, the equations of motion for the system give finite results, and in consequence become algorithmically soluble.

The infinities in the initial differential equations thus emerge as a consequence of the axiomatic structure of the calculus rather than a property of the real world.

Beggs and Tucker [2] also explore of the extent to which Newtonian mechanics would allow the hypothetical construction of infinitely parallel computing engines. They do not claim that such machines could actually exist, but ask: what sort of mechanics would allow such machines to exist?

This is an interesting, if speculative line of enquiry. But to pursue it one has to be consistent. It would be reasonable enough, when investigating the hypothetical computational possibilities of a Newtonian universe, to ignore constraints imposed by relativity theory and quantum theory. But Newtonian mechanics imposes other constraints that may not immediately be obvious.

Beggs and Tucker derive the ability to perform hypercomputation from an infinite plane of conventional computers, which purport to use tricks of Newtonian mechanics to allow infinitely fast transmission of information.

They use two tricks, on the one hand they propose to synchronise the clocks of the machines by using infinitely long, rigid rods. The rod is threaded through all the machines and a push on the rod starts all the computers synchronously. They concede that for this to happen the rod must not only be perfectly rigid, but it must either be massless or have a density which exponentially tends to zero as one moves away from the starting point. This is necessary if the rod is to be moved by applying a finite force.

It is not clear in what sense such rods can be said to be Newtonian.

There is an old technical term for rods like this: wands. When Turing had recourse to 'Oracles' he deliberately used magical language to indicate that this recourse was make-believe.

They propose that the infinite collection of computers will be able to return results to an initiating processor using an ability to fire cannonballs up and down along parabolic trajectories at arbitrarily high velocities. The arrival of such a cannonball transmits a binary truth value. They further propose that in a finite time interval an infinite number of cannonballs can be projected, in such a way that at any given instant only one is in flight.

Their argument is that given a projectile of mass m we can project it at arbitrarily high speed if we use enough energy. Given a distance b that the cannonball has to travel, we can make the time of flight arbitrarily small by selecting a sufficiently high velocity of travel. The proposal is that one use cannons and an arbitrarily large supply of gunpowder to achieve this. This argument is contradicts Newtonian mechanics on several points.

1. The immediate problem is that whilst there is according to Newton no limit to the ultimate velocity that a particle subjected to uniform acceleration can reach, this velocity is not reached instantaneously. Let us suppose we use perfect cannons, ones in which the accelerating force f due to the combustion of gunpowder remains constant along the length of the barrel.

Achieving this is very hard, it requires all sorts of constraints on the way

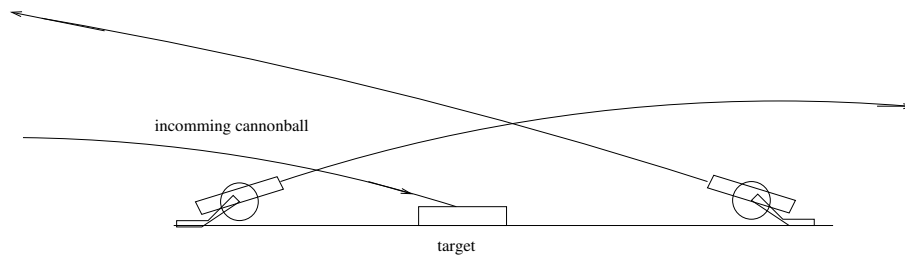


Figure 7.2: Cannonballs and their clearances.

the powder burns. It was a sought after but never achieved goal of 19th century ballistic engineering, much experimentation with powder grain size went into the quest, see for example [69].

The acceleration of a ball of mass m will then be $s = \frac{f}{m}$. A cannonball spends a period in the cannon $\frac{v}{a}$ being accelerated that is proportional to the velocity ultimately attained. Thus whilst the flight time $\frac{b}{v}$ tends to zero as velocity increases, total travel time $\frac{b}{v} + \frac{v}{a}$ does not.

2. There is a further problem with assuming that cannons can attain an arbitrary velocity. As one increases the charge in a gun an increasing amount of the work done by the powder consists in accelerating the powder itself down the barrel. The limiting velocity achievable is that of the exit velocity of the gas of a blank charge. This, in turn, is limited by the speed of sound in the driving gas. For a highly energetic hydrogen/oxygen explosion this sets a limit of about 2100 m/s. Techniques such as hybrid explosive and light gas guns can produce a limited improvement in velocity[17], but certainly not an arbitrary speed. Rail guns [67] can achieve higher velocities using electromagnetic acceleration. It is not clear whether eletro-magnetic propulsion is permissible in Beggs and Tucker's chosen model of mechanics.
3. There is also a problem of trajectory Begg and Tucker further assume parabolic trajectories to ensure that the cannonballs fly clear of intervening obstacles like other cannons prior to hitting their targets, see fig. 7.2. The balls will take a finite time for gravity to retard their upward velocities.

Even on its own Newtonian terms, the proposed hypercomputer is inconsistent. It is analogous to Fredkin's billiard ball computer, superficially plausible but on examination, inconsistent.

7.2.2 Bournez and Cosnard's analogue super-Turing computer

An examination of a concrete example of another couple of proposed super-Turing analogue computers[8, 5] illustrates the sorts of errors that would vitiate

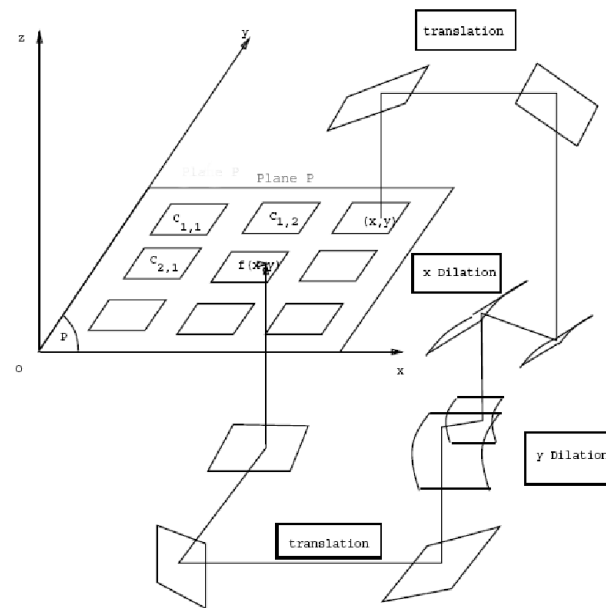


Figure 7.3: An analogue computer proposed by Bournez and Cosnard. Reproduced from [8].

its operation. Bournez and Cosnard propose to use two real valued variables corresponding to the x, y coordinates of particles, (presumably photons) passing through plane P in Figure 7.3. The binary expansion of these real valued coordinates could then be used to emulate the left and right parts of a TM tape [47, 10]. They argue that the machine could, in addition, be used to simulate a class of two stack automata whose computational powers might exceed those of TMs. The gain in power comes from the ability of their proposed stack automaton to switch on the basis of the entire contents of an unbounded stack, rather than on the basis of what is directly under the TM head. They suggest that if one had available iterated real valued functional systems based on piecewise affine transforms, such analogue automata could be implemented. In the proposed physical embodiment given in Figure 7.3, multiplication by reals would be implemented by pairs of parabolic mirrors, and translation by arrangements of planar ones.

The authors, in striking contrast to researchers active in the area 50 years earlier, fail to identify the likely sources of error in their calculator. Like any other analogue system it would be subject to parameter, operator and variable errors. The parameters of the system are set by the positions and curvatures of the mirrors. The placement of the mirrors would be subject to manufacturing errors, to distortions due to temperature, mechanical stress etc. The parabolic mirrors would have imperfections in their curvature and in their surfaces. All of these would limit the number of significant digits to which the machine could calculate. But let us, for the moment, ignore these manufacturing errors and concentrate on the inherent uncertainty in the variables.

Because of the wave particle duality any optical system has a diffraction limited circle of confusion. We can say that a certain percentage of the photons arriving from a particular direction will land within this circle. The radius of the circle of confusion is inversely proportional to the aperture of the optical system and directly proportional to the focal length of the apparatus and to the wavelength of the photons used. The angle to the first off-center diffraction peak $\Delta\theta$ is given by

$$\sin(\Delta\theta) = \frac{\lambda}{A} \quad (7.1)$$

where A is the aperture and λ the wavelength.

By constraining the position of the photon to be within the aperture, we induce, by the principle of Heisenberg, an uncertainty in its momentum within the plane of the aperture.

To see what this implies, we give some plausible dimensions to the machine. Assume the aperture of the mirrors to be 25mm and the path length of a single pass through the system from the first mirror shown in Figure 7.3 back to Plane P to be 500mm. Further assume that we use monochromatic light with wavelength $\lambda = 0.5\mu$. This would give us a circle of confusion with a radius $\Delta_{f(x,y)Mirror} \approx 10\mu$.

If plane P was $\frac{1}{10}$ of a meter across the system would resolve about 5000 distinct points in each direction as possible values for $f(x, y)$. This corresponds

to about 12 bits accuracy.

The dispersion $\Delta_{f(x,y)Mirror}$ accounts only for the first pass through the apparatus. Let us look at the parametric uncertainty in x, y to start out with.

One wants to specify x, y to greater accuracy than $f(x, y)$ so that $\Delta_{x,y} < \Delta_{f(x,y)}$. Assume we have a source of collimated light whose wavefronts are normal to the axis of the machine. A mask with a circular hole could then constrain the incoming photons to be within a radius $< 10\mu$. Any constraint on the position of the photons is an initial aperture. If this aperture $\leq 10\mu$, its diffraction cone would have a $\Delta\theta \approx 0.05$ radians. Going through the full optical path the resulting uncertainty in position $\Delta_{f(x,y)Mask(\Delta_{x,y})} \approx 25\text{mm}$. We have $\Delta_{f(x,y)Mask(\Delta_{x,y})} \gg \Delta_{f(x,y)Mirror}$.

The narrower the hole in the mask the more uncertain will be the result $f(x, y)$. In other words the lower the parametric error in the starting point, the greater the error in the result.

There will be a point at which the errors in the initial position and the errors due to the diffraction from the mirrors balance: when $\Delta_{f(x,y)Mirror} + \Delta_{f(x,y)Mask(\Delta_{x,y})} \leq \Delta_{x,y}$. From simple geometry this will come about when the ratio $\Delta_{x,y}/L \approx \lambda/\Delta_{x,y}$ so

$$\Delta_{x,y} \approx \sqrt{L\lambda} \quad (7.2)$$

where L is the optical path length of the computation. For the size of machine which we have assumed above, this implies $\Delta_{x,y} = 500\mu$. Its accuracy of representation of the reals is thus less than 8 bits ($= -\log_2(\frac{500\mu}{100\text{mm}})$), hardly competitive with existing digital computers.

The evaluation of $f(x, y)$ corresponds to a single step of a TM program. If n is the number of TM steps, optical path length is nL . By (7.2), the optimal initial aperture setting $\Delta_{x,y} \propto \sqrt{n}$. Each fourfold increase in the execution length of the program, will reduce by 1 bit the accuracy to which the machine can be set to compute.

If we want to make an optical machine more accurate we have to make it bigger - the growth in the size of astronomical telescopes bears witness to this. For every bit of accuracy we add, we double the linear dimensions. If M is the mass of the machine, and b its bit accuracy then $M \propto 2^{3b}$.

For a conventional digital VLSI machine, $M \propto b$ and the mass of the arithmetic unit grows as $b \log b$. For any but the least accurate calculations this sort of optical analogue machine will be inferior to conventional machines.

7.2.3 Optical prime factorisation

The fame accorded to Shor's prime factorisation algorithm encouraged another attempt at the use of interference as a mechanism for prime factorisation. Shor relied on quantum interference, whereas this proposal relies upon optical interference. Blakey's design for a prime factorisation machine[5, 4] displays considerable geometric ingenuity.

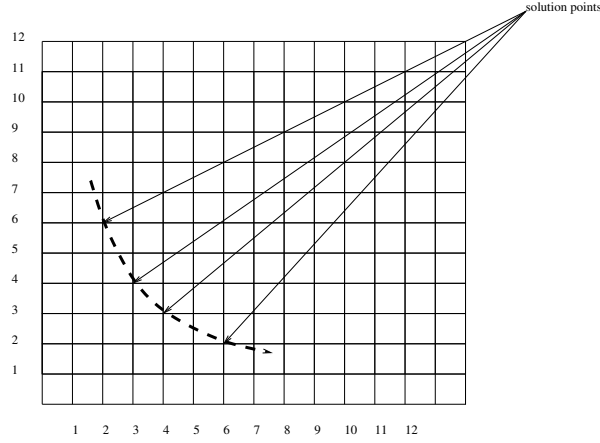


Figure 7.4: The hyperbolic curve $y = N/x$ passes through the grid points that are prime factors of N , illustrated with $N = 12$

It addresses the problem of finding the prime factors of a large integer. This problem is classically of order \sqrt{n} for an integer n . Basically one just tries dividing all possible factors up to the square root, at which point you will have found all the factorisations. \sqrt{n} . Blakey observes that if you have a grid in two dimensions as in Figure 7.4, then a hyperbolic curve $y = N/x$ will pass through the prime factors of N .

He then asks can we set up a physical apparatus that will generate the required grid and a hyperbolic curve. His suggested equipment is shown in Figure 7.5.

A point source of light S along with a group of 3 mirrors is used to set up a grid pattern of constructive and destructive interference across the plane between the mirrors M_2, M_3 . The hyperbola is then created as the plane intersection between the conic arc P, C and the mirror plane. Along the circle C which forms the base of the cone there are photo detectors.

The principle of operation is that:

Diminution of second-source radiation due to its having passed through an integer (that is, maximally active) point in the first-source interference pattern (which models the grid of integer points) is detected at the sensor; the coordinates of points of such detection can, Turing-computationally trivially, be converted into the coordinates of the sought integer points on the cone. These latter coordinates are factors of n . ([6], page 4)

It is unclear why the interference caused by the standing wave pattern in the mirror plane is supposed to impede the passage of light beams on the path from P to C . Light beams pass through one another without effect. One only

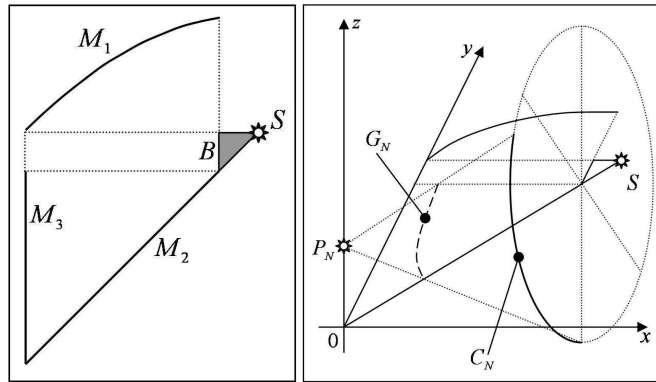


Figure 7.5: Left, plan view of the machine, M_i mirrors, S light source. Right, apparatus to generate the hyperbola as a conic section. P is another light source, and C a circle arc made up of detectors.

observes interference if there is more than one possible route that a photon can have taken between source and destination. The photons arriving at the detectors on C can only have come from P , so there will be no interference between these and the photons emitted from S .

As described the machine would not work. It can be modified in a sympathetic way to make it work roughly as intended. Blakey does not mention the use of any photosensitive materials. One could imagine a modification of his design so that, during a first phase, an initially transparent photosensitive plate is placed in the mirror plane. The light from S could be used to expose the plate, darkening it at points of constructive interference. There are of course a variety of other ways one could lay out such a grid, but Blakey's interference patterns are one possible technique.

Given that we have a grid, is the rest of the design plausible?

The basic design seems to assume that a very fine grid will cast a shadow pattern that will also be a very fine grid. But this is not so. A grid with holes whose size is comparable to a wavelength, will act as a two dimensional diffraction grating, which will split the original light rays from P into a strong principle component and a several diffracted component. There will be no shadows cast by the grid and hence the desired effect on the detector will not be observed, since the principal component will seem to pass directly through though with diminished brightness.

To get a shadow cast you would require a grid that was large relative to the wavelength of the light and the focal length used. Figure 7.6 illustrates the constraints. One constraint that Blakey fails to take into account is that the sensors will have gaps between them and there is the possibility that the peak in brightness corresponding to a hole in the mask will coincide with a gap. This is an instance the well known problem of aliasing in digital photography.

But ignoring that problem for a moment, let us consider how to overcome the blurring caused by diffraction. We could to constrain the separation between the principle and first diffracted components arriving on the sensor array to be no more than half the separation between sensors. This gives us the constraint

$$w/A < \frac{1}{2}B/F \quad (7.3)$$

Let us add the further constraint that $A = cB$ with $c > 1$ in order to have enough sensors to have some hope of compensating for the aliasing problem. Let us set $c = 2$. We thus have the derivation

$$\frac{w}{2B} = \frac{B}{2F}$$

$$2wF = 2B^2$$

$$B = \sqrt{wF}$$

Since we can only currently build sensors of this density of about 4cm across, we can use Blakey's diagrams to fix a plausible value of the focal length F to be about 2cm. Let us assume we use visible light with a wavelength 0.7 microns. This gives us a figure of B the sensor separation, of about 0.1 mm and a grid spacing of twice that. If we assume that the practical size of the grid is set by the sensor dimensions to also be about 4cm we see that the optical apparatus could hope to determine the prime factors of numbers up to about 200. Finding prime factors in this range is not a serious problem for digital techniques. Since the prime numbers used in cryptography tend to be of the order of 2^{256} it is evident that the analogue technique is of far too low a precision to be of use.

The algorithmic complexity would be worse than on an orthodox machine the problem would be to find the maximum brightness over the sensors, which are of order N so the time complexity is N rather than \sqrt{N} on the conventional computer. The space complexity is N^2 since the area of the grid grows in this manner.

7.3 Conclusions

We have reviewed a number of proposals for trans Turing machines. In each case we have seen that the machine does not live up to its promise. Now in one sense this is not surprising since many designs of conventional computers have faults in them initially. A computer is something very complicated and it is hard to get the design right initially. Bugs are uncovered in the initial designs as you try and fill in details. Some bugs survive until the first prototypes are built and tested.

Unless and until a hypercomputer is built we have to reserve engineering judgement on whether it is possible for one to exist. The current proposals all seem to hit snags well before the prototype stage.

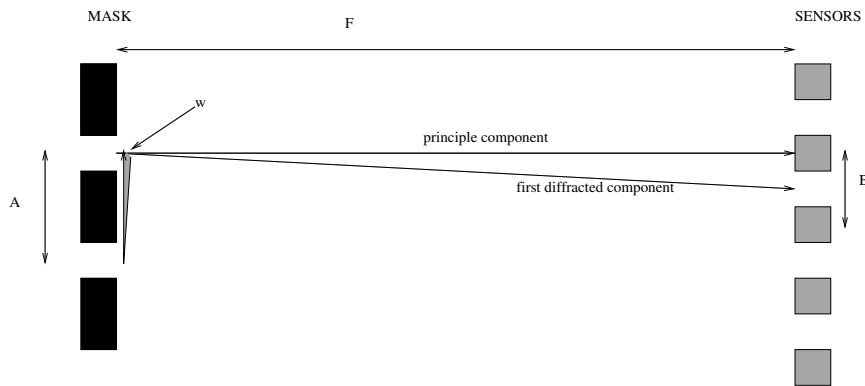


Figure 7.6: Detailed view of how a mask interacts with a sensor array. The wavelength of the light is w , the separation of the mask and the sensor array is F , and A, B are the separations of the mask holes and the sensors respectively.

But suppose that a machine could be built that would solve some hypercomputational problem what use would it be?

Suppose that it gave us answers about some uncomputable function. We would not be in a position to verify whether the answer was correct unless there was an independent way of verifying it by algorithmic means. We would need multiple different designs of hypercomputer, working in different ways so that they could check each other for consistency.

But in what sense would this hypercomputer be a computer?

It would not be a general purpose computer. If it had general purpose computing abilities, up to the level of self emulation, then it would be possible to construct a self referential problem analogous to the paradoxical problem that Turing constructed to demonstrate the halting problem.

Instead it would be a special purpose measuring device rather than a general purpose computer. A Turing Machine can not answer the question 'how much do I weigh'. From the standpoint of TM's this is an uncomputable number. But a bathroom scale can answer it, and we can check such scales for mutual consistency. In that sense we already have non-algorithmic devices that give us numeric answers, we just do not call them hyper-computers.

The key property of general purpose computers is that they are general purpose. We can use them to deterministically model any physical system, of which they are not themselves a part, to an arbitrary degree of accuracy. Their logical limits arise when we try to get them to model a part of the world which includes themselves.

Bibliography

- [1] C.J. Bashe, L.R. Johnson, J.H. Palmer, and E.W. Pugh. *IBM's early computers*. MIT Press Cambridge, MA, USA, 1986.
- [2] EJ Beggs and JV Tucker. Embedding infinitely parallel computation in Newtonian kinematics. *Applied mathematics and computation*, 178(1):25–43, 2006.
- [3] G. D. Bergman. A New Electronic Analogue Storage Device. In *Proceedings of the International Analogy Computation Meeting*, pages 92–95, Brussels, 1955. Presses Academiques Europeennes.
- [4] E. Blakey. Computational Complexity in Non-Turing Models of Computation. *Electronic Notes in Theoretical Computer Science*.
- [5] E. Blakey. A Model-Independent Theory of Computational Complexity. *Price: From Patience to Precision (and Beyond)*, 2008.
- [6] E. Blakey. Factorizing RSA Keys, an Improved Analogue Solution. *New Generation Computing*, 27(2):159–176, 2009.
- [7] L. Boltzmann. *Lectures on gas theory*. Dover Pubns, 1995.
- [8] Olivier Bournez and Michesl Cosnard. On the computational power and super-turing capabilities of dynamical systems. Technical Report 95-30, Ecole Normal Superior de Lyons, 1995.
- [9] J. Brooks. *Dreadnought Gunnery and the Battle of Jutland: The Question of Fire Control*. RoutledgeFalmer, 2005.
- [10] L.A. Bunimovich and M.A. Khabystova. Lorentz lattice gases and many-dimensional Turing machines. *Collision-Based Computing*, pages 443–468.
- [11] J.N. Burghartz, K.A. Jenkins, and M. Soyuer. Multilevel-spiral inductors using VLSI interconnect technology. *IEEE Electron device letters*, 17(9):428–430, 1996.
- [12] P.E. Ceruzzi. *A history of modern computing*. The MIT press, 2003.

- [13] N. Chomsky. Three models for the description of language. *IRE Trans. on Information Theory*, 2(3):113–124, 1956.
- [14] C. Cohen-Tannoudji, B. Diu, F. Laloë, and S.R. Hemley. *Quantum mechanics.: Vol. 1.* Wiley Interscience;, 1977.
- [15] J. Copeland. Accelerated Turing Machines. *Minds and Machines*, 12:281–301, 2002.
- [16] S.R. Cray Jr. Immersion cooled high density electronic assembly, May 20 1986. US Patent 4,590,538.
- [17] WD Crozier and W. Hume. High-Velocity, Light-Gas Gun. *Journal of Applied Physics*, 28(8):892–894, 1957.
- [18] NCA da Costa and FA Doria. How to build a hypercomputer. *Applied Mathematics and Computation*, 215(4):1361–1367, 2009.
- [19] M. Davis. Hilbert’s tenth problem is unsolvable. *The American Mathematical Monthly*, 80(3):233–269, 1973.
- [20] T. Dawson. Range Keeper For Guns, November 30 1909. US Patent 941,626.
- [21] D. de Solla Price. An ancient Greek computer. *Scientific American*, 201:60–67, 1959.
- [22] D. de Solla Price. Gears from the Greeks. The Antikythera Mechanism: A Calendar Computer from ca. 80 BC. *Transactions of the American Philosophical Society*, 64(7):1–70, 1974.
- [23] Bernard d’Espagnat. *Conceptual Foundations of Quantum Mechanics.* Addison-Wesley, 1976.
- [24] J.S. Dumaresq. A Rate of Change of Range and Deflection Finder to Facilitate the Accurate Shooting of Guns. 1905. GB patent GBD190417719 19040815.
- [25] P.D. Dunn and D.A. Reay. The heat pipe. *Physics in Technology*, 4:187–201, 1973.
- [26] A. Einstein and T. into English. Concerning an heuristic point of view toward the emission and transformation of light. *American Journal of Physics*, 33(5):367, 1965.
- [27] R.K. Englund. *The World’s Writing Systems*, chapter The proto-Elamite script, pages 160–164. 1996.
- [28] G. Etesi and I. Németi. Non-Turing Computations Via Malament–Hogarth Space-Times. *International Journal of Theoretical Physics*, 41(2):341–370, 2002.

- [29] R.P. Feynman. Quantum mechanical computers. *Foundations of Physics*, 16(6):507–531, 1986.
- [30] R. P. Feynmann. Simulating physics with computers. In A. Hey, editor, *Feynmann and computation: exploring the limits of computers*, pages 133–153. Perseus Books, Cambridge, MA, USA, 1999.
- [31] E. Fredkin and T. Toffoli. Conservative logic. *International Journal of Theoretical Physics*, 21(3):219–253, 1982.
- [32] T. Freeth, Y. Bitsakis, X. Moussas, JH Seiradakis, A. Tselikas, H. Mangou, M. Zafeiropoulou, R. Hadland, D. Bate, A. Ramsey, et al. Decoding the ancient Greek astronomical calculator known as the Antikythera Mechanism. *NATURE-LONDON*-, 444(7119):587, 2006.
- [33] P. Gannon. *Colossus: Bletchley Park's Greatest Secret*. Atlantic Books, London, 2006.
- [34] J. Gea-Banacloche. Minimum energy requirements for quantum computation. *Physical review letters*, 89(21):217901, 2002.
- [35] J. Gea-Banacloche and L.B. Kish. Comparison of energy requirements for classical and quantum information processing. *Fluct. Noise Lett*, 3, 2003.
- [36] P. Gordon. Numerical cognition without words: Evidence from Amazonia. *Science*, 306(5695):496, 2004.
- [37] J.D. Hamkins. Infinite Time Turing Machines. *Minds and Machines*, 12(4):521–539, 2002.
- [38] RW Hamming. The unreasonable effectiveness of mathematics. *American Mathematical Monthly*, pages 81–90, 1980.
- [39] DR Hartree. The mechanical integration of differential equations. *The Mathematical Gazette*, 22(251):342–364, 1938.
- [40] S.W. Hawking. Black hole explosions. *Nature*, 248(5443):30–31, 1974.
- [41] Andrew Hodges. *Alan Turing the enigma*. Touchstone, 1983.
- [42] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages*. Addison Wesley, 1979.
- [43] G. Ifrah. *Histoire universelle des chiffres: l'intelligence des hommes racontée par les nombres et le calcul*. R. Laffont, 1995.
- [44] LB Kish. Moore's law and the energy requirement of computing versus performance. *IEEE PROCEEDINGS CIRCUITS DEVICES AND SYSTEMS*, 151(2):190–194, 2004.
- [45] F.W. Kistermann. Blaise Pascal's adding machine: new findings and conclusions. *IEEE Annals of the History of Computing*, 20(1):69–76, 1998.

- [46] T.F. Knight Jr and S.G. Younis. Asymptotically zero energy computing using split-level charge recovery logic. Technical report, 1994.
- [47] P. Koiran and C. Moore. Closed-form analytic maps in one and two dimensions can simulate Turing machines. *Theoretical Computer Science*, 210(1):217–223, 1999.
- [48] G. Lakoff and R. Nunez. *Where mathematics comes from: How the embodied mind brings mathematics into being*. Basic Books, 2001.
- [49] R. Landauer. The physical nature of information. *Physics Letters A*, 217(4-5):188–193, 1996.
- [50] Rolf Landauer. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, 5:183–91, 1961.
- [51] Rolf Landauer. Information is physical. *Physics Today*, pages 23–29, May 1991.
- [52] Rolf Landauer. Information is inevitably physical. In Antony Hey, editor, *Feynmann and Computing*. Westview Press, Oxford, 2002.
- [53] S. Lavington. *A history of Manchester computers*. NCC Publications, 1975.
- [54] S. H. Lavington. The Manchester Mark I and atlas: a historical perspective. *Commun. ACM*, 21(1):4–12, 1978.
- [55] S.H. Lavington. *Early British computers: the story of vintage computers and the people who built them*. Manchester University Press, 1980.
- [56] Donald MacKay and Michael Fisher. *Analogue Computing at Ultra High Speed*. Chapman and Hall, London, 1962.
- [57] G. Mandler and B.J.mandler Shebo. Subitizing: An analysis of its component processes. *Journal of Experimental Psychology: General*, 111(1):1–22, 1982.
- [58] K. . Menninger. *Number words and number symbols: A cultural history of numbers*. Dover Pubns, 1992.
- [59] E.F. Moore. Gedanken-experiments on sequential machines. *Automata studies*, 34:129–153, 1956.
- [60] O. Neugebauer. Apollonius’ Planetary Theory. *Communications on Pure and Applied Mathematics*, 8(4), 1955.
- [61] H.J. Nissen, P. Damerow, and R.K. Englund. *Archaic bookkeeping: early writing and techniques of economic administration in the ancient Near East*, page 11..12. University of Chicago Press, 1993.
- [62] R. Omnes. *The interpretation of quantum mechanics*. Princeton University Press, NJ, 1994.

- [63] M.R. Polcari. Collaboration: The Semiconductor Industry's Path to Survival and Growth. In *AIP Conference Proceedings*, volume 788, page 3. IOP INSTITUTE OF PHYSICS PUBLISHING LTD, 2005.
- [64] RJD Power and HC Longuet-Higgins. Learning to Count: A Computational Model of Language Acquisition. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 200(1141):391, 1978.
- [65] V. Pratt. *Thinking machines: the evolution of artificial intelligence*. Blackwell Publishers, 1987.
- [66] Emerson W. Pugh, Lyle R. Johnson, and John H. Palmer. *IBM's 360 and Early 370 Systems*. MIT, 1991.
- [67] SC Rashleigh and RA Marshall. Electromagnetic acceleration of macroparticles to high velocities. *Journal of Applied Physics*, 49(4):2540–2542, 1978.
- [68] F. Scardigli. Generalised Uncertainty Principle in Quantum Gravity from Micro-Black Hole Gedanken Experiment. *Physics Letters B*, 452:39–44, 1999.
- [69] T.O.E.C. SCHENCK. MANUFACTURE OF GUNPOWDER, February 27 1883. US Patent 273,209.
- [70] C. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423 and 623–56, 1948.
- [71] W.D. Smith. Church's thesis meets the N-body problem. *Applied mathematics and computation*, 178(1):154–183, 2006.
- [72] D. Spinellis. The Antikythera Mechanism: A Computer Science Perspective. *Computer*, pages 22–27, 2008.
- [73] S. Suzuki and T. Matsuzawa. Choice between two discrimination tasks in chimpanzees (Pan troglodytes). *Japanese Psychological Research*, 39(3):226–235, 1997.
- [74] R.K.G. Temple. *The Crystal Sun: Rediscovering a lost technology of the ancient world*. Century, 2000.
- [75] W. Thomson. Mechanical Integration of the Linear Differential Equations of the Second Order with Variable Coefficients. *Proceedings of the Royal Society of London*, pages 269–271, 1875.
- [76] W. Thomson. On an Instrument for Calculating $(\int \varphi(x)\psi(x)dx)$, the Integral of the Product of Two Given Functions. *Proceedings of the royal society of London*, pages 266–268, 1875.
- [77] A. Turing. On Computable Numbers, With an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–65, 1937.

- [78] A. Turing. Computing Machinery and Intelligence. *Mind*, (49):433–460, 1950.
- [79] A. Turing. Lecture on the Automatic Computing Engine, 1947 . In B. J. Copeland, editor, *The Essential Turing*. OUP, 2004.
- [80] C. A. A. Wass. *Introduction to Electronic Analogue Computers*. Pergamon, 1955.
- [81] E. Wigner. The unreasonable effectiveness of mathematics in the natural sciences. *Communications in Pure and Applied Mathematics*, 13(1):1–14, 1960.
- [82] F. C. Williams. A Cathode Ray Tube Digit Store. *Proceedings of the Royal Society of London*, 195A:279–284, 1948.
- [83] JO Wisdom. Berkeley’s Criticism of the Infinitesimal. *The British Journal for the Philosophy of Science*, 4(13):22–25, 1953.
- [84] Stephen Wolfram. *A New Kind of Science*. Wolfram Media, Inc., Illinois, 2002.
- [85] SG Younis. *Asymptotically Zero Energy Computing Split-Level Charge Recovery Logic*. PhD thesis, Ph. D. Dissertation, MIT, Cambridge, MA, 1994.