Modelling Babbage's Difference Engine

0215594

Abstract

Charles Babbage's Difference Engine is commonly considered to be the world's first computer. As such it is an important part of any Computing teaching. Though much literature exists on Babbage, the construction of the engine and the algorithm it computes, little has been written about the actual operation of the machine. Constructionism is well known in the field of teaching and learning. This paper will attempt to identify the relative merits of constructionism in creating a model of the Difference Engine for educational purposes, describe the model and process and discuss this approach with regards to its relationship to Empirical Modelling.

1 Introduction

1.1 Historical Context

Before the invention of calculating engines, indeed up until the 1940s, calculations involving complex functions, such as trigonometric calculations, could only be done with the aid of a set of tables, which had been pre-computed by hand. This laborious task was often fraud with calculation errors. Such errors had grave consequences, such as the sinking of ships because of subsequent navigation errors. Babbage himself proclaimed in 1821: "I wish to God these calculations had been executed by steam" (Swade, 2000).

The first mention of a calculation machine was made by a German engineer called Johann Helfrich Müller in 1784 (Swade, 2000) in a letter to a colleague[1].

In 1812 Charles Babbage conceived the idea of the Difference Engine: "[a machine that uses] the fact that the n^{th} order differences of a polynomial of degree n are constant in order to calculate successive values [of the polynomial]" (d'Ocagne, 1986). Babbage never completed the Difference Engine, due to a number of intervening factors, his temper being not a small such factor.

1.2 Motivation

One of the main aspects of Empirical Modelling is teaching and educational technology (Beynon, 1997). Whilst researching for this paper, it became clear that to this date no physical models exist, which can be used in the teaching of either the history of computing or mathematics; the few models that do exist, such as the one built for the Science Museum in London are not available for demonstration to the public. It is difficult to see them in action and to get a first-hand, detailed account of how they operate. To the average museum goer this is not of great concern, however, considering that Charles Babbage's Difference Engine is by some regarded to be the first "proper computer", it seems important that students, and also teachers, understand the principal workings of the machine. Currently the teaching extends only as far as explaining the method of finite differences that the Difference Engine uses to do its computations. However, the *actual* workings, the internal calculations, are, if at all, only glossed over.

1.3 The Method of Finite Differences

This section will serve as a quick explanation of the idea behind the Difference Engine [2]. Many calculations for tables, such as trigonometric functions can be approximated to a polynomial equation in one variable, of the form:

$$f(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_2 x^2 + ax + a$$

These polynomials can then be computed by substituting a value for x. The idea behind the Method of Finite Differences is that when the difference between two successive values of f(x) is taken, and then the difference between these differences is taken at some point the difference is constant. For an n^{th} order polynomial this is the n^{th} difference. As an example: let $f(x)=x^2$:

[1] For the text of the letter see Klipstein, P.E., "*Beschreibung einer neu erfundenen Rechenmaschine*", Frankfurt, 1786

[2] For more information of the mechanical operation of the Difference Engine: http://www.satyam.com.ar/Babbage/en/ (last accessed 18/01/06)

Table 1 – First and Second Differences for $f(x) = x^2$

x	f(x)	1st difference	2nd difference
1	1	3	2
2	4	5	2
3	9	7	2
4	16	9	2
5	25	11	2
6	36	13	2
7	49	15	2
8	64	17	2
9	81	19	*2*
10	100	*21*	

Given this table, it is possible to work out the value for when x = 11. Since the second difference is constant at two, we can assume that the next value after 19 is 19+2=21. From that we can calculate that 11^2 is 121 and any calculator will verify this. For very long high-degree polynomials it was often easier to calculate successive results via this method, which often ran to 6 or 7 differences. As Babbage himself said: "All tables can be computed by differences." (Babbage, 1994)

This process was laborious, however, and a single slight miscalculation would carry forward to the rest of the table, rendering it useless. This is the reason why Babbage thought of building a machine to automate these calculations.

2 Previous Models

Babbage invented the Difference Engine over 150 years ago. Since then attempts have been made at actually building it, with varying degrees of success. Discussed here are three versions which were completed and adhere more or less closely to Babbage's design.

2.1 Scheutz Difference Engine

The two Swedes George and Edvard Scheutz, father and son, were engineers and built one of the first versions of the Difference Engine. It is often claimed that they were unaware of Babbage's design[3] and invented their own Difference Engine (d'Ocagne, 1986). They felt, however, that the machine was too large to be completed successfully quickly. Thus, they built a smaller scale version, which held Babbage's principle in mind, but deviated quite a lot from his designs.

[3] As it says in a foot note, the Scheutzes were probably inspired by an untitled article about Babbage's Difference Engine. See Lardner, D., Edinburgh Review, July 1834, LIX(CXX):263-327

2.2 Robinson's Meccano Engine

Tim Robinson's Difference Engine is probably one of the most curious. It is built entirely out of standard Meccano parts[4]. In some way it is ironic, that Babbage never built his own Difference Engine because of the lack of precision engineering and mass produced gears (and other mechanical parts). Whereas now, in principle, anyone with the right Meccano set can reproduce Babbage's machine.

2.3 The Science Museum Difference Engine

This Difference Engine, which is on permanent display at the London Science Museum was begun in 1985 and completed its first successful, error-free test run a month before Charles Babbage's bicentennial in November 1991.[5]

Though the Science Museum's version of Babbage's Difference Engine N° 2 deviated slightly from Babbage's original plans, these changes had been made so they were reversible and only implemented to facilitate operating the machine[6]. Since April, the 4,000 part printing mechanism is also in operation[7].

2.4 Software Models

Ed Thalen has created a version of the Difference Engine as a Java Applet [8]. This applet only describes the functioning of the engine in terms of outputs of columns, but does recall the Difference Engines ability to handle the sine function by approximating it to a polynomial of degree 7. He also provides a description of tools that may be used to implement a version of the Difference Engine.

3 Constructionism

3.1 The complexity of modelling

After reviewing the available material on the Difference Engine, it became clear very soon that any modelling done would have to be very basic. Not only does the author lack the specific mechanical engineering knowledge in order to design and construct the various elements, but also with very complicated and closely timed operations (such as for the carry) it was unfeasible to produce an actual rendition of the Difference Engine as software.

[4] http://www.meccano.us/difference_engines/rde_1/index.html (last accessed 18/01/06)

[5] http://www.sciencemuseum.org.uk/on-line/babbage/index.asp (last accessed 18/01/06)

[6] Described by Reg Crick (5th paragraph from the bottom) http://www.ftldesign.com/Babbage/ (last accessed 18/01/06)

[7] http://news.bbc.co.uk/1/hi/sci/tech/710950.stm (last accessed 18/01/06)

[8] http://ed-thelen.org/bab/bab-diff-JavaScript.html (last accessed 18/01/06)

However, this was not a deterrent from creating the Difference Engine, but rather forced a new view of how a model of the Difference Engine could be constructed empirically.

3.2 What is constructionism?

In its simplest form, constructionism is "learn by making" (Papert, 1991). By constructing artefacts to model the real world a deeper understanding of the workings of both the model and the real-world are fostered.

We perceive the *real* world artefacts through what in computing terms would be called its *output*, which in Empirical Modelling is called an *observable*. Indeed, the only way of knowing that some action had taken place inside the artefact is by what we can observe from the outside. Of course these observables generally do not act independently, but rather are connected by some *dependency*, which can either exist between observables in the artefact or a wider *system* of artefacts.

When we examine an artefact, we detect the changes in the observables; we can, given that we know the inputs then construe the dependencies between the observables. Doing this empirically, i.e. not from formal theory but from one's own observations enables one to construct a mental model and hence with the right tools, a physical (software) model of the artefact.

This resulting model may or may not have close resemblance to the inner workings of the actual artefact that is being modelled. The important part of constructionism is that by creating a model of the artefact a deeper understanding of the artefact is fostered. The fact that the model may have little to no resemblance to the real world *referent* is of little concern. In the end, the observables behave the same way and give the same values as the referent. Take a computer program as an example. It is not necessary to know in which programming language it was coded, nor whether object oriented practices were used, nor is it important whether the variables were named in an understandable fashion, nor whether the program was well commented. If all we have is the final, compiled program, we can recreate it, without having to know what it looks like on the inside. One could argue that such a program behaves like Schrödinger's cat. We don't know whether the model is the same as the program, until we can take a look at the program or the model behaves differently to the program.

3.3 Constructionism in Empirical Modelling

Constructionism and Empirical Modelling are much related fields. Both place an emphasis on artefacts as well as modelling in order to understand the artefact itself. While Constructionism approaches modelling from an educational angle, where the process of learning about the domain of an artefact is closely interwoven with the creation of a model to simulate that domain, Empirical Modelling focuses more on the philosophical aspects and the link between observations and the inner workings of artefacts. In (Beynon, 2004) the term *construal* is used to describe "a computer based artefact for active learning" it then goes on to say that "a construal is typically much more primitive than a program It is built with a referent in mind." This referent, of course, is the original real-world artefact. This argument begins to outline the relationship between Constructionism and Empirical Modelling. Empirical Modelling gives Constructionism the tools and a theoretical framework, whilst Constructionism gives Empirical Modelling a real-world application, a way to test theories.

Furthermore, in many ways, developing a model in EM *is* constructionism. Modelling begins with a basic model and then builds (or constructs?) on it. Modelling, unlike developing "classical" computer programs, is an iterative process, which begins with a crude model that is then refined as more observables and dependencies are added. As such, a model can also never be complete, since no model can ever recreate the real world to 100%. However, Empirical Modelling supplies the correct tools for this iterative process, as described in the next section.

3.4 Tools

3.4.1 Definitive Notation

When modelling dependencies in a constructionist manner, it is important that all or at least many parts of the model can be changed. This is not just in order to allow the model to be extended at will, or to be adapted to a previously unseen pattern in the observables, but also so that a variety of *what if* scenarios can be constructed.

The ability to change the values of observables or dependencies is important in modelling. When wanting to investigate further the possible effects that a change in certain dependencies would have, or if the starting conditions changed, one *has* to turn to a model. The real world conditions of the artefact cannot be changed, so in turn one must look to a model.

3.4.2 EDEN

Conventional programming languages have a flaw, which renders them less effective when trying to study the effects of changing observables in a model: With every change they need to be recompiled. It can be argued that one can expose the variables via textboxes and buttons to the user, and while that may be true, it can only expose those parts of the model that the modeller has thought about exposing. In other words, only those

variables that can be *foreseen* to be useful are exposed.
With a modelling tool, like tkeden[9], it is far simpler to build and interact with a model. Observables can be changed while the model is running, giving real-time feedback about the change in the model.
The EDEN suite of tools also includes support for a graphical representation of observables. This is particularly useful where a model is very large and potential users need a starting point to investigate the model. Furthermore, since modelling tries to mimic the real world, and real world artefacts usually have some sort of *visual* output, a model ought to as well.

3.4.3 The Downside

There are of course downsides to using these modelling tools. EDEN does not make programming knowledge obsolete. The necessity for having a modifiable l-value in order to assign to a variable is still present. As are programming constructs, such as for loops and if statements. Though these are of course necessary, they do limit the audience of the tool to those who already know how to program.
Thus, if it hadn't been for the author's previous knowledge of programming, EDEB would not have been a very useful tool.
Using the graphical tools, such as DoNaLD and Scout constrains the interaction with the model in a similar way that compiled programs do. Also, when seeing the final model, a user might not even think about the observables and dependencies, which are now hidden behind the graphical user interface.

4 The Model

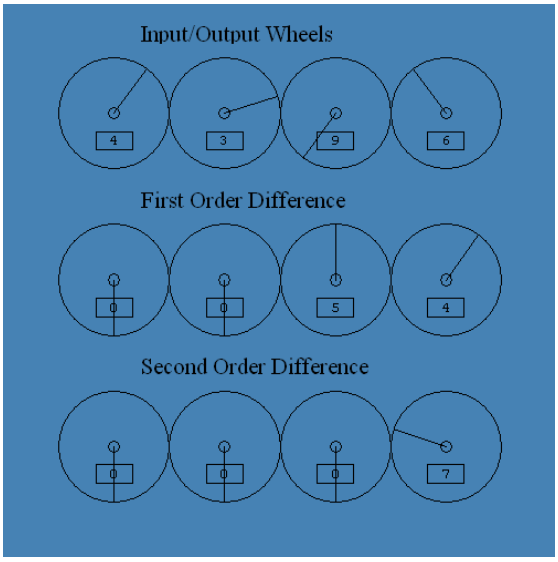

Figure 1 – The digit wheels in operation

[9] More information about tkeden and all its associated notations can be viewed here and a copy downloaded: http://www2.warwick.ac.uk/fac/sci/dcs/research/em/notations/intro/ (last accessed 18/01/06)

4.1 Initial Problems

With close to 4,000 gears, shafts, sprockets and other assorted parts, the Difference Engine was difficult to engineer in the 1830s due to the lack of precision engineering (Swade, 2000).
Fortunately, this is not much of a problem when modelling a machine such as the Difference Engine in software. The main problem with parts such as gears is that they must be modelled before they can be used. This, as it turned out, was a much greater problem than originally perceived. Attempts at making variable tooth gears in tkeden's 3D graphics tool, Sasami, failed. Thus with no way of creating all the necessary components in 3D, this course had to be abandoned. 2D construction of the Difference Engine was also considered and then discarded. A 3D machine simply cannot be modelled correctly and understandably in just two dimensions.
Thus the constructionist idea was taken up.

4.2 Outline of the Model

The model of the Difference Engine can compute 4 digit numbers to second order differences. The digits are arranged in wheels, running horizontally. The right-most wheel in each row represents the units, the next left wheel the tens and so on. Thus, the third wheel from the right actually represents the hundreds. Vertically arranged are from top to bottom the input/output wheels, first order difference wheels and then the second order difference wheels.
The wheels are represented with a window to read of the value that the wheel is currently turned to and a line indicating where on the wheel zero is located. This gives a quick visual clue to how far the wheel has turned and will be further explained in the animation section.
To the right of the wheels are the input textboxes. By entering a number here in the conventional form (i.e. 1234) the wheels for each particular row can be set. If a number larger than the number of digit wheels is entered, any digits greater than 10^3 are ignored. The wheels cannot be set via these textboxes once the calculation mechanism has been set into operation, however, the underlying observables can of course still be altered.

4.3 Operation

A button at the bottom of the screen starts the calculation. Babbage's machine calculated one result every four cranks of its main drive shaft. This button simulates four such cranks. While in operation, the button goes black, once the result appears on the input/output wheels, the button resets to its default colour.
The calculation begins by counting down the second order difference onto the first order

difference; this happens even when the second order difference is 0, in which case the first order difference remains unchanged. The wheels on the second order difference wheels will count down to zero whilst simultaneously the wheels on the first order difference will count upwards by the same amount. Once this is done the first order difference wheels will repeat the same process onto the input/output wheels. Once this has been completed, the first order and second order difference wheels will reset to the previous value they had. The model is then ready to calculate the next result. Initially the speed of the wheels turning is set to 1000 milliseconds. This allows the operation of the calculation steps to be observed more easily. In order to get a feel for a more realistic speed it should be set to 100 milliseconds.

4.4 Simplicity

Babbage himself was very concerned not only with making the Difference Engine, but also with its use. In his autobiography he states that the Difference Engine "would be of comparatively little value, unless it were easily set to do its work, and […] executed not only accurately, but with great rapidity." [BC31].

It seems though, that the machine was not at all "easily set to do its work". The technical manual prepared by Reg Crick and Barrie Holloway[10] on the setting of the Difference Engine built for the Science Museum in London, describes the process in a series of steps. This includes setting the odd and even columns separately as well as several locks that need to be disengaged and re-engaged at the right time. This is certainly not as simple as Babbage leads us to believe.

Furthermore, the digits on the Difference Engine read from bottom to top in order of increasing powers of ten. This can be confusing when attempting to read a value of the wheels. One would expect the digits to be aligned in the same way that one normally reads them, i.e. horizontally. For the Difference Engine after Babbage's design, this was not much of an issue, since Babbage envisioned from the start to have a printing apparatus attached to the output column. From that point of view, the output column never needed to be read, the results would be printed out, organised into tables. Babbage considered this to be very important "as there was the possibility of errors by the operators." (Nyman, 1982)

Both of these issues we simplified in the model. Since the real physical processes of the Difference Engine aren't modelled one-to-one, liberties were taken with the setting of the machine as well. Textboxes are used to set the wheels to the correct positions. Also the digit wheels are aligned horizontally to facilitate reading off numbers. Since Eden does not contain any support for printing, especially line-by-line printing, this part of the Difference Engine was not implemented at all.

4.5 The Modelling Process

4.5.1 The Carry mechanism

Though Babbage discovered himself whilst investigating the feasibility and design of the Difference Engine, "an intractable problem was always the 'carry system'." [NA48] This part of the model needed a great deal of attention. On the one hand, it was important that the model replicated the inner-workings of the Difference Engine; on the other hand, it would not have made sense to try and model the idea of an arm swinging around and knocking the next gear one tooth further.

Thus, the carry mechanism functions like this: When a digit wheel changes, it checks whether the number it turns to is greater than ten. If that is the case, a carry is generated. When a carry is generated, the carry indicator is set to one. This change is registered and the next wheel is incremented. Then the carry indicator is reset to zero. This intermediary step seemed to follow the idea behind the Difference Engine's carry mechanism more closely than if it had been omitted.

4.5.2 Animation

Anyone who has seen the Science Museum's Replica of the Difference Engine cannot help but be awed by its rattling and stomping, the whirring of gears and the clatter of the crank shaft. It was obvious right away that the visual aspect of the Difference Engine was important and that it would have to feature in the final model in some way.

The wheels are represented by a circle. Since a circle alone would not be able to convey the idea of movement, a line was added, which points to the zeroth position on the disk. The position of this line is directly linked to the value of the digit that it represents, so that if the value changes (whether it is within the model's code or from user intervention) the facing of the line is automatically updated. Whilst the calculation is running, the values are constantly updated and the wheels appear to be spinning. This, it was felt give more realism to the working of the model.

Also, for someone who is not familiar with the theory behind Babbage's Difference Engine it is easy to see what processes are going on inside the model.

When the model is set to the right clock speed (about 100 milliseconds) the wheels of the model spin fast enough to simulate actual movement. The clock can of course be slowed down to watch the process at a slower pace und thus understand it more easily.

[10] A copy can be found here: http://ed-thelen.org/bab/bab_inst.html (last accessed 18/01/06)

4.6 Extendibility

When the model was constructed, it was done in a very exploratory manner. First two wheels were constructed and a carry mechanism between was implemented. Content that this approach would work, the input/output wheels were extended to four digits. After the first order difference was added, the model was, in principal complete. By adding a second difference to the model, it was proved that the model can be extended fairly easily (certainly more easily than the original Difference Engine). By copying the definitions file for the first order wheels and replacing the prefixes to variables and functions (a simple find and replace in most text editors) and editing some settings for the animation, another difference could be added in a matter of minutes. Likewise the number of digits could be extended fairly easily.

One major advantage over this model to comparable software models is that there is no upper limit on the number of digits in the computation, other than memory and screen size limitations. Each digit is stored in an individual variable. EDEN does not have a "byte" data type (in fact a nybble would be sufficient) and therefore the digits had to be stored as integers.

4.7 Limitations

The main limitation of the model is the fact that it does not closely follow the engineering plans that Babbage drew up. It cannot convey the full extend of Babbage's Difference Engine. Though it recreates the working, theoretical aspect of the machine and through the use of animation gives some idea of the visual aspect of the engine, it does not fully model the Difference Engine. Though a user might learn the working principle of the machine, inner operation of the machine is by far different.

5 Conclusion

Constructing the model has indeed been an iterative process, starting with two wheels and a carry mechanism between them; moving all the way on to being able to extend Babbage's Difference Engine far beyond the capabilities that Babbage himself had dreamt of.

It is, of course, clear that the inner workings of the machine and the model are vastly different. The former is purely mechanical, whilst the latter is pays very little tribute to the engineering principles present in the Difference Engine.

The model, however, does mimic the function of the Difference Engine. Given the same problem and with the ability to synchronize the first step (the setting of the wheels), the model and the engine will both give the same answer every time[11]. In light of this, the model can be declared to be successful and to have demonstrated the principles of constructionism in action.

Furthermore, the close working relationship between Constructionism and Empirical Modelling was demonstrated, by creating the model in the definitive notation, EDEN.

The main question that remains at the end is in what way would the experience of making the model (and possibly even the experience of using the model) be enhanced, if the model were to be made to exact 3D specification of Babbage's plans and furthermore, in what way this experience would differ, would a traditional programming language be used.

References

C. Babbage, "Passages from the life of a Philosopher", Pickering & Chatto (Publishers) Limited, 1994

W.M. Beynon, C. Roe., "Computer support for constructionism in context", In Proc. of ICALT'04, Joensuu, Finland, August 2004, 216-220.

M. Beynon,, "Empirical Modelling for Educational Technology", In Proceedings of Cognitive Technology 1997, pages 54--68. University of Aizu, Japan, IEEE, 1997

M. d'Ocagne, "Le calcul simplifie", Charles Babbage Institute Reprint Series for the History of Computing, The MIT Press and Tomash Publishers, 1986

P. Morrison, E. Morrison, "Charles Babbage and his calculating engines", Dover Publications, 1961

A. Nyman, "Charles Babbage: Pioneer of the Computer", Oxford University Press, 1982

S. Papert, I. Harel, "Situating Constructionism", Ablex Publishing Corporation, 1991

D. Swade, "The Cogwheel Brain", Little, Brown and Company, 2000

[11] In fact, the Difference Engine would make calculation errors more often than the model, due to mechanical failures and the delicacy of the carry mechanism.