On chaotic pulsation in semiconductor laser diodes, the optical systems used in determining the occurrence of chaotic pulsation, and the application of this to the encryption of data signals

Luke Pomfrey[1], Mohammed Moussa[2], Alexander Shalashilin[2], Philip Smith[2], Wai Cheong Tou[2], and Gihan Weerasinghe[2]

[1] *University College London, Physics and Astronomy Dept., MSci Physics with Space Science*
[2] *University College London, Physics and Astronomy Dept., MSci Physics*

Monday 17th March, 2008

Contents

1 Introduction to the Project **3**
- 1.1 Executive Summary . . . 3
- 1.2 About the Group . . . 3
  - 1.2.1 Luke Pomfrey . . . 3
  - 1.2.2 Mohammed Moussa . . . 3
  - 1.2.3 Alexander Shalashilin . . . 4
  - 1.2.4 Philip Smith . . . 4
  - 1.2.5 Wai Cheong Tou . . . 4
  - 1.2.6 Gihan Weerasinghe . . . 4

2 Investigation report **5**
- 2.1 Introduction . . . 5
  - 2.1.1 Current methods of data encryption and the differences between hardware & software based encryption . . . 5
  - 2.1.2 The benefits of chaotic analogue encryption . . . 6
  - 2.1.3 Chaotic systems and the application of chaotic systems to data transfer . . . 7
  - 2.1.4 Feasibility of the proposed solution . . . 10
  - 2.1.5 How we will implement the proposed solution . . . 11
- 2.2 Technical information . . . 12
  - 2.2.1 The operation and theory behind the laser . . . 12
  - 2.2.2 The auto-correlator . . . 20
  - 2.2.3 Second harmonic generation . . . 23
- 2.3 The design and construction of the auto-correlator . . . 27
- 2.4 Computer modelling of the laser pulsation . . . 30
  - 2.4.1 Solving the rate equations . . . 34
  - 2.4.2 The *Mathematica* model . . . 35
- 2.5 Results from computer modelling . . . 39
  - 2.5.1 Comparison of the models . . . 39
  - 2.5.2 Mathematica output . . . 43

Appendix A: Computer modelling source code **47**
- Matlab . . . 49
- Mathematica . . . 53
- A *C++* Runge-Kutta method . . . 70

Bibliography **73**

Acknowledgements **74**

Index **75**
List of Figures 75
List of Tables 77
List of Attachments 78
Index 78

1 Introduction to the Project

1.1 Executive Summary

Self-pulsation is the term commonly used to describe the pulsed emission from semiconductor laser diodes operated under DC bias.[1] Self-pulsating lasers have the advantage that their short coherence length, due to the pulsation, makes them less sensitive to optical feedback and gives them a low relative noise intensity[1, 2]. These features are important in the use of semi-conductor lasers in Compact Disc (CD) players and other optical storage systems.[1]

The behaviour of these lasers can be described using a rate equation model[1, 3]. It can be seen that, if a suitable sinusoidal electrical modulation is added to the rate equations the frequency of free-running self-pulsations will become mode-locked to the frequency of the modulation.[4] By moving the modulation frequency away from the natural resonant frequency of the self-pulsation then the two frequencies can be made to compete, resulting in chaotic self-pulsation.[1]

In this paper we investigate the effects of changing the electrical modulation frequency applied to the laser, and the impact it has on the chaotic pulsation of the laser. We also investigate methods of discerning whether the laser is in fact pulsating chaotically using an auto-correlator with a second harmonic generation crystal.

We will be looking at the application of the chaotic self-pulsation of the laser to forms of hardware encryption in network interfaces.

1.2 About the Group

1.2.1 Luke Pomfrey

Luke Pomfrey contributed to the programming of the laser models, and created the *Fortran* and *C++* models, which were later scrapped due to the restrictive amount of time that would have been needed to implement functionality in them that was already present in the *Mathematica* and *Matlab* environments.

As the group secretary and resident TEXnician he was responsible for authoring agenda and minutes for the meetings, as well as presentations and this document. He authored the content of the introductory sections on encryption here (2.1.1 and 2.1.2) and co-authored the section on the operational theory of the lasers used (2.2.1).

1.2.2 Mohammed Moussa

Mohammed Moussa worked on implementing the computer simulation of the laser, more specifically the Matlab version. He researched mathematical techniques for solving coupled differential equations before writing a fourth-order Runge-Kutta algorithm implementation in Matlab. He cooperated with G. Weerasinghe, who worked on the Mathematica model, to analyze and compare both their models.

M. Moussa contributed to sections ?? and 2.5 of this document.

1.2.3 Alexander Shalashilin

Alex Shalashilin, having previously worked with the demonstrator Stephen Lynch in a similar optical experiment, was to be responsible for the construction of the apparatus. Having composed the equipment list and done research on the required components, he passed this to Canice Wai Tou and provided consultation while he drew the diagram, and afterwards assisted Wai Cheong Tou and Gihan Weerasinghe and to disassemble a CD-ROM to obtain the laser diode. Alex also helped to plan the presentation and composed the critical evaluation of the group.

Alex was the creator of section 2.3 of this document.

1.2.4 Philip Smith

Philip Smith carried out research into the second harmonic generation and operational theory of lasers. He authored section 2.2.3 and co-authored section 2.2.1.

1.2.5 Wai Cheong Tou

Wai Cheong Tou was chosen as the chair of the meetings, he organized locations for each board meeting, and was responsible for coordinating the flow during the meetings. He researched into the feasibility of using lasers driven into chaos as method of data encryption, and recorded how this solution was investigated by the group. He cooperated with A. Shalashilin in collating the equipment list required for building the intensity autocorrelator, and, with his graphical background, he produced the scaled graphical diagram of the autocorrelator(Fig. 2.16) which was used for reference and for the final presentation. Tou also worked together with A. Shalashilin to obtain the self-pulsating laser diode which is later used to build a simple model to demonstrate chaotic pulsation for presentation purpose by Tou and G. Weerasinghe.

W. Tou contributed to the report with sections 2.1.4 and 2.1.5.

1.2.6 Gihan Weerasinghe

Gihan Weerasinghe programmed and documented a simulation in Mathematica to model the self and chaotic pulsations of a semiconducting laser diode. He was also responsible for ensuring the final report met all the required criteria and prepared a basic structure to adhere to. In addition to this he contributed two additional documents for the report on chaotic systems and autocorrelators.

Gihan Weerasinghe contributed to sections 2.1.3, and 2.2.2 of this document.

2 Investigation report

2.1 Introduction

2.1.1 Current methods of data encryption and the differences between hardware & software based encryption

Author: L. Pomfrey

Current methods for encrypting network traffic

Current encryption methods for computer network traffic are generally digital encryption methods. These methods tend to be performed by software applications, this can be performed transparently to the user (as in TLS/SSL) or require user action (as in PGP/GPG). Applications of analogue network encryption can be found in analogue satellite and cable subscriber television systems, but these have largely been phased out with the advent of digital television. The prevalence in the mid- to late-1990's of pirate cable and satellite decoders was a testament to the insecurity of these systems.

In this section we will discuss the basics of two popular digital encryption systems performed in software;

1. Transport Layer Security (TLS) (and it's predecessor; Secure Sockets Layer (SSL))
2. *Pretty Good Privacy* (PGP) (and it's Free/Libre Open Source Software (FLOSS) implementation; GPG)

Many articles on encryption in computing can be found at [5] and in numerous textbooks and other references, as such, this will not be a comprehensive guide to current methods of encryption or indeed to the various methods of cryptography employed.

TLS/SSL: Probably the most well known (and, for all intensive purposes, the most widely used) methods of encryption for network traffic at the moment are the Transport Layer Security (TLS) protocol and it's predecessor, the Secure Sockets Layer (SSL) protocol. These are used heavily for the encryption of web (HTTP) traffic (*e.g.* on e-commerce and on-line banking websites) and for the security of e-mail authentication.

SSL and TLS are, for a substantial part of their mechanism, essentially the same protocol. TLS, however, fixes some of the inadequacies in the SSL protocol. They operate on a public/private key pair principle (known as asymmetric cryptography or public-key cryptography), thus, the security of the system is limited by the size of the keys involved and the specific algorithm used to generate the key pair. Short keys and insecure algorithms can significantly increase the possibility of brute force attacks on encrypted data being successful.

More information on TLS can be found at the Internet Engineering Task Force's TLS Status Pages.[6]

PGP/GPG: *Pretty Good Privacy* (PGP), and it's FLOSS implementation; *GNU Privacy Guard* (GPG), are applications providing both privacy and authentication for users. As with TLS/SSL, PGP/GPG is a form of public-key, or asymmetric, cryptography. As such, it is also vulnerable if a short key and/or insecure algorithm are used for key generation.

PGP/GPG can be used simply for authentication (*i.e.* ensuring that the message you have received is from who it purports to be from) by signing messages and files, or for full encryption of information.

Unlike TLS/SSL, PGP/GPG isn't transparent to the user. It generally requires the user to consciously sign/encrypt any files and/or messages to be sent over the network. This can cause a security risk if the user forgets to encrypt something before sending it.

More information on PGP/GPG can be found at the GnuPG website [7] and in the OpenPGP RFC specification [8].

The difference between hardware & software encryption

When attempting to encrypt network traffic there are two approaches that one can take, encrypting in hardware and encrypting in software. The examples of TLS/SSL and PGP/GPG given above are both software based encryption techniques.

Hardware and software encryption methods both have pros and cons compared to the other. Software based encryption is inherently digital in nature and thus generally can't take advantage of the same kind of chaotic processes that hardware based encryption can. Hardware encryption, however, requires additional hardware than that which is present in "stock" computers.

Taking the approach that encryption requires the generation of random numbers allows us to look at the mechanisms used in both hardware and software methods.

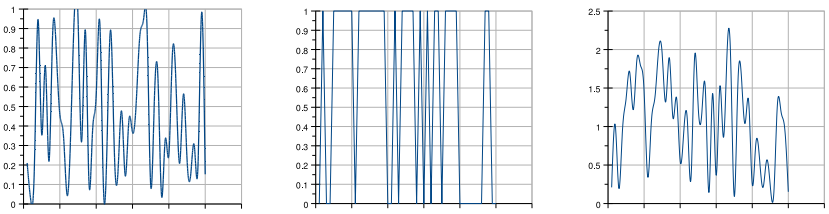
Hardware based random number generation generally utilises input from random processes such as radioactive decay, or thermal noise in semi-conductor components. These methods allow hardware based encryption to generate highly random numbers. The output from hardware based random numbers can then be digitised (which can reduce the randomness of the numbers) or left as an analogue signal, depending on the required usage. Whilst some methods of generating random numbers in hardware are, in a way, predictable they generally have a much higher degree of randomness than those generated in software.

Software methods of random number generation generally rely on the use of algorithms to create pseudorandom numbers. These are not strictly random numbers but are simply sequences of numbers that "appear" to be random. There are many methods for generating random numbers, for example on *GNU/Linux* and other *UNIX*-like operating systems the "special-file" `/dev/random` holds an "entropy pool" of environment data and hashes are used to generate pseudo-random numbers from this. The generation of pseudorandom numbers in software, and their limitations, is covered in-depth in [9].

2.1.2 The benefits of chaotic analogue encryption

Author: L. Pomfrey

The obvious benefit from analogue encryption is that unlike digital encryption,



(a) Noise carrier signal (b) Binary data signal (c) Data signal + Noise signal

Figure 2.1: A graphical example of the addition of a data signal to random noise showing the noise-like output waveform
Created by L. Pomfrey

where the values an encryption "key" can take are discreet in nature, the values a "key" can take are continuous in nature. This means there are in theory there are an unlimited number of encryption "keys" possible.

By using chaotic data as a carrier for an encrypted data stream (see section 2.1.3) we essentially bury the data in random noise, this means that not only will the data be encrypted incredibly securely (since, in order to decrypt the signal, the attacker will be required to create a chaotic signal identical to our carrier) but at a glance it will just look like white noise. Fig. 2.1 shows a graphical example of this, the random-looking nature of the chaotic carrier wave and the binary data signal combined is easily seen.

2.1.3 Chaotic systems and the application of chaotic systems to data transfer

Author: G. Weerasighe

The word "chaos" is defined as a system which exhibits "complete disorder", mathematically it is a feature of non-linear systems that are extremely sensitive to input conditions. It is usually associated with undesirable events, such as earthquakes, hurricanes and terrorist attacks. The chaos involved in these cases is largely a description of the consequences rather than the event itself. In each case, there is a momentary breakdown of civil law which leads to confusion and disorder or, in other words, the definition of chaos.

In physics, chaotic behaviour of a system has much in common with real world chaos, however, it is more appropriate to describe such a chaotic system as "a system which is completely unpredictable". It is hard to imagine any desirable application of chaos and, in most cases, chaos in any physical system is highly undesirable. However, the properties of chaotic systems happen to be ideally suited for one potential future application, and that is in secure data transmission.

When a DC bias current is applied to a semi-conducting laser diode, the result is self-pulsation.[1] The rate at which the laser pulses light is dependent on the applied current and the properties of the particular semi-conductor. The frequency of these pulsations is constant provided the current is DC. The frequency of the pulsations for a DC current is the natural fre-

quency of the laser. The behaviour of a laser diode changes if an alternating current is applied. If the alternating current is applied such that it's frequency is sufficiently different from the natural frequency of the laser, it is possible to make the laser diode pulse chaotically.[1]

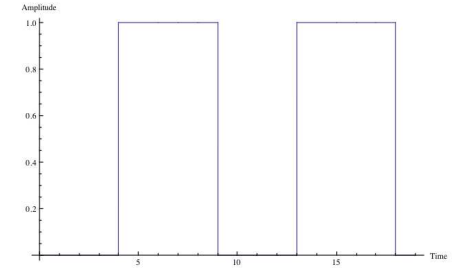
Chaotic synchronisation

A common way to transmit information is to use a carrier signal which is modulated by the data. Two methods are amplitude modulation and frequency modulation. In each case, either the amplitude or frequency of the carrier signal changes according to the data signal. At the receiving end, the carrier wave is extracted from the signal, leaving the data remaining. An example of a typical amplitude modulation is shown in Figure 2.2.

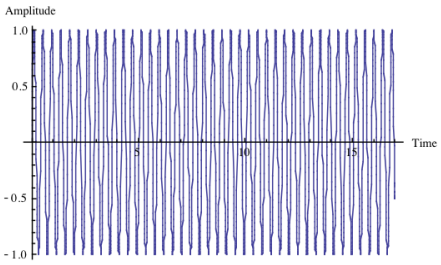
Chaotic carrier signals are potentially ideal for encrypting data. The chaotic signal is used, as before, as a carrier which is modulated by the data. However, since the carrier signal is completely unpredictable, it is impossible to intercept and extract the data, since the carrier signal cannot be distinguished from the modulated data by the interceptor. The data is therefore secured. Figure 2.1 illustrates data modulation using a chaotic carrier signal. It is, however, not immediately obvious how these chaotic pulsations could be feasibly used for data transmission. The transmission and receiving of data would require two diodes; one to transmit the data, and one to receive the data. The transmitting laser diode will transmit the chaotic carrier signal modulated by the data. At the receiving end, a photodiode will receive the same chaotic signal modulated by the data. However, the carrier signal cannot be removed since it again cannot be identified. The solution to this problem exploits a phenomena known as chaos synchronisation. If a second laser diode is introduced into the system at the receiving end and is driven by the chaotic output from the transmitting diode, then the transmitted chaos is perfectly replicated in the receiving diode. In other words, the chaos in both diodes synchronises. What is key to the data transmission is that only the chaotic signal from the transmitting laser diode is replicated in the second receiving laser diode.[10]

Since only the chaos is replicated at the receiving diode, it is possible to divide the chaotic signal out from the total signal at the receiving end which consists of the chaos and data product. Once this has been achieved, only the data remains at the receiving end, and thus the data has been successfully, and securely, transmitted. There are some conditions on the choice of semi conducting lasers for this "send and receive" construction, the most important of which is the similarity of the transmitting and receiving diodes. A small difference between the atomic properties of the transmitting and receiving diode (such as defects), would yield a large difference in the two different chaotic pulsations, in other words, it would not be possible to synchronise the chaos from the two lasers. For this reason, the two diodes need to be chosen from very close points on the wafer from which they are taken. In this way, the atomic properties of the diodes will be sufficiently similar. The similarity conditions of the diodes further secures the data transmission, since an interceptor would not be able to introduce an arbitrary laser diode into the transmission line, drive that diode into chaos, and essentially replicate what happens at the receiving end to cipher the data.

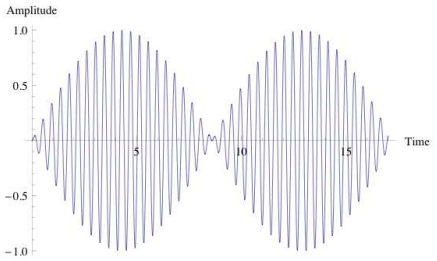
Other references: [11, 12, ?]



(a) Digital data



(b) Carrier signal



(c) Signal + data

Figure 2.2: An example of the use of typical amplitude modulation for data transfer. *Created by G. Weerasinghe.*

2.1.4 Feasibility of the proposed solution

Author: W. Tou

Unprotected data exchanged or stored at a company may reveal confidential information to unauthorized users, internal and external. Businesses need a way to exchange secure data with branch offices, vendors, and business partners to stay competitive and protected.

As previous chapters have mentioned, our team's proposed solution is to use chaotic optical pulses to encrypt signal for data protection, this has one big advantage, the encryption itself is done with an analogue system, most existing security systems today is done digitally, with digital signals, there can only be two kinds of output, 1 and 0, whereas for analogue, the signal is continuous, this makes it a lot harder to decipher thus providing more security.

Another advantage is the simplicity of the system, it doesn't require any particularly advanced equipment, a simple cheap CD laser diode1 from a compact disc player would be sufficient to produce the input signal, the only other piece of equipment in a production system would be a commonly available network interface card, this is discussed in the next paragraph.

In terms of price of production, how will we put our concept into practice outside the lab? The initial product cost to consumers would be high in order to cover the research and development costs (a look at Table 2.3 shows the price of the equipment needed for the initial research on the laser), a high retail price will directly affect sales and could mean that this product simply wouldn't sell. We can see that once in production the cost to the consumer would fall by looking at previous cases of technological advances. For example, the compact laser diode, which is widely used in a typical CD player today, cost scientists thousands of dollars to build, but after it became commercialized, and with engineers working and reducing its production cost, after a few years, with consideration of economies of scale, its' production cost has dropped to below a dollar each. From this we can predict that the same can be expected from our product.

Current tools for data encryption are mostly computer software, their price range from free to hundreds of pounds, their encryption key size are from 56, 128, 192, 256 bits etc. This method of data encryption is used on computers all around the world, the larger the bit size, the longer it will take for a computer to decode, the idea of encryption is to make it so complicated to decipher the data, that it can put off crackers from trying to steal the data.

This software method is effective, but it isn't unsolvable, as it is reliant on an encryption key, which only takes a matter of time to obtain, but for our system, there is simply no key or pattern to a chaotic signal.

Our method of encrypting is currently unsolvable, that's because this is a new idea, if this were to be released commercially, it will soon have people attempting to crack it; no encryption is ever going to be 100% secure. Let's assume that it is one day cracked, our whole system will all become scrap over-night, as it is not a scalable system, if it is a normal software encryption, they would simply increase its bit count, whereas for our hardware system, once they have crack it, would there be an existing solution to upgrade it? Or will It just become totally useless? In this

sense, software encryption would indeed have a much longer lifetime.

The big advantage of software encryption tools is that its reproduction cost is next to zero, as it is simply computer software which can be copied as many times as you like, and printing it onto a CD is an almost costless process when doing it with the right machinery.

Whereas for our encryption tool, it isn't software, instead it is hardware; it needs to be manually installed into an existing fiber optic cable system, which each set of tool will protect data transfer between two points, for example A and B. This can be a major disadvantage to our product, because in order to transfer data from either point A or B to a third party (e.g. point C). An additional set of tool will need to be installed in between A-C and B-C data point, this could be time consuming and would increase the total setup cost. Therefore this product could potentially put off companies which regular updates their network infrastructure. This will be a major factor when considering the cost of implementing the system, and should be considered at later stages.

As fore mentioned, our product is hardware, which will have an advantage over software as it doesn't use RAM or processor power on a computer, it will run faster and better than any software encryption tools, but the same goes to most hardware, it can age, and may will require regular maintenance and repairs.

All these factors affect whether this product is worth commercializing, and would definitely need to be analyzed before it leaves the lab.

2.1.5 How we will implement the proposed solution

Author: W. Tou

The investigation of the proposed solution of building an intensity autocorrelator to generate and measure chaotic optical pulses from compact disc laser diodes, first we looked at various research papers about auto-correlation to get a basic idea of how the laser will behave inside an auto-correlator, and the equipment we would need etc.

Next we split the task between computer modeling, research, equipment list and producing graphics.

Computer modelling

M. Moussa, L. Pomfrey, and G. Weerasinghe

In order to make prediction about the pulsing of the laser under autocorrelation, a computer programs is the ideal tool to test our theories, with the model, we will be able to make predictions on the behavior of the laser under autocorrelation, we can also use it to investigate how data is transmitted via optical data encryption.

The computer modeling was created using various programming languages: *MatLab*, *C++*, *Fortran*, and *Mathematica* (although the *Fortran* and *C++* models were later scrapped due to the amount of time that would be needed to implement the functionality lready found natively in *Fortran* and *Mathematica* in them). The reason why we used a number of languages to analyze this solution, is because we wish to analyze which programming method will be most suitable for our project.

Research

L. Pomfrey, P. Smith, W. Tou, and G. Weerasinghe

Continual research took place to collect information on chaotic pulsation, second harmonic generation and auto-correlation. The proposal was analyzed to examine the feasibility of the purposed solution, looking at whether this is practical economically and industrially.

Equipment research & pricing and construction planning

A. Shalashilin, and W. Tou

In preparation to build the system, the experimental setup was analyzed and compiled into an equipment check list for the auto-correlation experiment. Websites such as Thorlab and Newport were visited to collect information about our equipment. From this, a budget proposal was produced, and this leads to the detailed construction plan to show how the auto-correlator will be built.

To help explain the building process and for presentation of our project, a detailed diagram was required, experiment the basic setup of the autocorrelator was analyzed to produce a scaled diagram of the experiment, this diagram is mainly used for building, reference and for various meetings and presentations.

With all of the above tasks completed, we were able to procure a detailed plan showing how the autocorrelator will be built, predictions of our results is done with our computer simulations. For example, we can predict how a laser from a compact disc laser diode will behave under autocorrelation.

Next, we planned out our presentation, we needed a simple and effective way to communicate to our audience about how we can use a laser driven into chaos can be used for data protection, G. Weerasinghe and W. Tou decided that it is best to be done using a simple physical model which can be used to demonstrate how the laser is driven into chaos. So W. Tou and A. Shalashilin extracted a self pulsating laser diode from a typical compact disc drive from personal computers.

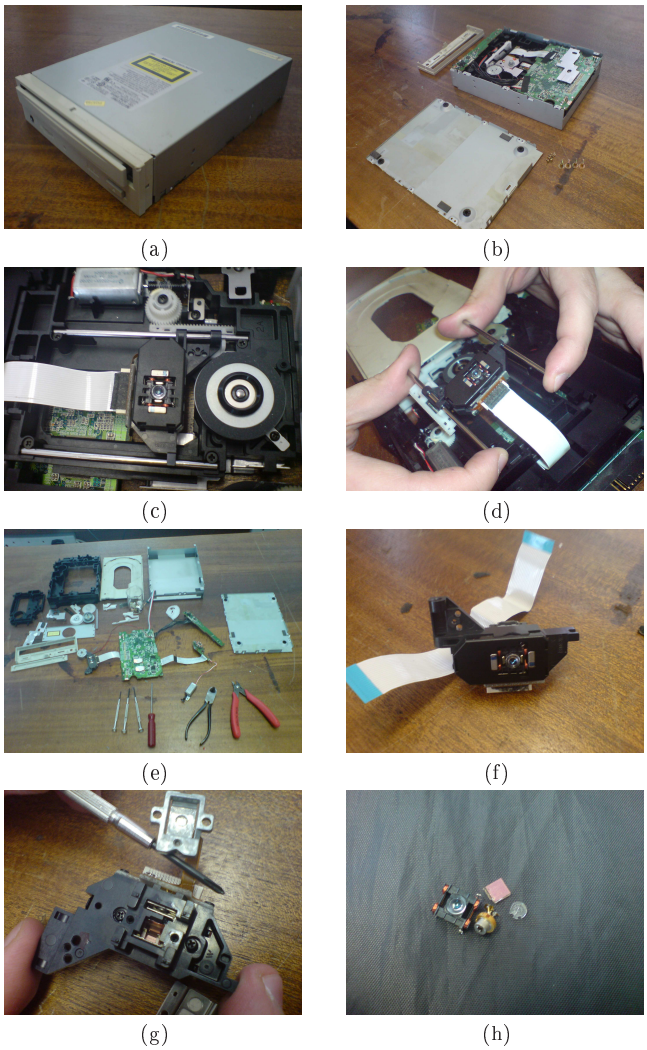
2.2 Technical information

2.2.1 The operation and theory behind the laser

Authors: L. Pomfrey and P. Smith

The purpose of the experiment is to see how a laser diode from a CD player can be driven into chaos. The usefulness of this is that another signal can be masked by the chaos, as can be seen in Fig. 2.1. It has been shown that phase-locking two lasers together allows for the chaotic signal to be removed and the original signal retrieved. This has been conducted using a master laser driven into chaos which then phase-locks a slave laser.[13] We will discuss the rate equations which will model the behavior of a self-pulsating laser diode and how the laser will be driven into chaos.

For this laser we are assuming it is a two level system. The rate equations are used to describe the population densities of the charge carriers in a laser. The laser diode or semi-conductor laser is essentially a p-n junction with cleaved edges that act as reflecting surfaces which supply the cavity feedback. A semi-conductor has two bands, a valence band and a conduction band with a forbidden space between them called the band gap. An amount of energy



(a) (b) (c) (d) (e) (f) (g) (h)

Figure 2.3: A series of photographs documenting the removal of the laser from a Compact Disc player in the lab. 2.3h shows the actual laser canister next to the lens normally mounted above it.
Created by W. Tou.

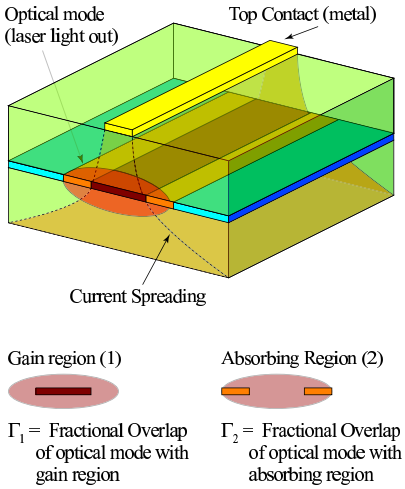


Figure 2.4: Schematic diagram showing the structure of a real CD laser.[11] *Created by, and used with permission of, S. Lynch.*

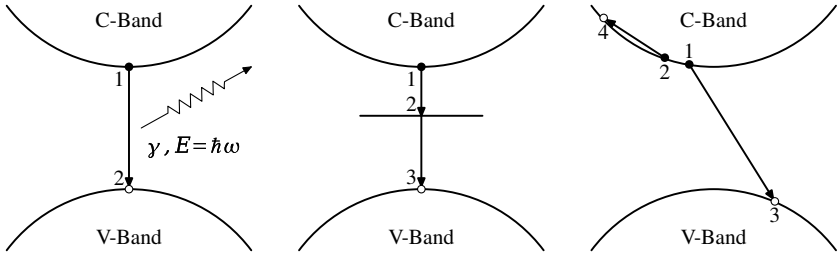
equal to the band gap needs to be added for an electron in the valence band to jump to the conduction band. In transitions between the bands photons will be emitted and under the right inversion conditions of the 2 levels lasing will occur.

Semi-conductors can be doped such that there are a small amount of impurities in the materials that make it up. These impurities can lead to there being electron gaps ("holes") act as positive charge carriers. This is a p-type semiconductor. The impurities can also lead to there being extra electrons with no space in an outer electron shell for them. This is an n-type semi-conductor. For instance Gallium Arsenide (GaAs) can be doped with Zinc. The Zinc with two outer shells replaces the Gallium with three outer shells and is a p-type material. Doping with Selenium (with six-outer shells) replaces the Arsenide (with five outer electrons) this means there are extra electrons in the lattice without spaces in electron clouds. This causes them to act as negative charge carriers. Placing together pieces of n- and p-type semiconductor creates a p-n junction. A voltage source drives the positive and negative charge carriers towards the junction. Laser diodes are forward biased p-n junctions.

Fig. 2.4 shows a schematic of a CD-laser.

Recombination of carriers and the generation of light

When an excited electron returns to it's ground state it releases a photon of light,



(a) Radiative recombination

(b) Non-radiative recombination

(c) Auger recombination

Figure 2.5: Illustration showing possible recombination processes in a semi-conductor on a simple band diagram.[11] 2.5a shows the radiative recombination process, this requires a vertical transition in k-space. 2.5b shows the processes of defect and surface recombination. These occur when an electron falls into a defect or surface level in the band gap and recombines non-radiatively from there. 2.5c shows the two-body Auger recombination process. There are many permutations of this process, denoted by the initial and final energy states. 2.5c depicts the CCCH process, which involves three electron states and one heavy hole state. In this example, two electrons collide in the conduction band. One of the electrons is excited into the valence band while the other is excited higher into the conduction band, where it eventually thermalises back down to the bottom of the conduction band, releasing the excess energy as heat.[11]
Created by L. Pomfrey.

with a frequency, ν, given by (2.1).

$$\nu = \frac{E}{h} \tag{2.1}$$

The process of radiative transition is essential to the created of light photons in semiconductors.[11] In this section we will discuss how this process is harnessed to cause light amplification and lasing.

The rate of change of the number of carriers in an electrically pumped semiconductor will depend on the input rate of carriers by electrical pumping and the rate of loss of carriers to the recombination processes shown in Fig. 2.5. This can be summarised mathematically, as in (2.2).

$$\frac{dN}{dt} = R_{\text{gen}} - R_{\text{rec}} = \frac{J}{eV} - R_{\text{rec}} \tag{2.2}$$

Where e is the electronic charge, V the active region volume, J the magnitude of the pumping current, R_{gen} the carrier generation rate, and R_{rec} the recombination rate.[11] The recombination rate can be written in terms of components as in (2.3).

$$\begin{aligned} R_{\text{rec}} &= R_{\text{sp}} + R_{\text{nr}} + R_{\text{l}} + R_{\text{st}} \\ &= \frac{N}{\tau(N)} + R_{\text{st}} \end{aligned} \tag{2.3}$$

Where we have the rate of carrier recombination giving spontaneous emission, R_{sp}, the rate of non-radiative carrier recombination, R_{nr}, the carrier leakage rate, R_{l}, and the rate of carrier recombination giving stimulated emission, R_{st}. The term relating to the rate of stimulated emission is the most important one for lasing. Since we are most interested in this term we can lump the other carrier recombination terms together and rewrite them in terms of the carrier density, N, and a time-constant, $\tau(N)$. This can be described by a power series in N, as shown in (2.4).[11]

$$\frac{N}{\tau(N)} = AN + BN^2 + CN^3 \tag{2.4}$$

Fig. 2.5 shows some of these recombination processes. The AN and CN^3 terms deal with non-radiative recombination (by the processes described by Fig. 2.5b and Fig. 2.5c respectively), and the BN^2 term deals with radiative recombination (as shown in Fig. 2.5a.)

We can now gather all of the recombinations rates into the simple rate equation in (2.5). We can now also describe the rate at which photons are generated as in (2.6).[11]

$$\frac{dN}{dt} = \frac{J}{eV} - \frac{N}{\tau(N)} - R_{\text{st}} \tag{2.5}$$

$$\frac{dP}{dt} = \Gamma R_{\text{st}} + \Gamma \beta_{\text{sp}} R_{\text{sp}} - \frac{P}{\tau_P} \tag{2.6}$$

Here, Γ is a pre-factor that takes into account the difference that the cavity volume (which the photons occupy) is larger than the active region volume. The last term is the rate of removal of photons from the cavity, this is defined in a way similar to the carrier recombination rate in (2.5) for the time being. β_{sp} is the reciprocal of the number of optical modes in the bandwidth of the spontaneous emission. R_{sp} is the rate of spontaneous emission.

Linear gain approximation and the generation of laser light

For the modeling of the behavior of a laser diode, several assumptions and approximations have to be made. The gain is approximated to be linear[2] with the carrier density N. In a real device the gain would vary with the carrier density. This linear approximation is made to avoid having to make the complication of performing the

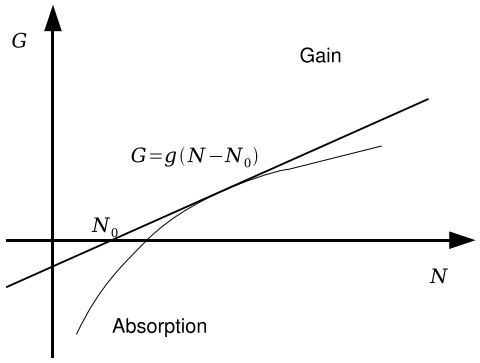


Figure 2.6: Illustration showing the linear gain approximation. Plotted on the graph is the peak gain as a function of carrier density.[11]
Created by L. Pomfrey.

full gain calculation. For a bulk active region the approximation $G = g(N - N_0)$ is used. At first in a real laser the gain is linear with increasing density of states but then saturates.

Another part of this approximation is that the non-linear term of the gain is ignored. It is argued that it is required for a full description of the behavior of a semiconductor laser.[3] This is because at high light intensities the gain saturates however for the purpose of this model only the linear approximation shall be used. The coefficient g is called the differential gain and corresponds to the gradient of the line. This has been shown to be a good approximation for the siuation where undoped or slightly doped GaAs is used as the material of the semi-conductor laser. N_0 is the crossing point of the linear approximation of the gain corresponding to transparency.

This approximation allows us to rewrite (2.5) and (2.6) as (2.7) and (2.8) respectively.

$$\frac{dN}{dt} = \frac{J}{eV} - \frac{N}{\tau(N)} - v_{\mathrm{gp}} G P \qquad (2.7)$$

$$\frac{dP}{dt} = \Gamma v_{\mathrm{gp}} + \Gamma \beta_{\mathrm{sp}} B N^2 - \frac{P}{\tau_P} \qquad (2.8)$$

Where G is the gain and v_{gp} is the group velocity.

Other modifications that must be made when dealing with a real CD laser and the final rate equations

Other adaptations that are required for the rate equations to model a real laser are for the spreading of the injection current from the contact plate as it travels down to the microchip at the base of the device (as is apparent when considering Fig. 2.4). This means that only some of the active region receives current. The optical mode is larger than this and therefore overlaps with some unbiased active region. In the unbiased region photons will be converted back into carriers. This is the opposite of the lasing process and must be taken into account in the form of a second rate equation for the charge carriers. This is considered as a negative gain. The photon density in the active region is taken to be uniform and the photon and carrier densities

have been averaged over the length of the device to make the calculation tractable.

We can now derive the three rate equations for the model. These have a similar form to the rate equations used by Yamada to model a semi-conductor laser.[3][11]

$$\frac{dN_1}{dt} = \frac{J}{eV} - \frac{N_1}{\tau(N_1)} - \frac{N_1 - N_2}{\tau_{12}} - \Gamma_1 v_{\mathrm{gp}} g_1 \left|(N_1 - N_{01})\right| P \quad (2.9a)$$

$$\frac{dN_2}{dt} = -\frac{N_2}{\tau(N_2)} - \frac{N_2 - N_1}{\tau_{21}} + \Gamma_2 v_{\mathrm{gp}} g_2 \left|(N_2 - N_{02})\right| P \quad (2.9b)$$

$$\frac{dP}{dt} = v_{\mathrm{gp}} \left[\Gamma_1 g_1 \left|(N_1 - N_{01})\right| - \Gamma_2 g_2 \left|(N_2 - N_{02})\right| - \alpha\right] P + \beta B N_1^2 \quad (2.9c)$$

$$\tau_{N_i} = \left(A + BN_i + CN_i^2\right)^{-1} \quad (2.10)$$

(2.9a) and (2.9b) deal with the rate of change of charge carriers in the gain region (denoted 1) and absorbing region (denoted 2) respectively. (2.9c) governs the rate at which photons are generated. Terms of the form $(N_i - N_j)/\tau_{ij}$ deal with the diffusion of carriers from one region to the other. The forms of the gain, G for regions 1 and 2 have been substituted for their respective equations. The prefactor Γ_1 is the fractional overlap of the optical mode with the gain region. Γ_2 is the fractional overlap of optical mode with the absorbing region. In (2.9c), α is the waveguide/mirror loss. β is the spontaneous emission factor, which has combined the Γ and β_{sp} terms from (2.8). (2.10) gives the carrier lifetime.

Some typical values of the terms in the rate equations, and their meanings, are given in Table 2.1.

Causing chaotic pulsations in the laser

For a real device a laser can be driven into chaotic behavior under certain conditions. Modelling this using the rate equations involves adding terms to the rate equations such that the pulsed output will becomes chaotic with time. This represents an additional electrical modulation to the laser. In general the natural frequency of the free running self-pulsations will lock to the frequency of the applied signal. If, however, the external frequency is too far away from the natural resonant frequency, the signals will compete and result in chaotic pulsation.

This external electrical modulation comes in the form of varying the DC current with a signal generator. The constant current will become sinusoidal, modifying (2.9a) to include this results in 2.11).

$$\frac{dN_1}{dt} = \frac{J_0 \sin(\omega_e t)}{eV} - \frac{N_1}{\tau(N_1)} - \frac{N_1 - N_2}{\tau_{12}} - \Gamma_1 v_{\mathrm{gp}} g_1 \left|(N_1 - N_{01})\right| P \quad (2.11)$$

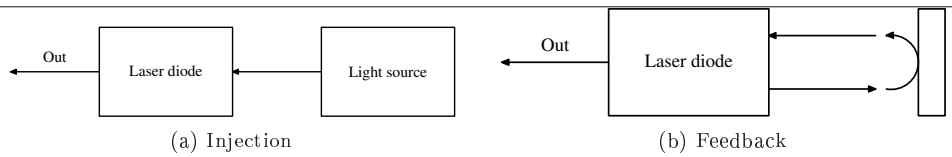
Where J_0 is the amplitude of the varying current, and ω_e is the angular freuency of the modulation.

Varying the value of the external frequency and seeing how the autocorrelation function changes allows us to determine when the pulses become chaotic. Inhomogeneous current injection has been previously used to investigate the bistability and pulsations of a semiconductor laser and how they vary with the current injection.[2]

Other ways to cause a semi-conductor laser to become chaotic are through optical injection, optical feedback and optoelectronic feedback.[14, 15] These have been investigated before as methods for causing chaotic pulsation. These have been used in investigating chaotic syn-

Parameter name	Symbol	Nominal value
Electronic charge	e	1.6×10^{-19} C
Active region volume	V	7.2×10^{-11} cm^3
Non-radiative coefficient	A	1×10^{8} s^{-1}
Radiative coefficient	B	3×10^{-10}cm^{3}s^{-1}
Auger coefficient	C	7×10^{-29} cm^{6}s^{-1}
Diffusion time, region $1 \rightarrow 2$	τ_{12}	1.5×10^{-9} s^{-1}
Diffusion time, region $2 \rightarrow 1$	τ_{21}	2.5×10^{-9} s^{-1}
Overlap region 1	Γ_1	0.1
Overlap region 2	Γ_2	0.2
Transparency density region 1	N_{01}	1.2×10^{18} cm^{-3}
Transparency density region 2	N_{02}	1×10^{18} cm^{-3}
Group velocity	v_{gp}	7.5×10^{9} cms^{-1}
Differential absorption/gain coefficient region 1	g_1	$\frac{3.08 \times 10^{-6}}{v_{\mathrm{gp}}}$ cm^2
Differential absorption/gain coefficient region 2	g_2	$4g_1$ cm^2
Waveguide/mirror losses	α	10 cm^{-1}
Spontaneous emission factor	β	1×10^{-5}
Stepsize	Δ	5×10^{-13} s^{-1}
Bias current	J_0	*variable* A

Table 2.1: List of parameters and their nominal values for the CD laser model.[11] *Created by L. Pomfrey.*



(a) Injection (b) Feedback

Figure 2.7: Possible methods for causing chaotic pusation in lasers. 2.7a shows optical injection using an external light source, whilst 2.7b shows feedback from an external cavity mirror.
Created by L. Pomfrey and P. Smith

chronization of two lasers for secure communications.

These methods differ from modulating the injection drive current by using optical perturbations. They are similar to the electrical modulation in that the external signal locks the diode laser to its frequency. The two conditions for external locking to work for optical injection are that the detuning of the external frequency and that of the natural resonant frequency of the laser diode must not be too large and the injected power has to be sufficiently large. However, it requires an external light source to act as the locking signal.

The optical feedback method involves using an external cavity mirror which causes optical feedback and induces chaos in the laser diode.[16]

For a small device meant for mass production as a solution for secure communications, electrical modulation is the best way to cause a laser to pulse chaotically. Optical injection and optical feedback methods are useful in a laboratory experiment but require precision lining up of a laser device or external cavity wall behind the laser diode.

2.2.2 The auto-correlator

Author: G. Weerasinghe

Lasers can be made to pulse light typically with durations between nanoseconds (ns) and femtoseconds (fs). To the naked eye, these pulses are so fast, that they naturally appear as a continuous stream of light. Photodiodes are also inadequate for resolving such fast pulses. A standard method for measuring fast light pulses is to perform an intensity auto- correlation. The technique of auto-correlation is to pass the incident light pulses through a beam splitter, creating two identical copies of the incident pulsed light. One copy of the light directly enters a second harmonic generation (SHG) crystal (see section 2.2.3). The second copy traverses a variable path, such that a path difference is created between the two beams before entering the SHG crystal.

The function of the SHG crystal is to recombine the two beams creating a single beam with an intensity proportional to the product of the two intensities. When the two beams have been combined, it is then possible to determine how well they "fit" with each other (correlate). Due to the non-linearity of the SHG crystal, the effect of combining the two separate beams will also double the frequency.

Given the output photons from the SHG crystal will have twice the incident frequency, the Planck energy law insists that light emerging from the SHG crystal must consequently have twice the energy per photon than the incident light. For energy to be conserved, this must come at the expense of the total number of photons leaving the SHG crystal, *i.e.* the intensity of the resulting beam. Semi-conductor lasers used in CD players, are low powered, therefore it is expected that the intensity of the light emerging from the SHG crystal will be immeasurably low. For the autocorrelated light to be detected, a photo-multiplier must be used between the SHG crystal and photodiode detector.

Convolution and intensity auto-correlation

Mathematically, the convolution of two functions is defined as in (2.12).

$$\int_{-\infty}^{\infty} f(x)g(-t+x)dx \qquad (2.12)$$

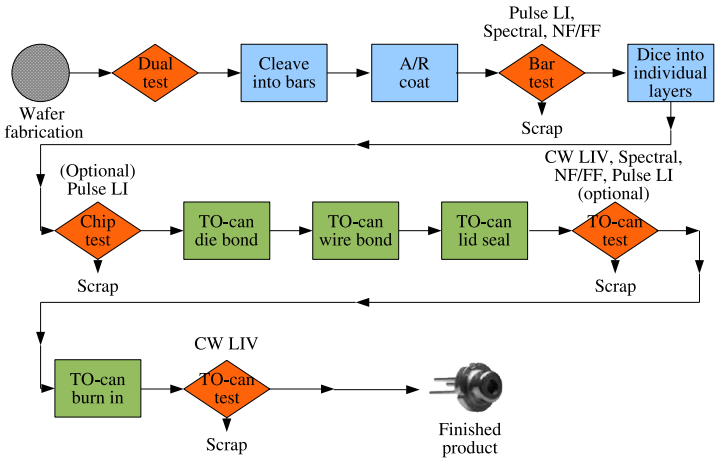


Figure 2.8: Compact Disc player laser production by the TO-can production line. *Created by L. Pomfrey.*

Qualitatively, the convolution of two functions is a measure of how much a function $g(x)$ overlaps with a function $f(x)$ as it is shifted over the function $f(x)$ by an amount t. The resulting convolution function is a function of t, thus, a plot of the convoluted function against t will yield a graphical representation of how the function $g(x)$ overlaps with $f(x)$ as it is shifted by t. The cross-correlation of two functions is similar to the convolution and is defined as in (2.13).

$$\int_{-\infty}^{\infty} g(t+x)f(x)^{\star}dx \qquad (2.13)$$

Where $f(x)^{\star}$ denotes the complex conjugate of the function $f(x)$. The cross-correlation is a description of the similarity between two functions $g(x)$ and $f(x)$, where the overlap of the functions can be altered by a parameter t.

To obtain an expression for the auto-correlation we replace $g(t+x)$ with $f(t+x)$, as in (2.14).

$$\int_{-\infty}^{\infty} f(t+x)f(x)^{\star}dx \qquad (2.14)$$

Using the definition of cross-correlation, the auto-correlation can now be interpreted as the similarity between $f(x)$ and $f(t+x)$, *i.e.* the similarity between a function and itself after it has been shifted by some amount t.

The intensity autocorrelation of two beams entering the SHG crystal can now be expressed mathematically. Since the laser will be pulsed, the intensity of light will be a function of time, t. The light traversing the variable path difference will represent the original intensity function shifted by a small variable time interval b, where b is the time taken for the beam to traverse the variable path difference.

An expression for the intensity auto-correlation of the two beams entering the

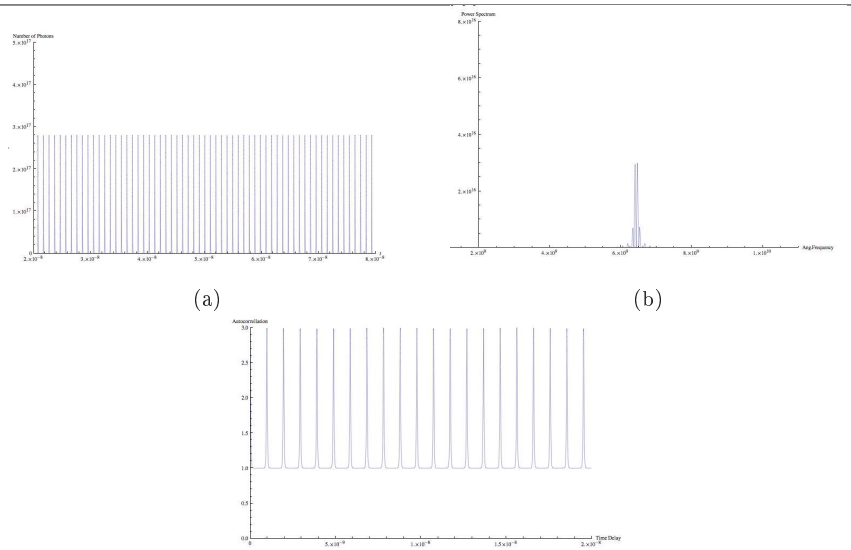


Figure 2.9: 2.9a shows a pulsation with a definite frequency. 2.9b shows the power spectrum of 2.9a. A sharp peak corresponding to a large component of energy concentrated at a particular frequency can clearly be seen. It should be noted that the power spectrum does not appear as a sharp vertical line because the pulsation, in this case, is not a perfect sinusoidal wave and actually contains a combination of frequencies. 2.9c shows the auto-correlation of 2.9a. It can be seen that an auto-correlation value of 3 corresponds to maximum correlation between a pulse and a pulse shifted by a small time delay. The distance between peaks is the pulse interval. For a real optical auto-correlator, this distance corresponding to the time delay would be varied and measured, therefore, the pulse interval can be found.
Created by G. Weerasinghe.

SHG crystal can be written as in (2.15).

$$\int_{-\infty}^{\infty} U(t)U(b+t)dt \tag{2.15}$$

It is conventional to introduce a normalisation constant such that when $b = 0$, *i.e.* both pulses are fully correlated, the maximum correlation of the two pulses will be represented by 1. This normalisation constant is given by the integral of the intensity squared over all possible values of t. The expression for the intensity auto-correlation is further changed when the geometry of the equipment is considered.[17] In the case of a single pulsed light source being split and recombined in a SHG crystal, the intensity auto-correlation function is given as in (2.16).

$$1 + \frac{\int_{-\infty}^{\infty} 2U(t)U(b+t)dt}{\int_{-\infty}^{\infty} [U(t)]^2 dt} \tag{2.16}$$

In this case, the maximum correlation of two pulses will result in this function equating to 3. This can be seen by setting $b = 0$ (if the time difference between the two pulses is zero, pulses should be fully correlated) as in (2.17).

$$1 + \frac{\int_{-\infty}^{\infty} 2U(t)U(b+t)dt}{\int_{-\infty}^{\infty} [U(t)]^2 dt} = 1 + \frac{2\int_{-\infty}^{\infty} [U(t)]^2 dt}{\int_{-\infty}^{\infty} [U(t)]^2 dt} = 3 \tag{2.17}$$

Power spectrum

The Wiener-Kinchin theorem states that the Fourier transform of a cross-correlation function (two functions, $F(t)$ and $G(t)$) can be expressed as in (2.18).

$$\sqrt{2\pi} G(k)F^{\star}(k) \tag{2.18}$$

For an auto-correlation, $G(k) \rightarrow F(k)$, therefore, the Fourier transform of an auto-correlation function will be as given in (2.19).[18]

$$\sqrt{2\pi}\,|F(k)|^2 \tag{2.19}$$

The Fourier transform of the auto-correlation function is the modulus squared of the Fourier transform of $F(t)$. The modulus squared of $F(k)$ is known as the power spectrum which describes how the energy of a pulsation is distributed amongst different frequency components.

Other references: [19, 1].

2.2.3 Second harmonic generation

Author: P. Smith

The experiment to generate and measure chaotic optical pulses from a Compact Disc laser diode utilises an intensity auto-correlation to investigate the pulse width. In the experiment the pulsed laser light is split into two equal parts by a beam splitter and travels along two different paths (with one path length being varied by a delay line) before being recombined in a second harmonic crystal. The intensity of the output from the second harmonic signal is then measured using a photo-multiplier tube (PMT).

Electric polarisation of linear and non-linear materials

Second harmonic generation is a phenomena associated with non linear materials. It is the response of the atoms inside the material to the associated electric field of the laser light that makes second harmonic generation possible. The outer electrons

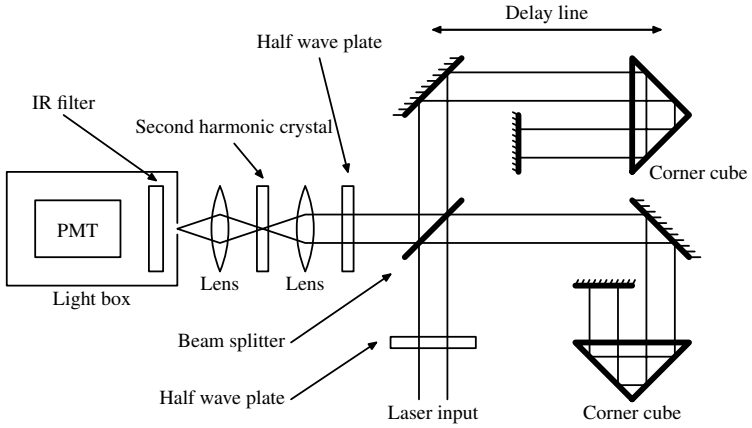


Figure 2.10: Schematic diagram of a simple intensity auto-correlator.[1]
Created by L. Pomfrey.

of atoms respond to an applied electric field in such a way that the atoms become electrically polarised. In linear media the macroscopic polarisation is directly proportional to the magnitude of the electric field, as can be seen in (2.20).

$$\mathbf{P} = \epsilon_0 \chi \mathbf{E} \tag{2.20}$$

Where $\mathbf{P}$ is the polarisation, ϵ_0 is the permittivity of free space, χ is the electric susceptibility constant, and $\mathbf{E}$ is the electric field.

The second orders (and higher order terms) do not make a sizeable contribution in linear, isotropic, homogeneous materials because the linear term is so large compared to them.

In a non linear material where the applied electric field is large enough, this linear response breaks down and higher order contributions become important. The higher order terms can be shown by expanding χ in a power series as in (2.21).

$$\chi = \chi_1 + \chi_2 \mathbf{E} + \chi_3 \mathbf{E}^2 + \ldots \tag{2.21}$$

Substituting into (2.20) gives (2.22).

$$\mathbf{P} = \epsilon_0 \left(\chi_1 \mathbf{E} + \chi_2 \mathbf{E}^2 + \chi_3 \mathbf{E}^3 + \ldots \right) \tag{2.22}$$

(2.20) can then be re-written as (2.23).

$$P = P_1 + (P_2 + P_3 + \ldots) \tag{2.23}$$

The linear P_1 completely describes the polarization of linear media and suits the purpose of linear optics (where $\mathbf{P}_1 = \epsilon \mathbf{E}$ and $\epsilon = \epsilon_0 \chi_1$).

Lasers however allow intense coherent light to be focused onto wavelength dimensions such that the associated electric field of the laser exceeds 10^{10} V.m^{-1}, on the order of strengths of fields binding electrons to nuclei. This means that the higher order terms can become sizeable in non-linear media to appreciably affect the first order polarization.

Second harmonic generation

A non-linear crystal can be used to produce second and even third harmonic gen-

eration providing a means to produce coherent, intense electromagnetic radiation at which there are no efficient laser transitions. From (2.22) and (2.23) the second order term is given by (2.24).

$$P_2 = \epsilon_0 \chi_2 \mathbf{E}^2 \tag{2.24}$$

It can be seen that the second order polarization term P_2 is proportional to the square of the applied electric field.

The signal produced from second harmonic generation is twice that of the fundamental wave and half the wavelength. This can be shown from a basic mathematical argument. If the applied electric field or one of its Fourier components is of the form given in (2.25),

$$\mathbf{E} = \mathbf{E}_0 \cos(\omega t) \tag{2.25}$$

Then, substituting into (2.24) gives (2.26).

$$\begin{aligned} P_2 &= \epsilon_0 \chi_2 \left(\mathbf{E}_0 \cos(\omega t)\right)^2 \\ &= \epsilon_0 \chi_2 \mathbf{E}_0^2 \left[\frac{1}{2}\left(1 + \cos(2\omega t)\right)\right] \\ P_2 &= \frac{1}{2}\epsilon_0 \chi_2 \mathbf{E}_0^2 + \frac{1}{2}\epsilon_0 \chi_2 \mathbf{E}_0^2 \cos(2\omega t) \end{aligned} \tag{2.26}$$

The second term of the second order polarization contains the term that is twice the frequency of the applied optical field, ω. The first term is a constant or DC component representing the optical rectification.

The P_2 polarization term causes dipole oscillations at 2ω in the medium. These dipole oscillations are what cause electromagnetic radiation of angular frequency 2ω to be generated and emitted from the medium together with the fundamental (pump wave) at frequency ω. This is why the infra-red pass filter at 780 nm is required to block out the fundamental wave, so that only the second order signal reaches the photo-multiplier tube.

Phase matching

The non-linear response that allows the conversion of energy from the fundamental wave to the second order electromagnetic wave also allows for energy to be converted back the other way. The direction of the energy flow is dependant on the phase matching between the fundamental wave of frequency ω and the second order wave of frequency 2ω. Due to dispersion the light emitted at frequency 2ω travels at a different speed in the optical medium than the light at frequency ω. Another way to view this is that the refractive index inside a non-linear material is slightly different for the fundamental and harmonic wavelengths.

This means the two waves will be periodically in and out of phase with each other. This will result in very little frequency doubled output being obtained. The irradiance of the second harmonic field is proportional to the irradiance factor given in (2.27).

$$\sin c^2 \left(\frac{\Delta K L}{2}\right) \tag{2.27}$$

Where $K = n\omega/c$ is the wave propagation constant, and L is the distance into the crystal.

For optimum non-linear phase conversion a proper phase relationship has to be maintained along the propagation direction through the material. The difference in wave number between the two beams should, ideally, be zero, where the difference is given by (2.28).

$$\Delta K = K_{2\omega} - 2K_{\omega} \tag{2.28}$$

Where $K_{2\omega}$ is the wave number of the secnd harmonic wave, and K_{ω} is the wave number of the fundamental wave.

When $\Delta K = 0$ the irradiance and signal of the second harmonic wave is maxi-

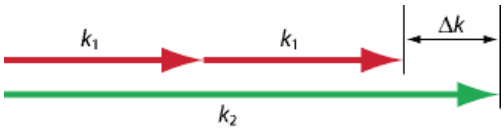


Figure 2.11: Phase mismatch for second harmonic generation. Due to chromatic dispersion the wave number of the second harmonic is more than twice as large as that for the fundamental wave.[20]
Created by P. Smith.

mized, however, due to dispersion ΔK is typically not zero and the irradiance will be reduced. A coherence length, L_C, can be defined as in (2.29).

$$L_C = \frac{\pi}{\Delta K} \tag{2.29}$$

When $L = L_C$, the $\sin c^2$ intensity factor is reduced to about 0.4 of its maximum value. This provides a method of defining the efficiency of a crystal for second harmonic generation. There are several methods for phase matching a non-linear crystal such that second harmonic generation is properly maximized. The efficiency of second harmonic generation is maximized by making small angular adjustments to the crystal to ensure correct phase matching.[1] This is best matched by critical phase matching which is a form of birefringent phase matching and quasi-phase matching (which is employed in the non-linear crystals produced by Thorlabs[21]).

Critical phase matching: This means that an angular adjustment of the crystal (or the beam) is used to find a phase matching configuration. In a birefringent (double refraction) crystal, non incident light will be refracted such that the beam is effectively split in two (creating an ordinary ray (O-ray)s and an extra-ordinary ray (E-ray)). The O- and E- rays travel along different paths and have different refractive indices such that there will be two rays emerging from the crystal. Changing the angle of propagation leaves the refractive index for the O-ray constant while the E-rays' refractive index will change. At a certain angle, the direction through the crystal will be such that $n_{2\omega}$ (where n is the refractive index) for the E-ray is equal to n_ω for the O-ray and the fundamental and second harmonic waves will remain in step and phase matching will be achieved.

Quasi-phase matching: This process involves the usage of a crystal such that every coherence length, L_C, the sign of the electric susceptibility term χ_2 changes. Over the coherence length, the energy is transferred from the fundamental field to the second harmonic field. Just as the transfer is about to switch and cause attenuation of the fundamental field the sign of the electric susceptibility changes and maintains the proper phase relation between the second harmonic field and the dipoles of the medium. A structure called periodically poled

lithium niobate (PPLN) which uses external fields to perform a periodic poling of a non-linear ferro-electric material.

A note on other second order processes

Apart from second harmonic generation there are other second order non-linear processes (where unlike second harmonic generation the resultant wavelengths are not determined by the initial wavelengths). These are listed in Table 2.2 with some third order processes. In SHG the lack of phase matching between the fundamental field and second harmonic field means that other second order processes do not occur with second harmonic generation. Frequency mixing of two or more incident beams can lead to sum frequency generation where $\omega_3 = \omega_1 + \omega_2$, and difference frequency generation where $\omega_3 = \omega_1 - \omega_2$.

Experimental consequences

In terms of the experiment, the actual phase matching of the fundamental and harmonic fields is a process of configuring the crystal such that the max efficiency of conversion between the two is achieved (probably by critical phase angle matching). The pulses will be split into two equal parts and, due to the delay line, will be out of phase when the two beams are recombined in the second harmonic generation crystal. When the two beams' path difference matches up, the second harmonic generation will be maximized and the intensity reading from the photo-multiplier tube will be maximized as well.

The second harmonic crystal is important for the experiment as it is this that recombines the two pulses back together into a single pulse. The intensity of which is then measured by the photo-multiplier tube. The amount of second harmonic radiation produced is determined by the temporal overlap of the two pulses. A maximum in intensity is measured at the photo-multiplier tube if the two signals overlap perfectly. The delay line varies the optical path length over which one of the signals travels such that the intensity of the combined signal is varied. The temporal resolution means that the one pulse will have a signal $E(t)$ and the other one will have $E(t + \tau)$ (where τ is varied though the optical path length by the delay line).

The intensity correlation function is measured if the two beams are recombined in the crystal. The magnitude of the second harmonic signal is proportional to the integral of the squared modulus of the intensity of the beam. Information is gained when a pulse is auto-correlated. When the time delay between the two arms is adjusted it can be arranged so that a pulse is correlated with the one that directly follows it. This is known as cross-correlation and it will be used to study the overall temporal chaos of the laser output.

2.3 The design and construction of the auto-correlator

Author: A. Shalashilin.

In order to test the theory of the chaotic laser auto-correlation, an experimental setup was developed. Although time pressure prevented us from constructing it, nonetheless a detailed plan was developed specifying individual components and resulting in a scale technical drawing of the

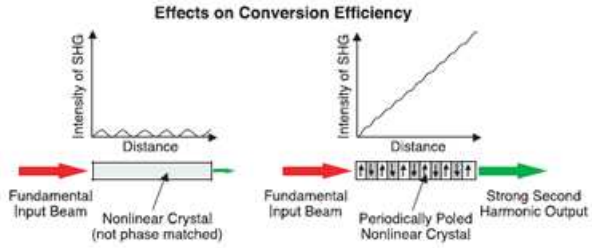


Figure 2.12: Graphs showing effectiveness of PPLN compared to non-phased matched situation.[21]
Created by P. Smith.

Second order non-linear processes	**Third order non-linear processes**
Second harmonic generation	Third harmonic generation
Three wave mixing	Four wave mixing
Optical rectification	Kerr effect
Parametric amplification	Raman scattering
Pockels effect	Brillouin scattering

Table 2.2: Examples of second- and third-order non-linear processes.
Created by P. Smith.

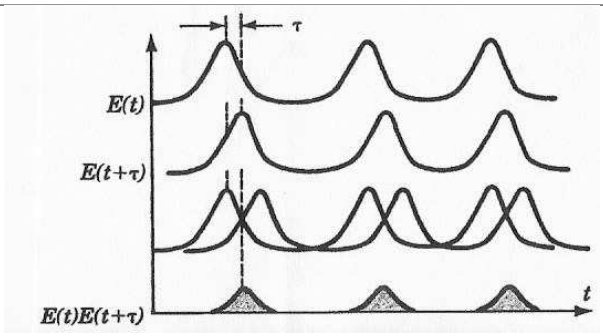
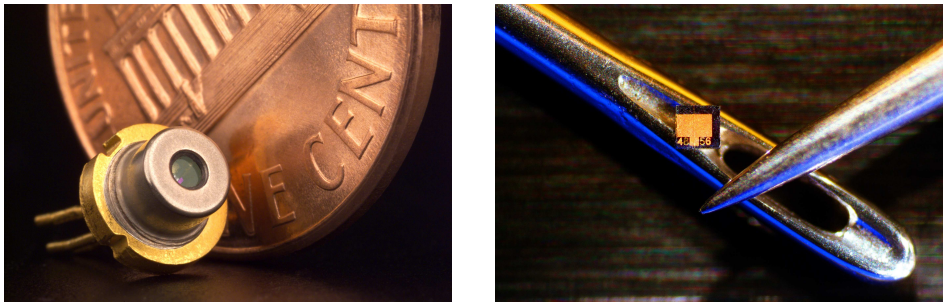


Figure 2.13: A graph of two signals, their overlap in the second harmonic crystal and the second harmonic signal. Only the signal under the pulse is detected by the photomultiplier tube. Varying τ obtains better temporal resolution.[22]
Created by P. Smith.



(a) (b)

Figure 2.14: A semi-conductor laser diode of the type used in the experiment. 2.14a shows the laser in it's enclosure ready to be used, a US one cent coin is shown for scale. 2.14b shows the laser chip itself, a sewing needle is shown for scale.
Created by W. Tou and A. Shalashilin.

setup.

Our laser was cannibalised from an ordinary CD-ROM drive, with standard 780 nm average output. The experiment would be safe to conduct and demonstrate without worrying about hazards caused by stray beams or accidental exposure into the naked eye, because the laser was rated as `Class I`.

The autocorrelation setup visibly reminds of a modified Michelson interferometer, which is not that far from the truth. Like in the famous example, there is a 50:50 beam splitter mirror, which would send part of the laser beam towards a set of two mirrors, a corner cube and another mirror which would reflect it back along the same path. The other beam would follow an identical type of path, except the final mirror to corner cube distance would be variable. This is achieved by placing the mirror on a motorised delay line. This allows a very high precision measurement of the path length, and allows for precise almost infinitesimal change in the distance between the corner cube and the mirror (given the scales at which we are working at, this is an important advantage).

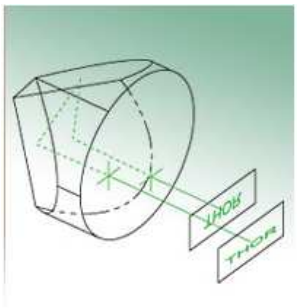


Figure 2.15: A schematic of a corner cube of the type used in the experiment.
Created by W. Tou and A. Shalashilin.

After the beams travel the two arm lengths and return to the beam splitter mirror, they are superimposed and in order to attain the data of the autocorrelation (which is essentially the difference between the lengths travelled by the two beams), we require for the re-

combined beam to travel first through a half-wave plate, and then a second harmonic crystal, made of monopotassium sulfate (KDP). This arrangement also includes two aspheric lenses which are necessary to focus the beam into the crystal and then re-focus the beam coming out. Finally the beam can enter a photo-multiplier tube, although since we only require the component of light that has been frequency doubled, the tube needs to be optically isolated and the beam filtered by an infra-red filter.

A schematic of the setup can be seen in Fig. 2.10, and the scale diagram can be seen in Fig. 2.16.

Like most optical experiments, ours is fairly costly. The most expensive part is the delay line, whose components alone total to £8,201.00. Luckily, through negotiations with Newport, we were able to attain a 15% discount on that figure.

All of the other components, which are manufactured by Thorlabs, add up to £1,208.20 (a detailed list is given in Table 2.3). An exception was the second harmonic crystal which neither of two optical suppliers had. Instead, we found a supplier called Red Optronics but have not yet had a reply from them. In any case, without that final piece of equipment (which is a costly piece of equipment in itself), our apparatus sums up to £8,179.05 which is quite expensive considering that this excludes VAT. Luckily some of the components such as the photo-multiplier tube and the CD-ROM drive, from which the laser was extracted by our team members, were provided by UCL Department of Physics and the London Centre of Nanotechnology.

It is highly unlikely that these expenses can be avoided as the experimental setup does require kinematic mirror mounts to directly reflect the beam and point it in the right direction, it requires the corner cubes to produce a perfect reflected beam parallel to the incident one, and it requires the delay line to precisely move the mirror so that we know an exact change in distance to a precision no simple ruler can measure. On the other hand, after our experiment would be concluded this apparatus can be reused for any other optical setups, this would justify the expenses slightly.

2.4 Computer modelling of the laser pulsation

Authors: M. Moussa and G. Weerasinghe

In order to understand what is going on in our experiment it is advantageous to build a simulation of the setup. The reasons for doing so are numerous, such as being able to observe effects and processes that would be impossible or difficult to measure/observe in a physical model (such as the region carrier densities in the laser cavity). A simulation allows us to vary different parameters and observe the consequences of doing so without a long down time compared to the setup. This also allows us to intercept errors in the setup that may not be obvious in the experiment, which would have wasted time and effort. Data may be represented visually in a variety of ways that are simply not possible with the physical model. This allows us to represent mission critical/interesting data to people in a useful way, which may have otherwise been impossible. Performance and experimental improvements are also easy to implement.

On the other hand it is important we do not oversimplify the model, which would almost certainly give us a different account of what is happening than the ex-

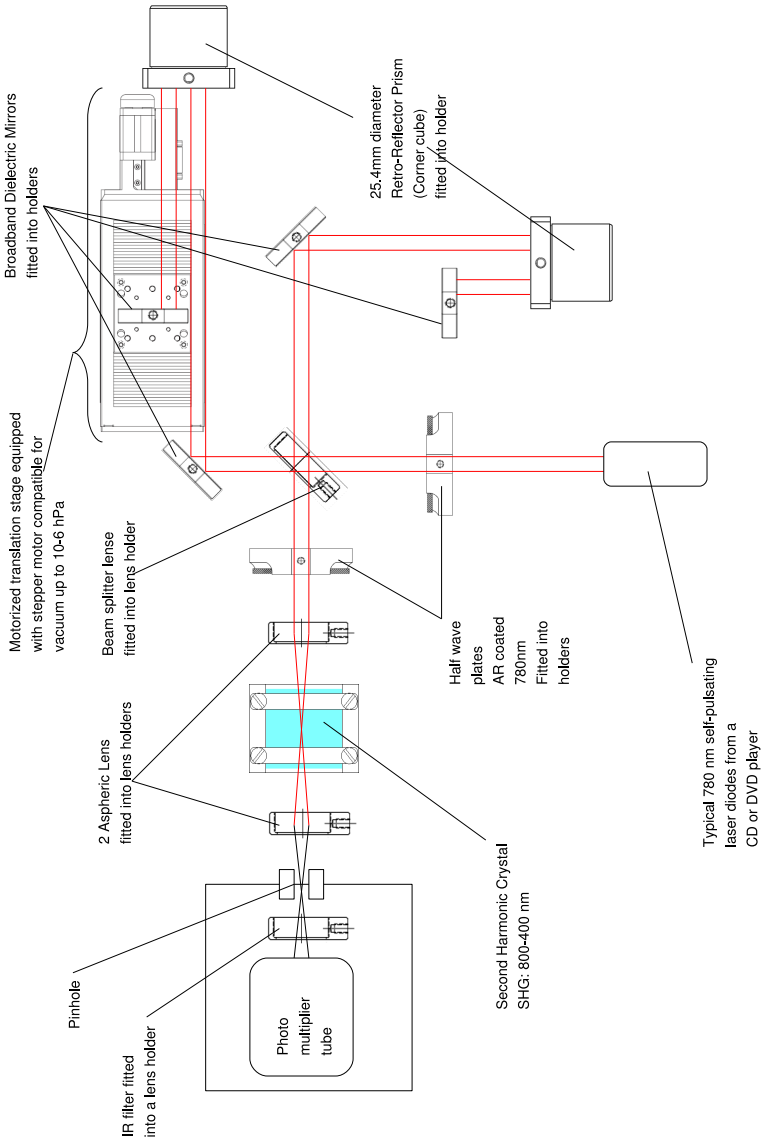


Figure 2.16: Scale drawing of the auto-correlator setup. *Note:* A larger version is available in the apppendix.
Created by W. Tou.

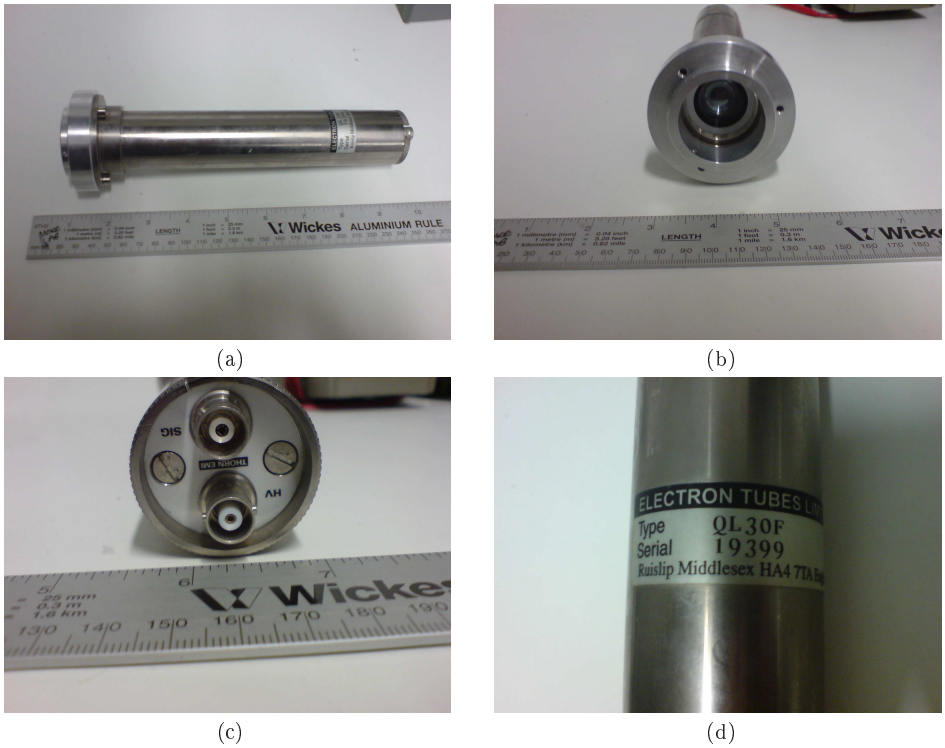
Component	Product code	Unit price	Quantity	Line price
Mounted half-wave plate	WPMH05M-780	£143.00	2	£286.00
Economy 50:50 Beamsplitter	EBS1	£17.80	1	£17.80
Mounted Geltech Aspheric Lens	C230TM-B	£53.10	2	£106.20
Broadband Dielectric Mirror	BB1-E03	£47.30	4	£189.20
Mounted Retro-Reflector (Corner cube)	PM1-RR300	£85.10	2	£170.20
Square Colored Glass Filter	FGL780S	£46.70	1	£46.70
Beamsplitter mount	LMR1	£9.90	1	£9.90
Mirror mount	KM100	£25.10	4	£100.40
IR Mount	FH2	£12.30	1	£12.30
Mounting Base	BA1	£3.50	10	£35.10
Mounting post	P100/M	£23.40	10	£234.00
			Subtotal:	£1,207.70

(a) Items from Thorlabs

Component	Product code	Unit price	Quantity	Line price
Motorized translation stage	UTS150PPV6	£2,945.00	2	£5,890.00
XPS controller	XPS-C6	£4,445.00	1	£4,445.00
Agilis Optical mount	AG-M100NV6	£276.00	2	£552.00
Drive module	XPS-DRV01	£359.00	2	£718.00
Hand held controller	AG-UC2	£176.00	1	£176.00
			Subtotal:	£11,781.00
			Total:	£12,988.70

(b) Items from Newport for the delay line

Table 2.3: The equipment required to build the auto-correlator and the prices of each item.
Created by A. Shalashilin.



(a) (b)

(c) (d)

Figure 2.17: Images of a photo-multiplier tube in the lab.
Created by W. Tou.

periment. It also important we check it is valid for a wide range of input. This requires us to check it with differential models that we know the analytical solution to and see if the model outputs a numerical solution that converges to the required accuracy.

2.4.1 Solving the rate equations

Author: M. Moussa A mathematical treatment of the laser simulation produces 3 first order coupled ordinary differential equations (ODEs) that need to be solved simultaneously to understand how the laser behaviour evolves over time.[11] The rate equations are given in 2.9, and are of the form given in (2.30).

$$\frac{\delta x(t)}{\delta t} = f(t, x(t), y(t), z(t)), \quad x(0) = x_0 \tag{2.30a}$$

$$\frac{\delta y(t)}{\delta t} = f(t, x(t), y(t), z(t)), \quad y(0) = y_0 \tag{2.30b}$$

$$\frac{\delta z(t)}{\delta t} = f(t, x(t), y(t), z(t)), \quad z(0) = z_0 \tag{2.30c}$$

These ordinary differential equations cannot be solved analytically due to the coupling from the dependent variables, *i.e.* the reliance on other variables at each time step. The only choice left is a numerical analysis which, rather importantly, will produce an approximation of the answer, not an exact answer, which may lead to complications explained later.

There are many mathematical tools at our disposal to begin an analysis, and so it is important to pick the right one for our specific needs. Numerical analysis requires us to consider three important aspects;

- Convergence: Whether the chosen numerical method converges to the solution.
- Stability: How sensitive the method is to errors.
- Order: How well the method approximates the solution.

An Euler method could be used[23], by which the differential is replaced with a finite difference approximation as in (2.31).

$$\frac{\delta n(t)}{\delta t} \approx \frac{n(t+h) - n(t)}{h} \tag{2.31}$$

Where h is the step size we choose.

This can easily be rewritten as a recursive function, for which a computer is well suited to solve. The problem with this, and similar methods, is that it is nowhere near accurate enough for small h, as required, especially when you consider that we expect the laser power to pulse sharply, which if not modelled accurately will throw us off. The Euler method is said to be first order, which means that the total error at the end of the approximation will be proportional to h.

A suitable candidate is a Runge-Kutta method. These are a well known family of numerical methods developed by C. Runge and M. W. Kutta in the 1900's. The most common implementation is the fourth order Runge-Kutta method, which in fact is so commonly used it is referred to as *the* Runge-Kutta method.[24]

For a given initial value problem (IVP) such as ours with the condition given in (2.32),

$$\frac{\delta n(t)}{\delta t} = f(t, y), \qquad y(t_0) = y_0 \tag{2.32}$$

Then the fourth order Runge-Kutta method is defined as in (2.33).

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{2.33}$$

Where the terms k_n are defined as in (2.34).

$$k_1 = f(t_n, y_n) \tag{2.34a}$$

$$k_2 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right) \tag{2.34b}$$

$$k_3 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right) \tag{2.34c}$$

$$k_4 = f(t_n + h, y_n + hk_3) \tag{2.34d}$$

2.4.2 The *Mathematica* model

Author: G. Weerasinghe

The simulation of chaotic and self pulsations from a laser diode was developed simultaneously in both *MatLab* and *Mathematica*. It was agreed that it would be sensible to not concentrate all efforts on a single solution but instead to develop the simulation across different languages. In this way, if obstacles or limitations occurred in one language, the hope would be that this may be overcome using another. There was also a division of skills amongst the group with members being competent in different programming languages.

The primary goal of the simulation was to accurately model the pulsations of a semiconductor laser when driven with both direct and alternating currents. This would involve solving the relevant rate equations numerically using the fourth order Runge-Kutta method. In addition to this, the autocorrelation and power spectrum functions were also implemented. As part of the simulation, I also thought it would be useful to implement an algorithm which would accurately detect the conditions needed to produce maximum chaotic output from the laser.

The simulation in Mathematica was designed using a modular approach from the start. This aided debugging and gave the code a more elegant and organised form. The end product was a library of modules rather than a mass of code. Individual modules could be readily and easily replaced with updated code, when needed. After evaluating the entire notebook in Mathematica, a user could open a blank canvas, and easily access the required modules. Ease of use was consistently a key criteria throughout the development of the simulation since the intention of the model was to supplement the physical development of the auto-correlator and it was important that every member of the group knew how to operate the simulation for cross-checking experimental and theoretical results. As a programming task, my aims were as follows:

- Produce a solution which would be easy to use and intuitive.
- Frequently group repetitive procedures into separate modules, thus reducing the bulk of the code.
- Produce a solution which would produce accurate results in a reasonable amount of time.
- Wherever possible, use Mathematica's own algorithms for calculations. These are usually optimized and generally much quicker.

Simulating the auto-correlation

The result of an intensity autocorrelation can be simulated using an algorithm, provided $P(t)$ has been calculated from solving the rate equations. The method of performing an auto-correlation on $P(t)$ is described in the section on autocorrelators. The formula used to yield the autocorrelation as a function of the time delay

is given in (2.16). To implement the function in code the fact given in (2.35).

$$\lim_{\Delta t \to 0} \sum_n f(t_n)\Delta t_n = \int_{x_1}^{x_2} f(t)dt \quad (2.35)$$

I therefore take the integral as a finite sum over small intervals Δt, where the summation is over a finite range covering a sufficient number of pulsations. *Mathematica's* `Sum` function is used to calculate the finite sum. This is nested inside a do loop which changes the values of time delay (b). The values are then appended into a list containing $b, ac[b]$ where $ac[b]$ is the value of the autocorrelation at a time delay b. At the end of the loop, the `acoutputfunction`[1] is produced as an interpolating function of the list.

Calculating the power spectrum

The power spectrum is the modulus squared of the Fourier transform of $P(t)$.[18] The Fourier transform of $P(t)$ is calculated from (2.36).

$$g(k) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} P(t)e^{-i\omega t}dt \quad (2.36)$$

To produce the function $g(k)$, a procedure similar to the calculation of the autocorrelation is used. The integral is replaced with a finite sum over small intervals of dt. The summation is again nested inside a `do` loop which increments values of k, in an attempt to produce a continuous function. It was found that the Fourier transform algorithm implemented was extremely sensitive to the total amount of iterations performed. The total amount of iterations is given by (2.37).

$$I_{\text{tot}} = \left(\frac{(t_2 - t_1)}{t_{\text{step}}}\right)\left(\frac{(f_2 - f_1)}{f_{\text{step}}}\right) \quad (2.37)$$

[1] See the *Mathematica* source code in the appendix.

Where t_{step} is the time interval to step through the time range $(t_2 - t_1)$ by each iteration, and similarly for f_{step} (frequency). To obtain a good result in a reasonable amount of time, the variables t_2, t_1, t_{step}, f_2, f_1 and f_{step} had to be "tuned" accordingly. Through extensive testing, the following was determined:

- Smaller time intervals produce less artifacts (spurious results) in the Fourier transform.
- Larger time intervals give between resolution between peaks, *i.e.* a higher peak would appear more prominently amongst a series of shorter peaks.
- Decreasing the frequency step adjusts the sharpness of the peaks and gives more defined peaks for larger ranges.
- Increasing the frequency range controls the ability to pick up peaks over a larger range.

Obtaining large volumes of data

It became increasingly apparent during development that observing in real time how the functions change with modulation frequency and driving current would require a database of acquired data. This is because each set of data, containing $p(t)$, $n_1(t)$, $n_2(t)$, the power spectrum and autocorrelation would take about 80 seconds to calculate.

The idea was to design a sub program which would automatically acquire data across a user defined frequency and current range. Sampled data output, however, would take up too much hard disk space per set (approximately 5 MB per set), since to represent each function accurately would require at least 4000 points

to be taken. The most feasible solution to this problem was to export all the graphed output as GIF image files. The GIF format was found to render the most quickly in *Mathematica* and provide a good quality image, whilst not occupying too much disk space. Once a library of GIF images had been created, it was possible to alter current and frequency as a parameter and allow *Mathematica* to import a corresponding set of image files for display. The function "get data" (`gdexp`) performs the following operations:

1. Asks for current range, frequency range and steps to take over each range per iteration.

2. Executes two do loops, one nested inside the other. The outer loop traverses the current range and the inner loop traverses the frequency range. For every current measurement, the entire frequency range is traversed, then the current is incremented in the outer loop, and the process repeats until the maximum current range is reached.

3. For each current-frequency set of data, all functions are calculated. The graphed output for each is then exported to a GIF file.

4. For each current-frequency "run" an index file is created containing all the variables defined in step 1. This is so that Mathematica knows how to read the data back.

Acquiring large amounts of data ($>$400 sets) consumes a considerable amount of system RAM. Mathematica crashes at the point when RAM runs out. For this reason, it is not possible to acquire a large amount of data in one run. Instead, the calculations are divided into smaller separate composite runs. After each composite run, *Mathematica's* kernel is restarted, and the next composite run commences. The function "continue get data" (`cgd`) divides a single run into sets of composite runs seamlessly and without corrupting the database. It was found that the stability of the computer worsened when data was acquired over particularly long periods of time ($\gtrsim$8 hours). This was most likely due to the CPU overheating. For this reason, *Mathematica* would usually crash midway during a long set of calculations. It was important that, in this event, procedures were implemented which protect the database from corruption and make it possible to resume data acquisition from the point at which *Mathematica* would crash. This would avoid having to re-acquire the data from scratch, which would be very time consuming.

The procedures installed to protect the database against Mathematica crashing are:

- Index file is written per current set. This means that if Mathematica crashes midway, the previous results are not affected. A user could allow the CPU to cool, then later use the `cgd` function to simply specify a maximum current, and the program will resuming acquiring data up to that maximum from the current point at which Mathematica crashed.

- Pulse information and chaos data are also written to a data file per current run. This eliminates the risk of data redundancy (where measurements would be written to the file twice in the even of *Mathematica* crashing). Using `cgd`, the program will load back in the chaos and pulse

data, and resume the data acquisition process.

- The function `cgd` does not allow the user to specify a maximum current range that is less than the point at which the integrity of database has been verified.

Detecting chaos

Chaotic pulsations cannot be distinguished by eye, instead, an algorithm must be employed which quantifies each chaotic output to determine what makes a signal chaotic. To accomplish this, I make use of the auto-correlation function of $P(t)$. If the function $P(t)$ is chaotic, then the probability the autocorrelation function will maximize (equal 3) at any time delay will be negligibly small. This can seen by imagining two completely random pulsations. If the same pulsation overlaps with itself shifted by a small amount then it would be unlikely that the shifted pulsation will be similar to the original pulse because the pulsations exhibit little or no periodicity. Therefore, one can expect a few peaks with heights around 1, or 2 but never 3.

Auto-correlation functions for chaotic pulsations tend to display the following characteristics;

- Few correlated peaks, having a height of less than 3.
- Majority of points lie within a small height range.

Given these characteristics, the algorithm to detect chaos does the following;

- Samples the entire autocorrelated function producing a list of points corresponding to values of the function.
- Sorts the list from higher to lower.
- Averages the highest 6 points with 183^{rd}-189^{th}highest point.
- Computes and stores this average for all frequencies and currents in a specified range.
- The lower the average, the more chaotic the pulsations.

The purpose of averaging the 183rd-189th highest points is part of identifying key characteristic features of a chaotic autocorrelation function. For non-chaotic pulsations, the $183^{\text{rd}}$and 189^{th}highest terms are still sufficiently high. However, for chaotic pulsations, these terms tend to be noticeably smaller than the 6 highest terms. The numbers 183 and 189 are chosen arbitrarily to lie approximately midway in the list.

Obtaining pulse interval and magnitudes for non-chaotic pulses

It is possible to obtain the pulse interval using the autocorrelation function. The pulse interval can only be calculated for non-chaotic pulsations, *i.e.* pulsations created by applying a direct current. To calculate the pulse interval, the temporal distance between two points separated by the period must be measured. The calculation of this heavily exploits a feature of non-chaotic autocorrelation functions which involves the instantaneous rate of change. Despite the appearance of a non-chaotic autocorrelation function, there are in fact no stationary points. This is largely due to the step sizes used for evaluation. However, at a given peak, the instantaneous rate of change of the autocorrelation function is discontinuous; it switches quickly between a positive rate of

change and a negative rate of change. By detecting these unique points, it is possible to locate the positions of the peaks in the autocorrelation. Once the positions have been found, the temporal distance is measured between consecutive peaks, and the pulse interval is calculated.

The algorithm `intervalfinder` therefore does the following:

- Scans the instantaneous rate of change of the autocorrelation function, $A(t)$.
- For a time t, $A(t)$ and $A(t + dt)$ are read in, where dt is a short interval.
- If the sign of $A(t+dt)$ is negative and $A(t)$ is positive, then this represents the location of the peak.
- The point (t_1) is marked and the loop repeats finding the next instance of the above condition.
- When the condition is satisfied again, the point is marked again, t_2.
- The difference $(t_2 - t_1)$ is the pulse interval.

The pulse magnitude is calculated by finding the maximum point of the function $P(t)$. This is found by sampling the function, producing a list of points, and then finding the maximum in that list.

2.5 Results from computer modelling

Authors: M. Moussa and G. Weerasinghe

2.5.1 Comparison of the models

The advantage of simultaneously producing two separate simulations is that it is possible to cross check the outputs for consistency. Comparing the result for photon density from both the *Matlab* and *Mathematica* models (see Figure 2.18), it is clearly seen that there is a consistency with the output from solving the rate equations.

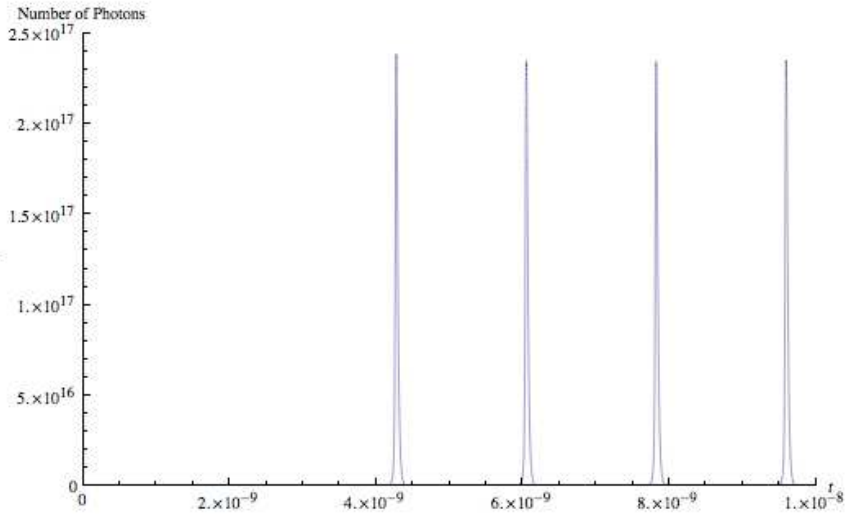
Figure 2.19 shows chaotic output from the *Mathematica* and *Matlab* simulations, respectively. The output was obtained at an input current of 32 mA and a modulation frequency of 2 GHz. The figures show consistency in the solutions to the chaotic rate equations.

Figure 2.20 shows a result obtained from the *Matlab* model that is not available using the *Mathematica* model. It shows a plot of the ratio of the carrier region densities (n_1/n_2). The long tail corresponds to the laser powering up, and beginning to lase when it reaches the circular region. It will remain in this circular loop as the laser pulses periodically.

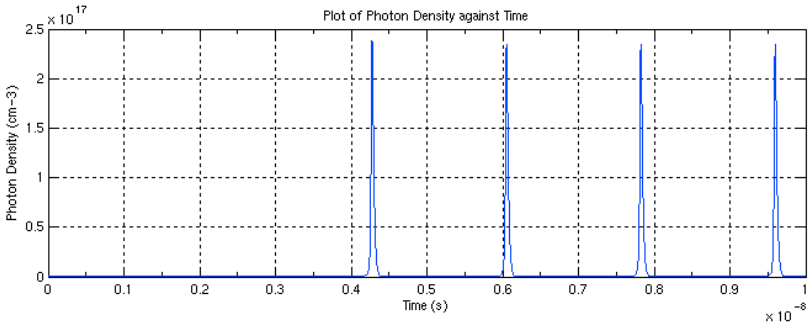
Figure 2.20b shows the case when chaos is introduced into the simulation. Again the laser takes time to lase, but once it does the pattern is significantly different than the periodic lasing. The path no longer traces over itself, instead deviating from the circular trace more and more with each pulse. This pattern is a good alternative way to distinguish between chaotic and non-chaotic laser pulsing.

The *Matlab* model did not progress as far as the *Mathematica* model due to many technical difficulties encountered during its development. These difficulties included;

- Having to write a Runge-Kutta ODE solver from scratch as *Matlab's* `ODE45` function encountered difficulties with the rate equations.
- Problems with the Fast Fourier

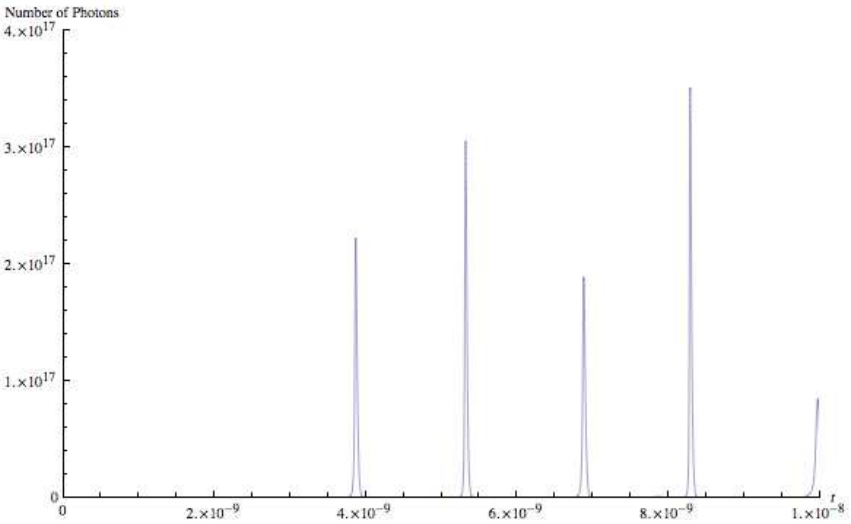


(a) Mathematica

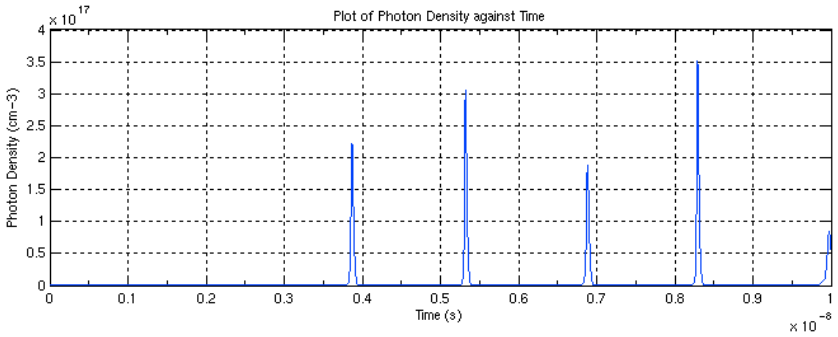


(b) Matlab

Figure 2.18: Graphs of photon density from the *Mathematica* and *Matlab* models respectively.
Created by M. Moussa and G. Weerasinghe.

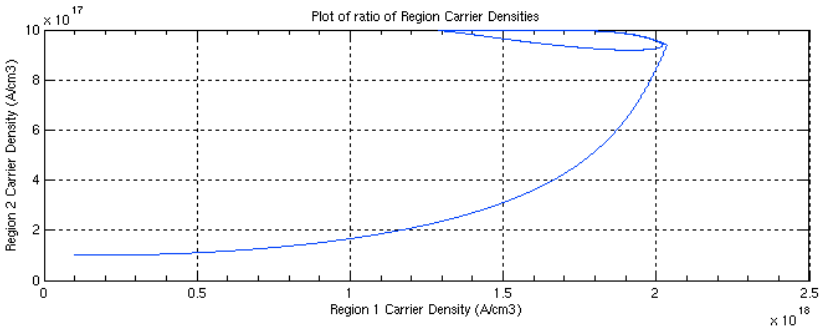


(a) Mathematica

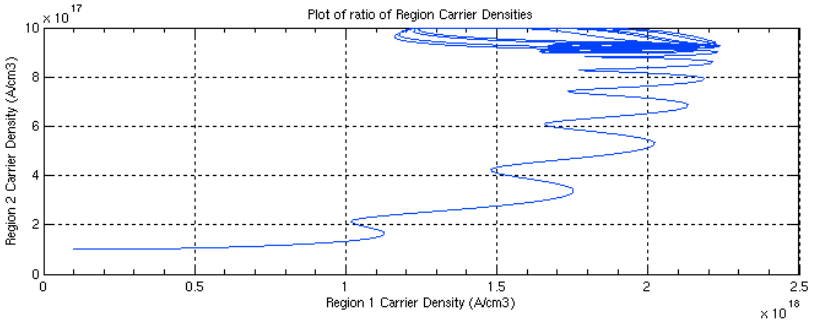


(b) Matlab

Figure 2.19: Chaotic output from the *Mathematica* and *Matlab* models, the input current amplitude was set at 32 mA and the modulation frequency was set at 2 GHz. *Created by M. Moussa and G. Weerasinghe.*



(a) Non-chaotic pulsation



(b) Chaotic pulsation

Figure 2.20: The plot of the ratio of the carrier region densities from the *Matlab* model. The difference between the plots when the laser is exhibiting chaotic and non-chaotic pulsation is easily visible.
Created by M. Moussa and G. Weerasinghe.

Transform (FFT) algorithm implemented in *Matlab* required to produce power spectrums and the auto-correlation function.

- The *Mathematica* simulation made heavy use of approximating definite integrals with finite summations for producing the Fourier Transform and the auto-correlation. An equivalent method was very difficult to implement in the *Matlab* model.

Running times were a factor with the models with both taking a long time to calculate solutions. The *Matlab* model takes a significantly longer time than the *Mathematica* model, due in part to running *Matlab* over *Citrix Metaframe*, rather than natively.

The source code for the models is included in the appendix.

A note on the C++ and Fortran models

Author: L. Pomfrey

Models created using *C++* and *Fortran* were originally planned. Whilst these models would have had the advantage of much faster running times than the *Mathematica* and *Matlab* models, there were a number of disadvantages that resulted in them being scrapped due to the amount of time implementing the necessary functionality would have taken, these included;

- The lack of many mathematical functions already present in the *Mathematica* and *Matlab* environments. For this reason a pre-packaged mathematical library of basic functions was used in the *C++* model.[25]
- The lack of a native graphical plotting library. Utilising the FLOSS graphing program *Gnuplot* was considered for this.
- Having to write, from scratch, both Runge-Kutta and Fast Fourier Transform methods.

The *C++* Runge-Kutta method is included in the appendix.

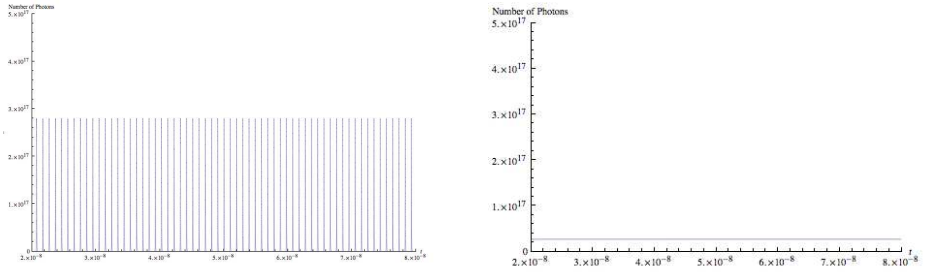
2.5.2 Mathematica output

Solutions to the rate equations

The solutions to the rate equations clearly show that driving a semiconducting laser diode with a DC current will create self pulsation. The rate of pulsation is found be proportional to the applied current. When a modulation frequency is applied, the simulation shows that at certain frequencies the pulsations become chaotic. The frequency at which this occurs is dependent on the driving amplitude of the current and hence the rate at which the laser would naturally pulsate without the modulation. The simulation predicts that for self pulsation to occur, the applied current must be between 29.6 and 41.65 mA.

Results from auto-correlation

The results from the simulated autocorrelation agree well with predictions made from theory. It was found that non-chaotic self pulsations correlated consistently, which was expected due to their periodicity. Chaotic pulsations were found to be consistently lower, which agreed well with the theory that chaotic pulsations have little or no periodicity. The pulse intervals were found to be between 1.41 ns and 0.557 ns for 29.7 and 41 mA respectively.



(a) (b)

Figure 2.21: The simulation predicts the self pulsation of a laser diode. The figure shows that when a DC bias is applied, the laser diode will not emit a continuous stream of light, but pulse packets of photons instead. When a DC bias current above 41.65 mA is applied, the pulsations cease. No output from the laser is obtained below a bias current of 29.6 mA.
Created by G. Weerasinghe.

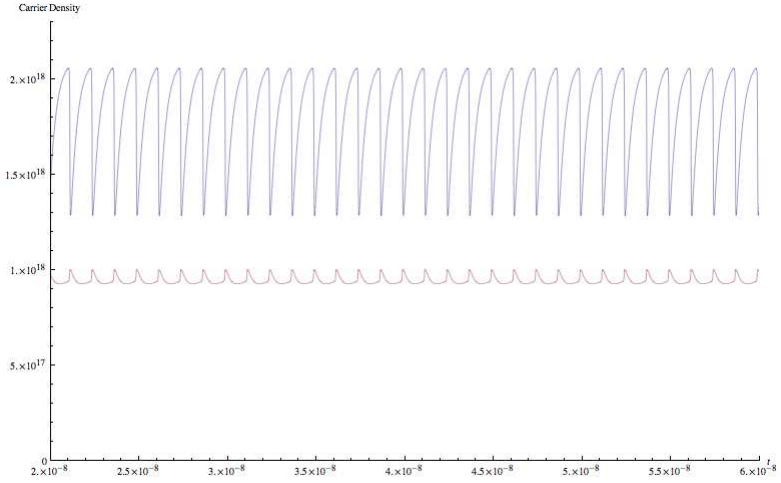


Figure 2.22: The plots of the two carrier densities in regions 1 and 2. The smaller amplitude plot represents the unbiased region 2.
Created by G. Weerasinghe.

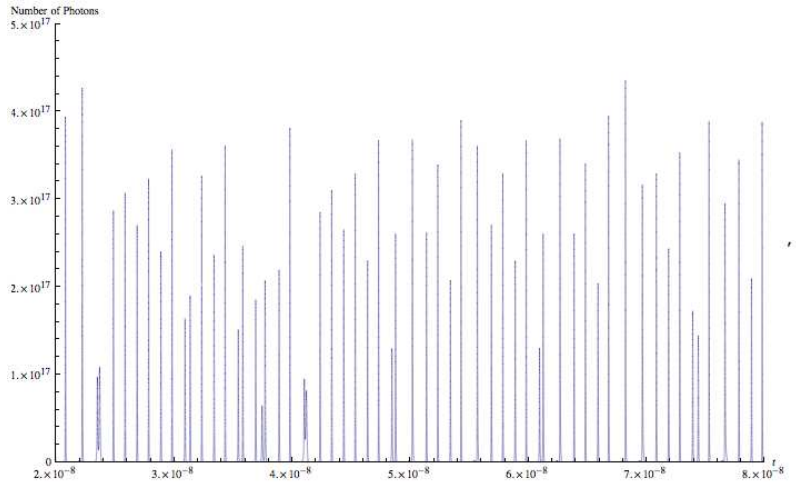


Figure 2.23: An example of chaotic output.
Created by G. Weerasinghe.

Results from the power spectrum

Results from the power spectrum clearly show peaks at angular frequencies corresponding roughly to $(2\pi)/(T)$, where T is the pulse interval calculated from the autocorrelation. Smaller peaks can be seen at other angular frequencies, however, this is typical of such waveforms that are not completely sinusoidal. As the current is increased, the peak is shifted to the right. The height of the peak also marginally increases indicating that more energy is being concentrated in a single frequency.

Results from chaos detection

The aim of the chaos detection algorithm is to find the current and the frequency which will generate the most chaotic pulsations from the laser. The first results from the chaos detection were obtained from data over the current range of 29.5-42 mA in steps of 0.5 mA and a modulation frequency range of 0 to 10 GHz in steps of 500 MHz. The result from applying the chaos detection algorithm showed that maximum chaos is obtained at 32.5 mA and 5 GHz modulation frequency.

The second set of results were taken around the region of maximum chaos found in the first set. The aim was to search for more chaotic regions over this particularly chaotic range. The results were taken at 32.5 mA over a modulation frequency range of 4GHz to 6 GHz in steps of 100 MHz. The chaos detection algorithm showed that maximum chaos was obtained at 32.5 mA and 5 GHz modulation frequency.

The final set of results were taken over an even narrower range, focussing particularly on this area of chaos. They were taken at 32.5 mA over a modulation frequency range of 4.9 GHz to 5 GHz, in steps of 4 MHz. The chaos detection algorithm showed that maximum chaos was

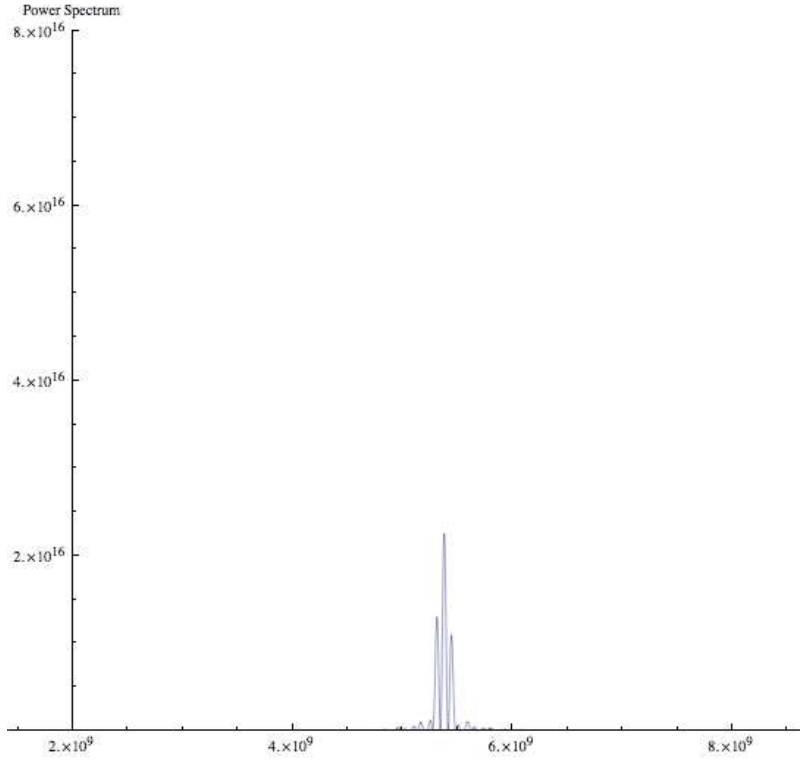
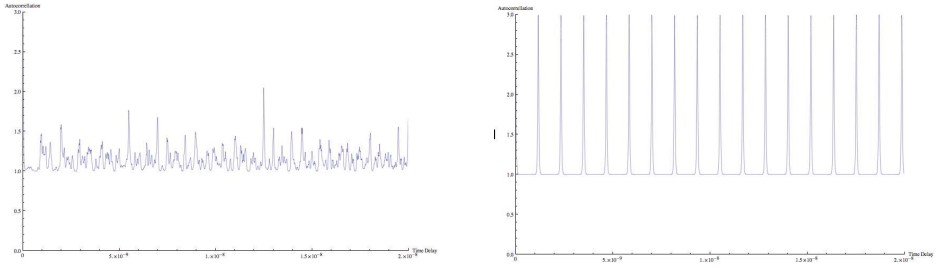


Figure 2.24: A typical power spectrum.
Created by G. Weerasinghe.

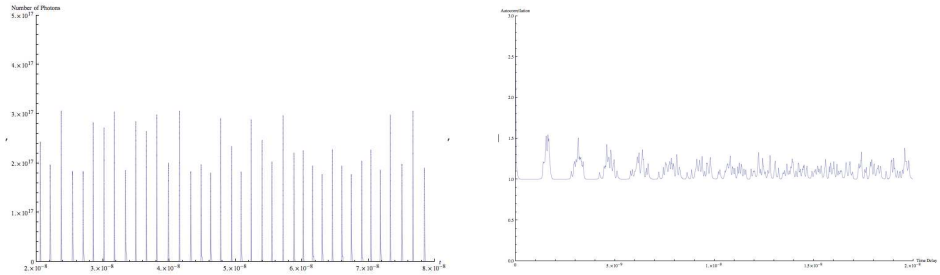


(a) Chaotic (b) Non-chaotic

Figure 2.25: The auto-correlation function for chaotic and non-chaotic pulses respectively.
Created by G. Weerasinghe.

once again found at 32.5 mA and 5 GHz modulation frequency.

To conclude, unanimously, the results from the chaos detection predict that a maximum chaotic output would be obtained from a semiconducting laser diode if driven with 32.5 mA modulated at a frequency of 5 GHz.



(a) Output (b) Power spectrum

Figure 2.26: Results taken at the point of maximum chaotic output, as determined by a chaos detection algorithm. This point is taken to be at 32.5 mA at a modulation frequency of 5 GHz.
Created by G. Weerasinghe.

Appendix A: Computer modelling source code

Matlab

RungeKuttaFixedGUIVersion.m

```
function [T,X] = RungeKuttaFixedGUIVersion(ODEFunctions, TimeSpan, InitialValues, StepSize, J

NumOfSteps = (TimeSpan(2) - TimeSpan(1)) / StepSize;
HalfStepSize = 0.5 * StepSize;
NumOfEquations = size(InitialValues);
X = zeros(NumOfEquations(1), NumOfSteps); % Creates a vector to store values at each step
T = zeros(1, NumOfSteps); % creates a vector to store time values
X(:,1) = InitialValues; % Inserts initial values into data vector
T(1) = TimeSpan(1); % Inserts initial time into time vector

Td = TimeSpan(1); % Set current time step
Xd = InitialValues; % Set current values

for i = 2:NumOfSteps,% Loop for evaluating each step

 K1 = feval(ODEFunctions,Td,Xd,J);

 Thalf = Td + HalfStepSize;
 Xtemp = Xd + HalfStepSize * K1;

 K2 = feval(ODEFunctions, Thalf, Xtemp, J);

 Xtemp = Xd + HalfStepSize * K2;

 K3 = feval(ODEFunctions, Thalf, Xtemp, J);

 Tfull = Td + StepSize;

 Xtemp = Xd + StepSize * K3;

 K4 = feval(ODEFunctions, Tfull, Xtemp, J);
```

```
 X(:,i) = Xd + StepSize * (K1 + 2.0 * (K2 + K3) + K4) / 6;

 T(i) = Tfull;

 Xd = X(:,i);
 Td = T(i);

end

X = X';
T = T';
```

LASERsimulation.m

```
function varargout = LASERSimulation(varargin)
% LASERSimulation Application M-file for LASERSimulation.fig
%   LASERSimulation, by itself, creates a new LASERSimulation or raises the existing
%   singleton*.
%
%   H = LASERSimulation returns the handle to a new LASERSimulation or the handle to
%   the existing singleton*.
%
%   LASERSimulation('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in LASERSimulation.M with the given input arguments.
%
%   LASERSimulation('Property','Value',...) creates a new LASERSimulation or raises the
%   existing singleton*.  Starting from the left, property value pairs are
%   applied to the GUI before LASERSimulation_OpeningFunction gets called.  An
%   unrecognized property name or invalid value makes property application
%   stop.  All inputs are passed to LASERSimulation_OpeningFcn via varargin.
%
%   *See GUI Options - GUI allows only one instance to run (singleton).
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help LASERSimulation

% Copyright 2001-2006 The MathWorks, Inc.

% Last Modified by GUIDE v2.5 27-Jan-2008 00:53:32

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',          mfilename, ...
```

```
                  'gui_Singleton',     gui_Singleton, ...
                  'gui_OpeningFcn',    @LASERSimulation_OpeningFcn, ...
                  'gui_OutputFcn',     @LASERSimulation_OutputFcn, ...
                  'gui_LayoutFcn',     [], ...
                  'gui_Callback',      []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before LASERSimulation is made visible.
function LASERSimulation_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to LASERSimulation (see VARARGIN)

% Choose default command line output for LASERSimulation
handles.output = hObject;

axes(handles.power_plot)
cla
axes(handles.n1_n2_plot)
cla

% Update handles structure
 guidata(hObject, handles);

% UIWAIT makes LASERSimulation wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = LASERSimulation_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Get default command line output from handles structure
varargout{1} = handles.output;


% --------------------------------------------------------------------
function plot_button_Callback(hObject, eventdata, handles, varargin)
% hObject    handle to plot_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get user input from GUI
j_value = str2double(get(handles.j_value_input,'String'));
t_value = str2double(get(handles.t_value_input,'String'));

% Calculate data
[t y] = RungeKuttaFixedGUIVersion('LaserModelGUIVersion', [0.0;t_value],[100000000000000000;1

% Create power plot plot
axes(handles.power_plot)
plot(t,y(:,3))
xlabel('Time (s)')
ylabel('Photon Density (cm-3)')
title('Plot of Photon Density against Time')
set(handles.power_plot,'XMinorTick','on')
grid on

% Create n1_n2 plot
axes(handles.n1_n2_plot)
plot(y(:,1),y(:,2))
xlabel('Region 1 Carrier Density (A/cm3)')
ylabel('Region 2 Carrier Density (A/cm3)')
title('Plot of ratio of Region Carrier Densities')
set(handles.n1_n2_plot,'XMinorTick','on')
grid on
```

LaserModelGUIVersion.m

```
function dydt = LaserModelGUIVersion(t,y,j)

dydt = zeros(size(y));

e = 0.00000000000000000016;
V = 0.000000000072;
B1 = 0.0000000003;
tau12 = 0.0000000015;
```

```
tau21 = 0.0000000025;
gamma1 = 0.1;
gamma2 = 0.2;
N01 = 1200000000000000000;
N02 = 1000000000000000000;
nugp = 7500000000;
g1 = 0.00000308 / nugp;
g2 = 4 * g1;
alpha = 10;
beta = 0.00001;
J = j/1000;

A = y(1);
B = y(2);
C = y(3);

dydt(1) = (J / (e * V)) - (A / CarrierLifetime(A)) - ((A - B) / tau12) - (gamma1 * nugp * g1
dydt(2) = (-B / CarrierLifetime(B)) - ((B - A) / tau21) + (gamma2 * nugp * g2 * abs(B - N02)
dydt(3) = (nugp * ((gamma1 * g1 * abs(A - N01)) - (gamma2 * g2 * abs(B - N02)) - alpha) * C)
```

CarrierLifetime.m

```
function lifetime = CarrierLifetime(Ni)

A = 100000000;
B = 0.0000000003;
C = 0.00000000000000000000000000007;

lifetime = (A + (B * Ni) + (C * (Ni.^2))).^(-1);
```

Mathematica

```
Needs["GUIKit`"];
Off[NDSolve::"mxst"];
Off[General::"spell1"];
Off[NIntegrate::"slwcon"];
Off[NIntegrate::"ncvb"];
Off[ReleaseGUIObject::"shdw"]


ac := {
```

```
  temp = ProgressDialog[], acexp[0], ReleaseGUIObject[temp]

  }



acatm := acexp[1]



sv := {

  modfreq = Input["Add modulation frequency? (Y for yes, N for no)"],


  If[Or[modfreq == Y,
    modfreq == y], {f = Input["Set modulation frequency (in Hz)"]}, ,
   f = 0],

  J = Input["Set Bias Current (in mA)"],
  loadequations,
  Print["Variables and equations loaded. Type rs to run the \
simulation"]

  }
acexp[iden_] := {

  Clear[result],

  result = {{-1, 0}},

  tlint = 3*10^-8,

  thint = 6*10^-8,

  delt = (thint - tlint)/1000,

  recur = 2*10^-8,

  ste = recur/1800,

  powtemp = Table[{t, pow[t]}, {t, 2*10^-8, 8*10^-8, 0.1*10^-11}],
  pow2 = Interpolation[powtemp],

  nm = (Sum[(pow2[t])^2*(delt), {t, tlint, thint, delt}]),
```

```
   Do[

   {
    If[iden == 0,
     progressbar["Calculating..", "Autocorrelating...", recur,
      b], ,],

    lk = (Sum[(pow2[t]*pow2[t + b])*(delt),

       {t, tlint, thint, delt}]),


    h = ((2*lk/nm) + 1),


    result = Append[result, {b, N[h]}]},


   {b, 0, recur, ste}],


  acoutputfunction = Interpolation[result];,


  acoutputgraph =
    Plot[acoutputfunction[x], {x, 0, recur},
     PlotRange -> {{0, recur}, {0, 3}}, PlotPoints -> 2000,
     AxesLabel -> {"Time Delay", "Autocorrellation"}];


  }

start := {

  Print["**************** Chaotic Optical Pulse Simulator \
***************"],
  Print["***************************** by \
*****************************"],
  Print["********************** Gihan \
Weerasinghe************************"],
  Print["***************************** 2008 \
*****************************"],
  Print[""],
```

```
  Print[""],

  A = 1*10^8,
  B = 3*10^-10,
  H = 7*10^-29,

  e = 1.6*10^-19,
  V = 7.2*10^-11,

  k12 = 1.5*10^-9,
  k21 = 2.5*10^-9,

  r1 = 0.1,
  r2 = 0.2,

  n01 = 1.2*10^18,
  n02 = 1*10^18,

  v = 7.5*10^9,

  alpha = 10,
  bertay = 1*10^-5,
  g1 = ((3.08*10^-6)/v),
  g2 = (4*g1),
  delt = 5*10^-11,
  lowint = 0*10^-8,
  highint = 8*10^-8,
  sedat = Input["Enter data set number"],
  sedat = ToString[sedat],
  mywrkingdir = "/Users/Gihan/CPS Data/" <> "Set " <> sedat,
  SetDirectory[mywrkingdir],

  sv




  }




loadequations := {
  J = J*10^-3,
```

```
  k[y_] = (A + (B*y[t]) + (H*(y[t])^2))^-1,

  eq1 = y'[
     t] == (J/(e*V)) + ((J*(Sin[(2*Pi*f)*t]))/(e*V)) - ((y[t])/(k[
         y])) - (((y[t] - x[t]))/(k12)) - (r1*v*
       g1*(Abs[(y[t] - n01)])*P[t]),

  eq2 = x'[
     t] == -(x[t]/k[x]) - ((x[t] - y[t])/k21) + (r2*v*
       g2*(Abs[(x[t] - n02)])*P[t]),

  eq3 = P'[
     t] == (v*P[t]*r1*g1*Abs[(y[t] - n01)]) - (v*r2*g2*P[t]*
       Abs[(x[t] - n02)]) - (v*alpha*P[t]) + ((bertay)*B*(y[t])^2)

  }

rsexp[iden_] :=
 {
  Solution =
   NDSolve[{eq1, y[0] == 1*10^17, eq2, x[0] == 1*10^17, eq3,
     P[0] == 0}, {y, x, P}, {t, 0, 1*10^-7},
    StartingStepSize -> 5*10^-13, MaxStepSize -> 5*10^-13,
    Method -> {"FixedStep", Method -> "ExplicitRungeKutta"},
    MaxSteps -> 300000],

  pow[t_] = Evaluate[P[t] /. Solution],

  n1[t_] = Evaluate[y[t] /. Solution],

  n2[t_] = Evaluate[x[t] /. Solution],

  Intensity[t_] = pow[t]
  }

rs := {temp = ProgressDialog[],
  progressbar["Solving Rate Equations...", "Calculating...", 1,
   0.9999], rsexp[0], ReleaseGUIObject[temp]}




sg :=

 {Print["For ", (N[J]*1000), " mA"],
```

```
  If[f == 0, , Print["And ", f, " Hz Modulation Frequency" ],
   Print["And ", f, " Hz Modulation Frequency" ]],

  Print[""],

  powerplot =
   Plot[pow[t], {t, 0, 8*10^-8},
    PlotRange -> {{2*10^-8, 8*10^-8}, {0, 50*10^16}},
    AxesLabel -> {t, "Number of Photons"}, PlotPoints -> 2000],

  n1plot =
   Plot[n1[t], {t, 0, 6*10^-8}, AxesLabel -> {t, "N1"},
    PlotPoints -> 2000],

  n2plot =
   Plot[n2[t], {t, 0, 6*10^-8}, AxesLabel -> {t, "N2"},
    PlotPoints -> 2000]


  }
help := {

  Print["*************** Chaotic Optical Pulse Simulator Help \
***************"],
  Print[""],
  Print[""],




  Print["sv  -> Set Variables. If the bias current or modulation \
frequency need changing or toggling, sv can be used to change them"],
  Print[""],
  Print[""],
  Print["start  -> The Start function must be run initially before \
beginning the simulation for the first time. Start loads inital \
variables and equations, avoiding the need repeatedly reload them. \
"],

  Print[""],
  Print[""],

  Print["sg  -> Show Graphs. After running the simulation, you may \
wish to see plots of the various Runge Kutta solutions. The sg \
function will display these."],
```

```
  Print[""],
  Print[""],

  Print["rs  -> Run Simulation. Runs the simulation obtaining the \
required Runge Kutta solutions for the specified bias current and \
modulation frequency. The bias current and modulation frequency \
require setting first either by using start or sv"],

  Print[""],
  Print[""],

  Print["ac[Delay Steps, Delay Range]  -> Performs Autocorrelaton for \
the last Runge Kutta solutions obtainied using rs (runsimulation \
function). User adds arguements to ac for a stepping delay interval, \
i.e. by how much delta t should be changed after each iteration and \
over which range to perform the autocorrelation. "],

  Print[""],
  Print[""],

  Print["ps  -> Calculates Power Spectrum for the last Runge Kutta \
solutions obtainied using rs (runsimulation function)."],

  Print[""],
  Print[""],

  Print["ic  -> Interval Change. Overrides the default setting for \
the interval over which the autocorrelation and power spectrum are \
calculated"],

  Print[""],
  Print[""],

  Print["gd  -> Get Data. Automatically gathers full set of data over \
a specified frequency and bias current range. Saves GIF snapshots of \
the output in working directory, creating a library of data"],

  Print[""],
  Print[""],

  Print["cgd  -> Continue Get Data. Continues acquiring data over a \
previous working range. Useful for dividing long calculations up into \
separate runs, or recovering from a system crash."],
```

```
  Print[""],
  Print[""],

  Print["changeset  -> Changes data set folder. A number (n) should \
be entered corresponding to a data folder present in the working \
directory set n. GIF output will be saved in this folder "]


  }



progressbar[title_, caption_, tot_, ita_] :=

 {
  inc = IntegerPart[N[(ita/tot)*100]],
  temp @ SetPropertyValue[{"bar", "value"}, inc],
  temp @ SetPropertyValue[{"frame", "title"}, title],
  temp @ SetPropertyValue[{"label", "text"}, caption],

  }






ProgressDialog[] := GUIRun[
  Widget["Frame", {
    WidgetGroup[{
      Widget["Label", {"text" -> "Percent complete:"},
       Name -> "label"], Widget["ProgressBar",
       {"minimum" -> 0, "maximum" -> 100,
        "preferredSize" ->
           Widget["Dimension", {"width" -> 300, "height" -> 25}]},
       Name -> "bar"]
        }, WidgetLayout -> {
       "Grouping" -> Column,
       "Border" -> {{15, 15}, {25, 20}}}],
    "location" -> Widget["Point", {"x" -> 400, "y" -> 400}],
    "title" -> "Computation Progress",
    "resizable" -> False},
   Name -> "frame"]]
psexp[iden_] :=
```

```
{

 nmcon = (1/(2*Pi)^0.5),


 Clear[discresult],
 powtemp = Table[{t, pow[t]}, {t, 2*10^-8, 7*10^-8, 0.1*10^-11}],
 pow2 = Interpolation[powtemp],

 discresult = {{-1, 0}},

 tlint = 2.5*10^-8,

 thint = 6.5*10^-8,

 flint = 7*10^8,

 fhint = 11*10^9,

 fcount = 4250,

 fint = (fhint - flint)/fcount,

 tcount = 1400,

 delt = (thint - tlint)/tcount,

 Do[ {

   If[iden == 0,
    progressbar["Calculating...", "Calculating Power Spectrum...",
     fhint, k], ,],

   discfourier =
    nmcon*(Sum[
       pow2[t]*(Exp[(-I*k*t)])*delt, {t, tlint, thint, delt}]),

   discresult = Append[discresult, {k, N[discfourier]}]},


  {k, flint, fhint, fint}],

 psoutputresult = Interpolation[Re[discresult]],
 psoutputgraph =
```

```
   Plot[(Abs[psoutputresult[k]])^2, {k, flint, fhint},
    PlotRange -> {{flint, fhint}, {0, 0.8*10^17}}, PlotPoints -> 2000,
     Axes -> True, AxesLabel -> {"Ang.Frequency", "Power Spectrum"}]

  }
ps := {temp = ProgressDialog[], psexp[0], ReleaseGUIObject[temp]

  }

psatm := {

  psexp[1]

  }


ic := {

  lowint =
   Input["Type lower interval to evaluate Runge Kutta solutions"],

  highint =
   Input["Type higher interval to evaluate Runge Kutta solutions"]

  }
WriteImage :=

 {imgdimens = {300, 300},
  mkfilename[jloop, floop],

  powerplot =
   Plot[pow[t], {t, 0, 8*10^-8},
    PlotRange -> {{2*10^-8, 8*10^-8}, {0, 50*10^16}},
    AxesLabel -> {t, "Number of Photons"}, PlotPoints -> 2000],

  n1plot =
   Plot[n1[t], {t, 2*10^-8, 8*10^-8}, AxesLabel -> {t, "N1"},
    PlotPoints -> 2000],

  n2plot =
   Plot[n2[t], {t, 2*10^-8, 8*10^-8}, AxesLabel -> {t, "N2"},
    PlotPoints -> 2000],


  Export["ac" <> filname, acoutputgraph, "GIF",
```

```
   ImageSize -> imgdimens],

  Export["ps" <> filname, psoutputgraph, "GIF",
   ImageSize -> imgdimens],

  Export["n1" <> filname, n1plot, "GIF", ImageSize -> imgdimens],

  Export["n2" <> filname, n2plot, "GIF", ImageSize -> imgdimens],

  Export["p" <> filname, powerplot, "GIF", ImageSize -> imgdimens]
  }

gatherdata :=

 {timemax = N[(itercount*realtime)/(3600), 1],
  hrpart = IntegerPart[timemax],
  secpart = (FractionalPart[timemax])*3600,
  hrpart = ToString[hrpart],
  secpart = ToString[secpart],

  strinfo =
   ToString[jloop] <> " mA" <> " and " <> ToString[f] <> " Hz   ",
  progressbar["Gathering Data... " <> strinfo,
   "Approx Time Left: " <> hrpart <> " hrs " <> secpart <> " secs",
   itermax, incoun],
  rsexp[1];,
  acatm;,
  chaoscalc;,
  psatm;,
  If[floop == 0, scanpulse],
  WriteImage;


  }
gd :=
 {jmin = Input["Enter minimum bias current boundary:"],
  jminold = jmin,
  jmax = Input["Enter maximum bias current boundary:"],
  jstep = Input["Increment J in steps of ?"],
  fmin = Input["Enter minimum frequency boundary"],
  fmax = Input["Enter maximum frequency boundary"],
  fstep = Input["Increment f in steps of?"],
  incoun = 0,
  gdexp;
  }
```

```
cgd :=

 {lisdat2 = Get[mywrkingdir <> "/index.cps"],
  jminold = Extract[lisdat2, 1],
  jmin = Extract[lisdat2, 2],
  jstep = Extract[lisdat2, 3],
  fmin = Extract[lisdat2, 4],
  fstep = Extract[lisdat2, 6],
  fmax = Extract[lisdat2, 5],

  jmax = Input["Enter maximum bias current boundary:"],
  If[jmax > jmin, {jmin = jmin + jstep,
    chaoslist = Get["chaosdata.cps"],
    pulselist = Get["pulsedata.cps"],
    incoun = 0.0001,
    gdexp;},
   Print["Error: Jmax cannot be less than previous maximum value!"]]

  }

gdexp := {


  realtime = 85,

  temp = ProgressDialog[],



  If[Not[fmax == 0],

   {itermax = (((jmax - jmin)/jstep) + 1)*(((fmax - fmin)/fstep) + 1),
     itercount = (itermax) - incoun,
    Do[{
      J = jloop,

      Do[
       {f = floop,

        loadequations,

        realtime = Extract[Timing[gatherdata], 1],
        incoun = incoun + 1,

        itercount = (itermax) - incoun,
```

```
        J = jloop},
       {floop, fmin, fmax, fstep}], WriteIndex[jloop], finishup},

     {jloop, jmin, jmax, jstep}

     ]}, ,],


  If[fmax == 0, {f = 0, floop = 0,
    itermax = (((jmax - jmin)/jstep) + 1),
    itercount = (itermax) - incoun,
    Do[{J = jloop, loadequations,
      realtime = Extract[Timing[gatherdata], 1], incoun = incoun + 1,
      WriteIndex[jloop], finishup,
      itercount = (itermax) - incoun}, {jloop, jmin, jmax,
      jstep}]}, ,],


  ReleaseGUIObject[temp]




  }
mkfilename[J_, f_] :=

 {inpar = ToString[IntegerPart[J]],
  frpar = ToString[FractionalPart[J]],
  dig = StringLength[frpar] - 2,
  frpar = StringTake[frpar, -dig],
  frepar = ToString[f],
  filname = inpar <> frpar <> frepar <> ".gif",

  }
exploredata :=
 {

  lisdat = Get[mywrkingdir <> "/index.cps"],
  jmin = Extract[lisdat, 1],
  jmax = Extract[lisdat, 2],
  jstep = Extract[lisdat, 3],
  fmin = Extract[lisdat, 4],
  fmax = Extract[lisdat, 5],
  fstep = Extract[lisdat, 6],
```

```
  jmin = N[jmin],
  dispdim = {200, 200},

  If[Not[fmax == 0],
   Manipulate[{mkfilename[cur, fr];,
     Import["p" <> filname, ImageSize -> dispdim],
     Import["ps" <> filname, ImageSize -> dispdim],
     Import["ac" <> filname, ImageSize -> dispdim],
     Import["n1" <> filname, ImageSize -> dispdim],
     Import["n2" <> filname, ImageSize -> dispdim]}, {{cur, jmin,
      "Bias Current (mA)"}, jmin, jmax,
     jstep}, {{fr, fmin, "Frequency (Hz)"}, fmin, fmax, fstep}], ,],

  If[fmax == 0,
   Manipulate[{mkfilename[cur, 0];, Import["p" <> filname],
     Import["ps" <> filname], Import["ac" <> filname],
     Import["n1" <> filname],
     Import["n2" <> filname]}, {{cur, jmin, "Bias Current (mA)"},
     jmin, jmax, jstep}], ,]



  }


changeset :=

 {
  sedat = Input["Enter data set number"],
  sedat = ToString[sedat],
  mywrkingdir = "/Users/Gihan/CPS Data/" <> "Set " <> sedat,
  SetDirectory[mywrkingdir]

  }


chaosdetect :=

 {

  chdata = Get["chaosdata.cps"],
  len = Length[chdata],
  a = Extract[chdata, {1, 1}],
  jch = Extract[chdata, {1, 2}],
  fch = Extract[chdata, {1, 3}],
```

```
  Do[{

    b = Extract[chdata, {n, 1}],

    If[b < a, {a = b, jch = Extract[chdata, {n, 2}],
      fch = Extract[chdata, {n, 3}]}, ,]

    },

   {n, 2, len}],

  curresult = ToString[jch],
  freqresult = ToString[fch],


  Print["Maximum Chaos Occurs At: "],
  Print[curresult <> " mA"],
  Print[freqresult <> " Hz"]







  }
intervalfinder :=

 {time1 = 0, delz = 1*10^-12,
  ratechang[
     z_] = (acoutputfunction[z + delz] -
       acoutputfunction[z])/(delz);,
  Do[

   If[And[

     Sign[ratechang[z]] == 1,

     Sign[ratechang[z + delz]] == -1],


    {If[

      time1 == 0, time1 = z + delz, {time2 = (z + delz), Break[]}]}
```

```
    ],

   {z, 1*10^-8, 2*10^-8, delz}


   ],

  pulseinterval = time2 - time1


  }

chaoscalc :=

 {sampac =
   Table[acoutputfunction[t], {t, 1*10^-8, 2*10^-8, 1*10^-12}],
  sampac = Flatten[sampac],
  sampac = Sort[sampac, Greater],
  samp1 = Extract[sampac, 1],
  samp2 = Extract[sampac, 2],
  samp3 = Extract[sampac, 3],
  samp4 = Extract[sampac, 4],
  samp5 = Extract[sampac, 5],
  samp6 = Extract[sampac, 6],
  samp7 = Extract[sampac, 183],
  samp8 = Extract[sampac, 184],
  samp9 = Extract[sampac, 185],
  samp10 = Extract[sampac, 186],
  samp11 = Extract[sampac, 194],
  samp12 = Extract[sampac, 200],
  avrg = ((samp1 + samp2 + samp3 + samp4 + samp5 + samp6 + samp7 +
       samp8 + samp9 + samp10 + samp11 + samp12)/12),

  If[incoun == 0, chaoslist = {{avrg, jloop, floop}},
   chaoslist = Append[chaoslist, {avrg, jloop, floop}]]

  }

scanpulse :=

 {maxpulse,
  intervalfinder,
  If[incoun == 0, pulselist = {{maxfunc, pulseinterval, jloop}},
   pulselist = Append[pulselist, {maxfunc, pulseinterval, jloop}]]
```

```
  }
maxpulse :=

 {func = Table[pow[t], {t, 3*10^-8, 7*10^-8, 1*10^-12}];,
  maxfunc = Max[func]
  }
finishup :=

 {Put[chaoslist, "chaosdata.cps"],
  Put[pulselist, "pulsedata.cps"]
  }
MultiRunge[h_, x0_, y0_, p0_] :=

 {

  f1[x_, y_, P_] = Extract[VectorX, 1],
  f2[x_, y_, P_] = Extract[VectorX, 2],
  f3[x_, y_, P_] = Extract[VectorX, 3], xresult = {{0, x0}},
  yresult = {{0, y0}}, presult = {{0, p0}}, wx = x0, wy = y0,
  wp = p0,

  Do[

   {

    a1 = f1[wx, wy, wp],
    a2 = f2[wx, wy, wp],
    a3 = f3[wx, wy, wp],

    b1 = f1[wx + ((h/2)*a1), wy + ((h/2)*a2), wp + ((h/2)*a3)],
    b2 = f2[wx + ((h/2)*a1), wy + ((h/2)*a2), wp + ((h/2)*a3)],
    b3 = f3[wx + ((h/2)*a1), wy + ((h/2)*a2), wp + ((h/2)*a3)],

    c1 = f1[wx + ((h/2)*b1), wy + ((h/2)*b2), wp + ((h/2)*b3)],
    c2 = f2[wx + ((h/2)*b1), wy + ((h/2)*b2), wp + ((h/2)*b3)],
    c3 = f3[wx + ((h/2)*b1), wy + ((h/2)*b2), wp + ((h/2)*b3)],

    d1 = f1[wx + ((h/2)*c1), wy + ((h/2)*c2), wp + ((h/2)*c3)],
    d2 = f2[wx + ((h/2)*c1), wy + ((h/2)*c2), wp + ((h/2)*c3)],
    d3 = f1[wx + ((h/2)*c1), wy + ((h/2)*c2), wp + ((h/2)*c3)],

    wy = wy + ((h/6)*(a1 + (2*b1) + (2*c1) + d1)),
    wx = wx + ((h/6)*(a2 + (2*b2) + (2*c2) + d2)),
    wp = wp + ((h/6)*(a3 + (2*b3) + (2*c3) + d3)),
```

```
    t = (n*h),

    xresult = Append[xresult, {t, wx}],
    yresult = Append[yresult, {t, wy}],
    presult = Append[presult, {t, wp}]

    }, {n, 1, 80000}],

  n1 = Interpolation[xresult],
  n2 = Interpolation[yresult],
  pow = Interpolation[presult]


  }

WriteIndex[jloop_] :=

 {stodat = {jminold, jloop, jstep, fmin, fmax, fstep},
  fildrop = mywrkingdir <> "/index.cps",
  Put[stodat, fildrop]
  }
```

A C++ Runge-Kutta method

rungekuttasys.h

```
#ifndef _rungekuttasys_h
#define _rungekuttasys_h

#include "ap.h"


void solvesystemrungekutta(const double& x,
     const double& x1,
     const int& m,
     const int& n,
     ap::real_1d_array& y);


void solvesystemrungekuttastep(const double& x,
     const double& h,
     const int& n,
     ap::real_1d_array& y);
```

```
#endif
```

rungekuttasys.cpp

```
#include <stdafx.h>
#include "rungekuttasys.h"

void solvesystemrungekutta(const double& x,
     const double& x1,
     const int& m,
     const int& n,
     ap::real_1d_array& y)
{
    double h;
    int i;

    for(i = 0; i <= m-1; i++)
    {
        solvesystemrungekuttastep(x+i*(x1-x)/m, (x1-x)/m, n, y);
    }
}


void solvesystemrungekuttastep(const double& x,
     const double& h,
     const int& n,
     ap::real_1d_array& y)
{
    int i;
    ap::real_1d_array yt;
    ap::real_1d_array k1;
    ap::real_1d_array k2;
    ap::real_1d_array k3;
    ap::real_1d_array k4;

    yt.setbounds(1, n);
    k1.setbounds(1, n);
    k2.setbounds(1, n);
    k3.setbounds(1, n);
    k4.setbounds(1, n);
    for(i = 1; i <= n; i++)
    {
```

```
        k1(i) = h*f(i, x, y);
    }
    for(i = 1; i <= n; i++)
    {
        yt(i) = y(i)+0.5*k1(i);
    }
    for(i = 1; i <= n; i++)
    {
        k2(i) = h*f(i, x+h*0.5, yt);
    }
    for(i = 1; i <= n; i++)
    {
        yt(i) = y(i)+0.5*k2(i);
    }
    for(i = 1; i <= n; i++)
    {
        k3(i) = h*f(i, x+h*0.5, yt);
    }
    for(i = 1; i <= n; i++)
    {
        yt(i) = y(i)+k3(i);
    }
    for(i = 1; i <= n; i++)
    {
        k4(i) = h*f(i, x+h, yt);
    }
    for(i = 1; i <= n; i++)
    {
        y(i) = y(i)+(k1(i)+2.0*k2(i)+2.0*k3(i)+k4(i))/6;
    }
}
```

Bibliography

[1] Stephen A. Lynch. Generating and measuring chaotic optical pulses from compact disc laser diodes. The project brief, Dec 2007.

[2] C. Harder, K. Lau, and A. Yariv. Bistability and pulsations in semiconductor lasers with inhomogeneous current injection. *IEEE Journal of Quantum Electronics*, 18(9):1351–1360, Sep 1982.

[3] M. Yamada. A theoretical analysis of self-sustained pulsation phenomena in narrow-stripe semiconductor lasers. *IEEE Journal of Quantum Electronics*, 29(5):1330–1336, May 1993.

[4] A. Egan, M. Harley-Steed, P. Rees, S. A. Lynch, J. O'Gorman, and J. Hegarty. An experimental and theoretical analysis of jitter in self-pulsating lasers synchronized to periodic electrical signals. *IEEE Photonics Technology Letters*, 8(6):758–760, Jun 1996.

[5] SecurityDocs.com. Encryption white papers. Website. `http://www.securitydocs.com/Encryption`.

[6] Internet Engineering Task Force. IETF TLS Status Pages. Website. `http://tools.ietf.org/wg/tls/`.

[7] The Free Software Foundation. GnuPG - GNU Privacy Guard. Website. `http://www.gnupg.org/`.

[8] Internet Engineering Task Force. RFC4880 - OpenPGP. Website. `http://www.ietf.org/rfc/rfc4880.txt`.

[9] Donald E. Knuth. *The Art of Computer Programming*, volume 2, chapter 3. Addison Wesley, 1999.

[10] M. Shinozuka, A. Uchida, T. Ogawa, Y. Shiniainura, K.Otani, S. Yoshimorii, and E Kanriari. Experiments on signal extraction from synchronized chaos inmicrochip lasers for secure communications. In *Lasers and Electro-Optics Society 2000 Annual Meeting. LEOS 2000. 13th Annual Meeting. IEEE*, volume 2, pages 740–741, Rio Grande, Puerto Rico, Nov 2000. IEEE.

[11] Stephen A. Lynch. Compact disc laser diode model. Supplementary material to the brief, Jan 2008.

[12] L. M. Pecora and T. L. Carroll. Synchronisation in chaotic systems. *Physical Review Letters*, 64(8):821–825, Feb 1990.

[13] A. Murakami. Phase locking and chaos synchronization in injection-locked semiconductor lasers. *IEEE Journal of Quantum Electronics*, 39(3):438–447, Mar 2003.

[14] M. Liu. Chaotic synchronization of semiconductor lasers. In *Lasers and Electro-Optics Society 1999 12th Annual Meeting. LEOS '99. IEEE*, vol-

ume 1, pages 377–378, San Francisco, CA, USA, Nov 1999. IEEE.

[15] G. van Tartwijk and D. Lenstra. Semiconductor lasers with optical injection and feedback. *Quantum Semiclassical Optics*, 7:87–143, 1995.

[16] J. Mørk, B. Tromborg, and J. Mark. Chaos in semiconductor lasers with optical feedback: theory and experiment. *IEEE Journal of Quantum Electronics*, 28(1):93–108, Jan 1992.

[17] J. C. M. Diels, J. J. Fontaine, I. C. McMichael, and F. Simoni. Control and measurement of ultrashort pulse shapes (in amplitude and phase) with femtosecond accuracy. *Applied Optics*, 24(9):1270–1282, 1985.

[18] Hobson, Riley, and Bence. *Mathematical Methods for Physics and Engineering*. Cambridge University Press, 2nd edition, 2004.

[19] Swamp Optics. Intensity autocorrelation tutorial. Website. `http://www.swampoptics.com/`.

[20] RP Photonics. Encyclopedia of laser physics and technology. Website. `http://www.rp-photonics.com/encyclopedia.html`.

[21] Thorlabs. Website. `http://www.thorlabs.com/NewGroupPage9.cfm?ObjectGroup_ID=1844&visNavID=693`.

[22] R. Guenther. *Modern Optics*. 1990.

[23] Wolfram Mathworld. Euler forward method. Website. `http://mathworld.wolfram.com/EulerForwardMethod.html`.

[24] Wolfram Mathworld. Runge-kutta method. Website. `http://mathworld.wolfram.com/Runge-KuttaMethod.html`.

[25] Alglib.net. AP mathematical libraries. Website. `http://www.alglib.net/`.

[26] R. Jones, P. Rees, P. Spencer, and K. Shore. Chaotic synchronization with self-pulsating semiconductor lasers. In *Lasers and Electro-Optics, 2000. (CLEO 2000). Conference on*, page 166, San Francisco, CA, USA, May 2000. IEEE.

[27] M. Liu, P. Davis, and T. Aida. Synchronized chaotic mode hopping in dbr lasers with delayed opto-electric feedback. *IEEE Journal of Quantum Electronics*, 37(3):337–352, Mar 2001.

[28] M. W. Lee and K. A. Shore. Demonstration of a chaotic optical message relay using dfb laser diodes. *IEEE Photonics Technology Letters*, 18(1):169–171, Jan 2006.

[29] D. Kanakidis, A. Argyris, A. Bogris, and D. Syvridis. Influence of the decoding process on the performance of chaos encrypted optical communication systems. *Journal of Lightwave Technology*, 24(1):335–341, Jan 2006.

[30] W. L. Mochán and B. S. Mendoza. Second harmonic generation at crystal surfaces. *Journal of Physics: Condensed Matter*, 5:A183–A184, Aug 1993.

Acknowledgements

Many thanks to Stephen Lynch, the group supervisor, for his time and tuition.

List of Figures

2.1 A graphical example of the addition of a data signal to random noise showing the noise-like output waveform
Created by L. Pomfrey 7

2.2 An example of the use of typical amplitude modulation for data transfer.
Created by G. Weerasinghe. 9

2.3 A series of photographs documenting the removal of the laser from a Compact Disc player in the lab. ?? shows the actual laser canister next to the lens normally mounted above it.
Created by W. Tou. 13

2.4 Schematic diagram showing the structure of a real CD laser.[11]
Created by, and used with permission of, S. Lynch. 14

2.5 Illustration showing possible recombination processes in a semiconductor on a simple band diagram.[11] ?? shows the radiative recombination process, this requires a vertical transition in k-space. ?? shows the processes of defect and surface recombination. These occur when an electron falls into a defect or surface level in the band gap and recombines non-radiatively from there. ?? shows the two-body Auger recombination process. There are many permutations of this process, denoted by the initial and final energy states. ?? depicts the CCCH process, which involves three electron states and one heavy hole state. In this example, two electrons collide in the conduction band. One of the electrons is excited into the valence band while the other is excited higher into the conduction band, where it eventually thermalises back down to the bottom of the conduction band, releasing the excess energy as heat.[11]
Created by L. Pomfrey. 15

2.6 Illustration showing the linear gain approximation. Plotted on the graph is the peak gain as a function of carrier density.[11]
Created by L. Pomfrey. 17

2.7 Possible methods for causing chaotic pusation in lasers. ?? shows optical injection using an external light source, whilst ?? shows feedback from an external cavity mirror.
Created by L. Pomfrey and P. Smith 19

2.8 Compact Disc player laser production by the TO-can production line.
Created by L. Pomfrey. 21

2.9 ?? shows a pulsation with a definite frequency. ?? shows the power spectrum of ??. A sharp peak corresponding to a large component of energy concentrated at a particular frequency can clearly be seen. It should be noted that the power spectrum does not appear as a sharp vertical line because the pulsation, in this case, is not a perfect sinusoidal wave and actually contains a combination of frequencies. ?? shows the auto-correlation of ??. It can be seen that an auto-correlation value of 3 corresponds to maximum correlation between a pulse and a pulse shifted by a small time delay. The distance between peaks is the pulse interval. For a real optical auto-correlator, this distance corresponding to the time delay would be varied and measured, therefore, the pulse interval can be found.
Created by G. Weerasinghe. 22

2.10 Schematic diagram of a simple intensity auto-correlator.[1]
Created by L. Pomfrey. 24

2.11 Phase mismatch for second harmonic generation. Due to chromatic dispersion the wave number of the second harmonic is more than twice as large as that for the fundamental wave.[20]
Created by P. Smith. 26

2.12 Graphs showing effectiveness of PPLN compared to non-phased matched situation.[21]
Created by P. Smith. 28

2.13 A graph of two signals, their overlap in the second harmonic crystal and the second harmonic signal. Only the signal under the pulse is detected by the photo-multiplier tube. Varying τ obtains better temporal resolution.[22]
Created by P. Smith. 28

2.14 A semi-conductor laser diode of the type used in the experiment. ?? shows the laser in it's enclosure ready to be used, a US one cent coin is shown for scale. ?? shows the laser chip itself, a sewing needle is shown for scale.
Created by W. Tou and A. Shalashilin. 29

2.15 A schematic of a corner cube of the type used in the experiment.
Created by W. Tou and A. Shalashilin. 29

2.16 Scale drawing of the auto-correlator setup. *Note:* A larger version is available in the apppendix.
Created by W. Tou. 31

2.17 Images of a photo-multiplier tube in the lab.
Created by W. Tou. 33

2.18 Graphs of photon density from the *Mathematica* and *Matlab* models respectively.
Created by M. Moussa and G. Weerasinghe. 40

2.19 Chaotic output from the *Mathematica* and *Matlab* models, the input current amplitude was set at 32 mA and the modulation frequency was set at 2 GHz.
Created by M. Moussa and G. Weerasinghe. 41
2.20 The plot of the ratio of the carrier region densities from the *Matlab* model. The difference between the plots when the laser is exhibiting chaotic and non-chaotic pulsation is easily visible.
Created by M. Moussa and G. Weerasinghe. 42
2.21 The simulation predicts the self pulsation of a laser diode. The figure shows that when a DC bias is applied, the laser diode will not emit a continuous stream of light, but pulse packets of photons instead. When a DC bias current above 41.65 mA is applied, the pulsations cease. No output from the laser is obtained below a bias current of 29.6 mA.
Created by G. Weerasinghe. 44
2.22 The plots of the two carrier densities in regions 1 and 2. The smaller amplitude plot represents the unbiased region 2.
Created by G. Weerasinghe. 44
2.23 An example of chaotic output.
Created by G. Weerasinghe. 45
2.24 A typical power spectrum.
Created by G. Weerasinghe. 46
2.25 The auto-correlation function for chaotic and non-chaotic pulses respectively.
Created by G. Weerasinghe. 46
2.26 Results taken at the point of maximum chaotic output, as determined by a chaos detection algorithm. This point is taken to be at 32.5 mA at a modulation frequency of 5 GHz.
Created by G. Weerasinghe. 48

List of Tables

2.1 List of parameters and their nominal values for the CD laser model.[11] *Created by L. Pomfrey.* 19
2.2 Examples of second- and third-order non-linear processes. *Created by P. Smith.* 28
2.3 The equipment required to build the auto-correlator and the prices of each item. *Created by A. Shalashilin.* 32

List of Attachments

This section contains a list of supplementary documents that can be found attached to this report. These include the agenda and reports from the group meetings and similar items.

- A3 version of the scale drawing (Figure 2.16).
- Minutes and Agenda for the meetings held on;
  - Tuesday 11th December, 2007 (Minutes only.)
  - Monday 14th January, 2008
  - Monday 21st January, 2008
  - Thursday 31st January, 2008
  - Friday 1st February, 2008
  - Thursday 28th February, 2008
  - Thursday 6th March, 2008