# Modelling Nonlinear optics with the Bloch-Messiah reduction

Oliver Thomas

Quantum Engineering CDT
University of Bristol

August 17, 2018

# Overview

- What is nonlinear optics?
- Why do we care about it?
- What I have been doing
- Gaussian optics
- Outlook

# Motivation

## The good

Spontaneous Parametric processes, SPDC, SFWM

- Heralded single photon sources
- Entangled photon pair generation (polarisation, spatial)

Kerr processes

- Self-Phase modulation (SPM) for generating Bannana states (CV)
- Cross-Phase modulation (XPM) for sensing

## The bad

- Generating more than two photons -> bad for quantum computing
-

All Kerr nonlinear processes

- SPM -> Spectral broadening
- XPM -> Unwanted phase shifts on single photons due to propagation of the pump

# What do we mean by nonlinear optics?

- Roughly processes that conserve energy but do not conserve photon number.

$$P = E_1 + \chi^{(1)} E_1 E_2 + \chi^{(2)} E_1 E_2 E_3 + \chi^{(3)} E_1 E_2 E_3 E_4 + \dots \quad (1)$$

Here we are going to talk about squeezing, i.e SPDC or SFWM, Hamiltonians are then of the form,

$$\hat{H} = A \hat{a}_S^\dagger \hat{a}_I^\dagger \hat{a}_P + h.c. \quad (2)$$

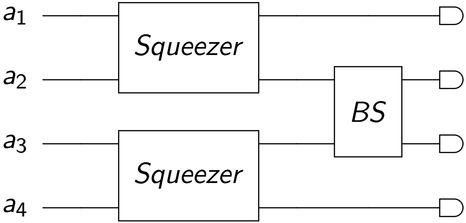
$$\hat{H} = A \hat{a}_S^\dagger \hat{a}_I^\dagger \hat{a}_P \hat{a}_P + h.c. \quad (3)$$

**Note** for the rest of this presentation I will drop the hat notatiaion and using the convention a, b are annihilation operators in modes a & b

# Gaussian Optics

- Using the undelpeted pump approximation we can write the Hamiltonians as terms which are at most quadratic in creation and annihilation operators.
- These are Gaussian transforms, they take Gaussian states to Gaussian states

$$\begin{bmatrix} \vec{b} \\ \vec{b}^\dagger \end{bmatrix} = M \begin{bmatrix} \vec{a} \\ \vec{a}^\dagger \end{bmatrix} \tag{4}$$



# Making a repository

- Go to the folder and right click `git with bash`
- You are now able to use bash for the rest of the talk!

# Basic Git commands

- There are four[1] important commands you will need for git:
- `git pull`
- `git add *`
- `git commit -a`
- `git push`

[1] I cheat here and write a bash script which does these in order so I only have to run a single command.

# Advanced Git commands

- One of the great things about Git is that you can get by with just the four above commands.
- The git man page is very useful, especially,

  ```
  man gittutorial
  man giteveryday
  ```

- `giteveryday` is a super useful collection of the 20 commands you will need regularly.

# Adding Collaborators

- Go to a repository and on the settings tab click collaborators, you can then search for the github username

# Why Python?

- Python is popular, multi-platform and becoming a standard language[2]
- It is a good high level language to know, it is a very flexible interpreted language.

[2] standard on most of the popular linux distributions

# Python syntax

- As with every programming language we should figure out how to do *Hello, world!*

Open python and type:

```
print 'Hello, world!'
```

- As Python is an interpreted language you can run command by command in python or use an IDE and then use python to run the program. For plotting it is more useful to write the program out in an IDE first.

# Adding your first commit

- Save your *hello, world!* program.
- Then either run:
- `git add *`
- `git commit -a`
- `git push`

Or use the windows GUI version and commit them to your repository.

# Plotting

- Python requires the `numpy` library[3] for a lot of basic maths functions (and arrays).
- We are going to use the `matplotlib` library[4] for the remainder of this talk.

[3] http://www.numpy.org/

[4] https://://matplotlib.org/api/_as_gen/matplotlib.pyplot.plot.html

# Example 1, Plotting functions

- Go to the `src` folder and open `ex1functions.py`
- Run `all.py` and choose 1

# Example 1, Plotting functions

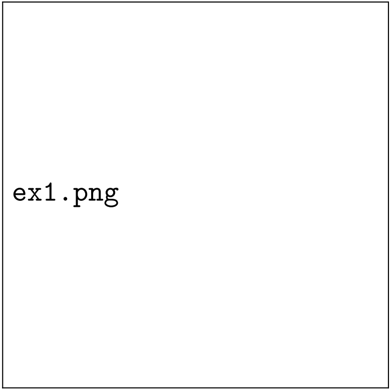


Figure: function plotting

- It could do with some axis labels.
- go into the program and find the line called `plt.ylabel=` and `plt.xlabel=`

# Example 2, Complicated functions!

- In the `src` folder open `ex2compfunctions.py`
- Run all.py and choose 2

# Example 2, Complicated functions!

- Figures!
-

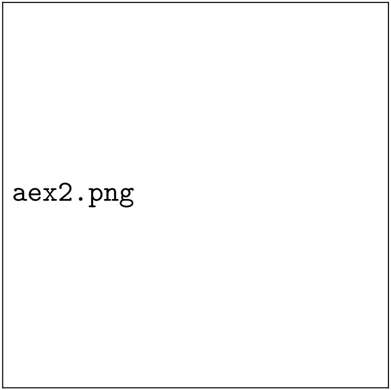


Figure: function plotting

# Example 3, Plotting data!

- once again, in the `src` folder open `ex3data.py`
- Run all.py and choose 3

# Example 3, Plotting data!

- figure

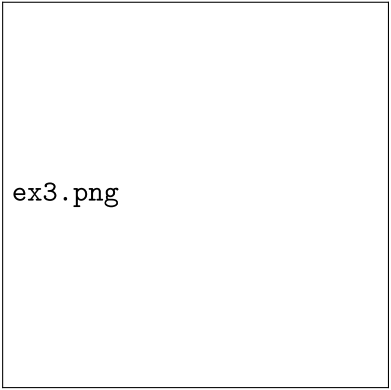


Figure: function plotting

# Example 4, Histograms!

- once again, in the `src` folder open `ex4hist.py`
- Run all.py and choose 4

# Example 4, Histograms!

- figure

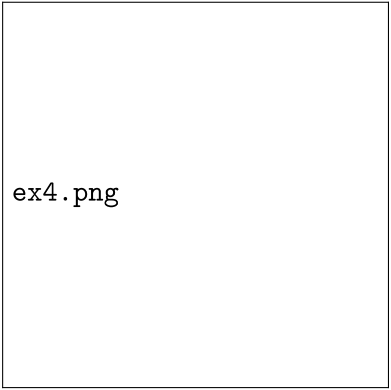


Figure: function plotting

# Example 5, Subplots!

- In the `src` folder open `ex5subplots.py`
- Run all.py and choose 5

# Example 5, Subplots!

- Figures!

`ex5.png`

Figure: function plotting

# Example 6, Art!

- In the `src` folder open `ex6art.py`
- Run all.py and choose 6

# Example 6, Art!

- Figures!

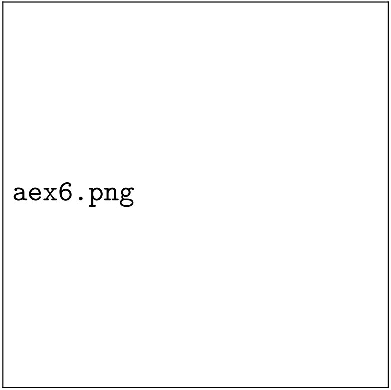


Figure: function plotting

# Branching

- Branching is useful, it lets you test something out separately to the main branch.
- To make a new branch called `test`

  ```
  git branch test
  ```
- You can check all of the current branches and which branch you are on with

  ```
  git branch
  ```

# Branching

- To switch to the test branch type:

  ```
  git checkout test
  ```

# Thanks for listening!

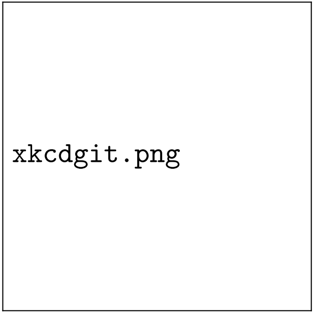


Figure: If it all goes wrong . . . [5]

[1] https://xkcd.com/1597/