



FACULTAD DE CIENCIAS

Simulación de peines de frecuencia óptica generados por láseres de semiconductor

Simulation of optical frequency comb generate by
semiconductor lasers

Trabajo de Fin de Grado para acceder al
Grado en Física

Autor
Jaime DÍEZ GONZÁLEZ-PARDO
Director
Dr. Á.A. VALLE GUTIERREZ

19 de agosto de 2019

Resumen

Hola

Abstract

Hola

Índice general

1. Introducción	4
1.1. Láseres de Semiconductor	4
1.2. Procesos Estocásticos	4
1.3. Dinámica No Lineal	4
1.4. Peines de Frecuencia Óptica	4
1.4.1. Gain-Switching	4
1.4.2. Inyección Óptica	4
1.4.3. Aplicaciones	4
1.5. Objetivo del Estudio	4
2. Modelo Computacional	5
2.1. RoF	5
2.2. Código de la Simulación	5
3. OFC (Gain Switching)	6
4. Inyeccion de Luz	12
5. Inyeccion de luz en OFC	15
6. Conclusiones	17
A. Código de la simulación	19

Índice de figuras

3.1. Espectros en continua de la simulacion	6
3.2. Espectros en continua de Chaves	7
3.3. Transitorio	8
3.4. RateEquations	9
3.5. PSD	9
3.6. Current	10
3.7. 500	10
3.8. 500mhz	11
4.1. el pie de pagina que le quieras poner a la imagen	12
4.2. Map	13
4.3. ZoneRtEq	13
4.4. P2zone	14
5.1. el pie de pagina que le quieras poner a la imagen	15
5.2. p1-P2	16

Capítulo 1

Introducción

1.1. Láseres de Semiconductor

1.2. Procesos Estocásticos

1.3. Dinámica No Lineal

1.4. Peines de Frecuencia Óptica

Que son, características principales y como se generan,...

1.4.1. *Gain-Switching*

hola que tal todos

1.4.2. Inyección Óptica

adios a todos

1.4.3. Aplicaciones

1.5. Objetivo del Estudio

Capítulo 2

Modelo Computacional

2.1. RoF

2.2. Código de la Simulación

Capítulo 3

OFC (Gain Switching)

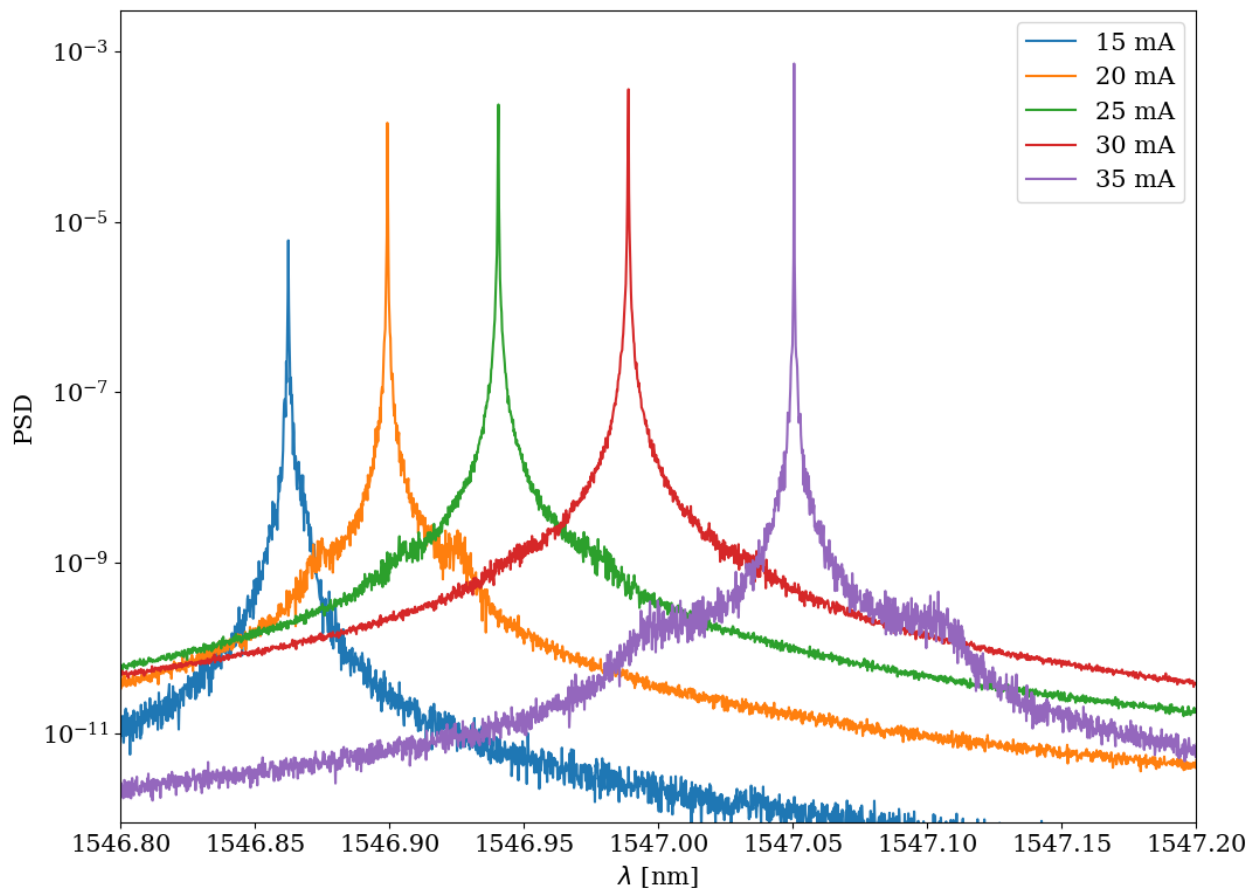


Figura 3.1: Espectros en continua de la simulacion

I_{Bias}	λ
15	1546.86
20	1546.90
25	1546.94
30	1546.99
35	1547.05

Tabla 3.1: Lambdas de la simulacion

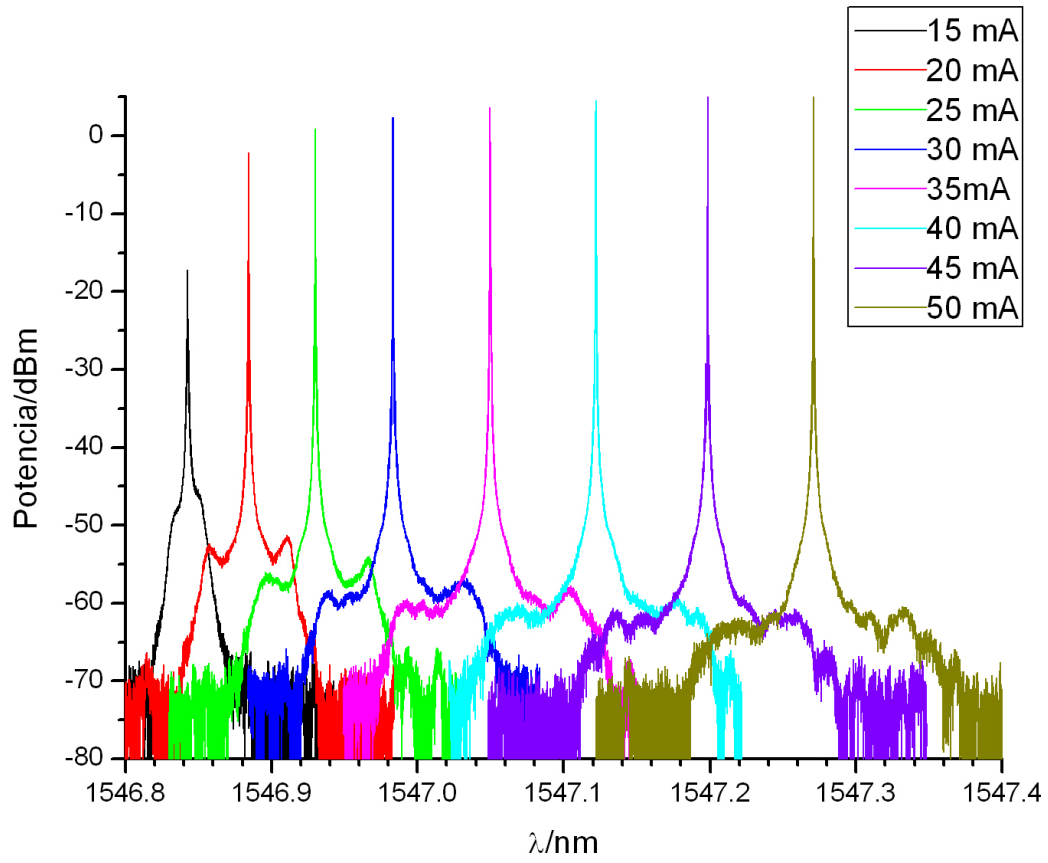


Figura 3.2: Espectros en continua de Chaves

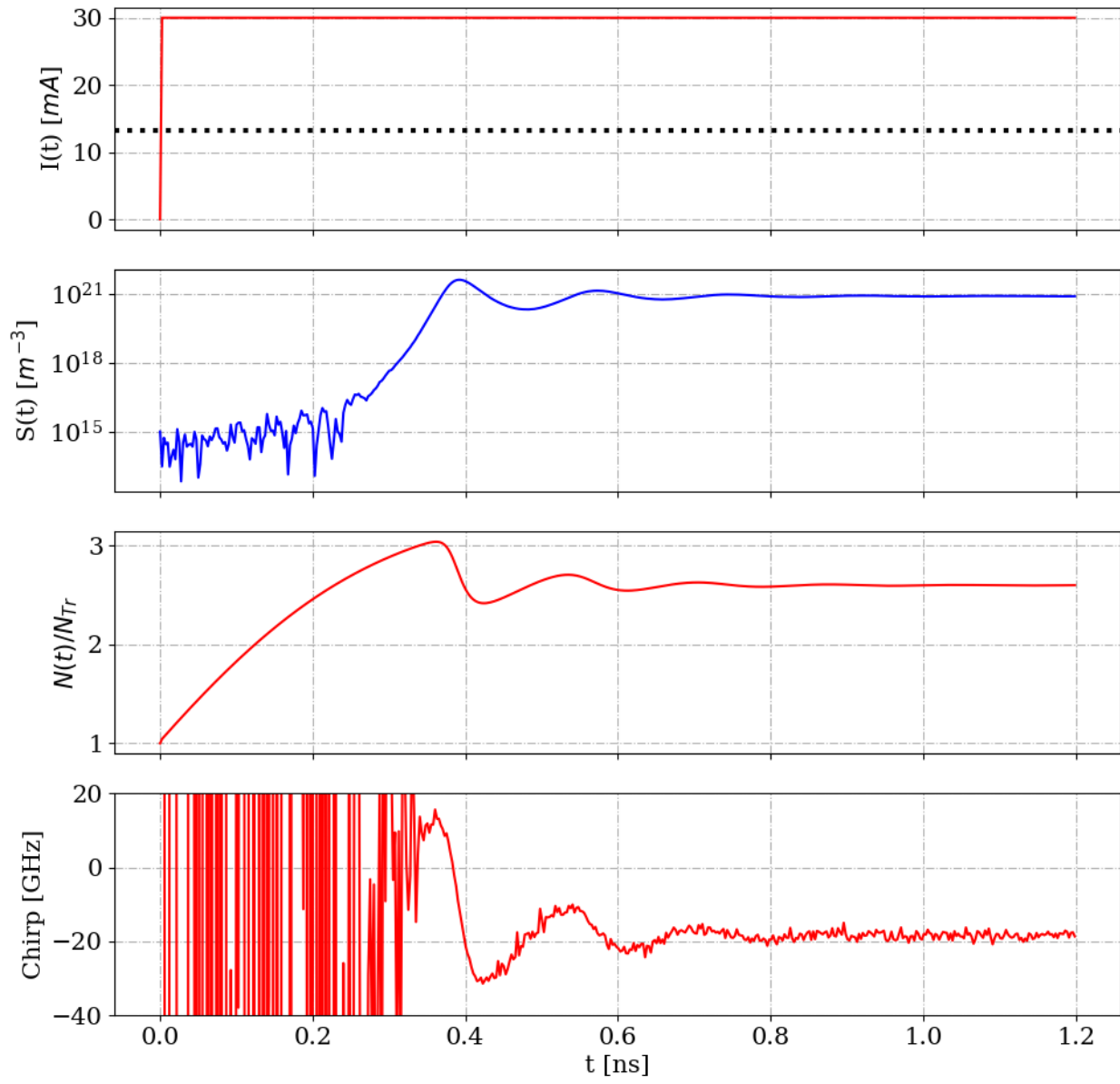


Figura 3.3: Transitorio

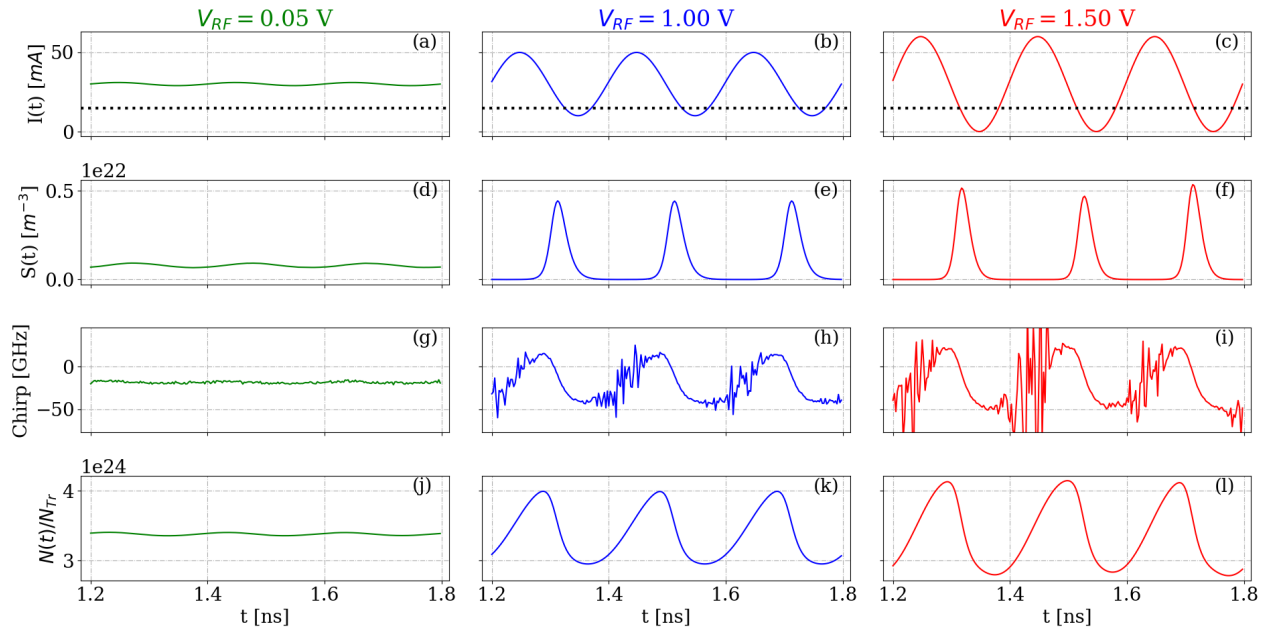


Figura 3.4: RateEquations

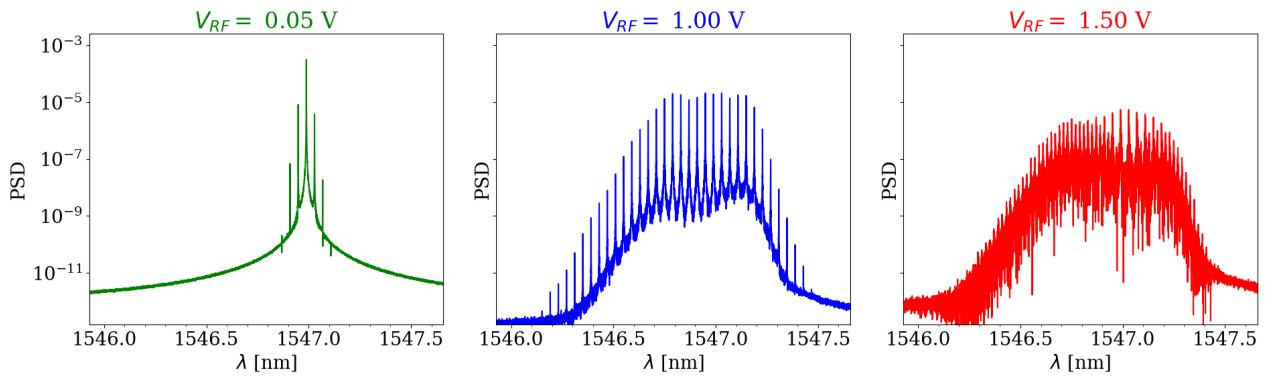


Figura 3.5: PSD

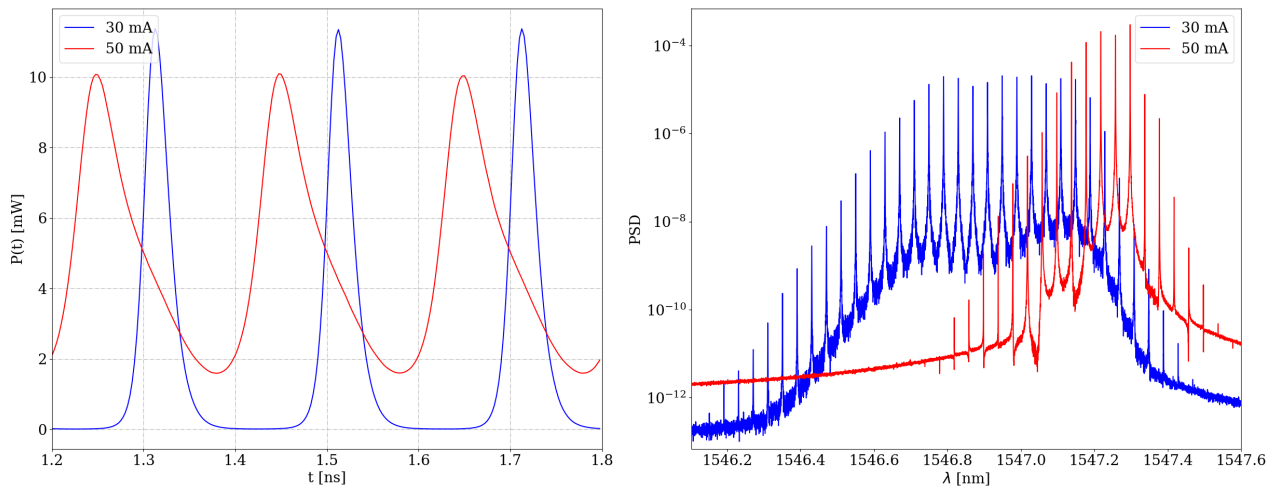


Figura 3.6: Current

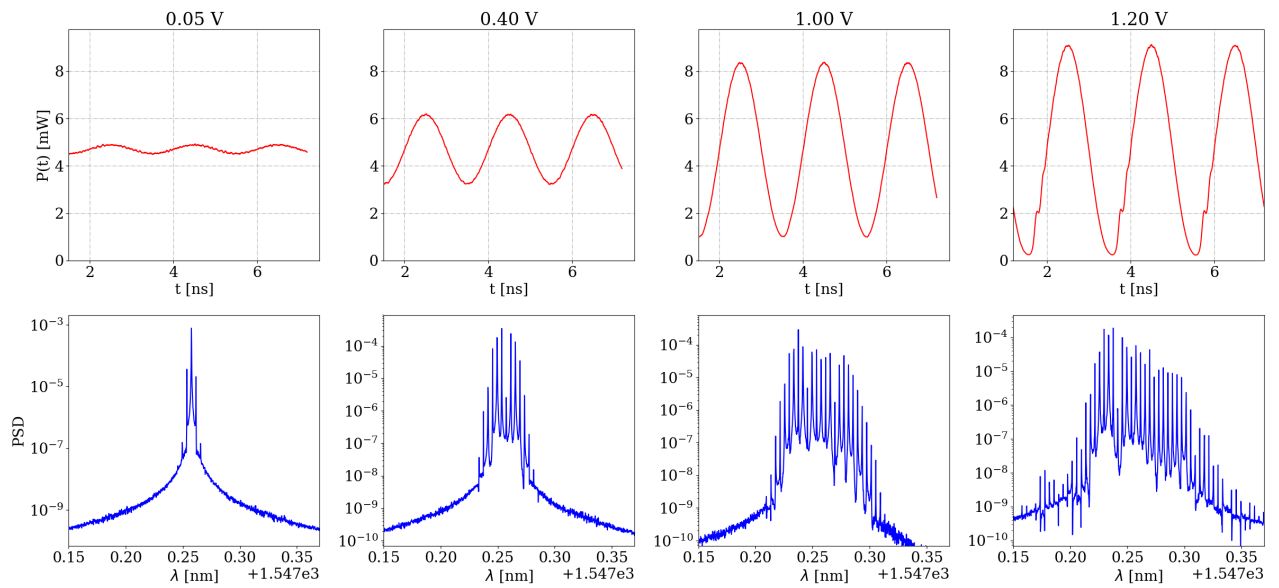


Figura 3.7: 500

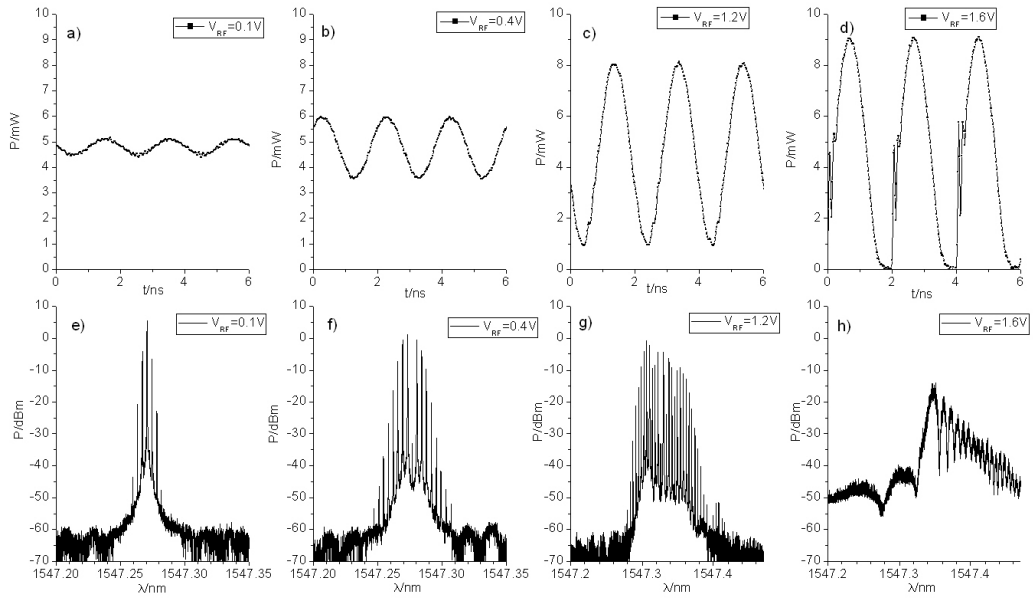


Figura 3.8: 500mhz

Capítulo 4

Inyección de Luz

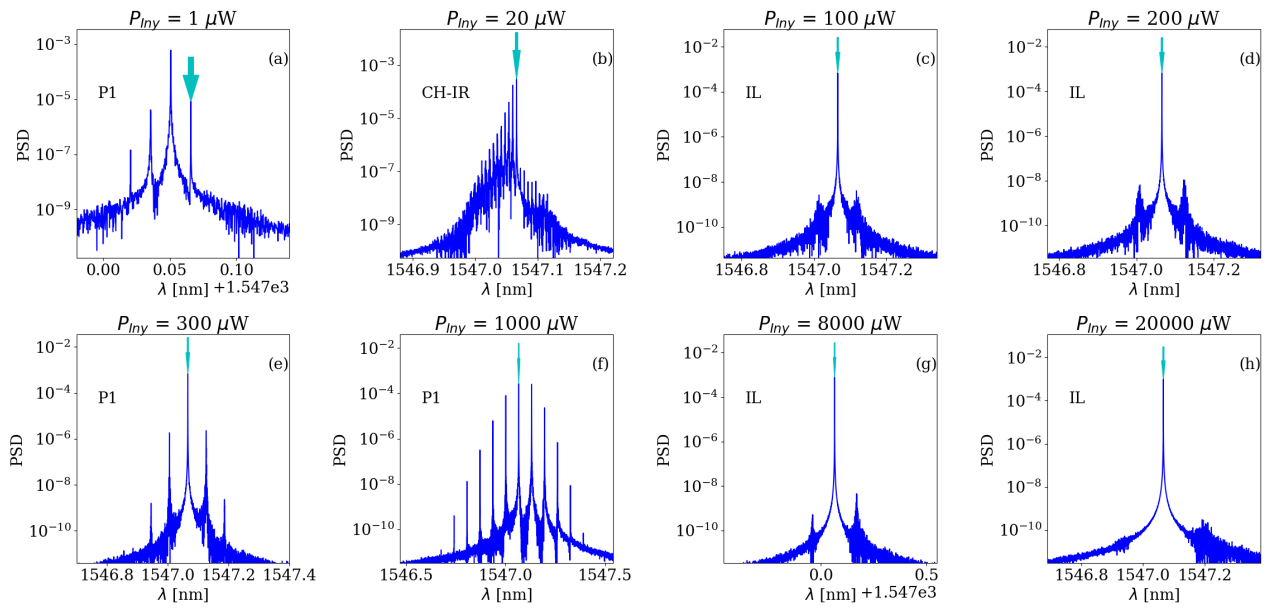


Figura 4.1: el pie de pagina que le quieras poner a la imagen

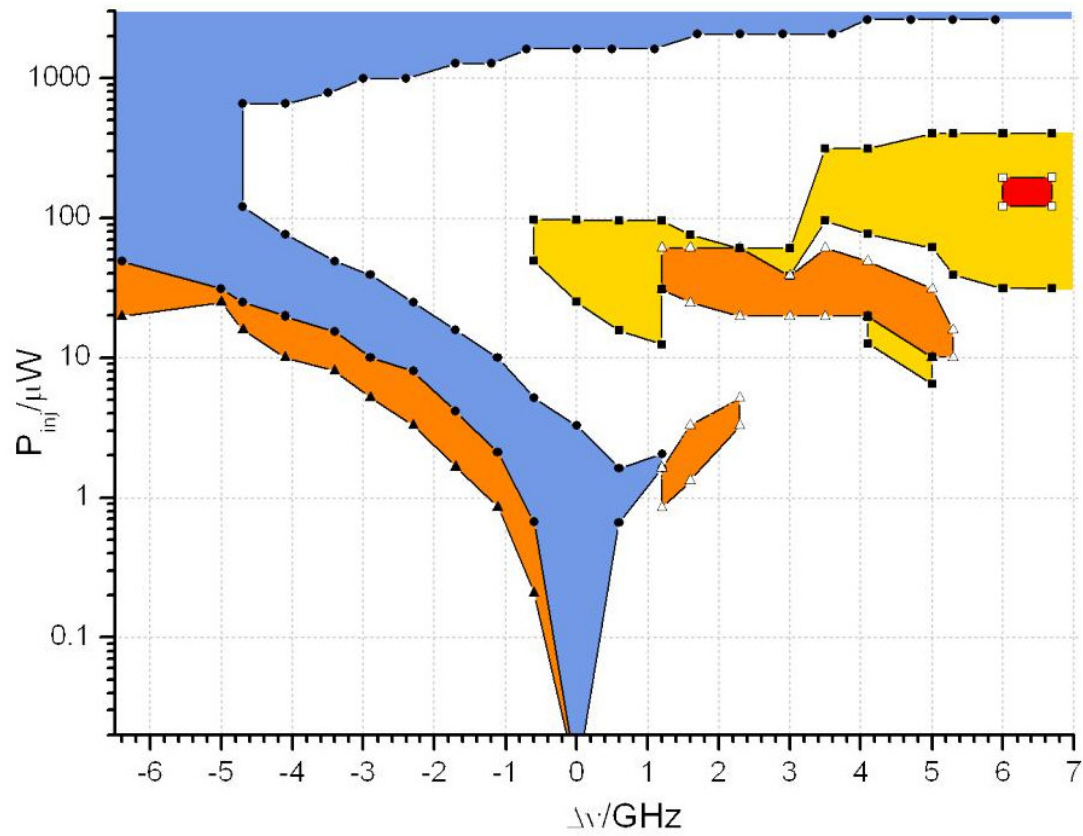


Figura 4.2: Map

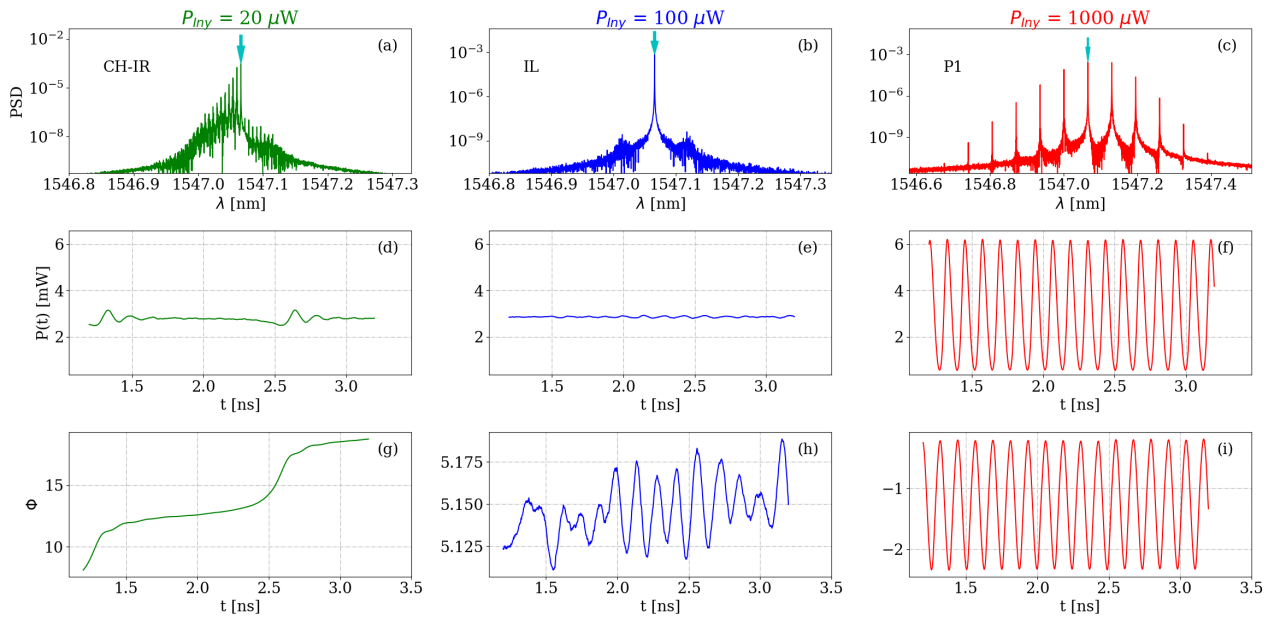


Figura 4.3: ZoneRtEq

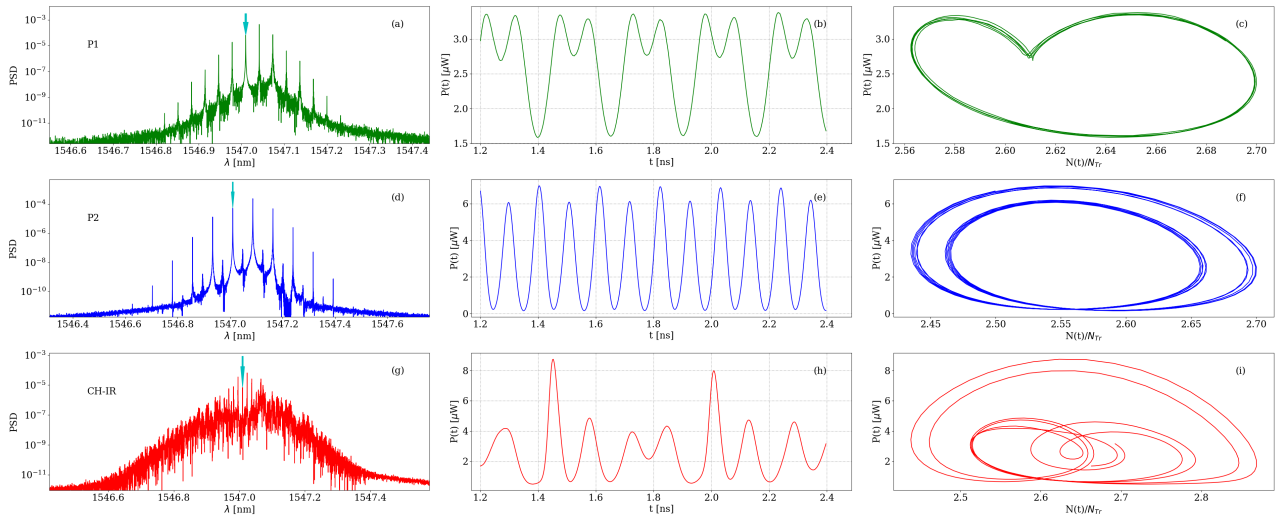


Figura 4.4: P2zone

Capítulo 5

Inyeccion de luz en OFC

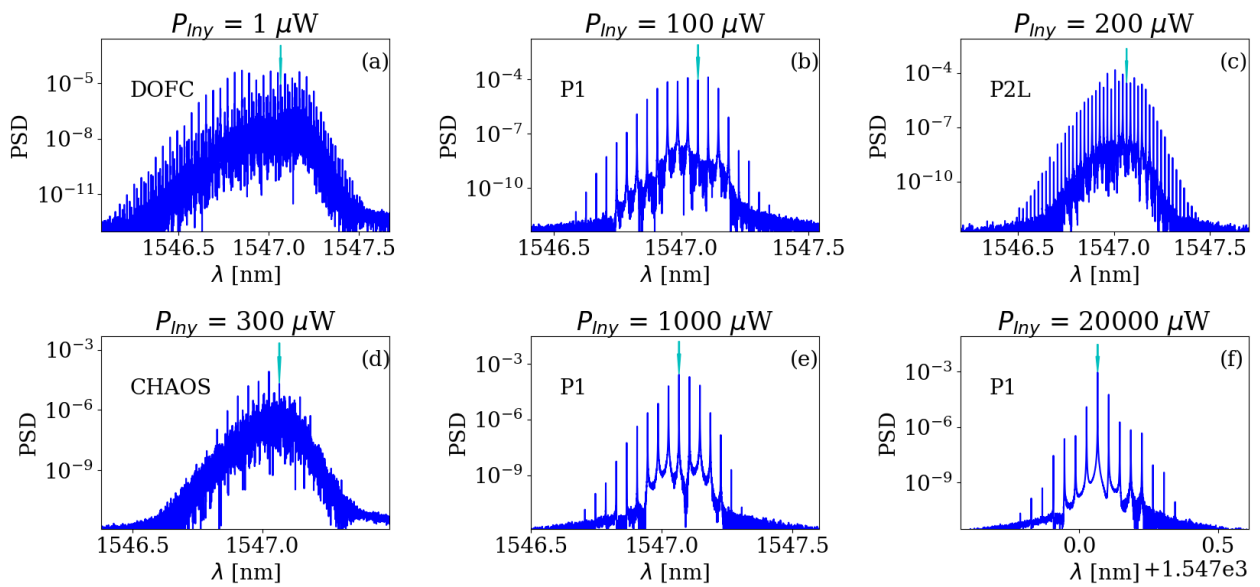


Figura 5.1: el pie de pagina que le quieras poner a la imagen

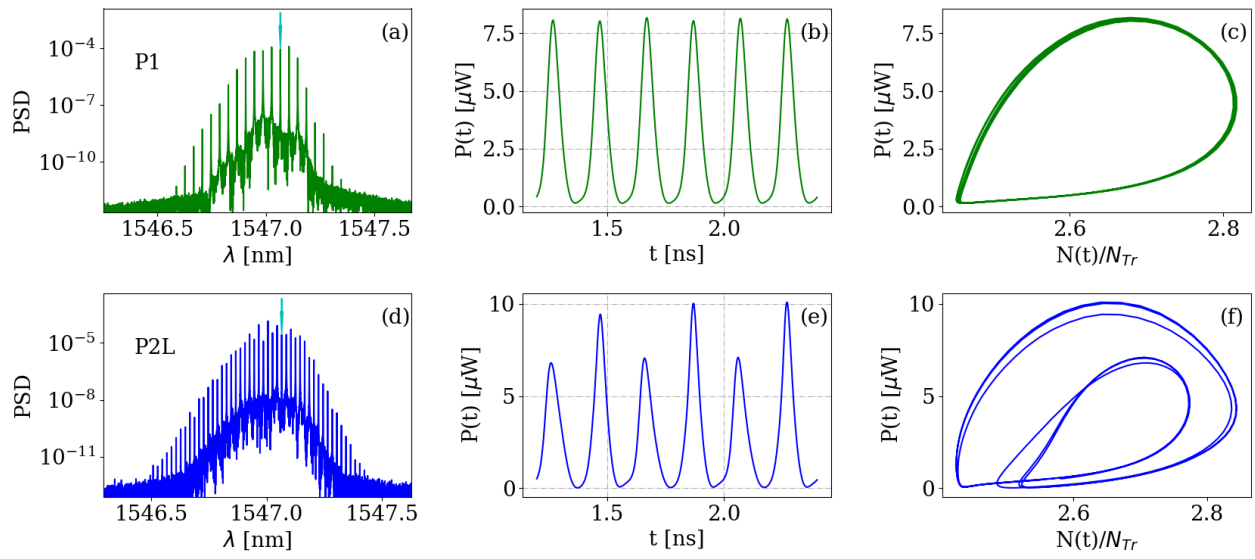


Figura 5.2: p1-P2

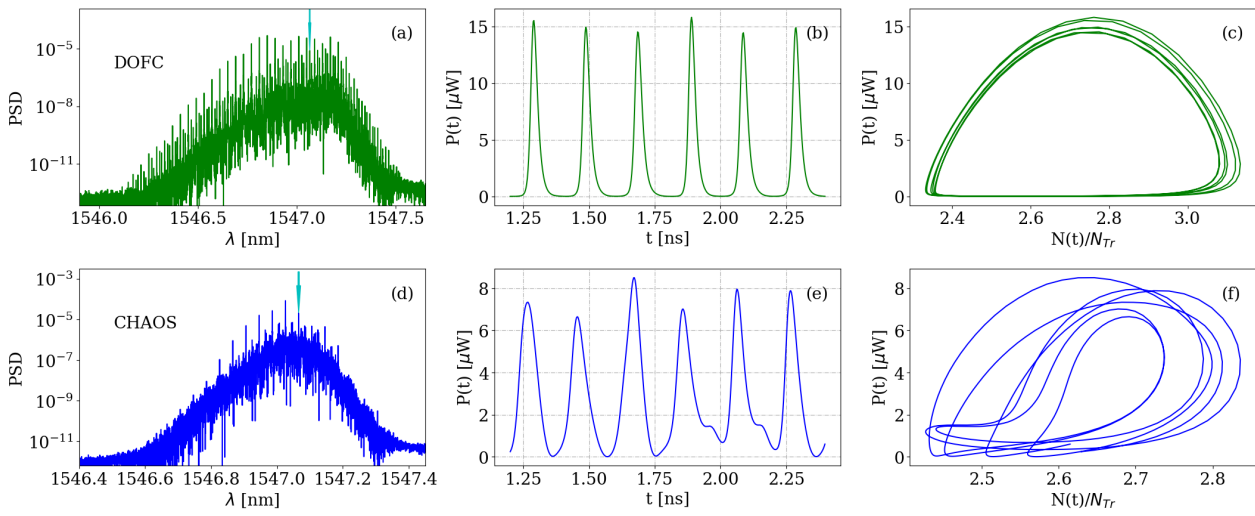


Figura 5.3: Chaos

Capítulo 6

Conclusiones

Bibliografía

Apéndice A

Código de la simulación

```
1  def allSimulation(self):
2      self.setArrays()
3
4      ampInject = kc*np.sqrt(self.sInjct)*tIntev
5      opFldInject = np.sqrt(rSL*constP*self.sInjct)*np.exp(1j*np.pi)
6
7      derivPhaseTerm = self.phaseTerm / tIntev
8      self.TFavg = 0
9      self.TFang = 0
10
11     frequencyLimits = 1 / (2*delta)
12     self.fftFreq = np.linspace(-frequencyLimits,
13                                frequencyLimits,
14                                len(self.opField)
15                                )
16
17     deltaFreq = deltaFreqs[self.iBias]
18     self.fftFreq += f0 - (deltaFreq/(2.0*np.pi))
19     self.fftWL = (c0/self.fftFreq) *10**(9) # Wavelength [nm]
20
21     for win in range(0, self.numWindw):
22
23         #-----
24         # Gaussian arrays N(0,1) for the Noise
25         #-----
26         X = np.random.normal(0, 1, nTotal)
27         Y = np.random.normal(0, 1, nTotal)
28
29         # Initial conditions are defined in order to resolved the SDE
30         tempN = nTr
31         tempS = float(10**(15))
32         tempPhi = 0
33
34         for q in range(0, mTrans):
35             for k in range(0, ndelta):
36
37                 index = q*ndelta + k
38
39                 bTN = bTIntv * tempN * tempN
40                 invS = 1 / ((1/tempS) + epsilon)
41                 sqrtS = np.sqrt(abs(tempS))
```

```

42         cosPhi = np.cos(tempPhi)
43         senPhi = np.sin(tempPhi)
44
45         tempPhi = (tempPhi + aphvgTGmm*tempN - self.phaseTerm
46                   + noisePhi*tempN*Y[index]/sqrtS
47                   - (ampInject/sqrtS)*senPhi*cosInject[index]
48                   + (ampInject/sqrtS)*cosPhi*senInject[index]
49                   )
50
51         tempS = (tempS + vgTGmm*tempN*invS - vgTGmmN*invS
52                 - intTtau*tempS + btGmm*bTN
53                 + noiseS*tempN*sqrtS*X[index]
54                 + 2*ampInject*sqrtS*cosPhi*cosInject[index]
55                 + 2*ampInject*sqrtS*senPhi*senInject[index]
56                 )
57
58         tempN = (tempN + self.currentTerm[index] - aTIntv*tempN
59                 - bTN - cTIntv*tempN**3 - vgT*tempN*invS + vgtN*invS
60                 )
61
62         self.I[q] = self.currentTerm[index] *10**(12) / eVinv
63         self.N[q] = tempN
64         self.S[q] = tempS
65         self.Phi[q] = tempPhi
66         self.dPhi[q] = (1/(2*np.pi))*(derivAphvgTGmm*tempN
67                               - derivPhaseTerm + derivNoisePhi*tempN*Y[q]/sqrtS
68                               )
69
70     for q in range(mTrans, mTotal):
71         for k in range(0, ndelta):
72
73             index = q*ndelta + k
74
75             bTN = bTIntv * tempN * tempN
76             invS = 1 / ((1/tempS) + epsilon)
77             sqrtS = np.sqrt(tempS)
78             cosPhi = np.cos(tempPhi)
79             senPhi = np.sin(tempPhi)
80
81             tempPhi = (tempPhi + aphvgTGmm*tempN - self.phaseTerm
82                       + noisePhi*tempN*Y[index]/sqrtS
83                       - (ampInject/sqrtS)*senPhi*cosInject[index]
84                       + (ampInject/sqrtS)*cosPhi*senInject[index]
85                       )
86
87             tempS = (tempS + vgTGmm*tempN*invS - vgTGmmN*invS
88                     - intTtau*tempS + btGmm*bTN
89                     + noiseS*tempN*sqrtS*X[index]
90                     + 2*ampInject*sqrtS*cosPhi*cosInject[index]
91                     + 2*ampInject*sqrtS*senPhi*senInject[index]
92                     )
93
94             tempN = (tempN + self.currentTerm[index]
95                     - aTIntv*tempN - bTN - (cTIntv*tempN**3)
96                     - vgT*tempN*invS + vgtN*invS
97                     )
98
99         self.I[q] = self.currentTerm[index]*10**(12) / eVinv
100        self.N[q] = tempN

```

```

101         self.S[q] = tempS
102         self.Phi[q] = tempPhi
103         self.dPhi[q] = ((derivAphvgTGmm*tempN - derivPhaseTerm
104                        + derivNoisePhi*tempN*Y[index]/np.sqrt(tempS))
105                        / (2*np.pi)
106                        )
107         self.opField[q-mTrans] = (np.sqrt(constP*tempS)
108                                * np.exp(1j*tempPhi)
109                                + opFldInject
110                                * np.exp(1j*angInject[index])
111                                )
112
113         self.I[0] = 0
114         self.N[0] = nTr
115         self.S[0] = float(10**(15))
116         self.Phi[0] = 0
117         transFourier = np.fft.fft(self.opField)
118         self.TFavg += (abs(np.fft.fftshift(transFourier))
119                      * abs(np.fft.fftshift(transFourier))
120                      / float(self.numWindw)
121                      )
122
123         self.TFang += (np.angle(np.fft.fftshift(transFourier))
124                      / float(self.numWindw)
125                      )

```