FACULTAD DE CIENCIAS

Simulación de peines de frecuencia óptica generados por láseres de semiconductor

Simulation of optical frequecy 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

31 de agosto de 2019

Índice general

Índice de figuras

Capítulo 1

Introducción

Hola a todos

1.1. Láseres de Semiconductor

1.2. Procesos Estocásticos

1.3. Dinámica No Lineal

1.4. Peines de Frecuencia Óptica

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

1.4.1. *Gain-Switching*

hola que tal todos

1.4.2. Inyección Ópticar

adios a todos

1.4.3. Aplicaciones

1.5. Objetivo del Estudio

Capítulo 2

Modelo Computacional

Hola a todos

2.1. RoF

2.2. Código de la Simulación

Se explicará el codigo utilizado para el trabajo

2.2.1. Término de la temperatura

Explicar el termino de la temperatura

2.2.2. Transitorio

Explicar el transitorio

Capítulo 3

Láser en solitario

Antes de abordar el estudio de la dinámica no lineal del láser de semiconductor de modo discreto se ha realizado la simulación del láser en solitario, sin inyección de luz del láser esclavo ($P_{Iny} = 0$). Se han realizado simulaciones para el láser tanto en corriente continua (CW de sus siglas en inglés), como en Gain-Switching, comparando los resultados con los obtenidos experimentalmente en condiciones similares [1].

3.1. Láser en corriente continua

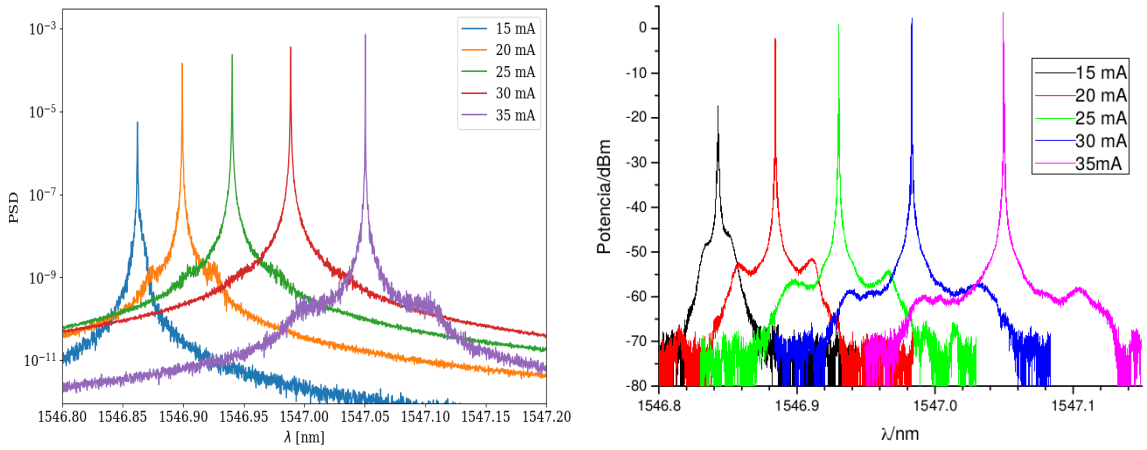
Para poder realizar las simulaciones en corriente continua se ha trabajado con una corriente de inyección constante e igual a la corriente de polarización ($I(t) = I_{Bias}$), tomando $V_{RF} = 0$.

HABLAR DE QUE EN CONTINUA SE TIENE QUE $\frac{dN}{dt} = \frac{dS}{dt} = \frac{d\Phi}{dt} = 0$

3.1.1. Espectros de emisión

Tal y como se vio en la sección 1.1, pequeñas variaciones en la corriente de inyección del láser producen desplazamientos de la línea de emisión del láser en el espectro de frecuencias. Por ello es importante conocer el valor de la longitud de onda del pico de emisión del láser en solitario en función de la corriente de polarización I_{Bias}, de cara a realizar el estudio de la inyección de luz.

En la Figura 3.1 se muestran las densidades espectrales de potencia del láser a diferentes corrientes de polarización, comparando los datos obtenidos mediante la simulación del láser(Figura 3.1a), con los obtenidos experimentalmente(Figura 3.1b).



(a) *Espectros ópticos obtenidos mediante simulación.*

(b) *Espectros ópticos obtenidos experimentalmente.*

Figura 3.1: *Espectros ópticos del DML para diferentes corrientes de polarización I_{Bias}obtenidos mediante simulación (izquierda 3.1a) y mediante experimento (derecha, 3.1b).*

Comparando los espectros obtenidos mediante simulación con los obtenidos esperimentalmente se observa un gran parecido en la forma, observando una forma más puntiaguda y estrecha en los espectros con corriente $I_{Bias} = 15$ mA para ambas gráficas. Además, la simulación permite observar los picos propios de las oscilaciones de relajación del láser que se observan en el experimento.

Además, a partir de los espectros de la Figura 3.1 se pueden obtener las longitudes de onda de los picos de emisión en funci'on de la corriente I_{Bias}. En la Tabla 3.1 se muestran los valores de las longitudes de onda *lambda* obtenidas de los espectros de la Figura 3.1 tanto para la simulación como para el experimento.

I_{Bias}	λ_{sim}	λ_{exp}
15	1546.86	1546.84
20	1546.90	1546.88
25	1546.94	1546.93
30	1546.99	1546.98
35	1547.05	1547.05

Tabla 3.1: *Longitud de onda de las lineas de emisión del DML en función de la I_{Bias}obtenidas de la figura 3.1. Se muestran los valores experimentales λ_{exp} obtenidos de la gráfica 3.1b con un error de $\delta\lambda_{exp} = 0.02$, y los valores obtenidos de la simulación de la gráfica 3.1a.*

Los valores de las longitudes de onda que se muestran en la Tabla 3.1 muestran una gran similitud entre los valores experimentales y los obtenidos mediante simulación, obteniendo una discrepancia máxima de 0.02 nm. De esta forma, la gran concordancia entre los valores de λ experimentales y los obtenidos a partir de la simulación, junto con la gran similitud en la forma de los espectros, muestra la capacidad de la simulación de reproducir computacionalmente los resultados obtenidos experimentalmente en el laboratorio.

Para el estudio de la inyecci'on de luz se trabajará con una corriente $I_{Bias} = 35$ mA. Por tanto, la Tabla 3.1 permite obtener su longitud de onda de emisión de $\lambda = 1547.05$ nm, siendo además la misma que la obtenida en el experimento.

3.1.2. Oscilaciones de Relajación

Para que el láser comience a emitir se ha de cumplir que la emisión estimulada domine frente a la emisión espontánea. Esto se produce cuando la densidad de portadores de carga en la región activa supera un valor umbral, N_{th}, a partir del cuál el láser comienza a emitir fotones. Si la inyecciín de corriente que se le aplica al láser es constante (corriente continua), la densidad de portadores de carga tenderá a estabilizarse en N_{th}. De el mismo modo, la densidad de fotones y el chirp se estabilizarán para valores S_{th} y Φ_{th}.

No obstante, si se parte de unas condiciones iniciales del láser con valores $N(t=0) < N_{th}$, será necesario que transcurra un cierto tiempo hasta que el láser alcance el equilibrio y se estabilice. A este tiempo se le denomina transitorio.

Cabe destacar que, tal y como se comentó en el apartado 2.2.2, para la simulación se ha trabajado con un tiempo de transitorio $t_{trans} = 1.2$ ns, en el cuál se ha operado con la raiz cuadrada del módulo de S en los términos de ruido para evitar resultados complejos, despeciando dicho intervalo en el estudio de los espectros. Sin embargo, en este apartado se estudiarán las ecuaciones de balance en el transitorio para corrinte continua. El trabajar con una corriente continua mayor que la corriente umbral permite disminuir el intervalo de tiempo en el que se operará con $\sqrt{|S|}$ hasta los 0.2 ns, pudiendo realizar un estudio más riguroso de las ecuaciones de balance en el transitorio.

Se considerará una intensidad de corriente $I(t)$ función escalón con $I(t > 0) = I_{Bias}$. En la ecuación 3.1 se muestra la función escalón $I(t)$ utilizada así como las condiciones iniciales para la densidad de portadores de carga $N(0)$, la densidad de fotones $S(0)$ y de la fase óptica $\Phi(0)$.

$$I(t) = \begin{cases} 0 & t \leq 0 \\ I_{Bias} = 30\text{mA} & t > 0 \end{cases} \qquad \begin{array}{c} N(0) = N_{tr} \\ S(0) = 10^{15}\text{m}^{-3} \\ \Phi(0) = 0 \end{array} \tag{3.1}$$

En la Figura 3.2 se muestra la evolución temporal de la corriente de inyección $I(t)$ junto con los valores obtenidos en la simulación para la densidad de portadores de carga $N(t)$ la densidad de fotones $S(t)$ y la fase óptica $\Phi(t)$ para el transitorio $t_{trans} = 1.2$ ns.

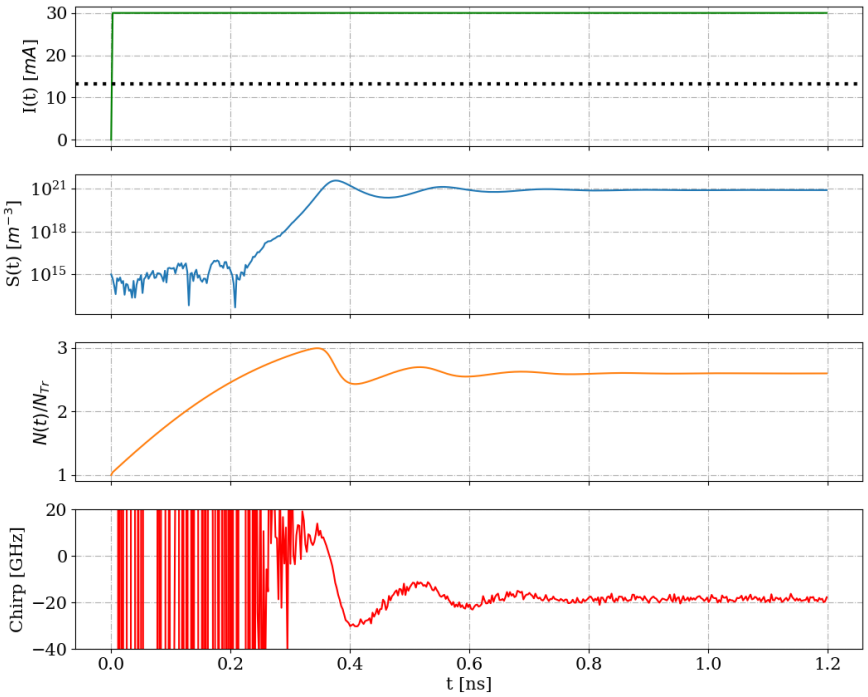


Figura 3.2: *Evolución temporal de la corriente de inyección $I(t)$, la densidad de fotones $S(t)$, la densidad de portadores de carga $N(t)$ y del Chirp durante el transitorio. Para la corriente de inyección $I(t)$ se ha marcado la corriente umbral del láser $I_{th} = 14.8$ mA con una línea horizontal discontinua.*

Se observan en las evoluciones temporales de densidad de portadores de carga $N(t)$, densidad de fotones $S(t)$ y fase óptica $\Phi(t)$ de la Figura 3.2 tres regiones diferentes en función del comportamiento de las tres magnitudes: *i*) Una vez que la corriente inyectada supera la corriente umbral I_{th} (en $t = 0$) la densidad de portadores de carga $N(t)$ comienza a aumentar. No obstante, el valor de $N(t)$ se mantiene inferior a N_{th} por lo que no se produce emisión estimulada, y así, la densidad de fotones no aumenta y toma valores aleatorios, debido a la emisión espontánea, alrededor de $S(0)$. Esto también se puede observar en el comportamiento también aleatorio del *Chirp*. *ii*) La densidad de portadores de carga $N(t)$ continua aumentando alcanzando el valor umbral N_{th} en $t = 0.23$ ns. En este punto la densidad de fotones comienza a aumentar debido a la emisión estimulada producida al superar N_{th}. Sin embargo, al encontrarse $S(t)$ por debajo de S_{th}, $N(t)$ continua creciendo tomando valores por encima N_{th} hasta que $S(t)$ alcanza el valor de S_{th}, en $t \approx 0.37$ ns. Al alcanzar $S(t)$ el valor de S_{th}, comienza a dominar la emisión estimulada frente a la emisión espontánea. Al alcanzar el valor S_{th} la emisión estimulada domina frente a la emisión espontánea, como se puede observar el *Chirp*, y la densidad de portadores de carga comienza a disminuir, alcanzando la densidad de portadores de carga $N(t)$ y *Chirp* un máximo. La densidad de fotones continúa aumentando y la densidad de portadores de carga disminuyendo, llegando nuevamente a tomar valores por debajo de N_{th}. Debido a ésto $S(t)$ alcanza un máximo cuando $N(t) = N_{th}$ y comienza a disminuir, volviendo a tomar valores inferiores a S_{th} y así, volviendo a aumentar $N(t)$. Estas oscilaciones entorno a los valores umbrales continuan, disminuyendo su amplitud, realizando un comportamiento anarmónico y se las conoce como oscilaciones de relajación. En la figura 3.2 se observan claramente estas oscilaciones, siendo iguales en el tiempo para densidad de portadores de carga $N(t)$ y para el *Chirp* (máximos en el mismo tiempo t). También se observa la relación entre las oscilaciones de éstas

magnitudes con las de $S(t)$, obteniendo un máximo en $S(t)$ cuando $N(t) = N_{th}$ de tal forma que ambas oscilaciones tengan el mismo periodo. *ii*) Las oscilaciones de relajación van disminuyendo a medida que el tiempo avanza alcanzando el equilibrio en el que las tres magnitudes se mantienen constante.

A partir de los datos de la figura 3.2 se pueden obtener las frecuencias de las oscilaciones de relajación en el transitorio, a patir del tiempo entre los máximos. Una primera estimación permite obtener una frecuencia de oscilación de $\nu_{RoF} \approx 5.9$ GHz, que pasado a longitud de onda equivale a $\lambda = 0.05$ nm. Comparando dicho valor con los picos debidos a las oscilaciones de relajación de los espectros para $I = 30$ mA de la figura 3.1 observamos que se encuentran en el mismo orden de magnitud, mostrando una gran concordancia entre la simulación y el experimento.

3.2. OFC (Gain-Switching)

Habrá que contar que se va a hacer aqui

3.2.1. Efecto de la amplitud de modulación a altas frecuencias

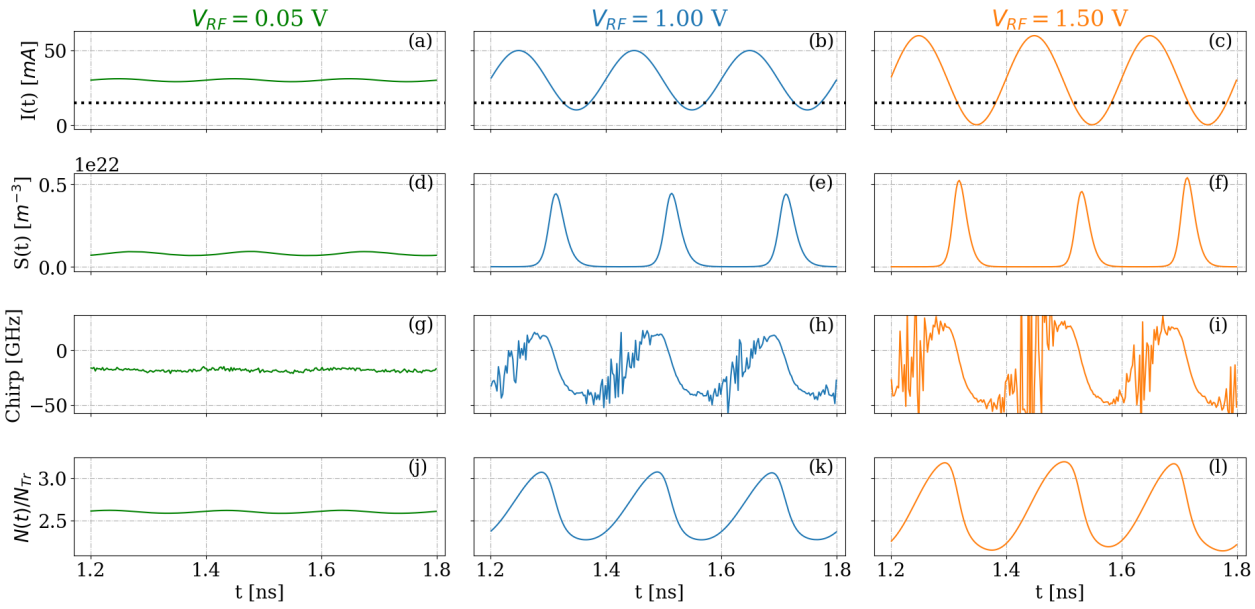


Figura 3.3: *RateEquations*

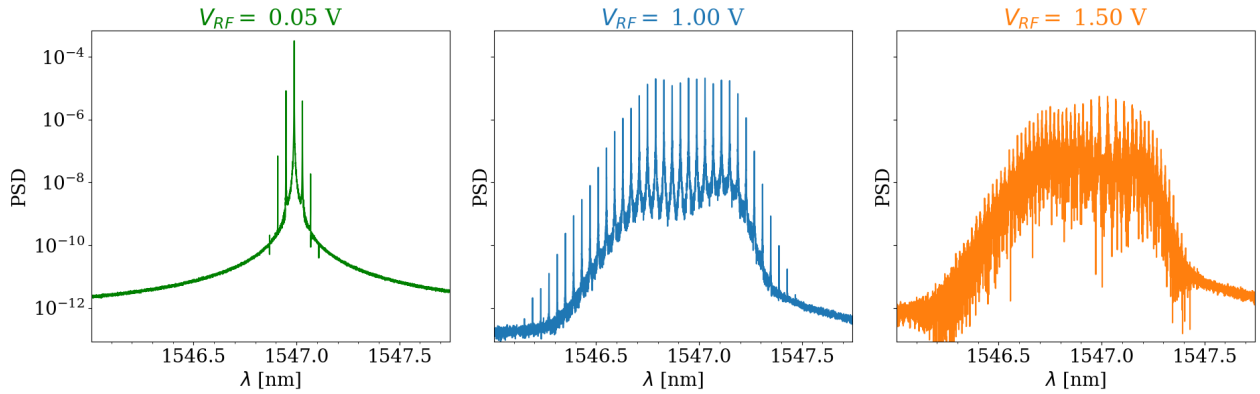


Figura 3.4: *PSD*

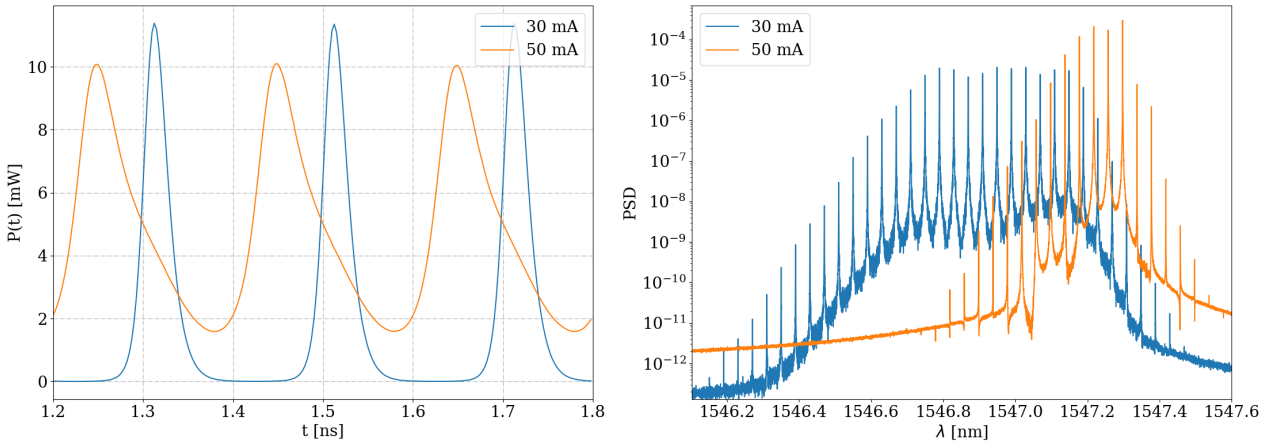


Figura 3.5: *Current*

3.2.2. Efecto de la amplitud de modulación a bajas frecuencias

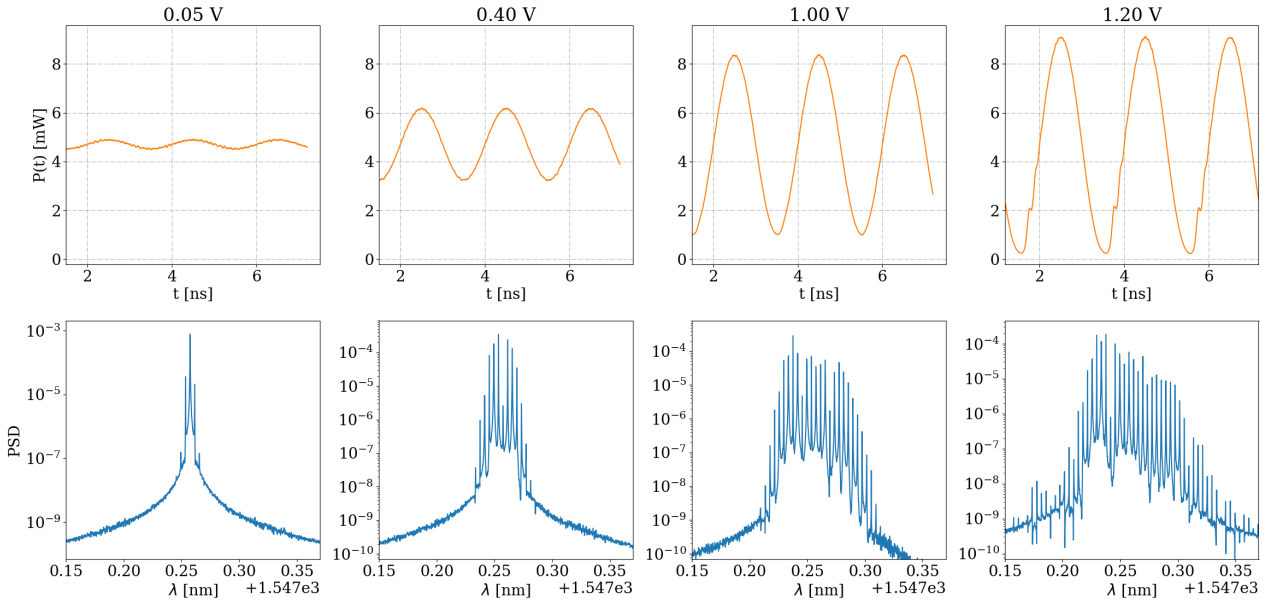


Figura 3.6: *500*

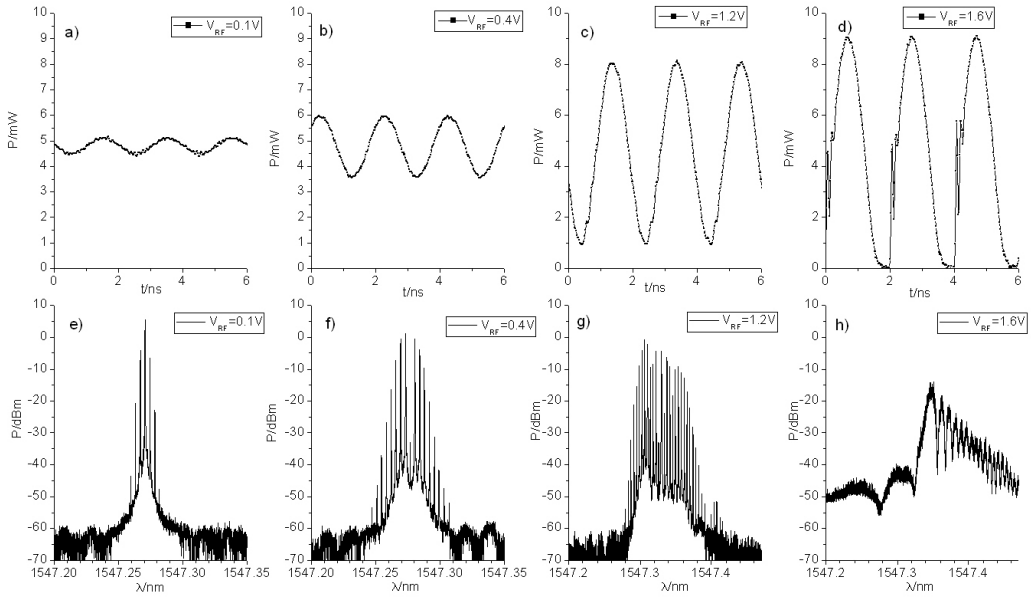


Figura 3.7: *500mhz*

Capítulo 4

Inyeccion de Luz

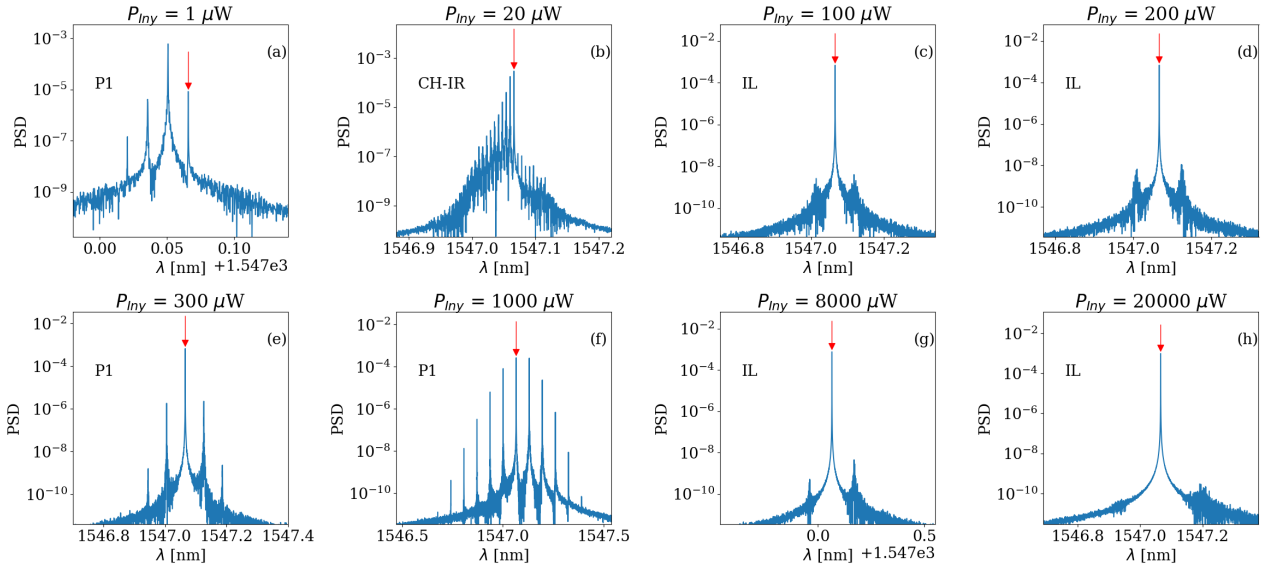


Figura 4.1: *el pie de pagina que le quieras poner a la imagen*

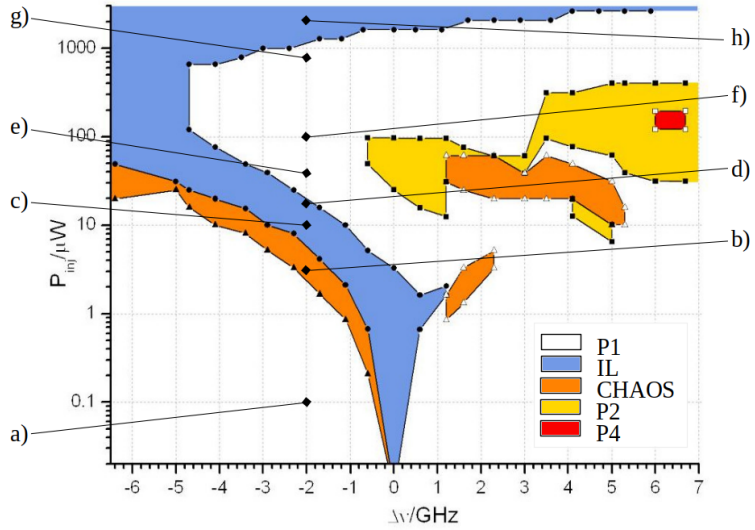


Figura 4.2: *Map*

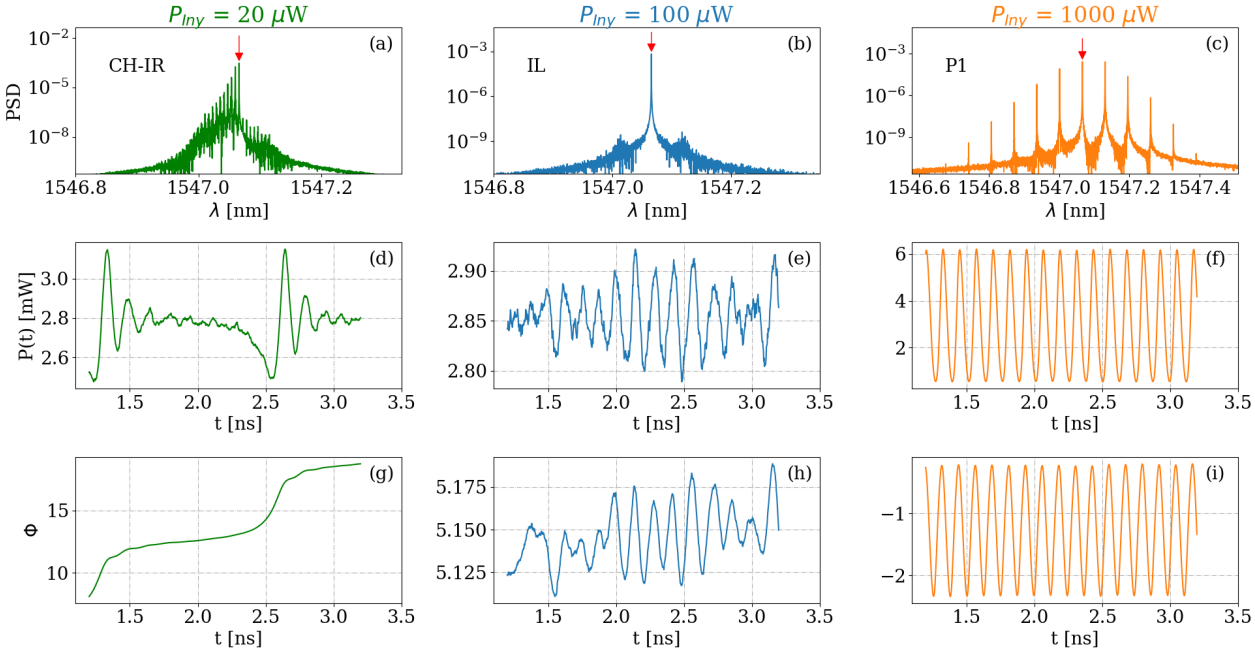


Figura 4.3: *ZoneRtEq*

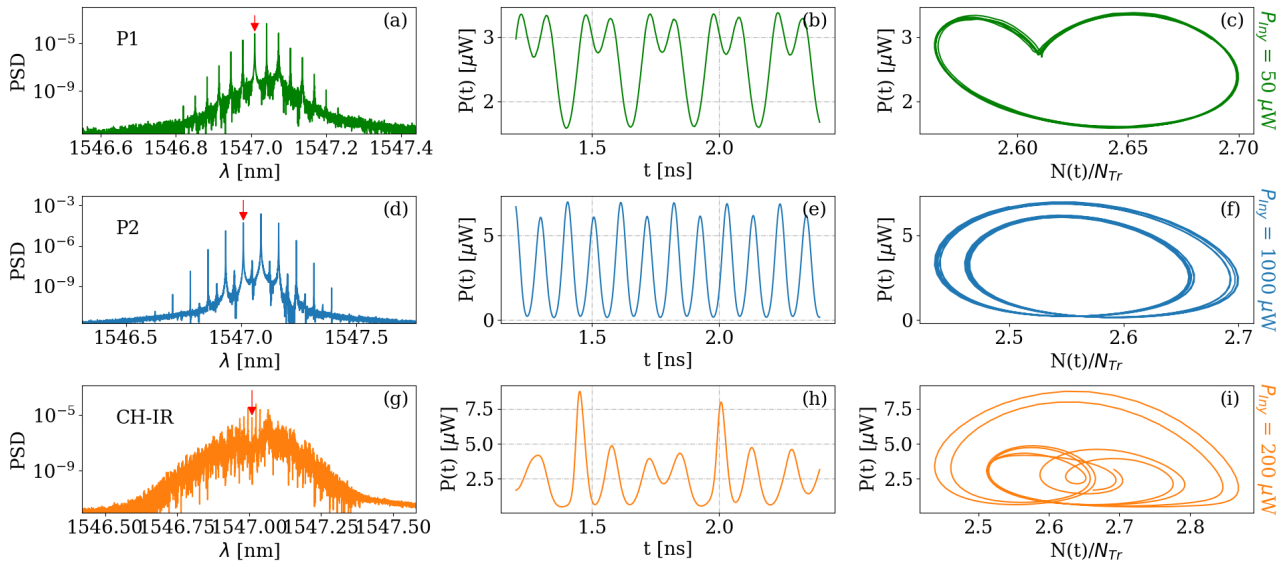


Figura 4.4: *P2zone*

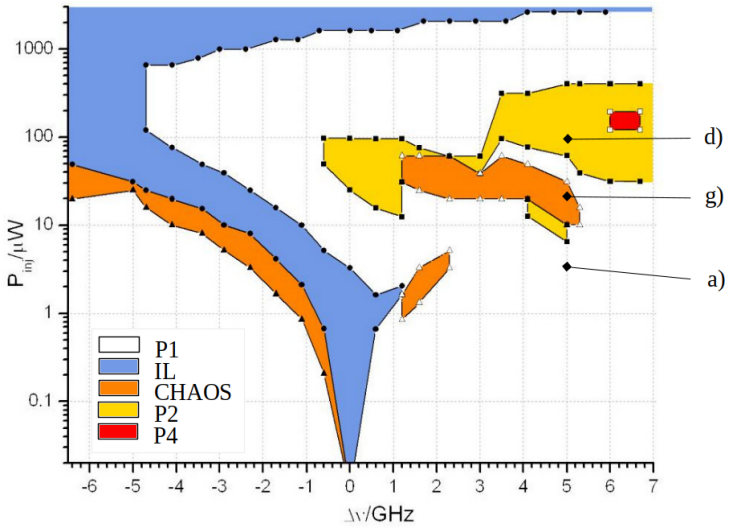


Figura 4.5: *Maps2*

Capítulo 5

Inyeccion de luz en OFC

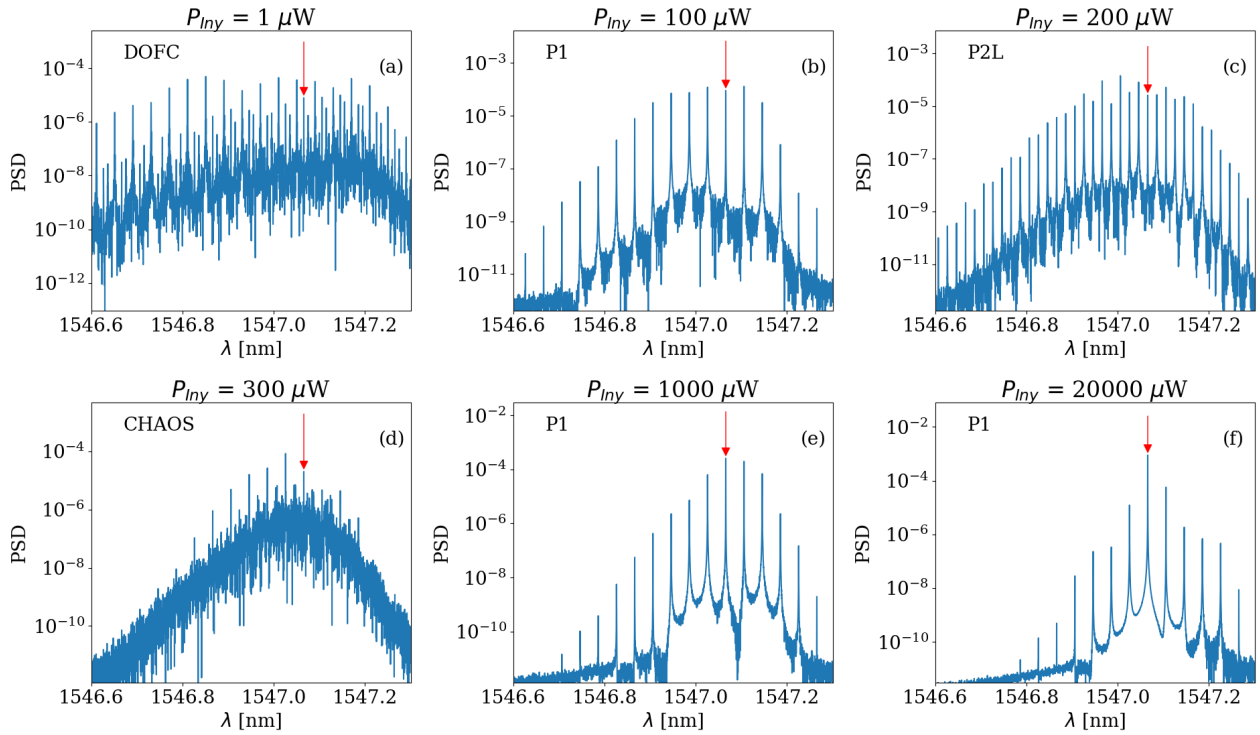


Figura 5.1: *el pie de pagina que le quieras poner a la imagen*

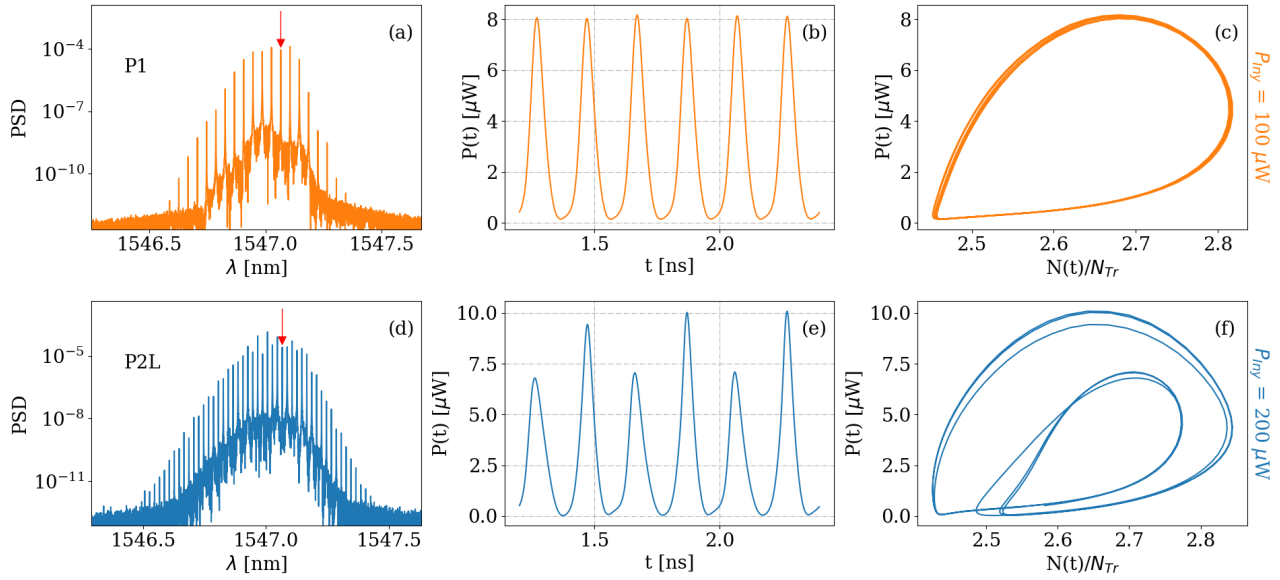


Figura 5.2: *p1-P2*

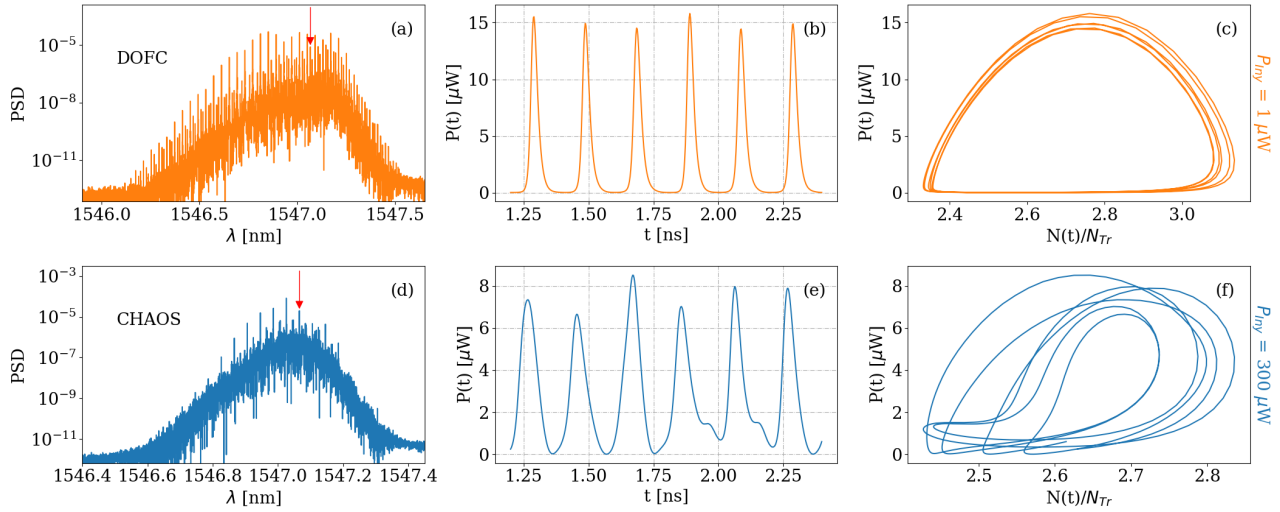


Figura 5.3: *Chaos*

Capítulo 6

Conclusiones

hola a todos

Bibliografía

[1] Diego Chaves Carriles and Ángel Alberto Valle Gutierrez. Peines de frecuencia óptica generados por láseres de semiconductor. *Trabajo Fin de Grado*, 2020.

Apéndice A

Código de la simulación

```
                              +- - - - - - - - - - - - - 18 lines - - - - - - - - - - - - -+
19  import numpy as np
20
21  from Constants import *
22  from getDictValues import *
23
24  class Simulation():
25
26      def __init__(self, iBias, vRF, fR, pwrInjct=0, nuDetng=0, numWindw=1):
                              +- - - - - - - - - - - - - 18 lines - - - - - - - - - - - - -+
44          #                  2 sqrt(2) vRF
45          # I_bias + cLoss ---------------- sin(2 pi fR t)
46          #                     z0 + zL
47          self.current = lambda t: (self.iBias*10**(-12)
48                                    + (cLoss*2.0*np.sqrt(2)*self.vRF
49                                    * np.sin(2*np.pi*fR*t)) / rInt
50                                   )
51
52          self.tWindw = 40.96
53          self.tTrans = 2.2
                              +- - - - - - - - - - - - - 24 lines - - - - - - - - - - - - -+
77          mWindw = int(self.tWindw / delta)
78          mTrans = int(self.tTrans / delta)
79          tTotal = self.tWindw + self.tTrans
80          mTotal = int(tTotal / delta)
81          nTotal = mTotal*ndelta #int(tTotal / tIntev)
                              +- - - - - - - - - - - - - 15 lines - - - - - - - - - - - - -+
96      def allSimulation(self):
                              +- - - - - - - - - - - - - 20 lines - - - - - - - - - - - - -+
116         for win in range(0, self.numWindw):
117
118             #-------------------------------------------------------
119             # Gaussian arrays N(0,1) for the Noise
120             #-------------------------------------------------------
121             X = np.random.normal(0, 1, nTotal)
122             Y = np.random.normal(0, 1, nTotal)
123
124             # Initial conditions are defined in order to resolved the SDE
125             tempN = nTr
126             tempS = float(10**(15))
127             tempPhi = 0
128
129             for q in range(0, mTrans):
```

```
                    for k in range(0, ndelta):

                        index = q*ndelta + k

                        bTN = bTIntv * tempN * tempN
                        invS = 1 / ((1/tempS) + epsilon)
                        sqrtS = np.sqrt(abs(tempS))
                        cosPhi = np.cos(tempPhi)
                        senPhi = np.sin(tempPhi)

                        tempPhi = (tempPhi + aphvgTGmm*tempN - self.phaseTerm
                                   + noisePhi*tempN*Y[index]/sqrtS
                                   - (ampInject/sqrtS)*senPhi*cosInject[index]
                                   + (ampInject/sqrtS)*cosPhi*senInject[index]
                                  )

                        tempS = (tempS + vgTGmm*tempN*invS - vgTGmmN*invS
                                 - intTtau*tempS + btGmm*bTN
                                 + noiseS*tempN*sqrtS*X[index]
                                 + 2*ampInject*sqrtS*cosPhi*cosInject[index]
                                 + 2*ampInject*sqrtS*senPhi*senInject[index]
                                )

                        tempN = (tempN + self.currentTerm[index] - aTIntv*tempN
                                 - bTN - cTIntv*tempN**3 - vgT*tempN*invS + vgtN*invS
                                )
                                +- - - - - - - - - - - - - 10 lines - - - - - - - - - - - - -+
                for q in range(mTrans, mTotal):
                    for k in range(0, ndelta):

                        index = q*ndelta + k

                        bTN = bTIntv * tempN * tempN
                        invS = 1 / ((1/tempS) + epsilon)
                        sqrtS = np.sqrt(tempS)
                        cosPhi = np.cos(tempPhi)
                        senPhi = np.sin(tempPhi)

                        tempPhi = (tempPhi + aphvgTGmm*tempN - self.phaseTerm
                                   + noisePhi*tempN*Y[index]/sqrtS
                                   - (ampInject/sqrtS)*senPhi*cosInject[index]
                                   + (ampInject/sqrtS)*cosPhi*senInject[index]
                                  )

                        tempS = (tempS + vgTGmm*tempN*invS - vgTGmmN*invS
                                 - intTtau*tempS + btGmm*bTN
                                 + noiseS*tempN*sqrtS*X[index]
                                 + 2*ampInject*sqrtS*cosPhi*cosInject[index]
                                 + 2*ampInject*sqrtS*senPhi*senInject[index]
                                )

                        tempN = (tempN + self.currentTerm[index]
                                 - aTIntv*tempN - bTN - (cTIntv*tempN**3)
                                 - vgT*tempN*invS + vgtN*invS
                                )
                                +- - - - - - - - - - - - - 10 lines - - - - - - - - - - - - -+
                    self.opField[q-mTrans] = (np.sqrt(constP*tempS)
                                              * np.exp(1j*tempPhi)
                                              + opFldInject
                                              * np.exp(1j*angInject[index])
                                             )
```

```
                                +- - - - - - - - - - - - - 6 lines - - - - - - - - - - - - -+
212                 transFourier = np.fft.fft(self.opField)
213                 self.TFavg += (abs(np.fft.fftshift(transFourier))
214                                * abs(np.fft.fftshift(transFourier))
215                                / float(self.numWindw)
216                               )
217
218                 self.TFang += (np.angle(np.fft.fftshift(transFourier))
219                                / float(self.numWindw)
220                               )
                                +- - - - - - - - - - - - - 39 lines - - - - - - - - - - - - -+
```