Master's Degree in Computational Mathematics

Master's Thesis

Numerical Solution of the Complex Ginzburg-Landau Equation. Alternatives for its Time Integration

AUTHOR: Miguel Ángel Rapado Tamarit
TUTOR: Fernando Casas Pérez

October 2022

Resumen

La ecuación Ginzburg-Landau compleja en su forma cúbica describe una amplia gama de fenómenos para muchos sistemas de la física y presenta muchas estructuras coherentes estables entre sus soluciones. De ahí que se haya estudiado intensamente a lo largo de los años y que sea un laboratorio ideal para aprender sobre los esquemas de integración temporal. Desde el punto de vista numérico se pueden utilizar métodos pseudoespectrales para resolver la discretización espacial, hay que resolverlo con variable compleja y se pueden utilizar métodos específicos para la integración temporal como la Exponential Time Differencing o Integrating Factor Methods gracias a la presencia de términos lineales y no lineales integrables. El objetivo de esta tesis es comparar algunos de estos métodos analizando su orden y eficiencia. Para ello, se han programado los diferentes esquemas en Fortran y se han validado sus resultados mediante una pila de casos de prueba. A continuación, utilizando uno de ellos como referencia y calculando la solución de una prueba con un paso de tiempo muy pequeño, se han comparado los demás. Entre otros resultados, se ha constatado la buena convergencia de los métodos de orden superior o la mejor velocidad de los métodos que integran analíticamente parte de la ecuación o que necesitan menos transformadas de Fourier.

Abstract

The Complex Ginzburg-Landau Equation in its cubic form describes a wide range of phenomena for many different systems in physics and presents many stable coherent structures within its solutions. Hence, it has been intensively studied over the years and it is an ideal laboratory to learn about time integration schemes. From the numerical point of view; pseudo-spectral methods can be used to solve the spatial discretization, it needs to be solved with complex variables and specific time integration methods like Exponential Time Differencing or Integrating Factor Methods can be used thanks to the presence of integrable linear and nonlinear terms. The purpose of this thesis is to compare some of these methods analysing their order and efficiency. For this purpose, the different schemes have been programmed in Fortran and their results validated using a stack of test cases. Then, using one as a reference and calculating the solution to a test with a very small time step, the others were compared. Among other results, it has been found the good convergence of the higher order methods or the better speed of the methods that analytically integrate part of the equation or need less Fourier transforms.

Contents

List of Figures

Chapter 1

Introduction

1.1 The Complex Ginzburg-Landau Equation

The Complex Ginzburg-Landau Equation (CGLE) in its cubic form has appeared over the years to model a big amount of physical systems in many different branches of physics. It can describe a wide range of phenomena and presents many stable coherent structures within its solutions. Among the systems where this equation becomes important it can be found hydrodynamic flows (Poiseuille flow), flames, reaction-diffusion systems, electro-convection in liquid crystals, chemical reactions or strings in field theory. Some phenomena that can be described with this model are superconductivity, superfluidity, Bose-Einstein condensation, nonlinear waves or second order phase transitions.

The CGLE is usually enclosed in the classification of pattern-forming systems due to the rich bunch of coherent structures and patterns that are found among their solutions. Patterns can be found in spatial extended dynamical systems like for example reaction-diffusion systems (RDS, see appendix A.2). For these systems the model includes transport and linear/nonlinear (or both) interactions, but not advection. They are usually seen as conservation laws in several branches of Physics like chemistry, thermodynamics, fluid mechanics, etc. Then, the field variable (u) under study take the meaning of concentration of chemical species, temperature, pressure, etc.

It was J.T. Stuart and J. Watson in 1960 who obtained the CGLE when studying the behaviour of Poiseuille and Couette flows, [15] and [19]. However, it is not in the scope of this work to understand the models that leads to a CGLE in Physics. Hence, instead of presenting the analysis from the governing equations (for example the Navier-Stokes equation) to the CGLE describing a specific phenomenon, this

work begins with the equation itself in its standard and parameterless form.

The equation in its more general form is usually written as:

$$\frac{\partial A}{\partial t} = c_2\nabla^2 A + c_1 A - c_3|A|^2 A \tag{1.1}$$

$A(x, y, z, t)$ in this expression is a complex function of time and space and plays the role of an amplitude. The parameters $c_1, c_2, c_3 \in \mathbb{C}$ are the linear growth coefficient, the amplitude diffusivity and the nonlinear interaction coefficient respectively. Finally, $\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$ for the three dimensional case. It is common to reduce to unity the real parts of the coefficients c_1, c_2, c_3 by rescaling $A, \vec{x}$ and t. In addition, a transformation like $A \to Ae^{i\phi t}$ (see appendix A.3) allows to discard the imaginary part of c_1, finally obtaining the standard form:

$$\frac{\partial A}{\partial t} = (1 + i\alpha)\nabla^2 A + A - (1 + i\beta)|A|^2 A \tag{1.2}$$

In this equation $\alpha, \beta \in \mathbb{R}$ are known as linear and nonlinear diffusivity. This work is focused on the one dimensional CGLE, for different values of the parameters and the initial condition and with periodic boundary conditions:

$$\frac{\partial A(x,t)}{\partial t} = (1 + i\alpha)\frac{\partial^2 A(x,t)}{\partial x^2} + A(x,t) - (1 + i\beta)|A(x,t)|^2 A(x,t) \tag{1.3}$$

The CGLE is usually explained as an amplitude (or modulational) equation. Actually, the Real Ginzburg-Landau Equation (RGLE), immediately obtained from CGLE by doing $\alpha, \beta = 0$, was first derived as a long-wave amplitude equation in the context of convection for binary mixtures and convection in heated motionless fluids. In the context of waves, amplitude equations describe the evolution of the wave amplitude and the modulation on an spatial scale. More specifically, the CGLE describes an **slow modulation in both space and time for isotropic spatially extended systems near the threshold of a long-wavelength supercritical oscillatory instability (supercritical Hopf bifurcation)** [18].

The appendix A delves into needed concepts to thresh this description of the CGLE. It is usually described as a generalization of the normal form of a Hopf bifurcation for spatially extended systems. Notice that the Hopf bifurcation treated in the appendix A.1 lives only in time, however, a pattern-formation dynamical system needs to exist both in space and time.

The modulation over time and space is defined as slow modulation since the amplitude evolves in a scale much bigger than the oscillation period of the solutions for both independent variables. To complete the description of the CGLE we need to delve into the idea of a finite wavelength instability. For our dynamical system, which is spatially extended, the solution can be decomposed in Fourier modes like $e^{-i\Omega t+ikx}$. These modes can be expressed as:

$$e^{-i\Omega t+ikx} = e^{-i\Re(\Omega)t+\Im(\Omega)t+ikx}$$

so $\Im(\Omega) < 0$, the real part of the exponent, decides the growth rate of each specific mode. When the parameter μ, causing the supercritical oscillatory instability (see appendix A.1), is negative, all these modes are decaying (negative growth rate, $\Im(\Omega) < 0$). However, for $\mu = 0$ there is a wave number k_c whose growth rate becomes 0 and for $\mu > 0$ a narrow band around k_c presents a positive growth rate [7]. For a supercritical bifurcation the non linearities above the bifurcation manifold have small amplitude and then we can expect a finite wavelength, close to $\frac{2\pi}{k_c}$.

1.2 Numerical Treatment

Exactly like the rich behaviour that the CGLE presents in terms of coherent structures and solutions, the numerical methods used to solve it are also varied. It is common to treat the spatial derivatives of the CGLE equation with one of the following methods: finite differences, pseudo-spectral Fourier methods or with the Chebyshev-Tau spectral method [5].

In this work, the Fourier pseudo-spectral method is used taking advantage of the fact that all problems to be solved have periodic boundary conditions (among others). These methods are ideal when the solution is smooth and regular. With them, part of the equation is solved in the Fourier space while the rest is solved in the spatial domain, the Discrete Fourier Transform (DFT) and Inverse Discrete Fourier Transform (IFT) are used to switch between both spaces. We will focus on the one dimensional case for spatial domains of different sizes ($L = 100$ or $L = 200$) centered in $x = 0$.

$$\begin{cases} \frac{d\hat{A}}{dt} = \left(1 - \left(k\frac{2\pi}{L}\right)^2 (1 + i\alpha)\right) \hat{A} - \mathcal{F}\left[(1 + i\beta)|A|^2 A\right] & k \in \left[-\frac{N}{2}, \frac{N}{2} - 1\right), t > 0 \\ \hat{A}(k, 0) = \hat{A}^0(k) \end{cases}$$

As can be checked in the expression above, the linear term of the CGLE in the Fourier space scales with $\left(k\frac{2\pi}{L}\right)^2$ which can be a problem for high wave numbers or said in other words, when we use a high frequencies spectrum (modes varying in really small times). A stiff behaviour appears due to the small time scale needed for explicit methods to solve the linear part, in contrast with the scale needed for the nonlinear terms.

However, several alternatives appear to solve this issue. On one side, Integrator Factor Methods are used to integrate analytically the linear part of the equation, automatically alleviating the stiffness behaviour:

$$\frac{d\hat{A}e^{-qt}}{dt} = -e^{-qt}\mathcal{F}\left[(1+i\beta)|A|^2A\right]$$

Together with this, Exponential Differencing Methods (ETD) are used to solve the exact integration of this equation and then only approximate an integral in the nonlinear terms. For a given nonlinear evolution equation like:

$$\frac{du}{dt} = \mathcal{L}u + \mathcal{N}(u,t) = qu + \mathcal{N}(u,t)$$

the following exact expression is used as origin for the different ETD methods:

$$u_{n+1} = u_n e^{q\Delta t} + e^{q\Delta t}\int_0^{\Delta t} e^{-q\tau}\mathcal{N}(u(t_n+\tau), t_n+\tau)d\tau$$

Finally, Splitting methods (Fractional Step methods or Split-Step methods) are also treated in this work. These methods become a good alternative when the right hand side of our system of ODEs can be split into integrable pieces [3]. For the CGLE, both the linear and the nonlinear term are integrable analytically. They are based on the idea of separating a single time step into different substeps and integrate the system by considering different compositions of the linear and nonlinear terms interchangeably.

Chapter 2

Test Cases

This chapter aims to present the test cases used to validate the code developed to solve the CGLE. The Fortran program is shown and explained in the appendix B and the theoretical foundations of the solvers used are explained in chapter 3. The following test cases are obtained from [7], the same parameters, conditions and time integration method are imitated in order to reproduce the same results. Seven simulations are performed:

- 2 with the aim of finding planar waves, the simplest solution of the CGLE,
- 2 related to the search of disordered states (chaos),
- 2 presenting an intermediate state between planar waves and a chaotic behaviour and
- 1 looking for other coherent structures like holes and shocks.

For the CGLE, to recover the physical field (u) close to the threshold from our complex amplitude (A), the following relation is used:

$$u = A(x,t)e^{-i\Omega_c t + ik_c x} + A^*(x,t)e^{i\Omega_c t - ik_c x} + \text{higher harmonics}$$

or for the real case:

$$u = A(x,t)e^{ik_c x} + A^*(x,t)e^{-ik_c x} + \text{higher harmonics}$$

where the higher harmonics term are proportional to $e^{2ik_c x}$ and k_c denotes the wave number of the mode whose growth rate becomes positive for $\mu > 0$ (see figure 2.1). Notice that this expression indicates that the dynamics of the patterns to be obtained are separated into:

- A fast wave that grows in time but is stationary in space when our parameter is $\mu > 0$ and for the real case ($e^{ik_c x}$). This same component for the complex equation is referred to a travelling wave ($e^{-i\Omega_c t + ik_c x}$). In both cases this fast component is ruled by the critical mode, k_c.
- An slowly varying amplitude in space and time that acts as an envelope for the fast component ($A(x,t)$).
- Higher harmonics.

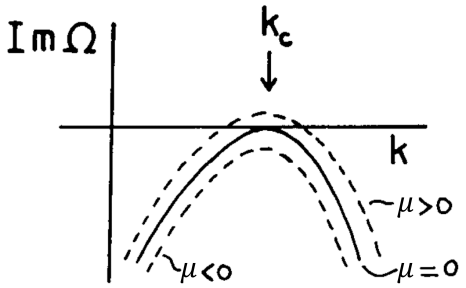


Figure 2.1: In this diagram, obtained from [18], the behaviour of the growth rate according to the variation of the parameter that generates the bifurcation is shown.

The main difference between the real and the complex case of the Ginzburg-Landau equation appears in the critical mode, the one that becomes unstable when $\mu = 0$. For the real case the resulting wave is stationary in space while for the complex dynamics we have a travelling wave. The RGLE allows a band of stationary solutions, periodic in space, like $A(x,t) = a_0 e^{iqx}$ with $q^2 = 1 - a_0^2$ called phase winding solutions. These stationary waves present a wave number above or below (but near) the critical for $q > 0$ and $q < 0$:

$$u = a_0 e^{iqx} e^{ik_c x} + a_0 e^{-iqx} e^{-ik_c x} + \text{higher harmonics}$$

$$u = a_0 e^{i(k_c+q)x} + a_0 e^{-i(k_c+q)x} + \text{higher harmonics}$$

For the complex case the phase winding solutions are also allowed. However, now the band of solutions are travelling waves $A(x,t) = a_0 e^{iqx+i\omega t}$. If we introduce these solutions in our CGLE we find:

$$a_0 e^{iqx+i\omega t}(i\omega) = (1+i\alpha)a_0 e^{iqx+i\omega t}(iq)^2 + a_0 e^{iqx+i\omega t} - (1+i\beta)a_0^2 a_0 e^{iqx+i\omega t}$$

which, separating real and imaginary parts and assuming $\omega \in \mathbb{R}$, leads to:

$$q^2 = 1 - a_0^2 \qquad \omega = -\alpha q^2 - a_0^2\beta$$

If we recover our physical field from the amplitude solution we find:

$$u = a_0 e^{iqx+i\omega t} e^{-i\Omega_c t + ik_c x} + a_0 e^{-iqx-i\omega t} e^{i\Omega_c t - ik_c x} + \text{higher harmonics}$$

$$u = a_0 e^{-i(\Omega_c-\omega)t + i(k_c+q)x} + a_0 e^{i(\Omega_c-\omega)t - i(k_c+q)x} + \text{higher harmonics}$$

and hence, ω measures the difference between the pattern frequency and the critical frequency and also indicates that the pattern frequency is depending on the wave number (q) and the pattern amplitude (a_0). This proves that α rules the linear dispersion and β rules the nonlinear dispersion.

As it is explained in [18], our CGLE is expressed in a frame that moves with the so called group velocity: $v_{gr} = \partial\omega/\partial k$ which is generally distinct from 0. When we want to study if the instability is convective or absolute and we want the equation to be expressed in a fixed frame of reference, an additional term is included in the left hand side of the equation: $v_{gr}\partial A/\partial x = \partial\omega/\partial k \cdot \partial A/\partial x$.

2.1 Planar waves

We've just seen that periodic travelling waves are some simple solutions of the CGLE. In this section we review the basics around the stability of these solutions following the same process as in [7]. Let's define $\tilde{a}_\pm$ as small amplitude perturbations, $\lambda \in \mathbb{C}$ and let's find solutions of the form:

$$A = \left(a_0 + \tilde{a}_+ e^{ikx+\lambda t} + \tilde{a}_- e^{-ikx+\lambda^* t}\right) e^{iqx+i\omega t}$$

By introducing this solution in the CGLE and expanding the resulting equation for λ around $k \sim 0$ the following expression is obtained:

$$\lambda = -2iq(\alpha - \beta)k - \left[1 + \alpha\beta - \frac{2q^2(1+\beta^2)}{a_0^2}\right] k^2 + \mathcal{O}(k^3)$$

Since we need the real part of the growth rate (λ) to be negative to have stable solutions in the long wave-length scale and $q, \alpha, \beta, k, a_0 \in \mathbb{R}$, we need that the following condition holds:

$$1 + \alpha\beta - \frac{2q^2(1+\beta^2)}{a_0^2} > 0$$

or, using the relation between the amplitude a_0 and the wave number q, we can say:

$$q^2 < \frac{\frac{1+\alpha\beta}{2(1+\beta^2)}}{1 + \frac{1+\alpha\beta}{2(1+\beta^2)}} = \frac{1+\alpha\beta}{2\beta^2 + 3 + \alpha\beta}$$

Analysing what happens for $q = 0$ or $|q| << 0$ we find the so-called Benjamin-Feir-Newell criterion:

$$1 + \alpha\beta > 0 \qquad \text{or} \qquad \alpha\beta > -1$$

and for the case $\alpha = \beta$ we have the Eckhaus condition: $q^2 < 1/3$. As a conclusion, all the periodic solutions are Benjamin-Feir unstable when $\alpha\beta < -1$. In the figure

2.2 obtained from [18] we can see how the stability is related to the values of q, α and β. The instability region is located within the dashed lines while the region that admits planar wave solutions is located within the solid line.

For the RGLE (or for α, β small) and for a given value $\mu > 0$ generating the instability, our planar waves are only stable for $|q| << 0$. Said in other words, only the solutions with a total wave length closer to the critical are stable. By making the product $\alpha\beta$ higher (in absolute value), when $\alpha\beta \leq -1$, all the region with planar wave solutions becomes unstable.

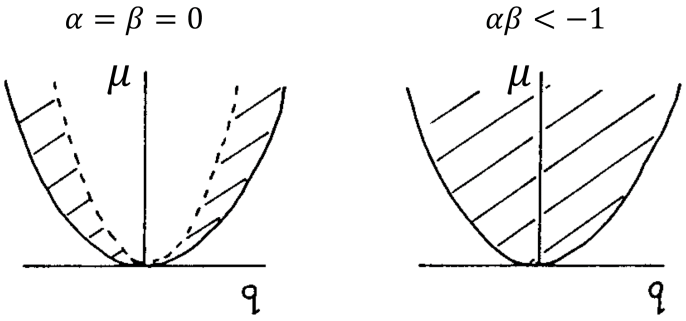


Figure 2.2: In this diagram, obtained from [18], the stability for the RGLE ($\alpha = \beta = 0$) and CGLE is shown depending on the values of μ and q. The region within the solid line admits planar waves and the region within the dashed line present stable planar waves.

The waves described until now differ in several properties from the waves that can be obtained as solutions of the linear wave equation like $\partial^2 U/dt^2 = \Delta U$. For linear waves the frequency does not depend on the amplitude of the wave and the amplitude does not depend on the wave number q. As we have seen, for the nonlinear waves obtained with the CGLE, the frequency ω is a function of amplitude and the amplitude a_0 depends on the wave number. In addition, the linear waves dissipate energy and lose amplitude but the nonlinear waves are consuming energy and do not decay. Finally, the linear waves do not interact between each other while for nonlinear waves we find shocks and collisions.

The first two simulations presented in [7], which we are using as test cases, find plane waves with two different initial conditions. To do that it chooses α, β such that there are a stable range of wave numbers q: $\alpha = 1, \beta = 2$. With these values we have $\alpha\beta > -1$ and we can expect stable solutions for $q^2 < 3/13$.

2.1.1 Test Case 1: Planar waves stable regime

By using an initial condition of small noise ($\pm 0.01 \pm 0.01i$) around $A = 0 + 0i$ the CGLE quickly converges to the expected planar waves (see figure 2.3). Since α, β ensures that we are in the stable regime and the noise is small enough, the solution does not diverge. Actually, $|A|$ quickly converges to a constant value while $\Re(A)$ and $\Im(A)$ present the planar waves.

$$1 + \alpha\beta = 3 > 0$$

Notice that all the tests made include periodic boundary conditions so the equation can be transformed to the Fourier space. The solver used for these simulations is the so-called Exponential Time-stepping 2 (ETD2).

2.1.2 Test Case 2: Planar waves unstable regime

To reproduce the Benjamin-Feir instability we can force the equation to start with an unstable wave as initial state. Notice that for the given same parameters used in the previous case, the unstable regime is obtained for:

$$q > \sqrt{\frac{1 + \alpha\beta}{2\beta^2 + 3 + \alpha\beta}} \simeq 0.480$$

so let's use an initial condition with a wave number of $q = \frac{20\pi}{L} = 0.6283 > 0.480$. The results show in the first place that the initial wave travels in space (see figure 2.4) until it mutates into a new plane wave with a stable wave number and generating in the change defects called phase singularities ($A = 0$). Notice that the solution for $t = 100$ has a wave number of around $q = 3\frac{2\pi}{L} = 0.1885$ (being in the stable region).

Test Case 1	
(α, β)	$(1, 2)$
Spatial domain	$x \in [-50, 50]$
Grid points	`Nl = 512`
Final time and timestep	`tf = 100, h = 0.05`
Initial condition	noise: $(\pm 0.01 \pm 0.01i)$
Solver used	Pseudospectral method + ETD2

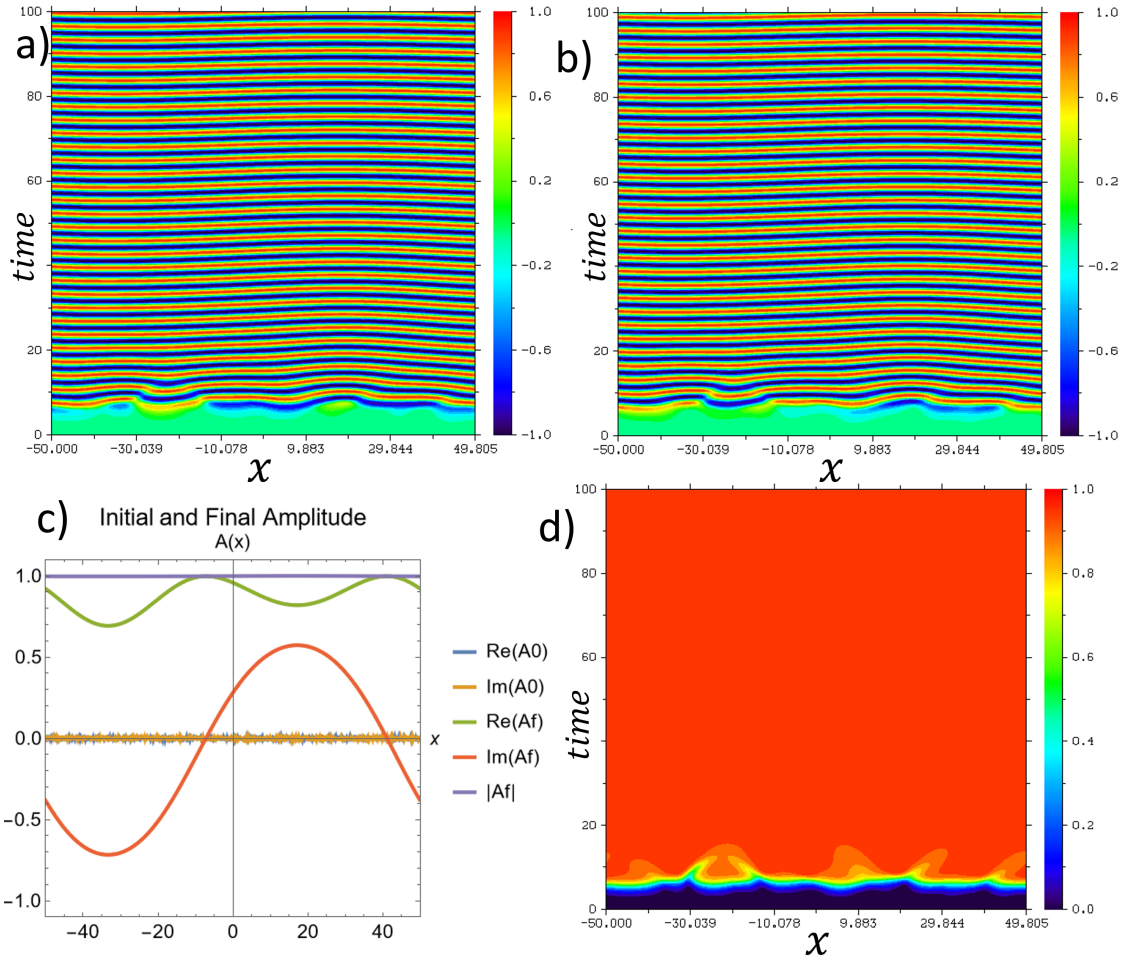


Figure 2.3: Space-time plots for a) $\Re(A)$, b) $\Im(A)$, c) Initial condition and final result and d) $|A|$ for Test Case 1.

Test Case 2	
(α, β)	$(1, 2)$
Spatial domain	$x \in [-50, 50]$
Grid points	`Nl = 512`
Final time and timestep	`tf = 100, h = 0.05`
Initial condition	$\sqrt{1 - \left(\frac{20\pi}{L}\right)} e^{i\left(\frac{20\pi}{L}\right)x} + \text{noise}$
Solver used	Pseudospectral method + ETD2

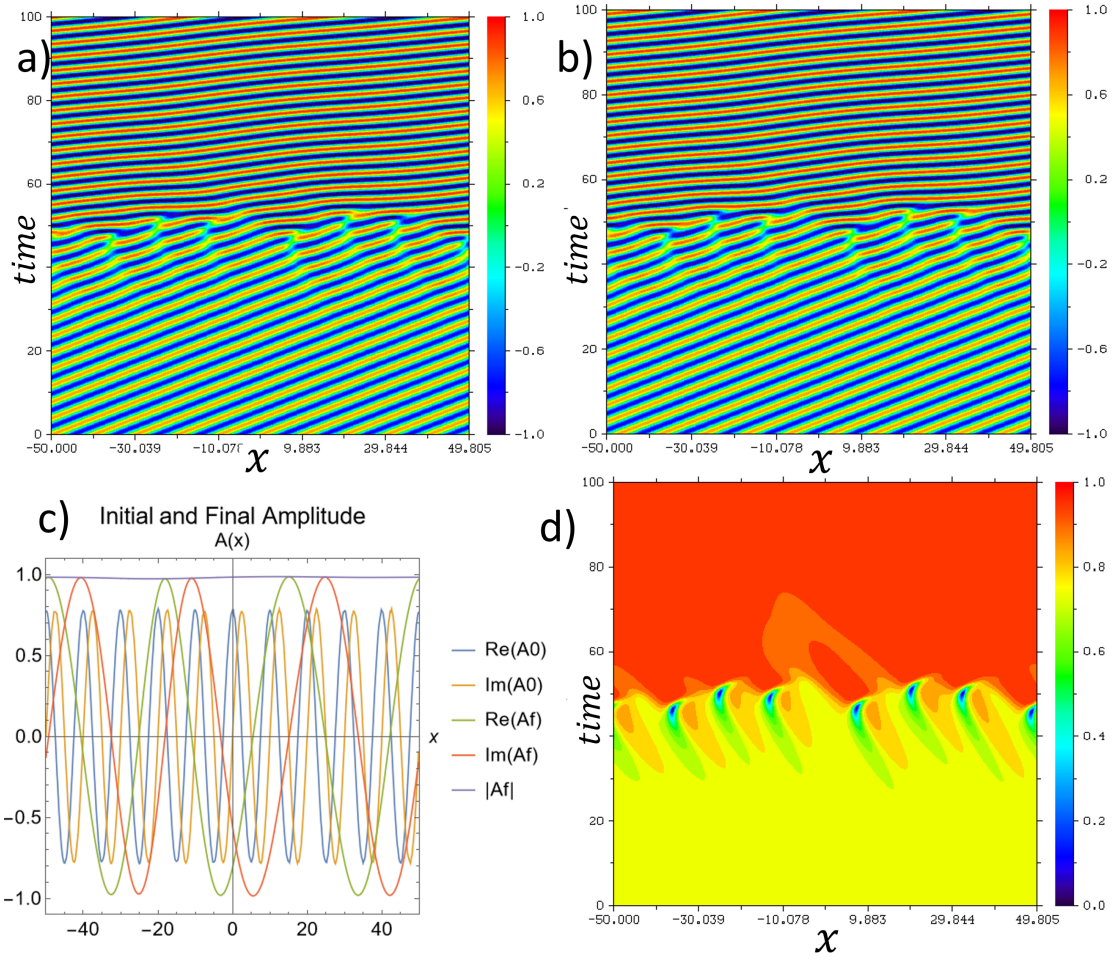


Figure 2.4: Space-time plots for a) $\Re(A)$, b) $\Im(A)$, c) Initial condition and final result and d) $|A|$ for Test Case 2.

2.2 Spatio-temporal chaos

2.2.1 Test Case 3: Phase turbulence

When the Benjamin-Feir-Newell criterion is violated (with $q \simeq 0$) and for the proper initial condition, a spatio-temporal chaotic state appears. This can happen for any non zero spatially constant initial condition like for example the one used in the following case tests ($A_0 = 1 + noise$). In this case we are using $\alpha = 2, \beta = -1$:

$$1 + \alpha\beta = -1 < 0$$

The first state encountered is phase chaos, this appears when the Benjamin-Feir-Newell line is crossed (above BF line plane waves are unstable) coming from the non chaotic region in the parameter space (see figure 2.5). Here, $|A|$ remains saturated, never reaches 0, and typically stays in values around $0.7 - 0.9$. The solutions obtained (see figure 2.6) show that the phase of the solution for different times presents a turbulence regime.

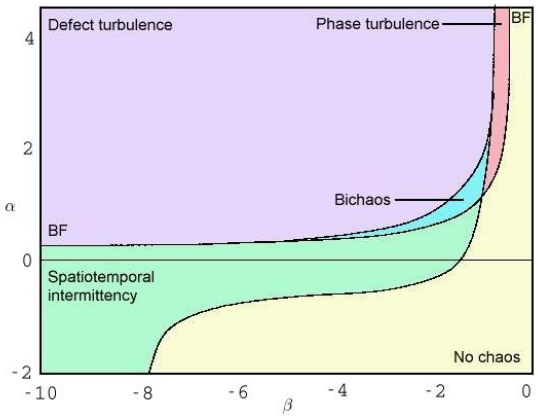


Figure 2.5: This diagram obtained form [12] shows the different regions in the parameter space that can be found in the 1D CGLE.

2.2.2 Test Case 4: Defect turbulence regime

In this spatio-temporal chaos, the same initial conditions can generate the so-called defect chaos when we make β smaller. Notice that in figure 2.5 we have crossed to the Defect Turbulence region with for example $\alpha = 2, \beta = -2$. This region is characterised by defects where $|A| = 0$ (see figure 2.7).

Test Case 3	
(α, β)	$(2, -1)$
Spatial domain	$x \in [-100, 100]$
Grid points	`Nl = 512`
Final time and timestep	`tf = 500, h = 0.05`
Initial condition	$1 +$ noise
Solver used	Pseudospectral method + ETD2

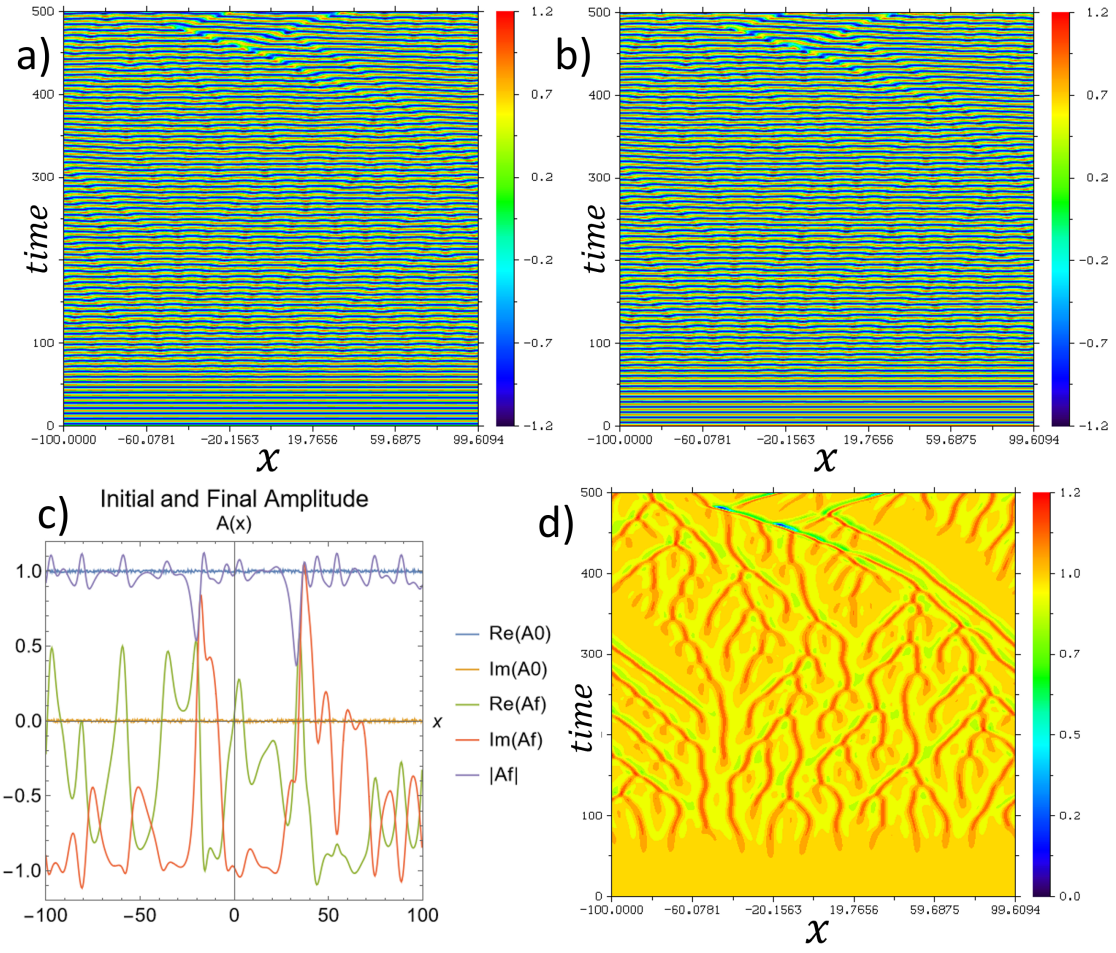


Figure 2.6: Space-time plots for a) $\Re(A)$, b) $\Im(A)$, c) Initial condition and final result and d) $|A|$ for Test Case 3.

Test Case 4	
(α, β)	$(2, -2)$
Spatial domain	$x \in [-100, 100]$
Grid points	`Nl = 512`
Final time and timestep	`tf = 100, h = 0.05`
Initial condition	$1 +$ noise
Solver used	Pseudospectral method + ETD2

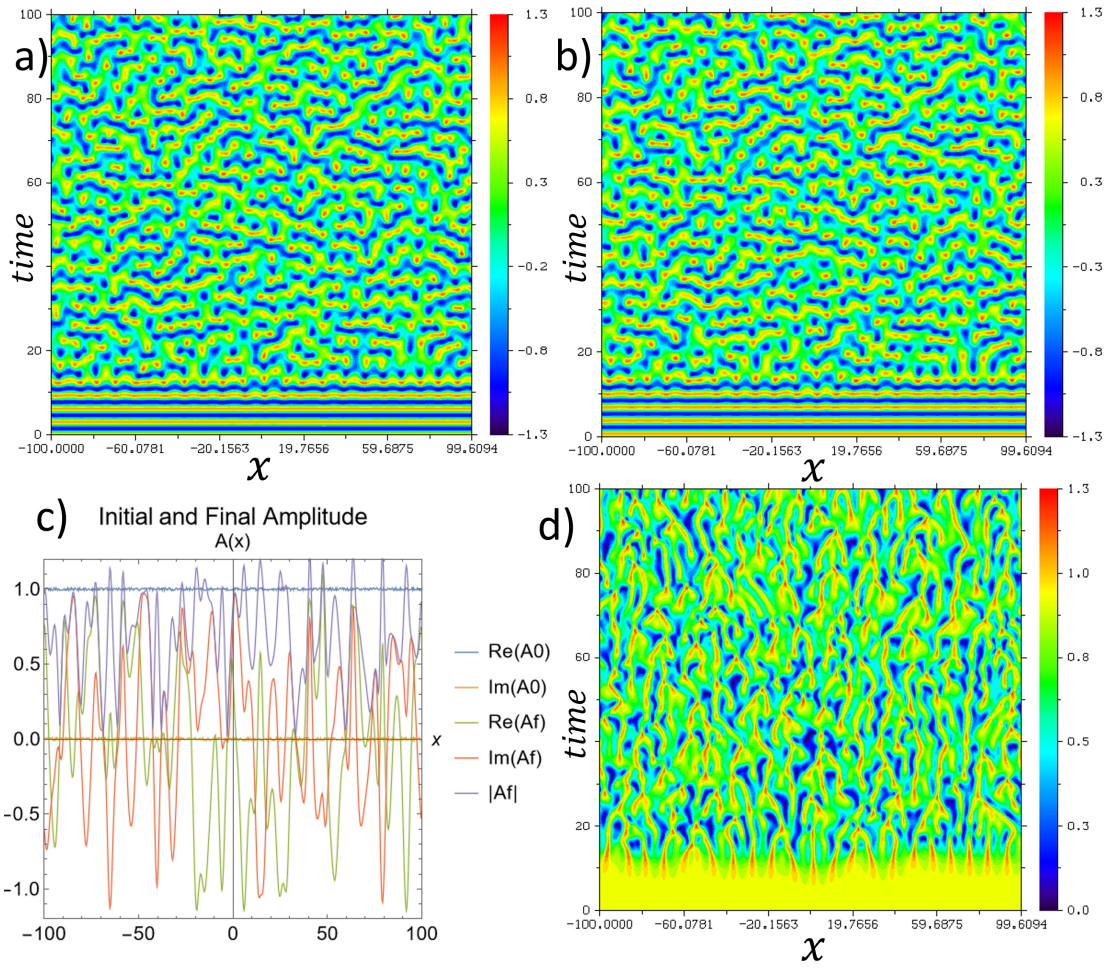


Figure 2.7: Space-time plots for a) $\Re(A)$, b) $\Im(A)$, c) Initial condition and final result and d) $|A|$ for Test Case 4.

2.3 Intermittency regime

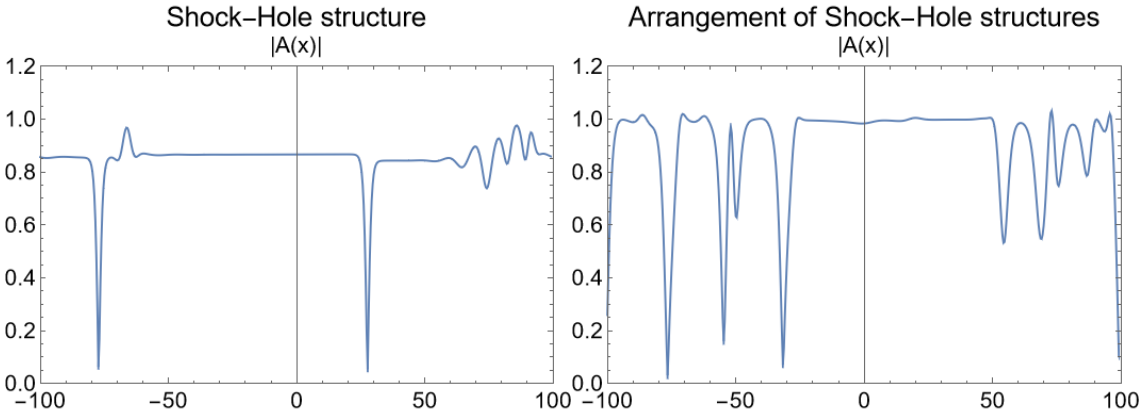
2.3.1 Test Case 5: plane waves + localized structures

The spatio-temporal intermittency regime is characterised by the coexistence of the stable planar waves with defect chaos. In order to obtain this regime, the initial condition includes localised pulses of amplitude preventing the equation to form only planar waves. As a result, after a transient period, localized structures ($|A| \sim 0$) separate patches of planar waves (constant $|A|$).

In this first case with ($\alpha = 0.5, \beta = -1.5$, right side of the region of spatio-temporal intermittency in figure 2.5) the moving localised structures are related to Nozaki-Bekki holes. These coherent structures in the form of sinks can be found analytically and they are characterised by a dip in $|A|$ moving at constant speed and generating planar waves on both sides but with different wave numbers ($q_r \neq q_l$). The behaviour is as follows: if the phase gradient of the hole spreads out the hole disappears but if the phase gradient becomes stronger, then a phase slip happens and two or more localised structures appears (see figure 2.9).

2.3.2 Test Case 6: turbulence + stationary shock-hole structures

In this second intermittency case simulated the parameters used are $\alpha = 0, \beta = -4$. With β decreasing we are nearer the turbulence region. Now, few stable planar waves coexist with arrangements of stationary holes and shocks that are separated by turbulence regions. Shocks appear naturally when two or more holes coexist creating the arrangements of these structures (see figure 2.8).



(a) Two shocks and holes.

(b) Arrangement os shocks and holes

Figure 2.8: A figure with two subfigures

Test Case 5	
(α, β)	$(0.5, -1.5)$
Spatial domain	$x \in [-100, 100]$
Grid points	`Nl = 512`
Final time and timestep	`tf = 100, h = 0.05`
Initial condition	$\text{sech}\left((x+10)^2\right) + 0.8\text{sech}\left((x-30)^2\right) + \text{noise}$
Solver used	Pseudospectral method + ETD2

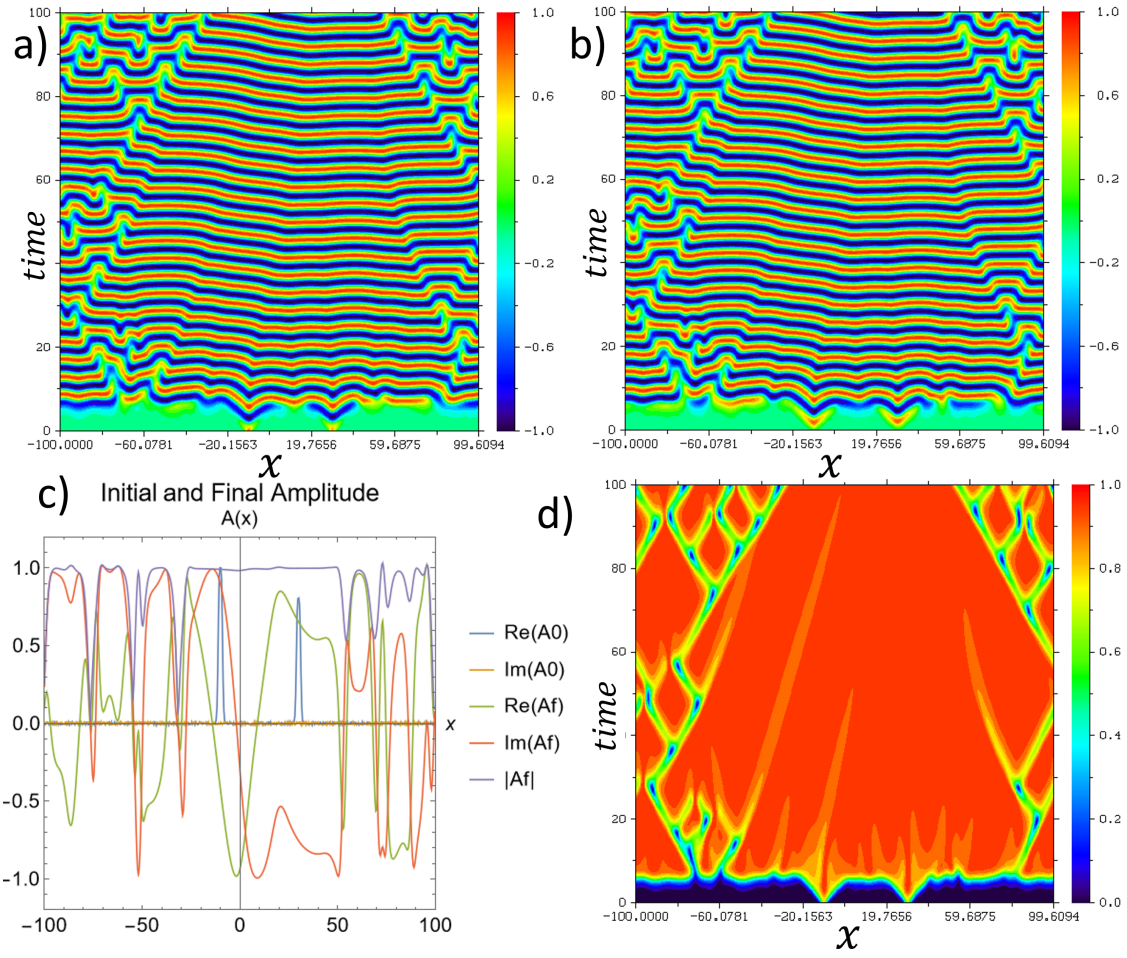


Figure 2.9: Space-time plots for a) $\Re(A)$, b) $\Im(A)$, c) Initial condition and final result and d) $|A|$ for Test Case 5.

Test Case 6	
(α, β)	$(0, -4)$
Spatial domain	$x \in [-100, 100]$
Grid points	`N1 = 512`
Final time and timestep	`tf = 100, h = 0.05`
Initial condition	$\text{sech}\left((x+\frac{L}{4})^2\right) + 0.8\text{sech}\left((x-\frac{L}{4})^2\right) + \text{noise}$
Solver used	Pseudospectral method + ETD2

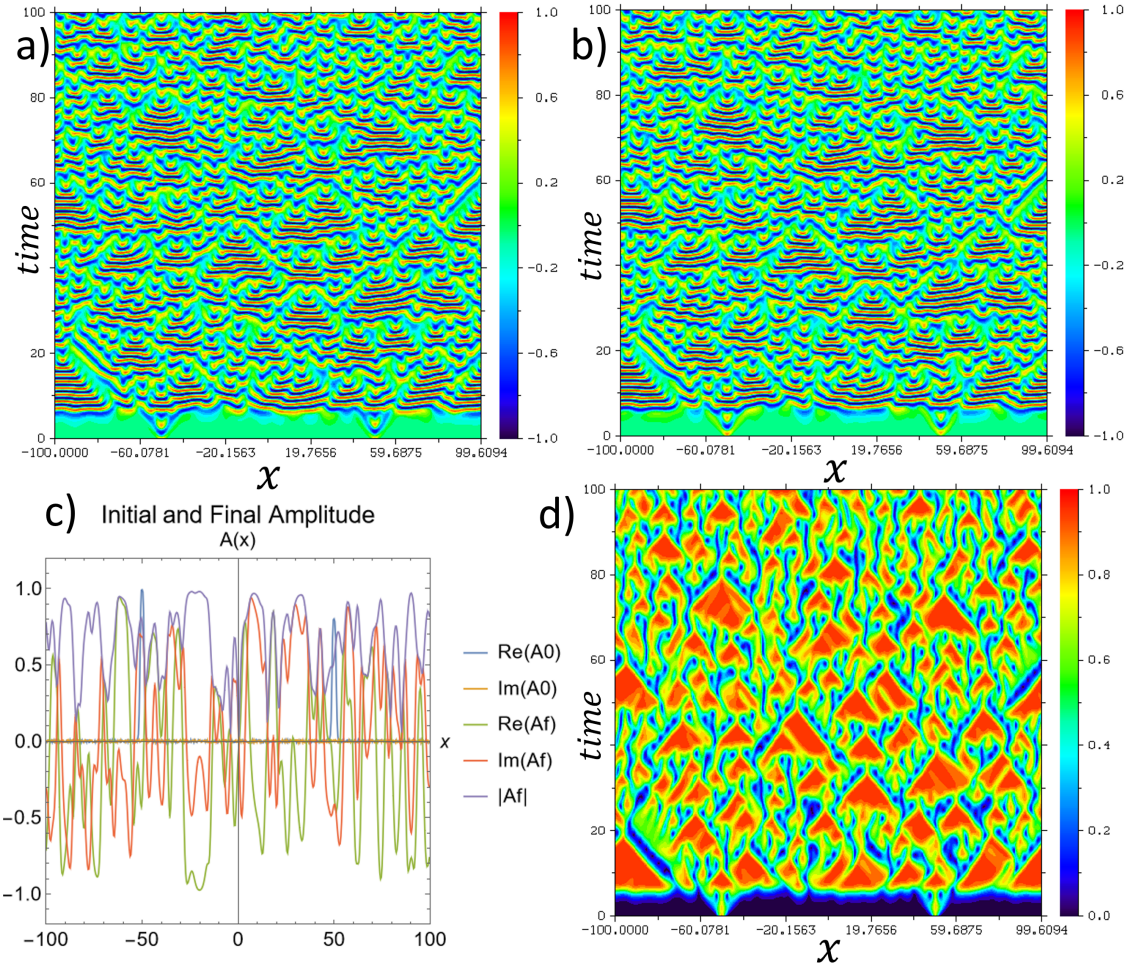


Figure 2.10: Space-time plots for a) $\Re(A)$, b) $\Im(A)$, c) Initial condition and final result and d) $|A|$ for Test Case 6.

2.4 Coherent structures

2.4.1 Test Case 7: Moving hole - shock pair

For this test, with parameters $\alpha = 0, \beta = 1.5$, we find two examples of the already mentioned Nozaki-Bekki holes, which are a kind of coherent structure (like planar waves). In this case the equation is outside the chaotic region so we expect to find planar waves together with the holes. In general, coherent structures are characterized by presenting regular patterns which are connected by interfaces. For the one dimensional case these structures can be found with the general form:

$$A(x,t) = a(x - \nu t)e^{i\phi(x-\nu t)-i\omega t}$$

where a, ϕ are real functions. If this solution is introduced in the CGLE, a system of three ODEs is obtained. In general, there are no exact analytical description for a sink (shock) that connects two travelling waves with arbitrary wave numbers, however, the so-called Nozaki-Bekki holes have an analytical expression but they are referred to a specific selection of the wave numbers of both sides waves.

Test Case 7	
(α, β)	$(0, 1.5)$
Spatial domain	$x \in [-100, 100]$
Grid points	`Nl = 512`
Final time and timestep	`tf = 500, h = 0.05`
Initial condition	noise: $(\pm 0.01 \pm 0.01i)$
Solver used	Pseudospectral method + ETD2

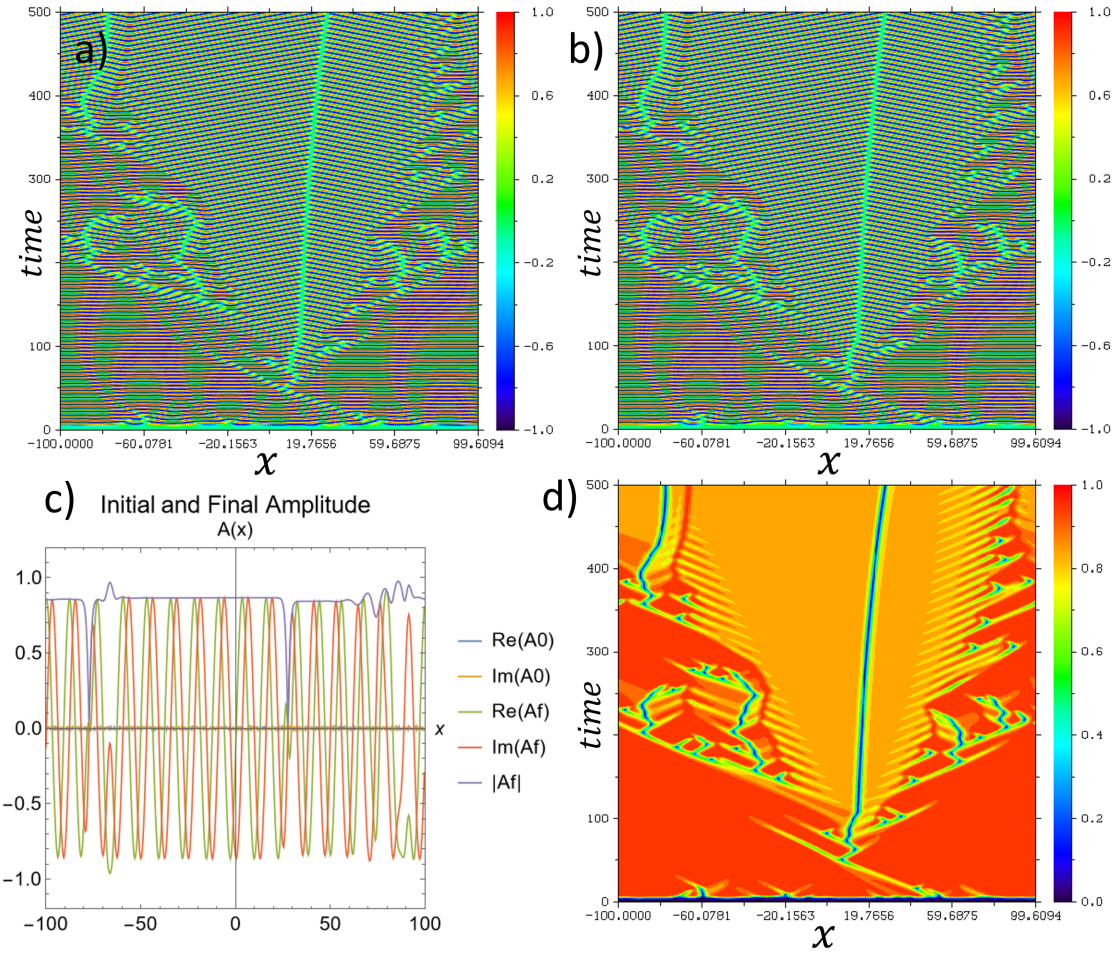


Figure 2.11: Space-time plots for a) $\Re(A)$, b) $\Im(A)$, c) Initial condition and final result and d) $|A|$ for Test Case 7.

Chapter 3

Numerical treatment

This chapter aims to give the theoretical framework of the different methods used to solve the time integration of the CGLE. The problem to solve is:

$$\begin{cases} \frac{\partial A}{\partial t} = (1+i\alpha)\frac{\partial^2 A}{\partial x^2} + A - (1+i\beta)|A|^2 A, \quad x \in \left[-\frac{L}{2}, \frac{L}{2}\right), t > 0 \\ A(x,0) = A^0(x) \\ A(-\frac{L}{2}, t) = A(\frac{L}{2}, t), \quad \frac{\partial A}{\partial x}(-\frac{L}{2}, t) = \frac{\partial A}{\partial x}(\frac{L}{2}, t) \end{cases}$$

with $A = A(x,t) \in \mathbb{C}$, $\alpha, \beta \in \mathbb{R}$ and $|\cdot|$ the modulus of a complex magnitude.

3.1 Spatial discretization

Taking advantage of the periodic boundary conditions, the spatial derivatives in this problem are suitable to be addressed with pseudo-spectral methods. With an spectral approach, the solution $A(x,t)$ would be expanded in an infinite sequence of orthogonal functions (basis functions) that, from the numerical point of view, we later have to truncate. This set of basis functions could be for example sines or cosines (expansion in Fourier series). We then need a fast convergence of the expansion so the truncated series approximate properly our solution function (spectral precision). However, in the pseudo-spectral approach, part of our equation is solved pointwise in the spatial domain (nonlinear term) and part in the frequency domain (space derivatives). In our case, the pseudo-spectral approach is performed by means of Fourier transforms.

With our numerical approximation we expect to find our solution $A(x,t)$ in a discrete and bounded grid. Hence, both the spatial and the frequency domain, are discrete and bounded (see figure 3.1), and the discrete Fourier Transform (DFT) is the suitable tool for this analysis [16]. It is clear that:

$$x_j = \frac{2\pi j}{N} \qquad \text{for } j = 0, ..., N-1$$

and

$$h = \frac{2\pi}{N}$$

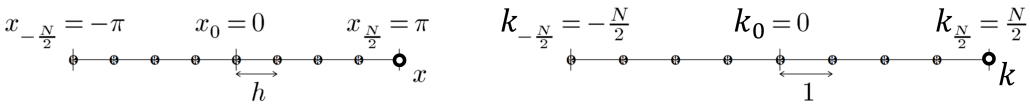


Figure 3.1: Discrete and bounded spatial and frequency (wave number) domains used in the DFT (obtained from [16]).

Let's obtain the DFT from the Discrete Fourier series [10], which for a given function $u(x)$, is the interpolant:

$$I_N(u) = \sum_{k=-\frac{N}{2}}^{k=\frac{N}{2}-1} \tilde{u}_k e^{ikx}$$

now let's force this interpolant to pass through our equispaced N grid points (see figure 3.1), called collocation points $(x_j, u(x_j))$:

$$u(x_j) = \sum_{k=-\frac{N}{2}}^{k=\frac{N}{2}-1} \tilde{u}_k e^{ikx_j} \qquad \text{for } j = 0, ..., N-1$$

This linear system of equations can be inverted to find the N coefficients $\tilde{u}_k$ depending on the collocation points. Similar to inverting the matrix we can multiply this equation by e^{-ikx_j} and sum for $j = 0, ..., N-1$ obtaining $(u_j = u(x_j))$:

$$\tilde{u}_k = \frac{1}{N} \sum_{j=0}^{j=N-1} u_j e^{-ikx_j} \qquad \text{for } k = -\frac{N}{2}, ..., \frac{N}{2} - 1$$

While this equation is the DFT, the previous allows us to recover the original discrete function $u(x_j)$ once we have the frequency components $\tilde{u}_k$ and is called the Inverse Discrete Fourier Transform (IFT). Some notes has to be mentioned for these expressions of the DFT-IFT:

1. In general, both u_j and $\tilde{u}_k$ are complex and N is an even integer quantity (for odd N the expressions differ).

2. The DFT can also be seen as an approximation of the expressions that gives the coefficients of a Fourier series for a periodic function:

   $$\tilde{u}_k = \frac{1}{2\pi} \int_0^{2\pi} u(x) e^{-ikx} dx$$

   when the trapezoidal rule is used to evaluate the integral.

3. For this analysis we've obtained the expression of the DFT which is normalized with the number of grid points $1/N$. Different expressions for the same purpose can be obtained ($1/\sqrt{N}$ normalization, positive exponent, etc.). However, with this expression the frequency components obtained can be used as coefficients for the continuous Fourier series.

4. Notice that the precaution of using square-integrable measurable functions or square-summable grid functions are needed for the Fourier transform and semi-discrete Fourier transform. However, for the DFT, with the correspondent norm:

   $$\|u\| = \left[h \sum_{j=-\frac{N}{2}}^{\frac{N}{2}-1} |u_j|^2 \right]^2$$

   we are sure that any N-periodic function on the grid $\{x_j\}$ is going to be square-summable on $\{x_j\}$ (since the sum is finite).

The previous expression for the DFT is developed for a fundamental spatial domain $[-\pi, \pi)$. Translations of this interval do not affect it but lengths different than 2π needs from a scale factor. For our CGLE equation, since the spatial domain has not necessarily length 2π, we would need this scale factor [17], however, another option is to transform the equation to the domain $[-\pi, \pi)$. For a given domain $x \in \left[-\frac{L}{2}, \frac{L}{2}\right)$, we can define:

$$\hat{x} = \frac{2\pi}{L}x$$

and redefine our problem as:

$$\begin{cases} \frac{\partial A}{\partial t} = (1+i\alpha)\frac{\partial^2 A}{\partial \hat{x}^2}\left(\frac{2\pi}{L}\right)^2 + A - (1+i\beta)|A|^2 A, \quad x \in [-\pi, \pi), t > 0 \\ A(\hat{x}, 0) = A^0(\hat{x}) \\ A(-\pi, t) = A(\pi, t), \quad \frac{\partial A}{\partial \hat{x}}(-\pi, t) = \frac{\partial A}{\partial \hat{x}}(\pi, t) \end{cases}$$

Finally, in the Fourier space, our problem can be written as:

$$\begin{cases} \frac{d\hat{A}}{dt} = \left(1 - \left(k\frac{2\pi}{L}\right)^2 (1+i\alpha)\right)\hat{A} - \mathcal{F}\left[(1+i\beta)|A|^2 A\right] \quad k \in \left[-\frac{N}{2}, \frac{N}{2} - 1\right), t > 0 \\ \hat{A}(k, 0) = \hat{A}^0(k) \end{cases}$$

Notice that we have converted the problem in a system of ODEs (with each component a frequency in the Fourier space) in time that can be treated as a Cauchy problem (or initial value problem).

3.2 Time integration

The initial value problems or Cauchy problems are evolution problems in the sense that, given an initial state, a numerical scheme is used to find the state in the next instant. Hence, from a given initial condition, the solution of the system of ODEs is obtained in discrete points of the independent variable. Normally, the independent variable is the time, however, it is not mandatory and some boundary value problems can be also stated as Cauchy problems.

Our start point to find the numerical schemes for the time integration is the following system of dimension s of first order ODEs (in general higher order equations/systems can be written as first order systems):

$$\frac{du(t)}{dt} = F(u, t) \qquad \text{with} \quad u(t_0) = u^0$$

where $u(t), u^0 \in \mathbb{C}^s$, $F : \mathbb{C}^s \times \mathbb{R} \to \mathbb{C}^s$. Notice that this equation is called autonomous when $F = F(u)$, which means that the function F does not depend explicitly on time. To obtain the solution the discretization is normally based on finding an approximation to both the function $F(u,t)$ and $\frac{du(t)}{dt}$ [9]. Hence, the problem is reduced to a system of difference equations like:

$$G(u^{n+1}, u^n, ..., u^{n+1-p}) = 0$$

with initial conditions and where p is the number of steps involved and $u^j = c^j$ (for $j = 0, ..., p-1$) are constants used to start the process. The value u^n is the approximation to $u(t)$ for $t = t_n$ so, in order to solve the problem, we need to find u^{n+1} based on the previous values u^n for all $n \geq p$. We are normally interested in finding the order q of a numerical scheme since this let us ensure that the global error $E^n = u(t_n) - u^n$ of our solution in a specific time can be bounded in the following way:

$$E^n = \mathcal{O}(\Delta t^q)$$

for Δt closely enough to 0. Notice that this global error is expressed as the difference between the exact solution of the equation and our approximation so our solution is closer to the exact value in the same order than Δt^q. Finally, it is common to denote F^n the values $F(u^n, t_n)$ and $\Delta t_n = t_{n+1} - t_n$.

The temporal schemes are usually categorised into different groups. For example a scheme is called multistep when the value in the instant u^{n+1} is obtained from p previous steps $((u^j, F^j))$ or single step when only one previous state is needed to determine it. They are called explicit when we can clear the value of u^{n+1} in the expression of $G(u^{n+1}, u^n, ..., u^{n+1-p})$ and write:

$$u^{n+1} = f(u^n, ..., u^{n+1-p})$$

or implicit when this is not possible and we have to solve a system of s nonlinear equations for each time step. Another interesting classification is made if the temporal scheme comes from a numerical integration or from a numerical differentiation. For the first case we can integrate the system of ODEs to obtain:

$$u^{n+1} = u^n + \int_{t_n}^{t_n+1} F(u,t)dt \tag{3.1}$$

and find the value of the integral by means of an interpolant or extrapolant polynomial of the function F. For the second case the interpolant polynomial is found for the function $u(t)$ using the instants $u^0, u^1, ..., u^{n+k}$, it is derived and forced to accomplish with our system of ODEs in the instant t_{n+k}:

$$\left(\frac{du(t)}{dt}\right)_{n+k} = F(u^{n+k}, t_{n+k}) \tag{3.2}$$

For our CGLE we are going to start using three well known schemes: Explicit Euler, second and fourth order Runge-Kutta. The three of them are single step and explicit schemes (simpler to implement than implicit schemes). However, the Runge-Kutta schemes are known as multistage methods since for each time step, e intermediate stages are considered between t_n and t_{n+1} to find u^{n+1}.

The explicit Euler, the simplest scheme, can be obtained by assuming that $F(u,t)$ is constant between t_n and t_{n+1}. Hence, we can evaluate the integral in the equation 3.1 as $(t_{n+1} - t_n)F^n$ resulting in the first-order method:

$$u^{n+1} = u^n + (t_{n+1} - t_n)F^n$$

Runge-Kutta schemes of e stages can be written with the general expression:

$$u^{n+1} = u^n + \Delta t \sum_{i=1}^{e} b_i k_i$$

where:

$$k_i = F(u^n + \Delta t \sum_{j=1}^{e} a_{ij} k_j, t_n + c_i \Delta t) \qquad \text{for} \quad i = 1, ..., e$$

and a_{ij}, c_i and b_i are specific constant for each numerical scheme. In these methods the function $F(u,t)$ needs to be evaluated in the intermediate points between $[t_n, t_{n+1}]$, to do that we need an estimation of $u(t)$ in the points since we do not know its real value. Once these stages are known, by integrating in 3.1 we obtain the value u^{n+1}.

The two stages and second-order method Runge-Kutta 2 can be built by estimating the values of $F(u,t)$ in the stages t_n and t_{n+1}. For t_n we already have the estimation for u^n from the previous step, so we can write:

$$k_1 = F(u^n, t_n)$$

For the estimation of u^{n+1} an explicit Euler can be used:

$$k_2 = F(u^n + \Delta t_n k_1, t_n + \Delta t_n)$$

With these values we can build the interpolant polynomial for $F(u,t)$ which, after being integrated, builds the scheme:

$$u^{n+1} = u^n + \frac{\Delta t_n}{2}(k_1 + k_2)$$

A fourth order Runge-Kutta scheme follows the expression:

$$u^{n+1} = u^n + \frac{\Delta t_n}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

with

$$\begin{cases} k_1 = F(u^n, t_n) \\ k_2 = F(u^n + 1/2\Delta t_n k_1, t_n + 1/2\Delta t_n) \\ k_3 = F(u^n + 1/2\Delta t_n k_2, t_n + 1/2\Delta t_n) \\ k_4 = F(u^n \Delta t_n k_3, t_n + \Delta t_n) \end{cases}$$

With no need of more manipulations in the equation or knowledge, we could already integrate in time the CGLE (code `subroutine CGLE`). The solution can be obtained just by calling:

$$F(\hat{A}, t) = \left(1 - \left(k\frac{2\pi}{L}\right)^2 (1 + i\alpha)\right)\hat{A} - \mathcal{F}\left[(1 + i\beta)|A|^2 A\right]$$

and using any of the described schemes for the vector of amplitudes $\hat{A}$ in the Fourier space:

$$\begin{cases} \frac{d\hat{A}}{dt} = F(\hat{A}, t) \quad k \in \left[-\frac{N}{2}, \frac{N}{2} - 1\right), t > 0 \\ \hat{A}(k, 0) = \hat{A}^0(k) \end{cases}$$

However, the method of Integrating Factor Method (IFM) can be also used with the CGLE. It consists on transforming the equation so that its linear part is solved analytically. Let's consider the variable change $\varphi = \hat{A}e^{-qt}$ with $q(k) = \left(1 - \left(k\frac{2\pi}{L}\right)^2 (1 + i\alpha)\right)$:

$$\frac{d\varphi}{dt} = \frac{d\hat{A}}{dt}e^{-qt} + \hat{A}e^{-qt}(-q)$$

and let's multiply our CGLE by e^{-qt}:

$$e^{-qt}\left(\frac{d\hat{A}}{dt} - q\hat{A}\right) = -e^{-qt}\mathcal{F}\left[(1 + i\beta)|A|^2 A\right]$$

$$\frac{d\varphi}{dt} = -e^{-qt}\mathcal{F}\left[(1 + i\beta)|A|^2 A\right] = F(\varphi, t)$$

Again, now any of the described schemes could be used over the vector φ in the Fourier space (code `subroutine CGLE_IFM_Gral`). Notice that several changes of variables are needed with this approach (dividing and multiplying by e^{-qt}). Instead of that, let's manipulate more the equation assuming that we want to use a Explicit Euler method (the same can be done for other explicit schemes). Let's integrate in time the previous equation assuming that $-e^{-qt}\mathcal{F}\left[(1 + i\beta)|A|^2 A\right]$ is constant in the interval:

$$\hat{A}_{n+1}e^{-qt_{n+1}} = \hat{A}_n e^{-qt_n} - \Delta t_n e^{-qt_n}\mathcal{F}\left[(1 + i\beta)|A_n|^2 A_n\right]$$

which, multiplying by $e^{qt_{n+1}}$, leads to (code `subroutine CGLE_IFM_Euler`):

$$\hat{A}_{n+1} = e^{q\Delta t_n}\left(\hat{A}_n - \Delta t_n \mathcal{F}\left[(1 + i\beta)|A_n|^2 A_n\right]\right)$$

It is common when solving Partial Differential Equations (PDEs) with spectral and pseudo-spectral methods that the systems of ODEs are stiff [4]. This happens because these equations involves time-dependent Fourier coefficients where a mode n-th, for large n, scales as $\mathcal{O}(n^{-m})$ with m the order of the highest spatial derivative. Let's take our CGLE equation for example, the term $\left(k\frac{2\pi}{L}\right)^2$ scales as $\mathcal{O}(k^2)$ since we have a second derivative. The time scale for that wave number scales as k^{-2} and, when we have high frequencies ($k \sim 256$) the evolution is happening in really short times. It is usually the linear term the responsible for this stiffness behaviour. The Exponential Time Differencing (ETD) methods born to solve this issue by performing an exact integration for the equation followed by an approximation of an integral related to the nonlinear term.

Let's obtain first the generic methods ETD1 and ETD2 and then apply them to the CGLE. For a given nonlinear evolution equation like:

$$\frac{du}{dt} = \mathcal{L}u + \mathcal{N}(u,t) = qu + \mathcal{N}(u,t)$$

with $u = u(k,t)$, $u(k,0) = u_0(k)$ and denoting $\mathcal{L}$ and $\mathcal{N}$ the linear and nonlinear operators. We can multiply it by the integrating factor e^{-qt} and integrate from t_n to t_{n+1}:

$$\int_{t_n}^{t_{n+1}} \frac{d\left(ue^{-qt}\right)}{dt} dt = \int_{t_n}^{t_{n+1}} e^{-qt}\mathcal{N}(u,t)dt$$

$$u_{n+1}e^{-qt_{n+1}} = u_n e^{-qt_n} + \int_{t_n}^{t_{n+1}} e^{-qt}\mathcal{N}(u,t)dt$$

Multiplying by $e^{qt_{n+1}}$ and defining $\tau = t - t_n$:

$$u_{n+1} = u_n e^{q\Delta t_n} + e^{q\Delta t_n} \int_0^{\Delta t_n} e^{-q\tau}\mathcal{N}(u(t_n+\tau), t_n+\tau)d\tau$$

This is an exact result and the variations of the ETD method are obtained by approximating the integral. Assuming that $\mathcal{N}(u,t) = \mathcal{N}(u_n,t_n)$ is constant in the whole time interval ($\mathcal{N} = \mathcal{N}_n + \mathcal{O}(\Delta t)$) we find the ETD1 method:

$$u_{n+1} = u_n e^{q\Delta t_n} + e^{q\Delta t_n}\left(\frac{-1}{q}\right)\mathcal{N}_n\left(e^{-q\Delta t_n} - 1\right)$$

$$u_{n+1} = u_n e^{q\Delta t_n} + \mathcal{N}_n \left(\frac{e^{q\Delta t_n} - 1}{q} \right)$$

When we use a higher order approximation for $\mathcal{N}$, let's say:

$$\mathcal{N} = \mathcal{N}_n + \tau \frac{\mathcal{N}_n - \mathcal{N}_{n-1}}{\Delta t_n} + \mathcal{O}(\Delta t^2)$$

and using $\int \tau e^{-q\tau}\, d\tau = -\frac{e^{-q\tau}(q\tau+1)}{q^2} + C$:

$$\begin{aligned} u_{n+1} = u_n e^{q\Delta t_n} + e^{q\Delta t_n} \Bigg[& \left(\frac{-1}{q} \right) \mathcal{N}_n \left(e^{-q\Delta t_n} - 1 \right) + \\ & + \left(\frac{\mathcal{N}_n - \mathcal{N}_{n-1}}{\Delta t_n} \right) \left(\frac{-1}{q^2} \right) \left(e^{-q\Delta t_n}(1 + q\Delta t_n) - 1 \right) \Bigg] \end{aligned} \tag{3.3}$$

the result is the so-called ETD2:

$$u_{n+1} = u_n e^{q\Delta t_n} + \frac{\mathcal{N}_n}{\Delta t_n q^2} \left(e^{q\Delta t_n} (\Delta t_n q + 1) - 2\Delta t_n q - 1 \right) + \frac{\mathcal{N}_{n-1}}{\Delta t_n q^2} \left(1 + q\Delta t_n - e^{q\Delta t_n} \right)$$

For our equation it is immediate to write the expressions for both the ETD1 and ETD2 (codes `subroutine CGLE_ETD1` and `subroutine CGLE_ETD2`) respectively:

$$\hat{A}_{n+1} = \hat{A}_n e^{q\Delta t_n} - \mathcal{F}\left[(1 + i\beta) |A_n|^2 A_n \right] \left(\frac{e^{q\Delta t_n} - 1}{q} \right) \tag{3.4}$$

$$\begin{aligned} \hat{A}_{n+1} = \hat{A}_n e^{q\Delta t_n} & - \frac{\mathcal{F}\left[(1 + i\beta) |A_n|^2 A_n \right]}{\Delta t_n q^2} \left(e^{q\Delta t_n} (\Delta t_n q + 1) - 2\Delta t_n q - 1 \right) - \\ & - \frac{\mathcal{F}\left[(1 + i\beta) |A_{n-1}|^2 A_{n-1} \right]}{\Delta t_n q^2} \left(1 + q\Delta t_n - e^{q\Delta t_n} \right) \end{aligned} \tag{3.5}$$

Notice that the first step for the scheme ETD2 must be done with a different method since it needs two previous steps to solve $\hat{A}_{n+1}$ and we only count with

the initial condition in one instant of time. ETD1 is being used for this purpose in the program developed. It is also interesting to remark that both schemes, for $|q| \to 0$ converge to other methods. While ETD2 converges to a second-order Adams-Bashforth method, the method ETD1 converge to the explicit Euler (or forward Euler) when no integrating factor is used, we can prove it by expanding $e^{q\Delta t_n}$ for $|q| \to 0$:

$$\begin{aligned} u_{n+1} &= u_n \left(1 + q\Delta t_n + \mathcal{O}\left((q\Delta t)^2\right)\right) + \mathcal{N}_n \left(\frac{1 + q\Delta t_n + \mathcal{O}\left((q\Delta t)^2\right) - 1}{q}\right) \simeq \\ &\simeq u_n + \Delta t_n \left(u_n q + \mathcal{N}_n\right) \end{aligned} \tag{3.6}$$

A precaution that must be taken with these methods is related to the low values of $q(k) = \left(1 - \left(k\frac{2\pi}{L}\right)^2 (1 + i\alpha)\right)$ since both schemes are divided by this value.

The last kind of schemes treated in this work are the splitting methods. More specifically the first-order method Lie-Trotter and the second-order method Strang Splitting are implemented. These methods appear as a natural response to equations (or systems of equations) with different terms representing different physics. This could be the example of the Navier-Stokes equations or Reaction-Diffusion PDEs like the CGLE. As we have seen, the two terms in our CGLE equation (already expressed in the Fourier space) represent two phenomena that occur in different time scales. As a result, a stiff problem is encountered, splitting methods can be used to speed up the calculation. A splitting method allows to use different numerical methods for different processes, for example manipulating a nonlinear term separated from a linear one. Hence, the integration can be optimized under some conditions.

Let's first get a basic idea of how they work. Imagine that we have a system of ODEs:

$$\frac{du}{dt} = f_0(u) + f_1(u)$$

with both f_0 and f_1 linear functions ($f_i = L_i u$). We can then write our solution as:

$$u(t) = u_0 e^{(L_1 + L_2)t}$$

On a single time step, the integration can be expressed as:

$$u_{n+1} = u_n e^{(L_1+L_2)\Delta t} = u_n e^{L_1\Delta t} e^{L_2\Delta t}$$

Then, the time step could be solved with the following two steps:

$$u^* = u_n e^{L_1\Delta t}$$
$$u_{n+1} = u^* e^{L_2\Delta t}$$

The order of the steps does not matter. While this method is exact for both functions being linear, when the functions are nonlinear this method is giving a first order approximation to our solution. To give a formal description of this method let's consider the following ODEs system as initial value problem [3]:

$$\frac{du}{dt} = f(u) \qquad \text{with} \quad u_0 = u(0) \in \mathbb{C}^D$$

where $f : \mathbb{C}^D \to \mathbb{C}^D$ and D is the dimension of the system. We assume that this function f can be decomposed in a sum of functions $f = \sum_{i=1}^{m} f^{[i]}(u)$ with $f^{[i]} : \mathbb{C}^D \to \mathbb{C}^D$ and such that:

$$\frac{du}{dt} = f^{[i]}(u) \qquad \text{with} \quad u_0 = u(0) \in \mathbb{C}^D \qquad \text{and} \quad i = 1, ..., m$$

can be integrated exactly or, in case that it is integrated approximately, they have to involve an improvement with respect to the integration of the complete equation, whether in terms of efficiency, speed, stability, etc. It is specially interesting in the case of the CGLE since the component functions can be integrated analytically.

Let's call $u(\Delta t) = \varphi^{[i]}_{\Delta t}(u_0)$ to the solution of the i-th equation along one time step of size Δt. Then, the composition of the m solutions $(\chi_{\Delta t})$ provides a first-order approximation to the exact solution (called $\varphi_{\Delta t}(u_0)$):

$$\chi_{\Delta t} = \varphi^{[m]}_{\Delta t} \circ ... \circ \varphi^{[2]}_{\Delta t} \circ \varphi^{[1]}_{\Delta t}$$

Let's use this method with our CGLE already expressed in the Fourier space (equation 3.1). We split it in the following two equations (linear and nonlinear terms) using $q = \left(1 - \left(k\frac{2\pi}{L}\right)^2 (1 + i\alpha)\right)$:

$$\frac{d\hat{A}}{dt} = q\hat{A}$$

$$\frac{d\hat{A}}{dt} = -\mathcal{F}\left[(1+i\beta)|A|^2 A\right]$$

The exact solution for the equation 3.2 is obtained as:

$$\hat{A}(k,t) = \hat{A}^0(k)e^{q(k)t} = \varphi_{\Delta t}^{[L]}$$

However, the integration for the nonlinear term can be done in the spatial domain instead on the Fourier space:

$$\frac{\partial A}{dt} = -(1+i\beta)|A|^2 A$$

By calling $M(x,t) = |A(x,t)|^2 = A(x,t)A^*(x,t)$ we can say:

$$\frac{\partial M(x,t)}{\partial t} = \frac{\partial A(x,t)}{\partial t}A^*(x,t) + A(x,t)\frac{\partial A^*(x,t)}{\partial t} =$$

$$= -(1+i\beta)|A|^2 AA^* - A(1-i\beta)|A|^2 A^* = -|A|^2|A|^2 - |A|^2|A|^2 = -M^2$$

By integrating $\frac{\partial M}{\partial t} = -M^2$ we obtain:

$$M(x,t) = \frac{M_0(x)}{1+2M_0(x)t}$$

and hence we can write the equation 3.2 as:

$$\frac{\partial A}{dt} = -(1+i\beta)M(x,t)A$$

and integrate to obtain:

$$A(x,t) = A_0(x)e^{-(1+i\beta)\int_0^t M(s)ds} = A_0(x)e^{-\frac{1+i\beta}{2}\log(1+2M_0(x)t)} = \varphi_{\Delta t}^{[N]}$$

Now we can use the simplest splitting method to solve our problem, the first-order scheme known as Lie-Trotter scheme (or its adjoint, see code `subroutine CGLE_SSLT`):

$$\psi_{\Delta t} = \varphi_{\Delta t}^{[L]} \circ \varphi_{\Delta t}^{[N]} \qquad \psi_{\Delta t}^{*} = \varphi_{\Delta t}^{[N]} \circ \varphi_{\Delta t}^{[L]}$$

or the second order scheme known as Strang splitting:

$$\mathcal{S}_{\Delta t} = \psi_{\frac{\Delta t}{2}} \circ \psi_{\frac{\Delta t}{2}}^{*} = \varphi_{\frac{\Delta t}{2}}^{[L]} \circ \varphi_{\Delta t}^{[N]} \circ \varphi_{\frac{\Delta t}{2}}^{[L]}$$

which admits the version (code `subroutine CGLE_Strang`):

$$\mathcal{S}_{\Delta t} = \psi_{\frac{\Delta t}{2}}^{*} \circ \psi_{\frac{\Delta t}{2}} = \varphi_{\frac{\Delta t}{2}}^{[N]} \circ \varphi_{\Delta t}^{[L]} \circ \varphi_{\frac{\Delta t}{2}}^{[N]}$$

Chapter 4

Methods comparison

To compare all the methods explained in the previous chapter, the test case number 5 (intermittency regime with planar waves and localised structures) was chosen. The way to do it is by running a simulation with one specific method using a extremely small time step and use its results as "exact" solution. Then, compare the convergence of the rest of methods to these results and also the convergence related to the CPU time or the number of DFTs performed in the computation. For this initial simulation the method ETD2 (second order) was chosen in the beginning since it is the method used for the test cases [7]. However, the results obtained with a fourth order Runge-Kutta scheme did not seem to converge in comparison with ETD2, staying almost constant for a big range of time steps (h).

Therefore, the RK4 method was finally chosen to be used as a basis for comparison. The following table summarises the parameters used for all the following simulations and the values `h` and `N` for the base simulation:

Test Case 5	
(α, β)	$(0.5, -1.5)$
Spatial domain	$x \in [-100, 100]$
Grid points	`Nl = 512`
Final time and timestep	`tf = 100, h = 0.0001, N = 1000000`
Initial condition	$\text{sech}\left((x+10)^2\right) + 0.8\text{sech}\left((x-30)^2\right) + \text{noise}$
Solver used	RK4

It is interesting to remark the big difference between the simulation made with

a really small time step and the results considered in the chapter 2. In the more accurate simulation the evolution of the right branch of localised structures changes and actually disappears for advanced times (see figure 4.1).

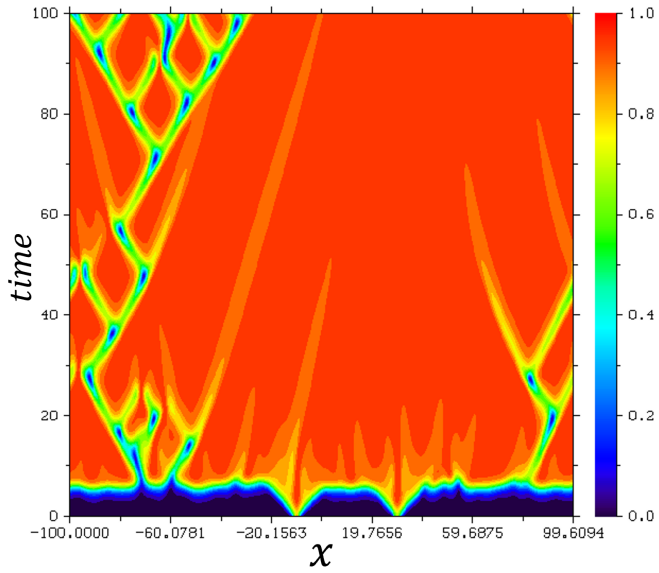


Figure 4.1: Results for the test case 5 obtained with a small time step in order to compare simulations with the different methods.

To compute the error for the different methods the last instant of time is used ($t = 100$). The Euclidean norm over the amplitude vector ($A(x,t)$) in the physical space is used as error measure. By calling $z = A(x, 10)$ for the base simulation and $\hat{z} = A(x, 10)$ for the rest, the norm is computed as:

$$E = z - \hat{z}$$

$$\|E\| = \sqrt{\sum_{i=1}^{512} E_i \overline{E}_i}.$$

The figure 4.2 shows this error depending on the number of points used for the time discretization and using logarithmic scales in both axes. The time step and the number of instants of time are related by $\frac{t_f}{h} = \frac{100}{h} = N$. In this figure, the slope of each line shows the order of the method: first order for Splitting Lie Trotter, second order for RK2, Strang Splitting and ETD2 and fourth order for RK4. Other first order simulations were performed using methods like IFM Euler or ETD1, however, the error for these methods were quite big compared to the base simulation and

hence, the results were not considered as representative. Although the error seemed to decrease for smaller h, the computation times needed were increasing (up to hours per test) what made difficult including more points in the plot.

Regarding the second and fourth order methods: RK4 is the one used for the comparison, hence, its convergence behaves really good in the plot. However, its slope seems to agree with a fourth order method. It has to be said that this method (and RK2) failed ot give results for h bigger than 0.01 being maybe related to the stiff behaviour associated to the linear part of the equation and the fact that is being treated without integrating factor, hence needing from really small time steps to solve the high frequencies. For the rest of methods they gave a result for $h > 0.01$ but with a big error.

The three second order methods seem to present a similar slope in their errors, being the best the Strang method and the RK2. The SSLT method also presents good error results (specially compared with the rest of other first order methods not included) but its convergence to the base simulation is the correspondent to a first order method. Here the advantages of the analytical integration of the linear (or both linear and nonlinear) terms are can be seen; bigger values of h converge to the solution although with a big value of error.

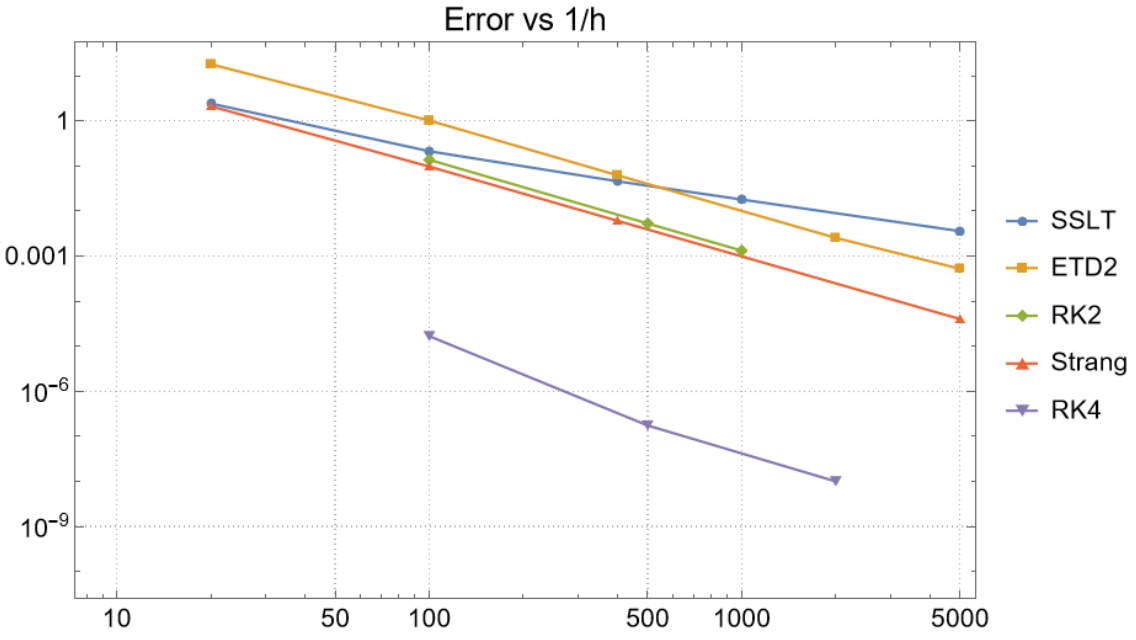


Figure 4.2: Error computing the test case ($t_f = 100$) for different methods depending on the inverse of the time step.

In order to compare the efficiency of each method two measures have been considered: the number of DFT's and the CPU time needed to perform a simulation.

The number of DFTs involved in a simulation is important since it needs from a big amount of time compared to the rest of operations that are performed in each time step. In this occasion is even more important since this solver is using the DFT ($\mathcal{O}(N_l^2)$ operations with N_l the number of spatial points) instead of the Fast Fourier Transform (FFT, $\mathcal{O}(N_l \log N_l)$ real operations).

Methods like RK4, that needs from 4 transforms in each step respectively, are penalised when this metric is considered (see figures 4.3 and 4.4). On the contrary, second order methods needs from less DFTs and less time in general and close the gap to fourth order methods. In general, all the first order methods considered were similar in this aspect since all of them need from the same amount of transforms (one per step) and need a quite similar time to be computed. However, among the second order methods, the Strang Splitting presents the best results, not only because its error is smaller but because, being second order, only needs from one transformation per step while the rest two transforms per step.

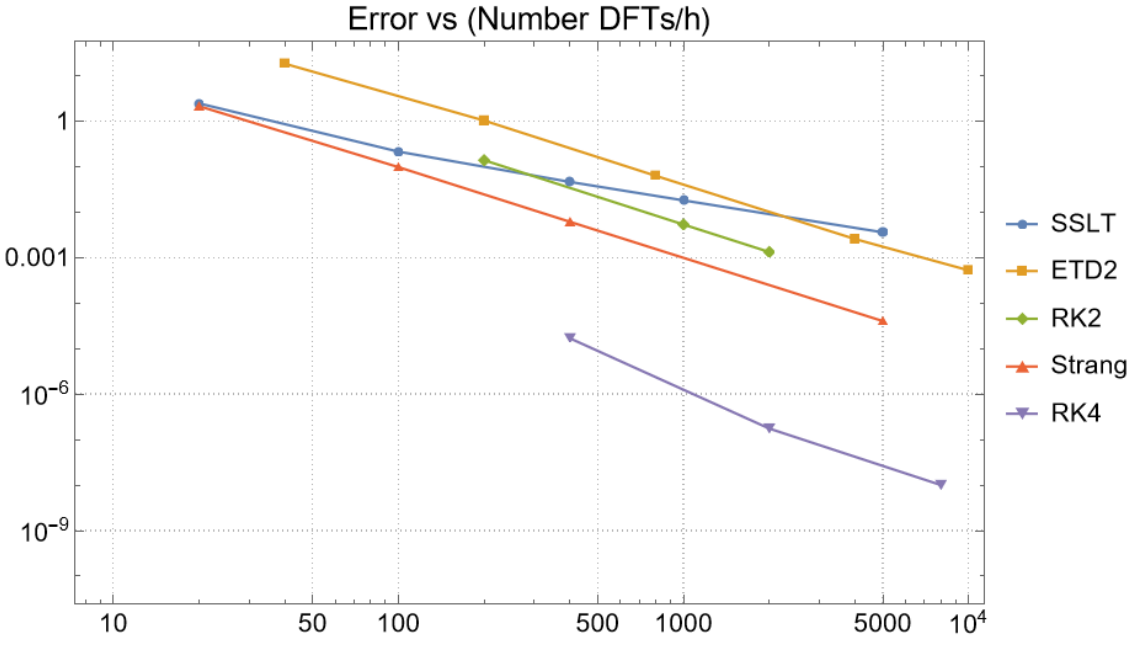


Figure 4.3: Error computing the test case ($t_f = 100$) for different methods related to the number of Fourier transforms needed and the inverse of the time step.

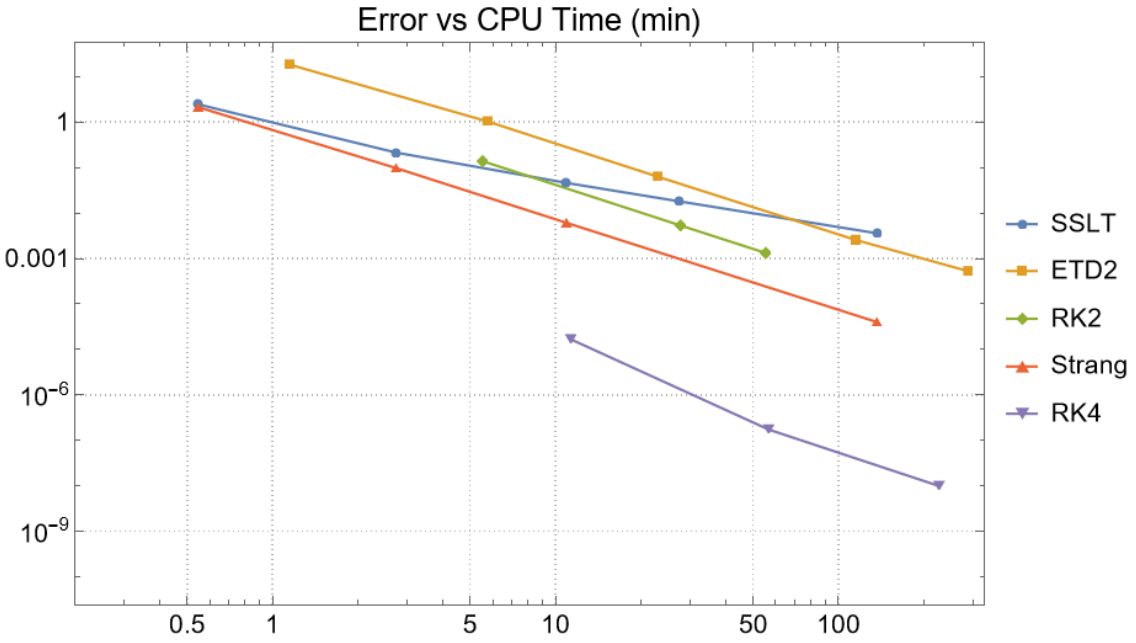


Figure 4.4: Error computing the test case ($t_f = 100$) for different methods related to the CPU time needed to perform the calculations.

Chapter 5

Conclusion

When the simulation to be performed has a high final time it can be essential to reduce the computation time and here Splitting Methods can be the best alternative among the options considered. The two methods tested ensure first and second order with similar computation times between them and requiring both from only one Direct Fourier Transform per time step. In this case, and being both simple to program (even simpler than classical schemes), it seems that Strang Splitting method would be a good option for the CGLE.

If there is enough computing processing power, or the final time is not so high (only transients), higher order methods could be considered. However, it is better to use methods that take advantage of the analytical integration that can be performed over the linear term (at least) like Exponential Time Differencing methods so the time step can be reduced as much as possible without increasing the error too much. It must be said that the approaches of Runge-Kutta 2 and 4 failed to give the proper solution for values of h bigger than 0.01 while the rest of the methods worked. This could be related to the fact that in the basic approach of solving the equation without exponential integrating factor, the equation presents a stiff behaviour, requiring a really small time step to solve the linear part of the equation.

Finally, except the SSLT method, the rest of first order methods compared present a big error and they are not efficient enough to be considered for high simulation times since they would need a lot of hours to compute the result. An alternative to make a better comparison could be choosing another simulation with smaller t_f, needing less computation times then, and use smaller time steps so the convergence of the results can be checked for all the other first order methods.

Appendix A

Notes about the equation

A.1 Notes about Hopf bifurcations for ODEs

Let's cover here some ideas behind the concept of Hopf bifurcation. Given a general autonomous dynamical system written as a system of ordinary differential equations (ODEs):

$$\frac{du_i}{dt} = f_i(\mathbf{u})$$

the dependent variables are usually ordered in the state array $\mathbf{u}$ and the components of the vector-function f_i are in general nonlinear functions. Notice that the state array can be seen as a vector in the phase space, in this phase space every state of the system (every combination of values for u_i) is mapped into a unique spatial location. To do that, one orthogonal axis in the phase space is used for each degree of freedom of the system.

The solutions $\mathbf{u} = \mathbf{u_s}$ of $\mathbf{f}(\mathbf{u}) = \mathbf{0}$ represent the fixed points (or critical points) of the system. For these solutions of the system of ODEs (called stationary states) and assuming that the solution is unique there ($\mathbf{f}$ Lipschitz continuous function of $\mathbf{u}$), the system stays indefinitely in the same state.

Trying to give a intuitive description of the concept of stability instead of a formal definition, we can think of fixed points in terms of how the system behaves in a neighbourhood of these points. If the initial state of a system is close to a fixed point and the trajectory of the solution stays close to the fixed point, then this

fixed point is stable and it is behaving as an attractor. When at least one trajectory starting close to the fixed point moves away from the it, then it is unstable and is behaving as a repeller.

Mathematically, the derivatives of $\mathbf{f}$ with respect to $\mathbf{u}$ are used to characterised the stability of the fixed points. While $f'(u) < 0$ ($f'(u) > 0$) is a sufficient condition to ensure the stability (unstability) of the fixed point in a one dimensional dynamical system, we can find both stable and unstable systems with $f'(u) = 0$, hence, these are not necessary conditions. For higher dimensional systems a linear approximation of the vector function $\mathbf{f}$ around the fixed point can be used. Depending on the eigenvalues of the Jacobian $\mathbf{f'(u)}$ the behaviour of the system in the neighbourhood of a fixed point can be characterised.

If a perturbation $\delta\mathbf{u}$ is introduced to our system in the fixed point $\mathbf{u_s}$:

$$\dot{\mathbf{u}}_\mathbf{s} + \dot{\delta\mathbf{u}} = \mathbf{f}(\mathbf{u_s} + \delta\mathbf{u})$$

we can use a Taylor expansion to linearise the equation:

$$\dot{\mathbf{u}}_\mathbf{s} + \delta\dot{\mathbf{u}} = \mathbf{f}(\mathbf{u_s}) + \mathbf{f'}(\mathbf{u_s})\delta\mathbf{u} + \mathbf{f''}(\mathbf{u_s})\frac{\delta\mathbf{u^2}}{\mathbf{2!}} + \dots$$

since $\dot{\mathbf{u}}_\mathbf{s} = \mathbf{f}(\mathbf{u_s}) = \mathbf{0}$ we can say:

$$\delta\dot{\mathbf{u}} = \mathbf{J}(u_s)\delta u + \text{higher order terms}$$

where $\mathbf{J}(u_s)$ denotes the Jacobian matrix of $\mathbf{f}(\mathbf{u})$ in the fixed point (constant matrix). Hence, now we are working with a linear differential equation whose solution can be expressed as a superposition of complex exponentials with the eigenvalues of the Jacobian $e^{\lambda t}$ in the exponent. Then, the eigenvalues of this Jacobian determine how the system behaves locally, near the fixed points. Right now we are interested in two conditions:when an eigenvalue have $\Re(\lambda) > 0$ the fixed point is unstable and if two eigenvalues are complex conjugate ($\Im(\lambda) \neq 0$) the system presents an oscillatory behaviour.

It has been said that the phase diagram can be used to represent the state of the system. This diagram takes on special importance when the equation depends on a parameter, specially with scalar autonomous dynamical systems:

$$\frac{du}{dt} = f(u;\mu)$$

Notice how in this case, when the parameter changes, the behaviour of the system also changes. Normally we can find quantitative changes but for some values of the parameter qualitative changes can also appear. Some fixed points can appear or disappear, change their behaviour from attractors to repellers or viceversa, etc. Two autonomous equations are qualitatively equivalent if they have the same number of critical points, of the same type and ordered in the same way along the phase diagram [6]. When a qualitative change occurs, for example for a value $\mu = \mu_0$, it is said that the equation $\frac{du}{dt} = f(u;\mu)$ has a bifurcation. These bifurcations are usually studied using a p-dimensional parametric spaces. Here, the boundary where the system changes its behaviour is called bifurcation manifold and it has a $p-1$ dimension in the p-dimensional space.

Several types of bifurcation exist depending on the change in the behaviour of the equation. CGL equations are intimately related to Hopf bifurcations. With this type (usually grouped into the local bifurcations) a periodic solution appears or disappears together with the stability change. In Hopf bifurcations a given fixed point, which is initially stable, loses stability due to the cross of a pair of complex conjugate eigenvalues of the imaginary axis. Hence, they are always related to eigenvalues with zero real part. Said in other words, they typically occur when the pair of eigenvalues of the linearised system becomes purely imaginary. Notice then that Hopf bifurcations occur in systems with two or more dimensions or, as we'll see, with complex systems.

Autonomous dynamical systems can present limit cycles within their trajectories. A limit cycle is an isolated closed trajectory that corresponds to a periodic (but non-constant) solution. It has the property that at least one trajectory generates a spiral into it for $t \to \infty$ or comes from it with a spiral shape ($t \to -\infty$). When a Hopf bifurcation appears in a system (of supercritical kind), the stable fixed point for a given value of the parameter becomes a small-amplitude limit cycle when the parameter crosses the boundaries.

For the CGLE we are interested in the supercritical Hopf bifurcations, those where the first Lyapunov coefficient is negative. To delve into this let's present first the normal form of a Hopf bifurcation. Notice that for a given 2 dimensional autonomous system in the variables u, v, the normal form can be obtained by using a complex variable $z = u + iv$, hence converting the system into a complex one dimensional equation:

$$\frac{du}{dt} = f(u, v; \mu)$$
$$\frac{dv}{dt} = g(u, v; \mu)$$

After a transformation of the system into a complex equation in the variable z the normal form is obtained:

$$\frac{dz}{dt} = z(\mu + i) + b|z|^2 z$$

where $b, z \in \mathbb{C}$ and $\mu \in \mathbb{R}$ is the parameter which generates the bifurcation. For $b = \alpha + i\beta$, α is known as first Lyapunov coefficient and it decides the type of Hopf bifurcation that is generated. With $\alpha < 0$, supercritical Hopf bifurcation, a stable fixed point ($\mu < 0$) becomes unstable and a stable limit cycle appears when μ becomes positive as the figure A.1 indicates [13]. We are not covering here the case $\alpha > 0$, subcritical Hopf bifurcation.

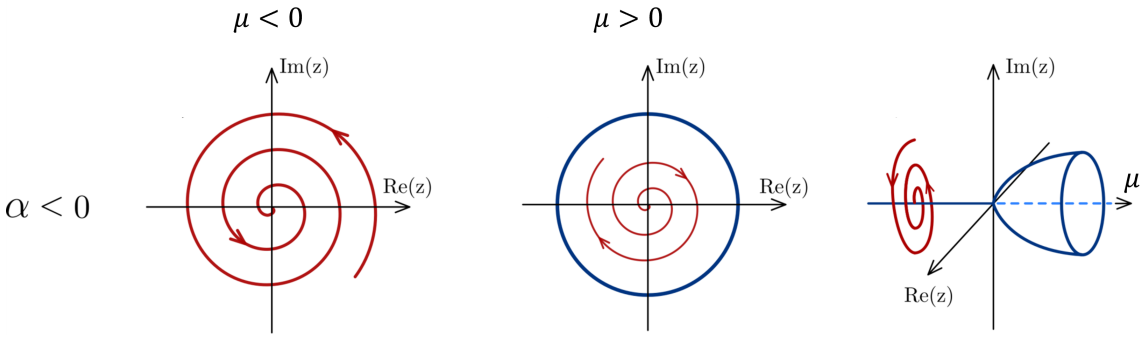


Figure A.1: In this diagram, obtained from Wikipedia, the phase diagram of both variables (or parts of the complex variable z) are represented before and after crossing the bifurcation manifold. Dark blue lines represent stable points, dashed blue lines unstable points and red lines the possible trajectories. The third diagram shows how the stable fixed point becomes an unstable fixed point while an stable limit cycle appears.

By using:

$$z = re^{i\theta}$$

let's write the normal form of the supercritical Hopf bifurcation using polar coordinates:

$$\frac{dz}{dt} = \frac{dr}{dt}e^{i\theta} + re^{i\theta}i\frac{d\theta}{dt} = re^{i\theta}(\mu + i) + br^2re^{i\theta}$$
$$\frac{dr}{dt} + ri\frac{d\theta}{dt} = r(\mu + i) + (\alpha + i\beta)r^2r$$

and separating into real and imaginary part for $\omega = 1 + \beta r^2$:

$$\frac{dr}{dt} = r(\mu + \alpha r^2)$$
$$\frac{d\theta}{dt} = 1 + \beta r^2 = \omega$$

When $\alpha < 0$, the cycle limit is given by $z(t) = re^{i\omega t}$ with $r = \sqrt{-\mu/\alpha}$ since:

$$\frac{dr}{dt} = r(\mu + \alpha r^2) = 0$$
$$\frac{d\theta}{dt} = \omega = \text{constant}$$

A.2 Notes about Reaction-Diffusion systems (RDS)

A general expression for these systems is [14]:

$$\gamma \frac{\partial u}{\partial t} = -\nabla \cdot \mathbf{j} + f(u)$$

where f represents the production rate (generally nonlinear and including parameters), γ is a capacitance factor and $\mathbf{j}$ is the flux. Notice that the model includes transport and linear/nonlinear (or both) interactions, but not advection. They are usually seen as conservation laws in several branches of Physics like chemistry, thermodynamics, fluid mechanics, etc. Then, the field variable (u) under study take the meaning of concentration of chemical species, temperature, pressure, etc. It is common to model fluxes with gradients of these field variables (u), for example:

$$\mathbf{j} = -D\nabla u$$

where D is interpreted as a diffusivity.

Some examples of pattern-forming systems are modelled through the cubic reaction-diffusion equation (gene propagation):

$$\frac{\partial u}{\partial t} = \nabla^2 u + u(1 - u^2)$$

or the Burgers equation (fluid mechanics, nonlinear acoustics):

$$\frac{\partial u}{\partial t} = \nabla^2 u + |\nabla u|^2$$

the already mentioned RGLE (where $\alpha, \beta = 0$ in CGLE):

$$\frac{\partial u}{\partial t} = \nabla^2 u + u(1 - |u|^2)$$

or the Nonlinear Schrödinger Equation (NLS, where $\alpha, \beta \to \infty$ in CGLE):

$$i\frac{\partial u}{\partial t} = \nabla^2 u + u(1 - |u|^2)$$

Notice that the first one is second order in the spatial derivatives and third order in u, similarly to the CGLE, the RGLE or the NLS. These equations, like the CGLE, are characterised by requiring spatial isotropy and a symmetry to transformations of u. For the cubic reaction-diffusion equation for example, the symmetry is found in the inversion of u [14]. The CGLE is invariant under a global change of gauge (among others), which means, invariant under a multiplication by $e^{i\Phi}$. It is common to write RDS equations parameterless, rescaled and with the signs already chosen to ensure stability against small perturbations.

An example of these equations that can be found in hydrodynamics would be something like [1]:

$$\tau(\partial_{\tilde{t}}\tilde{A} - \vec{v}_g \cdot \tilde{\nabla}\tilde{A}) = \epsilon(1 + ia)\tilde{A} + \xi^2(1 + ib)\tilde{\Delta}\tilde{A} - g(1 + ic)|\tilde{A}|^2\tilde{A}$$

After some assumptions and transformations like $\tilde{A} = (\epsilon/g)^{1/2}A\exp(-i(\epsilon a/\tau)\tilde{t})$, $\tilde{t} = (\tau/\epsilon)t$ or $\vec{\tilde{x}} - \vec{v}_g\tilde{t} = (\xi/\epsilon^{1/2})\vec{x}$, the following expression is obtained:

$$\frac{\partial A}{\partial t} = (1 + i\alpha)\nabla^2 A + A - (1 + i\beta)|A|^2 A$$

In a general situation the real parts of the coefficients of the Laplacian, linear and nonlinear terms can be reduced to unity. In addition, a transformation like $u \to ue^{i\phi t}$ allows to discard the imaginary part of the coefficient of the linear term.

A.3 Notes about the CGLE

Apart from the gauge invariance already mentioned (which is an invariance on space and time) there are other symmetries in the CGLE [2]. We can find another translation invariance due to the fact that it is an autonomous equation for both space and time (no explicit dependence of these variables). In addition, the system is isotropic (symmetry under rotations) and allows a symmetry for spatial reflections ($x \to -x$). Finally, the parameter space also allows an inversion with no changes in the equation ($\alpha, \beta, A \to -\alpha, -\beta, A^*$).

The variational treatment for this equation is possible when $\alpha = \beta$, contrary to the RGLE where it is always possible. For the CGLE, a rotating frame transformation is used ($A \to Ae^{i\alpha t}$) so:

$$\frac{\partial A}{\partial t} \to \frac{\partial A}{\partial t} e^{-i\alpha t} - e^{-i\alpha t} A i\alpha$$

and

$$(1 + i\alpha)\nabla^2 A + A - (1 + i\beta)|A|^2 A \to (1 + i\alpha)(\nabla^2 A - |A|^2 A)e^{-i\alpha t} + Ae^{-i\alpha t}$$

Hence, the equation can be written as:

$$\frac{\partial A}{\partial t} - Ai\alpha = (1 + i\alpha)(\nabla^2 A - |A|^2 A) + A$$

or

$$\frac{\partial A}{\partial t} = (1 + i\alpha)(\nabla^2 A - |A|^2 A + A)$$

This equation can be obtained through variation of a functional that acts as a global Lyapunov functional or complex generalized free energy [1]. Just as an example, for the real case the Lyapunov functional ($dV/dt < 0$) can be expressed as:

$$V = \int \left(\left| \frac{\partial A}{\partial x} \right|^2 - |A|^2 + \frac{1}{2}|A|^4 \right) dx$$

so our RGLE can be written as:

$$\frac{\partial A}{\partial t} = -\frac{\delta V}{\delta A^*}$$

Appendix B

Fortran Program

B.1 MAIN Program

The following code is the starting point of the project. Here only the solver to be used is decided and called. All the equation parameters, solver parameters and some constants are defined inside the module `Complex_Ginzburg_Landau`.

```
program MAIN

    use Complex_Ginzburg_Landau

    implicit none

!-------------------------------------------------------
! Select here the solver for the CGLE:
! dA/dt = (1+ia) d2A/dx2 + A - (1+ib) |A|^2 A
! Eq. parameters and Initial Condition inside the module
! Periodic Boundary Conditions
!-------------------------------------------------------

    !call CGLE
    !call  CGLE_IFM_Gral
    !call CGLE_IFM_Euler
    !call  CGLE_ETD1
    call  CGLE_ETD2
    !call CGLE_ETD4KR
    !call CGLE_SSLT
    !call CGLE_Strang


    !Base for comparison
    !call  CGLE_ETD2_MOD


end program
```

Listing B.1: `MAIN.f90`

B.2 Spatial discretization

The implementation of the Direct Fourier Transform (DFT) and its inverse (IFT) are stored in the module `Discrete_Fourier`. Notice that the normalization $\frac{1}{N}$ is used for the direct transform ([11] and [8]). Also, the indexes of the vectors involved (`U(x)` and `C(k)`) have physical sense, then $C(-256)$ means the Fourier mode with wave number $k = -256$.

```
function DFT(N, U) result(C)
    integer, intent(in) :: N
    complex, intent(in)  :: U(0:N-1)
    complex :: C(-N/2:N/2-1)

    integer   :: k, j
    complex :: S

    do k = -N/2, N/2-1
        S = 0
        do j=0, N-1
            S = S + U(j) * exp( -2*PI*II * j * k/real(N) )
        end do
        C(k)  = S / N
    end do

end function
```

Listing B.2: `Discrete_Fourier.f90`

```
function IFT(N, C) result(U)
    integer, intent(in) :: N
    complex, intent(in) :: C(-N/2:N/2-1)
    complex  :: U(0:N-1)

    integer   :: k, j
    complex :: S

    !write(*,*)  "C = ", C

    do j = 0, N-1
        S = 0
        do k=-N/2, N/2-1
            S = S + C(k) * exp( 2*PI*II * j * k/real(N) )
        end do
        U(j) = S
    end do
    !write(*,*)  "U = ", U

end function
```

Listing B.3: `Discrete_Fourier.f90`

B.3 Module `Complex_Ginzburg_Landau`

The following code shows the header of the main module. Here all the parameters are declared so the solvers located below use this declaration to find the solution. In the following page, the general structure for a solver is shown. For some solvers a general structure is encapsulated in the module `Cauchy_Problem` so the most common schemes can be used with no need of coding them each time. For more specific schemes, they are written in a loop inside the specific subroutine.

```
module Complex_Ginzburg_Landau

use Cauchy_Problem
use Temporal_Schemes
use plots
use Discrete_Fourier

implicit none

private
public ::   CGLE,           &
            CGLE_IFM_Gral,  &
            CGLE_IFM_Euler, &
            CGLE_ETD1,      &
            CGLE_ETD2,      &
            CGLE_ETD4KR,    &
            CGLE_SSLT,      &
            CGLE_Strang,  &

            CGLE_ETD2_MOD


real, parameter     :: PI = 4 * atan(1d0)
complex, parameter  :: II = (0, 1)
!Equation parameters
real, parameter :: alpha = .5, beta = -1.5
!Options for InitCond: "Noise", "NoisyWave",
!"NoisyConstant", "Intermittency1", "Intermittency2"
character(len=14) :: InitCond = "Intermittency1"
!Solver parameters
real :: t0 = 0, tf = 100, h
real :: x0 = -100, xf = 100, L
integer, parameter :: N = 2000, Nl = 512 !Nl must be even

contains
```

Listing B.4: Header of the module Complex_Ginzburg_Landau.f90

Essentially, each solver of the following are performing the same tasks: declaring the length of the interval, building the time domain and the spatial discretization, build the initial condition and its Fourier transform, solving the equation and transforming this solution to the physical space. For the rest of solvers only the code that is specific for the solver is shown.

```
subroutine CGLE

    integer :: i, k
    !real :: Time(0:N), x(0:Nl-1)
    !complex :: Ahat(0:N,0:Nl-1), A(0:N,0:Nl-1)
    real, allocatable :: Time(:), x(:)
    complex, allocatable :: Ahat(:,:), A(:,:)

    allocate( Time(0:N), x(0:Nl-1), Ahat(0:N,0:Nl-1), A(0:N,0:Nl-1) )

    L = xf - x0
    Time = [ (t0 + (tf - t0 ) * i / (1d0 * N), i=0, N ) ]
    x = [ (x0 + (xf - x0) * i / (1d0 * Nl), i=0, Nl-1 ) ]


    call Choose_A0( InitCond, A(0,:), Nl, x, L )
    Ahat(0,:) =  DFT( Nl, A(0,:) )


    call Cauchy_ProblemS(   Time_Domain = Time,              &
                            Differential_operator = F,       &
                            Solution = Ahat,                 &
                            Scheme = Runge_Kutta4  )

    do i = 0, N
        A(i,:) = IFT( Nl, Ahat(i,:) )
    enddo

    call CGLE_PLOT( x, Time, A, N, Nl )

    contains

    function F( Ahat, t )    !dU/dt = F(U,t)
        complex :: Ahat(:)
        real :: t
        complex :: F(size(Ahat))

        integer :: Nl, k
        complex, allocatable :: A(:)

        Nl = size(Ahat)
        allocate( A(0:Nl-1) )

        A = IFT( Nl, Ahat )
    F = [(1 - (k*(2*PI)/L)**2 * (1,alpha), k = -Nl/2, Nl/2-1)]*Ahat &
            - DFT( Nl, (1,beta) * abs(A)**2 * A )

    end function

    end subroutine
```

Listing B.5: Solver CGLE in Complex_Ginzburg_Landau.f90

```
    !Solver CGLE_IFM_Gral
    call Cauchy_ProblemS(   Time_Domain = Time,              &
                            Differential_operator = F,       &
                            Solution = Phi,                 &
                            Scheme = Runge_Kutta4  )


    do i = 0, N
        Ahat(i,:) = Phi(i,:)/exp(-q * Time(i))
        A(i,:) = IFT( Nl, Ahat(i,:) )
    enddo

    call CGLE_PLOT( x, Time, A, N, Nl )



    contains

    function F( Phi, t )
        complex :: Phi(:)
        real :: t
        complex :: F(size(Phi))

        integer :: Nl, k
        complex, allocatable :: q(:), A(:)

        Nl = size(Phi)
        allocate( q(-Nl/2:Nl/2-1), A(0:Nl-1) )

        q = [(1 - (k*(2*PI)/L)**2 * (1,alpha), k = -Nl/2, Nl/2-1)]
        A = IFT( Nl, Phi/exp(-q * t) )
        F = - DFT( Nl, (1,beta) * abs(A)**2 * A ) * exp(-q * t)

    end function

end subroutine
```

Listing B.6: Solver CGLE_IFM_Gral in Complex_Ginzburg_Landau.f90

```
    do i = 0, N-1     !Solver CGLE_IFM_Euler
        Ahat(i+1,:) = exp(q * h)*(Ahat(i,:) +    &
                      h * F(Ahat(i,:), Time(i)))

        if (mod(i,1000)==0) then;  write(*,*) "Steps = ", i; endif
    enddo

    do i = 0, N
        A(i,:) = IFT( Nl, Ahat(i,:) )
    enddo


    call CGLE_PLOT( x, Time, A, N, Nl )


    contains


    function F( Ahat, t )  result(Nhat)
        complex :: Ahat(:)
        real :: t !not needed, autonomous eq.
        complex :: Nhat(size(Ahat))

        integer :: Nl
        complex, allocatable :: A(:)

        Nl = size(Ahat)
        allocate( A(0:Nl-1) )

        A = IFT( Nl, Ahat )
        Nhat = -DFT( Nl, (1,beta) * abs(A)**2 * A )

    end function

    end subroutine
```

Listing B.7: Solver CGLE_IFM_Euler in Complex_Ginzburg_Landau.f90

```
do i = 0, N-1    !Solver CGLE_ETD1
    Ahat(i+1,:) = exp(q * h) * Ahat(i,:) +  &
        F( Ahat(i,:), Time(i) )*( exp(q * h) - 1 )/q

    if (mod(i,1000)==0) then;  write(*,*) "Steps = ", i; endif
enddo

do i = 0, N
    A(i,:) = IFT( Nl, Ahat(i,:) )
enddo


call CGLE_PLOT( x, Time, A, N, Nl )


contains


function F( Ahat, t )  result(Nhat)
    complex :: Ahat(:)
    real :: t !not needed, autonomous eq.
    complex :: Nhat(size(Ahat))

    integer :: Nl
    complex, allocatable :: A(:)

    Nl = size(Ahat)
    allocate( A(0:Nl-1) )

    A = IFT( Nl, Ahat )
    Nhat = -DFT( Nl, (1,beta) * abs(A)**2 * A )

end function

end subroutine
```

Listing B.8: Solver CGLE_ETD1 in Complex_Ginzburg_Landau.f90

```
    !Solver ETD2: first step with ETD1
    Ahat(1,:) = exp(q * h) * Ahat(0,:) +    &
        F( Ahat(0,:), Time(0) )*( exp(q * h) - 1 )/q
    do i = 1, N-1
        Ahat(i+1,:) =   Ahat(i,:)*exp(q * h) +              &
                    F( Ahat(i,:), Time(i) ) *               &
                    ( (1+h*q)*exp(q*h) - 1 - 2*h*q )/(h*q*q)&
                    + F( Ahat(i-1,:), Time(i-1) ) *         &
                    ( -exp(q*h) + 1 + h*q )/(h*q*q)

        if (mod(i,1000)==0) then;  write(*,*) "Steps = ", i; endif

    enddo

    do i = 0, N
        A(i,:) = IFT( Nl, Ahat(i,:) )
    enddo

    call CGLE_PLOT( x, Time, A, N, Nl )

    contains


    function F( Ahat, t )  result(Nhat)
        complex :: Ahat(:)
        real :: t !not needed, autonomous eq.
        complex :: Nhat(size(Ahat))

        integer :: Nl
        complex, allocatable :: A(:)

        Nl = size(Ahat)
        allocate( A(0:Nl-1) )

        A = IFT( Nl, Ahat )
        Nhat = -DFT( Nl, (1,beta) * abs(A)**2 * A )

    end function

    end subroutine
```

Listing B.9: Solver CGLE_ETD2 in Complex_Ginzburg_Landau.f90

```
!Solver ETD4RK
do i = 0, N-1

an = Ahat(i,:)*exp(qh/2) + (exp(qh/2)-1)*F(Ahat(i,:), Time(i))/q
bn = Ahat(i,:)*exp(qh/2) + (exp(qh/2)-1)*F(an, Time(i)+h/2)/q
cn = an*exp(qh/2) + (exp(qh/2)-1)*(2*F(bn, Time(i)+h/2) -  &
                                   F(Ahat(i,:), Time(i)))/q

Ahat(i+1,:) =   Ahat(i,:)*exp(qh) +   1/(h**2*q*q*q) *       &
( F( Ahat(i,:), Time(i) ) * (-4-qh+exp(qh)*(4-3*qh+qh*qh))   &
+ 2*(F(an,Time(i)+h/2)+F(bn,Time(i)+h/2))*(2+qh+exp(qh)*(-2+qh))&
+ F(cn,Time(i)+h)*(-4-3*qh-qh*qh+exp(qh)*(4-qh))    )

    if (mod(i,1000)==0) then;  write(*,*) "Steps = ", i; endif

enddo

do i = 0, N
    A(i,:) = IFT( Nl, Ahat(i,:) )
enddo

call CGLE_PLOT( x, Time, A, N, Nl )

contains


function F( Ahat, t )  result(Nhat)
    complex :: Ahat(:)
    real :: t !not needed, autonomous eq.
    complex :: Nhat(size(Ahat))

    integer :: Nl
    complex, allocatable :: A(:)

    Nl = size(Ahat)
    allocate( A(0:Nl-1) )

    A = IFT( Nl, Ahat )
    Nhat = -DFT( Nl, (1,beta) * abs(A)**2 * A )

end function

end subroutine
```

Listing B.10: Solver CGLE_ETDRK4 in Complex_Ginzburg_Landau.f90

```
    !Solver Splitting LT
    do i = 0, N-1
        A(i+1,:) = IFT(Nl, Lterm( DFT(Nl, Nterm( A(i,:), h )), h ))
        if (mod(i,1000)==0) then;  write(*,*) "Steps = ", i; endif
    enddo

    call CGLE_PLOT( x, Time, A, N, Nl )

    contains


    function Nterm( A, t )  result(N)
        complex :: A(:)
        real :: t
        complex :: N(size(A))

        N = A * exp( -(1,beta)/2 * log( 1 + 2*abs(A)**2 * t ) )

    end function

    function Lterm( Ahat, t )  result(L)
        complex :: Ahat(:)
        real :: t
        complex :: L(size(Ahat))

        L = Ahat * exp( q * t )

    end function

end subroutine
```

Listing B.11: Splitting Lie Trotter in Complex_Ginzburg_Landau.f90

```
    !Solver Strang Splitting
    do i = 0, N-1
    A(i+1,:) =                &
    Nterm( IFT(Nl, Lterm( DFT(Nl, Nterm( A(i,:), h/2 )), h )), h/2 )
        if (mod(i,1000)==0) then;  write(*,*) "Steps = ", i; endif
    enddo

    call CGLE_PLOT( x, Time, A, N, Nl )

    contains


    function Nterm( A, t )  result(N)
        complex :: A(:)
        real :: t
        complex :: N(size(A))

        N = A * exp( -(1,beta)/2 * log( 1 + 2*abs(A)**2 * t ) )

    end function

    function Lterm( Ahat, t )  result(L)
        complex :: Ahat(:)
        real :: t
        complex :: L(size(Ahat))

        L = Ahat * exp( q * t )

    end function

end subroutine
```

Listing B.12: Strang Splitting in Complex_Ginzburg_Landau.f90

B.4 Module Cauchy_Problem

This subroutine integrates the Cauchy problem for three different explicit schemes: Forward Euler, Runge-Kutta 2 and Runge-Kutta 4. It makes use of the module Temporal_Schemes where these schemes are coded in its more general form. Hence, any Cauchy Problem can be integrated automatically just by declaring the vector function and choosing a method.

```
subroutine Cauchy_ProblemS( Time_Domain, Differential_operator,  &
                            Solution, Scheme )

    real, intent(in) :: Time_Domain(:)
    procedure (ODES) :: Differential_operator ! F(U,t)
    complex, intent(out) :: Solution(:,:)
    !Scheme: ExplicitEuler, Runge_Kutta2, Runge_Kutta4
    procedure (Temporal_Scheme) :: Scheme

    real :: start, finish, t1, t2
    integer ::  i, N_steps, ierr


    call cpu_time(start)
    N_steps = size(Time_Domain) - 1
    do i = 1, N_steps
        t1 = Time_Domain(i); t2 = Time_Domain(i+1);

        call Scheme( Differential_operator, t1, t2,          &
                     Solution(i,:), Solution(i+1,:) )

        if (mod(i,1000)==0) then;  write(*,*) "Steps = ", i; endif

    enddo
    call cpu_time(finish)


    write(*,'("Cauchy_Problem, CPU Time=",f6.3," seconds.")') finish
       - start
    write(*,*)

end subroutine
```

Listing B.13: Cauchy_Problem.f90

B.5 Module `Temporal_Schemes`

```
module Temporal_Schemes

use ODE_Interface


implicit none

  private
  public ::     ExplicitEuler,       &
                Runge_Kutta2,        &
                Runge_Kutta4


!-------------------------------------------------------
! U^{n+1} = G( U^n... U^{n-1+p}, F^n... F^{n-1+p}, dt )
! F(U, t) in the system of ODEs  dU/dt = F(U, t)
! t1 and t2: initial and final times
! U1 and U2:  vectors for the initial and final state
!-------------------------------------------------------

    contains



 subroutine ExplicitEuler(F, t1, t2, U1, U2 )
    procedure (ODES) :: F

    real, intent(in) :: t1, t2
    complex, intent(in) ::  U1(:)
    complex, intent(out) ::  U2(:)

    real :: t, dt

    dt = t2 - t1
    t = t1
    U2 = U1 + dt * F(U1, t)

 end subroutine ExplicitEuler
```

Listing B.14: `Temporal_Schemes.f90`

```
subroutine Runge_Kutta2(F, t1, t2, U1, U2 )
    procedure (ODES) :: F

    real, intent(in) :: t1, t2
    complex, intent(in) ::  U1(:)
    complex, intent(out) ::  U2(:)

    real :: t, dt
    real,save :: t_old
    integer :: N
    complex, save, allocatable :: k1(:), k2(:)

    if (.not.allocated(k1)) then
        N = size(U1)
        allocate( k1(N), k2(N) )
    elseif (t1 < t_old) then
        N = size(U1)
        deallocate( k1, k2 )
        allocate( k1(N), k2(N) )
    endif

    dt = t2-t1;   t = t1

    k1 = F( U1, t )
    k2 = F( U1 + dt * k1, t + dt )

    U2 = U1 + dt * ( k1 + k2 )/2
    t_old = t2

end subroutine

subroutine Runge_Kutta4( F, t1, t2, U1, U2 )
    procedure (ODES) :: F
    real, intent(in) :: t1, t2
    complex, intent(in) :: U1(:)
    complex, intent(out) ::  U2(:)

    real :: t, dt
    complex :: k1(size(U1)), k2(size(U1)), k3(size(U1)), k4(size(U1))

    dt = t2-t1;   t = t1

    k1 = F( U1, t)
    k2 = F( U1 + dt * k1/2, t + dt/2 )
    k3 = F( U1 + dt * k2/2, t + dt/2 )
    k4 = F( U1 + dt * k3,   t + dt   )

    U2 = U1 + dt * ( k1 + 2*k2 + 2*k3 + k4 )/6

end subroutine Runge_Kutta4
```

Listing B.15: Temporal_Schemes.f90

B.6 Initial Conditions

The general subroutine `Choose_A0` is used to encapsulate the initialization of the initial condition for each solver. According to the variable `InitCond` declared at the beginning of the module it initialises `A(0,:)` by calling to the different functions needed for the test cases presented in the chapter 2.

```
subroutine Choose_A0( InitCond, A0, Nl, x, L )
    character(len=14), intent(in) :: InitCond
    integer, intent(in) :: Nl
    complex, intent(out) :: A0(0:Nl-1)
    real, intent(in) :: L, x(0:Nl-1)

    if (InitCond == "Noise") then
        A0(:) = A0_Noise( Nl, ampli = 1e-2 )
    else if (InitCond == "NoisyWave") then
        A0(:) = A0_NoisyWave( x, L, Nl, ampli = 1e-2 )
    else if (InitCond == "NoisyConstant") then
        A0(:) = A0_NoisyConstant( C = 1., Nl = Nl, ampli = 1e-2 )
    else if (InitCond == "Intermittency1") then
        A0(:) = A0_Intermittency1( x, Nl, ampli = 1e-2 )
    else if (InitCond == "Intermittency2") then
        A0(:) = A0_Intermittency2( x, L, Nl, ampli = 1e-2 )
    else
        write(*,*)  "Initial Condition not implemented"
        stop
    end if

end subroutine

!-------------------------------------------------------
! Pseudo-random complex noise: (noise + i*noise)
! in Nl points centered in 0 with an amplitude of +-ampli
!-------------------------------------------------------
function A0_Noise( Nl, ampli )  result(A0)
    integer :: Nl
    ! amplitud of the noise around A = 0
    real :: ampli
    complex :: A0(0:Nl-1)

    real :: re_noise(0:Nl-1), im_noise(0:Nl-1)
    integer :: nseed
    integer, allocatable :: seed(:)
    complex :: noise(0:Nl-1)

    call random_seed( size = nseed ); allocate( seed(nseed) )
    seed = 123456789
    call random_seed( put = seed ); deallocate(seed)

    call random_number( re_noise ); call random_number( im_noise )
    noise = ampli * (2*cmplx( re_noise, im_noise ) - (1,1))

    A0 = noise

end function
```

Listing B.16: `Complex_Ginzburg_Landau.f90`

The function A0_noise, which generates the noise for all the initial conditions, is also encapsulated and, in order to reproduce the same test case every time, the seed for the pseudo-random generator is always the same. By changing this seed the noise generated will change. Notice that the same noise is used for all the initial conditions.

```
!--------------------- wave and noise ------------------
! wave sqrt(1-(20*PI/L)^2) * e^(i(20PI/L)*x)  + noise
!-------------------------------------------------------
function A0_NoisyWave( x, L, Nl, ampli )  result(A0)
    integer :: Nl
    real :: x(0:Nl-1), L, ampli
    complex :: A0(0:Nl-1)

    A0 =  sqrt( 1 - (20*PI/L)**2 ) * exp( II*(20*PI/L)*x ) + &
                                        A0_Noise( Nl, ampli )

end function

!-------------------------------------------------------
! Constant + noise
!-------------------------------------------------------
function A0_NoisyConstant( C, Nl, ampli )  result(A0)
    integer :: Nl
    real :: C, ampli
    complex :: A0(0:Nl-1)

    A0 =  C + A0_Noise( Nl, ampli )

end function

!-------------------------------------------------------
! Sech((x+10)^2) + 0.8 Sech((x-30)^2) + noise
!-------------------------------------------------------
function A0_Intermittency1( x, Nl, ampli )  result(A0)
    integer :: Nl
    real :: x(0:Nl-1), ampli
    complex :: A0(0:Nl-1)

    A0 =  1./cosh( (x+10)**2 ) + 0.8/cosh( (x-30)**2 ) + &
                                    A0_Noise( Nl, ampli )

end function

!-------------------------------------------------------
! Sech((x+L/4)^2) + 0.8 Sech((x-L/4)^2) + noise
!-------------------------------------------------------
function A0_Intermittency2( x, L, Nl, ampli )  result(A0)
    integer :: Nl
    real :: x(0:Nl-1), L, ampli
    complex :: A0(0:Nl-1)

    A0 =  1./cosh( (x+L/4)**2 ) + 0.8/cosh( (x-L/4)**2 ) + &
                                    A0_Noise( Nl, ampli )

end function A0_Intermittency2
```

Listing B.17: Complex_Ginzburg_Landau.f90

References

[1] I. S. Aranson and L. Kramer. The world of the complex ginzburg-landau equation. *Rev. Mod. Phys.*, 74:99–143, Feb 2002. doi: 10.1103/RevModPhys.74.99. URL https://link.aps.org/doi/10.1103/RevModPhys.74.99.

[2] I. S. Aranson and L. Kramer. Complex ginzburg-landau equation lectures i and ii, 2002.

[3] S. Blanes, F. Casas, and A. Murua. Splitting and composition methods in the numerical integration of differential equations. *Bol. Soc. Esp. Mat. Apl.*, 45, 01 2009.

[4] S. Cox and P. Matthews. Exponential time differencing for stiff systems. *Journal of Computational Physics*, 176(2):430–455, 2002. ISSN 0021-9991. doi: https://doi.org/10.1006/jcph.2002.6995. URL https://www.sciencedirect.com/science/article/pii/S0021999102969950.

[5] X. Ding and S. Kang. Adaptive time-stepping exponential integrators for cubic-quintic complex ginzburg-landau equations. 03 2017.

[6] M. V. Ferrer González. Lecture notes in continuous systems modelization, 2022.

[7] I. for Theoretical Physics. *Patterns and Interfaces in Dissipative Dynamics*. WWU Münster, 2006.

[8] J. A. Hernández Ramos. Numericalhub. https://github.com/jahrWork/NumericalHUB.

[9] J. A. Hernández Ramos. *Cálculo Numérico en Ecuaciones Diferenciales Ordinarias*. ADI. Aula Documental de Investigación, 2001.

[10] J. A. Hernández Ramos. *Análisis y Cálculo Numérico en Ecuaciones en Derivadas Parciales*. Escuela Técnica Superior de Ingenieros Aeronáuticos, 2002.

[11] J. A. Hernández Ramos and J. Escoto López. *How to learn Applied Mathematics through modern Fortran*. Escuela Técnica Superior de Ingenieros Aeronáuticos, 2017.

[12] D. M. Winterbottom. The complex ginzburg-landau equation. https://codeinthehole.com/tutorial/index.html.

[13] R. Munoz-Alicea. Introduction to bifurcations and the hopf bifurcation theorem for planar systems. 2011. URL https://www.math.colostate.edu/~shipman/47/volume3b2011/M640_MunozAlicea.pdf.

[14] L. Pismen. *Patterns and Interfaces in Dissipative Dynamics*. Springer, 2006.

[15] J. T. Stuart. On the non-linear mechanics of wave disturbances in stable and unstable parallel flows part 1. the basic behaviour in plane poiseuille flow. *Journal of Fluid Mechanics*, 9(3):353370, 1960. doi: 10.1017/S002211206000116X.

[16] L. Trefethen. *Finite Difference and Spectral Methods for Ordinary and Partial Differential Equations*. The author, 1996.

[17] L. N. Trefethen. *Spectral Methods in MATLAB*. Society for Industrial and Applied Mathematics, USA, 2000. ISBN 0898714656.

[18] W. van Saarloos, P. E. Cladis, P. Palffy-Muhoray, and F. der Wiskunde en Natuurwetenschappen. The complex ginzburg-landau equation for beginners. 1994.

[19] J. Watson. On the non-linear mechanics of wave disturbances in stable and unstable parallel flows part 2. the development of a solution for plane poiseuille flow and for plane couette flow. *Journal of Fluid Mechanics*, 9(3):371389, 1960. doi: 10.1017/S0022112060001171.