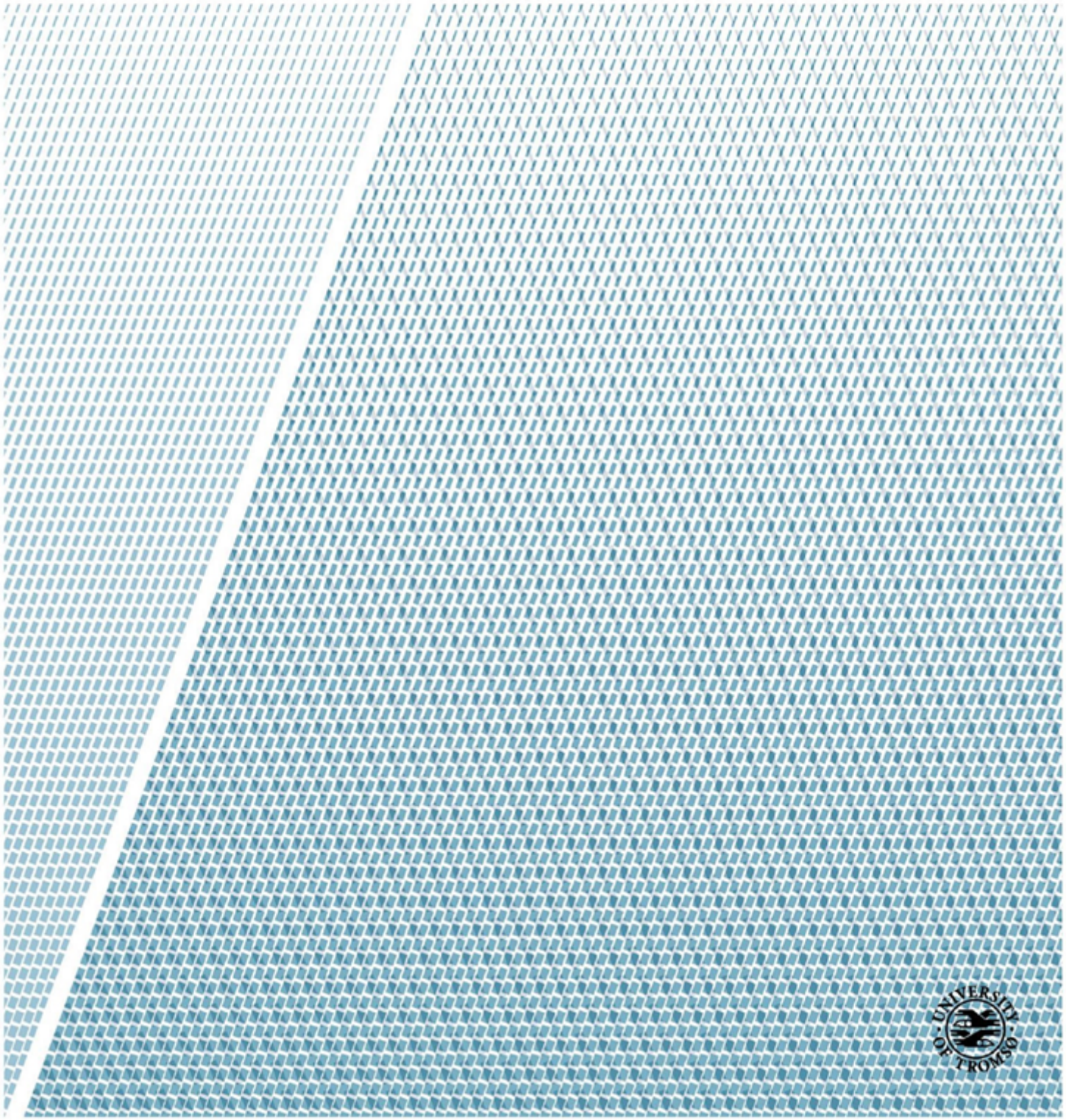
UiT
THE ARCTIC
UNIVERSITY
OF NORWAY

Department of Mathematics and Statistics
Faculty of Science and Technology

Modelling the evolution of ideal, infinite domain patterns, on a finite domain using a Perfectly Matched Layer

Andrey Antrushin
Master thesis in Applied Mathematics, January 2016



Abstract

The *Swift-Hohenberg* equation is an *evolution* equation which can produce a *Pattern*, or a pattern-like picture, to be more precise. For example, it could be used to model some simple natural patterns, like *stripes* and *rolls* that one may observe in a *Rayleigh-Bénard* convection experiment.

But for any pattern formation obtained by an *evolution* equation to look ideal, we have to consider this equation analytically on the *infinite domain*. If one wants to calculate and present the results *numerically*, the problem has to be *discretized*, the number of steps turns out to be *finite* then, and at some point the lateral boundaries appear. These boundaries cause a backward *reflection* and destroy the pattern eventually.

The idea of using a *Perfectly Matched Layer* in order to obtain some *reflectionless* boundaries was suggested. It should let us model the evolution of an ideal, infinite domain pattern, on an actual finite domain. From a set of *numerical methods* the most suitable ones will be chosen. The *MATLAB* environment is used to write a code to visualise the results.

Any investigations that could concern the fact of applying a *Perfectly Matched Layer* to an evolution *Swift-Hohenberg* equation seem to be absolutely new and yet untouched. This *Thesis* might be considered as a first step, as an introduction to the problem and its possible solutions.

Acknowledgements

I would like to thank my supervisor *Per Jakobsen* for his assistance in writing this *Master's Thesis:* from choosing an inspiringly exciting subject as *Patterns*, to the passing remarks about using a definite article in the text. I am very grateful for those moments, intentional or not, when I could feel his support and that I was not alone fighting with all the instabilities, as well as for those short, but brighter than anything else, moments of a sudden success and a triumph.

It is very important and just wonderful, when you get a chance to experience the way how teaching does actually teach and makes you learn even something more than a science or rules. When it makes you understand and see differently, or maybe inspires you for another new adventure. I am enormously happy that I got a supervisor who has this exciting and lively spark in his intelligent talks, in his unbelievable insights, in the helpful remarks he does.

I also want to thank the Head of the Office at the Department of Mathematics and Statistics, *Helge Johansen*, for his help with all the practical issues concerning this *Master's Thesis* and me being an international (quota) student at the University of Tromsø, for his great patient at some moments, and for being supportive.

Working on the *Thesis* I was somewhat 'on the road' most of the time. It will take another few pages if I try to express in words how thankful and pleased I am to meet and to spend the time with all the people from that 'road': to share meals, couches, ideas, experience, stories, happiness and smiles. Thanks for diverting me all the time from writing my *Thesis*, otherwise it would never be so overfilled with breathtaking moments, fun and amazing memories.

It happened that I was writing the last chapters being back home. So quiet and comfortable, with all the warmth and infinite love I thank my *Family* for just being my family.

Contents

1 Introduction

In this Master's Thesis we are going to face the problem of applying a relatively new approach of setting something that is called a *Perfectly Matched Layer (PML)* on the plane, and making it work for an evolution equation, such as a one-dimensional *Swift-Hohenberg* equation [1].

The main goal behind this investigation is to get a possibility to set absolutely *reflectionless boundaries* and be able to place them anywhere on the plane when proceeding with the numerical calculations of an evolution equation. This invention should let us obtain an exact picture of a *'Pattern'* that could be born by an evolution equation on the *infinite domain*, originally. The picture that won't be affected by any unwanted appearing instabilities, that are the subject of unavoidable reflections and disturbances from near the boundaries, while the boundaries are always present when solving a problem numerically.

In the very beginning there will be a short but inspiring (as it seems to me) review given about where and how the problem that we are about to study could be found or applied later on. Also we will be able to see how the things at first sight absolutely different in their nature could be considered as one, could be seen and studied as one, just due to the similar *Pattern* they might have. We will travel a bit in time to discover how incredibly exciting the history of a scientific 'underground' could be [2, 3], and we will take a ride through the Space, to go from the micro- to the macro-scales [4] and find out that it all looks the same sometimes, depending on how exactly you are going to watch at it.

As the process of doing this is absolutely new to us, and no previous examples of applying a *Perfectly Matched Layer* to an n-dimensional *Swift-Hohenberg* equation were found, we are going to do this step by step with a maximum possible thoroughness. There are few numerical and analytical methods to choose from [5], and in the very beginning it turns to be not quite obvious which one would strike out the best.

We will start with the decently simplified problem of considering a reduced *Schrödinger* equation on the plane where a mentioned above *Perfectly Matched Layer* has been applied. But first of all we will have to talk about the way we actually can set this 'magic' *Layer*. After choosing the best combination of the methods for our reduced problem we will move further to a one-dimensional case of a *Swift-Hohenberg* equation. But there will be also something more to learn about this evolution equation and its specifics [1]. And as a start-point here we will make it work with *Periodic Boundaries* first, which is basically not a new thing [6], but could be very useful in our future calculations.

The *Fourier Transform* [7] will be used as a tool to check the correctness of obtained results at some point. As well as a tool that provides a better understanding of the undergoing processes when running an evolution equation with an initial function introduced by a randomly distributed low-amplitude signal, such as a *White Noise* [8].

We will be confronted with few difficulties and some of them (the most important ones, of course) are going to be solved pretty unexpectedly and just amazingly wonderful in the very end of the work. Actually, exactly this thing

will let us finally introduce a solution for the setted goal and show the way how a one-dimensional *Swift-Hohenberg* evolution equation could be run on the plane with a *Perfectly Matched Layer (PML)*.

This work itself seems to be nothing but an introduction or the first step going into the investigation of a way to solve an evolution *Swift-Hohenberg* equation using a *Perfectly Matched Layer* set on the plane. We take our chance to do it as clearly for an understanding and as carefully about the calculations as it is possible. Because, we do realize that it might be used later by someone else to proceed on a further exploration, that is doubtlessly about to bring something new, awfully lot interesting and even more exciting.

***

It's funny and strange how hard it could be sometimes to start something. It always happens to me that I have to wait for a special moment, for a strong feeling about doing a thing. And what I'm waiting for at this moment I can compare, or I can even call it, the inspiration. Maybe the last one is the most right word for it. Inspiration.

What do we need to do, what should we look for to catch this feeling of an infusion? Moreover, should we really try our best to find it, turning up the heavy stones all around us, or dipping down to the very bottom of the unknown? Maybe the right way is just to wait? And when it comes, you will see and sense it clearly, you will understand that you are already there, you are inside the process! Probably, it would be hard to say, how it has happened that you have started something, but would it really be important? Now, when you have it as a part of you, when you do it with all your heart.

We need to be involved! We need to believe that we want to do this one thing in particular. Of course, there could be a lot of these things to do at the same time, but this one, it should be special, the unique one. It's not so easy sometimes to manage to build this strange belief and interest, and it's hard for me to say why. Most likely, because of the laziness and small futile fears. At the same time it's the most important step if you want to do something truly nice. I don't believe that without being involved with all your mind, heart and soul you can do a beautiful thing.

That what I was looking for all the time. And I call it the inspiration. From one hand because it comes suddenly, this feeling that you can start with the project, from the other hand, because almost every time it starts with an idea. An idea that could come from some random observation, an idea that you can catch from a song, or even feel in a friend's warm embrace.

And just after you got it, everything starts to evolve. It doesn't go really fast, it never goes fast. But it's everywhere around you, as well as it's always inside of you. It becomes so significant and so naturally simple at the same time, and you feel it as something essential in a way. The right thing to do now, probably, take your time and try to go until the very end, until the wonderful moment when you will understand that it's finished, that you have learned and understood everything you headed for. Certainly, there would be so many things

that you will discover on that way, and if you wish you can go for them as well, but later. Don't think they matter anyhow at this moment of approaching to the end. And it seems to be an exciting adventure, a breathtaking trip. You don't really remember how it has started, you are not sure any more what was there in the middle. You just know that you have enjoyed it and it was never in vain.

2 Patterns

I catch a snowflake, it lies quietly on my mitten and doesn't seem to melt away immediately, it is so easy to do here far up on the North. I have heard that every snowflake has its own unique way of flying down from huge clouds sitting and resting there high above our heads. It falls out of one of those clouds and let the wind and the gravity take it all the way through the changing air pressure and the gradient of the temperatures. Sometimes some sudden warm stream of the air will take it a bit up again, but then the endless or the momentary flying or falling (no, it is definitely the flight) will take its place again [4].

Suddenly, a snowflake will land here or there, right on the top of someone's hat or in the huge pile of the other snowflakes at the roadside [9]. They say, if you take a look at the snowflake and examine it carefully it is possible to tell quite accurate about the conditions it went through. In other words, you can tell what was going on with this tiny snowflake there in the air, so high above your head, until the moment you have found it, or it has found you. But the important and exciting thing now, is that you get a chance to observe its magical *Pattern* and learn maybe everything that has happened to it before it has landed down on your mitten.

If you have knowledge enough and a guess about what tools you have to use, and not the last, if you have a passion for it, then we can be sure that you will go for a try and tell us everything you've got to learn about its breathtaking adventures and nature.

It seems to me, that with the Thesis, or could be with almost any other work that one is going to do, it happens in a very similar way. It usually takes some time before you are done with a chosen project. It takes your attention, takes your strength, it might take you through the different distant or nearby places, real or imaginary ones, to show something. But after all, when looking on the completed work, looking carefully through all its 'chapters', one clearly sees, as in the story with a snowflake, how it was made. At least, this is how it should be, I guess. Otherwise, it means something is lost, something is missing, and probably, will stay misunderstood.

Maybe, if the work misses this part, eventually the result won't be shown and exposed the way it should or could be. Most likely, it would be possible to see the idea behind a single taken 'chapter' or some randomly appeared definition on one of the pages, but something essentially common like the properties of a water molecule that lies in the origin of every snowflake and its *Pattern* formation could stay unnoticed. Meanwhile, everything would make no sense without it. So, let the *story* begin.

***

I should probably start from the very beginning. How I came to the problem this work devoted to? I was very curious at one point about studying *Astrophysics* (and, I still feel a great interest and desire to do it). I remember, the first time when I asked if I could write my Master's Thesis on some problems of *Astrophysics*, I was told "no", but it wasn't that sharp and cold rejection. I could feel that there was something interesting and prepared behind this, and it seemed that everything had been decided already, without my involvement. But, actually, only for a good reason, because it was something I would like to do, even without knowing about it myself. *Patterns*, that what it's going to be about.

Figure 1: Pattern on the peel of a Melon.

Sounds exciting, isn't it? I have never been thinking a lot about *Patterns* before, I have never heard that much as well[1]. I remember only once, the first

[1]Though, since I had learned the topic of my Master Thesis, I started to pay more attention (not on purpose) to the patterns everywhere around us. Thus, this one (Fig. 1) I saw on the background of a Web-browser. But actually, there were more 'romantic' natural examples I had found. Unfortunately, they do always happen quite unpredictably and hard to be caught in the way to be visually presented later. While, this one just looked bright enough and easy to get. But my memory will stay full of those wonderful patterns I have seen by now and hope to find more.

time when I learned something about patterns and about their formation was in a documentary[2] from BBC [3]. And what I saw there was called a **Stirred Belousov-Zhabotinsky** reaction. The experiment showed that if you put two specific chemical solutions in one glass and mix them, then the substance will start changing the color periodically. Switching from white (clear) to yellow and back! All our class was shocked, and it was even more fantastic when we saw another example of this experiment when you observed some circles appearing on the surface of one substance, growing and interfering in time. This one was called a **Belousov-Zhabotinsky** reaction in a **Petri dish**[3].

After we had finished the documentary we ran to the Chemistry Department with our professor and asked their students if they had known something about it and if they could show us these reactions live. Of course, they knew. Moreover, they agreed to make an experiment of a Stirred Belousov-Zhabotynsky reaction, thus, in a few minutes we could see it with our own eyes.

But that was all! The glass with the chemical solution that changed its color periodically and impressed faces of students, carefully keeping an eye on it. After the experiment was over, honestly, I switched back to the Astrophysics really fast. I mean, I am pretty sure that I forgot about this experiment the next day. But it seems not forever. I was very surprised when in one of a Special Curriculum courses (I should probably add, 3 years later), I stumble upon this experiment again. I was reading through the book, I already knew that the subject for my Thesis was "Patterns", but hadn't recalled something I knew from before, until I really ran into it.

And that was a strange sudden happiness to understand that you are reading about something you have heard before, furthermore something that you have seen with your own eyes. Seems like you never know for sure, how the knowledge or the experience you obtained before, could be used in the future. You can not really predict what you will need later and what you can throw aside right now, maybe the best way to do, is to perceive and try as many things as you are interested in. Especially, if you have fair chances to do it.

I don't know why, but the first time I recalled the BBC documentary I was thinking about Belousov-Zhabotinsky stirred reaction (the one, when the solution changes its color periodically) as if it was the reaction called "Turing's reaction". Definitely, it was a mistake as I found later. Just because in the documentary they were talking a lot about **Alan Turing**, his life and work[4]. My memory, probably, played a trick on me and let me thinking that this pattern-forming reaction bears the name of Turing.

However, it was correct in another way, because Turing's mathematical model was exactly the one needed to describe Belousov's experiments. Unfortunately, it happened that neither Turing, no Belousov knew about each other work and

[2]Actually, we had been shown this documentary during my studies back to Russia, on the short course in Astrophysics that we had.

[3]There are plenty of video examples on both "stirred (oscillating)" and "in a Petri dish" **Belousov-Zhabotinsky** reaction on the Web, so if you feel curious just take a look.

[4]I am going to speak more concrete about both: Turing's theories and Belousov-Zhabotinsky reactions – later on, in Chapter 3.2.

explorations, also both had quite a tragic destiny [3]. But I will tell about it later (Chapter 3.2).

What I wanted to mention now, is that when I was reading the literature for my Special Curriculum course I was really happy to feel I know something about **Turing** from before, even though my ideas were wrong in a way. Nevertheless, this funny feeling made me more curious about the whole project and even the subject of studies. Maybe it gave me some confidence and pushed me forward. Quite queer, how just the feeling of knowing something, ignites the interest to go further and do more. And there I have started.

3 Examples of natural pattern systems

In this chapter I will give a short and simple description of few most known and important natural pattern systems. Saying "natural" I mean pattern that could be seen in real natural systems – *deserts, stars, animal coats, fingerprints* – but also I mean patterns, that could be produced in *laboratory* (or even home) experiments. I think it is the right thing to start with, just because it could be useful and interesting to observe the picture or idea of how different patterns can appear, how they look like and develop in time and space, what are the similarities between patterns formed by various systems, and what are the most studied or unknown aspects.

3.1 Convection (Rayleigh-Bénard)

Simply, convection is the overturning of a fluid that is heated from below. Imagine the experiment where you have two horizontal plates, such that the bottom plate is warm and the upper plate is cool. And between these two plates we observe a layer of fluid, e.g. air or water (Fig. 2). This experiment is called Rayleigh-Bénard convection.

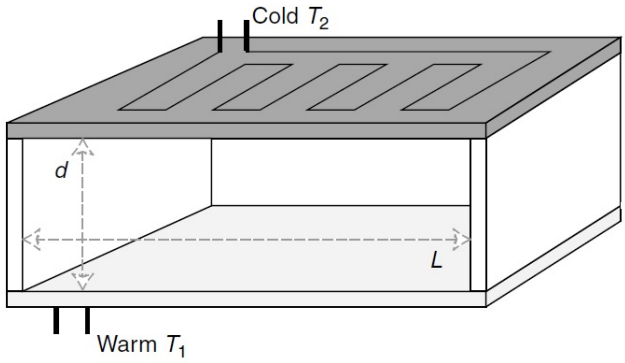


Figure 2: System for the Rayleigh-Bénard convection experiment.

Heat at the bottom plate causes the fluid to expand, become less dense and more buoyant and so to rise through the colder fluid above. As the fluid rises away from the heat source, it cools, becoming denser than the fluid below, and so falls

back down to the bottom under the influence of gravity. The cycle then repeats, so the fluid is constantly overturning. The rising and falling fluid forms spatial patterns, most commonly **stripes** or **convection rolls**[5] (Fig. 3).

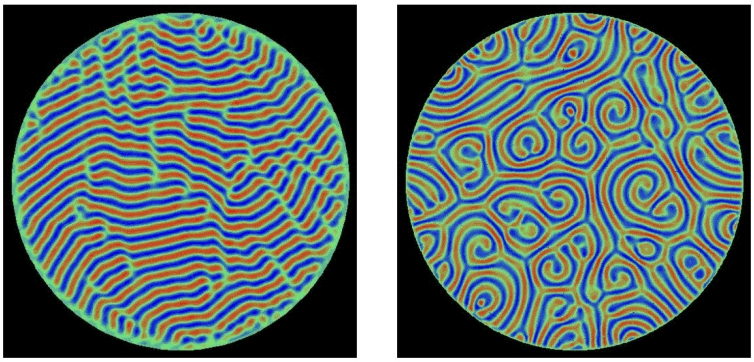


Figure 3: Possible patterns that can be observed in a Rayleigh-Bénard convection experiment, depending on the system and fluid properties. *Stripes* (on the left) and *Convection Rolls* (on the right) [10].

Though, more complicated patterns such as hexagons and squares are also possible, depending on the details of the physical system and the fluid properties. Convection is actually one of the most studied and known processes forming a pattern, and the reason it has been studied so extensively is that convection occurs naturally in the environment: in the Earth's mantle, convection leads to the movement of tectonic plates ('continental drift'); in the oceans it drives circulations such as the Gulf Stream that keeps north-western Europe so much warmer than its northern latitudes would suggest; in the atmosphere, convection creates thunderclouds and in stars, such as the Sun convection transports energy efficiently from the core where it is produced to the surface where it is released.

Another thing about the importance of convection is that the stripe patterns (Fig. 3), we can observe during the experiment[6], are very similar to the other different natural patterns and usually have the same kind of *dislocations* (when two stripes merge into one) which makes it very useful to work with, because once we find the solution (equations) to describe one of the patterns we will get the answer for other systems as well.

I extremely like this idea, how two, three, four, even more naturally different (as it seems) systems turn out to have a similar structure or behaviour, and could be called identical in a way, moreover, could be described with the same mathematical models. It brings this wonderful feeling about connection between everything in the world.

[5]The characteristic roll size is about the depth d of the air.

[6]Probably, it would be nice to mention, how actually it is possible to observe the pattern of convection. In the laboratory, it is usually visualised using the shadow-graph technique. In this method, a light is shone down onto the convection cell, which must have a transparent top plate and a reflective bottom plate. The warm rising fluid has a lower index of refraction that the cold falling fluid, and so the light is focused towards the cold regions, which appear bright, while the warmer regions remain dark. The pattern can be seen reflected off the bottom plate.

Just to visualise this aspect and emphasise the fact more thoroughly, I would like to make a comparison of patterns that belong to naturally different systems. I guess, everyone is most likely familiar with these systems. But it could be exciting and thoughtful enough if we pay our attention one more time to something like this and maybe make some new conclusions or get any fresh ideas.

Figure 4: Starting from the left: sand ripples, zebra stripes, and on the last picture you can see the image of the pattern obtained from convection in carbon dioxide under pressure [11].

Figure 5: On these pictures you can see hexagon patterns. Of course, everyone knows giraffe's coat with its gorgeous spots, but on the right you can see hexagons obtained in a kitchen experiment as it claims in [12]. The experiment is about mixing some black pepper and oil and frying it (actually, warming it up) on the pan. In the end one can obtain this interesting hexagon pattern.

3.2 Reaction-diffusion systems

Remember, on the page 7, we have mentioned for a while about **Turing's** theories and **Belousov-Zhabotinsky** reactions. That is what you will find in this Chapter! **Turing's** models and **Belousov-Zhabotinsky** reactions are the most representative cases of a *reaction-diffusion* system. Thus, it is time to talk a little bit more about them. In my own opinion, these reactions and patterns they create, are the most fantastic and amazing due to the way they appear and proceed and also from the historical point of discovering and working on them.

The best way to start is to start with **Alan Turing**, his insight and experiments. During the World War II, Turing managed to crack the Nazis' *Enigma Code*, but it seems that his real destiny was in cracking another kind of code, one that will help to understand how animals, human, everything in the nature could develop from chemical substrates. Turing believed development could be reduced to mathematical axioms and physical laws. In 1952, one of his articles called "The Chemical Basis of Morphogenesis" was published [2], it described the way in which non-uniformity (natural patterns such as stripes, spots and spirals) may arise naturally out of homogeneous, uniform state. Though it was a biological article, it made a great impact for understanding of nature, due to the absolutely new way of thinking and seeing the possibility to create a simple mathematical model which will describe pattern formation and evolution for life forms. The most shocking thing is that it took 60 years afterwards to prove (experimentally) his theory [13].

The central idea behind the theory is that two homogeneously distributed substances within a certain space, one "locally activated" and the other capable of "long-range inhibition", can produce novel shapes and gradients. What is special about such a model is that it can explain pattern formation without a preformed pattern. That is, the *reaction-diffusion model* can explain how those initial patterns form in the first place.

Turing's model would describe perfectly what the Soviet scientist Boris Belousov obtained in his experiments. Belousov studied the way how live organisms can get energy from sugar. One day mixing some chemical solutions together he noticed that one of the mixture changed its color from bright to dark. And maybe that was all right, but the strange thing happened after, the same solutions mixture, while being stirred, changed its color again, back to bright (Fig. 6). That was something unexpected and certainly unpredictable for Belousov. The system itself behaved as an oscillating system, but without any external influence and seemed without any reason for this.

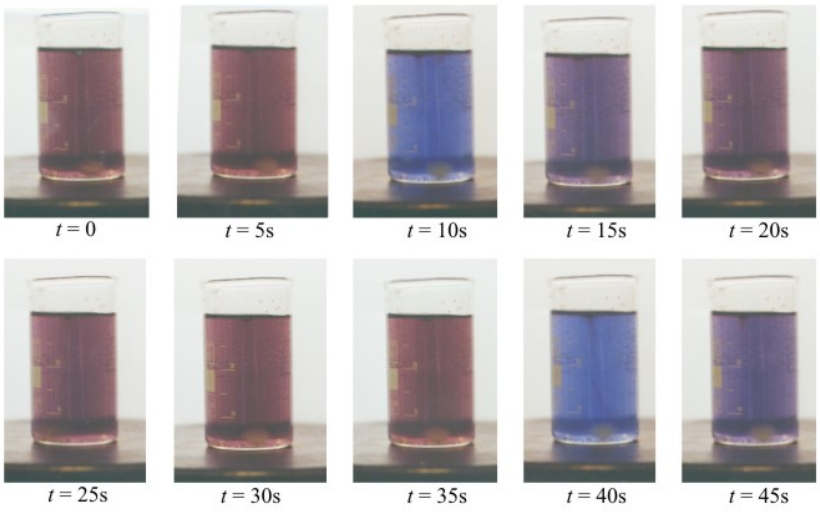


Figure 6: A stirred Belousov-Zhabotinsky reaction mixture showing changes in color over time.

Belousov checked his experiment several times, and always there was the same result – the mixture of several chemical solutions was changing its color in oscillating way while being stirred. He wrote an article about it, but of course without any mathematical model describing the process. It was where the tragic thing had started, the scientific society didn't take his article seriously, saying that it's simply impossible, what he observed, and that there should be a mistake in his calculations or experiment. It was the time of the *Cold War*, the time of an information blockade, and Belousov had no idea about the works of his English "college" Alan Turing. Otherwise, using Turing's theories (which described the pattern formation) he would be able to prove that his observations were correct and moreover study and explain them thoroughly.

After a rejection to publish his article and an accusation of being incompetent, Belousov was so depressed, that he quit his scientific research work. Alan Turing had even more tragic destiny, he was prosecuted for homosexuality, since homosexual acts were a criminal offences those times. Somehow police learned about it and instead of going to prison he chose a way of being undergone through hormonal treatment designed to reduce libido[7]. Choosing treatment he had a chance, of course, to keep working on his theory. However, one year later he committed suicide with the cyanide poisoned apple.

I don't know, nobody knows, how would the history go if two of these great men would keep on their work and experiments. We usually regret about something and start to take it more seriously only when we lose it. We start to create this horrible hypothesis called "what if", but they don't actually help a lot, obviously, they do nothing at all! It seems we just appreciate this feeling and condition of regretting. Sometimes it seems to me that everything goes the best way it could go, it goes the right way, even if the road and events you face, suddenly look strange or unknown. Who knows, maybe that was the only way (though, long way) to find out the truth and to learn something new.

In the end of this chapter I just want to inspire you and myself with the beautiful, wonderful pattern that could be obtained in **Belousov-Zhabotinsky** reaction in a **Petri dish**. And which could be described by Turing's theory. There is a nice still (Fig. 7) from when running this experiment under the laboratory conditions, but, of course, it is much more exciting to see it live[8]. At the same time, I want to mention again and want us not forget, that the theory and model of a pattern formation that Alan Turing had discovered you can see everywhere in the nature! So maybe take a look again on some *sand ripples* or *zebra stripes* (Fig. 4), as well as *giraffe's spots* (Fig. 5).

[7]The treatment rendered Turing impotent and caused gynecomastia.

[8]I encourage you to go on the Web, and see this experiment and other *reaction-diffusion* systems producing amazing patterns. There are some nice examples of Belousov-Zhabotinsky reaction here: https://www.youtube.com/watch?v=3JAqrRnKFHo and https://www.youtube.com/watch?v=fzcISz-ZcRk

Figure 7: A still of a pattern obtained from Belousov-Zhabotinsky reaction in a Petri dish.

3.3 Universe. Patterns on Huge Scales

Probably, there would be no practical use of this chapter, but I just really wanted to show some more of wonderful pattern pictures and, honestly, to go a little bit far beyond our usual understanding and expectations. Because, it seems pretty astonishing to me when you first have got a chance to compare something going from one place to another but staying mainly with the same scaling, and soon you are travelling through time and space until you reach such huge scales which are certainly hard to grasp and to believe in, but suddenly show you something you have seen before. Guess human is quite curious about finding similarities in everything.

It is still not clear how the existence of matter and light, together with the equations that determine their behaviour, produce the extraordinary complexity of the observed Universe. Instead of all matter in the Universe being clumped together in a single black hole, or spread out in a featureless cloud, we see with our telescopes a stunning variety of galaxies of different shapes and sizes. The galaxies are not randomly distributed throughout the space like molecules in a gas but are organized in clusters, the clusters are organized in super-clusters, and these super-clusters themselves are organized in voids and walls.

It is known now that our Universe is everywhere expanding, with all faraway galaxies moving away from each other and from the Earth. And according to the **Hubble law**, the galaxies that are furtherest away moving the fastest. Hubble

law looks very simple

$$\nu = H_0 \cdot d, \tag{1}$$

where ν is the so-called speed of a galaxy, H_0 is a Hubble constant, and d is the distance to a galaxy[9]. Another simple fact that the light from a galaxy that is moving away from Earth is *Doppler-shifted* to a longer wavelength (becomes more red) compared to the light coming from an identical but stationary galaxy, gives a possibility to measure the extent to which known spectral lines are red-shifted. Thus, astronomers can estimate the recessional speed ν of a galaxy and convert this speed to a distance d by using Hubble law (1). After we have all the distances to all observed galaxies, and we can actually observe all of them. Because as they say, the *Hubble Telescope* can see up to the edge of the Universe! Thus, having all these distances astronomers build a kind of a distribution map (Fig. 8) with the Earth (or more correctly, the measuring device) in the middle.

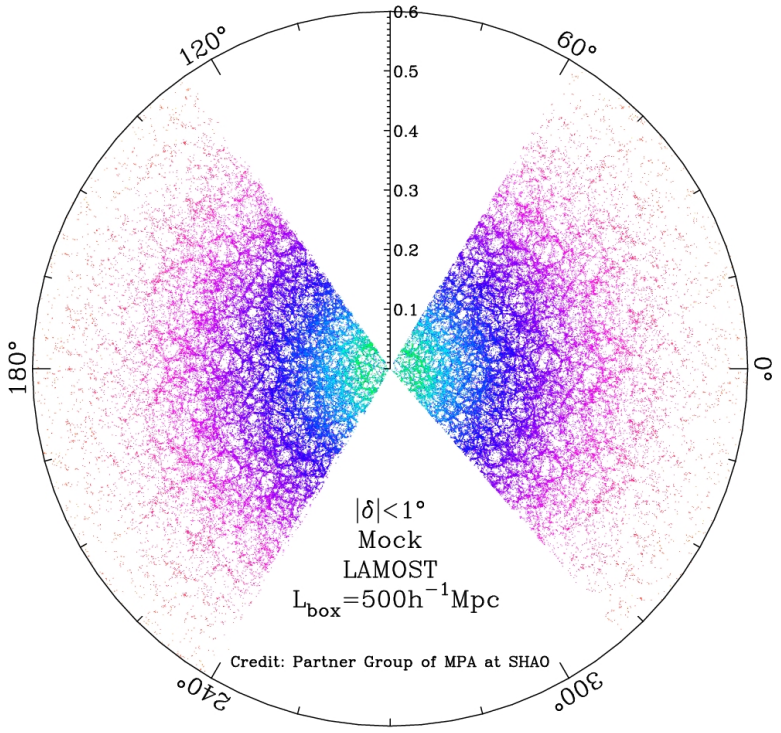


Figure 8: A "slice" through the 3-dimensional distribution of galaxies. The observer on Earth is located at the centre of the circle. The distance from the centre of the circle represents the red-shift of the galaxy. The galaxies within 1 degree of the equator are plotted in this diagram [14].

The pattern here won't be a geometric structure (e.g. a lattice) but will appear as statistical deviation from randomly and uniformly distributed points that is somehow difficult for the human visual system to quantify. Perhaps the closest earthly analogy would be a foam of bubbles in which the galaxies are concentrated on the surfaces of the bubbles. The reason for this galactic structure

[9] I feel very glad to write for a while about *Astrophysics* again. Especially, because this topic was part of my Bachelor's Thesis [4].

is not known at this time but is presumably a consequence of the details of the Big Bang (when matter first formed), the expansion of the Universe, the effects of gravity, and the effects of the mysterious dark matter that makes up most of the mass of the Universe but which has not yet been directly observed or identified.

Another interesting example of a grand pattern formation is any spiral galaxy. We can take the one called M74 (Fig. 9), located in the constellation *Pisces*. It is at a distance of about 32 million light-years away from Earth. The galaxy contains two clearly defined spiral arms and is therefore used as an archetypal example of a Grand Design Spiral Galaxy.

Figure 9: Photograph of the M74 spiral galaxy, a gravitationally bound island of 100 billion stars, approximately 100 000 light years wide, that lies about 32 000 000 light years from Earth in the Pisces constellation.

Why galaxies evolve to form spiral arms is poorly understood and is an important open question in current astrophysical research. Some laboratory experiments show that spiral formation is common for non-equilibrium media that have a tendency to oscillate in time or that support wave propagation. Further, experiments show that a tendency to form spirals is insensitive to details of the medium supporting the spiral. So a galactic spiral may not be too surprising since there are mechanisms in galaxies that can produce wave propagation [1]. For example, some researchers have proposed that the spiral arms are detonation waves of star formation that propagate through the galaxy, somewhat analogous to the excitation waves observed in the Belousov-Zhabotinsky reaction-diffusion system (Fig. 7).

It is wonderful in a way how fast we can jump back to the human scales from

Huge Universe, just finding similarities in the observed pattern formation. The last example that I would like to introduce in this chapter also seems to be a connection between different scales. I think it has been mentioned already in the very beginning. Our Sun, and it's convection rolls.

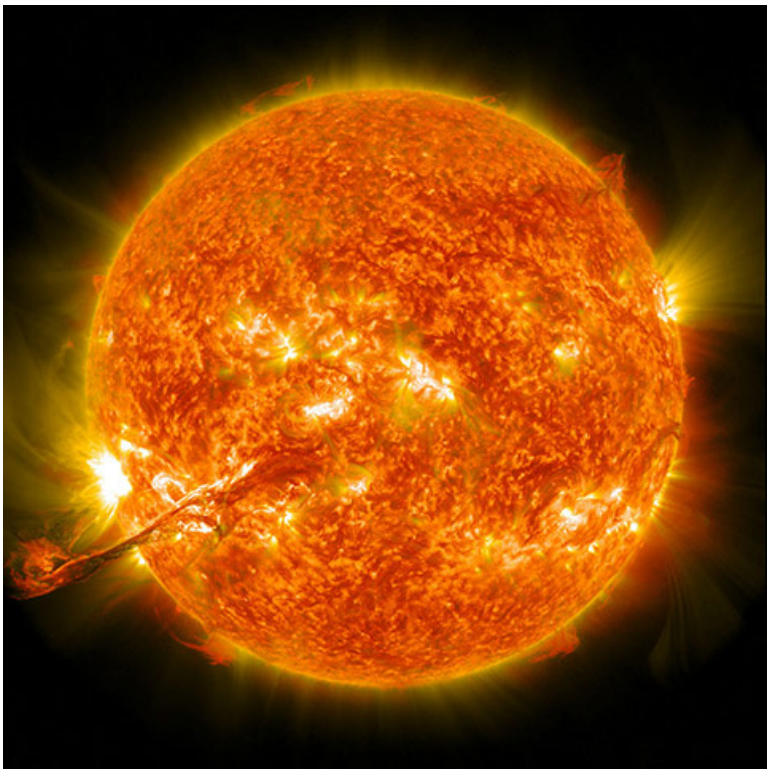


Figure 10: Photograph of the Sun's surface by *Hubble Telescope*, showing a complex time-dependent granular structure.

Heat diffuses by collisions from the Sun's small dense and extremely hot core (20 million degrees Kelvin) out to about two-thirds of the radius of the Sun, at which point the heat is transported to the cooler surface (about 6000 K) by convective motion of the Sun's plasma. The small bright dots (Fig. 10) are 1000 km-sized features are called "granules" and correspond to the top of convection cells, the darker boundaries are where the cooler plasma descends back into the interior[10].

Similar phenomenon of stripped pattern formation due to the convection is common to all of the gas giants (Jupiter, Saturn, Neptune, and Uranus). By example of Jupiter, careful observation of the bands and of their dynamics shows that they are highly turbulent time-dependent flows of the outer portion of Jupiter's atmosphere, with adjacent bands flowing in opposite directions with respect to Jupiter's axis of rotation.

The mechanisms that drive these non-equilibrium stripes and spots are not hard to identify. Jupiter's core is known to be hot and the transport of heat from

[10] The only difference from the convection we observed in Chapter 3.1, is that the Sun's plasma is a highly conducting electrical medium and its motion is influenced by the Sun's magnetic field and the magnetic field in turn is modified by the motion of the plasma.

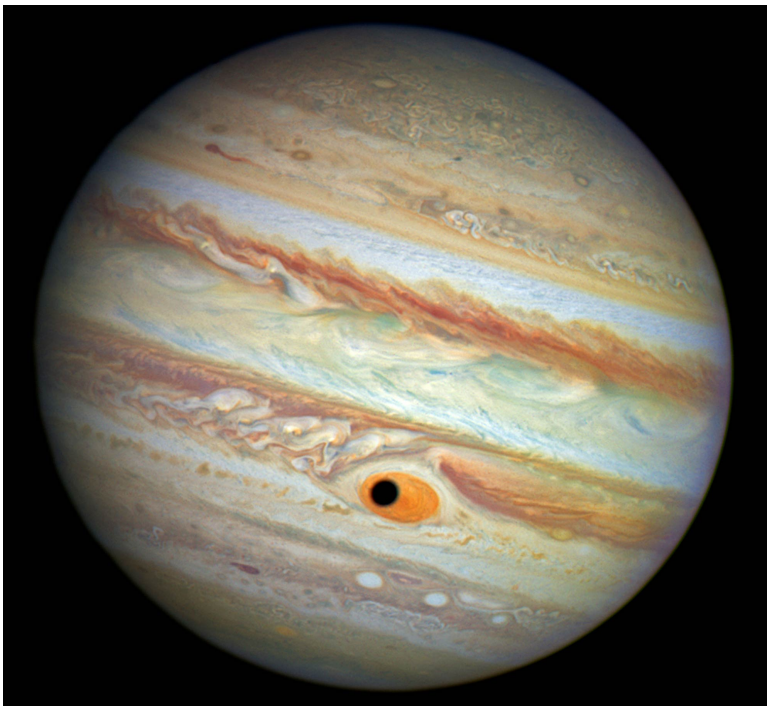


Figure 11: Jupiter by *Hubble Telescope*.

the core out through its atmosphere causes convection in the outermost layer, just as in Rayleigh-Bénard convection we discussed in Chapter 3.1. However, the convection is substantially modified by Jupiter's rapid rotation around its axis, about once every 10 hours. As warm and cold parcels of fluid rise and descend, they are pushed to the side by large Coriolis forces and so follow a spiralling path.

Hope these beautiful *Patterns* in the last three chapters and a magic idea of of comparing different systems, finding similarities and feeling some prescribed unique structure for everything, hope that it was inspiring and encouraging enough to move further to the real problem of this *Master's Thesis*, which I will try to make more clear in the following chapters. And if the pattern suddenly looks boring, or maybe even unknown, remember, it's just the simplification, it is just something to start with. But we shouldn't forget, in no circumstances, where all these patterns come from and how HUGE is the range of their application: from a Petri dish to the galaxies in the Universe.

4 PML and the reduced Schrödinger equation

We always need to find some starting points, anything that will let us push ourselves through all the unknown which will become clear and maybe even trivial soon. Something that will give us a chance to come close enough to the problem we are really interested in, and something that will later make us ready to try to solve this problem without any useless fear, but just with the pure curiosity and passion. It's hard to say what you should actually do when

the things you have to work with look completely unknown to you, and we usually start with something simple but similar in a way, slowly clambering up. This work is not an exception, especially because things I have to do: some of calculations, programming (coding) and some of mathematical modelling – is absolutely new to me. So for the beginning we use a simple reduced *Schrödinger* equation, we consider it only in 1D, and we use *MATLAB* instead of the "normal" programming environment (as *Python* or *C*).

The main idea for this chapter is to write such a code in *MATLAB* that will calculate and plot (animate) a moving wave packet, and before this moving wave packet hits the boundary (and there is one, because we want to solve the problem numerically) it should be maximum absorbed by the PML[11]. I had no idea what PML was, before I started working on this problem, and that's why I keep on writing it as an abbreviation. It was very funny to hear and even use it in discussions without really knowing what is it about. But it was all right, since I understood somehow the effect that this strange PML produces, so I even don't know how long it took me (maybe month) before I really decided to learn more about it. But here I decided first to make it clear about the "tools" we will use and then tell how it was actually going.

4.1 What is PML (Perfectly Matched Layer)?

Yes, *Perfectly Matched Layer*, that is what the abbreviation stands for. But I guess the name of these almost 'magic tool' doesn't completely reflect all the greatness of its idea and applicability. Basically, what PML is, turns to be some kind of an artificial absorbing layer (or material if you want), at least it behaves like it, or even better because the key property of a PML that distinguishes it from an ordinary absorbing material is that it is designed so that waves incident upon the PML from a non-PML medium do not reflect at the interface. This property allows the PML to strongly absorb outgoing waves from the interior of a computational region without reflecting them back into the interior [15].

Another thing that makes it more interesting and wonderful that it was derived not so long time ago, by Berenger in 1994. Before that we have been using so-called absorbing boundaries, or absorbing boundary conditions (ABCs) [16]. But there were several troubles about using them: the first one is that even if the boundary condition sets the solution to zero (*Dirichlet* boundary cond.), anyway we get some reflection from the boundary when the wave hits the edge; and second is that ABCs work perfectly only in one dimension, where waves can only propagate in two directions ($\pm x$). However, the main interest for numerical simulation lies in two and three dimensions, and in these cases the infinite number of possible propagation directions makes the ABC problem much harder [17].

But what Berenger did, he changed the idea! Instead of looking for an absorbing boundary *condition*, he created an absorbing boundary *layer*. This layer was kind of an artificial absorbing *material* that is placed adjacent to the edges

[11] And it will take us a lot of time to find a suitable numerical method to make it work with even this simple model.

of the grid, and what is more important it was completely independent of the *boundary condition*. When a wave enters the absorbing layer, it is attenuated by the absorption and decays exponentially; even if it reflects off the boundary, the returning wave after one round trip through the absorbing layer is exponentially tiny. The problem with this approach is that, whenever you have a transition from one material to another, waves generally reflect, and the transition from non-absorbing to absorbing material is no exception – so, instead of having reflections from the grid boundary, you now have reflections from the absorber boundary. However, Berenger showed that a special absorbing medium could be constructed so that waves do not reflect at the interface: a *perfectly matched layer*, or PML. Although PML was originally derived for electromagnetism (Maxwell's equations), the same ideas are immediately applicable to other wave equations.

Probably, aforesaid gives a nice picture about what PML is, but not about how we actually apply it to the equation. I think I will not give any explanation here about the way of setting up PML, because I believe it's much more easier to see and to learn by an example, and the next few chapters will give us such chance. But if you want to read more thoroughly about it I found this article [17] very helpful.

Maybe one more thing I want to mention, before we jump to the practical part, are some of the limitations and failure cases for PML. First, and most famously, PML is only reflectionless if you are solving the exact wave equations. As soon as you discretize the problem (whether for finite difference or finite elements), you are only solving an approximate wave equation and the analytical perfection of PML is no longer valid[12].

But even if we discretize the problem, PML is still the same absorbing material: waves that propagate within it are still attenuated, even though they are discrete waves. The boundary between the PML and the regular medium is no longer reflectionless, but the reflections are small because the discretization is (presumably) a good approximation for the exact wave equation. And we should not forget that the key fact, that even without a PML, reflections can be made arbitrarily small as long as the medium is slowly *varying*. Experience shows that a simple quadratic or cubic turn-on of the PML absorption usually produces negligible reflections for a PML layer of only half a wavelength or thinner. And, of course, increasing the resolution also increases the effectiveness of the PML, because it approaches the exact wave equation. I guess after this short observation we can finally start trying to solve our 'first-step' problem of adding PML to the plane where a reduced Schrödinger equation is going to be solved.

4.2 Several attempts (that failed)

I have all my thoughts and reflections in a big A4 notebook. Everyday when I was working on the problems: trying to make a code, figuring out the sense of

[12] Guess if I knew this aspect before I started to write my code it would help me a lot, because I was looking in a way for this really perfectly matched layer, but of course this fight with instabilities was worth it.

parameters in the equation, fighting with forgotten sign when making calculations – I was writing it down in this A4 notebook, and sometimes there were not only mathematical remarks but also some random ideas and thoughts coming to my mind under the influence of something that had happened or without. You could see the notes becoming really inspired after short talks I had with my supervisor. And honestly, I really wanted to make this *Thesis* look like a book, or maybe a pocket-book, or kind of a diary, whatever. I have been reading one of the Castaneda's volumes at that time, and all my notes were dated. I also felt this interesting, important connection between the teacher and the student [18].

But anyhow, I guess writing *Thesis* in that way would be inappropriate, it would just never be a *Thesis* then. Maybe something good and interesting, but not a *Thesis*. However, I let myself do these short digressions through the whole work. Seems it's like those tiny reflections from the PML – they don't really matter if you don't want them, and you can disregard them easily, but someone probably will be interested in studying and taking them into consideration, and even finding them useful[13].

So, with an idea to write a *MATLAB* code that is supposed to solve a wave equation (which is an evolution equation in time as well [19]) numerically using one of the methods, and also make the problem reflectionless for the boundaries by using PML – we start with the simple model, and as a 'first step' consider the next reduced *Schrödinger* equation,

$$i\frac{\partial U}{\partial t} = -\frac{\partial^2 U}{\partial x^2}. \tag{2}$$

Just to remind about the "normal" *Schrödinger* equation, here it is

$$i\hbar\frac{\partial}{\partial t}\Psi = -\hat{H}\Psi, \tag{3}$$

where i is the imaginary unit, $\hbar$ is the *Planck* constant divided by 2π, Ψ is the wave function of the quantum system, and $\hat{H}$ is the *Hamiltonian* operator (which characterizes the total energy of any given wave function and takes different forms depending on the situation). In our 1-dimensional case (2), it looks like $\partial^2/\partial x^2$, but originally for the non-relativistic *Schrödinger* equation for a single particle moving in an electric field it will take next form

$$i\hbar\frac{\partial}{\partial t}\Psi(\mathbf{r},t) = \Big[\frac{-\hbar^2}{2\mu}\nabla^2 + V(\mathbf{r},t)\Big]\Psi(\mathbf{r},t). \tag{4}$$

So what we do, just get rid of the terms that we don't really need for our simple, kind of sketch system. Thus the particle's 'reduced mass' μ and potential energy V go away, and for the start we choose to have only one dimension along x-axis, that's how we end up with eq. (2). But of course we want the equation stay time-dependent, which let our system to evolve with time, because any mathematical model that describes pattern is an evolution equation itself. So,

[13] Maybe I should change the color or font for this digressions, I will see and try it later, maybe.

we tend to make a simple model in the beginning, but at the same moment we try to build it as close in details as possible to the system we want to study later.

If we look for the solution of (2) it could be found in a form of a plane wave,

$$U(x,t) = Ae^{i(kx-\omega t)}, \tag{5}$$

where k is a wave number

$$k = \frac{2\pi}{\lambda} = \frac{2\pi\nu}{v_p} = \frac{\omega}{v_p}, \tag{6}$$

with ν is the frequency of the wave, λ is the wavelength, $\omega = 2\pi\nu$ is the angular frequency of the wave, and v_p is the phase velocity of the wave. If we try to plot the real part of $U(x,t)$, we will get the *wave packet* moving along the x-axis to the right ($k > 0$) or to the left ($k < 0$). Wave packet looks like this "monster" on (Fig. 12). I think it really does look like a friendly monster, especially when you see it moving on the plane.

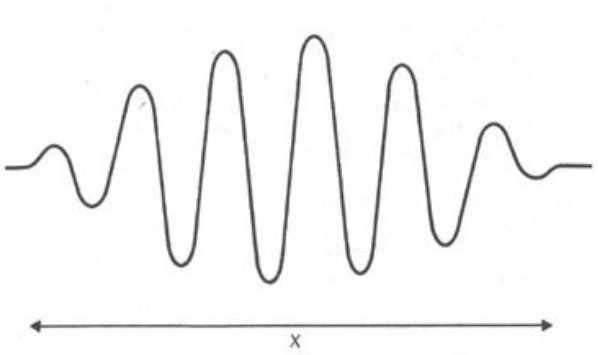


Figure 12: A wave packet corresponding to a particle located somewhere in the x region.

And to make it move on the plane we want to solve the equation numerically, but we don't really know which method should be used. Or maybe it's just me who had no idea which numerical method would be the most appropriate for solving this problem, however, it took quite a long time to figure out. I am almost sure that it was not so hard, and maybe even obvious from the very beginning what direction we should go, but at the same time I believe in usefulness of a studying based on making mistakes. So I took my time.

As soon as our purpose is to obtain the system with PML (Chapter 4.1) for the reduced *Schrödinger* equation (2) we first of all have to expand the problem to a complex plane, even before choosing any numerical method for calculation. That's actually how the PML works for this case – we need to expand our *'real'* problem to the *complex plane*. But I guess I will use a separate chapter (Chapter 4.3.1) for this, while in the present chapter, running few steps forward, I will just tell which numerical methods has failed when solving the problem. It won't take long now, though it took really a lot of time for me to go through all the crashes and instabilities.

The first numerical method that I used, was a *Finite Difference Method*. Maybe it is the most common method to start with, and maybe usually it works,

but not with this case. My slowness and uncertainty in coding took me almost half of a month to see and believe that all attempts were in vain. Maybe not in vain, but at least ineffective. Later with the help of my supervisor we managed to show why it was really impossible to do it with this method (I omit the explanation here). But I should say that using *Finite Difference Method* and even all its possible forms led to instability of a system. Even if we saw the wave packet almost perfectly vanishing when reaching the PML-point, it came back soon in the form of jumping and dancing instability, like from nowhere. I guess I was lost for a while, what are we going to do now? The method didn't work out! But I forget the most important thing, that it had really been only the method, only one of plenty. And that's how the true search of a method which would be free of instabilities had began.

Next thing I was suggested to do, was to look through two examples (called *Etude 12.2, 12.4*) in [1] and try to solve our problem using a similar method to the one that was used there. Actually, it is based on *Backward Euler Method*, but with some interesting and useful remark lying in calculating the *Tridiagonal Matrix*, which looks like this,

$$M = \begin{pmatrix} a_1 & b_1 & & & & \\ c_1 & a_2 & b_2 & & & \\ & c_2 & a_3 & b_3 & & \\ & & c_3 & \ddots & \ddots & \\ & & & \ddots & \ddots & b_{n-1} \\ & & & & c_{n-1} & a_n \end{pmatrix}, \tag{7}$$

for the right (spatial) part of (2), and later for the right (spatial) half of its version extended to the complex plane, I mean after applying the PML. And if for the solution **without** *Perfectly Matched Layer* it worked **perfectly**, just as well as it did in the book [1] – *exact* and *numerical* solutions matched very well, and *numerical* one decayed slowly with time, after reflection from the boundary; while exact solution became zero on the boundary, according to the boundary condition applied. But as soon as we added PML, and expanded the problem to the complex plane enormous, frightening instabilities appeared when wave packet reached the point of PML.

One of the ideas why this instability occurred, was the actual magnitude of our *Tridiagonal Matrix's* eigenvalues, because there were not meant to be bigger than 1. But after we had expanded equation (2) to the complex plane adding PML the elements of a *Tridiagonal Matrix* were not constant any more. When the wave packet reached the point where PML starts all elements of a *Tridiagonal Matrix* started to change, because since that moment wave packet began to move along the curve prescribed by PML, and of course it was supposed to decay, but seems that eigenvalues became bigger than 1 at some point and caused a really terrible instability. Of course I tried to check what eigenvalues we had, but while working on this problem, I realized one more important (as it seems to me) fact (because I never got any answer on it). What I found was the example from the book, on which our method was based, says that eigenvalues of a *Tridiagonal*

Matrix have to be negative in order to give stable results. But eigenvalues of our matrix were certainly positive. Thus, seems that this method was wrong from the very beginning (or maybe my assumption is incorrect), but don't think it matters any more. After another half month of calculations and fighting with *MATLAB* I ended up with choosing next approach to try to get stable solution for the problem. And this time it will work out!

4.3 Applying PML to the reduced Schrödinger equation

4.3.1 Something that almost worked

After all the fails[14], again we choose a new numerical method to solve a reduced *Schrödinger* equation. This method is actually a combination of two different approaches, and based on two main steps, which are quite simple in their essence:

1) We apply *Finite Difference Method (FDM)* to the spatial (right) part of the eq. (2);

2) For the partial time derivative ∂_t, on the left of the eq. (2), we apply one of the implemented ODE-solvers (from *MATLAB*).

Maybe, there was a third step as well, which should sound like *"and see how it works!"*, at least that what is written in my A4 notebook. But as I already told, it worked out *almost* nicely. I say "almost", because to run a few steps forward, it's actually not the final method that we are going to use, there will be one brilliant and important addition or change in the approach but we, probably, talk about it later (Chapter 4.3.2). As concerns this method, as I told before, it worked almost nice, but there were some small reflections all the time, very tiny, but still. That was the only reason to improve it.

Anyhow, I think it is worth to tell about the application of this method, because it seems to be the first very big step in the right direction, which helped to see what next step would be about! Of course, I was not moving smoothly and easily trying to carry it out: I was fighting again with stupid mistakes in calculation (like having the wrong sign somewhere for more then one week); I was improving slowly my programming skills, one of my codes took more than an hour to run and I was naive to believe that it's the equation (not my coding) that was so tough to solve, and got terrified with the idea what it would be like later, with the bigger equations and more dimensions. There were lots of hard work, disappointments and infusions, ups and downs, and definitely, I am sure, it all went the way it should. So in the next few paragraphs I will try to show step by step what we have got.

What we want to do, is to expand a reduced *Schrödinger* equation to the complex plane and apply PML. First of all, let's remind ourselves how our equation

[14]But I cannot say that these fails and bunch of mistakes I did, were useless or sad or maybe stupid. Quite the contrary, it was probably the best way to learn and to understand how things do actually work.

looks like, here it is

$$i\frac{\partial U}{\partial t} = -\frac{\partial^2 U}{\partial x^2}, \tag{8}$$

and in my notes, at one moment, it suddenly changes to

$$i\frac{\partial U}{\partial t} = -\frac{1}{2}\frac{\partial^2 U}{\partial x^2}. \tag{9}$$

It is not a big deal, having a $1/2$ constant factor, so we will just continue working on this eq. (9) now. And the thing we start with, is moving from our *real* plane to a *complex* plane:

$$x \quad \longrightarrow \quad z = x + if(x), \tag{10}$$

where $f(x)$ will be a function that describes PML behaviour (Fig. 13). It exists on the *complex* plane as you can see, and actually that's the trick of applying PML. It is some kind of an absorbing layer but it is on a *complex* plane.

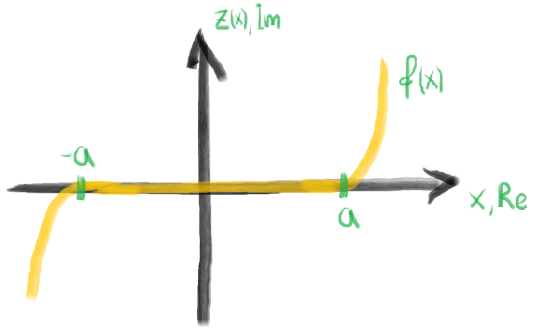


Figure 13: PML on a complex plain. PML starts at points $-a$ and a, and behaves according to the function $f(x)$, which is usually some parabolic function. In between points $-a$ and a there is no PML and $z(x) = x$.

Thereby, what we want to do, is to expand the problem we have to the *complex* plane and make it go not along the x-axis only (as it was before), but along *complex* z-axis which is defined by next values:

$$z = \begin{cases} a + if(x), & x \geqslant a \\ x, & -a < x < a \\ -a + if(x), & x \leqslant -a \end{cases}, \tag{11}$$

where a and $-a$, are the points where PML starts. So you can see that in the region where there is no PML $z = x$, and our system should behave as it did before, but as soon as it reaches point a or $-a$, it doesn't go along x-axis any more, instead it goes along axis: $z = a + if(x)$ or $z = -a + if(x)$ (depending on the direction) – thus, it will never reach the boundary (L or $-L$), and never reflect back from it, if we choose function $f(x)$ to be a PML which known to be a reflectionless absorbing layer. In my first "experiments" I usually chose $f(x)$ as some parabolic curve like

$$f(x) = C(x - a)^p, \tag{12}$$

where C is some constant (choose $C = 1$) and p is the power, I tried to make it greater then 2 usually. I guess, it was made this way to obtain a smoothly varying transition from environment without PML to the PML layer, to be sure that z is a smooth curve. And actually that was still the way I applied it for the method we are discussing now (based on *FDM* and built-in ODE-solver).

Next step of expanding to the *complex* plane is to define a new **complex function**, let's call it $\Phi\big(z(x),t\big)$. Using this new function, we can write down an equation analogous to (9), which will look like

$$i\frac{\partial\Phi\big(z(x),t\big)}{\partial t} = -\frac{1}{2}\frac{\partial^2\Phi\big(z(x),t\big)}{\partial z^2}. \tag{13}$$

Then the most important thing to do comes out, we claim that along the curve $z(x)$,

$$\Phi\big(z(x),t\big)\Big|_{z(x)} = \varphi(x,t). \tag{14}$$

As I understand, this 'trick' is the one that let us combine $U(x,t)$ function from (9) with PML (11), which behaves according to $f(x)$ (12). It let us find the solution following the curve $z(x)$. Thus, function $\varphi(x,t)$ is actually the solution we are looking for! That's very important to understand. It contains everything, and represents how the function (equation) we are interested in, goes along the plane and gets absorbed by the PML when reaching special points (a and $-a$). I would say we have come as close as never to the moment when we apply PML to the system, in a fact, we just did it, there are only few more calculating steps left to be figured out.

Using the condition from (14) we try to calculate all the derivatives for the function $\varphi(x,t)$, we are interested in, so as to create an equation for it, similar to (9) and its expanded form (13). As it goes,

$$\partial_x\varphi(x,t) = \partial_x\Big(\Phi\big(z(x),t\big)\Big|_{z(x)}\Big) = z'\partial_z\Phi\big(z(x),t\big). \tag{15}$$

If we try to find second derivative we will get

$$\begin{aligned}\partial_{xx}\varphi(x,t) =& \partial_x\Big(z'\partial_z\Phi\big(z(x),t\big)\Big|_{z(x)}\Big) = \\ &= z''\partial_z\Phi\big(z(x),t\big) + z'^2\partial_{zz}\Phi\big(z(x),t\big).\end{aligned} \tag{16}$$

From (15) we can find, that

$$\partial_z\Phi\big(z(x),t\big) = \frac{1}{z'}\partial_x\varphi(x,t), \tag{17}$$

and if we insert it into the first term on the right hand-side of (16) we will get

$$\partial_{xx}\varphi(x,t) = \frac{z''}{z'}\partial_x\varphi(x,t) + z'^2\partial_{zz}\Phi\big(z(x),t\big). \tag{18}$$

From here we can express $\partial_{zz}\Phi\big(z(x),t\big)$, and insert it into (13) afterwards, taking into account that nothing is changing for the time-dependence and everywhere we will have

$$\partial_t\Phi\big(z(x),t\big) = \partial_t\varphi(x,t). \tag{19}$$

Thus, after substitution we will get

$$i\partial_t\varphi(x,t) = -\frac{1}{2z'^2}\Big(\partial_{xx}\varphi(x,t) - \frac{z''}{z'}\partial_x\varphi(x,t)\Big). \tag{20}$$

As soon as we have chosen $f(x)$ (12), we will know all the values of $z(x)$ (11). It means, we can easily calculate all the derivatives needed (z' and z''). As concerns to the derivatives $\partial_x\varphi(x,t)$ and $\partial_{xx}\varphi(x,t)$ here comes the time to apply *Finite Difference Method* and discretize our problem.

When discretizing we have to do next things: we choose our boundaries to be $L = N \cdot dx$ and $-L = L$, where we set N - number of points we need for calculation (but from $-L$ to L it will take $n = 2N + 1$ points, because we have to include 0 as well), and dx is a spatial step-size ($dx \ll L$). But as soon as we want to have a PML that starts at some points $-a$ and a, it is important to be sure, that these two points are exactly on the grid, thus we just choose their values to be something in between $-L$ and L ($0 < a < L$), and the number of points we want for calculation $M < N$, and then we set the spatial step as $dx = a/M$. Now we can easily make a grid[15] for x, evaluating it as

$$x = \frac{2L}{2N+1}. \tag{21}$$

After discretizing we have values

$$x_j,\ j = 1, \ldots, n. \tag{22}$$

We do the same for the time to create a time-grid and obtain

$$t_k,\ k = 1, \ldots, m, \tag{23}$$

where $m = T/dt$ is the number of time steps, T is the whole period of time (and we start from 0, of course), dt is the size of a time-step. Now we are ready to apply the *Finite Difference Method* [5] to the spatial derivatives $\partial_x\varphi(x,t)$ and $\partial_{xx}\varphi(x,t)$ (remember that we do it only for the spatial part of equation!), for the single and double derivatives we get:

$$\begin{aligned} \partial_x\varphi &= \frac{\varphi_{j+1} - \varphi_j}{dx}, \\ \partial_{xx}\varphi &= \frac{\varphi_{j+1} - 2\varphi_j + \varphi_{j-1}}{dx^2}. \end{aligned} \tag{24}$$

If we insert it now into (20), and multiply by $-i$ from both sides, we will have

$$\partial_t\varphi_j = \frac{i}{2z'^2}\left(\frac{\varphi_{j+1} - 2\varphi_j + \varphi_{j-1}}{dx^2} - \frac{z''}{z'}\frac{\varphi_{j+1} - \varphi_j}{dx}\right), \tag{25}$$

[15] but in *MATLAB* I use command *'linspace(-L,L,n)'*.

and that is the way we use it in *MATLAB* code to get a numerical solution. Of course, to make *FDM* work we need to know the boundary conditions: $\varphi_{j=1}$ and $\varphi_{j=n}$, which we intentionally set to zero, and we need to choose an initial function at the moment of time $t = 0$. It is not hard to do, actually, we just choose any function $\varphi_0 = \varphi(x_j, 0)$, that could be a solution for eq. (25), and here we have stopped on a mix of a plane wave (5) and Gaussian function , which gives us next result:

$$\varphi_0 = e^{-\gamma(x-x_0)^P} e^{ikx}, \tag{26}$$

where γ is an arbitrary constant which controls the width of the wave packet, x_0 defines the position (the center) of the wave packet on the x-axis, the greater the power P the shaper is the slope (original Gaussian is parabolic, has $P = 2$). Thus, after we have chosen our initial and boundary conditions

$$\begin{aligned} \varphi(x, 0) &= \varphi_0 = e^{-\gamma(x-x_0)^P} e^{ikx}, \\ \varphi(-L, t) &= \varphi_{j=1} = 0, \\ \varphi(L, t) &= \varphi_{j=n} = 0, \end{aligned} \tag{27}$$

we can apply the *FDM* and calculate the right part of eq. (25). As concerns the time derivative on the left hand-side, we use a built-in ODE-solver. I have chosen the one called *'ode23t'* ("for moderately stiff problems if you need a solution without numerical damping") or *'ode23tb'* ("if using crude error tolerances to solve stiff systems") [20]. I found these two solvers to be the fastest and more precise.

The program works quite fast and gives us result in less then 10 seconds, but as I have already told, unfortunately with this method (even though it is very nice) we get some reflection, and its amount was considered to be too huge for the solution of our problem. You can see the plots on (Fig. 14) and observe how the wave packet is moving with time, gets absorbed by the PML at point $-a = -3$, but it's not absolutely absorbed, and reflects back, and even reaches PML on the other side at point $a = 3$ now, where after some time it gets totally absorbed. But our purpose is exactly in getting reflectionless boundaries, or being more correct, reflectionless PML layer. Thus, we are not completely satisfied with this code and approach. Nevertheless, I think half of the work is already done, regarding this exact problem, because the 'trick' we are about to use, to get rid of reflections, is only a little change in the method we have just used.

4.3.2 Something that worked

We have seen that the *Finite Difference Method* has worked almost fine, but still we have got some reflection from the PML (Fig. 14). Thus, we need to do something about it, we cannot change the built-in ODE-solver, but we can find another way, maybe a more precise way, of calculating numerically the spatial derivatives. And I don't really understand how the mind should work to find this kind of answers, is it based on an experience or was it a spark of an inspiration. Or maybe some things so obvious and clear to somebody could be absolutely far beyond the understanding of another. And still it seems to be not so much

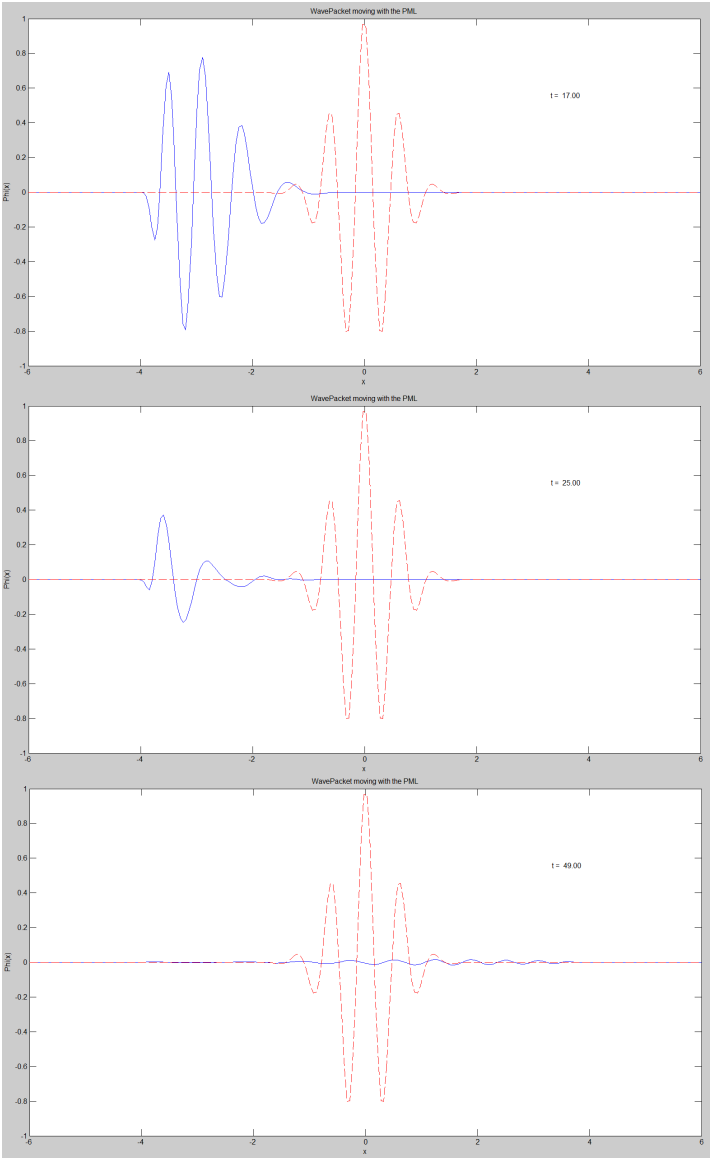


Figure 14: A wave packet (blue solid line) formed by $\varphi_0 = e^{-2(x-0)^2} e^{i(-10)x}$ and zero boundary conditions, calculated through *FDM*, is moving along the x-axis. PML starts at $a = 3$ and $-a = -3$, and the wave packet is moving to the left, because the wave number is negative, $k = -10$. The absorption starts at $-a$, but since the numerical method doesn't work perfectly we get small but impermissible reflections. The red dashed line is the initial condition φ_0, which have been used to compare the correctness of obtained results.

about understanding, but about creating. However, these two terms as if they are inseparable.

The idea of what we are doing now is next: we still apply built-in ODE-solver from *MATLAB*, but instead of *FDM* we do *Taylor Series* (or *Taylor's Expansion*) for the spatial part. And one more thing, we do it for the *complex* function $\Phi\big(z(x), t\big)$. Only after we Taylor-expand it we will go back (or more likely forward) to our desired function $\varphi(x, t)$. First of all, how does *Taylor's Expansion* look like? Basically, it is a representation of a function as an infinite sum of

terms that are calculated from the values of the function's derivatives at a single point,

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f'''(a)}{3!}(x-a)^3 + \ldots = \\ = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a). \tag{28}$$

We use the same way of discretizing the problem as we did in previous Chapter 4.3.1, but now we use it also for our *complex* function $\Phi\big(z(x),t\big)$, which will look like $\Phi(z(x_j),t) = \Phi(z_j,t)$, I assume that time is also discretized (23), it is just that we don't do *Taylor's Expansion* for the time part. Now if we Taylor-expand the function $\Phi(z_j,t)$ around the single point z_0 we will get

$$\Phi(z_j,t) = \Phi(z_0,t) + \Phi'(z_0,t)(z_j - z_0) + \frac{\Phi''(z_0,t)}{2}(z_j - z_0)^2 + \ldots \ . \tag{29}$$

Or we can write it even simpler as

$$\Phi(z_j,t) = a_0 + a_1(z_j - z_0) + a_2(z_j - z_0)^2 + \ldots \ , \tag{30}$$

where 'constants' (actually they are not): a_0, a_1 and a_2 – represent the spatial derivative of $\Phi(z,t)$ divided by the factorial of its (derivative's) power. And actually, we can choose as many steps q in *Taylor's Expansion* as we want, depending on a precision we are interested in, for a_q-coefficient we will have

$$a_q = \frac{\Phi^{(q)}(z,t)}{q!}. \tag{31}$$

We already know that if we follow the curve $z(x)$, then according to (14) we can come to the function $\varphi(x,t)$. Considering this, let's choose our $z_0 = z_j$ and expand those $\Phi(z_j,t)$ using (29) and (30), that are numbered with j, $j+1$ and $j-1$, what we get is

$$\begin{cases} \varphi_j = \Phi(z_j,t)\Big|_{z(x)} = a_0 \\ \varphi_{j+1} = \Phi(z_{j+1},t)\Big|_{z(x)} = a_0 + a_1(z_{j+1} - z_j) + a_2(z_{j+1} - z_j)^2 \\ \varphi_{j-1} = \Phi(z_{j-1},t)\Big|_{z(x)} = a_0 + a_1(z_{j-1} - z_j) + a_2(z_{j-1} - z_j)^2. \end{cases} \tag{32}$$

If we take now the double derivative $\partial_{zz}\Phi(z,t)$ we will see, that according to our expansion it is always

$$\partial_{zz}\Phi(z,t) = 2a_2, \tag{33}$$

even when we have more terms (when $q > 2$), because all the other terms will go to zero due to the differences $(z_j - z_j)$ in the brackets. But let's remember our reduced *Schrödinger* equation (13), write it down again

$$i\partial_t\Phi(z,t) = -\frac{1}{2}\partial_{zz}\Phi(z,t). \tag{34}$$

What we see on the right hand-side is exactly what we have just found, so we can rewrite this equation using (33) and also (14), to get a very beautiful result

$$\begin{gathered} i\partial_t\varphi(x,t) = -\frac{1}{2}\cdot 2a_2 \\ \Downarrow \\ \partial_t\varphi(x,t) = ia_2. \end{gathered} \tag{35}$$

And the most amazing fact is that from the system of equations (32) we can easily define our coefficient a_2 as a function of $(z_j, z_{j+1}, z_{j-1}, \varphi_j, \varphi_{j+1}, \varphi_{j-1})$. Because what we have in (32) is a system of 3 equations with 3 unknowns: a_0, a_1 and a_2 – which is obviously solvable.

In case if we want to add more terms, for better precision, say going to $q = 4$ our system will take the form

$$\begin{cases} \varphi_j = a_0 \\ \varphi_{j+1} = a_0 + a_1(z_{j+1} - z_j) + a_2(z_{j+1} - z_j)^2 + a_3(z_{j+1} - z_j)^3 + a_4(z_{j+1} - z_j)^4 \\ \varphi_{j-1} = a_0 + a_1(z_{j-1} - z_j) + a_2(z_{j-1} - z_j)^2 + a_3(z_{j-1} - z_j)^3 + a_4(z_{j-1} - z_j)^4 \\ \varphi_{j+2} = a_0 + a_1(z_{j+2} - z_j) + a_2(z_{j+2} - z_j)^2 + a_3(z_{j+2} - z_j)^3 + a_4(z_{j+2} - z_j)^4 \\ \varphi_{j-2} = a_0 + a_1(z_{j-2} - z_j) + a_2(z_{j-2} - z_j)^2 + a_3(z_{j-2} - z_j)^3 + a_4(z_{j-2} - z_j)^4. \end{cases} \tag{36}$$

Now it's a system of 5 equations with 5 unknowns and it means that in this case we can again express a_2 through all the values of z and φ. But yet we will stop on $q = 2$ and the nice equation (35) we have obtained. To express $a_2 = f(z_j, \varphi_j)$ I solved the system (32) in *Mathematica*, here is what we get

$$a_2 = -\frac{-z_j\varphi_{j-1} + z_{j+1}\varphi_{j-1} + z_{j-1}\varphi_j - z_{j+1}\varphi_j - z_{j-1}\varphi_{j+1} + z_{j-1}\varphi_{j+1}}{(z_{j-1} - z_j)(z_{j-1} - z_{j+1})(z_j - z_{j+1})}. \tag{37}$$

And this 'huge' but so elegant expression we use on the right-hand of (35) instead of using *Finite Difference Method*[16] that we used before (Chapter 4.3.1). For the left hand-side we still use the same built-in ODE-solver from *MATLAB*.

One more difference from the previous approach is another choice for the PML behaviour. Remember, that the PML is defined by a function $f(x)$, so this time we go really "extreme", but still very realistic and even more clear. We choose our PML to look like a straight vertical line (Fig. 15). As soon as we have it on a *complex* plane we set the complex part of z in the next form

$$if(x) = e^{i\theta}(x - a), \tag{38}$$

where θ is the parameter we use to change the inclination of the PML, in order to find the best one with less reflection, because even though it's supposed to be reflectionless, we have to keep in mind, and I have already mentioned it before (in the end of Chapter 4.1), that since we have a numerical way of solving problem, we will always get some small reflection, which can be reduced only by increasing the precision of calculations.

[16] Actually, this expression (37) is also a form of a *FDM*. It's just expended to a specific *complex* plane according to the PML (11). I didn't really get it from the very beginning, but it becomes more obvious when we go further.

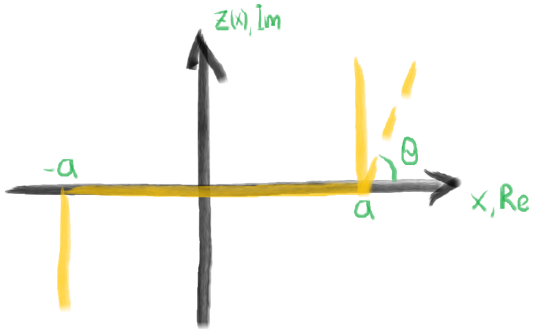


Figure 15: PML which is presented through the function $f(x) = -ie^{i\theta}(x-a)$, with $\theta = \pi/2$. As before, PML starts at the points $-a$ and a. In between the points $-a$ and a there is no PML and $z(x) = x$.

After this 'simplification' we will have a new system of values for $z(x)$, comparing to (11) with (12), now it will take a more pleasant form

$$z = \begin{cases} a + e^{i\theta}(x-a), & x \geqslant a \\ x, & -a < x < a \\ -a + e^{i\theta}(x+a), & x \leqslant -a. \end{cases} \tag{39}$$

And in my calculations I chose $\theta = \pi/2$, that gives $e^{i\frac{\pi}{2}} = i$. Then obviously,

$$z = \begin{cases} a + i(x-a), & x \geqslant a \\ x, & -a < x < a \\ -a + i(x+a), & x \leqslant -a, \end{cases} \tag{40}$$

which makes the PML look exactly like on a (Fig. 15). It could be quite surprising, and probably one would expect a strong reflection from this kind of a 'vertical wall'. But in practice we see, that using this type of a PML plus a method based on: *Taylor's Expansion* for the right hand-side; and built-in ODE-solver for the time derivative on the left hand-side of (34) – we get very nice plots for the wave packet moving along the x-axis. Also it gets (at least visually) totally absorbed by the PML at point a (or $-a$). That is exactly what we were looking for.

The numerical solution for this problem is going to be a first attachment in the *Appendices* section. I include it as a *MATLAB* code *[A]*. The only disadvantage of the code *[A]*, is that it takes 30-40 seconds (quite long time) to run. I found out that almost all this time is actually taken by a built-in ODE-solver (when calculating time derivative), thus, it means whether this is the way it works (without haste) or it is my mistake in representing the data for the solver. I bet on last one, of course. And I tried my best to fix it and put it in different way in the code, but have not succeeded yet. Except taking too much time, I believe that this code works perfectly and on (Fig. 16) you can see some freeze frames of the wave packet travelling from $x_0 = 0$ to the right (k is positive) PML's boundary which starts at $a = 4$.

Figure 16: A wave packet (blue solid line) formed by $\varphi_0 = e^{-\gamma(x-x_0)^P} e^{ikx}$, with $\gamma = 2$, $x_0 = 0$, $P = 2$, $k = 12$, and boundary conditions: $\varphi_{j=1} = \varphi(-L, t)$, $\varphi_{j=n} = \varphi(L, t)$ – set to zero. We can see (reading from left to right and then down) how a wave packet moves to the right ($k = 12$), in accordance with a reduced *Schrödinger* equation (34) applied. Later, when it reaches PML of a form (Fig. 15), that starts at the point $a = 4$, it gets absorbed. The red dashed line represents the exact initial function φ_0 (we used it to check the numerical solution). It is quite hard to notice any reflection on this scale, but on a bigger one, or when reducing the number of steps in calculations, we can observe small reflection. Though, we assume it to be negligible. The numerical approach used (code *[A]*) is based on *Taylor's Expansion* for the spatial derivative and a built-in ODE-solver for the time derivative of (34).

On this, I guess, the description of applying PML to a reduced *Schrödinger* equation could be finished. We see how it works and what it looks like (Fig. 16). Hopefully, it gives us a good basis to move to the next more interesting, but of course, more complicated problem to solve. And now, we will come closer to the *Patterns*, because the equation we are going to observe is exactly the one used to describe a *Rayleigh-Bénard* convection. But at the same time as we have discussed in Chapter 3.1, because of the visual similarities you can apply it to simulate something like *Zebra stripes* or *Sand dunes* (Fig. 4), if you want. And that's what makes it so amazing! Be ready for a *Swift-Hohenberg* equation.

5 The Swift-Hohenberg equation with Periodic Boundaries

In this chapter and its sub-chapters we will go through the steps that has been taken in order to calculate the result given by a one-dimensional *Swift-Hohenberg* equation if we solve it numerically. We will set the *Periodic Boundaries* and make everything behave as it runs on a circle, or a ring. Also there will be a *White noise* introduced as an initial function instead of a usual *Gaussian*. And in the very end we will find the *Fourier Transform* of our problem which will be not as easy as it might sound.

It could be hardly seen here in the text, even between the hundreds of lines of symbols how the time goes or how one place has been changed by another and then by one more. Most likely it looks like it is supposed to look – like a Thesis, a one-piece thing. And I hope it really does, otherwise it would be hard to follow rambling thoughts and conclusions we have to make.

But it still seems strange to me how something you do, could last forever being in the process, in the progress. Of course not *forever*, though at least for really long periods, as one comes to realise sometimes. And while the time passes by, this thing, that you do, will travel along with you. Through all the different amazing places! It will be somewhere on a side, glowing and changing its colors – blinking to you; or it could be just in front of your eyes all the time, but then it's hard in a way not to pay attention to it, and probably it will be just a part of anything that occupies you at the moment, or more likely, it will be this thing. And this time it won't last for a long period. Perhaps, you will try to concentrate on it. Seems, it won't let you go until it's solved in a right or trouble way.

5.1 One-dimensional Swift-Hohenberg equation

It is worth starting with the basics, with something that will be a basis for you later. Something that you can always return to, like a safe-point, when suddenly things go wrong. So let's take a look at the one-dimensional *Swift-Hohenberg* equation. It is usually written in the next form:

$$\partial_t u(x,t) = (r-1)u - 2\partial_x^2 u - \partial_x^4 u - N(u), \tag{41}$$

it is an evolution equation for a single field $u(x,t)$ in a one-dimensional domain described by the coordinate x, where r represents a *control parameter*. What does it mean and what do we want (or need) to control?

It comes, that when we proceed with an evolution equation we will have something that develops in time, and in most of the cases it's just some growth of the field $u(x,t)$ that is the function we are interested in. Usually there is some **critical value** $r = r_c$, that represents when the solution turns from a **stable** one to **unstable**. But we will discuss and see it more clearly when we do a *Stability Analysis* for the equation (Chapter 5.2). Also we are better to rewrite the one-dimensional *Swift-Hohenberg* equation in a more compact form[17], which, as one could see, is much more easier to understand and comfortable to work with:

$$\partial_t u(x,t) = ru - (\partial_x^2 + 1)^2 u - N(u). \tag{42}$$

Here I must mention the importance of the last term $N(u)$, which is a *non-linear term*, and plays a role of a stability parameter. Though we will see it later, but it happens that even if the control parameter $r > r_c$ causes an instability and a never-ending growth of the function, this non-linear term $N(u)$ could stop the growth somehow[18]. Isn't this magic? And to be honest, that's exactly what allows some certain or some unexpected pattern to be formed, if we talk about the evolution equations.

Maybe it's also worth mentioning, that the *Swift-Hohenberg* equation (42) is the one that could be used to describe *patterns* appearing in such a process as a **Rayleigh-Bénard** convection, that we have already slightly studied in the Chapter 3.1. But let us not forget about all the connections and similarities found between different patterns that belong to the absolutely different (at first sight) natural systems. I wish we could keep it in mind, because in my opinion it's one of the things that makes all these equations and theories to have something so huge and amazing behind, hidden in numbers and considerations.

5.2 Linear stability analysis (ordinary case)

I guess, the *linear stability analysis* is one of the first things to do when you consider any evolution equation. Right now we are going to apply this procedure to the *Swift-Hohenberg* equation (42). But let's write it down one more time choosing the non-linear term $N(u) = u^3$ as our *stability parameter*,

$$\partial_t u(x,t) = (r-1)u - 2\partial_x^2 u - \partial_x^4 u - u^3, \tag{43}$$

$$\partial_t u(x,t) = ru - (\partial_x^2 + 1)^2 u - u^3. \tag{44}$$

[17] This form (42) and the previous one (41) were actually obtained by a reduction of a more complicated form of the *Swift-Hohnebrg* equation, applying scaling and the substitution of variables. We did it within the **Special Curriculum** course that concerned a *Pattern Formation*.

[18] The non-linear term $N(u)$ could be presented by a different non-linear function of $u(x,t)$. In our further calculations we will use $N(u) = u^3$.

To carry out the linear stability analysis [1] we denote the solution of $u = 0$ as a base state u_b and ask whether the difference or perturbation field

$$u_p(x,t) = u(x,t) - u_b, \tag{45}$$

between an arbitrary nearby solution $u(x,t)$ and the base state will grow in magnitude over time. The perturbation u_p evolves according to the evolution equation

$$\partial_t u_p = \hat{N}[u_b + u_p] - \hat{N}[u_b], \tag{46}$$

here $\hat{N}$ is the non-linear operator that is defined according to the right side of (43) and is the function of the field u:

$$\hat{N}[u] = (r-1)u - 2\partial_x^2 u - \partial_x^4 u - u^3. \tag{47}$$

But then we can use one feature of the perturbation field u_p. We claim that if it is sufficiently small, then we can approximate $\hat{N}[u_b+u_p]$ in eq. (46) by linearizing about u_b. It means, that we are going to keep only the terms on the right-hand side of the equation that involve a **single factor** of u_p or its spatial derivative. So, keeping it in mind, we try to proceed with eq. (46) using the definition (47) for the operator $\hat{N}[u]$, what we will get is:

$$\begin{aligned} \partial_t u_p = (r-1)(u_b+u_p) - 2\partial_x^2(u_b+u_p) - \partial_x^4(u_b+u_p) + (u_b+u_p)^3 - \\ - (r-1)u_b - 2\partial_x^2 u_b - \partial_x^4 u_b + u_b^3 = \\ = (r-1)u_p - 2\partial_x^2 u_p - \partial_x^4 u_p + 3u_p u_b^2. \end{aligned} \tag{48}$$

So if we rewrite it taking u_p out of the brackets, we will get

$$\partial_t u_p = (r - 1 - 2\partial_x^2 - \partial_x^4 + 3u_b^2)u_p. \tag{49}$$

But let us not forget that we specialize it to a particular base state which is $u_b = 0$, thus the term $3u_b^2$ in (49) will vanish,

$$\partial_t u_p = (r - 1 - 2\partial_x^2 - \partial_x^4)u_p. \tag{50}$$

Notice that the equation is linear in the field u_p (for example, multiplying up by a constant factor leaves the equation unchanged). So what we have gotten is a linear differential equation with constant coefficients, and the coefficients are constant precisely, because the base state is stationary and uniform. To solve a linear constant-coefficient ODE we can choose a particular solution to eq. (50) that depends exponentially on time and exponentially on space

$$u_p(x,t) = Ae^{\sigma t}e^{\alpha x}, \tag{51}$$

Here the constant σ is called a *growth rate*, and both σ and the constant α are possibly complex. As an analogy that was made for eqs. (41) and (42) we can rewrite eq. (50) as

$$\partial_t u_p = r u_p - (\partial_x^2 + 1)^2 u_p. \tag{52}$$

Now, if we insert (51) into eq. (52) we will get next equation for the *growth rate*,

$$\sigma = r - (\alpha^2 + 1)^2 \tag{53}$$

The meaning of the constant α can be deduced by considering the boundary conditions that apply to the field u and to the perturbation u_p [1]. The simplest possible cases are the idealized geometries for which the lateral boundaries are eliminated by using *infinite* or *periodic boundaries*. And here we are approaching one of the cases that we will look at more carefully later and that is studied well nowadays. I am talking about the case when the lateral boundaries are eliminated by *periodic boundaries*. As concerns using *infinite* boundaries, it only makes sense if we consider the problem analytically. On practice, when we have to use numerical methods to find the result, there is no chance unfortunately to make infinite calculations. Even though computer technologies has gone quite far.

But let's return to the periodic boundaries that we want to use. In this case we can assume that the system is finite but periodic with length L, i.e. the system is topologically equivalent to a ring. A constant solution u_b is automatically periodic over any length but a perturbation $u_p(x,t)$ is periodic with period L only, one can write

$$u_p(x,t) = u_p(x+L,t), \tag{54}$$

for all times t and all positions x. At the same time if we insert (51) in a latter equation we will see, that the eq. (51) will be periodic with period L if and only if

$$e^{\alpha x} = e^{\alpha(x+L)} \quad \text{for all } x. \tag{55}$$

And exactly from here we get a very important result, we find that it implies that

$$e^{\alpha L} = 1 \quad \text{or} \quad \alpha L = (2\pi i)m, \tag{56}$$

for some integer m. And thus we find that $\alpha = iq$ for

$$q = m\left(\frac{2\pi}{L}\right), \quad m = 0, \pm 1, \pm 2, \ldots \tag{57}$$

Now we can conclude that the form of the mode (51) actually looks like

$$u_p(x,t) = Ae^{\sigma t}e^{iqx}, \tag{58}$$

with q a real number. The spatial dependence of u_p is then periodic with wave number q (alternatively, with wavelength $\lambda = 2\pi/q$). Substituting $\alpha = iq$ into eq. (53), will give us

$$\sigma = r - (q^2 - 1)^2. \tag{59}$$

This equation says us that a small-amplitude spatially periodic perturbation with a wave number q (about the base solution $u_b = 0$) will grow or decay exponentially in time with a growth rate σ_q that depends on q. Let us examine more carefully now what does exactly happens when we talk about the dependence of a growth rate σ_q on a wave-number q. We want to determine when the maximum of the curve $\mathrm{Re}\,\sigma_q$ vs. q changes from a negative to positive value as the

parameter r is varied. Because it will indicate the onset of **linear instability**. It is easy to see that the quantity $(q^2-1)^2$ is a non-negative one and vanishes when $q=1$, thus, the maximum for σ_q occurs when $q_{\max}=1$. And it happens independently of r. The value of σ_q at its maximum is therefore

$$\max_q \operatorname{Re}\sigma_q = r. \tag{60}$$

Also, as one can see from eq. (58), the uniform base state $u_b=0$ will be linearly stable if the exponential decays in the limit $t\to\infty$, well, this condition is true when

$$\max_q \operatorname{Re}\sigma_q < 0, \tag{61}$$

which means that the values of r should be negative if we look for the stable solution. To analyse it one more time, what have we discovered? We have found that the uniform state $u=0$ will be linearly stable if the control parameter in a *Swift-Hohenberg* equation $r<0$ and it turns unstable when we let $r>0$. Therefore, we can write down a **critical** control parameter value for the linear instability is $r_c=0$. Second thing, is the critical wave number q_c at which the curve $\operatorname{Re}\sigma_q$ first attains a positive value is $q_c=1$, since this is the location of the maximum, independent of the value of r. But I believe it is much easier to see and to analyse it with the figure. So, I will let myself steal one from [1], instead of drawing it.

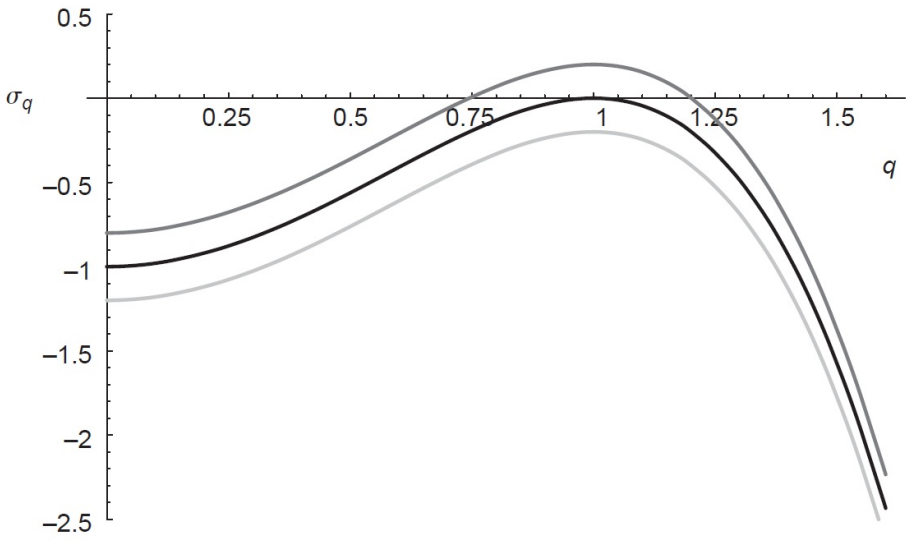


Figure 17: We see the plot based on the eq. (59) of the growth rate σ_q versus the wave number q for the uniform base state $u=0$ of the one-dimensional *Swift-Hohenberg* equation (41). Three curves are shown for parameter values of respectively $r=-0.2$ (**light gray**), $r=0$ (**black**), and $r=0.2$ (**dark gray**). These correspond to a **stable**, **marginally unstable**, and **unstable** base state. The critical parameter value $r_c=0$ and the critical wave number $q_c=1$ are identified from the $r=0$ curve as where $\operatorname{Re}\sigma_q$ first becomes zero as r is varied. For $r=0.2$ just positive, only a narrowband of *Fourier modes* centred on the critical wave number can grow, predicting the appearance of a pattern with a characteristic length scale $\frac{2\pi}{q_c}$.

In the end of this chapter I just want to pay attention one more time on the last sentence written in the caption for (Fig. 17). It says that for positive values of the control parameter $r > 0$ there will be a narrowband of *Fourier modes* centred on the critical wave number $q_c = 1$. And those are the only *Fourier modes* that will grow, moreover this process is actually predicts the appearance of a pattern with a characteristic length scale $\frac{2\pi}{q_c}$. We will see it as clear as it sounds in the Chapter 5.4, where we will take a chance to find a Fourier transform for our problem. Nevertheless, first we have to understand how a one-dimensional *Swift-Hohenberg* equation could be solved (numerically) when the *Periodic boundaries* are set.

5.3 Periodic Boundaries for the Swift-Hohenberg

Applying periodic boundaries to a one-dimensional *Swift-Hohenberg* equation is not something new[19]. But we do it because it gives us nice possibilities for the analysis of a problem as well as it is another good chance to check some parts of the **code** we have for solving the equation.

Perhaps, here I have to start exactly with a code, to be more precise, with the changes that have been made. These changes, they didn't changed anything crucially about the method we use, but they have definitely simplified the calculations and at the same time they have brought a lot to the understanding and to the analysing of a problem. I also think that we have done something similar before, and I accept that it could be that I got or did something wrong last time. But that's how it goes, and maybe it is not so bad, even funny a bit, to come back to a problem to return to a method that you considered wrong some months ago (feels like going through the history somehow). But I would claim it is all right when occurs on short periods of time only. Otherwise there would be too much to redo.

In Chapter 4.2, or to be more exact on the page *22*, we discussed the possibility of using a *Tridiagonal Matrix* for the right-hand side of the equation, which was actually a *finite difference* to represent the second order derivative. Most likely, there was some mistake in calculation (in my code) or maybe the way I was applying the method was not very right. But I don't think there is any reason to dig deep now, I just decided to redo it again, because obviously there was a point doing it.

In a certain sense it's just a way of representing the data that we have, so it shouldn't really be a problem. But it might be a bit more difficult now, since the equation we consider (43) has second order and fourth order derivatives, which means that we will have more terms in a *finite difference* method and the matrix will have more diagonals, at least five. Thus in the next few paragraphs I would

[19]Maybe, sometimes there is something that has been done many times before, done by the others. But still it doesn't mean it's not of any use for you to repeat and to see how it goes, especially when it comes to studies. It could be even better, if you try to do it yourself and then compare the result. Because, it is almost always that there are dozens of ways to do something, and it happens quite often that you cannot be sure which method will work the best and which one will bring you success or trouble.

like to go carefully through this, because the result that was obtained, did really simplify a lot of things afterwards and looks nice.

We consider a one-dimensional *Swift-Hohenberg* equation, let's write it down one more time,

$$\partial_t u(x,t) = (r-1)u - 2\partial_x^2 u - \partial_x^4 u - u^3. \tag{62}$$

To represent its spatial derivatives on the right-hand side we use the same method that has been successful for the *Schrödinger* equation (Chapter 4.3.2). We use the *Taylor Expansion* (28), and this time we don't need to apply **PML**, because of the **Periodic Boundaries**, thus we don't have to make any extension to the *complex plane*. But I would still prefer to use function $\varphi(x,t)$ in our notes instead of $u(x,t)$, so we just let $u(x,t) \to \varphi(x,t)$ in eq. (62). To consider a fourth order derivative with a *finite difference method*, we have to take at least 5 points on the grid, thus applying the *Taylor Expansion* one can obtain next system of equations

$$\begin{cases} \varphi_j = a_0 \\ \varphi_{j+1} = a_0 + a_1(x_{j+1}-x_j) + a_2(x_{j+1}-x_j)^2 + a_3(x_{j+1}-x_j)^3 + a_4(x_{j+1}-x_j)^4 \\ \varphi_{j-1} = a_0 + a_1(x_{j-1}-x_j) + a_2(x_{j-1}-x_j)^2 + a_3(x_{j-1}-x_j)^3 + a_4(x_{j-1}-x_j)^4 \\ \varphi_{j+2} = a_0 + a_1(x_{j+2}-x_j) + a_2(x_{j+2}-x_j)^2 + a_3(x_{j+2}-x_j)^3 + a_4(x_{j+2}-x_j)^4 \\ \varphi_{j-2} = a_0 + a_1(x_{j-2}-x_j) + a_2(x_{j-2}-x_j)^2 + a_3(x_{j-2}-x_j)^3 + a_4(x_{j-2}-x_j)^4. \end{cases} \tag{63}$$

Of course, to simplify it, first of all we have to rewrite all the terms such as $x_{j+1} - x_j = dx$, and $x_{j-2} - x_j = -2dx$, and etc[20]. It is not hard to do manually, but it will reduce the numerical calculations a lot[21]. So we rewrite the system using this feature. Also I suggest starting to write it from the further left point which is always x_{j-2} and moving to the right one x_{j+2}.

$$\begin{cases} \varphi_{j-2} = a_0 + a_1(-2dx) + a_2(-2dx)^2 + a_3(-2dx)^3 + a_4(-2dx)^4. \\ \varphi_{j-1} = a_0 + a_1(-dx) + a_2(-dx)^2 + a_3(-dx)^3 + a_4(-dx)^4 \\ \varphi_j = a_0 \\ \varphi_{j+1} = a_0 + a_1(dx) + a_2(dx)^2 + a_3(dx)^3 + a_4(dx)^4 \\ \varphi_{j+2} = a_0 + a_1(2dx) + a_2(2dx)^2 + a_3(2dx)^3 + a_4(2dx)^4 \end{cases} \tag{64}$$

From this system we can get formulas to calculate all the a_k-terms, but we do need only a_2 and a_4. According to the *Taylor Expansion* (28), they represent second and fourth order spatial derivatives of the functions φ_j that we need so much for our *Swift-Hohenberg* equation (62). Actually, what we obtain when we introduce coefficients a_2 and a_4 through the function $\varphi_j = \varphi(x_j, t)$ and dx is

[20] Here dx is a spatial step that we set when discretizing the space. We set the desired length L of the section and then choose the number of points n it should be split on, thus we calculate $dx = L/n$. It helps us to obtain the grid presented by x_j, where $j = 1, 2, \ldots, n$.

[21] By the way, that was something I forgot to do when building the matrix for the *Schrödinger* equation. That's why it took so long time for the code *[A]* to run. But I am happy to learn on my own mistakes and to see how they come to be solved out in the end.

called the *finite difference*. I have written a code *[B]* in *MATLAB* that solves a system of equations such as (63). Thus, the terms could be calculated every then and now, and later when it will come to the PML and the complex coefficients. For example, the result for a_2 will take form

$$a_2 = \left(-\frac{1}{24}\varphi_{j-2} + \frac{2}{3}\varphi_{j-1} - \frac{5}{4}\varphi_j + \frac{2}{3}\varphi_{j+1} - \frac{1}{24}\varphi_{j+2} \right) \frac{1}{dx^2}, \tag{65}$$

and if we want to proceed to find the derivative itself, according to eq. (28) we just have to multiply the result by 2! for the second order derivative and by 4! for the fourth order. Relying on this conclusion and the eq. (65) we can introduce the matrix which will let us to take the vector φ_j out of the brackets, while the diagonals of the matrix are built from the coefficients standing in front of every certain φ_j. But have to notice here, that it would be correct for all the points x_j except two first and two last points, since we want to set boundaries that are periodic, the rule will look differently for those 'special' points. Anyway, for all the others points we will have

$$\frac{\partial^2}{\partial x^2}(\varphi_j) = \frac{2!}{dx^2} \begin{pmatrix} \ddots & \ddots & \ddots & & & & & & \\ \ddots & \ddots & \ddots & \ddots & & & & & \\ & -\frac{1}{24} & \frac{2}{3} & -\frac{5}{4} & \frac{2}{3} & -\frac{1}{24} & & & \\ & & -\frac{1}{24} & \frac{2}{3} & -\frac{5}{4} & \frac{2}{3} & -\frac{1}{24} & & \\ & & & -\frac{1}{24} & \frac{2}{3} & -\frac{5}{4} & \frac{2}{3} & -\frac{1}{24} & \\ & & & & & \ddots & \ddots & \ddots & \ddots \\ & & & & & & \ddots & \ddots & \ddots \end{pmatrix} \varphi_j. \tag{66}$$

Let us try to find the rule to set down the value for the 'special' points: x_1, x_2 and x_{n-1}, x_n – where n is the total number of points on the spatial grid. When we want to apply the *periodic boundaries* one has to build kind of a circle on the plane. According to the 4^{th} order method we use, we need 4 neighbouring points to evaluate the value at some point x_j – two from the left (x_{j-2} and x_{j-1}) and two from the right (x_{j+1} and x_{j+2}). But using periodic boundaries it should be obvious that when we approach the points like x_{n-1} and x_n we will just use points from the very beginning x_1 and x_2 to set their values and operate vice-versa afterwards when being on the other 'end' (Fig. 18).

Figure 18: Applying *Periodic boundaries* on the plane: to calculate the value of a function at the point x_n (for the fourth order accuracy) we use two points from the left x_{n-2} and x_{n-1}, but then to make the right boundary *periodic* 'a ring has to be closed', thus we use the two first points x_1 and x_2.

Thus, after some calculations we can add first two and last two rows to the matrix (66), now it will look like

$$M_2 = \frac{1}{dx^2}\begin{pmatrix} -\frac{5}{4} & \frac{2}{3} & -\frac{1}{24} & \cdots & & & \cdots & -\frac{1}{24} & \frac{2}{3} \\ \frac{2}{3} & -\frac{5}{4} & \frac{2}{3} & -\frac{1}{24} & \cdots & & & \cdots & -\frac{1}{24} \\ -\frac{1}{24} & \frac{2}{3} & -\frac{5}{4} & \frac{2}{3} & -\frac{1}{24} & & & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & & & \\ & & -\frac{1}{24} & \frac{2}{3} & -\frac{5}{4} & \frac{2}{3} & -\frac{1}{24} & & \\ & & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & & -\frac{1}{24} & \frac{2}{3} & -\frac{5}{4} & \frac{2}{3} & -\frac{1}{24} \\ -\frac{1}{24} & \cdots & & & \cdots & -\frac{1}{24} & \frac{2}{3} & -\frac{5}{4} & \frac{2}{3} \\ \frac{2}{3} & -\frac{1}{24} & \cdots & & & \cdots & -\frac{1}{24} & \frac{2}{3} & -\frac{5}{4} \end{pmatrix}, \tag{67}$$

where in the middle we will still have those unchanged diagonals like before. Then we build the same kind of a matrix but for the 4^{th} order derivative which could be represented through the a_4 coefficient as

$$\frac{\partial^4}{\partial x^4}(\varphi_j) = 4! \cdot a_4, \tag{68}$$

and the a_4 coefficient has to be found from solving a system (64). The finite difference rule[22] for the a_4 coefficient will look like

$$a_4 = \left(\frac{1}{24}\varphi_{j-2} - \frac{1}{6}\varphi_{j-1} + \frac{1}{4}\varphi_j - \frac{1}{6}\varphi_{j+1}\frac{1}{24}\varphi_{j+2}\right)\frac{1}{dx^4}. \tag{69}$$

By the way I have checked, and the obtained results shown in eqs. (65) and (69) do match with the results for the finite difference coefficients from [21]. But it doesn't mean in any way that we have made the code *[B]* for no actual reason. It will be a lot of help for us, when later we will have to consider a problem with PML. Where the extension to the complex plane will take place. Thus, nothing is in vain.

Going back to the 4^{th} order derivative, again we prefer to present the result in the **matrix** form. In the same manner as we have done before for the second derivative and a_2, we proceed now for the 4^{th} derivative which is calculated with

[22] According to my *MATLAB* code *[B]* that is made to solve such a system of equations as (64).

a 2^{nd} order accuracy,

$$\frac{\partial^4}{\partial x^4}(\varphi_j) = \frac{4!}{dx^4}\begin{pmatrix} \frac{1}{4} & -\frac{1}{6} & \frac{1}{24} & \cdots & & & \cdots & \frac{1}{24} & -\frac{1}{6} \\ -\frac{1}{6} & \frac{1}{4} & -\frac{1}{6} & \frac{1}{24} & \cdots & & & \cdots & \frac{1}{24} \\ \frac{1}{24} & -\frac{1}{6} & \frac{1}{4} & -\frac{1}{6} & \frac{1}{24} & & & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & & & \\ & & \frac{1}{24} & -\frac{1}{6} & \frac{1}{4} & -\frac{1}{6} & \frac{1}{24} & & \\ & & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & & & \frac{1}{24} & -\frac{1}{6} & \frac{1}{4} & -\frac{1}{6} & \frac{1}{24} \\ \frac{1}{24} & \cdots & & & \cdots & \frac{1}{24} & -\frac{1}{6} & \frac{1}{4} & -\frac{1}{6} \\ -\frac{1}{6} & \frac{1}{24} & \cdots & & & \cdots & \frac{1}{24} & -\frac{1}{6} & \frac{1}{4} \end{pmatrix}\varphi_j. \tag{70}$$

Let's call the matrix here M_4, because it presents the coefficients that we need to calculate the 4^{th} derivative. But I put the factor $1/dx^4$ from (70) inside the matrix M_4, as we have already had $1/dx^2$ inside M_2 in eq. (67). It is fair to do, because what we try to represent by these matrices is a *Finite Difference* scheme and without those denominators it would not be complete.

It seems, now we can continue and rewrite the eq. (62) using matrices M_2 and M_4 instead of $\partial^2/\partial x^2$ and $\partial^4/\partial x^4$. We shouldn't forget that we let $u(x,t) \to \varphi(x,t)$, after the discretization in space we will have $\varphi_j = \varphi(x_j, t)$. Of course, φ_j could be taken out of the brackets,

$$\begin{aligned} \partial_t\varphi_j &= (r-1)\varphi_j - 2(2!\cdot M_2)\varphi_j - 4!\cdot M_4\varphi_j - \varphi_j^3 = \\ &= (r-1)\varphi_j - (4M_2 - 24M_4)\varphi_j - \varphi_j^3. \end{aligned} \tag{71}$$

It is clear from the equation above that now we can consider the problem in even a simpler form. First of all let's recall that the last term φ_j^3 is a non-linear term that we need to control the pattern evolution, so it is preferable to leave it as an independent term that we can vary at any time. As concerns the other terms, here we can just take the function φ_j out of brackets and introduce the matrix called M instead,

$$M = (r-1)\cdot \mathrm{I} - 2(2!\cdot M_2) - 4!\cdot M_4, \tag{72}$$

where I is an *Identity matrix*. It has a main diagonal built of 'ones', while all the other elements are zeros. The *Identity matrix* appears in (72) out from the main rules about operations and calculations with matrices [22]. Having the whole right-hand side (except for φ_j^3) of (71) represented by the matrix M does simplify the way it looks a lot and gives a clear understanding of what is going

on when it comes that we have to run the ODE-solver. Therefore, using (72) we can rewrite the eq. (71) in the next form

$$\partial_t \varphi_j = M\varphi_j - \varphi_j^3. \tag{73}$$

It is pleasure to know and to write that this method, this way of writing derivatives in the matrix form (actually, writing the whole right-hand side of the one-dimensional *Swift-Hohenberg* equation in the matrix form), it has worked out perfectly when solving the problem and using *periodic boundaries*. By the way, we solve it in the same way as we have solved the *Schrödinger* equation in the Chapter 4.3.1.

Equation (73) is a first order ordinary differential equation (ode) in time, since we have substituted spatial derivatives with obtained matrices. Thus we consider using a built-in ODE-solver[23] from *MATLAB* to calculate the result. And due to the fact that we have introduced matrix M simplifies it a lot[24]. Finally, the right way of presenting the data for the built-in ODE-solver has been found. It made me very happy, even though it took quite a long time to come to the solution that works.

In the *Appendices* section I attach my *MATLAB* codes *[C]* that calculate matrices M_2 and M_4 (second and fourth order derivatives) when the *Periodic Boundary* condition is applied. Those codes *[C]* are the *'functions'* that are supposed to be used in the code *[D]* or in the code *[E]* that calculate the evolution of a reduced *Schrödinger* equation (9) and a one-dimensional *Swift-Hohenberg* equation (62) respectively.

As one can remember, when solving the *Schrödinger* equation (9), we used a Gaussian function (26). For the one-dimensional *Swift-Hohenberg* equation we can also use this one[25], but it's much better to use some function that has a small amplitude variation, and also it would be quite interesting to use a randomly distributed signal. Because what we expect from such an evolution equation as the *Swift-Hohenberg* equation is, that it will let some certain Fourier modes to grow in time. We have mentioned it in the Chapter 5.2, and in the caption under the Fig. 17.

I am sure one would agree that it is much more interesting to see these *Fourier modes* (the pattern eventually) appearing, arising from a randomly distributed function, almost from chaos, almost from nothing (according to its tiny ampli-

[23] As in the Chapter 4.3.1, we use the ODE-solver called *'ode23tb'*, which is good for solving stiff problems [20].

[24] Of course, I have run it for the reduced *Schrödinger* equation (9) as well. But this time we have periodic boundaries instead of PML. It means that if we have a wave packet moving to the right or to the left it is not going to disappear like it did when it was absorbed with a PML, instead it will go through the periodic boundary and appear looking the same from the other side of the plot. Because periodic boundaries make it look as if the wave packet would run on a circle. Also it has to be mentioned that the Gaussian function tends to fade away slowly while moving along the axis. The *MATLAB* code *[D]*, helps to see it clearly if needed.

[25] In this case we will only see the Gaussian fading away for $r < 0$ and becoming unstable for $r > 0$, as in a case with a White noise the unstable mode could be controlled by the non-linear term, for example $N(u) = u^3$.

tudes); but maybe from everything (considering it's chaotic nature). For the role of this function the **White noise** has been chosen.

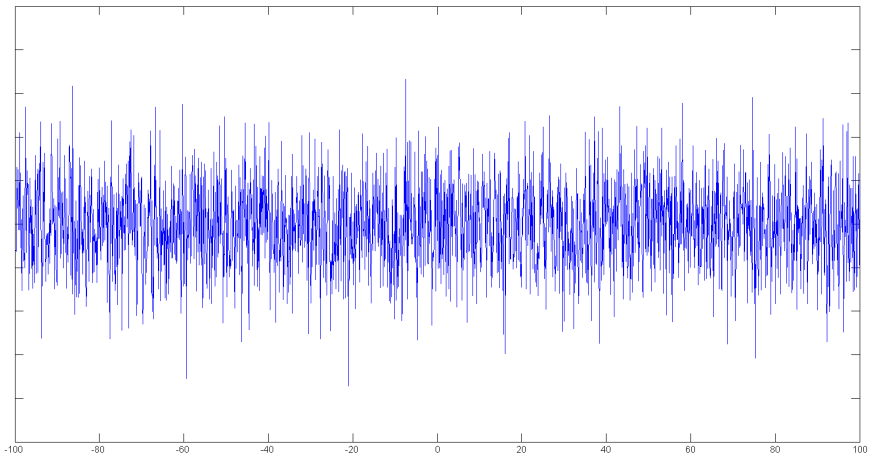


Figure 19: White noise that is generated by the built-in function '*wgn*', from *MATLAB* [20].

White noise is a random signal with a constant power spectral density [8]. In *MATLAB* one can easily generate a white Gaussian noise using the built-in function called '*wgn*' [20], thus we are not going to discuss it very thoroughly in here. If you are interested you can check [8] for an overview. You can see how the white noise performed by the *MATLAB* function looks like on the Fig. 19. But we will continue with the one-dimensional *Swift-Hohenberg* equation now, and let the White noise signal evolve, giving us a chance to check how it will look like.

We expect to see some certain modes growing up. According to the *linear stability analysis* we have done in Chapter 5.2, if we set the control parameter $r < 0$, then everything should be **stable**. I can assure you, that it is true, all the amplitudes of a signal they just dissolve in time. But the most interesting thing is to see what happens when an **instability** comes around. As we already know from before, the onset of instability is when we have $r > 0$. And that's what we could easily see letting the control parameter be positive (even slightly larger than zero).

First of all, let's do this experiment taking away a non-linear term from (73), which is φ_j^3. Thus the equation we consider will be

$$\partial_t \varphi_j = M\varphi_j. \tag{74}$$

In this case we have the instability coming around, and tiny amplitudes stay without any control. They keep on increasing infinitely as it could be easily seen (Fig. 20), and there is nothing to stop them.

From the other hand, we can leave a non-linear term φ_j^3 on its place. Then we are back to the eq. (73) without any changes applied. Having this non-linear term around, somehow we will be able to control this increasing instability and

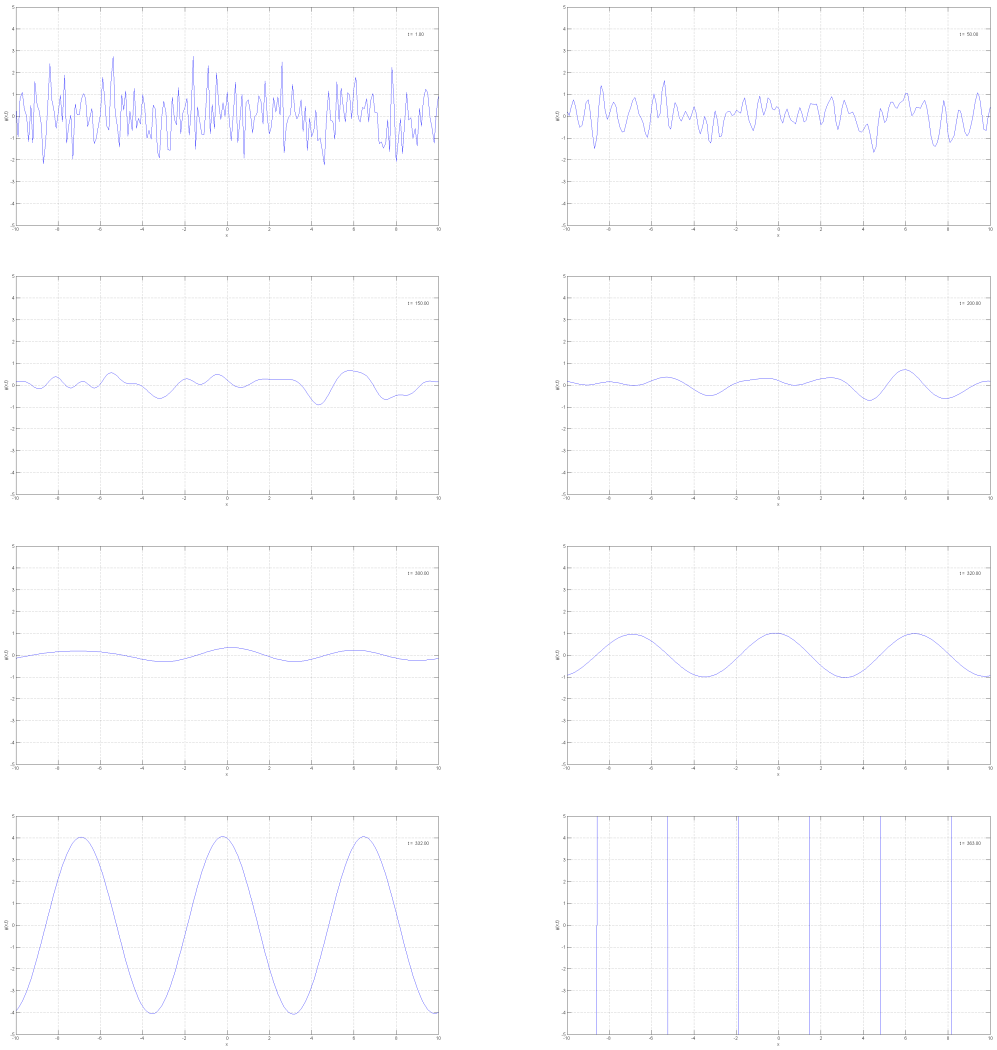


Figure 20: Unstable case of the evolution of a White noise when a one-dimensional *Swift-Hohenberg* equation (74) without a non-linear term is applied. The control parameter is set to be $r = 0.2$, slightly bigger then the critical value $r_c = 0$. You can see on the sequence of plots above (reading from left to right and then down) how: at first the randomly generated signal (White noise) fades away (until the fifth plot), and then at some moment (the sixth plot) expected instability appears and starts to grow infinitely.

a never-ending growth of the function $\varphi(x, t)$. The result can be observed on the next figure (Fig. 21).

It seems to me kind of magical! First of all, the fact that some similar structures (patterns) appear, arise from an absolutely randomly distributed medium as a white noise[26]. The second thing that makes it just wonderful, is that we are actually able to control the appearing instability using only a simple non-linear term such as φ^3. I would like to think it is really amazing and believe that

[26] As it was mentioned before any low-amplitude function (field) could be used here. For example, having a Gaussian function instead, one will still get the same kind of behaviour of the field $\varphi(x, t)$ and the plots (the pattern) will look very similar.

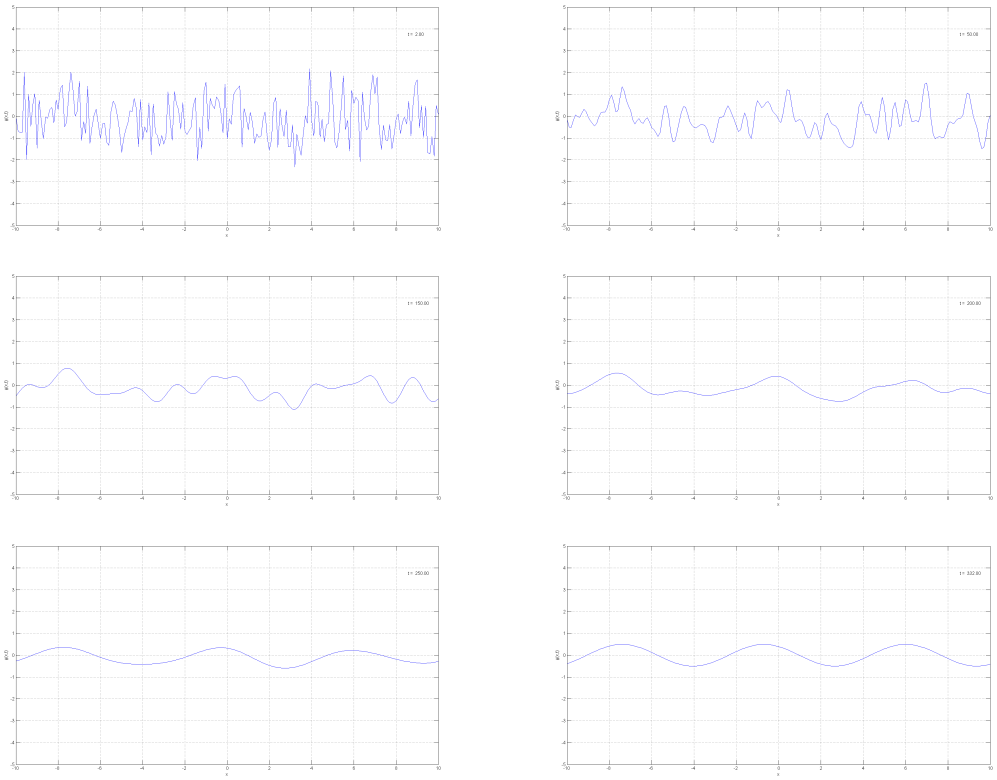


Figure 21: Unstable case of the evolution of a White noise when a one-dimensional *Swift-Hohenberg* equation (73) is applied. The control parameter is set to be $r = 0.2$, slightly bigger then the critical value $r_c = 0$. And the non-linear term is $N(\varphi) = \varphi^3$. On the sequence of plots above (reading from left to right and then down) you can see how: at first the randomly generated signal (White noise) fades down, and then at some moment (the fifth plot) expected instability appears and starts growing. But this time it is compensated by the non-linear term, so it won't increase infinitely. Instead it will stop after reaching some value and present us a desirable pattern.

one can find a lot of inspiration or some kind of a source for different exciting thoughts and ideas in it. Feels like this chapter has become long enough already, so maybe it is the right moment now to switch to the next one.

5.4 Fourier transform of a Swift-Hohenberg equation

Sometimes it happens that we have to do things to prove we are right, or maybe even to prove we are wrong. To prove this was right and that was wrong. Does it come that we don't have confidence enough or don't believe ourselves? Or maybe it says that you are not close yet to understand something, and you need to look at it from the other side, or from the inside. It is actually very nice to have such a possibility and not to be lazy to give it a chance. Especially, if you are not in haste.

Remember, in Chapter 5.2, we made a linear stability analysis for a one-dimensional *Swift-Hohenberg* equation and we have found that for $r = 0.2$ just positive, only a narrowband of *Fourier modes* centred on the critical wave num-

ber $q_c = 1$ can grow. This process does predict the appearance of a *pattern* with a characteristic length scale $\frac{2\pi}{q_c}$. To be honest, when working on this Thesis, an attempt to check how this works (Fourier transform), was taken only after we had stumbled over some problem finding a solution of a *Swift-Hohenberg* equation with PML settled on. But I think it is reasonable to go through it now, since we have just discussed a *Swift-Hohenberg* equation with periodic boundaries. Also, because until there everything will go smoothly in accordance with the theory and our analytics (to run a few steps forward).

So, what it is the *Fourier transform*? And how it could help us understand the process better? I really liked the simplicity of how it is described here [7], it says – all waveforms, no matter what you observe in the *Universe*, are actually just the sum of simple *sinusoids* of different frequencies. While the Fourier transform decomposes a waveform – basically any real world waveform, into sinusoids. That is, the Fourier transform gives us another way to represent a waveform[27]. Also the comparison that Wikipedia makes sounds really nice, it compares Fourier transform and the way it decomposes the function of time (a signal) into the frequencies that make it up, with a music chord! As the latter could be expressed as the amplitude (or loudness) of its constituent notes.

Thus, looking for the Fourier transform of a one-dimensional *Swift-Hohenberg* equation we will find which amplitudes are presented as the main ones, and the stability analysis predicts Fourier modes to grow around the critical wave number $q_c = 1$. To proceed with this a bit tricky thing I will go through the lecture notes we were given additionally on the course *'Mathematical Methods'* (at the UiT, University of Tromsø). But I will briefly make a sketch here as well. So, let's start.

If we let $\{u_r\}_{r=1}^n$ be a sequence of complex numbers. The discrete Fourier transform of the sequence $\{u\}_{r=1}^n$ is another complex sequence $\{v_s\}_{s=1}^n$. In *MATLAB*[28] this sequence is defined by

$$v_s = \sum_{r=1}^{n} u_r e^{-2\pi i \frac{(s-1)(r-1)}{n}}, \tag{75}$$

with the inverse as

$$u_r = \frac{1}{n} \sum_{s=1}^{n} v_s e^{2\pi i \frac{(s-1)(r-1)}{n}}. \tag{76}$$

If we go to the continuous variables, then the Fourier transform of a function $f(x)$ defined on $\mathbb{R}$ is

$$F(\lambda) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} dx\, f(x) e^{i\lambda x}, \tag{77}$$

[27] This goes for TV signals, cell phone signals, the sound waves that travel when you speak. In general, waveforms are not made up of a discrete number of frequencies, but rather a continuous range of frequencies. The Fourier transform is the mathematical tool that shows us how to deconstruct the waveform into its sinusoidal components.

[28] There are several different conventions when it comes to defining the Fourier transform. (75) and (76) is the default in *MATLAB*, in GSL – library that is used in the programming language *C*.

while its inverse is defined by

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} d\lambda \, F(\lambda) e^{-i\lambda x}. \tag{78}$$

Our goal now is to find a way how to use (75) and (76) to calculate (77) and (78) approximately.

Let us start by introducing the discretizations for the x-axis and λ-axis:

$$\alpha_j = (j - \frac{1}{2})\Delta x, \tag{79}$$

$$\beta_l = (l - \frac{1}{2})\Delta \lambda. \tag{80}$$

Observe that,

$$\alpha_{j+1} - \alpha_j = (j + \frac{1}{2})\Delta x - (j - \frac{1}{2})\Delta x = \Delta x, \tag{81}$$

$$\beta_{l+1} - \beta_l = (l + \frac{1}{2})\Delta \lambda - (l - \frac{1}{2})\Delta \lambda = \Delta \lambda. \tag{82}$$

Using discretizations $\{\alpha_j\}$ and $\{\beta_l\}$ we define the second pair of discretizations of the x-axis and λ-axis:

$$x_j = \frac{1}{2}(\alpha_{j+1} + \alpha_j) = j \cdot \Delta x, \tag{83}$$

$$\lambda_l = \frac{1}{2}(\beta_{l+1} + \beta_l) = l \cdot \Delta \lambda. \tag{84}$$

Using these descritizations and the midpoint rule for the integrals we have:

$$\begin{aligned} F(\lambda) = \frac{1}{\sqrt{2\pi}} \sum_{j=-\infty}^{+\infty} \int_{\alpha_j}^{\alpha_{j+1}} dx \, f(x) e^{i\lambda x} \approx \\ \approx \frac{\Delta x}{\sqrt{2\pi}} \sum_{j=-\infty}^{+\infty} f(x_j) e^{i\lambda x_j}, \end{aligned} \tag{85}$$

and in a similar way

$$f(x) \approx \frac{\Delta \lambda}{\sqrt{2\pi}} \sum_{l=-\infty}^{+\infty} F(\lambda_l) e^{-i\lambda_l x}. \tag{86}$$

In order to implement (85) and (86) on a computer the infinite series must be truncated. Thus, we say, if it is true that

$$F_l \equiv F(\lambda_l) \approx 0 \qquad \text{for } |l| > N, \tag{87}$$

then we have from (86)

$$f(x) \approx \frac{\Delta \lambda}{\sqrt{2\pi}} \sum_{l=-N}^{N} F_l e^{-i\lambda_l x}. \tag{88}$$

This formula tells us that we have approximated $f(x)$ with a function that is periodic with a longest period equal to

$$P_{\max} = \frac{2\pi}{\lambda_1} = \frac{2\pi}{\Delta\lambda}, \tag{89}$$

and the shortest period

$$P_{\min} = \frac{2\pi}{\lambda_N} = \frac{2\pi}{N \cdot \Delta\lambda}. \tag{90}$$

On the other hand, we want to approximate $f(x)$ by its values on the grid $\{x_j\}$, suggesting that $f_j \equiv f(x_j)$. The shortest period that can be detected on the grid $\{x_j\}$ is $2\Delta x$. Thus, in order for the approximation (88) for $f(x)$ to be well presented by the grid-values f_j we must have

$$\begin{gathered} 2\Delta x \leqslant P_{\min} \\ \Downarrow \\ \Delta x \leqslant \frac{\pi}{N\Delta\lambda}. \end{gathered} \tag{91}$$

On the other hand if eq. (88) is a good representation of $f(x)$ we gain nothing by making $2\Delta x$ smaller than $P_{\min}$. Thus, the optional choice is

$$\Delta x = \frac{\pi}{N\Delta\lambda} \quad \Rightarrow \quad \Delta x \Delta\lambda = \frac{\pi}{N}. \tag{92}$$

We now truncate the sum from eq. (85) in the same way, using the same N as in eq. (88). Evaluating $F(\lambda)$ on the grid $\{\lambda_l\}$ and $f(x)$ on the grid $\{x_j\}$ and using the found condition (92), one can get

$$F_l = \frac{\Delta x}{\sqrt{2\pi}} \sum_{j=-N}^{N} f_j e^{i\pi \frac{jl}{N}}, \tag{93}$$

$$f_j = \frac{\Delta\lambda}{\sqrt{2\pi}} \sum_{l=-N}^{N} F_l e^{-i\pi \frac{jl}{N}}. \tag{94}$$

Observe that for these two equations to be a good representation of the Fourier transform (77) and the inverse transform (78) we must have

$$F_N \approx 0 \quad \text{and} \quad f_N \approx 0. \tag{95}$$

Therefore, no accuracy is lost if we rather use the formulas:

$$F_l = \frac{\Delta x}{\sqrt{2\pi}} \sum_{j=-N}^{N-1} f_j e^{i\pi \frac{jl}{N}}, \qquad l = -N, \ldots, N-1, \tag{96}$$

$$f_j = \frac{\Delta\lambda}{\sqrt{2\pi}} \sum_{l=-N}^{N-1} F_l e^{-i\pi \frac{jl}{N}}, \qquad j = -N, \ldots, N-1. \tag{97}$$

What we have to do now is to find the way how to transform these two equations into (75) and (76), like they are defined in *MATLAB*. We first shift the indices

$$r = N + 1 + j, \qquad j = -N, \ldots, N-1, \tag{98}$$

$$s = N + 1 + l, \qquad l = -N, \ldots, N-1. \tag{99}$$

This will give us the formulas:

$$F_{s-N-1} = \frac{\Delta x}{\sqrt{2\pi}} \sum_{r=1}^{2N} f_{r-N-1} \cdot e^{i\pi \frac{(r-N-1)(s-N-1)}{N}}, \tag{100}$$

$$f_{r-N-1} = \frac{\Delta \lambda}{\sqrt{2\pi}} \sum_{s=1}^{2N} F_{s-N-1} \cdot e^{-i\pi \frac{(r-N-1)(s-N-1)}{N}}. \tag{101}$$

If we look at the power of the exponent in (100) and (101) more carefully, we will find that it could be written in another way as,

$$e^{i\pi \frac{(r-N-1)(s-N-1)}{N}} = e^{-i\pi(r-1)} e^{-i\pi(s-1)} e^{i\pi N} e^{i\pi \frac{(r-1)(s-1)}{N}}. \tag{102}$$

Next, we introduce our discrete Fourier transform v_s and u_r through the scaling from (79) and (80), this will give us

$$F_{s-N-1} = \alpha_s v_s, \tag{103}$$

$$f_{r-N-1} = \beta_r u_r. \tag{104}$$

After substituting these two to eqs. (75) and (76) we will get

$$v_s = \sum_{r=1}^{2N} u_r \frac{\beta_r \Delta x}{\alpha_s \sqrt{2\pi}} e^{-i\pi(r-1)} e^{-i\pi(s-1)} e^{i\pi N} e^{i\pi \frac{(r-1)(s-1)}{N}}, \tag{105}$$

$$u_r = \sum_{s=1}^{2N} v_s \frac{\alpha_s \Delta \lambda}{\beta_r \sqrt{2\pi}} e^{i\pi(r-1)} e^{i\pi(s-1)} e^{-i\pi N} e^{-i\pi \frac{(r-1)(s-1)}{N}}. \tag{106}$$

Let's choose

$$\beta_r = \beta_0 e^{i\pi(r-1)} e^{-i\pi N} \qquad \text{and} \qquad \alpha_s = \alpha_0 e^{-i\pi(s-1)}. \tag{107}$$

This will simplify two previous equations which we can write now as

$$v_s = \sum_{r=1}^{2N} u_r \frac{\beta_0 \Delta x}{\alpha_0 \sqrt{2\pi}} e^{i\pi \frac{(r-1)(s-1)}{N}}, \tag{108}$$

$$u_r = \sum_{s=1}^{2N} v_s \frac{\alpha_0 \Delta \lambda}{\beta_0 \sqrt{2\pi}} e^{-i\pi \frac{(r-1)(s-1)}{N}}. \tag{109}$$

Now we can 'play' a bit with constants, but don't forget that what we are actually trying to do is to make the last two equations look like (75) and (76), the default Fourier transform and its inverse from *MATLAB*. Therefore, we choose:

$$\begin{aligned} \frac{\beta_0 \Delta x}{\alpha_0 \sqrt{2\pi}} = 1 \quad &\Rightarrow \quad \Delta x = \frac{\alpha_0 \sqrt{2\pi}}{\beta_0}, \\ \frac{\alpha_0 \Delta \lambda}{\beta_0 \sqrt{2\pi}} = \frac{1}{2N} \quad &\Rightarrow \quad \Delta \lambda = \frac{\beta_0 \sqrt{2\pi}}{\alpha_0 \cdot 2N}. \end{aligned} \tag{110}$$

From here, by calculating $\Delta x \Delta \lambda$ we see that the condition (92) is already satisfied. Thus, letting $N = n/2$ we will be able to get desired (75) and (76) from eqs. (108) and (109). But let's do one more thing. Since the condition (92) is satisfied, that we see from (110). It means, that we can choose constants α_0 and β_0 to be any numbers (but zero, of course). Let's take it easy and just choose $\beta_0 = 1$, which will give us $\alpha_0 = \Delta x / \sqrt{2\pi}$. Now, with this features on, we are going to put obtained results from (110) and our chosen variables (107) to the scaled equations (103) and (104). After some calculations are done we get next formulas

$$v_s = \frac{\sqrt{2\pi}}{\Delta x} e^{i\pi(s-1)} F_{s-N-1}, \qquad s = 1, \ldots, n = 2N, \tag{111}$$

$$u_r = e^{i\pi N} e^{-i\pi(r-1)} f_{r-N-1}, \qquad r = 1, \ldots, n = 2N, \tag{112}$$

and

$$F_l = \frac{\Delta x}{\sqrt{2\pi}} e^{-i\pi(l+N)} v_{l+N+1}, \qquad l = -N, \ldots, N-1, \tag{113}$$

$$f_j = e^{i\pi(j+1)} e^{-i\pi N} u_{j+N+1}, \qquad j = -N, \ldots, N-1. \tag{114}$$

Thus, we have found a connection between approximated continuous and discrete Fourier transform – eqs. (111) and (113). As well as we have found this connection between their inverse transforms – eqs. (112) and (114). But how should we use it now? Would be probably a logical thing to ask. Honestly, it took me a while before I figured out the way of applying it to our problem, exactly, to the problem of finding numerically a Fourier transform of a one-dimensional *Swift-Hohenberg* equation with the initial signal in the form of a White noise. We do it in *MATLAB*, of course.

To start with, we use the same code *[E]* as before to calculate the solution of a *Swift-Hohenberg* equation. When it comes to find its Fourier transform we proceed next way. We should understand, that the function $\varphi(x_j, t)$, we have found, becomes an approximated continuous inverse transform $f_j(t)$, analogous to the (97) or (101) after the shift of indices, only now it also evolves in time. This gives us a possibility to find a discrete inverse Fourier transform u_r from the eq. (112). But then we make a trick! The one that took me a while to understand.

Using *MATLAB* procedure that calculates Fourier transform [20], we can find the discrete Fourier transform v_r by applying to the already found function u_r which is the discrete inverse transform. I know there are too many similar words in the last two sentences, maybe they need to be read few times before

it becomes clear. Though, I can try to formulate it in another words: we have to take our function $\varphi(x_j, t)$, call it approximately continuous inverse Fourier transform, make a transformation to the discrete inverse Fourier transform u_r, then calculate a discrete Fourier transform v_s (using *MATLAB* built-in function), and the last step we find the approximated continuous Fourier transform F_l using the relation (113).

When plotting results we also have to take into consideration that Fourier transform has to be plotted along the frequency domain, which is a grid λ_l, to refresh look at the eq. (84). We use the rule (92) to set the grid λ_l in the code *[F]*. Finally we can plot the Fourier transform $F(\lambda_l)$ versus the frequencies λ_l. And as one can see (Fig. 22) there is certainly a growth of *Fourier modes* centred on the critical wave-number, which in our case corresponds to the frequency $\lambda = 1$ (I denote it with a green line on both Fig. 22 and Fig. 23). By the way, on the Fig. 23 we find the zoomed still from the last frame of the sequence showed before. It could be seen there that a narrowband of *Fourier modes* around the critical frequency (wave-number) does actually grow.

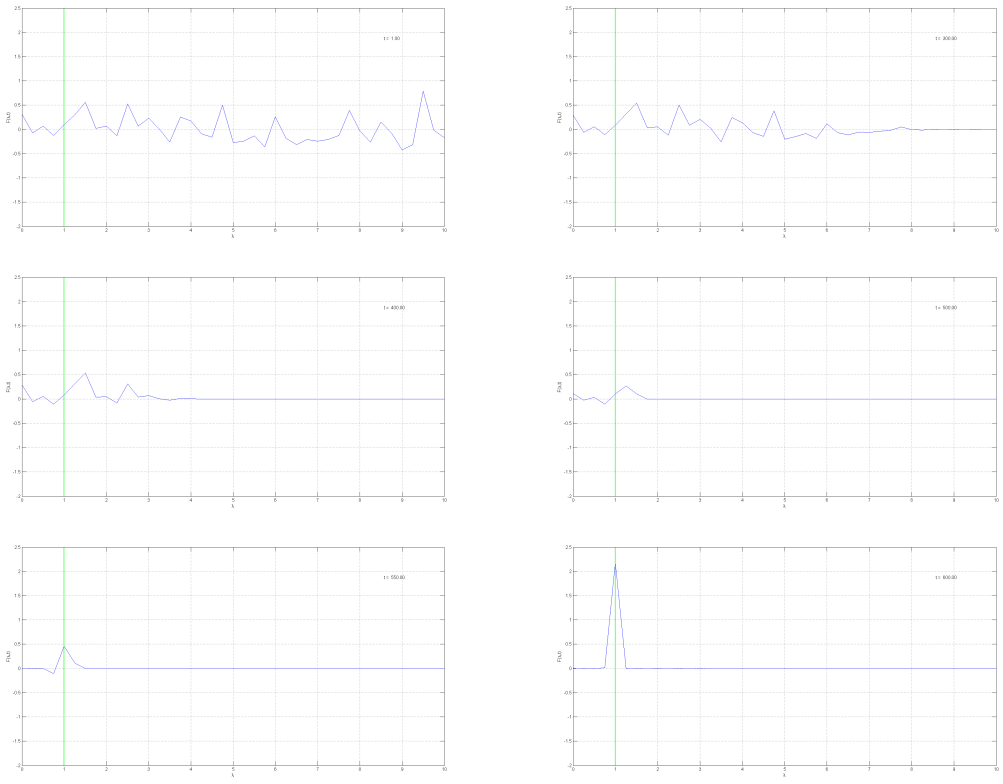


Figure 22: Fourier transform of a one-dimensional *Swift-Hohenberg* equation (with a White noise as an initial function), plotted along the frequency grid λ_l. We can see on the plots (reading from left to right and then down) how the most furthermost from the critical wave-number $q_c = 1$ (the green line on the plots) *Fourier modes* decay first. And those, close to the critical point, create a narrowband around it with the maximum in the middle. The control parameter is $r = 0.2$ here, and the non-linear term, to stop the increasing instability, is chosen as before to be φ^3. The results are obtained using code *[F]* from *Appendices* section.

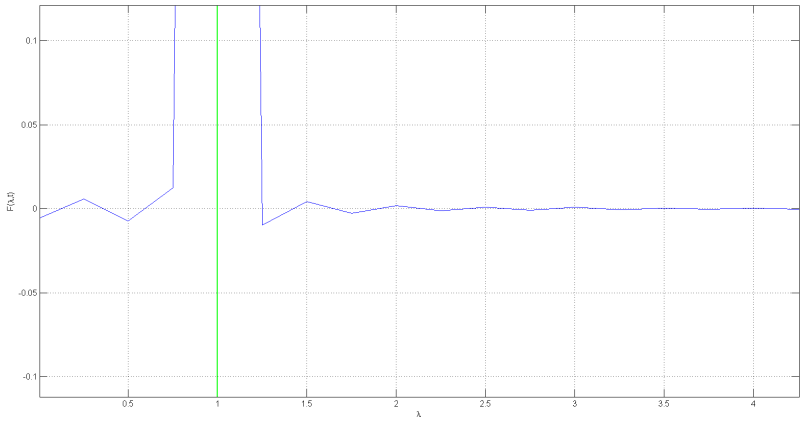


Figure 23: This is a zoomed still from the last frame on the Fig. 22. Just to see a narrowband of growing *Fourier modes* centred on the critical wave-number or frequency.

Sometimes we don't have time enough to explore everything we consider interesting. Or maybe just turn a bit different direction due to some of our preferences, and other questions standing in front of us. But it is nice that we are many, and if you look more carefully you will see that probably the problem you are busy with has been solved by someone else before.

As I have told in the introduction to the Chapter 5.3, applying Periodic Boundaries to the *Swift-Hohenberg* equation wasn't something new. And I have found some nice videos on *YouTube* about how this equation evolves in time and space (on xy-plane). Probably it could be my next step if we were not busy with applying PML. But I wish to share few frames from this videos just to show how the appearing pattern obtained by solving the *Swift-Hohenberg* equation could look like. Moreover, I have contacted the guy who made these films. And it happened that they were a part of his Master's Thesis and an article [6] some years ago. It feels amazingly nice to find things like this.

6 Preparing the Swift-Hohenberg equation for the plane with PML

When we first make a step in any direction. Do we really know where it will lead? We probably have some expectations or some preferences but even these two are subjects to change. It seems, there could be so many reasons for this transformation to happen. It could be something coming from the inside, as your own wish and desire. But it could also appear as an external foreign interference. There is a funny fact, because, in the first case it would probably look like an open door, the one that you have been trying to find for so long, it might turn into the river and let you float down or up the stream for some time. Nice. But in the second case, it will arise like some kind of a wall, like an obstruction. It won't occur that you already know how to deal with this. It won't occur that

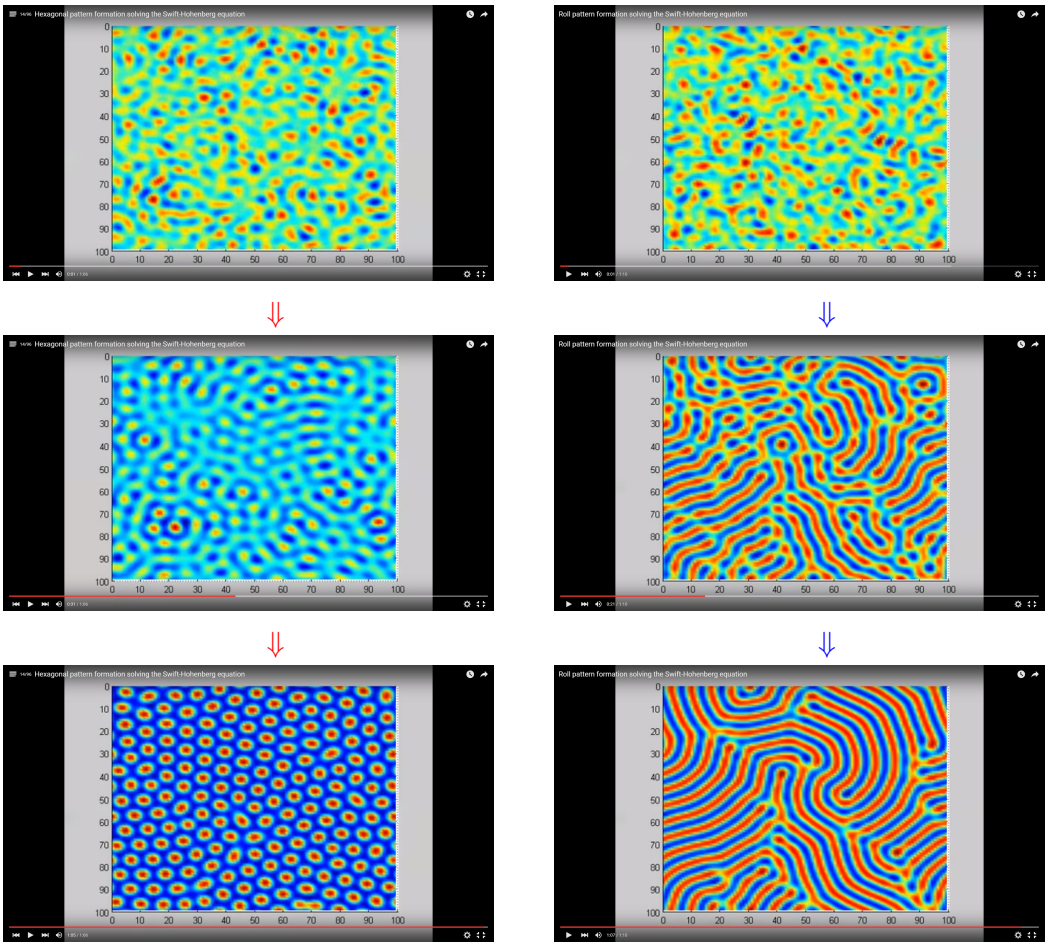


Figure 24: Simulation of the formation and evolution of a hexagonal pattern (on the left) and a roll pattern (on the right) by numerically solving the *Swift-Hohenberg* equation with *Periodic Boundaries* set on. These films appear to be a result of a Master's Thesis and an article from [6] and could be found on *YouTube* at: (left) https://www.youtube.com/watch?v=tJpTkU3LQdk, (right) https://www.youtube.com/watch?v=C0ZcQqN4rRY.

you can take it easy. And you should not! There is so much more there, even if it says the way to go is blocked. Be sure, the path will clear up, it just needs a bit more of your ideas and patience, or could be passion. I don't think it ever works for real without any passion.

In this Chapter we are going to see the way of applying **PML** to a one-dimensional *Swift-Hohenberg* equation. After we have figured out how it works for the reduced *Schrödinger* equation (Chapter 4) and for the *Swift-Hohenberg* equation with periodic boundaries (Chapter 5), that were both a simplification and a nice basis for the real problem we have to solve, it should seem that there are no obvious difficulties to overcome now. But it happened, there were some and unfortunately they didn't let us go as far as it had been planned and wished in the very beginning, so long time ago though. But we didn't give up and took it with all the courage and excitement, while the time was running out. I will try to go step by step to present the obtained results.

6.1 Let's make a new Matrix

Since we know (Chapter 5.3) that the *matrix* like (67) and (70) works fine as a way to represent the second and fourth order spatial derivatives of some discrete function $\varphi(x_j, t)$, we would like to use the same method of constructing our data. But this time we have a distinct difference, we want to set up a *PML (Perfectly Matched Layer)*, to refresh it in the mind take a look through the Chapter 4. This PML and an absence of *periodic boundaries* (that we don't need and don't want any more) would certainly make the *matrices* look differently. Also we should not forget that PML means that we will have to expand our problem to the *complex* plane, but having a good example as one with a *Schrödinger* equation, helps us to handle this.

I am far of being the best one when it comes to write a code, and honestly, there were a lot of mistakes floating out almost every time I was about to solve some problem numerically. So, it seemed reasonable to go through some intentionally made 'check points' to know for sure, that at the first, second and third stage there were no troubles.

Therefore, to reduce the problem we will start with the time independent case. We let our function be only a function of x, thus $\varphi(x_j)$. We will try to build a matrix that presents a first step of a one-dimensional *Swift-Hohenberg* equation on the plane with PML set at the points $-a$ and a. Afterwards we will compare results with an exact solution, because it is not so hard to do for an initial condition only. Exactly for this time independent case we will return to the Gaussian function as an initial one. But we will come back to the *White noise* signal when considering a time evolving process.

To start with, we need to discretize the problem correctly again. And in a similar way as for the *Schrödinger* equation we demand that PML-points $-a$ and a were exactly on the grid. So, after giving some value a we are going to set a step for the spatial grid as

$$dx = \frac{a}{M}, \tag{115}$$

where a is a value of the point where we want to place our PML, and M is a number of points we want to have on the grid line, starting from the origin of the coordinates – zero, and until the PML. Now we have no doubts that PML-points are exactly on the grid. Next, we choose some number $N > M$ to set the number of points on the grid from the origin of coordinates to the lateral boundary $L > a$. Since we want to consider the problem from the left boundary $-L$, passing through the left PML $-a$, to the right boundary L, passing through the right PML a, we will have to pass the point of origin. Thus, the whole number of points on the spatial grid will be

$$n = 2N + 1, \tag{116}$$

and we can set the right lateral boundary as

$$L = N \cdot dx. \tag{117}$$

It seems everything is ready to create a spatial grid – commonly we choose a discrete coordinate x_j, which takes all the values from $-L$ to L with the step dx.

According to (115) we are quite sure that it passes through the PML-points $-a$ and a. But to introduce the PML one has to expand the problem to the *complex* plane. Certainly, it will modify the spatial grid, we apply the same PML as in eq. (39), for a visual refresh one can take a look at the Fig. 15. But I think we better rewrite the PML condition again, just to have it within easy reach,

$$z = \begin{cases} a + e^{i\theta}(x - a), & x \geqslant a \\ x, & -a < x < a \\ -a + e^{i\theta}(x + a), & x \leqslant -a. \end{cases} \tag{118}$$

Again, we are interested in making our PML look as a straight vertical line, which simplifies it even more, we choose $\theta = \pi/2$ and the system above gets reduced to

$$z = \begin{cases} a + i(x - a), & x \geqslant a \\ x, & -a < x < a \\ -a + i(x + a), & x \leqslant -a. \end{cases} \tag{119}$$

If we try to visualise what actually happens, we could say that when the function $\varphi(z_j)$ (yes, we can have it complex if we want) is considered on the region between the PML-points (in between $-a$ and a) it should behave as it did before and go along the x-axis, but as soon as it occurs to be at the PML-point $-a$ or a it will start moving along the complex z-axis, which is calculated according (118) or (119) if we have chosen $\theta = \pi/2$. And here we expect it to decay rapidly with no reflection back, that's the idea of applying a PML.

We proceed with using the same method to introduce second and fourth spatial derivatives through the a_2 and a_4 coefficients of the *Taylor Expansion* (63). I think it is actually such a clear and beautiful way to go through this problem, because once it has been done it could be applied again, even for the problem with slightly different features. So the easiest way to rewrite the problem would be just jump from the real grid x_j to the new complex one with z_j which we discretize using eq. (118), thus we obtain

$$\begin{cases} \varphi_{j-2} = a_0 + a_1(z_{j-2} - z_j) + a_2(z_{j-2} - z_j)^2 + a_3(z_{j-2} - z_j)^3 + a_4(z_{j-2} - z_j)^4 \\ \varphi_{j-1} = a_0 + a_1(z_{j-1} - z_j) + a_2(z_{j-1} - z_j)^2 + a_3(z_{j-1} - z_j)^3 + a_4(z_{j-1} - z_j)^4 \\ \varphi_j = a_0 \\ \varphi_{j+1} = a_0 + a_1(z_{j+1} - z_j) + a_2(z_{j+1} - z_j)^2 + a_3(z_{j+1} - z_j)^3 + a_4(z_{j+1} - z_j)^4 \\ \varphi_{j+2} = a_0 + a_1(z_{j+2} - z_j) + a_2(z_{j+2} - z_j)^2 + a_3(z_{j+2} - z_j)^3 + a_4(z_{j+2} - z_j)^4. \end{cases} \tag{120}$$

It could seem familiar... Yes, it does look exactly like the system (36), but we haven't really used it last time, because we have been looking for the second order derivative, so we could take another system (32) with a lower precision. Nevertheless, even then I managed to use it in a very ineffective way, I didn't take care to simplify it, that's why the code for the reduced *Schrödinger* equation did run so slow (Chapter 4.3). Also because we didn't try to represent the data as a matrix before, instead I just gave it as it was to the ODE-solver with all

its huge formulas, that the solver needed to go through again and again as it was evaluating the problem in time. But here we are with our mistakes and our understanding that almost always comes a bit later, and there is always this kind of a mixed strange feeling when you find yourself being wrong but walking so close to the truth all the time, and finally you can face it. It's a nice feeling.

Let us see how we can simplify the problem. First of all, we know that on the region between the PML, we have everything in the same way it was before. Thus, in between the points $-a$ and a we could write that the system will take a form (63), and after simplification (64) which will lead consequently to the same second and fourth derivatives represented through the coefficients a_2 and a_4 ((65) and (69)). I would write them here again, then we don't need to page back all the time,

$$a_2 = \left(-\frac{1}{24}\varphi_{j-2} + \frac{2}{3}\varphi_{j-1} - \frac{5}{4}\varphi_j + \frac{2}{3}\varphi_{j+1} - \frac{1}{24}\varphi_{j+2} \right) \frac{1}{dx^2}, \tag{121}$$

and

$$a_4 = \left(\frac{1}{24}\varphi_{j-2} - \frac{1}{6}\varphi_{j-1} + \frac{1}{4}\varphi_j - \frac{1}{6}\varphi_{j+1} \frac{1}{24}\varphi_{j+2} \right) \frac{1}{dx^4}. \tag{122}$$

This let us to write the same lines in the matrices, that we are trying to create, as in (67) and (70). But don't forget, it is only for the elements lying in between the PML.

When it comes to any of the points where PML starts, rules are changing, and we have to apply a *Taylor Expansion* (120) for z from (119). Actually, if we pay more attention we will see that the rules start changing a bit earlier. Because when applying a *Taylor Expansion* to find the *Finite Difference* rule using fourth order accuracy (to introduce coefficients a_2 and a_4), we need two points from each side to calculate the value of the point in the middle.

It means, that when we want to know the solution for the point which is one step away from PML (say x_{M-1}) then we already need the value at the PML-point $x_M = a$ and even the value of the next one which is inside PML x_{M+1}. But wait, it is not x any more! According to the PML that we have set (119) we enter the *complex* plane now and the next point would be $z_{M+1} = a + i(x_{M+1} - a)$. I think it could be much easier to see it on the sketch (Fig. 25), so I just attach it below.

As it goes for the next point, which is exactly where the PML begins, we will need two points from the left (x_{M-1} and x_{M-2}) and two points from the inside of our PML. A similar kind of a picture we have, when one wants to find the solution for the second point inside the PML layer, which is z_{M+1}, to calculate it we will need next two points from PML (z_{M+2} and z_{M+3}), the PML starting point $z_M = x_M$, and one point from outside of the PML x_{M-1}.

It is quite obvious, that these three points around the right PML boarder a, and similar three points around the left one $-a$, are definitely very special points! Because they are out of any rules. They are a combination of the points from a region without PML and points from the inside of PML. So I suggest to consider carefully, step by step, at least one point like this and write down the solution. While the other similar points could be done analogically.

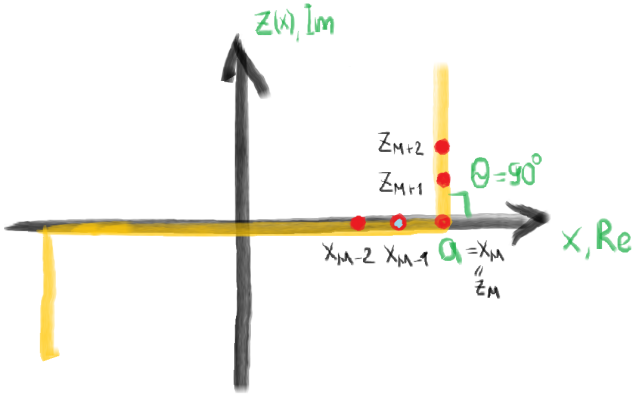


Figure 25: On the sketch you can see the points (red dots) that we have to use in order to calculate the value in the point x_{M-1} one step before the PML starts. According to the *Taylor Expansion* (120), if we want to find its value we already have to use one point from inside of the PML, which is z_{M+1} and could be calculated by (119).

Let's choose the point we have already talked about, a point one step before the right PML boarder x_{M-1} (Fig. 25), and write down the *Taylor Expansion* for the function $\varphi(x_j) = \varphi_j$ around this point, we will have

$$\begin{cases} \varphi_{M-3} = a_0 + a_1(x_{M-3} - x_{M-1}) + a_2(x_{M-3} - x_{M-1})^2 + a_3(x_{M-3} - x_{M-1})^3 + a_4(x_{M-3} - x_{M-1})^4 \\ \varphi_{M-2} = a_0 + a_1(x_{M-2} - x_{M-1}) + a_2(x_{M-2} - x_{M-1})^2 + a_3(x_{M-2} - x_{M-1})^3 + a_4(x_{M-2} - x_{M-1})^4 \\ \varphi_{M-1} = a_0 \\ \varphi_{M} = a_0 + a_1(z_{M} - x_{M-1}) + a_2(z_{M} - x_{M-1})^2 + a_3(z_{M} - x_{M-1})^3 + a_4(z_{M} - x_{M-1})^4 \\ \varphi_{M+1} = a_0 + a_1(z_{M+1} - x_{M-1}) + a_2(z_{M+1} - x_{M-1})^2 + a_3(z_{M+1} - x_{M-1})^3 + a_4(z_{M+1} - x_{M-1})^4. \end{cases} \tag{123}$$

Before introducing the coefficients a_2 and a_4 we are able to simplify the problem, in a similar way as we did when solving *Swift-Hohenberg* equation with *Periodic boundaries*. Such terms as $x_{M-3} - x_{M-1}$ and $x_{M-3} - x_{M-2}$ are obviously equal to $-2dx$ and $-dx$. As concerns z_M, it equals to $x_M = a$, thus the difference $z_M - x_{M-1} = dx$. But the last equation is the most interesting one because here we have z_{M+1}, using (119) let's calculate the term from the last equation from the system,

$$\begin{aligned} z_{M+1} - x_{M-1} &= \big[a + i(x_{M+1} - a)\big] - x_{M-1} = \\ &= x_M - x_{M-1} + i(x_{M+1} - x_M) = \\ &= dx + i(dx) = (i+1)dx. \end{aligned} \tag{124}$$

Isn't it fantastic?! Let's rewrite our simplified system now, using obtained results,

$$\begin{cases} \varphi_{M-3} = a_0 + a_1(-2dx) + a_2(-2dx)^2 + a_3(-2dx)^3 + a_4(-2dx)^4 \\ \varphi_{M-2} = a_0 + a_1(-dx) + a_2(-dx)^2 + a_3(-dx)^3 + a_4(-dx)^4 \\ \varphi_{M-1} = a_0 \\ \varphi_{M} = a_0 + a_1(dx) + a_2(dx)^2 + a_3(dx)^3 + a_4(dx)^4 \\ \varphi_{M+1} = a_0 + a_1\big((i+1)dx\big) + a_2\big((i+1)dx\big)^2 + a_3\big((i+1)dx\big)^3 + a_4\big((i+1)dx\big)^4. \end{cases} \tag{125}$$

This system (125) is exactly the one we try to solve now to find desired coefficients a_2 and a_4 at a certain step[29], around the point x_{M-1}. Let's see what we have for the coefficient a_2 which represents the second order derivative, I calculate it using the code *[B]* and get next result

$$a_2 = \Big(\Big(\frac{i}{60} - \frac{1}{20}\Big)\varphi_{M-3} + \Big(\frac{7}{10} - \frac{i}{10}\Big)\varphi_{M-2} + \Big(\frac{i}{4} - \frac{5}{4}\Big)\varphi_{M-1} + \\ + \Big(\frac{1}{2} - \frac{i}{6}\Big)\varphi_M + \frac{1}{10}\varphi_{M+1}\Big)\frac{1}{dx^2}. \tag{126}$$

Of course, it could be simplified even more, but it doesn't really seem to be of any use from the perspective of the numerical calculation. Thus we let ourselves to leave it like this, and write the coefficient a_4, which as one can find is

$$a_4 = \Big(\Big(\frac{1}{20} - \frac{i}{60}\Big)\varphi_{M-3} + \Big(\frac{i}{10} - \frac{1}{5}\Big)\varphi_{M-2} + \Big(\frac{1}{4} - \frac{i}{4}\Big)\varphi_{M-1} + \\ + \frac{i}{6}\varphi_M - \frac{1}{10}\varphi_{M+1}\Big)\frac{1}{dx^4}. \tag{127}$$

In the same way as we did before (Chapter 5.3) the derivatives could be found for this 'special' point now and the corresponding rows in the matrices could be written. Though we should not forget that it was only one out of 6 'special' points. But the rules for another 5 points could be found following the same scheme. When this is done, we can go to the next step.

It is needed now to find the rule to evaluate the function inside the PML. But it is not as difficult as it could seem. Using the same *Taylor Expansion* (120) we calculate it for the PML which is the extension to the *complex* plane according to the rule (119) we have set. Thus, it is obvious that the system (120) after the simplification will turn into

$$\begin{cases} \varphi_{j-2} = a_0 + a_1(-2idx) + a_2(-2idx)^2 + a_3(-2idx)^3 + a_4(-2idx)^4 \\ \varphi_{j-1} = a_0 + a_1(-idx) + a_2(-idx)^2 + a_3(-idx)^3 + a_4(-idx)^4 \\ \varphi_j = a_0 \\ \varphi_{j+1} = a_0 + a_1(idx) + a_2(idx)^2 + a_3(idx)^3 + a_4(idx)^4 \\ \varphi_{j+2} = a_0 + a_1(2idx) + a_2(2idx)^2 + a_3(2idx)^3 + a_4(2idx)^4. \end{cases} \tag{128}$$

If one solves this system one will find that inside the PML coefficients $a_{2_{\text{PML}}}$ and $a_{4_{\text{PML}}}$, that we need to calculate the derivatives, will be

$$a_{2_{\text{PML}}} = -a_2 \qquad \text{and} \qquad a_{4_{\text{PML}}} = a_4, \tag{129}$$

where a_2 and a_4 are the PML-points from eqs. (121) and (122). Which is basically means that the second derivative inside the PML changes its sign while the fourth derivative stays unchanged

$$\partial_{zz} \to -\partial_{xx} \qquad \text{and} \qquad \partial_{zzzz} \to \partial_{xxxx}, \qquad \text{(inside the PML)}. \tag{130}$$

[29] As before, I use my code *[B]* to solve such system of equations.

It gives us a nice possibility to write the next corresponding rows in the matrices M_2 and M_4. Those are the rows on the left from the point $-a$ and on the right from the point a where the PML starts. Let's not forget that we already have the values for the 6 'special' points around the PML's onsets.

We are almost done now, we have the values (121) and (122) to fill the middles of the matrices' diagonals where there is no PML and everything goes as it should in 'reality'. We also know the rule (129) which let us write down the rows on the left and on the right sides where the PML is set. Then in the same way as it shown through the example for one of the 'special' points, we can get equations similar to the eqs. (126) and (127), and we will have the corresponding matrices' diagonals filled with new elements. Last thing is left to do. We have to find the values for the edge points.

In Chapter 5.3 we set *Periodic boundaries*, and there was one way of defining the edge points. But now we wish to use only PML, so we keep on using a *Finite Difference Method* to find the edge points. Since we have already found the rule for defining derivatives (130) inside the PML we can simply use the forward and backward finite difference [21] to find values for the 4 edge points (two on the left-hand side and two on the right[30]). For example, the rules for the coefficients $a_2|_{j=1}$ and $a_4|_{j=1}$ belong to the very first point, will look like

$$a_2|_{j=1} = -\left(\frac{15}{4}\varphi_1 - \frac{77}{6}\varphi_2 + \frac{107}{6}\varphi_3 - 13\varphi_4 + \frac{61}{12}\varphi_5 - \frac{5}{6}\varphi_6\right)\frac{1}{2! \cdot dx^2}, \tag{131}$$

and (using second order accuracy here)

$$a_4|_{j=1} = \left(3\varphi_1 - 14\varphi_2 + 26\varphi_3 - 24\varphi_4 + 11\varphi_5 - 2\varphi_6\right)\frac{1}{4! \cdot dx^4}. \tag{132}$$

One can see that we have had to put minus sign to the equation for the $a_2|_{j=1}$ coefficient, and we have saved the sign for the $a_4|_{j=1}$, we do it according to the found rule (129). Also, relying on the formula (31) that represents the relation between coefficients a_q and the derivative of a function we add factorials 2! and 4! to these equations, because the numbers in the brackets from [21] are the one to calculate the derivatives according to the Finite Difference method, not the coefficients themselves from the *Taylor Expansion*.

Hope it wasn't too rambling and puzzled, but now we have all the elements we need to build the matrices M_2 and M_4 to find the spatial derivatives from a one-dimensional *Swift-Hohenberg* equation (62). To create these matrices I have written a *MATLAB* code *[G]*, it took a while to make it work correctly and put all the elements in the right rows and columns.

To check if the matrices are done correctly we take some simple initial function, the one that will let us calculate its derivatives analytically (to compare with numerical solution later). Therefore, the Gaussian 'bell' was chosen,

$$\varphi_0 = e^{-\gamma(z-z_0)^2}, \tag{133}$$

[30] When you do a *Finite Difference* for the edge points it is called *forward* (for the first point) and *backward* (for the last point) *Finite Difference* [5, 21].

where z is defined by (119), while γ and z_0 are constant that could be varied. Next step is to analytically calculate second and fourth order derivatives of this function, one can easily find that

$$\partial_{zz}\varphi_0 = -2\gamma \cdot \varphi_0 + 4\gamma(z - z_0)^2 \cdot \varphi_0, \tag{134}$$

and

$$\partial_{zzzz}\varphi_0 = 12\gamma^2 \cdot \varphi_0 - 48\gamma^3(z - z_0)^2 \cdot \varphi_0 + 16\gamma^4(z - z_0)^4 \cdot \varphi_0. \tag{135}$$

Now we can find an exact solution for the first step of a one-dimensional *Swift-Hohenberg* equation, we would like to reduce our problem a bit more though, and take away the non-linear term. we can do this, because what we truly trying to do, is to check if the exact and numerical solution for the derivatives do match. Thus, using (134) and (135) we calculate the function

$$\varphi_{j_{\text{exact}}}\Big|_{t=0} = (r-1)\varphi_{0_j} - 2 \cdot \partial_{zz}\varphi_{0_j} - \partial_{zzzz}\varphi_{0_j}. \tag{136}$$

And then, with the help of the matrices M_2 and M_4 that we have found, we solve this problem numerically, with the method based on the *Taylor Expansion* and the *Finite Deference*. The solution could be written as

$$\varphi_{j_{\text{numer}}}\Big|_{t=0} = \Big((r-1) \cdot \mathrm{I} - 2 \cdot 2! \cdot M_2 - 4! \cdot M_4\Big)\varphi_{0_j}, \tag{137}$$

where I, is the *Identity matrix*[31]. We need it to proceed in a right way with calculations when it comes to the matrices. When we have finally calculated our *exact* (136) and *numerical* (137) solutions we can plot them together (Fig. 26). Then we clearly see how both *real* and *imaginary* parts of the solution do match. Moreover, on this figure it seems that they even match perfectly, but if we zoom some places we will find that there is a tiny difference in results, but this error occurs to be too negligible to worry about. It is also possible to check the matrices directly comparing $2! \cdot M_2$ with the second order derivative (134) and $4! \cdot M_2$ with the fourth (135). And of course it has been done as well[32]. I just save place here not to overwhelm it with the figures.

Though, it could be interesting and important to look closer on the 'special' points and on the points at the edges. On the following Fig. 27 one can observe a slight difference between the values for the 3 'special' points around and exactly at the right PML-point a. As we can see the matching is not perfect, though it has been considered to be small enough to keep on this method. As concerns the edge points, the error occurs to be very small there, so I don't find it necessary to insert an extra plot for it.

I think here we can stop, and if I didn't know what is waiting us soon, I would say with enough of confidence, that the matrices we have made are quite reliable

[31] It has been mentioned before (Chapter 5.3), the *Identity matrix* is a square matrix with the main diagonal built out of 1 (ones), while all the other elements are 0 (zeros). It is needed here to proceed in a proper way with the calculations [22]. In *MATLAB* one can call the *Identity matrix* using a built-in function *'eye(n)'* [20].

[32] I leave the possibility of doing it again through the code *[G]* using 'commented' parts.

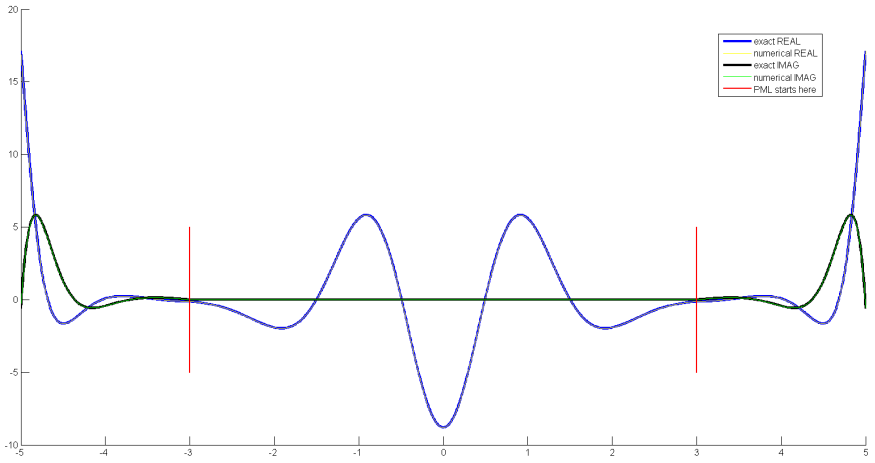


Figure 26: The matching of the *exact* (136) and *numerical* (137) solutions for the first step of a one-dimensional *Swift-Hohenberg* equation (62) with the initial function in the form of a Gaussian (133). It is obvious that both *real* and *imaginary* parts match really well. Two vertical red lines introduce the onset of a PML.

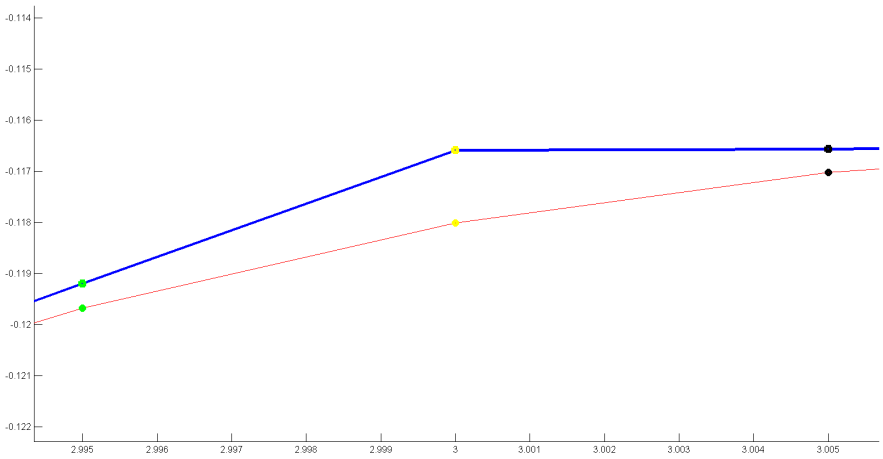


Figure 27: In a zoom from (Fig. 26), 3 'special' points around the right PML-point $a = 3$. On the top (blue line) is a real part of an exact solution (136) with its 3 'special' points, below (red line) is the real part of a numerical solution (137). We see that the result doesn't match perfectly but the error of *O(3)* was considered to be all right. Similar picture is observed if we compare imaginary parts of the result, but even with a less error of *O(4)*.

and could be used as an approximation for the second and fourth order derivatives to calculate the functions that behave according to an evolution equation (such as a one-dimensional *Swift-Hohenberg* or a one-dimensional reduced *Schrödinger* equation) on the plane where a PML of the form (119) was applied. Nevertheless, let's go and see what happens on practice[33].

[33]Writing this last paragraph, I still had no idea that the solution would be suddenly found.

6.2 Linear stability analysis (with PML)

In Chapter 5.2 we have already done a *linear stability analysis* for a one-dimensional *Swift-Hohenberg* equation with *Periodic boundaries*. We have talked about the control parameter r and found its critical value $r_c = 0$, which was kind of a signal for us about the onset of an instability when $r > r_c$. Also we discussed the critical wave number $q_c = 1$, and the growth rate σ (59), but I am going to write the formula for it once again, then we don't need to go back through the pages. So, the *growth rate* could be calculated as

$$\sigma = r - (q-1)^2. \tag{138}$$

But it was right only for that very case. What will happen now, when we have the PML around? How will the relation between the growth rate and the control parameter look like? Because this relation is definitely very important to be found if we need to learn about the critical values. Let's proceed with the analysis in the same way as for a *Swift-Hohenberg* equation with *Periodic boundaries* (Chapter 5.2), but pay attention at what we have to change now, since there is a PML applied, and the problem is extended on the *complex* plane.

Remember in the previous Chapter 6.1 we found interesting transformation of the second and fourth order derivatives (130), when we calculate them on the region with or without PML? But to be honest we have got this rule in a bit experimental way. Though, it could be easily obtained through the more common consideration. We have already done something similar (Chapter 4.3.1) while trying to find the second order spatial derivative, eqs. (14)–(16). We needed this derivative to solve a reduced *Schrödinger* equation. But I should confess that the calculation I made there was quite tricky. So I will take a chance to do it once again in a more clear way and calculate both second and fourth order derivatives needed to solve a *Swift-Hohenberg* equation (62).

To expand the problem to the complex plane, we start with the function $\varphi(x,t)$ and claim that along the curve $z(x)$ (Fig. 25) defined by (119),

$$\Phi\big(z(x),t\big)\big|_{z(x)} = \varphi(x,t). \tag{139}$$

The next step is to take a derivative of $\varphi(x,t)$ once, this will immediately give us

$$\partial_x \varphi(x,t) = z' \partial_z \Phi\Big(z(x),t\Big)\Big|_{z(x)}, \tag{140}$$

but according to the eq. (119), z' is nothing but $z' = i$. Thus, we can write

$$\partial_x \varphi(x,t) = i \partial_z \Phi\Big(z(x),t\Big)\Big|_{z(x)}. \tag{141}$$

Going further and calculating the second order derivative one gets

$$\partial_{xx} \varphi(x,t) = i \cdot z' \partial_{zz} \Phi\Big(z(x),t\Big)\Big|_{z(x)} = -\partial_{zz} \Phi\Big(z(x),t\Big)\Big|_{z(x)}. \tag{142}$$

Thus, when it comes to find fourth order derivative we repeat the procedure two more times and get

$$\partial_{xxxx}\varphi(x,t) = \partial_{zzzz}\Phi\Big(z(x),t\Big)\Big|_{z(x)}. \tag{143}$$

That is it! We have got the same rule again for the 'inside PML' region,

$$\begin{aligned} \partial_{xx} &\to -\partial_{zz}, \\ \partial_{xxxx} &\to \partial_{zzzz}. \end{aligned} \tag{144}$$

But let's return back to the *stability analysis*. Simply knowing now, that inside the PML our second order derivative changes its sign, we can predict that the *stability analysis* will look different for this term. While for the fourth order derivative it will stay the same (144). When proceeding with the *stability analysis* inside the PML one can do it in two different ways. First, we choose the mode (58) that represents the perturbation field $u_p(x,t)$ to have complex coordinates $z(x)$ and look like

$$u_p(z,t) = Ae^{\sigma t}e^{iqz}. \tag{145}$$

Inserting it into (50), one will get

$$\sigma = r - 1 - 2(iq\cdot i)^2 - (iq\cdot i)^4 = r - 1 - 2q^2 - q^4. \tag{146}$$

The same result could be obtained if we use the rule (144), so just change the sign of a second derivative in (58), then the equation will look like

$$\partial_t u_p(z,t) = (r - 1 + 2\cdot\partial_{zz} - \partial_{zzzz})u_p(z,t), \tag{147}$$

where $u_p(z,t)$ is the one from eq. (145). After substitution one will get again

$$\sigma = r - 1 + 2(iq)^2 - (iq)^4 = r - 1 - 2q^2 - q^4, \tag{148}$$

which could be nicely transformed to

$$\sigma = r - (q^2+1)^2. \tag{149}$$

The above equation is a new important relation between *growth rate* σ and the *control parameter* r for a one-dimensional *Swift-Hohenberg* equation on the region with PML (119). We observe that the sign in brackets of (149) has changed, comparing to the result (138) for the region without PML. It certainly means that the critical values for the control parameter and the wave number will change as well. Let's find them.

As before, the form of a mode (145) tells us that the problem is going to stay stable, while

$$\max_q \operatorname{Re}\sigma_q < 0. \tag{150}$$

If we try to find the maximum real value of a growth rate σ dependent on the wave number q, one can see that the second term on the right-hand side of (149)

is always positive and couldn't be turned to zero to let $\sigma = r$. But anyway, we can find its minimal value which is 1, when $q_c = 0$. It gives us

$$\max_q \operatorname{Re} \sigma_q = r - 1. \tag{151}$$

From here and the criteria of stability (150) we can find a critical value for the control parameter,

$$r_c = 1. \tag{152}$$

It means that while $r < 1$ we have a stable solution for a one-dimensional *Swift-Hohenberg* equation inside the PML, but as soon as $r = r_c = 1$ it is the onset of instability.

Recall, that the critical value of a control parameter for the region without PML was $r_c = 0$. One can conclude now, that to satisfy both regions – with and without PML, we can simply choose $r < 0$ for the stability everywhere on the plane, or choose $0 < r < 1$ to get unstable solution on the region in the middle (where there is no PML) and still have stability inside the PML. The last one is what we actually prefer to have there, because we are interested in the *pattern* appearing only inside the PML-free region in between $-a$ and a points. To sum it up, we have found the next critical values

$$r_c = 0 \qquad \text{and} \qquad r_{c_{\text{PML}}} = 1, \tag{153}$$

$$q_c = 1 \qquad \text{and} \qquad q_{c_{\text{PML}}} = 0. \tag{154}$$

Maybe it's worth it to make a plot similar to Fig. 17 to analyse visually the dependence of the *growth rate* σ on the *wave number* q (Fig. 28). I decided to plot it for $r_c = 0.5$, then we observe both – the stability of the modes inside PML, and instability in the region without PML.

6.3 Solving the evolution equations

It knocks you down for a while, when you make a step full of a certain confidence and exciting expectations, but suddenly everything goes the way you wanted the less. Moreover, it goes the way you cannot see and understand clearly, the way you cannot explain mainly to yourself, and there is not a single bright idea to light it up and look around. And it is such a revealing feeling if soon you get a chance or a smart thought how to fix it, but it keeps on pushing on you and promises to explode this tiny aching head and body, if the answer stays unknown, every time when you return to the problem. Honestly, there is a miserable chance that it will let you get rid of constant thoughts about it.

There is this slow bothering feeling that something stays undone. Does it mean that you care? Could it be considered as a positive circumstance? Oh, it is really hard to let it go. But it might be a very important and useful knowledge and ability to know how to let it go. Most likely it does work only in the sense of an ability but not a knowledge, because it asks for a different kind of treatment, if it makes any sense.

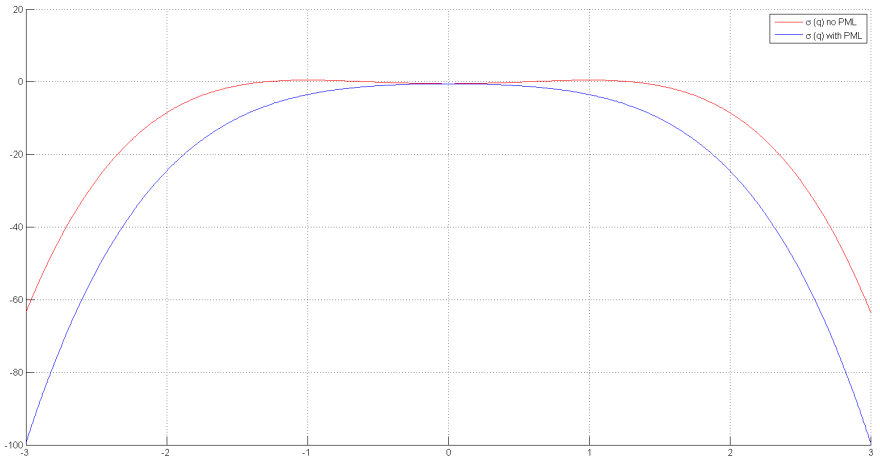


Figure 28: The growth rate σ vs the wave number q for the region without PML (red line (138)) and for the region where PML (119) is presented (blue line (149)). Control parameter $r = 0.5$, and provides instability for the PML-free region around the critical wave-number $q_c = 1$, while the region with PML stays stable, but still has its maximum at the critical wave-number $q_{c\text{PML}} = 0$. Reducing control parameter $r < 0$ will make both lines slide down, and escaping the instability. While setting the control parameter $r > 1$ will lead to the increasing instability, consequently making both lines slide up.

It would be so easy to think that we are close as never to see how the *White noise* (Fig. 19) or any other low-amplitude function will evolve according to a one-dimensional *Swift-Hohenberg* equation (62) on the plane with a PML of type (118). Maybe I expected some possibly appearing errors in my code, but I definitely thought that they would be solvable. I mean, I hoped that if they would appear I would be able to figure out the way to fix them. But at least at that step it was hard to guess that everything could suddenly stop, or to be more precise explode. It wasn't one of the main steps as it seems to me, it took much longer to understand how to apply the right method, how to set a PML, how to create the 'brilliant matrix'. It was more like "now let's apply what we have found" and move to the next, more exciting problem in two- or n-dimensions, let's play with parameters at least. No. Not this time.

In the following small sub-chapters I would present results that were obtained for the reduced *Schrödinger* equation and a one-dimensional *Swift-Hohenberg* equation solved on the plane with PML (119). First of all I wish I could show something that worked nicely. I did it after I had tried the method for the *Swift-Hohenberg*, but it made me feel better, it always does when you see that something goes the way you wish.

6.3.1 PML and the Schrödinger, again

... but much better this time. Not even much better, but probably the way it should be from the very beginning. We have already tried to run our reduced *Schrödinger* equation on the plane with Periodic boundaries, I mentioned it be-

fore (Chapter 5.3). I didn't insert any figures there, but can do it now (Fig. 29). Everything was working perfectly there, 'perfectly' means according to the analytics probably. But also, that test for a *Swift-Hohenberg* equation (Fig. 21) told us that the method and the matrix did work fine. Then, in the very beginning (Chapter 4.3) we have run this reduced *Schrödinger* equation on the plane with a PML applied (Fig. 14). And even though the code *[A]* was a bit tough and irrational, it gave correct results. After all, when I got some simplified formulas to introduce the second order derivative through the matrix M_2 which are discussed thoroughly in the previous Chapter 6.1 and could be generated by the *MATLAB* code *[G]*, it was a real pleasure to see how it works for the *Schrödinger* equation. Even though, or maybe 'especially because', I did it after a huge unexpected fail with the *Swift-Hohenberg*.

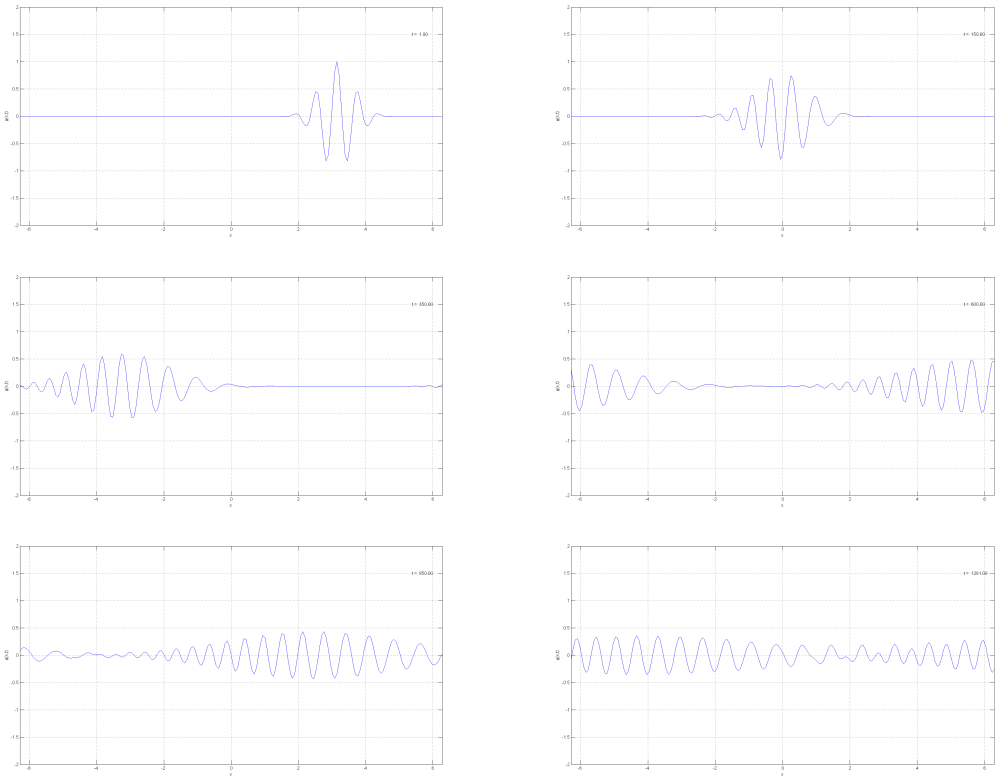


Figure 29: Evolution of a Gaussian function $\varphi = e^{-\gamma(x-x_0)^2}e^{ikx}$ according to a reduced *Shcrödnger* equation (9) on the plane where *Periodic boundaries* are applied. On the series of plots one can see (from left to right and then down) how the wave-packet is moving to the left (wave-number $k = -10$) and slowly dissolves, when it reaches the left boundary it appears from the right-hand side as expected and keeps on moving with a continuous dissolution.

There was nothing so special to do when it came that we wanted to calculate the reduced *Schrödinger* equation (9) on the plane with a PML (119). All we needed was just to make a new discretizastion for z, then generate the matrix M_2 (code *[G]*) which represents the second order derivative and run a new code *[H]*, which is very similar to the previous one (code *[A]*) but much more simplified and elegant, mainly due to a new representation of a *matrix* M_2.

It took less than 5 seconds when a beautiful *wave-packet* appeared and started to move slowly (Fig. 30), like it has nothing to worry about. Like it was on a walk, on one of those days when everything is so quiet and calm, and the harmony swallows you, letting you slide on the ground without really looking down at your feet – they don't need your control, you can just play with you thoughts and ideas now.

But at some magic point a it started to disappear. Like those kids running through the platform $9\frac{3}{4}$, 'it was walking briskly to the barrier he was almost there – and then, quite suddenly, he wasn't anywhere' [23]. But wasn't that what we were looking for? To see how it would disappear.

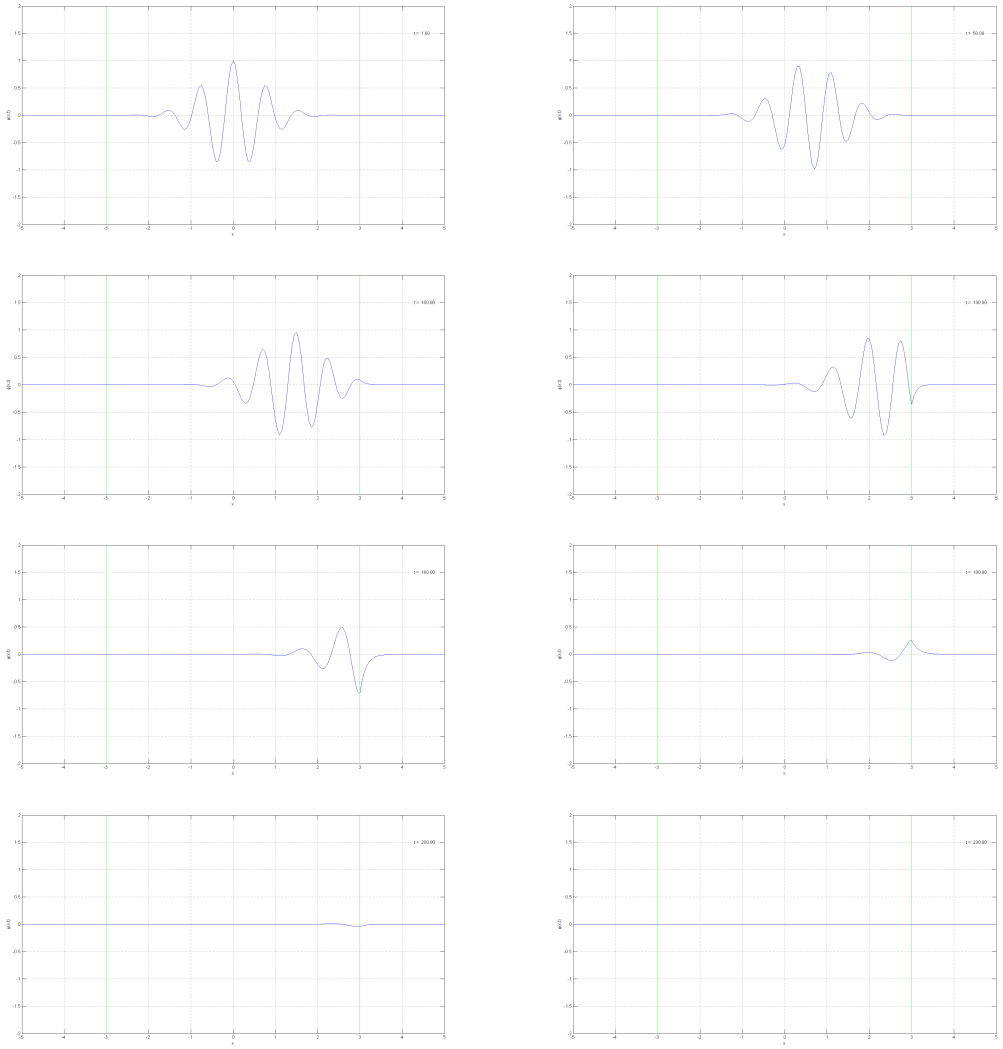


Figure 30: Evolution of a Gaussian function (155) according to a reduced *Shcrödnger* equation (9) on the plane where a PML (119) is applied. On the series of plots one can see (from left to right and then down) how the wave-packet is moving to the right (wave-number $k = 8$), at the point ($a = 3$) where the desired PML (green line) is set, it starts to disappear without any visible reflection back to the PML-free region.

The *wave-packet* on (Fig. 30) is formed by a Gaussian function,

$$\varphi(x,t) = e^{-\gamma(x-x0)^2} e^{ikx}, \tag{155}$$

which moves according to a reduced *Schrödinger* equation (9) on the plane with a PML set at the points $-a$ and a (Fig. 30). This time it takes the *MATLAB* code *[H]* to run really fast. And it is all due to the simplification of the data we have done through the last chapters. Also there is no reflection observed. These facts definitely let us make a reasonable conclusion, that at least the second order spatial derivative on the plane with a PML like (119) could be introduced quite accurately through the matrix M_2 (code *[G]*).

As it comes, some instability appears in the case of a *Swift-Hohenberg* equation (next Chapter 6.3.2). Thus, being pretty much satisfied with a second order derivative represented by the matrix M_2, we suggest that the trouble comes from a numerical approximation of a fourth order spatial derivative and the matrix M_4. There was actually a lot of time and strength spent to find the way to fix this problem or at least to find the reason for it to happen. Let's look one more time at the beautiful *wave-packet* running through the 'platform $9\frac{3}{4}$' (Fig. 30) and turn the page to the 'hopeless case' [24].

6.3.2 The Swift-Hohenberg with PML, fails

... without any obvious reason every time we try to run the code *[I]*, which is basically nothing new but the composition of everything we have already found. Again and again, day after day, we try to figure out the problem, we use all known methods: check the eigenvalues of the matrices M_2 and M_4, change the boundary conditions back to periodic and back again, we expend the possibility of forms that the PML can take through varying θ from (118) and etc. But nothing really helps[34].

But anyway, I think that the results are worth to be introduced here, at least because it really took us a very long time to work it out. All the time then, we were sure that there was no known way for us to solve the problem. So it looked like this.

We have a one-dimensional *Swift-Hohenberg* equation (62) on the plane with the PML (119) at points $-a$ and a, which looks like (Fig. 25). Using our *MATLAB* code *[G]*, we introduce the second and the fourth order spatial derivatives

[34] It is like, you were standing outside on a fresh day, playing music on the street, there were some people who started to gather around you, let's assume because they were interested in the sounds they heard. Interested in how the simplest chords and a confident strumming on that funny ukulele were interacting with your voice that was reflecting from every cold unassailable wall of huge buildings around. Time was passing by, the crowd was getting bigger and bigger, maybe the whole town was there? And at the moment when everyone until the last trouble kid was there, they were ready to... They probably had learned already almost everything about you, while gathering and listening to the alluring sounds, and now they were waiting only for the last very important song or a melody, or a chord. It should sound clearly and leave a certain pattern of interest on their minds or it should sound quietly and slowly fade away making them feel. But suddenly, the strings break down, all four of them! Four string, you hadn't even touch them. Let's see how it could be... and try to find the reason.

from a *Swift-Hohenberg* equation through the matrices M_2 and M_4 (Chapter 6.1). Then we prepare the code *[I]* using all known data and choose a randomly distributed **White noise** signal (Fig. 19) as an initial function. We run the code (Fig. 31).

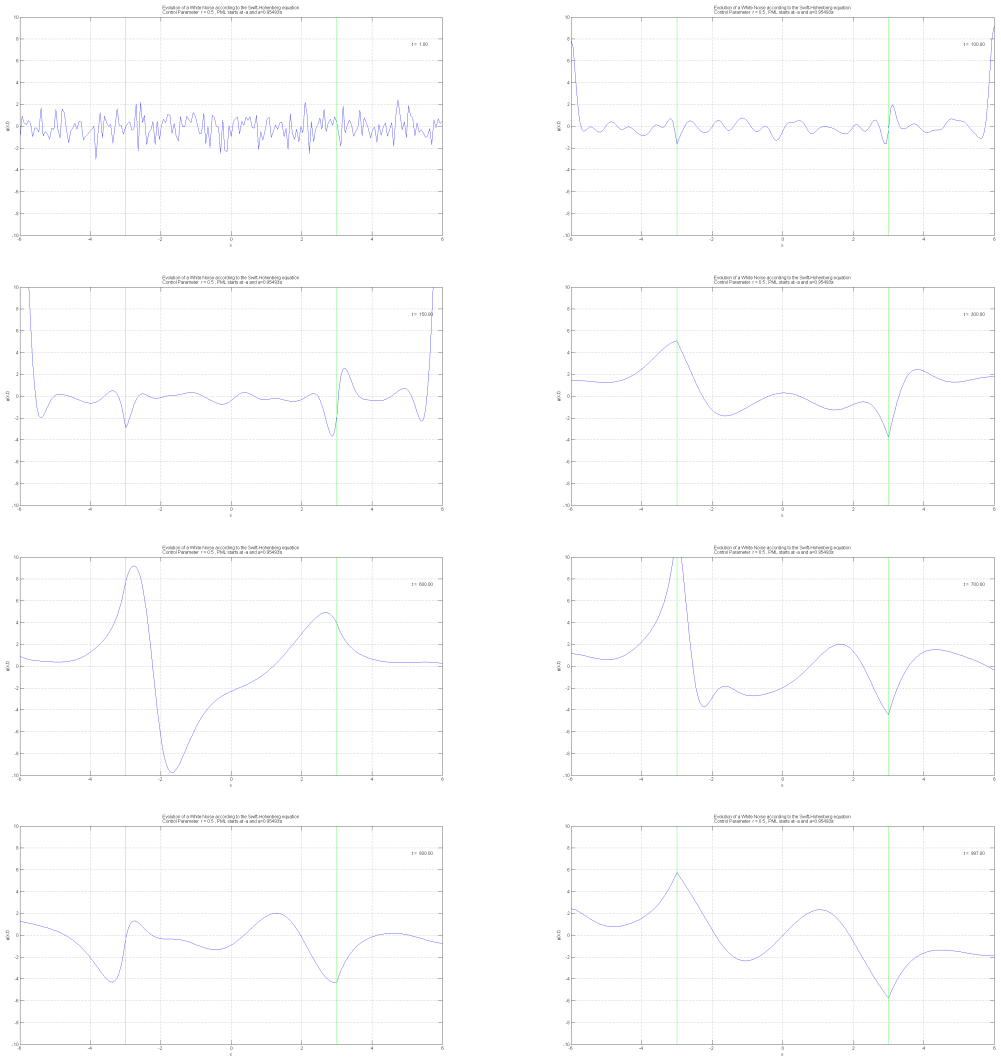


Figure 31: *White noise* (Fig. 19) that evolves according to the *Swift-Hohenberg* equation (62) on the plane with the PML (119) at points $-a$ and a, which looks like (Fig. 25). *Control parameter* $r = 0.5$ that according to (153) leads to the instability on the PML-free region (between the green lines) and to stable modes inside the PML. It is important to mention that the PML-points $a \approx \pi$ here. One can see (reading from left to right and then down) the evolution of the *White noise*, but also it becomes obvious that the function stays unstable everywhere. Thus, the predicted result doesn't appear to be true.

First time we have run the system with control parameter $r = 0.5$ (Fig. 31). Looking for any kind of a hope, we would set the control parameter to $r < 0$ now. It should obviously give us stable results everywhere, it should make the *White noise* signal vanish away slowly everywhere on the plane. Let's see what we actually get (Fig. 32).

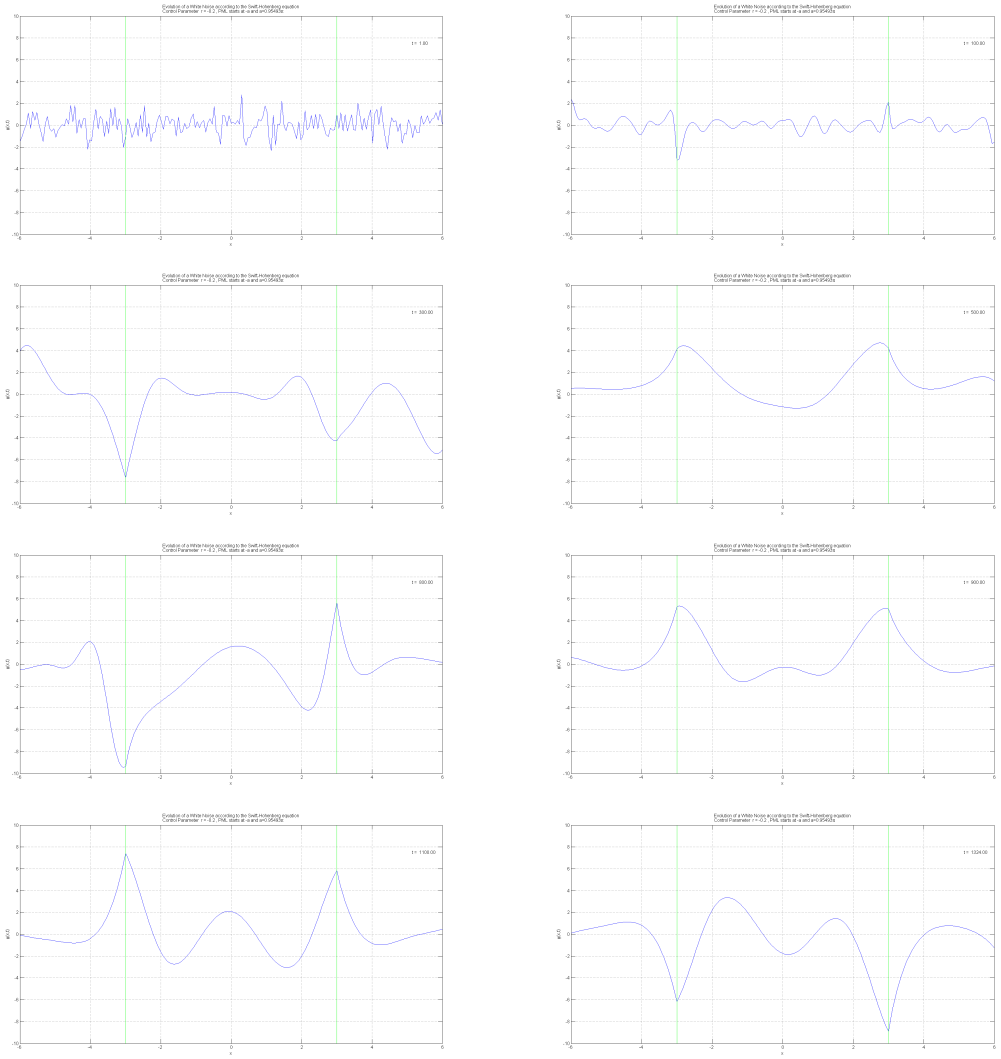


Figure 32: *White noise* (Fig. 19) that evolves according to the *Swift-Hohenberg* equation (62) on the plane with the PML (119) at points $-a$ and a, which looks like (Fig. 25). *Control parameter* is set to be $r < 0$ here, that according to (153) promises us stability everywhere on the plane. It is important to mention that the PML-points $a \approx \pi$ here (green lines denote where the PML starts). One can see (reading from left to right and then down) the evolution of the *White noise*, though it becomes obvious that the function doesn't vanish away as it should, quite the reverse it becomes unstable everywhere.

We have been working with these results (Fig. 31, Fig. 32) for some really long time I would say. The sudden instability was considered as a contradiction of our analytical results and numerical calculations. We tried to figure out where it comes from and the fact that it worked perfectly for the reduced *Schrödinger* equation made us believe that all the troubles are due to the fourth order derivative. Because in the reduced *Schrödinger* equation (9) we have only the second spatial derivative and as it's shown in the Chapter 6.3.1 there were no problem in applying obtained PML.

We had gone deep into considerations of how the appeared failure could be explained by any means of the theory. Meanwhile, everything needed was right

on the surface. But it is often too hard to notice small things on the tip of your nose and we keep on looking under a table or a bed. And it takes a lot of time of course, no doubts that there are so many things to find there and they could be useful or fun, but rarely they would solve your problem. Is it that we just wait for the right moment when we can finally discover this small simple "a" that is going to fix everything?

7 PML and the Swift-Hohenberg equation

Actually, I am very happy to have this chapter the way it looks now and not the other. The most interesting fact here, is that when writing my *Thesis*, all the way until the previous paragraph I had no idea that this current chapter will take place. It could be probably seen pretty clear through the slightly pessimistic moods and hopeless plots only one page back.

I decided not to change it, because it wasn't something that had appeared just for a while. Honestly, just a day ago or so it would be the end of the *Thesis*, with the exception of few more chapters that should present our desperate attempts to prove the appropriateness of these exploding figures as well as making ourselves believing that this not comforting result can also take place.

It is even funny how suddenly and unexpectedly everything can change, how a little thought, that comes just before you fall asleep could make things go so differently in the morning. Now everything seems to be so obvious and clear, there is a certain lack of understanding about how it could stay invisible from you all the time. There is a pinch of a sadness that burns with regrets of how everything could go if you had found it some months ago.

Of course, if this had appeared earlier in the research we made, "we could then have pushed our investigations into new pristine territory. However, this is not what occurred and we must accept to this". Moreover, we should write as much as possible about it now, because it can be the very last step for this *Thesis* but the very first one for someone who will take a chance to investigate it further on. Thus, I let myself to go back to the work and explain everything step by step.

7.1 There are still some dissensions

If we page back to the Chapter 5.2 and read the caption (Fig. 17) in the very end of it, in the very last sentence, we would find the remark that gave us a chance to find the 'mistake' if it could be called this way. It says, that when the *stability analysis* is done, we expect some set of *Fourier modes* close to the critical wave-number $q_c = 1$ to grow. It also means, that the *pattern* should appear. More precisely, the growth of certain *Fourier modes* close to the critical wave-number $q_c = 1$ predicts the appearance of a *pattern* with a characteristic length scale $2\pi/q_c$. Let's read it one more time,

$$\text{characteristic length scale for a pattern} = \frac{2\pi}{q_c}. \tag{156}$$

I could clearly remember that all the time I'd run the code *[I]*, PML-points $-a$ and a were set to $a = 3$. I had chosen it once as well as the number of spatial steps N and M to define the spatial-grid, and never bothered to change it. But obviously, it means that the PML-free region, where we had expected to observe a pattern formation had the length from $-a$ to a, which is $l = 6$. It turns to be less than the characteristic length scale for a pattern formation from (156), because on the PML-free region according to the stability analysis $q_c = 1$. So, all we needed to do was just to expend the plane of the observations. It is truly unbelievable how suddenly all of it could go so nicely, and just the way it has been derived analytically.

So, to do what we have just found, one has to change the value of the PML-point a in the code *[I]* for a bigger one. Once this is done, we can run the code and enjoy a pattern-like picture and check everything predicted by the theory while varying the *control parameter* r and turning the *non-linear term* on and off. All the 'beautiful' results are shown in the next series of figures (Fig. 35 – 37), but for a while we let them wait.

As it appears, after a more careful examination, it is not only a bigger value of a PML-point that fixes everything, the spatial step dx does also play an important role. It takes some time to check all the possible combinations, because there are few terms we can change: the control parameter r, the spatial step dx and the value of a PML-point a. Somehow, after several tries certain conclusions could be made.

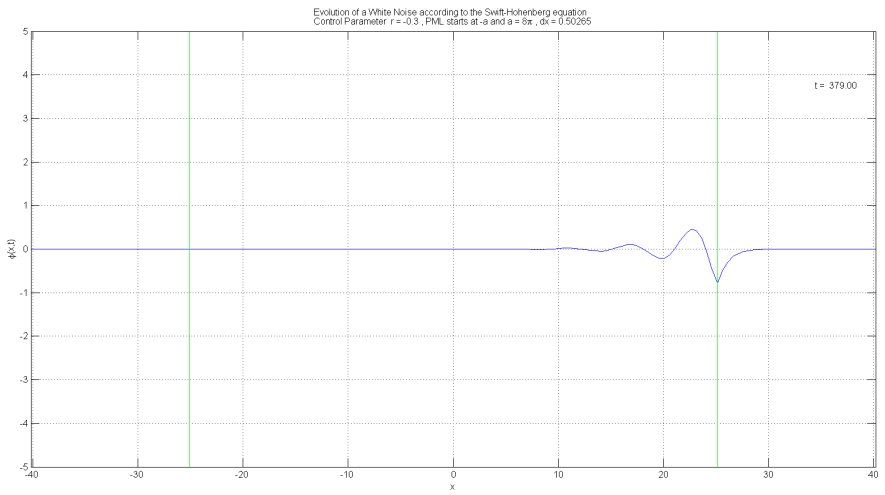


Figure 33: The result (last step) of running a *Swift-Hohenberg* equation (62) on the plane with a PML (119), when a *White noise* signal (Fig. 19) is used as an initial function. Here the control parameter is chosen to be $r = -0.3$. Thus, according to the theory (153), all the amplitudes should fade away. Still we can see some disturbance on the right. As the experiment shows, there is almost always some kind of a different disturbance near the PML-boarder (green lines) or on the edges, when the control parameter $-0.3 \leqslant r \leqslant 0$, even if the PML-point is set $a \geqslant 2\pi$.

First of all, there is still something that 'breaks' results when we run the system with a control parameter $-0.3 < r < 0$, (Fig. 33). According to the

stability analysis (153), with $r < 0$ we should have all the amplitudes fading away everywhere on the plane, somehow there is still something left all the time (Fig. 33) unless we decrease the value of a control parameter even more. Nevertheless, for all the other values of r it seems to work nice.

The second issue, is the value of a spatial step dx. It turns out that if dx is really small, then we cannot save the situation even with a huge values of a. We have already mentioned that it is possible to get correct results only if $a \geqslant 2\pi$, thus I tried to find kind of a critical value for the spatial step dx_c, when $a = 2\pi$. It appears to be, that $dx_c \approx 0.3$ when the PML-point is set to $a = 2\pi$ (Fig. 34). The value is approximate because we examine a signal, the *White noise*, which is always **randomly** distributed on the plane (Fig. 19). So, even with a prescribed behaviour by a *Swift-Hohenberg* equation, no two experiments are alike.

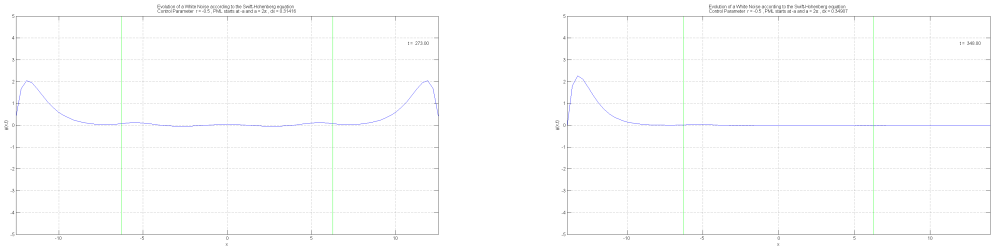


Figure 34: Two different disturbances that appear when running a *Swift-Hohenberg* equation (62) on the plane with a PML (119), when a *White noise* signal (Fig. 19) is used as an initial function. These disturbances appear as a mistake and a disagreement with a theory when the spatial step is chosen to be $dx \lesssim 0.3$. Here the control parameter is set to $r = -0.5$. According to the theory (153) and our careful examination, all the amplitudes usually fade away for this value. But since we set $dx = \pi/10 \approx 0.3142$, we can see that some disturbances do arise. As our experiment shows, there is almost always some kind of a different disturbance near the PML-boarder (green lines) or on the edges when the spatial step is set $dx \lesssim 0.3$.

Also when we increase the value of a PML-point, we see that the *critical value* for the spatial step begins to decrease, but not that much. So still, for the correct results it is preferably to use[35] $dx > 0.3$ and $a > 2\pi$. Maybe it is useful to make a little table to introduce what we have just discussed.

$a < 2\pi$	results are unstable and incorrect (Fig. 31–32)
$a = 2\pi$	set $dx > 0.3$ for the correct results (Fig. 34)
$a > 2\pi$	dx_c could be different now, and depends on a
set $a > 2\pi$ and $dx > 0.3$ for the correct results (Fig. 35 – 37).	

Now, after all the dissensions with the theory are up on the surface, we can use parameters that will give us the most close to the truth (and to the theory) solution of the problem. Since all these results are quite new for me, and have appeared just some days ago, though the *Thesis* has to be finished very soon, I don't try now to explain them or to fix the problem and make it work perfectly

[35] It has to be mentioned, that in the code *[I]* we set the spatial step as $dx = a/M$, where M is the number of points we want to have on the grid from the origin of coordinates to the PML-point a. Thus, the value of the spatial step dx is always set by this fraction.

in accordance with the analytics. But I would suggest it to be the next step in any further investigations, if one is about to start them.

7.2 'Patterns' do appear!

It seems to be the last chapter, that mostly will consists out of figures and plots presenting the evolution of a *White noise* signal (Fig. 19) according to a *Swift-Hohenberg* equation (62) on the plane where PML of type (119) has been applied. I guess, I have written the above sentence so many times already, and every time before it was just something incomplete, something that would need an improvement or a reconsideration, but now it is more or less what we can call the answer! Though, this kind of an answer that needs not to be kept untouched but has to be developed and upgraded to the next level. I mean, that it has to be developed, but not only around itself, working on this result could definitely turn it into something else, into something even more exciting.

Using the matrices calculated in Chapter 6.1 and presented by the code *[G]*, using the right (suitable) configuration of parameters, which were discussed in a previous Chapter 7.1, we can run the code *[I]* and get nice pictures. Also I am not going to comment all the figures through the main text here, but rather leave as much comments as possible in the caption for each. By the way, there is going to be *three* of them, according to the possible *three* cases for the control parameter (153) from the *stability analysis* (Chapter 6.2):

$$\begin{aligned} r < 0 \quad &\Rightarrow \quad \text{stability everywhere (all the amplitudes fade away),} \\ 0 < r < 1 \quad &\Rightarrow \quad \text{unstable on a PML-free region, stable inside PML,} \\ r > 1 \quad &\Rightarrow \quad \text{unstable everywhere on the plane.} \end{aligned} \tag{157}$$

It was really a pleasure to see the pattern-like picture appearing in the middle of the plots on the PML-free region, just as predicted by the theory (Fig. 35). Or to see how the function inside the PML couldn't calm down for $r > 1$, while in the middle everything looked still (Fig. 37). At the same time, the case with 'everywhere dissolution' for $r < 0$ (Fig. 36) brought more problems that had been expected and which still have to be studied more thoroughly. These were the most interesting and exciting moments about running the prepared code and examining the results. I hope, that the explanations below the figures are full and valuable enough to tell as much as possible on the results obtained. From here I let the figures 'speak for themselves'.

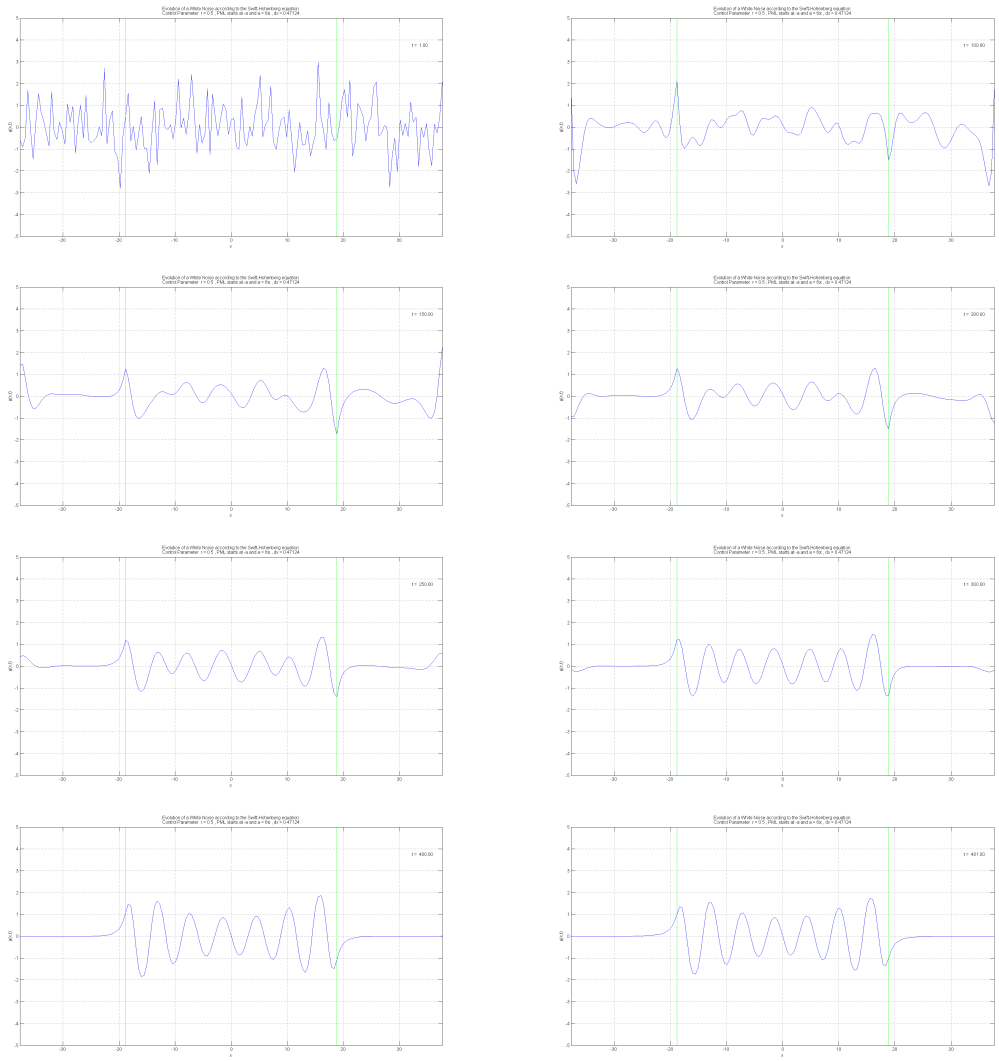


Figure 35: The first and the most 'classical' case (in our terms) that we would like to observe after a long working process, and probably the most important one. When the control parameter lies between $0 < r < 1$ which gives us an instability on the PML-free region (between green lines) but let everything to fade away inside the PML (153), these conclusions are based on the *stability analysis* we did before (Chapter 6.2). On the series of plots (reading from left to right and then down) we see how the *White noise* signal (Fig. 19) evolves according to a *Swift-Hohenberg* equation (62) on the plane with a PML (119) at points $-a$ and a. The title of every plot (if one zooms the page) claims that the PML-point is set $a = 6\pi$, the control parameter $r = 0.5$, and the spatial step occurs to be $dx \approx 0.47124$ (these parameters were chosen in accordance with the remarks we made through the Chapter 7.1). Also the 'endless' growth of the instability is controlled by a non-linear term u^3 from (62). Thus, we obtain a really nice picture with a pattern-like formation in the middle, between the PML-starting points.

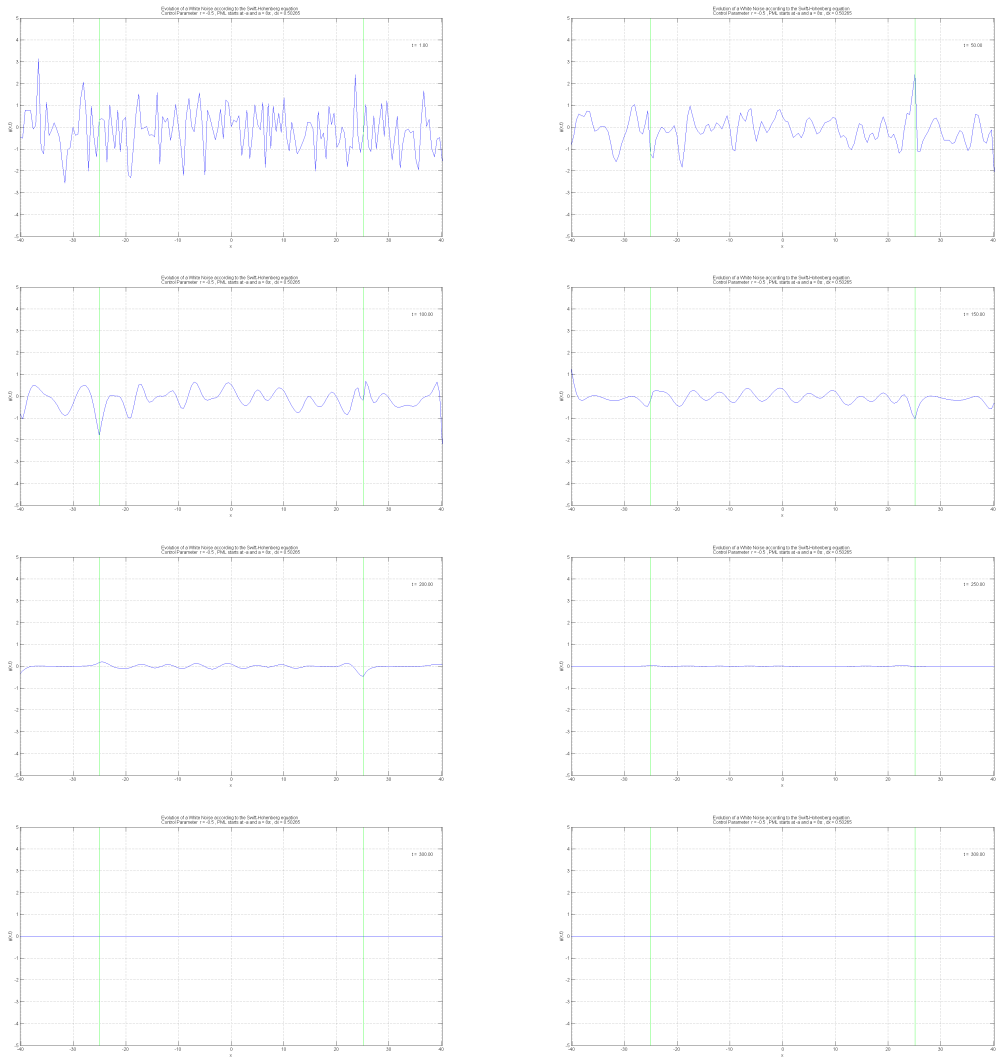


Figure 36: The second and, as could seem, the most boring case, with the value of a control parameter $r < 0$ that has to provide stable solutions everywhere, that should make all the amplitudes fade away slowly. But if we recall what is said in Chapter 7.1 and look once again on the Fig. 33, we will understand that this case turns to be more or less tricky and incomplete. The control parameter has to be chosen $r < -0.3$, if we want to get the correct, by all means, results. On the series of plots (reading from left to right and then down) we see how the *White noise* signal (Fig. 19) evolves according to a *Swift-Hohenberg* equation (62) on the plane with a PML (119) at points $-a$ and a. I have chosen different values for the PML-points here from the previous case, now $a = 8\pi$, the spatial step occurs to be $dx \approx 0.50265$, and the most important is the value for the control parameter, which is set to be $r = -0.5$ (these parameters were chosen in accordance with the remarks we made through the Chapter 7.1). Thus, when the 'special' parameters are set we observe how all the low amplitudes of a *White noise* signal dissolve in time and fade away.

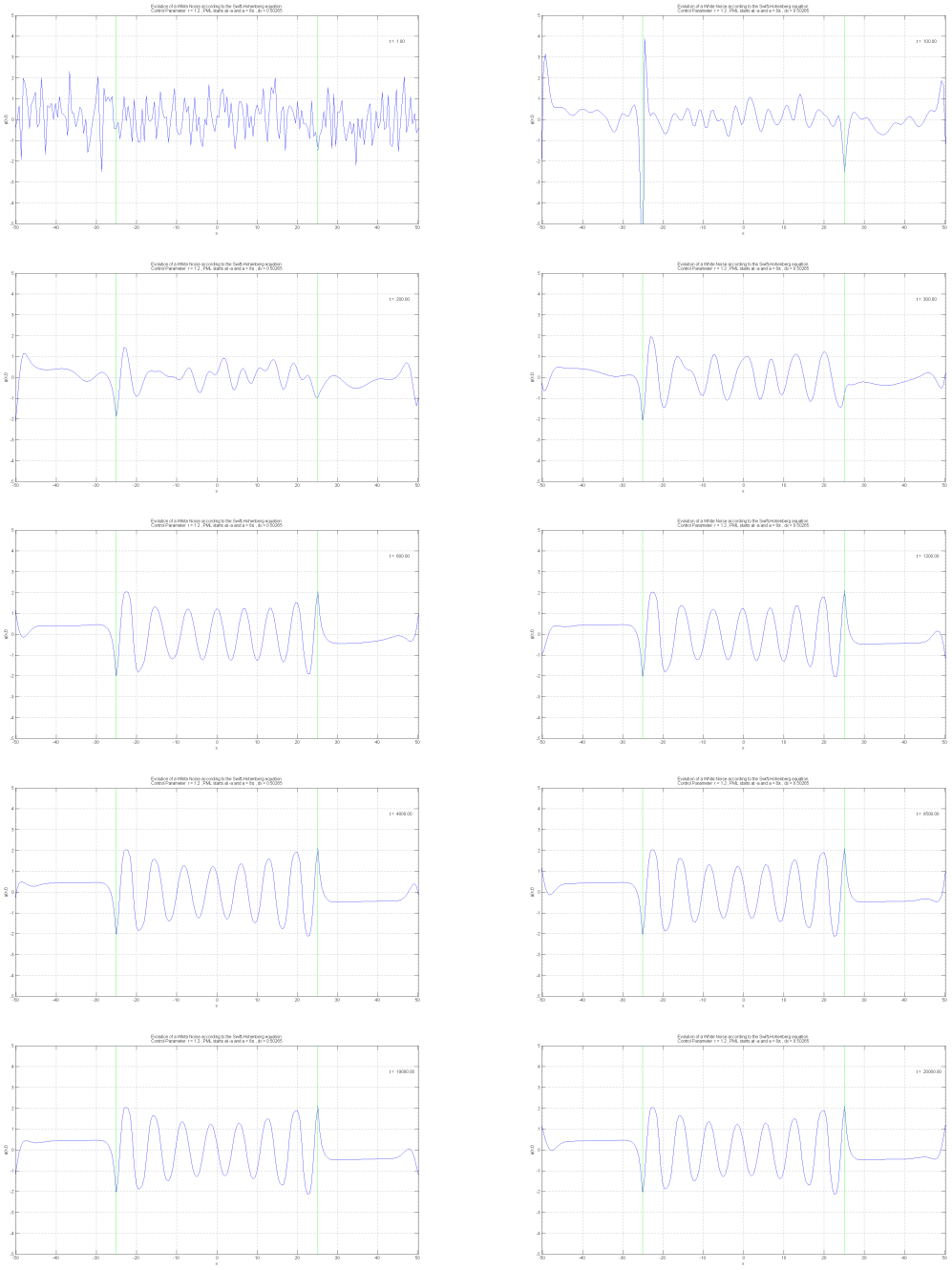


Figure 37: The third, the last, and the most exciting case, I would say. The control parameter is set $r > 1$, which is greater then both critical values r_c for the region with and without PML (153). It should give us instability (but controlled by the non-linear term u^3) on the PML-free middle (between the green lines) and some kind of an instability inside the PML as well (Chapter 6.2). And that's is exactly what we see on the series of plots (reading from left to right and then down). The *White noise* signal (Fig. 19) evolves according to a *Swift-Hohenberg* equation (62) on the plane with a PML (119) at points $-a$ and a. After some period of time we see how a pattern-like picture sets up in the middle (PML-free), but inside the PML-region nothing calms down, even if running the code for a really long time. The parameters chosen here (as it could be seen from the title to every plot) are: PML starts at $a = 8\pi$, the control parameter is $r = 1.2$, and the spatial step occurs to be $dx \approx 0.50265$ (these parameters were chosen in accordance with the remarks we made through the Chapter 7.1).

8 Conclusion

I remember after a series of failures, while working on the very first problem of my *Thesis* – a reduced *Schrödinger* equation on the plane with a *Perfectly Matched Layer (PML)* set on; I was asked to do some other tasks, which were definitely a bit aside from the main course of our investigations, though maybe they were supposed to switch my attention for a moment. Back then, I could not really guess if they would be of any help later, or might let me see some similarities, or whatever else.

Of course, I did all the tasks, which took me about a week or so. But in the very end, we were back to the point I had been stuck at before. I gave it a try and asked, for what reason we had done all those complicated and tricky things, if now we have returned to the same exact place in our research we did stand on in the beginning. The answer was – *'Oh! You know, before doing something you have to talk to people about complex analysis and etc. It is just good to know the history behind the question, to go through something tough before you can really evaluate the usefulness and simplicity of the other methods'.*

So, even if it seems to be only about getting the results, it is actually not. I want to believe, that in its origin *this work* is actually more about the process: of getting an experience, of exploring the inner and the outer places, of travelling backward and forward, of dreaming and thinking – while building a personal understanding of the whole thing.

From the first to the last chapter of this *Thesis* we were going through the steps that let us in the very end, even though quite unexpectedly, to build a system with a *Perfectly Matched Layer (PML)* of a special form that could be described as a *vertical reflectionless* boundary on the plane.

Two *evolution* equations on this plane with *PML* have been considered. First, a reduced *Schrödinger* equation with a Gaussian function as an initial signal, which let us observe a *wave-packet* moving along the x-axis and being totally absorbed by the *PML*. Second, and the most interesting case, was a one-dimensional *Swift-Hohenberg* equation with a randomly distributed *White noise* signal in the role of an initial function. In our experiments with a *Swift-Hohenberg* equation we could see how a randomly distributed signal would start fading away everywhere on the plane, but at some moment, depending on the chosen parameters, it might suddenly grow and form a simple but nice kind of a *pattern*, or it could continue on vanishing away.

There are lots of interesting, amazing, inspiring aspects appearing in the *Thesis* that you can pay attention to. One of them, is the *vertical* boundaries we have set with a *PML*, they are something like a wall on one's way, though turn out to be magically *reflectionless*, which let us observe *Patterns*, that the *evolution* equation is supposed to produce, as they are. Another thing, is my favourite, about how the *evolution* equation as a *Swift-Hohenberg* equation, with its *control parameter* and non-linear *stability* terms, can make a similar *Pattern* appear every time almost from nothing, from a randomly distributed signal, like a *White noise*. It seems to have a lot in common with all those great unbelievable things happening in the *Universe*, on the huge scales; or here around us, on the

scales so much smaller. It makes me think about the *chaos* and the *order* around and inside of us, or maybe around and inside of everything. It also brings us back to those pictures of very similar *Patterns* found in absolutely different, as we would think, *natural systems*.

We have studied and talked about plenty of *analytical* and *numerical methods* in this work. Our exploration starts with the experiments and systems that are a certain simplification of the real problem and the goal we set. Thus, a reduced *Schrödinger* equation was studied first, it has only a second order spatial derivative. Then, we took some time creating a method to calculate spatial derivatives, using *Taylor Expansion* (along with a *Finite Difference Method*), and based on building a *matrix* in *MATLAB*. We did it first of all for the system with *Periodic Boundaries*, and only after we were satisfied with the results, we moved further and implemented a *Perfectly Matched Layer (PML)*.

The results that we have got for a one-dimensional *Swift-Hohenberg* equation, when running it on the plane with a desired type of a *Perfectly Matched Layer (PML)* set, and a randomly distributed *White noise* signal chosen as an initial function, are assumed to be absolutely new, if we talk about the experiment and its application. These results are meant to give us a possibility to observe a *pattern formation*, produced by an evolution *Swift-Hohenberg* equation, the way it should appear on the *infinite domain* originally, but using an actual *finite* one, avoiding all the reflections that cause instabilities and destroy the real picture.

From the other hand, I can call results that were obtained some sort of *'new'*, just because we have discovered and presented them the way they are, only in the very end of the *Thesis*. It just happened that we got them, when already being in the finish. Maybe, it makes it even more exciting, to have something 'fresh' half a step before the conclusion. Also it makes you feel as if the problem was sort of 'alive'. Because the question to study and the further steps to go are lying right on the surface now, and they just wait for someone to keep on with the investigations. Only expanding the problem to a two-dimensional (spatially) case should already give us those wonderful pattern-like plots (as *hexagons* and *convection rolls*). I would say, even if it feels for a moment that we have reached the finish line, actually we have only clambered up the first step. It was a tough adventure, nevertheless, now we know that this 'step' is stable enough to let us proceed with developing our theory and push the investigations to the next level.

In the very beginning, almost everything was absolutely unknown to me, and of course, the goal we set for the *Thesis* at first, is still so far from the 'step' we are safely sitting on now. Although, I think that setting the expectations somewhere to a further place, than the one you can actually reach, is not a bad practice, especially if you kind of know it from the moment you have started, or maybe, get some understanding while working on the problem. If you do the thing carefully, trying not to miss any part that could be important and making the issues clear first of all for yourself, then almost always, it takes at least double time. But hopefully, it is the way it should be, only being on this road you might have a chance to get some important answers, it is more about the process, as I have told already. Process of learning, exploring, thinking and dreaming.

***

So, is it the end? You catch a snowflake and learn something from it just before it melts away, or you get some magical feeling watching its amazing pattern for those few seconds that it rests on your mitten. You will never forget the chance that you got and everything that happened. But then, there is always something next to do.

It is incredibly wonderful how the things you start, your *'projects'*, so different in their nature, would intertwine from time to time, while you keep on doing both. How the memories and ideas from one of them would affect the results of another, and how the inspiration for the first appears while working hard on the second. But of course, numbering here makes no sense, as well as the number of these *'projects'*. It is all about getting something done and learning new things along with getting the problems solved. Because they have to be, or because you want them to be solved. Quite often these two aspect do also appear to be one. Meanwhile, having fun is not the last thing here. One has to enjoy the process otherwise, it won't give you the real true outcome.

I bet, we never know for sure when exactly something will strike out and we will get this beautiful or inspiring 'answer'. Maybe, one day after two weeks of hard working on a Thesis, 'hiding' on the farm not far from *Oslo*, you will find a *caravan* standing next to the place where you currently live, and you will go there and spend the night doing a sudden music recording,

About Sun = https://soundcloud.com/andrew-antrushin/about-sun (158)

Being so not sure how it happens, I obviously have a strong feeling as if it was all about the same. The patterns and the equations, the patterns and the songs. Your projects and your dreams. The way how one turns to be the other equally. The moment when something appears out of nothing, or a second when everything fades away for a while.

It all turns to be a part of a 'bigger', but gracefully bears its own face and ignites this pleasant understanding, that you do something you belong to. And it helps a lot. It is needed.

This way we are able to find and to see the amazing connections: while being impressed by a snowflake that landed on your mitten, or thrilled by a spiral galaxy, that could be actually observed even situating millions of light years away from us. And then the moment comes, when patterns appear from equations, when equations become a song (158), when everything suddenly goes the way it has been always going, but now, after getting rid of some disturbing *reflections*, you just see it more clearly.

Bibliography

[1] M. Cross and H. Greenside. *Pattern Formation and Dynamics in Nonequilibrium Systems*. Cambridge University Press, United Kingdom, 2009.

[2] A. M. Turing. The Chemical Basis of Morphogenesis. *Philosophical Transactions of the Royal Society of London*, 237(641):37–72, Aug. 14 1952.

[3] BBC Four. *The Secret Life of Chaos*. Science Documentary Film, 2010. Duration: 59 min.

[4] Andrey Antrushin. Dark Energy and Dark Matter in the Universe. Axion as a particle of a Dark Matter. *Bachelor's thesis in Physics*, Northern (Arctic) Federal University named after M.V. Lomonosov. Arkhangelsk, 2011.

[5] Timothy Sauer. *Numerical Analysis*. Pearson, 2006.

[6] S. Sánchez Pérez-Moreno, S. Ruiz Chavarría, and G. Ruiz Chavarría. *Experimental and Computational Fluid Mechanics*, chapter Numerical Solution of the Swift-Hohenberg Equation, pages 409–416. Springer, 2014.

[7] *Fourier Transform tutorial*, 2010. http://www.thefouriertransform.com/.

[8] An article from Wikipedia. *White noise*. https://en.wikipedia.org/wiki/White_noise.

[9] Malin Kivelä. *Du eller aldrig*. Schildts & Söderströms, 2006.

[10] Janet Scheel, Keng-Hwee Chiam, Mark Paul, Henry Greenside, and Paul Fischer. Pattern Formation and Dynamics in Rayleigh-Bénard Convection. *Theoretical Aspects of Pattern Formation, Surrey*, 2005.

[11] J. Liu, K.M.S. Bajaj, and G. Ahlers. Images of Spatio-Temporal Chaos. Unpublished.

[12] Rebecca Hoyle. *Pattern Formation An Introduction to Methods*. Cambridge University Press, New York, 2006.

[13] Nathan Tompkins, Ning Li, Camille Girabawe, Michael Heymann, G. Bard Ermentrout, Irving R. Epstein, and Seth Fradena. Testing Turing's theory of morphogenesis in chemical cells. *Proceedings of the National Academy of Sciences of the United States of America(PNAS)*, 111(12):4397–4402, March 10 2014. DOI: 10.1073/pnas.1322005111.

[14] Cheng Li. *Mock Galaxy Redshift Surveys for SDSS and LAMOST*, 2007. http://wwwmpa.mpa-garching.mpg.de/~leech/research/clustering/main.php.

[15] Frank D. Hastings, John B. Schneider, and Shira L. Broschat. *Application of the perfectly matched layer (PML) absorbing boundary condition to elastic wave propagation*. School of Electrical Engineering and Computer Science. Washington State University, 1995.

[16] Frédéric Nataf. *Absorbing boundary conditions and perfectly matched layers in wave propagation problems*. Direct and Inverse problems in Wave Propagation and Applications, pages 219–231. Radon Ser. Comput. Appl. Math., 2013.

[17] Steven G. Johnson. *Notes on Perfectly Matched Layers (PMLs)*, August 2007. http://math.mit.edu/~stevenj/18.369/spring07/pml.pdf.

[18] Carlos Castaneda. *The Teachings of Don Juan: A Yaqui Way of Knowledge*. University of California Press, 1968.

[19] J.A. Walker. *Dynamical systems and evolution equations*. Plenum, 1980.

[20] *MathWorks Documentation*. Online at: http://se.mathworks.com/help/matlab/index.html.

[21] An article from Wikipedia. *Finite difference coefficient*. https://en.wikipedia.org/wiki/Finite_difference_coefficient.

[22] Howard Anton and Chris Rorres. *Elementary Linear Algebra: Applications Version*. Wiley, 11 edition, 2013.

[23] J. K. Rowling. *Harry Potter and the Philosopher's Stone*, chapter 6: The Journey From Platform Nine And Three-Quarters, pages 68–81. Scholastic, 1998.

[24] John Boyne. *The Boy in the Striped Pyjamas*. Black Swan, 2006.

Appendices

A The Schrödinger equation with PML (my first try)

```
% First attempt of making a Gaussian function evolve
% according to a reduced Schrodinger equation on the plane with PML.
% The code is quite tough, mainly due to using the matrix
% for the ODE-solver from the function odePML.m which is not simplified

clear all; clc;
tic; % Start timer

% define the spatial grid
M = 100; % number of points at (-a,a)
N = 200; % number of points at (-L,L)
a = 4; % the value of PML-point
dx = a/M; %define the step (to make sure that a PML-point is on the grid)
L = N*dx; % the value of the point where the lateral boundary is situated
n = 2*N+1; % the whloe number of grid points from '-L' to 'L'
x = linspace(-L,L,n); % define x-grid

% discretize z in accordance with chosen PML
% z = a + if(x) = a + exp(i*tetta)(x-a)
ar = N+1 + M; % number of the righ PML-point
al = N+1 - M; % number of the left PML-point
aR = x(ar); % value of the right PML-point
aL = x(al); % value of the left PML-point

theta = pi/2; % set incline 'theta'
z(1:al) = aL + exp(1j*theta)*(x(1:al) - aL); % left PML
z(al+1:ar-1) = x(al+1:ar-1); % middle without PML
z(ar:n) = aR + exp(1j*theta)*(x(ar:n) - aR); % right PML

% define time-grid
dt = 0.03;
T = 3;
m = T/dt;
t = linspace(0,T,m);

% generate a Gaussian function
% phi = exp(-gamma*(x - x0 -/+ C*t)^P)*exp(1j*K*x);
% parameters
C = 1;
gamma = 0.8; % smaller gamma -> wider wavepacket!!!
K = -6; % bigger wave number more waves in a wavepacket!!! and guess it
    also moves faster
P = 2; % should be even!!!
x0 = -1;
% initial function
phi0 = exp(-gamma*(x - x0).^P).*exp(1j*K.*x);
phi0(1) = 0;
phi0(n) = 0;

```

```
% finite difference (from expansion) to introduce second order derivative
num_dxx = zeros(1,n);
for i = 2:n-1
     num_dxx(i) = -2*(-z(i)*phi0(i-1) + z(i+1)*phi0(i-1) +...
         z(i-1)*phi0(i) - z(i+1)*phi0(i) - z(i-1)*phi0(i+1) +...
           z(i)*phi0(i+1))/((z(i-1) - z(i))*(z(i-1) - z(i+1))*(z(i) - z(i
              +1)));
end

% create a function for the ode-solver and calculate the result
f = @(tt,phi) odePML(tt,phi,n,z);
[tt, Phi] = ode23tb(f,t,phi0);

toc; % Stop timer

% plot the results
figure(1)
for k = 1:m
    drawnow;
    plot(x,real(Phi(k,:)));%, x, 1, 'g', x, -1, 'g');
    ymin = -T;
    ymax = -ymin;
    axis([-L L ymin ymax]);
    title('WavePacket moving with the PML');
    line([a,a],[ymin,ymax],'Color','g');
    line([-a,-a],[ymin,ymax],'Color','g');
    xlabel('x');
    ylabel('\Phi(x)');
    S1=sprintf('t =  %.2f',k);
    text(L/1.8,ymax/1.8,S1);
end
```

A.1 Function 'odePML.m' for the main code *[A]*

```
function dPhidt = odePML(~,phi,n,z)

dPhidt = zeros(n,1);
a2 = zeros(n,1);

for i = 3:n-2
    a2(i) = (phi(i-2)*(z(i-1)*z(i+1) - 2*z(i)*z(i+1) - 2*z(i)*z(i+2) -...
    2*z(i)*z(i-1) + z(i-1)*z(i+2) + z(i+1)*z(i+2) + 3*z(i)^2))/((z(i)
        -...
    z(i-2))*(z(i-1) - z(i-2))*(z(i-2) - z(i+1))*(z(i-2) - z(i+2))) -...
    (phi(i-1)*(z(i-2)*z(i+1) - 2*z(i)*z(i+1) - 2*z(i)*z(i+2) -...
    2*z(i)*z(i-2) + z(i-2)*z(i+2) + z(i+1)*z(i+2) + 3*z(i)^2))/((z(i)
        -...
    z(i-1))*(z(i-1) - z(i-2))*(z(i-1) - z(i+1))*(z(i-1) - z(i+2))) -...
    (phi(i+1)*(z(i-1)*z(i-2) - 2*z(i)*z(i-2) - 2*z(i)*z(i+2) -...
    2*z(i)*z(i-1) + z(i-1)*z(i+2) + z(i-2)*z(i+2) + 3*z(i)^2))/((z(i)
        -...
    z(i+1))*(z(i-1) - z(i+1))*(z(i-2) - z(i+1))*(z(i+1) - z(i+2))) +...
    (phi(i+2)*(z(i-1)*z(i-2) - 2*z(i)*z(i-2) - 2*z(i)*z(i+1) -...
    2*z(i)*z(i-1) + z(i-1)*z(i+1) + z(i-2)*z(i+1) + 3*z(i)^2))/((z(i)
        -...
```

```
18      z(i+2))*(z(i-1) - z(i+2))*(z(i-2) - z(i+2))*(z(i+1) - z(i+2))) +...
19      (phi(i)*(z(i-1)*z(i-2) - 3*z(i)*z(i-2) - 3*z(i)*z(i+1) -...
20      3*z(i)*z(i+2) - 3*z(i)*z(i-1) + z(i-1)*z(i+1) + z(i-1)*z(i+2) +...
21      z(i-2)*z(i+1) + z(i-2)*z(i+2) + z(i+1)*z(i+2) + 6*z(i)^2))/((z(i)
            -...
22      z(i-1))*(z(i) - z(i-2))*(z(i) - z(i+1))*(z(i) - z(i+2)));
23
24      dPhidt(i) = 1j*a2(i);
25  end
26  end
27
28  % it wasn't the very first presentation for the complex coefficients
29  % that I have used. It was even more complicated before. But still
30  % this presentation is very overwhelming in comparison with the elegant
31  % matrices we will get later. And it is only calculations for the second
32  % order  derivative, can imagine how huge the formula for the fourth
33  % order derivative will be.
```

B Solve a system of equations

```
1  % Solves a system of equations obtained from Taylor's Expansions
2  % and finds the Finite Difference rule
3  clear all; clc;
4  % set the variables
5  syms phi_im2 phi_im phi_i phi_ip phi_ip2 z_im2 z_im z_i z_ip z_ip2 dx
       real
6  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
7  % set the terms of the equation
8  a = (-1j*2*dx);
9  b = (-1j*dx);
10 c = 0;
11 d = (1j*dx);
12 e = ((1j+1)*dx);
13
14 % create a matrix with these terms according to Taylor's Expansion
15 M = [1 (a) (a)^2 (a)^3 (a)^4
16      1 (b) (b)^2 (b)^3 (b)^4
17      1 (c) (c)^2 (c)^3 (c)^4
18      1 (d) (d)^2 (d)^3 (d)^4
19      1 (e) (e)^2 (e)^3 (e)^4];
20
21 % create the vector, the function "phi_i"
22 B = [phi_im2 phi_im phi_i phi_ip phi_ip2]';
23
24 % find the solution for all a-coefficients that represent different
25 % order spatial derivatives of the function "phi_i"
26 a = M\B;
27
28 % find a2 and a4 coefficients, to calculate the second and the fourth
29 % order derivatives of the function "phi (x_i,t)"
30 a2 = collect(a(3),phi_im2);
31 a2 = collect(a2,phi_im);
32 a2 = collect(a2,phi_ip2);
33 a2 = collect(a2,phi_ip);
```

```
a2 = collect(a2,phi_i);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
a2 = char(a2);
a2 = strrep(a2, '_im2', '(i-2)');
a2 = strrep(a2, '_im', '(i-1)');
a2 = strrep(a2, '_ip2', '(i+2)');
a2 = strrep(a2, '_ip', '(i+1)');
a2 = strrep(a2, '_i', '(i)')
%%%%%%%%%%%%%
a4 = collect(a(5),phi_im2);
a4 = collect(a4,phi_im);
a4 = collect(a4,phi_ip2);
a4 = collect(a4,phi_ip);
a4 = collect(a4,phi_i);

a4 = char(a4);
a4 = strrep(a4, '_im2', '(i-2)');
a4 = strrep(a4, '_im', '(i-1)');
a4 = strrep(a4, '_ip2', '(i+2)');
a4 = strrep(a4, '_ip', '(i+1)');
a4 = strrep(a4, '_i', '(i)')
```

C Matrices when Periodic boundaries are applied

C.1 Matrix to calculate a second order derivative

```
% calculates the matrix M2 which is a finite difference method for the
% second order derivative. M2 is a square matrix of the size (n+1, n+1),
% where n+1 is the total number of grid points including zero.
function M2_pbc = Matrix_M2_pbc(dx,n)

s = 1/dx^2;
M2_pbc = zeros(n+1,n+1);

for i = 1:n
    M2_pbc(i,i) = -5/4*s;
    M2_pbc(i,i+1) = 2/3*s;
    M2_pbc(i,i+2) = -1/24*s;
    M2_pbc(i+1,i) = 2/3*s;
    M2_pbc(i+2,i) = -1/24*s;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% at the point x_1
M2_pbc(1,n) = -1/24*s;
M2_pbc(1,n+1) = 2/3*s;
M2_pbc(1,1) = -5/4*s;
M2_pbc(1,2) = 2/3*s;
M2_pbc(1,3) = -1/24*s;
% at the point x_2
M2_pbc(2,n+1) = -1/24*s;
M2_pbc(2,1) = 2/3*s;
M2_pbc(2,2) = -5/4*s;
M2_pbc(2,3) = 2/3*s;
```

```
M2_pbc(2,4) = -1/24*s;

% at the point x_(n+1)
M2_pbc(n+1,n-1) = -1/24*s;
M2_pbc(n+1,n) = 2/3*s;
M2_pbc(n+1,n+1) = -5/4*s;
M2_pbc(n+1,1) = 2/3*s;
M2_pbc(n+1,2) = -1/24*s;
% at the point x_n
M2_pbc(n,n-2) = -1/24*s;
M2_pbc(n,n-1) = 2/3*s;
M2_pbc(n,n) = -5/4*s;
M2_pbc(n,n+1) = 2/3*s;
M2_pbc(n,1) = -1/24*s;


M2_pbc = M2_pbc(1:n+1,1:n+1);
end
```

C.2 Matrix to calculate a fourth order derivative

```
% calculates the matrix M4 which is a finite difference method for the
% fourth order spatial derivative.
% M4 is a square matrix of the size (n+1, n+1),
% where n+1 is the total number of grid points including zero.
function M4_pbc = Matrix_M4_pbc(dx,n)

s = 1/dx^4;
M4_pbc = zeros(n+1,n+1);

for i = 1:n
    M4_pbc(i,i) = 1/4*s;
    M4_pbc(i,i+1) = -1/6*s;
    M4_pbc(i,i+2) = 1/24*s;
    M4_pbc(i+1,i) = -1/6*s;
    M4_pbc(i+2,i) = 1/24*s;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% at the point x_1
M4_pbc(1,n) = 1/24*s;
M4_pbc(1,n+1) = -1/6*s;
M4_pbc(1,1) = 1/4*s;
M4_pbc(1,2) = -1/6*s;
M4_pbc(1,3) = 1/24*s;
% at the point x_2
M4_pbc(2,n+1) = 1/24*s;
M4_pbc(2,1) = -1/6*s;
M4_pbc(2,2) = 1/4*s;
M4_pbc(2,3) = -1/6*s;
M4_pbc(2,4) = 1/24*s;

% at the point x_(n+1)
M4_pbc(n+1,n-1) = 1/24*s;
M4_pbc(n+1,n) = -1/6*s;
```

```
M4_pbc(n+1,n+1) = 1/4*s;
M4_pbc(n+1,1) = -1/6*s;
M4_pbc(n+1,2) = 1/24*s;
% at the point x_n
M4_pbc(n,n-2) = 1/24*s;
M4_pbc(n,n-1) = -1/6*s;
M4_pbc(n,n) = 1/4*s;
M4_pbc(n,n+1) = -1/6*s;
M4_pbc(n,1) = 1/24*s;


M4_pbc = M4_pbc(1:n+1,1:n+1);
end
```

D The Schrödinger equation with Periodic boundaries

```
% Calculate a one-dimensional reduced Schrodinger equation
% on the plane with Periodic Boundaries.
% Some kind of a Gaussian function is used as an initial function.
clear all; clc;

tic % start timer

L = pi*2;
n = 100;
dx = L/n;
x = -L:dx:L;

% generate Gaussian function
gamma = 2; K = -10; x0 = L/2;
Gauss = exp(-gamma*(x-x0).^2).*exp(1j*K*x);

% generate matrix M2 that represent the Finite Difference
% coefficients for a second derivative of the function "phi(x,t)"
M2_pb = Matrix_M2_pbc (dx, 2*n);

% set the function that represents the right-hand side of the
% reduced Schrodinger equation
Schr = @(t,u) 1j*M2_pb*u;

% set the intial function to a chosen Gaussian
% and find the solution of an ordinary differential equation
phi0 = Gauss;
[t, phi] = ode23tb(Schr, [0 2], phi0);

% plot the result
figure(1)
m = length(t);
for k = 1:5:m
    drawnow
    plot(x, real(phi(k,:)))
    xmin = -L; xmax = L; ymin = -2; ymax = 2;
    axis([xmin xmax ymin ymax])
    text(6*xmax/7, 3*ymax/4, sprintf('t = %.2f',k))
```

```
39      grid on
40      xlabel('x'); ylabel('\phi(x,t)')
41  end
42
43  toc % stop timer
```

E The Swift-Hohenberg equation with Periodic boundaries

```
1  % Calculate a one-dimensional Swift-Hohenberg equation on the plane with
2  % Periodic Boundaries. White noise is used as an initial function
3  clear all; clc;
4
5  tic % start timer
6
7  L = 10;
8  n = 100;
9  dx = L/n;
10 x = -L:dx:L;
11
12 % generate White noise
13 Wnoise = wgn(length(x),1,0);
14
15 % generate matrices M2 and M4 that represent the Finite Difference
16 % coefficients for second and fourth derivatives
17 % of the function "phi(x,t)"
18 M2_pb = Matrix_M2_pbc (dx, 2*n);
19 M4_pb = Matrix_M4_pbc (dx, 2*n);
20
21 % set M as a matrix that presents the whole right side
22 % of the one-dimensional Swift-Hohenberg equation.
23 r = 0.2; % control parameter
24 M = (r-1)*eye(2*n+1) - 2*2*M2_pb - 24*M4_pb;
25
26 % set the function that represents the right-hand side of the
27 % Swift-Hohenberg equation
28 SwHoh_pb = @(t,u) M*u - u.^3;
29
30 % set the intial function as a White noise signal
31 % and find the solution of an ordinary differential equation
32 phi0 = Wnoise;
33 [t, phi] = ode23tb(SwHoh_pb, [0 50], phi0);
34
35 % plot the result
36 figure(1)
37 m = length(t);
38 for k = 1:m
39      drawnow
40      plot(x, real(phi(k,:)))
41      xmin = -L; xmax = L; ymin = -5; ymax = 5;
42      axis([xmin xmax ymin ymax])
43      text(6*xmax/7, 3*ymax/4, sprintf('t =  %.2f',k))
44      grid on
```

```
%     title(['evolution of a white noise under the Swift-Hohenberg, with
    r = ', num2str(r)])
     xlabel('x'); ylabel('\phi(x,t)')
end

toc % stop timer
```

F The Fourier transform of a Swift-Hohenberg equation with Periodic Boundaries

```
% Calculates Fourier transform of a one-dimensional Swift-Hohenberg
% equation on the plane with Periodic Boundaries. White noise is used
% as an initial function.
clear all; clc; clf;

tic % start timer

L = pi*4;
n = 400;
dx = L/n;
x = -L:dx:L;

% generates a Gaussian function
gamma = 2;
x0 = 0;
Gauss = exp(-gamma*(x - x0).^2);

% generates a White noise signal
Wnoise = wgn(length(x),1,0);

% generate matrices M2 and M4 that represent the Finite Difference
% coefficients for second and fourth derivatives of function "phi(x,t)"
M2_pb = Matrix_M2_pbc(dx, 2*n);
M4_pb = Matrix_M4_pbc(dx, 2*n);

% set M as a matrix that presents the whole right side
% of a one-dimensional Swift-Hohenberg equation.
r = 0.2; % control parameter
M = (r-1)*eye(2*n+1) - 2*2*M2_pb - 24*M4_pb;

% set the function that represents the right-hand side of the
% Swift-Hohenberg equation
SwHoh_pb = @(t,u) M*u - u.^3;

% set the initial function as a White noise signal or a Gaussian function
phi0 = Wnoise; % enable to choose White noise as an intial function
% phi0 = Gauss; % enable to choose Gaussian function as an intial one

% and find the solution of an ordinary differential equation
[t, phi] = ode23tb(SwHoh_pb, [0 50], phi0);

% set function f for the Fourier transform
f = phi;
r = 1:1:2*n; % new grid
```

```
m = length(t);
u = zeros(m,2*n); v = zeros(m,2*n); Fl = zeros(m,2*n+1);
for k = 1:m
    % first we calculate the inverse Fourier transform through the
    % obtained formula
    u(k,1:2*n) = exp(1j*pi*n)*exp(-1j*pi*(r-1)).*f(k,1:2*n);
    % then we calculate the Fourier transform of the inverse using
    % MATLAB function
    v(k,1:2*n) = fft(u(k,1:2*n),2*n);
    % find through the obtained formula the approximated continuous
    % Fourier transform
    Fl(k,1:2*n) = dx*sqrt(1/(2*pi))*exp(-1j*pi*(r-1)).*v(k,1:2*n);
end

% we also set the grid for the frequency domain 'lambda'
l = -n:1:n;
dl = pi/(n*dx);
lgrid = zeros(1,2*n+1);
for i = 1:2*n+1
    lgrid(i) = l(i)*dl;
end

% plotting results
figure(1)
for k = 1:m
    drawnow
    plot(lgrid, real(Fl(k,:)))
    grid on
    xmin = 0; xmax = max(lgrid)/8;
    ymin = -2; ymax = 3;
    line([1 1], [ymin ymax], 'Color','g', 'LineWidth', 1.5)
    axis([xmin xmax ymin ymax])
%     title(['evolution of a fourier transform of a white noise according
    to the Swift-Hohenberg, with r = ', num2str(r)])
    xlabel('\lambda'); ylabel('F(\lambda,t)')
    text(6*xmax/7, 3*ymax/4, sprintf('t =  %.2f',k))
end

toc % stop timer
```

G Matrices when PML is applied

G.1 Matrix to calculate a second order derivative

```
% calculates the matrix M2 which is a finite difference method for the
% second order derivative when PML is applied.
function M2 = Matrix_M2_pml(dx,n,al,ar)

% you can use these commented lines below to check
% if the approximation for the derivative is correct
% you have to disable a 'function' mode and run the code,
% then observe the matrix
% dx = 1;
% n = 15;
```

```
% al = 5;
% ar = 11;

s = 1/dx^2;
M2 = zeros(n,n);

% values inside the PML from the left
for i = 1:al
    M2(i,i) = 5/4*s;
    M2(i,i+1) = -2/3*s;
    M2(i,i+2) = 1/24*s;
    M2(i+1,i) = -2/3*s;
    M2(i+2,i) = 1/24*s;
end
% values in the middle, where there is no PML
for i = al:ar
    M2(i,i) = -5/4*s;
    M2(i,i+1) = 2/3*s;
    M2(i,i+2) = -1/24*s;
    M2(i+1,i) = 2/3*s;
    M2(i+2,i) = -1/24*s;
end
% values inside the PML from the right
for i = ar:n-2
    M2(i,i) = 5/4*s;
    M2(i,i+1) = -2/3*s;
    M2(i,i+2) = 1/24*s;
    M2(i+1,i) = -2/3*s;
    M2(i+2,i) = 1/24*s;
end
% values for 4 edge points
% for x_1
M2(1,1) = -15/8*s;
M2(1,2) = 77/12*s;
M2(1,3) = -107/12*s;
M2(1,4) = 13/2*s;
M2(1,5) = -61/24*s;
M2(1,6) = 5/12*s;
% for x_2
M2(2,1) = 0;
M2(2,2) = -15/8*s;
M2(2,3) = 77/12*s;
M2(2,4) = -107/12*s;
M2(2,5) = 13/2*s;
M2(2,6) = -61/24*s;
M2(2,7) = 5/12*s;
% for x_{n-1}
M2(n-1,n-1) = -15/8*s;
M2(n-1,n-2) = 77/12*s;
M2(n-1,n-3) = -107/12*s;
M2(n-1,n-4) = 13/2*s;
M2(n-1,n-5) = -61/24*s;
M2(n-1,n-6) = 5/12*s;
% for x_{n}
```

```
M2(n,n) = -15/8*s;
M2(n,n-1) = 77/12*s;
M2(n,n-2) = -107/12*s;
M2(n,n-3) = 13/2*s;
M2(n,n-4) = -61/24*s;
M2(n,n-5) = 5/12*s;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% values for 6 special points
% a_2 at the point x_{al-1}
M2(al-1,al-1) = (1j/4 + 5/4)*s;
M2(al-1,al) = (- 1j/6 - 1/2)*s;
M2(al-1,al+1) = -1/10*s;
M2(al-1,al-2) = (- 1j/10 - 7/10)*s;
M2(al-1,al-3) = (1j/60 + 1/20)*s;
% a_2 at the point x_{al}
M2(al,al) = ((9*1j)/4)*s;
M2(al,al+1) = (8/5 - (6*1j)/5)*s;
M2(al,al+2) = ((3*1j)/40 - 11/40)*s;
M2(al,al-1) = (- (6*1j)/5 - 8/5)*s;
M2(al,al-2) = ((3*1j)/40 + 11/40)*s;
% a_2 at the point x_{al+1}
M2(al+1,al+1) = (1j/4 - 5/4)*s;
M2(al+1,al+2) = (7/10 - 1j/10)*s;
M2(al+1,al+3) = (1j/60 - 1/20)*s;
M2(al+1,al) = (1/2 - 1j/6)*s;
M2(al+1,al-1) = 1/10*s;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% a_2 at the point x_{ar-1}
M2(ar-1,ar-3) = (1j/60 - 1/20)*s;
M2(ar-1,ar-2) = (7/10 - 1j/10)*s;
M2(ar-1,ar-1) = (1j/4 - 5/4)*s;
M2(ar-1,ar) = (1/2 - 1j/6)*s;
M2(ar-1,ar+1) = 1/10*s;
% a_2 at the point x_{ar}
M2(ar,ar) = ((9*1j)/4)*s;
M2(ar,ar+1) = (- (6*1j)/5 - 8/5)*s;
M2(ar,ar+2) = ((3*1j)/40 + 11/40)*s;
M2(ar,ar-1) = (8/5 - (6*1j)/5)*s;
M2(ar,ar-2) = ((3*1j)/40 - 11/40)*s;
% a_2 at the point x_{ar+1}
M2(ar+1,ar+1) = (1j/4 + 5/4)*s;
M2(ar+1,ar+2) = (- 1j/10 - 7/10)*s;
M2(ar+1,ar+3) = (1j/60 + 1/20)*s;
M2(ar+1,ar) = (- 1j/6 - 1/2)*s;
M2(ar+1,ar-1) = -1/10*s;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
```

G.2 Matrix to calculate a fourth order derivative

```
% calculates the matrix M4 which is a finite difference method for the
% fourth order derivative when PML is applied.
function M4 = Matrix_M4_pml(dx,n,al,ar)

% you can use these commented lines below to check
```

```
% if the approximation for the derivative is correct
% you have to disable a 'function' mode and run the code,
% then observe the matrix
% dx = 1;
% n = 15;
% al = 5;
% ar = 11;

s = 1/dx^4;
M4 = zeros(n,n);

% values for the whole region from left to right, because the rule stays
% the same whether PML is applied or not
for i = 1:n-2
    M4(i,i) = 1/4*s;
    M4(i,i+1) = -1/6*s;
    M4(i,i+2) = 1/24*s;
    M4(i+1,i) = -1/6*s;
    M4(i+2,i) = 1/24*s;
end
% values for 4 edge points
% for x_1
M4(1,1) = 3/24*s;
M4(1,2) = -14/24*s;
M4(1,3) = 26/24*s;
M4(1,4) = -24/24*s;
M4(1,5) = 11/24*s;
M4(1,6) = -2/24*s;
% for x_2
M4(2,1) = 0;
M4(2,2) = 3/24*s;
M4(2,3) = -14/24*s;
M4(2,4) = 26/24*s;
M4(2,5) = -24/24*s;
M4(2,6) = 11/24*s;
M4(2,7) = -2/24*s;
% for x_{n-1}
M4(n-1,n-1) = 3/24*s;
M4(n-1,n-2) = -14/24*s;
M4(n-1,n-3) = 26/24*s;
M4(n-1,n-4) = -24/24*s;
M4(n-1,n-5) = 11/24*s;
M4(n-1,n-6) = -2/24*s;
% for x_{n}
M4(n,n) = 3/24*s;
M4(n,n-1) = -14/24*s;
M4(n,n-2) = 26/24*s;
M4(n,n-3) = -24/24*s;
M4(n,n-4) = 11/24*s;
M4(n,n-5) = -2/24*s;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% values for 6 special points
% a_4 at the point x_{al-1}
M4(al-1,al-1) = (1j/4 + 1/4)*s;
```

```
M4(al-1,al) = (-1j/6)*s;
M4(al-1,al+1) = -1/10*s;
M4(al-1,al-2) = (- 1j/10 - 1/5)*s;
M4(al-1,al-3) = (1j/60 + 1/20)*s;
% a_4 at the poimt x_{al}
M4(al,al) = -1/4*s;
M4(al,al+1) = ((3*1j)/10 + 1/10)*s;
M4(al,al+2) = (1/40 - (3*1j)/40)*s;
M4(al,al-1) = (1/10 - (3*1j)/10)*s;
M4(al,al-2) = ((3*1j)/40 + 1/40)*s;
% a_4 at the point x_{al+1}
M4(al+1,al+1) = (1/4 - 1j/4)*s;
M4(al+1,al+2) = (1j/10 - 1/5)*s;
M4(al+1,al+3) = (1/20 - 1j/60)*s;
M4(al+1,al) = (1j/6)*s;
M4(al+1,al-1) = -1/10*s;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% a_4 at the point x_{ar-1}
M4(ar-1,ar-1) = (1/4 - 1j/4)*s;
M4(ar-1,ar) = (1j/6)*s;
M4(ar-1,ar+1) = -1/10*s;
M4(ar-1,ar-2) = (1j/10 - 1/5)*s;
M4(ar-1,ar-3) = (1/20 - 1j/60)*s;
% a_4 at the point x_{ar}
M4(ar,ar) = -1/4*s;
M4(ar,ar+1) = (1/10 - (3*1j)/10)*s;
M4(ar,ar+2) = ((3*1j)/40 + 1/40)*s;
M4(ar,ar-1) = ((3*1j)/10 + 1/10)*s;
M4(ar,ar-2) = (1/40 - (3*1j)/40)*s;
% a_4 at the point x_{ar+1}
M4(ar+1,ar+1) = (1j/4 + 1/4)*s;
M4(ar+1,ar+2) = (- 1j/10 - 1/5)*s;
M4(ar+1,ar+3) = (1j/60 + 1/20)*s;
M4(ar+1,ar) = -1j/6*s;
M4(ar+1,ar-1) = -1/10*s;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end
```

H The Schrödinger equation with PML

```
% Calculates a reduced Schrodinger equation on the plane with PML.
% A Gaussian function is chosen as an initial one
clear all; clc; clf;

tic % start timer

% define the grid
M = 60; % number of grid steps from '0' to PML-point 'a'
a = 3; % the value of PML-point
dx = a/M; %define the step (to make sure that a PML-point is on the grid)
N = 100; % number of points from '0' to the lateral boundary
L = dx*N; % the value of the point where the lateral boundary is situated
x = -L:dx:L; % define x-grid
n = 2*N + 1; % the whloe number of grid points from '-L' to 'L'
```

```
15
16 % discretize z in accordance with a chosen PML
17 % z = a + if(x) = a + exp(i*tetta)(x-a)
18 ar = N+1 + M; % number of the right PML-point
19 al = N+1 - M; % number of the left PML-point
20
21 aR = x(ar); % value of the right PML-point
22 aL = x(al); % value of the left PML-point
23
24 theta = pi/2; % set incline 'theta'
25
26 z(1:al) = aL + exp(1j*theta)*(x(1:al) - aL); % left PML
27 z(al+1:ar-1) = x(al+1:ar-1); % middle without PML
28 z(ar:n) = aR + exp(1j*theta)*(x(ar:n) - aR); % right PML
29
30 % generate the initial Gaussian function
31 gamma = 1;
32 K = 8;
33 z0 = 0;
34 x0 = 0;
35 Gauss = exp(-gamma*(x - x0).^2).*exp(1j*K*x);
36
37 % generate matrix M2 that represent the Finite Difference
38 % coefficients for the second order derivative
39 M2 = Matrix_M2_pml(dx,n,al,ar);
40
41 % set the function that represents the right-hand side of the
42 % reduced Schrodinger equation
43 Schr = @(t,u) 1j/2*2*M2*u;
44
45 % set the initial function as a Gaussian function
46 % and find the solution of an ordinary differential equation
47 phi0 = Gauss;
48 [t, phi] = ode23tb(Schr, [0 1], phi0);
49
50 % plot the result
51 figure(1)
52 m = length(t);
53 for k = 1:m
54     drawnow
55     plot(x, real(phi(k,:)))
56     xmin = -L; xmax = L; ymin = -2; ymax = 2;
57     axis([xmin xmax ymin ymax])
58     text(6*xmax/7, 3*ymax/4, sprintf('t = %.2f',k))
59     line([a,a],[ymin,ymax],'Color','g');
60     line([-a,-a],[ymin,ymax],'Color','g');
61     grid on
62 %     title(['evolution of a white noise under the Swift-Hohenberg, with
    r = ', num2str(r)])
63     xlabel('x'); ylabel('\phi(x,t)')
64 end
65
66 toc % stop timer
```

I The Swift-Hohenberg equation with PML

```
% Calculates a one-dimensional Swift-Hohenberg equation on the plane
% with PML. A White noise signal is chosen as an initial function
clear all; clc; clf;

tic % start timer

r = -0.5; % control parameter
% define the grid
M = 50; % number of grid steps from '0' to PML-point 'a'
a = 6*pi; % the value of PML-point
dx = a/M; %define the step (to make sure that a PML-point is on the grid)
N = 100; % number of points from '0' to the lateral boundary
L = dx*N; % the value of the point where the lateral boundary is situated
x = -L:dx:L; % define x-grid
n = 2*N + 1; % the whole number of grid points from '-L' to 'L'

% discretize z in accordance with a chosen PML
% z = a + if(x) = a + exp(i*tetta)(x-a)
ar = N+1 + M; % number of the right PML-point
al = N+1 - M; % number of the left PML-point

aR = x(ar); % value of the right PML-point
aL = x(al); % value of the left PML-point

theta = pi/2; % set incline 'theta'

z(1:al) = aL + exp(1j*theta)*(x(1:al) - aL); % left PML
z(al+1:ar-1) = x(al+1:ar-1); % middle without PML
z(ar:n) = aR + exp(1j*theta)*(x(ar:n) - aR); % right PML

% generate White noise
Wnoise = wgn(length(x),1,0);

% generate White noise that is chopped before the PML starts
chop = -5; % the positive value will chop it a bit after the PML
Wnoise_ch = zeros(1,n);
Wnoise_ch(al-chop:ar+chop) = Wnoise(al-chop:ar+chop);

% generate matrices M2 and M4 that represent the Finite Difference
% coefficients for the second and fourth derivatives
% of the function "phi(x,t)"
M2_pml = Matrix_M2_pml (dx,n,al,ar);
M4_pml = Matrix_M4_pml (dx,n,al,ar);

% set M as a matrix that presents the whole right side
% of the one-dimensional Swift-Hohenberg equation.
M = (r-1)*eye(n) - 2*2*M2_pml - 24*M4_pml;

% set the function that represents the right-hand side of the
% Swift-Hohenberg equation
SwHoh_pml = @(t,u) M*u - u.^3;

```

```
% set the initial function as a White noise signal (or its chopped
% version) and find the solution of an ordinary differential equation
phi0 = Wnoise;
[t, phi] = ode23tb(SwHoh_pml, [0 100], phi0);

% plot the result
figure(1)
m = length(t);
for k = 1:m
    drawnow
    plot(x, real(phi(k,:)))
    xmin = -L; xmax = L; ymin = -5; ymax = 5;
    axis([xmin xmax ymin ymax])
    text(6*xmax/7, 3*ymax/4, sprintf('t =  %.2f',k))
    line([a,a],[ymin,ymax],'Color','g');
    line([-a,-a],[ymin,ymax],'Color','g');
    grid on
    title(['Evolution of a White Noise according to the Swift-Hohenberg
       equation'...
        '\newlineControl Parameter  r = ', num2str(r),' , PML starts at -
           a and a = ', num2str(a/pi),'\pi'...
        ' , dx = ', num2str(dx)])
    xlabel('x'); ylabel('\phi(x,t)')
end

toc % stop timer
```