Computing the Ginzburg-Landau Equation

Jan Kierzkowski

July 2024

1 Equations

The Ginzburg-Landau equation[1] is given by:

$$f(t, A) = \frac{\partial A}{\partial t} = (1 + i\alpha)\nabla^2 A + A - (1 + i\beta) \mid A \mid^2 A. \tag{1}$$

For the initial conditions, we choose $\alpha = 0$, $\beta = 1.5$, and $A(t = 0) = A_0$:

$$A_0(x, y) = (ix + y) \exp(-0.03(x^2 + y^2)), \tag{2}$$

where x and y are spatial coordinates, and i is the imaginary unit. We then evaluate the system over time.

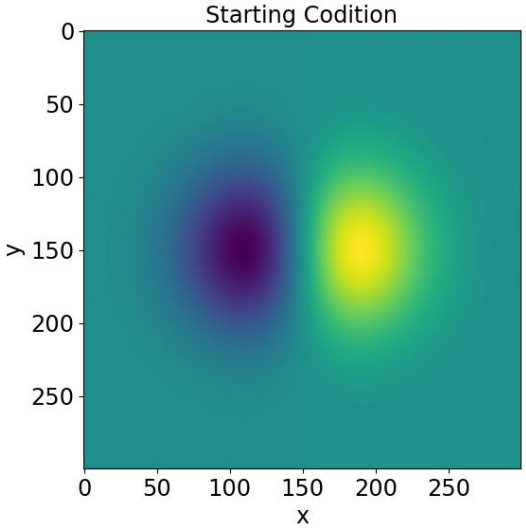


Figure 1: Initial conditions from equation (2).

2 Methods

The Ginzburg-Landau equation was solved using three numerical methods: the first-order Euler method, Heun's method, and the Runge–Kutta–Fehlberg method.

2.1 Euler Method

The simplest method used is the first-order Euler method. The approach is to take a constant time step h and approximate the next value of the function A based on the current value:

$$A_{n+1} = A_n + h \cdot f(t_n, A_n), \tag{3}$$

where $f(t_n, A_n)$ is the function defined by equation (1). In this calculation, the time step was chosen as $h = 0.001$.

2.2 Heun's Method

The next method in terms of complexity is Heun's method, which is an improved version of Euler's method. It introduces a half-step, $\overline{A_{n+1}}$, as follows:

$$\overline{A_{n+1}} = A_n + h \cdot f_n, \tag{4}$$

and then calculates the actual next value of A:

$$A_{n+1} = A_n + \frac{h}{2}\left(f(t_n, A_n) + f(t_{n+1}, \overline{A_{n+1}})\right). \tag{5}$$

The time step h was the same as in the Euler method, $h = 0.001$.

2.3 Runge–Kutta–Fehlberg Method

The most complex and accurate method used in this study is the Runge–Kutta–Fehlberg method (often referred to as RK45). This method differs from the others by using an adaptive time step h. The step is calculated based on a weighted average of six incrementally more accurate function approximations.

These six function approximations are:

1. $k_1 = h \cdot f(t_n + C_1(0)h, A_n)$
2. $k_2 = h \cdot f(t_n + C_1(1)h, A_n + C_2(1,0)k_1)$
3. $k_3 = h \cdot f(t_n + C_1(2)h, A_n + C_2(2,0)k_1 + C_2(2,1)k_2)$
4. $k_4 = h \cdot f(t_n + C_1(3)h, A_n + C_2(3,0)k_1 + C_2(3,1)k_2 + C_2(3,2)k_3)$
5. $k_5 = h \cdot f(t_n + C_1(4)h, A_n + C_2(4,0)k_1 + C_2(4,1)k_2 + C_2(4,2)k_3 + C_2(4,3)k_4)$

6. $k_6 = h \cdot f(t_n + C_1(5)h, A_n + C_2(5,0)k_1 + C_2(5,1)k_2 + C_2(5,2)k_3 + C_2(5,3)k_4 + C_2(5,4)k_5)$

where $C_1(k)$ and $C_2(k,l)$ are coefficients derived by Fehlberg. Values were taken from the second table on the Wikipedia page for the Runge–Kutta–Fehlberg method[3], and are reproduced in Tables 1 and 2 below.

To calculate the new time step h, we first compute the weighted average truncation error TE:

$$TE = \left| \sum_{i=1}^{6} CT(i) \cdot k_i \right|, \tag{6}$$

where the CT coefficients are also from Table 2. The new time step h_{new} is then calculated as:

$$h_{\text{new}} = 0.9h \left(\frac{\epsilon}{TE} \right)^{1/5}, \tag{7}$$

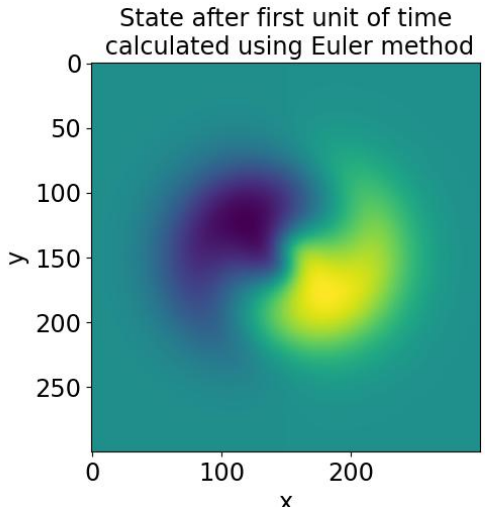
If $TE > \epsilon$, then h is replaced by h_{new} and the step is repeated. If $TE \leq \epsilon$, then the step is completed and h is updated to h_{new} for the next iteration.

3 The Code

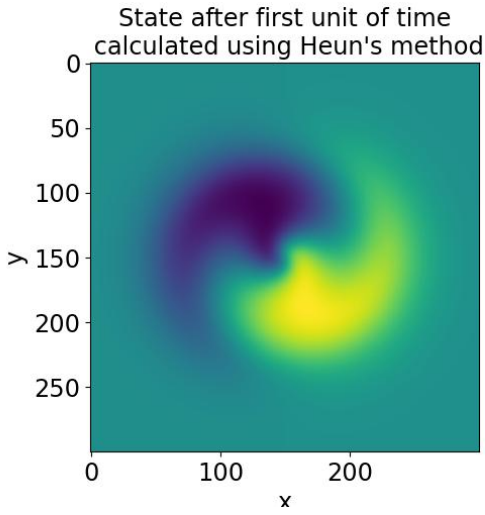
All calculations were performed using the Google Colab platform. The code is accessible via this link[2]. The script is written in Python, using standard libraries like NumPy and SciPy. The Laplacian in equation (1) was computed using the ‘laplace‘ function from SciPy, with the "wrap" option for boundary conditions. All matrices were generated as NumPy arrays. The grid used for these calculations consisted of 600 evenly distributed points from $-15-15i$ to $15+15i$ on the complex plane. All plots were generated using the Matplotlib library.

4 Results

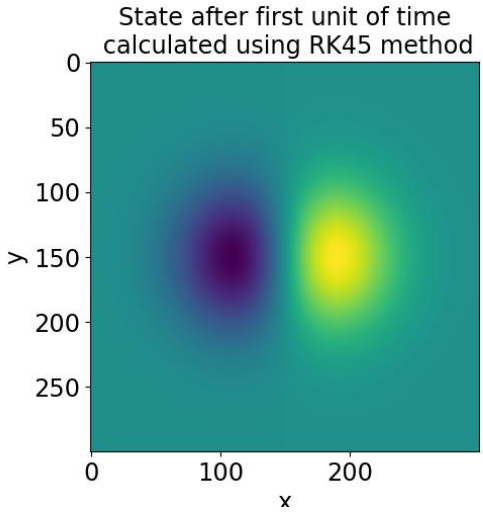
The results presented are the real parts of A_n at several time steps. The calculations were performed over a time range from 0 to 10 units. All methods completed in relatively short times. The fastest was the Runge–Kutta–Fehlberg method, which finished in 10 seconds due to its adaptive step. The Euler and Heun methods completed in about 30 seconds each.
In the initial phase, the merging of the two high spots from the initial state is observed. The most advanced state of "mixing" is seen in the calculations using the Runge–Kutta–Fehlberg method. The images below show the results after the first time unit. Heun's and Euler's methods yield visually similar results.



(a) Euler method

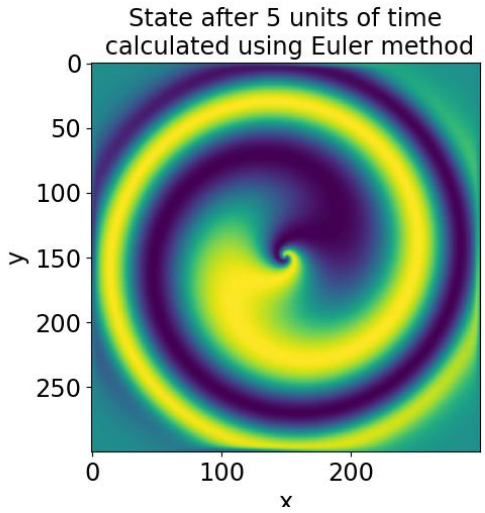


(b) Heun's method

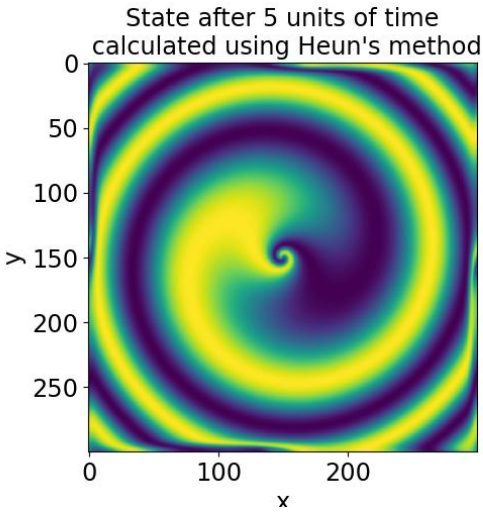


(c) RK4(5) method

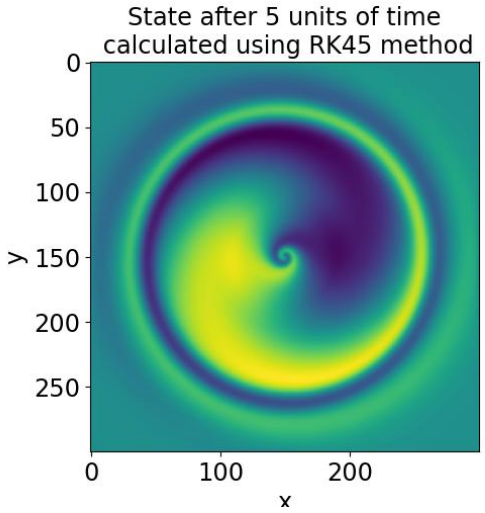
At the halfway point of the simulation, differences start to appear between the methods. Heun's method shows one more ring than the others. At this point, Euler's and Runge–Kutta–Fehlberg methods are much closer to each other.



(a) Euler method

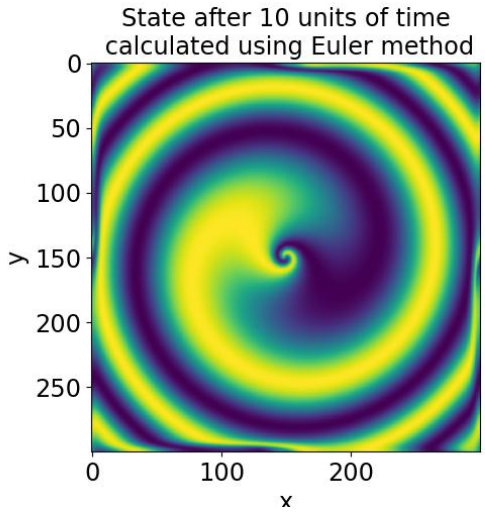


(b) Heun's method

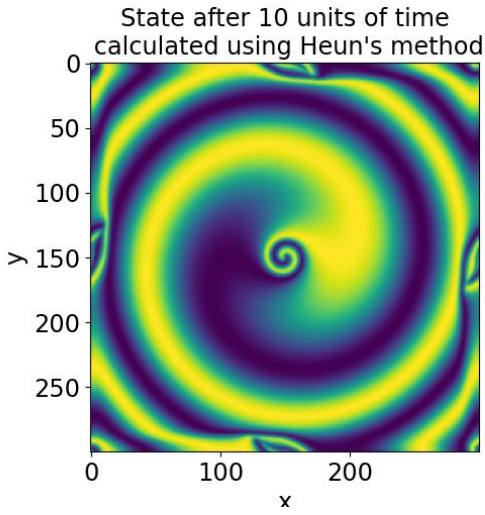


(c) RK4(5) method

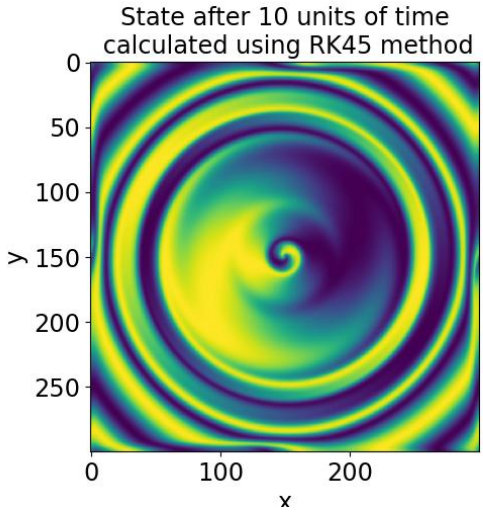
By the end of the simulation (at 10 time units), the differences are significant. Euler's and Heun's methods show the appearance of spirals and structures moving towards the edges of the grid. Runge–Kutta–Fehlberg results show the most uniform distribution of energy, which continues to evolve.



(a) Euler method



(b) Heun's method



(c) RK4(5) method

5 Summary

This paper presents the results of integrating the Ginzburg-Landau equation using three methods: Euler's method, Heun's method, and the Runge–Kutta–Fehlberg method (RK45). Differences between the methods appear at all stages of the simulations but are most pronounced at the final time (10 time units). The Runge–Kutta–Fehlberg method produced the most uniform and stable results, although Euler's and Heun's methods both performed well in the early stages of the simulation. In the final stage of Heun's method, some abnormal shapes appeared at the edges due to the wrap-around option in the Laplace function.

Table 1: Runge-Kutta-Felberg coefficients table[3].

K	**A(K)**	**B(K,L)**				
		L=1	L=2	L=3	L=4	L=5
1	0					
2	1/4	1/4				
3	3/8	3/32	9/32			
4	12/13	1932/2197	-7200/2197	7296/2197		
5	1	439/216	-8	3680/513	-845/4104	
6	1/2	-8/27	2	-3544/2565	1859/4104	-11/40

Table 2: Runge-Kutta-Felberg coefficients table[3].

K	**C(K)**	**CH(K)**	**CT(K)**
1	25/216	16/135	-1/360
2	0	0	0
3	1408/2565	6656/12825	128/4275
4	2197/4104	28561/56430	2197/75240
5	-1/5	-9/50	-1/50
6		2/55	-2/55

References

[1] *Ginzburg–Landau equation.* URL: https://en.wikipedia.org/wiki/Ginzburg%E2%80%93Landau_equation.

[2] *Google colab with a python script used in this calculations.* URL: https://colab.research.google.com/drive/1YzuOREvTIOKOZPPuUpTluCrTDNOJLA7c%5C?usp=sharing.

[3] *Runge–Kutta–Fehlberg method.* URL: https://en.wikipedia.org/wiki/Runge%5C%E2%5C%80%5C%93Kutta%5C%E2%80%5C%93Fehlberg%5C_method.