Multi-steps Methods for Calculating the Phase Portrait of the Incommensurate Superstructure with the Lifshitz's Invariant

S. Sveleba, I. Katerynchuk, I. Kunyo, I. Karpa
Ivan Franko National University of Lviv
107 Tarnavsky St.,
UA–79017 Lviv, Ukraine
incomlviv@gmail.com

Abstract—**Two multi-steps methods: Adams and BDF was applied for calculation of phase portrait of incommensurate superstructure with the Lifshitz's invariant. The Python language was used to create the correspond application. The result of numerical experiments was gives an excellent result in the calculation of complex dynamic systems.**

Index Terms—**multi-steps methods; Python language; incommensurate superstructure; Lifshitz's invariant; phase system portrait.**

The best accuracy of derivatives approximation is obtained by using information with a large number of points. In this case, for the calculation of the value y_{i+1}, we use the results of not one, but a *n* of the previous steps, in other words we use the value of the function in y_{i-n+1}, y_{i-n+2}, ..., y_i prevision points. So in this case the *n*-steps method is obtained.

There are explicit and implicit methods. A good result can be obtained using a combination of the explicit and the implicit method. They are called the methods of forecasting and correction. Each step consists of two stages and is calculated by multi-step methods. Using the explicit method (forecast) for the known values of the function in the previous nodes is the initial approximation in the new node. Using an implicit method (correction), the result of iterations is the correction of this value.

In explicit algorithms for calculating the value of a function at the current stage, we need the values of the function and its derivatives from the previous steps, and in implicit methods, this derivative must be predicted in the current step with the help of additional algorithms, which makes them less economical. However, the area of the stability zone for explicit methods is always less than that for the implicit, but when we increasing the step in explicit methods, they increases, and in the implicit - decreases [4]. Implicit and multi-step methods, as a rule, do not require setting the value of a step, it adapts automatically for reasons of accuracy. Methods of the second and higher orders are nonlinear, moreover, the higher order of accuracy needed more additional components in the formula of the method need to be calculated [1, 2].

In the simulation programs, the explicit one-step Runge-Kutta step methods that are multi-stage are widely used and perform several intermediate steps. After that, the basic stage is carried out, which allows you to increase the accuracy of the solution, using only the information from one current step. Algorithms with prediction-correction of the second and higher orders perform two calculations of functions at each step, with the usual use of the explicit Adams-Bashfort method, and as the corrector in the implicit Adams-Multon. The multifaceted (linear) methods (Gir’s, Adams-Bashfort, Adams-Multon) use information from previous steps. A separate case of linear methods is the explicit and implicit Euler method, which can be taken as Adams-Bashfort and Adams-Multon respectively in the first order with one step. Since there is a contradiction between the criteria of stability (explicit, implicit) and accuracy (order of the method), accuracy and economy (the cost in the machine time), the problem of optimal choice of integration method and modeling parameters, especially for rigid models [1], with a large scale of constant integration . The paper [3] analyzes the main methods that are widely used in the simulators and sorts them in accordance with the aforementioned criteria and areas of application. The presented methods can be grouped as follows.

1. Methods based on the Runge-Kutti algorithm: RK2 and RK4 are two stages with the calculation of the intermediate point [2]; RKF45 and Dormand-Prince (DOPRI853) require 4 to 6 intermediate calculations of the function [4].

2. Methods based on the formula of back differentiation: the Gyr’s method (BDF) of variable order with the first order of derivatives [4, 5], which launches in each step the iterative process of linearization by simple iterative methods, Bridene, Newton-Raffson; MeBDFi is a modified extended back differentiation formula [4].

3. Comprehensive algorithms: VODE and Vode Adams, consisting of a multi-steps implicit Adams-Moulton method of variable order and step, and multi-steps BDF of variable order and step. The resulting nonlinear system of algebraic equilibrium is solved by the iterative linearization method at each step of integration [1].

Thus, the most effective method for calculating systems of second-order differential equations is the multi-steps implicit Adams-Multon method of variable order, and the multi-steps method of BDF of variable order .

Therefore, the purpose of this work is to construct phase portraits for an incommensurate superstructure described by two differential equations of the second order [6]:

$$R'' - R^3 + \left(1 - \varphi'^2 + T\varphi'\right)R - R^{n-1}K\left(\cos n\varphi + 1\right) = 0\,, \quad (1)$$

$$\varphi'' + 2\frac{R'}{R}(\varphi' - \frac{T}{2}) + R^{n-2}K\sin n\varphi = 0 \quad (2)$$

.

here $T = \frac{\sigma}{(\gamma r)^{\frac{1}{2}}}$, $K = 2^{-\frac{n}{2}} r^{\frac{n}{2}-2} n\omega u^{1-\frac{n}{2}}$ — dimensionless parameters, n — an integer characterizing the potential symmetry, and dimensionless variables $\eta = \left(\frac{r}{2u}\right)^{1/2} R$, $z = \left(\frac{\gamma}{r}\right)^{1/2} \xi$, and checking the effectiveness of multi-steps Adams-Multon methods and BDF for solving the differential equation systems.

In this work, the construction of phase portraits of nonlinear dynamic systems, in the Python software environment using the scipy library. In this library, the class scipy.integrate.ode (f, jac = None) is the common interface class to numeric integrators. This class solves the system of equations ((y '(t) = f (t, y)) with jac = df / dy [7]. Using the set_integrator method, the integrator "vode", which is a common solver of the differential equation, was chosen from the implementation of a fixed-leading coefficient. In the set_integrator method of the ode class, the integrator accepts the following parameters:

1. atol: floating or definite accuracy (in this work it was chosen equal to atol = 1E-6 ÷ -12)

2. rtol: relative permissible parameter (rtol = 0);

3. method: " adams " or 'bdf'. In the calculation, the solver was used as "Adams" (for non-hard systems) and "BDF" (for hard systems);

4. with_jacobian: bool this parameter was entered when the Jacobian function was not considered and did not indicate that the Jacobian was grouped.

The import block looks like this:

```
from scipy.integrate import ode
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

From the scipy library, we call the common interface class to numeric integrators - scipy.integrate.ode (f, jac = None). Determine the function that is responsible for calculating the system of differential equations:

```
def f(t, y):
    c0=1.0
    c1=0.7
    c2=4
    y0, y1, y2, y3, = y
    return [y1,y0**3-(1+c0*y3-y3**2)*y0+c1*(y0**(c2-1))*(1+np.cos(c2*y2)),y3,-(y1/y0)*(2*y3-c0)-c1*(y0**(c2-2))*np.sin(c2*y2)]
```

The arguments of the function are:

y — matrix of the state of the variable

t — time

c0, c1, c2 — parameters of the differential equation (may be any number)

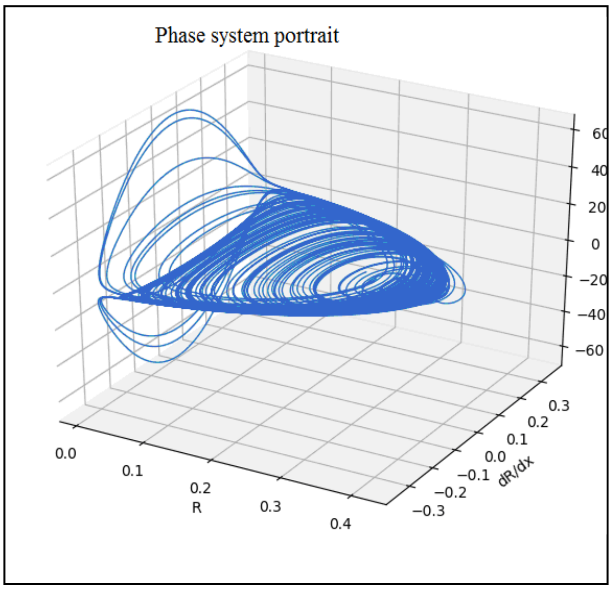
In this case c0=T, c1=K, c2=n. The function returns the matrix of derivatives. Next, the implementation of the function is obtained for a solution of a system of differential equations with given initial conditions.

```
r = ode(f).set_integrator("vode", method="BDF", order=10, rtol=0, atol=1e-6, with_jacobian=False)
y0 = [0.3,0.0,0.0,0.75]
r.set_initial_value(y0, 0)
T =300
dt = 0.0004
y = [ ]; t = [ ]
```

The Jacobi matrix or the Jacobian of a given system of functions is a matrix that is formed by partial derivatives of these functions for all variables. The value of this matrix should not exceed the specified accuracy value of the calculation. Failure to execute this condition returns the value to a repeated iterative loop. To implement a phase portrait, it is necessary to carry out a solution of a system of differential equations with different initial conditions. To implement, we also need an array of values of calculated functions.

```
while r.successful() and r.t <= T:
    r.integrate(r.t + dt)
    y.append(r.y); t.append(r.t)
y=np.array(y)
fig, ax = plt.subplots()
fig.set_facecolor('white')
ax=Axes3D(fig)
plt.xlabel('R')
plt.ylabel('dR/dx')
plt.title("Phase system portrait")
plt.plot(y[ :,0],y[ :,1],y[ :,3], linewidth=1)
plt.grid(True)
plt.show()
```

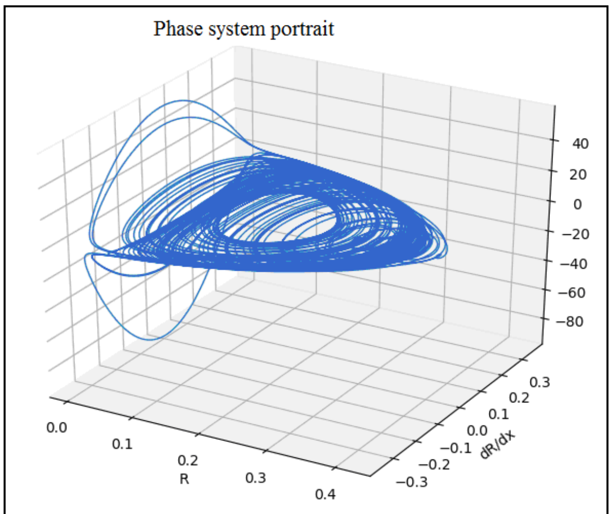
Fig. 1 and Fig. 2 show the results of the calculation and



```
def f(t, y):
    c0=1.0 #"c0=1.0;c1=1.5";
"c0=0.050;c1=1.5";"c0=0.050;c1=0.15";
"c0=0.05; c1=0.0015"
    c1=1.59
    c2=4
    y0, y1, y2, y3, = y
    return [y1,y0**3-(1+c0*y3-
y3**2)*y0+c1*(y0**(c2-
1))*(1+np.cos(c2*y2)),y3,-(y1/y0)*(2*y3-c0)-
c1*(y0**(c2-2))*np.sin(c2*y2)]
#from scipy.integrate import ode
r = ode(f).set_integrator("vode",
method="adams",order=10, rtol=0, atol=1e-6,
with_jacobian=False)
y0 = [0.3,0.0,0.0,0.75]
r.set_initial_value(y0, 0)
T =300
dt = 0.0004
y = []; t = []
while r.successful() and r.t <= T:
    r.integrate(r.t + dt)
    y.append(r.y); t.append(r.t)
```

Figure 1. Phase system portrait in coordinates R, dR/dx, dφ/dx provided: R_0=0,3; R'=0; φ_0=0; φ'=0,75 a) and the part of the code responsible for the calculation method (Adams) and the initial conditions b).

construction of phase portraits of a given system of differential equations obtained by implicit, multi-steps, variable-step calculation for soft systems by the Adams method in Fig. 1, and for hard systems by the BDF method in Fig. 2. The obtained phase portraits are qualitatively similar, only the value of the coefficient c1 differs, at which the system is still stable.



a

```
def f(t, y):
    c0=1.0 #"c0=1.0;c1=1.5"
;"c0=0.050;c1=1.5";"c0=0.050;c1=0.15";"c0=0.05;
c1=0.0015"
    c1=1.605
    c2=4
    y0, y1, y2, y3, = y
    return [y1,y0**3-(1+c0*y3-
y3**2)*y0+c1*(y0**(c2-
1))*(1+np.cos(c2*y2)),y3,-(y1/y0)*(2*y3-c0)-
c1*(y0**(c2-2))*np.sin(c2*y2)]
#from scipy.integrate import ode
r = ode(f).set_integrator("vode", method="BDF",
order=10, rtol=0, atol=1e-6,
with_jacobian=False)
y0 = [0.3,0.0,0.0,0.75]
r.set_initial_value(y0, 0)
T =300
dt = 0.0004
y = []; t = []
while r.successful() and r.t <= T:
    r.integrate(r.t + dt)
    y.append(r.y); t.append(r.t)
y=np.array(y)
```

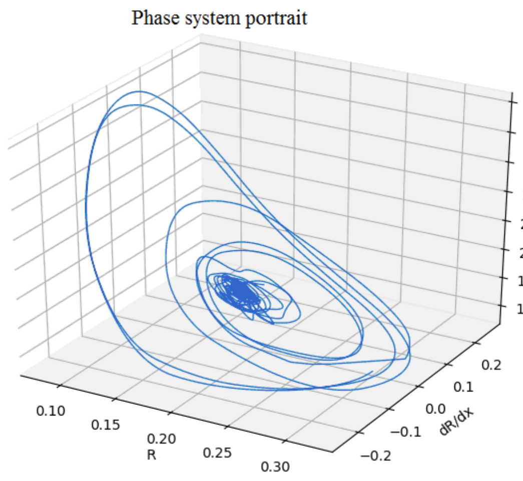
b

Figure 2. Phase system portrait in coordinates R, dR/dx, dφ/dx provided: R_0=0,3; R'=0; φ_0=0; φ'=0,75 a) and the part of the code responsible for the calculation method (BDF) and the initial conditions b).

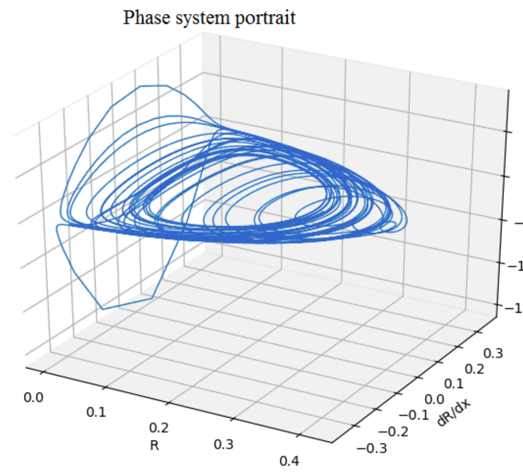
The peculiarity of these methods is that the solution at the next point depends on the solution in several previous points. To calculate the considered system of differential equations, this method is appropriate, since at this point it is necessary to calculate the quantities, namely dR/dx, dφ/dx, d^2R/dx^2, d$^2\varphi$/dx^2.

Also phase portraits for the considered calculation methods have a different values of a fixed step was calculated. At values of step> 0.01 monotonous behavior of attractors is observed. Under these conditions there is a slight difference in the received phase portraits by different methods. For the Adams method, there have at greater number of bifurcations than for the DBF method. With the subsequent decrease in the value of the step, this difference disappears.

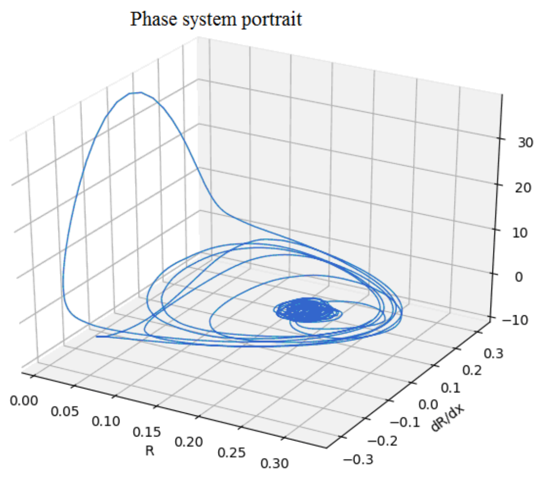
For hard and soft systems, the accuracy of the calculation is significant. For Adams and DBF methods, the accuracy of the calculation can either be by default or be given. Fig. 3 shows the calculated phase portraits of the research system under the condition of different accuracy values of the solution.



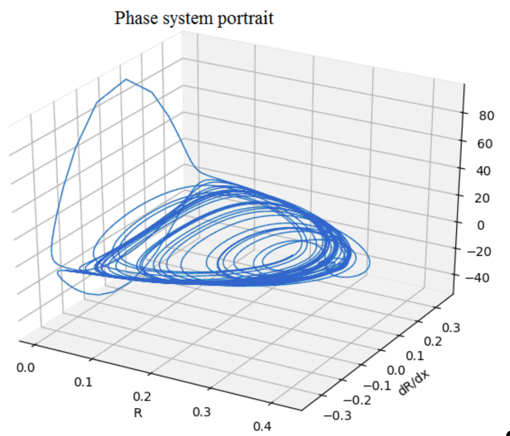
a



b



c



d

Figure 3. Phase system portrait in coordinates R, dR/dx, dφ/dx provided: R_0=0,3;R'=0; φ_0=0; φ'=0,75 at different values of calculation accuracy atol. a, b – Adams method; c, d – BDF method, atol=0.1 a, c atol=0.001 b, d.

The difference between the received phase portraits is observed only with low accuracy of calculation (atol = 0.1 ÷ 0.001). Given atol> 0.1, the shape of the phase portrait is far from ideal. In this case, the essential difference between the forms calculated by different methods is traced. In particular, the number of received attractors for the DBF method is greater than they were obtained in the calculation of the system by the Adams method. Given atol ≤ 0.001, this difference gradually disappears. Therefore, in the study of the influence of the stability parameter of the initial *T* = c0 and the parameter of anisotropic interaction *K* = c1, which is described by the Dzyaloshinsky's invariant on the phase portraits of this system, the accuracy of the calculation was atol ≤ 1e-6 ÷ 12.

Thus, the multi-step methods of Adams and DBF for calculating the systems of differential equations give an excellent result in the calculation of complex dynamic systems. It should be noted that these methods are relevant in the study of bifurcation processes, because they provide both high accuracy of calculation and small value of the calculation step.

REFERENCES

[1] Pereverzjev A. V., Vasylenko O. V., Prokopenko R. V. Zapobigannja algorytmichnyh zboi'v system ECAD, Radioelektronika, informatyka, upravlinnja, 2006, No. 1, pp. 123–128.

[2] Dushin S. E., Krasov A. V., Litvinov Yu. V. Modelirovanie system i kompleksov. Sankt-Peterburg, SPbGU ITMO, 2010, 177 p

[3] Vasilenko O.V., Petrenko Ya.I. Improvement of the Quality of Dynamic Systems Modeling by Choosing Optimal Simulation Algorithms Radioelektronika, Informatics, Management. - 2016. - No. 4. - September 11-18. [in Ukrainian]

[4] Press W. H., Teukolsky S. A., Vetterling W. T., Flannery B. P. Numerical Recipes. The Art of Scientific Computing (3rd ed.). Cambridge University Press, 2007, 1235 p.

[5] Cellier F. E., Kofman E. Continuous system simulation. Springer Verlag, New York, 2006, 643 p.

[6] I.M. Kuno, S.A. Sveleba, I.V. Karpa, I.M. Katerynchuk Inhomogeneous States of Thin-layer Crystals with Incommensurate Superstructure JOURNAL OF NANO- AND ELECTRONIC PHYSICS Vol. 10 No 2, 02026(6pp) (2018))

[7] http://man.hubwiz.com/docset/SciPy.docset/Contents/Resources/Documents/doc/generated/scipy.integrate.ode.html