ISSN NO: 2769-996X
Date of Publication: 26-05-2023

Practical application of Python in solving systems of linear and nonlinear equations

Qudratova Gulshoda Shodmonkulovna
student of Faculty of Economics and Management at Tashkent University of Information Technologies (TUIT), Tashkent, Republic of Uzbekistan

Annotation. The article studied methods for writing programs that solve scientific problems in the Python programming language.

Keywords: Python, numerical methods, scipy.optimize.root, spsolve_trianular, newton_krylov, matrix.

Introduction

The article studied how to write programs that solve scientific problems in the Python programming language has a clear and understandable syntax, and therefore it is easy to learn and well suited for programming. This will allow not to be distracted from the problem being solved by the features of the language and their explanation, and will naturally consistently introduce new tools. The language is close to MATLAB and is good for programming mathematical calculations. In addition, Python can work with languages such as Fortran, C and C ++, which are already widely used in scientific calculations. Among other things, most products and modules written in Python are distributed free of charge[7].

Modern scientific calculations are based on the use of numerical methods. Computational methods are the intellectual core of applied mathematical modeling, which is based primarily on solving non-linear non-stationary multidimensional problems for partial differential equations. This causes increasing attention to the training of specialists in numerical methods, both at the developer level and at the level of a qualified user [8].

In courses on numerical methods, the main attention is paid to numerical methods for solving problems of algebra and analysis, the problems of solving boundary value problems for ordinary differential equations and equations with partial derivatives are considered. In computational mathematics, the most important issues of constructing and theoretical substantiation of computational algorithms are studied. No less important is the problem of the practical use of numerical methods in solving applied problems [9].

Results and Discussions

Many applied problems lead to the need to find a general solution to a system of nonlinear equations. A general analytical solution to the system of nonlinear equations has not been found. There are only numerical methods. Python is considered to be an effective option for calculating numerical methods. And special Python library functions like scipy.optimize.root, spsolve_trianular, newton_krylov are the best choice for solving problems numerically. It’s hard to disagree with this, if only because the variety of modules has also raised Python to the top of popularity. However, there are cases when, even with a superficial consideration, the use of direct known methods without using the special functions of the SciPy library also gives good results[10].

A Python script is a text file containing some instructions. We can read the script and understand what the program is capable of doing, but the script itself does not perform any action on the computer until the Python interpreter reads the text of the script and converts it into some actions. Quite a lot of functionality is available in the default interpreter, but much more functionality is implemented in the Python libraries. In order to activate the use of additional functionality, we must explicitly import it. In Python, many mathematical functions are collected in the math library. Such a library is called a module in Python terms. Python has a large number of libraries that implement great features, of which we can only import the ones we need at the moment. Using Python, you can plot function graphs. For example, sometimes we need to plot two functions on the same figure. Let's say we have an array h as defined above and an array H. A graph with two curves can be constructed as follows[7]:

```
# -*- coding: utf-8 -*-
```

ISSN NO: 2769-996X
Date of Publication: 26-05-2023

```
import numpy as np
import matplotlib.pyplot as plt
h = np.zeros(4)
h[0] = 1.60; h[1] = 1.75; h[2] = 1.82; h[3] = 1.72
H = np.zeros(4)
H[0] = 0.60; H[1] = 0.30; H[2] = 1.90; H[3] = 1.99
numbers = np.zeros(4)
numbers[0] = 1; numbers[1] = 2; numbers[2] = 3; numbers[3] = 4
plt.plot(numbers, h, numbers, H)
plt.xlabel(u' Sequence number ')
plt.ylabel(u'Value')
plt.show()
```

For linear algebra, you may need standard operations, for example, multiplying a matrix by a vector. You can use a special type for this. For example, we want to calculate the vector y as follows

$$y=Ax,$$

where A is a 2×2 matrix, x is a vector. This can be implemented like this:

```
# -*- coding: utf-8 -*-
from numpy import zeros, mat, transpose
x = zeros(2)
x = mat(x)
x = transpose(x)
x[0] = 1.0; x[1] = 3.0
A = zeros((2,2))
A = mat(A)
A[0,0] = 1.0; A[0,1] = 2.0
A[1,0] = 3.0; A[1,1] = 4.0
y = A*x
print y
```

The problem of solving the linear system Ax=b (1) is central to scientific computing. To solve systems of the form (1), the Gauss elimination method and some iterative methods are used. One of the main problems of computational mathematics is the problem of solving systems of linear algebraic equations with real coefficients. Direct and iterative methods are used to find an approximate solution to systems of equations. The mathematical apparatus of linear algebra is based on the concepts of the norm of a vector and matrix, and the condition number. The classical methods of elimination of unknowns are considered, the features of solving problems with a symmetric real matrix are noted [1].

To solve triangular systems, we reduce general systems to a triangular form using Gaussian transformations. Given that the resulting method behaves very badly on a non-trivial class of problems, let's consider the concept of choosing pivots. As is known, the Gaussian method is direct, i.e. gives an exact solution to the system of linear equations. To check the implementation of the solution of a system of linear equations by the Gauss method, we can write the following function[2]:

```
def test_solve_lu():
    A = np.array([[1, 4, 7], [2, 5, 8], [3, 6, 10]])
    expected = np.array([-1./3, 1./3, 0])
    b = np.dot(A, expected)
    computed = solve_lu(A, b)
    tol = 1e-14
    success = np.linalg.norm(computed - expected) < tol
    msg = 'x_exact = ' + str(expected) + '; x_computed = ' + str(computed)
    assert success, msg
```

To solve nonlinear systems and equations, the solution is sought

$$f(x)=0, \qquad (2)$$

ISSN NO: 2769-996X
Date of Publication: 26-05-2023

where f(x) is a given function. The roots of equation (2) can be complex and multiple. It is singled out as an independent problem of separation of roots, when a region is selected in the complex plane containing one root. After that, on the basis of certain iterative processes with the chosen initial approximation, the solution of the nonlinear equation (2) [3] is found. In a more general case, we have not one equation (2), but a system of nonlinear equations

$$fi(x1,x2,\ldots,xn)=0, \quad i=1,2,\ldots n. \quad (3)$$

One of the methods for solving nonlinear equations is Newton's method. Code for the simplest implementation of Newton's method:

```
def naive_Newton(f, dfdx, x, eps):
   while abs(f(x)) > eps:
   x = x - float(f(x))/dfdx(x)
   return x
```

Or in corrected form:

```
def Newton(f, dfdx, x, eps):
   f_value = f(x)
   iteration_counter = 0
   while abs(f_value) > eps and iteration_counter < 100:
      try:
         x = x - f_value/dfdx(x)
      except ZeroDivisionError as err:
         print("Error: {}".format(err))
         sys.exit(1)
      f_value = f(x)
      iteration_counter += 1

   if abs(f_value) > eps:
      iteration_counter = -1
   return x, iteration_counter
```

If the initial approximation is close to the solution, Newton's method converges quickly. If the initial approximation is chosen incorrectly, Newton's method may diverge. Therefore, when the initial approximation is far from the exact solution, it may be possible to use several iterations of the bisection method, and then use the Newton method. When implementing Newton's method, it is necessary to know the analytical expression for the derivative f'(x)[4]. Python contains a SymPy package that can be used to create the dfdx function. For our task, this can be done as follows:

```
from sympy import symbol, sinh, diff, lambdify

x = symbol.Symbol('x')
f_expr = sinh(x)
dfdx_expr = diff(f_expr, x)
f = lambdify([x],
          f_expr)
dfdx = lambdify([x], dfdx_expr)
tol = 1e-1
sol, no_iterations = Newton(f, dfdx, x=1.09, eps=tol)
if no_iterations > 0:
   print("Equation: sinh(x) = 0. Count iteration: {}".format(no_iterations))
   print("Solve: {}, eps = {}".format(sol, tol))
else:
   print("No solve!")

def F(x):
   y = np.zeros_like(x)
```

ISSN NO: 2769-996X
Date of Publication: 26-05-2023

```
    y[0] = (3 + 2*x[0])*x[0] - 2*x[1] - 3
    y[1:-1] = (3 + 2*x[1:-1])*x[1:-1] - x[:-2] - 2*x[2:] -2
    y[-1] = (3 + 2*x[-1])*x[-1] - x[-2] - 4
    return y

def J(x):
    n = len(x)
    jac = np.zeros((n, n))
    jac[0, 0] = 3 + 4*x[0]
    jac[0, 1] = -2
    for i in range(n-1):
        jac[i, i-1] = -1
        jac[i, i] = 3 + 4*x[i]
        jac[i, i+1] = -2
    jac[-1, -2] = -1
    jac[-1, -1] = 3 + 4*x[-1]

    return jac

n = 10
guess = 3*np.ones(n)

sol, its = Newton_system(F, J, guess)

if its > 0:
    print("x = {}".format(sol))
else:
    print("No solve!")
```

Conclusions

Solutions obtained by numerical methods are numbers or a series of numbers, in contrast to formulas obtained from analytical solutions. This is a disadvantage of numerical solutions, since they are difficult to study. The advantage of numerical methods is that the solution method does not depend on the type of equations for the class of equations for which this method can be applied. For example, the same method can be applied to both algebraic and trigonometric equations [5].

The solutions obtained by numerical methods are approximate, so it is necessary to estimate the error with which the result is obtained. And this, in turn, depends on the resources of computer technology and the quality of software.

The purpose of this publication is to compare the number of iterations, speed, and most importantly, the result of solving a model problem in the form of a system of one hundred non-linear algebraic equations using Python and the Newton method implemented using libraries [6].

References

1. Qudratovich, S. S. (2022). The Role and Possibilities of Multimedia Technologies in Education. International Journal of Discoveries and Innovations in Applied Sciences, 2(3), 72–78. Retrieved from http://openaccessjournals.eu/index.php/ijdias/article/view/1148
2. Qudratovich, S. S. (2022). Technical and Software Capabilities of a Computer for Working with Multimedia Resources. International Journal of Discoveries and Innovations in Applied Sciences, 2(3), 64–71. Retrieved from http://openaccessjournals.eu/index.php/ijdias/article/view/1147
3. Sh.Q. Shoyqulov. (2022). The text is of the main components of multimedia technologies. Academicia Globe: Inderscience Research, 3(04), 573–580. https://doi.org/10.17605/OSF.IO/VBY8Z

ISSN NO: 2769-996X
Date of Publication: 26-05-2023

4. Shoyqulov Sh.Q. EditorJournals and Conferences. (2022, May 3). The graphics- is of the main components of multimedia technologies. https://doi.org/10.17605/OSF.IO/2KAM8
5. https://wos.academiascience.org/index.php/wos/article/view/1427
6. Shoykulova Dilorom Kudratovna, & Sh.Q. Shoyqulov. (2022). PHP is one of the main tools for creating a Web page in computer science lessons. Texas Journal of Engineering and Technology, 9, 83–87. Retrieved from https://zienjournals.com/index.php/tjet/article/view/2000
7. Sh.Q. Shoyqulov. (2021). Methods for plotting function graphs in computers using backend and frontend internet technologies. European Scholar Journal, 2(6), 161-165. Retrieved from https://scholarzest.com/index.php/esj/article/view/964
8. G.Sh. Qudratova. Using Ms Excel Function to Solve Economic Problems, Indonesian Journal of Innovation Studies. Vol. 18 (2022): April 2022. ISSN (ONLINE) 2598-9936
9. Sh.Q. Shoyqulov., METHODS FOR PLOTTING FUNCTION GRAPHS IN COMPUTERS USING BACKEND AND FRONTEND INTERNET TECHNOLOGIES. Published in European Scholar Journal (ESJ). Spain, Vol. 2 No. 6, June 2021, ISSN: 2660-5562. P.161-165. https://scholarzest.com/index.php/esj/article/view/964/826
10. Sh.Q. Shoyqulov., Methods for plotting function graphs in computers using modern software and programming languages. Published in ACADEMICIA An International Multidisciplinary Research Journal ISSN: 2249-7137, Vol. 11 Issue 6, JUNE 2021. India, P. 321-329, https://saarj.com/academicia-view-journal-current-issue/
11. Sh.Q. Shoyqulov., The graphics- is of the main components of multimedia technologies. Web of Scientist: International Scientific Research Journal (WoS), ISSN: 2776-0979, Vol. 3. No. 4 (2022): wos. Published: 2022-04-04. 1373-1381 p., DOI: https://doi.org/10.17605/OSF.IO/2KAM8