PYTHON ROUTINE FOR UNIFORM FLOW IN TRAPEZOIDAL OPEN CHANNELS

Lecturer PhD RALUCA MITROI, lecturer PhD ANCA ZABORILĂ,
lecturer PhD VALENTIN BOBOC, lecturer PhD MARIUS TELIŞCĂ,
prof. PhD. CATRINEL-RALUCA GIURMA-HANDLEY
"Gheorghe Asachi" Technical University of Iaşi, Romania

Abstract: *Natural rivers and man-made canals are open channels. They have many advantages over pipes and have been used for many centuries for water supply, for transport and for agriculture. [1] Critical and normal depth are two essential factors to consider when designing open channels. In trapezoidal channels, the governing equations are nonlinear for the normal and critical flow depths and thus the solution of the implicit equations involves numerical methods such as iterative ones (trial and error) and graphic methods. The complexity of equations often leads to an elevated risk of errors. A Python-based algorithm has been presented for the computation of uniform flow in open channels. This algorithm has the aim of enhancing students' comprehension of the subject matter and offers greater accuracy through automated iteration-solving. The algorithm's precision has been confirmed through a basic scenario, and its applicability is intended to extend to a broader range of channel shapes and flow conditions.*

Keywords: *trapezoidal open channel; Python; critical depth; normal depth;*

1. Introduction

The advancement of the automation calculation in the field of the hydraulics requires mathematical understanding and advanced programming to create a working code from the abstract concepts[1]. Recent advances in data science have led to the rise of Python usage that help with machine learning and deep learning tasks for hydraulics and hydrology. Feature engineering involves modifying the existing input data and generating new characteristics based on input data to improve data algorithms [2].

In hydraulic engineering, the study of open channel flow is crucial for understanding the behaviour of fluids in natural or man-made channels. Most of the flows with free surface, existing within hydrotechnical systems, are of unsteady type. But, on relatively small intervals of time and for some functional situations of the channels, the movement can be considered steady. Their analysis and calculation is carried depending on the spatial variation of flow parameters. In this case, the steady, turbulent movement of free-level flows can be either uniform or non-uniform[3].

Critical depth is a significant parameter in the designing and management of open channels and related hydraulic structures, understanding the flow characteristics and calculations of varied flows (gradually, spatially, etc.). The trapezoidal cross sections are the most commonly used geometric sections in the network of water transmission and distribution channels, thus discussing its geometrical and hydraulic parameters is inevitable. The used nonlinear and mathematical relationships governing the critical depth problem in the trapezoidal channels are implicit and complex, hence the methods of trial and error, graphical and numerical are used to calculate it[4].

Normal depth is of interest and plays an important role in designing the open channels, and its computation for a given set of variables is essential. Various explicit equations have been developed for normal flow depth in different channel sections.[5]

These depths are presently obtained by numerical method, manual trial-and-error method (time-consuming), graphical method (low accuracy owing to its log-scale representation) or by using explicit regression-based equations[6].

The main objective of this study was to develop a Python routine to generate the hydraulic computation of uniform flow in trapezoidal open channels. Results for different parameters were presented to validate the accuracy of the developed routine. The algorithm

can be a valuable tool for the hydrotechnical engineers or for researches that are working in the field of hydraulic modelling.

2.Materials and methods

To calculate the flow parameters in a trapezoidal open channel, a Python routine that include the principles of the hydraulic was created. The proposal of the work is the presentation of this routine, developed in Python language, for the hydraulic computation of uniform flow in trapezoidal open channels.

The methodology involves the following steps: collecting relevant equations, formulating the routine that takes input data and returns flow characteristics, implementing the routine by writing the code, verifying its functionality by comparing the results with manual methods, and optimizing the routine for improved performance.

Channels are determined by the interplay of geometric properties, representing the physical cross-sectional characteristics, and hydraulic factors, which comprehensively embody the principles of fluid dynamics governing flow behaviour within the channel (Fig. 1).

4. the hydraulic radius, denoted as R:

$$R = \frac{A}{P} \quad (4)$$

5. Chezy coefficient:

$$C = \frac{1}{n} R^{\frac{1}{6}} (5)$$

6. the flow carried by the channel:

$$Qc = (6)$$

The precision of the calculations for determining the normal depth is imposed by the following equation:

$$\delta_{adm} = 0{,}5\ ‰ = .$$

where: δ_{CQ} =.

In the case of the critical depth, the following equation must be in order to meet the precision criterion imposed.

$$\left| \frac{\frac{\alpha Q^2}{g} - \frac{A^3}{B}}{\frac{\alpha Q^2}{g}} \right| \cdot 100 \leq \delta_{C_Q} (8)$$

where: δ_{CQ} =.

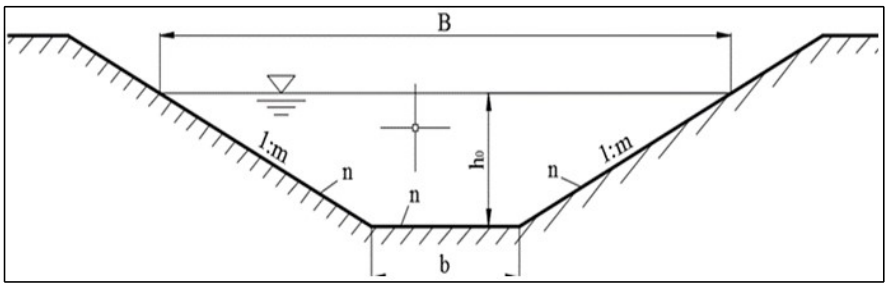

Fig. 1. Trapezoidal open channel

Trapezoidal channels, characterized by a side slope of 1:m, include the following geometric information: the channel's base width (b), side slope (m), and longitudinal slope (i). The hydraulic elements of this type of channel are:

1. the width of the channel section at the free water surface, denoted as B:

$$B = b + 2mh \quad (1)$$

2. the cross-sectional area of flow, denoted as A:

$$A = h(b + mh) (2)$$

3. the wetted perimeter, denoted as P:

$$P = b + 2h\sqrt{} (3)$$

The input data includes the channel's base width (b), side slope (m), longitudinal slope (i), roughness coefficient (n), Coriolis coefficient (α) and the required discharge (Q).

By providing the input data, the routine calculates values for the normal water depth (h0) and the determination of the flow regime based on the critical depth (hc), critical slope, and Froude number criteria. In the figures 2, 3 and 4 are illustrated parts of the script file.

The output parameter critical depth (hc)represents the depth at which the flow transitions from subcritical to supercritical or vice versa. The critical slope output parameter refers to the slope at which the flow regime transitions.

```
##### compute normal depth
h0 = 1
iterationCount = 1
while(True):
    A = h0 * (b + m * h0)
    P = b + 2 * h0 * (1 + m**2)**0.5
    R = A / P
    # Chezy coefficient
    C = (1 / n) * (R ** (1/6))
    Qc = A * C * (R * i) ** 0.5
    deltaQ = abs((Q - Qc) / Q) * 100
    if deltaQ <= deltaAdm:
        break

    h0 = h0 + normalDepthIncrement
    iterationCount += 1
print("\n")
print(f"Normal depth: {h0:.4f}\n")
```

Fig. 2. Partial extract from the routine listing – critical depth

```
##### critical slope
Pcr = b + 2 * hcr * (1 + m**2)**0.5
Rcr = Acr / Pcr
Ccr = (1 / n) * (Rcr ** (1.0 / 6.0))
iCr = (Q ** 2) / (Acr * Acr * Ccr * Ccr * Rcr)
print(f"\nicr = {iCr:.6f}")

if (i < iCr):
    print("Subcritical flow")
elif (i == iCr):
    print("Critical flow")
else:
    print("Supercritical flow")
```

Fig. 3. Partial extract from the routine listing – critical slope

While the Froude number criteriadetermine if the flow is subcritical (Fr < 1), critical (Fr = 1), or supercritical (Fr > 1).These outputs provide valuable insights into the behaviour and characteristics of the flow in the trapezoidal open channel.

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming.

Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an

```
##### Froude number
v0 = Q / A
Fr = (alpha * v0**2) / (g * h0)
print(f"\nFr = {Fr:.4f}")
if (Fr < 1):
    print("Subcritical flow")
elif (Fr == 1):
    print("Critical flow")
else:
    print("Supercritical flow")
```

Fig. 4. Partial extract from the routine listing – Froude number

ideal language for scripting and rapid application development in many areas on most platforms [7].

3. Results

In this section, extracts with input and output data and tables with the values obtained in the simulated situation are presented.

The example used consisted of the following input data: the channel's base width (b=3.5 m), side slope (m=2), longitudinal slope (I =0.7 ‰), roughness coefficient (n=0.014), Coriolis coefficient (α=1.1) and the required discharge (Q=50 m^3/s).

Critical depth and normal depth for the application mentioned above were computed using Python. In the case of normal depth, a total of 1469 simulations were necessary to ensure compliance with the specified limits. To enhance the precision of the calculation, the iteration step was set to 0.001.

A partial extract from the resulting table of values is presented in Table no. 1.

The correlation between normal depth and allowable error can be visualized through the results generated by the Python routine, as demonstrated in Figure 5.

To achieve a more accurate graphical representation, the generation of a significant number of additional iterations was chosen, equivalent to the initially generated ones – 1469 additional iterations.

It can be observed from the graph that after reaching the initial value, the graph exhibits a point of inflection.

From this point, a significant increase in the calculated error value can be observed, suggesting that the correct values were identified in the preceding steps.

For the computation of the critical depth, a number of 99 simulations were performed in order to meet the specified limits. Considering that the boundary limit for δCQ was 1, the iteration step was implemented as 0.01in order to ensure accurate results. The iteration process stopped when the precision criterion imposed was less than 1 (0.666). Thus, this iteration helped to refine the critical depth calculation.

The relation between the critical depth hcrand the allowable error can be observed through the outcomes produced by the Python routine, as illustrated in Figure 6. To enhance the precision of the graphical representation, it was chosen to generate an additional set of 99 iterations. From the graph, it is evident that after reaching the initial value, a point of inflection appears in the graph, indicating the fact that the results were found **previously.**

Identifying the water flow regime in open hydraulic systems represents an important factor intheir sizing and their verification. The flow regime through is determined by comparing the flow characteristics with critical ones. In the simulation performed using Python, the flow

Table 1. Partial extract from the results of the computation of normal depth

Simulation no.	h	A	P	R	C	Qc	δ_Q
1	1	5.5	7.972	0.69	67.143	8.115	83.769
2	1.001	5.508	7.977	0.69	67.152	8.131	83.738
3	1.002	5.515	7.981	0.691	67.161	8.146	83.708
4	1.003	5.523	7.986	0.692	67.17	8.162	83.677
5	1.004	5.53	7.99	0.692	67.179	8.177	83.646
6	1.005	5.538	7.994	0.693	67.188	8.193	83.615
7	1.006	5.545	7.999	0.693	67.197	8.208	83.584
8	1.007	5.553	8.003	0.694	67.206	8.224	83.553
9	1.008	5.56	8.008	0.694	67.215	8.239	83.522
10	1.009	5.568	8.012	0.695	67.224	8.255	83.491
1460	2.459	20.7	14.497	1.428	75.797	49.604	0.792
1461	2.46	20.713	14.501	1.428	75.802	49.647	0.706
1462	2.461	20.727	14.506	1.429	75.806	49.69	0.62
1463	2.462	20.74	14.51	1.429	75.81	49.733	0.534
1464	2.463	20.753	14.515	1.43	75.814	49.776	0.448
1465	2.464	20.767	14.519	1.43	75.818	49.819	0.361
1466	2.465	20.78	14.524	1.431	75.823	49.863	0.275
1467	2.466	20.793	14.528	1.431	75.827	49.906	0.188
1468	2.467	20.807	14.533	1.432	75.831	49.949	0.102
1469	**2.468**	**20.82**	**14.537**	**1.432**	**75.835**	**49.992**	**0.015**

Table 2. Partial extract from the results of the computation of critical depth

Simulation no.	h	A	B	A^3/B	$\alpha Q^2/g$	δ_{CQ}
1	1	5.5	7.5	280.326	22.183	92.087
2	1.01	5.575	7.54	280.326	22.983	91.801
3	1.02	5.651	7.58	280.326	23.805	91.508
4	1.03	5.727	7.62	280.326	24.648	91.207
5	1.04	5.803	7.66	280.326	25.514	90.899
6	1.05	5.88	7.7	280.326	26.402	90.582
7	1.06	5.957	7.74	280.326	27.314	90.256
8	1.07	6.035	7.78	280.326	28.249	89.923
9	1.08	6.113	7.82	280.326	29.209	89.58
10	1.09	6.191	7.86	280.326	30.193	89.229
90	1.89	13.759	11.06	280.326	235.518	15.984
91	1.9	13.87	11.1	280.326	240.384	14.248
92	1.91	13.981	11.14	280.326	245.329	12.485
93	1.92	14.093	11.18	280.326	250.351	10.693
94	1.93	14.205	11.22	280.326	255.454	8.873
95	1.94	14.317	11.26	280.326	260.637	7.024
96	1.95	14.43	11.3	280.326	265.901	5.146
97	1.96	14.543	11.34	280.326	271.248	3.238
98	1.97	14.657	11.38	280.326	276.678	1.301
99	**1.98**	**14.771**	**11.42**	**280.326**	**282.193**	**0.666**

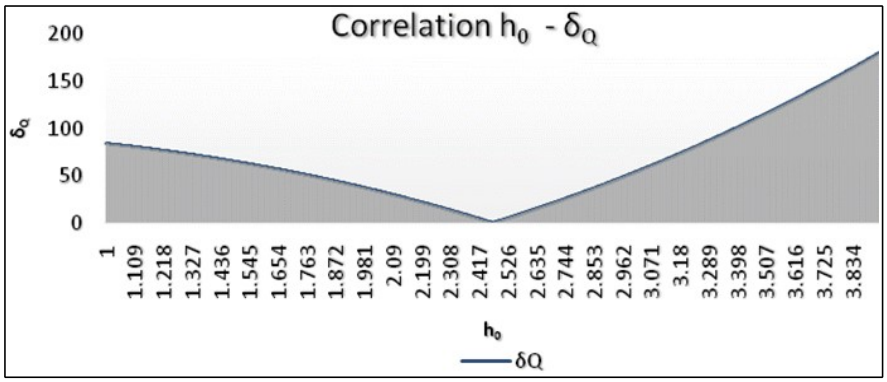

Fig. 5. Correlation h0 - δQ

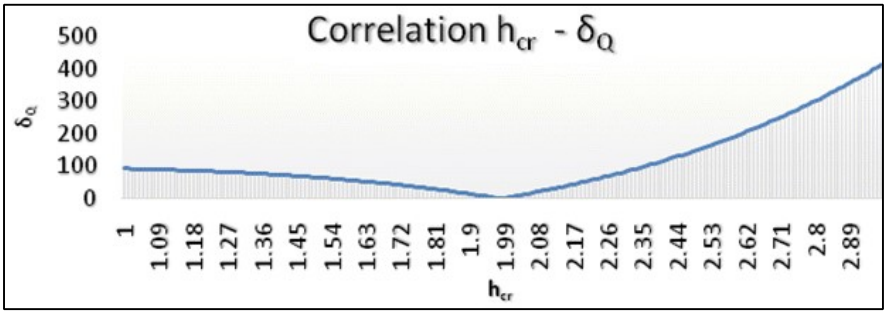

Fig. 6. Correlation hcr - δQ

regime was evaluated based on the critical slope criterion (icr) and the Froude number criterion.

The resulting values acquired from the simulationare as follows:

i=0,7 ‰ <icr=0.00177=1,77‰ **subcritical flow**

Fr=0.262<1 **subcritical flow**

4. Conclusions

In this article, a Python routine was presented with the aim to generate the hydraulic computation of uniform flow in trapezoidal open channels.

The routine incorporates empirical equations and provides information about the flow area, flow perimeter, hydraulic radius, flow velocity and flow rate.

Results for different parameters were presented to validate the accuracy of the developed routine as numerical values or in graphical form.

The presented routine represents a starting point for future studies of diverse channel sections under different types of flow. It can be further expanded or customized to suit specific needs. By leveraging this routine and adjusting the input parameters, you can analyse and predict the behaviour of flow in trapezoidal open channels efficiently and accurately.

References

1. Schnellbach, T.– *Hydraulic Data Analysis Using Python*, Master Thesis, Technische Darmstadt Universitat, 2022;
2. Ather, B., Laurie, B., Pachepsky, Y., Kyunghyun, K., Jong Ahn, C., Kyung Hwa, C.–AI4Water v1.0: *An open source python package for modelling hydrological time series using data-driven methods*, Geoscientific Model Development, 2021;

3. Bartha, I., Marcoie, N., Toma, D., Mitroi, R. – *Hydraulics course for civil engineers,* Editura Performantica, 2017;
4. Luca, M., *Hidraulică tehnică, Mişcarea permanentă în canale*, EdituraTehnopress, Vol. 1, Iaşi, 1998;
5. Ali R. Vatankhah - *Uniform flow depth in trapezoidal open channels*, Flow Measurement and Instrumentation, Volume 94, December 2023, 102458;
6. Varandili, S.A., Arvanaghi, H., Ghorbani, M.A., Yaseen, Z.M. *–A novel and exact analytical model for determination of critical depth in trapezoidal open channels*, Flow Meas. Instrum., 68 (2019), p. 101575;
7. The Python Tutorial: https://docs.python.org/3/tutorial/index.html#tutorial-index.