



Article

Can ChatGPT Generate Geotechnical Engineering Code? A Human-in-the-Loop Validation Study

Sophia Einspänner ¹, Reza Taherdangkoo ^{1,2,*} and Christoph Butscher ^{1,2}

¹ Institute of Geotechnics, TU Bergakademie Freiberg, Gustav-Zeuner-Str. 1, 09599 Freiberg, Germany; sophiaeinspa@outlook.de (S.E.); christoph.butscher@ifgt.tu-freiberg.de (C.B.)

² Freiberg Centre for Water Research (ZeWaF), TU Bergakademie Freiberg, 09599 Freiberg, Germany

* Correspondence: reza.taherdangkoo@ifgt.tu-freiberg.de

Abstract

Large language models are increasingly used to generate engineering code, but executable output does not establish correctness. This study examines an expert supervised workflow used to develop Python implementations for two analytical geotechnical problems: a prescribed circular slip surface calculation using the simplified Bishop method and a shallow foundation bearing capacity calculation. The workflow comprised problem decomposition, specification of geotechnical constraints, modular code generation, expert diagnosis, prompted correction, visual inspection, and comparison with independently configured reference calculations. The slope example documents implementation choices and failure modes for one prescribed slip surface. The bearing capacity study exercised five test groups and 44 calculations covering homogeneous soil, groundwater, two-layer profiles, horizontal loading, and limiting cases. The development process revealed safety-relevant failure modes, including syntax errors, unstable iterations, incorrect geometric extrapolation, and physically inadmissible failure mechanisms. These errors were not resolved reliably by autonomous model self-correction, but required expert diagnosis, modular testing, and constraint-based prompting. The results show that ChatGPT can support the development of geotechnical calculation tools for bounded analytical verification tasks, but only within a strict human-in-the-loop framework. The findings should be interpreted as a case-specific validation of one model system rather than as evidence for the reliability of large language models in general.

Keywords: large language models; ChatGPT; geotechnical code generation; human-in-the-loop; software validation

Academic Editor: Yunfeng Zhang

Received: 17 June 2026

Revised: 24 July 2026

Accepted: 29 July 2026

Published: 30 July 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Recent developments in large language models (LLMs) offer a new means of producing geotechnical calculation tools by translating natural-language instructions into executable code [1]. Geotechnical studies have considered LLMs for education, domain adaptation, and design assistance [2–4]. However, successful execution does not establish that generated code correctly represents the specified geotechnical formulation [1,5]. Reliable use therefore depends on verification of the mechanical and numerical formulation under expert control.

Geotechnical analyses are commonly conducted using specialized commercial software. These programs provide established computational implementations, but their

source code and many internal numerical procedures are generally not available for independent inspection. By contrast, source code available to the user can be inspected, modified, and tested. Direct access to source code facilitates the development of calculation tools for specific problems without relying exclusively on closed software environments.

Recent studies have used ChatGPT to develop geotechnical numerical models and calculations based on design standards [5–8]. In several reported examples, the generated code required iterative correction and expert input before agreement with reference results was obtained. Executable implementations also produced incorrect results when the mathematical formulation or numerical treatment was not represented correctly [6,7]. More generally, evaluations of code generation have shown that limited test sets can fail to detect functionally incorrect code [1]. Final code and numerical results therefore provide insufficient evidence of how implementation errors were identified and corrected. Assessment must consider the development and verification process [3,5,6].

A valid implementation must preserve the intended mechanical model and correctly represent the relevant problem conditions and design rules. Implementation errors may not cause an obvious runtime failure; the code may still return plausible but mechanically inconsistent or unconservative results. Evaluation must therefore combine inspection of the formulation with comparison against mechanically relevant reference cases and documentation of identified failure modes [9]. Responsibility for the formulation and final acceptance of the code remains with the engineer.

This study examines the use of ChatGPT to generate geotechnical calculation scripts within a workflow conducted under expert supervision. Two analytical problems are considered: evaluation of a prescribed circular slip surface using the simplified Bishop method and calculation of the bearing capacity of shallow foundations. The contribution is a geotechnical verification of the Python implementations and an analysis of the observed failure modes. The study documents where executable code required expert intervention and identifies the verification conditions applied before the scripts were accepted for the investigated cases. The resulting evidence is limited to the accepted implementations and investigated cases and does not establish the reliability of future code generations by ChatGPT.

2. Human–Model Interaction

2.1. Human-in-the-Loop Control

Code development was conducted under expert supervision. The engineer specified the calculation procedure and acceptance criteria, evaluated the outputs, and diagnosed the source of any mechanical or numerical discrepancy before requesting a revision. ChatGPT generated candidate code from the supplied requirements but did not determine whether the output was acceptable. Individual modules were reviewed and tested separately and, where available, compared with relevant reference quantities before integration into the complete implementation. Figure 1 summarizes the division of responsibility within the workflow.

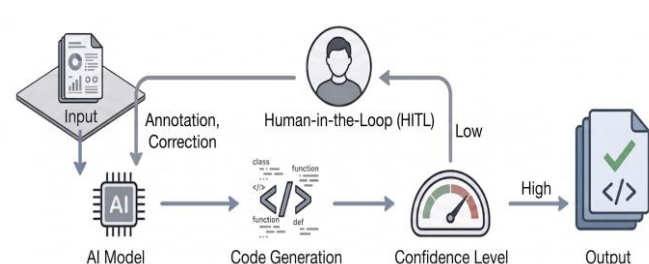


Figure 1. Workflow under expert supervision for generating geotechnical calculation code using ChatGPT. The engineer specifies the calculation, reviews the generated code, diagnoses errors, and accepts the script only after the checks applicable to the investigated case have been completed.

2.2. Prompt Engineering

Prompt engineering was treated as a form of engineering specification. The prompts defined the intended geotechnical formulation, applicable standards, numerical procedure, and acceptance conditions. Generic requests such as “write a slope stability program” or “calculate the bearing capacity” left essential engineering decisions unspecified. The workflow therefore used constrained and modular prompts followed by inspection and reference checks [3,5–7]. Table 1 summarizes the prompt controls used.

Table 1. Prompt controls used in the workflow under expert supervision.

Control	Application in This Study	Risk Addressed
Task definition	The intended geotechnical calculation and programming task were specified.	Misinterpretation of the requested scope
Modular development	The calculation was divided into functions that could be reviewed separately.	Errors concealed within integrated code
Physical constraints	Admissible inputs, special cases, and safety factor logic were specified.	Invalid inputs and incorrect calculation branches
Geometry specification	The discretization and admissible failure mechanisms were defined explicitly.	Mechanically inadmissible geometry
Code freezing	Later prompts were instructed not to modify previously checked functions.	Regression during staged development
Reference comparison	Candidate revisions were accepted only after comparison with relevant reference quantities.	Acceptance of plausible but incorrect results

The code freezing rule addressed regression during staged development. Once a backend function had passed the reported checks, later additions were implemented through separate routines where possible. This procedure did not establish that the accepted functions were free from errors, but it limited unintended changes during subsequent development.

For the simplified Bishop method, the prompt specified an iterative solution because the factor of safety appears on both sides of the equilibrium equation [10,11]:

$$F = \frac{\sum_i \left[\frac{c'_i b_i + (G_i - u_i b_i) \tan \phi'_i}{\cos \theta_i + \frac{\tan \phi'_i \sin \theta_i}{F}} \right]}{\sum_i G_i \sin \theta_i} \quad (1)$$

The prompt defined the convergence controls required for the Bishop iteration. For the bearing capacity calculation, the prompts specified the conditional branches governing foundation geometry, loading, ground conditions, and design situation. Prompting therefore served as an engineering specification, while acceptance depended on staged verification against the intended formulation and relevant reference results.

3. Expert Supervised Development Workflow

3.1. Development Documentation and Reproducibility

Code generation was conducted through the ChatGPT web interface using o4-mini-high in July 2025 and GPT-5 Thinking between August and December 2025. The bearing capacity code was executed with Python 3.13.5, NumPy 2.3.2, Matplotlib 3.10.5, and Tkinter 8.6. Reference calculations used GGU-FOOTING 9.15 and DC-Grundbruch 24.2.5 [12,13].

Complete interaction histories were not retained. Reproducibility is therefore limited to the archived accepted code, reported software environment, documented prompt excerpts, stated assumptions, and available calculation records. The current code version was archived after each major development stage. Candidate code was accepted only after

successful execution, inspection against the intended geotechnical formulation, and, where available, comparison with independently checked intermediate quantities or final reference results. Corrections were introduced through targeted prompts following expert diagnosis. Representative prompt excerpts are provided in Appendix A.

3.2. Development Stages

The tasks were decomposed into modules for input handling, geometry, ground and water conditions, calculation routines, and output. Each module was generated, inspected, and tested before subsequent functionality was added. Mechanical and normative assumptions were specified before the corresponding calculation routines were accepted.

When a module produced an invalid result, the correction prompt addressed the identified mechanical or numerical source rather than only the incorrect final value. Functions that had passed the relevant checks were protected from later modification where possible, while plotting and interface components were added separately. The completed implementations were assessed through code inspection, graphical plausibility checks, and comparison with independent reference calculations where applicable.

4. Benchmark Verification and Reference Comparison

The accepted implementations were assessed using checks defined for each example. The slope assessment examined the implementation and reported output for one prescribed slip surface, whereas the bearing capacity assessment compared intermediate and final outputs with independent software calculations. For the bearing capacity comparisons, a relative difference of 2% was used as an internal screening level rather than as an allowable design error.

4.1. Slope Stability

The slope example examined the implementation of the simplified Bishop equation for one prescribed circular slip surface based on the geometry in DIN 4084 Supplement 1 [10,11,14–16]. The model contains nonhorizontal terrain, a berm, two soil layers, and a phreatic surface (Figure 2). The terrain was represented by piecewise linear interpolation. The prescribed circle passes through (0, 0) m and (36, 14) m and has its center at (11.69, 35.00) m with a radius of 36.90 m. The sliding mass was divided into ten vertical slices. Slice tops were clipped to the terrain, pore pressure was calculated from the groundwater elevation above the slice base, and strength parameters were assigned according to the location of the slice base. One transition slice was assigned width-weighted parameters.

Each of the ten slices had a width of 3.6 m. The calculation used an initial factor of safety of 1.0, a relative stopping criterion of 0.03, and a maximum of 100 iterations. Slice weight was calculated from the average upper and lower elevations, and the simplified Bishop equation was solved iteratively. Table 2 reports the slice quantities. The accepted implementation produced $F = 1.1495$ after two iterations.

Figure 3 shows the result produced by the Python notebook. GGU-STABILITY returned a critical factor of safety of $F = 1.10$ by searching for a critical surface within the same overall geometry. The GGU surface differs from the prescribed circle evaluated by the Python notebook, so the two values are not treated as a direct numerical comparison. Figure 4 shows the independently obtained critical mechanism.

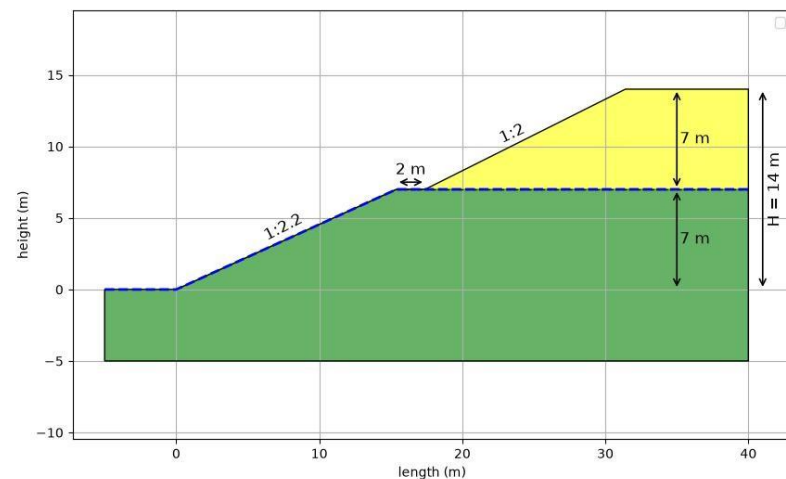


Figure 2. Two-layer slope geometry based on DIN 4084 Supplement 1. The model includes a 14 m high slope, a 2 m berm, two soil layers, and a groundwater table intersecting the stratigraphic boundary.

Table 2. Slice quantities reproduced from the accepted Bishop notebook for the prescribed circle. W is the slice weight, ub is the pore pressure resultant along the horizontal slice width, R is the resisting contribution to the Bishop numerator, and $D = W \sin \alpha$.

Slice	W (kN/m)	ub (kN/m)	α (°)	c (kPa)	φ (°)	R	D
1	78.4	39.2	−8.55	18.00	18.30	82.2	−11.7
2	222.0	111.0	−2.75	18.00	18.30	103.0	−10.6
3	339.2	169.6	3.03	18.00	18.30	119.2	17.9
4	429.9	215.0	8.84	18.00	18.30	131.6	66.1
5	461.6	233.6	14.74	18.00	18.30	134.8	117.5
6	474.4	191.9	20.81	18.00	18.30	152.6	168.6
7	494.6	133.9	27.13	18.00	18.30	180.3	225.6
8	477.0	57.1	33.84	18.00	18.30	205.6	265.6
9	389.2	0.0	41.12	2.43	27.93	203.5	255.9
10	159.5	0.0	49.33	0.00	29.30	87.6	121.0

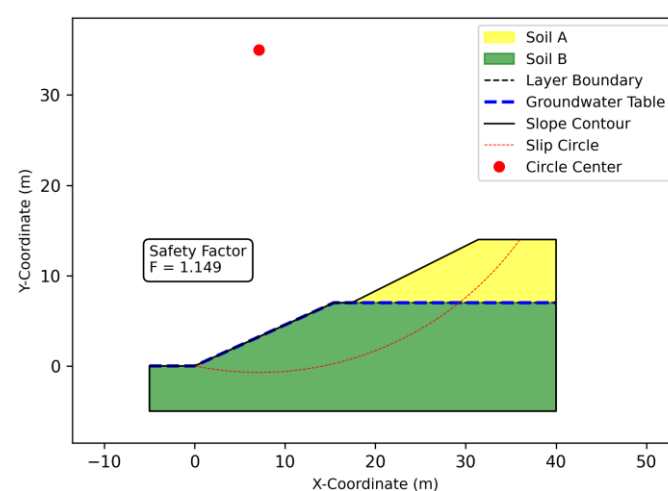


Figure 3. Slope stability calculation produced by the Python notebook.

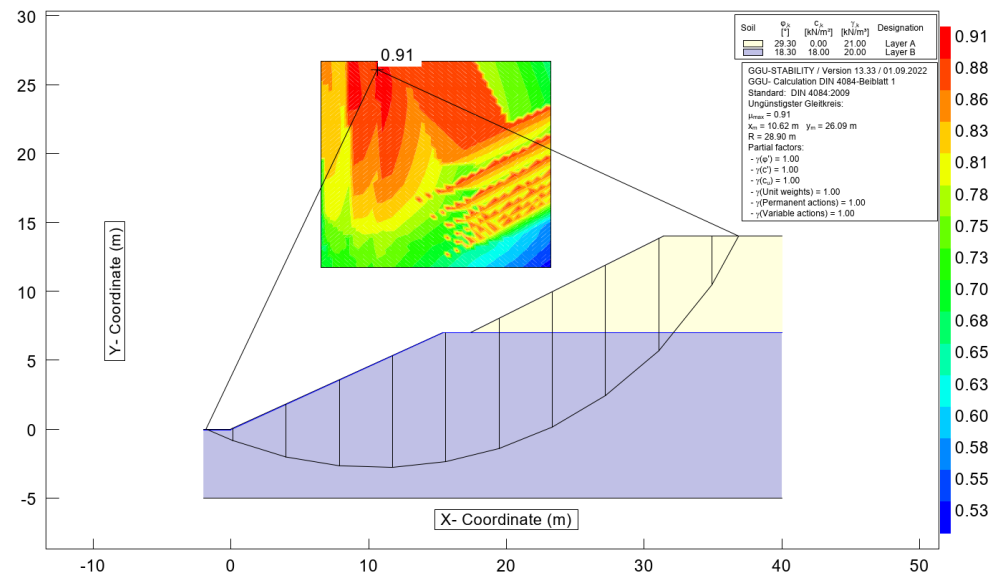


Figure 4. Critical slip surface obtained independently with GGU-STABILITY for the same overall slope geometry. The surface differs from the prescribed circle evaluated by the Python notebook.

4.2. Bearing Capacity

The bearing capacity calculation provides the principal external comparison. The assessment followed DIN 4017 and the German safety concept under Eurocode [11,14–16]. Five test groups comprised eight scenarios and 44 calculations. Reference calculations were performed using GGU-FOOTING 9.15 and, for strip foundations, DC-Grundbruch 24.2.5. Intermediate quantities, final resistances, and utilization ratios were compared. Table 3 reports the maximum relative difference within each group.

Table 3. Summary of bearing capacity comparisons with DC-Grundbruch and GGU-FOOTING.

Case	Scenario	Maximum Difference
TF01	Homogeneous ground; rectangular and strip foundations; BS-P and BS-T	0.0%
TF02	Groundwater varied from the ground surface to the foundation base	0.0%
TF03	Two-layer ground; interface depth varied from 3 to 14 m	1.96%
TF04	Horizontal load varied from 0 to 3000 kN	≈1.50%
TF05	Limiting friction angle cases, including $\varphi = 0^\circ$	0.05% at $\varphi = 0^\circ$; < 0.4% for reported N-factors

The homogeneous and groundwater cases reproduced the reference outputs after rounding. For the two-layer profile, the relative difference varied with interface depth and reached 1.96% at a depth of 8.0 m below ground level. The difference decreased to zero when the failure mechanism was located entirely within one layer and was largest when both layers influenced the equivalent parameters. This pattern is consistent with sensitivity to the geometric averaging and iterative equivalent-parameter procedure. Agreement with the commercial implementation does not, however, establish that either implementation is exact.

For horizontal loading, the relative difference in characteristic bearing resistance increased with load and reached approximately 1.50% at $Hk = 3000$ kN. The corresponding utilization ratios differed by up to approximately 0.01 after rounding. The results confirm that the branches for layered ground and inclined loading required checks distinct from those applied to the homogeneous cases. An example output for validation case TF 04 is shown in Figure 5.

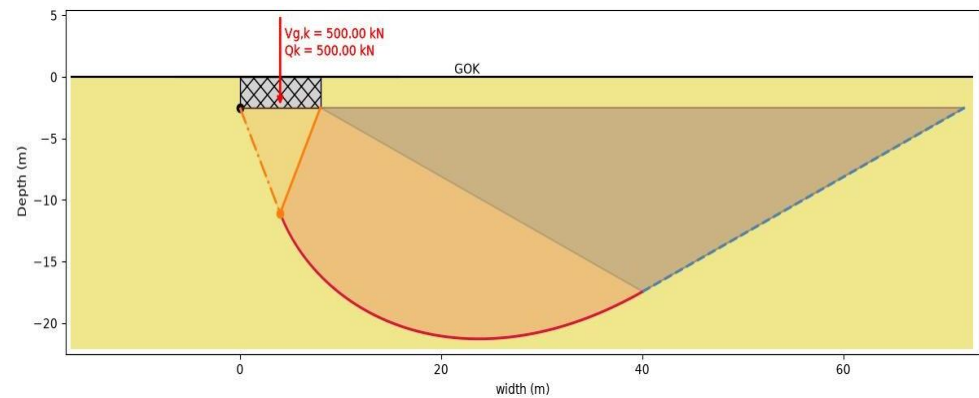


Figure 5. Output from the bearing capacity implementation for comparison case TF04 under horizontal loading.

5. Failure Modes and Corrective Measures

The observed failures were classified according to whether they prevented execution or allowed the code to run while violating the intended geotechnical formulation. The second category was more critical because it could produce plausible numerical results without an obvious program error.

Table 4 summarizes the failures identified during development and the corrective measures applied. Providing ChatGPT only with the incorrect final result was insufficient when the error concerned the mechanical or numerical formulation. In some cases, the revised output changed the calculation logic without correcting the underlying source of the discrepancy. Correction prompts were therefore formulated after expert diagnosis, and the revised module was checked again against the intended formulation. The applied measures addressed the identified failures, but their individual effects were not evaluated separately and do not certify the generated code.

Table 4. Observed failure modes and corrective measures.

Failure Mode	Observed Problem	Corrective Measure
Geometric inconsistency	Inadmissible slip surfaces or bearing capacity mechanisms were generated.	Geometric constraints were specified explicitly, and the resulting mechanisms were inspected visually.
Terrain extrapolation	Slices extended beyond the defined slope surface.	Slice tops were limited to the interpolated terrain profile.
Incorrect discretization	The slip surface was divided by arc length rather than into vertical Bishop slices.	Vertical slices with constant horizontal width were specified.
Iteration instability	Bishop iteration failed for poorly conditioned circles.	Iteration limits, denominator checks, and explicit failure conditions were introduced.
Geometry library exception	Tangential intersections returned geometry object types that were not handled by the initial code.	Object type checks and exception handling were added.
Syntax and notation error	Mathematical notation was occasionally inserted directly into Python code.	Candidate modules were executed and inspected before integration.
Regression during extension	New plotting or interface functions modified previously accepted calculation routines.	Accepted functions were protected from modification, and additional features were introduced separately.
Inadequate input handling	Invalid dimensions, loads, or decimal formats were not consistently rejected.	Input checks were added for the identified cases.

6. Limitations

This study evaluates the accepted versions of two implementations for deterministic analytical examples. It does not compare language models, repeated code generations, alternative prompting configurations, automated development workflows, advanced AI-assisted strategies, or independently written expert code. In particular, no API-based verification loop was implemented to execute generated scripts, compare their outputs with reference results, and return discrepancies for iterative revision. Such a workflow could automate testing and regression checks, but its reliability would depend on the quality and coverage of the reference cases, fixed acceptance criteria, and safeguards against modification of tests or tolerances. This study therefore cannot estimate generation success rates, differences among models, or gains in development efficiency. Other approaches, including automated test generation, retrieval assisted prompting, tool integrated execution, and multi-agent code review, were also not investigated. The findings should not be generalized to more complex numerical analyses or engineering design applications.

The slope stability evidence is particularly limited. Only one prescribed circular slip surface divided into ten slices was evaluated. The archived notebook uses a relative iteration stopping criterion of 3% and does not include a study of slice number or convergence sensitivity, an external calculation of the same prescribed circle, or a search for the critical slip surface. The slope example therefore documents the accepted implementation and its observed failure modes but does not establish numerical accuracy or robustness across other slope configurations.

The bearing capacity evidence covers a larger set of cases but remains limited to the parameter ranges investigated and the selected commercial reference programs. The 2% screening level was an internal criterion for identifying discrepancies and does not represent a normative tolerance. Complete interaction histories and prospective development metrics were not retained. The supplied software also lacks an automated test suite and a formal assessment of maintainability and test coverage. Reproducibility is consequently limited to the archived code, reported software environment, stated cases, prompt excerpts, and available calculation records. Use in engineering design would require independent verification, controlled software management, documented testing, and review by a qualified engineer.

7. Conclusions

The findings support the following conclusions.

- ChatGPT supported the development and revision of geotechnical calculation code when the analytical formulation, applicable standards, assumptions, and acceptance criteria were defined by an engineer. This does not demonstrate autonomous geotechnical competence.
- The most consequential failures were executable implementations that violated the intended geometry, calculation branches, or numerical formulation. Successful execution and plausible output were therefore insufficient to establish mechanical validity. Expert diagnosis, formulation checks, graphical inspection, and reference calculations remained necessary.
- The scripts should be regarded as inspectable research artifacts rather than autonomous design software. Their practical value remains conditional on domain-specific verification, and use in engineering design would require further testing, controlled software management, and independent review.

Author Contributions: S.E.: Methodology, Formal analysis, Validation, Investigation, Writing—Original & Draft. R.T.: Writing—Review & Editing. C.B.: Conceptualization, Supervision, Writing—Review & Editing. All authors have read and agreed to the published version of the manuscript.

Funding: R.T. receives funding from the Deutsche Forschungsgemeinschaft (DFG) under grant no. TA 2076/1-1.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The source code and reported verification data are available in the repositories listed in the Code Availability Statement: Slope stability: https://github.com/sophia-eins/slope_stability_AI_Calculation.git (accessed on 28 July 2026). Bearing capacity: https://github.com/sophia-eins/Ground_Failure_AI_Calculation.git (accessed on 28 July 2026).

Acknowledgments: Open Access Funding by the Publication Fund of the TU Bergakademie Freiberg.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Exemplary Prompts

Representative prompt excerpts are documented below to illustrate initial task specification, targeted correction, and invalid model output. They are not a complete interaction log and cannot support quantitative conclusions about prompt efficiency or correction frequency. The executable scripts, rather than exact regeneration of model responses, are the reproducible objects of this study.

Appendix A.1. Example 1: Initial Task Specification

Description: This prompt demonstrates the definition of the model role, task context, and code freezing instruction used to align the generated output with the target calculation task.

Context: Initialisation of the basic calculation logic for the normative foundation failure analysis.

Prompt:

‘Imagine you are a geotechnical engineer with a lot of programming experience. Create an interactive Python script to calculate safety against foundation failure for shallow foundations according to Eurocode 7. We already have a functioning code block for querying the foundation dimensions, which should be kept exactly as is. Now add the logic for the design situations (BS-P, BS-T, BS-A/BS-E) and assign the correct partial safety factors (γ_G , γ_Q , γ_M). The script must check whether the foundation is a rectangular or strip foundation and adjust the calculation accordingly.’

Result: Based on this prompt, the language model generated error-free input loops for user input and implemented the correct partial safety factors strictly according to standard specifications.

Appendix A.2. Example 2: Correction Prompt for Layered Soil Averaging

Description: This prompt shows how precise geometric instructions were used to revise an incorrect solution approach.

Context: Correction of a severe integration error in the iteration for area averaging of the effective friction angle in two-layer soils.

Prompt:

‘In the current calculation of the averaged unit weight and friction angle, the area of the slices is only estimated by linear height proportions of the triangles. This leads to an unacceptable deviation. Correct the function for iterative averaging. Instead, clip the complete failure body polygon exactly at the layer boundary using the *shapely* geometry li-

brary and scale the partial areas from the actual geometric intersection to calculate the parameters weighted by length and area.'

Result: The correction produced the geometric clipping and equivalent-parameter procedure used in the accepted code. In the reported two-layer comparison set, relative differences varied with interface depth and reached 1.96%; the correction therefore reduced the earlier error but did not produce exact agreement for every layered case.

Appendix A.3. Example 3: Invalid Model Output for Prandtl Spiral Construction

This example documents a safety relevant failure in which the generated code combined invalid syntax with a physically inadmissible failure mechanism.

Context: Autonomous generation of code for the construction of the logarithmic Prandtl spiral by the LLM.

Incorrect Output of the LLM:

```
alpha pass deg = 45.0 - 0.5 * phi deg m pass = -math. tan(deg2rad (alpha pass deg)) r1 =
r2 * exp ((pi/180) * v * tan(phi)) print (f'→ G = {y_G:.2 f } gamma Q = {gamma Q:.2 f}')
```

Interpretation of the safety relevant error: The generated output mixed Python syntax with non-executable mathematical notation and produced a physically inadmissible geometric construction. The syntax problem would stop execution, but the more relevant engineering issue was that the constructed spiral curved into the positive y domain, outside the soil body, instead of penetrating the ground. The human supervisor therefore provided a targeted correction prompt specifying that valid Python syntax had to be used and that the spanning angle of the spiral had to be defined so that the slip surface remained within the subsoil.

References

1. Liu, J.; Xia, C.S.; Wang, Y.; Zhang, L. Is your code generated by ChatGPT really correct? Rigorous evaluation of large language models for code generation. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 21558–21572.
2. Chen, L.; Hettiyadura, U.; Kodikara, J.; Tophel, A. An investigation into the utility of large language models in geotechnical engineering education. *Geotechnics* **2024**, *4*, 441–461.
3. Fan, L.; Liu, F.; Chen, C. Domain adaptation of large language models for geotechnical applications. *Solid Earth Sci.* **2026**, *11*, 100285.
4. Xu, H.R.; Zhang, N.; Yin, Z.Y.; Guy Atangana Njock, P. GeoLLM: A specialized large language model framework for intelligent geotechnical design. *Comput. Geotech.* **2025**, *177*, 106849.
5. Kim, S.; Kim, D.; Youn, H. Automating vertical bearing capacity calculations using Python: Prompt engineering of ChatGPT on API RP 2A. *Dev. Built Environ.* **2025**, *21*, 100628.
6. Kim, D.; Kim, T.; Kim, Y.; Byun, Y.H.; Yun, T.S. A ChatGPT MATLAB framework for numerical modeling in geotechnical engineering applications. *Comput. Geotech.* **2024**, *169*, 106237.
7. Kim, T.; Yun, T.S.; Suh, H.S. Can ChatGPT implement finite element models for geotechnical engineering applications? *Int. J. Numer. Anal. Methods Geomech.* **2025**, *49*, 1747–1766.
8. Kwak, J.; Won, J. Application of ChatGPT in seepage induced slope stability. *KSCE J. Civ. Eng.* **2026**, *30*, 100457.
9. Taherdangkoo, R.; Arab, A.; Shakiba, S.; Doulati Ardejani, F.; Nagel, T. What Does “Good Performance” Mean in Geotechnical Machine Learning? *Geomech. Energy Environ.* **2026**, *47*, 100865.
10. Bishop, A.W. The Use of the Slip Circle in the Stability Analysis of Slopes. *Geotechnique* **1955**, *5*, 7–17.
11. DIN 4084:2021-11; Baugrund–Geländebruchberechnungen. DIN: Berlin, Germany, 2021.
12. GGU-FOOTING Version 10: Analysis of Footings to DIN 4017, DIN 4019, DIN 1054 and EC 7, version 10; Civilserve GmbH: Steinfeld, Germany, 2022.
13. GGU-STABILITY Version 14: Slope Stability Analysis and Analysis of Soil Nailing and Reinforced Soil Walls to DIN 4084 and EC 7, version 14; Civilserve GmbH: Steinfeld, Germany, 2022.
14. DIN 1054:2021-04; Baugrund—Sicherheitsnachweise im Erd- und Grundbau—Ergänzende Regelungen zu DIN EN 1997-1, Deutsches Institut für Normung e.V.: Berlin, Germany, 2021. Available online: <https://www.dinmedia.de/en/standard/din-1054/335418422> (accessed on 28 July 2026).

15. *DIN EN 1997-1*; Eurocode 7: Entwurf, Berechnung und Bemessung in der Geotechnik—Teil 1: Allgemeine Regeln. Beuth Verlag GmbH: Berlin, Germany, 2022.
16. *DIN 4017*; Baugrund—Berechnung des Grundbruchwiderstands Unter Flachgründungen. Beuth Verlag GmbH: Berlin, Germany, 2025.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.