A free educational software application for the design of concrete dam components

Michael Dupuis[1]
[1] Geosyntec Consultants Inc.

Abstract: Detailed design packages are required to ensure that structural components of concrete dams satisfy applicable design guidelines. These design packages are important for the design and analysis of appurtenant dam structures—such as concrete anchors, spillway and powerhouse slabs, retaining and training walls, powerhouse beams, and foundations—because they are the primary record of design and capacity checks by the engineer of record. However, these design checks are tedious and time consuming to complete, especially for unique one-off structural elements which are common for dams. Furthermore, project changes can require multiple iterations of a design concept which can result in repetition of these design checks prior to final design. Commercially available software programs are available for structural design checks, and partially address this problem; however, these programs suffer from three main shortcomings: (i) lack of an application programing interface limits the ability to rapid prototype design alternatives, (ii) awkward compilation of multiple structural components into well-organized project design documents, and (iii) trial student licences have limited durations and full licenses are not freely available. To address these shortcomings, I have developed a freely available educational software program which can be used as either a graphical user interface or as an application programming interface. The stand-alone graphical user interface requires no programming knowledge and can be run from a downloaded executable file. The application programming interface can be accessed as a Python module and provides additional power for engineers knowledgeable in Python to interact with the software directly through their own Python scripts. The software is provided for educational use only with the hope that it will motivate companies and organizations within the hydropower industry to develop similar technologies.

Keywords: Concrete Dam, Structural Design, Python, Graphical User Interface, Application Programming Interface

Introduction

Structural design calculation packages are time consuming to produce, repetitious, and susceptible to required rework due to project changes; however, these design reports are critically important for the design and analysis of structural components of concrete dams. Commercial software packages are available which partially address these challenges; however, these software packages are expensive, limited by the constraints of a graphical user interface, and have poor integration of the design and analysis results into the narrative of a structural design package.

To address these shortcomings, I have developed a software program called Structural Design Package (SDP). This name is intended to have a double meaning, with "package" referring to both the software program itself and the pdf design documents which the software program produces.

The program offers two alternate methods through which a user can interact with the software. A compiled executable (*.exe*) is provided for non-programmers to run the program and interact with the software through a graphical user interface (GUI), as shown in Figure 1. For advanced users with knowledge of Python, the program offers a convenient application programming interface (API), as shown in Figure 2, which offers greater analysis power and flexibility.

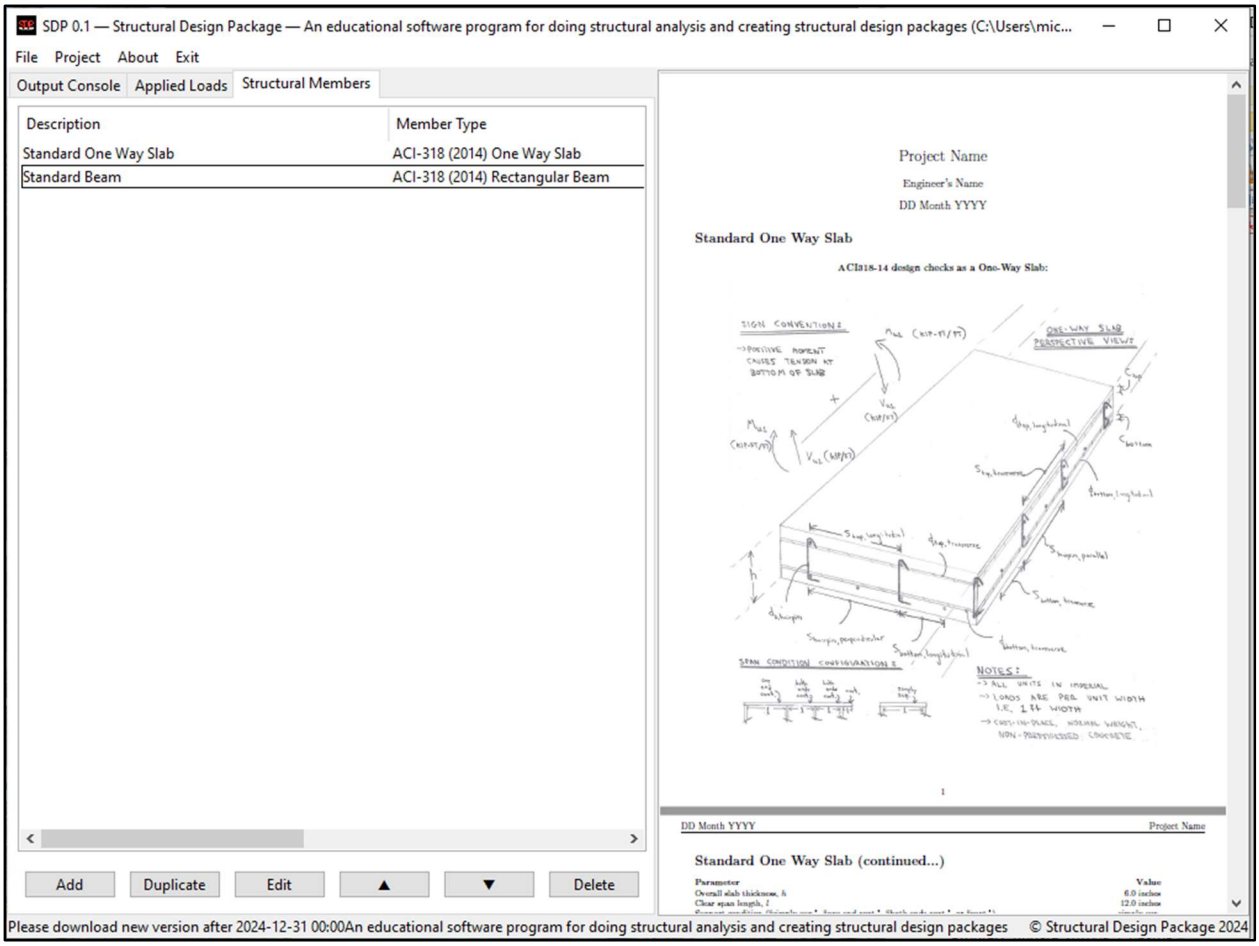

Figure 1: Graphical user interface of the compiled executable showing example elements (left panel)) and a rendering of a compiled pdf design report (right panel).

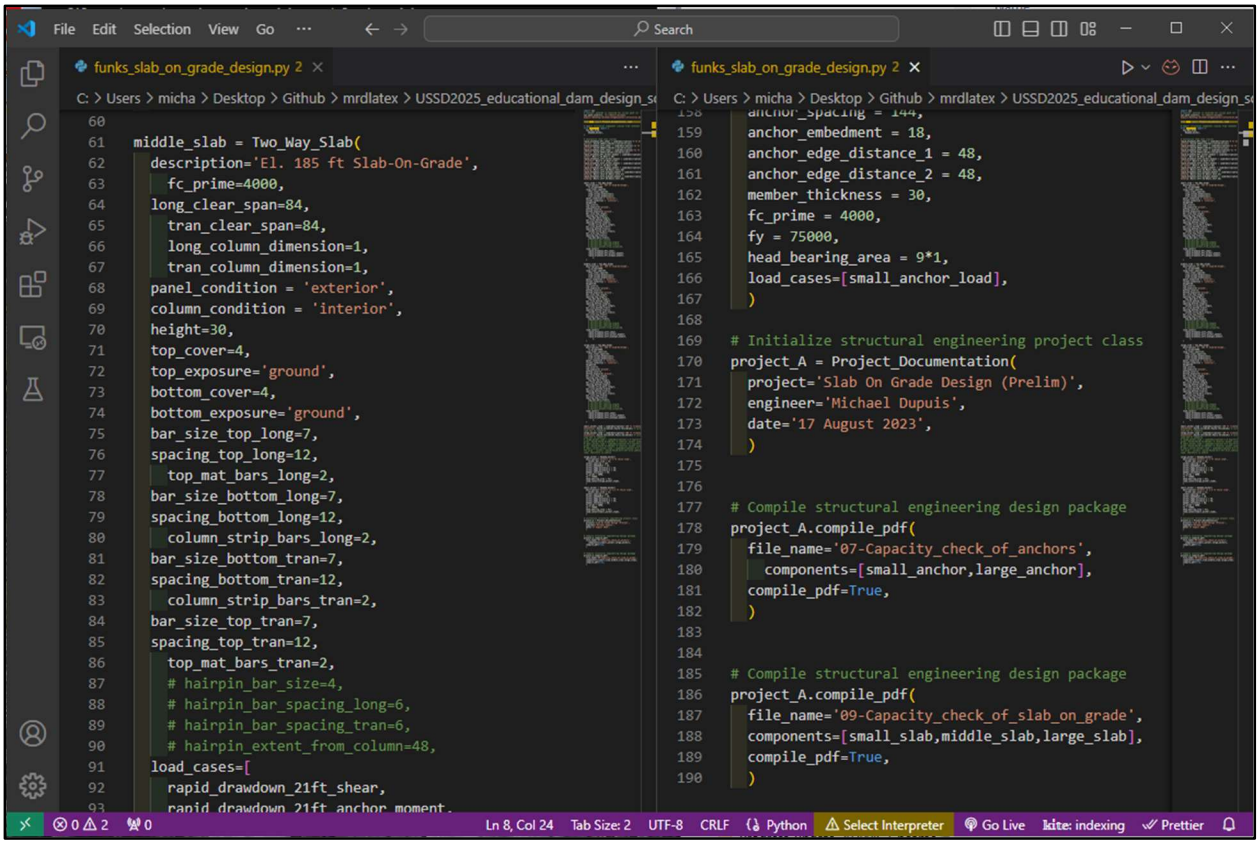

Figure 2: Application programming interface showing instantiation of Member classes (left panel), Project classes (right panel), and compilation of pdf design reports (right panel).

SDP is distributed for educational purposes only and is not to be used for commercial or professional purposes. The program is provided without warranty of any kind and has been developed with the intent to motivate other organizations, companies, and individuals in hydropower industry to develop similar tools for their own work and pleasure.

Background

SDP leverages free and open-source programs, primarily Python and LaTeX, to provide a simple yet powerful platform from which structural design packages can be rapidly and consistently produced.

Python is a high-level programming language which emphasizes code readability and simplicity, making it accessible to both novice programmers and seasoned professionals. As an open-source language, Python is freely available and continuously enhanced by a robust community of developers. The language's architecture supports cross-platform portability, ensuring consistent behavior irrespective of the operating system (e.g., Windows, macOS, GNU/Linux, etc.). Python supports version control through Git, which facilitates collaborative development and efficient version control which is essential for maintaining the integrity of software projects throughout their development.

Within Python, wxPython is a GUI toolkit for the Python programming language. WxPython is portable across all major desktop platforms; therefore, the software will maintain a consistent look and feel regardless of operating system. This toolkit was used to develop a GUI and another Python package, PyInstaller, converts Python applications into stand-alone executables, thus making them easily accessible to non-programmers. PyInstaller is free, open-source, and supports the locking of source code to protect against unauthorized modifications. Like wxPython, it is actively developed and utilizes version control, ensuring robustness and security for deployment in various environments.

The coupling of Python with LaTeX, an independent high-quality typesetting system, offers substantial benefits, particularly in scenarios requiring automated, repeatable, and precise document creation. LaTeX is notoriously cumbersome, difficult to debug, and susceptible to typos; however, the use of Python to automate and control LaTeX processing transforms the cumbersome manual coding process into a more efficient, programmatically controlled workflow, thereby harnessing the strengths of both technologies. LaTeX, which is required as a separate (free) installation, integrates seamlessly with both the GUI or API, enhances the power and flexibility of Python's computational capabilities with the typesetting quality of LaTeX.

Software architecture

SDP relies on three levels of abstraction, with classes developed for each level, which are combined in a hierarchical structure. The classes at each level—*Project* (top level), *Member* (middle level), and *Load* (bottom level)—can be conceptualized as forms or recipes which are used to collect and organize information for each object.

The top level of this architecture consists of the *Project* class, which is used to combine multiple structural components into a single structural design package. The *Project* class includes attributes for the project including the project name, data, and the engineer of record. Python methods defined for the *Project* class allow for the compilation of a combined pdf design package which includes all structural components assigned to an instance of the *Project* class.

The middle level—just below the *Project* class—is the *Member* class which is used to describe the structural components which can be analyzed. Child classes, which inherit common attributes and functionality defined for the *Member* class, are defined for each type of structural component of each design standard. Currently only the ACI318-14 design standard (concrete) is implemented, with seven types of structural components: footing, rectangular beam, one-way slab, two-way slab, structural wall, headed anchor, and plain (unreinforced) concrete. Generic, hand-drawn, component diagrams, as shown in Figure 3, are provided for each *Member* class to illustrate the sign convention and orientation of applied forces, specify the units used, and define all parameters used to define the geometry, dimensions, and materials which comprise the structural component.

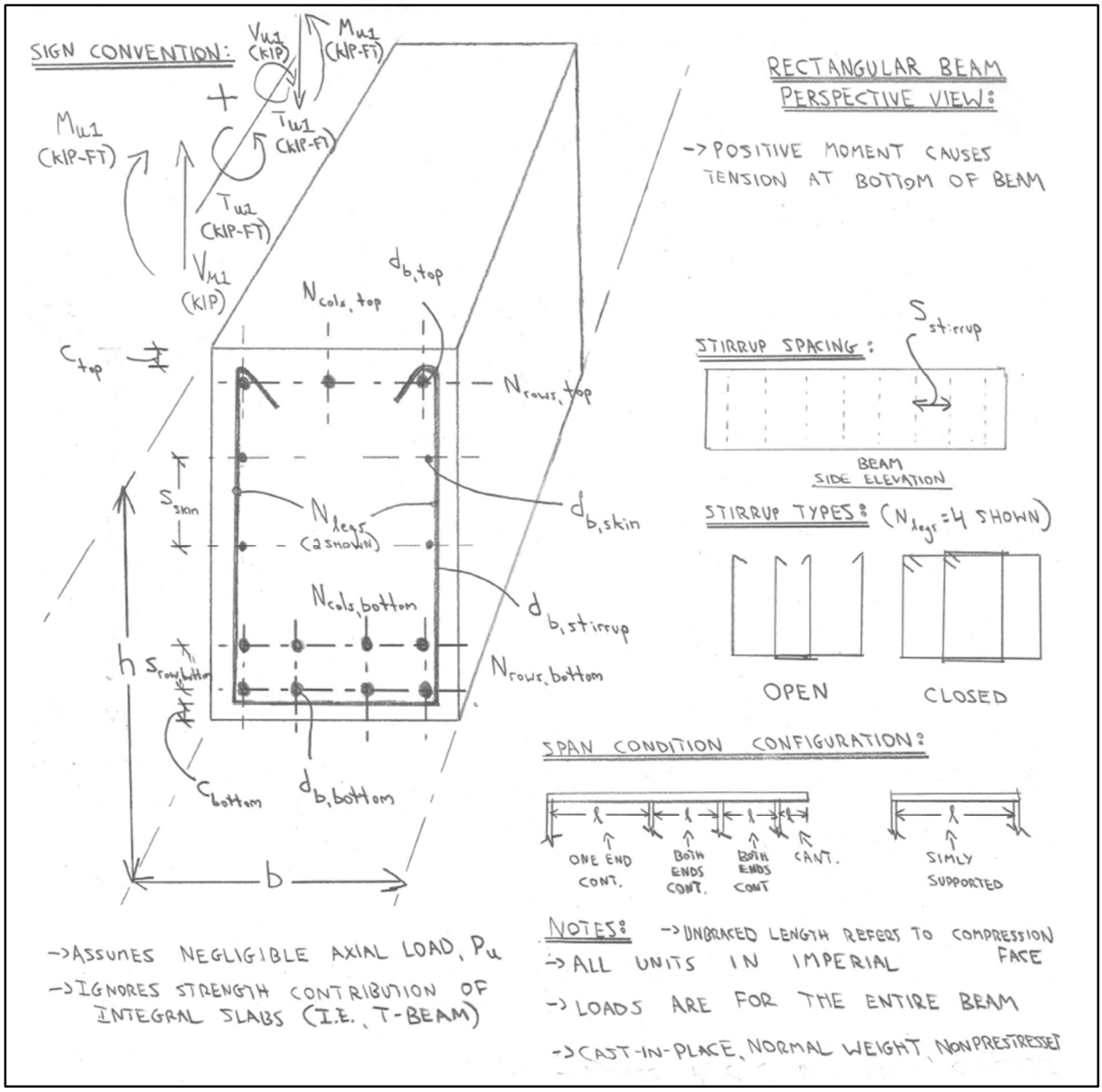

Figure 3: Example hand-drawn component diagram for a rectangular beam Member class showing the sign convention for applied loads, units, and the definition of various properties which are used to define the member geometry and materials.

Each *Member* class can be assigned multiple load cases, which are defined using the *Load* class which exists are the bottom level of the three-level hierarchy. The *Load* class is generic, and allows for the definition of factored axial force, bending moment, shear force, and torsion about two axes. The units of each load case are not explicitly defined and are determined based on the *Member* class to which the load is assigned. For American codes (e.g., ACI318-14) imperial units—kip and ft—are used; for international codes (which have not been implemented yet), metric units—kN and m—will be used. Similar to the treatment of units, the orientation and sign convention for the applied loads is determined based on the *Member* to which the load is assigned and is communicated to the user via a sign convention diagram. A single instance of the *Load* class may

be assigned to multiple instances of *Member* classes, including different types of structural components, with the units and orientation of the applied load dictated by each *Member* class.

The program requires LaTeX to be installed on the user's machine; if the user attempts to compile a pdf without a LaTeX installation, SDP will prompt the user to install MiKTeX, which is a preferred version of LaTeX. If Adobe Reader is installed on the user's machine, it will be used to enhance the rendering of pdfs within SDP; if Adobe Reader is not installed, SDP relies on a relatively rudimentary pdf viewer which is built into the Python and has limited display options.

Loads and *Members* can be easily renamed, edited, duplicated, and reorganized to suite the design narrative. All information which is input into the GUI can be saved as a *.sdp* file which can be loaded at a future date, moved between machines, maintained for future revisions, or archived for record keeping. Pdf design reports are compiled from within the GUI via drop-down menu options or from within the API via a method of the *Project* class.

Dam component design checks

In the compiled pdf design report, the program provides a summary of the dam component properties specified by the user, as illustrated in Figure 4, and completes two types of design checks: (i) capacity checks and (ii) code clause checks. Some capacity checks are repeated as code clauses checks, e.g., code clauses which require strength minimums be achieved for certain structural configurations.

Standard Beam (continued...)

Parameter	Value
Overall depth, h	24.0 inches
Overall width, b	12.0 inches
Unbraced length, l_u	20.0 inches
Clear span length, l	20.0 inches
Support condition ("simply sup.", "one end cont.", "both ends cont.", or "cant.")	simply sup.
Concrete strength, f_c'	4000.0 psi
Reinforcement strength, f_y	60000.0 psi
Top concrete cover, c_{top}	2.0 inches
Top exposure condition ("ground", "exterior", or "interior")	interior
Bottom concrete cover, c_{bottom}	2.0 inches
Bottom exposure condition ("ground", "exterior", or "interior")	interior
Top bar size, $d_{b,top}$	#5
Number of top rows, $N_{rows,top}$	2.0 row(s)
Bars per top row, $N_{cols,top}$	3.0 bars
Top row spacing, c-c, $s_{row,top}$	6.0 inches
Bottom bar size, $d_{b,bottom}$	#5
Number of bottom rows, $N_{rows,bottom}$	2.0 row(s)
Bars per bottom row, $N_{cols,bottom}$	3.0 bars
Bottom row spacing, c-c, $s_{row,bottom}$	6.0 inches
Stirrup bar size, $d_{b,stirrup}$	#3
Stirrup spacing, $s_{stirrup}$	12.0 inches
Num stirrup legs, N_{legs}	2.0 bars
Stirrup Type ("open", "closed")	closed
Skin bar size, $d_{b,skin}$	#4
Skin reinforcement spacing, s_{skin}	18.0 inches

Figure 4: Summary of structural properties input by the user for a rectangular beam in a powerhouse.

The first type of check, the capacity check, compares the structural capacity relative to the applied load cases, and is shown in Figure 5. A summary of the resistance factors, nominal capacities, and factored capacities for the structural component is presented under the heading of "Computed component capacities"; the applied load cases for the structural component are compared against these capacities under the heading of "Factored load description". For each load case, the demand-to-capacity ratio is computed—analogous, but not strictly equivalent, to a factor of safety (FOS)—and this ratio is presented for each load, with ratios above one shown in blue and ratios below one shown in red.

Standard Beam (continued...)

Computed component capacities	**Value**
Flexural resistance factor, ϕ (Sagging)	0.9
Nominal moment resistance, M_n (Sagging)	155.34 kip-ft
Factored moment resistance, ϕM_n (Sagging)	139.81 kip-ft
Flexural resistance factor, ϕ (Hogging)	0.9
Nominal moment resistance, M_n (Hogging)	155.34 kip-ft
Factored moment resistance, ϕM_n (Hogging)	139.81 kip-ft
Shear resistance factor, ϕ	0.75
Concrete shear resistance, V_c	27.8 kip
Steel shear resistance, V_s	20.14 kip
Nominal shear resistance, V_n	47.94 kip
Factored shear resistance, ϕV_n	35.96 kip
Torsion resistance factor, ϕ	0.75
Nominal torsion resistance, T_n	13.72 kip-ft
Factored torsion resistance, ϕT_n	10.29 kip-ft
Factored load description	**Demands (FOS)**
Example Load Case A	-
Example Load Case B	$V_{u1} = 50.0$ kip (0.72); $M_{u1} = 100.0$ kip-ft (1.4)

Figure 5: Example structural capacity checks for a rectangular beam in a powerhouse and computation of capacity-to-demand ratios, with ratios greater than one shown in blue and ratios less than one shown in red.

A second type of check is also performed which evaluates a selection of pertinent code clauses applicable to each structural component to determine whether the requirements of each clause are satisfied (blue) or not satisfied (red), as is shown in Figure 6. Not every code clause applicable to each type of structural member is checked. Some code clauses relate to conditions involving the broader structural configurations and require knowledge of storey heights, bar splice locations, etc. To limit the scope and complexity of SDP, a decision of which factors to consider and which to ignore was necessary. In general, only factors related to the immediate environment of the structural component are considered and code clauses which are not listed in the design report from SDP are not checked.

Standard Beam (continued...)

Clause 9.2.3.1: If a beam is not continuously laterally braced, (a) and (b) shall be satisfied: (a) Spacing of lateral bracing shall not exceed 50 times the least width of compression flange or face; and (b) Spacing of lateral bracing shall take into account effects of eccentric loads.
Satisfied
Clause 9.3.1.1: For solid nonprestressed beams not supporting or attached to partitions or other construction likely to be damaged by large deflections, overall beam depth h shall satisfy the limits in Table 9.3.1.1, unless the calculated deflection limits of 9.3.2 are satisfied.
Satisfied
Clause 9.5.1.1: For each applicable factored load combination, design strength at all sections shall satisfy $\phi S_n > U$ including (a) through (d). Interaction between load effects shall be considered: (a) $\phi M_n > M_u$; (b) $\phi V_n > V_u$; (c) $\phi T_n > T_u$; (d) Not applicable because assumed negligable axial loads.
Satisfied
Clause 9.5.3.1: V_n shall be calculated in accordance with 22.5 (22.5.1.2 dimensions shall be satisfied.)
Satisfied
Clause 9.6.1.1: A minimum area of flexural reinforcement, $A_{s,min}$, shall be provided at every section where tension reinforcement is required by analysis.
Satisfied
Clause 9.6.3.1: A minimum area of shear reinforcement, $A_{v,min}$, shall be provided in all regions where $V_u > 0.5\phi V_c$ except for all the cases in Table 9.6.3.1. For these cases, at least $A_{v,min}$ shall be provided where $V_u > \phi V_c$.
Satisfied
Clause 9.6.3.3: If shear reinforcement is required and torsional effects can be neglected according to 9.5.4.1, $A_{v,min}$, shall be in accordance with Table 9.6.3.3.
Satisfied
Clause 9.6.4.1: A minimum area of torsional reinforcement shall be provided in all regions where $T_u > \phi T_{th}$ in accordance with 22.7.
Satisfied
Clause 9.6.4.2: If torsional reinforcement is required, minimum transverse reinforcement $(A_v+2A_{t,min})/s$ shall be the greater of (a) and (b): (a) $0.75{f_c}'^{0.5}(b_w/f_{yt})$; and (b) $50(b_w/f_{yt})$
Satisfied
Clause 9.6.4.3: If torsional reinforcement is required, minimum area of longitudinal reinforcement, $A_{l,min}$, shall be the lesser of (a) and (b): (a) $5{f_c}'^{0.5}A_{cp}/f_y - (25b_w/f_{yt})p_h(f_{yt}/f_y)$; and (b) $5{f_c}'^{0.5}A_{cp}/f_y - (A_t/s)p_h(f_{yt}/f_y)$
Satisfied
Clause 9.7.1.1: Concrete cover for reinforcement shall be in acordance with 20.6.1. (check of top cover)
Satisfied
Clause 9.7.1.1: Concrete cover for reinforcement shall be in acordance with 20.6.1. (check of bottom cover)
Satisfied
Clause 9.7.2.1: Minimum spacing s shall be in accordance with 25.2
Satisfied
Clause 9.7.2.2: For nonprestressed and Class C prestressed beams, spacing of bonded longitudinal reinforcment closest to the tension face shall not exceed s given in 24.3.
Satisfied
Clause 9.7.2.3: For nonprestressed and Class C prestressed beams with h exceeding 36 in., longitudinal skin reinforcement shall be uniformly distributed on both sides of the beam for a distance $h/2$ from the tension face. Spacing of skin reinforcement shall not exceed s given in 24.3.2, where c_c is the clear cover from the sking reinforcement to the side face. It shall be permitted to include skin reinforcement in strength calculations if a strain compatibility analysis is made.
Satisfied
Clause 9.7.5.1: If torsional reinforcement is required, longitudinal torsional reinforcement shall be distributed around the perimeter of closed stirrups that satisfy 25.7.1.6 or hoops with a spacing not greater than 12 in. the longitudinal reinforcement shall be inside the stirrup or hoop, and at least one longitudinal bar or tendon shall be placed in each corner.
Satisfied
Clause 9.7.5.2: Longitudinal torsional reinforcement shall have a diameter at least 0.042 times the transverse reinforcement spacing, but not less than 3/8 in.
Satisfied
Clause 9.7.6.2.1: If required, shear reinforcement shall be provided using stirrups, hoops, or longitudinal bent bars.
Satisfied
Clause 9.7.6.2.2: Maximum spacing of shear reinforcement shall be in accordance with Table 9.7.6.2.2.
Not satisfied
Clause 9.7.6.3.1: If required, transverse torsional reinforcement shall be closed stirrups satisfying 25.7.1.6 or hoops.
Satisfied
Clause 9.7.6.3.3: Spacing of transverse torsional reinforcement shall not exceed the lesser of $P_h/8$ and 12 in.
Satisfied
Clause 9.9.1.1: Deep beams are members that are loaded on one face and supported on the opposite face such that strut-like compression elements can develop between the loads and supports and that satisfy (a) or (b): (a) Clear span does not exceed four times the overall member depth h; and (b) concentrated loads exist within a distance $2h$ from the face of the support.
Satisfied

Figure 6: Example code clause checks for a rectangular beam in a powerhouse indicating whether each clause is satisfied (blue) or not satisfied (red).

Limitations and planned improvements

The current release of SDP (version 0.1) has several import limitations. Some of these limitations are a result of the software architecture and will persist in subsequent releases. Other limitations are a result of limited resources (time) and will be addressed in future releases.

In the current release, only seven types of concrete components (*Member* classes) (footing, rectangular beam, one-way slab, two-way slab, structural wall, headed anchor, and plain concrete) for a single design standard (ACI318-14) are implemented. Future improvements will expand the library of components to include additional types of concrete components, such as columns, and to provide options to complete the design checks for alternate design standards including ACI318-19, ACI318-24, and CSA A23.3. Furthermore, structural members comprised of other materials, such steel and timber, will also be added.

For each component type, e.g., rectangular beam or one-way slab, the range and variety of structural forms considered by the program has been curtailed to limit the number of configurations

the software must analyze. For example, less common structural forms, such as hollow core slabs, composite beam-slab sections, and non-rectangular beams, are not considered. Such components, which are less common in practice, are unlikely to be implemented in future releases. For these less common structural forms, approximate checks can be completed through simplified geometric idealizations of components.

While as many clause checks have been included for each component as was practicable, the software does not attempt to complete an exhaustive check of every applicable code clause for each type of structural component. Code checks which require knowledge of the broader structural configuration within the dam, spillway, or powerhouse, such as checks of bar cutoffs and bar splices along the lengths of beams and heights of walls, are not included. Code checks focus on sectional analysis but may also include component level checks. Only code clauses listed in the compiled design document are completed.

Conclusions

A software program, called Structural Design Package, has been created which addresses several challenges faced by structural designers in the hydropower industry. Detailed structural design packages are required to ensure that structural components of concrete dams satisfy applicable design guidelines, and these design packages are important for the design and analysis of appurtenant dam structures because they are the primary record of design and capacity checks by the engineer of record. However, these design checks are tedious and time consuming to complete, especially for unique one-off structural elements which are common for dams.

Commercially available software programs are available for structural design checks, and partially address these problems; however, these programs suffer from three main shortcomings: (i) lack of an application programing interface limits the ability to rapid prototype design alternatives, (ii) awkward compilation of multiple structural components into well-organized project design documents, and (iii) trial student licences have limited durations and full licenses are not freely available.

I have developed an educational software program for doing structural design and analysis of dam components which can be used via a graphical user interface or applied as an application programming interface. The stand-alone graphical user interface requires no programming knowledge and can be run from a downloaded executable file. The application programming interface can be accessed as a Python module and provides additional power and flexibility for engineers knowledgeable in Python to interact with the software directly through their own Python scripts. The software is provided for educational use only, with no warranty of any kind, with the hope that it will motivate companies, organizations, and stewards of the hydropower industry to develop similar forward-looking technologies which empower the industry.

References

American Concrete Institute. (2014). ACI 318-14: Building Code Requirements for Structural Concrete and Commentary. Farmington Hills, MI: American Concrete Institute.

Canadian Standards Association. (2014). CSA A23.3-14: Design of Concrete Structures. Mississauga, ON: CSA Group.

Van Rossum, G., & Drake, F. L. Jr. (2009). Python 3 Reference Manual. Scotts Valley, CA: CreateSpace.

Lamport, L. (1994). LaTeX: A Document Preparation System. 2nd edition. Reading, MA: Addison-Wesley.

Schenk, C. (2023). MiKTeX. [Online]. Available: https://miktex.org/

Rappin, N., & Dunn, R. (2006). wxPython in Action. Greenwich, CT: Manning Publications.

Hartmann, G., Kuhlmann, H., & Schougaard, S. V. (2015). PyInstaller Manual. [Online]. Available: https://www.pyinstaller.org