ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

NILSON VIANA GOLIM

DESENVOLVIMENTO DE PROGRAMA EM PYTHON PARA O AUXÍLIO DO DIMENSIONAMENTO E DETALHAMENTO DE PAREDES DE CONCRETO ARMADO BIAPOIADAS EM RESERVATÓRIOS, INTEGRADO AO *SOFTWARE* TQS

São Paulo

2024

NILSON VIANA GOLIM

Desenvolvimento de programa em Python para o auxílio do dimensionamento e detalhamento de paredes de concreto armado biapoiadas em reservatórios, integrado ao software tqs

Versão Corrigida

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo, como requisito parcial para a obtenção do título de Especialista – Gestão de Projetos de Sistemas Estruturais - Edificações

Orientador: Prof. Me. Januário Pellegrino Neto

São Paulo

2024

Catalogação-na-publicação

Golim, Nilson

Programa em linguagem Python para dimensionamento e detalhamento de paredes de concreto armado para reservatórios com aplicação no programa TQS / N. Golim -- São Paulo, 2024.

p.

Monografia (Especialização em Gestão de Projetos de Sistemas Estruturais Edificações) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Estruturas e Geotécnica.

1.Paredes de concreto armado para reservatórios I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Estruturas e Geotécnica II.t.

Nome: GOLIM, Nilson Viana

Título: Desenvolvimento de programa em Python para o auxílio do dimensionamento e detalhamento de paredes de concreto armado biapoiadas em reservatórios, integrado ao software tqs

Monografia apresentada à Escola Politécnica da Universidade de São Paulo, como requisito parcial para a obtenção do título de Especialista em Gestão de Projetos de Sistemas Estruturais – Edificações.

Aprovado em:

Banda Examinadora

Prof. _____

Instituição: _____

Julgamento: _____

Prof. _____

Instituição: _____

Julgamento: _____

Prof. _____

Instituição: _____

Julgamento: _____

AGRADECIMENTOS

Agradeço primeiramente a Deus e à minha família, que sempre foram a base de todo o suporte e apoio ao longo da minha vida.

Ao engenheiro Ítalo Leone, pelo incentivo constante aos meus estudos e por proporcionar a oportunidade de realizar esta especialização. Sua confiança e as oportunidades concedidas foram essenciais para o meu crescimento profissional.

Aos amigos e colegas, que de forma direta ou indireta, fizeram parte desta formação. A todos, o meu muito obrigado.

RESUMO

O cálculo e dimensionamento de paredes de reservatórios é uma etapa essencial em projetos de concreto armado, atualmente com métodos bem estabelecidos para garantir segurança e eficiência estrutural. Embora atualmente exista uma teoria solida sobre como dimensionar e detalhar esses elementos, a prática cotidiana busca constantemente por ferramentas que otimizem o tempo e mantenham a precisão dos resultados. No contexto do *software* TQS, onde há a opção de dimensionar e detalhar automaticamente reservatórios e piscinas, este trabalho propõe uma abordagem alternativa, com o desenvolvimento de um programa complementar em linguagem Python. Este programa será integrado ao TQS com a api python e terá como objetivo realizar o dimensionamento das paredes utilizando o método manual mais comumente empregado, gerando um desenho dwg TQS de maneira automática apenas preenchendo os dados do elemento. O principal benefício dessa ferramenta será a otimização do tempo de trabalho, ao mesmo tempo em que oferece maior segurança dos cálculos e detalhamento. Além do desenvolvimento do programa, este trabalho inclui uma análise comparativa entre os resultados gerados pela metodologia manual proposta e os obtidos pelo detalhamento automático do TQS. Essa comparação destacará as diferenças, vantagens e limitações, contribuindo para o entendimento das possibilidades e limitações da automação no dimensionamento de estruturas de concreto armado. Dessa forma, o trabalho não apenas propõe uma solução prática para otimizar processos, mas também fomenta discussões relevantes sobre a aplicação de metodologias tradicionais em um contexto tecnológico, promovendo avanços na área de engenharia estrutural.

Palavras-chave: Parede – Reservatório - TQS - Python

ABSTRACT

The calculation and design of reservoir walls are essential steps in reinforced concrete projects, currently relying on well-established methods to ensure structural safety and efficiency. Although there is a solid theoretical framework for designing and detailing these elements, everyday practice continuously seeks tools to optimize time while maintaining precision in results. In the context of the TQS *software*, which offers the option to automatically design and detail reservoirs and pools, this work proposes an alternative approach by developing a complementary program in Python. This program will be integrated into TQS using its Python API and aims to perform the wall design using the most employed manual method, generating a TQS DWG drawing automatically by simply filling in the element data. The main benefit of this tool will be optimizing work time while providing greater reliability in calculations and detailing. In addition to the program's development, this work includes a comparative analysis between the results generated by the proposed manual methodology and those obtained through TQS's automatic detailing. This comparison will highlight the differences, advantages, and limitations, contributing to a deeper understanding of the possibilities and constraints of automation in the design of reinforced concrete structures. Thus, this work not only proposes a practical solution for process optimization but also fosters relevant discussions on the application of traditional methodologies in a technological context, promoting advancements in structural engineering.

Keywords: Wall – Reservoir - TQS - Python

SUMÁRIO

1 INTRODUÇÃO

1.1 Contexto

Assim como ocorre em outros setores, o mercado de projetos estruturais de concreto armado sempre demandou melhorias na produtividade. O ganho de eficiência é essencial para atender aos prazos estipulados e às necessidades das obras, tornando-se um fator determinante no sucesso de qualquer empreendimento.

Com a crescente necessidade de desenvolver projetos de forma cada vez mais ágil e segura, surgiu a ideia de criar ferramentas específicas que facilitassem e otimizassem esse processo. Foi nesse contexto que nasceu o *software* Tecnologia e Qualidade de Sistemas, mais conhecido como TQS. Fundada em 1986 por engenheiros civis, a TQS é uma empresa brasileira que se destacou pelo desenvolvimento de *softwares* voltados à elaboração de projetos estruturais de edificações, com foco principal no concreto armado. Ao longo dos anos, o TQS tornou-se referência nacional nesse segmento.

Segundo Nelson Covas, sócio fundador da empresa: “Disponibilizar *softwares* inovadores, abrangentes e robustos, de forma profissional e transparente, alinhada com um suporte técnico competente, a fim de auxiliar os seus clientes na complexa e competitiva arte de elaborar projetos estruturais. Esse é o foco da TQS.”

O *software* TQS realiza o cálculo estrutural e o detalhamento de diversos elementos estruturais, como pilares, lajes, vigas e outros elementos especiais, como blocos de fundação, armadura de punção em lajes e reservatórios, consolidando-se como uma solução abrangente para projetos de concreto armado.

Apesar de sua robustez, o TQS possui algumas limitações, especialmente no que diz respeito à personalização e padronização de detalhamentos conforme as preferências individuais de cada engenheiro. Essa flexibilidade, quando combinada a outros métodos de dimensionamento, permite que o profissional adote abordagens personalizadas, ampliando o escopo do *software* e garantindo maior alinhamento com as demandas específicas de cada projeto.

1.2 Justificativa

As paredes de reservatórios são elementos recorrentes em projetos de edificações em concreto armado, uma vez que os edifícios demandam reservatórios para diferentes usos.

Contudo, esses elementos apresentam características distintas em relação aos componentes estruturais mais comuns, como vigas, lajes e pilares, exigindo atenção especial em sua concepção e detalhamento.

O autor deste trabalho atua em um contexto em que a rapidez na entrega dos projetos, aliada à padronização e à qualidade, é uma exigência constante. Essa realidade motivou a concepção de um programa de computador destinado a auxiliar no dimensionamento e detalhamento de paredes de concreto para reservatórios, atendendo às demandas do mercado com maior eficiência.

A implementação de uma solução computacional para apoiar o desenvolvimento de projetos estruturais é extremamente vantajosa. Além de proporcionar agilidade ao processo, ela garante maior segurança tanto no dimensionamento quanto no detalhamento, minimizando a ocorrência de erros e assegurando um padrão de qualidade nos resultados.

1.3 Objetivo

1.3.1 Objetivo Geral

O objetivo geral é compreender como é desenvolvido o projeto estrutural de uma parede de reservatório em concreto armado e desenvolver um programa de computador para auxiliar na elaboração deste projeto.

1.3.2 Objetivos específicos

Para atingir o objetivo final, foram estabelecidos objetivos específicos:

I. Compreender como é realizado o dimensionamento e detalhamento de uma parede de concreto armado.
II. Compreender como utilizar a API disponibilizada pela TQS para o desenvolvimento do programa auxiliar.
III. Realizar o desenvolvimento do programa de computador buscando os objetivos para ganho de produtividade e evitar erros.
IV. Comparar os resultados obtidos pelo programa auxiliar com os resultados obtidos pelo programa de reservatórios disponível no TQS.

1.4 Metodologia

Para a construção de um embasamento teórico, a primeira etapa realizada neste trabalho, é a revisão bibliográfica.

Nela, foram reunidos os conceitos que fundamentam este trabalho, de maneira simples e resumida, por meio da coleta de dados com a leitura de livros, obras, artigos científicos e normas.

A segunda etapa constitui em desenvolver o programa computacional, com a demonstração de cada etapa desenvolvida, que foi realizada com o auxílio de livros sobre a linguagem python e o manual da API do TQS.

Na terceira etapa foi mostrado um exemplo prático do programa auxiliar e a comparação dos resultados obtidos com o TQS.

1.5 Estrutura do trabalho

A estrutura deste trabalho está organizada da seguinte forma:

Capítulo 1

Esse capítulo contém a descrição geral do tema, seus objetivos, a justificativa para seu desenvolvimento, a metodologia utilizada e a sua estrutura.

Capítulo 2

Neste capítulo contém a revisão bibliográfica, com a demonstração dos conceitos para análise, dimensionamento e detalhamento de paredes de concreto armado para reservatórios.

Capítulo 3

Esse capítulo tem a demonstração das etapas de criação do programa auxiliar, com o desenvolvimento do código de cada etapa.

Capítulo 4

Neste, tem a demonstração de um caso prático exemplificado para demonstrar o uso do programa e validação dos resultados.

Capítulo 5

No capítulo 5 tem a comparação dos resultados do programa com os resultados obtidos pelo TQS.

Capítulo 6

Esse capítulo consta uma conclusão da comparação dos resultados obtidos e a eficiência do programa computacional desenvolvido.

Apêndice A

Contém um manual de instruções do programa computacional desenvolvido para dimensionamento de parede de reservatório biapoiada.

Apêndice B

Contém um manual de instruções do programa computacional desenvolvido para o detalhamento de parede de reservatório.

Anexo A

Contém o código do programa computacional desenvolvido para dimensionamento de parede de reservatório biapoiada.

Anexo B

Contém o código do programa computacional desenvolvido para o detalhamento de parede de reservatório.

2 REVISÃO BIBLIOGRAFICA

Para o dimensionamento de uma parede de concreto armado, na qual apresenta esforços de flexão em conjunto com esforços laterais de empuxo, as bibliografias recomendam uma soma de dimensionamentos, somando o dimensionamento para flexão de uma viga-parede e o dimensionamento de uma placa de concreto como o empregado em lajes de concreto armado.

Será apresentado um exemplo de um método de cálculo baseado em fontes reconhecidas para seguir um passo a passo no exemplo apresentado posteriormente neste trabalho.

2.1 Conceituação

A NBR 6118:2023 menciona algumas recomendações para o dimensionamento para uma viga-parede.

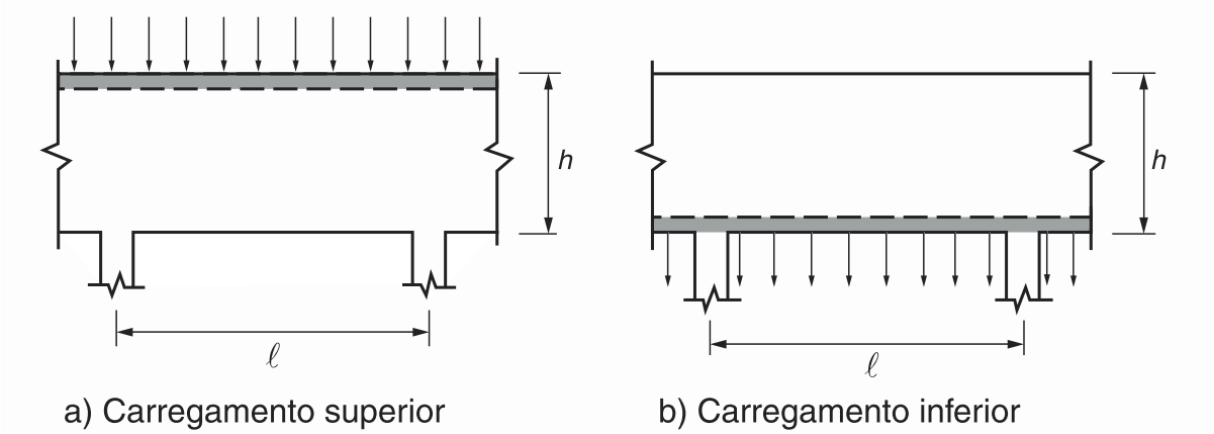
De acordo com a NBR 6118:2023 são consideradas vigas-parede as vigas altas em que a relação entre o vão e a altura seja:

$$\frac{l}{h} < 2 \text{ para vigas bi} - \text{apoiadas}$$

$$\frac{l}{h} < 3 \text{ para vigas contínuas}$$

As vigas-parede podem receber carregamentos superior ou inferior:

Figura 2.1 – Dois tipos comuns de vigas-parede em relação ao carregamento



Fonte: ABNT NBR 6118:2023.

O comportamento das vigas-parede apresenta características especificas que as diferenciam das vigas comuns. Entre essas particularidades, destaca-se sua menor eficiência tanto à flexão quanto ao cisalhamento. Devido à sua altura, essas vigas frequentemente enfrentam problemas de instabilidade, seja como corpo rígido ou em termos de estabilidade elástica. Para mitigar essas questões, é comum a necessidade de enrijecedores de apoio ou travamentos adicionais. Além disso, deve-se considerar os efeitos das perturbações causadas por cargas concentradas, aberturas ou variações de espessura. Tais fatores podem impactar de forma significativa o comportamento estrutural e a capacidade resistente das vigas-parede, exigindo atenção especial durante o dimensionamento e a análise desse tipo de elemento.

A NBR 6118 também permite modelos planos elásticos lineares e não lineares, como o método dos elementos finitos. Permite também modelos das bielas e tirantes.

2.2 Armadura de flexão

Para os tirantes de tração, as armaduras não podem ser concentradas em uma ou poucas camadas, mas deve cobrir toda a zona efetivamente tracionada conforme o modelo de cálculo adotado.

Nas vigas-parede bi-apoiadas a armadura deve ser distribuída em altura da ordem de 0,15h.

Nas contínuas, a altura de distribuição da armadura negativa deve ser feita considerando 3 faixas na altura h, não considerando para h os valores superiores ao vão teórico l ($3 \geq l/h \geq 1$):

- 20% superiores de h: $As_1 = (l/2h - 0{,}50) \cdot As$
- 60% centrais de h: $As_2 = (1{,}50 - l/2h) \cdot As$
- 20% inferiores de h: $As_3 = 0$

Para armadura horizontal mínima, é considerado 0,075% *b* por face, por metro.

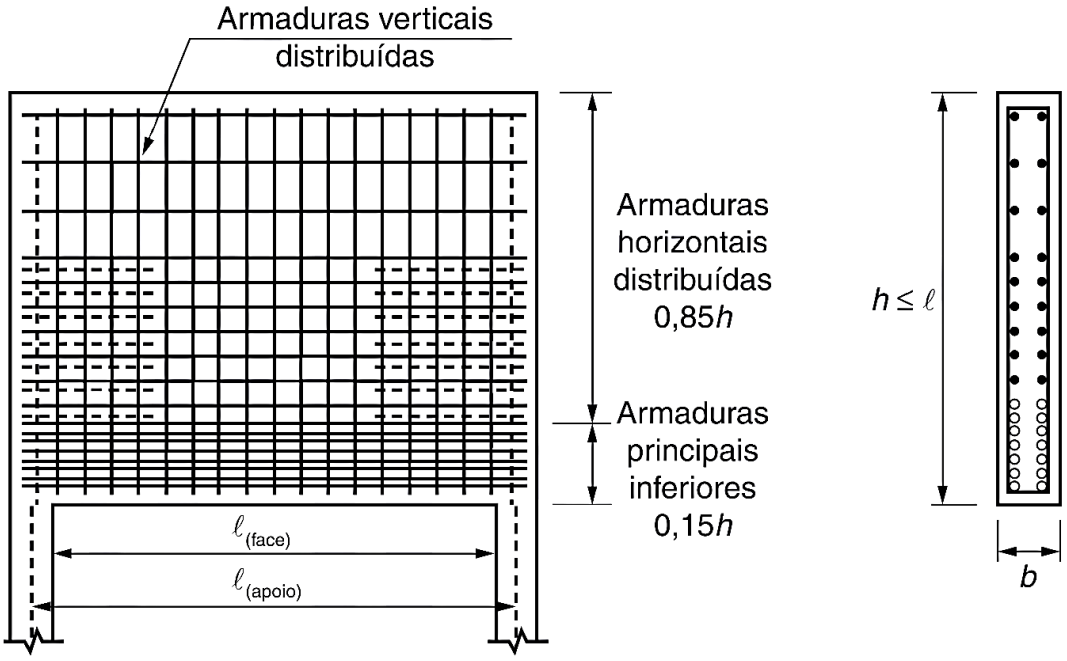
2.3 Armadura vertical

A armadura vertical deve ser calculada respeitando um valor mínimo de 0,075% *b* por face, por metro.

Caso exista carregamento pela parte inferior da viga, essa armadura deverá ser capaz de suspender a totalidade da carga aplicada.

As armaduras devem envolver as armaduras horizontais, principais e secundarias.

Figura 2.2 – Armação típica de viga-parede com h ≤ l



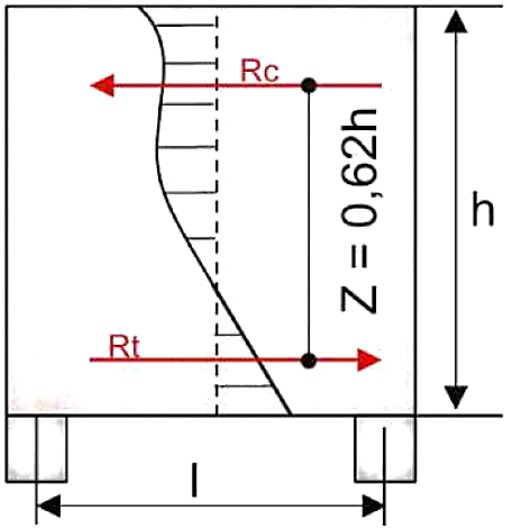
Fonte: ABNT NBR 6118:2023.

2.4 Dimensionamento para viga-parede

Em reservatórios, além da análise estrutural como laje por conta do esforço devido ao empuxo, é necessário verificar os esforços de flexão nas paredes como uma vigas-parede. É comum a utilização da armadura mínima conforme a NBR 6118 por conta da elevada altura dessas vigas, e às baixas cargas verticais que elas suportam. Para os reservatórios enterrados que se apoiam diretamente no solo, essa verificação pode ser dispensada.

A espessura mínima para uma viga-parede é de 15 cm, conforme o item 13.2.2 da NBR 6118. Já o item 22.4.1 define vigas-parede como aquelas em que a relação entre o vão e a altura (l/h) é menor que 2 para vigas biapoiadas e menor que 3 para vigas contínuas, é importante salientar que no cálculo desse tipo de estrutura a hipótese das seções planas não se aplica, devido às significativas distorções que as vigas-parede sofrem. As deformações normais não apresentam comportamento linear ao longo da seção, o que implica que as tensões também não variam de forma linear, diferentemente do que ocorre em vigas comuns.

Figura 2.3 – Dois tipos comuns de vigas-parede em relação ao carregamento



Fonte: PORTELA, 2021.

A figura 2.3 tem a demonstração de tensões normais para uma viga-parede com relação h/l = 1, comum em formato quadrado, mas é comum que para razões h/l = 2, a distribuição seja próxima a de vigas mais comuns, com variação linear normais ao longo da seção.

O cálculo dos esforços solicitantes em uma viga-parede segue os mesmos princípios de uma viga comum, em uma viga simplesmente apoiada com carga distribuída linearmente, o momento solicitante pode ser estimado como $ql^2/8$. Contudo, a distribuição de tensões em uma viga-parede varia conforme o ponto de aplicação da carga. É essencial considerar o local de entrada da carga ao dimensionar as armaduras. A análise estrutural será influenciada dependendo se a carga é aplicada pelo fundo ou pelo topo da viga.

É importante para esse tipo de elemento estrutural verificar o esmagamento do concreto nos apoios, escoamento da armadura no banzo tracionado e a ruptura da armadura de suspensão.

Para a região tracionada:

$$As = \frac{Md}{Z.fyd}$$

Onde:

As = Area de aço necessária.

Md = Momento fletor de cálculo.

Z = Braço de alavanca.

fyd = Tensão de cálculo do aço.

O valor de Z para viga-parede biapoiada é dado por:

$$Z = 0{,}15h\left(3 + \frac{l}{h}\right) \ \ se \ \ 1 < \frac{l}{h} < 2$$

$$Z = 0{,}6l \ \ se \ \ \frac{l}{h} \leq 1$$

Para viga-parede de dois vãos:

$$Z = 0{,}10h\left(2{,}5 + 2\frac{l}{h}\right) \ \ se \ \ 1 < \frac{l}{h} < 2{,}5$$

$$Z = 0{,}45l \ \ se \ \ \frac{l}{h} \leq 1$$

Para mais de dois vãos:

$$Z = 0{,}15h\left(2 + \frac{l}{h}\right) \ \ se \ \ 1 < \frac{l}{h} < 3$$

$$Z = 0{,}45l \ \ se \ \ \frac{l}{h} \leq 1$$

Onde:

Z = Braço de alavanca.

h = Altura da parede

l = Largura da parede

Para a armadura mínima de tração horizontal:

$$As, minVP = \lambda As.minVE$$

Onde:

As,minVP = Armadura mínima das vigas-parede.

As,minVE = Armadura mínima de vigas esbeltas.

λ depende da relação l/h.

Tabela 2.1 – Tabela de λ e relação l/h

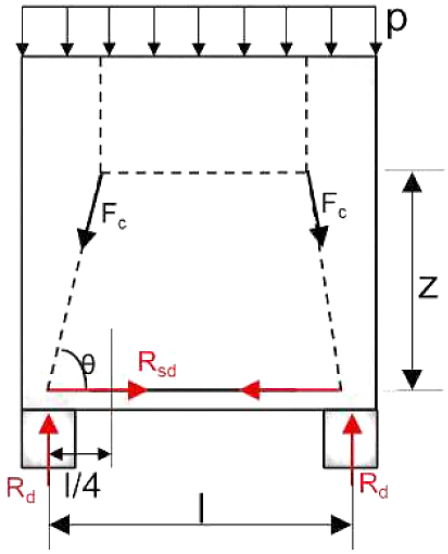
l/h	λ
2,0	1,00
1,5	0,90
1,25	0,75
1,0	0,55

Fonte: TQSDocs, 2022.

A armadura mínima lateral pode ser calculada por 0,10b, em cm^2/m, onde b é a largura da parede.

Para o esmagamento do concreto nos apoios, pode ser utilizado o modelo de cálculo de bielas e tirantes da seguinte forma:

Figura 2.4 – Modelo de bielas e tirantes para viga-parede biapoiada.



Fonte: PORTELA, 2021.

$$Rsd.Z = Rd\frac{l}{4}$$

Onde:

Rsd = Força de cálculo resultante do banzo tracionado, que pode ser dado por Asfyd.

Z = Braço de alavanca.

Rd = Força de cálculo da reação de apoio, que pode ser dado por pl/2.

l = Vão entre eixos dos apoios da viga-parede.

$$tg\theta = 4\frac{Z}{l}$$

$$Fc = \frac{Rd}{sen\theta}$$

Onde:

θ = Ângulo da biela.

Z = Braço de alavanca.

l = Vão entre eixos dos apoios da viga-parede.

Fc = Força da biela comprimida.

Rd = Força de cálculo da reação de apoio, que pode ser dado por pl/2.

$$\sigma_d = \frac{Rd}{bc} \leq fcdr$$

Onde:

σ_d = Tensão no apoio

Rd = Força de cálculo da reação de apoio, que pode ser dado por pl/2.

b = largura da viga-parede

c = largura do apoio

fcdr = Resistencia de compressão reduzida.

$$\sigma_{2d} = \frac{Fc}{bc_2} \leq fcdr$$

$$c_1 = c + ucotg\theta$$

$$u = 2d'$$

$$\sigma_{2d} = \frac{Rd}{b(c + ucotg\theta)sen^2\theta}$$

Onde:

σ_{d2} = Tensão da biela inclinada

Fc = Força da biela comprimida.

b = largura da viga-parede

c = largura do apoio

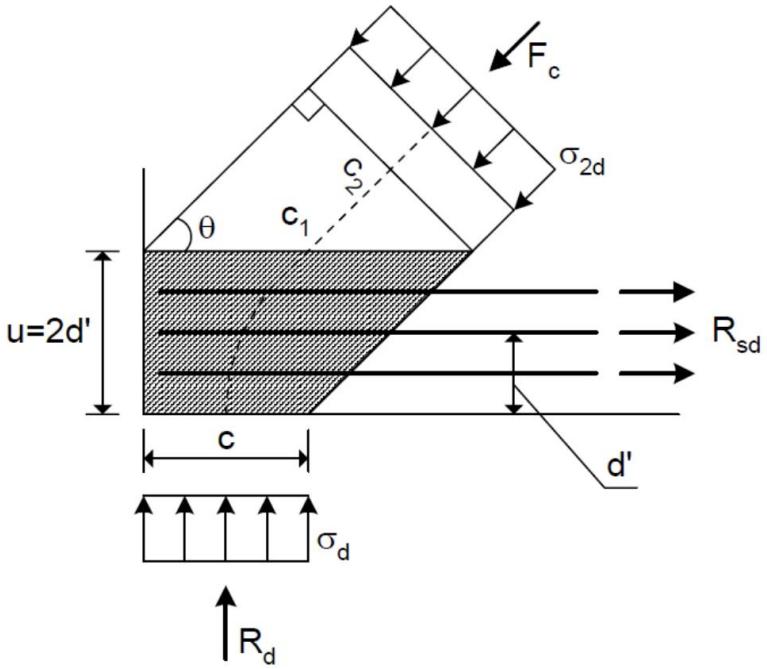
c1 = Projeção da largura da biela de compressão horizontal + largura do apoio.

u = altura do nó de apoio, depende da distribuição das armaduras de tração.

fcdr = Resistencia de compressão reduzida.

Rd = Força de cálculo da reação de apoio, que pode ser dado por pl/2.

Figura 2.5 – Nó no apoio de extremidade.



Fonte: TQSDocs, 2022.

$$fcdr = 0{,}60\alpha_v fcd = 0{,}60\left(1 - \frac{fck}{250}\right)fcd$$

Onde:

fck = Resistencia a compressão do concreto em MPa.

fcd = Resistencia de cálculo a compressão do concreto em MPa.

Realizar as seguintes verificações:

$$\text{se } u \geq c.cotg\theta \textbf{ basta } \sigma d \leq fcdr$$

$$\text{se } u < c.cotg\theta \textbf{ basta } \sigma 2d \leq fcdr$$

Para apoios internos:

$$\sigma_d \leq 0{,}85 fcd$$

Onde:

σ_d = Tensão no apoio

fcd = Resistencia de cálculo a compressão do concreto em MPa.

Quando a carga na viga-parede chega pela face de baixo, é necessário calcular a armadura de suspensão, que pode ser por estribos, para o cálculo da armadura de suspensão:

$$A_{susp} = \frac{p_{fi}}{fyd}$$

Onde:

Asusp = Area de armadura de suspensão em cm^2/m.

p_{fi} = carga distribuída do fundo da viga-parede.

fyd = Resistencia de cálculo do aço.

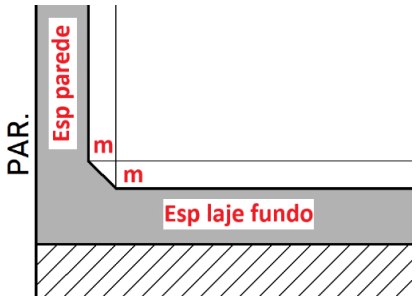
2.5 Dimensões mínimas

Para o dimensionamento da viga-parede é importante fazer as considerações de dimensões mínimas. Estas medidas são encontradas na NBR 6118:2023 no item 13.2.2.

A norma define como largura mínima para uma viga-parede o valor de 15cm, para ajudar na transferência de esforços, para a laje de fundo o ideal é ter sua espessura igual ou superior a esta, sendo assim, no mínimo 15cm, considerando que esta laje de fundo recebe uma carga elevada por conta da carga do reservatório, é comum encontrar projetos de reservatório com espessuras na ordem de 15 a 20cm, para a laje da tampa é possível reduzir a espessura, mais comum entre 10 e 12cm, pois essa laje não recebe a mesma ordem de grandeza da carga aplicada na laje do fundo do reservatório.

É recomendado para reservatórios, principalmente os de uso como de água potável ou reuso, a inclusão de mísulas, em resumo as mísulas são os chanfros entre os encontros de parede e laje de fundo, a fim de facilitar impermeabilização e limpeza, e estruturalmente ajuda a aumentar o engastamento entre os elementos e reduz os riscos de fissuração.

Figura 2.6– Mísula entre parede e laje de fundo.



Fonte: Autor próprio, 2024.

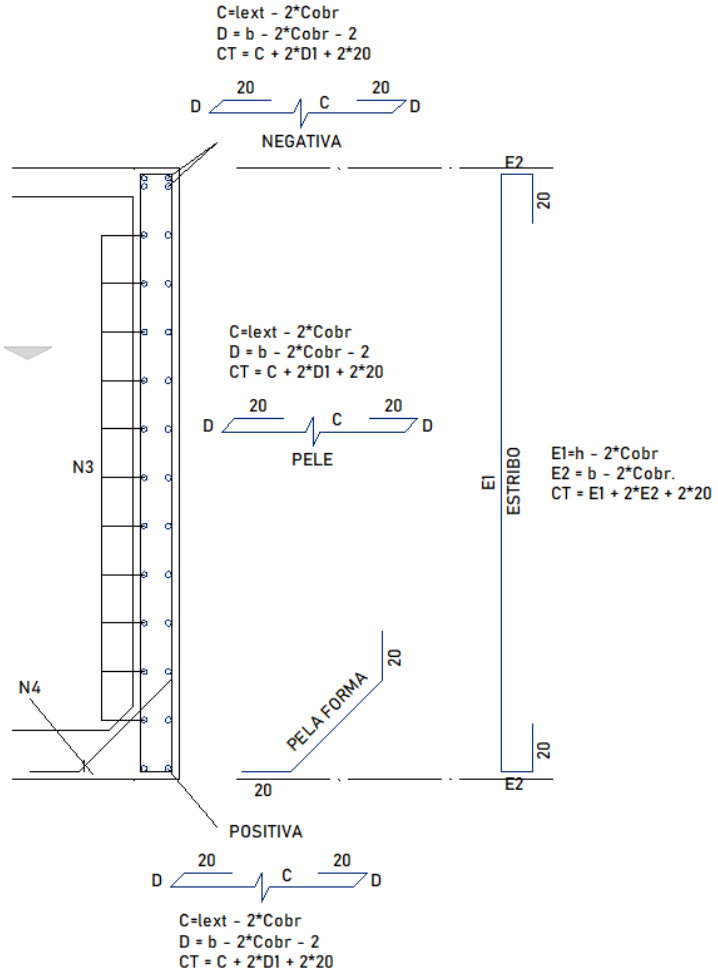
Onde m é a espessura das mísulas.

2.6 Detalhamento

O detalhamento da viga-parede pode ser feito de algumas maneiras, o mais importante é se atentar quanto aos engastamentos (ligações) definidos, para garantir que sejam eficazes.

Sugestão para o detalhamento da viga-parede pelo método de cálculo apresentado:

Figura 2.7 – Sugestão para o detalhamento da viga-parede



Fonte: Autor próprio, 2024.

Onde:

b = Largura da parede.

h = Altura da parede.

D = Dobra das barras horizontais.

C = Comprimento reto das barras.

CT = Comprimento total das barras.

E1 = Comprimento reto do estribo.

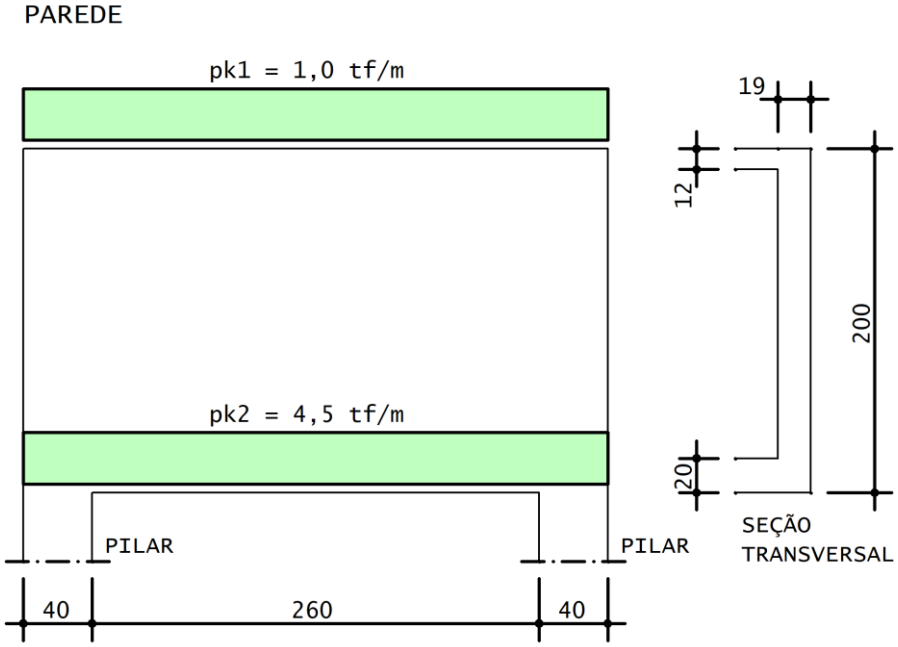
E2 = Dobra do estribo.

O controle de abertura de fissuras deve ser feito de acordo com as recomendações da NBR 6118:2023 demonstrados nos itens 13.4 e 17.3.3.

2.7 Exemplo de dimensionamento de uma parede de reservatório

Dimensionar a parede indicada abaixo:

Figura 2.8 – Geometria e carregamento da parede



Fonte: Autor próprio, 2024.

Peso próprio da parede:

$$pk3 = pp.b.h \rightarrow pk3 = 2{,}5\,.\,0{,}19\,.\,2{,}00 = 0{,}95\ tf/m$$

Onde:

Pp = Peso próprio do concreto em tf/m^3

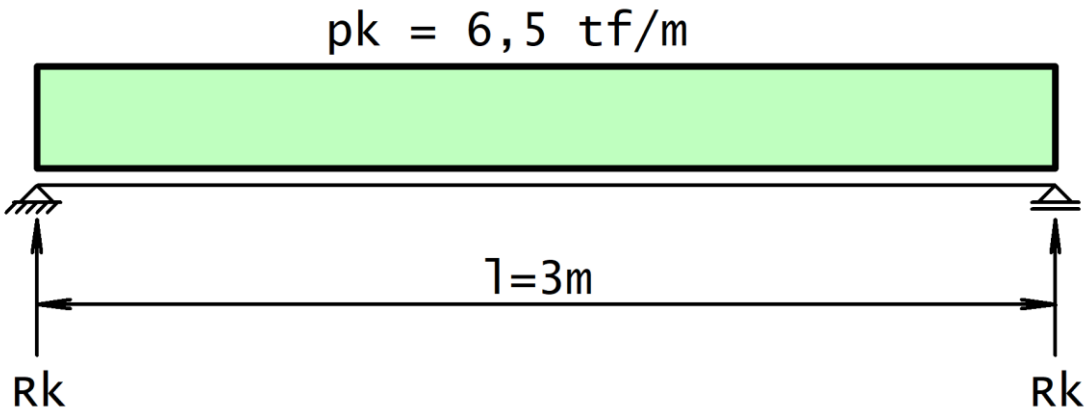
b = Largura da parede

h = Altura da parede

A carga total é:

$$pk = pk1 + pk2 + pk3 = 1{,}0 + 4{,}5 + 0{,}95 \cong 6{,}5\ tf/m$$

Figura 2.9 – Diagrama de corpo livre da parede



Fonte: Autor próprio, 2024.

O momento fletor e a reação de apoio Rk pode ser calculada como o de uma viga biapoiada comum:

$$Mk = \frac{pkl^2}{8} = \frac{6{,}5.3^2}{8} \cong 7{,}5\ tfm$$

$$Rk = \frac{pkl}{2} = \frac{6{,}5.3}{2} \cong 10\ tf$$

A relação entre o vão e a altura da parede é de l/h = 3/2 = 1,5 < 2.

Desta forma a parede se enquadra como uma viga-parede.

Armadura horizontal:

Aço CA-50 → fyk = 500 MPa = 5 tf/m^2 → fyd = 500/1,15 = 434,7 → 4,347 tf/m^2

$$Md = 1{,}4.Mk = 1{,}4.7{,}5 = 10{,}5\ tfm$$

$$Z = 0{,}15h\left(3 + \frac{l}{h}\right) = 0{,}15.2\left(3 + \frac{3}{2}\right) = 1{,}35m$$

$$As = \frac{Md}{Z.fyd} = \frac{10,5}{1,35\,.\,4,347} \cong 1,8\ cm^2$$

Armadura mínima

$$As, minVP = 0,9.19.200.\frac{0,15}{100} = 5,13\ cm^2$$

Adotando Ø 12,5mm para a distribuição da armadura temos:

$$A_{Ø12,5} = \frac{\pi.d^2}{4} = \frac{\pi.1,25^2}{4} \cong 1,23\ cm^2$$

Portanto:

$6x1,23 = 7,38\ cm^2 > 5,13\ cm^2$

Armadura horizontal → Ase = 6 Ø 12,5.

Tensão nos apoios:

Força de cálculo da reação de apoio:

$$Rd = 1,4.Rk = 1,4.10 = 14\ tf$$

A inclinação da biela de compressão:

$$tg\theta = \frac{4Z}{l} = \frac{4.1,35}{3} = 1,8 \rightarrow \theta \cong 61°$$

Considerando cobrimento das armaduras de 3,0cm, tem-se d' = 7,0cm, pois a armadura do banzo tracionado tem 3 camadas. A altura do nó de apoio é u = 2d' = 14,0cm.

$$c.cotg\theta = 40.cotg61° = 22,17\ cm$$

se $u < c.cotg\theta$ **basta** $\sigma 2d \leq fcdr$

$$\sigma_{2d} = \frac{Rd}{b(c + ucotg\theta)sen^2\theta} = \frac{14}{19.\left(40 + 22{,}17.\frac{1}{1{,}8}\right).0{,}874^2} = \frac{14}{759{,}305} = 0{,}018\ \text{tf/cm}^2$$

$$\sigma_{2d} = 0{,}018\frac{\text{tf}}{\text{cm}^2} \rightarrow 0{,}18\ \frac{\text{kN}}{\text{cm}^2}\ \rightarrow 1{,}8\ \text{MPa}$$

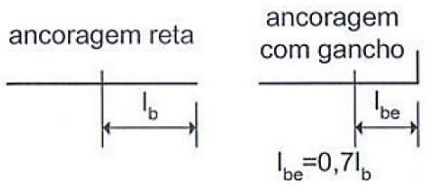
Utilizando um concreto de fck = 25 MPa:

$$fcdr = 0{,}60\left(1 - \frac{25}{250}\right).\frac{25}{1{,}4} \cong 9{,}7\ \text{MPa}$$

$\sigma_{2d} < fcdr\ \rightarrow 1{,}8 < 9{,}7\ \text{MPa}\ \rightarrow$ Segurança contra o esmagamento de biela ok!

Comprimento básico para ancoragem da armadura de flexão:

Tabela 2.2 – Valores para comprimento básico de ancoragem.
Aço CA-50 ; $f_{cd} = f_{ck}/1{,}4$; $f_{yd} = f_{yk}/1{,}15$



BOA ADERÊNCIA

Bitola	$f_{ck} = 20$ MPa		$f_{ck} = 25$ MPa		$f_{ck} = 30$ MPa	
ϕ	l_b	l_{be}	l_b	l_{be}	l_b	l_{be}
6,3	28	19	24	17	21	15
8	35	25	30	21	27	19
10	44	31	38	27	34	23
12,5	55	38	47	33	42	29
16	70	49	61	43	54	38
20	88	62	76	53	67	47
25	110	77	95	66	84	59

MÁ ADERÊNCIA

Bitola	$f_{ck} = 20$ MPa		$f_{ck} = 25$ MPa		$f_{ck} = 30$ MPa	
ϕ	l_b	l_{be}	l_b	l_{be}	l_b	l_{be}
6,3	40	28	34	24	30	21
8	50	35	43	30	38	27
10	63	44	54	38	48	34
12,5	79	55	68	47	60	42
16	100	70	87	61	77	54
20	126	88	108	76	96	67
25	157	110	135	95	120	84

Fonte: ARAUJO, 2023.

O *lb* necessário para fck 25 MPa e Ø 12,5 em região de boa aderência e com gancho é igual a 33cm.

A força a ser ancorada é de 0,8.As.fyd, a armadura calculada nos apoios é de:

$$As, cal = 0{,}8As = 0{,}8\,.\,1{,}8 \rightarrow 1{,}44\ \text{cm}^2$$

Portanto, o comprimento de ancoragem necessário é:

$$lb, nec = 0{,}7.lb.\frac{As, cal}{Ase} = 33.\frac{1{,}44}{7{,}38} \cong 6{,}44\ cm$$

Valores mínimos para comprimento de ancoragem são de → 0,3.33=9,9cm, 10.1,25=12,5cm e 10cm, portanto o lb necessário real é de 12,5 cm, como o comprimento dos pilares de apoio é de 40cm, é possível realizar a ancoragem com gancho.

Armadura de suspensão:

$$Pd = 1{,}4pk2 = 1{,}4\,.\,4{,}5 = 6{,}3\,\frac{\text{tf}}{\text{m}}$$

$$Asusp = \frac{Pd}{fyd} = \frac{6{,}3}{4{,}348} \cong 1{,}45\,\frac{\text{cm}^2}{\text{m}} \rightarrow 0{,}725\ \frac{\text{cm}^2}{\text{m}}\ \text{por face}$$

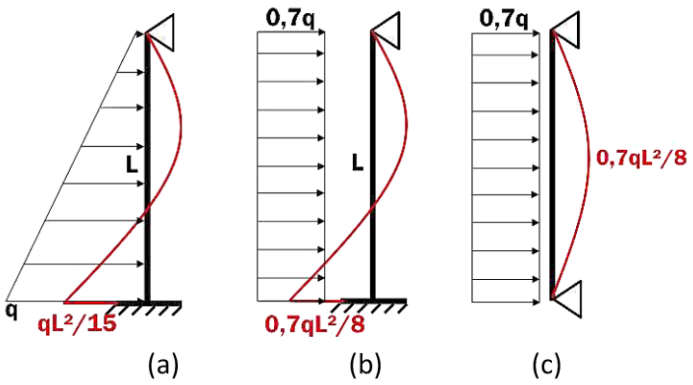
Armadura de pele e estribo:

Armadura mínima:

$$Asw, pmin = \ 0{,}10.b = 0{,}10\,.\,19 = 1{,}9\ \frac{\text{cm}^2}{\text{m}}\ \text{por face}$$

Os esforços resultantes do empuxo da água podem ser determinados utilizando o método de cálculo para lajes maciças, seja por meio de análises por elementos finitos ou com o auxílio de tabelas, como a tabela de Czerny. Alternativamente, é possível simplificar o cálculo adotando o seguinte modelo:

Figura 2.10 – Compatibilização de momentos fletores.



Fonte: PORTELA, 2021.

O método baseia-se em considerar tanto a laje de tampa quanto a laje de fundo como estruturas simplesmente apoiadas. O momento positivo é dimensionado com base nos valores calculados, e a armadura negativa da laje de fundo é definida como equivalente à armadura da parede. A parede é sempre tratada como unidirecional, sendo dimensionada para o momento $ql^2/8$ em ambas as faces. Para esse cálculo, a carga q é aproximadamente 70% do maior empuxo.

Quando a altura da parede do reservatório é significativamente maior que seu comprimento (2:1), recomenda-se que o cálculo dos momentos seja realizado na direção horizontal. Na direção vertical, pode-se adotar apenas a armadura mínima, atendendo às exigências normativas.

De acordo com Portela, Enson (2021, p. 17)

> Esse método alternativo é bastante seguro e produz armaduras trabalhando com baixas tensões de tração, o que é recomendável para reservatórios. Lembre-se que neste tipo de estrutura fissuras mínimas devem ser evitadas por conta da água. Deixar então a armadura trabalhando com baixas tensões é sempre uma boa prática.

Sendo assim:

$$Mpar = 0{,}7\frac{ql^2}{8}$$

Onde:

q = Carga máxima de empuxo horizontal da água.

l = O vão vertical da parede entre os eixos das lajes de fundo e tampa.

$$q = h_i.\gamma_{agua}$$

Onde:

h_i = Altura interna da parede (descontando as lajes).

γ_{agua} = 1 tf/m^3.

Portanto:

$$q = 1{,}68.1 = 1{,}68\ \text{tf/m/m}$$

$$Mpar = \frac{0{,}7.1{,}68.1{,}84^2}{8} = 0{,}5\ \text{tfm/m}$$

Com o esforço do momento na parede podemos calcular qual a armadura necessária.

Para o cálculo da armadura da seção vertical da parede:

b = 100 cm (armação distribuída)
h = 19 cm (largura da parede)
d = 19 – 4 = 15cm
d' = 4cm
fck = 25 MPa
Aço CA50 → fyk = 500 MPa
M = 0,5 tfm/m

x_{lim} = 0,45d
x_{23} = 0,259d
x_{34} = 0,628d
γf = 1,4
γc = 1,4
γs = 1,15

$$fcd = \frac{2{,}5}{1{,}4} = 1{,}79\frac{\text{kN}}{\text{cm}^2}$$

$$fyd = \frac{50}{1{,}15} = 43{,}48\frac{\text{kN}}{\text{cm}^2}$$

$$x_{lim} = 0{,}45.15 = 6{,}75\ \text{cm}$$

$$x_{23} = 0{,}259.15 \cong 3{,}89\ \text{cm}$$

$$x_{34} = 0{,}628.15 \cong 9{,}42\ \text{cm}$$

$$Md = 1{,}4.0{,}5 = 0{,}70\frac{\text{tfm}}{\text{m}} = 700\ \frac{\text{kNcm}}{\text{m}}$$

$$Msd = 0{,}68.b.x.fcd.(d - 0{,}4x)$$

$$x = 1{,}25.d.\left[1 - \sqrt{1 - \frac{Md}{0{,}425.b.d^2.fcd}}\right]$$

$$x = 1{,}25.15.\left[1 - \sqrt{1 - \frac{700}{0{,}425.100.15^2.1{,}79}}\right]$$

$$x = 0{,}39\ cm \ \rightarrow \text{Domínio 2} \ \rightarrow \ \sigma sd = fyd = 43{,}48\frac{\text{kN}}{\text{cm}^2}$$

$$Asw = \frac{1}{fyd}.\frac{Md}{(d - 0{,}4x)} = \frac{1}{43{,}48}.\frac{700}{(15 - 0{,}4.0{,}54)} \cong 1{,}08\ \frac{\text{cm}^2}{\text{m}}$$

$$Asusp = \ 0{,}725\ \frac{\text{cm}^2}{\text{m}}$$

$$Asw, pmin = \ 1{,}9\ \frac{\text{cm}^2}{\text{m}}$$

O maior valor dentre as armaduras calcula, suspensão e mínima, foi a armadura mínima, portanto, utilizar a armadura mínima de 1,9 cm^2/m.

Considerando uma armadura distribuída de Ø 8 C/20 temos:

$$A_{ø8} = \frac{\pi.d^2}{4} = \ \frac{\pi.0{,}8^2}{4} \ \cong 0{,}5\ cm^2$$

Em 1 metro tem-se a quantidade de barras:

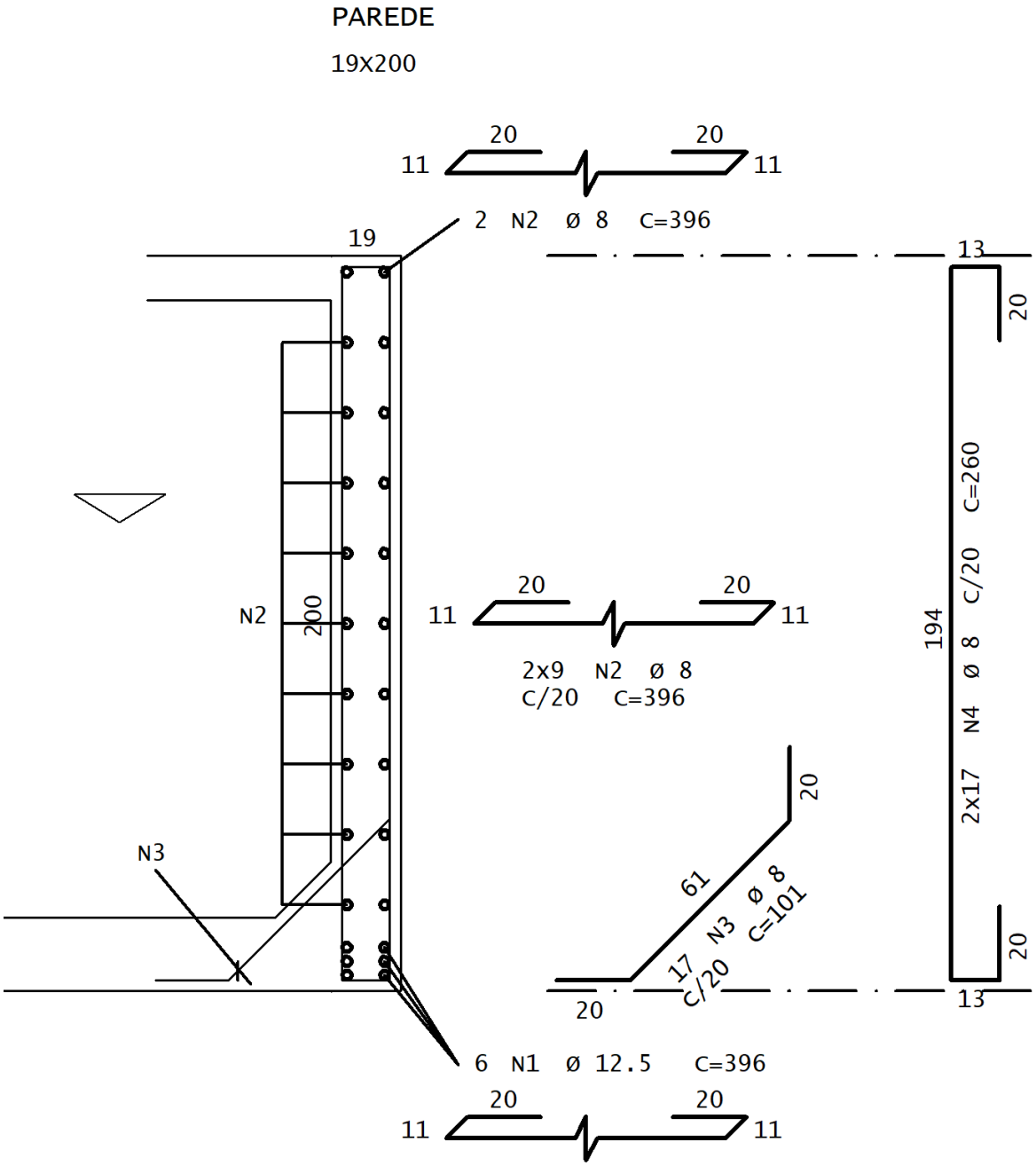
$$qb = \frac{100}{20} = 5 \text{ barras}$$

$$Asw, total = 5\ .0{,}5 = 2{,}5\frac{\text{cm}^2}{\text{m}} > 1{,}9\frac{\text{cm}^2}{\text{m}} \quad \text{Ok}$$

Em resumo, armadura do tirante tracionado (barras positivas) 6 Ø 12.5, armaduras para estribo, pele e mísula Ø 8 C/20.

A figura abaixo demonstra o detalhamento da parede calculada no exemplo:

Figura 2.11 – Detalhamento da parede exemplo.



Fonte: Autor próprio, 2024.

3 PROGRAMA COMPUTACIONAL

Para este trabalho foi criado 2 programas simples para auxiliar no dimensionamento e detalhamento de uma parede de reservatório biapoiada.

3.1 Linguagem utilizada

Para os programas desenvolvidos foi utilizada a linguagem Python e a IDE Visual Studio Code.

A escolha da linguagem é dada em função do programa ser um auxiliar junto ao TQS, que disponibiliza um API em Python para desenvolver automatizações, desde criar desenhos a alterar dados de projetos criado no programa.

3.2 Programa de dimensionamento

O programa para auxiliar no dimensionamento da parede foi desenvolvido baseado no exemplo apresentado neste trabalho.

O programa dimensiona uma parede biapoiada levando em conta o dimensionamento para flexão de uma viga-parede e o dimensionamento semelhante ao de uma laje biapoiada para o esforço devido ao empuxo da água.

Inicialmente o programa irá pedir para o usuário inserir os dados de entrada de geometria e carga da parede:

```
# Dados de entrada
nometxt = ""                        # Nome do arquivo gerado .txt
titulo = ""                         # Título da parede
largura = 0                         # Largura da parede
altura = 0                          # Altura da parede
vao = 0                             # Vão para o cálculo do comprimento das
armações horizontais
cobr = 0                            # Cobrimento da armação
hli = 0                             # altura da laje inferior cm
hls = 0                             # altura da laje superior cm
cae = 0                             # comprimento do apoio esquerdo cm
cad = 0                             # comprimento do apoio direito cm
pk1 = 0                             # carga superior na parede tf/m
pk2 = 0                             # carga inferior na parede tf/m
pp = 2.5                            # peso próprio do concreto armado
```

Algumas entradas foram feitas dentro de um *while* para evitar que o programa se encerre caso o dado preenchido não seja do tipo correto.

```
def obter_titulo():
    while True:
        try:
            return str(input("Digite o título da parede: "))
        except ValueError:
            print("Dado incorreto. Tente novamente.")
…
```

Após a entrada de dados, se inicia o cálculo dos esforços na parede.

```
# peso proprio da parede

print()
print("Calculo dos esforços na parede:")

pk3 = pp*largura/100*altura/100                 # calculo do peso próprio da
parede
print()
print(f"Peso próprio da parede {pk3} tf/m")

# Calculo dos esforços para viga biapoiada

pk = pk1 + pk2 + pk3                # carga total em tf/m
l = (vao - cae/2 - cad/2)/100       # vão de calculo de eixo a eixo de apoio
em m
Mk = pk*l**2/8                      # momento maximo em tfm
Rk = pk*l/2                         # reação de apoio em tf
…
```

Depois dos esforços calculados, o programa irá realizar as verificações e dimensionamentos necessários para uma viga-parede.

```
# relação vao/altura para viga-parede biapoiada

if vao/altura <= 2:
    print("Calcular como viga-parede")
else:
    print("Não é viga-parede, dimensionar como viga")
    sys.exit()

# coeficientes para concreto e aço

gamaf = 1.4         # coeficiente de majoração de esforços
gamac = 1.4         # coeficiente de minoração do concreto
gamas = 1.15        # coeficiente de minoração do aço …
```

Depois da armadura dimensionada o programa vai pedir para o usuário selecionar uma opção de alojamento desejada para a armadura do banzo tracionada, deixando livre a escolha para quantidade de barras em relação as bitolas.

```
# Escolha da armadura para o banzo tracionado

# Lista de diâmetros em centímetros
diametros = [0.63, 0.8, 1, 1.25, 1.6, 2, 2.5]

# Calculando as áreas e armazenando como lista de dicionários
bite = [{"diametro": d, "area": math.pi * (d / 2) ** 2} for d in diametros]

# Exibindo as opções de alojamento
print("\nOpções de alojamento:\n")
for i, bitola in enumerate(bite, 1):
    bitpos = bitola["diametro"] * 10  # Convertendo para milímetros
    # Calculando a quantidade de barras necessárias
    quantidade_barras = math.ceil(Ast / bitola["area"])  # Número de barras
arredondado para cima
    # Garantindo que a quantidade de barras seja par
    if quantidade_barras % 2 != 0:
        quantidade_barras += 1
    print(f"{i}. {quantidade_barras} Ø {bitpos}")

# Pedindo para o usuário escolher uma opção
escolha = int(input("\nEscolha o número da bitola desejada: "))

# Verificando se a escolha é válida
if 1 <= escolha <= len(bite):
    bitola_escolhida = bite[escolha - 1]  # Obtém a bitola escolhida
    qpos = Ast / bitola_escolhida["area"]  # Calcula o valor inicial
    qpos = math.ceil(qpos)  # Arredonda para cima
    # Ajusta para o próximo número par, se necessário
    if qpos % 2 != 0:
        qpos += 1
    # Exibe o resultado para a bitola escolhida
    bitpos = bitola_escolhida["diametro"] * 10  # Convertendo para milímetros
    quantidade_barras = math.ceil(Ast / bitola_escolhida["area"])  #
Quantidade de barras para a bitola escolhida
    # Garantindo que a quantidade de barras seja par
    if quantidade_barras % 2 != 0:
        quantidade_barras += 1
    print(f"\nArmadura positiva: {qpos} Ø {bitpos}")
else:
    print("\nEscolha inválida! Tente novamente.")
…
```

Em seguida o programa vai pedir para o usuário selecionar mais algumas opções, como por exemplo, escolher como a armadura do banzo tracionado (positiva) será distribuída.

Depois será realizada a verificação nos apoios, o programa vai realizar automaticamente a verificação do esmagamento do concreto e o comprimento de ancoragem necessário.

Logo após essas verificações, o dimensionamento do banzo tracionado vai ser finalizado e se iniciara o dimensionamento da pele e estribo devido à força de empuxo da água, onde é verificada a armadura de suspensão, a armadura mínima para o empuxo e a armadura da seção devido ao empuxo da água aplicado na lateral da parede.

```
# Armadura de suspensão

Pds = gamac * pk2   # Força a ser suspensa
print()
print("Armadura de suspensão: ")
print()
print(f"Força a ser suspensa: {Pds:.2f} tf/m")

Asusp = (Pds/(fyd/100)) / 2     # calculo da armadura de suspensão
print()
print(f"Armadura de suspensão: {Asusp:.3f} cm²/m por face")

# Armadura de pele e estribo

print()
print("Armadura de pele e estribo: ")
Aswmin = 0.1 * largura      # armadura mínima para pele e estribo
print()
print(f"Armadura mínima para pele e estribo = {Aswmin:.2f} cm²/m por face")

# Calculo do esforço para pele e estribo
...

# Calculo da armadura de flexão para estribo, pele e misula devido ao empuxo

# dados para o dimensionamento da seção da parede
b = 100                         # largura (unidade de m)
h = largura                     # altura util da parede onde o empuxo é
aplicado, no caso da parede, em sua largura
d = largura - 4                 # altura util de calculo
dlinhae = 4                     # d'
xlim = 0.45 * d                 # limite para armadura dupla segundo NBR 6118
x23 = 0.259 * d                 # limite para domínio 2
x34 = 0.628 * d                 # limite para domínio 3
Md = Mpar * gamaf * 1000        # Momento de calculo em kNm/m
```

```
# Calculo do valro de x - linha neutra
divisao = Md / (0.425 * b * (d**2) * (fcd/10))
raiz = (1 - divisao) ** 0.5
x = 1.25 * d * (1 - raiz)
…
```

Ao final do dimensionamento o programa vai pedir novamente para o usuário selecionar uma opção de alojamento desejada, para o estribo, pele e mísula, será mostrada as opções mais próximas da área de aço necessária com espaçamentos mais usuais e bitolas.

```
# escolha do alojamento para estribo, pele e misula
# Lista de diâmetros em centímetros e suas áreas
bite = [
    {"diametro": 0.63, "area": 0.3117},
    {"diametro": 0.8, "area": 0.5027},
    {"diametro": 1.0, "area": 0.7854},
    {"diametro": 1.25, "area": 1.2272},
    {"diametro": 1.6, "area": 2.0106},
    {"diametro": 2.0, "area": 3.1416},
    {"diametro": 2.5, "area": 4.9087},
]

# Espaçamentos permitidos (em cm) - limitados aos espaçamentos mais usuais
espaçamento = [10, 12, 15, 20]

# Comprimento de referência (1 metro) - devido a armadura ser distribuida
comprimento = 100  # em cm

print("\nOpções de alojamento para estribo, pele e misula:")

melhores_opcoes = []

# Encontrar a melhor combinação para cada espaçamento
for esp in espaçamento:
    melhor_opcao = None  # Inicializa a melhor opção para cada espaçamento
    menor_sobra = float('inf')  # Inicializa a menor sobra como infinito

    for bitola in bite:
        # Calculando o número de barras no comprimento de 1 metro
        quantidade_barras = comprimento / esp  # Divisão inteira para número
de barras

        # Calculando a área total fornecida corretamente
        Aste = (comprimento / esp) * bitola["area"]  # Área total de aço
fornecida
```

```
        # Se o alojamento atender o Asw
        if Aste >= Aswe:
            sobra = abs(Aste - Aswe)  # Diferença absoluta entre a área
fornecida e a necessária
            if sobra < menor_sobra:
                menor_sobra = sobra
                melhor_opcao = {
                    "espaçamento": esp,
                    "quantidade_barras": quantidade_barras,
                    "diametro": bitola["diametro"] * 10,  # Convertendo para
mm
                    "Aste": round(Aste, 2),  # Arredondando a área total de
aço para 2 casas decimais
                }

    # Adiciona o melhor alojamento encontrado para o espaçamento
    if melhor_opcao:
        melhores_opcoes.append(melhor_opcao)

# Ordenando as opções pela menor diferença da área total de aço em relação ao
Aswe
melhores_opcoes.sort(key=lambda x: abs(x['Aste'] - Aswe))

# Exibindo apenas as opções mais próximas
print("Escolha uma das opções de alojamento mais próximas:\n")
for i, opcao in enumerate(melhores_opcoes, 1):
    print(f"{i}. Ø {opcao['diametro']} C/{opcao['espaçamento']} → Área total
de aço = {opcao['Aste']} cm²")

# Pedindo para o usuário escolher uma opção
escolha = int(input("\nEscolha o número da opção desejada: "))

# Verificando se a escolha é válida
if 1 <= escolha <= len(melhores_opcoes):
    opcao_escolhida = melhores_opcoes[escolha - 1]  # Obtém a opção escolhida
    print(f"\nArmadura de pele e estribo = Ø {opcao_escolhida['diametro']}
C/{opcao_escolhida['espaçamento']}")
else:
    print("\nEscolha inválida! Tente novamente.")
…
```

Por fim, será informado um resumo com as armaduras escolhidas.

```
print("Resumo do dimensionamento para a parede:")
print(f"{NEGRITO}\nArmadura positiva →  {qpos} Ø {bitpos}")
print(f"Armadura de pele, estribo e misula → Ø {opcao_escolhida['diametro']}
C/{opcao_escolhida['espaçamento']}")
```

```
print(f"Para armadura negativa de viga-parede, pode-se repetir a armadura de
pele → 2 Ø {opcao_escolhida['diametro']}")
```

3.3 Programa para o detalhamento

Para o desenho da parede, a programação é feita baseada na linguagem Python com a API do TQS, portanto, muitos comandos desse código são pertencentes a essas bibliotecas especificas disponibilizadas.

Antes de iniciar a programação é necessário instalar a API do TQS, quando instalada a versão 23 do programa, junto aos arquivos é criada uma pasta chamada Python, que fica em \TQSW\EXEC\PYTHON. Dentro desta pasta existem os arquivos de exemplo, além de um manual da programação TQS x Python e o arquivo .whl da API para ser instalado. O pacote TQS é fornecido no formato de um arquivo do tipo *Wheel* e pode ser instalado utilizando o utilitário PIP, que é incluído na instalação do Python. Para instalar o pacote, dentro do terminal da IDE ou dentro do cmd, basta digitar:

```
pip install TQSPythonInterface-1.2.1-py310-none-any.whl
```

O arquivo TQSPythonInterface-1.2.1-py310-none-any.whl pode mudar de acordo com a versão do TQS.

Para o programa de detalhamento, foi adotada uma abordagem mais flexível, aproveitando que estão separados. Isso proporcionou maior liberdade para o usuário, permitindo a geração de desenhos de diversas maneiras. Entre as opções disponíveis, destacam-se a definição de desníveis de lajes, a possibilidade de incluir ou não armadura de mísula, a inclusão de armadura de nó de pórtico, entre outras funcionalidades.

No início é comum importar as bibliotecas, para este em específico nota-se a importação das bibliotecas do TQS.

```
# Inicio do programa para detalhamento de parede de reservatório no TQS

# Importando as bibliotecas Python

import os
import math
import tkinter as tk
from tkinter import filedialog
from TQS import TQSDwg, TQSGeo
…
```

Logo no início já é utilizado o comando da API para inicializar um desenho formato DWG para TQS.

```
# Inicializar o desenho
dwg = TQSDwg.Dwg()
dwg.settings.systemId    = TQSDwg.IEDTAB       # AGC & DP
dwg.settings.subSystemId = 13                  # Desenho de armação
dwg.file.LoadColors()   # carrega as cores
…
```

Como mencionado, devido a maior liberdade para o desenho, tem-se uma grande quantidade de entrada de dados onde é possível ver as possibilidades.

```
# entrada de dados
# variaveis
nomedwg = ""                          # Nome do arquivo gerado .dwg
titulo = ""                           # Titulo da parede no desenho
largura = 0                           # Largura da parede
altura = 0                            # Altura da parede
vao = 0                               # Vão para o calculo do comprimento das
armações horizontais
cobr = 0                              # Cobrimento da armação
agua = "N"                            # Indicação de agua
lie = "N"                             # Laje inferior a esquerda?
hlie = 0                              # Altura da laje inferior esquerda
deslie = 0                            # Desnivel do fundo da parede ao fundo da
laje esquerda
mislie = "N"                          # Misula na laje inferior esquerda
hmislie = 0                           # Altura da misula inferior esquerda
bitmislie = 0                         # Bitola da misula inferior esquerda
espmislie = 0                         # Espaçamento da misula infeior esquerda
aeve = "N"                            # Armadura contra empuxo ao vazio na laje
inferior esquerda
bitaeve = 0                           # Bitola da armadura contra empuxo ao
vazio na laje inferior esquerda
espaeve = 0                           # Espaçamento da armadura contra empuxo ao
vazio na laje inferior esquerda
aevep = 0                             # Tamanho da 'perna' da armadura contra
empuxo ao vazio na laje inferior esquerda
lse = "N"                             # Laje superior esquerda
hlse = 0                              # Altura da laje superior esquerda
deslse = 0                            # Desnivel do topo da parede ao topo da
laje superior esquerda
lid = "N"                             # Laje inferior a direita?
hlid = 0                              # Altura da laje inferior direita
deslid = 0                            # Desnivel do fundo da parede ao fundo da
laje direita
mislid = "N"                          # Misula na laje inferior direita
```

```
hmislid = 0                          # Altura da misula inferior direita
bitmislid = 0                        # Bitola da misula inferior direita
espmislid = 0                        # Espaçamento da misula infeior direita
aevd = "N"                           # Armadura contra empuxo ao vazio na laje
inferior direita
bitaevd = 0                          # Bitola da armadura contra empuxo ao
vazio na laje inferior direita
espaevd = 0                          # Espaçamento da armadura contra empuxo ao
vazio na laje inferior direita
aevdp = 0                            # Tamanho da 'perna' da armadura contra
empuxo ao vazio na laje inferior direita
lsd = "N"                            # Laje superior direita
hlsd = 0                             # Altura da laje superior direita
deslsd = 0                           # Desnivel do topo da parede ao topo da
laje superior direita
qpos = 0                             # Quantidade de barras positivas
bitpos = 0                           # Bitola armação positiva
qneg = 0                             # Quantidade de barras negativas
bitneg = 0                           # Bitola armação negativa
espele = 0                           # Espaçamento da armadura de pele
bitpele = 0                          # Bitola da Armadura de pele
vest = 0                             # Vão para o calculo da quantidade dos
estribos e misulas
espest = 0                           # Espaçamento dos estribos
bitest = 0                           # Bitola dos estribos
tipoestribo = 0                      # Tipo de estribo, aberto ou fechado
…
```

Sendo um programa para gerar um desenho DWG, é comum ao longo do código ter a programação para referência de coordenadas X e Y no plano.

```
# Indicação de agua
xae1 = 0-90-50              # Pontos para desenho da representação
yae1 = altura+70
xae2 = xae1+50
yae2 = yae1
xae3 = xae1+25
yae3 = yae1-15

xad1 = 0+largura*2+90
yad1 = yae1
xad2 = xad1+50
yad2 = yad1
xad3 = xad1+25
yad3 = yae3
…
```

Também é comum o uso de comandos para criação de linhas através das coordenadas definidas.

```
if agua == "E":
    dwg.draw.level = 0
    dwg.draw.Line (xae1,yae1,xae2,yae2)
    dwg.draw.Line (xae1,yae1,xae3,yae3)
    dwg.draw.Line (xae3,yae3,xae2,yae2)
…
```

O comando *dwg.draw.level* serve para definir o nível utilizado no desenho, nível no TQS é correspondente ao *layer* no *AutoCad*. O comando *dwg.draw.Line* utiliza 2 pontos para traçar uma linha, com o respectivo ponto X,Y no plano.

Após toda a programação de algumas definições e da geometria da parede, tem-se a utilização dos comandos para criação das barras de armadura.

Existem comandos específicos para cada tipo de armadura, por exemplo, existem diferenças entre um ferro reto, um ferro genérico e um estribo.

Segue abaixo o exemplo da criação de um ferro reto:

```
# textos dos ferros longitudinais

positivo = TQSDwg.SmartRebar (dwg)              # Definindo o ferro positivo
positivo.type = TQSDwg.ICPFRT                   # Tipo de ferro - ferro reto
positivo.cover                  = cobr          # Cobrimento p/uso geral
positivo.mark                   = 1             # Numero da posição
positivo.quantity               = qpos          # Num ferros
positivo.multiplier             = 1             # Multipl do numero de ferros
positivo.diameter               = bitpos        # Bitola mm
positivo.straightBarMainLength  = vao-2*cobr    # Comprimento horizontal (cm)
positivo.straightBarLeftLength  = dp            # Dobra a esquerda
positivo.straightBarRightLength = dp            # Dobra a direita
positivo.straightBarLeftLength2 = 20            # 2a dobra a esquerda
positivo.straightBarRightLength2 = 20           # 2a dobra a direita
positivo.straightBarZone        = TQSDwg.ICPPOS # Tipo de ferro reto positivo
ou negativo
positivo.RebarTextDisplayPosition (xlp1+15,ylp7+25,10,0,0,1,0) # Texto do
ferro

negativo = TQSDwg.SmartRebar (dwg)              # Definindo o ferro negativo
negativo.type = TQSDwg.ICPFRT                   # Tipo de ferro - ferro reto
negativo.cover                  = cobr          # Cobrimento p/uso geral
negativo.mark                   = 2             # Numero da posição
negativo.quantity               = qneg          # Num ferros
negativo.multiplier             = 1             # Multipl do numero de ferros
negativo.diameter               = bitneg        # Bitola mm
```

```
negativo.straightBarMainLength    = vao-2*cobr    # Comprimento horizontal (cm)
negativo.straightBarLeftLength    = dn            # Dobra a esquerda
negativo.straightBarRightLength   = dn            # Dobra a direita
negativo.straightBarLeftLength2   = 20            # 2a dobra a esquerda
negativo.straightBarRightLength2  = 20            # 2a dobra a direita
negativo.straightBarZone          = TQSDwg.ICPNEG # Tipo de ferro reto positivo
ou negativo
negativo.RebarTextDisplayPosition (xln1+15,yln1-30,10,0,0,1,0) # Texto do
ferro
…
```

Existem comandos para cada opção do ferro no TQS, como tipo, cobrimento, quantidade, bitola, dobras e outros dados.

Segue o exemplo do código para criação de um estribo:

```
estribo       = TQSDwg.SmartRebar (dwg) # Definindo o ferro estribo fechado
estribo.type  = TQSDwg.ICPSTR           # Estribo
estribo.mark  = 4                       # Numero da posição
estribo.quantity = int(vest/espest)     # Num ferros
estribo.diameter = bitest               # Bitola mm
estribo.spacing  = espest               # Espaçamento
estribo.cover    = cobr                 # Cobrimento p/uso geral
inivel           = -1                   # Nivel defaultp
iestilo          = -1                   # Estilo default
icor             = -1                   # Cor default
estribo.stirrupType = 0                 # Tipo de estribo
estribo.stirrupLegs = 0                 # Numero de ramos
estribo.stirrupSectionWidth  = largura  # Base da seção
estribo.stirrupSectionHeight = altura   # Altura da seção
estribo.stirrupSectionWidth2 = 0        # Base max seção var de min/max
estribo.stirrupSectionHeight2     = 0   # Altura max idem
estribo.stirrupEffectiveLeftWidth = 0   # Largura colab a esquerda
estribo.stirrupEffectiveRightWidth= 0   # Largura colab a direita
estribo.stirrupSlabBendLength     = 0   # Dobra do estribo na laje
estribo.RebarTextDisplayPosition (largura*2+570,y1s-15,10,0,0,0,1)
estribo.RebarLine (largura*2+550,y1s,0,2,1,1,1,0,-1,-1,-1)  # Linha de ferro
…
```

Depois de todas as armaduras definidas, existe um comando que renumera automaticamente as posições, para evitar erros de numeração repetida e pulo de numeração, com isso é possível dentro do TQS extrair automaticamente a tabela de ferros.

Ao final do programa, existem 2 comandos do TQS para limpar os blocos e níveis carregados no desenho, para tornar o arquivo DWG mais limpo e mais leve, ocupando menos

espaço no armazenamento do computador. Para este tipo de desenho isso não é um item tão crucial, mas pode se tornar para desenhos maiores.

```
# limpar blocos e niveis
dwg.file.PurgeBlocks()
dwg.file.PurgeLevels()
```

Com todos os dados preenchidos, o usuário pode selecionar uma pasta na qual o desenho vai ser salvo, e logo em seguida o programa vai abrir o desenho para o usuário visualizar e caso prefira, realizar edições, e assim o desenho está pronto para ser inserido no projeto.

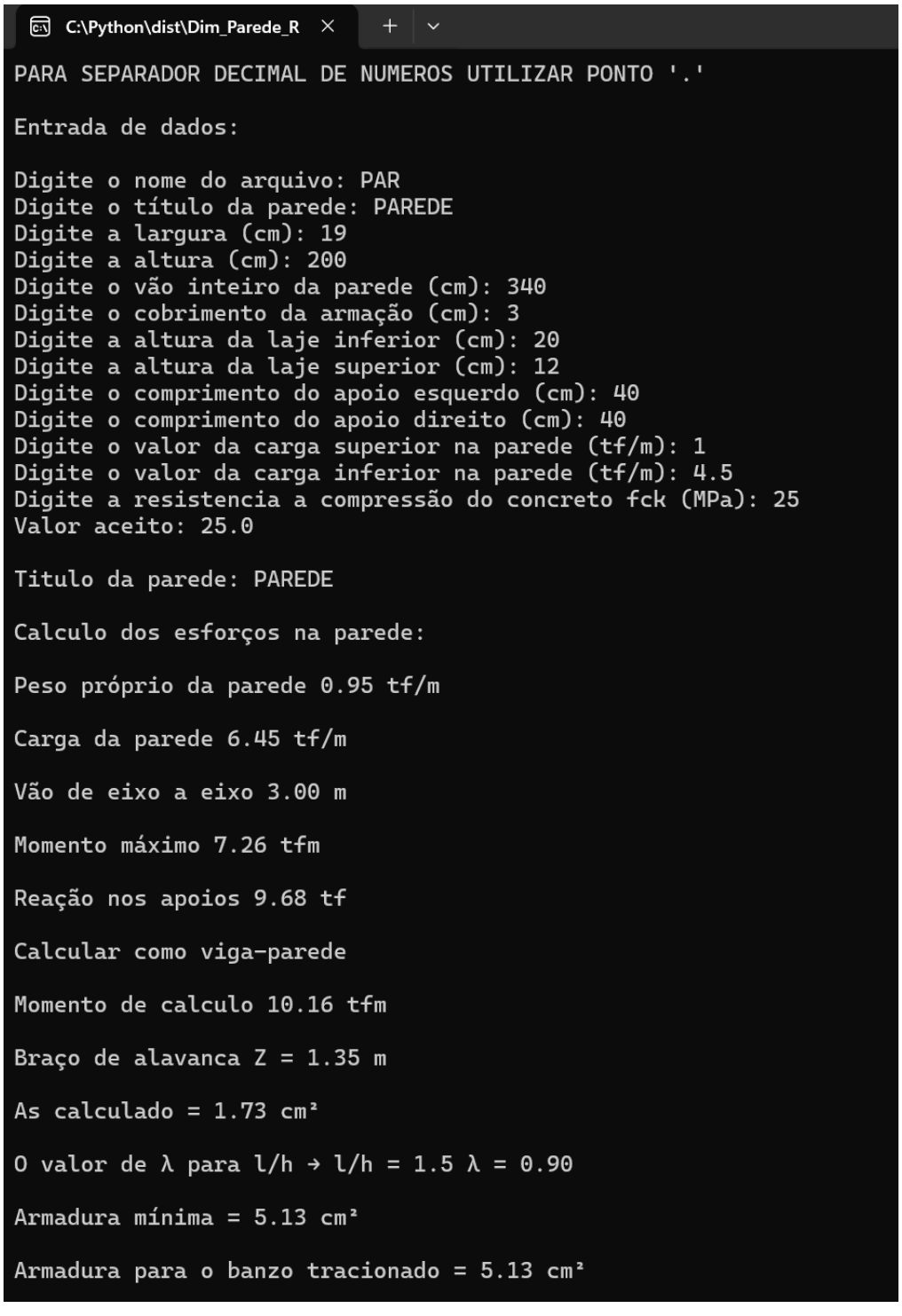
```
def escolher_pasta():
    # Inicializa o tkinter
    root = tk.Tk()
    root.withdraw()  # Oculta a janela principal

    # Força a janela a estar no topo
    root.attributes("-topmost", True)

    # Abre o diálogo para seleção da pasta
    caminho_pasta = filedialog.askdirectory(title="Selecione uma pasta")

    # Fecha a janela principal
    root.destroy()

    if caminho_pasta:
        print(f"Caminho selecionado: {caminho_pasta}")
        return caminho_pasta
    else:
        print("Nenhuma pasta foi selecionada.")
        return None

# Chamando a função
caminho_salvo = escolher_pasta()

#if caminho_salvo:
    # Use o caminho salvo para outras operações
    #print(f"Caminho salvo: {caminho_salvo}")

#salvar o arquivo
dwg.file.SaveAs (caminho_salvo + "/" + nomedwg + ".dwg")

#abrir o arquivo após o salvamento
os.startfile(caminho_salvo + "/" + nomedwg + ".dwg")

system('cls')
```

4 EXEMPLO PRATICO DO PROGRAMA

4.1 Exemplo do programa de dimensionamento

Demonstração do programa de dimensionamento conforme o exemplo apresentado no Capítulo 2.

Entrada de dados e resultados:



```
Opções de alojamento:

1. 18 Ø 6.3
2. 12 Ø 8.0
3. 8 Ø 10
4. 6 Ø 12.5
5. 4 Ø 16.0
6. 2 Ø 20
7. 2 Ø 25.0

Escolha o número da bitola desejada: 4

Armadura positiva: 6 Ø 12.5

Area de aço total escolhida para o banzo tracionado = 7.36 cm²

Distribuição da armadura positiva em camadas (C) ou 0.15h (H)?c
Valor aceito

Verificação da tensão nos apoios:

Força de calculo da reação de apoio = 13.54

tg teta = 1.80

Angulo teta = 60.95°

d' = 6.875 cm

Altura do nó de apoio 'u' = 13.75 cm

σd = 15.18 MPa

σ2d do apoio esquerdo = 1.96 MPa

σ2d do apoio direito = 1.96 MPa

Para o concreto de 25 MPa:
fcdr = 9.64 MPa

Verificação do esmagamento do concreto na região dos apoios:

Segurança contra o esmagamento do concreto no apoio esquerdo OK!

Segurança contra o esmagamento do concreto no apoio direito OK!

Comprimento básico de ancoragem necessário:

O lb necessário para 25 MPa e para a Ø 12.5 em região de boa aderência com gancho é igual a 33 cm

Valores mínimos para o comprimento de ancoragem:

lb necessário: 4.34 cm
lb mínimo 1 = 9.90 cm
lb mínimo 2 = 12.50 cm
lb mínimo 3 = 10.00 cm

Portanto o lb necessário é de = 12.50 cm

O comprimento do apoio esquerdo de 40.0 cm é maior que o lb necessário de 12.5 cm,
portanto é possível realizar a ancoragem no apoio
```

```
Armadura de suspensão:

Força a ser suspensa: 6.30 tf/m

Armadura de suspensão: 0.725 cm²/m por face

Armadura de pele e estribo:

Armadura mínima para pele e estribo = 1.90 cm²/m por face

Momento devido ao empuxo na parede = 0.50 tfm/m

Calculo da seção devido ao empuxo da água na parede:

Base = 100 cm
Altura = 19.0 cm
Altura útil d = 15.0 cm
d' = 4 cm
Altura limite da linha neutra = 6.75 cm
Altura limite para Domínio 2 = 3.89 cm
Altura limite para Domínio 3 = 9.42 cm
Momento de calculo devido ao empuxo = 696.76 kNcm/m

Posição da linha neutra = 0.39 cm
Domínio 2

Posição da linha neutra Ok!

Armadura calculada para pele e estribo = 1.08 cm²/m

Armaduras calculadas para pele e estribo:

Armadura de suspensão: 0.725 cm²/m
Armadura mínima : 1.90 cm²/m
Armadura calculada para pele e estribo = 1.08 cm²/m

Portanto armadura para pele e estribo → Asw = 1.90 cm²/m

Opções de alojamento para estribo, pele e misula:
Escolha uma das opções de alojamento mais próximas:

1. Ø 6.3 C/15 → Área total de aço = 2.08 cm²
2. Ø 8.0 C/20 → Área total de aço = 2.51 cm²
3. Ø 6.3 C/12 → Área total de aço = 2.6 cm²
4. Ø 6.3 C/10 → Área total de aço = 3.12 cm²

Escolha o número da opção desejada: 2

Armadura de pele e estribo = Ø 8.0 C/20

Resumo do dimensionamento para a parede:

Armadura positiva →  6 Ø 12.5
Armadura de pele, estribo e misula → Ø 8.0 C/20
Para armadura negativa de viga-parede, pode-se repetir a armadura de pele → 2 Ø 8.0
```

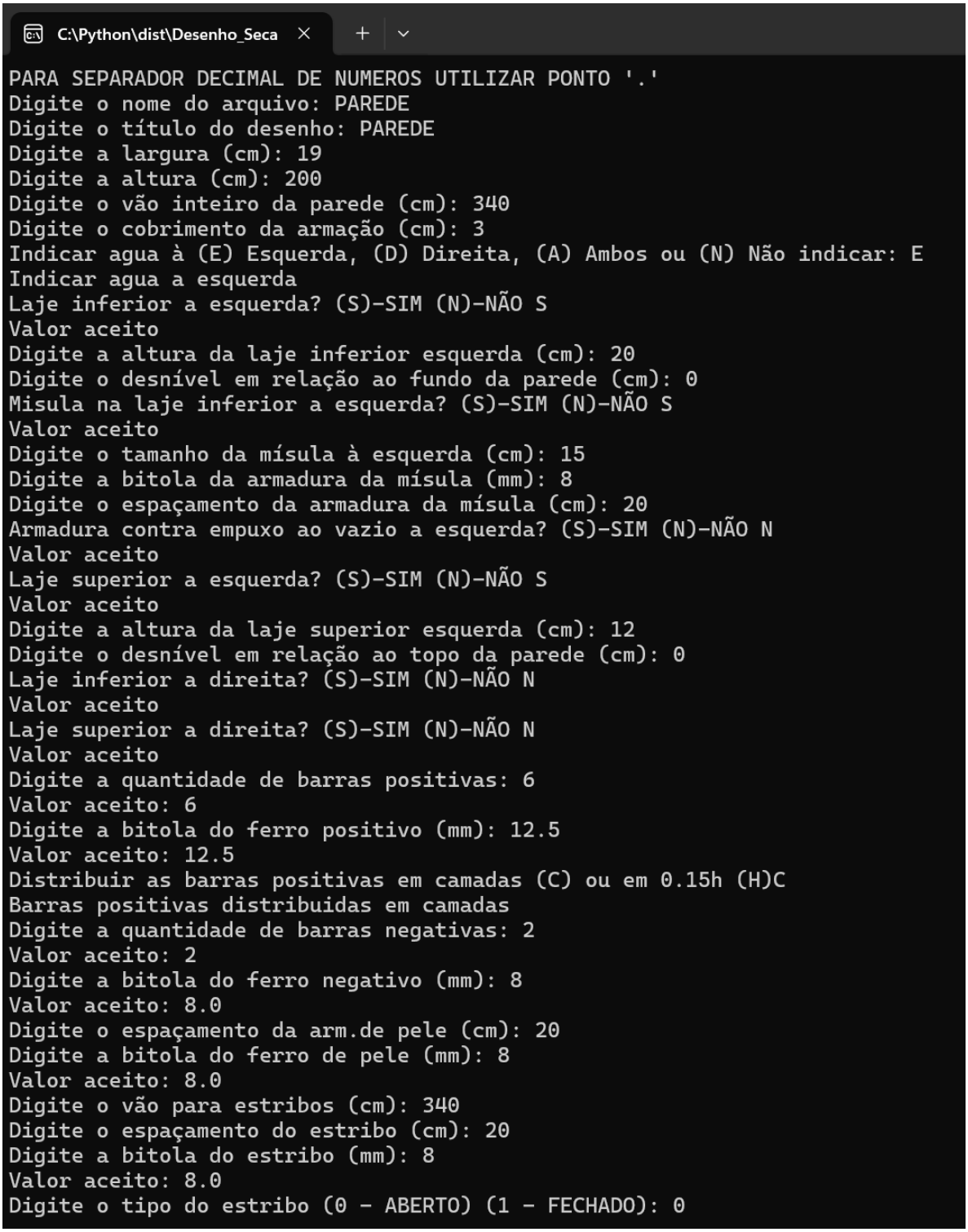
Foram obtidos os mesmos resultados do exemplo, validando os resultados do programa.

Desta forma o dimensionamento de uma parede de reservatório biapoiada qualquer fica muito rápido de ser obtido.

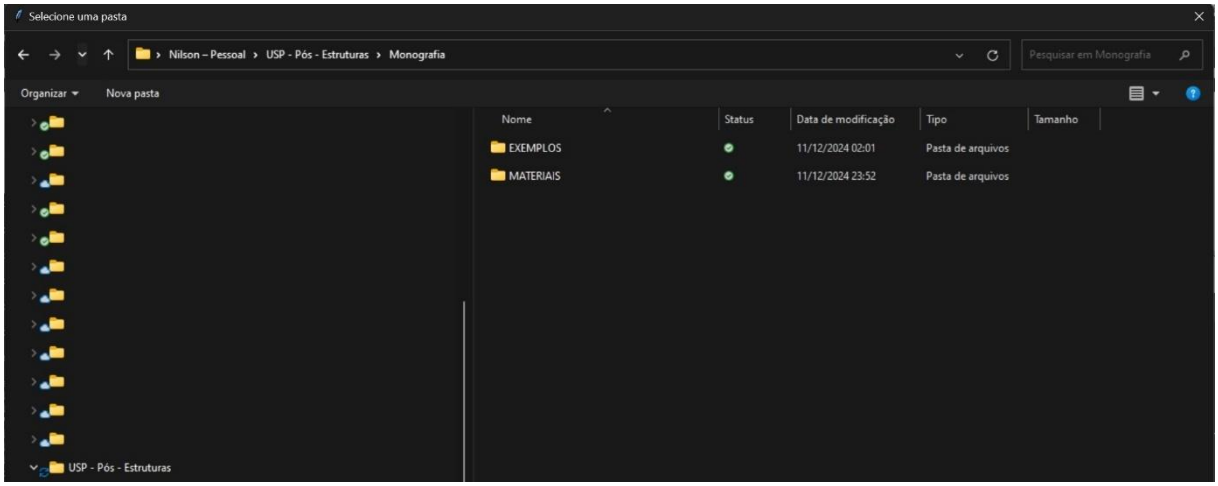
4.2 Exemplo do programa de detalhamento

Demonstração do programa de detalhamento conforme o exemplo apresentado no Capítulo 2.

Entrada de dados:

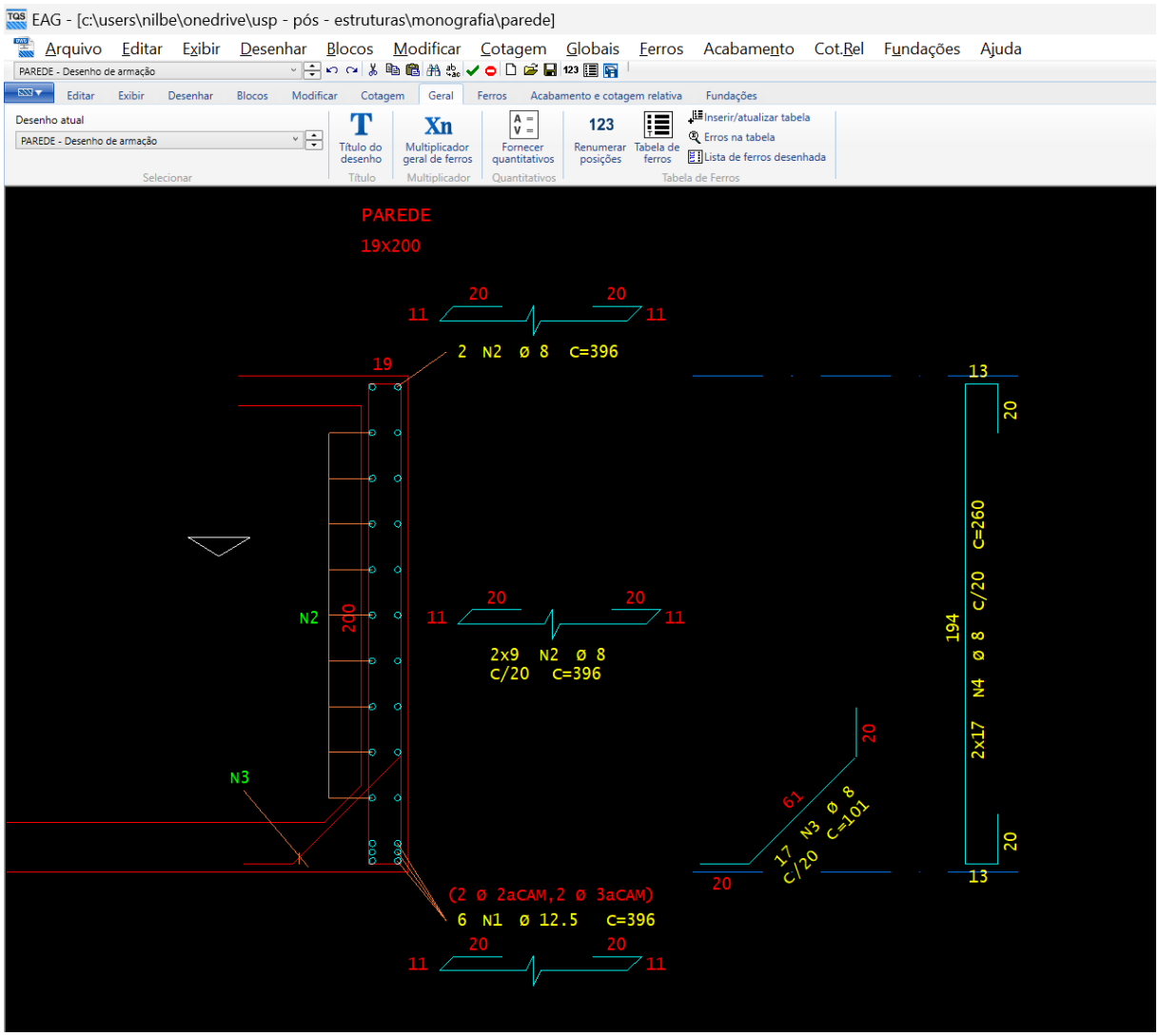


Seleção da pasta de destino do arquivo .DWG:



Desenho gerado pelo programa aberto no TQS:

Figura 4.1 – Detalhamento da parede pelo programa criado no TQS.



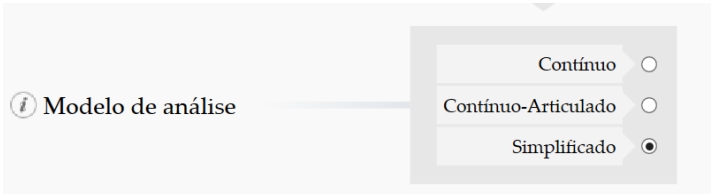
Fonte: Autor próprio, 2024.

5 RESULTADOS COM O MÓDULO DE RESERVATÓRIOS DO TQS

Para tentar aproximar o dimensionamento e detalhamento do TQS com o exemplo do trabalho foram feitas algumas alterações de critérios. Vale lembrar que o exemplo está sendo realizado na versão 23 do TQS.

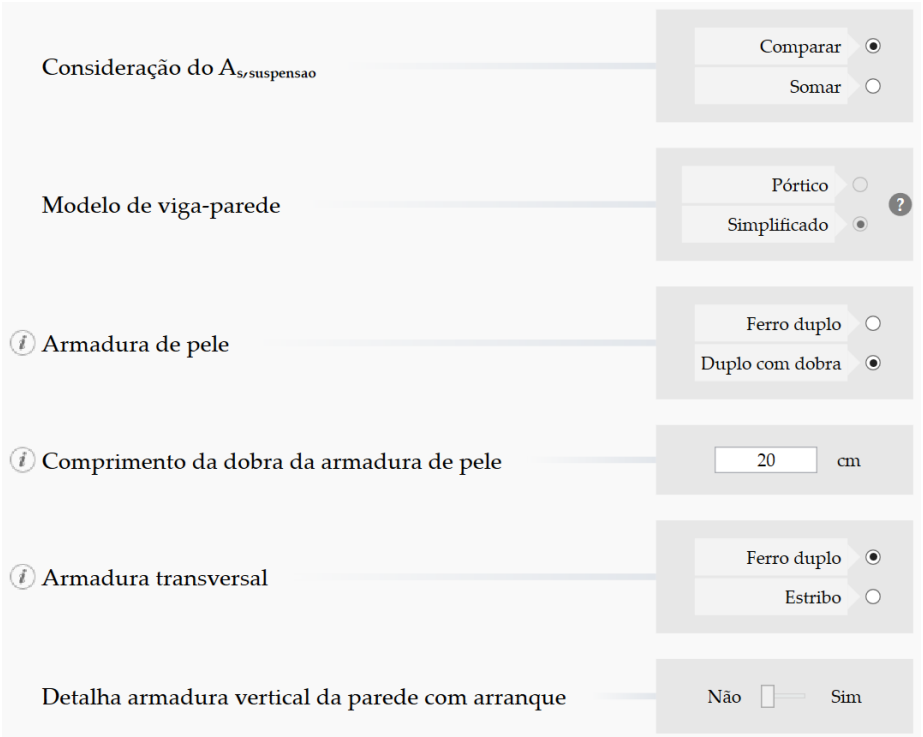
5.1 Dados do reservatório pelo TQS

No critério de modelagem foi utilizada a opção Simplificado em Modelo de análise, para tentar se aproximar ao máximo do exemplificado anteriormente:



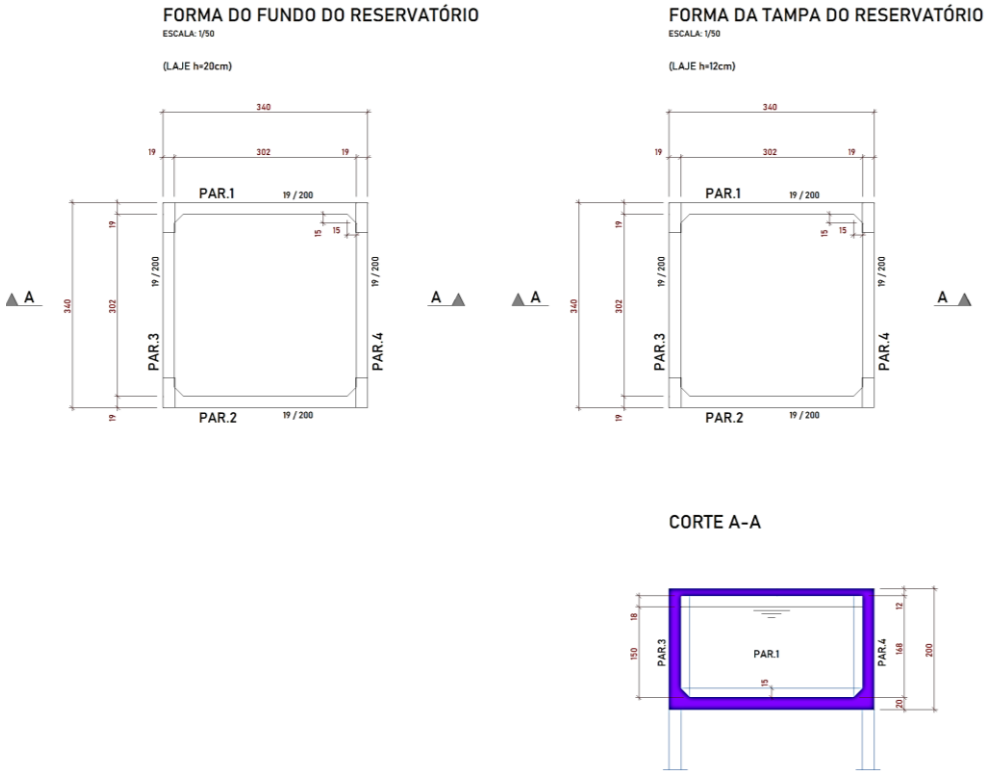
Simplificado
Realizado pelos projetistas em geral, engasta todos os lados das lajes de fundo, engasta os três lados adjacentes entre o fundo e as paredes e o lado entre a parede e tampa é articulada, ou seja, torna-se um apoio fixo. A tampa é definida como totalmente articulada e apoiada nos seus lados

Em detalhamento foram utilizadas as opções 'Comparar' para 'As/suspensão', 'Duplo com dobra' para 'Armadura de pele', '20cm' para 'Comprimento da dobra da armadura de pele', 'Ferro duplo' para 'Armadura transversal' e 'Não' para 'Detalha armadura vertical da parede com arranque'.



Tentando aproximar ao máximo a geometria da parede do trabalho dentro das limitações do módulo do TQS temos:

Figura 5.1 – Detalhamento do reservatório pelo TQS.



Fonte: Autor próprio, 2024.

Figura 5.2 – Dados de geometria e cargas considerados pelo cálculo do TQS.

```
-----------------------------------------------------------------------------------------------------------
     RELATÓRIO DOS DADOS DE ENTRADA DO RESERVATÓRIO
                    V22.12.029

===========================================================================================================
Reservatório Elevado

Reservatório Elevado, apoiado nos vértices

Peso Específico do Concreto = 2.50   tf/m3      Peso Específico do Líquido = 1.00    tf/m3

Modelo de cálculo: Simplificado
-----------------------------------------------------------------------------------------------------------
Geometria da Célula 1
    Lx(cm)      Ly(cm)      Altura(cm)      Altura Lâmina dágua(cm)      Volume dágua cálculo(m3 )    Volume dágua
    321         321          184                 150                          15.46                  13.68

Espessuras:     Tampa     Fundo     PAR.1a     PAR.2a     PAR.3     PAR.4
                 12        20         19         19         19        19          [cm]

Cargas:
          Sobrec.(tf/m2  )     Peso P.(tf/m2  )      Peso água(tf/m2  )   Soma(tf/m2  )   Resultante(tf )
Tampa         0.000              0.300                  *****              0.300            3.09
Fundo         0.000              0.500                  1.500              2.000           20.61
          Peso P.(tf/m2  )     Empuxo água(tf/m2  )  Resultante Vert.(tf )  Resultante Empuxo(tf )
PAR.1a        0.475              1.500                  2.81               4.43
PAR.2a        0.475              1.500                  2.81               4.43
PAR.3         0.475              1.500                  2.81               4.43
PAR.4         0.475              1.500                  2.81               4.43
-----------------------------------------------------------------------------------------------------------
```

Fonte: Autor próprio, 2024.

Figura 5.3 – Dados do dimensionamento da parede calculada pelo TQS.

LAJE: PAR.1a
Lx = 321 Ly = 184 h = 19 [cm]

Direção X	Astot (cm2/m)	Bitola(mm)	Espaç(cm)	Direção Y	Astot (cm2/m)	Bitola(mm)	Espaç(cm)
Esq	2.85	6.3	10	Inf	2.85	6.3	10
Meio	2.85	6.3	10	Meio	2.85	6.3	10
Dir	2.85	6.3	10	Sup	2.85	6.3	10

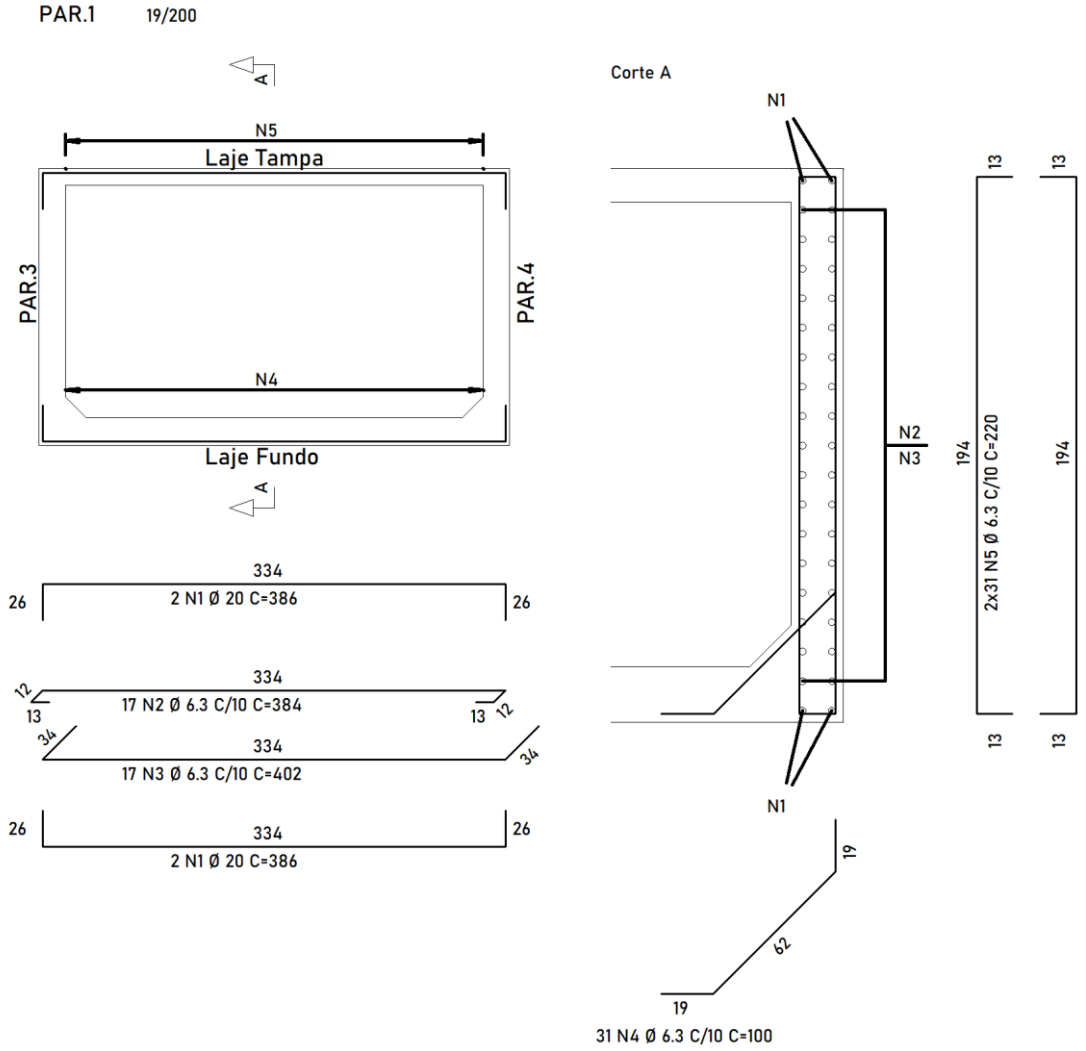
Armaduras nas Vigas-Paredes:

	Longitudinal na face Inferior/Superior*																
Parede	qtampa	qfundo	qpp	Area	Vd	Md	L	He	bw	Z	Ascal	Asmin	hs	NFer	Bit	NFer*	Bit*
	tf/m2	tf/m2	tf/m	m2	tf	tfm	m	m	cm	cm	cm2	cm2	cm		mm		mm
PAR.1a	0.300	2.000	0.95	2.58	6.28	5.04	3.21	1.84	19	1.4	0.84	5.24	29.9	2	20	2	20
PAR.2a	0.300	2.000	0.95	2.58	6.28	5.04	3.21	1.84	19	1.4	0.84	5.24	29.9	2	20	2	20
PAR.3	0.300	2.000	0.95	2.58	6.28	5.04	3.21	1.84	19	1.4	0.84	5.24	29.9	2	20	2	20
PAR.4	0.300	2.000	0.95	2.58	6.28	5.04	3.21	1.84	19	1.4	0.84	5.24	29.9	2	20	2	20

	Suspensao		de Pele (de alma)			Complementar	
Parede	Asusp	Nd	Bnec	Aspel	Aspelmin	Asc	0.3*He
	(cm2/m)	(tf)	(cm)	(cm2/m)	(cm2/m)	(cm2/m)	(cm)
PAR.1a	0.67	29.12	1.4	0.27	2.85	2.85	55.2
PAR.2a	0.67	29.12	1.4	0.27	2.85	2.85	55.2
PAR.3	0.67	29.12	1.4	0.27	2.85	2.85	55.2
PAR.4	0.67	29.12	1.4	0.27	2.85	2.85	55.2

Fonte: Autor próprio, 2024.

Figura 5.4 – Detalhamento da parede gerado pelo TQS.



Fonte: Autor próprio, 2024

6 CONCLUSÃO E SUGESTÕES FUTURAS

6.1 Conclusão

Após os resultados foi encontrada uma diferença entre a parede dimensionada pelo programa desenvolvido neste trabalho, seguindo o roteiro manual simplificado, e a gerada automaticamente pelo módulo de reservatórios do TQS, devido a uma consideração de armadura mínima negativa, onde, sendo esta considerada para os cálculos manuais, os resultados atingem valores muito próximos.

Uma limitação do TQS é a ausência de uma opção para preencher o comprimento dos apoios, o que torna pouco claro como essa consideração é realizada. Essa questão impacta diretamente o cálculo do esmagamento do concreto na região de apoio e a verificação da ancoragem da armadura principal. No cálculo do vão, o TQS considera apenas a largura padrão de 19 cm para as paredes, resultando em um vão de 321 cm, conforme descrito no relatório.

Outro ponto relevante é o método adotado pelo TQS para calcular a armadura mínima positiva, baseado no percentual mínimo da seção de uma viga comum. Embora esse valor seja próximo ao obtido pelo programa criado, o TQS também aplica esse cálculo às barras negativas, onde foi encontrada a maior diferença entre os métodos, no TQS é considerada uma armadura mínima de 0,15% da seção transversal. No entanto, a revisão bibliográfica realizada demonstrou que não há necessidade de armadura negativa mínima nas condições avaliadas, porém na prática, é recomendado o uso da armadura mínima para proteção do elemento e proteção contra efeitos em elementos do concreto armado, como a retração do concreto.

Também foi observada uma diferença insignificante entre os valores de força de suspensão e da área de armadura de suspensão (Asusp) apresentados no relatório do TQS e os resultados obtidos no exemplo estudado neste trabalho, onde o método com os cálculos manuais foi levemente superior.

De forma geral, o dimensionamento obtido com ambos os métodos é compatível, com exceção da armadura negativa, que no detalhamento do TQS apresenta-se excessiva se comparada com as recomendações das bibliografias.

Uma vantagem importante do TQS é ser um software completo, capaz de realizar a análise de todo o corpo do reservatório, e não apenas da parede isolada. Contudo, é necessário cautela na análise e interpretação de seus resultados.

O programa desenvolvido neste trabalho apresenta algumas vantagens, como a possibilidade de selecionar livremente a quantidade e os diâmetros das barras de armadura, sem

a necessidade de editar manualmente o desenho DWG. Também evita erros frequentes no uso do TQS, que ocorreram em algumas etapas durante o estudo de caso. Além disso, a programação permite a personalização do detalhamento, alinhando-se aos padrões desejados e oferecendo maior produtividade no processo.

Portanto, os resultados apresentados reforçam a hipótese de que ferramentas personalizadas, desenvolvidas para atender às demandas de um projeto específico, podem auxiliar muito bem o uso de *softwares* comerciais em cenários onde a flexibilidade, o controle e a adaptabilidade são fatores determinantes. Ao combinar precisão técnica com a personalização do processo de detalhamento, o programa desenvolvido não apenas atende aos objetivos propostos, mas também se mostra como uma alternativa prática e eficiente para o dimensionamento de paredes de reservatórios em projetos de concreto armado.

6.2 Sugestões futuras

Segue algumas sugestões para desenvolvimentos futuros baseados nos programas criados:

- Inclusão de mais opções de cálculo, como viga-parede de 2 ou mais vãos, reservatórios enterrados e piscinas.

- Detalhamento de desenho mais completo como por exemplo o detalhamento da vista da parede.

- Desenvolver uma interface gráfica para facilitar a utilização e visualização dos programas.

- Após as melhorias realizadas, juntar e desenvolver apenas um único programa completo para o dimensionamento e detalhamento de paredes de reservatórios mais abrangente.

REFERÊNCIAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 6118:** projeto de estruturas de concreto. 4. ed. Rio de Janeiro, RJ: Ibracon, 2023.

TQS INFORMÁTICA LTDA. **TQSdocs**: Tratamento de Viga-parede. São Paulo, SP: TQS Informática LTDA, 2022. Disponível em: https://docs.tqs.com.br/Docs/PrintAllReport/176950357?language=pt-BR

CHAER, A.V. **Vigas-parede de concreto armado**. Goiânia, Go: Pontifícia Universidade Católica de Goiás, 2004. Disponível em: https://professor.pucgoias.edu.br/SiteDocente/admin/arquivosUpload/3922/material/Chaer_vigas-parede.pdf

ARAÚJO, J. M. DE. **Curso De Concreto Armado**. 5. ed. Rio Grande, RS: Editora Dunas, 2023.

TQS INFORMÁTICA LTDA. **Manual de programação Python**. São Paulo, SP: TQS Informática LTDA, 2022.

LUTZ, M.; ASCHER, D. **Aprendendo Python**. 2. ed. [s.l.] Bookman, 2022.

MATTHES, E. **Curso intensivo de Python: uma introdução prática e baseada em projetos à programação**. [s.l.] Novatec Editora, 2022.

PORTELA, E. D. L. **Projeto De Reservatórios E Piscinas De Concreto Armado: Da Concepção a Fundação**. 1. ed. TIANGUÁ, CE: Enson Portela, 2021.

SANTOS, L. E. B. DOS. **Python para desenvolvedores**. [s.l.] Luiz Eduardo Borges, 2010.

SWEIGART, A. **Automatize tarefas maçantes com Python**. San Francisco: No Starch Press, 2015.

APÊNDICE A

Manual de Instruções para o Programa de Dimensionamento de Parede Biapoiada

Objetivo do Programa

Este programa calcula e dimensiona as armaduras necessárias para uma parede de reservatório biapoiada com base em parâmetros de entrada fornecidos pelo usuário. Ele realiza verificações de segurança estrutural e gera sugestões para otimização do projeto.

Requisitos do Sistema

Sistema Operacional: Windows 10 ou superior.

Nenhuma instalação extra é necessária.

1) Instruções para Uso:

Para executar o Programa basta clicar 2x ou teclar *enter* no executável 'Dim_Parede_Res_Biapoiada.exe'.

Cuidado com antivírus e proteções que podem excluir o arquivo acusando um falso positivo.

O *Windows* normalmente vai perguntar se você realmente deseja executar o arquivo, clique em executar mesmo assim.

O *SmartScreen* do *Windows* pode interromper a execução do arquivo, caso aconteça, clique em 'mais informações' e clique em executar.

2) Entrada de Dados:

O programa solicita informações sequencialmente. Preencha os valores conforme exibidos pelo programa.

3) Cálculos Realizados

O programa irá realizar os cálculos como os esforços na parede, as armaduras necessárias e as verificações de segurança.

4) **Interações do Usuário**

Escolher bitolas e espaçamentos para as armaduras a partir das opções sugeridas pelo programa.

5) **Saída dos Resultados**

O programa apresenta os resultados diretamente no terminal e salva um resumo no arquivo especificado pelo usuário no início da execução.

6) **Finalização**

O programa aguarda o usuário pressionar *Enter* para encerrar, garantindo que todos os resultados sejam visualizados.

APÊNDICE B

Manual de Instruções para o Programa de Detalhamento de Parede Biapoiada

Objetivo do Programa

Este programa realiza o detalhamento de uma parede de reservatório com base em parâmetros de entrada fornecidos pelo usuário. Ele cria um desenho DWG compatível com o programa TQS e serve como um programa auxiliar.

Requisitos do Sistema

Sistema Operacional: Windows 10 ou superior.

Exige a instalação do programa TQS.

1) Instruções para Uso:

Para executar o Programa basta clicar 2x ou teclar *enter* no executável 'Desenho_Secao_Par.exe'.

Cuidado com antivírus e proteções que podem excluir o arquivo acusando um falso positivo.

O *Windows* normalmente vai perguntar se você realmente deseja executar o arquivo, clique em executar mesmo assim.

O *SmartScreen* do *Windows* pode interromper a execução do arquivo, caso aconteça, clique em 'mais informações' e clique em executar.

2) Entrada de Dados:

O programa solicita informações sequencialmente. Preencha os valores conforme exibidos pelo programa.

3) Interações do Usuário

Além dos dados da parede, escolher opções para formatos e armaduras opcionais.

O usuário deverá também selecionar uma pasta de destino do arquivo DWG que será criado.

4) Finalização

Após todos os dados forem preenchidos e a pasta selecionada, o programa irá fechar a janela do terminal e abrirá o DWG TQS para visualizar o desenho e realizar possíveis edições caso deseje.

ANEXO A

Código do programa para dimensionamento de uma parede de reservatório biapoiada em Python:

```
# Inicio do programa para dimensionamento de parede de reservatório biapoiada

# Importando as bibliotecas Python

import os
import math
import tkinter as tk
import sys
from tkinter import filedialog
from os import system

# Limpa a janela do terminal
system('cls')

# Dados de entrada
nometxt = ""                # Nome do arquivo gerado .txt
titulo = ""                 # Titulo da parede
largura = 0                 # Largura da parede
altura = 0                  # Altura da parede
vao = 0                     # Vão para o calculo do comprimento das armações horizontais
cobr = 0                    # Cobrimento da armação
hli = 0                     # altura da laje inferior cm
hls = 0                     # altura da laje superior cm
cae = 0                     # comprimento do apoio esquerdo cm
cad = 0                     # comprimento do apoio direito cm
pk1 = 0                     # carga superior na parede tf/m
pk2 = 0                     # carga inferior na parede tf/m
pp = 2.5                    # peso proprio do concreto armado
fck = 25                    # resistencia a compressão do concreto em MPa
fyk = 500                   # resistencia do aço em MPa

print("PARA SEPARADOR DECIMAL DE NUMEROS UTILIZAR PONTO '.'\n")
print("Entrada de dados:\n")

# Obter dados dentro de um while para evitar que o programa se encerre caso o dado preenchido não seja do tipo correto
def obter_nometxt():
    while True:
        try:
            return str(input("Digite o nome do arquivo: "))
        except ValueError:
            print("Dado incorreto. Tente novamente.")

def obter_titulo():
    while True:
        try:
            return str(input("Digite o título da parede: "))
        except ValueError:
            print("Dado incorreto. Tente novamente.")

def obter_largura():
```

```
    while True:
        try:
            return float(input("Digite a largura (cm): "))
        except ValueError:
            print("Digite um numero. Tente novamente.")

def obter_altura():
    while True:
        try:
            return float(input("Digite a altura (cm): "))
        except ValueError:
            print("Digite um numero. Tente novamente.")

def obter_vao():
    while True:
        try:
            return float(input("Digite o vão inteiro da parede (cm): "))
        except ValueError:
            print("Digite um numero. Tente novamente.")

def obter_cobr():
    while True:
        try:
            return float(input("Digite o cobrimento da armação (cm): "))
        except ValueError:
            print("Digite um numero. Tente novamente.")

def obter_dados():
    nometxt = obter_nometxt()
    titulo = obter_titulo()
    largura = obter_largura()
    altura = obter_altura()
    vao = obter_vao()
    cobr = obter_cobr()
    return nometxt, titulo, largura, altura, vao, cobr

# Uso
nometxt, titulo, largura, altura, vao, cobr = obter_dados()

def obter_hli():
    while True:
        try:
            return float(input("Digite a altura da laje inferior (cm): "))
        except ValueError:
            print("Digite um numero. Tente novamente.")
hli = obter_hli()

def obter_hls():
    while True:
        try:
            return float(input("Digite a altura da laje superior (cm): "))
        except ValueError:
            print("Digite um numero. Tente novamente.")
hls = obter_hls()

def obter_cae():
    while True:
        try:
            return float(input("Digite o comprimento do apoio esquerdo (cm): "))
        except ValueError:
```

```
            print("Digite um numero. Tente novamente.")
cae = obter_cae()

def obter_cad():
    while True:
        try:
            return float(input("Digite o comprimento do apoio direito (cm): "))
        except ValueError:
            print("Digite um numero. Tente novamente.")
cad = obter_cad()

def obter_pk1():
    while True:
        try:
            return float(input("Digite o valor da carga superior na parede (tf/m): "))
        except ValueError:
            print("Digite um numero. Tente novamente.")
pk1 = obter_pk1()

def obter_pk2():
    while True:
        try:
            return float(input("Digite o valor da carga inferior na parede (tf/m): "))
        except ValueError:
            print("Digite um numero. Tente novamente.")
pk2 = obter_pk2()

while True:
    try:
        fck = float(input("Digite a resistencia a compressão do concreto fck (MPa): "))
        if fck == 25:
            print(f"Valor aceito: {fck}")
            break
        if fck == 30:
            print(f"Valor aceito: {fck}")
            break
        if fck == 35:
            print(f"Valor aceito: {fck}")
            break
        if fck == 40:
            print(f"Valor aceito: {fck}")
            break
        if fck == 45:
            print(f"Valor aceito: {fck}")
            break
        if fck == 50:
            print(f"Valor aceito: {fck}")
            break
        else:
            print("Valor incorreto. Tente novamente.")
    except ValueError:
        print("Entrada inválida. Por favor, digite um número aceitável.")

print(f"\nTitulo da parede: {titulo}")

# peso proprio da parede

print()
print("Calculo dos esforços na parede:")
```

```
pk3 = pp*largura/100*altura/100                # calculo do peso próprio da parede
print()
print(f"Peso próprio da parede {pk3} tf/m")

# Calculo dos esforços para viga biapoiada

pk = pk1 + pk2 + pk3                # carga total em tf/m
l = (vao - cae/2 - cad/2)/100       # vão de calculo de eixo a eixo de apoio em m
Mk = pk*l**2/8                      # momento maximo em tfm
Rk = pk*l/2                         # reação de apoio em tf

print()
print(f"Carga da parede {pk:.2f} tf/m")
print()
print(f"Vão de eixo a eixo {l:.2f} m")
print()
print(f"Momento máximo {Mk:.2f} tfm")
print()
print(f"Reação nos apoios {Rk:.2f} tf")
print()

# relação vao/altura para viga-parede biapoiada

if vao/altura <= 2:
   print("Calcular como viga-parede")
else:
   print("Não é viga-parede, dimensionar como viga")
   sys.exit()

# coeficientes para concreto e aço

gamaf = 1.4        # coeficiente de majoração de esforços
gamac = 1.4        # coeficiente de minoração do concreto
gamas = 1.15       # coeficiente de minoração do aço

Md = gamaf*Mk      # momento de calculo
print()
print(f"Momento de calculo {Md:.2f} tfm")

# braço de alavanca Z para viga-parede biapoiada

alturam = altura/100    # converter altura de cm para m

if 1 < l/alturam <= 2:
   z = 0.15*alturam*(3+l/alturam)

if l/alturam <= 1:
   z = 0.6*vao/100

print()
print(f"Braço de alavanca Z = {z:.2f} m")

# Armadura do banzo tracionado

fyd = fyk/gamas             # resistencia de calculo do aço
As = Md/(z*(fyd/100))       # area de aço do banzo tracionado

print()
print(f"As calculado = {As:.2f} cm²")
```

```
# Armadura mínima

# interpolar lambda

def interpolar(x, pontos):
    """
    Interpola o valor de y para um dado x baseado nos pontos fornecidos.

    :param x: Valor de x para o qual se deseja interpolar.
    :param pontos: Lista de pontos [(x1, y1), (x2, y2), ...], ordenados por x.
    :return: Valor interpolado de y.
    """
    pontos = sorted(pontos)  # Garantir que os pontos estão ordenados
    for i in range(len(pontos) - 1):
        x1, y1 = pontos[i]
        x2, y2 = pontos[i + 1]
        if x1 <= x <= x2:  # Verificar intervalo
            # Interpolação linear
            return y1 + (x - x1) * (y2 - y1) / (x2 - x1)
    raise ValueError("Valor fora do intervalo dos pontos fornecidos.")

# Pontos conhecidos
pontos = [(2.0, 1.00), (1.5, 0.90), (1.0, 0.55)]

# Valor de l/h para interpolar
lh_valor = l/alturam

# Resultado
lambda_interpolado = interpolar(lh_valor, pontos)
print()
print(f"O valor de λ para l/h → l/h = {lh_valor} λ = {lambda_interpolado:.2f}")

# ro mínimo em relação ao fck

if fck <= 30:
    romin = 0.15/100
if fck == 35:
    romin = 0.164/100
if fck == 40:
    romin = 0.179/100
if fck == 45:
    romin = 0.194/100
if fck == 50:
    romin = 0.208/100

AsminVP = lambda_interpolado*largura*altura*romin

print()
print(f"Armadura mínima = {AsminVP:.2f} cm²")

Ast = max(As,AsminVP)

print()
print(f"Armadura para o banzo tracionado = {Ast:.2f} cm²")

# Escolha da armadura para o banzo tracionado

# Lista de diâmetros em centímetros
diametros = [0.63, 0.8, 1, 1.25, 1.6, 2, 2.5]
```

```
# Calculando as áreas e armazenando como lista de dicionários
bite = [{"diametro": d, "area": math.pi * (d / 2) ** 2} for d in diametros]

# Exibindo as opções de alojamento
print("\nOpções de alojamento:\n")
for i, bitola in enumerate(bite, 1):
    bitpos = bitola["diametro"] * 10  # Convertendo para milímetros
    # Calculando a quantidade de barras necessárias
    quantidade_barras = math.ceil(Ast / bitola["area"])  # Número de barras arredondado para cima
    # Garantindo que a quantidade de barras seja par
    if quantidade_barras % 2 != 0:
        quantidade_barras += 1
    print(f"{i}. {quantidade_barras} Ø {bitpos}")

# Pedindo para o usuário escolher uma opção
escolha = int(input("\nEscolha o número da bitola desejada: "))

# Verificando se a escolha é válida
if 1 <= escolha <= len(bite):
    bitola_escolhida = bite[escolha - 1]  # Obtém a bitola escolhida
    qpos = Ast / bitola_escolhida["area"]  # Calcula o valor inicial
    qpos = math.ceil(qpos)  # Arredonda para cima
    # Ajusta para o próximo número par, se necessário
    if qpos % 2 != 0:
        qpos += 1
    # Exibe o resultado para a bitola escolhida
    bitpos = bitola_escolhida["diametro"] * 10  # Convertendo para milímetros
    quantidade_barras = math.ceil(Ast / bitola_escolhida["area"])  # Quantidade de barras para a bitola escolhida
    # Garantindo que a quantidade de barras seja par
    if quantidade_barras % 2 != 0:
        quantidade_barras += 1
    print(f"\nArmadura positiva: {qpos} Ø {bitpos}")
else:
    print("\nEscolha inválida! Tente novamente.")

Asbe = math.pi * ((bitpos/2) / 10) ** 2
Aste = qpos * Asbe
print()
print(f"Area de aço total escolhida para o banzo tracionado = {Aste:.2f} cm²")

# distancia entre as barras positivas para posteriormente calcular o valor correto de d'

while True:
    try:
        distarmt = str(input("\nDistribuição da armadura positiva em camadas (C) ou 0.15h (H)?")).upper()
        if distarmt == "C":
            print(f"Valor aceito")
            break
        if distarmt == "H":
            print(f"Valor aceito")
            break
        else:
            print("Valor incorreto, digite C ou H. Tente novamente.")
    except ValueError:
        print("Entrada inválida. Por favor, digite um valor aceitável.")

if distarmt == "H":
    dat = 0.15*altura
else:
    dat = qpos/2 * (bitpos/10) + 2 * (qpos/2 - 1)
```

```
#print(f"\n{dat}")

# Tensão nos apoios

print("\nVerificação da tensão nos apoios:")
Rd = gamaf * Rk     # força de calculo da reação de apoio
print()
print(f"Força de calculo da reação de apoio = {Rd:.2f}")

tgteta = 4 * z / l
print()
print(f"tg teta = {tgteta:.2f}")

tetarad = math.atan(tgteta)
teta = math.degrees(tetarad)          # convertendo o angulo para graus
print()
print(f"Angulo teta = {teta:.2f}°")

dlinha = float(cobr + (dat/2))      # calculo para o d'
print()
print(f"d' = {dlinha} cm")

# Altura do nó de apoio

u = 2 * dlinha
print()
print(f"Altura do nó de apoio 'u' = {u} cm")

ce = cae                            # comprimento do apoio esquerdo
cd = cad                            # comprimento do apoio direito
cotgteta = 1 / tgteta                # cotangente do angulo teta
cecotgteta = ce * cotgteta
cdcotgteta = cd * cotgteta
fcd = fck / gamac                    # resistencia de calculo do concreto
senoteta = math.sin(tetarad)          # seno do angulo teta
senotetaquadrado = senoteta ** 2        # seno do angulo teta ao quadrado

Sigmad = 0.85 * fcd                  # tensão sigmad para verificação do apoio

Sigma2de = (Rd / (largura * (ce + u * cotgteta) * senotetaquadrado)) * 100        # tensão sigmad2 para
verificação do apoio esquerdo
Sigma2dd = (Rd / (largura * (cd + u * cotgteta) * senotetaquadrado)) * 100        # tensão sigmad2 para
verificação do apoio direito

print()
print(f"σd = {Sigmad:.2f} MPa")
print()
print(f"σ2d do apoio esquerdo = {Sigma2de:.2f} MPa")
print()
print(f"σ2d do apoio direito = {Sigma2dd:.2f} MPa")

# Verificação quanto ao esmagamento do concreto no apoio

fcdr = 0.6 * (1 - fck/250) * fcd    # valor resistente limite para verificação para verificação do esmagamento do
concreto

print()
print(f"Para o concreto de {fck:.0f} MPa:")
print(f"fcdr = {fcdr:.2f} MPa")
print()
```

```
print("Verificação do esmagamento do concreto na região dos apoios: ")

# condições de verificação

if u < cecotgteta:
    if Sigma2de <= fcdr:
        print()
        print("Segurança contra o esmagamento do concreto no apoio esquerdo OK!")
    else:
        print()
        print("Esmagamento do concreto no apoio esquerdo, rever distribuição da armadura positiva, fck do
concreto ou dimensão do apoio, sendo essa ultima opção apenas em ultimo caso.")

if u < cdcotgteta:
    if Sigma2dd <= fcdr:
        print()
        print("Segurança contra o esmagamento do concreto no apoio direito OK!")
    else:
        print()
        print("Esmagamento do concreto no apoio direito, rever distribuição da armadura positiva, fck do concreto
ou dimensão do apoio, sendo essa ultima opção apenas em ultimo caso.")

if u >= cecotgteta:
    if Sigmad <= fcdr:
        print()
        print("Segurança contra o esmagamento do concreto no apoio esquerdo OK!")
    else:
        print()
        print("Esmagamento do concreto no apoio esquerdo, rever distribuição da armadura positiva, fck do
concreto ou dimensão do apoio, sendo essa ultima opção apenas em ultimo caso.")

if u >= cdcotgteta:
    if Sigmad <= fcdr:
        print()
        print("Segurança contra o esmagamento do concreto no apoio direito OK!")
    else:
        print()
        print("Esmagamento do concreto no apoio direito, rever distribuição da armadura positiva, fck do concreto
ou dimensão do apoio, sendo essa ultima opção apenas em ultimo caso.")

# Comprimento basico de ancoragem
# Para garantir um lb de forma simples e segura vamos utilizar o C25 como referência,
# em região de boa aderência, já que este calculo é para a armadura positiva na região inferior da parede

Raiodobra = 5 * bitpos / 10

if bitpos == 8:
    blb = 30
    blbe = 21

if bitpos == 10:
    blb = 38
    blbe = 27

if bitpos == 12.5:
    blb = 47
    blbe = 33

if bitpos == 16:
    blb = 61
```

```
    blbe = 43

if bitpos == 20:
    blb = 76
    blbe = 53

if bitpos == 25:
    blb = 95
    blbe = 66

print()
print("Comprimento básico de ancoragem necessário:")
print()
print(f"O lb necessário para {fck:.0f} MPa e para a Ø {bitpos} em região de boa aderência com gancho é igual a
{blbe:.0f} cm")

# Força a ser ancorada para a armadura tracionada

Ascal = 0.8 * As                        # Força a ser ancorada para a armadura tracionada
lbnec = 0.7 * blbe * Ascal / Aste             # calculo do lb necessário
lbmin1 = 0.3 * blbe                      # lb minimo de norma
lbmin2 = 10 * bitpos/10                     # lb minimo de norma
lbmin3 = 10                             # lb minimo de norma
lbreal = max(lbmin1, lbmin2, lbmin3, lbnec)       # pegar o maior valor para lb necessário

print()
print("Valores mínimos para o comprimento de ancoragem:")
print()
print(f"lb necessário: {lbnec:.2f} cm \nlb mínimo 1 = {lbmin1:.2f} cm \nlb mínimo 2 = {lbmin2:.2f} cm \nlb
mínimo 3 = {lbmin3:.2f} cm")
print()
print(f"Portanto o lb necessário é de = {lbreal:.2f} cm")

# verificação do lb necessário

if cae >= lbreal:
    print()
    print(f"O comprimento do apoio esquerdo de {cae} cm é maior que o lb necessário de {lbreal} cm, portanto é
possível realizar a ancoragem no apoio")
else:
    print()
    print("Não é possível realizar a ancoragem no apoio, rever a armadura positiva ou tamanho do apoio")
    sys.exit()

# Armadura de suspensão

Pds = gamac * pk2   # Força a ser suspensa
print()
print("Armadura de suspensão: ")
print()
print(f"Força a ser suspensa: {Pds:.2f} tf/m")

Asusp = (Pds/(fyd/100)) / 2     # calculo da armadura de suspensão
print()
print(f"Armadura de suspensão: {Asusp:.3f} cm²/m por face")

# Armadura de pele e estribo

print()
print("Armadura de pele e estribo: ")
```

```
Aswmin = 0.1 * largura      # armadura mínima para pele e estribo
print()
print(f"Armadura mínima para pele e estribo = {Aswmin:.2f} cm²/m por face")

# Calculo do esforço para pele e estribo

hi = altura - hli - hls                    # altura de água
gamaagua = 1                               # peso especifido da agua em tf/m³
q = hi * gamaagua / 100                    # força de empuxo da agua na parede
hil = (altura - ((hli + hls) / 2)) / 100   # altura para o calculo do esforço de momento na parede devido a agua

# condição de calculo do momento

if altura / vao < 2:
   Mpar = 0.7 * q * (hil ** 2) / 8

else:
   Mpar = 0.7 * q * (l ** 2) / 8

print()
print(f"Momento devido ao empuxo na parede = {Mpar:.2f} tfm/m")

# Calculo da armadura de flexão para estribo, pele e misula devido ao empuxo

# dados para o dimensionamento da seção da parede
b = 100                    # largura (unidade de m)
h = largura                # altura util da parede onde o empuxo é aplicado, no caso da parede, em sua largura
d = largura - 4            # altura util de calculo
dlinhae = 4                # d'
xlim = 0.45 * d            # limite para armadura dupla segundo NBR 6118
x23 = 0.259 * d            # limite para domínio 2
x34 = 0.628 * d            # limite para domínio 3
Md = Mpar * gamaf * 1000   # Momento de calculo em kNm/m

# Calculo do valro de x - linha neutra
divisao = Md / (0.425 * b * (d**2) * (fcd/10))
raiz = (1 - divisao) ** 0.5
x = 1.25 * d * (1 - raiz)

print()
print("Calculo da seção devido ao empuxo da água na parede:")
print()
print(f"Base = {b} cm")
print(f"Altura = {h} cm")
print(f"Altura útil d = {d} cm")
print(f"d' = {dlinhae} cm")
print(f"Altura limite da linha neutra = {xlim:.2f} cm")
print(f"Altura limite para Domínio 2 = {x23:.2f} cm")
print(f"Altura limite para Domínio 3 = {x34:.2f} cm")
print(f"Momento de calculo devido ao empuxo = {Md:.2f} kNcm/m")
print()
print(f"Posição da linha neutra = {x:.2f} cm")

# verificação da linha neutra
if x <= x23:
   print("Domínio 2")
if x <= x34 and x > x23:
   print("Domínio 3")
if x <= xlim:
   print()
```

```
    print("Posição da linha neutra Ok!")
if x > xlim:
    print()
    print("Posição da linha neutra acima do limite, rever largura da parede")
    sys.exit()

# area de armadura para estribo, pele e misula
Asw = ( 1 / (fyd/10) ) * ( Md / (d - 0.4 * x))
Aswe = max(Asusp, Aswmin, Asw)

print()
print(f"Armadura calculada para pele e estribo = {Asw:.2f} cm²/m")
print()
print("Armaduras calculadas para pele e estribo: ")
print()
print(f"Armadura de suspensão: {Asusp:.3f} cm²/m")
print(f"Armadura mínima : {Aswmin:.2f} cm²/m")
print(f"Armadura calculada para pele e estribo = {Asw:.2f} cm²/m")
print()
print(f"Portanto armadura para pele e estribo → Asw = {Aswe:.2f} cm²/m")

# escolha do alojamento para estribo, pele e misula
# Lista de diâmetros em centímetros e suas áreas
bite = [
    {"diametro": 0.63, "area": 0.3117},
    {"diametro": 0.8, "area": 0.5027},
    {"diametro": 1.0, "area": 0.7854},
    {"diametro": 1.25, "area": 1.2272},
    {"diametro": 1.6, "area": 2.0106},
    {"diametro": 2.0, "area": 3.1416},
    {"diametro": 2.5, "area": 4.9087},
]

# Espaçamentos permitidos (em cm) - limitados aos espaçamentos mais usuais
espaçamento = [10, 12, 15, 20]

# Comprimento de referência (1 metro) - devido a armadura ser distribuida
comprimento = 100  # em cm

print("\nOpções de alojamento para estribo, pele e misula:")

melhores_opcoes = []

# Encontrar a melhor combinação para cada espaçamento
for esp in espaçamento:
    melhor_opcao = None  # Inicializa a melhor opção para cada espaçamento
    menor_sobra = float('inf')  # Inicializa a menor sobra como infinito

    for bitola in bite:
        # Calculando o número de barras no comprimento de 1 metro
        quantidade_barras = comprimento / esp  # Divisão inteira para número de barras

        # Calculando a área total fornecida corretamente
        Aste = (comprimento / esp) * bitola["area"]  # Área total de aço fornecida

        # Se o alojamento atender o Asw
        if Aste >= Aswe:
            sobra = abs(Aste - Aswe)  # Diferença absoluta entre a área fornecida e a necessária
            if sobra < menor_sobra:
                menor_sobra = sobra
```

```
            melhor_opcao = {
                "espaçamento": esp,
                "quantidade_barras": quantidade_barras,
                "diametro": bitola["diametro"] * 10,  # Convertendo para mm
                "Aste": round(Aste, 2),  # Arredondando a área total de aço para 2 casas decimais
            }

    # Adiciona o melhor alojamento encontrado para o espaçamento
    if melhor_opcao:
        melhores_opcoes.append(melhor_opcao)

# Ordenando as opções pela menor diferença da área total de aço em relação ao Aswe
melhores_opcoes.sort(key=lambda x: abs(x['Aste'] - Aswe))

# Exibindo apenas as opções mais próximas
print("Escolha uma das opções de alojamento mais próximas:\n")
for i, opcao in enumerate(melhores_opcoes, 1):
    print(f"{i}. Ø {opcao['diametro']} C/{opcao['espaçamento']} → Área total de aço = {opcao['Aste']} cm²")

# Pedindo para o usuário escolher uma opção
escolha = int(input("\nEscolha o número da opção desejada: "))

# Verificando se a escolha é válida
if 1 <= escolha <= len(melhores_opcoes):
    opcao_escolhida = melhores_opcoes[escolha - 1]  # Obtém a opção escolhida
    print(f"\nArmadura de pele e estribo = Ø {opcao_escolhida['diametro']}
C/{opcao_escolhida['espaçamento']}")
else:
    print("\nEscolha inválida! Tente novamente.")

# mostra resumo do dimensionamento das armaduras
# ANSI escape code para negrito
NEGRITO = '\033[1m'
RESET = '\033[0m'
print()
print("Resumo do dimensionamento para a parede:")
print(f"{NEGRITO}\nArmadura positiva →  {qpos} Ø {bitpos}")
print(f"Armadura de pele, estribo e misula → Ø {opcao_escolhida['diametro']}
C/{opcao_escolhida['espaçamento']}")
print(f"Para armadura negativa de viga-parede, pode-se repetir a armadura de pele → 2 Ø
{opcao_escolhida['diametro']}")

# interação para manter janela com informações abertas até o usuario decidir fechar o programa
input("\nPressione Enter para sair...")
```

ANEXO B

Código do programa para detalhamento de uma parede de reservatório em DWG TQS utilizando a API - Python x TQS:

```
# Inicio do programa para detalhamento de parede de reservatório no TQS

# Importando as bibliotecas Python

import os
import math
import tkinter as tk
from tkinter import filedialog
from TQS import TQSDwg, TQSGeo

# Limpa a janela do terminal
from os import system
system('cls')

# Inicializar o desenho
dwg = TQSDwg.Dwg()
dwg.settings.systemId    = TQSDwg.IEDTAB      # AGC & DP
dwg.settings.subSystemId = 13                 # Desenho de armação
dwg.file.LoadColors()   # carrega as cores

# entrada de dados
# variaveis
nomedwg = ""                  # Nome do arquivo gerado .dwg
titulo = ""                  # Titulo da parede no desenho
largura = 0                   # Largura da parede
altura = 0                   # Altura da parede
vao = 0                      # Vão para o calculo do comprimento das armações horizontais
cobr = 0                     # Cobrimento da armação
agua = "N"                     # Indicação de agua
lie = "N"                    # Laje inferior a esquerda?
hlie = 0                     # Altura da laje inferior esquerda
deslie = 0                    # Desnivel do fundo da parede ao fundo da laje esquerda
mislie = "N"                    # Misula na laje inferior esquerda
hmislie = 0                   # Altura da misula inferior esquerda
bitmislie = 0                 # Bitola da misula inferior esquerda
espmislie = 0                  # Espaçamento da misula infeior esquerda
aeve = "N"                    # Armadura contra empuxo ao vazio na laje inferior esquerda
bitaeve = 0                   # Bitola da armadura contra empuxo ao vazio na laje inferior esquerda
espaeve = 0                    # Espaçamento da armadura contra empuxo ao vazio na laje inferior esquerda
aevep = 0                     # Tamanho da 'perna' da armadura contra empuxo ao vazio na laje inferior esquerda
lse = "N"                    # Laje superior esquerda
hlse = 0                     # Altura da laje superior esquerda
deslse = 0                    # Desnivel do topo da parede ao topo da laje superior esquerda
lid = "N"                    # Laje inferior a direita?
hlid = 0                     # Altura da laje inferior direita
deslid = 0                    # Desnivel do fundo da parede ao fundo da laje direita
mislid = "N"                    # Misula na laje inferior direita
hmislid = 0                   # Altura da misula inferior direita
bitmislid = 0                 # Bitola da misula inferior direita
espmislid = 0                  # Espaçamento da misula infeior direita
aevd = "N"                    # Armadura contra empuxo ao vazio na laje inferior direita
```

```
bitaevd = 0                 # Bitola da armadura contra empuxo ao vazio na laje inferior direita
espaevd = 0                 # Espaçamento da armadura contra empuxo ao vazio na laje inferior direita
aevdp = 0                   # Tamanho da 'perna' da armadura contra empuxo ao vazio na laje inferior direita
lsd = "N"                   # Laje superior direita
hlsd = 0                    # Altura da laje superior direita
deslsd = 0                  # Desnivel do topo da parede ao topo da laje superior direita
qpos = 0                    # Quantidade de barras positivas
bitpos = 0                  # Bitola armação positiva
qneg = 0                    # Quantidade de barras negativas
bitneg = 0                  # Bitola armação negativa
espele = 0                  # Espaçamento da armadura de pele
bitpele = 0                 # Bitola da Armadura de pele
vest = 0                    # Vão para o calculo da quantidade dos estribos e misulas
espest = 0                  # Espaçamento dos estribos
bitest = 0                  # Bitola dos estribos
tipoestribo = 0             # Tipo de estribo, aberto ou fechado

print("PARA SEPARADOR DECIMAL DE NUMEROS UTILIZAR PONTO '.'")
# Obter dados dentro de um while para evitar que o programa se encerre caso o dado preenchido não seja do tipo correto
def obter_nomedwg():
    while True:
        try:
            return str(input("Digite o nome do arquivo: "))
        except ValueError:
            print("Dado incorreto. Tente novamente.")

def obter_titulo():
    while True:
        try:
            return str(input("Digite o título do desenho: "))
        except ValueError:
            print("Dado incorreto. Tente novamente.")

def obter_largura():
    while True:
        try:
            return float(input("Digite a largura (cm): "))
        except ValueError:
            print("Digite um numero. Tente novamente.")

def obter_altura():
    while True:
        try:
            return float(input("Digite a altura (cm): "))
        except ValueError:
            print("Digite um numero. Tente novamente.")

def obter_vao():
    while True:
        try:
            return float(input("Digite o vão inteiro da parede (cm): "))
        except ValueError:
            print("Digite um numero. Tente novamente.")

def obter_cobr():
    while True:
        try:
            return float(input("Digite o cobrimento da armação (cm): "))
        except ValueError:
```

```
            print("Digite um numero. Tente novamente.")

def obter_dados():
    nomedwg = obter_nomedwg()
    titulo = obter_titulo()
    largura = obter_largura()
    altura = obter_altura()
    vao = obter_vao()
    cobr = obter_cobr()
    return nomedwg, titulo, largura, altura, vao, cobr

# Uso
nomedwg, titulo, largura, altura, vao, cobr = obter_dados()

while True:
    try:
        agua = str(input("Indicar agua à (E) Esquerda, (D) Direita, (A) Ambos ou (N) Não indicar: ")).upper()
        if agua == "E":
            print(f"Indicar agua a esquerda")
            break
        if agua == "D":
            print(f"Indicar agua a direita")
            break
        if agua == "A":
            print(f"Indicar agua em ambos os lados")
            break
        if agua == "N":
            print(f"Sem indicação de agua")
            break
        else:
            print("Valor incorreto. Tente novamente.")
    except ValueError:
        print("Dado incorreto.E, D, A ou N. Tente novamente.")

# Indicação de agua
xae1 = 0-90-50              # Pontos para desenho da representação
yae1 = altura+70
xae2 = xae1+50
yae2 = yae1
xae3 = xae1+25
yae3 = yae1-15

xad1 = 0+largura*2+90
yad1 = yae1
xad2 = xad1+50
yad2 = yad1
xad3 = xad1+25
yad3 = yae3

if agua == "E":
    dwg.draw.level = 0
    dwg.draw.Line (xae1,yae1,xae2,yae2)
    dwg.draw.Line (xae1,yae1,xae3,yae3)
    dwg.draw.Line (xae3,yae3,xae2,yae2)

if agua == "D":
    dwg.draw.level = 0
    dwg.draw.Line (xad1,yad1,xad2,yad2)
    dwg.draw.Line (xad1,yad1,xad3,yad3)
    dwg.draw.Line (xad3,yad3,xad2,yad2)
```

```
if agua == "A":
    dwg.draw.level = 0
    dwg.draw.Line (xae1,yae1,xae2,yae2)
    dwg.draw.Line (xae1,yae1,xae3,yae3)
    dwg.draw.Line (xae3,yae3,xae2,yae2)
    dwg.draw.Line (xad1,yad1,xad2,yad2)
    dwg.draw.Line (xad1,yad1,xad3,yad3)
    dwg.draw.Line (xad3,yad3,xad2,yad2)

# pontos da seção sem laje
x1s = 0
y1s = 0
x2s = largura*2
y2s = 0
x3s = largura*2
y3s = altura*2
x4s = 0
y4s = altura*2

while True:
    try:
        lie = str(input("Laje inferior a esquerda? (S)-SIM (N)-NÃO ")).upper()
        if lie == "S":
            print(f"Valor aceito")
            break
        if lie == "N":
            print(f"Valor aceito")
            break
        else:
            print("Valor incorreto, digite S ou N. Tente novamente.")
    except ValueError:
        print("Entrada inválida. Por favor, digite um valor aceitável.")
if lie == "S":
    def obter_hlie():
        while True:
            try:
                return float(input("Digite a altura da laje inferior esquerda (cm): "))
            except ValueError:
                print("Digite um numero. Tente novamente.")

    def obter_deslie():
        while True:
            try:
                return float(input("Digite o desnível em relação ao fundo da parede (cm): "))
            except ValueError:
                print("Digite um numero. Tente novamente.")

    def hlie_deslie():
        hlie = obter_hlie()
        deslie = obter_deslie()
        return hlie, deslie

    # Uso
    hlie, deslie = hlie_deslie()

    # pontos para seção
    xlie1 = 0
    ylie1 = 0+deslie*2
    xlie2 = 0-320
```

```
ylie2 = ylie1
xlie3 = 0
ylie3 = ylie1+hlie*2
xlie4 = xlie2
ylie4 = ylie3
mislie = "N"
while True:
    try:
        mislie = str(input("Misula na laje inferior a esquerda? (S)-SIM (N)-NÃO ")).upper()
        if mislie == "S":
            print(f"Valor aceito")
            break
        if mislie == "N":
            print(f"Valor aceito")
            break
        else:
            print("Valor incorreto, digite S ou N. Tente novamente.")
    except ValueError:
        print("Entrada inválida. Por favor, digite um valor aceitável.")
if mislie == "S":
    def obter_hmislie():
        while True:
            try:
                return float(input("Digite o tamanho da mísula à esquerda (cm): "))
            except ValueError:
                print("Digite um numero. Tente novamente.")

    def obter_bitmislie():
        while True:
            try:
                return float(input("Digite a bitola da armadura da mísula (mm): "))
            except ValueError:
                print("Digite um numero. Tente novamente.")

    def obter_espmislie():
        while True:
            try:
                return float(input("Digite o espaçamento da armadura da mísula (cm): "))
            except ValueError:
                print("Digite um numero. Tente novamente.")

    def hmislie_bitmislie_espmislie():
        hmislie = obter_hmislie()
        bitmislie = obter_bitmislie()
        espmislie = obter_espmislie()
        return hmislie, bitmislie, espmislie

    # Uso
    hmislie, bitmislie, espmislie = hmislie_bitmislie_espmislie()

    # pontos para seção
    xme1 = 0-hmislie*2
    yme1 = ylie3
    xme2 = 0
    yme2 = ylie3+hmislie*2
while True:
    try:
        aeve = str(input("Armadura contra empuxo ao vazio a esquerda? (S)-SIM (N)-NÃO ")).upper()
        if aeve == "S":
            print(f"Valor aceito")
```

```
            break
        if aeve == "N":
            print(f"Valor aceito")
            break
        else:
            print("Valor incorreto, digite S ou N. Tente novamente.")
    except ValueError:
        print("Entrada inválida. Por favor, digite um valor aceitável.")
if aeve == "S":
    def obter_bitaeve():
        while True:
            try:
                return float(input("Digite a bitola da armadura contra empuxo ao vazio (mm): "))
            except ValueError:
                print("Digite um numero. Tente novamente.")

    def obter_espaeve():
        while True:
            try:
                return float(input("Digite o espaçamento da armadura contra empuxo ao vazio (cm): "))
            except ValueError:
                print("Digite um numero. Tente novamente.")

    def obter_aevep():
        while True:
            try:
                return float(input("Digite o tamanho da 'perna' da armadura (cm): "))
            except ValueError:
                print("Digite um numero. Tente novamente.")

    def bitaeve_espaeve_aevep():
        bitaeve = obter_bitaeve()
        espaeve = obter_espaeve()
        aevep = obter_aevep()
        return bitaeve, espaeve, aevep

    # Uso
    bitaeve, espaeve, aevep = bitaeve_espaeve_aevep()

while True:
    try:
        lse = str(input("Laje superior a esquerda? (S)-SIM (N)-NÃO ")).upper()
        if lse == "S":
            print(f"Valor aceito")
            break
        if lse == "N":
            print(f"Valor aceito")
            break
        else:
            print("Valor incorreto, digite S ou N. Tente novamente.")
    except ValueError:
        print("Entrada inválida. Por favor, digite um valor aceitável.")
if lse == "S":
    def obter_hlse():
        while True:
            try:
                return float(input("Digite a altura da laje superior esquerda (cm): "))
            except ValueError:
                print("Digite um numero. Tente novamente.")
```

```
    def obter_deslse():
        while True:
            try:
                return float(input("Digite o desnível em relação ao topo da parede (cm): "))
            except ValueError:
                print("Digite um numero. Tente novamente.")

    def hlse_deslse():
        hlse = obter_hlse()
        deslse = obter_deslse()
        return hlse, deslse

    # Uso
    hlse, deslse = hlse_deslse()

    # pontos para seção
    xlse1 = 0
    ylse1 = altura*2-deslse*2
    xlse2 = 0-100
    ylse2 = ylse1
    xlse3 = 0
    ylse3 = ylse1-hlse*2
    xlse4 = 0-100
    ylse4 = ylse3

while True:
    try:
        lid = str(input("Laje inferior a direita? (S)-SIM (N)-NÃO ")).upper()
        if lid == "S":
            print(f"Valor aceito")
            break
        if lid == "N":
            print(f"Valor aceito")
            break
        else:
            print("Valor incorreto, digite S ou N. Tente novamente.")
    except ValueError:
        print("Entrada inválida. Por favor, digite um número aceitável.")
if lid == "S":
    def obter_hlid():
        while True:
            try:
                return float(input("Digite a altura da laje inferior direita (cm): "))
            except ValueError:
                print("Digite um numero. Tente novamente.")

    def obter_deslid():
        while True:
            try:
                return float(input("Digite o desnível em relação ao fundo da parede (cm): "))
            except ValueError:
                print("Digite um numero. Tente novamente.")

    def hlid_deslid():
        hlid = obter_hlid()
        deslid = obter_deslid()
        return hlid, deslid

    # Uso
    hlid, deslid = hlid_deslid()
```

```
# pontos para seção
xlid1 = 0+largura*2
ylid1 = 0+deslid*2
xlid2 = xlid1+320
ylid2 = ylid1
xlid3 = xlid1
ylid3 = ylid1+hlid*2
xlid4 = xlid2
ylid4 = ylid3
mislid = "N"
while True:
    try:
        mislid = str(input("Misula na laje inferior a direita? (S)-SIM (N)-NÃO ")).upper()
        if mislid == "S":
            print(f"Valor aceito")
            break
        if mislid == "N":
            print(f"Valor aceito")
            break
        else:
            print("Valor incorreto, digite S ou N. Tente novamente.")
    except ValueError:
        print("Entrada inválida. Por favor, digite um número aceitável.")
if mislid == "S":
    def obter_hmislid():
        while True:
            try:
                return float(input("Digite o tamanho da mísula à direita (cm): "))
            except ValueError:
                print("Digite um numero. Tente novamente.")

    def obter_bitmislid():
        while True:
            try:
                return float(input("Digite a bitola da armadura da mísula (mm): "))
            except ValueError:
                print("Digite um numero. Tente novamente.")

    def obter_espmislid():
        while True:
            try:
                return float(input("Digite o espaçamento da armadura da mísula (cm): "))
            except ValueError:
                print("Digite um numero. Tente novamente.")

    def hmislid_bitmislid_espmislid():
        hmislid = obter_hmislid()
        bitmislid = obter_bitmislid()
        espmislid = obter_espmislid()
        return hmislid, bitmislid, espmislid

    # Uso
    hmislid, bitmislid, espmislid = hmislid_bitmislid_espmislid()

    # pontos para seção
    xmd1 = xlid1+hmislid*2
    ymd1 = ylid3
    xmd2 = xlid1
    ymd2 = ylid3+hmislid*2
```

```
    while True:
        try:
            aevd = str(input("Armadura contra empuxo ao vazio a direita? (S)-SIM (N)-NÃO ")).upper()
            if aevd == "S":
                print(f"Valor aceito")
                break
            if aevd == "N":
                print(f"Valor aceito")
                break
            else:
                print("Valor incorreto, digite S ou N. Tente novamente.")
        except ValueError:
            print("Entrada inválida. Por favor, digite um número aceitável.")
    if aevd == "S":
        def obter_bitaevd():
            while True:
                try:
                    return float(input("Digite a bitola da armadura contra empuxo ao vazio (mm): "))
                except ValueError:
                    print("Digite um numero. Tente novamente.")

        def obter_espaevd():
            while True:
                try:
                    return float(input("Digite o espaçamento da armadura contra empuxo ao vazio (cm): "))
                except ValueError:
                    print("Digite um numero. Tente novamente.")

        def obter_aevdp():
            while True:
                try:
                    return float(input("Digite o tamanho da 'perna' da armadura (cm): "))
                except ValueError:
                    print("Digite um numero. Tente novamente.")

        def bitaevd_espaevd_aevdp():
            bitaevd = obter_bitaevd()
            espaevd = obter_espaevd()
            aevdp = obter_aevdp()
            return bitaevd, espaevd, aevdp

        # Uso
        bitaevd, espaevd, aevdp = bitaevd_espaevd_aevdp()

while True:
    try:
        lsd = str(input("Laje superior a direita? (S)-SIM (N)-NÃO ")).upper()
        if lsd == "S":
            print(f"Valor aceito")
            break
        if lsd == "N":
            print(f"Valor aceito")
            break
        else:
            print("Valor incorreto, digite S ou N. Tente novamente.")
    except ValueError:
        print("Entrada inválida. Por favor, digite um valor aceitável.")
if lsd == "S":
    def obter_hlsd():
        while True:
```

```
            try:
                return float(input("Digite a altura da laje superior direita (cm): "))
            except ValueError:
                print("Digite um numero. Tente novamente.")

    def obter_deslsd():
        while True:
            try:
                return float(input("Digite o desnível em relação ao topo da parede (cm): "))
            except ValueError:
                print("Digite um numero. Tente novamente.")

    def hlsd_deslsd():
        hlsd = obter_hlsd()
        deslsd = obter_deslsd()
        return hlsd, deslsd

    # Uso
    hlsd, deslsd = hlsd_deslsd()


    # pontos para seção
    xlsd1 = 0+largura*2
    ylsd1 = altura*2-deslsd*2
    xlsd2 = xlsd1+100
    ylsd2 = ylsd1
    xlsd3 = xlsd1
    ylsd3 = ylsd1-hlsd*2
    xlsd4 = xlsd3+100
    ylsd4 = ylsd3

while True:
    try:
        qpos = int(input("Digite a quantidade de barras positivas: "))
        if qpos % 2 == 0:
            print(f"Valor aceito: {qpos}")
            break
        else:
            print("O valor não é par. Tente novamente.")
    except ValueError:
        print("Entrada inválida. Por favor, digite um número inteiro.")

while True:
    try:
        bitpos = float(input("Digite a bitola do ferro positivo (mm): "))
        if bitpos == 6.3:
            print(f"Valor aceito: {bitpos}")
            break
        if bitpos == 8:
            print(f"Valor aceito: {bitpos}")
            break
        if bitpos == 10:
            print(f"Valor aceito: {bitpos}")
            break
        if bitpos == 12.5:
            print(f"Valor aceito: {bitpos}")
            break
        if bitpos == 16:
            print(f"Valor aceito: {bitpos}")
            break
```

```
        if bitpos == 20:
            print(f"Valor aceito: {bitpos}")
            break
        if bitpos == 25:
            print(f"Valor aceito: {bitpos}")
            break
        else:
            print("Valor incorreto. Tente novamente.")
    except ValueError:
        print("Entrada inválida. Por favor, digite um número aceitável.")

while True:
    try:
        distpos = str(input("Distribuir as barras positivas em camadas (C) ou em 0.15h (H): ")).upper()
        if distpos == "C":
            print(f"Barras positivas distribuidas em camadas")
            break
        if distpos == "H":
            print(f"Barras positivas distribuidas em 0.15h")
            break
        else:
            print("Entrada inválida. Por favor, digite 'C' ou 'H'")
    except ValueError:
        print("Entrada inválida. Por favor, digite 'C' ou 'H'")

while True:
    try:
        qneg = int(input("Digite a quantidade de barras negativas: "))
        if qneg % 2 == 0:
            print(f"Valor aceito: {qneg}")
            break
        else:
            print("O valor não é par. Tente novamente.")
    except ValueError:
        print("Entrada inválida. Por favor, digite um número inteiro.")

while True:
    try:
        bitneg = float(input("Digite a bitola do ferro negativo (mm): "))
        if bitneg == 6.3:
            print(f"Valor aceito: {bitneg}")
            break
        if bitneg == 8:
            print(f"Valor aceito: {bitneg}")
            break
        if bitneg == 10:
            print(f"Valor aceito: {bitneg}")
            break
        if bitneg == 12.5:
            print(f"Valor aceito: {bitneg}")
            break
        if bitneg == 16:
            print(f"Valor aceito: {bitneg}")
            break
        if bitneg == 20:
            print(f"Valor aceito: {bitneg}")
            break
        if bitneg == 25:
            print(f"Valor aceito: {bitneg}")
            break
```

```
        else:
            print("Valor incorreto. Tente novamente.")
    except ValueError:
        print("Entrada inválida. Por favor, digite um número aceitável.")

def arm_pele():
    while True:
        try:
            espele = float(input("Digite o espaçamento da arm.de pele (cm): "))
            if espele <= 0:
                print("O espaçamento deve ser maior que zero")
                continue
            return espele
        except ValueError:
            print("Dado incorreto. Só pode ser números. Tente novamente.")
espele = arm_pele()

try:
    qpele = int(altura/espele)-1    # quantidade de barras de armadura de pele
except ZeroDivisionError:
    print("Erro: divisão por zero")
#print (qpele)

while True:
    try:
        bitpele = float(input("Digite a bitola do ferro de pele (mm): "))
        if bitpele == 8:
            print(f"Valor aceito: {bitpele}")
            break
        if bitpele == 10:
            print(f"Valor aceito: {bitpele}")
            break
        if bitpele == 12.5:
            print(f"Valor aceito: {bitpele}")
            break
        if bitpele == 16:
            print(f"Valor aceito: {bitpele}")
            break
        if bitpele == 20:
            print(f"Valor aceito: {bitpele}")
            break
        if bitpele == 25:
            print(f"Valor aceito: {bitpele}")
            break
        else:
            print("Valor incorreto. Tente novamente.")
    except ValueError:
        print("Entrada inválida. Por favor, digite um número aceitável.")

def obter_vest():
    while True:
        try:
            return float(input("Digite o vão para estribos (cm): "))
        except ValueError:
            print("Digite um numero. Só pode ser números. Tente novamente.")

def obter_espest():
    while True:
        try:
            return float(input("Digite o espaçamento do estribo (cm): "))
```

```
        except ValueError:
            print("Digite um numero. Só pode ser números. Tente novamente.")

def arm_est():
    vest = obter_vest()
    espest = obter_espest()
    return vest, espest

# Uso
vest, espest = arm_est()


while True:
    try:
        bitest = float(input("Digite a bitola do estribo (mm): "))
        if bitest == 5:
            print(f"Valor aceito: {bitest}")
            break
        if bitest == 6.3:
            print(f"Valor aceito: {bitest}")
            break
        if bitest == 8:
            print(f"Valor aceito: {bitest}")
            break
        if bitest == 10:
            print(f"Valor aceito: {bitest}")
            break
        if bitest == 12.5:
            print(f"Valor aceito: {bitest}")
            break
        if bitest == 16:
            print(f"Valor aceito: {bitest}")
            break
        if bitest == 20:
            print(f"Valor aceito: {bitest}")
            break
        if bitest == 25:
            print(f"Valor aceito: {bitest}")
            break
        else:
            print("Valor incorreto. Tente novamente.")
    except ValueError:
        print("Entrada inválida. Por favor, digite um número aceitável.")

while True:
    try:
        tipoestribo = int(input("Digite o tipo do estribo (0 - ABERTO) (1 - FECHADO): "))
        if tipoestribo == 0:
            print(f"Valor aceito: {bitest}")
            break
        if tipoestribo == 1:
            print(f"Valor aceito: {bitest}")
            break
        else:
            print("Valor incorreto, digite 0 ou 1. Tente novamente.")
    except ValueError:
        print("Entrada inválida. Por favor, digite um número aceitável.")

# texto do titulo
dwg.draw.level = 222                        # nivel do texto
```

```
dwg.draw.Text (0, altura*2+125, 12, 0, titulo)      # texto do titulo no desenho

# texto da seção
dwg.draw.level = 224    # nivel do texto
dwg.draw.Text (0, altura*2+100, 10, 0, str(int(largura)) + "X" + str(int(altura)))   # texto da dimensão no desenho

# linhas da seção
dwg.draw.level = 224    # nivel da linha

# desenho do retangulo da seção sem lajes
if lie == "N" and lse == "N" and lid == "N" and lsd == "N":
    dwg.draw.Line   (x1s,y1s,x2s,y2s)    # linha inferior
    dwg.draw.Line   (x2s,y2s,x3s,y3s)    # linha direita
    dwg.draw.Line   (x3s,y3s,x4s,y4s)    # linha superior
    dwg.draw.Line   (x4s,y4s,x1s,y1s)    # linha esquerda

# desenho do retangulo da seção com laje inferior esquerda sem misula
if lie == "S" and lse == "N" and lid == "N" and lsd == "N" and mislie == "N":
    dwg.draw.Line   (x1s,y1s,x2s,y2s)              # linha inferior
    dwg.draw.Line   (x2s,y2s,x3s,y3s)              # linha direita
    dwg.draw.Line   (x3s,y3s,x4s,y4s)              # linha superior
    dwg.draw.Line   (x4s,y4s,xlie3,ylie3)           # linha esquerda superior
    dwg.draw.Line   (xlie3,ylie3,xlie4,ylie4)       # linha superior laje inferior esquerda
    dwg.draw.Line   (xlie1,ylie1,xlie2,ylie2)       # linha inferior laje inferior esquerda
    dwg.draw.Line   (x1s,y1s,xlie1,ylie1)           # linha esquerda inferior

# desenho do retangulo da seção com laje inferior esquerda com misula
if lie == "S" and lse == "N" and lid == "N" and lsd == "N" and mislie == "S":
    dwg.draw.Line   (x1s,y1s,x2s,y2s)              # linha inferior
    dwg.draw.Line   (x2s,y2s,x3s,y3s)              # linha direita
    dwg.draw.Line   (x3s,y3s,x4s,y4s)              # linha superior
    dwg.draw.Line   (x4s,y4s,xme2,yme2)             # linha esquerda superior
    dwg.draw.Line   (xme2,yme2,xme1,yme1)           # linha misula
    dwg.draw.Line   (xme1,yme1,xlie4,ylie4)         # linha superior laje inferior esquerda
    dwg.draw.Line   (xlie1,ylie1,xlie2,ylie2)       # linha inferior laje inferior esquerda
    dwg.draw.Line   (x1s,y1s,xlie1,ylie1)           # linha esquerda inferior

# desenho do retangulo da seção com laje inferior e superior esquerda sem misula
if lie == "S" and lse == "S" and lid == "N" and lsd == "N" and mislie == "N":
    dwg.draw.Line   (x1s,y1s,x2s,y2s)              # linha inferior
    dwg.draw.Line   (x2s,y2s,x3s,y3s)              # linha direita
    dwg.draw.Line   (x3s,y3s,x4s,y4s)              # linha superior
    dwg.draw.Line   (x4s,y4s,xlse1,ylse1)           # linha esquerda superior
    dwg.draw.Line   (xlse1,ylse1,xlse2,ylse2)       # linha superior laje superior esquerda
    dwg.draw.Line   (xlse3,ylse3,xlse4,ylse4)       # linha inferior laje superior esquerda
    dwg.draw.Line   (xlse3,ylse3,xlie3,ylie3)       # linha intermediária esquerda
    dwg.draw.Line   (xlie3,ylie3,xlie4,ylie4)       # linha superior laje inferior esquerda
    dwg.draw.Line   (xlie1,ylie1,xlie2,ylie2)       # linha inferior laje inferior esquerda
    dwg.draw.Line   (x1s,y1s,xlie1,ylie1)           # linha esquerda inferior

# desenho do retangulo da seção com laje inferior e superior esquerda com misula
if lie == "S" and lse == "S" and lid == "N" and lsd == "N" and mislie == "S":
    dwg.draw.Line   (x1s,y1s,x2s,y2s)              # linha inferior
    dwg.draw.Line   (x2s,y2s,x3s,y3s)              # linha direita
    dwg.draw.Line   (x3s,y3s,x4s,y4s)              # linha superior
    dwg.draw.Line   (x4s,y4s,xlse1,ylse1)           # linha esquerda superior
    dwg.draw.Line   (xlse1,ylse1,xlse2,ylse2)       # linha superior laje superior esquerda
    dwg.draw.Line   (xlse3,ylse3,xlse4,ylse4)       # linha inferior laje superior esquerda
    dwg.draw.Line   (xlse3,ylse3,xme2,yme2)         # linha intermediária esquerda
    dwg.draw.Line   (xme2,yme2,xme1,yme1)           # linha misula
```

```
    dwg.draw.Line   (xme1,yme1,xlie4,ylie4)         # linha superior laje inferior esquerda
    dwg.draw.Line   (xlie1,ylie1,xlie2,ylie2)       # linha inferior laje inferior esquerda
    dwg.draw.Line   (x1s,y1s,xlie1,ylie1)           # linha esquerda inferior

# desenho do retangulo da seção com laje inferior e superior esquerda e laje superior direita sem misula
if lie == "S" and lse == "S" and lid == "N" and lsd == "S" and mislie == "N":
    dwg.draw.Line   (x1s,y1s,x2s,y2s)               # linha inferior
    dwg.draw.Line   (x2s,y2s,xlsd3,ylsd3)           # linha direita inferior
    dwg.draw.Line   (xlsd3,ylsd3,xlsd4,ylsd4)       # linha inferior laje superior direita
    dwg.draw.Line   (xlsd1,ylsd1,xlsd2,ylsd2)       # linha superior laje superior direita
    dwg.draw.Line   (xlsd1,ylsd1,x3s,y3s)           # linha superior direita
    dwg.draw.Line   (x3s,y3s,x4s,y4s)               # linha superior
    dwg.draw.Line   (x4s,y4s,xlse1,ylse1)           # linha esquerda superior
    dwg.draw.Line   (xlse1,ylse1,xlse2,ylse2)       # linha superior laje esquerda superior
    dwg.draw.Line   (xlse3,ylse3,xlse4,ylse4)       # linha inferior laje esquerda superior
    dwg.draw.Line   (xlse3,ylse3,xlie3,ylie3)       # linha esquerda intermediária
    dwg.draw.Line   (xlie3,ylie3,xlie4,ylie4)       # linha superior laje inferior esquerda
    dwg.draw.Line   (xlie1,ylie1,xlie2,ylie2)       # linha inferior laje inferior esquerda
    dwg.draw.Line   (xlie1,ylie1,x1s,y1s)           # linha esquerda inferior

# desenho do retangulo da seção com laje inferior e superior esquerda e laje superior direita com misula
if lie == "S" and lse == "S" and lid == "N" and lsd == "S" and mislie == "S":
    dwg.draw.Line   (x1s,y1s,x2s,y2s)               # linha inferior
    dwg.draw.Line   (x2s,y2s,xlsd3,ylsd3)           # linha direita inferior
    dwg.draw.Line   (xlsd3,ylsd3,xlsd4,ylsd4)       # linha inferior laje superior direita
    dwg.draw.Line   (xlsd1,ylsd1,xlsd2,ylsd2)       # linha superior laje superior direita
    dwg.draw.Line   (xlsd1,ylsd1,x3s,y3s)           # linha superior direita
    dwg.draw.Line   (x3s,y3s,x4s,y4s)               # linha superior
    dwg.draw.Line   (x4s,y4s,xlse1,ylse1)           # linha esquerda superior
    dwg.draw.Line   (xlse1,ylse1,xlse2,ylse2)       # linha superior laje esquerda superior
    dwg.draw.Line   (xlse3,ylse3,xlse4,ylse4)       # linha inferior laje esquerda superior
    dwg.draw.Line   (xlse3,ylse3,xme2,yme2)         # linha esquerda intermediária
    dwg.draw.Line   (xme2,yme2,xme1,yme1)           # linha misula
    dwg.draw.Line   (xme1,yme1,xlie4,ylie4)         # linha superior laje inferior esquerda
    dwg.draw.Line   (xlie1,ylie1,xlie2,ylie2)       # linha inferior laje inferior esquerda
    dwg.draw.Line   (xlie1,ylie1,x1s,y1s)           # linha esquerda inferior

# desenho do retangulo da seção com laje inferior e superior esquerda e laje inferior direita sem misula
if lie == "S" and lse == "S" and lid == "S" and lsd == "N" and mislie == "N" and mislid == "N":
    dwg.draw.Line   (x1s,y1s,x2s,y2s)               # linha inferior
    dwg.draw.Line   (x2s,y2s,xlid1,ylid1)           # linha direita inferior
    dwg.draw.Line   (xlid1,ylid1,xlid2,ylid2)       # linha inferior laje superior direita
    dwg.draw.Line   (xlid3,ylid3,xlid4,ylid4)       # linha superior laje superior direita
    dwg.draw.Line   (xlid3,ylid3,x3s,y3s)           # linha superior direita
    dwg.draw.Line   (x3s,y3s,x4s,y4s)               # linha superior
    dwg.draw.Line   (x4s,y4s,xlse1,ylse1)           # linha esquerda superior
    dwg.draw.Line   (xlse1,ylse1,xlse2,ylse2)       # linha superior laje esquerda superior
    dwg.draw.Line   (xlse3,ylse3,xlse4,ylse4)       # linha inferior laje esquerda superior
    dwg.draw.Line   (xlse3,ylse3,xlie3,ylie3)       # linha esquerda intermediária
    dwg.draw.Line   (xlie3,ylie3,xlie4,ylie4)       # linha superior laje inferior esquerda
    dwg.draw.Line   (xlie1,ylie1,xlie2,ylie2)       # linha inferior laje inferior esquerda
    dwg.draw.Line   (xlie1,ylie1,x1s,y1s)           # linha esquerda inferior

# desenho do retangulo da seção com laje inferior e superior esquerda e laje inferior direita com misula esquerda
if lie == "S" and lse == "S" and lid == "S" and lsd == "N" and mislie == "S" and mislid == "N":
    dwg.draw.Line   (x1s,y1s,x2s,y2s)               # linha inferior
    dwg.draw.Line   (x2s,y2s,xlid1,ylid1)           # linha direita inferior
    dwg.draw.Line   (xlid1,ylid1,xlid2,ylid2)       # linha inferior laje superior direita
    dwg.draw.Line   (xlid3,ylid3,xlid4,ylid4)       # linha superior laje superior direita
    dwg.draw.Line   (xlid3,ylid3,x3s,y3s)           # linha superior direita
```

```
    dwg.draw.Line   (x3s,y3s,x4s,y4s)             # linha superior
    dwg.draw.Line   (x4s,y4s,xlse1,ylse1)          # linha esquerda superior
    dwg.draw.Line   (xlse1,ylse1,xlse2,ylse2)       # linha superior laje esquerda superior
    dwg.draw.Line   (xlse3,ylse3,xlse4,ylse4)       # linha inferior laje esquerda superior
    dwg.draw.Line   (xlse3,ylse3,xme2,yme2)         # linha esquerda intermediária
    dwg.draw.Line   (xme2,yme2,xme1,yme1)           # linha misula
    dwg.draw.Line   (xme1,yme1,xlie4,ylie4)         # linha superior laje inferior esquerda
    dwg.draw.Line   (xlie1,ylie1,xlie2,ylie2)       # linha inferior laje inferior esquerda
    dwg.draw.Line   (xlie1,ylie1,x1s,y1s)           # linha esquerda inferior

# desenho do retangulo da seção com laje inferior e superior esquerda e laje inferior direita com misula esquerda
e direita
if lie == "S" and lse == "S" and lid == "S" and lsd == "N" and mislie == "S" and mislid == "N":
    dwg.draw.Line   (x1s,y1s,x2s,y2s)             # linha inferior
    dwg.draw.Line   (x2s,y2s,xlid1,ylid1)          # linha direita inferior
    dwg.draw.Line   (xlid1,ylid1,xlid2,ylid2)       # linha inferior laje superior direita
    dwg.draw.Line   (xmd1,ymd1,xlid4,ylid4)         # linha superior laje superior direita
    dwg.draw.Line   (xmd1,ymd1,xmd2,ymd2)           # linha misula direita
    dwg.draw.Line   (xmd2,ymd2,x3s,y3s)            # linha superior direita
    dwg.draw.Line   (x3s,y3s,x4s,y4s)             # linha superior
    dwg.draw.Line   (x4s,y4s,xlse1,ylse1)          # linha esquerda superior
    dwg.draw.Line   (xlse1,ylse1,xlse2,ylse2)       # linha superior laje esquerda superior
    dwg.draw.Line   (xlse3,ylse3,xlse4,ylse4)       # linha inferior laje esquerda superior
    dwg.draw.Line   (xlse3,ylse3,xme2,yme2)         # linha esquerda intermediária
    dwg.draw.Line   (xme2,yme2,xme1,yme1)           # linha misula esquerda
    dwg.draw.Line   (xme1,yme1,xlie4,ylie4)         # linha superior laje inferior esquerda
    dwg.draw.Line   (xlie1,ylie1,xlie2,ylie2)       # linha inferior laje inferior esquerda
    dwg.draw.Line   (xlie1,ylie1,x1s,y1s)           # linha esquerda inferior

# desenho do retangulo da seção com laje inferior direita sem misula
if lie == "N" and lse == "N" and lid == "S" and lsd == "N" and mislid == "N":
    dwg.draw.Line   (x1s,y1s,x2s,y2s)             # linha inferior
    dwg.draw.Line   (x2s,y2s,xlid1,ylid1)          # linha direita inferior
    dwg.draw.Line   (xlid1,ylid1,xlid2,ylid2)       # linha inferior laje inferior direita
    dwg.draw.Line   (xlid3,ylid3,xlid4,ylid4)       # linha superior laje inferior direita
    dwg.draw.Line   (xlid3,ylid3,x3s,y3s)          # linha direita superior
    dwg.draw.Line   (x3s,y3s,x4s,y4s)             # linha superior
    dwg.draw.Line   (x4s,y4s,x1s,y1s)             # linha esquerda

# desenho do retangulo da seção com laje inferior direita com misula
if lie == "N" and lse == "N" and lid == "S" and lsd == "N" and mislid == "S":
    dwg.draw.Line   (x1s,y1s,x2s,y2s)             # linha inferior
    dwg.draw.Line   (x2s,y2s,xlid1,ylid1)          # linha direita inferior
    dwg.draw.Line   (xlid1,ylid1,xlid2,ylid2)       # linha inferior laje inferior direita
    dwg.draw.Line   (xmd1,ymd1,xlid4,ylid4)         # linha superior laje inferior direita
    dwg.draw.Line   (xmd1,ymd1,xmd2,ymd2)           # linha misula
    dwg.draw.Line   (xmd2,ymd2,x3s,y3s)            # linha direita superior
    dwg.draw.Line   (x3s,y3s,x4s,y4s)             # linha superior
    dwg.draw.Line   (x4s,y4s,x1s,y1s)             # linha esquerda

# desenho do retangulo da seção com laje inferior e superior direita sem misula
if lie == "N" and lse == "N" and lid == "S" and lsd == "S" and mislid == "N":
    dwg.draw.Line   (x1s,y1s,x2s,y2s)             # linha inferior
    dwg.draw.Line   (x2s,y2s,xlid1,ylid1)          # linha direita inferior
    dwg.draw.Line   (xlid1,ylid1,xlid2,ylid2)       # linha inferior laje inferior direita
    dwg.draw.Line   (xlid3,ylid3,xlid4,ylid4)       # linha superior laje inferior direita
    dwg.draw.Line   (xlid3,ylid3,xlsd3,ylsd3)       # linha direita intermediária
    dwg.draw.Line   (xlsd3,ylsd3,xlsd4,ylsd4)       # linha inferior laje superior direita
    dwg.draw.Line   (xlsd1,ylsd1,xlsd2,ylsd2)       # linha superior laje superior direita
    dwg.draw.Line   (xlsd1,ylsd1,x3s,y3s)          # linha direita superior
```

```
    dwg.draw.Line   (x3s,y3s,x4s,y4s)            # linha superior
    dwg.draw.Line   (x4s,y4s,x1s,y1s)            # linha esquerda

# desenho do retangulo da seção com laje inferior e superior direita com misula
if lie == "N" and lse == "N" and lid == "S" and lsd == "S" and mislid == "S":
    dwg.draw.Line   (x1s,y1s,x2s,y2s)            # linha inferior
    dwg.draw.Line   (x2s,y2s,xlid1,ylid1)         # linha direita inferior
    dwg.draw.Line   (xlid1,ylid1,xlid2,ylid2)      # linha inferior laje inferior direita
    dwg.draw.Line   (xmd1,ymd1,xlid4,ylid4)        # linha superior laje inferior direita
    dwg.draw.Line   (xmd1,ymd1,xmd2,ymd2)          # linha misula
    dwg.draw.Line   (xmd2,ymd2,xlsd3,ylsd3)        # linha direita intermediária
    dwg.draw.Line   (xlsd3,ylsd3,xlsd4,ylsd4)      # linha inferior laje superior direita
    dwg.draw.Line   (xlsd1,ylsd1,xlsd2,ylsd2)      # linha superior laje superior direita
    dwg.draw.Line   (xlsd1,ylsd1,x3s,y3s)          # linha direita superior
    dwg.draw.Line   (x3s,y3s,x4s,y4s)            # linha superior
    dwg.draw.Line   (x4s,y4s,x1s,y1s)            # linha esquerda

# desenho do retangulo da seção com laje inferior e superior direita e laje superior esquerda sem misula
if lie == "N" and lse == "S" and lid == "S" and lsd == "S" and mislid == "N":
    dwg.draw.Line   (x1s,y1s,x2s,y2s)            # linha inferior
    dwg.draw.Line   (x2s,y2s,xlid1,ylid1)         # linha direita inferior
    dwg.draw.Line   (xlid1,ylid1,xlid2,ylid2)      # linha inferior laje inferior direita
    dwg.draw.Line   (xlid3,ylid3,xlid4,ylid4)      # linha superior laje inferior direita
    dwg.draw.Line   (xlid3,ylid3,xlsd3,ylsd3)      # linha direita intermediária
    dwg.draw.Line   (xlsd3,ylsd3,xlsd4,ylsd4)      # linha inferior laje superior direita
    dwg.draw.Line   (xlsd1,ylsd1,xlsd2,ylsd2)      # linha superior laje superior direita
    dwg.draw.Line   (xlsd1,ylsd1,x3s,y3s)          # linha direita superior
    dwg.draw.Line   (x3s,y3s,x4s,y4s)            # linha superior
    dwg.draw.Line   (x4s,y4s,xlse1,ylse1)         # linha esquerda superior
    dwg.draw.Line   (xlse1,ylse1,xlse2,ylse2)      # linha superior laje superior esquerda
    dwg.draw.Line   (xlse3,ylse3,xlse4,ylse4)      # linha inferior laje superior esquerda
    dwg.draw.Line   (xlse3,ylse3,x1s,y1s)          # linha esquerda inferior

# desenho do retangulo da seção com laje inferior e superior direita e laje superior esquerda com misula
if lie == "N" and lse == "S" and lid == "S" and lsd == "S" and mislid == "S":
    dwg.draw.Line   (x1s,y1s,x2s,y2s)            # linha inferior
    dwg.draw.Line   (x2s,y2s,xlid1,ylid1)         # linha direita inferior
    dwg.draw.Line   (xlid1,ylid1,xlid2,ylid2)      # linha inferior laje inferior direita
    dwg.draw.Line   (xmd1,ymd1,xlid4,ylid4)        # linha superior laje inferior direita
    dwg.draw.Line   (xmd1,ymd1,xmd2,ymd2)          # linha misula
    dwg.draw.Line   (xmd2,ymd2,xlsd3,ylsd3)        # linha direita intermediária
    dwg.draw.Line   (xlsd3,ylsd3,xlsd4,ylsd4)      # linha inferior laje superior direita
    dwg.draw.Line   (xlsd1,ylsd1,xlsd2,ylsd2)      # linha superior laje superior direita
    dwg.draw.Line   (xlsd1,ylsd1,x3s,y3s)          # linha direita superior
    dwg.draw.Line   (x3s,y3s,x4s,y4s)            # linha superior
    dwg.draw.Line   (x4s,y4s,xlse1,ylse1)         # linha esquerda superior
    dwg.draw.Line   (xlse1,ylse1,xlse2,ylse2)      # linha superior laje superior esquerda
    dwg.draw.Line   (xlse3,ylse3,xlse4,ylse4)      # linha inferior laje superior esquerda
    dwg.draw.Line   (xlse3,ylse3,x1s,y1s)          # linha esquerda inferior

# desenho do retangulo da seção com laje inferior e superior direita e laje inferior esquerda sem misula
if lie == "S" and lse == "N" and lid == "S" and lsd == "S" and mislie == "N" and mislid == "N":
    dwg.draw.Line   (x1s,y1s,x2s,y2s)            # linha inferior
    dwg.draw.Line   (x2s,y2s,xlid1,ylid1)         # linha direita inferior
    dwg.draw.Line   (xlid1,ylid1,xlid2,ylid2)      # linha inferior laje inferior direita
    dwg.draw.Line   (xlid3,ylid3,xlid4,ylid4)      # linha superior laje inferior direita
    dwg.draw.Line   (xlid3,ylid3,xlsd3,ylsd3)      # linha direita intermediária
    dwg.draw.Line   (xlsd3,ylsd3,xlsd4,ylsd4)      # linha inferior laje superior direita
    dwg.draw.Line   (xlsd1,ylsd1,xlsd2,ylsd2)      # linha superior laje superior direita
    dwg.draw.Line   (xlsd1,ylsd1,x3s,y3s)          # linha direita superior
```

```
    dwg.draw.Line   (x3s,y3s,x4s,y4s)             # linha superior
    dwg.draw.Line   (x4s,y4s,xlie3,ylie3)         # linha esquerda superior
    dwg.draw.Line   (xlie3,ylie3,xlie4,ylie4)     # linha superior laje inferior esquerda
    dwg.draw.Line   (xlie1,ylie1,xlie2,ylie2)     # linha inferior laje inferior esquerda
    dwg.draw.Line   (xlie1,ylie1,x1s,y1s)         # linha esquerda inferior

# desenho do retangulo da seção com laje inferior e superior direita e laje inferior esquerda com misula direita
if lie == "S" and lse == "N" and lid == "S" and lsd == "S" and mislie == "N" and mislid == "S":
    dwg.draw.Line   (x1s,y1s,x2s,y2s)             # linha inferior
    dwg.draw.Line   (x2s,y2s,xlid1,ylid1)         # linha direita inferior
    dwg.draw.Line   (xlid1,ylid1,xlid2,ylid2)     # linha inferior laje inferior direita
    dwg.draw.Line   (xmd1,ymd1,xlid4,ylid4)       # linha superior laje inferior direita
    dwg.draw.Line   (xmd1,ymd1,xmd2,ymd2)         # linha misula
    dwg.draw.Line   (xmd2,ymd2,xlsd3,ylsd3)       # linha direita intermediária
    dwg.draw.Line   (xlsd3,ylsd3,xlsd4,ylsd4)     # linha inferior laje superior direita
    dwg.draw.Line   (xlsd1,ylsd1,xlsd2,ylsd2)     # linha superior laje superior direita
    dwg.draw.Line   (xlsd1,ylsd1,x3s,y3s)         # linha direita superior
    dwg.draw.Line   (x3s,y3s,x4s,y4s)             # linha superior
    dwg.draw.Line   (x4s,y4s,xlie3,ylie3)         # linha esquerda superior
    dwg.draw.Line   (xlie3,ylie3,xlie4,ylie4)     # linha superior laje inferior esquerda
    dwg.draw.Line   (xlie1,ylie1,xlie2,ylie2)     # linha inferior laje inferior esquerda
    dwg.draw.Line   (xlie1,ylie1,x1s,y1s)         # linha esquerda inferior

# desenho do retangulo da seção com laje inferior e superior direita e laje inferior esquerda com misula direita e
esquerda
if lie == "S" and lse == "N" and lid == "S" and lsd == "S" and mislie == "S" and mislid == "S":
    dwg.draw.Line   (x1s,y1s,x2s,y2s)             # linha inferior
    dwg.draw.Line   (x2s,y2s,xlid1,ylid1)         # linha direita inferior
    dwg.draw.Line   (xlid1,ylid1,xlid2,ylid2)     # linha inferior laje inferior direita
    dwg.draw.Line   (xmd1,ymd1,xlid4,ylid4)       # linha superior laje inferior direita
    dwg.draw.Line   (xmd1,ymd1,xmd2,ymd2)         # linha misula direita
    dwg.draw.Line   (xmd2,ymd2,xlsd3,ylsd3)       # linha direita intermediária
    dwg.draw.Line   (xlsd3,ylsd3,xlsd4,ylsd4)     # linha inferior laje superior direita
    dwg.draw.Line   (xlsd1,ylsd1,xlsd2,ylsd2)     # linha superior laje superior direita
    dwg.draw.Line   (xlsd1,ylsd1,x3s,y3s)         # linha direita superior
    dwg.draw.Line   (x3s,y3s,x4s,y4s)             # linha superior
    dwg.draw.Line   (x4s,y4s,xme2,yme2)           # linha esquerda superior
    dwg.draw.Line   (xme2,yme2,xme1,yme1)         # linha misula esquerda
    dwg.draw.Line   (xme1,yme1,xlie4,ylie4)       # linha superior laje inferior esquerda
    dwg.draw.Line   (xlie1,ylie1,xlie2,ylie2)     # linha inferior laje inferior esquerda
    dwg.draw.Line   (xlie1,ylie1,x1s,y1s)         # linha esquerda inferior

# desenho do retangulo da seção com laje inferior e superior direita e laje inferior e superior esquerda sem misula
if lie == "S" and lse == "S" and lid == "S" and lsd == "S" and mislie == "N" and mislid == "N":
    dwg.draw.Line   (x1s,y1s,x2s,y2s)             # linha inferior
    dwg.draw.Line   (x2s,y2s,xlid1,ylid1)         # linha direita inferior
    dwg.draw.Line   (xlid1,ylid1,xlid2,ylid2)     # linha inferior laje inferior direita
    dwg.draw.Line   (xlid3,ylid3,xlid4,ylid4)     # linha superior laje inferior direita
    dwg.draw.Line   (xlid3,ylid3,xlsd3,ylsd3)     # linha direita intermediária
    dwg.draw.Line   (xlsd3,ylsd3,xlsd4,ylsd4)     # linha inferior laje superior direita
    dwg.draw.Line   (xlsd1,ylsd1,xlsd2,ylsd2)     # linha superior laje superior direita
    dwg.draw.Line   (xlsd1,ylsd1,x3s,y3s)         # linha direita superior
    dwg.draw.Line   (x3s,y3s,x4s,y4s)             # linha superior
    dwg.draw.Line   (x4s,y4s,xlse1,ylse1)         # linha esquerda superior
    dwg.draw.Line   (xlse1,ylse1,xlse2,ylse2)     # linha superior laje superior esquerda
    dwg.draw.Line   (xlse3,ylse3,xlse4,ylse4)     # linha inferior laje superior esquerda
    dwg.draw.Line   (xlse3,ylse3,xlie3,ylie3)     # linha esquerda intermediária
    dwg.draw.Line   (xlie3,ylie3,xlie4,ylie4)     # linha superior laje inferior esquerda
    dwg.draw.Line   (xlie1,ylie1,xlie2,ylie2)     # linha inferior laje inferior esquerda
    dwg.draw.Line   (xlie1,ylie1,x1s,y1s)         # linha esquerda inferior
```

```
# desenho do retangulo da seção com laje inferior e superior direita e laje inferior e superior esquerda com misula esquerda
if lie == "S" and lse == "S" and lid == "S" and lsd == "S" and mislie == "S" and mislid == "N":
    dwg.draw.Line   (x1s,y1s,x2s,y2s)             # linha inferior
    dwg.draw.Line   (x2s,y2s,xlid1,ylid1)          # linha direita inferior
    dwg.draw.Line   (xlid1,ylid1,xlid2,ylid2)       # linha inferior laje inferior direita
    dwg.draw.Line   (xlid3,ylid3,xlid4,ylid4)       # linha superior laje inferior direita
    dwg.draw.Line   (xlid3,ylid3,xlsd3,ylsd3)       # linha direita intermediária
    dwg.draw.Line   (xlsd3,ylsd3,xlsd4,ylsd4)       # linha inferior laje superior direita
    dwg.draw.Line   (xlsd1,ylsd1,xlsd2,ylsd2)       # linha superior laje superior direita
    dwg.draw.Line   (xlsd1,ylsd1,x3s,y3s)          # linha direita superior
    dwg.draw.Line   (x3s,y3s,x4s,y4s)             # linha superior
    dwg.draw.Line   (x4s,y4s,xlse1,ylse1)          # linha esquerda superior
    dwg.draw.Line   (xlse1,ylse1,xlse2,ylse2)       # linha superior laje superior esquerda
    dwg.draw.Line   (xlse3,ylse3,xlse4,ylse4)       # linha inferior laje superior esquerda
    dwg.draw.Line   (xlse3,ylse3,xme2,yme2)         # linha esquerda intermediária
    dwg.draw.Line   (xme2,yme2,xme1,yme1)           # linha misula
    dwg.draw.Line   (xme1,yme1,xlie4,ylie4)         # linha superior laje inferior esquerda
    dwg.draw.Line   (xlie1,ylie1,xlie2,ylie2)       # linha inferior laje inferior esquerda
    dwg.draw.Line   (xlie1,ylie1,x1s,y1s)          # linha esquerda inferior

# desenho do retangulo da seção com laje inferior e superior direita e laje inferior e superior esquerda com misula direita
if lie == "S" and lse == "S" and lid == "S" and lsd == "S" and mislie == "S" and mislid == "S":
    dwg.draw.Line   (x1s,y1s,x2s,y2s)             # linha inferior
    dwg.draw.Line   (x2s,y2s,xlid1,ylid1)          # linha direita inferior
    dwg.draw.Line   (xlid1,ylid1,xlid2,ylid2)       # linha inferior laje inferior direita
    dwg.draw.Line   (xmd1,ymd1,xlid4,ylid4)         # linha superior laje inferior direita
    dwg.draw.Line   (xmd1,ymd1,xmd2,ymd2)           # linha misula
    dwg.draw.Line   (xmd2,ymd2,xlsd3,ylsd3)         # linha direita intermediária
    dwg.draw.Line   (xlsd3,ylsd3,xlsd4,ylsd4)       # linha inferior laje superior direita
    dwg.draw.Line   (xlsd1,ylsd1,xlsd2,ylsd2)       # linha superior laje superior direita
    dwg.draw.Line   (xlsd1,ylsd1,x3s,y3s)          # linha direita superior
    dwg.draw.Line   (x3s,y3s,x4s,y4s)             # linha superior
    dwg.draw.Line   (x4s,y4s,xlse1,ylse1)          # linha esquerda superior
    dwg.draw.Line   (xlse1,ylse1,xlse2,ylse2)       # linha superior laje superior esquerda
    dwg.draw.Line   (xlse3,ylse3,xlse4,ylse4)       # linha inferior laje superior esquerda
    dwg.draw.Line   (xlse3,ylse3,xme2,yme2)         # linha esquerda intermediária
    dwg.draw.Line   (xme2,yme2,xme1,yme1)           # linha misula
    dwg.draw.Line   (xme1,yme1,xlie4,ylie4)         # linha superior laje inferior esquerda
    dwg.draw.Line   (xlie1,ylie1,xlie2,ylie2)       # linha inferior laje inferior esquerda
    dwg.draw.Line   (xlie1,ylie1,x1s,y1s)          # linha esquerda inferior

# desenho da representação do estribo na seção
dwg.draw.Line   (0+cobr*2,0+cobr*2,largura*2-cobr*2,0+cobr*2)              # linha inferior
dwg.draw.Line   (largura*2-cobr*2,0+cobr*2,largura*2-cobr*2,altura*2-cobr*2)    # linha direita
dwg.draw.Line   (largura*2-cobr*2,altura*2-cobr*2,0+cobr*2,altura*2-cobr*2)     # linha superior
dwg.draw.Line   (0+cobr*2,altura*2-cobr*2,0+cobr*2,0+cobr*2)               # linha esquerda

# desenho da representação da armadura da misula na seção

ang = 45
angrad = math.radians(ang)
distmisula1e = hlie*2-4*cobr*math.cos(angrad)-2*cobr          # distancia para o ponto da misula na laje
distmisula1d = hlid*2-4*cobr*math.cos(angrad)-2*cobr
distmisula2 = largura*2-4*cobr*math.cos(angrad)-2*cobr        # distancia para o ponto da misula na parede

if mislie == "S":
    xame1 = xme1-distmisula1e                  # pontos para desenho da representação da barra da misula
```

```
    yame1 = ylie1+cobr*2
    xame2 = x2s-cobr*2
    yame2 = yme2+distmisula2
    xame3 = xame1-40
    yame3 = yame1
    xame4 = xame1+5
    yame4 = yame1+5
    xame5 = xame3
    yame5 = ylie4+26
    xame6 = xame4+7.07
    yame6 = yame4-7.07
    dwg.draw.level = 224
    dwg.draw.Line   (xame1,yame1,xame2,yame2)      # desenho da linha de representação da barra da misula
    dwg.draw.Line   (xame1,yame1,xame3,yame3)
    dwg.draw.level = 225
    dwg.draw.Line   (xame6,yame6,xame5,yame5)      # Linha da identificação da barra
    dwg.draw.Line   (xame4,yame4-5,xame4,yame4+5)

if mislid == "S":
    xamd1 = xmd1+distmisula1d                    # pontos para desenho da representação da barra da misula
    yamd1 = ylid1+cobr*2
    xamd2 = x1s+cobr*2
    yamd2 = ymd2+distmisula2
    xamd3 = xamd1+40
    yamd3 = yamd1
    xamd4 = xamd1-5
    yamd4 = yamd1+5
    xamd5 = xamd3
    yamd5 = ylid3+26
    xamd6 = xamd4-7.07
    yamd6 = yamd4-7.07
    dwg.draw.level = 224
    dwg.draw.Line   (xamd1,yamd1,xamd2,yamd2)      # desenho da linha de representação da barra da misula
    dwg.draw.Line   (xamd1,yamd1,xamd3,yamd3)
    dwg.draw.level = 225
    dwg.draw.Line   (xamd6,yamd6,xamd5,yamd5)      # Linha da identificação da barra
    dwg.draw.Line   (xamd4,yamd4-5,xamd4,yamd4+5)


# textos das dimensões da seção
dwg.draw.level = 224
dwg.draw.Text (largura/2, altura*2+5, 10, 0, str(int(largura)))     # texto largura
dwg.draw.Text (0-5, altura-8, 10, 90, str(int(altura)))             # texto altura

# representação da armação longitudinal na seção

dwg.draw.level = 220    # nivel do circulo
dwg.draw.Circle (0+cobr*2+2.75,0+cobr*2+2.75,2.5)              # armação positiva
dwg.draw.Circle (0+largura*2-cobr*2-2.75,0+cobr*2+2.75,2.5)    # armação positiva

dwg.draw.level = 225    # nivel linha de chamada
dwg.draw.Line (0+largura*2-cobr*2-2.75,0+cobr*2+2.75,0+largura*2-cobr*2-2.75+40,0-40)   #linha de
chamada

# altura de distribuição da armação positiva
hlong = 0.15*altura

if distpos == "H":
    if qpos > 2:                          # distribuição das barras positivas
        epos1 = 0+cobr*2+2.75                 # posição inicial no desenho
```

```
      epos2 = 0+0.15*altura*2-cobr*2         # posição final no desenho
      epos = (epos2-epos1) / ((qpos/2)-1)    # espaçamento das barras
   #   print (epos)
      for i in range(int(qpos/2 - 1)):           # comando para repetição dos desenhos
         x1 = 0+cobr*2+2.75                  # posição incial
         x2 = 0+largura*2-cobr*2-2.75
         y1 = 0+cobr*2+2.75 + epos + i * epos    # posição inicial até o intervalo da repetição, já que a barra é
distribuida na vertical Y
         y2 = 0+cobr*2+2.75 + epos + i * epos

         dwg.draw.level = 220
         dwg.draw.Circle (x1,y1,2.5)           # desenho da representação das barras até o intervalo de repetição
         dwg.draw.Circle (x2,y2,2.5)

         dwg.draw.level = 225
         dwg.draw.Line (x2,y2,x2+40,0-40)    # desenhos das linhas de chamada

if distpos == "C":
   if qpos > 2:                          # distribuição das barras positivas
      epos1 = 0+cobr*2+2.75                # posição inicial no desenho
      epos = 7                           # espaçamento das barras
   #   print (epos)
      for i in range(int(qpos/2 - 1)):           # comando para repetição dos desenhos
         x1 = 0+cobr*2+2.75                  # posição incial
         x2 = 0+largura*2-cobr*2-2.75
         y1 = 0+cobr*2+2.75 + epos + i * epos    # posição inicial até o intervalo da repetição, já que a barra é
distribuida na vertical Y
         y2 = 0+cobr*2+2.75 + epos + i * epos

         dwg.draw.level = 220
         dwg.draw.Circle (x1,y1,2.5)           # desenho da representação das barras até o intervalo de repetição
         dwg.draw.Circle (x2,y2,2.5)

         dwg.draw.level = 225
         dwg.draw.Line (x2,y2,x2+40,0-40)    # desenhos das linhas de chamada

dwg.draw.level = 220
dwg.draw.Circle (0+cobr*2+2.75,0+altura*2-cobr*2-2.75,2.5)             # armação negativa
dwg.draw.Circle (0+largura*2-cobr*2-2.75,0+altura*2-cobr*2-2.75,2.5)    # armação negativa

dwg.draw.level = 225    # nivel linha de chamada
dwg.draw.Line (0+largura*2-cobr*2-2.75,0+altura*2-cobr*2-2.75,0+largura*2-cobr*2-2.75+40,altura*2+20)
#linha de chamada

if qneg > 2:          # distribuição das barras negativas
   eneg = 7         # espaçamento das barras
   for i in range(int(qneg/2 - 1)):           # comando para repetição dos desenhos
      x1 = 0+cobr*2+2.75                   # posição incial
      x2 = 0+largura*2-cobr*2-2.75
      y1 = 0+altura*2-cobr*2-2.75 - eneg - i * eneg    # posição inicial até o intervalo da repetição, já que a barra
é distribuida na vertical Y
      y2 = 0+altura*2-cobr*2-2.75 - eneg - i * eneg

      dwg.draw.level = 220
      dwg.draw.Circle (x1,y1,2.5)           # desenho da representação das barras até o intervalo de repetição
      dwg.draw.Circle (x2,y2,2.5)

      dwg.draw.level = 225
      dwg.draw.Line (x2,y2,x2+40,altura*2+20)    # desenhos das linhas de chamada
```

```
# armadura de pele

xp1 = 0+cobr*2+2.75              # posição em X esquerda
xp2 = 0+largura*2-cobr*2-2.75      # posição em X direita

if qpos == 2:
   yp1 = 0+cobr*2+2.75           # posição Y para positivos com apenas 1 camada

if qpos > 2:
   if distpos == "H":
      yp1 = 0-cobr*2+(0.15*altura)*2            # posição Y para positivos com mais de 1 camada
   elif distpos == "C":
      yp1 = 0+cobr*2+2.75+(qpos/2-1)*7    # posição Y para positivos com mais de 1 camada

if qneg == 2:
   yp2 = 0+altura*2-cobr*2-2.75              # posição Y para negativos com apenas 1 camada
elif qneg > 2:
   yp2 = altura*2-cobr*2-2.75-(7*(qneg/2-1))   # posição Y para negativos com mais de 1 camada

epele = (yp2-yp1) / (qpele+1)
#print (yp2-yp1)

for i in range(int(qpele)):        # comando para repetição dos desenhos
   x1 = xp1                    # posição incial
   x2 = xp2
   y = yp1 + epele + i * epele     # posição inicial até o intervalo da repetição, já que a barra é distribuida na
vertical Y

   dwg.draw.level = 220
   dwg.draw.Circle (x1,y,2.5)           # desenho da representação das barras até o intervalo de repetição
   dwg.draw.Circle (x2,y,2.5)

   dwg.draw.level = 225
   dwg.draw.Line (x1,y,x1-35,y)                  # desenho das linhas de chamada da armadura de pele
   dwg.draw.Line (x1-35,yp1+epele,x1-35,yp2-epele)

# desenho das armaduras longitudinais na seção
# desenho das linhas de representação das barras
dwg.draw.level = 220
xlp1 = largura*2+25          # pontos para desenho da linha de ferro positivo
ylp1 = 0-80
xlp2 = xlp1+70
ylp2 = ylp1
xlp3 = xlp2+6
ylp3 = ylp2+12
xlp4 = xlp3
ylp4 = ylp3-24
xlp5 = xlp4+6
ylp5 = ylp2
xlp6 = xlp5+70
ylp6 = ylp5
xlp7 = xlp1+11.31           # pontos para as dobras
ylp7 = ylp1+11.31
xlp8 = xlp7+40
ylp8 = ylp7
xlp9 = xlp6+11.31
ylp9 = ylp6+11.31
xlp10 = xlp9-40
ylp10 = ylp9
```

```
dwg.draw.Line (xlp1,ylp1,xlp2,ylp2)     # desenho das linhas
dwg.draw.Line (xlp2,ylp2,xlp3,ylp3)
dwg.draw.Line (xlp3,ylp3,xlp4,ylp4)
dwg.draw.Line (xlp4,ylp4,xlp5,ylp5)
dwg.draw.Line (xlp5,ylp5,xlp6,ylp6)
dwg.draw.Line (xlp1,ylp1,xlp7,ylp7)
dwg.draw.Line (xlp7,ylp7,xlp8,ylp8)
dwg.draw.Line (xlp6,ylp6,xlp9,ylp9)
dwg.draw.Line (xlp9,ylp9,xlp10,ylp10)

dwg.draw.level = 224
dwg.draw.Text ((xlp7+xlp8)/2-8,ylp7+5,10,0,str(20))       # textos das dobras
dwg.draw.Text ((xlp9+xlp10)/2-8,ylp9+5,10,0,str(20))
dp = largura-cobr*2-2
dwg.draw.Text (xlp1-25,ylp1,10,0,str(int(dp)))
dwg.draw.Text (xlp6+15,ylp6,10,0,str(int(dp)))

xln1 = largura*2+25          # pontos para desenho da linha de ferro negativo
yln1 = altura*2+45
xln2 = xln1+70
yln2 = yln1
xln3 = xln2+6
yln3 = yln2+12
xln4 = xln3
yln4 = yln3-24
xln5 = xln4+6
yln5 = yln2
xln6 = xln5+70
yln6 = yln5
xln7 = xln1+11.31           # pontos para as dobras
yln7 = yln1+11.31
xln8 = xln7+40
yln8 = yln7
xln9 = xln6+11.31
yln9 = yln6+11.31
xln10 = xln9-40
yln10 = yln9

dwg.draw.level = 220
dwg.draw.Line (xln1,yln1,xln2,yln2)     # desenho das linhas
dwg.draw.Line (xln2,yln2,xln3,yln3)
dwg.draw.Line (xln3,yln3,xln4,yln4)
dwg.draw.Line (xln4,yln4,xln5,yln5)
dwg.draw.Line (xln5,yln5,xln6,yln6)
dwg.draw.Line (xln1,yln1,xln7,yln7)
dwg.draw.Line (xln7,yln7,xln8,yln8)
dwg.draw.Line (xln6,yln6,xln9,yln9)
dwg.draw.Line (xln9,yln9,xln10,yln10)

dwg.draw.level = 224
dwg.draw.Text ((xln7+xln8)/2-8,yln7+5,10,0,str(20))       # textos das dobras
dwg.draw.Text ((xln9+xln10)/2-8,yln9+5,10,0,str(20))
dn = largura-cobr*2-2
dwg.draw.Text (xln1-25,yln1,10,0,str(int(dn)))
dwg.draw.Text (xln6+15,yln6,10,0,str(int(dn)))

xlpe1 = largura*2+40          # pontos para desenho da linha de ferro pele
ylpe1 = altura
xlpe2 = xlpe1+70
ylpe2 = ylpe1
```

```
xlpe3 = xlpe2+6
ylpe3 = ylpe2+12
xlpe4 = xlpe3
ylpe4 = ylpe3-24
xlpe5 = xlpe4+6
ylpe5 = ylpe2
xlpe6 = xlpe5+70
ylpe6 = ylpe5
xlpe7 = xlpe1+11.31          # pontos para as dobras
ylpe7 = ylpe1+11.31
xlpe8 = xlpe7+40
ylpe8 = ylpe7
xlpe9 = xlpe6+11.31
ylpe9 = ylpe6+11.31
xlpe10 = xlpe9-40
ylpe10 = ylpe9

dwg.draw.level = 220
dwg.draw.Line (xlpe1,ylpe1,xlpe2,ylpe2)     # desenho das linhas
dwg.draw.Line (xlpe2,ylpe2,xlpe3,ylpe3)
dwg.draw.Line (xlpe3,ylpe3,xlpe4,ylpe4)
dwg.draw.Line (xlpe4,ylpe4,xlpe5,ylpe5)
dwg.draw.Line (xlpe5,ylpe5,xlpe6,ylpe6)
dwg.draw.Line (xlpe1,ylpe1,xlpe7,ylpe7)
dwg.draw.Line (xlpe7,ylpe7,xlpe8,ylpe8)
dwg.draw.Line (xlpe6,ylpe6,xlpe9,ylpe9)
dwg.draw.Line (xlpe9,ylpe9,xlpe10,ylpe10)

dwg.draw.level = 224
dwg.draw.Text ((xlpe7+xlpe8)/2-8,ylpe7+5,10,0,str(20))       # textos das dobras
dwg.draw.Text ((xlpe9+xlpe10)/2-8,ylpe9+5,10,0,str(20))
dpe = largura-cobr*2-2
dwg.draw.Text (xlpe1-25,ylpe1,10,0,str(int(dpe)))
dwg.draw.Text (xlpe6+15,ylpe6,10,0,str(int(dpe)))

# textos dos ferros longitudinais

positivo = TQSDwg.SmartRebar (dwg)                          # Definindo o ferro positivo
positivo.type = TQSDwg.ICPFRT                              # Tipo de ferro - ferro reto
positivo.cover                = cobr                       # Cobrimento p/uso geral
positivo.mark                 = 1                          # Numero da posição
positivo.quantity             = qpos                       # Num ferros
positivo.multiplier           = 1                          # Multipl do numero de ferros
positivo.diameter             = bitpos                     # Bitola mm
positivo.straightBarMainLength   = vao-2*cobr              # Comprimento horizontal (cm)
positivo.straightBarLeftLength   = dp                      # Dobra a esquerda
positivo.straightBarRightLength  = dp                      # Dobra a direita
positivo.straightBarLeftLength2  = 20                      # 2a dobra a esquerda
positivo.straightBarRightLength2 = 20                      # 2a dobra a direita
positivo.straightBarZone      = TQSDwg.ICPPOS              # Tipo de ferro reto positivo ou negativo
positivo.RebarTextDisplayPosition (xlp1+15,ylp7+25,10,0,0,1,0)      # Texto do ferro
if distpos == "C":
   if qpos == 4:
      dwg.draw.Text (xln3-40,ylp7+45,10,0,"(2 { 2aCAM)")
   if qpos == 6:
      dwg.draw.Text (xln3-70,ylp7+45,10,0,"(2 { 2aCAM,2 { 3aCAM)")
   if qpos == 8:
      dwg.draw.Text (xln3-70,ylp7+57,10,0,"(2 { 2aCAM,2 { 3aCAM,")
      dwg.draw.Text (xln3-20,ylp7+45,10,0,"2 { 4aCAM)")
```

```
negativo = TQSDwg.SmartRebar (dwg)                    # Definindo o ferro negativo
negativo.type = TQSDwg.ICPFRT                         # Tipo de ferro - ferro reto
negativo.cover              = cobr                    # Cobrimento p/uso geral
negativo.mark               = 2                       # Numero da posição
negativo.quantity           = qneg                    # Num ferros
negativo.multiplier         = 1                       # Multipl do numero de ferros
negativo.diameter           = bitneg                  # Bitola mm
negativo.straightBarMainLength   = vao-2*cobr         # Comprimento horizontal (cm)
negativo.straightBarLeftLength   = dn                 # Dobra a esquerda
negativo.straightBarRightLength  = dn                 # Dobra a direita
negativo.straightBarLeftLength2  = 20                 # 2a dobra a esquerda
negativo.straightBarRightLength2 = 20                 # 2a dobra a direita
negativo.straightBarZone        = TQSDwg.ICPNEG       # Tipo de ferro reto positivo ou negativo
negativo.RebarTextDisplayPosition (xln1+15,yln1-30,10,0,0,1,0)      # Texto do ferro
if qneg == 4:
   dwg.draw.Text (xln3-40,yln1-58,10,0,"(2 { 2aCAM)")
if qneg == 6:
   dwg.draw.Text (xln3-75,yln1-58,10,0,"(2 { 2aCAM,2 { 3aCAM)")
if qneg == 8:
   dwg.draw.Text (xln3-75,yln1-58,10,0,"(2 { 2aCAM,2 { 3aCAM,")
   dwg.draw.Text (xln3-25,yln1-73,10,0,"2 { 4aCAM)")

pele = TQSDwg.SmartRebar (dwg)                        # Definindo o ferro de pele
pele.type = TQSDwg.ICPFRT                             # Tipo de ferro - ferro reto
pele.cover              = cobr                        # Cobrimento p/uso geral
pele.mark               = 3                           # Numero da posição
pele.quantity           = qpele                       # Num ferros
pele.multiplier         = 2                           # Multipl do numero de ferros
pele.diameter           = bitpele                     # Bitola mm
pele.spacing            = espele                      # Espaçamento
pele.straightBarMainLength   = vao-2*cobr             # Comprimento horizontal (cm)
pele.straightBarLeftLength   = dpe                    # Dobra a esquerda
pele.straightBarRightLength  = dpe                    # Dobra a direita
pele.straightBarLeftLength2  = 20                     # 2a dobra a esquerda
pele.straightBarRightLength2 = 20                     # 2a dobra a direita
pele.straightBarZone        = TQSDwg.ICPPOS           # Tipo de ferro reto positivo ou negativo
pele.RebarTextDisplayPosition (xlpe3,ylpe1-30,10,0,0,0,1)     # Texto do ferro
pele.RebarMarkIdentify      (1,0-40,altura,0,1,0,0)   # Texto de identificação

if tipoestribo == 0:

   estribo = TQSDwg.SmartRebar (dwg)                  # Definindo o ferro estribo aberto
   estribo.type = TQSDwg.ICPFRT                       # Tipo de ferro - ferro reto
   estribo.cover            = cobr                    # Cobrimento p/uso geral
   estribo.mark             = 4                       # Numero da posição
   estribo.quantity         = int(vest/espest)        # Num ferros
   estribo.multiplier       = 2                       # Multipl do numero de ferros
   estribo.diameter         = bitest                  # Bitola mm
   estribo.spacing          = espest                  # Espaçamento
   estribo.straightBarMainLength   = altura-cobr*2    # Comprimento horizontal (cm)
   estribo.straightBarLeftLength   = largura-cobr*2   # Dobra a esquerda
   estribo.straightBarRightLength  = largura-cobr*2   # Dobra a direita
   estribo.straightBarLeftLength2  = 20               # 2a dobra a esquerda
   estribo.straightBarRightLength2 = 20               # 2a dobra a direita
   estribo.straightBarTextDirection = 0               # Direção do texto das dobras
   estribo.straightBarZone        = TQSDwg.ICPNEG     # Tipo de ferro reto positivo ou
negativo
   estribo.RebarTextDisplayPosition (largura*2+565,altura,10,90,0,0,0)     # Texto do ferro
   estribo.RebarLine          (largura*2+550,cobr*2,90,2,1,1,1,0,-1,-1,-1)  # Linha de ferro
```

```
else:

    estribo                         = TQSDwg.SmartRebar (dwg)                  # Definindo o ferro estribo fechado
    estribo.type                    = TQSDwg.ICPSTR                            # Estribo
    estribo.mark                    = 4                                        # Numero da posição
    estribo.quantity                = int(vest/espest)                         # Num ferros
    estribo.diameter                = bitest                                   # Bitola mm
    estribo.spacing                 = espest                                   # Espaçamento
    estribo.cover                   = cobr                                     # Cobrimento p/uso geral
    inivel                          = -1                                       # Nivel defaultp
    iestilo                         = -1                                       # Estilo default
    icor                            = -1                                       # Cor default
    estribo.stirrupType             = 0                                        # Tipo de estribo
    estribo.stirrupLegs             = 0                                        # Numero de ramos
    estribo.stirrupSectionWidth     = largura                                  # Base da seção
    estribo.stirrupSectionHeight    = altura                                   # Altura da seção
    estribo.stirrupSectionWidth2    = 0                                        # Base max seção var de min/max
    estribo.stirrupSectionHeight2   = 0                                        # Altura max idem
    estribo.stirrupEffectiveLeftWidth  = 0                                     # Largura colab a esquerda
    estribo.stirrupEffectiveRightWidth = 0                                     # Largura colab a direita
    estribo.stirrupSlabBendLength   = 0                                        # Dobra do estribo na laje
    estribo.RebarTextDisplayPosition  (largura*2+570,y1s-15,10,0,0,0,1)
    estribo.RebarLine               (largura*2+550,y1s,0,2,1,1,1,0,-1,-1,-1)   # Linha de ferro

if mislie == "S" and mislid == "N":

    c1 = cobr*4*math.cos(angrad)+hmislie*2                                     # cateto interno
    c2 = c1**2+c1**2                                                           # soma dos quadrados do cateto interno
    c3 = math.sqrt(c2)                                                         # hipotenusa interno
    c4 = largura*2-cobr*2-cobr*4*math.cos(angrad)                              # cateto largura da parede
    c5 = c4**2+c4**2                                                           # soma dos quadrados do cateto da parede
    c6 = math.sqrt(c5)                                                         # hipotenusa da parede
    c7 = hlie*2-cobr*2-cobr*4*math.cos(angrad)                                 # cateto da laje
    c8 = c7**2+c7**2                                                           # soma dos quadrados do cateto da laje
    c9 = math.sqrt(c8)                                                         # hipotenusa da laje
    ctotal = int(c3+c6+c9)                                                     # comprimento da misula na seção
    ctotal2 = int(ctotal/2)                                                    # comprimento real
    d1 = 20
    d2 = 20

    misulae = TQSDwg.SmartRebar (dwg)                                          # Definindo o ferro
estribo aberto
    misulae.type = TQSDwg.ICPFRT                                               # Tipo de ferro - ferro
reto
    misulae.cover                   = cobr                                     # Cobrimento p/uso geral
    misulae.mark                    = 5                                        # Numero da posição
    misulae.quantity                = int(vest/espmislie)                      # Num ferros
    misulae.multiplier              = 1                                        # Multipl do numero de ferros
    misulae.diameter                = bitmislie                                # Bitola mm
    misulae.spacing                 = espmislie                                # Espaçamento
    misulae.straightBarMainLength   = ctotal2                                  # Comprimento
horizontal (cm)
    misulae.straightBarLeftLength   = 20                                       # Dobra a esquerda
    misulae.straightBarRightLength  = 20                                       # Dobra a direita
    misulae.straightBarTextDirection = 0                                       # Direção do texto das
dobras
    misulae.straightBarZone         = TQSDwg.ICPPOS                            # Tipo de ferro reto
positivo ou negativo
```

```
   misulae.RebarTextDisplayPosition  (largura*2+320+70+ctotal2*math.cos(angrad),ylie1+35,10,45,0,0,1)
# Texto do ferro
   misulae.RebarLine              (largura*2+320+55,ylie1+cobr*2,45,2,1,1,0,0,-1,-1,-1)                  # Linha de
ferro
   misulae.RebarMarkIdentify        (1,xame5,yame5+5,0,1,0,0)                                            # Texto de
identificação

   dwg.draw.level                 = 220
# Nivel do desenho
   dwg.draw.Line                  (largura*2+320+55,ylie1+cobr*2,largura*2+320+15,ylie1+cobr*2)
# Desenho das linhas das dobras
   dwg.draw.Line
(largura*2+320+55+(ctotal*math.cos(angrad)),ylie1+cobr*2+(ctotal*math.cos(angrad)),largura*2+320+55+(ctot
al*math.cos(angrad)),ylie1+cobr*2+(ctotal*math.cos(angrad))+40)
   dwg.draw.level                 = 224
# Nivel do desenho
   dwg.draw.Text
(largura*2+320+45+ctotal2*math.cos(angrad),ylie1+cobr*2+ctotal2*math.cos(angrad),10,45,str(ctotal2))
# Textos de identificação das dobras
   dwg.draw.Text                  (largura*2+320+25,ylie1-15,10,0,str(d1))
   dwg.draw.Text
(largura*2+320+70+ctotal*math.cos(angrad),ylie1+cobr*2+ctotal*math.cos(angrad)+10,10,90,str(d2))
   if deslie != 0:
      dwg.draw.level              = 204
      dwg.draw.Line               (largura*2+330,ylie1,largura*2+330+55+ctotal*math.cos(angrad),ylie1)       #
linha de chamada

if mislie == "N" and mislid == "S":

   c1 = cobr*4*math.cos(angrad)+hmislid*2               # cateto interno
   c2 = c1**2+c1**2                                    # soma dos quadrados do cateto interno
   c3 = math.sqrt(c2)                                  # hipotenusa interno
   c4 = largura*2-cobr*2-cobr*4*math.cos(angrad)         # cateto largura da parede
   c5 = c4**2+c4**2                                    # soma dos quadrados do cateto da parede
   c6 = math.sqrt(c5)                                  # hipotenusa da parede
   c7 = hlid*2-cobr*2-cobr*4*math.cos(angrad)           # cateto da laje
   c8 = c7**2+c7**2                                    # soma dos quadrados do cateto da laje
   c9 = math.sqrt(c8)                                  # hipotenusa da laje
   ctotal = int(c3+c6+c9)                              # comprimento da misula na seção
   ctotal2 = int(ctotal/2)                             # comprimento real
   d1 = 20                                            # dobras misula
   d2 = 20

   misulae = TQSDwg.SmartRebar (dwg)                                                                  # Definindo o
ferro estribo aberto
   misulae.type = TQSDwg.ICPFRT                                                                       # Tipo de ferro -
ferro reto
   misulae.cover                  = cobr                                                            # Cobrimento p/uso
geral
   misulae.mark                   = 5                                                              # Numero da posição
   misulae.quantity               = int(vest/espmislid)                                                 # Num ferros
   misulae.multiplier             = 1                                                              # Multipl do numero de
ferros
   misulae.diameter               = bitmislid                                                       # Bitola mm
   misulae.spacing                = espmislid                                                       # Espaçamento
   misulae.straightBarMainLength     = ctotal2                                                          # Comprimento
horizontal (cm)
   misulae.straightBarLeftLength     = 20                                                           # Dobra a
esquerda
   misulae.straightBarRightLength    = 20                                                            # Dobra a direita
```

```
   misulae.straightBarTextDirection  = 0                                                    # Direção do texto
das dobras
   misulae.straightBarZone          = TQSDwg.ICPPOS                                          # Tipo de
ferro reto positivo ou negativo
   misulae.RebarTextDisplayPosition  (largura*2+290+45+ctotal2*math.cos(angrad),ylid1+35,10,-45,0,0,1)
# Texto do ferro
   misulae.RebarLine                (largura*2+290+55,ylid1+cobr*2+ctotal*math.cos(angrad),-45,2,1,1,0,0,-1,-1,-
1)          # Linha de ferro
   misulae.RebarMarkIdentify        (1,xamd5,yamd5+5,0,1,0,0)                                #
Texto de identificação

   dwg.draw.level                   = 220                                                    #
Nivel do desenho
   dwg.draw.Line
(largura*2+290+55+ctotal*math.cos(angrad),ylid1+cobr*2,largura*2+290+95+ctotal*math.cos(angrad),ylid1+c
obr*2)             # Desenho das linhas das dobras
   dwg.draw.Line
(largura*2+290+55,ylid1+cobr*2+(ctotal*math.cos(angrad)),largura*2+290+55,ylid1+cobr*2+(ctotal*math.cos(
angrad))+40)
   dwg.draw.level                   = 224                                                    #
Nivel do desenho
   dwg.draw.Text
(largura*2+290+50+ctotal2*math.cos(angrad),ylid1+cobr*2+ctotal2*math.cos(angrad)+15,10,-45,str(ctotal2))
# Textos de identificação das dobras
   dwg.draw.Text                    (largura*2+290+65+ctotal*math.cos(angrad),ylid1-15,10,0,str(d1))
   dwg.draw.Text                    (largura*2+290+50,ylid1+cobr*2+ctotal*math.cos(angrad)+10,10,90,str(d2))
   if deslid != 0:
      dwg.draw.level                = 204
      dwg.draw.Line                 (largura*2+330,ylie1,largura*2+330+55+ctotal*math.cos(angrad),ylie1)       #
linha de chamada

if mislie == "S" and mislid == "S":

   c1 = cobr*4*math.cos(angrad)+hmislie*2              # cateto interno
   c2 = c1**2+c1**2                                    # soma dos quadrados do cateto interno
   c3 = math.sqrt(c2)                                  # hipotenusa interno
   c4 = largura*2-cobr*2-cobr*4*math.cos(angrad)       # cateto largura da parede
   c5 = c4**2+c4**2                                    # soma dos quadrados do cateto da parede
   c6 = math.sqrt(c5)                                  # hipotenusa da parede
   c7 = hlie*2-cobr*2-cobr*4*math.cos(angrad)          # cateto da laje
   c8 = c7**2+c7**2                                    # soma dos quadrados do cateto da laje
   c9 = math.sqrt(c8)                                  # hipotenusa da laje
   ctotal = int(c3+c6+c9)                              # comprimento da misula na seção
   ctotal2 = int(ctotal/2)                             # comprimento real
   d1 = 20                                          # dobras misula
   d2 = 20

   misulae = TQSDwg.SmartRebar (dwg)                                                        # Definindo o ferro
estribo aberto
   misulae.type = TQSDwg.ICPFRT                                                             # Tipo de ferro - ferro
reto
   misulae.cover                = cobr                                                    # Cobrimento p/ uso geral
   misulae.mark                 = 5                                                       # Numero da posição
   misulae.quantity             = int(vest/espmislie)                                     # Num ferros
   misulae.multiplier           = 2                                                       # Multipl do numero de ferros
   misulae.diameter             = bitmislie                                               # Bitola mm
   misulae.spacing              = espmislie                                               # Espaçamento
   misulae.straightBarMainLength    = ctotal2                                             # Comprimento
horizontal (cm)
   misulae.straightBarLeftLength    = 20                                                  # Dobra a esquerda
```

```
   misulae.straightBarRightLength    = 20                                          # Dobra a direita
   misulae.straightBarTextDirection  = 0                                           # Direção do texto das
dobras
   misulae.straightBarZone           = TQSDwg.ICPPOS                               # Tipo de ferro reto
positivo ou negativo
   misulae.RebarTextDisplayPosition  (largura*2+320+70+ctotal2*math.cos(angrad),ylie1+35,10,45,0,0,1)
# Texto do ferro
   misulae.RebarLine                 (largura*2+320+55,ylie1+cobr*2,45,2,1,1,0,0,-1,-1,-1)        # Linha de
ferro
   misulae.RebarMarkIdentify         (1,xame5,yame5+5,0,1,0,0)                     # Texto de
identificação
   misulae.RebarMarkIdentify         (1,xamd5,yamd5+5,0,1,0,0)                     # Texto de
identificação

   dwg.draw.level                    = 220
# Nivel do desenho
   dwg.draw.Line                     (largura*2+320+55,ylie1+cobr*2,largura*2+320+15,ylie1+cobr*2)
# Desenho das linhas das dobras
   dwg.draw.Line
(largura*2+320+55+(ctotal*math.cos(angrad)),ylie1+cobr*2+(ctotal*math.cos(angrad)),largura*2+320+55+(ctot
al*math.cos(angrad)),ylie1+cobr*2+(ctotal*math.cos(angrad))+40)
   dwg.draw.level                    = 224
# Nivel do desenho
   dwg.draw.Text
(largura*2+320+45+ctotal2*math.cos(angrad),ylie1+cobr*2+ctotal2*math.cos(angrad),10,45,str(ctotal2))
# Textos de identificação das dobras
   dwg.draw.Text                     (largura*2+320+25,ylie1-15,10,0,str(d1))
   dwg.draw.Text
(largura*2+320+70+ctotal*math.cos(angrad),ylie1+cobr*2+ctotal*math.cos(angrad)+10,10,90,str(d2))
   if deslie != 0 or deslid != 0:
      dwg.draw.level                 = 204
      dwg.draw.Line                  (largura*2+330,ylie1,largura*2+330+55+ctotal*math.cos(angrad),ylie1)      #
linha de chamada

# armadura contra empuxo ao vazio

if lie == "S" and aeve == "S":
   dwg.draw.level     = 224          # Nivel do desenho
   # desenho da linha de representação na seção
   xeve1 = 0+cobr
   yeve1 = xeve1
   xeve2 = xeve1+largura-cobr*2
   yeve2 = yeve1
   xeve3 = xeve2
   yeve3 = deslie+hlie-cobr
   xeve4 = xeve3-aevep
   yeve4 = yeve3
   xeve5 = xeve4
   yeve5 = deslie+cobr
   xeve6 = xeve1
   yeve6 = yeve1+aevep+deslie
   xeve7 = xeve2
   yeve7 = yeve6
   dwg.draw.Line   (xeve1*2,yeve1*2,xeve2*2,yeve2*2)
   dwg.draw.Line   (xeve2*2,yeve2*2,xeve3*2,yeve3*2)
   dwg.draw.Line   (xeve3*2,yeve3*2,xeve4*2,yeve4*2)
   dwg.draw.Line   (xeve4*2,yeve4*2,xeve5*2,yeve5*2)
   dwg.draw.Line   (xeve1*2,yeve1*2,xeve6*2,yeve6*2)
   dwg.draw.Line   (xeve6*2,yeve6*2,xeve7*2,yeve7*2)
```

```
   aeve = TQSDwg.SmartRebar (dwg)                                    # Definindo o ferro armadura contra
empuxo ao vazio esquerda
   aeve.type = TQSDwg.ICPFGN                                         # Tipo de ferro - ferro genérico
   aeve.cover                = cobr                                  # Cobrimento p/uso geral
   aeve.mark                 = 6                                     # Numero da posição
   aeve.quantity             = int(vest/espaeve)                     # Num ferros
   aeve.multiplier           = 1                                     # Multipl do numero de ferros
   aeve.diameter             = bitaeve                               # Bitola mm
   aeve.spacing              = espaeve                               # Espaçamento
   aeve.straightBarTextDirection = 0                                 # Direção do texto das dobras
   aeve.straightBarZone      = TQSDwg.ICPNEG                         # Tipo de ferro reto positivo ou
negativo
   aeve.bendBendLengthMode   = TQSDwg.ICPDOBSMS                      # Calculo das dobras em
Comprimento do trecho para corrigir comprimentos
   aeve.bendTotalLengthMode  = TQSDwg.ICPSMS                         # Calculo de comprimento total
em Soma simples para corrigir comprimentos
   aeve.GenRebarPoint        (xeve5,yeve5,0,0,1,0)                   # Pontos para o ferro generico
   aeve.GenRebarPoint        (xeve4,yeve4,0,0,1,0)
   aeve.GenRebarPoint        (xeve3,yeve3,0,0,1,0)
   aeve.GenRebarPoint        (xeve2,yeve2,0,0,1,0)
   aeve.GenRebarPoint        (xeve1,yeve1,0,0,1,0)
   aeve.GenRebarPoint        (xeve6,yeve6,0,0,1,0)
   aeve.GenRebarPoint        (xeve7,yeve7,0,0,1,0)
   aeve.RebarTextDisplayPosition (xlie4-70-aevep/1.5,yeve3,10,0,0,0,1)   # Texto do ferro
   aeve.RebarLine            (xlie4-70,y1s,0,2,1,1,1,0,-1,-1,-1)     # Linha de ferro
   dwg.draw.level            = 225                                   # Nivel da linha de chamada
   xieve1 = x2s-cobr*2-aevep*2+20
   yieve1 = ylie2+hlie*2-cobr*2
   xieve2 = xieve1+5
   yieve2 = yieve1-5
   xieve3 = xieve1-30
   yieve3 = yieve1+30
   xieve4 = xieve1-5
   yieve4 = yieve1-5
   xieve5 = xieve1+5
   yieve5 = yieve1+5
   dwg.draw.Line             (xieve2,yieve2,xieve3,yieve3)           # Desenho da linha de chamada
   dwg.draw.Line             (xieve4,yieve4,xieve5,yieve5)           # Desenho do 'tick'
   aeve.RebarMarkIdentify    (1,xieve3,yieve3+5,0,1,0,0)             # Texto de identificação
   dwg.draw.level            = 204                                   # Nivel da linha de referencia
   dwg.draw.Line             (xlie2-10,y1s,xlie2-50-aevep*2-15,y1s)  # Linha de referencia

if lid == "S" and aevd == "S":
   dwg.draw.level    = 224       # Nivel do desenho
   # desenho da linha de representação na seção
   xevd1 = x1s+largura-cobr
   yevd1 = y2s+cobr
   xevd2 = xevd1-largura+cobr*2
   yevd2 = yevd1
   xevd3 = xevd2
   yevd3 = deslid+hlid-cobr
   xevd4 = xevd3+aevdp
   yevd4 = yevd3
   xevd5 = xevd4
   yevd5 = deslid+cobr
   xevd6 = xevd1
   yevd6 = yevd1+aevdp+deslid
   xevd7 = xevd2
   yevd7 = yevd6
   dwg.draw.Line   (xevd1*2,yevd1*2,xevd2*2,yevd2*2)
```

```
   dwg.draw.Line   (xevd2*2,yevd2*2,xevd3*2,yevd3*2)
   dwg.draw.Line   (xevd3*2,yevd3*2,xevd4*2,yevd4*2)
   dwg.draw.Line   (xevd4*2,yevd4*2,xevd5*2,yevd5*2)
   dwg.draw.Line   (xevd1*2,yevd1*2,xevd6*2,yevd6*2)
   dwg.draw.Line   (xevd6*2,yevd6*2,xevd7*2,yevd7*2)
   aevd = TQSDwg.SmartRebar (dwg)                                  # Definindo o ferro armadura contra
empuxo ao vazio esquerda
   aevd.type = TQSDwg.ICPFGN                                  # Tipo de ferro - ferro genérico
   aevd.cover                = cobr                          # Cobrimento p/uso geral
   aevd.mark                 = 7                             # Numero da posição
   aevd.quantity              = int(vest/espaevd)                  # Num ferros
   aevd.multiplier            = 1                             # Multipl do numero de ferros
   aevd.diameter              = bitaevd                          # Bitola mm
   aevd.spacing               = espaevd                          # Espaçamento
   aevd.straightBarTextDirection = 0                             # Direção do texto das dobras
   aevd.straightBarZone         = TQSDwg.ICPNEG                        # Tipo de ferro reto positivo ou
negativo
   aevd.bendBendLengthMode      = TQSDwg.ICPDOBSMS                        # Calculo das dobras em
Comprimento do trecho para corrigir comprimentos
   aevd.bendTotalLengthMode     = TQSDwg.ICPSMS                        # Calculo de comprimento total
em Soma simples para corrigir comprimentos
   aevd.GenRebarPoint          (xevd5,yevd5,0,0,1,0)                 # Pontos para o ferro generico
   aevd.GenRebarPoint          (xevd4,yevd4,0,0,1,0)
   aevd.GenRebarPoint          (xevd3,yevd3,0,0,1,0)
   aevd.GenRebarPoint          (xevd2,yevd2,0,0,1,0)
   aevd.GenRebarPoint          (xevd1,yevd1,0,0,1,0)
   aevd.GenRebarPoint          (xevd6,yevd6,0,0,1,0)
   aevd.GenRebarPoint          (xevd7,yevd7,0,0,1,0)
   aevd.RebarTextDisplayPosition (largura*4+500+100+aevdp,yevd3,10,0,0,0,1)      # Texto do ferro
   aevd.RebarLine              (largura*4+500+100,y1s,0,2,1,1,1,0,-1,-1,-1)       # Linha de ferro
   dwg.draw.level             = 225                          # Nivel da linha de chamada
   xievd1 = x1s+cobr*2+aevdp*2-20
   yievd1 = ylid2+hlid*2-cobr*2
   xievd2 = xievd1-5
   yievd2 = yievd1-5
   xievd3 = xievd1+30
   yievd3 = yievd1+30
   xievd4 = xievd1+5
   yievd4 = yievd1-5
   xievd5 = xievd1-5
   yievd5 = yievd1+5
   dwg.draw.Line              (xievd2,yievd2,xievd3,yievd3)                  # Desenho da linha de
chamada
   dwg.draw.Line              (xievd4,yievd4,xievd5,yievd5)                  # Desenho do 'tick'
   aevd.RebarMarkIdentify       (1,xievd3,yievd3+5,0,1,0,0)                    # Texto de identificação
   dwg.draw.level             = 204                          # Nivel da linha de referencia
   dwg.draw.Line              (largura*2+550+largura*2+5,0,largura*2+550+largura*2+5+aevdp*2+65,0)    #
Linha de referencia

# desenho das linhas de chamada para o estribo
dwg.draw.level = 204
dwg.draw.Line   (largura*2+330,0,largura*2+550+largura*2+5,0)
dwg.draw.Line   (largura*2+330,altura*2,largura*2+550+largura*2+5,altura*2)

dwg.globalrebar.RenumerateMarks()

# limpar blocos e niveis
dwg.file.PurgeBlocks()
dwg.file.PurgeLevels()
```

```
def escolher_pasta():
    # Inicializa o tkinter
    root = tk.Tk()
    root.withdraw()  # Oculta a janela principal

    # Força a janela a estar no topo
    root.attributes("-topmost", True)

    # Abre o diálogo para seleção da pasta
    caminho_pasta = filedialog.askdirectory(title="Selecione uma pasta")

    # Fecha a janela principal
    root.destroy()

    if caminho_pasta:
        print(f"Caminho selecionado: {caminho_pasta}")
        return caminho_pasta
    else:
        print("Nenhuma pasta foi selecionada.")
        return None

# Chamando a função
caminho_salvo = escolher_pasta()

#if caminho_salvo:
    # Use o caminho salvo para outras operações
    #print(f"Caminho salvo: {caminho_salvo}")

#salvar o arquivo
dwg.file.SaveAs (caminho_salvo + "/" + nomedwg + ".dwg")

#abrir o arquivo após o salvamento
os.startfile(caminho_salvo + "/" + nomedwg + ".dwg")

system('cls')
```