



UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA CIVIL



TRABALHO DE CONCLUSÃO DE CURSO

DESENVOLVIMENTO DE PROGRAMA DE DIMENSIONAMENTO DE SEÇÃO TRANSVERSAL DE VIGAS DE CONCRETO PROTENDIDO COM ADERÊNCIA INICIAL

Victor Carvalho Lima

Uberlândia
2019



UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA CIVIL



Victor Carvalho Lima

DESENVOLVIMENTO DE PROGRAMA DE DIMENSIONAMENTO
DE SEÇÃO TRANSVERSAL DE VIGAS DE CONCRETO PROTENDIDO COM
ADERÊNCIA INICIAL

Trabalho de Conclusão de Curso apresentado à
Faculdade de Engenharia Civil da
Universidade Federal de Uberlândia como
parte dos requisitos para a conclusão da
graduação em Engenharia Civil

Orientador: Arquimedes Diógenes Ciloni

Uberlândia

2019

Agradecimentos

Agradeço à minha família, que sempre se dedicou a me proporcionar as melhores oportunidades de estudo possíveis e me apoiar até chegar aonde cheguei.

Agradeço aos meus amigos por estarem sempre comigo nas horas difíceis e nas horas muito difíceis da faculdade.

Agradeço aos meus professores, que me instruíram ao longo do curso e possibilitaram minha formação profissional na área de engenharia civil.

Resumo

Este trabalho teve como objetivo a elaboração de um programa de dimensionamento de seção transversal de vigas de concreto protendido com aderência inicial, utilizando o roteiro de cálculo proposto por Hanai (2005). O programa foi desenvolvido com a linguagem de programação C#, com o auxílio do *software* Visual Studio, da Microsoft. Foram implementadas no programa as etapas de inserção dos dados iniciais do concreto, aço e da seção transversal, o cálculo de tensões atuantes na viga, o cálculo das armaduras ativas e das perdas da força de protensão. O programa é capaz de calcular as áreas da armadura de protensão, as forças efetivas e as perdas na vida útil da viga, mas não são realizadas as verificações necessárias das seções ao longo do elemento.

Palavras-chave: concreto, programa, protensão

Abstract:

This study aimed the elaboration of a software for projecting cross sections of prestressed concrete beams with initial adherence, using the routine proposed by Hanai (2005). The software was developed using the programming language C#, with the integrated development environment Microsoft Visual Studio. In the software, were implemented functions for insertion of basic beam data, such as concrete and steel material properties, cross-section, stresses applied to the beam, prestressed rebar calculations and rebar tension loss. The program is capable of calculating the area of the prestressed bars, the prestress forces and the tension losses through the lifespan of the beam, but the verifications of the cross sections along the beam aren't calculated.

Keywords: concrete, software, prestress

Lista de figuras:

Figura 1 - Pista de protensão	5
Figura 2 - Ancoragem da armadura de uma viga protendida.....	6
Figura 3 - Cabos de protensão curvos	6
Figura 4 - Tipos de ondulações que causam perdas por atrito	8
Figura 5 - Formulário gerado no Visual Studio	9
Figura 6 - Ambiente de trabalho do Microsoft Visual Studio.....	9
Figura 7 - Tela para inserção dos dados dos materiais da viga	20
Figura 8 - Exemplo de inserção de seção retangular no programa	21
Figura 9 - Exemplo de seção T no programa	22
Figura 10 - Exemplo de seção duplo T no programa	22
Figura 11 - Exemplo de seção I no programa	22
Figura 12 - Tela para inserção dos dados geométricos da viga.....	23
Figura 13 - Tipos de cargas acidentais disponíveis para seleção	23
Figura 14 – Tela para inserção dos carregamentos e cálculo das forças de protensão.	25
Figura 15 - Tela para inserção da armadura de protensão efetiva.....	26
Figura 16 – Quadro para inserção dos dados iniciais para cálculo de perdas.....	27
Figura 17 – Quadro dos dados relativos às perdas por retração do concreto.....	27
Figura 18 – Quadro dos dados relativos às perdas por fluência do concreto.....	28
Figura 19 - Tela para cálculo das perdas de protensão.....	28
Figura 20 – Dados iniciais da viga de exemplo	29
Figura 21 – Dados geométricos da viga de exemplo.....	30
Figura 22 – Dados das cargas atuantes sobre a viga de exemplo.....	31

Figura 23 – Dados das armaduras efetivas da viga de exemplo	32
Figura 24 – Dados das perdas de protensão da viga de exemplo	33

Lista de tabelas:

Tabela 1 - Coeficientes Ψ para cargas acidentais	14
Tabela 2 – Coeficientes γ, ϕ e ϵ para cálculo de perdas de protensão	18
Tabela 3 - Coeficientes Ψ de minoração estatística de cargas acidentais.....	24

Sumário

1.	Introdução	3
1.1	Objetivo	3
1.2	Método.....	4
1.3	Justificativa	4
2.	Revisão bibliográfica	4
2.1	Métodos de protensão.....	4
2.2	Tipos de protensão	6
2.3	Perdas de protensão	7
3.	Ambiente de desenvolvimento – Visual Studio	8
4.	Roteiro de cálculo	10
4.1	Dados iniciais do concreto.....	10
4.2	Dados iniciais da armadura ativa	10
4.3	Dados da seção transversal	11
4.4	Carregamentos	14
4.5	Forças de protensão estimadas.....	15
4.6	Definição das armaduras	16
4.7	Cálculo de perdas	17
5.	Utilização do programa.....	20
5.1	Definição dos materiais	20
5.2	Geometria da peça.....	21
5.3	Carregamentos e ações	23

5.4	Definição da armadura	25
5.5	Cálculo de perdas	26
6.	Exemplo de utilização do programa	29
6.1	Dados iniciais.....	29
6.2	Seção transversal.....	29
6.3	Esforços e forças estimadas	30
6.4	Área de aço efetiva e novas forças de protensão	31
6.5	Cálculo de perdas	32
7.	Conclusão	33
8.	Sugestões para futuros trabalhos	34
	Referências.....	35
	ANEXO A – Código fonte do programa	36

1. INTRODUÇÃO

Nos elementos estruturais de concreto armado, emprega-se apenas o uso de armaduras de aço passivas. Assim, numa seção transversal de um elemento do tipo sob flexão simples, por exemplo, parte da área da seção é submetida a tensões de compressão e outra parte a tensões de tração.

Como muitas vezes a tensão de tração ao qual o concreto é submetido é maior que a sua resistência a este esforço, a peça de concreto armado normalmente apresenta fissuras que devem ser verificadas durante o dimensionamento.

Dessa forma, peças de concreto armado simples não são indicadas para vencer grandes vãos; a dimensão e o custo de peças de concreto armado seriam excessivamente altos e impraticáveis, como no caso de vãos livres de mais de 10 metros, em projetos de pontes, por exemplo.

A solução para esta limitação é adotar armaduras ativas nas peças de concreto. Armaduras ativas são inicialmente tracionadas, imprimindo à peça de concreto tensões de compressão, mesmo antes da aplicação de cargas no elemento. Uma peça de concreto submetida à flexão simples possui, normalmente, tensões de compressão e tração. Com a protensão, estas tensões de tração são combatidas pela compressão prévia da peça.

Assim, é possível que uma peça de concreto protendido tenha tensões de tração inferiores à resistência à tração do concreto ou sequer tenha tensões do tipo, mesmo que a peça vença grandes vãos e tenha uma seção transversal com área reduzida. Além disso, é possível eliminar a fissuração da peça, garantindo mais proteção contra a corrosão da armadura.

1.1 Objetivo

O objetivo deste trabalho é desenvolver um programa que possibilite o dimensionamento e detalhamento de vigas de concreto protendido com protensão inicial, com as verificações exigidas pela ABNT NBR 6118:2014.

1.2 Método

O programa será desenvolvido usando o ambiente de trabalho Visual Studio, um programa criado pela Microsoft, voltado para o desenvolvimento de aplicações, tanto para sistemas Windows quanto dispositivos móveis e páginas da web, em diversas linguagens de programação, como C#, C++, Visual Basic e Python.

Será desenvolvido um programa do tipo formulários do Windows, usando a linguagem C#, e que pode ser instalado e executado em qualquer computador com sistema operacional Windows.

1.3 Justificativa

O setor de construção civil no Brasil passa por um lento e contínuo processo de modernização. Muitas edificações são construídas usando métodos “industriais” de construção, como a utilização de peças pré-moldadas e pré-fabricadas. Estes métodos construtivos aumentam a eficiência nas obras e reduzem o tempo em construção, bem como reduzem o desperdício de materiais na obra.

Com a crescente utilização de peças pré-moldadas e pré-fabricadas, aumenta-se, conseqüentemente, o uso de peças com aço protendido, característico desse tipo de elemento.

Devido ao aumento da demanda pela execução de obras com concreto protendido, torna-se conveniente ofertar novos softwares de cálculo de peças protendidas de forma a popularizar e reduzir os custos deste método, visto que há pouca oferta de programas de cálculo de armaduras de concreto protendido no mercado.

2. REVISÃO BIBLIOGRÁFICA

2.1 Métodos de protensão

Um elemento de concreto pode ser protendido antes ou depois do processo de concretagem. O primeiro método é denominado pré-tração enquanto o segundo é chamado de pós-tração.

No método da pré-tração, os cabos de aço são posicionados sobre uma forma. Os cabos são tensionados por um macaco hidráulico e, enquanto são submetidos a essa tração,

realiza-se a concretagem da peça. Após o endurecimento do concreto, retiram-se os macacos hidráulicos e as armaduras ativas imprimem uma tensão de compressão sobre o concreto. Normalmente, a forma consiste em uma longa pista de concretagem, podendo chegar até cerca de 200 metros de comprimento. Após certo tempo de endurecimento do concreto, podem-se cortar elementos protendidos com o comprimento desejado.

Figura 1 - Pista de protensão



Fonte: Maia, 2018

Como este método é feito antes da concretagem, só é possível obter armaduras de protensão com trechos retilíneos. Normalmente, as armaduras pré-tracionadas têm excentricidade constante ao longo de todo o elemento estrutural, como no caso de pré-moldados em pistas de protensão, mas é possível variar a direção da armadura usando-se cabos poligonais.

Este método é comumente empregado em fábricas de elementos pré-moldados padronizados, como vigotas treliçadas.

Já num elemento pós-tracionado, antes da concretagem, são colocadas ao longo do comprimento do elemento bainhas com os cabos de protensão. A peça é concretada. Após o endurecimento do concreto, usa-se um macaco hidráulico para tracionar as armaduras. Com as armaduras tensionadas, injeta-se uma pasta de cimento dentro da bainha, preenchendo o espaço interno desta. Depois de endurecida a pasta, retiram-se os macacos hidráulicos e os cabos passam a comprimir o concreto.

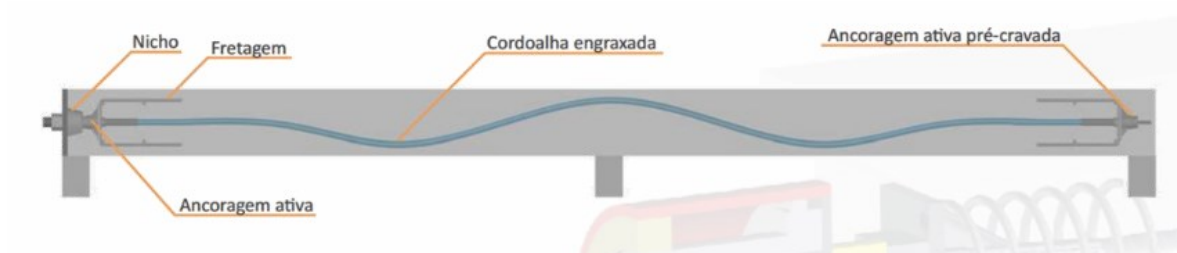
Com este método, a excentricidade dos cabos de protensão pode ser variável, permitindo o uso de cabos curvos que combatem as tensões de tração onde houver necessidade, a saber, na parte inferior do elemento no meio dos vãos e na parte superior quando próximo aos apoios de vigas com mais de dois apoios.

Figura 2 - Ancoragem da armadura de uma viga protendida



Fonte: Full Estruturas, 2016

Figura 3 - Cabos de protensão curvos



Fonte: Lobato, [20--]

2.2 Tipos de protensão

Para efetuar a protensão de um elemento estrutural, deve-se considerar alguns fatores, sendo o potencial corrosivo da armadura um dos principais a serem levados em conta.

No método de protensão parcial, admite-se a abertura de fissuras até 0,2mm, segundo a ABNT NBR 6118:2014.

No método de protensão limitada, considera-se que o elemento, em condição de serviço, sofra tensões de tração não nulas no concreto e abaixo do seu limite de resistência a este tipo de esforço, de forma a não permitir a abertura de fissuras.

Na protensão completa, não é admitida qualquer abertura de fissura no concreto e, portanto, os esforços de tração não podem ser observados no elemento. Utiliza-se este tipo de protensão em casos de ambientes com alta agressividade ambiental, como em locais com variação do nível da maré, e em casos que a abertura de fissuras represente um risco à população e ao meio ambiente, como no caso de reatores nucleares.

2.3 Perdas de protensão

Ao realizar o dimensionamento de uma peça protendida, deve-se verificar as perdas de força dos cabos de protensão que irão ocorrer durante toda a vida útil da peça, desde o desacoplamento do macaco hidráulico das armaduras até as perdas por fluência até o fim do tempo de serviço da peça.

As perdas de protensão podem ocorrer devido à retração do concreto – que ocorre nos primeiros dias, durante a secagem do concreto –, à fluência – devido a deformações plásticas do concreto (HANAI, 2005).

Soma-se a essas perdas a queda de protensão causada pela relaxação e fluência dos cabos de protensão. A perda por relaxação é causada pelo alívio de tensões em um cabo de comprimento mantido constante ao longo do tempo, enquanto a perda por fluência ocorre quando o cabo, submetido a uma tensão constante, aumenta seu comprimento, numa deformação plástica.

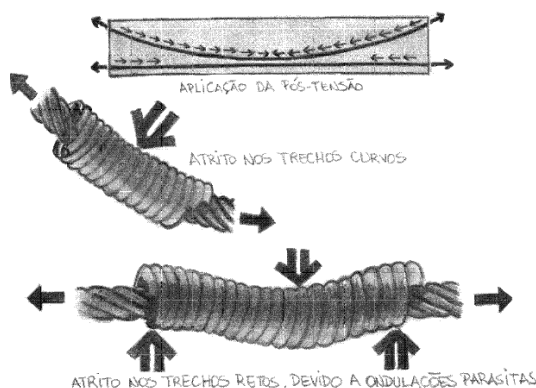
Segundo Hanai (2005), a perda por fluência do aço é de menor relevância no cálculo total das perdas, visto que o comprimento da armadura ativa é praticamente constante durante a vida útil da peça protendida.

No caso de armaduras pós tracionadas, ocorrem perdas de protensão provocadas pelo atrito da armadura com a bainha.

Em armaduras com excentricidade constante, esta perda por atrito deve-se a pequenas irregularidades, chamadas de “ondulações parasitas”, que causam trechos de contato entre o cabo e a bainha.

Já em cabos curvos, além das ondulações parasitas, a curvatura projetada para o cabo incorre em valores adicionais de atrito que devem ser considerados no projeto.

Figura 4 - Tipos de ondulações que causam perdas por atrito



Fonte: HANAI, 2005

Em cabos pré-tracionados, ao se liberar os cabos, a tensão da protensão imprime no concreto uma tensão inicial – esperada – de compressão. Esta tensão causa, de imediato, um encurtamento elástico da peça, e esta acomodação inicial leva à redução da força de protensão.

Além destas perdas, outro fator considerável a se verificar é a perda por acomodação das ancoragens da armadura. Após a liberação das armaduras, pode haver, por exemplo, acomodação das cunhas que restringem o deslocamento dos cabos, provocando um pequeno encurtamento da armadura, mas que causa perdas significativas de força de protensão.

3. AMBIENTE DE DESENVOLVIMENTO – VISUAL STUDIO

Para o desenvolvimento do programa deste trabalho, utilizou-se o programa Visual Studio, da Microsoft.

O Visual Studio oferece uma interface de programação que permite criar, entre outros modelos, aplicações com interfaces gráficas de formulários, chamados de Windows Forms, em que é possível inserir campos de entrada de dados fornecidos pelo usuário e, com base nestes dados, realizar os cálculos desejados.

Figura 5 - Formulário gerado no Visual Studio

Vigas de concreto protendido

Arquivo | Dados Iniciais | Geometria | Forças e carregamentos | Armaduras | Cálculo de perdas | Relatório

Carregamentos (kN/m)

Peso próprio: Carga permanente: Cargas acidentais: Tipo de carga acidental:

ψ_0 ψ_1 ψ_2

Protensão

Tipo de protensão: Excentricidade dos cabos:

Combinações para o estado limite de serviço ELS

Estado Limite de Descompressão - ELS-D: Estado Limite de Formação de fissuras - ELS-F:

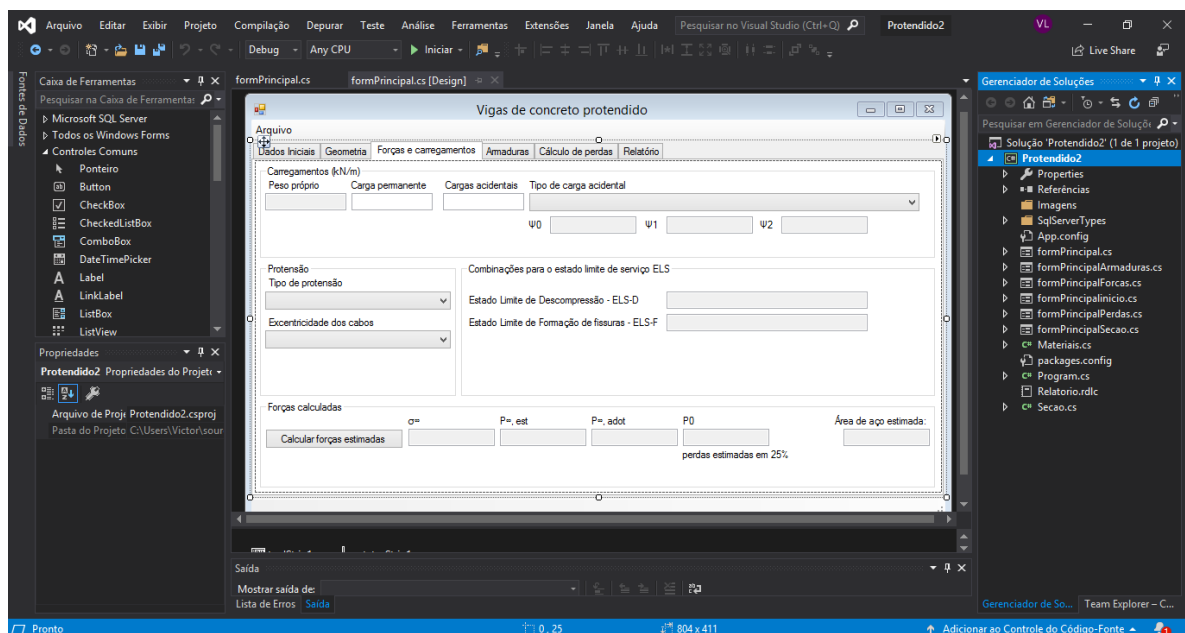
Forças calculadas

σ_{pe} $P_{pe, est}$ $P_{pe, adot}$ P_0 Área de aço estimada:

perdas estimadas em 25%

Fonte: Autor

Figura 6 - Ambiente de trabalho do Microsoft Visual Studio



Fonte: Autor

A linguagem de programação utilizada foi C#, também desenvolvida pela Microsoft, e que possui uma grande gama de aplicações, desde programas simples até sistemas próprios de grandes empresas e desenvolvimento de jogos para computador e consoles de videogames.

A linguagem C# é uma continuação das linguagens C e C++, historicamente usadas no desenvolvimento de sistemas operacionais Windows.

4. ROTEIRO DE CÁLCULO

Para o cálculo dos esforços atuantes na viga, será adotada a hipótese de que o elemento é bi-apoiado.

4.1 Dados iniciais do concreto

No início do projeto da viga protendida, seleciona-se uma classe de resistência do concreto, o tipo de agregado utilizado e a classe de agressividade ambiental, que regula o cobrimento mínimo da peça (em função do qual será definida a excentricidade máxima de protensão) e a resistência mínima necessária, conforme a ABNT NBR 6118:2014.

Após a seleção destes dados, são calculadas outras importantes características do concreto usado, como o módulo de elasticidade e a resistência à tração de acordo com a ABNT NBR 6118:2014.

Para concretos de resistência até 50 MPa, a resistência à tração média é calculada por:

$$f_{ct,m} = 0,3 \cdot \sqrt[3]{f_{ck}^2} \quad (1)$$

Para concretos acima de 50 MPa de resistência, a fórmula usada é

$$f_{ct,m} = 2,12 \cdot \ln(1 + 0,11 \cdot f_{ck}) \quad (2)$$

Os valores superiores e inferiores da resistência à tração são dados por:

$$f_{ctk,inf} = 0,7 \cdot f_{ct,m} \quad (3)$$

$$f_{ctk,sup} = 1,3 \cdot f_{ct,m} \quad (4)$$

4.2 Dados iniciais da armadura ativa

No início do projeto, é definida a resistência do aço utilizado, o grau de relaxação das cordoalhas ou fios (baixa ou normal) e a máxima excentricidade dos cabos da armadura em relação às fibras inferiores da seção transversal.

A tensão limite de tração a ser considerada nos cálculos, no caso de armaduras pré-tracionadas, para o aço de relaxação normal é dada por:

$$\sigma_{lim} \leq \begin{cases} 0,77 \cdot f_{ptk} \\ 0,90 \cdot f_{pyk} \end{cases} \quad (5)$$

e, para o aço de relaxação baixa:

$$\sigma_{lim} \leq \begin{cases} 0,77 \cdot f_{ptk} \\ 0,85 \cdot f_{pyk} \end{cases} \quad (6)$$

em que

$$f_{pyk} = 0,9 \cdot f_{ptk}$$

No caso de armaduras pós-tracionadas, deve-se obedecer a seguinte condição de tensão limite para aços de relaxação normal:

$$\sigma_{lim} \leq \begin{cases} 0,74 \cdot f_{ptk} \\ 0,87 \cdot f_{pyk} \end{cases} \quad (7)$$

e, para o aço de relaxação baixa:

$$\sigma_{lim} \leq \begin{cases} 0,74 \cdot f_{ptk} \\ 0,82 \cdot f_{pyk} \end{cases} \quad (8)$$

4.3 Dados da seção transversal

Em seguida, define-se o tipo de seção será utilizada para a viga, entre os formatos retangular, seção T, seção duplo T e seção I.

Dado o tipo de seção, configura-se as dimensões características dos elementos da seção, a saber: a largura da alma (em seções T, duplo T e I), largura da mesa, espessura das almas e a altura da seção.

Inseridas todas as medidas necessárias, são realizados os cálculos das características geométricas da seção.

As áreas das seções são calculadas por geometria simples, dividindo a seção em triângulos e retângulos. O centroide da seção é calculado por:

$$y_{CG} = \frac{\sum_{i=1}^n A_i \cdot y_{CG,i}}{\sum_{i=1}^n A_i} \quad (9)$$

em que

y_{CG} é o centroide da seção transversal

A_i é a área de cada forma simples da seção e

$y_{CG,i}$ é o centroide de cada área simples da seção

Calculado o centroide da seção, obtém-se, então, o momento de inércia da viga. Para as formas retangulares, o momento de inércia é dado por:

$$I_{ret} = \frac{b \cdot h^3}{12} \quad (10)$$

Para as formas triangulares, tem-se:

$$I_{tri} = \frac{b \cdot h^3}{36} \quad (11)$$

O momento de inércia total da seção transversal é obtido pelo teorema de Steiner, transferindo-se o eixo de inércia do centroide de cada forma simples para o centroide da seção composta.

$$I = \sum_{i=1}^n \left(I_i + A_i \cdot (y_{CG,i} - y_{CG})^2 \right) \quad (12)$$

Em seguida, calculam-se os módulos de resistência da seção.

$$w_1 = \frac{I}{y_1} \quad (13)$$

$$w_2 = \frac{I}{y_2} \quad (14)$$

em que

y_1 é a distância do centroide à fibra inferior da seção

y_2 é a distância do centroide à fibra superior da seção

w_1 é o módulo de resistência da fibra inferior da seção

w_2 é o módulo de resistência da fibra superior da seção

I é o momento de inércia da seção.

A relação geométrica é dada por:

$$R_g = \frac{2 \cdot A_c}{\mu} \quad (15)$$

em que

A_c é a área da seção transversal e

μ é o perímetro da seção exposto ao ar

Quando não se sabe o perímetro total que será exposto durante a vida útil da peça, considera-se todo o perímetro da seção.

4.3.1 Outras fórmulas para cálculo das características geométricas

Para futuras alterações deste programa e inserção de novas funcionalidades, foi implementado no código fonte outras funções para cálculo de perímetro, área, centroide e momento de inércia de seções transversais genéricas. Os cálculos são baseados nas coordenadas do polígono que representa a seção transversal e se aplicam a qualquer seção delimitada por este polígono.

O cálculo do perímetro da seção é dado por:

$$P = \sum_{i=1}^{n-1} \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} \quad (16)$$

A área é calculada pela fórmula da área de Gauss:

$$A = \frac{1}{2} \left| \sum_{i=1}^{n-1} (x_i \cdot y_{i+1}) - \sum_{i=1}^{n-1} (x_{i+1} \cdot y_i) \right| \quad (17)$$

O centroide da seção é dado por:

$$y_{CG} = \frac{1}{6A} \sum_{i=1}^{n-1} (y_i + y_{i+1}) (x_i \cdot y_{y+1} - x_{i+1} \cdot y_i) \quad (18)$$

em que

A é a área da seção transversal

O momento de inércia da seção é calculado pela fórmula:

$$I = \frac{1}{12} \sum_{i=1}^{n-1} (x_i^2 + x_i \cdot x_{i+1} + x_{i+1}^2) (x_i \cdot y_{i+1} - x_{i+1} \cdot y_i) \quad (19)$$

Caso as coordenadas dos vértices do polígono já estejam em função do centroide da seção, não é necessário aplicar o teorema de Steiner. Caso contrário, deve-se adaptar o momento de inércia para o eixo que passa pelo centroide da seção.

$$I = I_0 + A \cdot d^2 \quad (20)$$

em que

I_0 é o momento de inércia calculado fora do eixo do centroide

A é a área da seção transversal e

d é a distância entre o eixo do momento de inércia calculado e o eixo que passa pelo centroide.

4.4 Carregamentos

Após a definição da seção transversal da viga e do vão entre os apoios, calcula-se a carga distribuída, em kN/m, do peso próprio da seção, considerando o peso específico do concreto armado de $\gamma_c = 25 \text{ kN/m}^3$.

Também são inseridas as cargas permanente e acidental – e seu respectivo tipo – atuantes na viga, ambos em kN/m e distribuídas ao longo do vão da viga.

Os coeficientes Ψ para as ações acidentais são definidos em função da frequência de atuação pela seguinte tabela:

Tabela 1 - Coeficientes Ψ para cargas acidentais

Ações		γ_z		
		ψ_0	$\psi_1^{1)}$	ψ_2
Cargas acidentais de edifícios	Locais em que não há predominância de pesos de equipamentos que permanecem fixos por longos períodos de tempo, nem de elevadas concentrações de pessoas ²⁾	0,5	0,4	0,3
	Locais em que há predominância de pesos de equipamentos que permanecem fixos por longos períodos de tempo, ou de elevada concentração de pessoas ³⁾	0,7	0,6	0,4
	Biblioteca, arquivos, oficinas e garagens	0,8	0,7	0,6
Vento	Pressão dinâmica do vento nas estruturas em geral	0,6	0,3	0
Temperatura	Variações uniformes de temperatura em relação à média anual local	0,6	0,5	0,3
¹⁾ Para os valores de ψ_1 relativos às pontes e principalmente aos problemas de fadiga, ver seção 23. ²⁾ Edifícios residenciais. ³⁾ Edifícios comerciais, de escritórios, estações e edifícios públicos.				

Fonte: ABNT NBR 6118:2014

Para o cálculo de vigas em estado de serviço no Estádio I, ou seja, sem abertura de fissuras, utiliza-se o Estado Limite de Abertura de Fissuras – ELS-F –, em que se considera a resistência do concreto à tração, e o Estado Limite de Descompressão – ELS-D –, em que a

tensão de tração nas fibras inferiores é anulada pela tensão de compressão proveniente da armadura protendida.

Para o ELS-D da protensão limitada, utiliza-se a combinação quase permanente de ações, enquanto para o ELS-F, é usada a combinação frequente.

Na protensão completa, o ELS-D é regido pela combinação frequente e o ELS-F pela combinação rara de ações.

4.5 Forças de protensão estimadas

A estimativa da força de protensão no tempo infinito - P_{∞} - é definida pelo menor valor obtido entre as duas combinações para os estados limites de serviço.

$$P_{\infty} \leq \begin{cases} P_{\infty}(ELS - D) \\ P_{\infty}(ELS - F) \end{cases} \quad (21)$$

No caso da protensão limitada, as equações de tensões são as seguintes:

Para o Estado Limite de Formação de Fissuras:

$$\sigma_{g1} + \sigma_{g2} + \Psi_1 \sigma_q + \sigma_{P0} \leq \sigma_{ct} \quad (22)$$

Para o Estado Limite de Descompressão:

$$\sigma_{g1} + \sigma_{g2} + \Psi_2 \sigma_q + \sigma_{P0} \leq 0 \quad (23)$$

Para a protensão completa, tem-se:

Para o ELS-F:

$$\sigma_{g1} + \sigma_{g2} + \sigma_q + \sigma_{P0} \leq \sigma_{ct} \quad (24)$$

Para o ELS-D:

$$\sigma_{g1} + \sigma_{g2} + \Psi_1 \sigma_q + \sigma_{P0} \leq 0 \quad (25)$$

No programa desenvolvido, foi considerada apenas uma solicitação acidental; pretende-se, porém, em futuras implementações, permitir ao usuário adicionar quantas ações acidentais forem necessárias e realizar a análise combinatória dos esforços.

4.6 Definição das armaduras

A área de armadura ativa estimada é dada por

$$A_{p,est} = \frac{P_{0,est}}{\sigma_{s,max}} \quad (26)$$

em que

$\sigma_{s,max}$ é a máxima tensão permitida na armadura

Com a área estimada, é definida então a área efetiva da armadura de protensão, escolhendo-se o tipo de cordoalha empregada, o número de cabos e a respectiva área nominal. Com a área efetiva calculada, as forças de protensão são recalculadas a partir dos novos valores.

Calcula-se a nova força P_0 pela fórmula:

$$P_{0,calc} = A_{p,ef} \cdot \sigma_{s,max} \quad (27)$$

em que

$A_{p,ef}$ é a área de aço efetiva da armadura ativa

Definida a área de aço, é necessário então obter as características da seção homogeneizada, como a área efetiva, o momento de inércia e os módulos de resistência das fibras superiores e inferiores.

A área de concreto equivalente à área do aço da armadura é dada por:

$$A_{c,eq} = \alpha_p \cdot A_s \quad (28)$$

O coeficiente α_p é um valor de equivalência entre o módulo de elasticidade do aço e do concreto:

$$\alpha_p = \frac{E_s}{E_{cs}} \quad (29)$$

A área “adaptada” de concreto se situa virtualmente no centroide das barras da armadura de protensão.

Calcula-se, então, o novo centroide da peça e, posteriormente, o momento de inércia da seção homogeneizada:

$$I_h = I_0 + A_c \cdot d^2 + A_{c,eq} \cdot e^2 \quad (30)$$

em que

I_0 é o momento de inércia da seção bruta

A_c é a área da seção bruta

d é a distância entre o centroide da seção bruta e o centroide da seção homogeneizada

$A_{c,eq}$ é a área equivalente em concreto da armadura ativa

e é a distância entre o centroide da seção homogeneizada e o centroide da armadura ativa

4.7 Cálculo de perdas

Para o cálculo das perdas progressivas, foi utilizado o método aproximado, como apresentado na seção 9.6.3.4.2 da ABNT NBR 6118:2014.

Por este processo, a perda de protensão é dada por:

$$\Delta\sigma_p(t, t_0) = \frac{\varepsilon_{cs}(t, t_0)E_p - \alpha_p \sigma_{c,p0g} \varphi(t, t_0) - \sigma_{p0} \chi(t, t_0)}{\chi_p + \chi_c \alpha_p \eta \rho_p} \quad (31)$$

em que

$\sigma_{c,p0g}$ é a tensão de compressão no concreto no centroide das armaduras ativas, quando atuam a força de protensão e o carregamento permanente,

$\varphi(t, t_0)$ é o coeficiente de fluência do concreto no instante t com a força de protensão aplicada no momento t_0

4.7.1 Espessura fictícia

Para obter as perdas progressivas de protensão da peça, foi calculada a espessura fictícia do elemento, baseado na umidade média do ar nos primeiros dias após a concretagem e a relação geométrica da seção transversal.

A espessura fictícia do elemento protendido é dada por:

$$h_{fict} = \gamma \cdot \frac{2 \cdot A_c}{4} \quad (32)$$

O coeficiente γ é obtido na tabela A.1 da ABNT NBR 6118:2014, e é dependente do tipo de ambiente ao qual a peça de concreto está exposta.

Tabela 2 – Coeficientes γ , ϕ e ε para cálculo de perdas de protensão

Ambiente	Umidade U %	Fluência $\varphi_{1c}^{1) 3)}$			Retração $10^4 \varepsilon_{1s}^{2) 3)}$			$\gamma^{4)}$
		Abatimento de acordo com a ABNT NBR NM 67						
		cm						
		0 - 4	5 - 9	10 - 15	0 - 4	5 - 9	10 - 15	
Na água	-	0,6	0,8	1,0	+ 1,0	+ 1,0	+ 1,0	30,0
Em ambiente muito úmido imediatamente acima da água	90	1,0	1,3	1,6	- 1,0	- 1,3	- 1,6	5,0
Ao ar livre, em geral	70	1,5	2,0	2,5	- 2,5	- 3,2	- 4,0	1,5
Em ambiente seco	40	2,3	3,0	3,8	- 4,0	- 5,2	- 6,5	1,0

¹⁾ $\varphi_{1c} = 4,45 - 0,035U$ para abatimento no intervalo de 5 cm a 9 cm e $U \leq 90\%$.

²⁾ $10^4 \varepsilon_{1s} = -6,16 - (U/484) + (U^2/1590)$ para abatimentos de 5 cm a 9 cm e $U \leq 90\%$.

³⁾ Os valores de φ_{1c} e ε_{1s} para $U \leq 90\%$ e abatimento entre 0 cm e 4 cm são 25% menores e para abatimentos entre 10 cm e 15 cm são 25% maiores.

⁴⁾ $\gamma = 1 + \exp (-7,8 + 0,1 U)$ para $U \leq 90 \%$.

NOTAS

1 Para efeito de cálculo, as mesmas expressões e os mesmos valores numéricos podem ser empregados no caso de tração.

2 Para o cálculo dos valores de fluência e retração, a consistência do concreto é aquela correspondente à obtida com o mesmo traço sem a adição de superplastificantes e superfluidificantes.

Fonte: ABNT NBR 6118:2014

No programa desenvolvido, o valor de γ é calculado pela fórmula sugerida pela norma, a partir do valor de umidade do ar fornecido pelo usuário.

$$\gamma = 1 + \exp(-7,8 + 0,1U) \quad (33)$$

4.7.2 Deformação por retração do concreto

Segundo a ABNT NBR 6118:2014, a retração do concreto é dada pela seguinte equação:

$$\varepsilon_{cs}(t, t_0) = \varepsilon_{cs\infty}[\beta_s(t) - \beta_s(t_0)] \quad (34)$$

A função β_s é dada pela fórmula:

$$\beta_s(t) = \frac{\left(\frac{t}{100}\right)^3 + A\left(\frac{t}{100}\right)^2 + B\left(\frac{t}{100}\right)}{\left(\frac{t}{100}\right)^3 + C\left(\frac{t}{100}\right)^2 + D\left(\frac{t}{100}\right) + E} \quad (35)$$

em que:

$$A = 40$$

$$B = 116h^3 - 282h^2 + 220h - 4,8$$

$$C = 2,5h^3 - 8,8h + 40,7$$

$$D = -75h^3 + 585h^2 + 496h - 6,8$$

$$E = -169hh^4 + 88h^3 + 584h^2 - 39h + 0,8$$

h é a espessura fictícia do elemento dimensionado

4.7.3 Deformação por fluência do concreto

O valor da deformação por fluência do concreto é dado por:

$$\varepsilon_{cc}(t, t_0) = \frac{\sigma_c}{E_{c28}} \cdot \varphi(t, t_0) \quad (36)$$

em que

E_{c28} é o módulo de elasticidade do concreto aos 28 dias

$\varphi(t, t_0)$ é o coeficiente de fluência do concreto, e é calculado por:

$$\varphi(t, t_0) = \varphi_a + \varphi_{f\infty} [\beta_f(t) - \beta_f(t_0)] + \varphi_{d\infty} \beta_d \quad (37)$$

A função β_f no programa é dada por:

$$\beta_f(t) = \frac{t^2 + At + B}{t^2 + Ct + D} \quad (38)$$

em que:

$$A = 42h^3 - 350h^2 + 588h + 113$$

$$B = 768h^3 - 3060h^2 + 3234h - 23$$

$$C = -200h^3 + 13h^2 + 1090h + 183$$

$$D = 7579h^3 - 31916h^2 + 35343h + 1931$$

h é a altura fictícia da peça dimensionada

5. UTILIZAÇÃO DO PROGRAMA

5.1 Definição dos materiais

Ao abrir o programa, a primeira tela indica as informações a respeito dos materiais da viga a ser dimensionada.

Na primeira caixa de listagem, o usuário deve selecionar o grau de agressividade ambiental do meio, o que servirá para definir a resistência necessária do concreto, bem como limitar o cobrimento nominal da peça.

Em seguida, deve-se definir a resistência do concreto, observando as limitações previamente impostas. Escolhe-se também o tipo de agregado a ser usado no concreto.

Após a definição do concreto, o usuário escolhe o tipo de aço para as armaduras ativas, entre as opções de resistência do aço (CP 175, CP 190 ou CP 210), o nível de relaxação da armadura (normal ou baixa) e, por fim, o tipo de cabo a ser empregado (cordoalhas, barras ou fios).

Figura 7 - Tela para inserção dos dados dos materiais da viga

A imagem mostra a interface de um software chamado "Vigas de concreto protendido". A janela possui uma barra de menu com "Arquivo" e uma barra de abas com "Dados Iniciais", "Geometria", "Forças e carregamentos", "Armaduras", "Cálculo de perdas" e "Relatório". A aba "Dados Iniciais" está selecionada e contém duas seções principais:

- Dados do concreto:**
 - Classe de agressividade ambiental: Menu suspenso com "Classe II - Áreas urbanas" selecionado.
 - Tipo de concreto: Menu suspenso com "C30" selecionado.
 - Agregado: Menu suspenso com "Basalto" selecionado.
- Dados do aço:**
 - Aço para armadura ativa: Menu suspenso com "CP 190" selecionado.
 - Relaxação do aço: Menu suspenso com "RB" selecionado.
 - d' mínimo da armadura (cm): Campo de texto com o valor "5" digitado.

Fonte: Autor

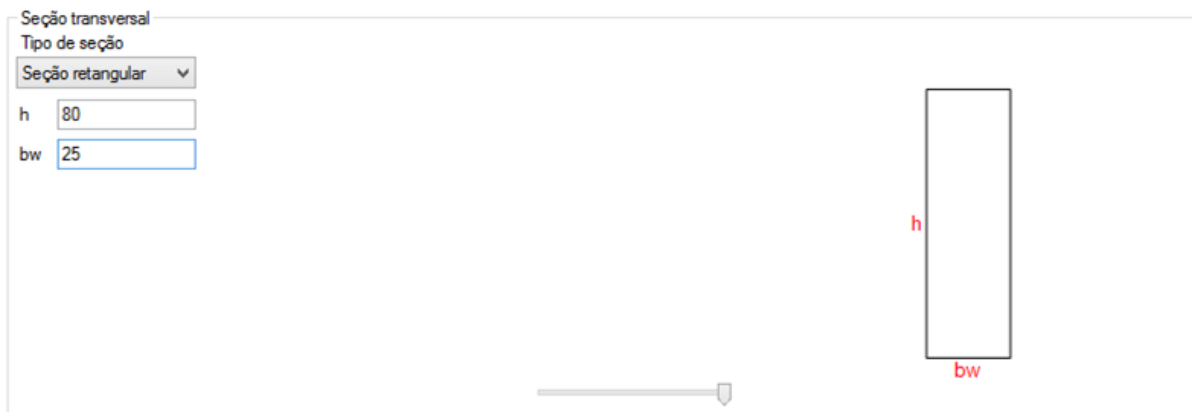
5.2 Geometria da peça

Na segunda etapa, o usuário define a geometria da peça dimensionada. No programa desenvolvido, há quatro tipos de seções disponíveis (retangular, seção T, duplo T e seção I). Ao optar por uma das seções, o usuário insere as medidas nos campos correspondentes, que são exibidos a depender do tipo de peça especificada.

Ao realizar cada modificação nas medidas da peça, o programa recalcula automaticamente as características geométricas, como a área, o centroide, o momento de inércia e os módulos de resistência para as fibras superiores e inferiores. O formato da seção é exibido em escala no lado direito da tela e é possível ajustar o tamanho do desenho para melhor visualização.

Nesta parte também é inserido o comprimento do vão entre os apoios da peça, com o valor em metros.

Figura 8 - Exemplo de inserção de seção retangular no programa



Fonte: Autor

Figura 9 - Exemplo de seção T no programa

Seção transversal

Tipo de seção

Seção T ▼

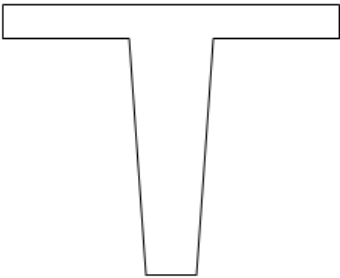
h 80

bw 15

bw2 25

hf 10

bf 100



Fonte: Autor

Figura 10 - Exemplo de seção duplo T no programa

Seção transversal

Tipo de seção

Seção duplo T ▼

h 80

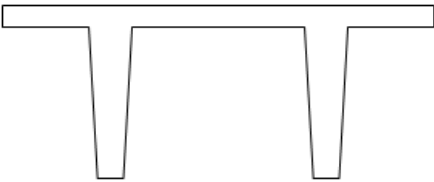
bw 12

bw2 20

hf 10

bf 200

esp 100



Fonte: Autor

Figura 11 - Exemplo de seção I no programa

Seção transversal

Tipo de seção

Seção I ▼

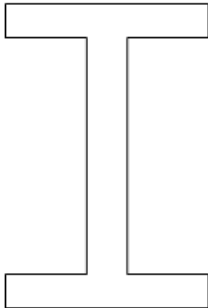
h 90

bw 12

bw2 12

hf 10

bf 60



Fonte: Autor

Figura 12 - Tela para inserção dos dados geométricos da viga

Fonte: Autor

5.3 Carregamentos e ações

Ao modificar a seção transversal da peça, o peso próprio da estrutura é recalculado automaticamente. O usuário, porém, deve inserir manualmente os valores das cargas permanentes e acidentais, dados em kN/m.

Nesta versão do programa, só é possível inserir um tipo de carga acidental. Após inserir o valor da carga, pede-se que o usuário especifique na caixa de listagem qual o tipo de carga acidental atuante, como mostrado na Figura 13.

Figura 13 - Tipos de cargas acidentais disponíveis para seleção

Fonte: Autor

Ao selecionar o tipo de carga, o programa seleciona os valores de Ψ_0 , Ψ_1 e Ψ_2 de acordo com a Tabela 11.2 da ABNT NBR 6118:2014.

Tabela 3 - Coeficientes Ψ de minoração estatística de cargas acidentais

Ações		γ_{f2}		
		Ψ_0	$\Psi_1^{1)}$	Ψ_2
Cargas acidentais de edifícios	Locais em que não há predominância de pesos de equipamentos que permanecem fixos por longos períodos de tempo, nem de elevadas concentrações de pessoas ²⁾	0,5	0,4	0,3
	Locais em que há predominância de pesos de equipamentos que permanecem fixos por longos períodos de tempo, ou de elevada concentração de pessoas ³⁾	0,7	0,6	0,4
	Biblioteca, arquivos, oficinas e garagens	0,8	0,7	0,6
Vento	Pressão dinâmica do vento nas estruturas em geral	0,6	0,3	0
Temperatura	Variações uniformes de temperatura em relação à média anual local	0,6	0,5	0,3
¹⁾ Para os valores de Ψ_1 relativos às pontes e principalmente aos problemas de fadiga, ver seção 23. ²⁾ Edifícios residenciais. ³⁾ Edifícios comerciais, de escritórios, estações e edifícios públicos.				

Fonte: ABNT NBR 6118:2014

Nesta tela, também é especificada a excentricidade dos cabos da armadura ativa (constante ou variável), e o tipo de protensão a ser utilizada (parcial, limitada ou completa).

Ao selecionar o tipo de protensão, o programa exibe quais combinações de ações serão empregadas no cálculo das forças necessárias para protender a armadura.

Ao clicar no botão “Calcular forças”, o programa obtém o valor das forças estimadas de protensão, a saber, a força inicial P_0 , a força ancorada P_a e a força de protensão final P_∞ , em como suas respectivas tensões no aço em função da tensão máxima permitida. Em função da tensão máxima admitida, é calculada a área de aço estimada para a armadura ativa.

Figura 14 – Tela para inserção dos carregamentos e cálculo das forças de protensão

Fonte: Autor

5.4 Definição da armadura

Na tela seguinte, o primeiro campo exibe a área estimada calculada previamente e serve como referência para definir a área efetiva.

O próximo passo é, então, escolher a quantidade de barras, fios ou cordoalhas para a armadura e a sua respectiva área nominal unitária. Com base nos dados inseridos na tabela, o programa calcula a área de aço efetiva e exibe a diferença, em porcentagem, entre o valor estimado e o valor estipulado pelo usuário.

Após a definição da área de aço, o usuário deve clicar no botão “Calcular área homogeneizada e novas forças”. O programa calcula, então, as novas características da seção transversal da peça, como a área homogeneizada, o momento de inércia e os módulos de resistência, em função da área de aço efetiva.

Com as características atualizadas da seção, o programa recalcula as forças de protensão P_0 , P_i e P_a .

Figura 15 - Tela para inserção da armadura de protensão efetiva

Vigas de concreto protendido

Arquivo

Dados Iniciais Geometria Forças e carregamentos **Armaduras** Cálculo de perdas Relatório

Área de armadura estimada: 6,13 cm²

Área efetiva calculada: 6,08 cm²

Diferença: -0,80%

Calcular seção homogeneizada e novas forças

Insira as informações das cordoalhas nos campos abaixo:

	Nº de cordoalhas	Área nominal (cm ²)	Área total
▶	6	1,014	6,084001
*			

Consulte o catálogo do fabricante para obter a área nominal dos fios/cordoalhas

Dados da seção homogeneizada

lx: 2154879

w1: 40241,94

w2: -81464,06

Novas forças de protensão

σP_i	P_i (força inicial)	P_a (força ancorada)	P_0 (força calculada)
145,35 kN/cm ²	884,31 kN	857,78 kN	910,77 kN

Estima-se perda de 3% para aços de relaxação baixa e 7% para relaxação normal

Fonte: Autor

5.5 Cálculo de perdas

Na seção Cálculo de perdas, há três quadros onde o usuário deve inserir algumas informações.

No primeiro, deve-se informar qual será a umidade do ar esperada durante a concretagem, a protensão e os primeiros dias da peça. Com este valor, o programa calcula a altura fictícia da peça, que será utilizada para os cálculos de perdas por retração e fluência do concreto.

Figura 16 – Quadro para inserção dos dados iniciais para cálculo de perdas

1. Cálculo de espessuras fictícias	
Umidade do ar (%)	70
Coef. alfa (fluência)	3
Coef. alfa (retração)	1
t0 (fluência)	21 dias
t0 (retração)	7 dias
tempo final (dias)	10000
Coef. gamma	1,45
Espessura fictícia	12,15 cm

Fonte: Autor

No segundo quadro, o usuário deve inserir o valor do coeficiente ε_{1s} , usando como referência a tabela A.1 da ABNT NBR 6118:2014. Os outros valores são calculados automaticamente e são usados no cálculo do coeficiente de fluência do concreto χ_c .

Figura 17 – Quadro dos dados relativos às perdas por retração do concreto

2. Deformação por retração do concreto	
ε_{1s}^*	-0,0004
ε_{2s}	1,0008
ε_{cs}^*	-0,0004
$\varepsilon_{cs}(t,t_0)$	-3,38e-4

* Tabela A.1 NBR 6118:2014

Fonte: Autor

Analogamente, o usuário também deve preencher o campo do coeficiente φ_{1c} no terceiro quadro, também com base na tabela A.1 da ABNT NBR 6118:2014, enquanto os dados seguintes são obtidos de forma automática. É feito então o cálculo do coeficiente de fluência do aço χ_p .

Figura 18 – Quadro dos dados relativos às perdas por fluência do concreto

3. Deformação por fluência do concreto

ϕ_{1c^*}	1,5
ϕ_{2c}	1,684
$\phi_{f=}$	2,526
ϕ_a	0,098
$\phi_{d=}$	0,400
$\phi(t,t_0)$	2,097

* Tabela A.1 NBR 6118:2014

Fonte: Autor

Ao pressionar o botão “Calcular perdas”, o programa calcula o valor de $\Delta\sigma$ das perdas progressivas, pelo método simplificado, definido na seção 9.6.3.4.2 da ABNT NBR 6118:2014 e, então, o valor da força de protensão P_∞ .

Figura 19 - Tela para cálculo das perdas de protensão

Vigas de concreto protendido

Arquivo

Dados Iniciais Geometria Forças e carregamentos Amaduras **Cálculo de perdas** Relatório

1. Cálculo de espessuras fictícias

Umidade do ar (%) 70

Coef. alfa (fluência) 3

Coef. alfa (retração) 1

t0 (fluência) 21 dias

t0 (retração) 7 dias

tempo final (dias) 10000

Coef. gamma 1,45

Espessura fictícia 12,15 cm

2. Deformação por retração do concreto

ϵ_{1s^*} -0,0004

ϵ_{2s} 1,0008

$\epsilon_{cs=}$ -0,0004

$\epsilon_{cs(t,t_0)}$ -3,38e-4

* Tabela A.1 NBR 6118:2014

3. Deformação por fluência do concreto

ϕ_{1c^*} 1,5

ϕ_{2c} 1,684

$\phi_{f=}$ 2,526

ϕ_a 0,098

$\phi_{d=}$ 0,400

$\phi(t,t_0)$ 2,097

* Tabela A.1 NBR 6118:2014

Calcular

X_c 2,048

X_p 1,080

P0 calculado 910,77 kN

Perdas 30,02 kN

P= calculado 880,75 kN

Fonte: Autor

6. EXEMPLO DE UTILIZAÇÃO DO PROGRAMA

Pra demonstração da utilização do programa, será dimensionada uma viga T, usando aço CP-190, concreto C40 e um vão de 12 metros.

6.1 Dados iniciais

Na parte de definição dos dados iniciais, foi definida a classe de agressividade ambiental como sendo Classe II, equivalente a áreas urbanas. O concreto é do tipo C40, com agregado gráúdo de basalto. A armadura ativa é do tipo CP-190 de relaxação baixa, e seu centroide está localizado a 5 cm da fibra inferior da viga.

Figura 20 – Dados iniciais da viga de exemplo

A imagem mostra a interface do programa "Vigas de concreto protendido". No topo, há uma barra de menu com o ícone de arquivo e o nome do programa. Abaixo, há uma barra de abas com as seguintes opções: "Dados Iniciais", "Geometria", "Forças e carregamentos", "Armaduras", "Cálculo de perdas" e "Relatório". A aba "Dados Iniciais" está selecionada. O formulário é dividido em duas seções principais: "Dados do concreto" e "Dados do aço".

Dados do concreto:

- Classe de agressividade ambiental: Classe II - Áreas urbanas (menu suspenso)
- Tipo de concreto: C40 (menu suspenso)
- Agregado: Basalto (menu suspenso)

Dados do aço:

- Aço para armadura ativa: CP 190 (menu suspenso)
- Relaxação do aço: RB (menu suspenso)
- d' mínimo da armadura (cm): 5 (campo de texto)

Fonte: Autor

6.2 Seção transversal

A seção transversal definida tem as dimensões mostradas na figura 22. A partir desta seção, o momento de inércia correspondente calculado foi de 1031678,0 cm⁴. Os valores dos módulos de resistência das fibras inferior e superior calculados foram, respectivamente, 19068,25 cm³ e -39840 cm³.

Figura 21 – Dados geométricos da viga de exemplo

The screenshot shows a software window titled "Vigas de concreto protendido". It has a menu bar with "Arquivo" and a tabbed interface with "Dados Iniciais", "Geometria", "Forças e carregamentos", "Armaduras", "Cálculo de perdas", and "Relatório". The "Geometria" tab is active. On the left, under "Seção transversal", there is a dropdown for "Tipo de seção" set to "Seção T". Below it are input fields for dimensions: h (80), bw (11), bw2 (17), hf (5), and bf (125). To the right of these fields is a diagram of a T-beam cross-section. Below the dimension fields are two groups of input fields: "Dados da seção" with lx (1031678), w1 (19068,25), and w2 (-39840); and "Vão da viga" with "Comprimento da viga" (12). A horizontal slider is located between the dimension fields and the section data fields.

Fonte: Autor

6.3 Esforços e forças estimadas

Com base na geometria da seção transversal, o valor do peso próprio da viga calculado foi de 4,19 kN/m.

A carga permanente definida foi de 2,5 kN/m e a carga acidental de 10 kN/m, sendo considerado o uso da edificação equivalente ao de bibliotecas, oficinas e garagens. Com isso, os coeficientes de minoração estatísticos das ações acidentais são $\Psi_0 = 0,8$, $\Psi_1 = 0,7$ e $\Psi_2 = 0,6$.

O tipo de protensão escolhida foi a protensão limitada, de forma que as combinações utilizadas para o estado limite de descompressão e de formação de fissuras são, respectivamente, a quase-permanente e a frequente.

A partir dos dados inseridos, a força de protensão P_∞ estimada foi de 377,55 kN, arredondada para 380 kN. Assumindo 25% de perdas de protensão, a força P_0 estimada foi de 506,67 kN.

Assim, a área de aço estimada para a armadura ativa foi de 3,49 cm²,

Figura 22 – Dados das cargas atuantes sobre a viga de exemplo

The screenshot shows a software window titled "Vigas de concreto protendido". It has a menu bar with "Arquivo" and a tab bar with "Dados Iniciais", "Geometria", "Forças e carregamentos", "Armaduras", "Cálculo de perdas", and "Relatório". The "Forças e carregamentos" tab is active.

Carregamentos (kN/m)

Peso próprio	Carga permanente	Cargas acidentais	Tipo de carga acidental
4,19	2,5	10	Bibliotecas, arquivos, oficinas e garagens

ψ_0 0,8 ψ_1 0,7 ψ_2 0,6

Protensão

Tipo de protensão: Protensão Limitada

Excentricidade dos cabos: Excentricidade constante

Combinações para o estado limite de serviço ELS

Estado Limite de Descompressão - ELS-D	Combinação quase permanente
Estado Limite de Formação de fissuras - ELS-F	Combinação frequente

Forças calculadas

Calcular forças estimadas	σ_{pe}	$P_{pe, est}$	$P_{pe, adot}$	P_0	Área de aço estimada:
	-11,98 MPa	377,55 kN	380,00 kN	506,67 kN	3,49 cm ²

perdas estimadas em 25%

Fonte: Autor

6.4 Área de aço efetiva e novas forças de protensão

Com base na área estimada de 3,49 cm², foram consideradas na armadura efetiva 3 cordoalhas com área nominal de 1,2 cm² cada, totalizando 3,60 cm² de área efetiva, 3,28% a mais do que o estimado anteriormente.

A nova seção homogeneizada com a armadura efetiva tem momento de inércia de 1075936 cm⁴, e os módulos de resistência das fibras inferiores e superiores são, respectivamente, 20086,03 cm³ e -40703,31 cm³, ligeiramente maiores que os valores da seção bruta de concreto.

A força de protensão inicial efetiva foi de 523,26 kN. Com perdas estimadas em 3% para relaxação baixa, a força ancorada foi de 507,56 kN. Após as perdas iniciais, a força de protensão P_0 calculada foi de 476,27 kN.

Figura 23 – Dados das armaduras efetivas da viga de exemplo

The screenshot shows a software window titled "Vigas de concreto protendido". It has several tabs: "Arquivo", "Dados Iniciais", "Geometria", "Forças e carregamentos", "Armaduras", "Cálculo de perdas", and "Relatório". The "Armaduras" tab is active.

On the left, under "Área de armadura estimada", the value is 3,49 cm². Below it, "Área efetiva calculada" is 3,60 cm², and "Diferença" is 3,28%. A button "Calcular seção homogeneizada e novas forças" is present.

In the center, a table for "Insira as informações das cordoalhas nos campos abaixo:" has the following data:

	Nº de cordoalhas	Área nominal (cm²)	Área total
▶	3	1,2	3,6
*			

On the right, a note says: "Consulte o catálogo do fabricante para obter a área nominal dos fios/cordoalhas".

At the bottom, under "Dados da seção homogeneizada", there are input fields for b_x (1075936), w_1 (20086,03), and w_2 (-40703,31).

Under "Novas forças de protensão", there are four input fields: σP_i (145,35 kN/cm²), P_i (força inicial) (523,26 kN), P_a (força ancorada) (507,56 kN), and P_0 (força calculada) (476,27 kN). A note below these fields states: "Estima-se perda de 3% para aços de relaxação baixa e 7% para relaxação normal".

Fonte: Autor

6.5 Cálculo de perdas

Para o cálculo das perdas, a umidade do ar considerada nos dias iniciais da viga foi de 70%. Adotou-se os coeficientes de fluência e retração de, respectivamente, 3 e 1. O tempo inicial do cálculo de fluência foi de 21 dias e o de retração, 7 dias. O tempo final adotado foi de 10000 dias.

Com base na tabela A.1 da ABNT NBR 6118:2014 (ver Tabela 2 deste trabalho), e considerando o slump do concreto de 12 cm, foi adotado o coeficiente de retração do concreto de -0,0004 e de fluência de 2,5. Com isso, o coeficiente de fluência χ_c encontrado foi de 2,583, enquanto o coeficiente de fluência χ_p do aço da armadura ativa foi de 1,058.

Figura 24 – Dados das perdas de protensão da viga de exemplo

1. Cálculo de espessuras fictícias			2. Deformação por retração do concreto		3. Deformação por fluência do concreto	
Umidade do ar (%)	70		ϵ_{1s}^*	-0,0004	ϕ_{1c}^*	2,5
Coef. alfa (fluência)	3		ϵ_{2s}	1,0035	ϕ_{2c}	1,687
Coef. alfa (retração)	1		ϵ_{cs}^*	-0,0004	$\phi_{f=}$	4,219
t0 (fluência)	21 dias		$\epsilon_{cs(t,t0)}$	-3,38e-4	ϕ_a	0,098
t0 (retração)	7 dias				$\phi_{d=}$	0,400
tempo final (dias)	10000				$\phi(t,t0)$	3,166
Coef. gamma	1,45		* Tabela A.1 NBR 6118:2014		* Tabela A.1 NBR 6118:2014	
Espessura fictícia	12,00 cm				X_c	2,583
					X_p	1,058

P0 calculado	Perdas	P= calculado
476,27 kN	27,36 kN	448,90 kN

Fonte: Autor

Com os dados calculados, as perdas obtidas foram de 27,36 kN. A partir da força P_0 efetiva, obteve-se, então, a força $P_\infty = 448,90$ kN.

7. CONCLUSÃO

Devido à crescente demanda por rapidez e qualidade na construção civil no Brasil, o concreto protendido se torna uma alternativa cada vez mais viável e popular. Com a utilização de peças pré-moldadas e protendidas é possível reduzir o tempo total de obra, bem como permitir a construção de edificações mais esbeltas e com maiores vãos entre apoios.

Dessa forma, é conveniente desenvolver soluções que visem atender a esta crescente demanda de mercado, visto que, apesar de haver diversas opções de *softwares* de cálculo de concreto armado, a disponibilidade de programas voltados ao cálculo de elementos protendidos é mais restrita.

A partir desta análise, desenvolveu-se um programa de cálculo de vigas de concreto protendido, abrangendo a definição das características do aço e do concreto, a seção

transversal e o cálculo dos esforços solicitantes, das forças de protensão e das perdas de protensão.

O programa apresentou o resultado esperado para a obtenção de áreas de armadura ativa, bem como os valores de perdas de protensão, sendo possível, por meio do programa desenvolvido, definir as armaduras necessárias para a protensão de uma viga com armadura pré-tracionada, embora ainda seja necessário executar verificações de tensões ao longo do comprimento da viga de forma manual ou com o auxílio de outros softwares.

8. SUGESTÕES PARA FUTUROS TRABALHOS

Em futuras versões do programa, pretende-se adicionar funções como a possibilidade de adoção de cabos curvos, cálculo de cabos individuais, seções transversais livremente personalizáveis, inserção de múltiplas cargas acidentais, bem como verificações de tensões nas seções, cálculo das curvas limite e de fuso limite.

Serão implementadas também outras funcionalidades que forem consideradas relevantes para que o programa seja usado profissionalmente, por empresas e calculistas estruturais, e possivelmente no ambiente acadêmico, de forma a complementar a formação dos alunos com auxílio da tecnologia.

Como sugestão para continuação deste trabalho, pode-se avaliar a possibilidade de se obter os valores de esforços a partir de cálculos matriciais para estruturas hiperestáticas, de modo a permitir, por exemplo, o cálculo de vigas com três ou mais apoios.

REFERÊNCIAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 6118: Projeto de estruturas de concreto - Procedimento**. Rio de Janeiro, 2014.

CONCRETO protendido: vantagens e características. [S. l.], 23 dez. 2016. Disponível em: <http://fullestruturas.com.br/concreto-protendido-vantagens-e-caracteristicas/>. Acesso em: 1 dez. 2019.

HANAI, João Bento de. **Fundamentos do concreto protendido**. [S. l.: s. n.], 2005.

LOBATO, Raul. **Sistemas de aplicação da protensão**. [S. l.], 26 dez. 2019. Disponível em: http://sinop.unemat.br/site_antigo/prof/foto_p_downloads/fot_12221aula_02_pdf_AULA_02.pdf. Acesso em: 20 dez. 2019.

MAIA, Victor. **Edificações – Concreto Protendido**. [S. l.], 15 nov. 2018. Disponível em: <https://maquinadeaprovacao.com.br/engenharia/edificacoes-concreto-protendido/>. Acesso em: 1 dez. 2019.

ANEXO A – CÓDIGO FONTE DO PROGRAMA

Observação: todos os arquivos cujo título começam com “FormPrincipal” são extensões da classe parcial FormPrincipal, que é o formulário principal e único do programa. Optou-se por dividir a mesma classe em arquivos diferentes para organizar as etapas do dimensionamento e otimizar o desenvolvimento do programa, apesar de, por este método, as funções e a performance do programa permanecerem inalteradas.

Arquivo: FormPrincipal.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Protendido2
{
    public partial class formPrincipal : Form
    {
        public formPrincipal()
        {
            InitializeComponent();
            imagemSecao = desenhoSecao.CreateGraphics();
        }

        private void atualizarDesenhos(object sender, EventArgs e)
        {
            desenharSecao();
        }

        private void FormPrincipal_Load(object sender, EventArgs e)
        {
        }
    }
}
```

Arquivo: FormPrincipalInicio.cs

```
namespace Protendido2
{
    public partial class formPrincipal : Form {

        public float e_min, exc;
        public float[] concretos = { 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75,
80, 85, 90 };

        public float[] aços = { 175, 190, 210 };
    }
}
```

```

public string[] relaxacao = { "RN", "RB" };

Concreto concreto;
Aço aço;

private void CampoConcreto_SelectedIndexChanged(object sender, EventArgs e)
{
    gravarConcreto();
}
private void CampoAgregado_SelectedIndexChanged(object sender, EventArgs e)
{
    gravarConcreto();
}

private void gravarConcreto()
{
    if (campoConcreto.SelectedIndex != -1 && campoAgregado.SelectedIndex != -
1)
        concreto = new Concreto(concretos[campoConcreto.SelectedIndex],
campoAgregado.SelectedItem.ToString());
}

private void GravarTipoAco(object sender, EventArgs e)
{
    if (campoTipoAco.SelectedIndex != -1 && campoRelaxacaoAco.SelectedIndex !=
-1)
        aço = new Aço(aços[campoTipoAco.SelectedIndex],
relaxacao[campoRelaxacaoAco.SelectedIndex]);
}

private void CampoExcentricidade_TextChanged(object sender, EventArgs e)
{
    float.TryParse(campoExcentricidade.Text, out exc);
}

}
}

```

Arquivo: Materiais.cs

```

namespace Protendido2
{
    public class Concreto
    {
        public float gammaC = 25; // kN/m³

        // Todas as unidades em MPa
        public float fck;
        public float fctm;
        public float fctkinf;
        public float fctksup;

        public string agregado;
        private float alfaE, alfaI;
        public float Eci, Ecs;

        public Concreto(float fck)
        {
            this.fck = fck;

```

```

        if (fck <= 50)
            this.fctm = 0.3f * (float)Math.Pow(this.fck, 2f / 3f);
        else
            this.fctm = 2.12f * (float)Math.Log(1 + .11 * (float)fck);

        this.fctkinf = 0.7f * this.fctm;
        this.fctksup = 1.3f * this.fctm;

        this.alfaE = 1f;
    }

    public Concreto(float fck, string agregado)
    {
        this.fck = fck;

        if (fck <= 50)
            this.fctm = 0.3f * (float)Math.Pow(this.fck, 2f / 3f);
        else
            this.fctm = 2.12f * (float)Math.Log(1 + .11 * (float)fck);

        this.fctkinf = 0.7f * this.fctm;
        this.fctksup = 1.3f * this.fctm;
        this.agregado = agregado;

        switch (agregado)
        {
            case "Basalto":
            case "Diabásio":
                alfaE = 1.2f;
                break;
            case "Granito":
            case "Gnaiss":
                alfaE = 1.0f;
                break;
            case "Calcário":
                alfaE = 0.9f;
                break;
            case "Arenito":
                alfaE = 0.7f;
                break;
        }

        if (fck <= 50)
            Eci = alfaE * 5600f * (float)Math.Sqrt(fck);
        else
            Eci = 21.5f * 1000 * alfaE * (float)Math.Pow((fck / 10 + 1.25), (1 /
3)));

        alfaI = (0.8f + 0.2 * fck / 80f) > 1 ? 1 : (0.8f + 0.2f * fck / 80f);
        Ecs = alfaI * Eci;
    }

    public float fcj(float dias)
    {
        return fck * (float)Math.Exp(0.119 * (1 - Math.Sqrt(28f / dias)));
    }

    public float Ecsj(float dias)
    {
        return alfaI * alfaE * 5600f * (float)Math.Sqrt(this.fcj(dias));
    }
}

```

```

public class Aço
{
    // Resistência * Um aço do tipo CP 190 tem resistência de 1900 MPa, o
    // equivalente a 190 kN/cm²
    public float fptk; // kN/cm²
    public float fpyk; // kN/cm²

    // Elasticidade
    public float Ep = 200000; // MPa

    public float sigmaMax; // kN/cm²

    public string relaxacao; // Baixa ou normal

    public string tipo; // Barras, cordoalhas ou fios

    public Aço(float fptk, string relaxacao)
    {
        this.fptk = fptk;
        this.fpyk = .9f * fptk;

        if (relaxacao == "RN")
            sigmaMax = Math.Min(.77f * fptk, .90f * fpyk);
        else
            sigmaMax = Math.Min(.77f * fptk, .85f * fpyk);

        this.relaxacao = relaxacao;

        this.tipo = "Cordoalhas";
    }

    public float sigmaPi() // NBR 6118:2014 item 9.6.1.2.1
    {
        if (relaxacao == "RN")
            return Math.Min(.77f * fptk, .90f * fpyk);
        else
            return Math.Min(.77f * fptk, .85f * fpyk);
    }
}

```

Arquivo: FormPrincipalSecao.cs

```

namespace Protendido2
{
    public partial class formPrincipal : Form
    {
        /* -----
        * ----- VARIÁVEIS -----
        * -----
        */

        // Variável para desenhar a seção transversal
        Graphics imagemSecao;

        // Dados da seção transversal
        float h;
        float bw;
        float bw2;
    }
}

```

```

float hf;
float bf;
float esp;

Secao secaoTransversal;

/* -----
 * -----  FUNÇÕES PRINCIPAIS  -----
 * -----
 */

public void desenharSecao()
{
    imagemSecao.SmoothingMode =
System.Drawing.Drawing2D.SmoothingMode.AntiAlias;

    // Limpa o desenho atual
    imagemSecao.Clear(Color.White);

    // Verifica se há campos vazios
    foreach (Control campo in groupSecao.Controls)
    {
        if (campo is TextBox)
        {
            if (campo.Text == "" && campo.Visible) {
imagemSecao.Clear(Color.White); return; }
        }
    }

    // Variáveis auxiliares para desenho
    float escala = (1f + alterarEscala.Value) / 5f;
    Font font = new Font("Arial", 10);
    PointF[] pontos = new PointF[12];

    // Centro do desenho
    float centroX = desenhoSecao.Width / 2;
    float centroY = desenhoSecao.Height / 2;

    // Dados principais (usados para seção retangular
    float.TryParse(campoH.Text, out h);
    float.TryParse(campoBw.Text, out bw);

    // Seção T
    float.TryParse(campoHf.Text, out hf);
    float.TryParse(campoBf.Text, out bf);
    float.TryParse(campoBw2.Text, out bw2);

    // Seção Duplo T
    float.TryParse(campoEsp.Text, out esp);

    switch (campoTipoSecao.SelectedIndex)
    {
        case 0: // Seção retangular

            imagemSecao.DrawString(
                "bw",
                font,
                System.Drawing.Brushes.Red,
                centroX - imagemSecao.MeasureString("bw", font).Width / 2,
                centroY + escala * h / 2);

```

```

        imagemSecao.DrawString(
            "h",
            font,
            System.Drawing.Brushes.Red,
            centroX - escala * bw / 2 - imagemSecao.MeasureString("h",
font).Width,
            centroY - imagemSecao.MeasureString("h", font).Height / 2);

        pontos[0] = new PointF(centroX - escala * bw / 2, centroY - escala
* h / 2);
        pontos[1] = new PointF(centroX + escala * bw / 2, centroY - escala
* h / 2);
        pontos[2] = new PointF(centroX + escala * bw / 2, centroY + escala
* h / 2);
        pontos[3] = new PointF(centroX - escala * bw / 2, centroY + escala
* h / 2);
        PointF[] retangulo = { pontos[0], pontos[1], pontos[2], pontos[3]
};

        imagemSecao.DrawPolygon(Pens.Black, retangulo);

        secaoTransversal = new Secao(h, bw);

        break;
    case 1: // Seção T
        pontos[0] = new PointF(centroX - escala * bf / 2, centroY - escala
* h / 2);
        pontos[1] = new PointF(centroX + escala * bf / 2, centroY - escala
* h / 2);
        pontos[2] = new PointF(centroX + escala * bf / 2, centroY - escala
* (h / 2 - hf));
        pontos[3] = new PointF(centroX + escala * bw2 / 2, centroY -
escala * (h / 2 - hf));
        pontos[4] = new PointF(centroX + escala * bw / 2, centroY + escala
* h / 2);
        pontos[5] = new PointF(centroX - escala * bw / 2, centroY + escala
* h / 2);
        pontos[6] = new PointF(centroX - escala * bw2 / 2, centroY -
escala * (h / 2 - hf));
        pontos[7] = new PointF(centroX - escala * bf / 2, centroY - escala
* (h / 2 - hf));
        PointF[] secaoT = new PointF[8];
        for (int i = 0; i < 8; i++) { secaoT[i] = pontos[i]; }

        imagemSecao.DrawPolygon(Pens.Black, secaoT);

        secaoTransversal = new Secao(h, bw, bw2, hf, bf);

        break;
    case 2: // Seção Duplo T
        pontos[0] = new PointF(centroX - escala * bf / 2, centroY - escala
* h / 2);
        pontos[1] = new PointF(centroX + escala * bf / 2, centroY - escala
* h / 2);
        pontos[2] = new PointF(centroX + escala * bf / 2, centroY - escala
* (h / 2 - hf));
        pontos[3] = new PointF(centroX + escala * (esp/2 + bw2/2), centroY
- escala * (h/2 - hf));

```

```

        pontos[4] = new PointF(centroX + escala * (esp / 2 + bw / 2),
centroY + escala * h / 2);
        pontos[5] = new PointF(centroX + escala * (esp / 2 - bw / 2),
centroY + escala * h / 2);
        pontos[6] = new PointF(centroX + escala * (esp / 2 - bw2 / 2),
centroY - escala * (h / 2 - hf));
        pontos[7] = new PointF(centroX - escala * (esp / 2 - bw2 / 2),
centroY - escala * (h / 2 - hf));
        pontos[8] = new PointF(centroX - escala * (esp / 2 - bw / 2),
centroY + escala * h / 2);
        pontos[9] = new PointF(centroX - escala * (esp / 2 + bw / 2),
centroY + escala * h / 2);
        pontos[10] = new PointF(centroX - escala * (esp / 2 + bw2 / 2),
centroY - escala * (h / 2 - hf));
        pontos[11] = new PointF(centroX - escala * bf / 2, centroY -
escala * (h / 2 - hf));

        imagemSecao.DrawPolygon(Pens.Black, pontos);

        secaoTransversal = new Secao(h, bw, bw2, hf, bf, esp);
        break;

    case 3:

        pontos[0] = new PointF(centroX - escala * bf / 2, centroY - escala
* h / 2);
        pontos[1] = new PointF(centroX + escala * bf / 2, centroY - escala
* h / 2);
        pontos[2] = new PointF(centroX + escala * bf / 2, centroY - escala
* (h / 2 - hf));
        pontos[3] = new PointF(centroX + escala * bw / 2, centroY - escala
* (h / 2 - hf));
        pontos[4] = new PointF(centroX + escala * bw2 / 2, centroY +
escala * (h / 2 - hf));
        pontos[5] = new PointF(centroX + escala * (bf / 2), centroY +
escala * (h / 2 - hf));
        pontos[6] = new PointF(centroX + escala * (bf / 2), centroY +
escala * h / 2);
        pontos[7] = new PointF(centroX - escala * (bf / 2), centroY +
escala * h / 2);
        pontos[8] = new PointF(centroX - escala * (bf / 2), centroY +
escala * (h / 2 - hf));
        pontos[9] = new PointF(centroX - escala * bw2 / 2, centroY +
escala * (h / 2 - hf));
        pontos[10] = new PointF(centroX - escala * bw / 2, centroY -
escala * (h / 2 - hf));
        pontos[11] = new PointF(centroX - escala * (bf / 2), centroY -
escala * (h / 2 - hf));

        imagemSecao.DrawPolygon(Pens.Black, pontos);
        break;
    default:

        imagemSecao.Clear(Color.White);
        break;
}

if (secaoTransversal is Secao)
{
    campoInercia.Text = secaoTransversal.inercia.ToString();
    campoW1.Text = secaoTransversal.w1.ToString();
    campoW2.Text = secaoTransversal.w2.ToString();
}

```

```

        calcularPesoProprio();
    }
}

/* -----
 * ----- CONFIGURAÇÃO DE CAMPOS -----
 * -----
 */
private void campoTipoSecao_SelectedIndexChanged(object sender, EventArgs e)
{
    campoH.Visible = true; textoH.Visible = true;
    campoBw.Visible = true; textoBw.Visible = true;

    switch (campoTipoSecao.SelectedIndex)
    {
        case 0:
            campoH.Visible = true; textoH.Visible = true;
            campoBw.Visible = true; textoBw.Visible = true;

            campoBw2.Visible = false; textoBw2.Visible = false;
            campoHf.Visible = false; textoHf.Visible = false;
            campoBf.Visible = false; textoBf.Visible = false;
            campoEsp.Visible = false; textoEsp.Visible = false;
            break;
        case 1:

            campoBw2.Visible = true; textoBw2.Visible = true;
            campoHf.Visible = true; textoHf.Visible = true;
            campoBf.Visible = true; textoBf.Visible = true;

            campoEsp.Visible = false; textoEsp.Visible = false;
            break;
        case 2:
            campoBw2.Visible = true; textoBw2.Visible = true;
            campoHf.Visible = true; textoHf.Visible = true;
            campoBf.Visible = true; textoBf.Visible = true;
            campoEsp.Visible = true; textoEsp.Visible = true;
            break;
        case 3:
            campoBw2.Visible = true; textoBw2.Visible = true;
            campoHf.Visible = true; textoHf.Visible = true;
            campoBf.Visible = true; textoBf.Visible = true;

            campoEsp.Visible = false; textoEsp.Visible = false;
            break;
        default:
            break;
    }

    desenharSecao();
}

// Função chamada ao se alterar o valor das medidas da seção
private void alterarSecao(object sender, EventArgs e)
{
    if (imagemSecao is Graphics && tabControl.SelectedIndex == 1)
        desenharSecao();
}
}
}

```

Arquivo: Secao.cs

```
namespace Protendido2
{
    public class Secao
    {
        public string tipo;

        // Seção original
        public float area;

        public float h, bw;
        public float bw2, hf, bf;
        public float esp;

        public float relGeometrica;

        public float inercia;
        public float y1, y2, w1, w2;
        public float centroide;

        // Seção homogeneizada
        public float areaH;
        public float inerciaH;
        public float y1h, y2h, w1h, w2h;
        public float centroideH;
        public float alfaP;

        public float alfa;

        public Secao(float h, float bw)
        {
            this.tipo = "Retangular";

            this.h = h;
            this.bw = bw;

            this.area = h * bw;

            this.relGeometrica = 2 * this.area / (2 * (h + bw));

            this.centroide = h / 2;

            this.y1 = this.h / 2;
            this.y2 = -this.h / 2;

            this.inercia = CalcularInercia("retângulo", h, bw);

            this.w1 = this.inercia / this.y1;
            this.w2 = this.inercia / this.y2;

            this.alfa = 1.5f;
        }

        public Secao(float h, float bw, float bw2, float hf, float bf)
        {
            this.tipo = "T";

            this.h = h;
            this.bw = bw;
            this.bw2 = bw2;
            this.hf = hf;
        }
    }
}
```

```

        this.bf = bf;

        this.area = hf * bf + (h - hf) * (bw + bw2) / 2;

        this.relGeometrica = 2 * this.area / (bf + 2 * hf + 2 *
(float)Math.Sqrt(Math.Pow((h - hf), 2) + Math.Pow(bw - bw2, 2)) + bw + (bf - bw2));

        this.centroide = CalcularCentroide(this.h, this.bw, this.bw2, this.hf,
this.bf);

        this.y1 = this.centroide;
        this.y2 = -(this.h - this.centroide);

        this.inercia = CalcularInercia(h, bw, bw2, hf, bf);

        this.w1 = this.inercia / this.y1;
        this.w2 = this.inercia / this.y2;

        this.alfa = 1.2f;
    }

    public Secao(float h, float bw, float bw2, float hf, float bf, float esp)
    {
        this.tipo = "Duplo T";

        this.area = hf * bf + 2 * (h - hf) * (bw + bw2) / 2;

        this.h = h;
        this.bw = bw;
        this.bw2 = bw2;
        this.hf = hf;
        this.bf = bf;
        this.esp = esp;

        this.relGeometrica = 2 * this.area / (bf + 2 * hf + 4 *
(float)Math.Sqrt(Math.Pow((h - hf), 2) + Math.Pow(bw - bw2, 2)) + 2 * bw + (bf - 2 *
bw2));

        this.centroide = CalcularCentroide(this.h, 2 * this.bw, 2 * this.bw2,
this.hf, this.bf);

        this.y1 = this.centroide;
        this.y2 = -(this.h - this.centroide);

        this.inercia = CalcularInercia(h, 2 * bw, 2 * bw2, hf, bf);

        this.w1 = this.inercia / this.y1;
        this.w2 = this.inercia / this.y2;

        this.alfa = 1.2f;
    }

    public Secao(List<PointF> pontos, string tipo)
    {
        this.tipo = tipo;
        this.area = CalcularArea(pontos);
        this.relGeometrica = this.area / CalcularPerimetro(pontos);
        this.centroide = CalcularCentroide(pontos);
        this.inercia = CalcularInercia(pontos);
    }
}

```

```

// Seção T
private static float CalcularCentroide(float h, float bw, float bw2, float hf,
float bf)
{
    float areaMesa = hf * bf;
    float yMesa = h - (hf / 2f);

    float areaAlmaRet = (h - hf) * (bw < bw2 ? bw : bw2);
    float yAlmaRet = (h - hf) / 2f;

    float areaAlmaTri = (h - hf) * (bw < bw2 ? bw2 - bw : bw - bw2) / 2f;
    float yAlmaTri = bw < bw2 ? (h - hf) * (2f / 3f) : (h - hf) / 3f;

    float somaArea = areaMesa + areaAlmaRet + areaAlmaTri;
    float somaPeso = yMesa*areaMesa + yAlmaRet*areaAlmaRet +
yAlmaTri*areaAlmaTri;

    return somaPeso/somaArea;
}

//Seção retangular
public static float CalcularInercia(string tipo, float h, float b)
{
    switch (tipo) {
        case "retângulo":
            return b * (float)Math.Pow(h, 3) / 12f;
        case "triângulo":
            return b * (float)Math.Pow(h, 3) / 36f;
        default:
            return 0;
    }
}

//Seção T
public static float CalcularInercia(float h, float bw, float bw2, float hf,
float bf)
{
    float centroide = CalcularCentroide(h, bw, bw2, hf, bf);

    float areaMesa = hf * bf;
    float yMesa = h - (hf / 2f);
    float inerciaMesa = CalcularInercia("retângulo", hf, bf);
    inerciaMesa += areaMesa * (float)Math.Pow(yMesa - centroide, 2);

    float areaAlmaRet = (h - hf) * (bw < bw2 ? bw : bw2);
    float yAlmaRet = (h - hf) / 2f;
    float inerciaAlmaRet = CalcularInercia("retângulo", (h - hf), Math.Min(bw,
bw2));
    inerciaAlmaRet += areaAlmaRet * (float)Math.Pow(yAlmaRet - centroide, 2);

    float areaAlmaTri = (h - hf) * (bw < bw2 ? bw2 - bw : bw - bw2) / 2f;
    float yAlmaTri = bw < bw2 ? (h - hf) * (2f / 3f) : (h - hf) / 3f;
    float inerciaAlmaTri = CalcularInercia("triângulo", (h - hf), (bw < bw2 ?
bw2 - bw : bw - bw2));
    inerciaAlmaTri += areaAlmaTri * (float)Math.Pow(yAlmaTri - centroide, 2);

    float inercia = inerciaMesa + inerciaAlmaRet + inerciaAlmaTri;

    //Console.WriteLine(inercia);
    return inercia;
}

```

```

    }

    // Calcular seção homogeneizada
    public void CalcularSecaoHomogeneizada(Concreto concreto, float dias, Aço aço,
float areaAço, float exc)
    {
        alfaP = aço.Ep / concreto.Ecsj(dias);

        areaH = area + areaAço * (alfaP - 1);

        centroideH = (area * centroide + areaAço * (alfaP - 1) * exc) / areaH;

        y1h = centroideH;
        y2h = -(h - centroideH);

        inerciaH = inercia + (area) * (float)Math.Pow(centroide - centroideH, 2) +
areaAço * (alfaP - 1) * (float)Math.Pow(centroideH - exc, 2);

        w1h = inerciaH / y1h;
        w2h = inerciaH / y2h;
    }

    // Calcular perímetro
    public static float CalcularPerimetro(List<PointF> pontos)
    {
        float perimetro = 0;
        for (int i = 0; i < pontos.Count - 2; i++)
            perimetro += (float)Math.Sqrt(Math.Pow((pontos[i+1].X - pontos[i].X),
2) + Math.Pow((pontos[i + 1].Y - pontos[i].Y),2));

        return perimetro;
    }

    // Calcular area por Gauss
    public static float CalcularArea(List<PointF> pontos)
    {
        float soma = 0;
        for (int i = 0; i < pontos.Count - 2; i++)
            soma += (pontos[i].X * pontos[i + 1].Y) - (pontos[i + 1].X *
pontos[i].Y);

        return Math.Abs(soma) / 2;
    }

    // Calcular centroide
    public static float CalcularCentroide(List<PointF> pontos)
    {
        float soma = 0;
        for (int i = 0; i < pontos.Count - 2; i++)
            soma += (pontos[i].Y + pontos[i + 1].Y) * (pontos[i].X * pontos[i +
1].Y - pontos[i + 1].X * pontos[i].Y);

        return soma / (6 * CalcularArea(pontos));
    }

    public static float CalcularInercia(List<PointF> pontos)
    {
        float soma = 0;
        for (int i = 0; i < pontos.Count - 2; i++)
            soma += (float)(Math.Pow(pontos[i].X, 2) + pontos[i].X * pontos[i +
1].X + Math.Pow(pontos[i + 1].X, 2)) * (pontos[i].X * pontos[i + 1].Y - pontos[i +
1].X * pontos[i].Y);
    }

```

```

        return (soma / 12) + CalcularArea(pontos) *
(float)Math.Pow(CalcularCentroide(pontos), 2);
    }
}

```

Arquivo: FormPrincipalForcas.cs

```

namespace Protendido2
{
    public partial class formPrincipal : Form
    {
        /* -----
        * -----  VARIÁVEIS -----
        * -----
        */

        // Carregamentos
        public float pesoProprio, cargaPermanente, cargaAcidental;
        public float vaoDaViga;

        // Coeficientes para combinações de esforços
        float phi0, phi1, phi2;
        float[] phi = { 0, 0, 0 };

        float[,] tabelaPhi = {
            { .5f, .4f, .3f },
            { .7f, .6f, .4f },
            { .8f, .7f, .6f },
            { .6f, .3f, .0f },
            { .6f, .5f, .3f } };

        // Tensões e forças de protensão
        float sigma1PInf, sigma2PInf, PInf_est, PInf, P0;

        /* -----
        * -----  DEFINIÇÃO DOS DADOS -----
        * -----
        */
        public void calcularPesoProprio()
        {
            pesoProprio = 25 * secaoTransversal.area / (100 * 100);
            campoModuloPP.Text = String.Format("{0:0.00}", pesoProprio);
        }

        private void CampoCargaPermanente_TextChanged(object sender, EventArgs e)
        {
            float.TryParse(campoCargaPermanente.Text, out cargaPermanente);
        }

        private void CampoCargaAcidental_TextChanged(object sender, EventArgs e)
        {
            float.TryParse(campoCargaAcidental.Text, out cargaAcidental);
        }

        private void CampoComprimento_TextChanged(object sender, EventArgs e)
        {
            float.TryParse(campoComprimento.Text, out vaoDaViga);
        }
    }
}

```

```

private void ListaTipoCargaAcidental_SelectedIndexChanged(object sender,
EventArgs e)
{
    for (int i = 0; i < 3; i++)
        phi[i] = tabelaPhi[listaTipoCargaAcidental.SelectedIndex, i];

    campoPhi0.Text = phi[0].ToString();
    campoPhi1.Text = phi[1].ToString();
    campoPhi2.Text = phi[2].ToString();
}

e) private void CampoTipoProtensao_SelectedIndexChanged(object sender, EventArgs
{
    switch (campoTipoProtensao.SelectedIndex)
    {
        case 0:
            break;
        case 1:
            campoLimiteDescompressao.Text = "Combinação quase permanente";
            campoLimiteFissuras.Text = "Combinação frequente";
            break;
        case 2:
            campoLimiteDescompressao.Text = "Combinação frequente";
            campoLimiteFissuras.Text = "Combinação rara";
            break;
    }
}

/* -----
* -----  CALCULAR FORÇAS ESTIMADAS  -----
* -----
*/

private void BtCalcularForcas_Click(object sender, EventArgs e)
{
    aviso.Text = "";

    // VERIFICAÇÕES
    if (pesoProprio == 0)
        aviso.Text = "Peso próprio não definido";

    if (vaoDaViga <= 0)
        aviso.Text = "Vão da viga não definido ou inválido";

    if (!(secaoTransversal is Secao))
        aviso.Text = "Nenhuma seção transversal definida";

    if (campoTipoProtensao.SelectedIndex == -1)
        aviso.Text = "Nenhum tipo de protensão selecionado";

    if (!(concreto is Concreto))
        aviso.Text = "Concreto não definido";

    if (exc <= 0)
        aviso.Text = "Excentricidade da armadura não definida ou inválida";

    if (!(aço is Aço))
        aviso.Text = "Nenhum tipo de aço selecionado";
}

```

```

if (aviso.Text != "")
    return;
// FIM DAS VERIFICAÇÕES

// Estado limite de fissuras

float sigma1PInf1, sigma1PInf2;

// Estado limite de descompressão
switch (campoLimiteDescompressao.Text)
{
    case "Combinação quase permanente":
        sigma1PInf1 = -(sigma1(pesoProprio, vaoDaViga / 2f,
secaoTransversal) +
            sigma1(cargaPermanente, vaoDaViga / 2f, secaoTransversal) +
            phi[2] * sigma1(cargaAcidental, vaoDaViga / 2f,
secaoTransversal)) / 1000;
        break;
    case "Combinação frequente":
        sigma1PInf1 = -(sigma1(pesoProprio, vaoDaViga / 2f,
secaoTransversal) +
            sigma1(cargaPermanente, vaoDaViga / 2f, secaoTransversal) +
            phi[1] * sigma1(cargaAcidental, vaoDaViga / 2f,
secaoTransversal)) / 1000;
        break;
    default:
        sigma1PInf1 = 0;
        break;
}

// Estado limite de formação de fissura
switch (campoLimiteFissuras.Text)
{
    case "Combinação frequente":
        sigma1PInf2 = -(sigma1(pesoProprio, vaoDaViga / 2f,
secaoTransversal) +
            sigma1(cargaPermanente, vaoDaViga / 2f, secaoTransversal) +
            phi[1] * sigma1(cargaAcidental, vaoDaViga / 2f,
secaoTransversal)) / 1000 + // MPa
            secaoTransversal.alfa * concreto.fctkinf;
        break;
    case "Combinação rara":
        sigma1PInf2 = -(sigma1(pesoProprio, vaoDaViga / 2f,
secaoTransversal) +
            sigma1(cargaPermanente, vaoDaViga / 2f, secaoTransversal) +
            phi[1] * sigma1(cargaAcidental, vaoDaViga / 2f,
secaoTransversal)) / 1000 + // MPa
            secaoTransversal.alfa * concreto.fctkinf;
        break;
    default:
        sigma1PInf2 = 0;
        break;
}

// Grava a tensão de protensão na fibra inferior
sigma1PInf = Math.Min(sigma1PInf1, sigma1PInf2);
campoTensaoPInf.Text = String.Format("{0:0.00} MPa", sigma1PInf);

```

```

        // Força de protensão estimada
        PInf_est = -(sigma1PInf / 10f) / (1f / secaoTransversal.area +
(secaoTransversal.centroide - exc) / secaoTransversal.w1);
        campoPInf_est.Text = String.Format("{0:0.00} kN", PInf_est);

        // Força de protensão adotada
        PInf = PInf_est - PInf_est % 5 + 5;
        campoPInf.Text = String.Format("{0:0.00} kN", PInf);

        // Força de protensão P0
        P0 = PInf / .75f;
        campoP0.Text = String.Format("{0:0.00} kN", P0);

        // Área de aço estimada (variável declarada no arquivo
FormPrincipalArmaduras)
        a_est = P0 / aço.sigmaMax;
        campoAreaAcoEstimada.Text = string.Format("{0:0.00} cm²", a_est);
        campoAreaEstimada.Text = string.Format("{0:0.00} cm²", a_est);
    }

    /* -----
    * ----- FUNÇÕES IMPORTANTES -----
    * -----
    */

    private float momento(float moduloCarga, float posicao)
    {
        float reacao1 = moduloCarga * vaoDaViga / 2;
        return reacao1 * posicao - moduloCarga * (posicao * posicao) / 2f;
    }

    private float sigma1(float moduloCarga, float posicao, Secao secao)
    {
        return momento(moduloCarga, posicao) / secao.w1 * (float)(Math.Pow(100,
3));
    }

    private float sigma2(float moduloCarga, float posicao, Secao secao)
    {
        return momento(moduloCarga, posicao) / secao.w2 * (float)(Math.Pow(100,
3));
    }
}
}
}

```

Arquivo: FormPrincipalArmaduras.cs

```

namespace Protendido2
{
    public partial class formPrincipal : Form
    {
        /* -----
        * ----- VARIÁVEIS -----
        * -----
        */

        // Dados da armadura pré-dimensionamento
        public float a_est;

        // Dados da armadura dimensionada
        public float a_ef;
    }
}

```

```

// Novas tensões e forças de protensão
public float sigmaPi, Pi;
public float Pa;
public float sigmaPa;

public float sigmacPa; // Tensão no concreto adjacente à armadura
public float sigmaP0;
public float P0_calc;

/* -----
 * ----- FUNÇÕES IMPORTANTES -----
 * -----
 */

private void CalcularArmaduraEfetiva()
{
    float num_cord, area, area_total;
    float diferenca;

    a_ef = 0;
    for (int i = 0; i < configArmaduras.RowCount - 1; i++)
    {
        if (configArmaduras["num_cord", i].Value != null &&
            configArmaduras["area_nominal", i].Value != null)
        {
            float.TryParse(configArmaduras["num_cord", i].Value.ToString(),
                out num_cord);
            float.TryParse(configArmaduras["area_nominal",
                i].Value.ToString(), out area);
            //area =
            (float)Convert.ToDouble(configArmaduras.Rows[i].Cells["num_cord"].Value) *
            (float)Convert.ToDouble(configArmaduras.Rows[i].Cells["area_nominal"].Value);
            area_total = num_cord * area;
            a_ef += area_total;
            configArmaduras["area_total", i].Value = area_total;
        }
    }
    campoAreaEfetiva.Text = String.Format("{0:0.00} cm²", a_ef);

    if (a_est != 0)
    {
        diferenca = ((a_ef - a_est) / a_est) * 100;
        campoDiffArea.Text = String.Format("{0:0.00}%", diferenca);
    }
}

private void BtCalcularSecaoHomogeneizada_Click(object sender, EventArgs e)
{
    aviso.Text = "";
    if (a_ef <= 0)
        aviso.Text = "Área de aço não definina ou inválida";

    if (a_est <= 0)
        aviso.Text = "Área de aço estimada não foi calculada ou é inválida";

    if (aviso.Text != "")
        return;

    // Cálculo da seção homogeneizada
    secaoTransversal.CalcularSecaoHomogeneizada(concreto, 2, aço, a_ef, exc);
    campoIh.Text = secaoTransversal.inerciaH.ToString();
    campow1h.Text = secaoTransversal.w1h.ToString();
}

```

```

        campow2h.Text = secaoTransversal.w2h.ToString();

        // Cálculo das novas forças de protensão
        sigmaPi = aço.sigmaPi();
        Pi = sigmaPi * a_ef;
        Pa = Pi * ((aço.relaxacao == "RB") ? 0.97f : 0.93f);

        sigmacPa = -(float)((Pa / secaoTransversal.areaH) + (Pa *
Math.Pow(secaoTransversal.centroideH - exc, 2) / secaoTransversal.inerciaH)); //
kN/cm²

        sigmaPa = Pa / a_ef; // kN/cm²

        sigmaP0 = sigmaPa - secaoTransversal.alfaP * sigmacPa; // kN/cm²

        P0_calc = sigmaP0 * a_ef;

        // Exibir valores
        campoSigmaPi.Text = String.Format("{0:0.00} kN/cm²", sigmaPi);
        campoPi.Text = String.Format("{0:0.00} kN", Pi);
        campoPa.Text = String.Format("{0:0.00} kN", Pa);
        campoP0calc.Text = String.Format("{0:0.00} kN", P0_calc);
        campoP0final.Text = String.Format("{0:0.00} kN", P0_calc);

    }

    /* -----
    * ----- CONFIGURAÇÕES DE CAMPOS -----
    * -----
    */
    private void ConfigArmaduras_Validated(object sender, EventArgs e)
    {
        CalcularArmaduraEfetiva();
    }

    private void ConfigArmaduras_CellValidated(object sender,
DataGridViewCellEventArgs e)
    {
        CalcularArmaduraEfetiva();
    }

}
}

```

Arquivo: FormPrincipalPerdas.cs

```

namespace Protendido2
{
    public partial class formPrincipal : Form
    {
        /*
        * VARIÁVEIS
        */

        // ALTURA FICTÍCIA
        float U; // umidade do ar
        float alfaFl = 3; // Coeficientes alfa para fluência e retração
        float alfaRt = 1;
    }
}

```

```

float t0_fl = 7 * 3;      // Tempo inicial (dias)
float t0_rt = 7 * 1;
float t_inf = 10000;     // Tempo final (dias)
float gamma;
float h_fict;            // Altura fictícia da viga

// RETRAÇÃO
float e1s;
float e2s;
float ecs_inf;
float ecs_t_t0;

// FLUÊNCIA
float phiA;
float phi1c;
float phi2c;
float phiF_inf;
float phiD_inf;
float phi_t_t0;

float Xc;                // Coeficiente de fluência do concreto
float Xp;                // Coeficiente de fluência do aço
float Psi_t_t0;
float x_t_t0;

// Cálculos de forças finais
float sigmaC_P0g;
float sigmaP_inf;
float P_inf;

// Variáveis auxiliares
float[,] valoresPhi1000 = {
    {0.5f, 0.0f, 0.0f, 0.0f, 0.0f, 0.0f},
    {0.6f, 3.5f, 2.3f, 2.5f, 1.0f, 1.5f},
    {0.7f, 7.0f, 2.5f, 5.0f, 2.0f, 4.0f},
    {0.8f, 12.0f, 3.5f, 8.5f, 3.0f, 7.0f}
};
float Phi1000;

/*
 * FUNÇÕES PRINCIPAIS
 */

private void CalcularPerdas()
{
    // Calcular espessura fictícia
    float.TryParse(campoUmidade.Text, out U);
    float.TryParse(campoAlfaFluencia.Text, out alfaFl);
    float.TryParse(campoAlfaRetracao.Text, out alfaRt);

    float.TryParse(campoTfinal.Text, out t_inf);
    t0_fl = 7 * alfaFl;
    t0_rt = 7 * alfaRt;

    gamma = 1 + (float)Math.Exp(-7.8 + 0.1 * U); // Tabela A.1 ABNT

    h_fict = gamma * secaoTransversal.relGeometrica;

    // Calcular perdas por retração do concreto
    float.TryParse(campoE1s.Text, out e1s);
    e2s = (float)(33f + 2f * h_fict) / (float)(20.8 + 3f * h_fict);

```

```

        ecs_inf = e1s * e2s;
        ecs_t_t0 = ecs_inf * (Bs(t_inf, h_fict) - Bs(t0_rt, h_fict));

        // Calcular perdas por fluência do concreto
        phiA = 0.8f * (1 - (concreto.fcj(t0_fl) / concreto.fcj(t_inf)));
        float.TryParse(campoPhi1c.Text, out phi1c);
        phi2c = (42f + h_fict) / (20f + h_fict);
        phiF_inf = phi1c * phi2c;
        phiD_inf = 0.4f;
        phi_t_t0 = phiA + phiF_inf * (Bf(t_inf, h_fict) - Bf(t0_fl, h_fict)) +
        phiD_inf * Bd(t_inf, t0_fl);
        Xc = 1f + .5f * phi_t_t0;

        // Perdas por fluência do aço
        float taxaProtensao = sigmaP0 / aço.fptk;
        bool relaxacaoAço = taxaProtensao > 0.5;
        if (!relaxacaoAço)
            Phi1000 = 0;
        else
        {
            int coluna;
            if (aço.tipo == "Cordoalhas")
            {
                if (aço.relaxacao == "RN")
                    coluna = 1;
                else
                    coluna = 2;
            }
            else if (aço.tipo == "Fios")
            {
                if (aço.relaxacao == "RN")
                    coluna = 3;
                else
                    coluna = 4;
            }
            else
            {
                coluna = 5;
            }

            for (int i = 0; i < valoresPhi1000.GetLength(0) - 1; i++)
                if (valoresPhi1000[i, 0] < taxaProtensao && valoresPhi1000[i + 1,
0] > taxaProtensao)
                    Phi1000 = (((taxaProtensao - valoresPhi1000[i, 0]) /
(valoresPhi1000[i + 1, 0] - valoresPhi1000[i, 0])) * (valoresPhi1000[i + 1, coluna] -
valoresPhi1000[i, coluna]) + valoresPhi1000[i, coluna]) / 100; // divisão por 100,
trata-se de porcentagem
            }

            Psi_t_t0 = Phi1000 * (float)Math.Pow((t_inf - t0_fl) / 41.67f, .15f);
            // Fórmula do item 9.6.3.4.5 da NBR 6118:2014
            x_t_t0 = -(float)Math.Log(1 - Psi_t_t0);
            Xp = 1 + x_t_t0;

            // Tensão de compressão no concreto no centroide da armadura de protensão
            float n = 1 + secaoTransversal.area *
(float)Math.Pow(secaoTransversal.centroide - exc, 2) / secaoTransversal.inercia;
            float Mg = momento(pesoProprio, vaoDaViga / 2) + momento(cargaPermanente,
vaoDaViga/2);
            sigmaC_P0g = ((Mg * 100f * (secaoTransversal.centroide - exc) /
secaoTransversal.inercia) - (P0_calc / secaoTransversal.area) * n)*10; // MPa

```

```

        float deltaSigmaP = (aço.Ep * ecs_t_t0 - secaoTransversal.alfaP *
sigmaC_P0g * phi_t_t0 + sigmaP0 / 100 * x_t_t0)
        / (Xp + Xc * secaoTransversal.alfaP * n * (a_ef /
secaoTransversal.area)); // MPa
        sigmaP_inf = sigmaP0 - deltaSigmaP / 10; // kN/cm²
        P_inf = sigmaP_inf * a_ef;

// Exibir resultados nos campos
campot0Fluencia.Text = t0_fl.ToString("0 dias");
campot0Retracao.Text = t0_rt.ToString("0 dias");
campoGamma.Text = gamma.ToString("0.00");
campoHfict.Text = h_fict.ToString("0.00 cm");

campoE2s.Text = e2s.ToString("0.0000");
campoEcsinf.Text = ecs_inf.ToString("0.0000");
campoEcst_t0.Text = ecs_t_t0.ToString("0.00e-0");

campoPhi2c.Text = phi2c.ToString("0.000");
campoPhiFinf.Text = phiF_inf.ToString("0.000");
campoPhiA.Text = phiA.ToString("0.000");
campoPhiD_inf.Text = phiD_inf.ToString("0.000");
campoPhit_t0.Text = phi_t_t0.ToString("0.000");

campoXc.Text = Xc.ToString("0.000");
campoXp.Text = Xp.ToString("0.000");

campoPerdas.Text = (deltaSigmaP / 10 * a_ef).ToString("0.00 kN");
campoPinfFinal.Text = P_inf.ToString("0.00 kN");
}

/*
 * FUNÇÕES AUXILIARES
 */

private float Bs(float t, float h_fict)
{
    float h = (float)(h_fict / 100f > 1.6 ? 1.6 : (h_fict / 100f < .05 ? .05 :
h_fict / 100));

    float A = 40;
    float B = (float)(116f * Math.Pow(h, 3) - 282f * Math.Pow(h, 2) + 220f * h
- 4.8);
    float C = (float)(2.5 * Math.Pow(h, 3) - 8.8 * h + 40.7);
    float D = (float)(-75f * Math.Pow(h, 3) + 585f * Math.Pow(h, 2) + 496f * h
- 6.8);
    float E = (float)(-169f * Math.Pow(h, 4) + 88f * Math.Pow(h, 3) + 584f *
Math.Pow(h, 2) - 39f * h + 0.8);

    float Bs = (float)((Math.Pow(t / 100f, 3) + A * Math.Pow(t / 100f, 2) + B
* (t / 100f)) / (Math.Pow(t / 100f, 3) + C * Math.Pow(t / 100f, 2) + D * (t / 100f) +
E));

    return Bs;
}

private float Bf(float t, float h_fict)
{
    float h = (float)(h_fict / 100f > 1.6 ? 1.6 : (h_fict / 100f < .05 ? .05 :
h_fict / 100));

    float A = (float)(42f * Math.Pow(h, 3) - 350f * Math.Pow(h, 2) + 588f * h
+ 113);

```

```

        float B = (float)(768f * Math.Pow(h, 3) - 3060f * Math.Pow(h, 2) + 3234f *
h - 23);
        float C = (float)(-200f * Math.Pow(h, 3) + 13f * Math.Pow(h, 2) + 1090f *
h + 183f);
        float D = (float)(7579f * Math.Pow(h, 3) - 31916f * Math.Pow(h, 2) +
35343f * h + 1931f);

        float Bf = (float)((Math.Pow(t, 2) + A * t + B) / (Math.Pow(t, 2) + C * t
+ D));

        return Bf;
    }

    private float Bd(float t, float t0)
    {
        return (t - t0 + 20f) / (t - t0 + 70);
    }

    /*
     * BOTÕES
     */
    private void BtCalcularPerdas_Click(object sender, EventArgs e)
    {
        CalcularPerdas();
    }
}
}

```