UCLouvain

École polytechnique de Louvain

Optimisation des gîtages de plancher sur base d'une reformulation adimensionnelle des critères de l'EC5: développement d'une application hybride.

Auteur: **Benjamin Moeremans**
Promoteur: **Pierre Latteur**
Lecteurs: **Catherine Doneux**, **Michel Monhonval**, **Jean-François Rondeaux**
Année académique 2018–2019
Master [120] : ingénieur civil des constructions

Remerciements

Je tiens à adresser mes plus sincères remerciements au Professeur Pierre Latteur, promoteur de ce mémoire, pour l'inspiration, l'aide et le temps qu'il m'a consacrés. Non seulement, il a réussi à me transmettre durablement son vif intérêt pour les structures, mais ses conseils judicieux, son questionnement continu et son aide bienveillante ont été très positifs dans ma démarche. Sans lui, ce mémoire n'aurait pas été ce qu'il est aujourd'hui. Je le remercie également pour la confiance et la liberté qu'il m'a accordées, en particulier dans le volet informatique de ce mémoire.

Je tiens également à remercier particulièrement Monsieur Michel Monhonval pour son engagement et pour le temps qu'il a consacrés à ce mémoire, pour tous ses précieux conseils et pour m'avoir permis de bénéficier de sa passion et de sa large expérience dans le domaine du bois et du génie civil.

Résumé

Le bois est un matériau très largement utilisé dans la réalisation de structures, en particulier de gîtages. Le dimensionnement de ces gîtages est un élément capital de la conception d'une construction, puisqu'il est la structure soutenant les planchers. Les professionnels semblent pourtant souvent se limiter au calcul et à la vérification d'une option de dimensionnement, ou d'un nombre fort limité d'options, sans chercher la solution qui serait, pour eux, optimale. Les logiciels de calcul disponibles sur le marché ne permettent d'ailleurs généralement que la vérification d'une option de dimensionnement. Ce mémoire comprend un volet de recherche destiné à analyser de manière théorique un gîtage, la configuration et le dimensionnement qui minimisent le volume de bois utilisé. A cette fin, est établie une reformulation adimensionnelle des critères de l'Eurocode 5, basée sur une publication de Descamps, Monhonval et Latteur (2018). Cette reformulation rassemble les différents critères de dimensionnement en un critère unique et permet le développement d'un indicateur adimensionnel du volume de bois d'un gîtage, conduisant à des lois générales valables pour tous les problèmes de plancher. Cette recherche d'optimisation du volume de bois d'un gîtage se poursuit dans la réalisation d'une application hybride pour smartphone et tablette, basée sur la reformulation adimensionnelle des critères précités et destinée à contribuer à l'amélioration de la réalisation de gîtages. Cette application permet d'obtenir immédiatement la configuration du gîtage minimisant le volume de bois à mettre en œuvre d'une part, et celle minimisant le coût total de réalisation d'autre part.

Abstract

Wood is a widely used material in the construction of structures, particularly subfloor framing structures. Due to their role in supporting the floor, the design of these subfloor structures is a crucial element in the conception of any building. Despite their importance, professionals often reduce their calculation and verification to a single or a limited number of design options, missing the opportunity for the optimal solution. Calculation software currently available typically only allows the verification of a design option. This master's thesis includes a research component, intended to analyze a subfloor structure from a theoretical perspective, its required configuration and sizing to minimize wood volume. In order to do so, a dimensionless reformulation of the Eurocode 5 design criteria is established, based on a publication of Descamps, Monhonval, and Latteur (2018). This reformulation gathers the different design criteria into a single criterion, allowing for the development of a dimensionless indicator for the volume of wood used in subfloor structures and leading to the establishment of general laws applicable for all flooring problems. This research about the optimization of wood volume used for a subfloor structure is further pursued with the development of a cross-platform application for smartphones and tablets. The app development is based on the dimensionless reformulation of the abovementioned criteria and aims at improving the realization of such structures. The application allows to immediately obtain the configuration of the subfloor structure that, on the one hand, minimizes the volume of wood to be used and, on the other hand, provides the lowest cost configuration for such structure.

Table des matières

Introduction **1**

1 Description du cadre de la recherche **3**
1.1 Objectifs du mémoire . . . 3
1.2 Hypothèses et définitions générales . . . 5
1.2.1 Hypothèses de dimensionnement . . . 5
1.2.2 Notations principales . . . 6
1.2.3 Types de configurations considérées . . . 7

2 Synthèse des éléments des Eurocodes utiles pour ce mémoire **9**
2.1 Eléments de la norme EN-1990 . . . 9
2.1.1 Vérification aux Etats Limites Ultimes (ELU) . . . 10
2.1.2 Vérification aux Etats Limites de Service (ELS) . . . 10
2.2 Eléments de la norme EN-1991 . . . 11
2.3 Eléments de la norme EN-1995 . . . 12
2.3.1 Eléments nécessaires aux calculs . . . 12
2.3.2 Le calcul à l'Etat Limite Ultime (ELU) . . . 13
2.3.3 Le calcul à l'Etat Limite de Service (ELS) . . . 15

3 Reformulation des critères de dimensionnement **19**
3.1 Résumé de l'article "An assessment methodology for timber beams in a restoration context, based on a dimensionless formulation of EC5 design criteria" . . . 19
3.1.1 Facteurs adimensionnels développés dans le cadre de l'article . . . 20
3.1.2 Reformulation adimensionnelle des critères de dimensionnement ELU et ELS 20
3.1.3 Critère de dimensionnement global . . . 23
3.2 Définition de nouveaux paramètres . . . 23
3.2.1 Paramètre Y_d . . . 23
3.2.2 Paramètres B et θ . . . 24
3.2.3 Paramètre C . . . 24
3.3 Gammes de variations des différents paramètres . . . 24
3.3.1 Hypothèses et approximations de base . . . 25
3.3.2 Paramètre Y_d . . . 25
3.3.3 Paramètre k_{inst} . . . 25
3.3.4 Paramètre k_{fin} . . . 26
3.3.5 Paramètre k_{M-V} . . . 27
3.3.6 Paramètre θ . . . 27
3.3.7 Paramètre B . . . 27
3.3.8 Paramètre C . . . 27
3.4 Adaptation des critères au cas d'un gîtage : élément d'optimisation du volume de bois utilisé . . . 27
3.4.1 Calcul à l'ELU . . . 28

3.4.2 Calcul à l'ELS . . . 29
3.4.3 Conclusion . . . 30
3.5 Ultime reformulation des critères selon la hauteur de section nécessaire . . . 30
3.5.1 Critère ELU de flexion . . . 31
3.5.2 Critère ELU de cisaillement . . . 31
3.5.3 Critères ELS : flèche instantanée et flèche finale . . . 32
3.5.4 Expression générale du critère de dimensionnement . . . 33

4 Expression générale du volume de bois et optimisation 35
4.1 Développement de l'expression de l'indicateur du volume de bois . . . 35
4.1.1 Volume total de bois d'un gîtage . . . 35
4.1.2 Première approche . . . 36
4.1.3 Deuxième approche . . . 37
4.1.4 Troisième approche . . . 38
4.1.5 Choix de l'expression finale de l'indicateur de volume . . . 38
4.2 Hypothèse sur la classe de résistance du bois . . . 39
4.3 Analyse graphique de W . . . 39
4.3.1 Graphiques pour $\beta = 1,0$ . . . 40
4.3.2 Graphiques pour $\beta = 0,5$ . . . 42
4.3.3 Graphiques pour $\beta = 2,0$ . . . 44
4.3.4 Analyse des graphes 4.1 à 4.12 . . . 46
4.4 Introduction de concepts basés sur l'indicateur de volume : volume minimal théorique et rendement d'un gîtage . . . 48
4.5 Comparaison des sens court et long . . . 48
4.5.1 Comparaison graphique . . . 48
4.5.2 Comparaison analytique . . . 54
4.6 Conclusion . . . 56

5 Conception de l'application 57
5.1 Environnement de programmation : Ionic Framework . . . 57
5.1.1 Origine d'Ionic . . . 57
5.1.2 Dépendances d'Ionic . . . 58
5.1.3 Programmation avec Ionic . . . 58
5.2 Structure de l'application . . . 58
5.2.1 Description générale . . . 59
5.2.2 Modèles . . . 60
5.2.3 Services . . . 63
5.2.4 Pages . . . 65
5.3 Algorithme de calcul . . . 76
5.3.1 Variables globales et méthodes de la page `SolutionPage` . . . 77
5.3.2 Choix de la solution optimale . . . 81
5.3.3 Solution la moins volumineuse . . . 86
5.3.4 Solution la moins coûteuse . . . 94
5.3.5 Recours à une poutre intermédiaire . . . 96
5.4 Mode d'emploi de l'application . . . 99
5.4.1 Page d'accueil . . . 99
5.4.2 Gestion des plages de sections . . . 100
5.4.3 Calculer les solutions optimales d'un problème de gîtage . . . 104
5.5 Pistes d'enrichissement pour un développement futur . . . 114

6 Etude de cas de planchers et analyse des résultats **117**
6.1 Hypothèses sur les plages de sections . . . 117
6.1.1 Plage de sections en bois massif . . . 117
6.1.2 Plage de sections en BLC . . . 119
6.2 Résolution d'un problème type . . . 119
6.2.1 Description du problème . . . 119
6.2.2 Résolution analytique . . . 119
6.2.3 Résolution à l'aide de l'application . . . 122
6.3 Comparaison graphique des solutions pour différentes dimensions de plancher . . . 127
6.3.1 Rendement des solutions . . . 127
6.3.2 Comparaison du coût des solutions optimales . . . 128
6.3.3 Comparaison du volume des solutions optimales . . . 129
6.3.4 Comparaison des solutions optimales pour le sens court et le sens long . . . 130
6.3.5 Comparaison des solutions optimales des cas avec et sans poutre intermédiaire 132

Conclusion **135**

Bibliographie **136**

A Tableaux de valeurs issus des Eurocodes **139**
A.1 Valeurs des coefficients ψ (EN1990) . . . 139
A.2 Charges d'exploitation pour les bâtiments (EN1991) . . . 139
A.3 Valeurs de k_{def} (EN1995) . . . 140
A.4 Valeurs de k_{mod} (EN1995) . . . 140
A.5 Classes de durée de chargement (EN1995) . . . 140
A.6 Valeurs de γ_M (EN1995) . . . 140
A.7 Valeurs maximales de flèche (EN1995) . . . 141
A.8 Valeurs caractéristiques des classes de résistance de bois massif (EN338) . . . 141
A.9 Valeurs caractéristiques des classes de résistance de bois lamellé-collé (EN14080) 142

B Code source de l'application **143**
B.1 Modèles . . . 143
B.1.1 Paramètres du problème . . . 143
B.1.2 Largeur de section et coûts associés aux hauteurs disponibles . . . 144
B.1.3 Plage de largeurs de sections . . . 144
B.2 Services . . . 144
B.2.1 Service des Eurocodes . . . 144
B.2.2 Service de base de données des plages de sections . . . 146
B.3 Pages . . . 149
B.3.1 Page d'accueil . . . 149
B.3.2 Page d'informations à propos de l'application . . . 151
B.3.3 Liste des plages de sections . . . 152
B.3.4 Liste des largeurs d'une plage de sections . . . 154
B.3.5 Liste des hauteurs associées à une largeur d'une plage de sections . . . 156
B.3.6 Formulaire d'ajout d'une plage de sections . . . 159
B.3.7 Formulaire d'ajout d'une largeur à une plage de sections . . . 160
B.3.8 Formulaire d'ajout d'une hauteur associée à une largeur d'une plage de sections . . . 163
B.3.9 Formulaire des paramètres pour un calcul basé sur les Eurocodes . . . 165
B.3.10 Formulaire des paramètres pour un calcul personnalisé . . . 172
B.3.11 Page des solutions optimales en termes de volume et de coût . . . 179

Liste des symboles et abréviations

Symbole	[Unité [1]]	Signification

Symboles latins majuscules

A	$[m^2]$	Aire d'une section rectangulaire, tenant compte du facteur de fissuration ($= k_{cr}bh$)
A_v	$[m^2]$	Aire réduite d'une section rectangulaire ($= 5bh/6$)
B	[/]	Facteur adimensionnel développé dans le cadre de ce mémoire
C	[/]	Facteur adimensionnel développé dans le cadre de ce mémoire
E	[MPa]	Module d'élasticité
G	[MPa]	Module de cisaillement
G_j	[kN/m]	Action permanente j
I_y	$[m^4]$	Module d'inertie selon l'axe fort
L	$[m]$	Longueur (=portée) d'une poutre
M_{Ed}	[kNm]	Valeur de calcul d'un moment de flexion
P	$[m]$	Dimension du plancher qui est perpendiculaire à la portée des poutres
Q_i	[kN/m]	Action variable i
$Q_{ELS,inst}$	[kN/m]	Charge surfacique uniformément répartie issue de la combinaison de charge à l'ELS en début de vie d'un ouvrage
$Q_{ELS,fin}$	[kN/m]	Charge surfacique uniformément répartie issue de la combinaison de charge à l'ELS en fin de vie d'un ouvrage
Q_{ELU}	[kN/m]	Charge surfacique uniformément répartie issue de la combinaison de charge à l'ELU
S	$[m^2]$	Aire d'une section transversale d'une poutre
V_{Ed}	[kN]	Valeur de calcul d'un effort tranchant
Vol	$[m^3]$	Volume total de bois mis en œuvre
$Vol_{min,th}$	$[m^3]$	Volume minimal théorique de bois à mettre en œuvre
W_{el}	$[m^3]$	Module de flexion élastique
W_{L^3}	[/]	Indicateur de volume valant Vol/L^3
W_{LP}	[/]	Indicateur de volume valant $Vol/(LP)^{(3/2)}$
W_β	[/]	Indicateur de volume valant $Vol/(L^2P)$
W	[/]	Indicateur de volume (égal à W_{LP})
W_c	[/]	Indicateur de volume correspondant à une disposition des poutres selon le "sens court"
W_l	[/]	Indicateur de volume correspondant à une disposition des poutres selon le "sens long"
X_d	[·]	Valeur de calcul d'une propriété de résistance X_k
X_k	[·]	Valeur caractéristique d'une propriété de résistance d'un matériau
Y_d	[/]	Facteur adimensionnel développé dans le cadre de ce mémoire
Z_d	[/]	Facteur adimensionnel développé dans le cadre de l'article [1]

1. La notation [/] signifie "adimensionnel" ; la notation [·] signifie "unité variable".

Symboles latins minuscules

b	$[m]$	Largeur d'une section rectangulaire
b_{ef}	$[m]$	Largeur efficace d'une section rectangulaire ($= k_{cr}h$)
b/e	$[/]$	Taux de remplissage d'un plancher
e	$[m]$	Entraxe
$f_{ELU,f}$	$[/]$	Facteur adimensionnel lié au critère ELU de résistance à la flexion
$f_{ELU,v}$	$[/]$	Facteur adimensionnel lié au critère ELU de résistance au cisaillement
f_{ELS}	$[/]$	Facteur adimensionnel lié aux critères ELS de flèche
$f_{m,d}$	[MPa]	Valeur de calcul de la résistance à la flexion
$f_{m,k}$	[MPa]	Valeur caractéristique de la résistance à la flexion
$f_{v,d}$	[MPa]	Valeur de calcul de la résistance au cisaillement
$f_{v,k}$	[MPa]	Valeur caractéristique de la résistance au cisaillement
g_j	$[\mathrm{kN}/m^2]$	Action permanente j
h	$[m]$	Hauteur d'une section rectangulaire
$h_{ELU,f}$	$[m]$	Hauteur de section minimale satisfaisant le critère ELU de résistance à la flexion
$h_{ELU,v}$	$[m]$	Hauteur de section minimale satisfaisant le critère ELU de résistance au cisaillement
h_{ELS}	$[m]$	Hauteur de section minimale satisfaisant les critères ELS de flèche
h_{min}	$[m]$	Hauteur de section minimale satisfaisant tous les critères de dimensionnement
k_{cr}	$[/]$	Facteur de fissuration, égal à $0,67$ pour du bois massif et du BLC
k_{crit}	$[/]$	Facteur utilisé pour le déversement latéral
k_{def}	$[/]$	Facteur de déformation
k_h	$[/]$	Facteur de hauteur
k_{fin}	$[/]$	Facteur adimensionnel développé dans le cadre de l'article [1]
k_{inst}	$[/]$	Facteur adimensionnel développé dans le cadre de l'article [1]
k_{M-V}	$[/]$	Indicateur de transition M-V
k_{mod}	$[/]$	Facteur de modification
k_{sys}	$[/]$	Facteur système
n	$[/]$	Nombre de poutres en bois à mettre en oeuvre
$q_{ELS,inst}$	$[\mathrm{kN}/m]$	Charge linéique uniformément répartie issue de la combinaison de charge à l'ELS en début de vie d'un ouvrage
$q_{ELS,fin}$	$[\mathrm{kN}/m]$	Charge linéique uniformément répartie issue de la combinaison de charge à l'ELS en fin de vie d'un ouvrage
q_{ELU}	$[\mathrm{kN}/m]$	Charge linéique uniformément répartie issue de la combinaison de charge à l'ELU
q_i	$[\mathrm{kN}/m^2]$	Action variable i
w_c	$[m]$	Contreflèche
w_{creep}	$[m]$	Flèche de fluage
w_{fin}	$[m]$	Flèche finale
w_{inst}	$[m]$	Flèche instantanée
$w_{net,fin}$	$[m]$	Flèche résultante finale

Symboles grecs

β	[/]	Rapport de portée
β_c	[/]	Rapport de portée correspondant à une disposition des poutres selon le "sens court"
β_l	[/]	Rapport de portée correspondant à une disposition des poutres selon le "sens long"
$\gamma_{G,j}$	[/]	Coefficient pour la valeur de combinaison d'une action permanente j
$\gamma_{Q,i}$	[/]	Coefficient pour la valeur de combinaison d'une action variable i
γ_M	[/]	Coefficient partiel pour la propriété des matériaux
η_{inst}	[/]	Coefficient de limite de flèche instantanée
η_{fin}	[/]	Coefficient de limite de flèche finale
η_t	[/]	Désigne à la fois η_{inst} et η_{fin}
θ	[/]	Facteur adimensionnel développé dans le cadre de ce mémoire
μ	[/]	Rendement d'un gîtage
$\psi_{0,i}$	[/]	Facteur pour la valeur de combinaison d'une action variable i
$\psi_{1,i}$	[/]	Facteur pour la valeur fréquente d'une action variable i
$\psi_{2,i}$	[/]	Facteur pour la valeur quasi-permanante d'une action variable i
$\sigma_{m,Ed}$	[MPa]	Valeur de calcul de la contrainte de flexion
τ_{Ed}	[MPa]	Valeur de calcul de la contrainte de cisaillement

Autres symboles

$\triangleq$	Egalité par définition
$\Leftrightarrow$	Equivalence
$max()$	Fonction maximum
[/]	Adimensionnel
$[\cdot]$	Unité variable

Abréviations

BLC	Bois Lamellé-Collé
CLI	Command Line Interface
EC	Eurocode
ELS	Etat Limite de Service
ELU	Etat Limite Ultime
OSB	Oriented Strand Board (panneau de particules orientées)
SDK	Software Development Kit

Liste des figures

1.1 Photographies de gîtages en bois 3
1.2 Exemples de plans de configurations d'un gîtage 4
1.3 Phases de recherche 5
1.4 Dimensions de la surface de plancher 7
1.5 Deux cas décrits par le paramètre β 8
1.6 Recours à une poutre intermédiaire 8

2.1 Critères de dimensionnement définis par la norme EN1990 10
2.2 Variation de k_h avec la hauteur de l'élément, pour du bois massif et du bois lamellé-collé 14
2.3 Composantes de la flèche - Figure issue de la norme EN1995 §7.2 16

3.1 Illustration du facteur k_{M-V} pour des bois de classes C24 et D70 22
3.2 Exemples de planchers ayant différents taux de remplissage $\frac{b}{e}$ 24
3.3 Evolution de S en fonction de W_{el} pour différentes largeurs b 29
3.4 Evolution de S en fonction de I_y pour différentes largeurs b 30

4.1 Evolution de W en fonction de b/e pour différentes valeurs de n pour : $\beta = 1,0$ et $Y_d = 5 \times 10^{-5}$ 40
4.2 Evolution de W en fonction de b/e pour différentes valeurs de n pour : $\beta = 1,0$ et $Y_d = 40 \times 10^{-5}$ 40
4.3 Evolution de W en fonction de b/e pour différentes valeurs de n pour : $\beta = 1,0$ et $Y_d = 70 \times 10^{-5}$ 41
4.4 Evolution de W en fonction de b/e pour différentes valeurs de n pour : $\beta = 1,0$ et $Y_d = 100 \times 10^{-5}$ 41
4.5 Evolution de W en fonction de b/e pour différentes valeurs de n pour : $\beta = 0,5$ et $Y_d = 5 \times 10^{-5}$ 42
4.6 Evolution de W en fonction de b/e pour différentes valeurs de n pour : $\beta = 0,5$ et $Y_d = 40 \times 10^{-5}$ 42
4.7 Evolution de W en fonction de b/e pour différentes valeurs de n pour : $\beta = 0,5$ et $Y_d = 70 \times 10^{-5}$ 43
4.8 Evolution de W en fonction de b/e pour différentes valeurs de n pour : $\beta = 0,5$ et $Y_d = 100 \times 10^{-5}$ 43
4.9 Evolution de W en fonction de b/e pour différentes valeurs de n pour : $\beta = 2,0$ et $Y_d = 5 \times 10^{-5}$ 44
4.10 Evolution de W en fonction de b/e pour différentes valeurs de n pour : $\beta = 2,0$ et $Y_d = 40 \times 10^{-5}$ 44
4.11 Evolution de W en fonction de b/e pour différentes valeurs de n pour : $\beta = 2,0$ et $Y_d = 70 \times 10^{-5}$ 45
4.12 Evolution de W en fonction de b/e pour différentes valeurs de n pour : $\beta = 2,0$ et $Y_d = 100 \times 10^{-5}$ 45

4.13 Schémas de planchers illustrant les différentes valeurs de β et de n considérées pour les graphiques pour $b/e = 0,3$ 46
4.14 Illustration de la variation de n pour un même β et un même rapport $\frac{b_1}{e_1} = \frac{b_2}{e_2}$ . . 47
4.15 Comparaison de l'évolution de l'indicateur de volume en fonction du taux de remplissage b/e pour différentes valeurs de Y_d, pour $\beta = 2,0$ et $\beta = 0,5$ et pour $n = 30$ 49
4.16 Comparaison de l'évolution de l'indicateur de volume en fonction du taux de remplissage b/e pour différentes valeurs de Y_d, pour $\beta = 4,0$ et $\beta = 0,25$ et pour $n = 30$ 50
4.17 Evolution des indicateurs de volume minimaux pour les deux dispositions en fonction de β_c pour $Y_d = 5 \times 10^{-5}$ 51
4.18 Evolution des indicateurs de volume minimaux pour les deux dispositions en fonction de β_c pour $Y_d = 40 \times 10^{-5}$ 51
4.19 Evolution des indicateurs de volume minimaux pour les deux dispositions en fonction de β_c pour $Y_d = 70 \times 10^{-5}$ 52
4.20 Evolution des indicateurs de volume minimaux pour les deux dispositions en fonction de β_c pour $Y_d = 100 \times 10^{-5}$ 52
4.21 Evolution de $W_{l,min} - W_{c,min}$ en fonction de β_c pour $Y_d = 5 \times 10^{-5}$ et $Y_d = 100 \times 10^{-5}$ 53
4.22 Evolution de $W_{c,min}/W_{l,min}$ en fonction de β_c pour $Y_d = 5 \times 10^{-5}$ et $Y_d = 100 \times 10^{-5}$ 54

5.1 Structure générale de l'application (pages) 59
5.2 Représentation des modèles `Range` et `Width` et de leur contenu 62
5.3 Evénements du cycle de vie d'une page Ionic 66
5.4 Cycle de vie de la page `SolutionPage` 76
5.5 Exécution des méthodes `ionViewWillEnter()` et `ionViewDidEnter()` de la page `SolutionPage` 82
5.6 Exécution de la méthode `calculOptimalVolume()` de la page `SolutionPage` . . 87
5.7 Exécution de la méthode `calculOptimalCost()` de la page `SolutionPage` 95
5.8 Définition de deux nouvelles surfaces de plancher (en vert) de part et d'autre de la poutre intermédiaire (en rouge) 98
5.9 Page d'accueil de l'application 99
5.10 Liste des plages de sections 100
5.11 Liste des largeurs d'une plage de sections (1) 100
5.12 Liste des largeurs d'une plage de sections (2) 101
5.13 Liste des hauteurs associées à une largeur 101
5.14 Formulaire d'ajout d'une plage de sections (1) 102
5.15 Formulaire d'ajout d'une plage de sections (2) 102
5.16 Formulaire d'ajout d'une largeur à une plage de sections (1) 103
5.17 Formulaire d'ajout d'une largeur à une plage de sections (2) 103
5.18 Formulaire d'ajout d'une hauteur pour une largeur 104
5.19 Formulaire des paramètres nécessaires pour un calcul rapide (1) 104
5.20 Formulaire des paramètres nécessaires pour un calcul rapide (2) 105
5.21 Formulaire des paramètres nécessaires pour un calcul rapide (3) 105
5.22 Formulaire des paramètres nécessaires pour un calcul rapide (4) 106
5.23 Formulaire des paramètres nécessaires pour un calcul rapide (5) 106
5.24 Formulaire des paramètres nécessaires pour un calcul rapide (6) 107
5.25 Formulaire des paramètres nécessaires pour un calcul rapide (7) 107
5.26 Formulaire des paramètres nécessaires pour un calcul personnalisé (1) 108
5.27 Formulaire des paramètres nécessaires pour un calcul personnalisé (2) 108
5.28 Formulaire des paramètres nécessaires pour un calcul personnalisé (3) 109
5.29 Formulaire des paramètres nécessaires pour un calcul personnalisé (4) 109
5.30 Formulaire des paramètres nécessaires pour un calcul personnalisé (5) 110

5.31 Formulaire des paramètres nécessaires pour un calcul personnalisé (6) 110
5.32 Formulaire des paramètres nécessaires pour un calcul personnalisé (7) 111
5.33 Formulaire des paramètres nécessaires pour un calcul personnalisé (8) 111
5.34 Page des solutions 112
5.35 Exemples d'écrans de la page des solutions dans le cas où au moins une solution existe 112
5.36 Alerte dans le cas où aucune solution n'existe 113
5.37 Exemples d'écrans de la page des solutions dans le cas où aucune solution n'existe 113

6.1 Captures d'écrans de la page de solutions de l'application 123
6.2 Schéma des solutions optimales fournies par l'application 124
6.3 Rendement de la solution optimale en termes de volume en fonction d'une des dimensions du plancher 127
6.4 Economie de coût pour la solution optimale en termes de coût par rapport au coût de la solution optimale en termes de volume 128
6.5 Economie de volume pour la solution optimale en termes de volume par rapport au volume de la solution optimale en termes de coût 129
6.6 Economie de coût entre le coût minimal pour un gîtage disposé dans le sens court part rapport à un gîtage disposé dans le sens long 131
6.7 Economie de volume entre le volume minimal pour un gîtage disposé dans le sens court part rapport à un gîtage disposé dans le sens long 131
6.8 Economie de coût entre le coût minimal pour un gîtage sans poutre intermédiaire part rapport à un gîtage avec poutre intermédiaire 133
6.9 Economie de volume entre le volume minimal pour un gîtage avec poutre intermédiaire part rapport à un gîtage sans poutre intermédiaire 134

Liste des tableaux

5.1 Méthodes contenues dans le modèle `ProblemParams` 60
5.2 Données contenues dans le modèle `ProblemParams` 61
5.3 Données contenues dans le modèle `Width` 62
5.4 Données contenues dans le modèle `Range` 62
5.5 Données contenues dans le service `EurocodesService` 63
5.6 Constantes contenues dans le service `EurocodesService` 63
5.7 Méthode contenue dans le service `EurocodesService` 63
5.8 Méthodes contenues dans le service `RangesService` (partie 1) 64
5.9 Méthodes contenues dans le service `RangesService` (partie 2) 65
5.10 Description des événements du cycle de vie d'une page Ionic 66
5.11 Méthodes contenues dans le page `RangesPage` 68
5.12 Méthodes contenues dans le page `SingleRangePage` 69
5.13 Méthodes contenues dans le page `SingleWidthPage` 70
5.14 Contrôles du formulaire de la page `RangeFormPage` 71
5.15 Contrôles du formulaire de la page `WidthFormPage` 72
5.16 Contrôles du formulaire de la page `HeightFormPage` 73
5.17 Contrôles du formulaire de la page `CalcRapPage` 74
5.18 Contrôles du formulaire de la page `CalcPersPage` 75
5.19 Variables globales de la page `SolutionPage` (partie 1) 77
5.20 Variables globales de la page `SolutionPage` (partie 2) 78
5.21 Tuples contenant une solution 78
5.22 Méthodes contenues dans le page `SolutionPage` (partie 1) 79
5.23 Méthodes contenues dans le page `SolutionPage` (partie 2) 80
5.24 Méthodes contenues dans le page `SolutionPage` (partie 3) 81

6.1 Plage de sections de bois massif en épicéa (C24) 118
6.2 Plage de sections de bois lamellé-collé en épicéa (GL24h) 120
6.3 Solution optimale en termes de volume pour les différentes dispositions et comparaison à la solution optimale globale en termes de volume 124
6.4 Solution optimale en termes de coût pour les différentes dispositions et comparaison à la solution optimale globale en termes de coût 125

A.1 Valeur des coefficients ψ pour des catégories de charges d'exploitation des bâtiments 139
A.2 Catégories de charges d'exploitations pour les bâtiments 139
A.3 Valeurs de k_{def} 140
A.4 Valeurs de k_{mod} 140
A.5 Classes de durée de chargement 140
A.6 Coefficients partiels recommandés pour les propriétés des matériaux 140
A.7 Valeurs maximales de flèche 141
A.8 Extrait des valeurs caractéristiques des classes de résistance de bois massif 141
A.9 Extrait des valeurs caractéristiques des classes de résistance de bois lamellé-collé 142

Liste des codes informatiques

5.1 Extrait du code de `SolutionPage` correspondant à l'étape 1 de la figure 5.5 . . . 82
5.2 Extrait du code de `SolutionPage` correspondant à l'étape 2 de la figure 5.5 . . . 83
5.3 Extrait du code de `SolutionPage` correspondant à l'étape 7 de la figure 5.5 . . . 84
5.4 Extrait du code de `SolutionPage` correspondant aux étapes 12 et 13 de la figure 5.5 86
5.5 Extrait du code de la méthode `calculOptimalVolume()` de la page `SolutionPage` correspondant à l'étape 1 de la figure 5.6 . . . 88
5.6 Extrait du code de la méthode `calculOptimalVolume()` de la page `SolutionPage` correspondant à l'étape 2 de la figure 5.6 . . . 88
5.7 Extrait du code de la méthode `calculOptimalVolume()` de la page `SolutionPage` correspondant à l'étape 3 de la figure 5.6 . . . 88
5.8 Extrait du code de la méthode `calculOptimalVolume()` de la page `SolutionPage` correspondant aux étapes 4 à 20 de la figure 5.6 . . . 89
5.9 Extrait du code de la méthode `calculOptimalVolume()` de la page `SolutionPage` correspondant à l'étape 21 de la figure 5.6 . . . 94
5.10 Extrait du code de la méthode `calculOptimalCost()` de la page `SolutionPage` correspondant à l'étape 1 de la figure 5.7 . . . 94
5.11 Extrait du code de la méthode `calculOptimalCost()` de la page `SolutionPage` correspondant à l'étape 16 de la figure 5.7 . . . 96
5.12 Extrait du code de la méthode `calculOptimalCost()` de la page `SolutionPage` correspondant à l'étape 17 de la figure 5.7 . . . 96
5.13 Extrait du code des méthodes `calculOptimalVolume_inter()` et `calculOptimalCost_inter()` de la page `SolutionPage` correspondant à l'adaptation de l'étape 2 des figures 5.6 et 5.7 . . . 97
5.14 Extrait du code de la méthode `calculOptimalVolume_inter()` de la page `SolutionPage` correspondant à l'étape (21bis) . . . 97
5.15 Code de la méthode `calculOptimalVolumeBIS()` de la page `SolutionPage` . . . 99

B.1 `ProblemParams.ts` . . . 143
B.2 `Width.ts` . . . 144
B.3 `Range.ts` . . . 144
B.4 `eurocodes.service.ts` . . . 144
B.5 `ranges.service.ts` . . . 146
B.6 `home.ts` . . . 149
B.7 `home.html` . . . 150
B.8 `about.ts` . . . 151
B.9 `about.html` . . . 151
B.10 `ranges.ts` . . . 152
B.11 `ranges.html` . . . 153
B.12 `single-range.ts` . . . 154
B.13 `single-range.html` . . . 155
B.14 `single-width.ts` . . . 156
B.15 `single-width.html` . . . 157

B.16 range-form.ts 159
B.17 range-form.html 160
B.18 width-form.ts 160
B.19 width-form.html 162
B.20 height-form.ts 163
B.21 height-form.html 164
B.22 calculrap.ts 165
B.23 calculrap.html 168
B.24 calculpers.ts 172
B.25 calculpers.html 175
B.26 solution.ts 179
B.27 solution.html 188

Introduction

Si l'art de la construction et les matériaux utilisés ont fortement évolué au fil du temps, le bois est resté un matériau très largement utilisé, en particulier dans la confection de gîtages, structures destinées à soutenir un plancher. Pour ce type de constructions notamment, les ingénieurs de structures doivent respecter un nombre important de règles élaborées dans des codes harmonisés au niveau européen.

Le stage que j'ai pu réaliser durant ma première année de master sur le site du chantier *Handelsbeurs*, situé dans le cœur d'Anvers, m'a confronté aux nombreux défis d'ingénierie que représentent la réhabilitation et la transformation de bâtiments dont certaines parties datent du $17^{\text{ème}}$ siècle. J'ai notamment été amené à analyser des gîtages en bois vieux de plusieurs siècles. La conception de ces gîtages ne faisait à l'époque visiblement pas l'objet d'une étude approfondie : irrégularité dans les dimensions des poutres en bois, irrégularité dans leur espacement, etc. Si les professionnels semblent aujourd'hui accorder plus d'importance aux gîtages, ils se contentent en général du calcul et de la vérification d'une option de dimensionnement, ou d'un nombre fort limité d'options, sans chercher la solution qui serait, pour eux, optimale.

En effet, les logiciels de calcul disponibles sur le marché (citons par exemple SCIA Engineer ou MiTek Pamir) ne permettent en général que la vérification d'une option de dimensionnement. Dans tous les cas, même si certains logiciels comme Hout Info Bois vont jusqu'à proposer des options de dimensionnement, il n'y a apparemment aucun logiciel de calcul disponible qui permette de calculer immédiatement la solution optimale pour le dimensionnement d'un gîtage, que ce soit en termes de volume de bois mis en œuvre ou en termes de coût total.

Ce mémoire comprend un volet de recherche dont l'objectif est d'analyser de manière théorique un gîtage, sa configuration et le dimensionnement qui minimisent le volume de bois utilisé. A cette fin, il est utile de reformuler les critères de dimensionnement de structures en bois donnés dans la norme européenne. Une telle reformulation adimensionnelle permet de réduire considérablement le nombre de paramètres qui entrent en jeu, d'alléger les calculs en rassemblant tous les critères en un critère unique, et de déduire des lois générales valables pour tous les problèmes de planchers rectangulaires, et ce quels que soient les paramètres initiaux ou les dimensions du plancher. La reformulation adimensionnelle des critères est basée sur la récente publication "An assessment methodology for timber beams in a restauration context, based on a dimensionless formulation of EC5 design criteria" [1].

Ce mémoire comprend aussi la réalisation d'un outil informatique, basé sur la reformulation adimensionnelle des critères de dimensionnement. L'outil développé, destiné à contribuer à l'amélioration de la réalisation de gîtages, donne les solutions optimales (configuration et dimensionnement) permettant de minimiser le volume de bois mis en œuvre d'une part, et le coût de réalisation d'autre part. Il prend la forme d'une application hybride destinée à une utilisation sur smartphone et tablette, c'est-à-dire conçue pour fonctionner sur différents systèmes d'exploitation (Android, iOS et Windows Phone). Cela en fait un outil portable qui peut être utilisé directement sur chantier.

La description du cadre de l'étude et une présentation détaillée de celle-ci font l'objet du premier chapitre de ce mémoire. Les hypothèses et définitions générales y sont présentées.

La prise en compte des normes européennes de conception, de dimensionnement et de justification des structures, documentées dans les Eurocodes, est la base de tout travail de structure. C'est pourquoi le deuxième chapitre est consacré à une synthèse des éléments des Eurocodes qui sont utiles à notre étude.

Prenant comme point de départ l'article déjà mentionné ci-dessus, le troisième chapitre introduit de nouveaux paramètres adimensionnels qui permettent de poursuivre la reformulation des critères de l'Eurocode 5 de manière à pouvoir aborder le sujet de l'optimisation du volume de bois utilisé dans un gîtage.

L'objectif de l'étude théorique étant l'optimisation du volume de bois d'un gîtage, le quatrième chapitre détaille le développement d'un indicateur adimensionnel du volume de bois sur base des reformulations adimensionnelles des critères de dimensionnement. Cela permet de conduire une analyse graphique de laquelle il est possible de déduire l'approche générale à suivre, en théorie, pour minimiser le volume de bois d'un gîtage.

Le cinquième chapitre est, quant à lui, consacré à la conception de l'application. L'environnement de programmation choisi pour le développement de l'application y est décrit, ainsi que la structure de l'application et ses fonctionnalités. Une partie importante de ce chapitre est aussi consacrée à la description de l'algorithme, basé sur les recherches et analyses développées dans les chapitres précédents, permettant d'aboutir à la solution la moins volumineuse et à la solution la moins coûteuse. Un mode d'emploi de l'application et des pistes d'enrichissement de ses fonctionnalités sont aussi présentés en fin de ce chapitre.

La validation et l'analyse des résultats fournis par l'application sur base de problèmes concrets font l'objet du sixième chapitre. Différents concepts développés dans les chapitres précédents sont aussi illustrés sur base des problèmes résolus. Enfin une comparaison des solutions de différents problèmes et pour différentes configurations y est présentée.

Les deux objectifs de ce mémoire sont ainsi réalisés, apportant un développement dans la méthode de calcul d'un gîtage en bois et fournissant un outil informatique concret utilisable directement dans le travail des ingénieurs de structures.

Chapitre 1

Description du cadre de la recherche

Dans ce premier chapitre, nous décrivons d'abord l'objet de ce mémoire, les grandes lignes de la démarche que nous allons suivre et les hypothèses générales de notre recherche. Ensuite, nous introduisons les notations et définissons les conventions que nous utilisons dans ce mémoire.

1.1 Objectifs du mémoire

Par définition, le gîtage est la structure soutenant un plancher (figure 1.1). L'objectif de ce mémoire est de rechercher une méthode qui permettra d'optimiser le dimensionnement d'un gîtage en bois porteur d'un plancher rectangulaire. Une telle optimisation doit prendre en compte un grand nombre de paramètres et peut porter sur différentes variables. En effet, le dimensionnement optimal d'un gîtage pourrait être, par exemple, le plus économique, celui consommant le plus petit volume de bois, le plus léger, le moins épais, ..., ou encore celui qui optimiserait une combinaison de plusieurs facteurs.

FIGURE 1.1 – Photographies de gîtages en bois

Lorsque nous parlons de dimensionnement d'un gîtage, nous entendons bien sûr les dimensions des différents éléments en bois qui constituent ce gîtage, mais aussi la configuration dans laquelle sont disposés ces éléments. Une solution de dimensionnement d'un gîtage est donc définie par les paramètres suivants, qui seront expliqués en détail dans la suite de ce chapitre :

- la section des éléments en bois constituant le gîtage, qui est elle-même définie par une largeur b et une hauteur h pour une section rectangulaire ;
- le nombre d'éléments à mettre en œuvre ;
- la disposition de ces éléments.

La figure 1.2 montre quelques exemples de configurations possibles.

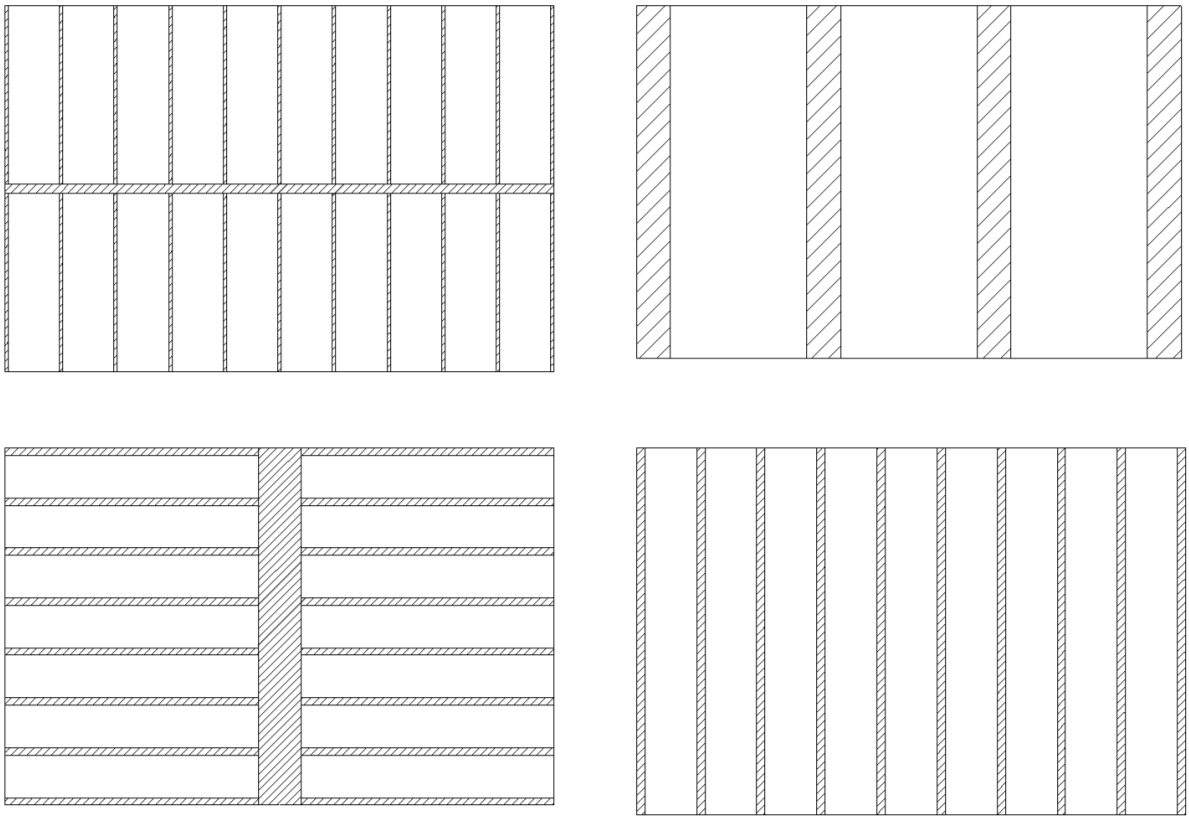


FIGURE 1.2 – Exemples de plans de configurations d'un gîtage

Premier objectif : optimisation théorique du volume de bois

Dans ce mémoire, nous cherchons dans un premier temps à optimiser ce dimensionnement en termes de volume de bois utilisé. A cette fin, nous allons d'abord réduire le nombre de paramètres qui entrent en jeu et chercher à reformuler, afin de les rendre adimensionnels, les critères de dimensionnement de structures, et plus particulièrement de structures en bois, donnés par les normes européennes. Cette reformulation adimensionnelle va nous permettre de déduire des lois générales valables pour tous les problèmes de planchers rectangulaires, et ce quels que soient les paramètres initiaux ou les dimensions du plancher.

Deuxième objectif : optimisation grâce à une application mobile

Dans un second temps, nous concevons, sur base des concepts et formules qui auront été développés, une application mobile destinée aux architectes et ingénieurs qui permet d'obtenir directement le dimensionnement optimal d'un gîtage en fonction de la surface à couvrir, cette fois en incluant également des paramètres économiques. En effet, pour obtenir un optimum économique pour le dimensionnement d'un gîtage, il faut prendre en compte un certain nombre de coûts qui ne sont pas proportionnels au volume de bois (cfr. figure 1.3). Parmi ces coûts, on tient compte du prix des sections courantes, d'un coût fixe (par élément) incluant la préparation et la mise en œuvre sur chantier, ou encore du prix des fixations des éléments. Nous ne prenons par contre pas en compte le prix des panneaux de plancher [1].

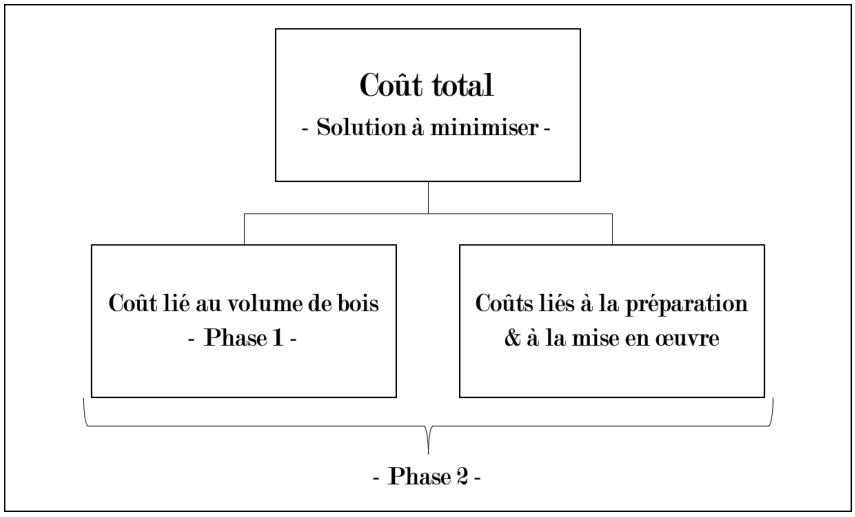


FIGURE 1.3 – Phases de recherche

Les analyses et résultats que nous avons établis dans ce mémoire sont valables pour tout type de gîtage rectangulaire en bois, à savoir aussi bien les gîtages destinés à reprendre des charges d'exploitation que ceux qui devront reprendre des charges climatiques (toitures plates).

1.2 Hypothèses et définitions générales

Afin de rester dans le cadre d'un travail de fin d'études, il est nécessaire de poser des hypothèses et des limites à notre recherche. Nous reprenons dans cette section les hypothèses sur lesquelles reposent nos calculs de dimensionnement.

1.2.1 Hypothèses de dimensionnement

Lors du dimensionnement d'un gîtage en bois, un nombre important de contraintes entrent en jeu, aussi bien d'un point de vue architectural (hauteur sous plafond minimale, présence de baies vitrées qui forcerait à placer le gîtage dans un certain sens, ...) que d'un point de vue structural (critères à l'ELU [2] et à l'ELS [3]).

1. Par exemple, des panneaux OSB (Oriented Strand Board)
2. ELU : Etat Limite Ultime
3. ELS : Etat Limite de Service

Dans ce mémoire, on retient les critères suivants pour le calcul du dimensionnement d'un gîtage :

- le critère ELU de résistance à la flexion simple ;
- le critère ELU de résistance au cisaillement ;
- le critère ELS de limite de flèche instantanée ;
- le critère ELS de limite de flèche finale.

Les critères non considérés dans ce mémoire pour le dimensionnement des gîtages sont :

- la résistance au feu ;
- la compression perpendiculaire aux appuis ;
- le glissement des assemblages ;
- le comportement aux vibrations ;
- la résistance aux organismes biologiques ;
- la résistance à la corrosion.

Le calcul des assemblages des éléments du gîtage entre eux ou avec la structure portant le gîtage ne fait pas partie de l'étude présentée dans ce mémoire.

Il est également à noter que l'on ne tient pas compte d'une méthode de calcul proposée par la norme européenne [4], méthode permettant de faire participer le panneau de plancher à la résistance du complexe gîtage-plancher. L'analyse présentée dans ce mémoire et l'application qui y est développée pourraient néanmoins être adaptées, dans le cadre d'une étude ultérieure, afin de prendre en compte cette possibilité.

Des critères architecturaux sont intégrés à partir du développement de l'application, mais ne font pas partie de la phase de recherche théorique sur l'optimisation du volume du bois.

1.2.2 Notations principales

Nous précisons ici les notions sur lesquelles se base notre étude.

Définition de la surface de plancher

Dans le cadre de ce mémoire, on ne considère que des planchers de forme rectangulaire. Ces rectangles seront de dimensions $P \times L$, où L désigne la portée des éléments (la longueur des poutres), et P la dimension du plancher transversale aux poutres. Ces grandeurs sont illustrées dans la figure 1.4.

Entraxe des éléments

L'entraxe est défini comme la distance séparant les axes de deux poutres adjacentes. On considère qu'il est constant pour tout le gîtage.

Théoriquement, dans le cadre d'une optimisation, on pourrait envisager un entraxe e allant jusqu'à P. Cependant, dans la pratique, les panneaux de plancher qui recouvrent le gîtage

4. Documentée dans l'Annexe B de l'Eurocode 5 : *Poutres assemblées mécaniquement*

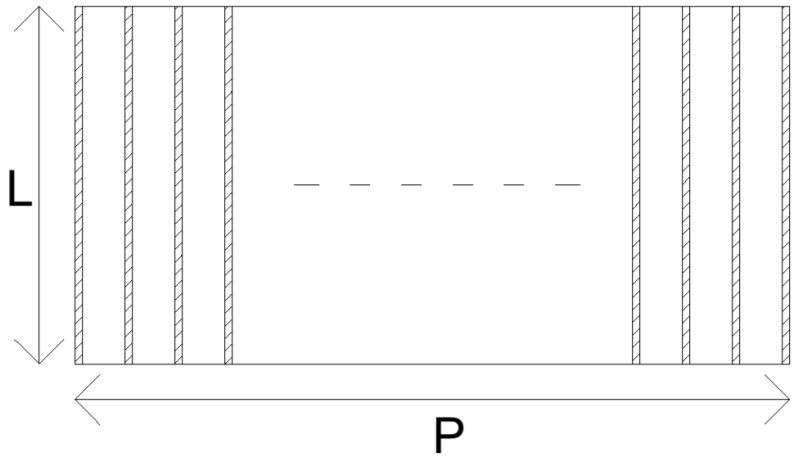


FIGURE 1.4 – Dimensions de la surface de plancher

pourraient ne pas supporter le poids du revêtement et des autres sollicitations si on choisit de trop grands entraxes. La réalité du terrain montre qu'un entraxe supérieur à 80[cm] conduit à des surcoûts de réalisation de l'ensemble du complexe gîtage-plancher. Par ailleurs, il est important de rappeler que le coût du plancher, qui peut dépendre de la grandeur de l'entraxe, n'est pas pris en compte dans la recherche de la solution la plus économique.

L'entraxe e est illustré à la figure 1.5.

Sens du gîtage

Afin de couvrir aussi bien les cas où $L > P$ et $L < P$, on définit un paramètre adimensionnel β qu'on appelle rapport de portée :

$$\beta = \frac{L}{P} \tag{1.1}$$

Ce paramètre permet de définir à lui seul la géométrie de la surface de plancher. La valeur de β dépend du choix de la dimension du plancher rectangulaire qui est associée à L ou à P. Cela nous conduit à définir deux nouvelles notions sur base de la valeur du paramètre β :

- $\beta < 1$: "portée selon le sens court"
- $\beta > 1$: "portée selon le sens long"

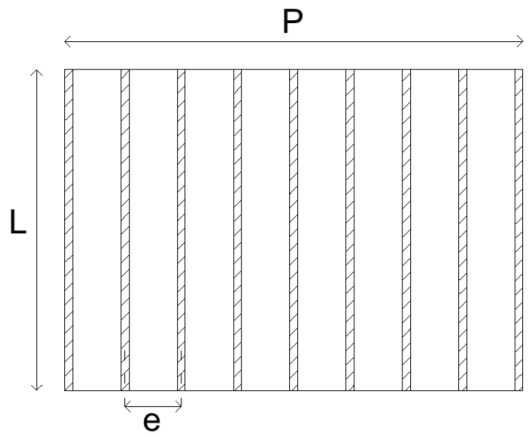
Les figures 1.5a et 1.5b illustrent ces deux cas.

1.2.3 Types de configurations considérées

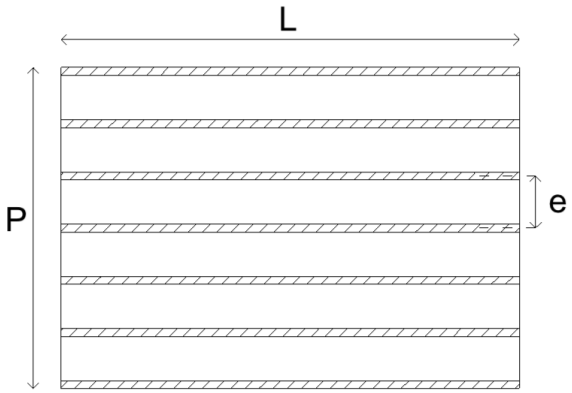
Il existe un grand nombre de possibilités de dispositions du gîtage. Dans le cadre de ce mémoire, on limitera la recherche de la solution optimale en considérant deux types de dispositions.

La première disposition envisagée, déjà représentée à la figure 1.5, est la méthode la plus simple qui consiste à placer des poutres de portée L espacées d'un entraxe e.

Ensuite, on considérera également la possibilité d'avoir recours à une poutre intermédiaire. Cette disposition, représentée à la figure 1.6, ne sera prise en compte, comme variante, qu'à partir de la conception de l'application.

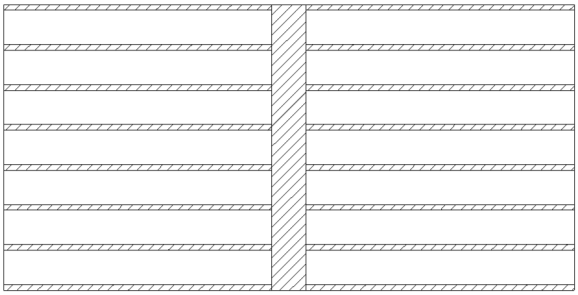


(a) Sens court : $\beta < 1$

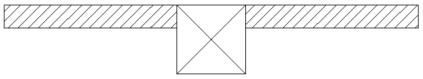


(b) Sens long : $\beta > 1$

FIGURE 1.5 – Deux cas décrits par le paramètre β



(a) Plan



(b) Coupe

FIGURE 1.6 – Recours à une poutre intermédiaire

L'objectif de la recherche étant fixé et les bases du problème étant posées, nous allons pouvoir, dans le prochain chapitre, nous étendre sur les normes européennes de calcul de structures en bois qui nous permettront de poursuivre notre étude.

Chapitre 2

Synthèse des éléments des Eurocodes utiles pour ce mémoire

Les Eurocodes sont une série de 10 documents reprenant les normes de conception, de dimensionnement et de justification des structures de bâtiments et d'ouvrages de génie civil au niveau européen. Dans le cadre de ce mémoire, nous utilisons essentiellement l'Eurocode 5 (norme EN-1995) [2], ainsi que l'Eurocode 0 (norme EN-1990) [3] et l'Eurocode 1 (norme EN-1991) [4]. Nous utilisons aussi les valeurs fournies dans les normes EN338 [5] et EN14080 [6].

2.1 Eléments de la norme EN-1990

> (1) L'EN1990 définit des Principes et des exigences en matière de sécurité, d'aptitude au service et de durabilité des structures, décrit les bases pour le dimensionnement et la vérification de celles-ci, et fournit des lignes directrices concernant les aspects de la fiabilité structurale qui s'y rattachent.
>
> (2) L'EN1990 est destinée à être utilisée conjointement avec les EN1991 à EN1999 pour la conception structurale des bâtiments et ouvrages de génie civil (...).
>
> Extrait de la norme EN1990.

Dans cette section, nous verrons les combinaisons de charges permanentes G_j et variables Q_i utilisées dans le cadre de ce mémoire. La formule générique des combinaisons de charges est donnée à l'équation 2.1.

$$\underbrace{\sum_j \gamma_{G,j} G_{k,j}}_{\text{actions permanentes}} + \underbrace{\gamma_{Q,1} Q_{k,1}}_{\text{action variable dominante}} + \underbrace{\sum_{i \geq 2} \gamma_{Q,i} \psi_{0,i} Q_{k,i}}_{\text{autres actions variables}} \tag{2.1}$$

Les coefficients γ, aussi appelés coefficients de sécurité, et les coefficients ψ, qui tiennent compte de la probabilité réduite d'avoir plusieurs charges variables agissant en même temps avec un effet maximum, sont spécifiés dans l'Annexe A1 de la norme EN1990. Leurs valeurs dépendront du critère de calcul. Ces critères sont définis par la norme EN1990 et sont résumés à la figure 2.1. Les valeurs prescrites pour les coefficients ψ sont donnés dans le tableau A.1.

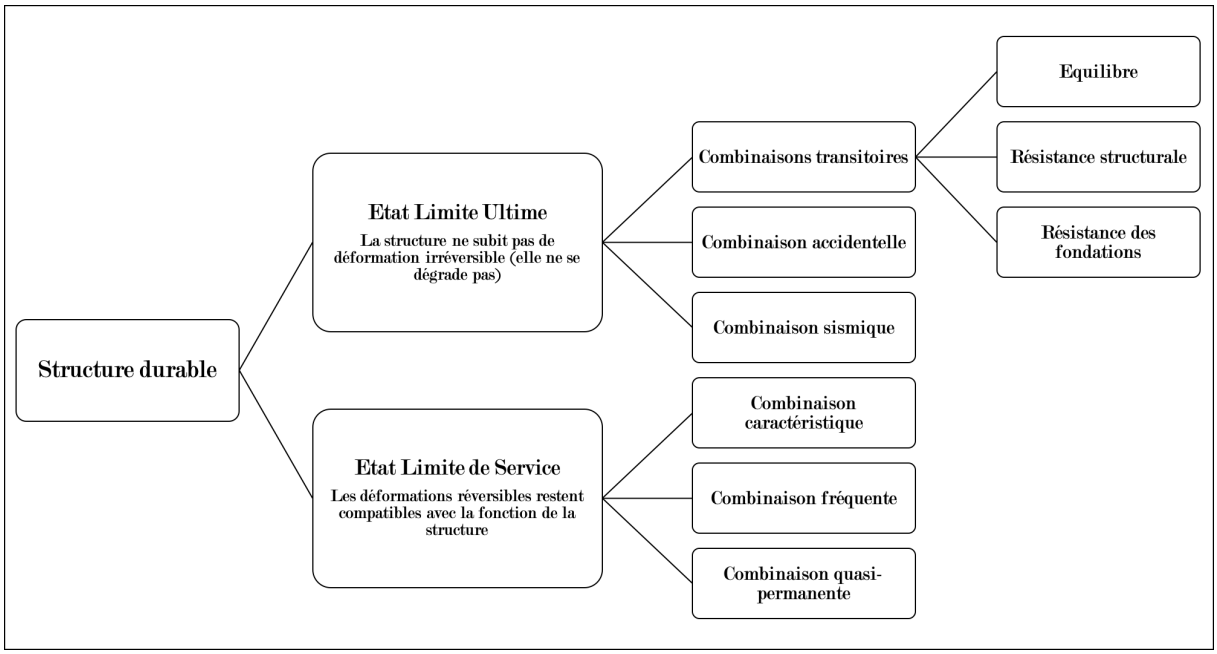


FIGURE 2.1 – Critères de dimensionnement définis par la norme EN1990

2.1.1 Vérification aux Etats Limites Ultimes (ELU)

En ce qui concerne la vérification aux ELU, nous retiendrons uniquement, dans le cadre de ce mémoire, la combinaison de résistance structurale (cfr. figure 2.1). Le but étant de déduire des tendances quant à l'optimisation du dimensionnement d'un gîtage, il n'est en effet pas nécessaire de considérer des combinaisons accidentelles ou sismiques. De plus, le critère de résistance structurale est plus contraignant que celui d'équilibre (coefficients γ plus élevés). Le critère de fondation n'est quant à lui simplement pas pertinent dans le cadre d'un gîtage.

La combinaison de charges utilisée pour la vérification à l'ELU de la résistance structurale s'écrit, selon la norme EN1990 :

$$1,35 \sum G_j + 1,50 Q_1 + \sum 1,50 \psi_{0,i} Q_i \tag{2.2}$$

Dans la suite, on notera q_{ELU} pour la charge linéique (en $[kN/m]$) uniformément répartie issue de cette combinaison de charge. Si on simplifie l'équation 2.2 afin de ne prendre en compte qu'une seule combinaison et en prenant une valeur sécuritaire de $\psi_0 = 1$, q_{ELU} est donné par :

$$\boxed{q_{ELU} = 1,35 \sum G_j + 1,50 \sum Q_i} \tag{2.3}$$

où G_j et Q_i sont respectivement les charges permanentes et variables en $[kN/m]$.

2.1.2 Vérification aux Etats Limites de Service (ELS)

En se référant à la figure 2.1, les combinaisons de charges pour les vérifications aux ELS sont :

1. Combinaison caractéristique (ou rare), qui s'écrit, selon la norme EN1990 :

$$\sum G_j + Q_1 + \sum \psi_{0,i} Q_i \tag{2.4}$$

2. Combinaison fréquente, qui s'écrit, selon la norme EN1990 :

$$\sum G_j + \psi_{1,1} Q_1 + \sum \psi_{2,i} Q_i \tag{2.5}$$

3. Combinaison quasi-permanente, qui s'écrit, selon la norme EN1990 :

$$\sum G_j + \psi_{2,1} Q_1 + \sum \psi_{2,i} Q_i \tag{2.6}$$

Dans ce mémoire, nous nous baserons sur la combinaison caractéristique, la plus contraignante, que nous allons toutefois adapter au bois et plus particulièrement à son comportement à long terme.

En début de vie d'un ouvrage en bois, l'équation 2.4 peut être appliquée telle quelle [7]. On notera $q_{ELS,inst}$ pour la charge linéique (en $[kN/m]$) uniformément répartie issue de cette combinaison de charge. Si on simplifie l'équation 2.4 afin de ne prendre en compte qu'une seule combinaison et en prenant une valeur sécuritaire de $\psi_0 = 1$, $q_{ELS,inst}$ est donné par :

$$\boxed{q_{ELS,inst} = \sum G_j + \sum Q_i} \tag{2.7}$$

En fin de vie d'un ouvrage en bois, il faut ajouter un terme à l'équation 2.4 afin de prendre en compte l'effet du fluage du bois [7]. La combinaison de charge de l'équation 2.4 devient :

$$\left[\sum G_j + Q_1 + \sum \psi_{0,i} Q_i\right] + k_{def} \left[sum G_j + \sum \psi_{2,i} Q_i\right] \tag{2.8}$$

où k_{def} est le facteur de déformation, qui est défini dans la norme EN1995-1-1 et dont les valeurs sont disponibles en annexe au tableau A.3. On notera $q_{ELS,fin}$ pour la charge linéique (en $[kN/m]$) uniformément répartie issue de la combinaison de charge de l'équation 2.8. Si on simplifie cette équation afin de ne prendre en compte qu'une seule combinaison et en prenant une valeur sécuritaire de $\psi_0 = 1$, $q_{ELS,fin}$ est donné par :

$$\boxed{q_{ELS,fin} = (1 + k_{def}) \sum G_j + \sum Q_i + k_{def} \sum \psi_{2,i} Q_i} \tag{2.9}$$

Dans les équations 2.7 et 2.9, G_j et Q_i sont respectivement les charges permanentes et variables en $[kN/m]$.

2.2 Eléments de la norme EN-1991

L'Eurocode 1 indique comment calculer les charges à prendre en compte lors du calcul d'une structure. La première partie de l'Eurocode 1, EN1991-1, traite des actions générales sur les structures. Cette première partie est elle-même subdivisée en plusieurs sous-parties qui définissent chacune les charges à prendre en compte pour les différents types d'actions (charges d'exploitation, neige, vent, actions thermiques, ...). Dans le cadre de ce mémoire, seule la partie EN1991-1-1 est concernée : Actions générales - Poids volumiques, poids propres, charges d'exploitation bâtiments.

Le tableau A.2 reprend les charges d'exploitation prescrites pour les bâtiments selon les différents usages.

2.3 Eléments de la norme EN-1995

L'Eurocode 5 est la norme européenne reprenant les principes de calcul pour les structures en bois.
Conformément à ce qui a été dit au Chapitre 1, nous nous limitons aux critères suivants pour l'optimisation du dimensionnement d'un gîtage en bois :

- le critère ELU de résistance à la flexion ;
- le critère ELU de résistance au cisaillement ;
- le critère ELS de limite de flèche instantanée ;
- le critère ELS de limite de flèche finale.

Dans cette section, nous détaillons les principes calculatoires de la norme EN-1995 qui correspondent à ces quatre critères.

2.3.1 Eléments nécessaires aux calculs

Classes de service

> Le système de classes de service a pour objectif principal d'affecter les valeurs de résistance et de calculer les déformations sous des conditions d'environnement définies.
>
> Les structures doivent être affectées à l'une des classes de service données ci-dessous :
>
> Classe de service 1 : est caractérisée par une humidité dans les matériaux correspondant à une température de $20°C$ et une humidité relative de l'air environnant ne dépassant 65% que quelques semaines par an.
>
> Classe de service 2 : est caractérisée par une humidité dans les matériaux correspondant à une température de $20°C$ et une humidité relative de l'air environnant ne dépassant 85% que quelques semaines par an.
>
> Classe de service 3 : est caractérisée par des conditions climatiques amenant à une humidité supérieure à celle de la classe de service 2.
>
> Extrait de la norme EN1995.

Classes de durée de chargement

> Les classes de durée de chargement sont caractérisées par l'effet d'une charge constante agissant pendant une certaine période de temps au cours de la vie de la structure. Pour une action variable, la classe appropriée doit être déterminée sur la base d'une estimation de la variation typique de la charge avec le temps.
>
> Extrait de la norme EN1995.

Valeur de calcul d'une propriété matérielle

La valeur de calcul X_d d'une propriété de résistance X_k doit être calculée selon [1] :

$$X_d = k_{mod}\frac{X_k}{\gamma_M} \tag{2.10}$$

Les valeurs de k_{mod} et γ_M sont données respectivement dans les tableaux A.4 et A.6.

2.3.2 Le calcul à l'Etat Limite Ultime (ELU)

Dans le cadre de l'étude présentée dans ce mémoire, nous nous limitons au cas de la flexion simple pour les calculs à l'ELU. Il y aura alors 2 critères de résistance à vérifier : la résistance à la flexion et la résistance au cisaillement.

Pour la vérification de ces deux critères, nous tiendrons compte de l'effet système. L'effet système intervient lorsque plusieurs éléments espacés uniformément sont connectés latéralement par un système qui peut redistribuer des charges continues en transférant les efforts d'un élément aux éléments voisins [2]. Comme cela est typiquement le cas de gîtages, on peut, d'après l'Eurocode 5, tenir compte de cet effet en multipliant les propriétés de résistance des éléments par un facteur $k_{sys} = 1,1$.

Le critère de résistance à la flexion

Il faut vérifier que la valeur de calcul de la contrainte de flexion n'excède pas la valeur de calcul correspondante pour la résistance en flexion :

$$\sigma_{m,Ed} \leqslant f_{m,d} \tag{2.11}$$

En considérant que $q_{ELU}[kN/m]$ est la charge linéique uniformément distribuée provenant de la pire combinaison ELU, que M_{Ed} est la valeur de calcul du moment de flexion et W_{el} le module de flexion élastique, on a, pour une section rectangulaire :

$$\sigma_{m,Ed} = \frac{M_{Ed}}{W_{el}} = \frac{q_{ELU}L^2/8}{bh^2/6} \tag{2.12}$$

La valeur de calcul de la résistance en flexion peut être établie en tenant compte de l'effet de la dimension des éléments, traduit par un facteur de hauteur k_h. On multiplie alors les propriétés de résistance des éléments par ce facteur k_h.

Pour un élément en bois massif de section rectangulaire de hauteur $h \leq 150[mm]$ dont la masse volumique caractéristique est inférieure à 700 $[kg/m^3]$, on a [3] :

$$k_h = min\begin{cases}\left(\frac{150}{h}\right)^{0,2}\\ 1,3\end{cases} \tag{2.13}$$

Pour un élément en bois lamellé-collé (BLC) de section rectangulaire de hauteur $h \leq 600[mm]$, on a [4] :

$$k_h = min\begin{cases}\left(\frac{600}{h}\right)^{0,1}\\ 1,1\end{cases} \tag{2.14}$$

1. EN1995-1-1 §2.4.1 (2.14)
2. EN1995-1-1 §6.6
3. EN1995-1-1 §3.2 (3.1)
4. EN1995-1-1 §3.3 (3.2)

La variation du facteur k_h peut être observée à la figure 2.2.

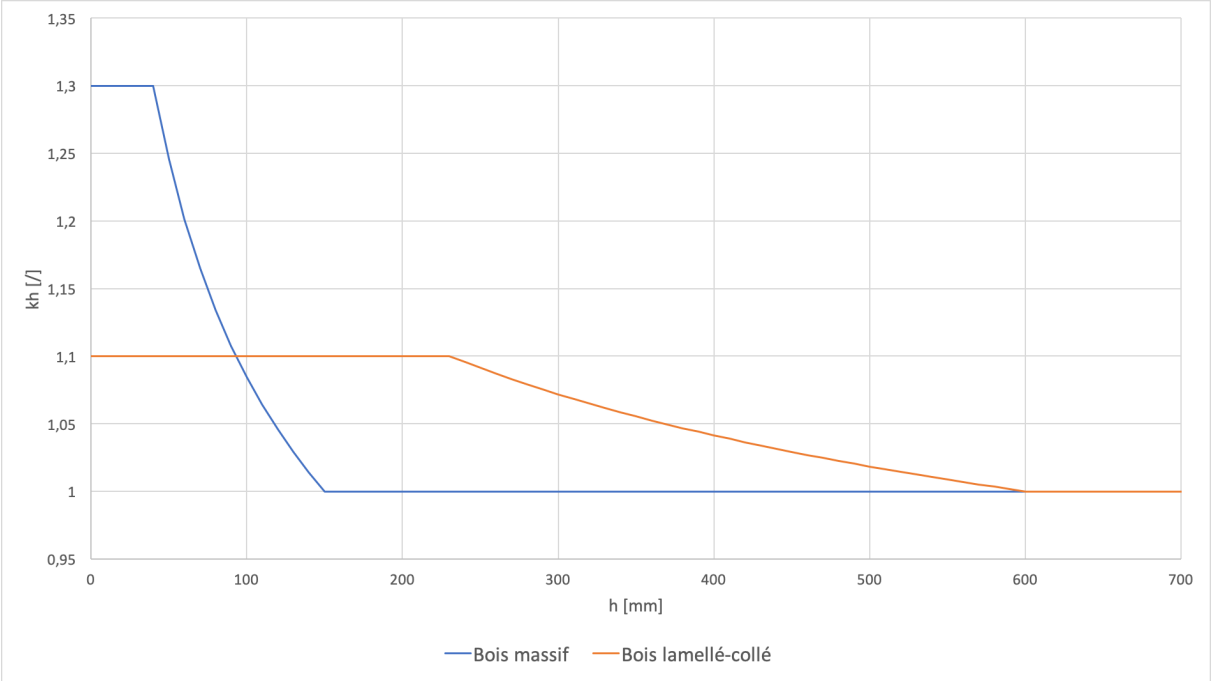


FIGURE 2.2 – Variation de k_h avec la hauteur de l'élément, pour du bois massif et du bois lamellé-collé

Conformément à l'équation 2.10 et en tenant compte de l'effet système et du facteur de hauteur, la valeur de calcul de la résistance en flexion vaut :

$$f_{m,d} = k_{mod} k_h k_{sys} \frac{f_{m,k}}{\gamma_M} \tag{2.15}$$

Les valeurs de $f_{m,k}$ sont données dans le Tableau A.8 pour les bois massifs et dans le Tableau A.9 pour les bois lamellés collés.

Finalement, le critère ELU de résistance à la flexion devient, en combinant les équations 2.11, 2.12 et 2.15 :

$$\boxed{\frac{M_{Ed}}{W_{el}} = \frac{q_{ELU} L^2/8}{bh^2/6} \leqslant k_{mod} k_h k_{sys} \frac{f_{m,k}}{\gamma_M}} \tag{2.16}$$

Le critère de résistance au cisaillement

Il faut vérifier que la valeur de calcul de la contrainte de cisaillement due à l'effort tranchant n'excède pas la valeur de calcul de la résistance au cisaillement [5] :

$$\tau_{Ed} \leqslant f_{v,d} \tag{2.17}$$

En considérant que $q_{ELU}[kN/m]$ est la charge linéique uniformément distribuée provenant de la pire combinaison ELU et que V_{Ed} est la valeur de calcul de l'effort tranchant, la sollicitation

5. EN1995-1-1 §6.1.7 (6.13)

due à cet effort tranchant est donnée, pour une section rectangulaire, par :

$$\tau_{Ed} = \frac{3V_{Ed}}{2A} = \frac{3q_{ELU}L/2}{2k_{cr}bh} \tag{2.18}$$

La valeur du facteur k_{cr} est prescrite dans l'Eurocode 5 et vaut $0,67$ pour le bois massif et le bois lamellé-collé. Ce facteur tient compte de l'influence des fissures et définit la largeur efficace de l'élément [6] : $b_{ef} = k_{cr}b$.

Conformément à l'équation 2.10 et en tenant compte de l'effet système, la valeur de calcul de la résistance au cisaillement vaut :

$$f_{v,d} = k_{mod}k_{sys}\frac{f_{v,k}}{\gamma_M} \tag{2.19}$$

Les valeurs de $f_{v,k}$ sont données dans le Tableau A.8 pour les bois massifs et dans le Tableau A.9 pour les bois lamellés-collés.

Finalement, le critère ELU de résistance au cisaillement devient, en combinant les équations 2.17, 2.18 et 2.19 :

$$\boxed{\frac{3V_{Ed}}{2A} = \frac{3q_{ELU}L/2}{2k_{cr}bh} \leqslant k_{mod}k_{sys}\frac{f_{v,k}}{\gamma_M}} \tag{2.20}$$

2.3.3 Le calcul à l'Etat Limite de Service (ELS)

Pour les calculs aux ELS, nous nous limitons dans le cadre de l'étude présentée ici aux valeurs limites pour les flèches des poutres [7]. Il y aura alors deux critères de flèche à vérifier : celui de la flèche instantanée ou élastique, et celui de la flèche finale.

> Les composantes de la flèche qui résulte d'une combinaison d'actions sont illustrées dans la figure 2.3, où les symboles sont définis comme suit :
>
> - w_c est la contreflèche (si elle existe)
> - w_{inst} est la flèche instantanée
> - w_{creep} est la flèche de fluage
> - w_{fin} est la flèche finale
> - $w_{net,fin}$ est la flèche résultante finale
>
> Extrait de la norme EN1995.

Le critère de la flèche instantanée

La flèche instantanée est élastique, et donc elle s'annule lorsque les charges sont retirées. La valeur maximale que cette flèche pourra avoir est donnée par :

$$w_{inst} \leqslant \frac{L}{\eta_{inst}} \tag{2.21}$$

6. Amendement A1 de la norme EN1995-1-1

7. La norme EN1995-1-1 prescrit également des règles tenant compte des glissements des assemblages (§7.1) et des vibrations (§7.3).

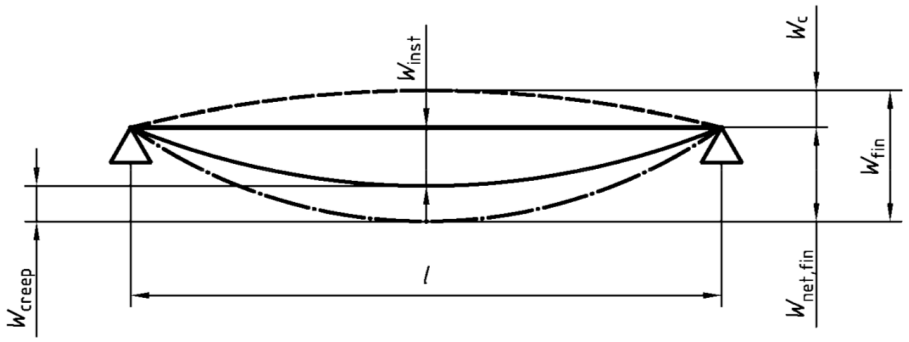


FIGURE 2.3 – Composantes de la flèche - Figure issue de la norme EN1995 §7.2

Pour une poutre sur deux appuis, les valeurs de η_{inst} données dans le tableau 7.2 de l'Eurocode 5 peuvent aller de 300 à 500. Ce tableau est repris en annexe dans le tableau A.7.

Pour une poutre en flexion simple, en considérant que $q_{ELS,inst}[kN/m]$ est la charge linéique uniformément distribuée provenant de la pire combinaison ELS de l'équation 2.7, la flèche instantanée est donnée par :

$$w_{inst} = \frac{5q_{ELS,inst}L^4}{384EI_y} + \frac{q_{ELS,inst}L^2}{8GA_v} \tag{2.22}$$

Le premier terme du membre de droite de l'équation 2.22 exprime la flèche due au moment de flexion, tandis que le second terme exprime la flèche due à l'effort tranchant.

Finalement, le critère ELS de la flèche instantanée devient, en combinant les équations 2.21 et 2.22 :

$$\boxed{\frac{5q_{ELS,inst}L^4}{384EI_y} + \frac{q_{ELS,inst}L^2}{8GA_v} \leqslant \frac{L}{\eta_{inst}}} \tag{2.23}$$

où E est le module d'élasticité du bois, G son module de cisaillement, I_y le module d'inertie de la poutre selon l'axe fort et A_v l'aire réduite de la section de la poutre.

Le critère de la flèche finale

La flèche finale est la somme de la flèche instantanée et de la flèche due au fluage. La flèche de fluage est une flèche résiduelle qui subsiste même lorsque toutes les charges sont retirées. La valeur maximale que cette flèche pourra avoir est donnée par :

$$w_{fin} \leqslant \frac{L}{\eta_{fin}} \tag{2.24}$$

Pour une poutre sur deux appuis, les valeurs de η_{fin} données dans le tableau 7.2 de l'Eurocode 5 peuvent aller de 150 à 300. Ce tableau est repris en annexe, tableau A.7.

Pour une poutre en flexion simple, en considérant que $q_{ELS,fin}[kN/m]$ est la charge linéique uniformément distribuée provenant de la pire combinaison ELS de l'équation 2.9, la flèche finale est donnée par :

$$w_{fin} = \frac{5q_{ELS,fin}L^4}{384EI_y} + \frac{q_{ELS,fin}L^2}{8GA_v} \tag{2.25}$$

On remarque que le membre de droite de l'équation 2.25 correspond au membre de droite de l'équation 2.22 que l'on aurait divisé par $q_{ELS,inst}$ puis multiplié par $q_{ELS,fin}$. Ceci nous permet d'exprimer la flèche finale w_{fin} en fonction de la flèche instantanée w_{inst} :

$$w_{fin} = \frac{q_{ELS,fin}}{q_{ELS,inst}} w_{inst} \tag{2.26}$$

Finalement, le critère ELS de la flèche finale devient, en combinant les équations 2.24, 2.26 et 2.22 :

$$\boxed{\frac{q_{ELS,fin}}{q_{ELS,inst}} \left(\frac{5 q_{ELS,inst} L^4}{384 E I_y} + \frac{q_{ELS,inst} L^2}{8 G A_v} \right) \leqslant \frac{L}{\eta_{fin}}} \tag{2.27}$$

Ainsi se trouvent résumés les éléments des Eurocodes nécessaires à notre recherche dans le cadre de ce travail de fin d'études.

Chapitre 3

Reformulation des critères de dimensionnement

Dans ce chapitre, nous établissons une reformulation adimensionnelle des critères de dimensionnement vus au chapitre précédent. Cette reformulation permet de réduire considérablement le nombre de paramètres qui entrent en jeu et de rassembler l'ensemble des critères en un unique critère. L'avantage d'une telle reformulation, outre le fait d'alléger les calculs, est de pouvoir facilement comparer différents problèmes de planchers entre eux et de permettre une déduction de lois générales valables pour tous les problèmes de planchers rectangulaires, quels que soient les paramètres initiaux ou les dimensions du plancher.

A cette fin, nous passons d'abord en revue une première reformulation adimensionnelle des quatre critères retenus [1], faite par T. Descamps, M. Monhonval et P. Latteur dans l'article "An assessment methodology for timber beams in a restoration context, based on a dimensionless formulation of EC5 design criteria" (2018) [1].
Dans un second temps, nous introduisons de nouveaux paramètres et nous effectuons une étude des plages de variation de ceux-ci.
Cela nous conduit ensuite à l'établissement de nouvelles expressions de ces quatre critères, qui font intervenir des paramètres directement liés à la géométrie du plancher.
Enfin, nous montrons en quoi ces critères peuvent être utilisés dans la recherche de la solution optimale en termes de volume de bois.

Les symboles utilisés dans ce chapitre sont repris dans la liste des symboles, page ix.

3.1 Résumé de l'article "An assessment methodology for timber beams in a restoration context, based on a dimensionless formulation of EC5 design criteria"

Cet article propose une méthodologie simple et efficace qui permet d'alléger considérablement les calculs relatifs au dimensionnement des poutres en bois de section rectangulaire chargées verticalement. Il est inspiré de la théorie d'indicateurs adimensionnels morphologiques (Latteur 2000, Latteur et al. 2017, Samyn 2000). La méthode proposée permet de ne prendre en compte qu'un nombre réduit de paramètres, essentiellement géométriques, tandis que les autres paramètres sont inclus dans un facteur adimensionnel Z_d, ce qui simplifie la démarche de calcul.

1. Deux critères ELU : flexion simple et cisaillement ; deux critères ELS : flèche instantanée et flèche finale

3.1.1 Facteurs adimensionnels développés dans le cadre de l'article

Facteur Z_d

Le facteur adimensionnel Z_d fait notamment intervenir la largeur de l'élément en bois $b[m]$, sa hauteur $h[m]$ et la charge linéique $q_{ELU}[N/m]$.

$$Z_d \triangleq \frac{b \cdot k_h k_{sys} k_{mod} \cdot f_{m,k}}{0,75 \cdot \gamma_M \cdot q_{ELU}} \tag{3.1}$$

Facteur k_{M-V}

Le facteur adimensionnel k_{M-V}, appelé indicateur de transition M-V, fait notamment intervenir le rapport entre la valeur caractéristique de la résistance en flexion $f_{m,k}$ et la valeur caractéristique de la résistance au cisaillement $f_{v,k}$.

$$k_{M-V} \triangleq \frac{k_h f_{m,k}}{k_{cr} f_{v,k}} \tag{3.2}$$

Facteurs k_{inst} et k_{fin}

Les facteurs adimensionnels k_{inst} et k_{fin} sont définis par les expressions 3.3 et 3.4 respectivement.

$$k_{inst} \triangleq \frac{q_{ELS,inst}}{bE} Z_d \tag{3.3}$$

$$k_{fin} \triangleq \frac{q_{ELS,fin}}{q_{ELS,inst}} \tag{3.4}$$

3.1.2 Reformulation adimensionnelle des critères de dimensionnement ELU et ELS

Critère ELU de flexion

En reprenant la relation 2.16, le critère de dimensionnement pour une poutre rectangulaire de portée L et de section $b \times h$ est :

$$\frac{q_{ELU} L^2/8}{bh^2/6} \leq \frac{k_h k_{sys} k_{mod} \cdot f_{m,k}}{\gamma_M} \tag{3.5}$$

En réarrangeant l'expression 3.5, on obtient successivement :

$$\frac{L^2}{h^2} \leq \frac{8 \cdot b \cdot k_h k_{sys} k_{mod} \cdot f_{m,k}}{6 \cdot \gamma_M \cdot q_{ELU}} \tag{3.6}$$

$$\left(\frac{L}{h}\right)^2 \leq \frac{b \cdot k_h k_{sys} k_{mod} \cdot f_{m,k}}{0,75 \cdot \gamma_M \cdot q_{ELU}} \tag{3.7}$$

On retrouve le paramètre Z_d, dont l'expression est donnée par l'équation 3.1, dans le membre de droite de l'expression 3.7. Finalement, on trouve donc :

$$\boxed{\left(\frac{L}{h}\right)^2 \leq Z_d} \tag{3.8}$$

Critère ELU de cisaillement

En reprenant la relation 2.20, le critère de dimensionnement pour une poutre rectangulaire de portée L et de section $b \times h$ est :

$$\frac{3q_{ELU}L/2}{2k_{cr}bh} \leq \frac{k_{sys}k_{mod} \cdot f_{v,k}}{\gamma_M} \tag{3.9}$$

En réarrangeant l'expression 3.9, on obtient :

$$\frac{L}{h} \leq k_{cr}f_{v,k}\frac{b \cdot k_{sys}k_{mod}}{0,75 \cdot \gamma_M \cdot q_{ELU}} \tag{3.10}$$

En introduisant le paramètre k_{M-V}, dont l'expression est donnée à l'équation 3.2, l'expression 3.10 devient :

$$k_{M-V}\frac{L}{h} \leq \frac{b \cdot k_h k_{sys}k_{mod} \cdot f_{m,k}}{0,75 \cdot \gamma_M \cdot q_{ELU}} \tag{3.11}$$

On retrouve immédiatement l'expression du paramètre Z_d dans le membre de droite de l'équation 3.11. Le critère ELU de cisaillement devient donc :

$$\boxed{k_{M-V}\left(\frac{L}{h}\right) \leq Z_d} \tag{3.12}$$

Comparaison des deux critères ELU

Le seuil qui définira le critère déterminant parmi les deux critères ELU peut être trouvé en égalant les équations 3.8 et 3.12. On obtient alors :

$$\left(\frac{L}{h}\right)^2 = k_{M-V}\left(\frac{L}{h}\right) \tag{3.13}$$

En divisant les deux membres de l'égalité 3.13 par L/h, on trouve :

$$k_{M-V} = \frac{L}{h} \tag{3.14}$$

Autrement dit, le critère ELU de flexion sera déterminant lorsque $L/h \geq k_{M-V}$ et, inversement, le critère ELU de cisaillement sera déterminant lorsque $L/h \leq k_{M-V}$. La figure 3.1 illustre ce phénomène : la valeur minimale de Z_d pour satisfaire le critère de cisaillement est plus grande que celle pour satisfaire le critère de flexion lorsque $L/h \leq k_{M-V}$ et inversement.

Cela explique pourquoi le facteur k_{M-V} est appelé *indicateur de transition M-V*.

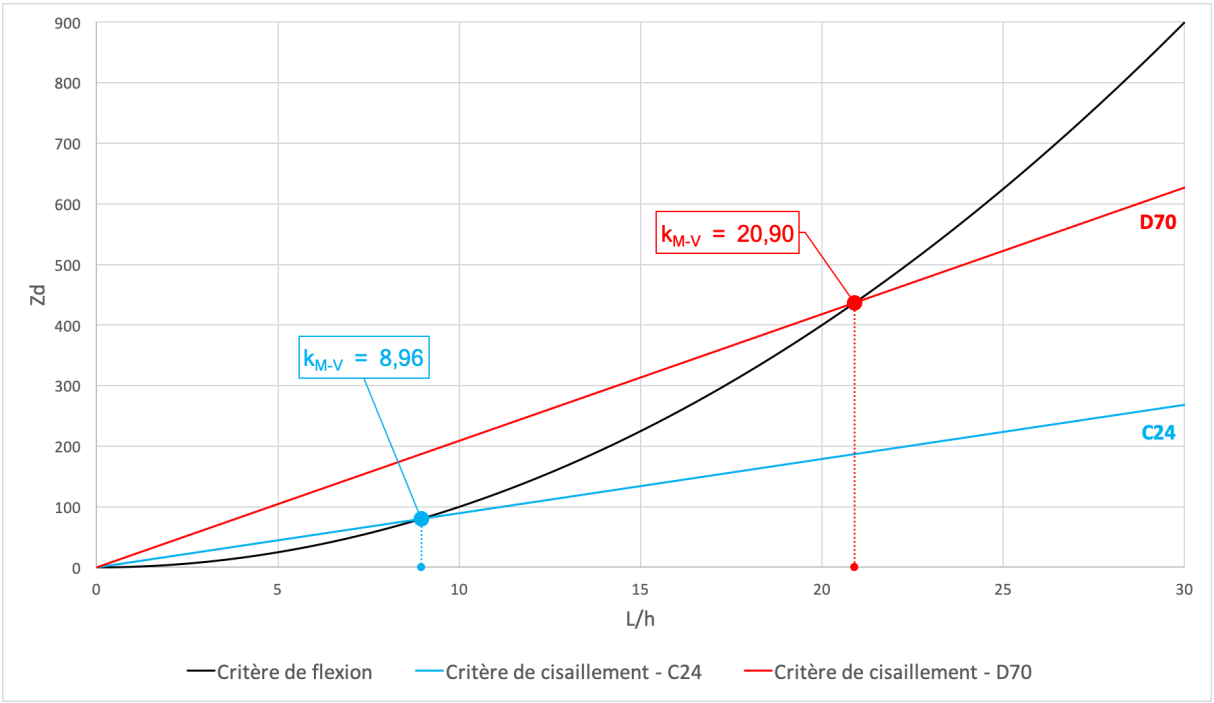


FIGURE 3.1 – Illustration du facteur k_{M-V} pour des bois de classes C24 et D70

Critère ELS de flèche instantanée

En reprenant la relation 2.23, le critère de dimensionnement tel que défini dans l'Eurocode 5 pour une poutre rectangulaire de portée L et de section $b \times h$ est :

$$w_{inst} = \frac{5 q_{ELS,inst} L^4}{384 E I_y} + \frac{q_{ELS,inst} L^2}{8 G A_v} \leq \frac{L}{\eta_{inst}} \tag{3.15}$$

En réarrangeant l'expression 3.15 et étant donné que

- $I_y = bh^3/12$
- $A_v = 5bh/6$

on obtient :

$$\frac{5}{32}\left(\frac{L}{h}\right)^3 + \frac{3}{20}\frac{E}{G}\left(\frac{L}{h}\right) \leq \frac{bE}{\eta_{inst} q_{ELS,inst}} \tag{3.16}$$

On retrouve l'expression du paramètre k_{inst} (équation 3.3) dans le membre de droite de l'équation 3.16. Le critère ELS de flèche instantanée devient dès lors :

$$\boxed{(\eta_{inst} k_{inst}) \left[\frac{5}{32}\left(\frac{L}{h}\right)^3 + \frac{3}{20}\frac{E}{G}\left(\frac{L}{h}\right)\right] \leq Z_d} \tag{3.17}$$

Critère ELS de flèche finale

Le critère de dimensionnement tel que défini dans l'Eurocode 5 pour une poutre rectangulaire de portée L et de section $b \times h$ est :

$$w_{fin} = \frac{5 q_{ELS,fin} L^4}{384 E I_y} + \frac{q_{ELS,fin} L^2}{8 G A_v} = k_{fin} w_{inst} \leq \frac{L}{\eta_{fin}} \tag{3.18}$$

On peut facilement adapter le résultat obtenu pour le critère de flèche instantanée (équation 3.17) afin de trouver le même type d'expression pour le critère ELS de flèche finale :

$$\boxed{k_{fin}\left(\eta_{fin}k_{inst}\right)\left[\frac{5}{32}\left(\frac{L}{h}\right)^3+\frac{3}{20}\frac{E}{G}\left(\frac{L}{h}\right)\right]\leq Z_d} \tag{3.19}$$

3.1.3 Critère de dimensionnement global

En comparant les quatre critères reformulés ci-dessus (équations 3.8, 3.12, 3.17 et 3.19), on peut les résumer en un unique critère de dimensionnement global :

$$\boxed{max\left[\left(\frac{L}{h}\right)^2, k_{M-V}\left(\frac{L}{h}\right), max\left(\eta_{inst}, k_{fin}\eta_{fin}\right)k_{inst}\left(\frac{5}{32}\left(\frac{L}{h}\right)^3+\frac{3}{20}\frac{E}{G}\left(\frac{L}{h}\right)\right)\right]\leq Z_d} \tag{3.20}$$

Ainsi se trouve résumée la reformulation adimensionnelle des critères de dimensionnement de l'Eurocode 5 telle qu'établie dans l'article de Descamps, Monhonval et Latteur (2018).

3.2 Définition de nouveaux paramètres

Afin d'élaborer de nouvelles expressions des critères de dimensionnement, plus adaptées à l'optimisation du volume de bois utilisé dans un gîtage, nous proposons de définir de nouveaux paramètres adimensionnels. Nous verrons plus loin dans ce chapitre comment les introduire dans les calculs.

3.2.1 Paramètre Y_d

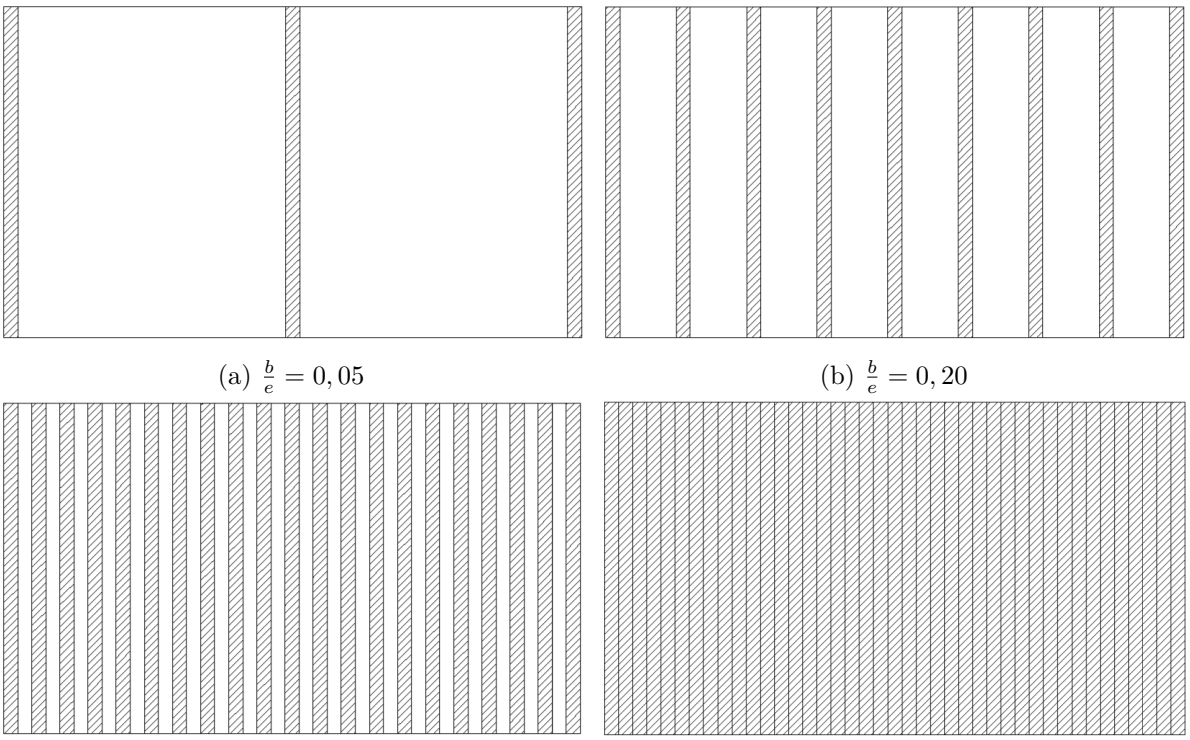
Nous allons dans un premier temps réécrire l'équation 3.1 afin d'y faire apparaître des paramètres spécifiques à un problème de gîtage. Pour ce faire, on remplace la charge linéique $q_{ELU}[N/m]$ par une charge surfacique $Q_{ELU}[N/m^2]$, donnée incontournable d'un problème de gîtage, multipliée par l'entraxe $e[m]$ espaçant les éléments du gîtage.

$$Z_d = \frac{k_h k_{sys} k_{mod} f_{m,k}}{0,75\gamma_M Q_{ELU}}\frac{b}{e} \tag{3.21}$$

On note l'apparition d'un paramètre adimensionnel b/e que l'on appellera *taux de remplissage* du plancher ($b/e < 1$). Quelques exemples de planchers ayant différents taux de remplissage sont illustrés à la figure 3.2.

On remarque que le premier terme du membre de droite de l'équation 3.21 est entièrement déterminé par les données du problème, alors que le deuxième terme du membre de droite, le taux de remplissage, est quant à lui une inconnue. Afin de pouvoir séparer ces deux termes dans la suite de l'étude, on définit un paramètre adimensionnel Y_d valant l'inverse du premier terme précité :

$$Y_d = \frac{0,75\gamma_M Q_{ELU}}{k_h k_{sys} k_{mod} f_{m,k}} = Z_d^{-1}\frac{b}{e} \tag{3.22}$$



(a) $\frac{b}{e} = 0,05$ (b) $\frac{b}{e} = 0,20$

(c) $\frac{b}{e} = 0,50$ (d) $\frac{b}{e} = 1,00$

FIGURE 3.2 – Exemples de planchers ayant différents taux de remplissage $\frac{b}{e}$

3.2.2 Paramètres B et θ

Le facteur adimensionnel B permettra plus tard de simplifier les calculs de la reformulation des critères de dimensionnement ELS. On le définit comme suit :

$$B \triangleq \theta Z_d = \theta Y_d^{-1} \frac{b}{e} \tag{3.23}$$

avec

$$\theta \triangleq \frac{32}{5 \cdot max\left(\eta_{inst}, k_{fin}\eta_{fin}\right) \cdot k_{inst}} \tag{3.24}$$

3.2.3 Paramètre C

De même, le facteur adimensionnel C permettra de simplifier les calculs lors de la reformulation des critères de dimensionnement ELS. On le définit comme suit :

$$C \triangleq \frac{24}{25} \frac{E}{G} \tag{3.25}$$

3.3 Gammes de variations des différents paramètres

Dans cette section, nous cherchons les bornes inférieures et supérieures des différents paramètres développés ci-dessus. A cette fin, nous allons poser certaines hypothèses.

3.3.1 Hypothèses et approximations de base

On considère les valeurs suivantes pour les facteurs k_h, k_{cr} et k_{sys} :

- $k_h = 1$
- $k_{cr} = 0,67$
- $k_{sys} = 1,1$

On fait l'hypothèse qu'un gîtage se trouvera toujours dans les classes de service 1 ou 2.

On suppose également qu'on travaille avec du bois massif, et donc, en se référant aux tableaux A.6 et A.4, on a :

- $\gamma_M = 1,30$
- $0,6 < k_{mod} < 0,9$ (on ne tient pas compte d'actions instantanées)

On se réfère aux valeurs caractéristiques des classes de résistance de bois massif données dans le tableau A.8.

En se basant sur le tableau A.2, on considère des charges d'exploitation allant de $2,0[kN/m^2]$ à $5,0[kN/m^2]$ (en omettant les espaces de stockage et les toitures). On suppose que les charges permanentes sur le gîtage (panneaux OSB, revêtement, plafond suspendu, ...) varient entre $0,5[kN/m^2]$ et $1,0[kN/m^2]$. En utilisant la combinaison 2.3, on obtient une charge Q_{ELU} comprise entre $3,67[kN/m^2]$ et $8,85[kN/m^2]$.

Enfin, on rappelle que le taux de remplissage du plancher $\frac{b}{e}$ est toujours compris entre 0 et 1.

3.3.2 Paramètre Y_d

Compte tenu des hypothèses et approximations faites ci-dessus, le paramètre Y_d sera borné par les valeurs suivantes :

- Pour les résineux : $7,2 \cdot 10^{-5} < Y_d < 93,4 \cdot 10^{-5}$
- Pour les feuillus : $5,1 \cdot 10^{-5} < Y_d < 72,7 \cdot 10^{-5}$

3.3.3 Paramètre k_{inst}

En étendant Z_d dans l'expression 3.3, on obtient :

$$k_{inst} = \left(\frac{4}{3} k_h k_{sys} \frac{q_{ELS,inst}}{q_{ELU}}\right) k_{mod} \left(\frac{f_{m,k}}{\gamma_M E}\right) \tag{3.26}$$

On peut montrer que $q_{ELS,inst}/q_{ELU}$ est toujours compris entre $1,50^{-1}$ et $1,35^{-1}$ [1]. En effet, pour une structure très légère (charges permanentes négligeables), on a :

$$\lim_{\left(\sum_j G_j\right) \to 0} \left(\frac{\sum_j G_j + \left[Q_1 + \sum_{i \geqslant 2} \psi_{0,i} Q_i\right]}{1,35 \sum_j G_j + 1,5 \left[Q_1 + \sum_{i \geqslant 2} \psi_{0,i} Q_i\right]}\right) = \frac{1}{1,50} \tag{3.27}$$

De même, dans le cas d'une structure très lourde (charges variables négligeables), on a :

$$\lim_{\left(Q_1+\sum_{i\geqslant 2}\psi_{0,i}Q_i\right)\to 0}\left(\frac{\sum_j G_j+\left[Q_1+\sum_{i\geqslant 2}\psi_{0,i}Q_i\right]}{1,35\sum_j G_j+1,5\left[Q_1+\sum_{i\geqslant 2}\psi_{0,i}Q_i\right]}\right)=\frac{1}{1,35} \tag{3.28}$$

Le terme $f_{m,k}/\left(\gamma_M E\right)$ varie quant à lui entre $1,54\times 10^{-3}$ et $2,40\times 10^{-3}$ pour les résineux et entre $1,46\times 10^{-3}$ et $2,69\times 10^{-3}$ pour les feuillus. Finalement, on trouve les différentes bornes pour le facteur k_{inst} :

- Pour les résineux : $9,0\cdot 10^{-4}<k_{inst}<23,5\cdot 10^{-4}$
- Pour les feuillus : $8,5\cdot 10^{-4}<k_{inst}<26,4\cdot 10^{-4}$

Pour simplifier l'expression de k_{inst}, on peut raisonnablement faire l'approximation que $q_{ELS,inst}/q_{ELU}=1,421^{-1}$. En effet, cette hypothèse générera au maximum une erreur de $5,3\%$ dans tous les cas [1] : $(1,5-1,421)/1,5=0,053$ et $(1,421-1,35)/1,5=0,053$. L'expression de k_{inst} qu'on utilisera dans la suite de cette étude est dès lors :

$$k_{inst}=1,0321\left(\frac{f_{m,k}}{\gamma_M E}\right)k_{mod} \tag{3.29}$$

3.3.4 Paramètre k_{fin}

On peut montrer qu'on aura toujours $1\leq k_{fin}=q_{ELS,fin}/q_{ELS,inst}\leq 1+k_{def}$ [1]. En effet, pour une structure très légère (charges permanentes négligeables) soumise uniquement au vent ou à la neige ($\psi_2=0$), on a :

$$\lim_{\left(\sum_j G_j\right)\to 0}\left(\frac{(1+k_{def})\sum_j G_j+Q_1+\left(\sum_{i\geqslant 2}\psi_{0,i}Q_i+k_{def}\sum_{i\geqslant 1}\psi_{2,i}Q_i\right)}{\sum_j G_j+Q_1+\sum_{i\geqslant 2}\psi_{0,i}Q_i}\right)=1 \tag{3.30}$$

De même, dans le cas d'une structure très lourde (charges variables négligeables), on a :

$$\lim_{\left(\sum_j G_j\right)\to 0}\left(\frac{(1+k_{def})\sum_j G_j+Q_1+\left(\sum_{i\geqslant 2}\psi_{0,i}Q_i+k_{def}\sum_{i\geqslant 1}\psi_{2,i}Q_i\right)}{\sum_j G_j+Q_1+\sum_{i\geqslant 2}\psi_{0,i}Q_i}\right)=1+k_{def} \tag{3.31}$$

Le facteur k_{fin} prend les valeurs suivantes, en fonction de la classe de service dans laquelle on se trouve :

- Classe de service 1 : $1<k_{fin}<1,6$
- Classe de service 2 : $1<k_{fin}<1,8$

3.3.5 Paramètre k_{M-V}

Le paramètre k_{M-V}, défini par l'équation 3.2, dépend uniquement de la classe de résistance du bois (les facteurs k_h et k_{cr} étant fixés). On a donc :

- Pour les résineux : $7,0 < k_{M-V} < 18,7$
- Pour les feuillus : $7,7 < k_{M-V} < 20,9$

3.3.6 Paramètre θ

Rappelons premièrement que les valeurs de η_{inst} et η_{fin} pour une poutre sur deux appuis, données dans l'Eurocode 5, sont les suivantes :

- η_{inst} : 300 à 500
- η_{fin} : 150 à 300

Sachant que $1 < k_{fin} < 1,8$, on a donc :

$$300 \leq max\left(\eta_{inst}, k_{fin}\eta_{fin}\right) \leq 540$$

En combinant cela à l'étude de variation du paramètre k_{inst}, on trouve :

- Pour les résineux : $5,0 < \theta < 23,8$
- Pour les feuillus : $4,4 < \theta < 25,1$

3.3.7 Paramètre B

Sur base de l'analyse de variation des paramètres θ et Y_d, et puisque, théoriquement, $0 < b/e < 1$, on trouve :

- Pour les résineux : $0 < B < 17,2 \cdot 10^6$
- Pour les feuillus : $0 < B < 13,0 \cdot 10^6$

3.3.8 Paramètre C

Pour les bois massifs, comme $\frac{E}{G} \approx 16$, on aura $C \approx 15,36$ (en étant plus précis, on aura toujours $15,26 < C < 15,46$).

3.4 Adaptation des critères au cas d'un gîtage : élément d'optimisation du volume de bois utilisé

Pour l'instant, les critères de dimensionnement (résumés par le critère de dimensionnement global de l'équation 3.20) ont encore une forme trop générale pour pouvoir les appliquer à notre étude. En effet, le volume qu'aura la poutre qu'on cherche à dimensionner grâce à ces critères est un élément primordial de notre étude, puisqu'on cherche à le minimiser. Ce volume d'une poutre vaut $Vol = L \times S$, où $S = bh$ est l'aire de la section de la poutre. Or, pour

l'instant, les paramètres intervenant dans ce volume n'apparaissent pas tous dans l'équation 3.20. De plus, il n'est pas encore établi sur quel paramètre doit porter ce critère de dimensionnement : sur le rapport L/h minimal ? Sur la largeur b minimale ? Sur la section $S = bh$ minimale ?

Dans cette section, nous démontrons précisément sur quel paramètre doit porter le critère de dimensionnement global afin que cela mène au volume minimal de la poutre.

3.4.1 Calcul à l'ELU

Le critère ELU de résistance au moment de flexion pour une poutre sur deux appuis est donné par l'équation 2.16. En réarrangeant les termes, cela revient à résoudre le problème suivant :

$$W_{el} \geqslant \frac{\gamma_M q_{ELU} L^2}{8 k_{mod} k_h k_{sys} f_{m,k}} \tag{3.32}$$

Le problème 3.32 impose donc de trouver un module de flexion élastique W_{el} minimal auquel les éléments du gîtage doivent satisfaire afin de répondre au critère de résistance 2.16. Or, il existe plusieurs sections S (combinaison d'une largeur b et d'une hauteur h) qui peuvent satisfaire un même module de flexion élastique W_{el}. Afin d'optimiser le volume de bois utilisé pour chaque poutre, il faudrait donc choisir la plus petite section $S = b \times h$ possible qui satisfait au module de flexion élastique W_{el} minumum requis.

Le critère ELU de résistance au cisaillement est donné par l'équation 2.20. En réarrangeant les termes, cela revient à résoudre le problème suivant :

$$A \geqslant \frac{3 \gamma_M q_{ELU} L}{4 k_{mod} k_{sys} f_{v,k}} \tag{3.33}$$

Le problème 3.33 impose donc de trouver une aire $A = k_{cr} bh$ minimale, et donc une section $S = b \times h$ minimale, à laquelle les éléments du gîtage doivent satisfaire afin de répondre au critère de résistance 2.20. Cela donnera automatiquement un volume de bois minimal ; dans cette section on retiendra donc surtout le critère de résistance au moment de flexion.

Afin de trouver la section minimale satisfaisant à un certain W_{el}, on aura besoin d'établir une relation liant ces deux grandeurs. Comme on a :

$$S = bh \text{ et } W_{el} = \tfrac{bh^2}{6}$$

on trouve :

$$S = \sqrt{6 W_{el} b} = (6b)^{0,5} \, (W_{el})^{0,5} \tag{3.34}$$

La figure 3.3 montre l'évolution de S en fonction de W_{el}, d'après l'équation 3.34, pour différentes largeurs courantes d'éléments.

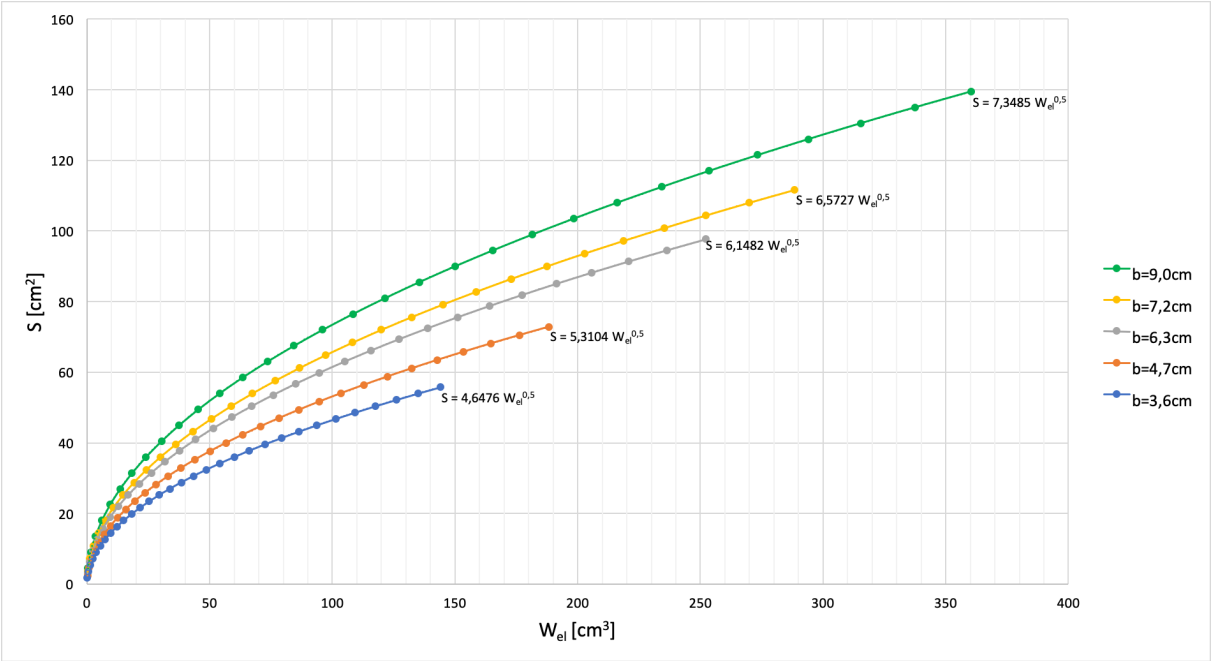


FIGURE 3.3 – Evolution de S en fonction de W_{el} pour différentes largeurs b

On remarque dans la figure 3.3 que la plus petite section satisfaisant une valeur donnée de W_{el} correspond à la plus petite largeur b disponible.

3.4.2 Calcul à l'ELS

Les critères de dimensionnement ELS pour une poutre sur deux appuis sont donnés par les équations 2.23 et 2.27. On peut les résumer par l'équation 3.35, où l'indice t peut désigner l'indice $inst$ ou l'indice fin.

$$\frac{5q_{ELS,t}L^4}{384EI_y} + \frac{q_{ELS,t}L^2}{8GA_v} \leqslant \frac{L}{\eta_t} \tag{3.35}$$

Le problème se résume donc à trouver un module d'inertie I_y minimal (et l'aire réduite qui y est associée, $A_v = 5bh/6$) tel que les critères 3.35 soient satisfaits. En menant le même raisonnement que dans le cas de l'ELU, on devra choisir la plus petite section $S = b \times h$ possible qui satisfait au module d'inertie I_y minumum requis afin d'optimiser le volume de bois utilisé pour chaque poutre.

De manière analogue au cas de l'ELU, on a ici besoin d'établir une relation liant la section S au module d'inertie I_y. Comme on a :

$$S = bh \text{ et } I_y = \frac{bh^3}{12}$$

on trouve :

$$S = \sqrt[3]{12I_y b^2} = \left(12b^2\right)^{0,33} (I_y)^{0,33} \tag{3.36}$$

La figure 3.4 montre l'évolution de S en fonction de I_y, d'après l'équation 3.36, pour différentes largeurs courantes d'éléments.

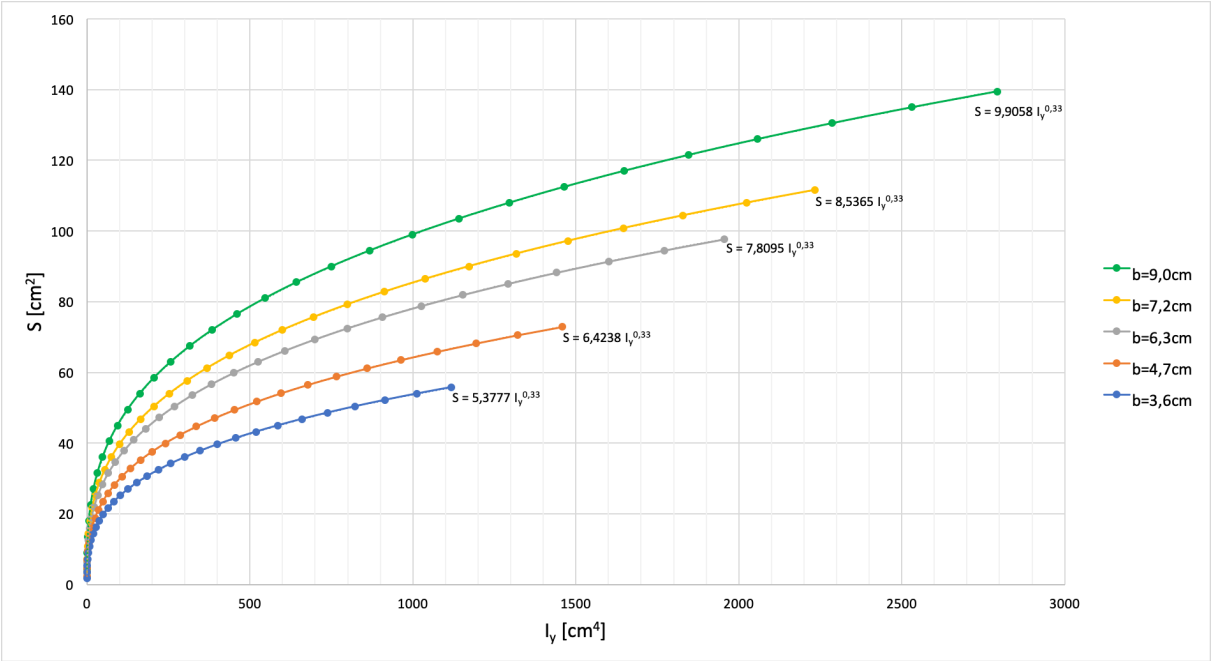


FIGURE 3.4 – Evolution de S en fonction de I_y pour différentes largeurs b

On remarque dans la figure 3.4 que la plus petite section satisfaisant à une valeur donnée de I_y correspond à la plus petite largeur b disponible.

3.4.3 Conclusion

Les observations précédentes montrent que, afin d'optimiser le volume de bois utilisé, il faut opter pour une section de largeur la plus petite possible. En pratique, lors du dimensionnement d'une structure, on aura à notre disposition une certaine plage de largeurs et de hauteurs de sections disponibles. Il faut alors procéder par itération, en commençant le calcul avec la plus petite largeur b disponible ; si la hauteur minimale h ainsi calculée n'est pas disponible pour cette largeur, on recommence le calcul avec la valeur supérieure disponible pour b.

Dans le cadre de notre recherche théorique du volume minimal de bois d'un gîtage, on part donc du principe que le paramètre b n'est plus une inconnue mais une donnée du problème, et qu'on recherche la hauteur minimale h requise pour la section.

3.5 Ultime reformulation des critères selon la hauteur de section nécessaire

Dans la section précédente, nous avons montré qu'afin de minimiser le volume d'une poutre satisfaisant les critères de dimensionnement il fallait recourir à la plus petite largeur de section b possible, puis calculer la hauteur minimale h nécessaire pour que la section satisfasse à tous ces critères. Par conséquent, le paramètre b peut être considéré comme une donnée connue au moment où on vérifiera les critères de dimensionnement, tout comme la portée L.

Nous cherchons donc, dans cette section, à reformuler les critères des équations 3.8, 3.12, 3.17 et 3.19 afin qu'ils portent sur la hauteur de section uniquement. On y fera également apparaître les autres paramètres déterminant la configuration du gîtage, à savoir b, L et e.

3.5.1 Critère ELU de flexion

On cherche à définir un critère sur la hauteur de section minimale $h_{ELU,f}$ qu'il faut pour satisfaire au critère ELU sur le moment de flexion.

On prend comme point de départ le critère 3.8, en y remplaçant Z_d par sa définition (équation 3.21) :

$$\left(\frac{L}{h_{ELU,f}}\right)^2 \leq \frac{k_h k_{sys} k_{mod} f_{m,k}}{0,75\gamma_M Q_{ELU}} \frac{b}{e} \tag{3.37}$$

En isolant $h_{ELU,f}$, on trouve :

$$L^2 \frac{0,75\gamma_M Q_{ELU}}{k_h k_{sys} k_{mod} f_{m,k}} \frac{e}{b} \leq h_{ELU,f}^2 \tag{3.38}$$

$$h_{ELU,f} \geq L\sqrt{\frac{0,75\gamma_M Q_{ELU}}{k_h k_{sys} k_{mod} f_{m,k}} \frac{e}{b}} \tag{3.39}$$

On remarque qu'on retrouve le paramètre Y_d, défini par l'équation 3.22, à l'intérieur de la racine dans l'équation 3.39. Finalement, le critère s'écrit :

$$\boxed{h_{ELU,f} \geq L Y_d^{\frac{1}{2}} \left(\frac{b}{e}\right)^{-\frac{1}{2}}} \tag{3.40}$$

3.5.2 Critère ELU de cisaillement

De manière similaire, on cherche à définir un critère sur la hauteur de section minimale $h_{ELU,v}$ qu'il faut pour satisfaire au critère ELU sur le cisaillement.

On prend comme point de départ le critère 3.12, en y remplaçant Z_d par sa définition (équation 3.21) :

$$k_{M-V}\left(\frac{L}{h_{ELU,v}}\right) \leq \frac{k_h k_{sys} k_{mod} f_{m,k}}{0,75\gamma_M Q_{ELU}} \frac{b}{e} \tag{3.41}$$

En isolant $h_{ELU,v}$, on trouve :

$$h_{ELU,v} \geq L k_{M-V} \frac{0,75\gamma_M Q_{ELU}}{k_h k_{sys} k_{mod} f_{m,k}} \frac{e}{b} \tag{3.42}$$

On remarque qu'on retrouve le paramètre Y_d, défini par l'équation 3.22, dans le membre de droite de l'équation 3.42. Finalement, le critère s'écrit :

$$\boxed{h_{ELU,v} \geq L Y_d k_{M-V} \left(\frac{b}{e}\right)^{-1}} \tag{3.43}$$

3.5.3 Critères ELS : flèche instantanée et flèche finale

Enfin, on cherche à définir un critère sur la hauteur de section minimale h_{ELS} qu'il faut pour satisfaire aux critères ELS sur la flèche (court et long terme).

On prend comme point de départ les critères 3.17 et 3.19. On peut les regrouper en un seul critère de la manière suivante :

$$k_{inst} max\left(\eta_{inst}, k_{fin}\eta_{fin}\right)\left[\frac{5}{32}\left(\frac{L}{h_{ELS}}\right)^3 + \frac{3}{20}\frac{E}{G}\left(\frac{L}{h_{ELS}}\right)\right] \leq Z_d \tag{3.44}$$

Afin de pouvoir isoler h_{ELS}, on réarrange d'abord les termes afin de faire apparaître un polynôme en L/h, de la forme $x^3 + ax^2 + bx + c \leq 0$:

$$\left(\frac{L}{h_{ELS}}\right)^3 + \frac{3 \times 32}{20 \times 5}\frac{E}{G}\left(\frac{L}{h_{ELS}}\right) - \frac{32 Z_d}{5 max\left(\eta_{inst}, k_{fin}\eta_{fin}\right) k_{inst}} \leq 0 \tag{3.45}$$

On y retrouve les paramètres B et C (définis par les équations 3.23 et 3.25) dans le troisième et deuxième terme du polynôme respectivement. On peut donc réécrire l'équation 3.45 comme suit :

$$\left(\frac{L}{h_{ELS}}\right)^3 + C\left(\frac{L}{h_{ELS}}\right) - B \leq 0 \tag{3.46}$$

La méthode de Cardan (voir encadré ci-dessous) permet de trouver les solutions d'un polynôme de la forme $x^3 + px + q = 0$. Dans notre cas, on obtient :

$$\left(\frac{L}{h_{ELS}}\right) \leq \left(\frac{B - \sqrt{B^2 + \frac{4C^3}{27}}}{2}\right)^{\frac{1}{3}} + \left(\frac{B + \sqrt{B^2 + \frac{4C^3}{27}}}{2}\right)^{\frac{1}{3}} \tag{3.47}$$

Démonstration de l'équation 3.47 :

La méthode de Cardan (1545) permet de trouver les trois racines x_k d'un polynôme de la forme $x^3 + px + q = 0$, où p et q sont des coefficients réels. Ces racines sont données par :

$$x_k = u_k + v_k \text{ avec } 0 \leq k \leq 2$$

avec :

$$u_k = j^k \left(\frac{-q + \sqrt{\frac{-\Delta}{27}}}{2}\right)^{\frac{1}{3}}$$

$$v_k = j^{-k} \left(\frac{-q - \sqrt{\frac{-\Delta}{27}}}{2}\right)^{\frac{1}{3}}$$

où j est un nombre complexe donné par :

$$j = e^{\frac{2i\pi}{3}}$$

et où Δ est le discriminant de l'équation et vaut :

$$\Delta = -(4p^3 + 27q^2)$$

Démonstration de l'équation 3.47 (suite) :

Le signe du discriminant indiquera la forme de l'ensemble des solutions.

1. si $\Delta > 0$: il y a trois solutions réelles distinctes
2. si $\Delta = 0$: les trois solutions sont réelles et confondues
3. si $\Delta < 0$: il y a une seule solution réelle et deux solutions complexes conjuguées

Dans notre cas, le discriminant vaut :

$$\Delta = -(4C^3 + 27(-B)^2) = -(4C^3 + 27B^2)$$

Comme C sera toujours positif, Δ sera toujours négatif, et donc l'équation 3.47 admet une seule solution réelle. Pour la trouver, nous réécrivons j^k et j^{-k} sous forme trigonométrique :

$$j^k = e^{\frac{2i\pi k}{3}} = \cos\frac{2\pi k}{3} + i\sin\frac{2\pi k}{3}$$

$$j^{-k} = e^{-\frac{2i\pi k}{3}} = \cos -\frac{2\pi k}{3} + i\sin -\frac{2\pi k}{3}$$

La solution réelle correspond au cas où j^k et j^{-k} sont réels, et donc à la valeur de k pour laquelle on a :

$$\sin\frac{2\pi k}{3} = \sin -\frac{2\pi k}{3} = 0$$

Cela correspond à $k = 0$. Pour cette valeur de k, on a :

$$\cos\frac{2\pi k}{3} = \cos -\frac{2\pi k}{3} = \cos 0 = 1$$

et donc $j^k = j^{-k} = 1$.

La seule solution réelle de l'équation 3.47 est donc bien donnée par, en remplaçant x par (L/h_{ELS}), p par C et q par B :

$$\left(\frac{L}{h_{ELS}}\right) = \left(\frac{-B + \sqrt{\frac{4C^3+27B^2}{27}}}{2}\right)^{\frac{1}{3}} + \left(\frac{-B - \sqrt{\frac{4C^3+27B^2}{27}}}{2}\right)^{\frac{1}{3}}$$

Finalement, en isolant h_{ELS}, on trouve le critère suivant pour les calculs à l'ELS :

$$\boxed{h_{ELS} \geq L\left[\left(\frac{B - \sqrt{B^2 + \frac{4C^3}{27}}}{2}\right)^{\frac{1}{3}} + \left(\frac{B + \sqrt{B^2 + \frac{4C^3}{27}}}{2}\right)^{\frac{1}{3}}\right]^{-1}} \tag{3.48}$$

3.5.4 Expression générale du critère de dimensionnement

Les critères que nous avons élaborés aux équations 3.40, 3.43 et 3.48, servent à déterminer la hauteur $h_{min} \geq max\,(h_{ELU,f}, h_{ELU,v}, h_{ELS})$ de section minimale requise pour les éléments du gîtage. On peut donc les combiner en une unique expression.

Pour simplifier les notations, on commence par définir trois nouvelles fonctions, $f_{ELU,f}$,$f_{ELU,v}$ et f_{ELS} :

$$f_{ELU,f}\left(\frac{b}{e}, Y_d\right) = Y_d^{\frac{1}{2}}\left(\frac{b}{e}\right)^{-\frac{1}{2}} \tag{3.49}$$

$$f_{ELU,v}\left(\frac{b}{e}, Y_d, k_{M-V}\right) = Y_d k_{M-V}\left(\frac{b}{e}\right)^{-1} \tag{3.50}$$

$$f_{ELS}\left(B, C\right) = \left[\left(\frac{B - \sqrt{B^2 + \frac{4C^3}{27}}}{2}\right)^{\frac{1}{3}} + \left(\frac{B + \sqrt{B^2 + \frac{4C^3}{27}}}{2}\right)^{\frac{1}{3}}\right]^{-1} \tag{3.51}$$

Sur base des fonctions définies ci-dessus, le critère général de dimensionnement peut se noter de façon à tenir compte de chaque critère ELU et ELS :

$$h_{min} = max\left(Lf_{ELU,f}, Lf_{ELU,v}, Lf_{ELS}\right) \tag{3.52}$$

On remarque qu'on peut mettre le paramètre L en évidence, afin d'obtenir finalement :

$$\boxed{h_{min} = max\left(f_{ELU,f}, f_{ELU,v}, f_{ELS}\right) L} \tag{3.53}$$

Nous avons finalement établi un critère de dimensionnement adimensionnel sous forme d'une expression unique (équation 3.53). Cette reformulation des critères de dimensionnement tels que présentés dans l'Eurocode 5 nous aidera à établir, dans le prochain chapitre, une démarche qui nous permettra de déterminer la solution théorique minimisant le volume de bois utilisé dans un gîtage.

Chapitre 4

Expression générale du volume de bois et optimisation

Sur base des expressions reformulées des critères de dimensionnement établies au chapitre précédent, nous allons maintenant rechercher la solution optimale théorique pour le dimensionnement d'un gîtage. L'utilité de calculer ce volume minimal théorique réside dans le fait de pouvoir comparer les solutions obtenues pour différents problèmes sur base d'un paramètre adimensionnel valable pour tous les problèmes de planchers et toutes les configurations du gîtage.

Pour que nous puissions trouver des tendances qui correspondent à toutes les géométries de planchers rectangulaires et à n'importe quelle combinaison de paramètres, nous allons d'abord exprimer le volume à l'aide d'un indicateur adimensionnel qui dépend lui-même de paramètres adimensionnels. Ensuite, nous analyserons l'évolution graphique de cet indicateur adimensionnel en fonction de ces différents paramètres. Enfin, nous en déduirons la solution de dimensionnement qui, théoriquement, minimise le volume de bois utilisé dans un gîtage, répondant ainsi au premier objectif de ce mémoire.

4.1 Développement de l'expression de l'indicateur du volume de bois

4.1.1 Volume total de bois d'un gîtage

Le volume total de bois Vol qui sera mis en œuvre dans un gîtage dépend du nombre n d'éléments qu'il y aura dans ce gîtage ainsi que des dimensions de ces éléments. Il est donné par :

$$Vol = n \times L \times b \times h \tag{4.1}$$

En remplaçant h par l'expression 3.53 dans l'équation 4.1, on obtient :

$$Vol = n \times L^2 \times b \times max\left(f_{ELU,f}, f_{ELU,v}, f_{ELS}\right) \tag{4.2}$$

Notre objectif est de formuler une expression liant un paramètre adimensionnel lié directement au volume du gîtage, que l'on appelle *indicateur de volume* et que l'on note W, à d'autres paramètres adimensionnels. Nous allons considérer trois approches, et choisirons la plus adaptée.

4.1.2 Première approche

Afin de rendre le terme de gauche de l'équation 4.2 adimensionnel, on propose de diviser les deux membres de cette équation par L^3. On obtient :

$$\frac{Vol}{L^3} = n \times \frac{b}{L} \times max\left(f_{ELU,f}, f_{ELU,v}, f_{ELS}\right) \tag{4.3}$$

Tous les termes de cette équation sont dès lors adimensionnels. Cette expression nous permet de définir l'indicateur de volume W_{L^3} :

$$W_{L^3} = \frac{Vol}{L^3} \tag{4.4}$$

Le paramètre b/L qui intervient dans le membre de droite de l'équation 4.3 n'a pas d'interprétation très intuitive : il s'agit du rapport entre la largeur de la section d'une poutre et sa portée, ce qui n'a pas vraiment de sens physique. On aimerait donc se débarrasser du terme b/L au profit de paramètres déjà utilisés et ayant un sens plus physique. Pour ce faire, attardons-nous sur le nombre n d'éléments qu'il y aura dans le gîtage. Instinctivement, on sait que ce nombre dépend de la largeur P du gîtage (cfr. figure 1.4) et de la taille de l'entraxe e. On a en effet :

$$n = \frac{P}{e} + 1 \tag{4.5}$$

En passant par la définition du paramètre β (équation 1.1), on remplace le paramètre P par le paramètre L, qui est déjà utilisé dans les critères de dimensionnement.

$$n = \frac{L}{\beta e} + 1 \tag{4.6}$$

On aimerait cependant aussi se débarrasser de l'inconnue e au profit d'une inconnue déjà utilisée dans les expressions des critères de dimensionnement. Il suffit de multiplier le premier terme du membre de droite par une expression valant l'unité :

$$n = \frac{L}{\beta e}\frac{e}{b}\frac{b}{e} + 1 \tag{4.7}$$

En effet, on retrouve le rapport b/e, déjà utilisé auparavant. On obtient dès lors l'expression suivante pour le nombre n d'éléments qui constitueront le gîtage :

$$n = \frac{L}{\beta b}\frac{b}{e} + 1 \tag{4.8}$$

L'équation 4.8 permet de remplacer le terme b/L de l'équation 4.3 par des paramètres qui sont déjà présents dans les critères de dimensionnement. En effet, en isolant L/b dans l'équation 4.8, on obtient :

$$\frac{L}{b} = (n-1)\beta\frac{e}{b} \tag{4.9}$$

Et donc, en inversant l'équation 4.9, on a :

$$\frac{b}{L} = \left((n-1)\beta\frac{e}{b}\right)^{-1} = \frac{1}{\beta}\frac{b}{e}\frac{1}{n-1} \tag{4.10}$$

Grâce à l'expression 4.10, on peut réécrire l'équation 4.3 en n'utilisant que des paramètres bien connus, à savoir : le taux de remplissage b/e, le rapport de portée β et le nombre d'éléments n. Finalement, en incluant l'équation 4.4 on trouve :

$$\boxed{W_{L^3} = \frac{1}{\beta}\frac{b}{e}\frac{n}{n-1} \times max\left(f_{ELU,f}, f_{ELU,v}, f_{ELS}\right)} \tag{4.11}$$

L'expression de W_{L^3} dépend du sens de la portée, puisque L est la portée des poutres. Donc, la valeur de L^3 n'est pas la même selon que les éléments du gîtage sont disposés selon le sens long ou le sens court du plancher. Dans l'approche suivante, nous nous basons sur une expression de W ne dépendant pas du sens de la portée, contrairement à l'équation 4.4.

4.1.3 Deuxième approche

Ici, au lieu de diviser les deux membres de l'équation 4.2 par L^3, on les divise par $(LP)^{\frac{3}{2}}$. Ainsi, le membre de gauche est également adimensionnel et il ne dépend plus du sens de la portée des poutres puisque $L \times P$ est la surface du plancher. On définit un nouvel indicateur de volume W_{LP}, qui vaut :

$$W_{LP} = \frac{Vol}{(LP)^{\frac{3}{2}}} \tag{4.12}$$

L'équation 4.2 devient, après avoir divisé ses deux membres par $(LP)^{\frac{3}{2}}$:

$$W_{LP} = \frac{Vol}{(LP)^{\frac{3}{2}}} = n \times \frac{L^2}{L^{\frac{3}{2}}} \times b \times \frac{1}{P^{\frac{3}{2}}} \times max\left(f_{ELU,f}, f_{ELU,v}, f_{ELS}\right) \tag{4.13}$$

En simplifiant les puissances de L, on obtient :

$$W_{LP} = n \times \frac{L^{\frac{1}{2}}}{P^{\frac{3}{2}}} \times b \times max\left(f_{ELU,f}, f_{ELU,v}, f_{ELS}\right) \tag{4.14}$$

Revenons à l'équation 4.5 qui exprime n en fonction de P et e. Si on isole P, on trouve :

$$P = (n-1)e \tag{4.15}$$

En substituant P dans l'équation 4.14 par l'expression donnée à l'équation 4.15, on a :

$$W_{LP} = n \times \frac{L^{\frac{1}{2}}}{(n-1)^{\frac{3}{2}} e^{\frac{3}{2}}} \times b \times max\left(f_{ELU,f}, f_{ELU,v}, f_{ELS}\right) \tag{4.16}$$

En réarrangeant les termes de l'équation 4.16, on peut la réécrire comme suit :

$$W_{LP} = \frac{n}{(n-1)^{\frac{3}{2}}} \times \frac{L^{\frac{1}{2}}}{e^{\frac{1}{2}}} \times \frac{b}{e} \times max\left(f_{ELU,f}, f_{ELU,v}, f_{ELS}\right) \tag{4.17}$$

L'équation 4.6 nous permet d'exprimer L/e en fonction de n et β. En isolant L/e dans cette équation, on trouve :

$$\frac{L}{e} = (n-1)\beta \tag{4.18}$$

En substituant L/e dans l'équation 4.17 par l'expression de l'équation 4.18, on obtient :

$$W_{LP} = \frac{n}{(n-1)^{\frac{3}{2}}} \times ((n-1)\beta)^{\frac{1}{2}} \times \frac{b}{e} \times max\left(f_{ELU,f}, f_{ELU,v}, f_{ELS}\right) \tag{4.19}$$

En simplifiant les puissances de $(n-1)$ dans l'équation 4.19, on obtient finalement :

$$\boxed{W_{LP} = \frac{n}{n-1} \times \beta^{\frac{1}{2}} \times \frac{b}{e} \times max\left(f_{ELU,f}, f_{ELU,v}, f_{ELS}\right)} \tag{4.20}$$

Ici, la définition de W_{LP} (équation 4.12) ne dépend plus du sens de la portée, puisque LP est la surface du plancher et ne change donc pas en fonction du sens de la portée. Cependant, la valeur de W_{LP} (équation 4.20) dépend encore de β et donc du sens de la portée.

4.1.4 Troisième approche

Afin que la valeur de l'indicateur de volume ne dépende plus du sens de la portée mais seulement de la géométrie du plancher, on peut inclure le paramètre β dans l'expression de l'indicateur de volume. On se base sur l'équation 4.20, dont on divise les deux membres par $\beta^{\frac{1}{2}}$. On définit alors un nouvel indicateur de volume W_β, qui vaut :

$$W_\beta = \frac{Vol}{(LP)^{\frac{3}{2}}\beta^{\frac{1}{2}}} \tag{4.21}$$

Comme $\beta = L/P$, on peut simplifier l'équation 4.21 comme suit :

$$W_\beta = \frac{Vol}{L^2 P} \tag{4.22}$$

Comme dit ci-dessus, on divise les deux membres de l'équation 4.20 par $\beta^{\frac{1}{2}}$ afin d'obtenir l'indicateur de volume W_β. Donc, cet indicateur de volume vaut :

$$\boxed{W_\beta = \frac{n}{n-1} \times \frac{b}{e} \times max\left(f_{ELU,f}, f_{ELU,v}, f_{ELS}\right)} \tag{4.23}$$

L'équation 4.23 ne dépend pas directement du sens de la portée, puisqu'on n'y retrouve plus L, P ni β (les fonctions $f_{ELU,f}$, $f_{ELU,v}$ et f_{ELS} ne dépendent pas non plus de ces trois grandeurs). Pour un certain nombre de poutres n et un certain rapport b/e, la valeur de l'indicateur de volume W_β est donc constante pour un plancher, quel que soit le sens de la portée.

4.1.5 Choix de l'expression finale de l'indicateur de volume

La troisième approche, celle de l'indicateur de volume W_β, a le désavantage qu'un plancher aura une unique valeur de W_β pour exprimer deux volumes différents. En effet, le volume du gîtage sera donné, en réarrangeant les termes de l'équation 4.22, par :

$$Vol = W_\beta \times PL^2 \tag{4.24}$$

Comme les valeurs de L et de P dépendent du sens de la portée des poutres, le terme PL^2 ne sera pas le même selon que les poutres sont placées dans le sens long ou dans le sens court. La valeur de W_β étant unique pour exprimer ces deux volumes différents, il ne sera pas possible de comparer les deux cas sur base de cet indicateur de volume.

L'indicateur de volume W_{L^3} développé dans la première approche a bien une valeur différente selon le sens de la portée des poutres puisqu'il dépend de β (cfr. équation 4.11). Cependant, la formule menant au volume du gîtage dépendra aussi du sens de la portée des poutres. En effet, le volume du gîtage sera donné, en réarrangeant les termes de l'équation 4.4, par :

$$Vol = W_{L^3} \times L^3 \tag{4.25}$$

Comme la valeur de L dépend du sens de la portée des poutres, le terme L^3 ne sera pas le même selon que les poutres sont placées dans le sens long ou dans le sens court.

La deuxième approche répond aux inconvénients des deux autres : l'indicateur de volume W_{LP} a une valeur différente selon le sens de la portée des poutres, ce qui permet de comparer les deux cas, et la formule menant au volume ne dépend pas du sens de la portée des poutres. Le volume associé à une certaine valeur de W_{LP} ne dépend en effet que de la surface LP du plancher étudié. Ce volume est donné par, en réarrangeant les termes de l'équation 4.12 :

$$Vol = W_{LP} \times (LP)^{\frac{3}{2}} \tag{4.26}$$

Dans la suite de ce mémoire, c'est donc cet indicateur de volume W_{LP} que l'on choisit d'analyser. A partir d'ici, on notera l'indicateur de volume W_{LP} simplement W. De manière très générale, on peut référencer comme suit les paramètres adimensionnels desquels sa valeur dépend :

$$\boxed{W = fonction\left(n, \frac{b}{e}, Y_d, k_{M-V}, \beta, \theta, C\right)} \tag{4.27}$$

4.2 Hypothèse sur la classe de résistance du bois

Pour la suite, on fera l'hypothèse qu'on utilise du bois C24, qui est la classe de résistance la plus commune en Belgique et dont les caractéristiques sont données en annexe dans le tableau A.8. De plus, on considère une valeur moyenne de $k_{mod} = 0,75$ et de $\eta_{inst} = 400 = k_{fin}\eta_{fin}$. On en déduit, grâce aux équations 3.2, 3.3, 3.22, 3.24 et 3.25, les valeurs suivantes :

- $f_{m,k} = 24$ [MPa]
- $f_{v,k} = 4$ [MPa]
- $E = 11000$ [MPa]
- $G = 690$ [MPa]
- $k_{M-V} = 8,96$
- $k_{inst} = 13 \times 10^{-4}$
- $\theta = 12,3$
- $C = 15,30$
- $0,15 < Y_d < 0,55$

Cela nous permet de retirer plusieurs variables de l'expression de W. La fonction 4.27 devient :

$$W = fonction\left(n, \frac{b}{e}, Y_d, \beta\right) \tag{4.28}$$

Sur base de la définition de l'indicateur de volume W et des hypothèses faites ci-dessus, nous conduirons, dans la section suivante, une analyse graphique de l'évolution de W en fonction des paramètres n, b/e, Y_d et β.

4.3 Analyse graphique de W

On propose, pour différentes valeurs courantes de Y_d, de n et de β, de tracer l'évolution de l'indicateur de volume W en fonction du taux de remplissage b/e.

Le paramètre β décrit la géométrie du plancher, mais aussi et surtout le sens de la portée des poutres (cfr. section 1.2.2). Nous tracerons des graphes pour les valeurs suivantes de β :

- $\beta = 1,0$
- $\beta = 0,5$
- $\beta = 2,0$

Le paramètre Y_d varie, pour le bois massif, entre approximativement 5×10^{-5} et 100×10^{-5} (cfr. section 3.3.2). Nous tracerons des graphes pour les valeurs suivantes :

- $Y_d = 5 \times 10^{-5}$
- $Y_d = 40 \times 10^{-5}$
- $Y_d = 70 \times 10^{-5}$
- $Y_d = 100 \times 10^{-5}$

Enfin, dans chaque graphique, nous tracerons les courbes pour des valeurs de n allant de 5 à 30. Ces graphiques sont présentés aux figures 4.1 à 4.12.

Des schémas de planchers illustrant les différentes valeurs de β et de n considérées sont fournis à la figure 4.13 pour un taux de remplissage du plancher valant $b/e = 0,3$.

4.3.1 Graphiques pour $\beta = 1,0$

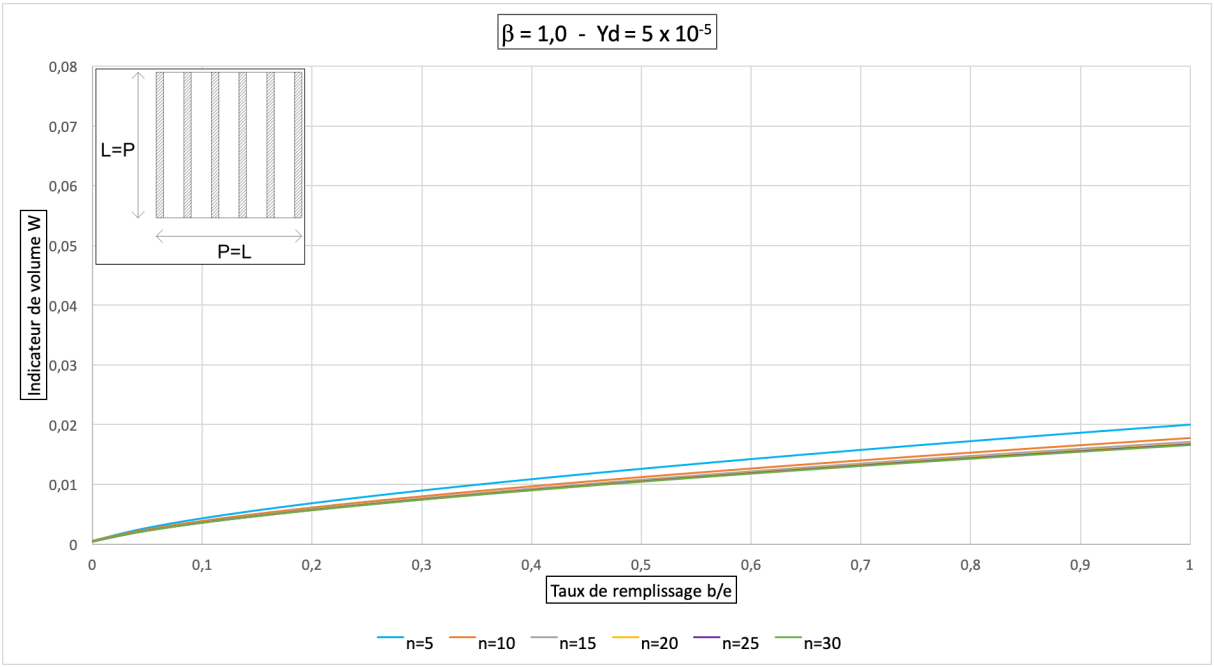


FIGURE 4.1 – Evolution de W en fonction de b/e pour différentes valeurs de n pour : $\beta = 1,0$ et $Y_d = 5 \times 10^{-5}$

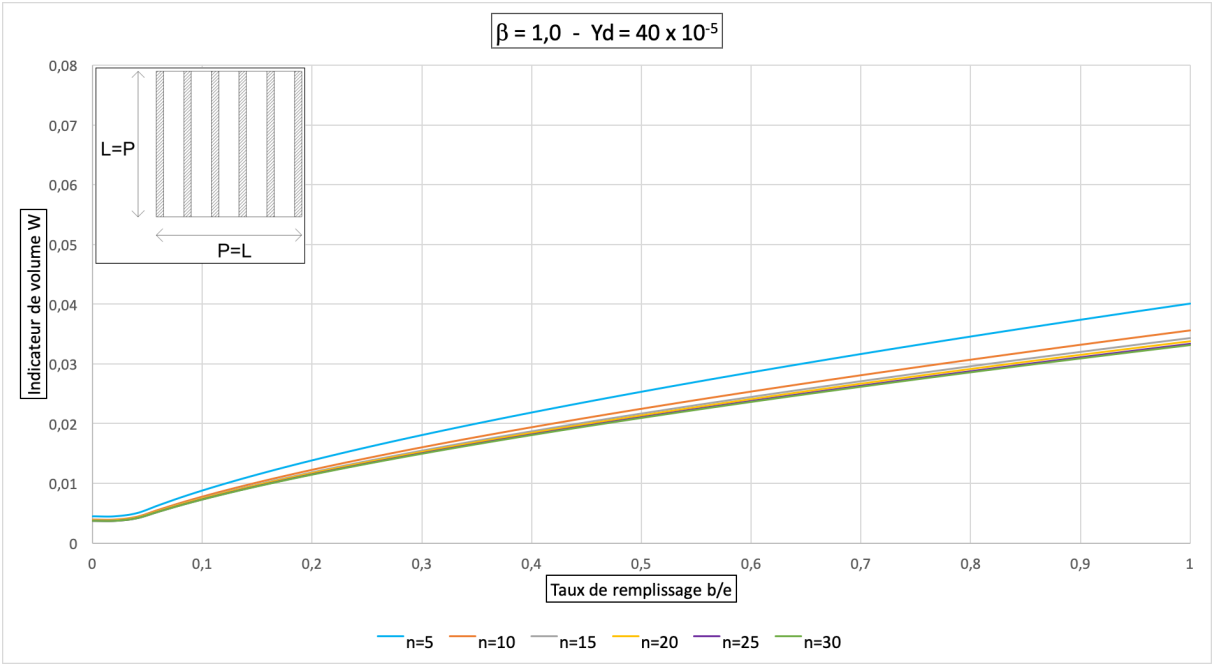


FIGURE 4.2 – Evolution de W en fonction de b/e pour différentes valeurs de n pour : $\beta = 1,0$ et $Y_d = 40 \times 10^{-5}$

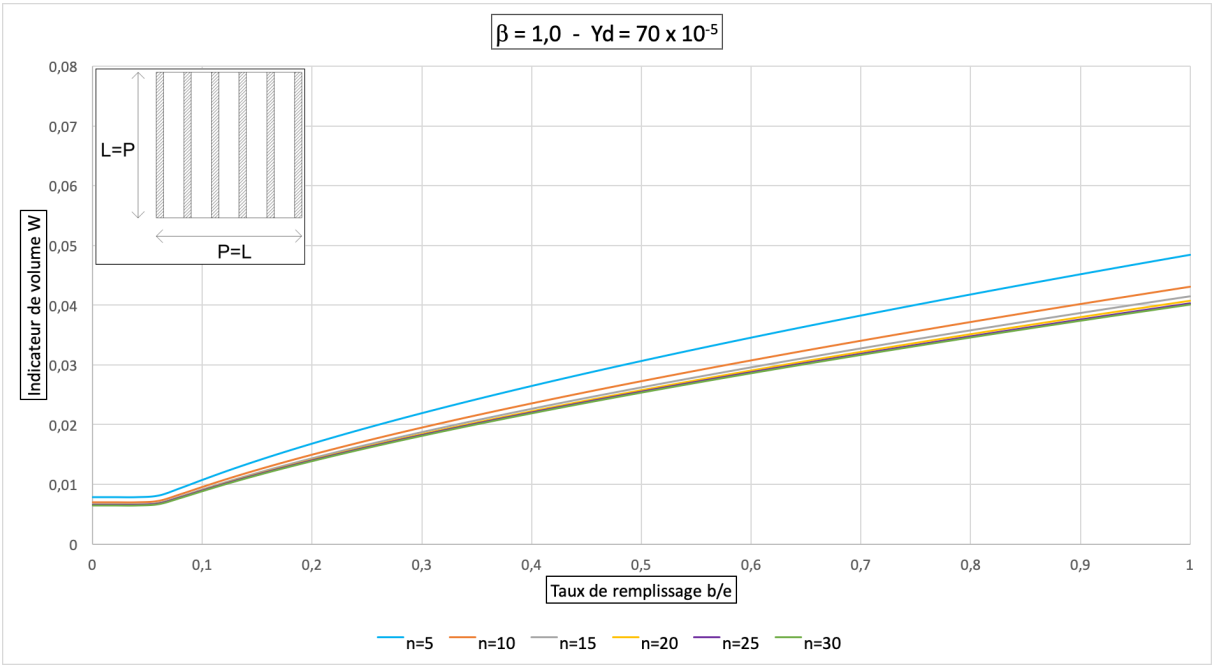


FIGURE 4.3 – Evolution de W en fonction de b/e pour différentes valeurs de n pour : $\beta = 1,0$ et $Y_d = 70 \times 10^{-5}$

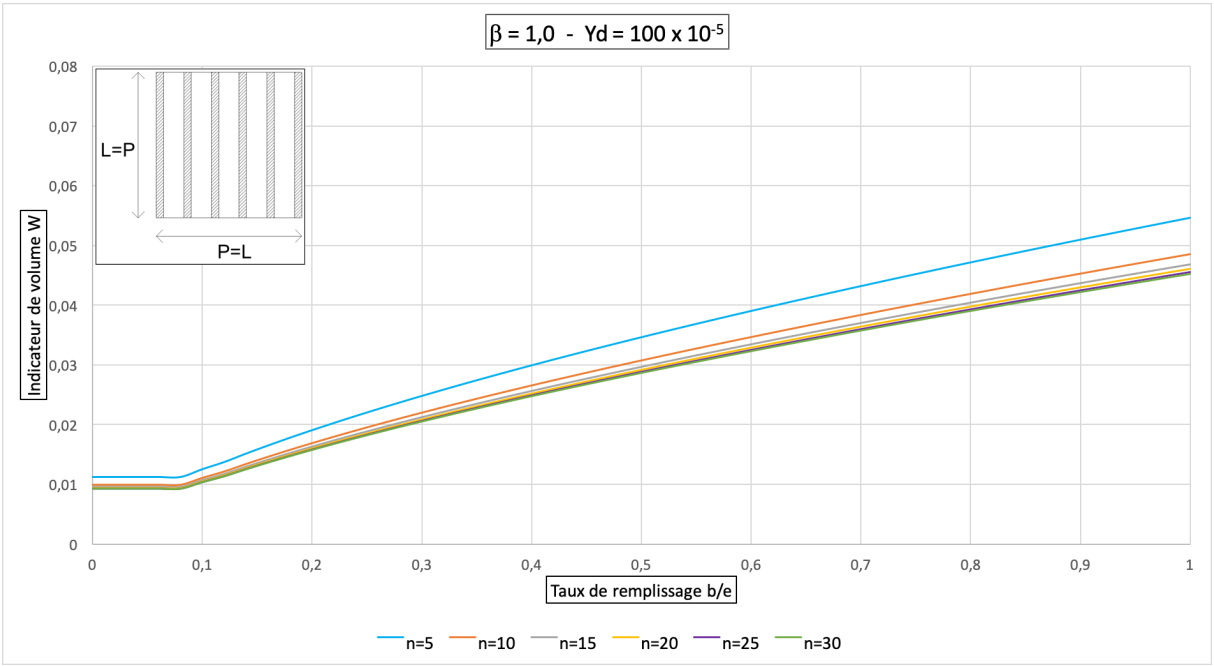


FIGURE 4.4 – Evolution de W en fonction de b/e pour différentes valeurs de n pour : $\beta = 1,0$ et $Y_d = 100 \times 10^{-5}$

4.3.2 Graphiques pour $\beta = 0,5$

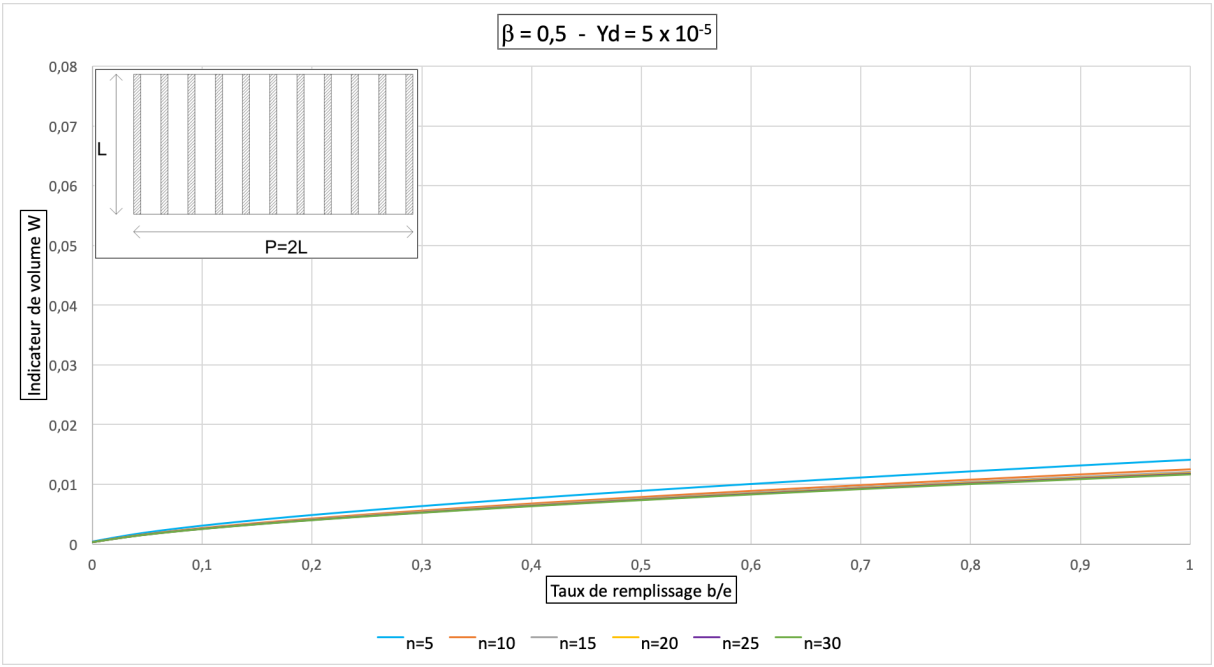


FIGURE 4.5 – Evolution de W en fonction de b/e pour différentes valeurs de n pour : $\beta = 0,5$ et $Y_d = 5 \times 10^{-5}$

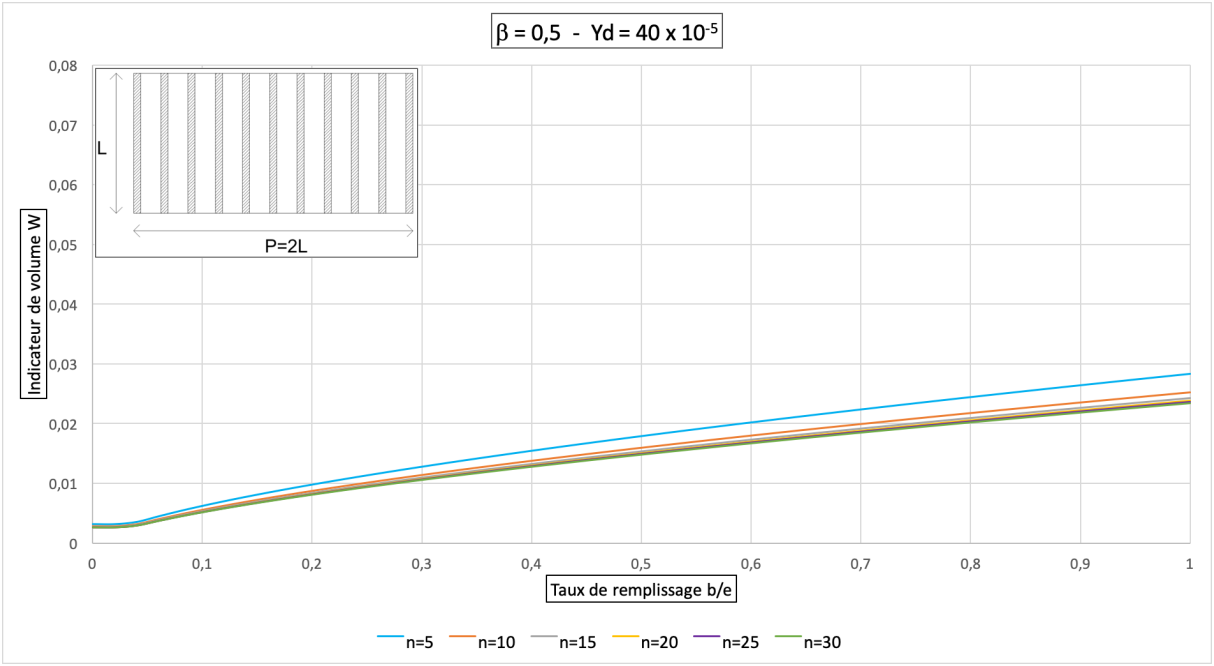


FIGURE 4.6 – Evolution de W en fonction de b/e pour différentes valeurs de n pour : $\beta = 0,5$ et $Y_d = 40 \times 10^{-5}$

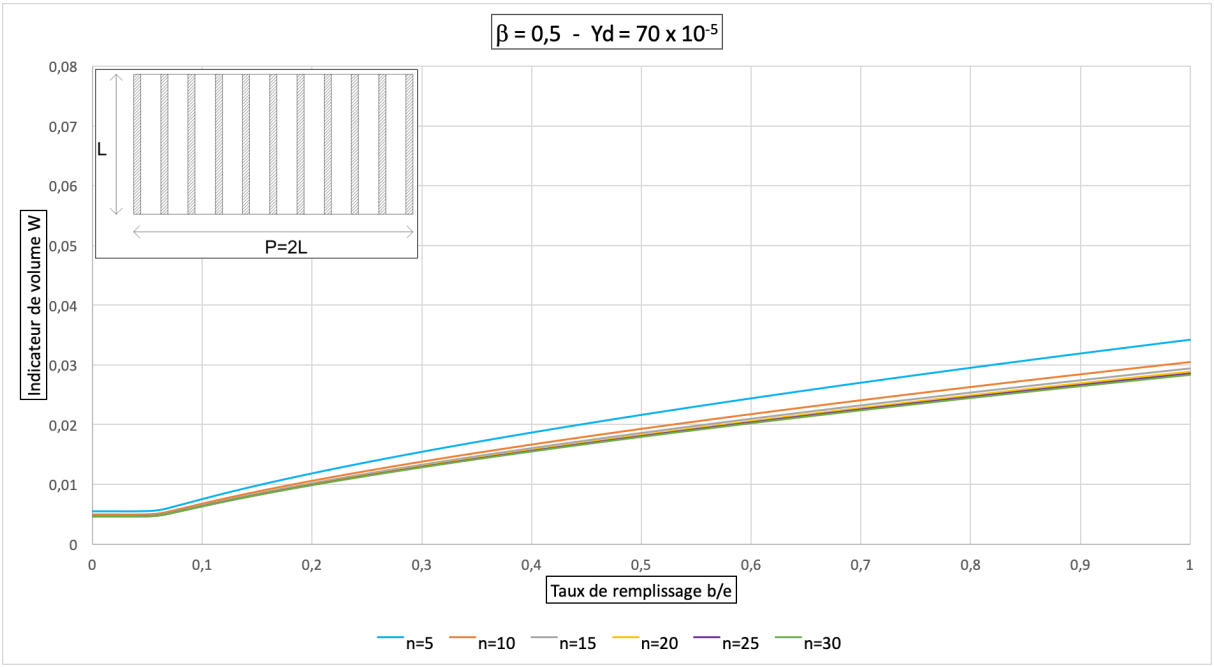


FIGURE 4.7 – Evolution de W en fonction de b/e pour différentes valeurs de n pour : $\beta = 0,5$ et $Y_d = 70 \times 10^{-5}$

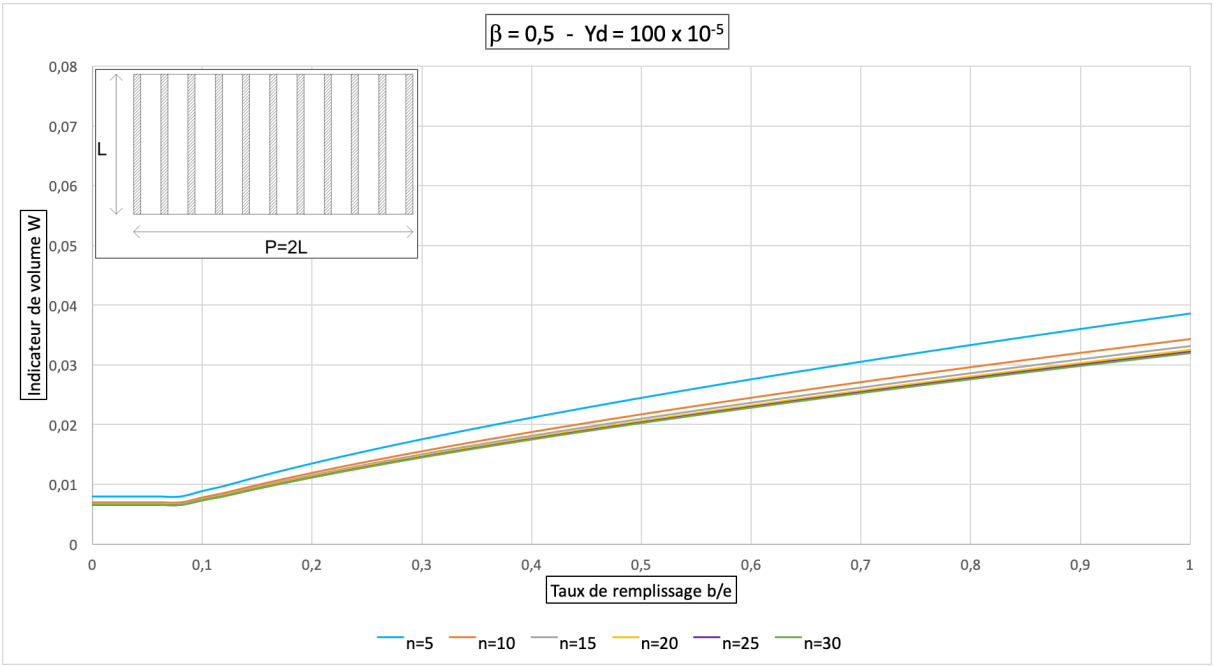


FIGURE 4.8 – Evolution de W en fonction de b/e pour différentes valeurs de n pour : $\beta = 0,5$ et $Y_d = 100 \times 10^{-5}$

4.3.3 Graphiques pour $\beta = 2,0$

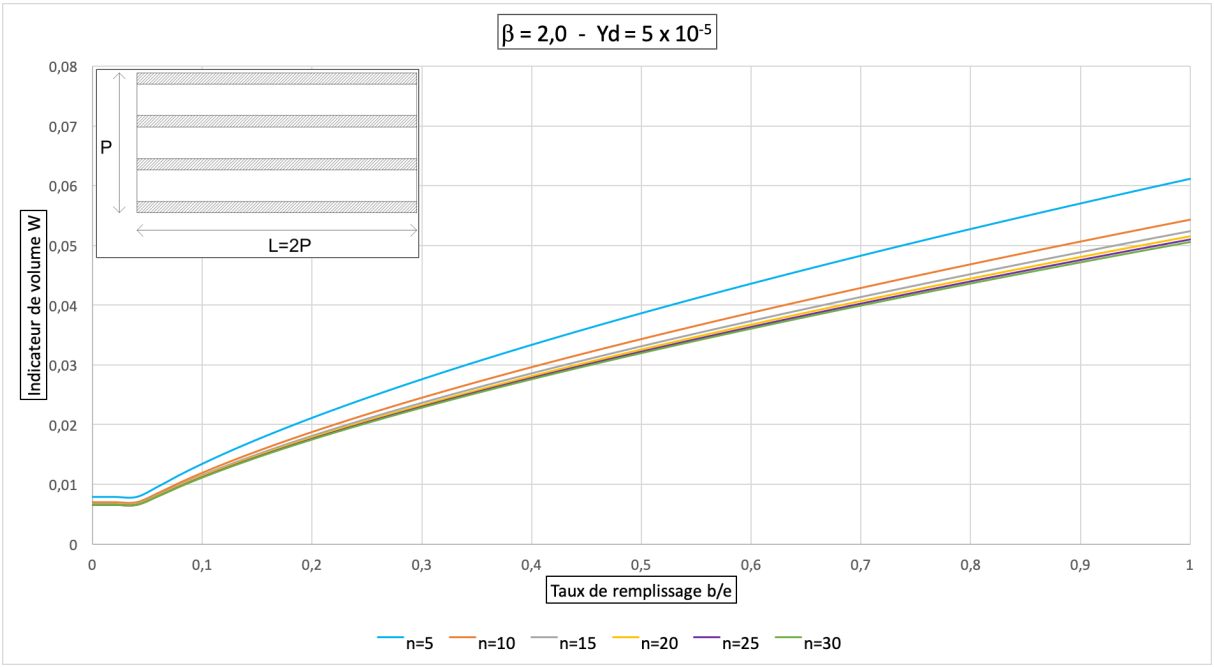


FIGURE 4.9 – Evolution de W en fonction de b/e pour différentes valeurs de n pour : $\beta = 2,0$ et $Y_d = 5 \times 10^{-5}$

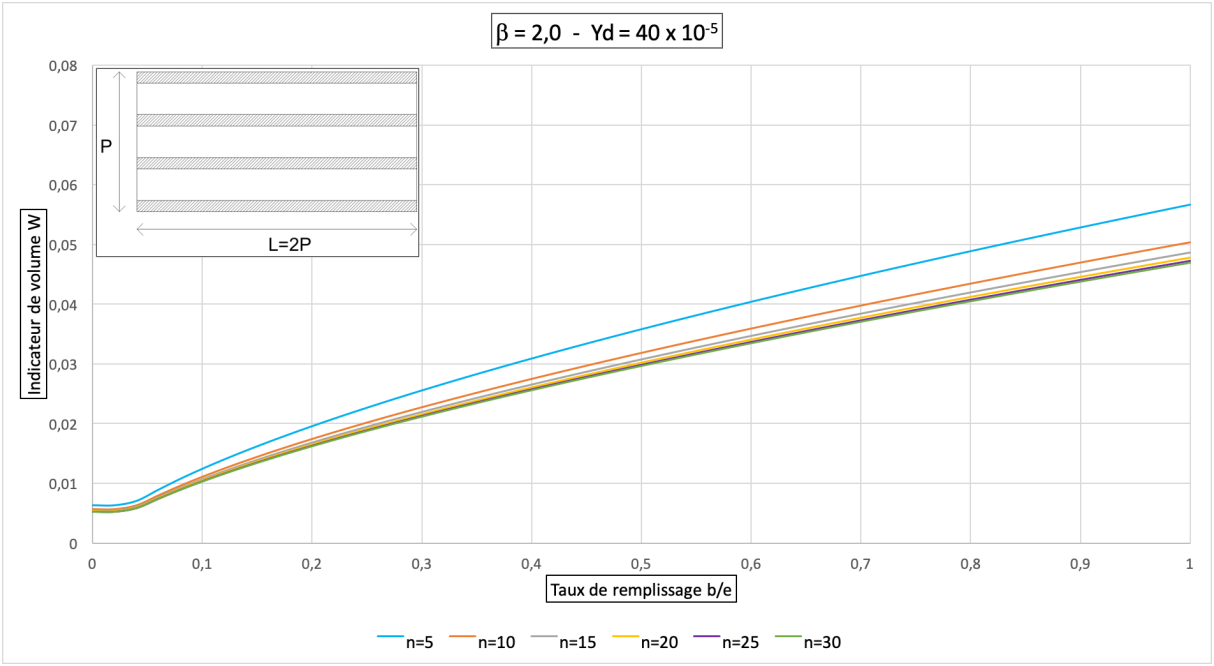


FIGURE 4.10 – Evolution de W en fonction de b/e pour différentes valeurs de n pour : $\beta = 2,0$ et $Y_d = 40 \times 10^{-5}$

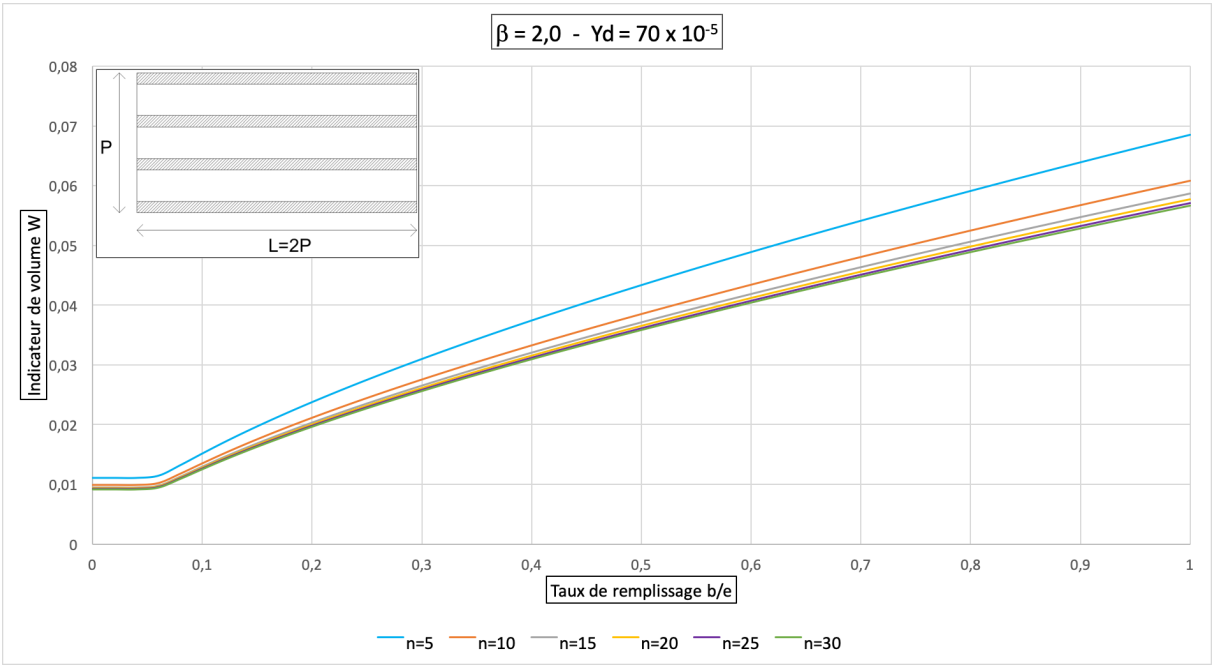


FIGURE 4.11 – Evolution de W en fonction de b/e pour différentes valeurs de n pour : $\beta = 2,0$ et $Y_d = 70 \times 10^{-5}$

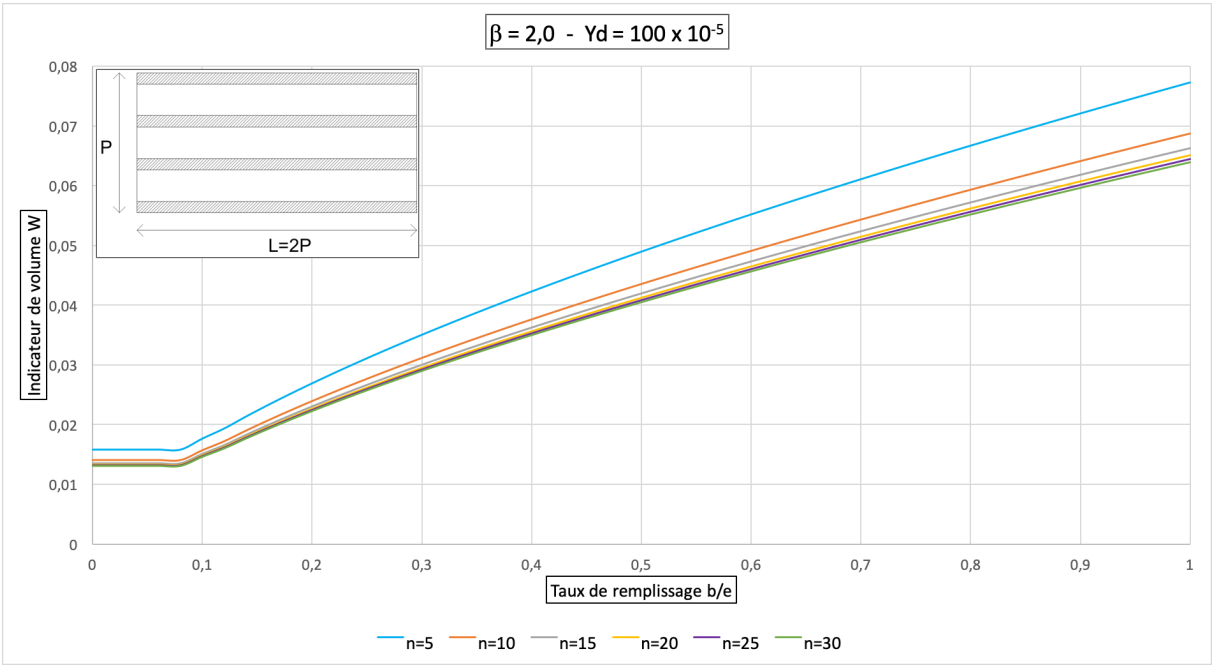
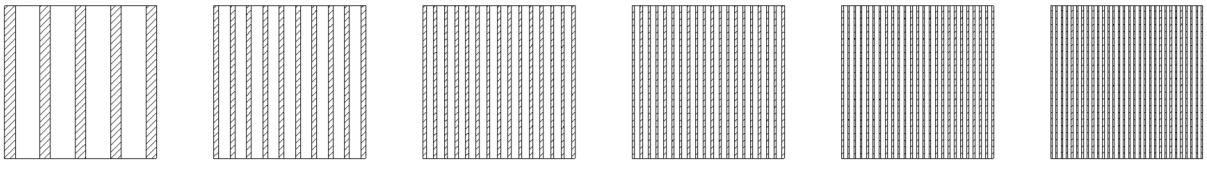
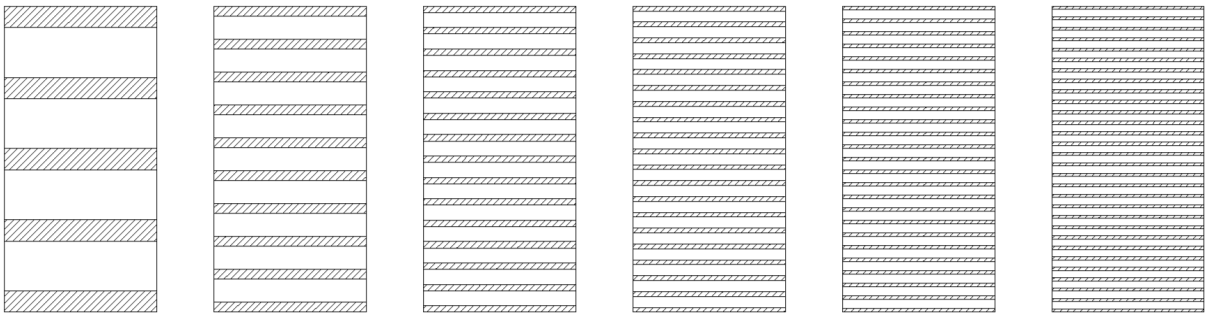


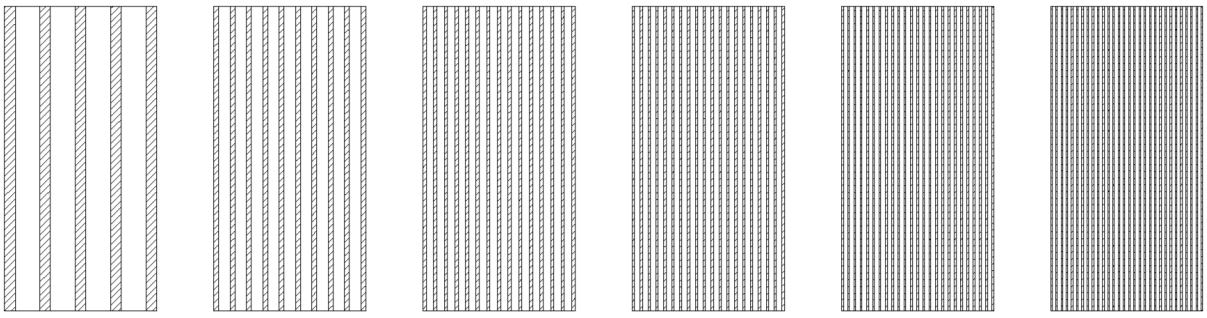
FIGURE 4.12 – Evolution de W en fonction de b/e pour différentes valeurs de n pour : $\beta = 2,0$ et $Y_d = 100 \times 10^{-5}$



(a) $\beta = 1,0$ et $n = 5, 10, 15, 20, 25, 30$



(b) $\beta = 0,5$ et $n = 5, 10, 15, 20, 25, 30$



(c) $\beta = 2,0$ et $n = 5, 10, 15, 20, 25, 30$

FIGURE 4.13 – Schémas de planchers illustrant les différentes valeurs de β et de n considérées pour les graphiques pour $b/e = 0,3$

4.3.4 Analyse des graphes 4.1 à 4.12

En étudiant les graphiques présentés aux figures 4.1 à 4.12, on peut faire les observations suivantes :

- Pour une même valeur de β, l'indicateur de volume W sera plus petit pour des valeurs de Y_d plus petite (et pour un même rapport b/e). La valeur de Y_d diminue lorsque le rapport entre les sollicitations (au numérateur) et la résistance (au dénominateur) diminue. Il est donc logique que le volume de bois diminue aussi.
- Dans tous les cas, l'indicateur de volume W est à son minimum lorsque le taux de remplissage b/e est (quasi) nul, ce qui équivaut à avoir un entraxe maximal par rapport à une largeur d'élément infinitésimale (et donc une hauteur d'élément très importante). Cette observation très théorique est à nuancer par le fait que, d'une part, l'entraxe e aura en réalité une borne supérieure (par exemple, $80[cm]$) qui dépend des autres éléments constituant le plancher. D'autre part, la largeur d'une poutre aura quant à elle une limite inférieure (par exemple, $3[cm]$).
- Dans tous les cas, pour un même rapport b/e, l'indicateur de volume W est plus faible pour un plus grand nombre n d'éléments. Compte tenu de l'observation précédente, cela veut dire qu'il sera préférable d'avoir recours à plus de poutres plus fines et plus rapprochées que d'utiliser moins de poutres plus espacées mais plus épaisses. Ce phénomène est illustré à la figure 4.14.

- Pour des mêmes valeurs de Y_d, de b/e et de n, l'indicateur de volume W est plus faible pour un β inférieur à 1, ce qui correspond au cas où la portée des poutres est disposée selon le sens court (cfr. section 1.2.2). Or, le volume d'un gîtage correspondant à une certaine valeur de W est proportionnel à la surface du plancher et ne dépend pas du sens de la portée (cfr. équation 4.12). Cela indique que, pour un même plancher, le volume du gîtage dont les poutres sont disposées dans le sens court est toujours plus faible que celui du gîtage dont les poutres sont disposées dans le sens long. Cette observation est à nuancer, puisqu'on peut supposer qu'il n'est pas réaliste de considérer un même nombre de poutres et un même rapport b/e pour les deux solutions, à savoir la disposition dans le sens long ou dans le sens court. Cette observation reste cependant intéressante si nous l'analysons pour des valeurs limites de b/e et de n. Nous y reviendrons plus tard dans ce chapitre.

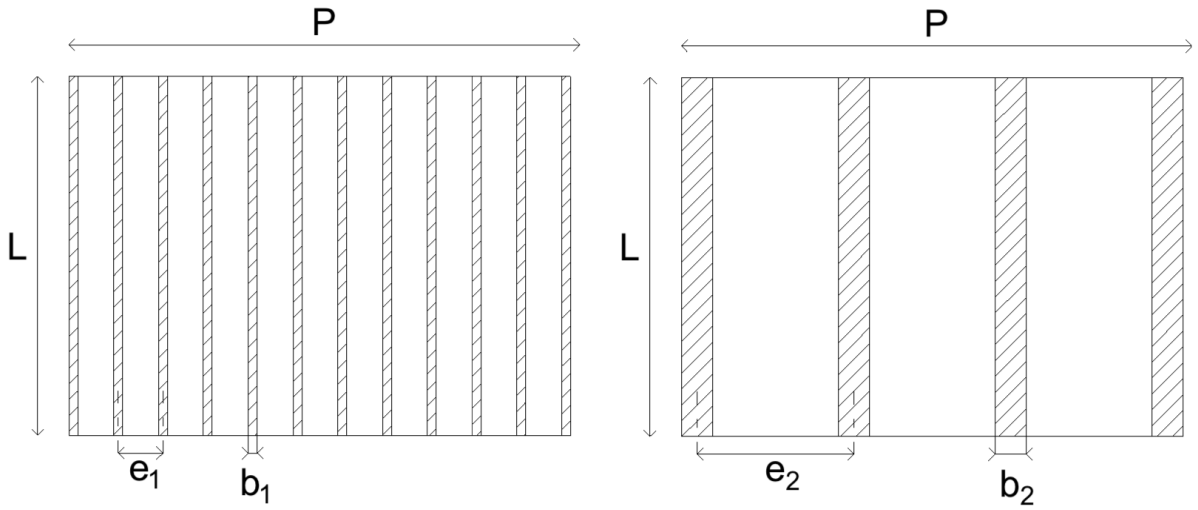


FIGURE 4.14 – Illustration de la variation de n pour un même β et un même rapport $\frac{b_1}{e_1} = \frac{b_2}{e_2}$

La zone horizontale, plus ou moins grande, au début des courbes s'explique par le fait que c'est le critère ELU de résistance au cisaillement qui est déterminant (c'est-à-dire lorsque $max(f_{ELU,f}, f_{ELU,v}, f_{ELS}) = f_{ELU,v}$). Dans ce cas, l'indicateur de volume W vaut (cfr. équation 4.20) :

$$W = \frac{n}{n-1} \times \beta^{\frac{1}{2}} \times \frac{b}{e} \times f_{ELU,v} \tag{4.29}$$

En reprenant la définition de $f_{ELU,v}$ à l'équation 3.50, W vaut donc :

$$W = \frac{n}{n-1} \times \beta^{\frac{1}{2}} \times \frac{b}{e} \times Y_d \times k_{M-V} \times \left(\frac{b}{e}\right)^{-1} \tag{4.30}$$

Dans l'équation 4.30, les termes (b/e) et $(b/e)^{-1}$ se simplifient. La valeur de W ne dépend donc pas de b/e, ce qui explique la zone horizontale puisque b/e correspond à l'axe des abscisses. Cela permet aussi de constater que le critère ELU de résistance au cisaillement est déterminant pour de petites valeurs de b/e. Par ailleurs, la valeur de b/e pour laquelle un autre critère devient déterminant augmente lorsque Y_d augmente.

4.4 Introduction de concepts basés sur l'indicateur de volume : volume minimal théorique et rendement d'un gîtage

Les observations faites dans la section précédente nous permettent de définir un nouveau concept, celui de *volume minimal théorique* d'un gîtage pour un certain plancher. En effet, on sait maintenant que la valeur minimale de l'indicateur de volume W sera atteinte pour un rapport b/e très petit et un nombre n d'éléments très important. Le volume minimal théorique d'un gîtage vaut donc, en se référant à l'équation 4.26 :

$$Vol_{min,th} = W_{\frac{b}{e}\to 0;n\to\infty} \times (LP)^{\frac{3}{2}} \tag{4.31}$$

En pratique, il sera bien entendu impossible d'atteindre cette valeur puisque les sections disponibles auront une largeur minimale et une hauteur maximale. De plus, l'épaisseur d'un plancher sera souvent limitée à une certaine hauteur maximale.

L'utilité de calculer ce volume minimal théorique réside dans le fait de pouvoir comparer les solutions obtenues pour différents problèmes sur base d'un paramètre adimensionnel valable pour tous les problèmes de plancher et toutes les configurations du gîtage. On définit ce paramètre, que l'on appelle *rendement du gîtage*, par :

$$\mu = \frac{Vol_{min,th}}{Vol_{sol}} \tag{4.32}$$

où Vol_{sol} est le volume de bois utilisé par la solution que l'on souhaite comparer. Ce concept de rendement permet de situer une certaine solution obtenue pour un certain gîtage sur une échelle en pourcentage, comprise entre 0% et 100%, commune à tous les gîtages. Ce concept sera illustré pour des exemples réels de planchers au chapitre 6.

4.5 Comparaison des sens court et long

Dans cette section, nous allons comparer les indicateurs de volume W_l et W_c correspondant respectivement à une disposition dans le sens long et à une disposition dans le sens court du gîtage, pour un même plancher rectangulaire. Dans un premier temps, nous comparerons ces deux grandeurs graphiquement. Ensuite, nous les comparerons analytiquement, afin de confirmer l'analyse graphique.

4.5.1 Comparaison graphique

D'abord, on propose de tracer dans un même graphique les courbes correspondant à $n = 30$ des figures 4.5 à 4.12. Les valeurs de $\beta = 2,0$ et $\beta = 0,5$ correspondent à un même plancher, mais une disposition différente du gîtage [1] :

1. $\beta_l = 2,0$ correspond à une disposition des poutres selon le sens long ;
2. $\beta_c = 1/\beta_l = 0,5$ correspond à une disposition des poutres selon le sens court.

Dans la figure 4.15, les courbes correspondant à $\beta = 2,0$ sont tracées en pointillés et celles correspondant à $\beta = 0,5$ en trait continu. Les courbes calculées pour une même valeur de Y_d sont tracées dans une même couleur.

1. Pour différencier les deux valeurs de β correspondant à un même plancher, on utilisera dans cette section les notations β_l pour le cas où $\beta > 1$ (sens long), et β_c pour le cas où $\beta < 1$ (sens court).

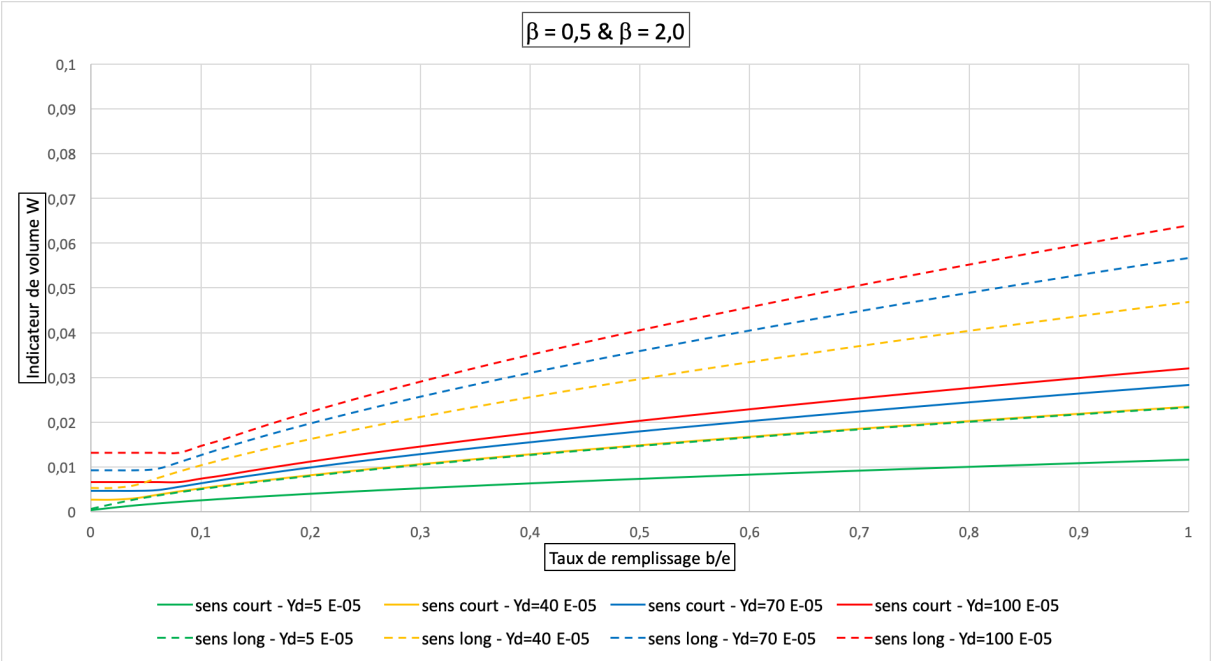


FIGURE 4.15 – Comparaison de l'évolution de l'indicateur de volume en fonction du taux de remplissage b/e pour différentes valeurs de Y_d, pour $\beta = 2,0$ et $\beta = 0,5$ et pour $n = 30$

Cette figure [2] permet de mettre en évidence un fait que nous avons déjà constaté précédemment (section 4.3) : pour des mêmes valeurs de Y_d et de b/e (et de n), l'indicateur de volume W est plus élevé pour un β supérieur à 1 (traits en pointillés), c'est-à-dire le cas où la portée des poutres est disposée selon le sens long. Comme le rapport entre le volume d'un gîtage et la valeur correspondante de W est proportionnel à la surface du plancher et ne dépend pas du sens de la portée, le volume sera également toujours plus élevé dans le cas où $\beta > 1$ (pour de mêmes Y_d, b/e et n). Nous avions cependant noté que cela était à nuancer, puisqu'on peut supposer qu'en réalité le nombre de poutres n et/ou le rapport b/e ne sera pas identique dans les deux dispositions.

La figure 4.16 illustre la même chose que la figure 4.15 mais pour des valeurs de β valant $\beta_l = 4,0$ et $\beta_c = 0,25$. Par rapport à la figure 4.15, les écarts entre les valeurs de W des deux dispositions sont plus importants. Cette différence d'écart fait l'objet d'une analyse détaillée plus tard dans cette section.

2. Les courbes "sens court - $Y_d = 40E-05$" et "sens long - $Y_d = 5E-05$" de la figure 4.15 semblent confondues. En réalité, elles sont très légèrement écartées (d'une différence de l'ordre de 10^{-6}). Cette proximité est fortuite et n'est due qu'au choix des paramètres. On peut d'ailleurs constater que cela n'est plus le cas lorsqu'on change les valeurs de β, à la figure 4.16.

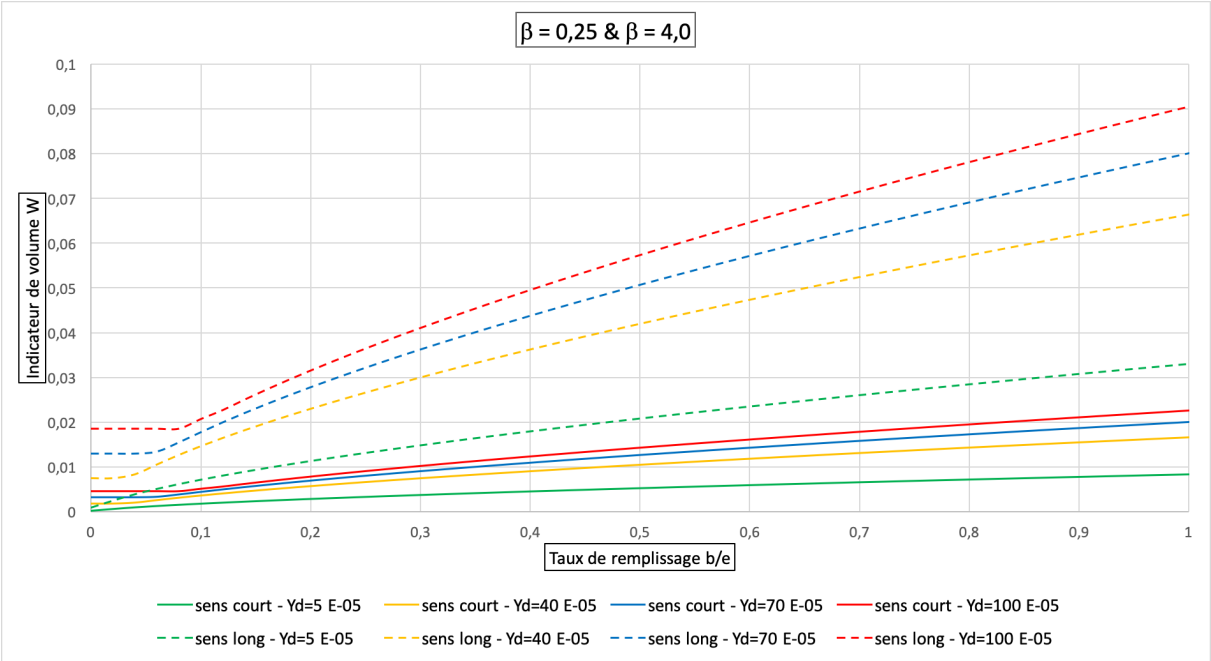


FIGURE 4.16 – Comparaison de l'évolution de l'indicateur de volume en fonction du taux de remplissage b/e pour différentes valeurs de Y_d, pour $\beta = 4,0$ et $\beta = 0,25$ et pour $n = 30$

Ce qui nous intéresse particulièrement est la comparaison entre les volumes minimaux théoriques des deux cas. Or, nous avons vu précédemment dans ce chapitre que l'indicateur de volume menant au volume minimal théorique correspond à $(b/e) \to 0$ et $n \to \infty$. Ces valeurs limites sont valables dans les deux dispositions (dans le sens long ou court), et nous pouvons dès lors comparer les volumes minimaux théoriques des deux dispositions sans nous soucier de la valeur réelle de n ou de b/e dans les deux cas.

Les figures 4.17 à 4.20 comparent les évolutions des indicateurs de volume minimaux pour les deux dispositions (selon le sens long et court) en fonction du rapport de portée β dans le cas de la disposition selon le sens court ($\beta = \beta_c$) pour différentes valeurs de Y_d. L'avantage d'utiliser le paramètre β_c est de pouvoir couvrir toutes les formes de planchers rectangulaires possibles, en le faisant varier de 0 à 1. En effet, le cas $\beta_c = 1 = \beta_l$ correspond à un plancher carré, tandis que $\beta_c \to 0 \Leftrightarrow \beta_l \to \infty$ correspond à une forme qui tend vers une ligne. La valeur des indicateurs de volume y est donc calculée pour $(b/e) \to 0$ et $n \to \infty$. Dans la suite, on note $W_{c,min}$ et $W_{l,min}$ pour les indicateurs de volume minimaux pour la disposition selon le sens court et selon le sens long respectivement.

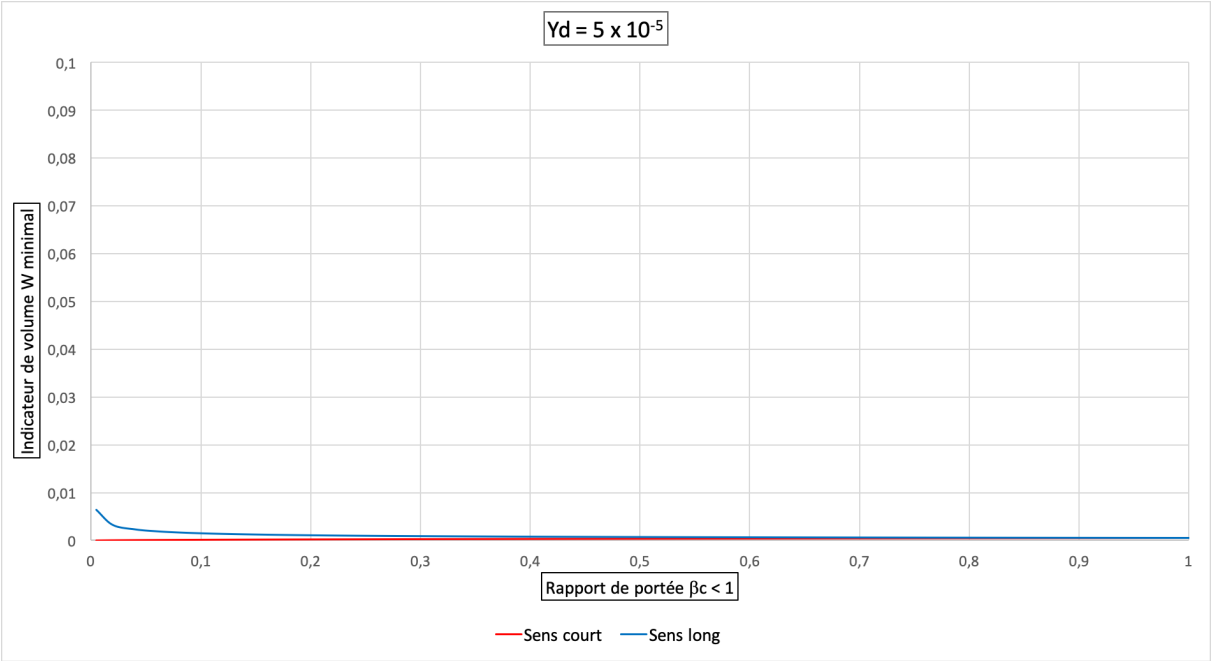


FIGURE 4.17 – Evolution des indicateurs de volume minimaux pour les deux dispositions en fonction de β_c pour $Y_d = 5 \times 10^{-5}$

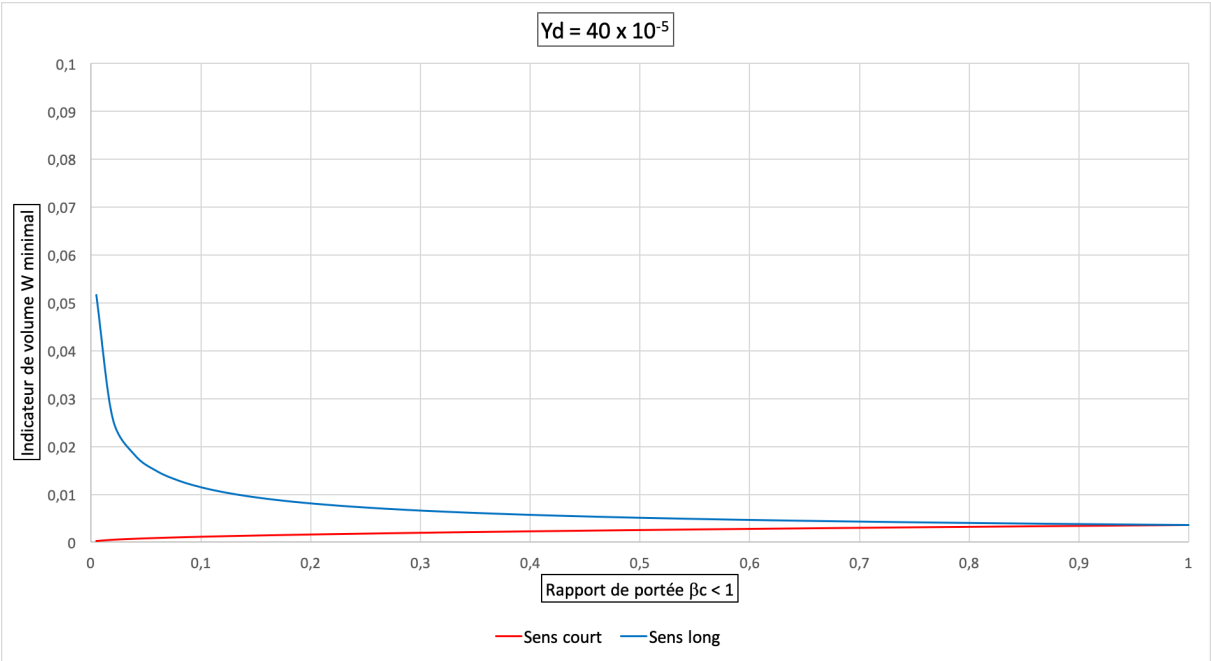


FIGURE 4.18 – Evolution des indicateurs de volume minimaux pour les deux dispositions en fonction de β_c pour $Y_d = 40 \times 10^{-5}$

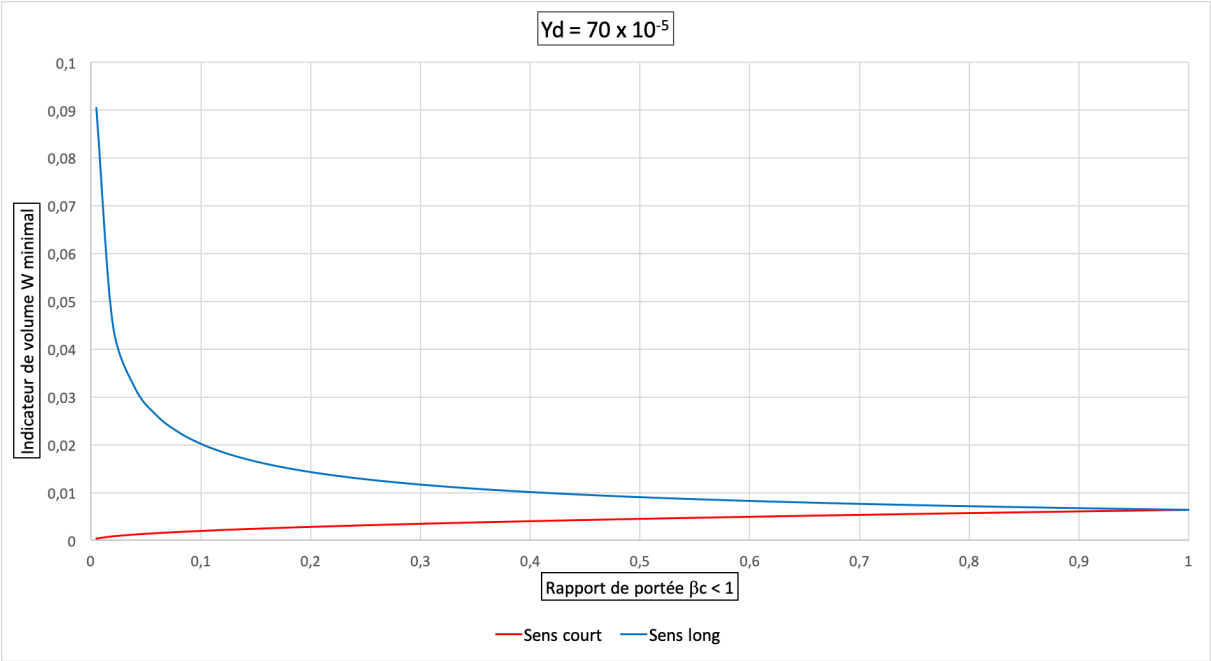


FIGURE 4.19 – Evolution des indicateurs de volume minimaux pour les deux dispositions en fonction de β_c pour $Y_d = 70 \times 10^{-5}$

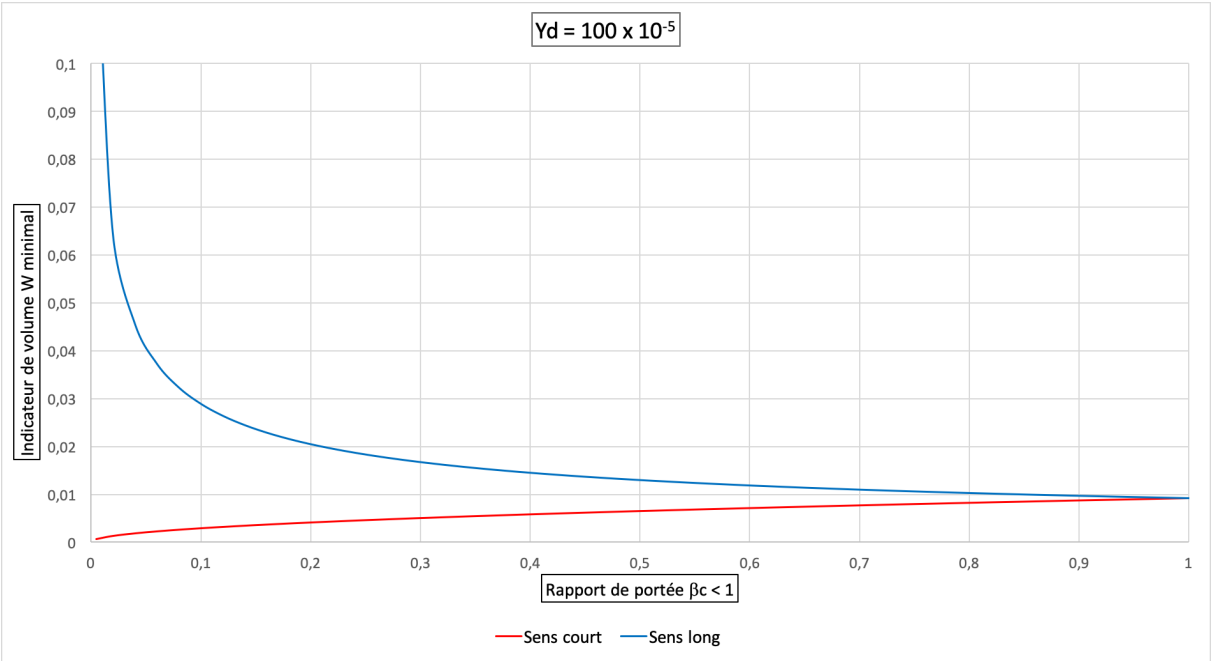


FIGURE 4.20 – Evolution des indicateurs de volume minimaux pour les deux dispositions en fonction de β_c pour $Y_d = 100 \times 10^{-5}$

Premièrement, l'indicateur de volume minimal pour la disposition selon le sens court ($W_{c,min}$) est toujours inférieur à celui pour la disposition selon le sens long ($W_{l,min}$). Puisque le volume d'un gîtage lié à une certaine valeur de W dépend uniquement de la surface du plancher (et donc pas du sens de la portée des poutres, cfr. équation 4.12), on peut conclure que le volume minimal théorique d'un gîtage disposé dans le sens court sera toujours plus petit que celui d'un gîtage disposé dans le sens long pour un même plancher rectangulaire.

Deuxièmement, l'écart entre les indicateurs de volume correspondant aux deux dispositions pour un même plancher se réduit au plus on s'approche d'un plancher carré ($\beta = 1$). Cela est logique puisque, pour un plancher carré, il n'y a pas de différence entre "sens long" et "sens court". Au plus la forme du plancher s'écarte de celle d'un carré, au plus la différence $W_{l,min} - W_{c,min}$ entre les indicateurs de volume correspondant aux deux dispositions sera grande. Cette différence évolue de manière exponentielle par rapport à la forme du plancher ; son évolution est illustrée graphiquement à la figure 4.21 en fonction de β_c pour $Y_d = 5 \times 10^{-5}$ et $Y_d = 100 \times 10^{-5}$.

Finalement, en comparant les figures 4.17 à 4.20, on constate que la différence $W_{l,min} - W_{c,min}$ se réduit lorsque la valeur de Y_d diminue. Cela est aussi observable à la figure 4.21.

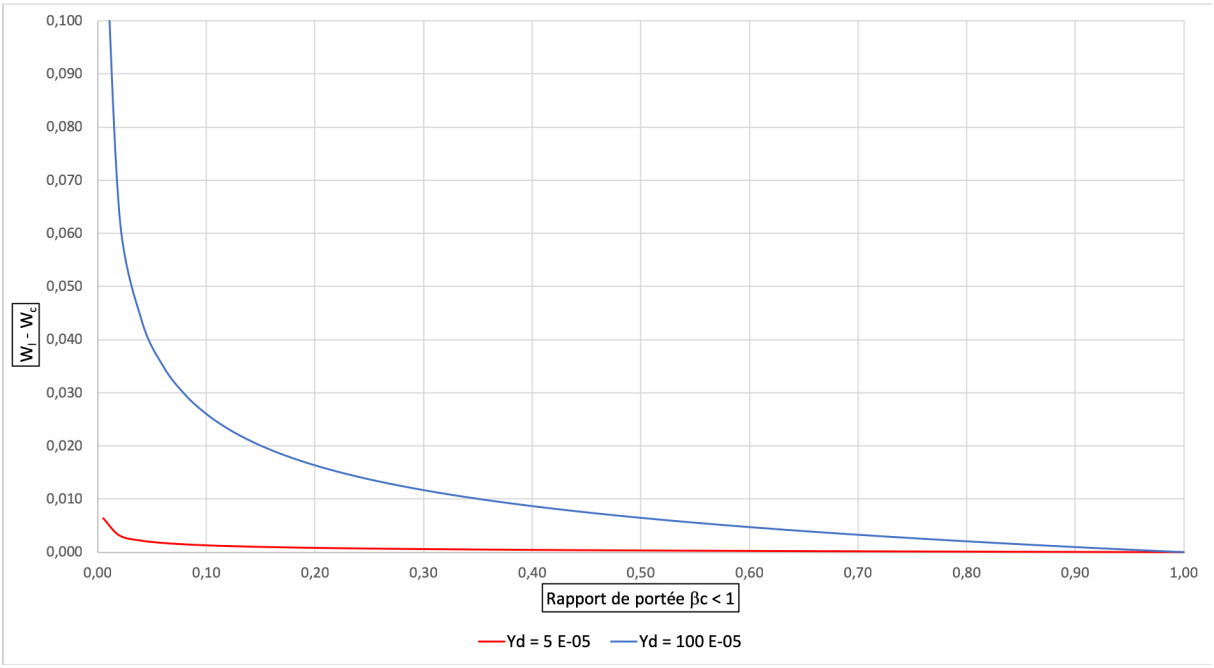


FIGURE 4.21 – Evolution de $W_{l,min} - W_{c,min}$ en fonction de β_c pour $Y_d = 5 \times 10^{-5}$ et $Y_d = 100 \times 10^{-5}$

Une façon plus explicite d'analyser la différence entre les indicateurs de volume correspondant aux deux dispositions pour un même plancher est de s'intéresser au rapport de ceux-ci, soit $W_{c,min}/W_{l,min}$. Ce rapport est illustré graphiquement à la figure 4.22 en fonction de β_c pour $Y_d = 5 \times 10^{-5}$ et $Y_d = 100 \times 10^{-5}$.

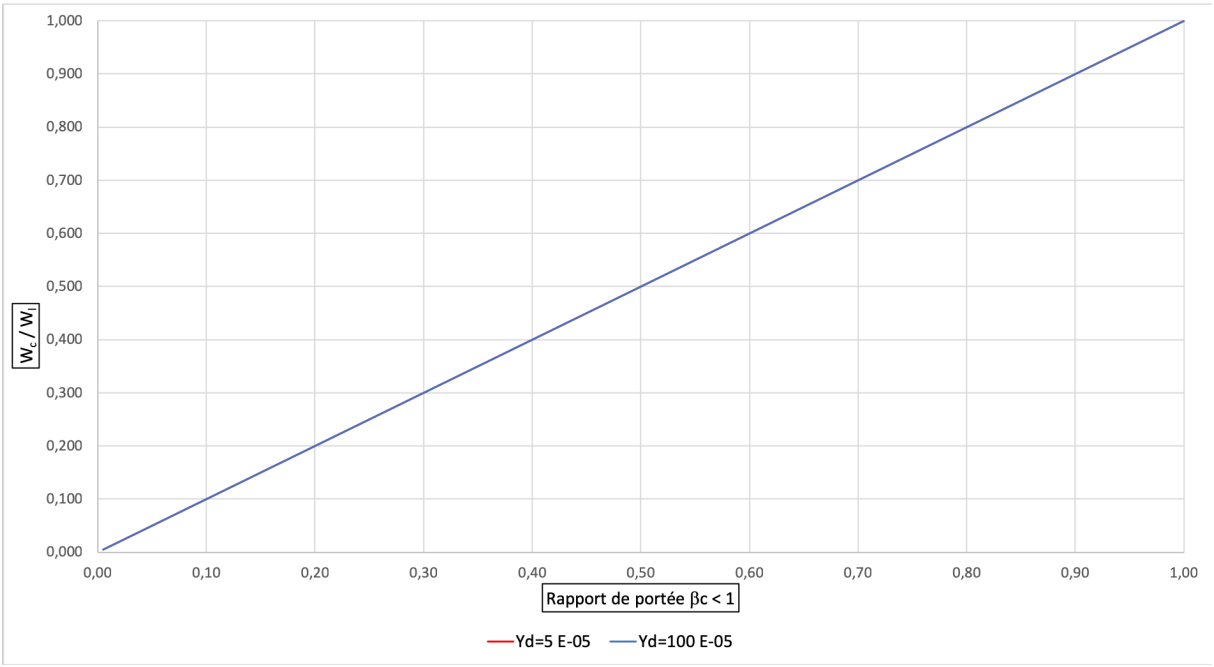


FIGURE 4.22 – Evolution de $W_{c,min}/W_{l,min}$ en fonction de β_c pour $Y_d = 5\times10^{-5}$ et $Y_d = 100\times10^{-5}$

La figure 4.22 permet de faire deux observations intéressantes :

- le rapport $W_{c,min}/W_{l,min}$ est égal à la valeur de β_c ;
- ce rapport ne dépend pas de la valeur de Y_d (les deux droites étant confondues).

Dans la section suivante, nous allons confirmer analytiquement toutes les observations que nous avons faites ci-dessus.

4.5.2 Comparaison analytique

Nous allons comparer les indicateurs de volume, puis les volumes, correspondant aux deux configurations suivantes du gîtage pour un même plancher de surface $L \times P$:

- une disposition des poutres selon le sens long, avec : $n = n_l$, $\frac{b}{e} = \left(\frac{b}{e}\right)_l$ et $\beta = \beta_l$;
- une disposition des poutres selon le sens court, avec : $n = n_c$, $\frac{b}{e} = \left(\frac{b}{e}\right)_c$ et $\beta = \beta_c = 1/\beta_l$.

On note les indicateurs de volume de ces deux configurations W_l et W_c respectivement. Ces deux indicateurs de volume valent, selon l'équation 4.20 :

$$W_l = \frac{n_l}{n_l - 1} \times \beta_l^{\frac{1}{2}} \times \left(\frac{b}{e}\right)_l \times max\left(f_{ELU,f}, f_{ELU,v}, f_{ELS}\right) \tag{4.33}$$

$$W_c = \frac{n_c}{n_c - 1} \times \beta_c^{\frac{1}{2}} \times \left(\frac{b}{e}\right)_c \times max\left(f_{ELU,f}, f_{ELU,v}, f_{ELS}\right) \tag{4.34}$$

Les fonctions $f_{ELU,f}$, $f_{ELU,v}$ et f_{ELS} dépendent de plusieurs paramètres, mais un seul d'entre eux, le taux de remplissage b/e, peut différer selon la disposition du gîtage. Pour rendre cela explicite, nous réécrivons les équations 4.33 et 4.34 comme suit :

$$W_l = \frac{n_l}{n_l - 1} \times \beta_l^{\frac{1}{2}} \times \left(\frac{b}{e}\right)_l \times f_{max}\left(\left(\frac{b}{e}\right)_l\right) \tag{4.35}$$

$$W_c = \frac{n_c}{n_c - 1} \times \beta_c^{\frac{1}{2}} \times \left(\frac{b}{e}\right)_c \times f_{max}\left(\left(\frac{b}{e}\right)_c\right) \tag{4.36}$$

où $f_{max}\left(\frac{b}{e}\right) = max\left(f_{ELU,f}, f_{ELU,v}, f_{ELS}\right)$. Le rapport entre l'indicateur de volume pour le sens long W_l et celui pour le sens court W_c est calculé en divisant l'équation 4.35 par l'équation 4.36. Il vaut :

$$\frac{W_l}{W_c} = \frac{n_l(n_c-1)}{n_c(n_l-1)} \times \left(\frac{\beta_l}{\beta_c}\right)^{\frac{1}{2}} \times \frac{\left(\frac{b}{e}\right)_l}{\left(\frac{b}{e}\right)_c} \times \frac{f_{max}\left(\left(\frac{b}{e}\right)_l\right)}{f_{max}\left(\left(\frac{b}{e}\right)_c\right)} \tag{4.37}$$

Etant donné que $\beta_l = 1/\beta_c$ et que $1/(b/e) = e/b$, on peut réécrire l'équation 4.37 comme suit :

$$\frac{W_l}{W_c} = \frac{n_l(n_c-1)}{n_c(n_l-1)} \times \beta_l \times \left(\frac{b}{e}\right)_l \times \left(\frac{e}{b}\right)_c \times \frac{f_{max}\left(\left(\frac{b}{e}\right)_l\right)}{f_{max}\left(\left(\frac{b}{e}\right)_c\right)} \tag{4.38}$$

Ce qui nous intéresse particulièrement est la comparaison entre les volumes minimaux théoriques des deux cas. Or, nous avons vu précédemment dans ce chapitre que l'indicateur de volume menant au volume minimal théorique correspondait au cas où $(b/e) \to 0$ et $n \to \infty$. Puisque ces valeurs limites sont les mêmes dans les deux cas (long et court), on peut faire les hypothèses suivantes si l'on s'en tient au volume minimal théorique :

1. $n_l = n_c$
2. $\left(\frac{b}{e}\right)_l = \left(\frac{b}{e}\right)_c$

L'équation 4.38 se simplifie alors largement et devient :

$$\frac{W_{l,min}}{W_{c,min}} = \beta_l \tag{4.39}$$

De même, en inversant l'équation 4.39 et puisque $1/\beta_l = \beta_c$, on a aussi :

$$\frac{W_{c,min}}{W_{l,min}} = \beta_c \tag{4.40}$$

Notons $Vol_{l,min}$ le volume minimal théorique correspondant au cas de la disposition dans le sens long et $Vol_{c,min}$ le volume minimal théorique correspondant au cas de la disposition dans le sens court. En combinant les équations 4.12 et 4.39, on trouve :

$$\frac{Vol_{l,min}}{Vol_{c,min}} = \frac{W_{l,min}(LP)^{\frac{3}{2}}}{W_{c,min}(LP)^{\frac{3}{2}}} = \beta_l \tag{4.41}$$

L'équation 4.41 montre que, pour n'importe quel plancher rectangulaire, le volume minimal théorique d'un gîtage disposé dans le sens court sera toujours plus petit que celui d'un gîtage disposé dans le sens long puisque $\beta_l > 1$. On peut d'ailleurs réécrire l'équation 4.41 comme suit :

$$Vol_{l,min} = \beta_l \cdot Vol_{c,min} \tag{4.42}$$

ou encore, vu que $\beta_l = 1/\beta_c$:

$$Vol_{c,min} = \beta_c \cdot Vol_{l,min} \tag{4.43}$$

4.6 Conclusion

Au Chapitre 3 (section 3.4), nous avions déjà vu que, pour obtenir un volume de bois minimal, il fallait opter pour une largeur d'élément b la plus petite possible. Dans ce quatrième chapitre (section 4.3), nous avons constaté que l'entraxe qui conduit au volume le plus faible est l'entraxe maximal possible pour une certaine largeur b (ce qui minimise le rapport b/e). Nous avons vu également qu'il est préférable d'opter pour un nombre d'éléments maximal pour une certaine valeur de b/e. En d'autres mots, le gîtage théoriquement idéal d'un point de vue du volume de bois mis en œuvre serait constitué d'une infinité d'éléments infiniment fins et infiniment hauts.

En pratique, il est impensable d'avoir recours à de tels gîtages, puisque les sections disponibles auront une largeur minimale et une hauteur maximale. De plus, l'épaisseur d'un plancher sera souvent limitée à une certaine hauteur maximale. Il ne sera donc jamais possible d'atteindre le volume minimum théorique de bois nécessaire $Vol_{min,th}$, qui vaut $W_{\frac{b}{e}\to 0;n\to\infty} \times (LP)^{\frac{3}{2}}$.

De plus, lors de la conception d'un gîtage en bois, nous serons toujours limités à une certaine plage de sections commerciales. Les valeurs de b et h devront donc être choisies dans un certain ensemble de valeurs prédéfinies. Ces valeurs changent d'ailleurs souvent d'un fournisseur à l'autre. Afin d'obtenir le couple $(b/e; n)$ optimal, il faudra donc avoir recours à un processus automatisé qui puisse tester un grand nombre de possibilités et en retirer la meilleure, et qui puisse s'adapter à différentes plages de sections disponibles.

C'est pourquoi, afin d'obtenir la section et les valeurs de b/e et de n qui conduiront réellement au gîtage consommant le plus petit volume de bois, nous proposons de créer une application pour smartphone et tablette, dans laquelle l'utilisateur pourra introduire ses propres plages de sections ainsi que les paramètres de son problème de plancher.

Une telle application a l'avantage de pouvoir également prendre en compte des paramètres de prix propres aux sections disponibles, aux éléments utilisés pour les fixations ainsi qu'au coût de la main d'œuvre. L'application permettra ainsi de comparer la solution la moins volumineuse à la solution la moins coûteuse.

Un autre avantage d'une telle application est de pouvoir l'utiliser facilement sur chantier, par exemple pour trouver rapidement la meilleure solution de gîtage sur base de certaines sections qui restent en stock.

Les prochains chapitres sont consacrés au second objectif de ce mémoire, c'est-à-dire la conception de l'application, ainsi qu'à son guide d'utilisation et à l'analyse des résultats fournis par celle-ci. En outre, nous appliquerons les concepts introduits dans ce chapitre (indicateur de volume, volume minimal théorique, rendement d'un gîtage) à des exemples concrets. Ces exemples nous permettront aussi de valider les résultats de l'application.

Chapitre 5

Conception de l'application

Dans les chapitres précédents, nous avons montré quelle est la solution optimale théorique en termes de volume pour le dimensionnement d'un gîtage. Nous sommes également arrivés à la conclusion qu'un logiciel de calcul est, dans la pratique, indispensable afin de trouver la solution réellement optimale sur base d'une plage de sections donnée.

Dans ce chapitre, nous allons développer une application hybride [1] permettant de calculer à la fois la solution optimale en termes de volume de bois et celle en termes de coût. Opter pour une application hybride permettra de pouvoir faire fonctionner l'application sur les trois principales plateformes, à savoir Android, iOS et Windows Phone, à partir d'un seul code source. Nous nous servons du framework Ionic, qui permet de générer des applications pour les différentes plateformes visées sur base des outils Web, en incluant des fonctionnalités natives de chaque plateforme.

Dans un premier temps, nous évoquons brièvement le framework Ionic, ses dépendances et son utilisation. Ensuite, nous passons en revue la structure de l'application et ses différentes fonctionnalités. Nous détaillons, après cela, l'algorithme qui permet de calculer les solutions optimales en termes de volume et de coût. Enfin, un mode d'emploi plus formel est établi en fin de chapitre, indiquant aussi les limitations de l'application, avant de proposer des pistes d'amélioration qui pourraient être incluses dans un éventuel développement futur.

5.1 Environnement de programmation : Ionic Framework

5.1.1 Origine d'Ionic

Ionic est un SDK [2] open-source créé en 2013 par Max Lynch, Ben Sperry, et Adam Bradley. Il est sous licence MIT [3] [9], ce qui signifie qu'Ionic peut être utilisé gratuitement et sans restriction pour des projets personnels ou commerciaux.

Ionic est construit sur base des frameworks Angular et Apache Cordova. Les outils de la CLI [4] sont pour leur part principalement basés sur l'environnement NodeJS.

1. Contrairement à une *application native*, qui est développée spécifiquement pour un certain système d'exploitation (Android, iOS, Windows Phone, ...), une application hybride utilise le navigateur Web intégré du smartphone ou de la tablette ainsi que des technologies Web (HTML, JavaScript et CSS) afin de fonctionner sur différents systèmes d'exploitation, tout en utilisant les fonctionnalités natives des smartphones ou tablettes (notifications, stockage, ...).
2. SDK : Software Development Kit
3. "MIT License : Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so." [8]
4. CLI : Command Line Interface. C'est l'interface qui permet de communiquer directement avec son ordinateur.

Nous utilisons, pour le développement de notre application, la version Ionic 3. Les langages utilisés sont TypeScript, HTML5 et CSS.

5.1.2 Dépendances d'Ionic

Angular

Angular est un framework JavaScript open-source développé par Google qui permet de construire des applications (ou pages) Web. La version d'Ionic que nous utilisons, Ionic 3, est basée sur la version Angular 2. Cette version d'Angular utilise TypeScript, et permet de convertir du code TypeScript en JavaScript (nécessaire pour le Web) [10].

Apache Cordova

Apache Cordova est un framework de développement open-source créé par la Fondation Apache qui permet d'utiliser les technologies web standard (HTML, JavaScript et CSS) pour développer des applications mobiles multi-plateformes. L'application s'exécute à l'intérieur d'un "emballage" spécifique à chaque platforme (Android, iOS, ...) [11].

NodeJS

NodeJS est une platforme open-source créée par Ryan Dahl qui permet d'exécuter du code JavaScript en dehors d'un navigateur Web [12].

5.1.3 Programmation avec Ionic

TypeScript

TypeScript est un langage de programmation développé par Microsoft et qui a pour but d'améliorer JavaScript. Il permet, entre autres, d'étendre complètement JavaScript à la programmation Orientée Objet. TypeScript est une extension de JavaScript et, par conséquent, tout code valide en JavaScript le sera aussi en TypeScript [13].

HTML5

HTML5 est la dernière version majeure du langage informatique HTML (HyperText Markup Language), utilisé pour écrire des pages Web. Le langage HTML est un langage de balisage, ce qui signifie qu'il permet d'ajouter des "balises" à un document de texte afin de structurer une page Web. A l'origine, HTML a été développé pour définir la structure d'une page Web (titres, paragraphes, ...). Il permet aujourd'hui de formater des pages Web en y incluant des ressources multimédias, des programmes informatiques, des boutons, etc. [14] [15].

CSS

CSS (Cascading Style Sheets) est un langage informatique conçu pour contrôler la présentation de documents HTML. Il permet notamment de définir la couleur du texte, le type de police d'écriture, l'espacement du texte, etc. [16].

5.2 Structure de l'application

Dans cette section, nous allons passer en revue la structure de l'application. Nous nous concentrerons uniquement sur les codes développés dans le cadre de notre problème de gîtage

et omettrons tous les codes (et autres documents) générés automatiquement par Ionic pour le fonctionnement de l'application.

5.2.1 Description générale

La figure 5.1 illustre la structure générale de l'application : elle montre les liens qui existent entre les différentes pages. L'utilisateur pourra, à partir de la page d'accueil, effectuer un calcul rapide sur base des Eurocodes et de la théorie développée dans ce mémoire, effectuer un calcul personnalisé, accéder à la liste des plages de sections ou afficher les informations à propos de l'application. Ces quatre fonctionnalités principales correspondent au deuxième niveau de l'arborescence illustrée à la figure 5.1.

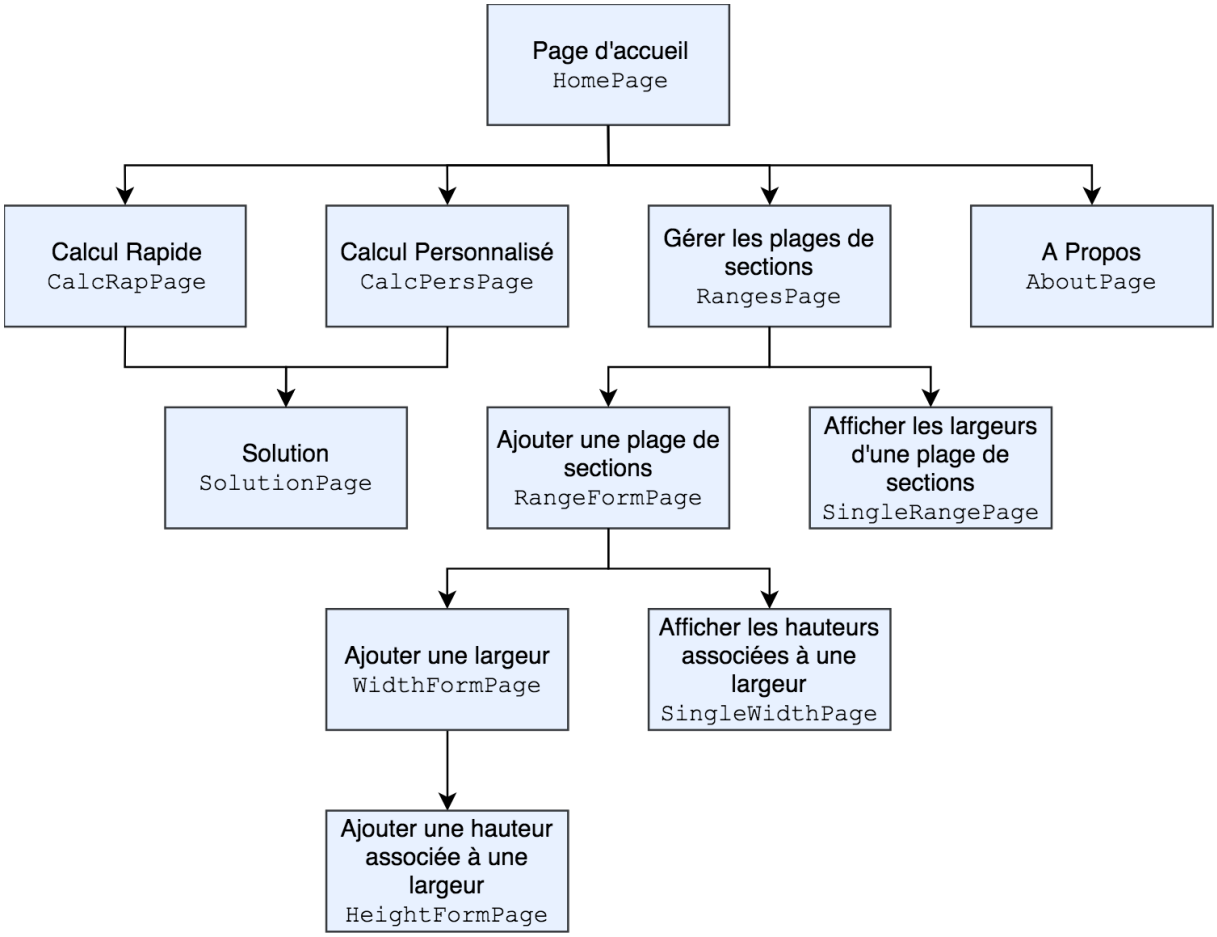


FIGURE 5.1 – Structure générale de l'application (pages)

Les codes sources des pages de l'application, toutes reprises à la figure 5.1, utilisent d'autres composants de l'application, dont des modèles et des services. En résumé, les codes qui nous intéressent sont divisés en trois catégories :

- **Les modèles :** il s'agit de classes qui nous permettent de définir nos propres types de données (objets). Nous en avons un qui permet de représenter le problème à résoudre, et deux qui permettent de représenter une plage de sections. Ils sont codés en TypeScript.
- **Les services :** il s'agit de classes qui sont injectées dans d'autres classes (en l'occurrence, les pages), afin de leur fournir un service. Dans notre cas, il s'agit de services de données

et il en existe deux : un relatif aux données des Eurocodes et un relatif aux données des plages de sections. Ils sont codés en TypeScript.

- **Les pages :** chaque page a un code HTML, qui spécifie sa mise en page et sa structure, un code TypeScript, qui permet par exemple de définir les actions à faire lorsqu'un bouton est sollicité, et un code CSS pour contrôler l'esthétique de la page.

Ci-dessous, nous allons décrire les codes appartenant à chacune de ces catégories, dans cet ordre.

5.2.2 Modèles

Les modèles sont des classes qui servent de structure de données. Ils permettent de définir nos propres types de données et de créer des objets dans les autres classes.

Paramètres du problème

Le modèle `ProblemParams` est une structure de données qui contient tous les paramètres permettant de définir entièrement le problème qu'on cherche à résoudre. Le code du fichier `ProblemParams.ts` est disponible en annexe dans le code B.1.

Le tableau 5.1 reprend les méthodes (autres que le constructeur) contenues dans le modèle `ProblemParams`, en spécifiant leur résultat, le type du résultat et son unité conventionnelle dans l'application.

Le tableau 5.2 reprend toutes les données contenues dans le modèle `ProblemParams` en spécifiant leur type, le paramètre qu'elles représentent et leur unité conventionnelle dans l'application.

Méthode	Type du résultat	Paramètre	Unité
`getCombinaisonELU()`	`number`	Q_{ELU}	$[kN/m^2]$
`getCombinaisonELSinst()`	`number`	$Q_{ELS,inst}$	$[kN/m^2]$
`getCombinaisonELSfin()`	`number`	$Q_{ELS,fin}$	$[kN/m^2]$

Tableau 5.1 – Méthodes contenues dans le modèle `ProblemParams`

Certaines données du tableau 5.2 méritent quelques précisions supplémentaires :

- `dimLP` : est un tableau contenant les deux dimensions du plancher. La plus petite des deux dimensions sera stockée à la première place de `dimLP` (à l'indice 0), tandis que la plus grande sera stockée à la seconde place de `dimLP` (à l'indice 1).
- `sensUnic` : est une chaîne de caractères qui spécifie le sens de la portée des poutres s'il est imposé.
- `cat` : est une chaîne de caractères qui spécifie le matériau (bois massif ou bois lamellé-collé).
- `inter_beam` : est un booléen qui sera vrai si l'utilisateur permet d'avoir recours à une poutre intermédiaire et faux dans le cas contraire.

Nom	Type	Paramètre	Unité
Paramètres généraux du problème			
`dimLP`	`number[]`	L et P	-
`dimLP[0]`	`number`	$\min(L; P)$	$[m]$
`dimLP[1]`	`number`	$\max(L; P)$	$[m]$
`entraxeMax`	`number`	e_{max}	$[cm]$
`ninst`	`number`	η_{inst}	$[/]$
`nfin`	`number`	η_{fin}	$[/]$
`gammaM`	`number`	γ_M	$[/]$
`kdef`	`number`	k_{def}	$[/]$
`kmod`	`number`	k_{mod}	$[/]$
`kcr`	`number`	k_{cr}	$[/]$
`ksys`	`number`	k_{sys}	$[/]$
`gammaG`	`number`	γ_G	$[/]$
`gammaQ`	`number`	γ_Q	$[/]$
`psi0`	`number`	ψ_0	$[/]$
`psi1`	`number`	ψ_1	$[/]$
`psi2`	`number`	ψ_2	$[/]$
`chargePermG`	`number`	G	$[kN/m^2]$
`chargeVarQ`	`number`	Q	$[kN/m^2]$
Contraintes architecturales éventuelles			
`hmax`	`number`	h_{max}	$[cm]$
`emin`	`number`	e_{min}	$[cm]$
`sensUnic`	`string`	Disposition	-
Paramètres des éléments principaux du gîtage			
`range_index`	`number`	Plage de section	-
`cat`	`string`	Matériau	$[/]$
`classeBois`	`string`	Classe de résistance	$[/]$
`fmk`	`number`	f_{mk}	$[MPa]$
`fvk`	`number`	f_{vk}	$[MPa]$
`modulE`	`number`	E	$[MPa]$
`modulG`	`number`	G	$[MPa]$
Paramètres de l'éventuelle poutre intermédiaire			
`inter_beam`	`boolean`	Poutre intermédiaire	-
`range_index_inter`	`number`	Plage de section	-
`cat_inter`	`string`	Matériau	-
`classeBois_inter`	`string`	Classe de résistance	$[/]$
`fmk_inter`	`number`	f_{mk}	$[MPa]$
`fvk_inter`	`number`	f_{mk}	$[MPa]$
`modulE_inter`	`number`	E	$[MPa]$
`modulG_inter`	`number`	G	$[MPa]$

TABLEAU 5.2 – Données contenues dans le modèle `ProblemParams`

Plage de sections

Une plage de sections sera représentée grâce à deux modèles. Le premier, `Width`, représente une certaine largeur b, les hauteurs h qui sont associées à cette largeur, ainsi que les coûts correspondants (par mètre linéaire, par pièce et par extrémité). Le deuxième modèle, `Range` représente une plage, qui est un ensemble de largeurs. Les codes des fichiers `Width.ts` et `Range.ts` sont disponibles en annexe dans les codes B.2 et B.3.

Le tableau 5.3 reprend les données contenues dans le modèle `Width`, tandis que le tableau 5.4 reprend les données contenues dans le modèle `Range`.

Nom	Type	Paramètre	Unité
`b`	`number`	b	$[cm]$
`h`	`number[]`	hauteurs h	$[cm]$
`m`	`number[]`	coûts par mètre	$[€/m]$
`p`	`number[]`	coûts par pièce	$[€/pce]$
`f`	`number[]`	coûts par extrémité	$[€/fixation]$

TABLEAU 5.3 – Données contenues dans le modèle `Width`

Nom	Type	Paramètre	Unité
`name`	`string`	nom de la plage	-
`largeurs`	`Width[]`	largeurs	-

TABLEAU 5.4 – Données contenues dans le modèle `Range`

Les modèles `Range` et `Width` peuvent être considérés comme des ensembles de données. Leur lien et leurs contenus sont représentés à la figure 5.2.

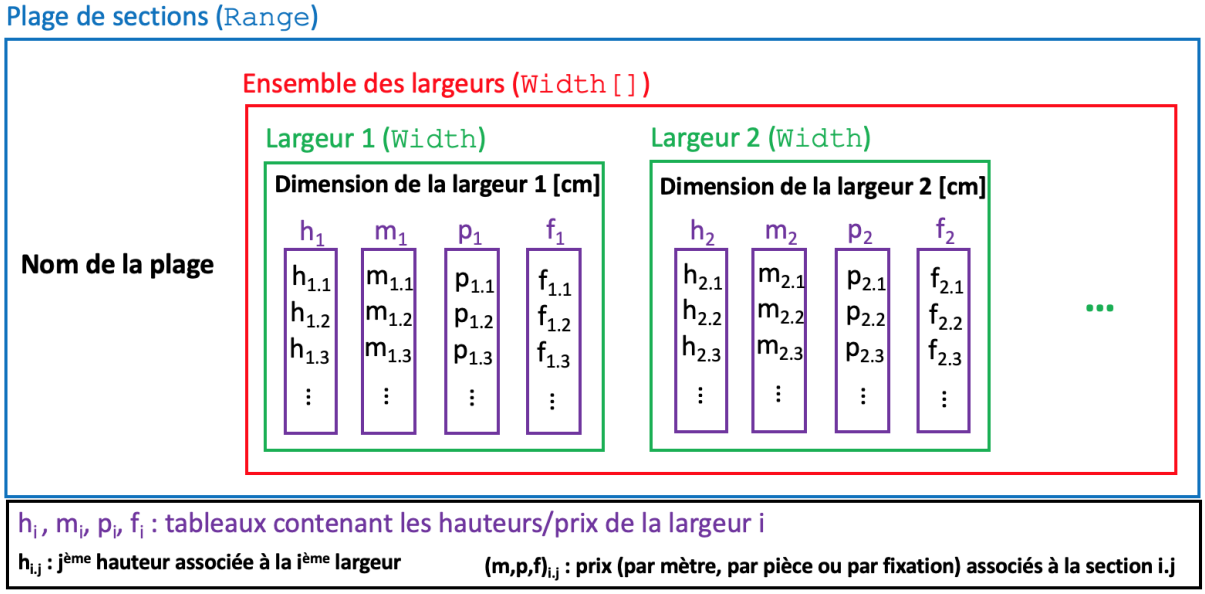


FIGURE 5.2 – Représentation des modèles `Range` et `Width` et de leur contenu

5.2.3 Services

Les services sont aussi appelés *"injectables"* ou *"fournisseurs"*. Cela nous permet de mieux décrire ce qu'est un service : il s'agit d'une classe qui peut être injectée dans une autre classe afin de lui fournir un service. Dans notre application, il s'agit de services qui fournissent des données.

Eurocodes

Le service `EurocodesService` contient les tableaux utiles des Eurocodes sous forme d'objets TypeScript, des constantes, ainsi qu'une méthode permettant d'obtenir la valeur du facteur k_h sur base des formules de l'Eurocode 5. Le code du fichier `eurocodes.service.ts` est disponible en annexe dans le code B.4.

Les tableaux 5.5, 5.6 et 5.7 reprennent respectivement les tableaux, les constantes et la méthode fournie par `EurocodesService`.

On précise que toutes les données utiles provenant des Eurocodes ne sont pas reprises dans ce service. En effet, comme nous le verrons plus tard, certaines seront directement incluses dans des formulaires afin d'alléger le code.

Nom de l'objet	Tableau(x) contenu(s) (cfr. Annexes)
`categUtil`	Tableaux A.1 et A.2
`kdef`	Tableau A.3
`kmod`	Tableau A.4
`gammaM`	Tableau A.6
`classesBois`	Tableaux A.8 et A.9

Tableau 5.5 – Données contenues dans le service `EurocodesService`

Nom	Type	Facteur	Valeur
`kcr`	`number`	k_{cr}	$0,67$
`ksys`	`number`	k_{sys}	$1,10$
`gammaG`	`number`	γ_G	$1,35$
`gammaQ`	`number`	γ_Q	$1,50$

Tableau 5.6 – Constantes contenues dans le service `EurocodesService`

Dans le tableau 5.6, les valeurs de γ_Q et γ_G sont prises conformément à l'équation 2.2, tandis que la valeur prise pour k_{sys} permet de tenir compte de l'effet système.

Méthode	Type du résultat	Facteur	Arguments
`getkh(height: number, woodtype: String)`	`number`	k_h (éq. 2.13 et 2.14)	hauteur de section matériau

Tableau 5.7 – Méthode contenue dans le service `EurocodesService`

Base de données

Le service `RangesService` fournit un service de base de données (BD) des plages de sections. Il contient deux plages de sections par défaut ainsi que toutes les méthodes utiles à la gestion de la base de données.

Ionic fournit un système de stockage de base de données qui permet de stocker des paires *(clé ; valeur)*. Pour stocker ces données sur la machine (smartphone ou tablette), Ionic utilise le système de stockage [5] qu'il considère être le meilleur en fonction de la plateforme (Android, iOS, ...) [9]. Les méthodes `get` et `set` qui permettent de lire et de remplir la base de données retournent des objets `Promise` [6].

Le code du fichier `ranges.service.ts` est disponible en annexe dans le code B.5.

Les tableaux 5.8 et 5.9 décrivent les méthodes fournies par `RangesService`.

`resetDefaultRanges()`	
Type du résultat	`Promise`
Arguments	-
Description	Réinitialise la BD avec les plages par défaut
`getRangesList()`	
Type du résultat	`Promise`
Arguments	-
Description	Obtient l'accès à la BD
`getRange(range_index: number)`	
Type du résultat	`Range`
Arguments	Clé de la plage
Description	Obtient une plage de sections
`getWidth(range_index: number, width_index: number)`	
Type du résultat	`Width`
Arguments	Clé de la plage Clé de la largeur
Description	Obtient une largeur d'une plage
`addRange(plage: Range)`	
Type du résultat	`Promise`
Arguments	Plage de sections
Description	Ajoute une plage de sections à la BD
`editRangeName(range_index: number, new_name: string)`	
Type du résultat	`Promise`
Arguments	Clé de la plage Nouveau nom
Description	Modifie le nom d'une plage de sections

Tableau 5.8 – Méthodes contenues dans le service `RangesService` (partie 1)

5. Par exemple : SQLite, IndexedDB, ...

6. `Promise` : *"L'objet Promise est utilisé pour réaliser des traitements de façon asynchrone. Une promesse représente une valeur qui peut être disponible maintenant, dans le futur, voire jamais."* [17]

deleteRange(range_index: number)	
Type du résultat	`Promise`
Arguments	Clé de la plage
Description	Retire une plage de sections de la BD
addWidth(range_index: number, new_largeur: Width)	
Type du résultat	`Promise`
Arguments	Clé de la plage Largeur
Description	Ajoute une largeur à une plage
deleteWidth(range_index: number, width_index: number)	
Type du résultat	`Promise`
Arguments	Clé de la plage Clé de la largeur
Description	Supprime une largeur d'une plage
addHeight(range_index: number, width_index: number, new_hmpf: number[])	
Type du résultat	`Promise`
Arguments	Clé de la plage Clé de la largeur Hauteur et coûts
Description	Ajoute une hauteur à une largeur d'une plage
deleteHeight(range_index: number, width_index: number, height_index: number)	
Type du résultat	`Promise`
Arguments	Clé de la plage Clé de la largeur Clé de la hauteur
Description	Supprime une hauteur d'une largeur d'une plage

Tableau 5.9 – Méthodes contenues dans le service `RangesService` (partie 2)

5.2.4 Pages

Les pages sont les principales composantes de notre application. Chaque page est représentée par :

1. une classe, codée en TypeScript
2. un document HTML, qui définit sa présentation à l'écran
3. un fichier CSS, qui permet d'ajuster l'apparence définie par le code HTML

La navigation entre les pages est assurée par Ionic et sa classe `NavController`, et fonctionne comme une pile. D'abord, une première page est chargée et définie comme étant la racine de l'application (via la méthode `setRoot()` de la classe `NavController`). La méthode `push()` de la classe `NavController` permet de charger une nouvelle page, qui sera placée au sommet de la pile. Enfin, la méthode `pop()` de la classe `NavController` permet de revenir vers la page précédente en éjectant la page du sommet de la pile. La méthode `push()` permet aussi de passer des arguments à la page qui va être chargée. Ces arguments seront récupérés dans la nouvelle page grâce à une autre classe d'Ionic, `NavParams`, et sa méthode `get()` [9]. On précise que, dans notre application, chaque page a un bouton permettant d'appeler la méthode `pop()` et donc de revenir vers la page précédente, sauf la page d'accueil qui est la page racine de l'application.

Ionic, toujours via sa classe `NavController`, prévoit des événements du cycle de vie d'une page, qui sont des méthodes qu'on peut implémenter, si nécessaire, dans chaque page. Lorsque

l'on voudra naviguer vers une nouvelle page, par exemple à l'aide de `push()`, Ionic appellera automatiquement ces méthodes dans un ordre bien précis qui dépend du cycle de vie de la page. Cet ordre est illustré à la figure 5.3. Certains événements ne se produisent que lors du premier chargement d'une page, c'est-à-dire lorsqu'elle est ajoutée à la pile (par exemple, avec `push()` ou `setRoot()`) ou retirée de la pile (avec `pop()`). Les autres événements seront appelés à chaque fois que la page sera affichée ou qu'elle sera quittée, même si elle est encore en cache (c'est-à-dire, dans la pile) [9]. Les événements du cycle de vie d'une page Ionic sont détaillés dans le tableau 5.10.

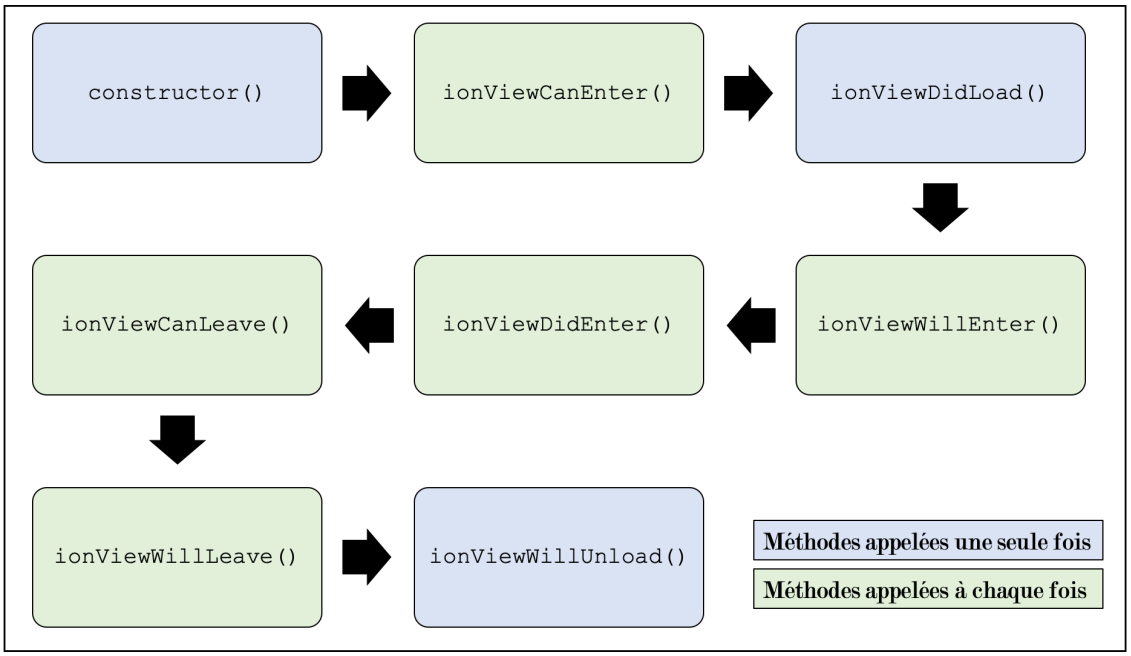


FIGURE 5.3 – Evénements du cycle de vie d'une page Ionic

Evénements	**Description**
`constructor()`	Constructeur de la classe de la page
`ionViewCanEnter()`	Appelé avant que la page ne soit active Permet de vérifier que l'accès à la page est permis
`ionViewDidLoad()`	Appelé lorsque la page est chargée Permet d'initialiser le code de la page
`ionViewWillEnter()`	Appelé lorsque la page est sur le point de devenir la page active
`ionViewDidEnter()`	Appelé lorsque la page est effectivement devenue la page active
`ionViewCanLeave()`	Appelé avant que la page ne soit quittée Permet de vérifier que la sortie de la page est permise
`ionViewWillLeave()`	Appelé avant que la page ne charge Permet de vérifier que l'accès à la page est permis
`ionViewWillUnload()`	Appelé lorsque la page est sur le point d'être retirée de la pile et détruite

TABLEAU 5.10 – Description des événements du cycle de vie d'une page Ionic

Page d'accueil

Lorsque l'application démarre, la page d'accueil (`HomePage`) est la première page à être chargée et est définie comme la 'racine' de la pile de navigation. Cette page est assez simple : sa seule fonctionnalité est de naviguer vers d'autres pages. Le code source de cette page est contenu

dans les fichiers `home.ts` (TypeScript) et `home.html` (HTML), disponibles en annexe dans les codes B.6 et B.7 respectivement.

A partir de la page d'accueil, il est possible d'effectuer les actions suivantes :

- Naviguer vers la page d'information (`AboutPage`)
- Naviguer vers la page affichant la liste des plages de sections (`RangesPage`)
- Naviguer vers la page permettant d'effectuer un calcul rapide sur base des Eurocodes (`CalcRapPage`)
- Naviguer vers la page permettant d'effectuer un calcul entièrement personnalisé (`CalcPersPage`)

Page d'information à propos de l'application

La page "A propos" (`AboutPage`) est la page la plus simple de l'application. Son seul but est d'afficher les informations concernant l'application et ses auteurs, et ne contient donc que du texte. La seule action possible à partir de cette page est de revenir à la page précédente, c'est-à-dire de retirer la page de la pile de navigation. Le code source de cette page est contenu dans les fichiers `about.ts` (TypeScript) et `about.html` (HTML), disponibles en annexe dans les codes B.8 et B.9 respectivement.

Liste des plages de sections

La page "Gérer les sections" (`RangesPage`) permet d'afficher la liste des plages de sections enregistrées dans la base de données et de la modifier. Le code source de cette page est contenu dans les fichiers `ranges.ts` (TypeScript) et `ranges.html` (HTML), disponibles en annexe dans les codes B.10 et B.11 respectivement.

A partir de cette page, il est possible d'effectuer les actions suivantes :

- Naviguer vers la page affichant la liste des largeurs d'une certaine plage de sections (`SingleRangePage`)
- Naviguer vers la page contenant le formulaire qui permet d'ajouter une plage de sections à la base de données (`RangeFormPage`)
- Supprimer une certaine plage de sections de la base de données
- Réinitialiser la base de données avec les sections par défaut ; c'est-à-dire supprimer toutes les plages de sections de la base de données et y réintroduire les plages de sections prédéfinies dans l'application

Cette page implémente la méthode `ionViewWillEnter()`, afin de charger toutes les plages de sections dans sa variable `rangesList` (de type `Ranges[]`) à partir de la base de données, avant qu'elle ne devienne active. Il est logique que cela doive être fait avant que la page ne s'affiche, puisqu'elle a précisément besoin de la liste des plages de sections afin de savoir quoi afficher. De plus, il est nécessaire que cela soit fait avant chaque affichage de la page et non la première fois uniquement, comme cela aurait été le cas avec la méthode `ionViewDidLoad()`. La liste affichée restera ainsi à jour lorsqu'une plage de section sera ajoutée ou supprimée.

Les autres méthodes de la page `RangesPage` sont énumérées et décrites dans le tableau 5.11.

7. Type de résultat d'une méthode qui exécute du code mais ne retourne pas de résultat.

onLoadRange(range_index: number)	
Type du résultat	`void` [7]
Arguments	Index d'une plage de sections
Description	Navigue vers la page affichant la liste des largeurs de la plage de sections
onNewRange()	
Type du résultat	`void`
Arguments	-
Description	Navigue vers la page contenant le formulaire qui permet d'ajouter une plage de sections à la base de données
onDeleteRange(range_index: number)	
Type du résultat	`void`
Arguments	Index d'une plage de sections
Description	Supprime la plage de sections
onDeleteRangeConfirm(range_index: number, event: Event)	
Type du résultat	`void`
Arguments	Index d'une plage de sections Evénement lié à un bouton HTML
Description	Demande de confirmer la suppression ; si confirmé, appelle `onDeleteRange(range_index: number)`
onResetRanges()	
Type du résultat	`void`
Arguments	-
Description	Réinitialise la base de données des plages de sections
onResetRangesConfirm()	
Type du résultat	`void`
Arguments	-
Description	Demande de confirmer la réinitialisation ; si confirmé, appelle `onResetRanges()`

Tableau 5.11 – Méthodes contenues dans le page `RangesPage`

Liste des largeurs d'une plage de sections

La page `SingleRangePage` permet d'afficher la liste des largeurs d'une certaine plage de sections enregistrée dans la base de données et de la modifier. Le code source de cette page est contenu dans les fichiers `single-range.ts` (TypeScript) et `single-range.html` (HTML), disponibles en annexe dans les codes B.12 et B.13 respectivement.

A partir de cette page, il est possible d'effectuer les actions suivantes :

- Naviguer vers la page affichant la liste des hauteurs et coûts associés à une certaine largeur de la plage de sections affichée (`SingleWidthPage`)
- Naviguer vers la page contenant le formulaire qui permet d'ajouter une largeur à la plage de sections affichée (`WidthFormPage`)
- Supprimer une certaine largeur de la plage de sections affichée
- Modifier le nom de la plage de sections affichée

Cette page implémente la méthode `ionViewWillEnter()`. Elle permet, dans un premier temps, de récupérer l'index de la plage de sections qu'il faut afficher. Cet index a été passé en argument

à la méthode `push()` de la classe Ionic `NavController` dans la page `RangesPage`, et est récupéré dans la page `SingleRangePage` grâce à la méthode `get()` de la classe Ionic `NavParams`. Ensuite, toujours avant que la page ne devienne active, elle charge la plage de sections dans la variable `range` (de type `Ranges`) à partir de la base de données. Une fois que la page `SingleRangePage` détient ces informations, elle pourra s'afficher correctement. Il est nécessaire que tout cela soit fait avant chaque affichage de la page et non la première fois uniquement, ce qui aurait été le cas avec la méthode `ionViewDidLoad()`. La liste affichée restera ainsi à jour lorsqu'une largeur sera ajoutée ou supprimée, par exemple.

Les autres méthodes de la page `SingleRangePage` sont énumérées et décrites dans le tableau 5.12.

`onLoadWidth(width_index: number)`	
Type du résultat	`void`
Arguments	Index d'une largeur de la plage de sections
Description	Navigue vers la page affichant la liste des hauteurs et coûts associés à la largeur de la plage de sections
`onNewWidth()`	
Type du résultat	`void`
Arguments	-
Description	Navigue vers la page contenant le formulaire qui permet d'ajouter une largeur à la plage de sections
`onDeleteWidth(width_index: number)`	
Type du résultat	`void`
Arguments	Index d'une largeur de la plage de sections
Description	Supprime la largeur
`onDeleteWidthConfirm(width_index: number, event: Event)`	
Type du résultat	`void`
Arguments	Index d'une largeur de la plage de sections Evénement lié à un bouton HTML
Description	Demande de confirmer la suppression ; si confirmé, appelle `onDeleteWidth(width_index: number)`
`onShowNameEdit()`	
Type du résultat	`void`
Arguments	-
Description	Affiche le champ pour entrer un nouveau nom
`onEditName(new_name: string, event: Event)`	
Type du résultat	`void`
Arguments	Nouveau nom Evénement lié à un bouton HTML
Description	Modifie le nom de la plage affichée

Tableau 5.12 – Méthodes contenues dans le page `SingleRangePage`

Liste des hauteurs associées à une largeur d'une plage de sections

La page (`SingleWidthPage`) permet d'afficher la liste des hauteurs associées à une certaine largeur d'une plage de sections, ainsi que les coûts (par mètre linéaire, par pièce et par extrémité), et de modifier ces données. Le code source de cette page est contenu dans les fichiers `single-width.ts` (TypeScript) et `single-width.html` (HTML), disponibles en annexe dans les codes B.14 et B.15 respectivement.

A partir de cette page, il est possible d'effectuer les actions suivantes :

- Naviguer vers la page contenant le formulaire qui permet d'ajouter une hauteur et ses coûts à la largeur affichée de la plage de sections (`HeightFormPage`)
- Supprimer une certaine hauteur pour la largeur affichée de la plage de sections

Cette page implémente la méthode `ionViewWillEnter()`. Elle permet, dans un premier temps, de récupérer l'index de la plage de sections dont il faut afficher une largeur et l'index de la largeur en question. Ces index ont été passés en argument à la méthode `push()` de la classe Ionic `NavController` dans la page `SingleRangePage`, et sont récupérés dans la page `SingleWidthPage` grâce à la méthode `get()` de la classe Ionic `NavParams`. Ensuite, toujours avant que la page ne devienne active, elle charge la liste des hauteurs (et coûts) associées à la largeur à afficher dans la variable `info` (de type `Width`), et le nom de la plage courante dans la variable `range_name` (de type `string`), à partir de la base de données. Une fois que la page `SingleWidthPage` détient ces informations, elle pourra s'afficher correctement. Il est nécessaire que tout cela soit fait avant chaque affichage de la page et non la première fois uniquement, ce qui aurait été le cas avec la méthode `ionViewDidLoad`. La liste affichée restera ainsi à jour lorsqu'une hauteur sera ajoutée ou supprimée.

Les autres méthodes de la page `SingleWidthPage` sont énumérées et décrites dans le tableau 5.13.

`onNewHeight()`	
Type du résultat	`void`
Arguments	-
Description	Navigue vers la plage contenant le formulaire qui permet d'ajouter une hauteur (et coûts) à la largeur actuelle
`onDeleteHeight(height_index: number)`	
Type du résultat	`void`
Arguments	Index d'une hauteur associée à la largeur actuelle
Description	Supprime la hauteur
`onDeleteHeightConfirm(height_index: number)`	
Type du résultat	`void`
Arguments	Index d'une hauteur associée à la largeur actuelle
Description	Demande de confirmer la suppression ; si confirmé, appelle `onDeleteHeight(height_index: number)`

Tableau 5.13 – Méthodes contenues dans le page `SingleWidthPage`

Formulaire d'ajout d'une plage de sections

La page (`RangeFormPage`) est un formulaire permettant d'enregistrer une nouvelle plage de sections dans la base de données, en précisant son nom et éventuellement une ou plusieurs largeur(s) de section. Le code source de cette page est contenu dans les fichiers `range-form.ts` (TypeScript) et `range-form.html` (HTML), disponibles en annexe dans les codes B.16 et B.17 respectivement.

Ionic, via Angular, fournit des classes [8] qui permettent de créer des formulaires. L'avantage d'utiliser un formulaire est qu'il offre des fonctionnalités optimisées qui rendent facile la transmission de données de l'utilisateur vers l'application. Dans le code TypeScript du formulaire il faudra, entre autres, implémenter les méthodes `onInitForm()` et `onSubmitForm()`.

Dans la méthode `onInitForm()`, qui permet d'initialiser le formulaire, on devra créer un objet `FormGroup`, contenant lui-même un objet `FormControl` par entrée du formulaire. Chaque `FormControl` contient un identificateur (par exemple, dans notre classe `RangeFormPage`, `name` et `largeursTab`), un modèle (ici, il s'agit d'un `string` vide pour `name` et d'un objet `FormArray` pour `largeursTab`) et d'éventuels validateurs. Dans notre application, nous utilisons uniquement, dans les objets `FormControl`, le validateur `required` qui oblige l'utilisateur à entrer une valeur avant de soumettre le formulaire. D'autres restrictions peuvent être imposées sur les entrées du formulaire directement depuis le code HTML, comme par exemple le type (nombre, ...) ou une valeur minimale pour un nombre. Les objets `FormGroup` et `FormControl` ne sont pas explicitement utilisés dans le code, mais sont créés par la classe `FormBuilder` et sa méthode `group()`. Les contrôles du formulaire sont repris dans le tableau 5.14.

Nom du contrôle	**Description**	**Requis**
`name`	Nom de la nouvelle plage	oui
`largeursTab`	Tableau des largeurs de la nouvelle plage	non

Tableau 5.14 – Contrôles du formulaire de la page `RangeFormPage`

La méthode `onSubmitForm()` est lié au bouton HTML qui permet de soumettre le formulaire, mais ne s'exécute que si tous les validateurs du formulaire sont satisfaits. Nous devons donc implémenter la méthode `onSubmitForm()` pour définir ce qu'il faut faire avec les données entrées dans le formulaire. Dans le cas de `RangeFormPage`, la méthode `onSubmitForm()` effectuera, lorsqu'elle s'exécutera, les actions suivantes :

- Construire un nouvel objet de type `Range`, avec comme paramètre le nom entré dans le formulaire
- Ajouter la ou les largeur(s) entrée(s) dans le formulaire à la liste des largeurs de la plage de sections créée
- Ajouter la nouvelle plage de sections à la base de données grâce au service `RangesService`
- Enfin, quitter la page du formulaire et revenir à la page précédente en utilisant la méthode `pop()` de la classe `NavController`

Cette page implémente la méthode `ngOnInit()`. Son fonctionnement est très similaire à la méthode `ionViewDidLoad()`, sauf qu'elle est fournie par la classe `OnInit` de Angular. Comme les classes permettant la création d'un formulaire viennent aussi d'Angular, il est plus naturel d'utiliser `ngOnInit`. On aurait aussi pu utiliser `ionViewDidLoad()` sans constater de différence dans l'application. La méthode `ngOnInit` s'occupe d'obtenir l'accès à la base de données contenant les plages de sections, ce qui sera nécessaire pour ajouter une nouvelle plage de sections à cette base de données, ainsi que d'appeler la méthode `initForm()` qui créera le formulaire. Une fois que le formulaire est créé, la page `RangeFormPage` pourra s'afficher correctement.

8. Classes utiles pour la création de formulaires : `FormBuilder`, `FormGroup`, `FormArray` et `Validators`

Formulaire d'ajout d'une largeur à une plage de sections

La page (`WidthFormPage`) est un formulaire permettant d'enregistrer une nouvelle largeur dans une certaine plage de sections de la base de données, en y associant éventuellement déjà une ou plusieurs hauteur(s) de section et ses coûts. Le code source de cette page est contenu dans les fichiers `width-form.ts` (TypeScript) et `width-form.html` (HTML), disponibles en annexe dans les codes B.18 et B.19 respectivement.

Cette page est très semblable à la page `RangeFormPage` et utilisera les mêmes classes d'Ionic et d'Angular afin de créer un formulaire. Les contrôles du formulaire sont repris dans le tableau 5.15.

Nom du contrôle	Description	Requis
`new_b`	Dimension de la nouvelle largeur	oui
`hauteursTab`	Tableau des hauteurs associées à la nouvelle largeur	non
`prixMetTab`	Tableau des coûts par mètre	non
`prixPiecTab`	Tableau des coûts par pièce	non
`prixFixTab`	Tableau des coûts par extrémité	non

Tableau 5.15 – Contrôles du formulaire de la page `WidthFormPage`

La méthode `onSubmitForm()` effectuera, lorsqu'elle s'exécutera, les actions suivantes :

- Construire un nouvel objet de type `Width`, avec comme paramètre la largeur entrée dans le formulaire
- Ajouter la ou les hauteurs(s) entrée(s) dans le formulaire à la liste des hauteurs de la largeur créée
- Faire de même pour les différents coûts
- Enfin, quitter la page du formulaire et revenir à la page précédente en utilisant la méthode `pop()` de la classe `NavController`

Cette page implémente également la méthode `ngOnInit()`. Elle s'occupe ici aussi d'obtenir l'accès à la base de données contenant les plages de sections, ce qui sera nécessaire pour ajouter une nouvelle largeur à une plage de sections à cette base de données, ainsi que d'appeler la méthode `initForm()` qui créera le formulaire. Une fois que le formulaire est créé, la page `WidthFormPage` pourra s'afficher correctement.

Formulaire d'ajout d'une hauteur associée à une largeur d'une plage de sections

La page (`HeightFormPage`) est un formulaire permettant d'enregistrer une nouvelle hauteur à une certaine largeur d'une plage de sections de la base de données, ainsi que les différents coûts qui y sont associés. Le code source de cette page est contenu dans les fichiers `height-form.ts` (TypeScript) et `height-form.html` (HTML), disponibles en annexe dans les codes B.20 et B.21 respectivement.

Cette page est très semblable aux pages `RangeFormPage` et `WidthFormPage` et utilisera les mêmes classes d'Ionic et d'Angular afin de créer un formulaire. Les contrôles du formulaire sont repris dans le tableau 5.16.

Nom du contrôle	Description	Requis
`h_new`	Nouvelle hauteur	oui
`m_new`	Coût par mètre	oui
`p_new`	Coût par pièce	oui
`f_new`	Coût par extrémité	oui

Tableau 5.16 – Contrôles du formulaire de la page `HeightFormPage`

La méthode `onSubmitForm()` effectuera, lorsqu'elle s'exécutera, les actions suivantes :

- Construire un nouveau tableau de type `number[]` contenant la hauteur et les coûts
- Ajouter ce tableau à la liste des hauteurs de la largeur concernée
- Enfin, quitter la page du formulaire et revenir à la page précédente en utilisant la méthode `pop()` de la classe `NavController`

Cette page implémente également la méthode `ngOnInit()`. Elle s'occupe ici aussi d'obtenir l'accès à la base de données contenant les plages de sections, ce qui sera nécessaire pour ajouter une nouvelle hauteur, ainsi que d'appeler la méthode `initForm()` qui créera le formulaire. Une fois que le formulaire est créé, la page `HeightFormPage` pourra s'afficher correctement.

Formulaire des paramètres pour un calcul basé sur les Eurocodes

La page (`CalcRapPage`) fonctionne comme un formulaire. Il permet à l'utilisateur d'entrer les paramètres d'un problème de gîtage dont il cherche les solutions optimales en termes de volume ou de coût, et fait usage du service `EurocodesService`. Le code source de cette page est contenu dans les fichiers `calculrap.ts` (TypeScript) et `calculrap.html` (HTML), disponibles en annexe dans les codes B.22 et B.23 respectivement.

Comme il s'agit d'un formulaire, cette page est semblable aux pages `RangeFormPage`, `WidthFormPage` et `HeightFormPage` et utilisera les mêmes classes d'Ionic et d'Angular afin de créer un formulaire. Les contrôles du formulaire de `CalcRapPage` correspondent à une partie des éléments contenus dans le modèle `ProblemParams` et sont repris dans le tableau 5.17. Une partie d'entre eux est directement entrée par l'utilisateur, d'autres sont choisis dans des listes déroulantes.

La méthode `onSubmitForm()` effectuera, lorsqu'elle s'exécutera, les actions suivantes :

- Initialiser tous les éléments de l'objet `parameters` de type `ProblemParams` sur base des entrées du formulaire et du service `EurocodesService`
- Quitter la page du formulaire et naviguer vers la page `SolutionPage` en utilisant la méthode `push()` de la classe `NavController`, en passant l'objet `parameters` en argument

Contrairement aux autres pages contenant des formulaires, cette page implémente la méthode `ionViewWillEnter()` de la classe `NavController`. Elle permet de charger les plages de sections depuis la base de données et d'initialiser le formulaire en appelant la méthode `initForm()`. Comme la page reste sur la pile après l'appel de la méthode `onSubmitForm`, puisqu'elle ne fait pas appel à la méthode `pop()` comme c'était le cas dans les autres formulaires, il faut en effet que le formulaire se réinitialise lorsqu'on retourne sur la page après avoir quitté la page des solutions.

Nom du contrôle	Description	Requis
`dim1`	Une des dimensions du plancher	oui
`dim2`	L'autre dimension du plancher	oui
`classeService`	Classe de service de la structure	oui
`categUtil`	Catégorie d'utilisation du plancher	oui
`chargePermG`	Charge permanente subie par le plancher	oui
`entraxeMax`	Entraxe maximal entre les éléments	oui
`ninst`	Paramètre η_{inst}	oui
`nfin`	Paramètre η_{fin}	oui
`ksys`	Paramètre k_{sys}	oui
`range_index`	Plage de sections pour les éléments	oui
`classeBois`	Classe de résistance des éléments	oui
`range_index_inter`	Plage de sections pour la poutre intermédiaire	non
`classeBois_inter`	Classe de résistance pour la poutre intermédiaire	non
`h_max`	Hauteur maximale des poutres	non
`e_min`	Entraxe minimal entre les éléments	non
`sensUnic`	Sens imposé pour la disposition du gîtage	non

Tableau 5.17 – Contrôles du formulaire de la page `CalcRapPage`

La page `CalcRapPage` a également une méthode permettant de réinitialiser le formulaire tout en restant sur la page. Cette méthode, qui est associée à un bouton HTML, s'appelle `reInitialisation()`.

Formulaire des paramètres pour un calcul personnalisé

La page (`CalcPersPage`) est très semblable à la page `CalcRapPage`, sauf qu'elle n'utilise pas le service `EurocodesService`. En effet, l'objectif de cette page est de permettre à l'utilisateur de personnaliser entièrement les paramètres de son problème de gîtage. Le code source de cette page est contenu dans les fichiers `calculpers.ts` (TypeScript) et `calculpers.html` (HTML), disponibles en annexe dans les codes B.24 et B.25 respectivement.

Les contrôles du formulaire de `CalcPersPage` seront beaucoup plus nombreux que pour le formulaire de la page `CalcRapPage` puisqu'il faut passer en revue tous les paramètres contenus dans le modèle `ProblemParams`. Les contrôles du formulaire sont repris dans le tableau 5.18. Une partie d'entre eux est directement entrée par l'utilisateur, d'autres sont choisis dans des listes déroulantes.

La méthode `onSubmitForm()` effectuera, lorsqu'elle s'exécutera, les actions suivantes :

- Initialiser tous les éléments de l'objet `parameters` de type `ProblemParams` sur base des entrées du formulaire
- Quitter la page du formulaire et naviguer vers la page `SolutionPage` en utilisant la méthode `push()` de la classe `NavController`, en passant l'objet `parameters` en argument

Comme la page `CalcRapPage`, cette page implémente la méthode `ionViewWillEnter()` de la classe `NavController`. Elle permet de charger les plages de sections depuis la base de données et d'initialiser le formulaire en appelant la méthode `initForm()`. Comme la page reste sur la pile après l'appel de la méthode `onSubmitForm`, il faut en effet que le formulaire se réinitialise lorsqu'on retourne sur la page après avoir quitté la page des solutions.

La page `CalcPersPage` a aussi une méthode permettant de réinitialiser le formulaire tout en restant sur la page. Cette méthode, qui est associée à un bouton HTML, s'appelle `reInitialisation()`.

Nom du contrôle	Description	Requis
`dim1`	Une des dimensions du plancher	oui
`dim2`	L'autre dimension du plancher	oui
`fmk`	Résistance du bois à la flexion ($f_{m,k}$)	oui
`fvk`	Résistance du bois au cisaillement ($f_{v,k}$)	oui
`modE`	Module d'élasticité du bois (E)	oui
`modG`	Module de cisaillement du bois (G)	oui
`cat`	Matériau (bois massif ou BLC)	oui
`chargePermG`	Charge permanente subie par le plancher	oui
`chargeVarQ`	Charge d'exploitation subie par le plancher	oui
`entraxeMax`	Entraxe maximal entre les éléments	oui
`gammaG`	Coefficient γ_G	oui
`gammaQ`	Coefficient γ_Q	oui
`psi0`	Coefficient ψ_0	oui
`psi1`	Coefficient ψ_1	oui
`psi2`	Coefficient ψ_2	oui
`ninst`	Paramètre η_{inst}	oui
`nfin`	Paramètre η_{fin}	oui
`gammaM`	Coefficient γ_M	oui
`kdef`	Paramètre k_{def}	oui
`kmod`	Paramètre k_{mod}	oui
`kcr`	Paramètre k_{cr}	oui
`ksys`	Paramètre k_{sys}	oui
`range_index`	Plage de sections pour les éléments	oui
`fmk_inter`	Résistance de la poutre intermédiaire à la flexion ($f_{m,k}$)	non
`fvk_inter`	Résistance de la poutre intermédiaire au cisaillement ($f_{v,k}$)	non
`modE_inter`	Module d'élasticité de la poutre intermédiaire (E)	non
`modG_inter`	Module de cisaillement de la poutre intermédiaire (G)	non
`cat_inter`	Matériau de la poutre intermédiaire (bois massif ou BLC)	non
`range_index_inter`	Plage de sections pour la poutre intermédiaire	non
`h_max`	Hauteur maximale des poutres	non
`e_min`	Entraxe minimal entre les éléments	non
`sensUnic`	Sens imposé pour la disposition du gîtage	non

Tableau 5.18 – Contrôles du formulaire de la page `CalcPersPage`

Page des solutions optimales en termes de volume et de coût

La page `SolutionPage` permet d'afficher la solution qui minimise le volume de bois à mettre en œuvre, la solution qui minimise le coût total du gîtage et la liste des paramètres utilisés pour les calculs. Le code source de cette page est contenu dans les fichiers `solution.ts` (TypeScript) et `solution.html` (HTML), disponibles en annexe dans les codes B.26 et B.27 respectivement.

Cette page n'a pour objectif que l'affichage de la solution et la seule action possible à partir de cette page est de revenir à la page précédente, c'est-à-dire de retirer la page de la pile de navigation.

La page `SolutionPage` contient l'algorithme de calcul de la solution, qui est détaillé dans la section suivante. Comme nous l'avons fait pour les autres pages, nous n'abordons ici que la structure générale de la page.

Le constructeur de la page va, au moment de charger la page, récupérer les paramètres du problème qui, pour rappel, lui ont été transmis par les pages `CalcRapPage` ou `CalcPersPage`. Pour ce faire, on utilise la méthode `get()` de la classe Ionic `NavParams` et on attribue les valeurs utiles aux variables globales de la page `SolutionPage`.

Cette page implémente la méthode `ionViewWillEnter()`. Elle permet, dans un premier temps, de récupérer la plage de sections (ou les plages de sections, s'il faut considérer une poutre intermédiaire) à utiliser pour calculer la solution. Les index de ces plages ont déjà été récupérés grâce au constructeur de la page. Ensuite, toujours avant que la page ne devienne active, elle doit déterminer la solution optimale en termes de volume et la solution optimale en termes de coût. Une fois que la page `SolutionPage` détient ces solutions, elle pourra s'afficher correctement.

Toutes les méthodes de la page `SolutionPage` sont détaillées dans la section suivante.

5.3 Algorithme de calcul

Dans cette section, nous détaillons l'algorithme déterminant la solution optimale en termes de volume et celle en termes de coût. Cet algorithme fait partie du code de la page `SolutionPage`, contenu dans le fichier `SolutionPage.ts`, qui est disponible en annexe au code B.26. Comme la structure de la page `SolutionPage` a déjà été abordée dans la section précédente, nous nous concentrons ici uniquement sur l'algorithme de calcul en lui-même. L'entièreté du processus de calcul est effectué durant l'appel à la méthode `ionViewWillEnter` puisque la page a besoin de la solution finale pour pouvoir s'afficher.

Les étapes importantes du cycle de vie de la page `SolutionPage` sont illustrées à la figure 5.4.

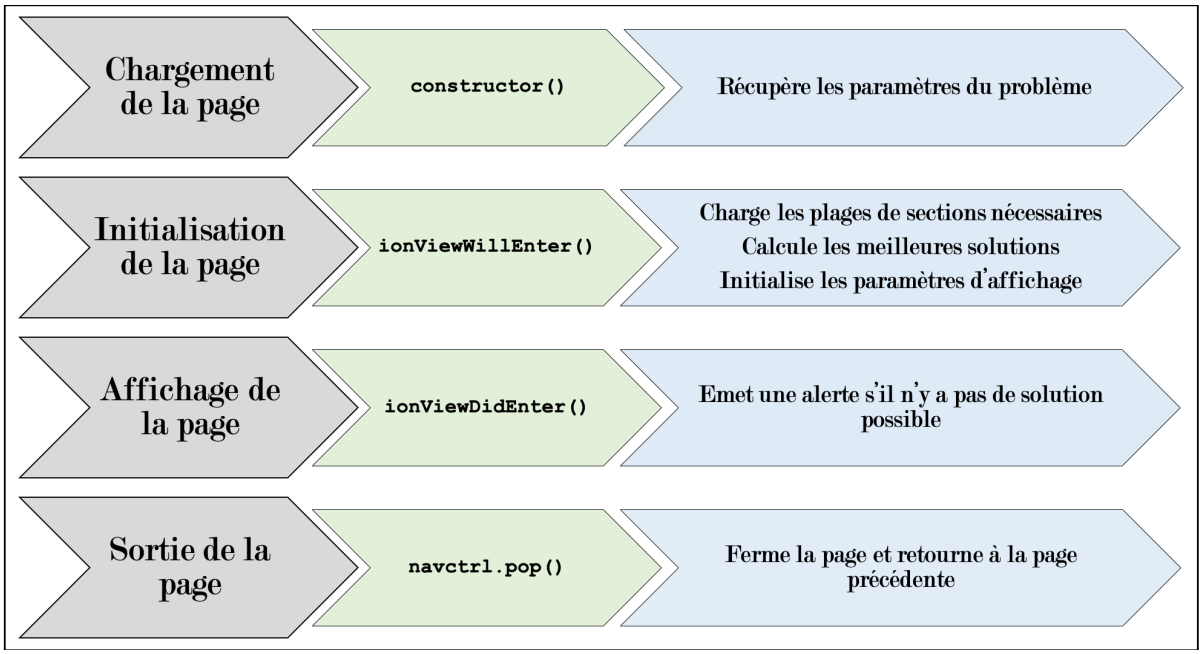


FIGURE 5.4 – Cycle de vie de la page `SolutionPage`

5.3.1 Variables globales et méthodes de la page `SolutionPage`

Les variables globales de la page `SolutionPage` sont énumérées et décrites dans les tableaux 5.19 et 5.20, tandis que les méthodes de la page `SolutionPage` (autres que le constructeur) le sont dans les tableaux 5.22, 5.23 et 5.24. Nous ferons de nombreuses références à ces variables et à ces méthodes dans la suite de cette section dédiée à l'algorithme.

Nom	**Type**	**Description**
`parameters`	`ProblemParams`	Contient tous les paramètres du problème
`range_index`	`number`	Contient l'index de la plage de sections à utiliser pour les éléments standards
`range`	`Range`	Contient la plage de sections à utiliser pour les éléments standards
`inter_beam`	`boolean`	Vaut `true` s'il faut considérer une poutre intermédiaire et `false` dans le cas contraire
`range_index_inter`	`number`	Contient l'index de la plage de sections à utiliser pour l'éventuelle poutre intermédiaire
`range_inter`	`Range`	Contient la plage de sections à utiliser pour la poutre intermédiaire
`range_inter_name`	`string`	Contient le nom de la plage de sections à utiliser pour la poutre intermédiaire
`if_sensUnic`	`boolean`	Indique si le sens de la portée des éléments est imposé ou non
`inter_beam_display_V`	`boolean`	Permet de gérer l'affichage de la solution en fonction de la présence ou non d'une poutre intermédiaire pour la solution en termes de volume
`inter_beam_display_C`	`boolean`	Permet de gérer l'affichage de la solution en fonction de la présence ou non d'une poutre intermédiaire pour la solution en termes de coût
`volumeSelected`	`boolean`	Permet d'afficher/masquer la solution optimale en termes de volume
`costSelected`	`boolean`	Permet d'afficher/masquer la solution optimale en termes de coût
`paramSelected`	`boolean`	Permet d'afficher/masquer les paramètres utilisés pour les calculs

Tableau 5.19 – Variables globales de la page `SolutionPage` (partie 1)

Nom	Type	Description
`volArrayLong`	Tuple [9]	Contient la solution minimisant le volume dans le cas d'une disposition dans le sens long sans poutre intermédiaire
`volArrayCourt`	Tuple	Contient la solution minimisant le volume dans le cas d'une disposition dans le sens court sans poutre intermédiaire
`volArrayLong_inter`	Tuple	Contient la solution minimisant le volume dans le cas d'une disposition dans le sens long avec une poutre intermédiaire
`volArrayCourt_inter`	Tuple	Contient la solution minimisant le volume dans le cas d'une disposition dans le sens court avec une poutre intermédiaire
`costArrayLong`	Tuple	Contient la solution minimisant le coût dans le cas d'une disposition dans le sens long sans poutre intermédiaire
`costArrayCourt`	Tuple	Contient la solution minimisant le coût dans le cas d'une disposition dans le sens court sans poutre intermédiaire
`costArrayLong_inter`	Tuple	Contient la solution minimisant le coût dans le cas d'une disposition dans le sens long avec une poutre intermédiaire
`costArrayCourt_inter`	Tuple	Contient la solution minimisant le coût dans le cas d'une disposition dans le sens court avec une poutre intermédiaire
`resultsVolArray`	Tuple	Contient la solution minimisant le volume
`resultsCostArray`	Tuple	Contient la solution minimisant le coût

Tableau 5.20 – Variables globales de la page `SolutionPage` (partie 2)

Chaque méthode permettant de calculer la solution optimale pour une certaine disposition retournera cette solution dans un tuple qui pourra être attribué à une des variables du tableau 5.20. Ces tuples ne peuvent être des tableaux ordinaires, puisque leurs entrées mélangeront des données de type `number` et des données de type `string`. Si une méthode ne trouve aucune solution possible, elle retournera un tuple dont toutes les entrées seront nulles.

Les données contenues dans ces tuples sont reprises, avec leur index dans le tuple, dans le tableau 5.21. Les tuples contenant une solution pour une disposition simple auront simplement leur deux dernières entrées valant 0.

Index	Type	Contenu
0	`string`	Sens de la portée
1	`number`	Volume de bois total
2	`number`	Coût total
3	`number`	Largeur des éléments standards
4	`number`	Hauteur des éléments standards
5	`number`	Nombre d'éléments standards
6	`number`	Entraxe entre les éléments standards
7	`number`	Largeur de l'éventuelle poutre intermédiaire
8	`number`	Hauteur de l'éventuelle poutre intermédiaire

Tableau 5.21 – Tuples contenant une solution

9. Tuple : comparable à un tableau (`Array`), sauf que, à l'inverse d'un tableau, un tuple permet de rassembler des variables de types différents. Dans notre application, ils seront du type `(string|number)[]`

ionViewWillEnter()	
Type du résultat	`void`
Arguments	-
Description	Appelée avant que la page ne soit active, mais après qu'elle soit chargée ; Charge les plages de sections nécessaires depuis la base de données ; Calcule les solutions optimales ; Initialise les paramètres d'affichage utilisés dans le document HTML
ionViewDidEnter()	
Type du résultat	`void`
Arguments	-
Description	Affiche une alerte si aucune solution n'existe pour les paramètres donnés
indexOfMinimum(arr: Number[])	
Type du résultat	`number`
Arguments	Tableau de nombres
Description	Donne l'index du plus petit nombre non nul du tableau `arr`
clone(paramset)	
Type du résultat	Même type que celui de l'argument
Arguments	Objet de type quelconque
Description	Retourne une copie de l'objet passé en argument
showHideVolume()	
Type du résultat	`void`
Arguments	-
Description	Est lié à un bouton HTML et permet d'afficher/masquer la solution optimale en termes de volume
showHideCost()	
Type du résultat	`void`
Arguments	-
Description	Est lié à un bouton HTML et permet d'afficher/masquer la solution optimale en termes de coût
showHideParam()	
Type du résultat	`void`
Arguments	-
Description	Est lié à un bouton HTML et permet d'afficher/masquer les paramètres utilisés pour les calculs
isSensCourt(resultArray: (string|number)[])	
Type du résultat	boolean
Arguments	Tuple de type `(string|number)[]`
Description	Permet de savoir si un tuple contenant une solution correspond à une disposition des poutres dans le sens court
isSensLong(resultArray: (string|number)[])	
Type du résultat	boolean
Arguments	Tuple de type `(string|number)[]`
Description	Permet de savoir si un tuple contenant une solution correspond à une disposition des poutres dans le sens long

Tableau 5.22 – Méthodes contenues dans le page `SolutionPage` (partie 1)

`calculOptimalVolume(paramset: ProblemParams, largeurs: Width[], sens: string)`	
Type du résultat	Tuple de type `(string|number)[]`
Arguments	Paramètres du problème Liste de largeurs Le sens de la portée des poutres
Description	Calcule la solution optimale en termes de volume pour un gîtage sans poutre intermédiaire et dont les poutres sont placées dans un certain sens
`calculOptimalCost(paramset: ProblemParams, largeurs: Width[], sens: string)`	
Type du résultat	Tuple de type `(string|number)[]`
Arguments	Paramètres du problème Liste de largeurs Le sens de la portée des poutres
Description	Calcule la solution optimale en termes de coût pour un gîtage sans poutre intermédiaire et dont les poutres sont placées dans un certain sens
`calculOptimalVolume_Inter(paramset: ProblemParams, largeurs: Width[], largeurs_inter: Width[], sens_inter: string)`	
Type du résultat	Tuple de type `(string|number)[]`
Arguments	Paramètres du problème Liste de largeurs pour les éléments standards Liste de largeurs pour la poutre intermédiaire Le sens de la portée de la poutre intermédiaire
Description	Calcule la solution optimale en termes de volume pour un gîtage avec une poutre intermédiaire placée dans un certain sens
`calculOptimalCost_Inter(paramset: ProblemParams, largeurs: Width[], largeurs_inter: Width[], sens_inter: string)`	
Type du résultat	Tuple de type `(string|number)[]`
Arguments	Paramètres du problème Liste de largeurs pour les éléments standards Liste de largeurs pour la poutre intermédiaire Le sens de la portée de la poutre intermédiaire
Description	Calcule la solution optimale en termes de coût pour un gîtage avec une poutre intermédiaire placée dans un certain sens

TABLEAU 5.23 – Méthodes contenues dans le page `SolutionPage` (partie 2)

`calculOptimalVolumeBIS(paramset_bis: ProblemParams, largeurs: Width[], sens: string, dimLbis: number, dimPbis: number)`	
Type du résultat	Tuple de type `(string|number)[]`
Arguments	Paramètres du problème Liste de largeurs Le sens de la portée des éléments Les dimensions du plancher
Description	Calcule la solution optimale en termes de volume pour un gîtage sans poutre intermédiaire et dont les poutres sont placées dans un certain sens pour un plancher de dimension donnée
`calculOptimalCostBIS(paramset_bis: ProblemParams, largeurs: Width[], sens: string, dimLbis: number, dimPbis: number)`	
Type du résultat	Tuple de type `(string|number)[]`
Arguments	Paramètres du problème Liste de largeurs Le sens de la portée des éléments Les dimensions du plancher
Description	Calcule la solution optimale en termes de coût pour un gîtage sans poutre intermédiaire et dont les poutres sont placées dans un certain sens pour un plancher de dimension donnée

Tableau 5.24 – Méthodes contenues dans le page `SolutionPage` (partie 3)

5.3.2 Choix de la solution optimale

L'entièreté du calcul se fait lors de l'exécution de la méthode `ionViewWillEnter()`. Un schéma de l'exécution de `ionViewWillEnter()` (et `ionViewDidEnter()`) est fourni à la figure 5.5. Ci-dessous, nous allons analyser les différentes étapes illustrées dans ce schéma (cfr. numéros des cadres).

L'étape (1) du schéma représenté à la figure 5.5 consiste à récupérer la ou les plage(s) de sections nécessaire(s) aux calculs. L'extrait de code qui correspond à cette étape est disponible au code 5.1.

On fait d'abord appel à la méthode `getRangesList()` du service `rangesService`, qui retourne la liste de toutes les plages de sections sous forme de `Promise`. On associe ce résultat à une nouvelle variable locale `ranges` de type `Range[]`. Ensuite, on associe la plage de sections qui doit être utilisée pour les éléments standards à la variable globale `range`, grâce à une autre variable globale, `range_index`.

Le premier `if` sert simplement à s'assurer que `ranges` a bien pu être associé à la `Promise` avant de l'utiliser. Le deuxième `if` teste la valeur de la variable globale `inter_beam`, qui vaudra `true` si l'utilisateur a permis l'utilisation d'une poutre intermédiaire. Si c'est le cas, on associe la plage de sections à utiliser pour l'éventuelle poutre intermédiaire à la variable globale `range_inter` en utilisant la variable globale `range_index_inter`.

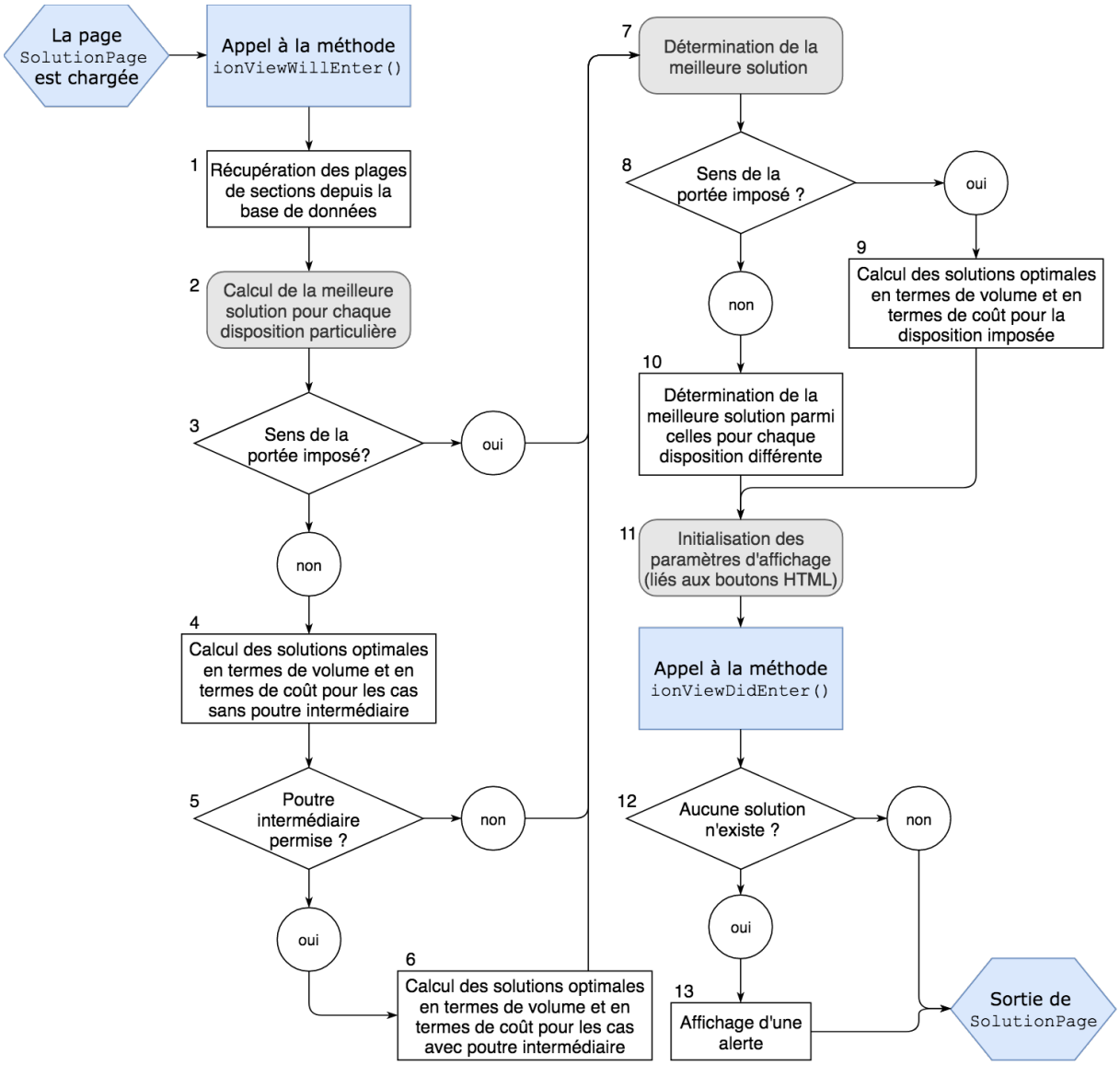


FIGURE 5.5 – Exécution des méthodes `ionViewWillEnter()` et `ionViewDidEnter()` de la page `SolutionPage`

```
1 this.rangesService.getRangesList().then((ranges: Range[]) => {
2   if (ranges) {
3     this.range = ranges[this.range_index];
4     if (this.inter_beam) {
5       this.range_inter = ranges[this.range_index_inter];
6     }
7   }
8 });
```

CODE 5.1 – Extrait du code de `SolutionPage` correspondant à l'étape 1 de la figure 5.5

L'étape (2) consiste à calculer la solution optimale en termes de coût et la solution optimale en termes de volume pour chaque disposition des poutres prise en compte. L'extrait de code qui correspond à cette étape est disponible au code 5.2.

Le premier `if` teste la valeur de la variable globale `if_sensUnic`, qui vaudra `true` si l'utilisateur a imposé un certain sens pour la portée des poutres. Cela correspond à l'étape (3) du

schéma de la figure 5.5, et à la ligne 1 du code 5.2. Si un certain sens est imposé, comme le `if` teste la valeur inverse de `if_sensUnic` (avec l'opérateur `"!"`), le code situé à l'intérieur de ce `if` ne sera pas exécuté et on passera directement à l'étape (7) du schéma représenté à la figure 5.5, dont le code correspondant est repris au code 5.3.

En revanche, si le sens n'est pas imposé, on passe à l'étape (4) du schéma représenté à la figure 5.5, qui correspond aux lignes 2 à 5 du code 5.2. Dans celle-ci, on fera deux appels à la méthode `calculOptimalVolume()` et deux appels à la méthode `calculOptimalCost()`, pour tenir compte des deux sens possibles. On associe les résultats aux variables globales `volArrayCourt`, `volArrayLong`, `costArrayCourt` et `costArrayLong`.

Un second `if`, imbriqué dans le premier et correspondant à l'étape (5) du schéma représenté à la figure 5.5, permet de tester la valeur de la variable globale `inter_beam`, qui vaudra `true` si l'utilisateur a permis l'utilisation d'une poutre intermédiaire. Si c'est le cas, on passe à l'étape (6), qui correspond aux lignes 7 à 10 du code 5.2. On y fera deux appels à la méthode `calculOptimalVolume_Inter()` et deux appels à la méthode `calculOptimalCost_Inter()`, pour tenir compte des deux sens possibles. On associe les résultats aux variables globales `volArrayLong_inter`, `volArrayCourt_inter`, `costArrayLong_inter` et `costArrayCourt_inter`.

Il est important de noter que le deuxième `if` du code 5.2 peut être imbriqué dans le premier puisqu'un utilisateur ne peut pas à la fois imposer un sens pour la portée des poutres et permettre le recours à une poutre intermédiaire. En effet, si on a recours à une poutre intermédiaire, alors il y aura de toute façon au moins une poutre dans chaque sens.

```
1  if (!this.if_sensUnic) {
2    this.volArrayCourt = this.calculOptimalVolume(this.parameters, this.range.
       largeurs, 'court');
3    this.volArrayLong = this.calculOptimalVolume(this.parameters, this.range.
       largeurs, 'long');
4    this.costArrayCourt = this.calculOptimalCost(this.parameters, this.range.
       largeurs, 'court');
5    this.costArrayLong = this.calculOptimalCost(this.parameters, this.range.
       largeurs, 'long');
6    if (this.inter_beam) {
7      this.volArrayLong_inter = this.calculOptimalVolume_Inter(this.parameters,
       this.range.largeurs, this.range_inter.largeurs, 'long');
8      this.volArrayCourt_inter = this.calculOptimalVolume_Inter(this.parameters,
       this.range.largeurs, this.range_inter.largeurs, 'court');
9      this.costArrayLong_inter = this.calculOptimalCost_Inter(this.parameters,
       this.range.largeurs, this.range_inter.largeurs, 'long');
10     this.costArrayCourt_inter = this.calculOptimalCost_Inter(this.parameters,
       this.range.largeurs, this.range_inter.largeurs, 'court');
11   }
12 }
```

Code 5.2 – Extrait du code de `SolutionPage` correspondant à l'étape 2 de la figure 5.5

L'étape (7) du schéma représenté à la figure 5.5 consiste à sélectionner la solution optimale en termes de volume et la solution optimale en termes de coût parmi les solutions calculées à l'étape (2), et de les placer dans les variables globales `resultsVolArray` et `resultsCostArray` respectivement. L'extrait de code qui correspond à cette étape est disponible au code 5.3.

Le premier `if`, qui correspond à l'étape (8) du schéma représenté à la figure 5.5 et à la ligne 1 du code 5.3, teste la valeur de la variable globale `if_sensUnic`, qui vaudra `true` si l'utilisateur a imposé un certain sens pour la portée des poutres. Si c'est le cas, cela veut dire qu'il ne faut calculer la solution optimale en termes de volume et celle en termes de coût que pour une seule disposition et on associe directement aux variables globales `resultsVolArray` et `resultsCostArray` les résultats des méthodes `calculOptimalVolume()` et `calculOptimalCost()` respectivement. Cela

correspond à l'étape (9) du schéma de la figure 5.5 et aux lignes 2 et 3 du code 5.3.

Si ce n'est pas le cas, alors il faudra déterminer laquelle des solutions calculées aux étapes (4) et, éventuellement, (6) est la meilleure solution. Ce procédé correspond à l'étape (10) du schéma représenté à la figure 5.5. Afin de comparer efficacement les volumes de chaque solution optimale en termes de volume et les coûts de chaque solution optimale en termes de coût, on crée deux variables globales `volArray` et `costArray`, aux lignes 6 et 7 du code 5.3, dans lesquelles on copiera par la suite uniquement les volumes et les coûts de chaque solution respectivement.

Après cela, on teste la valeur de `inter_beam`, à la ligne 8 du code 5.3. Si elle vaut `true`, c'est qu'on a calculé, lors de l'étape (2), les solutions optimales pour les cas avec poutre intermédiaire. Les lignes 9 à 46 du code 5.3 sont alors exécutées. Sinon, ce sont les lignes 50 à 70 du code 5.3 qui seront exécutées. Dans les deux cas, le procédé est le même. Nous l'expliquons ci-dessous pour le cas où `inter_beam` vaut `false`. L'adaptation au cas où `inter_beam` vaut `true` se fait simplement en prenant également en compte les solutions `volArrayCourt_inter` et `costArrayCourt_inter`, calculées à l'étape (2).

Dans le cas où `inter_beam` vaut `false`, on place d'abord les volumes des solutions `volArrayCourt` et `volArrayLong` dans le tableau `volArray` et les coûts des solutions `costArrayCourt` et `costArrayLong` dans le tableau `costArray` (lignes 50 et 51 du code 5.3). Les lignes 52 à 60 du code 5.3 vont permettre de déterminer la solution finale optimale en termes de volume, qui sera placée dans la variable globale `resultsVolArray`, et les lignes 62 à 70 du code 5.3 vont permettre de déterminer la solution finale optimale en termes de coût, qui sera placée dans la variable globale `resultsCostArray`.

Afin de tester si une entrée du tableau `volArray` (ou `costArray`) est la valeur minimum (strictement positive) de ce tableau, on utilise la méthode `indexOfMinimum()` (cfr. ligne 52 du code 5.3, par exemple). Il est important que la méthode `indexOfMinimum()` ne tienne pas compte des valeurs nulles, puisque cela correspondrait à un cas où aucune solution n'existe. Cependant, si toutes les entrées du tableau `volArray` (ou `costArray`) sont nulles, la méthode `indexOfMinimum()` retournera quand même l'index 0. Il est donc nécessaire de vérifier que, si la méthode `indexOfMinimum()` retourne l'index 0, l'entrée du tableau `volArray` (ou `costArray`) correspondant à cet index n'est pas nulle. Par prudence et pour rendre le code homogène, on effectuera aussi cette vérification dans les cas où la méthode `indexOfMinimum()` retournerait une autre valeur que 0.

Si ni les conditions à la ligne 52 du code 5.3 ni celles à la ligne 56 du code 5.3 ne sont respectées, cela signifie que toutes les entrées de `volArray` sont nulles et qu'il n'existe donc aucune solution pour le plancher et les paramètres donnés. Afin d'indiquer cela, on remplit le tableau `resultsVolArray` de zéros.

Lorsque les tableaux `resultsVolArray` et `resultsCostArray` sont remplis, on peut passer à l'étape (11) du schéma représenté à la figure 5.5. Lors de cette étape, les variables globales `volumeSelected`, `costSelected` et `paramSelected` seront initialisées à `false`. Ces variables sont utiles au code HTML de la page `SolutionPage` et sont liées aux boutons qui permettent d'afficher/masquer les solutions ou les paramètres.

```
1 if (this.if_sensUnic) {
2   this.resultsVolArray = this.calculOptimalVolume(this.parameters, this.range.
      largeurs, this.parameters.sensUnic);
3   this.resultsCostArray = this.calculOptimalCost(this.parameters, this.range.
      largeurs, this.parameters.sensUnic);
4 }
5 else {
6   let volArray: number[];
7   let costArray: number[];
```

```
8   if (this.inter_beam) {
9     volArray = [Number(this.volArrayCourt[1]), Number(this.volArrayLong[1]),
      Number(this.volArrayCourt_inter[1]), Number(this.volArrayLong_inter[1])];
10    costArray = [Number(this.costArrayCourt[2]), Number(this.costArrayLong[2]),
      Number(this.costArrayCourt_inter[2]), Number(this.costArrayLong_inter[2])];
11
12    if (this.indexOfMinimum(volArray)==0 && volArray[0]!=0){
13      this.resultsVolArray = this.volArrayCourt;
14      this.inter_beam_display_V = false;
15    }
16    else if (this.indexOfMinimum(volArray)==1 && volArray[1]!=0){
17      this.resultsVolArray = this.volArrayLong;
18      this.inter_beam_display_V = false;
19    }
20    else if (this.indexOfMinimum(volArray)==2 && volArray[2]!=0){
21      this.resultsVolArray = this.volArrayCourt_inter;
22      this.inter_beam_display_V = true;
23    }
24    else if (this.indexOfMinimum(volArray)==3 && volArray[3]!=0){
25      this.resultsVolArray = this.volArrayLong_inter;
26      this.inter_beam_display_V = true;
27    }
28    else {this.resultsVolArray = [0, 0, 0, 0, 0, 0, 0, 0, 0];}
29
30    if (this.indexOfMinimum(costArray)==0 && this.costArrayCourt[1]!=0){
31      this.resultsCostArray = this.costArrayCourt;
32      this.inter_beam_display_C = false;
33    }
34    else if (this.indexOfMinimum(costArray)==1 && this.costArrayLong[1]!=0){
35      this.resultsCostArray = this.costArrayLong;
36      this.inter_beam_display_C = false;
37    }
38    else if (this.indexOfMinimum(costArray)==2 && this.costArrayCourt_inter[1]!
      =0){
39      this.resultsCostArray = this.costArrayCourt_inter;
40      this.inter_beam_display_C = true;
41    }
42    else if (this.indexOfMinimum(costArray)==3 && this.costArrayLong_inter[1]!=0
      ){
43      this.resultsCostArray = this.costArrayLong_inter;
44      this.inter_beam_display_C = true;
45    }
46    else {this.resultsCostArray = [0, 0, 0, 0, 0, 0, 0, 0, 0];}
47  }
48
49  else {
50    volArray = [Number(this.volArrayCourt[1]), Number(this.volArrayLong[1])];
51    costArray = [Number(this.costArrayCourt[2]), Number(this.costArrayLong[2])];
52    if (this.indexOfMinimum(volArray)==0){
53      this.resultsVolArray = this.volArrayCourt;
54      this.inter_beam_display_V = false;
55    }
56    else if (this.indexOfMinimum(volArray)==1){
57      this.resultsVolArray = this.volArrayLong;
58      this.inter_beam_display_V = false;
59    }
60    else {this.resultsVolArray = [0, 0, 0, 0, 0, 0, 0, 0, 0];}
61
62    if (this.indexOfMinimum(costArray)==0 && this.costArrayCourt[1]!==0){
63      this.resultsCostArray = this.costArrayCourt;
64      this.inter_beam_display_C = false;
65    }
66    else if (this.indexOfMinimum(costArray)==1 && this.costArrayLong[1]!==0){
```

```
67        this.resultsCostArray = this.costArrayLong;
68        this.inter_beam_display_C = false;
69      }
70      else {this.resultsCostArray = [0, 0, 0, 0, 0, 0, 0, 0, 0];}
71    }
72  }
```

Code 5.3 – Extrait du code de `SolutionPage` correspondant à l'étape 7 de la figure 5.5

Après l'étape (11), on quitte la méthode `ionViewWillEnter()` et on entre dans la méthode `ionViewDidEnter()`. Ici, on vérifie si la solution existe et, si ce n'est pas le cas, on affiche une alerte l'indiquant à l'utilisateur.

La vérification correspond à l'étape (12) du schéma représenté à la figure 5.5 et à la ligne 2 du code 5.4. Si la valeur contenue à l'index correspondant au volume du tableau `resultsVolArray` (ou `resultsCostArray`) est nulle, cela signifie qu'aucune solution n'existe. Dans ce cas, on passe à l'étape (13), qui consiste à créer et présenter une alerte. L'étape (13) correspond aux lignes 3 à 8 du code 5.4. On y crée, à la ligne 3 du code 5.4, une alerte grâce à la fonction `create()` de la classe `AlertController` fournie par Ionic [9]. Les lignes 4 à 6 du code 5.4 permettent de définir le texte qui sera affiché dans les différents champs de l'alerte. Enfin, la ligne 8 du code 5.4 s'occupe de présenter l'alerte à l'écran via la méthode `present()` de la classe `AlertController`.

```
1  ionViewDidEnter() {
2    if (this.resultsVolArray && (this.resultsVolArray[1]==0 || this.
       resultsCostArray[1]==0)){
3      let alert = this.alertCtrl.create({
4        title: 'Erreur',
5        subTitle: 'Pas de solution possible: vérifiez les paramètres.',
6        buttons: ['OK']
7      });
8      alert.present();
9    }
10 }
```

Code 5.4 – Extrait du code de `SolutionPage` correspondant aux étapes 12 et 13 de la figure 5.5

5.3.3 Solution la moins volumineuse

Le calcul de la solution optimale en termes de volume, pour une disposition simple du gîtage et dans un sens donné, se fait en exécutant la méthode `calculOptimalVolume()`, dont les arguments sont, pour rappel, donnés dans le tableau 5.23. Un schéma de l'exécution de `calculOptimalVolume()` est fourni à la figure 5.6. Ci-dessous, nous allons analyser les différentes étapes illustrées dans ce schéma.

Lorsque la méthode `calculOptimalVolume()` est appelée, la première chose qu'elle va faire (étape (1) du schéma représenté à la figure 5.6) est de créer des variables locales contenant les éléments d'une solution et de les initialiser à 0. Ces variables sont :

- `optiVol`, qui contient le volume total de bois à mettre en œuvre (entrée 1 du tuple de solution)
- `cost`, qui contient le coût total du gîtage (entrée 2 du tuple de solution)
- `optiWidth`, qui contient la largeur des éléments à utiliser (entrée 3 du tuple de solution)
- `optiHeight`, qui contient la hauteur des éléments à utiliser (entrée 4 du tuple de solution)
- `optiNumber`, qui contient le nombre d'éléments à utiliser (entrée 5 du tuple de solution)
- `entraxe`, qui contient l'entraxe à prévoir entre les éléments (entrée 6 du tuple de solution)

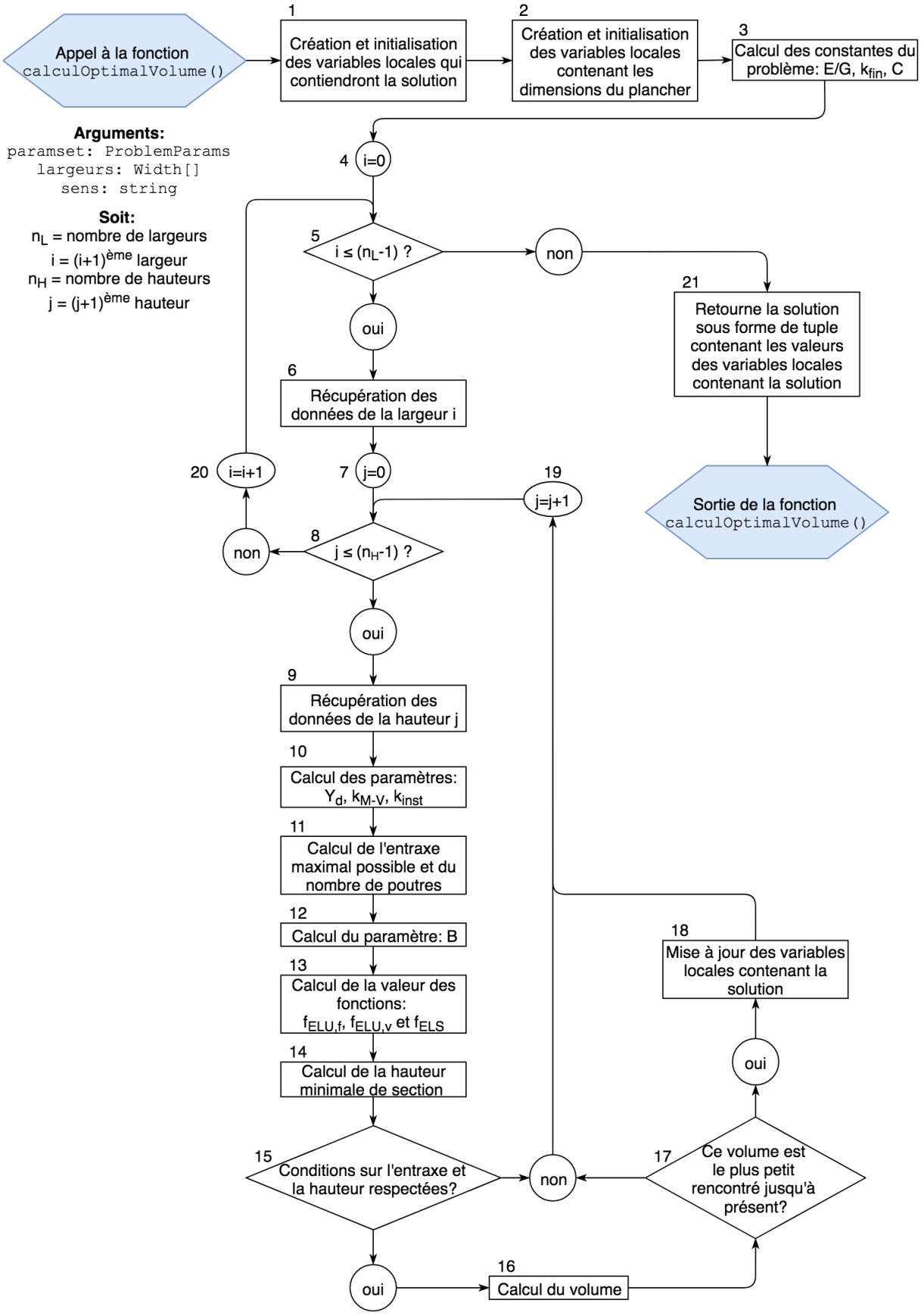


FIGURE 5.6 – Exécution de la méthode `calculOptimalVolume()` de la page `SolutionPage`

Les valeurs finales de ces variables locales seront, lors de l'étape (20) du schéma représenté à la figure 5.6, retournées sous forme d'un tuple comme décrit au tableau 5.21. L'extrait de code correspondant à cette première étape est donné au code 5.5.

```
let optiWidth: number = 0;
let optiHeight: number = 0;
let optiNumber: number = 0;
let optiVol: number = 0;
let cost: number = 0;
let entraxe: number = 0;
```

CODE 5.5 – Extrait du code de la méthode `calculOptimalVolume()` de la page `SolutionPage` correspondant à l'étape 1 de la figure 5.6

L'étape (2) du schéma représenté à la figure 5.6 consiste à créer et initialiser deux variables locales, `dimL` et `dimP`, qui contiennent les dimensions du plancher. La variable `dimL` correspond au paramètre L, c'est à dire la portée des poutres, et la variable `dimP` correspond au paramètre P, qui est l'autre dimension du plancher. Donc, lorsqu'on est dans le cas `sens='long'`, on associe la plus grande dimension du plancher à `dimL` et la plus petite à `dimP`, et inversément lorsque `sens='court'`. On rappelle que, par convention, la plus petite dimension est stockée à la première place du tableau `dimLP[]` du modèle `ProblemParams`. L'extrait de code correspondant à l'étape (2) est donné au code 5.6.

```
let dimL: number;
let dimP: number;

if (sens === 'long'){
  dimL = paramset.dimLP[1];
  dimP = paramset.dimLP[0];
}
else {
  dimL = paramset.dimLP[0];
  dimP = paramset.dimLP[1];
}
```

CODE 5.6 – Extrait du code de la méthode `calculOptimalVolume()` de la page `SolutionPage` correspondant à l'étape 2 de la figure 5.6

L'étape (3) de la méthode `calculOptimalVolum()` consiste simplement à calculer la valeur des paramètres $\frac{E}{G}$, k_{fin} et C, constants pour le problème. On utilise les paramètres stockés dans `parameters` et les équations 3.4 et 3.25. L'extrait de code correspondant à cette étape est donné au code 5.7.

```
let eg = paramset.modulE/paramset.modulG;
let kfin = paramset.getCombinaisonELSfin()/paramset.getCombinaisonELSinst();
let paramC = (24*eg)/25;
```

CODE 5.7 – Extrait du code de la méthode `calculOptimalVolume()` de la page `SolutionPage` correspondant à l'étape 3 de la figure 5.6

Les étapes (4) à (20) du schéma représenté à la figure 5.6 font partie de deux boucles `for` imbriquées, qui permettent de tester la meilleure solution pour chaque section de poutre possible. Ces étapes sont détaillées une par une ci-dessous. L'extrait de code correspondant à ces étapes est donné au code 5.8.

```
for (var i=0; i<largeurs.length; i++){
  let width = largeurs[i];
  let b = width.b;

  for (var j=0; j<width.h.length; j++){
    let height = width.h[j];

    let yd = (0.75*paramset.gammaM*paramset.getCombinaisonELU())/(this.
    eurocodesService.getkh(height,paramset.cat)*this.eurocodesService.ksys*
    paramset.kmod*paramset.fmk*1000);
    let kmv = (this.eurocodesService.getkh(height,paramset.cat)*paramset.fmk)/(
    paramset.kcr*paramset.fvk);
    let kinst = ((4*this.eurocodesService.getkh(height,paramset.cat)*this.
    eurocodesService.ksys*paramset.getCombinaisonELSinst()*paramset.kmod*
    paramset.fmk)/(3*paramset.getCombinaisonELU()*paramset.gammaM*paramset.
    modulE));

    let entraxeMaxELUf = (b/yd) * Math.pow(height/(100*dimL),2);
    let entraxeMaxELUv = (b/yd) * height/(kmv*100*dimL)
    let entraxeMaxELS = (b/yd) / ( Math.max(paramset.ninst,kfin*paramset.nfin) *
     kinst * (5*Math.pow(100*dimL/height,3)/32 + (3*eg*100*dimL)/(height*20)) );
    let entraxeMaxH = Math.min(entraxeMaxELUf,entraxeMaxELUv,entraxeMaxELS,
    paramset.entraxeMax);
    let number = Math.ceil(dimP*100/entraxeMaxH)+1;
    let entraxeMaxReal = dimP*100/(number-1);

    let paramB = (32*b)/(5*Math.max(paramset.ninst,kfin*paramset.nfin)*kinst*yd*
    entraxeMaxReal);

    let f_ELU_f = Math.sqrt(yd)*Math.sqrt(entraxeMaxReal/b);
    let f_ELU_v = (yd*kmv*entraxeMaxReal)/b;
    let f_ELS = 1/ ( Math.cbrt((paramB - Math.sqrt( Math.pow(paramB,2) + 4*Math.
    pow(paramC,3)/27 ))/2) + Math.cbrt((paramB + Math.sqrt( Math.pow(paramB,2) +
     4*Math.pow(paramC,3)/27 ))/2) );

    let hmin = Math.max(f_ELU_f, f_ELU_v, f_ELS)*dimL*100;

    if(height >= hmin && paramset.hmax >= height && entraxeMaxReal >= b &&
    entraxeMaxReal >= paramset.emin) {

      let volume = number*height*b*dimL/10000; //[m3]
      if(volume < optiVol || optiVol==0) {
        optiVol = volume; //[m3]
        optiNumber = number;
        optiWidth = b; //[cm]
        optiHeight = height; //[cm]
        cost = number*dimL*width.m[j] + number*width.p[j] + 2*number*width.f[j];
        entraxe = entraxeMaxReal; //[cm]
      }
    }
  }
}
```

Code 5.8 – Extrait du code de la méthode `calculOptimalVolume()` de la page `SolutionPage` correspondant aux étapes 4 à 20 de la figure 5.6

La première boucle `for` permet de parcourir toutes les largeurs une par une. L'étape (4) correspond à l'initialisation de cette boucle, en créant une variable locale `i` et en l'initialisant à 0. La variable `i` représente la largeur actuellement prise en compte dans la boucle. On l'initialise à 0 car l'indexation dans les tableaux commence à 0 dans TypeScript. L'étape (5) permet de vérifier que `i` correspond bien à un index du tableau `largeurs`. Ces deux étapes font partie de la

déclaration de la boucle `for` à la ligne 1 du code 5.8. L'étape (20), qui consiste à incrémenter `i` lorsqu'on ré-entre dans la boucle, fait aussi partie de cette déclaration à la ligne 1.

Si la condition de l'étape (5) n'est pas vérifiée, cela veut dire qu'on est arrivé au bout de la liste des largeurs de la plage de sections considérée. On passe alors à l'étape (21), où la solution est retournée, avant de sortir de la méthode `calculOptimalVolume()`.

Lorsqu'on rentre dans la première boucle `for`, après avoir vérifié la condition de l'étape (5), on passe à l'étape (6) qui consiste à créer deux variables locales, `width` et `b`, et à leur assigner respectivement l'objet de type `Width` correspondant à la largeur `i` et la dimension b de cette largeur respectivement. L'étape (6) correspond aux lignes 2 et 3 du code 5.8.

Après cela, on va entrer dans la deuxième boucle `for` qui permet de parcourir toutes les hauteurs associées à la largeur `i`, une par une. De la même manière que pour la première boucle, l'étape (7) correspond à l'initialisation de cette boucle, en créant une variable locale `j` et en l'initialisant à 0. La variable `j` représente la hauteur actuellement prise en compte dans la boucle. On l'initialise à 0 pour la même raison que précédemment. L'étape (8) permet de vérifier que `j` correspond bien à un index du tableau `h`. Ces deux étapes font partie de la déclaration de la boucle `for` à la ligne 5 du code 5.8. L'étape (19), qui consiste à incrémenter `j` lorsqu'on ré-entre dans la boucle, fait aussi partie de cette déclaration à la ligne 5.

Si la condition de l'étape (8) n'est pas vérifiée, cela veut dire qu'on est arrivé au bout de la liste des hauteurs associées à la largeur i. On passe alors à l'étape (20), où i est incrémenté de 1, avant de revenir à l'étape (5) et refaire les calculs précédents pour une nouvelle largeur.

Lorsqu'on rentre dans la deuxième boucle `for`, après avoir vérifié la condition de l'étape (8), on passe à l'étape (9) qui consiste à créer une variable locale `height` et à lui assigner la dimension h correspondant à la hauteur `j`. L'étape (9) correspond à la ligne 6 du code 5.8.

L'étape suivante (étape (10)) consiste à calculer la valeur des paramètres Y_d, k_{M-V} et k_{inst} sur base des équations les équations 3.22, 3.2 et 3.3, en utilisant les paramètres stockés dans `parameters` et les dimensions b et h de la section (i, j). Cette étape correspond aux lignes 8 à 10 du code 5.8.

L'objectif de l'étape (11) est de calculer le nombre minimal de poutres de section (i, j) nécessaire pour satisfaire tous les critères. Cette étape correspond aux lignes 12 à 17 du code 5.8. D'abord, aux lignes 12, 13 et 14, on calcule l'entraxe maximal qu'il est possible d'atteindre en considérant uniquement le critère ELU de flexion, le critère ELU de cisaillement et les critères ELS et on associe ces valeurs aux variables locales `entraxeMaxELUf`, `entraxeMaxELUv` et `entraxeMaxELS` respectivement.

Pour déterminer ces entraxes maximaux, on se réfère aux équations 3.40, 3.43 et 3.44 (pour cette dernière, on garde à l'esprit que $Z_d = Y_d^{-1} \cdot b/e$). En réarrangeant les termes, on trouve 3 nouvelles équations (5.1, 5.2 et 5.3) :

$$e_{max_{ELU,f}} \leq \frac{b}{Y_d} \left(\frac{h}{100 \cdot L} \right)^2 \tag{5.1}$$

$$e_{max_{ELU,v}} \leq \frac{b \cdot h}{100 \cdot L \cdot k_{M-V} \cdot Y_d} \tag{5.2}$$

$$e_{max_{ELS}} \leq \frac{b}{Y_d} \cdot \left[k_{inst} \cdot \max\left(\eta_{inst},\, k_{fin} \cdot \eta_{fin}\right) \left[\frac{5}{32} \left(\frac{100 \cdot L}{h} \right)^3 + \frac{3}{20} \frac{E}{G} \left(\frac{100 \cdot L}{h} \right) \right] \right]^{-1} \tag{5.3}$$

Passage de l'équation 3.40 à l'équation 5.1 :

Equation de départ :

$$h_{ELU,f} \geq LY_d^{\frac{1}{2}} \left(\frac{b}{e}\right)^{-\frac{1}{2}}$$

Comme le but n'est plus de trouver la hauteur minimale mais de trouver l'entraxe maximal, on renomme ces deux variables : $h_{ELU,f} \rightarrow h$ et $e \rightarrow e_{max_{ELU,f}}$

1 : On divise les deux membres par L :

$$\frac{h}{L} \geq Y_d^{\frac{1}{2}} \left(\frac{b}{e_{max_{ELU,f}}}\right)^{-\frac{1}{2}}$$

2 : On élève les deux membres au carré :

$$\left(\frac{h}{L}\right)^2 \geq Y_d \left(\frac{b}{e_{max_{ELU,f}}}\right)^{-1} = Y_d \frac{e_{max_{ELU,f}}}{b}$$

3 : On multiplie les deux membres par b/Y_d :

$$\left(\frac{h}{L}\right)^2 \frac{b}{Y_d} \geq e_{max_{ELU,f}}$$

4 : Pour respecter les unités utilisées dans l'application et obtenir un entraxe en $[cm]$, on doit multiplier L, donné en $[m]$, par 100. Finalement, on obtient bien :

$$e_{max_{ELU,f}} \leq \frac{b}{Y_d} \left(\frac{h}{100 \cdot L}\right)^2$$

Passage de l'équation 3.43 à l'équation 5.2 :

Equation de départ :

$$h_{ELU,v} \geq LY_d k_{M-V} \left(\frac{b}{e}\right)^{-1} = LY_d k_{M-V} \frac{e}{b}$$

Comme le but n'est plus de trouver la hauteur minimale mais de trouver l'entraxe maximal, on renomme ces deux variables : $h_{ELU,v} \rightarrow h$ et $e \rightarrow e_{max_{ELU,v}}$

1 : On divise les deux membres par L, Y_d et k_{M-V} :

$$\frac{h}{L \cdot Y_d \cdot k_{M-V}} \geq \frac{e_{max_{ELU,v}}}{b}$$

2 : On multiplie les deux membres par b :

$$\frac{b \cdot h}{L \cdot Y_d \cdot k_{M-V}} \geq e_{max_{ELU,v}}$$

4 : Pour respecter les unités utilisées dans l'application et obtenir un entraxe en $[cm]$, on doit multiplier L, donné en $[m]$, par 100. Finalement, on obtient bien :

$$e_{max_{ELU,v}} \leq \frac{b \cdot h}{100 \cdot L \cdot k_{M-V} \cdot Y_d}$$

Passage de l'équation 3.48 à l'équation 5.3 :

Equation de départ :

$$k_{inst} \cdot \max\left(\eta_{inst},\, k_{fin} \cdot \eta_{fin}\right) \left[\frac{5}{32}\left(\frac{L}{h_{ELS}}\right)^3 + \frac{3}{20}\frac{E}{G}\left(\frac{L}{h_{ELS}}\right)\right] \leq Z_d = \frac{1}{Y_d} \cdot \frac{b}{e}$$

Comme le but n'est plus de trouver la hauteur minimale mais de trouver l'entraxe maximal, on renomme ces deux variables : $h_{ELS} \rightarrow h$ et $e \rightarrow e_{max_{ELS}}$

1 : On divise les deux membres par le terme de gauche :

$$1 \leq \frac{1}{Y_d} \cdot \frac{b}{e_{max_{ELS}}} \cdot \left[k_{inst} \cdot \max\left(\eta_{inst},\, k_{fin} \cdot \eta_{fin}\right) \left[\frac{5}{32}\left(\frac{L}{h}\right)^3 + \frac{3}{20}\frac{E}{G}\left(\frac{L}{h}\right)\right]\right]^{-1}$$

2 : On multiplie les deux membres par $e_{max_{ELS}}$:

$$e_{max_{ELS}} \leq \frac{b}{Y_d} \cdot \left[k_{inst} \cdot \max\left(\eta_{inst},\, k_{fin} \cdot \eta_{fin}\right) \left[\frac{5}{32}\left(\frac{L}{h}\right)^3 + \frac{3}{20}\frac{E}{G}\left(\frac{L}{h}\right)\right]\right]^{-1}$$

4 : Pour respecter les unités utilisées dans l'application et obtenir un entraxe en $[cm]$, on doit multiplier L, donné en $[m]$, par 100. Finalement, on obtient bien :

$$e_{max_{ELS}} \leq \frac{b}{Y_d} \cdot \left[k_{inst} \cdot \max\left(\eta_{inst},\, k_{fin} \cdot \eta_{fin}\right) \left[\frac{5}{32}\left(\frac{100 \cdot L}{h}\right)^3 + \frac{3}{20}\frac{E}{G}\left(\frac{100 \cdot L}{h}\right)\right]\right]^{-1}$$

Dans la suite de l'étape (11), à la ligne 15 du code 5.8, on sélectionne la valeur la plus contraignante des valeurs données par les équations 5.1, 5.2 et 5.3, sans oublier de prendre en compte la valeur maximale de l'entraxe entrée par l'utilisateur. On place cette valeur, qui est donc le minimum parmi `entraxeMaxELUf`, `entraxeMaxELUv`, `entraxeMaxELS` et `paramset.entraxeMax`, dans la variable locale `entraxeMaxH`.

On calcule ensuite le nombre de poutres de section (i, j) qui sont nécessaires pour respecter cet entraxe maximal théorique. Ce nombre est donné par l'équation 5.4 où P est donné en $[m]$ et e_{max} en $[cm]$:

$$n = \frac{100 \cdot P}{e_{max}} + 1 \tag{5.4}$$

Cette opération est faite à la ligne 16 du code 5.8. Comme le nombre de poutres doit être un nombre entier, il faut arrondir le résultat à l'unité supérieure, ce dont s'occupe la méthode `ceil()` fournie par la classe `Math`.

Finalement, on recalcule l'entraxe réel qu'il faudra prévoir sur base du nombre de poutres obtenu (ligne 17 du code 5.8). Celui-ci vaut :

$$e_{real} = \frac{100 \cdot P}{n - 1} \tag{5.5}$$

Ayant la valeur réelle de l'entraxe e, on peut désormais calculer le dernier paramètre qu'il manquait, le paramètre B. Ceci correspond à l'étape (12) du schéma représenté à la figure 5.6 et à la ligne 19 du code 5.8. La valeur de B est donnée par l'équation 3.23.

Nous avons maintenant tous les ingrédients pour passer à l'étape (13) et calculer les valeurs des fonctions $f_{ELU,f}$, $f_{ELU,v}$ et f_{ELS}, définies par les équations 3.49, 3.50 et 3.51 respectivement. Cette étape correspond aux lignes 21 à 23 du code 5.8.

Ces valeurs permettent de calculer la hauteur minimale requise sur base du critère donné par l'équation 3.53. Nous plaçons cette valeur dans la variable locale `hmin`, sans oublier de la multiplier par 100 puisque `dimL` est donné en $[m]$ et qu'on souhaite avoir `hmin` en $[cm]$. Cette opération correspond à l'étape (14) du schéma représenté à la figure 5.6 et à la ligne 25 du code 5.8.

L'étape (15) est une condition `if`, à la ligne 27 du code 5.8, qui doit vérifier que 4 conditions soient satisfaites afin que la solution calculée puisse être acceptée. Ces 4 conditions sont :

1. `height >= hmin` : la hauteur de section doit être supérieure à la hauteur minimale requise
2. `paramset.hmax >= height` : la hauteur de section doit être inférieure à l'éventuelle hauteur maximale imposée par l'utilisateur
3. `entraxeMaxReal >= b` : la largeur de section doit être inférieure à l'entraxe maximal réel ; cela revient à imposer que les poutres ne se chevauchent pas (si `entraxeMaxReal = b`, on a un plancher plein)
4. `entraxeMaxReal >= paramset.emin` : l'entraxe maximal réel doit être supérieur à l'éventuel entraxe minimal imposé par l'utilisateur

Si la condition de l'étape (15) n'est pas satisfaite, cela veut dire que la solution ne peut pas être acceptée et on passe alors directement à l'étape (19), où j est incrémenté de 1, avant de revenir à l'étape (9) et refaire les calculs précédents pour une nouvelle hauteur.

En revanche, si la condition de l'étape (15) est satisfaite, cela veut dire que la solution satisfait à tous les critères requis et on passe à l'étape (16), qui consiste à calculer le volume de bois qui correspond à cette solution et à le placer dans la variable locale `volume`. Ce volume est calculé à la ligne 29 du code 5.8 et vaut, en $[m^3]$:

$$Vol = \frac{n \cdot h \cdot b \cdot L}{10000} \tag{5.6}$$

A l'étape (17), on regarde si ce volume, contenu dans la variable `volume`, est inférieur à celui contenu dans la variable `optiVol`. Pour rappel, `optiVol` contient le volume de ce qui sera la solution finale, et donc le plus petit volume calculé jusqu'à présent parmi tous les couples (i, j) déjà considérés. Comme nous avions initialisé la variable `optiVol` à 0 à l'étape (1), il faut aussi vérifier que `optiVol` n'est plus égal à 0. En effet, cela voudrait dire que c'est la première fois qu'on obtient une solution acceptable. Cette étape correspond à la ligne 30 du code 5.8.

Si les deux conditions testées à l'étape (17) sont satisfaites, on passe à l'étape (18) qui consiste à mettre à jour toutes les variables locales initialisées à l'étape (1) (lignes 31 à 36 du code 5.8). Sinon, on passe directement à l'étape (19) où j est incrémenté de 1, avant de revenir à l'étape (8) et refaire les calculs précédents pour une nouvelle hauteur.

La dernière étape, étape (21) du schéma de la figure 5.6, est celle de l'exécution de la méthode `calculOptimalVolume()` et consiste à retourner la solution finale sous la forme d'un tuple, de la forme décrite par le tableau 5.21, contenant les variables initialisées à l'étape (1) et mises à jour, dans le cas où au moins une solution a été trouvée, à l'étape (18). La ligne de code correspondant à cette étape est donnée au code 5.9.

```
return [sens, optiVol.toFixed(3), cost.toFixed(2), optiWidth, optiHeight,
    optiNumber, entraxe.toFixed(2), 0, 0];
```

CODE 5.9 – Extrait du code de la méthode `calculOptimalVolume()` de la page `SolutionPage` correspondant à l'étape 21 de la figure 5.6

On note que, comme prévu, tous les éléments de solution du tuple retourné seront nuls si aucune solution n'a été trouvée.

5.3.4 Solution la moins coûteuse

Le calcul de la solution optimale en termes de coût, pour une disposition simple du gîtage et dans un sens donné, se fait en exécutant la méthode `calculOptimalCost()`, dont les arguments sont, pour rappel, donnés dans le tableau 5.23. Un schéma de l'exécution de `calculOptimalCost()` est fourni à la figure 5.7.

L'exécution de la méthode `calculOptimalCost()` et très semblable à celle de la méthode `calculOptimalVolume()`. La seule différence se trouve aux étapes (16) et (17) : au lieu de calculer le volume de la solution et de le comparer au plus petit volume trouvé jusqu'à présent, on calcule le coût et on le compare au plus petit coût trouvé jusqu'à présent. A l'étape (18), on mettra alors les variables contenant les éléments de la solution à jour si le coût calculé à l'étape (16) est le plus petit qu'on ait trouvé jusqu'à présent. Ci-dessous, nous allons uniquement nous pencher sur les étapes qui diffèrent de celles de la méthode `calculOptimalVolume()`.

Les variables locales qui contiendront les éléments de la solution sont déclarées et initialisées à l'étape (1). Par rapport à celles de la méthode `calculOptimalVolume()`, on change le nom des variables contenant le coût et le volume pour mettre en évidence qu'on cherche cette fois à optimiser le coût et non le volume. L'extrait de code correspondant à l'étape (1) de la méthode `calculOptimalCost()` est donné au code 5.10.

```
let optiWidth: number = 0;
let optiHeight: number = 0;
let optiNumber: number = 0;
let optiCost: number = 0;
let vol: number = 0;
let entraxe: number = 0;
```

CODE 5.10 – Extrait du code de la méthode `calculOptimalCost()` de la page `SolutionPage` correspondant à l'étape 1 de la figure 5.7

Comme dans le cas de la méthode `calculOptimalVolume()`, on arrive ensuite à l'étape (16) uniquement si la solution calculée pour la section (i, j) est acceptable. On calcule alors le coût qu'engendre cette solution avec la formule suivante (code 5.11) :

$$\text{coût} = n \cdot L \cdot m(i,j) + n \cdot p(i,j) + 2 \cdot n \cdot f(i,j) \tag{5.7}$$

où $m(i, j)$ est le coût par mètre linéaire de la section (i, j) en [€/m], $p(i, j)$ le coût par pièce de section (i, j) en [€/pièce] et $f(i, j)$ le coût par fixation pour une section (i, j) en [€/extrémité].

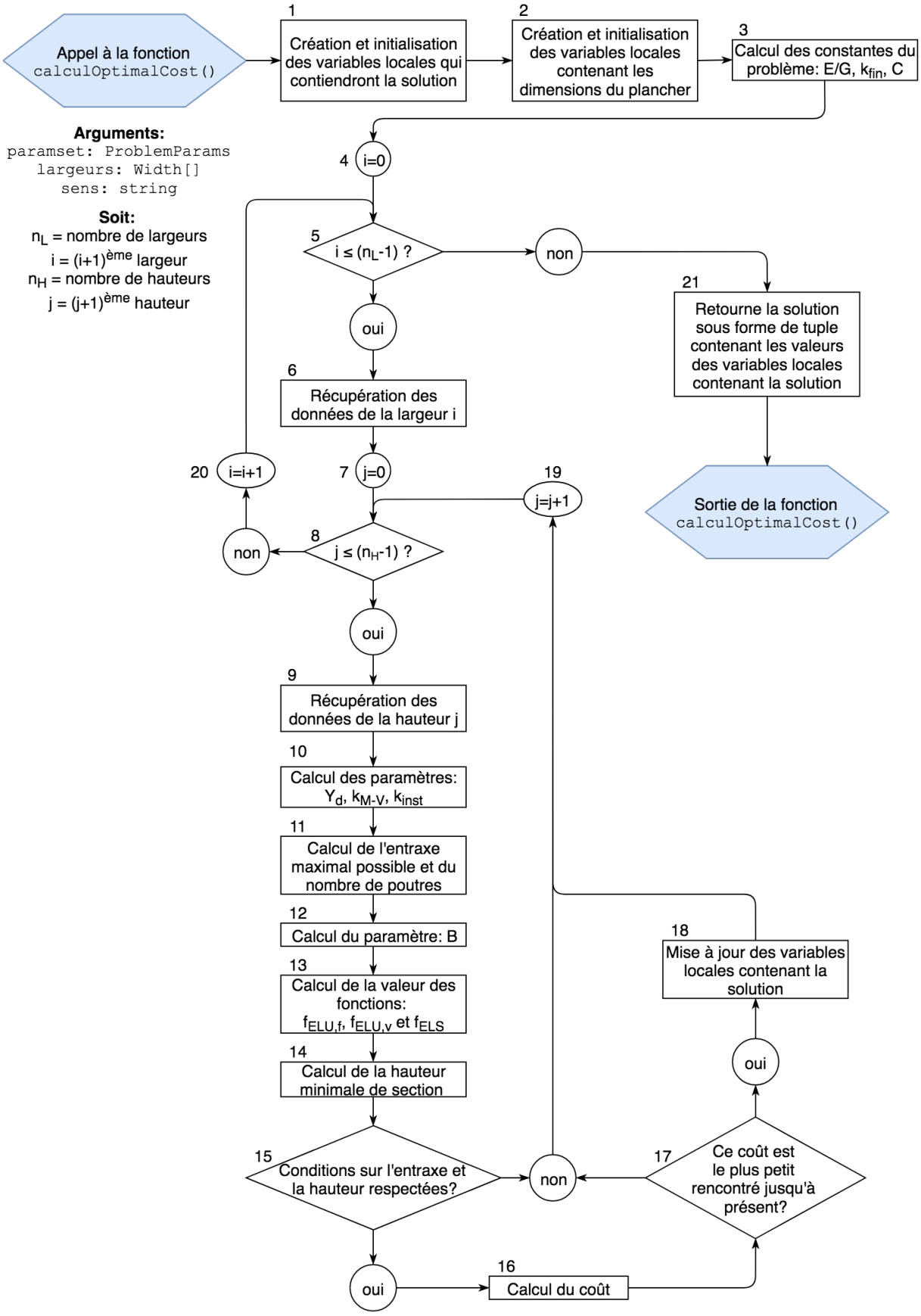


FIGURE 5.7 – Exécution de la méthode `calculOptimalCost()` de la page `SolutionPage`

```
1 let cost = number*dimL*width.m[j] + number*width.p[j] + 2*number*width.f[j]; //[
    €]
```

CODE 5.11 – Extrait du code de la méthode `calculOptimalCost()` de la page `SolutionPage` correspondant à l'étape 16 de la figure 5.7

A l'étape (17), on vérifie si ce coût, contenu dans la variable `cost`, est inférieur à celui contenu dans la variable `optiCost`. Pour rappel, `optiCost` contient le coût de ce qui sera la solution finale, et donc le plus petit coût calculé jusqu'à présent parmi tous les couple (i, j) déjà considérés. Comme nous avions initialisé la variable `optiCost` à 0 à l'étape (1), il faut aussi vérifier que `optiCost` n'est plus égal à 0. En effet, cela voudrait dire que c'est la première fois qu'on obtient une solution acceptable. Cette étape correspond à la ligne 1 du code 5.12.

Si les deux conditions testées à l'étape (17) sont satisfaites, on passe à l'étape (18) qui consiste à mettre à jour toutes la variables locales initialisées à l'étape (1). L'étape (18) correspond aux lignes 2 à 7 du code 5.12. Sinon, de la même manière que pour la méthode `calculOptimalVolume()`, on passe directement à l'étape (19) où j est incrémenté de 1, avant de revenir à l'étape (8) et refaire les calculs précédents pour une nouvelle hauteur.

```
1 if(cost < optiCost || optiCost==0) {
2   optiCost = cost; //[€]
3   optiNumber = number;
4   optiWidth = b; //[cm]
5   optiHeight = height; //[cm]
6   vol = optiNumber*optiHeight*optiWidth*dimL/10000; //[m3]
7   entraxe = entraxeMaxReal; //[cm]
8 }
```

CODE 5.12 – Extrait du code de la méthode `calculOptimalCost()` de la page `SolutionPage` correspondant à l'étape 17 de la figure 5.7

Comme pour la méthode `calculOptimalVolume()`, la dernière étape de l'exécution de la méthode `calculOptimalCost()` est de retourner la solution finale sous la forme d'un tuple, de la forme décrite par le tableau 5.21, contenant les variables initialisées à l'étape (1) et mises à jour, dans le cas où au moins une solution a été trouvée à l'étape (18). De nouveau, tous les éléments de solution du tuple retourné seront nuls si aucune solution n'a été trouvée.

5.3.5 Recours à une poutre intermédiaire

Pour calculer la meilleure solution dans le cas où on utilise une poutre intermédiaire, on utilise les méthodes `calculOptimalVolume_Inter()` et `calculOptimalCost_Inter()`, qui sont une adaptation des méthodes `calculOptimalVolume()` et `calculOptimalCost()`. Ci dessous, nous expliquons ces adaptations en nous référant aux figures 5.6 et 5.7.

Les méthodes `calculOptimalVolume_Inter()` et `calculOptimalCost_Inter()` fonctionnent dans un premier temps comme `calculOptimalVolume()` et `calculOptimalCost()`, sauf qu'on fait l'hypothèse d'un gîtage avec une seule poutre (la poutre intermédiaire). L'étape (11) n'a donc plus lieu d'être et l'entraxe qui sera utilisé dans les calculs aux étapes (12) à (14) peut être calculé dès l'étape (2). En effet, la valeur qu'on utilisera pour l'entraxe, dans le cas d'un gîtage constitué d'une seule poutre, vaut la moitié de la dimension transversale à la portée des poutres (`dimP`). On adapte donc le code de l'étape (2), qui est donné au code 5.13.

```
let dimL: number;
let dimP: number;

if (sens_inter === 'long'){
  dimL = paramset.dimLP[1];
  dimP = paramset.dimLP[0];
  entraxe_inter = (dimP*100)/2;
}
else {
  dimL = paramset.dimLP[0];
  dimP = paramset.dimLP[1];
  entraxe_inter = (dimP*100)/2;
}
```

CODE 5.13 – Extrait du code des méthodes `calculOptimalVolume_inter()` et `calculOptimalCost_inter()` de la page `SolutionPage` correspondant à l'adaptation de l'étape 2 des figures 5.6 et 5.7

Ensuite, toutes les étapes à partir de l'étape (3) jusqu'à l'étape (20) comprise sont identiques (hormis l'étape (11) qui disparaît). A la fin de ce processus, on trouve donc la section qui donnera la poutre intermédiaire la moins volumineuse (ou la moins coûteuse, selon la méthode) qui satisfait aux différents critères de dimensionnement et éventuellement de hauteur maximale. Avant de pouvoir retourner la solution, il reste encore à trouver le nombre d'éléments transversaux et leur section qu'il faudra placer transversalement de part et d'autre de la poutre intermédiaire pour compléter le gîtage. Nous inclurons ce calcul dans la dernière étape des méthodes `calculOptimalVolume_Inter()` et `calculOptimalCost_Inter()`, que nous appelons l'étape (21bis).

L'extrait de la méthode `calculOptimalVolume_inter()` correspondant à l'étape (21bis) est donné au code 5.14.

```
1  if (optiVol_inter != 0) {
2    // CALCUL DU DEMI-GITAGE - LE MOINS VOLUMINEUX
3
4    let dimLbis = (dimP-optiWidth_inter/100)/2;
5    let dimPbis = dimL;
6
7    let bis_result: (string|number)[];
8    let paramset_bis = this.clone(paramset);
9    bis_result = this.calculOptimalVolumeBIS(paramset_bis, largeurs, 'court',
       dimLbis, dimPbis);
10
11   return [sens_inter, (optiVol_inter+2*Number(bis_result[1])).toFixed(3), (
       cost_inter+2*Number(bis_result[2])).toFixed(2), Number(bis_result[3]),
       Number(bis_result[4]), Number(bis_result[5]), Number(bis_result[6]),
       optiWidth_inter, optiHeight_inter];
12 }
13 else {
14   return [0, 0, 0, 0, 0, 0, 0, 0, 0];
15 }
```

CODE 5.14 – Extrait du code de la méthode `calculOptimalVolume_inter()` de la page `SolutionPage` correspondant à l'étape (21bis)

A la ligne 1, on vérifie qu'une solution impliquant une poutre intermédiaire est possible. Cela revient à vérifier que la variable `optiVol_inter`, déclarée et initialisée à l'étape (1), a été mise à

jour et n'est donc plus égale à 0. Si c'est le cas, on exécute le code à l'intérieur du `if` ; sinon, on exécute le code à l'intérieur du `else`, qui retourne simplement un tuple dont tous les éléments sont nuls pour montrer qu'il n'existe pas de solution.

Le code situé à l'intérieur du `if` comprend lui-même plusieurs étapes. L'objectif est de définir deux nouvelles surfaces de plancher de mêmes dimensions $L_{bis} \times P_{bis}$, qui correspondent aux espaces situés de part et d'autre de la poutre intermédiaire (comme illustré à la figure 5.8), puis de les traiter comme des gîtages à part entière.

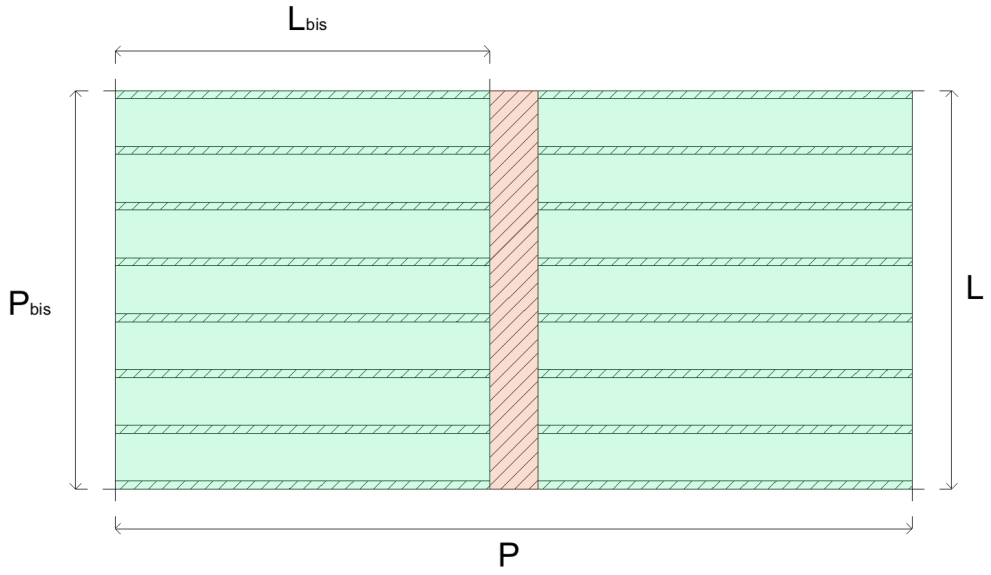


FIGURE 5.8 – Définition de deux nouvelles surfaces de plancher (en vert) de part et d'autre de la poutre intermédiaire (en rouge)

D'abord, on définit deux nouvelles variables (lignes 4 et 5 du code 5.14) : `dimLbis` et `dimPbis` qui correspondent aux dimensions L_{bis} et P_{bis}. Les valeurs de L_{bis} et P_{bis} se déduisent à partir de la figure 5.8 :

$$L_{bis} = \frac{P - b_{inter}}{2} \tag{5.8}$$

$$P_{bis} = L \tag{5.9}$$

où b_{inter} est la largeur de la poutre intermédiaire exprimée en $[m]$.

Ensuite, nous allons calculer la meilleure solution pour un plancher de dimension $L_{bis} \times P_{bis}$, en utilisant les méthodes `calculOptimalVolume()` ou `calculOptimalCost()`. A la ligne 7 du code 5.14, on déclare déjà le tuple `bis_result` qui contiendra la solution de ce plancher. A la ligne 8 du même code, on crée une copie de l'objet `paramset`, qui contient tous les paramètres du problème de base (plancher $L \times P$). En effet, les seuls paramètres qu'il faudra modifier par rapport aux paramètres du problème de base sont les dimensions du plancher.

Afin de calculer la solution optimale pour le plancher ($L_{bis} \times P_{bis}$) en termes de volume ou en termes de coût, on définit deux nouvelles méthodes : `calculOptimalVolumeBIS()` et `calculOptimalCostBIS()`, décrites dans le tableau 5.24. La méthode `calculOptimalVolumeBIS()` est donnée dans le code 5.15. La méthode `calculOptimalCostBIS()` est identique, sauf qu'on y fait appel à la méthode `calculOptimalCost()` au lieu de `calculOptimalVolume()`.

```
1  calculOptimalVolumeBIS(paramset_bis: ProblemParams, largeurs: Width[], sens:
       string, dimLbis: number, dimPbis: number){
2    paramset_bis.dimLP[0] = dimLbis;
3    paramset_bis.dimLP[1] = dimPbis;
4    return this.calculOptimalVolume(paramset_bis, largeurs, sens);
5  }
```

CODE 5.15 – Code de la méthode `calculOptimalVolumeBIS()` de la page `SolutionPage`

La méthode `calculOptimalVolumeBIS()` s'occupe de modifier les paramètres du problème de base en tenant compte des nouvelles dimensions L_{bis} et P_{bis} (lignes 2 et 3 du code 5.15). Ensuite, elle retourne le résultat de la méthode `calculOptimalVolume()` après lui avoir passé en argument l'ensemble des paramètres modifiés.

Revenons à l'étape (21bis). Nous avons maintenant la section optimale pour la poutre intermédiaire et, dans la variable `bis_result`, la solution optimale pour un plancher de dimensions $L_{bis} \times P_{bis}$. Il ne reste donc plus qu'a retourner l'ensemble de la solution optimale, ce qui est fait à la ligne 11 du code 5.14. La solution aura la forme du tuple décrit au tableau 5.21.

5.4 Mode d'emploi de l'application

Nous examinons, dans cette section, l'interface utilisateur de l'application, en en parcourant les différentes pages, en rappelant brièvement leur fonction et en y indiquant les différents boutons, champs, etc. Nous commençons par la page d'accueil, représentée à la figure 5.9.

5.4.1 Page d'accueil

1. Menu
2. Bouton permettant de changer la langue de l'application (la version actuelle de l'application n'est disponible qu'en français)
3. Bouton permettant de naviguer vers le formulaire de calcul rapide (figure 5.19)
4. Bouton permettant de naviguer vers le formulaire de calcul personnalisé (figure 5.26)
5. Bouton permettant de naviguer vers la liste des plages de sections (figure 5.10)
6. Bouton permettant de naviguer vers la page "à propos"

FIGURE 5.9 – Page d'accueil de l'application

5.4.2 Gestion des plages de sections

Liste des plages de sections

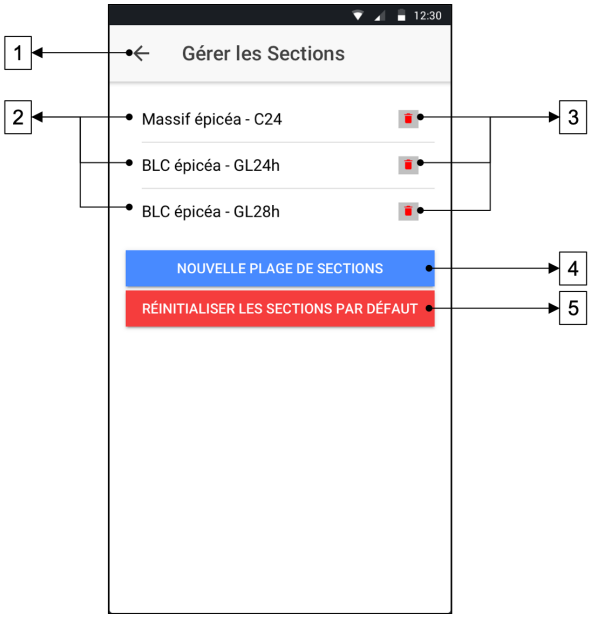


FIGURE 5.10 – Liste des plages de sections

1. Bouton retour (vers la page d'accueil)
2. Bouton permettant de naviguer vers la liste des largeurs de la plage de sections (figure 5.11)
3. Bouton permettant de supprimer la plage de sections
4. Bouton permettant de naviguer vers le formulaire d'ajout d'une plage de sections (figure 5.14)
5. Bouton permettant de réinitialiser la liste des plages de sections

Liste des largeurs d'une plage de sections

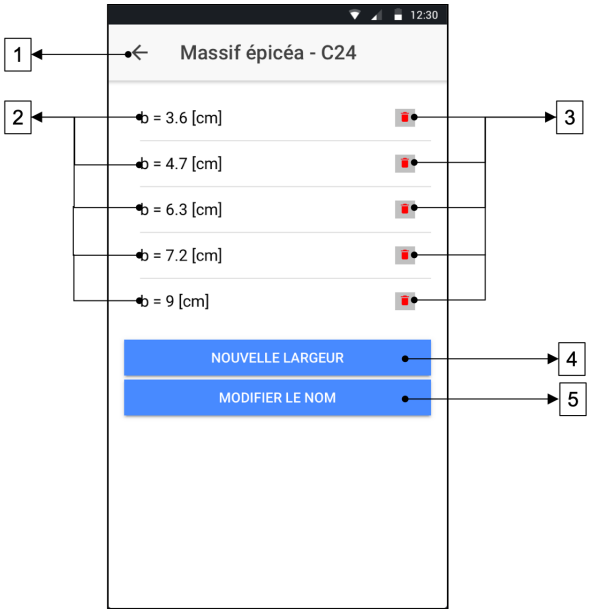


FIGURE 5.11 – Liste des largeurs d'une plage de sections (1)

1. Bouton retour (vers la liste des plages de sections)
2. Bouton permettant de naviguer vers la liste des hauteurs (et coûts) associées à la largeur (figure 5.13)
3. Bouton permettant de supprimer la largeur
4. Bouton permettant de naviguer vers le formulaire d'ajout d'une largeur (figure 5.16)
5. Bouton permettant de modifier le nom de la plage de sections (ajoute un champ et un bouton, cfr. figure 5.12)

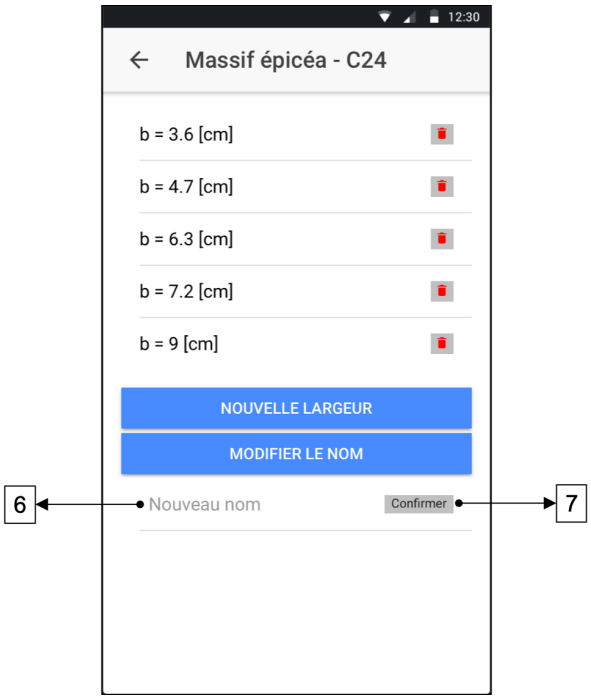


FIGURE 5.12 – Liste des largeurs d'une plage de sections (2)

6. Champ à remplir avec le nouveau nom de la plage de sections
7. Bouton permettant de confirmer le nouveau nom

Liste des hauteurs associées à une largeur d'une plage de sections

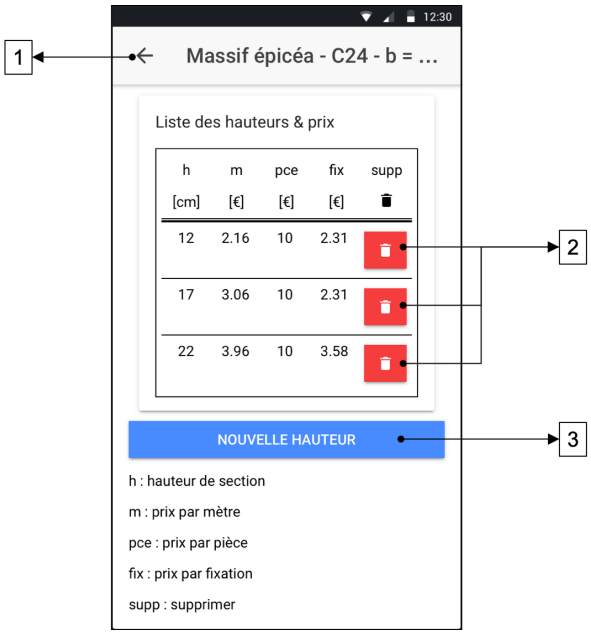


FIGURE 5.13 – Liste des hauteurs associées à une largeur

1. Bouton retour (vers la liste des largeurs)
2. Bouton permettant de supprimer la hauteur (et les coûts qui y sont associés)
3. Bouton permettant de naviguer vers le formulaire d'ajout d'une hauteur et ses coûts (figure 5.18)

Ajouter une plage de sections

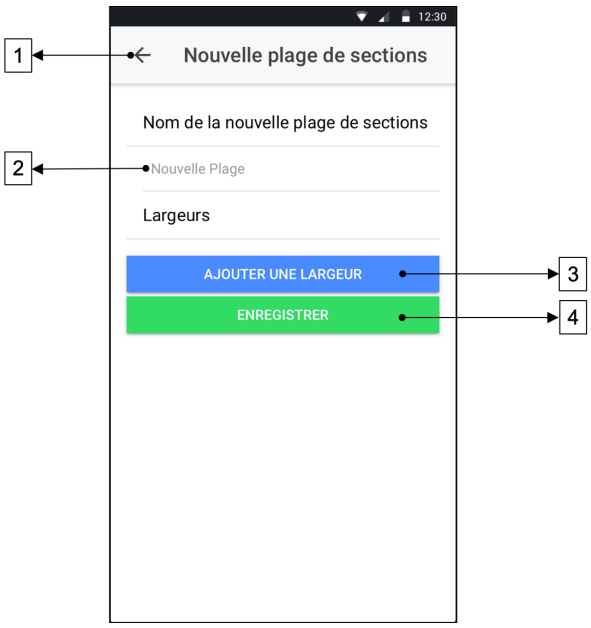


FIGURE 5.14 – Formulaire d'ajout d'une plage de sections (1)

1. Bouton retour (vers la liste des plages de sections)
2. Champ à remplir avec le nom à donner à la nouvelle plage de sections
3. Bouton permettant d'ajouter une largeur à la plage de sections (ajoute un champ, cfr. figure 5.15)
4. Bouton permettant d'enregistrer la nouvelle plage de sections

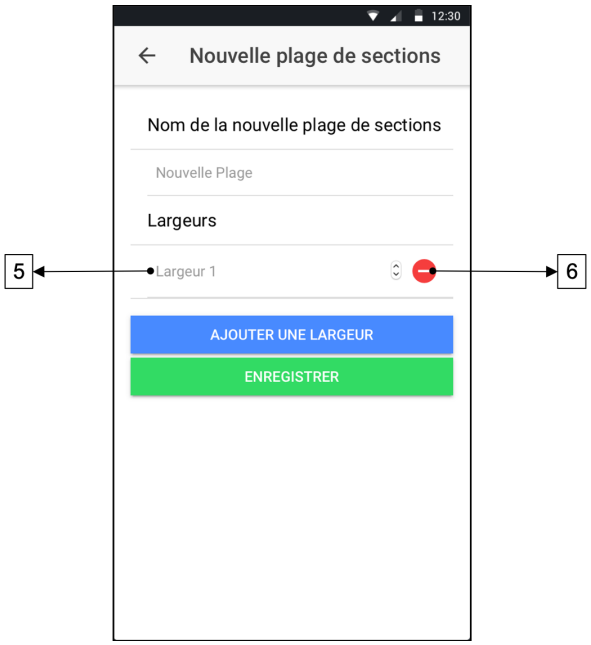


FIGURE 5.15 – Formulaire d'ajout d'une plage de sections (2)

5. Champ à remplir avec la dimension de la nouvelle largeur (en $[cm]$)
6. Bouton permettant de supprimer le champ destiné à ajouter une largeur

Ajouter une largeur à une plage de sections

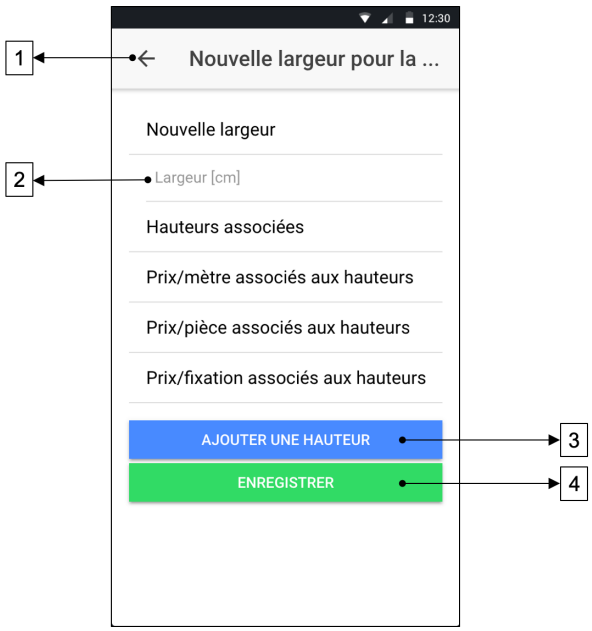


FIGURE 5.16 – Formulaire d'ajout d'une largeur à une plage de sections (1)

1. Bouton retour (vers la liste des largeurs)
2. Champ à remplir avec la dimension de la nouvelle largeur (en $[cm]$)
3. Bouton permettant d'associer une nouvelle hauteur (et ses coûts) à la largeur (ajoute des champs, cfr. figure 5.17)
4. Bouton permettant d'enregistrer la nouvelle largeur

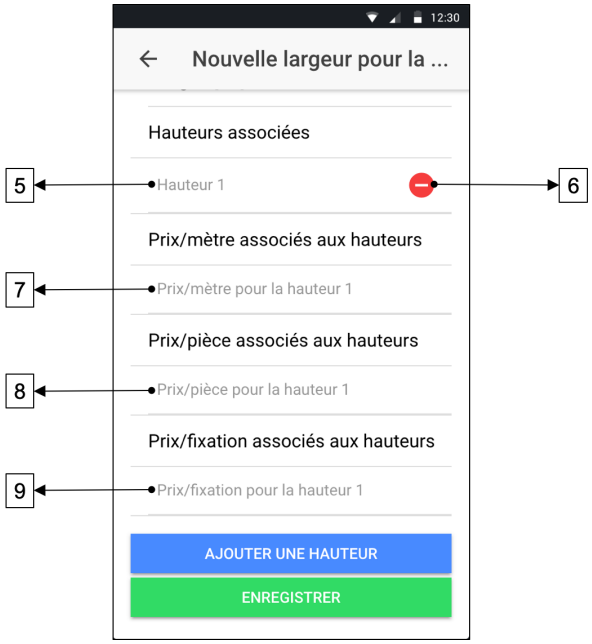


FIGURE 5.17 – Formulaire d'ajout d'une largeur à une plage de sections (2)

5. Champ à remplir avec la dimension de la nouvelle hauteur (en $[cm]$)
6. Bouton permettant de supprimer les champs destinés à ajouter une hauteur et ses coûts
7. Champ à remplir avec le coût par mètre de la section (en [€])
8. Champ à remplir avec le coût par pièce de la section (en [€])
9. Champ à remplir avec le coût par fixation de la section (en [€])

Ajouter une hauteur à une largeur d'une plage de sections

FIGURE 5.18 – Formulaire d'ajout d'une hauteur pour une largeur

1. Bouton retour (vers la liste des largeurs)
2. Champ à remplir avec la dimension de la nouvelle hauteur (en $[cm]$)
3. Champ à remplir avec coût par mètre correspondant (en [€])
4. Champ à remplir avec coût par pièce correspondant (en [€])
5. Champ à remplir avec coût par fixation correspondant (en [€])
6. Bouton permettant d'enregistrer la nouvelle hauteur

5.4.3 Calculer les solutions optimales d'un problème de gîtage

Calcul rapide

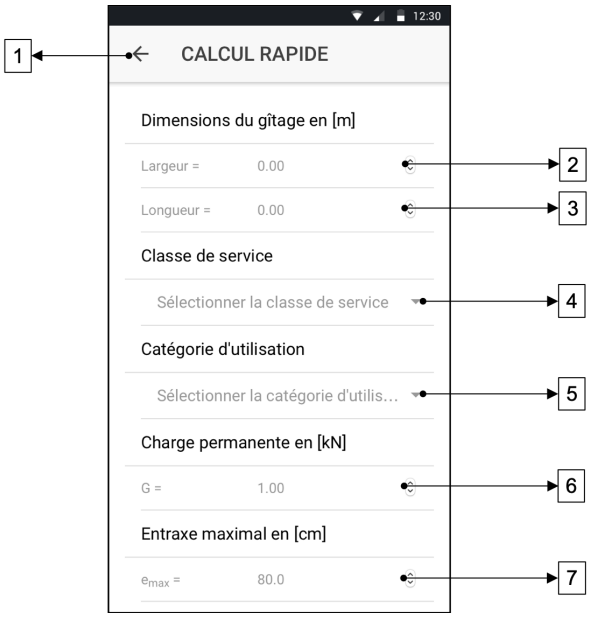


FIGURE 5.19 – Formulaire des paramètres nécessaires pour un calcul rapide (1)

1. Bouton retour (vers la page d'accueil)
2. Champ à remplir avec une des dimensions du plancher (en $[m]$)
3. Champ à remplir avec l'autre dimension du plancher (en $[m]$)
4. Liste déroulante permettant de sélectionner la classe de service du plancher (1, 2 ou 3)
5. Liste déroulante permettant de sélectionner la catégorie d'utilisation du plancher (cfr. tableau A.2)
6. Champ à remplir avec la charge permanente G subie par le plancher (en $[kN/m^2]$)
7. Champ à remplir avec l'entraxe maximal permis entre les poutres (en $[cm]$)

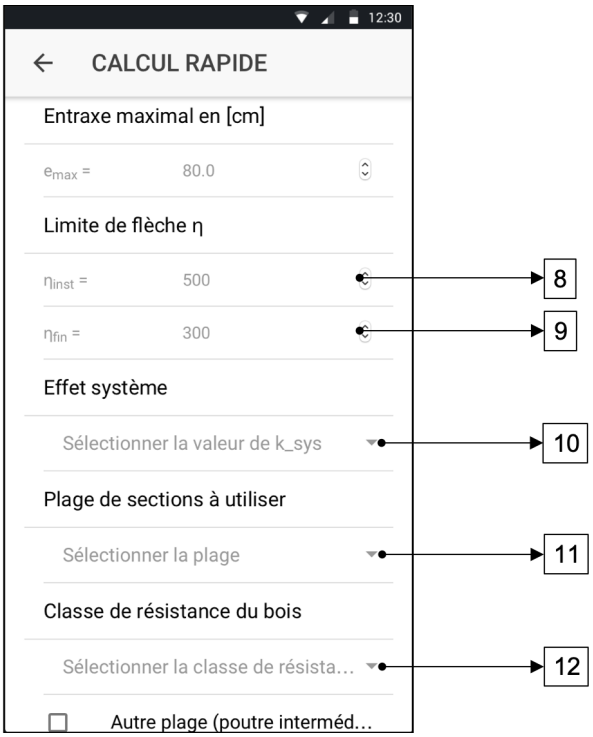


FIGURE 5.20 – Formulaire des paramètres nécessaires pour un calcul rapide (2)

8. Champ à remplir avec la valeur du paramètre η_{inst}
9. Champ à remplir avec la valeur du paramètre η_{fin}
10. Liste déroulante permettant de sélectionner la valeur du paramètre k_{sys} parmi les valeurs 1.0 (effet système non pris en compte), 1.1 (effet système pris en compte), ou la valeur automatique déterminée par l'application
11. Liste déroulante permettant de sélectionner la plage de sections à considérer
12. Liste déroulante permettant de sélectionner la classe de résistance du bois (cfr. tableaux A.8 et A.9)

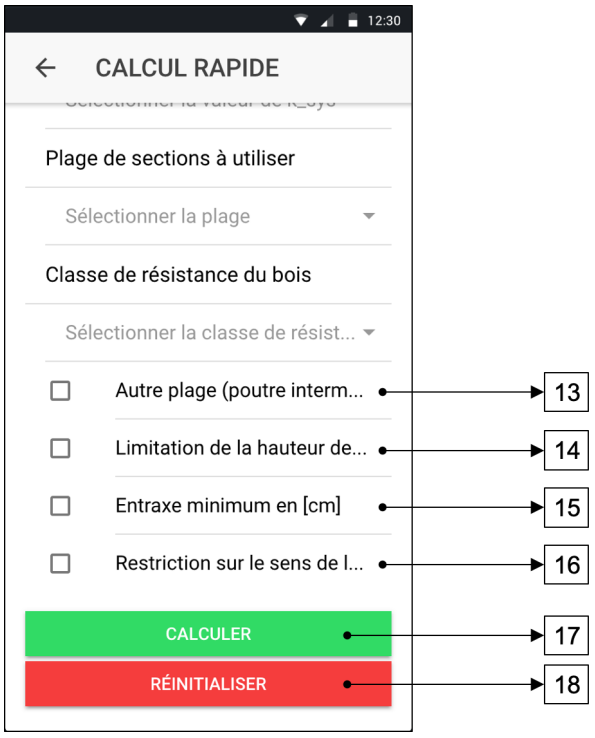


FIGURE 5.21 – Formulaire des paramètres nécessaires pour un calcul rapide (3)

13. Case à cocher si une poutre intermédiaire peut être considérée lors du calcul de la solution optimale (affiche les champs pour ajouter les informations nécessaires à propos de cette poutre, cfr. figure 5.22)
14. Case à cocher si la hauteur des poutres doit être limitée (affiche le champ pour préciser cette hauteur, cfr. figure 5.23)
15. Case à cocher si un entraxe minimal est requis entre les poutres (affiche le champ pour préciser cet entraxe, cfr. figure 5.24)
16. Case à cocher si le sens de la portée des poutres est imposé (affiche le champ pour préciser ce sens, cfr. figure 5.25)
17. Bouton permettant de lancer le calcul des solutions optimales en termes de volume de bois et en termes de coût et de les afficher
18. Bouton permettant de réinitialiser tous les champs du formulaire

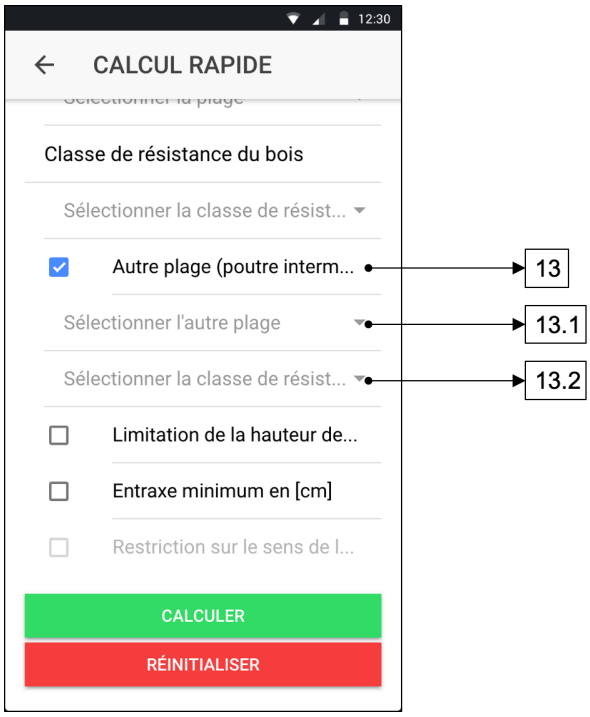


FIGURE 5.22 – Formulaire des paramètres nécessaires pour un calcul rapide (4)

13. Case à cocher si une poutre intermédiaire peut être considérée lors du calcul de la solution optimale
    - 13.1. Liste déroulante permettant de sélectionner la plage de sections à considérer pour la poutre intermédiaire
    - 13.2. Liste déroulante permettant de sélectionner la classe de résistance du bois de la poutre intermédiaire (cfr. tableaux A.8 et A.9)

Note : lorsque la case 13 est cochée, il n'est pas possible de sélectionner la case 16. En effet, si on a recours à une poutre intermédiaire, alors il y aura de toute façon au moins une poutre dans chaque sens.

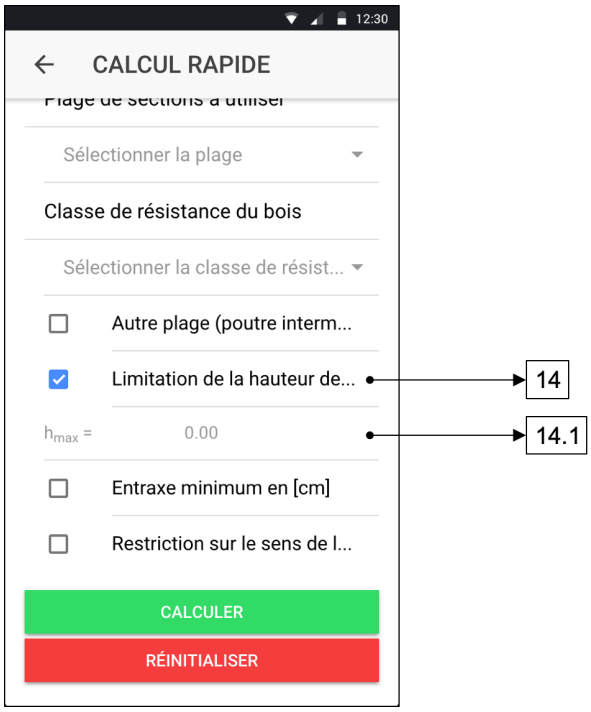


FIGURE 5.23 – Formulaire des paramètres nécessaires pour un calcul rapide (5)

14. Case à cocher si la hauteur des poutres doit être limitée
    - 14.1. Champ à remplir avec la valeur de la hauteur maximale les poutres (en $[cm]$)

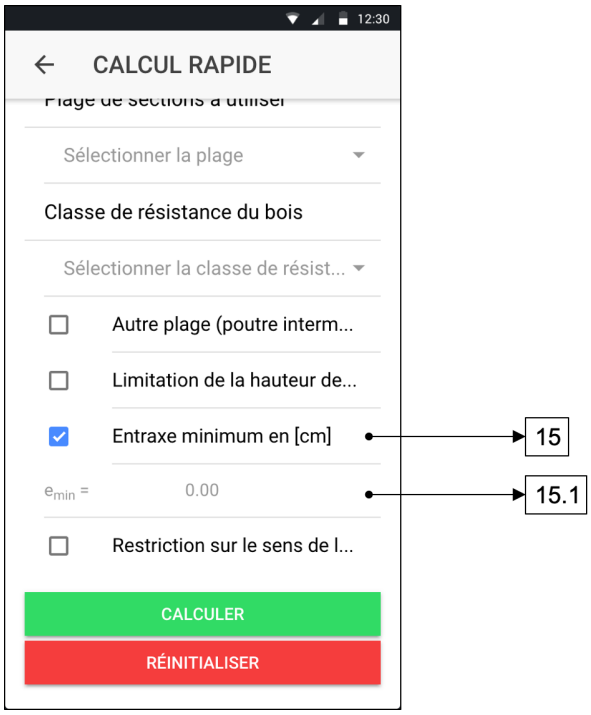


FIGURE 5.24 – Formulaire des paramètres nécessaires pour un calcul rapide (6)

15. Case à cocher si un entraxe minimal est requis entre les poutres
    15.1. Champ à remplir avec la valeur de l'entraxe minimal à prévoir entre les poutres (en $[cm]$)

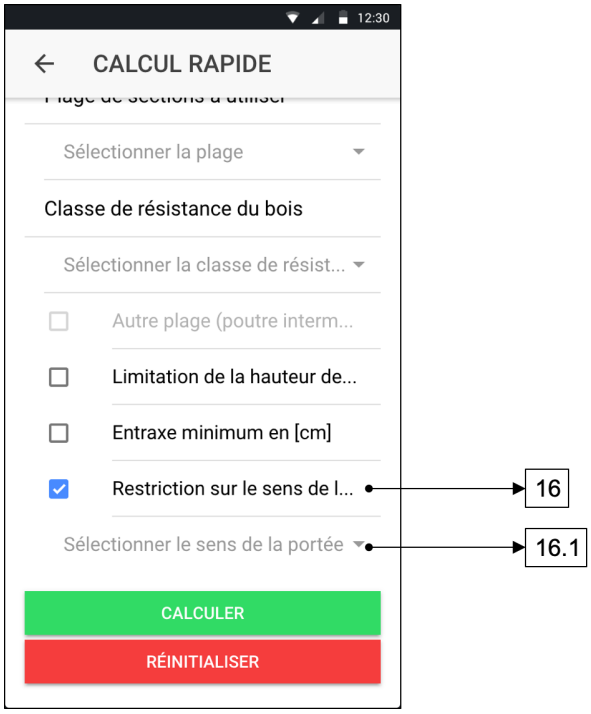


FIGURE 5.25 – Formulaire des paramètres nécessaires pour un calcul rapide (7)

16. Case à cocher si le sens de la portée des poutres est imposé
    16.1. Liste déroulante permettant de sélectionner le sens que doit avoir la portée des poutres (choisir entre sens le plus long ou sens le plus court)

Note : lorsque la case 16 est cochée, il n'est pas possible de sélectionner la case 13. En effet, si on impose le sens de la portée, on ne peut pas avoir recours à une poutre intermédiaire, puisqu'il y aurait dans ce cas de toute façon au moins une poutre dans chaque sens.

Calcul personnalisé

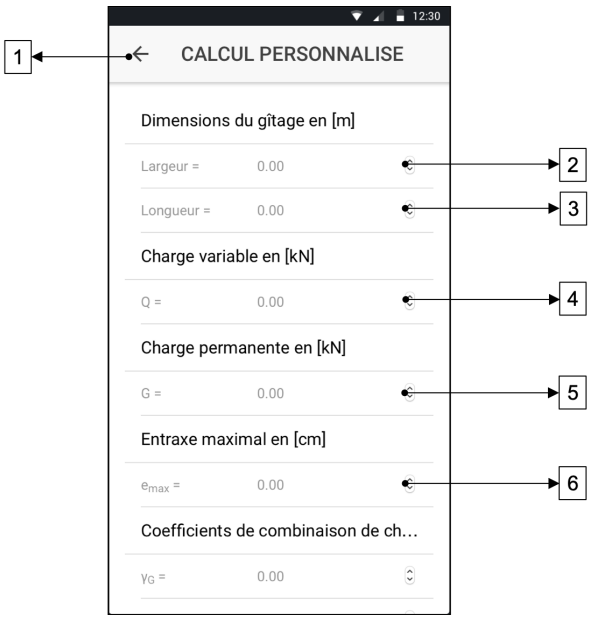


FIGURE 5.26 – Formulaire des paramètres nécessaires pour un calcul personnalisé (1)

1. Bouton retour (vers la page d'accueil)
2. Champ à remplir avec une des dimensions du plancher (en $[m]$)
3. Champ à remplir avec l'autre dimension du plancher (en $[m]$)
4. Champ à remplir avec la charge variable Q subie par le plancher (en $[kN/m^2]$)
5. Champ à remplir avec la charge permanente G subie par le plancher (en $[kN/m^2]$)
6. Champ à remplir avec l'entraxe maximal permis entre les poutres (en $[cm]$)

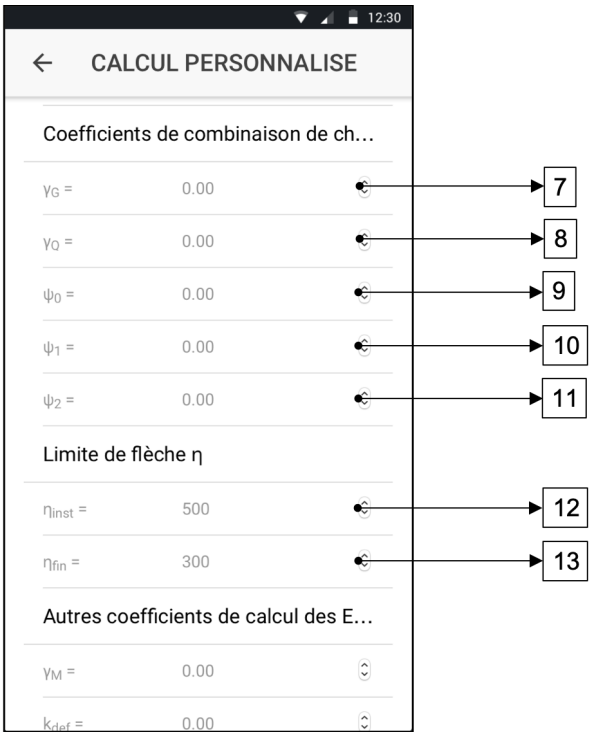


FIGURE 5.27 – Formulaire des paramètres nécessaires pour un calcul personnalisé (2)

7. Champ à remplir avec la valeur du coefficient γ_G
8. Champ à remplir avec la valeur du coefficient γ_Q
9. Champ à remplir avec la valeur du coefficient ψ_0
10. Champ à remplir avec la valeur du coefficient ψ_1
11. Champ à remplir avec la valeur du coefficient ψ_2
12. Champ à remplir avec la valeur du paramètre η_{inst}
13. Champ à remplir avec la valeur du paramètre η_{fin}

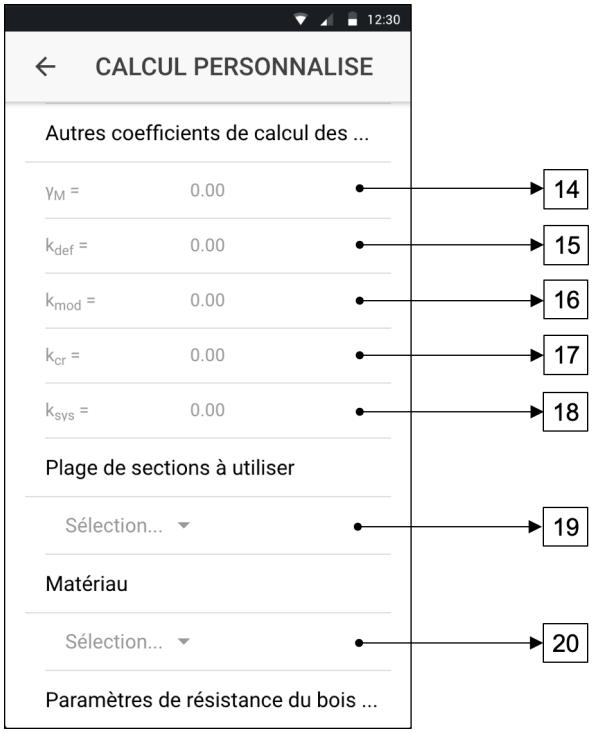


FIGURE 5.28 – Formulaire des paramètres nécessaires pour un calcul personnalisé (3)

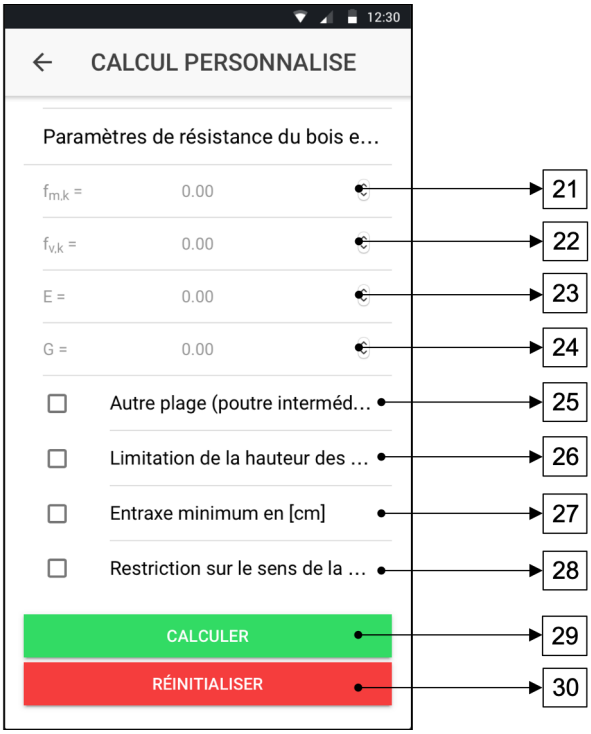


FIGURE 5.29 – Formulaire des paramètres nécessaires pour un calcul personnalisé (4)

14. Champ à remplir avec la valeur du coefficient partiel γ_M
15. Champ à remplir avec la valeur du facteur de déformation k_{def}
16. Champ à remplir avec la valeur du facteur de modification k_{mod}
17. Champ à remplir avec la valeur du facteur de fissuration k_{cr}
18. Champ à remplir avec la valeur du facteur système k_{sys}
19. Liste déroulante permettant de sélectionner la plage de sections à considérer
20. Liste déroulante permettant de sélectionner le matériau qui sera utilisé : bois massif ou bois lamellé-collé (BLC)
21. Champ à remplir avec la valeur de la résistance caractéristique du bois à la flexion $f_{m,k}$ en $[MPa]$
22. Champ à remplir avec la valeur de la résistance caractéristique du bois au cisaillement $f_{v,k}$ en $[MPa]$
23. Champ à remplir avec la valeur du module d'élasticité E du bois en $[MPa]$
24. Champ à remplir avec la valeur du module de cisaillement G du bois en $[MPa]$
25. Case à cocher si une poutre intermédiaire peut être considérée lors du calcul de la solution optimale (affiche les champs pour ajouter les informations nécessaires à propos de cette poutre, cfr. figure 5.30)
26. Case à cocher si la hauteur des poutres doit être limitée (affiche le champ pour préciser cette hauteur, cfr. figure 5.31)
27. Case à cocher si un entraxe minimal est requis entre les poutres (affiche le champ pour préciser cet entraxe, cfr. figure 5.32)
28. Case à cocher si le sens de la portée des poutres est imposé (affiche le champ pour préciser ce sens, cfr. figure 5.33)
29. Bouton permettant de lancer le calcul des solutions optimales en termes de volume de bois et en termes de coût et de les afficher
30. Bouton permettant de réinitialiser tous les champs du formulaire

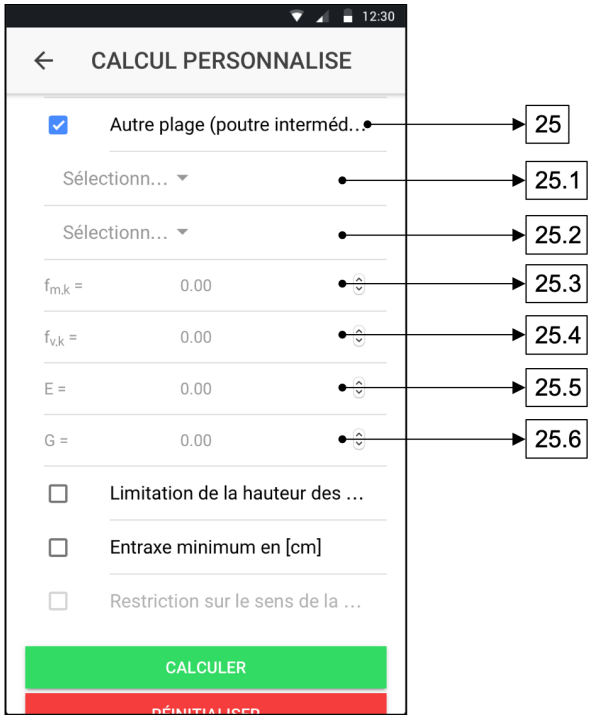


FIGURE 5.30 – Formulaire des paramètres nécessaires pour un calcul personnalisé (5)

25. Case à cocher si une poutre intermédiaire peut être considérée lors du calcul de la solution optimale
    - 25.1. Liste déroulante permettant de sélectionner la plage de sections à considérer pour la poutre intermédiaire
    - 25.2. Liste déroulante permettant de sélectionner le matériau de la poutre intermédiaire : bois massif ou bois lamellé-collé (BLC)
    - 25.3. Champ à remplir avec la valeur de la résistance caractéristique de la poutre intermédiaire à la flexion $f_{m,k}$ en $[MPa]$
    - 25.4. Champ à remplir avec la valeur de la résistance caractéristique de la poutre intermédiaire au cisaillement $f_{v,k}$ en $[MPa]$
    - 25.5. Champ à remplir avec la valeur du module d'élasticité E de la poutre intermédiaire en $[MPa]$
    - 25.6. Champ à remplir avec la valeur du module de cisaillement G de la poutre intermédiaire en $[MPa]$

Note : pour les mêmes raisons que dans le cas d'un calcul rapide, il n'est pas possible de sélectionner la case 28 lorsque la case 25 est cochée.

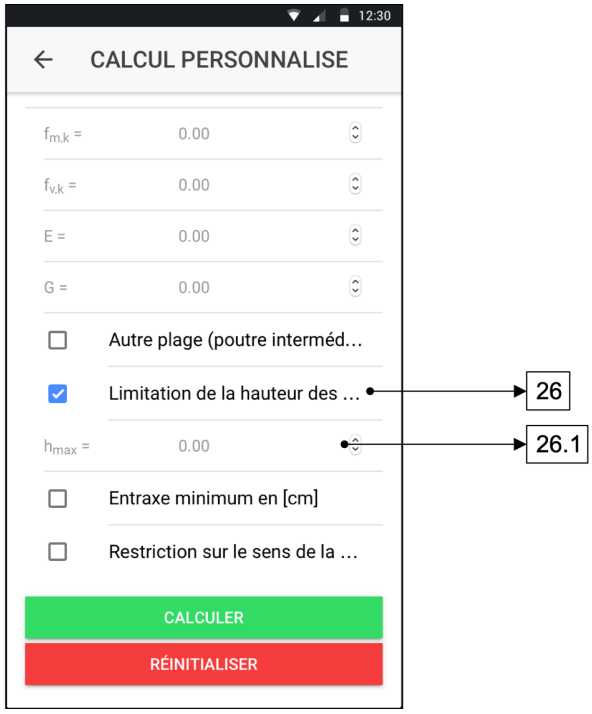


FIGURE 5.31 – Formulaire des paramètres nécessaires pour un calcul personnalisé (6)

26. Case à cocher si la hauteur des poutres doit être limitée
    - 26.1. Champ à remplir avec la valeur de la hauteur maximale les poutres (en $[cm]$)

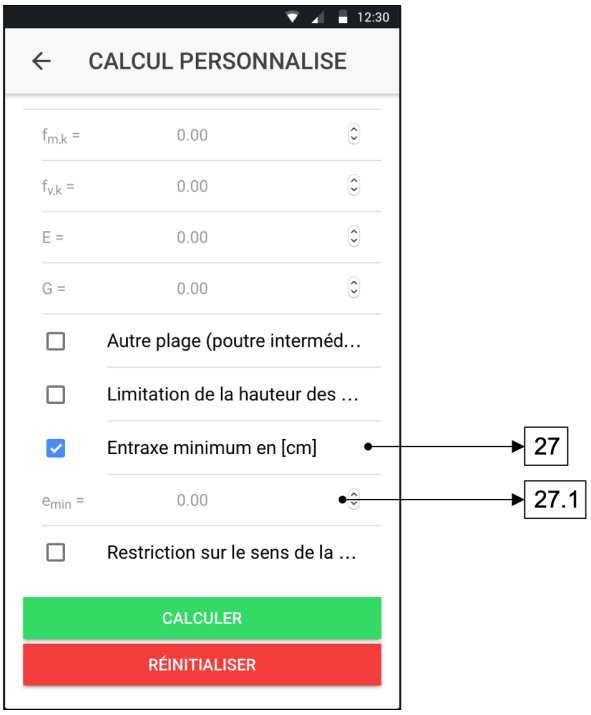


FIGURE 5.32 – Formulaire des paramètres nécessaires pour un calcul personnalisé (7)

27. Case à cocher si un entraxe minimal est requis entre les poutres
    27.1. Champ à remplir avec la valeur de l'entraxe minimal à prévoir entre les poutres (en $[cm]$)

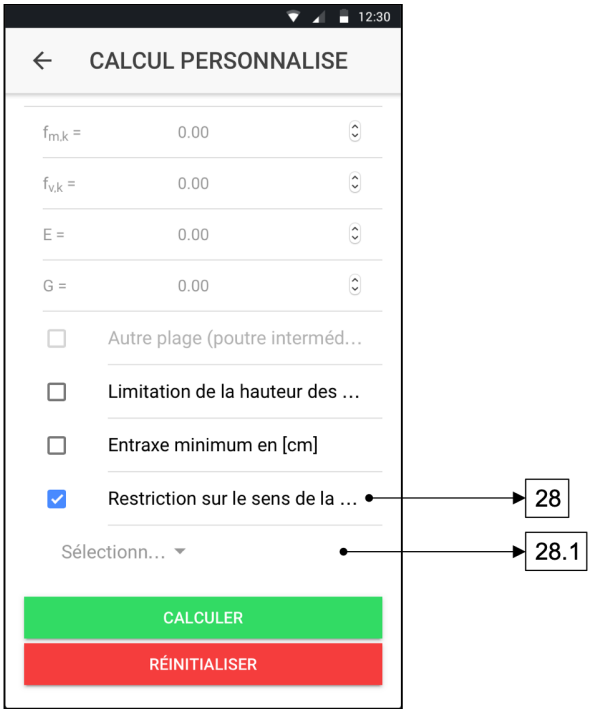
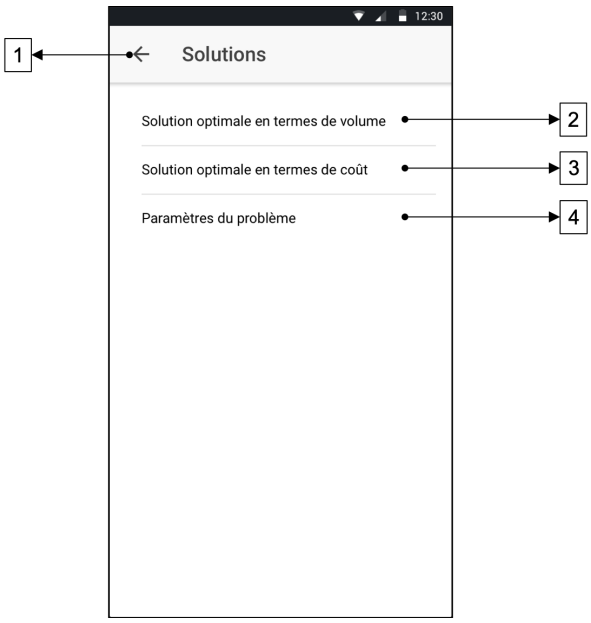


FIGURE 5.33 – Formulaire des paramètres nécessaires pour un calcul personnalisé (8)

28. Case à cocher si le sens de la portée des poutres est imposé
    28.1. Liste déroulante permettant de sélectionner le sens que doit avoir la portée des poutres (choisir entre sens le plus long ou sens le plus court)

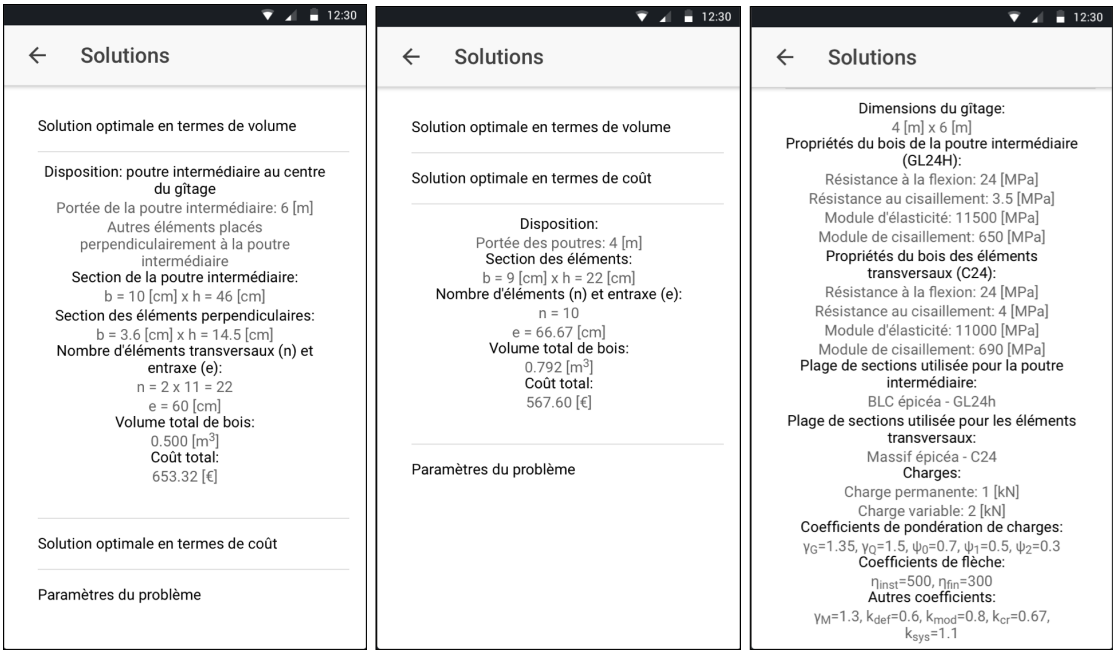
Note : lorsque la case 28 est cochée, il est n'est pas possible de sélectionner la case 25. En effet, si on a recours à une poutre intermédiaire, alors il y aura de toute façon au moins une poutre dans chaque sens.

Affichage des solutions - une solution existe



1. Bouton retour (vers le formulaire de calcul)
2. Bouton permettant d'afficher/masquer les éléments de la solution optimale en termes de volume (cfr. figure 5.35a)
3. Bouton permettant d'afficher/masquer les éléments de la solution optimale en termes de coût (cfr. figure 5.35b)
4. Bouton permettant d'afficher/masquer la valeur des paramètres utilisés pour le calcul (cfr. figure 5.35c)

FIGURE 5.34 – Page des solutions



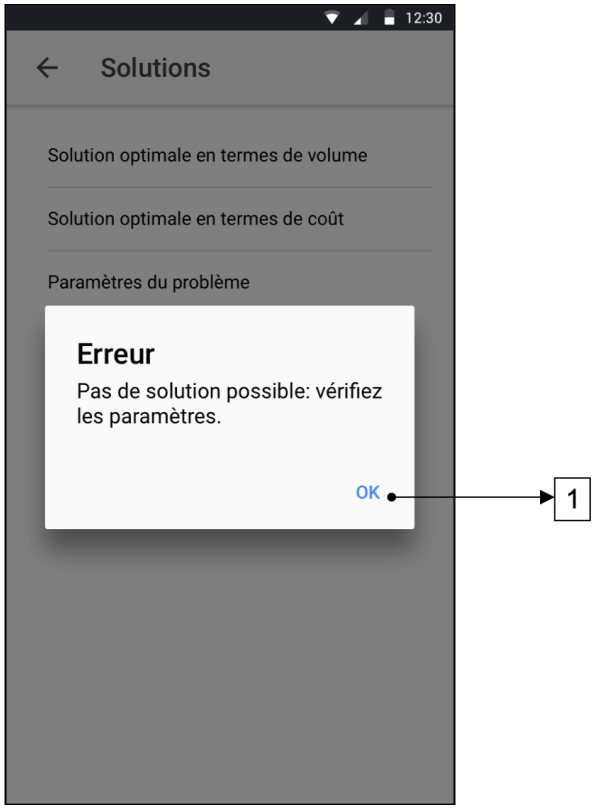
(a) Affichage de la solution optimale en termes de volume

(b) Affichage de la solution optimale en termes de coût

(c) Affichage des paramètres utilisés pour le calcul

FIGURE 5.35 – Exemples d'écrans de la page des solutions dans le cas où au moins une solution existe

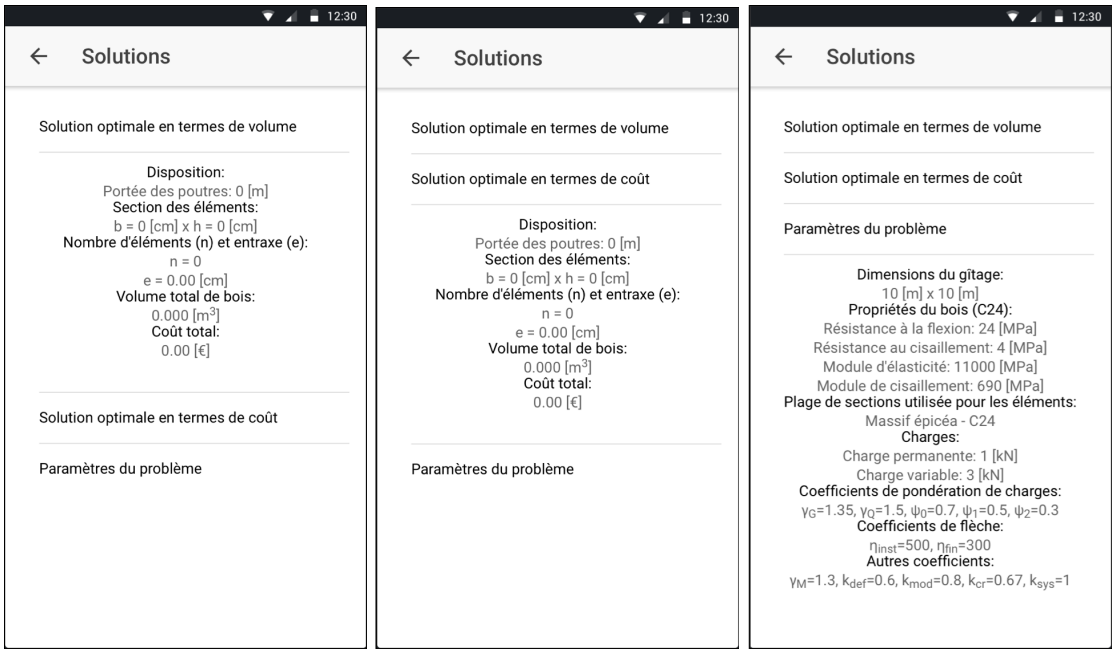
Affichage des solutions - aucune solution n'existe



1. Bouton permettant de fermer l'alerte et d'atterrir sur la page classique affichant les solutions (cfr. figure 5.34)

Note : un fois sur la page correspondant à la figure 5.34, les mêmes actions peuvent être effectuées. L'affichage des solutions est possible, mais tous les éléments seront nuls puisqu'il n'existe pas de solution (figures 5.37a et 5.37b). L'utilité de cette page est de pouvoir afficher les paramètres utilisés et vérifier qu'aucune erreur n'a été commise lors du remplissage du formulaire de calcul (figure 5.37c).

FIGURE 5.36 – Alerte dans le cas où aucune solution n'existe



(a) Affichage de la solution optimale en termes de volume

(b) Affichage de la solution optimale en termes de coût

(c) Affichage des paramètres utilisés pour le calcul

FIGURE 5.37 – Exemples d'écrans de la page des solutions dans le cas où aucune solution n'existe

5.5 Pistes d'enrichissement pour un développement futur

L'application développée dans le cadre de ce mémoire peut être considérée comme une version 1.0. Comme tout outil de travail, cette application peut être améliorée et étendue afin d'offrir une expérience plus confortable à l'utilisateur ou afin de fournir plus d'options, de résultats ou d'analyses de résultats. Ci-dessous, nous proposons diverses pistes d'améliorations qui pourraient être codées dans un développement futur de l'application (version 2.0).

Premièrement, ajouter des fonctionnalités permettant de raccourcir certaines opérations liées à l'encodage ou à la modification de plages de sections ferait gagner du temps à l'utilisateur et améliorerait son expérience d'utilisation. Nous proposons l'ajout des fonctionnalités suivantes pour la version 2.0 de l'application :

1. possibilité de modifier un champ, en particulier celui d'une largeur ou d'une hauteur ;
2. possibilité de fusionner deux plages de sections en une seule plage de sections ;
3. possibilité de copier/coller des plages, des largeurs ou des hauteurs entières.

Ensuite, on peut envisager d'élargir la base de données afin de prendre en compte un plus grand nombre de paramètres. Par exemple :

4. ajout d'un champ précisant, pour une certaine section, la longueur (portée) maximale de poutre ;
5. ajout d'un champ précisant le poids volumique des éléments issus d'une certaine plage de sections ;
6. ajout d'un champ précisant, pour chaque section, le poids du sabot qui devra être utilisé en plus de son prix ("poids par fixation" en plus de "prix par fixation") ;
7. ajout d'une base de données contenant les panneaux de plancher (par exemple, OSB) disponibles, avec des champs pour leur coût, l'entraxe maximal qu'ils permettent, leurs propriétés de résistance, etc.

Enfin, notamment à partir d'une base de données élargie, on pourrait étendre les options de calcul, les cas pris en compte et les solutions affichées :

8. possibilité de sélectionner plusieurs plages de sections de classes de résistance différentes pour le calcul des solutions ;
9. prise en compte, lors du calcul, de dispositions avec plusieurs poutres intermédiaires ;
10. prise en compte d'une disposition hyperstatique des éléments sur la poutre intermédiaire [10] ;
11. prise en compte des différents panneaux de plancher pour le calcul des solutions optimales en termes de coût et de volume :
    - utilisation de l'Annexe B de l'Eurocode 5 [11] ;
    - détermination de l'entraxe maximal en fonction du panneau considéré ;
    - prise en compte du coût (et volume) des panneaux dans le calcul du coût (et volume) total ;
12. indication, pour chaque solution, du rendement du gitage (cfr. chapitre 4) ;

10. Au lieu d'être assemblés à la poutre intermédiaire avec, par exemple, des sabots, les éléments reposeraient simplement dessus ; cela change néanmoins la répartition des efforts dans la poutre et il faudrait donc développer de nouveaux critères adimensionnels.

11. *Poutres assemblées mécaniquement* : méthode de calcul permettant de faire participer le panneau de plancher à la résistance du complexe gîtage-plancher.

13. prise en compte du poids du bois et des fixations et ajout d'une solution optimale en termes de poids ;
14. ajout d'une solution qui optimise un paramètre combinant le coût, le volume et éventuellement le poids du gîtage.

Le dernier point mérite plus de précisions. Nous pourrions, par exemple, définir un paramètre λ correspondant à une certaine solution qui dépendrait des différences de coût et de volume (et éventuellement de poids), exprimées en pourcentage, entre cette solution et les solutions optimales en termes de coût, de volume (et de poids). Par exemple, si on définit

- $\Delta c[\%]$ comme étant la différence de coût, en pourcentage, entre la solution en question et la solution optimale en termes de coût ;
- $\Delta v[\%]$ comme étant la différence de volume, en pourcentage, entre la solution en question et la solution optimale en termes de volume ;
- $\Delta p[\%]$ comme étant la différence de poids, en pourcentage, entre la solution en question et la solution optimale en termes de poids ;

alors on pourrait définir le paramètre λ comme suit :

$$\lambda = \Delta c^2 + \Delta v^2 + \Delta p^2 \tag{5.10}$$

Nous pourrions alors mettre en évidence la solution qui minimise ce paramètre λ et qui se rapproche donc le plus (au sens des moindres carrés [12]) de toutes les solutions optimales (en termes de coût, de volume et de poids) à la fois.

Dans un tout autre registre, on pourrait aussi envisager de traduire l'application dans d'autres langues (anglais, néerlandais, ...).

12. Méthode élaborée par Gausse et Legendre permettant d'approximer des données expérimentales en minimisant les écarts avec les mesures

Chapitre 6

Etude de cas de planchers et analyse des résultats

Dans ce chapitre, nous utilisons l'application développée précédemment afin d'analyser des cas concrets de planchers. Dans un premier temps, nous résolvons analytiquement un problème de plancher que nous avons préalablement défini, puis nous comparons la solution obtenue avec celles fournies par l'application. Dans un second temps, nous menons, grâce à l'application, des analyses comparatives sur base de différents cas.

En résumé, ce chapitre apporte des éléments probants quant à l'exactitude et à la pertinence de l'application.

6.1 Hypothèses sur les plages de sections

Pour effectuer cette démarche, nous avons besoin d'au moins une plage de sections et des coûts s'y rapportant.

Nous décidons de considérer deux plages de sections : l'une de bois massif, que nous utiliserons pour les éléments standards [1] du gîtage, et l'autre de bois lamellé-collé (BLC), que nous utiliserons pour l'éventuelle poutre intermédiaire.

6.1.1 Plage de sections en bois massif

On choisit d'utiliser une plage de sections homogène, avec 5 largeurs et, pour chacune de ces 5 largeurs, 7 hauteurs. Les sections retenues sont données au tableau 6.1. L'essence choisie est de l'épicéa (classe de résistance C24), largement répandue en Belgique.

Afin d'établir les coûts pour chaque section, qui sont également donnés au tableau 6.1, on utilise les données et hypothèses suivantes :

- le coût de chaque section par mètre courant est calculé sur base d'une estimation du coût moyen par unité de volume en Belgique pour des poutres en épicéa (C24), soit 500 $[€/m^3]$ [2] ;
- le coût supplémentaire par élément est considéré comme constant et égal à 10 [€] ;
- le coût des fixations est établi sur base de la gamme de sabots [3] à ailes extérieures de la marque SIMPSON Strong-Tie, et des prix proposés par le revendeur Eurabo [19].

1. On distingue les éléments standards de la poutre intermédiaire.

2. Donnée récoltée en décembre 2018 auprès du groupe RICHE, qui regroupe 3 entreprises spécialisées dans le bois, notamment dans la construction.

3. La norme concernant les sabots pour poutre en bois, rappelée par l'entreprise SIMPSON Strong-Tie, indique que "le flanc du sabot doit couvrir au minimum les 2/3 de la hauteur de l'élément porté, afin d'éviter le déversement de la poutre portée" [18].

Largeurs b [cm]	Hauteurs h [cm]	Prix/mètre [€]	Prix/pièce [€]	Prix/fixation [€]
3,6	7,0	1,26	10	2,31
	9,5	1,71	10	2,31
	12,0	2,16	10	2,31
	14,5	2,61	10	2,31
	17,0	3,06	10	2,31
	19,5	3,51	10	2,31
	22,0	3,96	10	3,58
4,7	7,0	1,65	10	2,52
	9,5	2,23	10	2,52
	12,0	2,82	10	2,52
	14,5	3,41	10	2,52
	17,0	4,00	10	2,52
	19,5	4,58	10	2,52
	22,0	5,17	10	3,59
6,3	7,0	2,21	10	2,31
	9,5	2,99	10	2,31
	12,0	3,78	10	2,31
	14,5	4,57	10	2,31
	17,0	5,36	10	2,31
	19,5	6,14	10	1,98
	22,0	6,93	10	2,37
7,2	7,0	2,52	10	3,58
	9,5	3,42	10	3,58
	12,0	4,32	10	3,58
	14,5	5,22	10	3,58
	17,0	6,12	10	3,58
	19,5	7,02	10	3,58
	22,0	7,92	10	3,58
9,0	7,0	3,15	10	3,93
	9,5	4,28	10	3,93
	12,0	5,40	10	3,93
	14,5	6,53	10	3,93
	17,0	7,65	10	3,93
	19,5	8,78	10	3,58
	22,0	9,90	10	3,58

Tableau 6.1 – Plage de sections de bois massif en épicéa (C24)

6.1.2 Plage de sections en BLC

Pour cette plage de sections, on se base sur les sections de BLC disponibles en épicéa (classe de résistance GL24h) dans le catalogue d'Eurabo [20] et leur prix par mètre courant. Le coût des fixations est établi sur la même base que pour la plage de sections de bois massif. De même, on considère à nouveau un coût supplémentaire par élément constant et égal à 10 [€]. Les sections et leurs coûts sont donnés au tableau 6.2.

6.2 Résolution d'un problème type

Dans cette section, nous allons calculer le gîtage nécessaire pour une plancher type dont nous aurons établi le contexte et les paramètres. Nous allons d'abord le calculer de manière "instinctive", comme nous l'aurions fait sans l'aide de l'application. Ensuite, nous comparons cette solution aux solutions optimales en termes de volume et en termes de coût données par l'application. Enfin, à l'aide de l'application, nous allons comparer les meilleures solutions pour différentes dispositions du gîtage, ainsi qu'en tenant compte de certaines contraintes architecturales éventuelles.

6.2.1 Description du problème

On suppose un plancher rectangulaire de 6 mètres sur 4 mètres dans une habitation unifamiliale. Le plancher étant réservé à un usage résidentiel, nous nous retrouvons dans la catégorie A du tableau A.2. De plus, le plancher se situant à l'intérieur, il sera de classe de service 1.

On suppose qu'on utilise des panneaux OSB3 de 22$[mm]$ d'épaisseur. L'entraxe maximal doit dès lors être limité à 70$[cm]$ [4].

On suppose que le poids propre de l'ensemble du plancher (panneaux OSB, revêtement, panneaux d'isolation, plafond suspendu) équivaut à 1$[kN/m^2]$ (environ 100$[kg/m^2]$).

Enfin, on souhaite limiter la flèche instantanée des poutres à $L/500$ et leur flèche finale à $L/300$.

6.2.2 Résolution analytique

Instinctivement, nous prenons l'entraxe le plus grand possible, qui est de $66,67[cm]$ et qui correspond à 10 poutres. En effet, si on retirait une poutre, on aurait un entraxe égal à 75$[cm]$, ce qui dépasse le maximum de 70$[cm]$.

Les combinaisons de charges ELU et ELS sont, étant donné que $G = 1[kN/m^2]$, que $Q = 2[kN/m^2]$ (cfr. tableau A.2) et que $k_{def} = 0,6$ (cfr. tableau A.3) :

$$Q_{ELU} = 1,35 \times G + 1,50 \times Q = 4,35[kN/m^2] \tag{6.1}$$

$$Q_{ELS,inst} = G + Q = 3,00[kN/m^2] \tag{6.2}$$

$$Q_{ELS,fin} = (1/k_{def}) \times G + Q = 3,60[kN/m^2] \tag{6.3}$$

4. Selon les abaques de charge sur panneaux OSB du fabriquant Swiss Krono [21].

Largeurs b [cm]	Hauteurs h [cm]	Prix/mètre [€]	Prix/pièce [€]	Prix/fixation [€]
4,5	10,0	3,10	10	2,52
	30,0	9,29	10	3,59
	36,0	11,15	10	10,01
	40,0	12,38	10	10,01
	45,0	13,93	10	10,01
8,0	16,0	8,81	10	3,46
	20,0	11,01	10	3,46
	24,0	12,06	10	3,46
	28,0	15,41	10	7,77
	32,0	17,61	10	8,16
	36,0	19,81	10	8,16
10,0	20,0	13,76	10	2,96
	32,0	22,02	10	9,91
	36,0	24,77	10	9,91
	40,0	27,52	10	9,91
	44,0	30,27	10	9,91
	46,0	31,65	10	9,91
12,0	20,0	16,51	10	9,55
	24,0	19,81	10	9,55
	28,0	23,12	10	9,91
	32,0	26,42	10	9,91
	36,0	19,72	10	9,91
	40,0	33,02	10	9,91
14,0	16,0	15,41	10	3,59
	24,0	23,12	10	3,59
	28,0	26,97	10	9,91
	30,0	29,39	10	9,91
	36,0	33,06	10	9,91
	40,0	38,53	10	9,91
	44,0	42,38	10	21,78
	46,0	44,31	10	21,78
	50,0	47,76	10	21,78
	60,0	55,10	10	21,78
16,0	16,0	17,61	10	7,34
	28,0	30,82	10	7,34
	30,0	35,23	10	11,30
	36,0	39,63	10	11,30
18,0	32,0	39,63	10	16,55
	52,0	64,40	10	36,70
20,0	20,0	27,52	10	11,30
	32,0	44,03	10	11,30
	44,0	60,54	10	31,31
	46,0	63,30	10	31,31
	52,0	71,55	10	31,31
22,0	24,0	36,33	10	20,02
	28,0	42,38	10	20,02
	36,0	54,49	10	20,02
	52,0	78,71	10	37,31

Tableau 6.2 – Plage de sections de bois lamellé-collé en épicéa (GL24h)

Commençons par le critère ELU de résistance à la flexion (cfr. équation 2.16). Il faut que :

$$\frac{q_{ELU}L^2/8}{bh^2/6} \leqslant k_{mod}k_h k_{sys}\frac{f_{m,k}}{\gamma_M} \tag{6.4}$$

Dans l'équation 6.4, la charge q_{ELU} est donnée en $[kN/m]$. Elle vaut :

$$q_{ELU} = Q_{ELU} \times e = 2,9[kN/m] \tag{6.5}$$

La résistance caractéristique à la flexion $f_{m,k}$ vaut, pour un bois de classe C24, $24[MPa]$ (cfr. A.8). Le paramètre k_{mod} vaut $0,8$ (cfr. tableau A.4). Comme nous tenons compte de l'effet système, $k_{sys} = 1,1$. Comme nous ne connaissons pas encore la hauteur h des éléments, nous prenons $k_h = 1$, ce qui est de toute manière plus sécuritaire. Enfin, comme nous travaillons avec du bois massif, on a $\gamma_M = 1,3$ (cfr. tableau A.6).

Puisque $L = 4[m]$, l'équation 6.4 devient :

$$\boxed{\frac{34,8}{bh^2} \leqslant 16246,154} \tag{6.6}$$

avec b et h en mètres.

Ensuite, nous faisons de même pour le critère ELU de résistance au cisaillement. Pour rappel, il est donné par l'équation 2.20. Il faut que :

$$\frac{3q_{ELU}L/2}{2k_{cr}bh} \leqslant k_{mod}k_{sys}\frac{f_{v,k}}{\gamma_M} \tag{6.7}$$

La résistance caractéristique au cisaillement $f_{v,k}$ vaut, pour un bois de classe C24, $4[MPa]$ (cfr. A.8) et le facteur de fissuration k_{cr} vaut $0,67$. Les autres paramètres ont déjà été évoqués ci-dessus. L'équation 6.7 devient :

$$\boxed{\frac{12,985}{bh} \leqslant 2707,7} \tag{6.8}$$

avec b et h en mètres.
Le critère ELS de flèche instantanée, donné à l'équation 2.23, est :

$$\frac{5q_{ELS,inst}L^4}{384EI_y} + \frac{q_{ELS,inst}L^2}{8GA_v} \leqslant \frac{L}{\eta_{inst}} \tag{6.9}$$

où $I_y = bh^3/12$ et $A_v = 5bh/6$.
Dans l'équation 6.9, la charge $q_{ELS,inst}$ est donnée en $[kN/m]$. Elle vaut :

$$q_{ELS,inst} = Q_{ELS,inst} \times e = 2,0[kN/m] \tag{6.10}$$

Le module d'élasticité E vaut, pour un bois de classe C24, $11000[MPa]$, tandis que le module de cisaillement G vaut $690[MPa]$ (cfr. A.8). Les autres paramètres ont déjà été évoqués ci-dessus. Le critère ELS de flèche instantanée devient, après calcul :

$$\boxed{7,31 \cdot 10^{-6}\frac{1}{bh^3} + 6,99 \cdot 10^{-6}\frac{1}{bh} \leq 8 \cdot 10^{-3}} \tag{6.11}$$

avec b et h en mètres.

Enfin, on fait de même avec le critère ELS de la flèche finale, donné par l'équation 2.27 :

$$\frac{q_{ELS,fin}}{q_{ELS,inst}}\left(\frac{5q_{ELS,inst}L^4}{384EI_y}+\frac{q_{ELS,inst}L^2}{8GA_v}\right)\leqslant\frac{L}{\eta_{fin}} \tag{6.12}$$

Le membre de gauche correspond au membre de gauche de l'équation 6.9, multiplié par k_{fin} qui vaut :

$$k_{fin}=\frac{q_{ELS,fin}}{q_{ELS,inst}}=\frac{Q_{ELS,fin}\cdot e}{Q_{ELS,inst}\cdot e}=1,2 \tag{6.13}$$

Finalement, l'équation 6.12 devient :

$$\boxed{8,77\cdot10^{-6}\frac{1}{bh^3}+8,39\cdot10^{-6}\frac{1}{bh}\leq13,3\cdot10^{-3}} \tag{6.14}$$

avec b et h en mètres.

En vérifiant ces quatre critères (équations 6.6, 6.8, 6.11 et 6.14) pour chaque section du tableau 6.1, on arrive à la conclusion qu'une seule section satisfait aux quatre critères en même temps. Il s'agit de la section $b=9,0[cm]\times h=22,0[cm]$.

En conclusion, nous devrons disposer 10 poutres de section $b=9,0[cm]\times h=22,0[cm]$, d'une portée de $4[m]$, et espacées de $67[cm]$ [5]. Nous pouvons calculer le volume total de bois que cela représente :

$$Vol=10\times0,09[m]\times0,22[m]\times4[m]=0,792\ m^3 \tag{6.15}$$

Sur base du tableau 6.1, nous pouvons également estimer le coût de ce gîtage :

$$Coût=(10\times9,90[€/m]\times4[m])+(10\times10,00[€/poutre])+(2\times10\times3,58[€/fixation])=567,60\ € \tag{6.16}$$

Dans la suite, nous ferons référence à cette solution en l'appelant la *solution instinctive*.

6.2.3 Résolution à l'aide de l'application

Solutions optimales

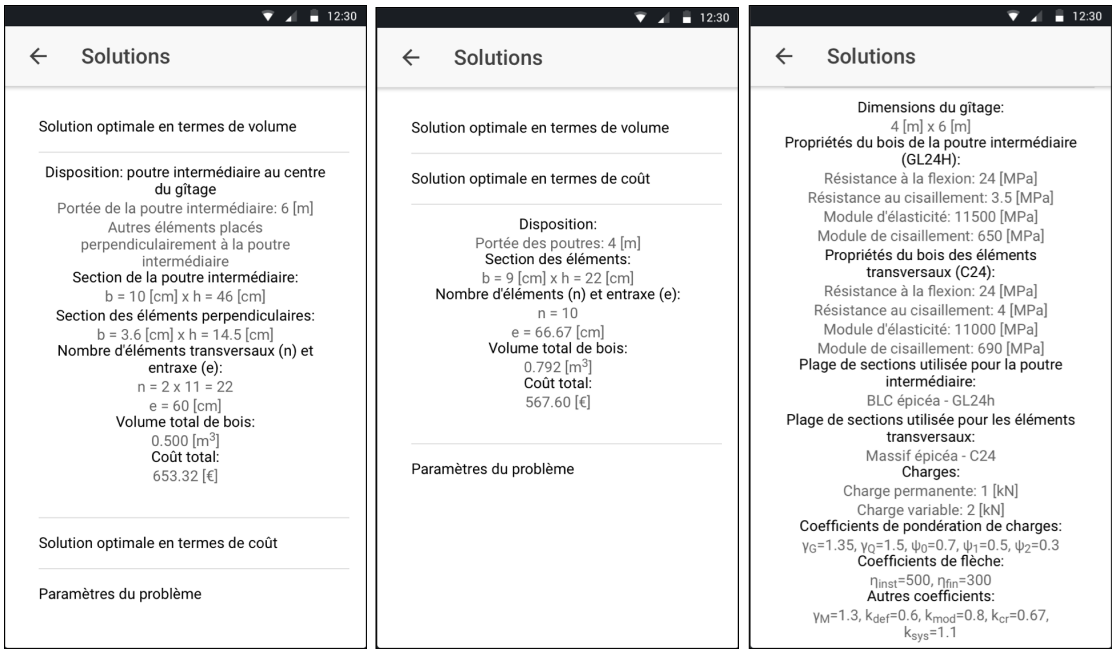
Ici, nous utilisons l'application que nous avons développée afin d'obtenir les solutions optimales en termes de volume et en termes de coût pour le problème décrit. Les paramètres évoqués ci-dessus ne changent pas. On choisit la plage de sections donnée au tableau 6.1 et, pour l'éventuelle poutre intermédiaire, la plage de sections du tableau 6.2. Les résultats fournis par l'application, ainsi que le rappel des paramètres utilisés, sont montrés à la figure 6.1.

La solution optimale en termes de volume consiste à disposer une poutre intermédiaire (en BLC) de section $b=10,0[cm]\times h=46,0[cm]$ ayant une portée de $6[m]$ au centre du gîtage, et de disposer, de chaque côté, 11 poutres (en bois massif) de section $b=3,6[cm]\times h=14,5[cm]$, d'une portée de $1,95[m]$ et espacées de $60[cm]$ [6] (cfr. figure 6.2a). Le volume total de bois utilisé par cette solution et son coût sont :

- Volume total : $0,5\ [m^3]$
- Coût total : $653,32\ [€]$

5. Pour rappel, l'entraxe est la distance entre les axes de deux poutres voisines. De plus, il y a une poutre en plus que d'espacements. L'espacement réel entre deux poutres sera donc ici de $(600-10\cdot9)/(10-1)=56,67[cm]$.

6. Espacement réel $=(600-11\cdot3,6)/(11-1)=56,04[cm]$



(a) Solution optimale en termes de volume

(b) Solution optimale en termes de coût

(c) Rappel des paramètres utilisés pour le calcul

FIGURE 6.1 – Captures d'écrans de la page de solutions de l'application

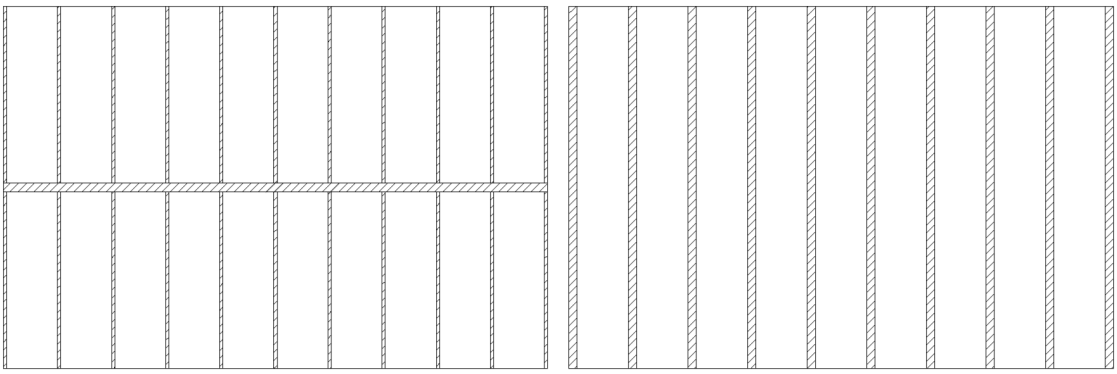
Par rapport à la solution instinctive, la solution optimale en termes de volume permet d'économiser $0,292[m^3]$ de bois, ce qui représente quasi 37% de moins. Cela correspond à une économie d'environ $130[kg]$ de bois si on considère un poids volumique moyen de $450[kg/m^3]$ pour l'épicéa. L'économie non négligeable réalisée sur le volume de bois nous coûterait tout de même $85,72[€]$ de plus par rapport à la solution optimale en termes de coût, soit un sur-coût d'un peu plus de 15%. Cela montre, dans notre cas, l'importance des coûts non proportionnels au volume bois.

La solution optimale en termes de coût ne fait quant à elle pas usage d'une poutre intermédiaire et consiste à disposer 10 poutres de section $b = 9,0[cm] \times h = 22,0[cm]$, d'une portée de $4[m]$ et espacées de $67[cm]$ [7] (cfr. figure 6.2b). Le volume total de bois utilisé par cette solution et son coût sont :

- Volume total : $0,792$ $[m^3]$
- Coût total : $567,60$ $[€]$

La solution optimale en termes de coût correspond ici à la solution instinctive. Cela est cependant très difficile à prévoir sans l'aide de l'application. En effet, dans notre cas, l'application a testé 472 possibilités avant de tirer ses conclusions, ce qui nous aurait mis un temps considérable si nous avions dû le faire analytiquement.

7. Espacement réel $= (600 - 10 \cdot 9)/(10 - 1) = 56,67[cm]$



(a) Solution optimale en termes de volume (b) Solution optimale en termes de coût

FIGURE 6.2 – Schéma des solutions optimales fournies par l'application

Solutions optimales pour différentes dispositions du gîtage

Sur base d'une version légèrement modifiée [8] de l'application qui permet d'afficher les solutions optimales pour chaque disposition possible du gîtage (sens long/sens court, avec/sans poutre intermédiaire), nous allons comparer les solutions optimales pour chacune des dispositions. Nous utilisons exactement les mêmes paramètres que précédemment.

Les solutions optimales en termes de volume pour chaque disposition prise en compte par l'application sont données dans le tableau 6.3. La $2^{\text{ème}}$ colonne donne le volume minimal qu'il est possible d'atteindre pour la disposition du gîtage, tandis que la $4^{\text{ème}}$ colonne donne le coût que cette solution représente. Les $3^{\text{ème}}$ et $5^{\text{ème}}$ colonnes permettent de comparer le volume et le coût de la solution au volume de la solution optimale en termes de volume et au coût de la solution optimale en termes de coût respectivement.

Solutions optimales en termes de volume				
	Volume $[m^3]$	Diff. [%]	Coût [€]	Diff. [%]
Sans poutre intermédiaire Sens court	0,760	52,0	792,00	39,5
Sans poutre intermédiaire Sens long	2,420	384,0	1879,80	231,2
Avec poutre intermédiaire dans le sens court	0,518	3,6	590,38	4,0
Avec poutre intermédiaire dans le sens long	0,500	0,0	653,32	15,1

TABLEAU 6.3 – Solution optimale en termes de volume pour les différentes dispositions et comparaison à la solution optimale globale en termes de volume

Les solutions optimales en termes de coût pour chaque disposition prise en compte par l'application sont données dans le tableau 6.4. La $4^{\text{ème}}$ colonne donne le coût minimal qu'il est possible d'atteindre pour la disposition du gîtage, tandis que la $2^{\text{ème}}$ colonne donne le volume de bois correspondant à cette solution. Les $3^{\text{ème}}$ et $5^{\text{ème}}$ colonnes permettent de comparer le volume et le coût de la solution au volume de la solution optimale en termes de volume et au coût de la solution optimale en termes de coût respectivement.

8. Spécifiquement pour ce chapitre.

Solutions optimales en termes de coût				
	Volume $[m^3]$	Diff. [%]	Coût [€]	Diff. [%]
Sans poutre intermédiaire Sens court	0,792	58,4	567,60	0,0
Sans poutre intermédiaire Sens long	2,495	399,0	1607,76	183,3
Avec poutre intermédiaire dans le sens court	0,652	30,4	577,92	1,8
Avec poutre intermédiaire dans le sens long	0,514	2,8	631,46	11,3

TABLEAU 6.4 – Solution optimale en termes de coût pour les différentes dispositions et comparaison à la solution optimale globale en termes de coût

On remarque que, si nous étions contraints de placer les poutres selon le sens long du plancher pour une quelconque raison architecturale, cela nous coûterait au minimum 1040€ de plus que le gîtage le moins cher, soit quasi trois fois plus. Une telle contrainte nous obligerait aussi à utiliser quasiment le quintuple de bois par rapport à la solution la moins volumineuse.

On retiendra aussi que la solution avec une poutre intermédiaire placée dans le sens court ne nous coûterait que 4% de plus que la solution la moins chère, et n'utiliserait que 3,6% de plus que la solution la moins volumineuse. Cette solution permet à la fois de se rapprocher du coût minimum atteignable et du volume minimum atteignable pour ce problème, et constituerait donc une bonne option. Dans sa version actuelle, l'application ne permet pas de déterminer la solution qui se rapproche le plus de ces deux optimums. Comme nous l'avions suggéré à la dernière section du chapitre précédent, il serait possible d'ajouter une telle fonction à l'application.

Application du concept de volume minimal théorique

Pour rappel, dans le chapitre 4, nous avions évoqué le concept de volume minimal théorique de bois nécessaire pour un gîtage (dans une disposition simple), qui vaut (équation 4.31) :

$$Vol_{min,th} = W_{\frac{b}{e}\to 0;n\to\infty} \times (LP)^{\frac{3}{2}} \tag{6.17}$$

Nous calculons la valeur de W sur base de l'équation 4.20, toujours pour le même problème défini ci-dessus, pour un rapport b/e infinitésimal et un nombre n très grand, et pour $\beta = 4/6 < 1$ (sens court). En la multipliant par la portée des poutres (ici, $4[m]$) au cube, on obtient :

$$Vol_{min,th} = 0,1726[m^3] \tag{6.18}$$

Ce volume minimal théorique correspond, comme nous l'avons vu au chapitre 4, à un très grand nombre de poutres infiniment fines et infiniment hautes, espacées au maximum. Dans la pratique, il sera impossible d'atteindre ce volume, mais il nous permet de calculer le rendement du gîtage, concept que nous avions également défini au chapitre 4 (équation 4.32). Celui-ci vaut, dans le cas de la solution instinctive, qui est aussi la solution optimale en termes de coût :

$$\mu = \frac{Vol_{min,th}}{Vol_{sol}} = \frac{0,1726}{0,792} = 21,8\% \tag{6.19}$$

Autrement dit, la solution instinctive sera $1/\mu = 4,6$ fois plus volumineuse que le volume minimal théorique.

En comparaison, le rendement de la solution optimale en termes de volume pour ce problème vaut 34,5%, et celui de la solution optimale en termes de volume pour une disposition des poutres dans le sens long est de seulement 7,1%.

Prise en compte d'une contrainte architecturale

Admettons que nous sommes contraints d'avoir recours à un gîtage dont l'épaisseur doit être inférieure à 20$[cm]$. L'application nous permet d'entrer une valeur maximale que la hauteur des poutres ne peut pas dépasser.

La solution optimale en termes de coût consisterait à disposer 14 poutres de section $b = 9,0[cm] \times h = 19,5[cm]$, d'une portée de 4$[m]$ et espacées de $46,15[cm]$ [9]. Le volume total de bois utilisé par cette solution et son coût sont :

- Volume total : $0,983$ $[m^3]$
- Coût total : $731,92$ [€]

Comparée à la solution optimale en termes de coût sans contrainte sur l'épaisseur du plancher, le fait d'imposer une hauteur maximale nous coûterait 29% de plus.

La solution optimale en termes de volume consisterait à disposer 33 poutres de section $b = 3,6[cm] \times h = 19,5[cm]$, d'une portée de 4$[m]$ et espacées de $18,75[cm]$ [10]. Le volume total de bois utilisé par cette solution et son coût sont :

- Volume total : $0,927$ $[m^3]$
- Coût total : $945,78$ [€]

Cela représente plus de 85% de volume de bois supplémentaire par rapport à la solution optimale en termes de volume sans contrainte sur l'épaisseur du plancher. Le fait d'imposer une hauteur maximale nous oblige donc à utiliser presque le double de bois.

Cette solution optimale en termes de volume nous impose cependant de placer les éléments très proches les uns des autres, avec un espacement réel entre les poutres qui n'est que de 15,04$[cm]$. Cela pourrait possiblement nous gêner si les ailes de nos sabots de fixation sont trop larges, par exemple. On pourrait, pour éviter ce problème, imposer un entraxe minimum entre les éléments de, par exemple, 25$[cm]$. La solution optimale en termes de volume consiste alors, dans ce cas, à disposer 17 poutres de section $b = 7,2[cm] \times h = 19,5[cm]$, d'une portée de 4$[m]$ et espacées de $37,50[cm]$ [11]. Le volume total de bois utilisé par cette solution et son coût sont :

- Volume total : $0,955$ $[m^3]$
- Coût total : $769,08$ [€]

Cette contrainte supplémentaire nous oblige à consommer encore 3% de bois en plus, ce qui reste très raisonnable. De plus, on remarque que cette solution se rapproche de la solution optimale en termes de coût.

9. Espacement réel $= (600 - 14 \cdot 9)/(14 - 1) = 36,46[cm]$
10. Espacement réel $= (600 - 33 \cdot 3,6)/(33 - 1) = 15,04[cm]$
11. Espacement réel $= (600 - 17 \cdot 7,2)/(17 - 1) = 29,85[cm]$

6.3 Comparaison graphique des solutions pour différentes dimensions de plancher

Dans cette section, nous allons utiliser notre application afin de pouvoir tracer plusieurs graphiques représentant des rapports entre les coûts ou entre les volumes de différentes solutions en fonction des dimensions du plancher.

Pour cela, on se base sur le problème traité à la section 6.2. Tous les paramètres que nous utilisons dans cette section sont identiques à ceux de ce problème, hormis une des dimensions du plancher : nous gardons une dimension fixe ($4[m]$), et faisons varier l'autre dimension entre 2 et 10 mètres (on l'appellera par la suite la *dimension variable*). Cela nous permet de tracer les graphiques désirés et d'en déduire certaines tendances. L'axe des abscisses de ces graphiques correspond à la dimension variable.

Comme ces tendances sont déduites pour un exemple chiffré de problème, elles ne peuvent pas être considérées comme des règles générales. Elles servent à mettre en évidence certaines observations que l'on peut faire pour cet exemple de problème type, avec des plages de sections types.

6.3.1 Rendement des solutions

Le graphique représenté à la figure 6.3 illustre le rendement des solutions optimales pour chaque dimension de plancher en fonction de la dimension variable (l'autre dimension étant, pour rappel, fixée à $4[m]$). Rappelons que le concept rendement pour un gîtage a été défini par l'équation 4.32.

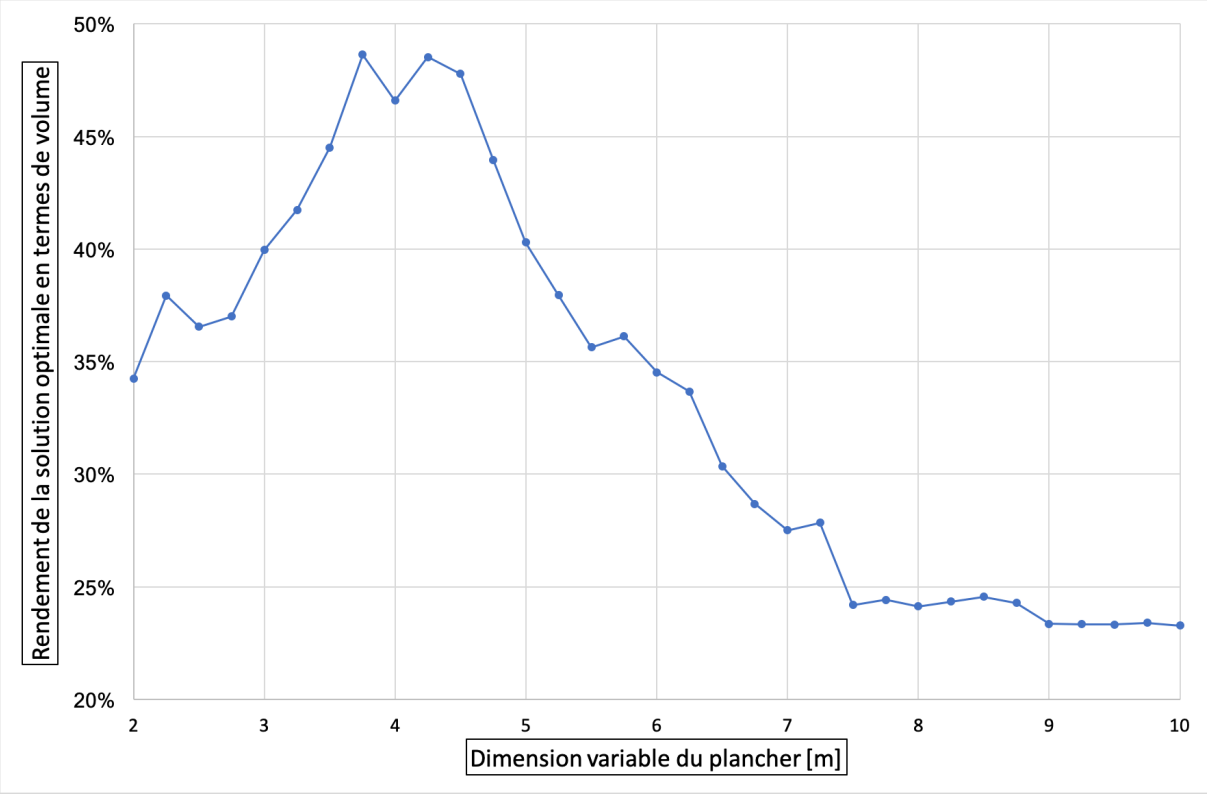


FIGURE 6.3 – Rendement de la solution optimale en termes de volume en fonction d'une des dimensions du plancher

On remarque que, pour les paramètres et les plages de sections considérées dans ce chapitre, on obtient le meilleur rendement pour un plancher se rapprochant d'un carré. A ce moment là, le volume minimal qu'il est possible d'atteindre avec nos plages de sections vaut à peine plus que le double du volume minimal théorique. A l'inverse, on remarque que, lorsque la différence entre les deux dimensions du plancher grandit, le volume minimal réellement atteignable s'écartera bien plus du volume minimal théorique.

6.3.2 Comparaison du coût des solutions optimales

Nous allons comparer, en fonction de la dimension variable, la différence entre le coût de la solution optimale en termes de volume et entre celui de la solution optimale en termes de coût. Autrement dit, nous allons, pour chaque valeur considérée pour la dimension variable, calculer l'économie que permet de faire la solution optimale en termes de coût sur la solution optimale en termes de volume, exprimée en pourcentage. Cette économie se calcule comme suit :

$$\text{économie}_{€}[\%] = \frac{\text{coût}_{optiV} - \text{coût}_{optiC}}{\text{coût}_{optiC}} \tag{6.20}$$

où coût_{optiV} est le coût de la solution optimale en termes de volume, et coût_{optiC} le coût de la solution optimale en termes de coût (et donc le coût minimal du gîtage).

Le graphique représenté à la figure 6.4 illustre cette économie en fonction de la dimension variable.

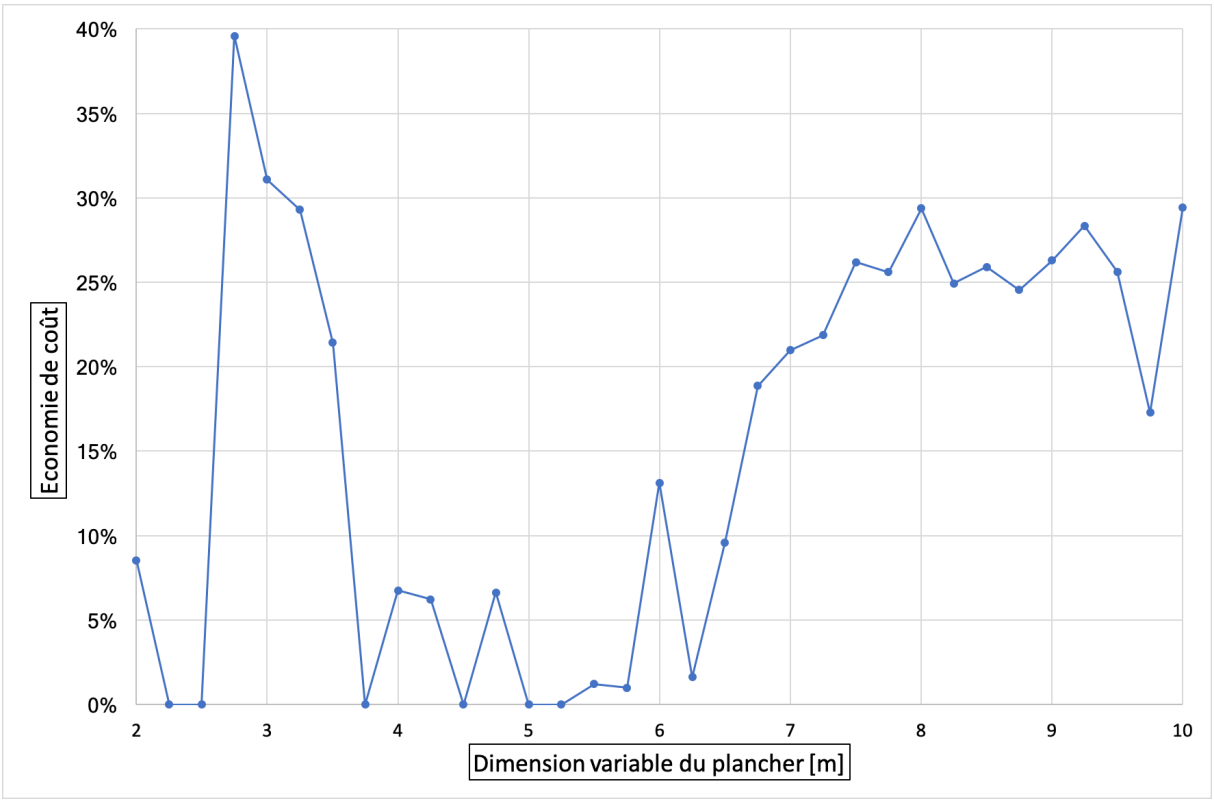


FIGURE 6.4 – Economie de coût pour la solution optimale en termes de coût par rapport au coût de la solution optimale en termes de volume

Logiquement, on remarque que cette économie est toujours positive, puisque par définition il n'y a pas de solution ayant un coût plus faible que la solution optimale en termes de coût. On

constate cependant que, dans certains cas, la solution optimale en termes de volume est identique à la solution optimale en termes de coût.

L'allure très erratique de ce graphe illustre le fait que le choix des plages de sections et les coûts associés à celles-ci détermineront entièrement l'allure de ce graphique.

6.3.3 Comparaison du volume des solutions optimales

Ici, c'est la différence entre le volume de la solution optimale en termes de coût et entre celui de la solution optimale en termes de volume que nous allons comparer en fonction de la dimension variable. Autrement dit, nous allons, pour chaque valeur considérée pour la dimension variable, calculer l'économie sur le volume que permet de faire la solution optimale en termes de volume sur la solution optimale en termes de coût. Cette économie se calcule comme suit :

$$\text{économie}_{vol}[\%] = \frac{\text{volume}_{optiC} - \text{volume}_{optiV}}{\text{volume}_{optiV}} \tag{6.21}$$

où volume_{optiV} est le volume de la solution optimale en termes de volume (et donc le volume minimal du gîtage), et volume_{optiC} le volume de la solution optimale en termes de coût.

Le graphique représenté à la figure 6.5 illustre cette économie de volume en fonction de la dimension variable.

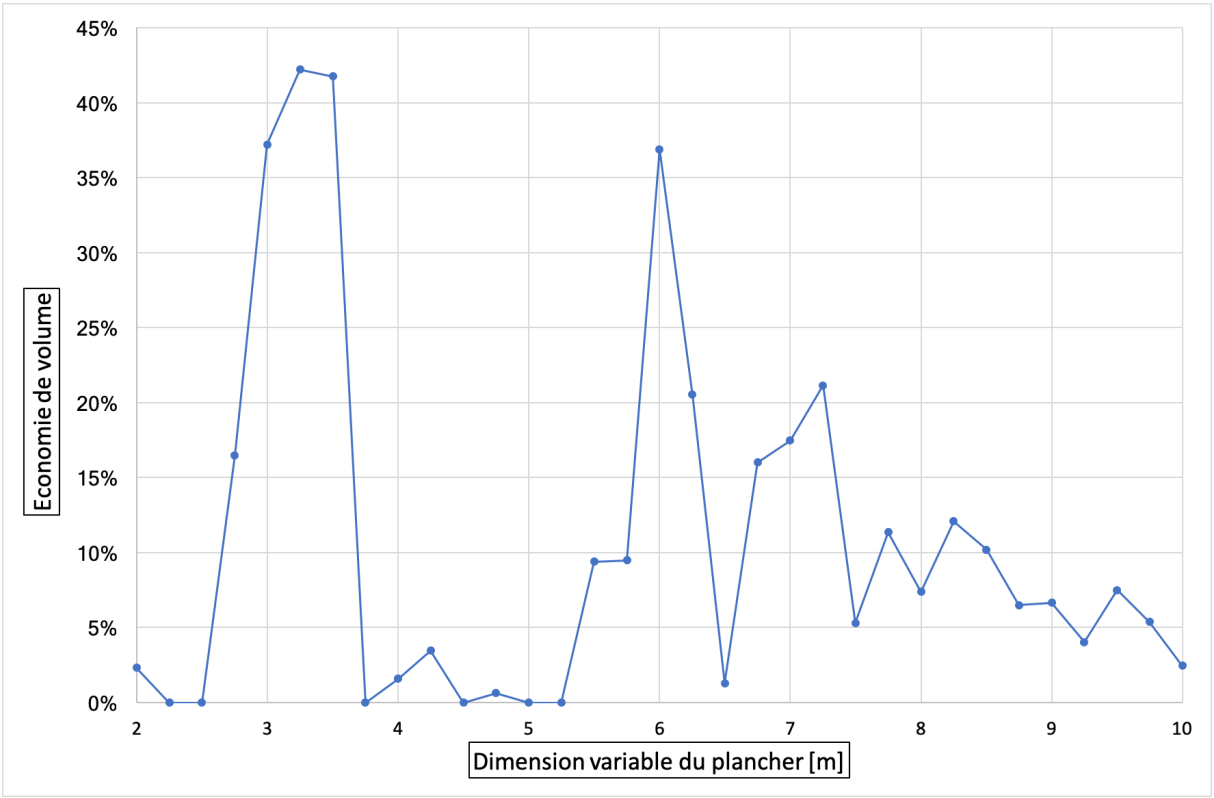


FIGURE 6.5 – Economie de volume pour la solution optimale en termes de volume par rapport au volume de la solution optimale en termes de coût

Les observations faites pour la figure 6.4 sont valables ici aussi. Il est d'ailleurs intéressant de remarquer que les deux graphiques ont grosso modo la même allure (avant $7[m]$), ce qui veut dire que lorsque la solution optimale en termes de coût est financièrement beaucoup plus

avantageuse que celle en termes de volume, la solution optimale en termes de volume sera aussi bien plus avantageuse que celle en terme de coût en ce qui concerne le volume de bois mis en œuvre. Autrement dit, si l'écart de coût entre les deux solutions est important, l'écart de volume le sera aussi, et inversement.

Il est important de souligner que cette conclusion ne sera pas non plus nécessairement valable si on change de plage de sections ou que l'on modifie les coûts, par exemple.

6.3.4 Comparaison des solutions optimales pour le sens court et le sens long

Dans cette section, nous allons comparer, pour un gîtage sans poutre intermédiaire, les solutions optimales dans le cas où les éléments sont disposés selon le sens court par rapport au cas où ils le sont selon le sens long (ici encore, en fonction de la dimension variable).

Autrement dit, nous allons, pour chaque valeur considérée pour la dimension variable, calculer l'économie sur le coût (équation 6.22) et sur le volume (équation 6.23) que nous faisons si nous plaçons les éléments selon le sens court par rapport au sens long.

$$\text{économie}_{€}[\%] = \frac{\text{coût}_{long} - \text{coût}_{court}}{\text{coût}_{court}} \tag{6.22}$$

$$\text{économie}_{vol}[\%] = \frac{\text{volume}_{long} - \text{volume}_{court}}{\text{volume}_{court}} \tag{6.23}$$

où volume_{court} et coût_{court} sont respectivement le volume minimum et le coût minimum qu'on puisse atteindre pour le plancher donné si nous disposons les éléments selon le sens court, et volume_{long} et coût_{long} sont respectivement le volume minimum et le coût minimum qu'on puisse atteindre pour le même plancher si nous disposons les éléments selon le sens long.

Le graphique représenté à la figure 6.6 illustre cette économie de coût en fonction de la dimension variable, tandis que le graphique représenté à la figure 6.7 illustre l'économie de volume qui est réalisée.

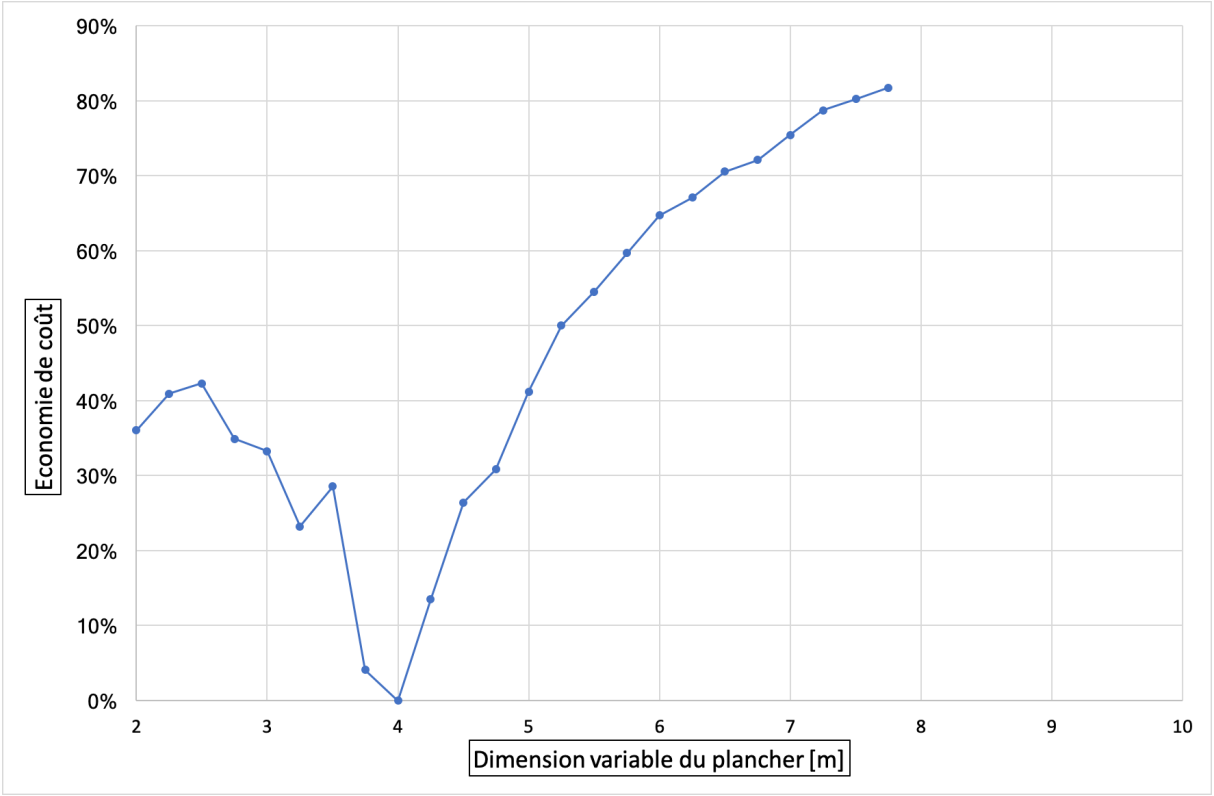


FIGURE 6.6 – Economie de coût entre le coût minimal pour un gîtage disposé dans le sens court part rapport à un gîtage disposé dans le sens long

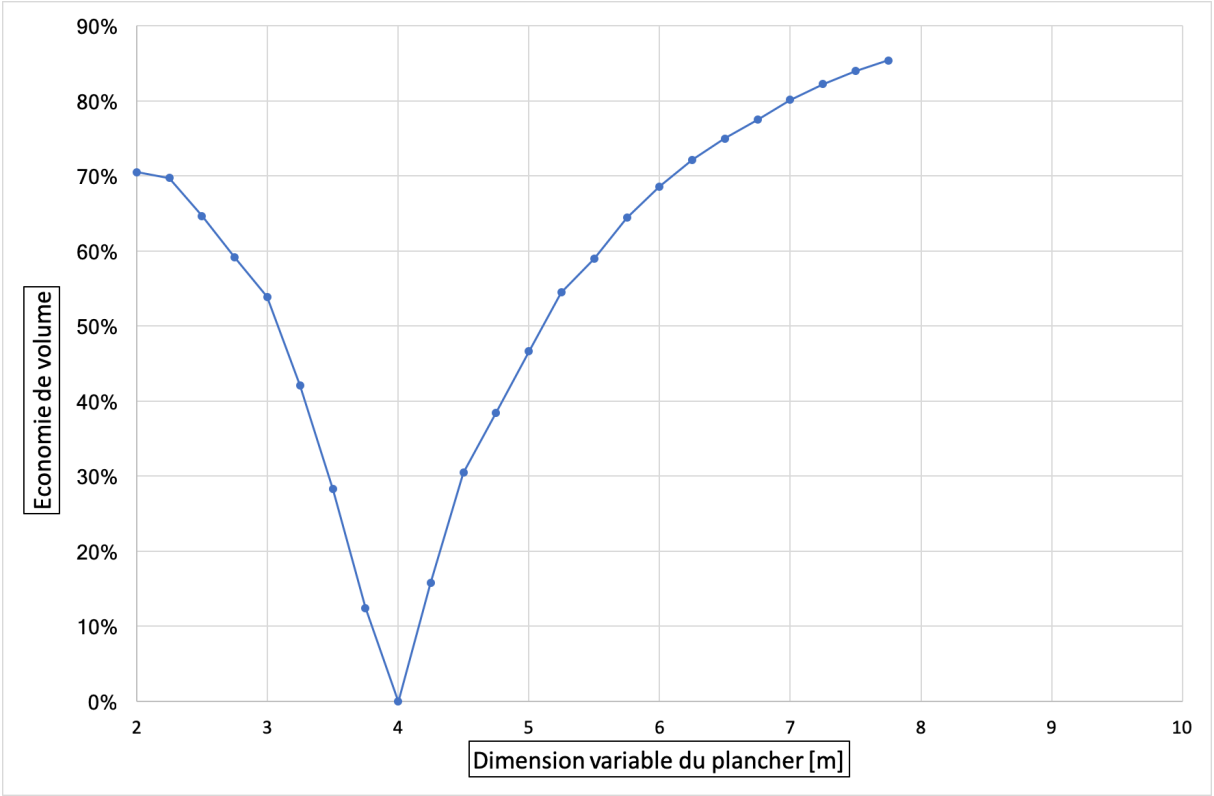


FIGURE 6.7 – Economie de volume entre le volume minimal pour un gîtage disposé dans le sens court part rapport à un gîtage disposé dans le sens long

En observant ces deux graphiques, on peut faire les remarques suivantes.

- Dans les deux cas, l'économie ne sera jamais négative. Cela signifie que, pour les paramètres de notre problème et pour la plage de sections considérée, il ne sera jamais plus avantageux d'opter pour une solution dans le sens long. Et cela, ni en termes de coût, ni en termes de volume de bois.
- Logiquement, on constate que, dans les deux cas, l'économie est nulle dans le cas d'un plancher carré (dans notre cas, $4[m]$ sur $4[m]$) puisqu'il n'y a pas de côté plus long/court que l'autre. On remarque aussi qu'au plus on se rapproche de cette configuration, au moins la différence de coût ou de volume sera importante. On peut raisonnablement supposer que cette observation restera valable pour d'autres paramètres du problème ou pour d'autres plages de sections.
- L'économie, de coût ou de volume, augmente très vite dès qu'on s'écarte un petit peu de la configuration carrée, puis se stabilise lorsque le rapport entre les deux dimensions du plancher devient important (ou faible, selon que la grande dimension soit au numérateur ou au dénominateur). Lorsque la grande dimension devient trop importante (ici, à partir de $7{,}75[m]$), il n'existera plus de solution permettant de placer les poutres dans le sens long et sans faire appel à une poutre intermédiaire. Cette limite dépend bien entendu de la plage de sections que l'on considère.

6.3.5 Comparaison des solutions optimales des cas avec et sans poutre intermédiaire

Enfin, nous allons effectuer les mêmes types de comparaisons pour une disposition simple du gîtage d'une part (sans poutre intermédiaire), et une disposition du gîtage avec une poutre intermédiaire d'autre part.

Premièrement, pour chaque valeur considérée pour la dimension variable, nous allons calculer l'économie sur le coût (équation 6.24) que nous réalisons si nous optons pour une disposition simple par rapport à une disposition avec poutre intermédiaire (quel que soit le sens de la portée des éléments ou de la poutre intermédiaire). Cette économie est représentée à la figure 6.8, en fonction de la dimension variable.

$$\text{économie}_{€}[\%] = \frac{\text{coût}_{inter} - \text{coût}_{simp}}{\text{coût}_{simp}} \tag{6.24}$$

où coût_{inter} est le coût minimum qu'on puisse atteindre pour le plancher donné si nous utilisons une poutre intermédiaire, et coût_{simp} le coût minimum qu'il est possible d'atteindre pour le même plancher si nous n'utilisons pas de poutre intermédiaire.

On peut faire les observations suivantes sur le graphique présenté à la figure 6.8.

- On remarque ici, qu'opter pour une disposition simple du gîtage (c'est-à-dire lorsque l'économie calculée à l'équation 6.24 est positive) est préférable dans deux cas :
  - pour des valeurs inférieures à $4[m]$ pour la dimension variable, car peu de très petites sections suffiront à porter le plancher et l'ajout d'une poutre intermédiaire ne fera qu'ajouter des éléments et des coûts inutiles, ne serait-ce qu'en termes de fixations ;
  - pour de grandes valeurs de la dimension variable (ici, supérieures à $7[m]$), car la poutre intermédiaire devrait avoir une section assez importante et que cela coûte cher ; de plus, dans notre plage de sections, le choix de grandes sections de BLC est limité et il faut donc parfois opter inutilement pour des sections bien plus grandes et plus coûteuses que la section minimale qui suffirait.

- Malgré ces observations, l'économie qu'il est possible de faire entre les deux dispositions dépendra au cas par cas des problèmes et des plages de sections disponibles ; il est donc difficile de prévoir quelle sera la meilleure solution pour un cas donné sans l'aide de l'application.

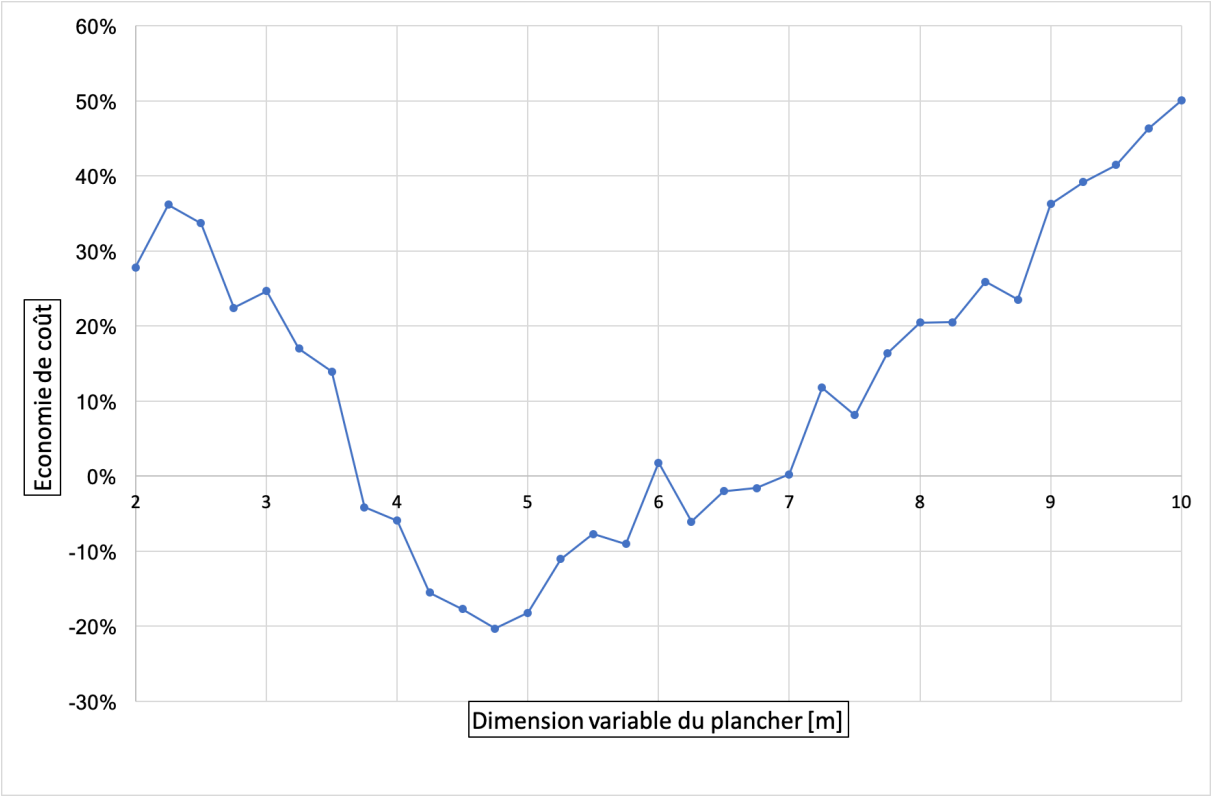


FIGURE 6.8 – Economie de coût entre le coût minimal pour un gîtage sans poutre intermédiaire part rapport à un gîtage avec poutre intermédiaire

En second lieu, pour chaque valeur considérée pour la dimension variable, nous allons cette fois calculer l'économie sur le volume (équation 6.25) que nous réalisons si nous optons pour une disposition avec poutre intermédiaire par rapport à une disposition simple (sans poutre intermédiaire), et ce quel que soit le sens de la portée des éléments ou de la poutre intermédiaire. Cette économie est représentée à la figure 6.9 en fonction de la dimension variable.

$$\text{économie}_{vol}[\%] = \frac{\text{volume}_{simp} - \text{volume}_{inter}}{\text{volume}_{inter}} \tag{6.25}$$

où volume_{inter} est le volume minimum qu'on puisse atteindre pour le plancher donné si nous utilisons une poutre intermédiaire, et volume_{simp} le volume minimum qu'il est possible d'atteindre pour le même plancher si nous n'utilisons pas de poutre intermédiaire.

Les observations que l'on peut faire à propos du graphique présenté à la figure 6.9 sont assez similaires à celles faites à propos du graphique de la figure 6.8.

- Il est préférable d'opter pour une disposition simple du gîtage (lorsque l'économie calculée à l'équation 6.25 est négative) dans deux cas :
  - pour des valeurs inférieures à 2,5$[m]$ pour la dimension variable, car peu de très petites sections suffiront à porter le plancher et l'ajout d'une poutre intermédiaire ne fera qu'ajouter du volume inutile ; cependant, cela se produit pour des valeurs encore

inférieures à celles permettant une économie sur le coût, car ici il n'y a que le volume de bois qui interviendra et pas les surcoûts des fixations, etc. ;

- pour de grandes valeurs de la dimension variable (ici, supérieures à 8,75$[m]$), de nouveau car la poutre intermédiaire devrait avoir une section assez importante et que, dans notre plage de sections, le choix de grandes sections de BLC est limité. Il faut donc parfois opter inutilement pour des sections bien plus grandes que la section minimale théorique qui suffirait.

- Malgré ces observations, même si on voit qu'il est ici généralement préférable d'avoir recours à une poutre intermédiaire, l'économie sur le volume de bois qu'il est possible de faire entre les deux dispositions dépendra des problèmes et des plages de sections disponibles ; il est donc difficile de prévoir la meilleure solution pour un cas donné sans l'aide de l'application.

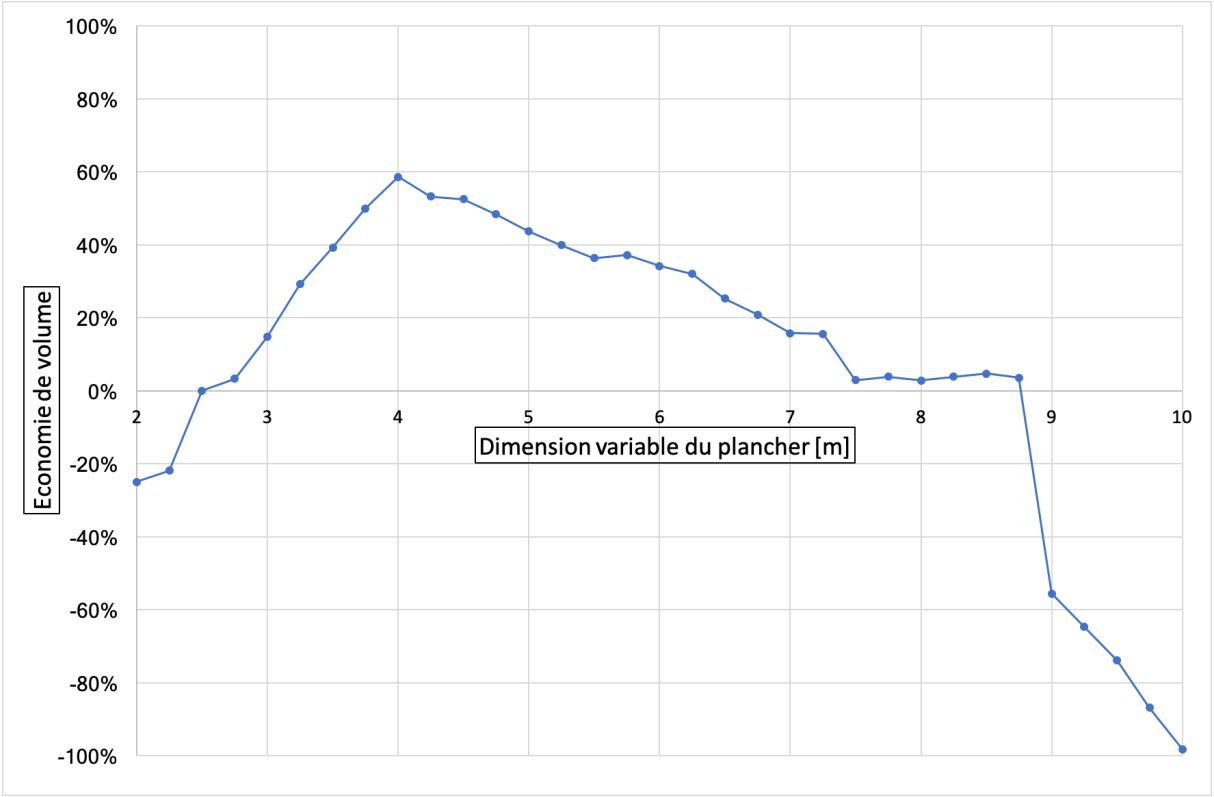


FIGURE 6.9 – Economie de volume entre le volume minimal pour un gîtage avec poutre intermédiaire part rapport à un gîtage sans poutre intermédiaire

Ce chapitre nous a permis de tester et de démontrer le bon fonctionnement de l'application que nous avons développée dans le cadre de ce mémoire, mais aussi et surtout son utilité, que ce soit en gain de volume de bois, en gain de coût ou en gain de temps pour l'utilisateur. Nous avons ainsi réalisé le second objectif de ce mémoire.

Conclusion

Le travail accompli dans le cadre de ce mémoire a permis d'atteindre les objectifs suivants :

- la formulation d'un critère adimensionnel général pour le dimensionnement d'une poutre en bois de section rectangulaire, adapté à la conception d'un gîtage ;
- la déduction de la configuration d'un gîtage qui conduit, en théorie, au volume minimal de bois à mettre en œuvre, valable pour tout plancher rectangulaire et pour tout cas de charge ;
- l'introduction de nouveaux concepts permettant de situer une certaine configuration de gîtage par rapport à d'autres et par rapport à l'optimum théorique ;
- le développement d'une application hybride permettant d'obtenir immédiatement la configuration du gîtage minimisant le volume de bois et celle minimisant le coût total, et dont les paramètres, parmi lesquels l'ensemble des sections de poutres disponibles, peuvent être facilement personnalisés.

Ce travail de recherche sur les gîtages et le développement de l'application ne forment qu'une étape dans l'optimisation du dimensionnement de gîtages. La recherche peut en effet être progressivement étendue en intégrant plus de paramètres, tels que la participation des panneaux de plancher, ou en incluant d'autres cas possibles de disposition du gîtage. De nombreuses pistes d'enrichissement ont d'ailleurs été proposées en fin du cinquième chapitre.

Atteindre les objectifs fixés est un aboutissement académique certain, mais il est permis de penser que ce travail et l'application qui en résulte permettront d'apporter un impact environnemental, économique et sociétal au-delà de ces objectifs. En effet, la solution optimale en termes de volume de bois fournie par l'application conduit à réduire la consommation de cette ressource naturelle. De plus, la solution optimale en termes de coût pour un plancher, calculée sur base de paramètres personnalisés au cas par cas, permet de faire de réelles économies sur cette partie du bâtiment. A ne pas négliger non plus, l'économie de temps pour les ingénieurs en bureau d'études ou sur chantier qui pourront rapidement et fiablement, grâce à l'application, déterminer le dimensionnement optimal d'un gîtage.

Enfin, à titre personnel, ce mémoire m'a permis de faire idéalement le lien entre les compétences et connaissances acquises lors de mon master en construction et de la mineure en informatique que j'ai choisie durant mon bac, associant une partie de recherche dans les calculs de structures et un outil technologique directement utilisable par les ingénieurs.

Bibliographie

[1] Thierry Descamps, Michel Monhonval, Pierre Latteur. An assessment methodology for timber beams in a restoration context, based on a dimensionless formulation of EC5 design criteria. *International Journal of Architectural Heritage*, 2018. DOI : 10.1080/15583058.2018.1442527.

[2] EN 1995-1-1. *Eurocode 5 : Conception et calcul des structures en bois - Partie 1-1 : Généralités - Règles communes et règles pour les bâtiments.* Comité Européen de Normalisation, CEN, 2004.

[3] EN 1990. *Eurocode 0 : Bases de calcul des structures.* Comité Européen de Normalisation, CEN, 2002.

[4] EN 1991-1-1. *Eurocode 1 : Actions sur les structures - Partie 1-1 : Actions générales - Poids volumiques, poids propres, charges d'exploitation bâtiments.* Comité Européen de Normalisation, CEN, 2002.

[5] EN 338. *Bois de structure - Classes de résistance.* Comité Européen de Normalisation, CEN, 2016.

[6] EN 14080. *Structures en bois - Bois lamellé-collé et bois massif reconstitué - Exigences.* Comité Européen de Normalisation, CEN, 2013.

[7] Pierre Latteur. LGCIV2043 : Structures en bois. Notes de cours. *Ecole Polytechnique de Louvain*, 2017.

[8] Open Source Initiative. MIT License. `https://ionicframework.com/docs/v1/overview/`. Consulté en octobre 2018.

[9] Ionic. Ionic Documentation. `https://ionicframework.com/docs/`. Consulté en octobre 2018.

[10] Angular. Angular Documentation. `https://angular.io/docs/`. Consulté en octobre 2018.

[11] Apache Cordova. Apache Cordova Documentation. `https://cordova.apache.org/docs/`. Consulté en octobre 2018.

[12] NodeJS. NodeJS Documentation. `https://nodejs.org/en/docs/`. Consulté en octobre 2018.

[13] Tutorialspoint. TypeScript - Overview. `http://www.tutorialspoint.com/typescript/typescript_overview.htm`. Consulté en octobre 2018.

[14] Tutorialspoint. HTML - Overview. `http://www.tutorialspoint.com/html/html_overview.htm`. Consulté en octobre 2018.

[15] Tutorialspoint. HTML5 - Overview. `http://www.tutorialspoint.com/html5/html5_overview.htm`. Consulté en octobre 2018.

[16] Tutorialspoint. What is CSS ? `http://www.tutorialspoint.com/css/what_is_css.htm`. Consulté en octobre 2018.

[17] Mozilla et contributeurs individuels. Promise. `http://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Promise`. Consulté en octobre 2018.

[18] Simpson Strong-Tie. *Guide de Préconisation*, 2017.

[19] Eurabo cvba. Catalogue Simpson Stong-Tie. `http://www.eurabo.be/fr/produits/marques/simpson-strong-tie`. Consulté en novembre 2018.

[20] Eurabo cvba. Catalogue Bois d'ingénierie. http://www.eurabo.be/fr/produits/categories/bois-d-ingenierie-joist-lvl-gl-fsc-pefc. Consulté en novembre 2018.

[21] Swiss Krono. Abaques Swiss Krono OSB en plancher et toiture. http://www.kronofrance.fr/IMG/pdf/Abaques_de_charges_OSB.pdf. Consulté en novembre 2018.

Annexe A

Tableaux de valeurs issus des Eurocodes

A.1 Valeurs des coefficients ψ (EN1990)

Catégorie	ψ_0	ψ_1	ψ_2
A : Habitations & zones résidentielles	0,7	0,5	0,3
B : Bureaux	0,7	0,5	0,3
C : Lieux de réunion	0,7	0,7	0,6
D : Commerces	0,7	0,7	0,6
E : Stockage	1,0	0,9	0,8
H : Toits	0,0	0,0	0,0

Tableau A.1 – Valeur des coefficients ψ pour des catégories de charges d'exploitation des bâtiments

A.2 Charges d'exploitation pour les bâtiments (EN1991)

Catégorie	Usage spécifique	$q[kN/m^2]$
A	Habitation, résidentiel	2,0
B	Bureaux	3,0
C1	Espaces équipés de tables	3,0
C2	Espaces équipés de sièges fixes	4,0
C3	Espaces ne présentant pas d'obstacle à la circulation des personnes	5,0
C4	Espaces permettant des activités physiques	5,0
C5	Espaces susceptibles d'accueillir des foules importantes	5,0
D1	Commerces de détail	5,0
D2	Grands magasins	5,0
E1	Espaces susceptibles de recevoir une accumulation de marchandises	7,5
E2	Usage industriel	5,0
H	Toitures inaccessibles	0,8

Tableau A.2 – Catégories de charges d'exploitations pour les bâtiments

A.3 Valeurs de k_{def} (EN1995)

Matériau	Classes de Service		
	1	2	3
Bois massif	0,60	0,80	2,00
Bois lamellé-collé	0,60	0,80	2,00

Tableau A.3 – Valeurs de k_{def}

A.4 Valeurs de k_{mod} (EN1995)

Matériau	Classes de Service	Classes de durée de chargement				
		Action permanente	Action long terme	Action moyen terme	Action court terme	Action instantanée
Bois massif	1	0,60	0,70	0,80	0,90	1,10
	2	0,60	0,70	0,80	0,90	1,10
	3	0,50	0,55	0,65	0,70	0,90
Bois lamellé collé	1	0,60	0,70	0,80	0,90	1,10
	2	0,60	0,70	0,80	0,90	1,10
	3	0,50	0,55	0,65	0,70	0,90

Tableau A.4 – Valeurs de k_{mod}

A.5 Classes de durée de chargement (EN1995)

Classe de durée de chargement cumulée	Ordre de grandeur de la durée de la charge caractéristique
Permanent	plus de 10 ans
Long terme	6 mois à 10 ans
Moyen terme	1 semaine à 6 mois
Court terme	moins d'une semaine
Instantané	

Tableau A.5 – Classes de durée de chargement

A.6 Valeurs de γ_M (EN1995)

Etats Limites Ultimes	γ_M
Combinaison fondamentale	
Bois massif	1,30
Bois lamellé-collé	1,25
OSB	1,20
Combinaison accidentelle	1,00

Tableau A.6 – Coefficients partiels recommandés pour les propriétés des matériaux

A.7 Valeurs maximales de flèche (EN1995)

	w_{inst}	$w_{net,fin}$	w_{fin}
Poutre sur deux appuis	$L/300$ à $L/500$	$L/250$ à $L/350$	$L/150$ à $L/300$
Poutre en porte-à-faux	$L/150$ à $L/250$	$L/125$ à $L/175$	$L/75$ à $L/150$

Tableau A.7 – Valeurs maximales de flèche

A.8 Valeurs caractéristiques des classes de résistance de bois massif (EN338)

Classes de résistance	Valeurs caractéristiques			
	$f_{m,k}$	$f_{v,k}$	E	G
C14	14	3	7000	440
C16	16	3,2	8000	500
C18	18	3,4	9000	560
C20	20	3,6	9500	590
C22	22	3,8	10000	630
C24	24	4	11000	690
C27	27	4	11500	720
C30	30	4	12000	750
C35	35	4	13000	810
C40	40	4	14000	880
C45	45	4	15000	940
C50	50	4	16000	1000
D18	18	3,5	9500	590
D24	24	3,7	10000	620
D27	27	3,8	10500	660
D30	30	3,9	11000	690
D35	35	4,1	12000	750
D40	40	4,2	13000	810
D50	50	4,5	14000	880
D60	60	4,8	17000	1060
D70	70	5	20000	1250

Tableau A.8 – Extrait des valeurs caractéristiques des classes de résistance de bois massif

A.9 Valeurs caractéristiques des classes de résistance de bois lamellé-collé (EN14080)

Classes de résistance	Valeurs caractéristiques			
	$f_{m,k}$	$f_{v,k}$	E	G
GL20h	20	3,5	8400	650
GL22h	22	3,5	10500	650
GL24h	24	3,5	11500	650
GL26h	26	3,5	12100	650
GL28h	28	3,5	12600	650
GL30h	30	3,5	13600	650
GL32h	32	3,5	14200	650

Tableau A.9 – Extrait des valeurs caractéristiques des classes de résistance de bois lamellé-collé

Annexe B

Code source de l'application

B.1 Modèles

B.1.1 Paramètres du problème

```
1
2  export class ProblemParams {
3
4      // Parametres generaux du probleme
5      dimLP: number[];
6      entraxeMax: number;
7      ninst: number; nfin: number;
8      gammaM: number;
9      kdef: number; kmod: number; kcr: number; ksys: number;
10     gammaG: number; gammaQ: number;
11     psi0: number; psi1: number; psi2: number;
12     chargePermG: number; chargeVarQ: number;
13     // Contraintes Architecturales
14     hmax: number;
15     sensUnic: string;
16     emin: number;
17     // Parametres des elements standards
18     range_index: number;
19     cat: string; classeBois: string;
20     fmk: number; fvk: number;
21     modulE: number; modulG: number;
22     // Parametres de la poutre intermediaire
23     inter_beam: boolean;
24     range_index_inter: number;
25     cat_inter: string; classeBois_inter: string;
26     fmk_inter: number; fvk_inter: number;
27     modulE_inter: number; modulG_inter: number;
28
29     constructor(){
30       this.dimLP = [];
31     }
32
33     // Combinaison ELU
34     getCombinaisonELU(){
35       return ((this.gammaG*this.chargePermG)+(this.gammaQ*this.chargeVarQ));
36     }
37
38     // Combinaison ELS court terme
39     getCombinaisonELSinst(){
40       return (this.chargePermG+this.chargeVarQ);
41     }
42
```

```
43      // Combinaison ELS long terme
44      getCombinaisonELSfin(){
45        return (((1+this.kdef)*this.chargePermG)+this.chargeVarQ);
46      }
47
48  }
```

Code B.1 – ProblemParams.ts

B.1.2 Largeur de section et coûts associés aux hauteurs disponibles

```
1   export class Width {
2
3       b: number;
4       h: number[];
5       m: number[];
6       p: number[];
7       f: number[];
8
9       constructor(b: number){
10        this.b = b;
11        this.h = [];
12        this.m = [];
13        this.p = [];
14        this.f = [];
15      }
16
17  }
```

Code B.2 – Width.ts

B.1.3 Plage de largeurs de sections

```
1   import { Width } from './Width';
2
3   export class Range {
4
5       largeurs: Width[];
6
7       constructor(public name: string){
8         this.largeurs = [];
9       }
10
11  }
```

Code B.3 – Range.ts

B.2 Services

B.2.1 Service des Eurocodes

```
1   export class EurocodesService {
2
3     // Classes de resistances de bois massif et de blc
4     public classesBois =
5     {c14:{cat:'massif', fmk:14, fvk:3, modulE:7000, modulG:440},
6      c16:{cat:'massif', fmk:16, fvk:3.2, modulE:8000, modulG:500},
7      c18:{cat:'massif', fmk:18, fvk:3.4, modulE:9000, modulG:560},
```

```
    c20:{cat:'massif', fmk:20, fvk:3.6, modulE:9500, modulG:590},
    c22:{cat:'massif', fmk:22, fvk:3.8, modulE:10000, modulG:630},
    c24:{cat:'massif', fmk:24, fvk:4, modulE:11000, modulG:690},
    c27:{cat:'massif', fmk:27, fvk:4, modulE:11500, modulG:720},
    c30:{cat:'massif', fmk:30, fvk:4, modulE:12000, modulG:750},
    c35:{cat:'massif', fmk:35, fvk:4, modulE:13000, modulG:810},
    c40:{cat:'massif', fmk:40, fvk:4, modulE:14000, modulG:880},
    c45:{cat:'massif', fmk:45, fvk:4, modulE:15000, modulG:940},
    c50:{cat:'massif', fmk:50, fvk:4, modulE:16000, modulG:1000},
    d18:{cat:'massif', fmk:18, fvk:3.5, modulE:9500, modulG:590},
    d24:{cat:'massif', fmk:24, fvk:3.7, modulE:10000, modulG:620},
    d27:{cat:'massif', fmk:27, fvk:3.8, modulE:10500, modulG:660},
    d30:{cat:'massif', fmk:30, fvk:3.9, modulE:11000, modulG:690},
    d35:{cat:'massif', fmk:35, fvk:4.1, modulE:12000, modulG:750},
    d40:{cat:'massif', fmk:40, fvk:4.2, modulE:13000, modulG:810},
    d50:{cat:'massif', fmk:50, fvk:4.5, modulE:14000, modulG:880},
    d60:{cat:'massif', fmk:60, fvk:4.8, modulE:17000, modulG:1060},
    d70:{cat:'massif', fmk:70, fvk:5, modulE:20000, modulG:1250},
    gl20h:{cat:'blc', fmk:20, fvk:3.5, modulE:8400, modulG:650},
    gl22h:{cat:'blc', fmk:22, fvk:3.5, modulE:10500, modulG:650},
    gl24h:{cat:'blc', fmk:24, fvk:3.5, modulE:11500, modulG:650},
    gl26h:{cat:'blc', fmk:26, fvk:3.5, modulE:12100, modulG:650},
    gl28h:{cat:'blc', fmk:28, fvk:3.5, modulE:12600, modulG:650},
    gl30h:{cat:'blc', fmk:30, fvk:3.5, modulE:13600, modulG:650},
    gl32h:{cat:'blc', fmk:32, fvk:3.5, modulE:14200, modulG:650}};

  // Categorie d'utilisation
  public categUtil =
  {a:  {q: 2.0, psi0:0.7, psi1:0.5, psi2:0.3},
   b:  {q: 3.0, psi0:0.7, psi1:0.5, psi2:0.3},
   c1: {q: 3.0, psi0:0.7, psi1:0.7, psi2:0.6},
   c2: {q: 4.0, psi0:0.7, psi1:0.7, psi2:0.6},
   c3: {q: 5.0, psi0:0.7, psi1:0.7, psi2:0.6},
   c4: {q: 5.0, psi0:0.7, psi1:0.7, psi2:0.6},
   c5: {q: 5.0, psi0:0.7, psi1:0.7, psi2:0.6},
   d1: {q: 5.0, psi0:0.7, psi1:0.7, psi2:0.6},
   d2: {q: 5.0, psi0:0.7, psi1:0.7, psi2:0.6},
   e1: {q: 7.5, psi0:0.1, psi1:0.9, psi2:0.8},
   e2: {q: 5.0, psi0:0.1, psi1:0.9, psi2:0.8},
   h:  {q: 0.8, psi0:0.0, psi1:0.0, psi2:0.0}};

  // Facteurs adimensionnels
  public gammaM = {massif: 1.3, blc: 1.25};

  public kdef = {cs1: 0.6, cs2: 0.8, cs3: 2.0};

  public kmod =
  {cs1: {perm: 0.6, lt: 0.7, mt: 0.8, ct: 0.9, inst: 1.1},
   cs2: {perm: 0.6, lt: 0.7, mt: 0.8, ct: 0.9, inst: 1.1},
   cs3: {perm: 0.5, lt: 0.55, mt: 0.65, ct: 0.7, inst: 0.9}};

  public kcr = 0.67;

  public ksys = 1.10;

  public gammaG = 1.35;

  public gammaQ = 1.50;

  getkh(height: number, woodtype: String){
    if(woodtype=='massif'){
      return Math.max(Math.min(Math.pow(150/(height*10),0.2),1.3),1);
    }
```

```
    else if(woodtype=='blc'){
      return Math.max(Math.min(Math.pow(600/(height*10),0.1),1.1),1);
    }
    else {
      return 1;
    }
  }
}
```

CODE B.4 – `eurocodes.service.ts`

B.2.2 Service de base de données des plages de sections

```
import { Range } from '../models/Range';
import { Width } from '../models/Width';
import { Injectable } from '@angular/core';
import { Storage } from '@ionic/storage';

@Injectable()
export class RangesService {

  // Plages de sections par defaut
  private rangesList: Range[] = [
    {name: 'Massif épicéa - C24',
     largeurs:[{b:3.6, h:[7.0, 9.5, 12.0, 14.5, 17.0, 19.5, 22.0],
                       m:[1.26, 1.71, 2.16, 2.61, 3.06, 3.51, 3.96],
                       p:[10, 10, 10, 10, 10, 10, 10],
                       f:[2.31, 2.31, 2.31, 2.31, 2.31, 2.31, 3.58]},
               {b:4.7, h:[7.0, 9.5, 12.0, 14.5, 17.0, 19.5, 22.0],
                       m:[1.65, 2.23, 2.82, 3.41, 4.00, 4.58, 5.17],
                       p:[10, 10, 10, 10, 10, 10, 10],
                       f:[2.52, 2.52, 2.52, 2.52, 2.52, 2.52, 3.59]},
               {b:6.3, h:[7.0, 9.5, 12.0, 14.5, 17.0, 19.5, 22.0],
                       m:[2.21, 2.99, 3.78, 4.57, 5.36, 6.14, 6.93],
                       p:[10, 10, 10, 10, 10, 10, 10],
                       f:[2.31, 2.31, 2.31, 2.31, 2.31, 1.98, 2.37]},
               {b:7.2, h:[7.0, 9.5, 12.0, 14.5, 17.0, 19.5, 22.0],
                       m:[2.52, 3.42, 4.32, 5.22, 6.12, 7.02, 7.92],
                       p:[10, 10, 10, 10, 10, 10, 10],
                       f:[3.58, 3.58, 3.58, 3.58, 3.58, 3.58, 3.58]},
               {b:9.0, h:[7.0, 9.5, 12.0, 14.5, 17.0, 19.5, 22.0],
                       m:[3.15, 4.28, 5.40, 6.53, 7.65, 8.78, 9.90],
                       p:[10, 10, 10, 10, 10, 10, 10],
                       f:[3.93, 3.93, 3.93, 3.93, 3.93, 3.58, 3.58]}]},
    {name: 'BLC épicéa - GL24h',
     largeurs:[{b:4.5, h:[10, 30, 36, 40, 45],
                       m:[3.1, 9.29, 11.15, 12.38, 13.93],
                       p:[10, 10, 10, 10, 10],
                       f:[2.52, 3.59, 10.01, 10.01, 10.01]},
               {b:8.0, h:[16, 20, 24, 28, 32, 36],
                       m:[8.81, 11.01, 12.06, 15.41, 17.61, 19.81],
                       p:[10, 10, 10, 10, 10, 10],
                       f:[3.46, 3.46, 3.46, 7.77, 8.16, 8.16]},
               {b:10,  h:[20, 32, 36, 40, 44, 46],
                       m:[13.76, 22.02, 24.77, 27.52, 30.27, 31.65],
                       p:[10, 10, 10, 10, 10, 10],
                       f:[2.96, 9.91, 9.91, 9.91, 9.91, 9.91]},
               {b:12,  h:[20, 24, 28, 32, 36, 40],
                       m:[16.51, 19.81, 23.12, 26.42, 29.72, 33.02],
                       p:[10, 10, 10, 10, 10, 10],
                       f:[9.55, 9.55, 9.91, 9.91, 9.91, 9.91]},
               {b:14,  h:[16, 24, 28, 30, 36, 40, 44, 46, 50, 60],
```

```
                        m:[15.41, 23.12, 26.97, 29.39, 33.06, 38.53, 42.38,
    44.31, 47.76, 55.10],
                        p:[10, 10, 10, 10, 10, 10, 10, 10, 10, 10],
                        f:[3.59, 3.59, 9.91, 9.91, 9.91, 9.91, 21.78, 21.78,
    21.78, 21.78]},
                {b:16,  h:[16, 28, 30, 36],
                        m:[17.61, 30.82, 35.23, 39.63],
                        p:[10, 10, 10, 10],
                        f:[7.34, 7.34, 11.30, 11.30]},
                {b:18,  h:[32, 52],
                        m:[39.63, 64.40],
                        p:[10, 10],
                        f:[16.55, 36.70]},
                {b:20,  h:[20, 32, 44, 46, 52],
                        m:[27.52, 44.03, 60.54, 63.3, 71.55],
                        p:[10, 10, 10, 10, 10],
                        f:[11.30, 11.30, 31.31, 31.31, 31.31]},
                {b:22,  h:[24, 28, 36, 52],
                        m:[36.33, 42.38, 54.49, 78.71],
                        p:[10, 10, 10, 10],
                        f:[20.02, 20.02, 20.02, 37.31]}]},
    {name: 'BLC épicéa - GL28h',
     largeurs:[{b:8.0, h:[16, 32, 40],
                        m:[9.63, 19.25, 24.06],
                        p:[10, 10, 10],
                        f:[3.46, 8.16, 10.01]},
                {b:10,  h:[32, 40, 48, 60],
                        m:[24.58, 30.72, 36.86, 46.08],
                        p:[10, 10, 10, 10],
                        f:[9.91, 9.91, 21.78, 21.78]},
                {b:14,  h:[60],
                        m:[63.17],
                        p:[10],
                        f:[21.78]},
                {b:16,  h:[16, 20],
                        m:[19.66, 24.58],
                        p:[10, 10],
                        f:[9.06, 9.06]},
                {b:18,  h:[24],
                        m:[32.49],
                        p:[10],
                        f:[9.90]},
                {b:20,  h:[28.1, 32, 40, 68],
                        m:[42.11, 48.13, 60.16, 102.27],
                        p:[10, 10, 10, 10],
                        f:[11.3, 11.3, 37.31, 37.31]},
                {b:24,  h:[32, 52],
                        m:[57.75, 91.85],
                        p:[10, 10],
                        f:[37.31, 37.31]},
                {b:26,  h:[56],
                        m:[109.49],
                        p:[10],
                        f:[37.31]}]}
];

// Initialisation du stockage
constructor(private storage: Storage) {
  this.storage=new Storage({name:"__db"});
  this.storage.set('ranges', this.rangesList);
}

// Reinitialisation des plages de section
```

```
111   resetDefaultRanges() {
112     return this.storage.set('ranges',this.rangesList);
113   }
114
115   // Obtenir toutes les plages de sections
116   getRangesList(){
117     return this.storage.get('ranges');
118   }
119
120   // Obtenir une plage de section particuliere
121   getRange(range_index: number){
122     return this.getRangesList().then((list: Range[]) => {
123       return list[range_index];
124     });
125   }
126
127   // Obtenir une largeur d'une certaine plage de sectione
128   getWidth(range_index: number, width_index: number){
129     return this.getRangesList().then((list: Range[]) => {
130       return list[range_index].largeurs[width_index];
131     });
132   }
133
134   // Ajouter une plage de section
135   addRange(plage: Range) {
136     return this.getRangesList().then((list: Range[]) => {
137       if (list) {
138         list.push(plage);
139         return this.storage.set('ranges', list);
140       }
141       else {
142         return this.storage.set('ranges', [plage]);
143       }
144     });
145   }
146
147   // Modifier le nom d'une plage de sections
148   editRangeName(range_index: number, new_name: string) {
149     return this.getRangesList().then((list: Range[]) => {
150       if (list) {
151         list[range_index].name = new_name;
152         return this.storage.set('ranges', list);
153       }
154     });
155   }
156
157   // Supprimer une plage de section
158   deleteRange(range_index: number){
159     return this.getRangesList().then((list: Range[]) => {
160       if (list) {
161         list.splice(range_index, 1);
162         return this.storage.set('ranges', list);
163       }
164     });
165   }
166
167   // Ajouter une largeur a une plage de section
168   addWidth(range_index: number, new_largeur: Width) {
169     return this.getRangesList().then((list: Range[]) => {
170       if (list) {
171         list[range_index].largeurs.push(new_largeur);
172         return this.storage.set('ranges', list);
173       }
```

```
      else {
        let new_range: Range={name:'unnamed', largeurs:[new_largeur]};
        return this.storage.set('ranges', [new_range]);
      }
    });
  }

  // Supprimer une largeur d'une plage de section
  deleteWidth(range_index: number, width_index: number){
    return this.getRangesList().then((list: Range[]) => {
      if (list) {
        list[range_index].largeurs.splice(width_index, 1);
        return this.storage.set('ranges', list);
      }
    });
  }

  // Ajouter une hauteur a une largeur d'une plage de section
  addHeight(range_index: number, width_index: number, new_hmpf: number[]) {
    return this.getRangesList().then((list: Range[]) => {
      if (list) {
        list[range_index].largeurs[width_index].h.push(new_hmpf[0]);
        list[range_index].largeurs[width_index].m.push(new_hmpf[1]);
        list[range_index].largeurs[width_index].p.push(new_hmpf[2]);
        list[range_index].largeurs[width_index].f.push(new_hmpf[3]);
        return this.storage.set('ranges', list);
      }
    });
  }

  // Supprimer une hauteur d'une largeur d'une plage de section
  deleteHeight(range_index: number, width_index: number, height_index: number){
    return this.getRangesList().then((list: Range[]) => {
      if (list) {
        list[range_index].largeurs[width_index].h.splice(height_index, 1);
        list[range_index].largeurs[width_index].m.splice(height_index, 1);
        list[range_index].largeurs[width_index].p.splice(height_index, 1);
        list[range_index].largeurs[width_index].f.splice(height_index, 1);
        return this.storage.set('ranges', list);
      }
    });
  }

}
```

Code B.5 – `ranges.service.ts`

B.3 Pages

B.3.1 Page d'accueil

```
import { Component } from '@angular/core';
import { NavController } from 'ionic-angular';

import { CalculrapPage } from '../calculrap/calculrap';
import { CalculpersPage } from '../calculpers/calculpers';
import { RangesPage } from '../ranges/ranges';
import { AboutPage } from '../about/about';

@Component({
  selector: 'page-home',
```

```
  templateUrl: 'home.html'
})
export class HomePage {

  constructor(public navCtrl: NavController) {}

  onGoToCalcRap(){
    this.navCtrl.push(CalculrapPage);
  }

  onGoToCalcPers(){
    this.navCtrl.push(CalculpersPage);
  }

  onGoToRanges(){
    this.navCtrl.push(RangesPage);
  }

  onGoToAbout(){
    this.navCtrl.push(AboutPage);
  }

}
```

Code B.6 – home.ts

```
<ion-header charset=UTF-8>
    <ion-navbar>
        <button ion-button menuToggle>
            <ion-icon name="menu"></ion-icon>
        </button>

        <ion-title>ACCUEIL</ion-title>

        <ion-buttons end>
            <button ion-button clear color="dark">
                <ion-icon name="globe"></ion-icon>
            </button>
        </ion-buttons>
    </ion-navbar>
</ion-header>

<ion-content padding>
    <h3>Bienvenue dans GitageCalculator</h3>

    <button ion-button block (click)="onGoToCalcRap()">CALCUL RAPIDE</button>
    <div text-justify padding-bottom>
        Définissez les principaux paramètres du gîtage pour un calcul rapide bas
    é sur les Eurocodes.
    </div>

    <button ion-button block (click)="onGoToCalcPers()">CALCUL PERSONNALISE</
    button>
    <div text-justify padding-bottom>
        Définissez tous les paramètres du gîtage et ceux définis dans les
    Eurocodes de manière personnalisée.
    </div>

    <button ion-button block (click)="onGoToRanges()">GERER LES SECTIONS</button
    >
    <div text-justify padding-bottom>
        Visualisez, ajoutez ou supprimez des sections ou des plages de sections.
     Les calculs s'effectuent sur base d'une certaine plage de sections.
    </div>
```

```
34
35      <button ion-button block (click)="onGoToAbout()">A PROPOS</button>
36      <div text-justify padding-bottom>
37          Informations à propos des auteurs et de la méthode de calcul.
38      </div>
39
40  </ion-content>
```

Code B.7 – home.html

B.3.2 Page d'informations à propos de l'application

```
1  import { Component } from '@angular/core';
2  import { IonicPage, NavController } from 'ionic-angular';
3
4  @IonicPage()
5  @Component({
6    selector: 'page-about',
7    templateUrl: 'about.html',
8  })
9  export class AboutPage {
10
11   constructor(public navCtrl: NavController) {}
12
13 }
```

Code B.8 – about.ts

```
1  <ion-header charset=UTF-8>
2
3    <ion-navbar>
4          <button ion-button menuToggle>
5              <ion-icon name="menu"></ion-icon>
6          </button>
7
8          <ion-title>A PROPOS</ion-title>
9
10         <ion-buttons end>
11             <button ion-button clear color="dark">
12                 <ion-icon name="globe"></ion-icon>
13             </button>
14         </ion-buttons>
15   </ion-navbar>
16
17 </ion-header>
18
19
20 <ion-content padding>
21
22     <div text-justify>
23         Cette application a été développée dans le cadre du TFE de Benjamin
       Moeremans dans le but de l'obtention de son diplôme d'ingénieur civil des
       constructions à l'Université Catholique de Louvain, avec la collaboration de
        son promoteur, le Professeur Pierre Latteur, et de l'Ingénieur Michel
       Monhonval.
24     </div>
25     <div text-justify>
26         La solution optimale pour le dimensionnement d'un gîtage fournie par
       cette application est basée sur une analyse théorique détaillée dans le TFE,
        elle-même basée, d'une part, sur les méthodes de calcul prescrites par les
       normes européennes et, d'autre part, sur un article de Descamps, Monhonval
       et Latteur (2018).
```

```
    </div>

</ion-content>
```

CODE B.9 – about.html

B.3.3 Liste des plages de sections

```
// ANGULAR ELEMENTS
import { Component } from '@angular/core';
import { NavController, AlertController } from 'ionic-angular';
// MODELS & SERVICES
import { Range } from '../../models/Range';
import { RangesService } from '../../services/ranges.service';
// PAGES
import { SingleRangePage } from './single-range/single-range';
import { RangeFormPage } from './range-form/range-form';


@IonicPage()
@Component({
  selector: 'page-ranges',
  templateUrl: 'ranges.html',
})
export class RangesPage {

  rangesList: Range[];

  constructor(public navCtrl: NavController,
              private alertCtrl: AlertController,
              private rangesService: RangesService) {}

  ionViewWillEnter() {
    this.rangesService.getRangesList().then((ranges: Range[]) => {
      if (ranges) {
        this.rangesList = ranges.slice();
      }
    });
  }

  onLoadRange(range_index: number) {
    this.navCtrl.push(SingleRangePage, {range_index: range_index});
  }

  onNewRange() {
    this.navCtrl.push(RangeFormPage);
  }

  onDeleteRange(range_index: number) {
      this.rangesService.deleteRange(range_index).then(result => {
        this.navCtrl.pop();
      });
  }

  onDeleteRangeConfirm(range_index: number, event: Event) {
    event.stopPropagation();
    let alert = this.alertCtrl.create({
      title: 'Alerte',
      message: 'Supprimer '+this.rangesList[range_index].name+'?',
      buttons: [
        {
          text: 'Annuler',
```

```
          role: 'cancel',
          handler: () => {console.log('Annule');}
        },
        {
          text: 'Confirmer',
          handler: () => {this.onDeleteRange(range_index);}
        }
      ]
    });
    alert.present();
  }

  onResetRanges() {
      this.rangesService.resetDefaultRanges().then(result => {
        this.navCtrl.pop();
      });
  }

  onResetRangesConfirm() {
    let alert = this.alertCtrl.create({
      title: 'Alerte',
      message: "Supprimer toutes les plages et restaurer les plages d'origine?",
      buttons: [
        {
          text: 'Annuler',
          role: 'cancel',
          handler: () => {console.log('Annule');}
        },
        {
          text: 'Confirmer',
          handler: () => {this.onResetRanges();}
        }
      ]
    });
    alert.present();
  }

}
```

CODE B.10 – `ranges.ts`

```
<ion-header charset=UTF-8>

  <ion-navbar>
    <button ion-button menuToggle>
        <ion-icon name="menu"></ion-icon>
    </button>
    <ion-title>Gérer les Sections</ion-title>
  </ion-navbar>

</ion-header>


<ion-content padding>

  <ion-list>
      <button ion-item *ngFor="let range of rangesList; let i = index" (click)="
    onLoadRange(i)">
        {{ range.name }}
        <button item-end (click)="onDeleteRangeConfirm(i, $event)">
          <ion-icon clear name="trash" class="red"></ion-icon>
        </button>
      </button>
  </ion-list>
```

```
23
24    <button ion-button full (click)="onNewRange()">Nouvelle plage de sections</
        button>
25    <button ion-button full (click)="onResetRangesConfirm()" color="danger">Ré
        initialiser les sections par défaut</button>
26
27  </ion-content>
```

Code B.11 – ranges.html

B.3.4 Liste des largeurs d'une plage de sections

```
1   // ANGULAR ELEMENTS
2   import { Component } from '@angular/core';
3   import { NavController, NavParams, AlertController, ViewController } from 'ionic
        -angular';
4   // MODELS & SERVICES
5   import { Range } from '../../../models/Range';
6   import { RangesService } from '../../../services/ranges.service';
7   // PAGES
8   import { SingleWidthPage } from '../single-width/single-width';
9   import { WidthFormPage } from '../width-form/width-form';
10
11
12  @Component({
13    selector: 'page-single-range',
14    templateUrl: 'single-range.html',
15  })
16  export class SingleRangePage {
17
18    range_index: number;
19    range: Range;
20
21    name_edit: boolean;
22
23    new_name: string = "";
24
25    constructor(public navCtrl: NavController,
26                public navParams: NavParams,
27                private alertCtrl: AlertController,
28                public viewCtrl: ViewController,
29                private rangesService: RangesService) {
30      this.range = {name:'', largeurs:[]};
31      this.name_edit = false;
32    }
33
34    ionViewWillEnter() {
35      this.range_index = this.navParams.get('range_index');
36      this.name_edit = false;
37      this.rangesService.getRangesList().then((ranges: Range[]) => {
38        if (ranges) {
39          this.range = ranges[this.range_index];
40        }
41      });
42    }
43
44    onLoadWidth(width_index: number) {
45      this.navCtrl.push(SingleWidthPage, {width_index: width_index, range_index:
          this.range_index});
46    }
47
48    onNewWidth() {
```

```
49      this.navCtrl.push(WidthFormPage, {range_index: this.range_index, range_name:
         this.range.name});
50    }
51
52    onDeleteWidth(width_index: number) {
53        this.rangesService.deleteWidth(this.range_index, width_index).then(result
       => {
54          this.navCtrl.pop();
55        });
56    }
57
58    onDeleteWidthConfirm(width_index: number, event: Event) {
59      event.stopPropagation();
60      let alert = this.alertCtrl.create({
61        title: 'Alerte',
62        message: 'Supprimer b='+this.range.largeurs[width_index].b+'?',
63        buttons: [
64          {
65            text: 'Annuler',
66            role: 'cancel',
67            handler: () => {console.log('Annule');}
68          },
69          {
70            text: 'Confirmer',
71            handler: () => {this.onDeleteWidth(width_index);}
72          }
73        ]
74      });
75      alert.present();
76    }
77
78    onShowNameEdit() {
79      this.name_edit = true;
80    }
81
82    onEditName(new_name: string, event: Event) {
83      if (new_name==="") {
84        this.rangesService.editRangeName(this.range_index, "Plage Sans Nom").then
       (() => {
85          this.navCtrl.pop();
86        });
87      }
88      else {
89        this.rangesService.editRangeName(this.range_index, new_name).then(() => {
90          this.navCtrl.pop();
91        });
92      }
93    }
94
95  }
```

Code B.12 – single-range.ts

```
1  <ion-header charset=UTF-8>
2
3    <ion-navbar>
4      <button ion-button menuToggle>
5          <ion-icon name="menu"></ion-icon>
6      </button>
7      <ion-title>{{range.name}}</ion-title>
8    </ion-navbar>
9
10 </ion-header>
11
```

```
12
13 <ion-content padding>
14
15   <ion-list>
16     <button ion-item *ngFor="let width of range.largeurs; let i = index" (click)
       ="onLoadWidth(i)">
17             b = {{ range.largeurs[i].b }} [cm]
18             <button item-end (click)="onDeleteWidthConfirm(i, $event)">
19               <ion-icon clear name="trash" class="red"></ion-icon>
20             </button>
21     </button>
22   </ion-list>
23
24   <button ion-button full (click)="onNewWidth()">Nouvelle largeur</button>
25   <button ion-button full (click)="onShowNameEdit()">Modifier le nom</button>
26
27   <ion-item *ngIf="name_edit">
28     <ion-input placeholder="Nouveau nom" [(ngModel)]="new_name" name="new_name"
       type="text" class="placeholders"></ion-input>
29     <button item-end (click)="onEditName(new_name, $event)" color="secondary">
30       Confirmer
31     </button>
32   </ion-item>
33
34 </ion-content>
```

Code B.13 – single-range.html

B.3.5 Liste des hauteurs associées à une largeur d'une plage de sections

```
1  // ANGULAR ELEMENTS
2  import { Component } from '@angular/core';
3  import { NavController, NavParams, AlertController, ViewController } from 'ionic
       -angular';
4  // MODELS & SERVICES
5  import { Range } from '../../../models/Range';
6  import { Width } from '../../../models/Width';
7  import { RangesService } from '../../../services/ranges.service';
8  // PAGES
9  import { HeightFormPage } from '../height-form/height-form';
10
11 @Component({
12   selector: 'page-single-width',
13   templateUrl: 'single-width.html',
14 })
15 export class SingleWidthPage {
16
17   range_index: number;
18   width_index: number;
19   info: Width;
20   range_name: String;
21
22   constructor(public navCtrl: NavController,
23               public navParams: NavParams,
24               private alertCtrl: AlertController,
25               public viewCtrl: ViewController,
26               public rangesService: RangesService) {
27     this.info = {b:0, h:[], m:[], p:[], f:[]};
28   }
29
30   ionViewWillEnter() {
31     this.width_index = this.navParams.get('width_index');
```

```
    this.range_index = this.navParams.get('range_index');
    this.rangesService.getRangesList().then((ranges: Range[]) => {
      if (ranges) {
        this.info = ranges[this.range_index].largeurs[this.width_index];
        this.range_name = ranges[this.range_index].name;
      }
    });
  }

  onDeleteHeight(height_index: number) {
    this.rangesService.deleteHeight(this.range_index, this.width_index,
    height_index).then(result => {
      this.navCtrl.pop();
    });
  }

  onDeleteHeightConfirm(height_index: number) {
    let alert = this.alertCtrl.create({
      title: 'Alerte',
      message: 'Supprimer h='+this.info.h[height_index]+'?',
      buttons: [
        {
          text: 'Annuler',
          role: 'cancel',
          handler: () => {console.log('Annule');}
        },
        {
          text: 'Confirmer',
          handler: () => {this.onDeleteHeight(height_index);}
        }
      ]
    });
    alert.present();
  }

  onNewHeight() {
    this.navCtrl.push(HeightFormPage, {range_index: this.range_index,
    width_index: this.width_index, b_value: this.info.b, range_name: this.
    range_name});
  }

}
```

Code B.14 – single-width.ts

```
<ion-header charset=UTF-8>

  <ion-navbar>
    <button ion-button menuToggle>
        <ion-icon name="menu"></ion-icon>
    </button>
    <ion-title>{{range_name}} - b = {{info.b}} [cm]</ion-title>
  </ion-navbar>

</ion-header>


<ion-content padding>

  <ion-card>
     <ion-card-header>Liste des hauteurs & prix</ion-card-header>
     <ion-card-content>
         <ion-grid text-center class="borderOuter">
           <ion-row>
```

```
            <ion-col>
              h
            </ion-col>
            <ion-col>
              m
            </ion-col>
            <ion-col>
              pce
            </ion-col>
            <ion-col>
              fix
            </ion-col>
            <ion-col>
              supp
            </ion-col>
          </ion-row>
          <ion-row class="borderBottomDouble">
            <ion-col>
              [cm]
            </ion-col>
            <ion-col>
              [€]
            </ion-col>
            <ion-col>
              [€]
            </ion-col>
            <ion-col>
              [€]
            </ion-col>
            <ion-col>
              <ion-icon name="trash"></ion-icon>
            </ion-col>
          </ion-row>
          <ion-row *ngFor="let h_item of info.h" class="borderTopSimple">
            <ion-col>
              {{ h_item }}
            </ion-col>
            <ion-col>
              {{ info.m[info.h.indexOf(h_item)] }}
            </ion-col>
            <ion-col>
              {{ info.p[info.h.indexOf(h_item)] }}
            </ion-col>
            <ion-col>
              {{ info.f[info.h.indexOf(h_item)] }}
            </ion-col>
            <ion-col>
              <button ion-button
                      full
                      color="danger"
                      (click)="onDeleteHeightConfirm(info.h.indexOf(h_item))">
                      <ion-icon name="trash"></ion-icon>
                    </button>
            </ion-col>
          </ion-row>
        </ion-grid>
    </ion-card-content>
  </ion-card>

  <button ion-button full (click)="onNewHeight()">Nouvelle hauteur</button>

  <p>h : hauteur de section</p>
  <p>m : prix par mètre</p>
```

```
    <p>pce : prix par pièce</p>
    <p>fix : prix par fixation</p>
    <p>supp : supprimer</p>

</ion-content>
```

Code B.15 – single-width.html

B.3.6 Formulaire d'ajout d'une plage de sections

```
// ANGULAR ELEMENTS
import { Component, OnInit } from '@angular/core';
import { NavController } from 'ionic-angular';
import { FormArray, FormBuilder, FormGroup, Validators } from '@angular/forms';
// MODELS & SERVICES
import { Range } from '../../../models/Range';
import { Width } from '../../../models/Width';
import { RangesService } from '../../../services/ranges.service';


@Component({
  selector: 'page-range-form',
  templateUrl: './range-form.html'
})

export class RangeFormPage implements OnInit {

  rangeForm: FormGroup;

  constructor(public navCtrl: NavController,
              private formBuilder: FormBuilder,
              public rangesService: RangesService) {}

  ngOnInit() {
    this.initForm();
    this.rangesService.getRangesList().then(result => {
    });
  }

  initForm() {
    this.rangeForm = this.formBuilder.group({
      name: ['', Validators.required],
      largeursTab: this.formBuilder.array([])
    });
  }

  getWidthArray() {
    return this.rangeForm.get('largeursTab') as FormArray;
  }

  onAddWidth() {
    let newControl = this.formBuilder.control('');
    this.getWidthArray().controls.push(newControl);
  }

  onRemoveWidth(index: number) {
    this.getWidthArray().removeAt(index);
  }

  onSubmitForm() {
    let newRange = new Range(this.rangeForm.get('name').value);
    for (let control of this.getWidthArray().controls) {
```

```
      let newWidth = new Width(control.value);
      newRange.largeurs.push(newWidth);
    }
    this.rangesService.addRange(newRange).then(result => {
      this.navCtrl.pop();
    });
  }
}
```

Code B.16 – range-form.ts

```
<ion-header charset=UTF-8>
  <ion-navbar>
    <ion-title>Nouvelle plage de sections</ion-title>
  </ion-navbar>
</ion-header>

<ion-content padding>
  <form *ngIf="rangeForm" [formGroup]="rangeForm">
    <ion-list>
      <ion-item-divider class="dividers">Nom de la nouvelle plage de sections</ion-item-divider>
      <ion-item>
        <ion-input placeholder="Nouvelle Plage" formControlName="name" type="text" class="placeholders"></ion-input>
      </ion-item>

      <ion-item-divider class="dividers">Largeurs</ion-item-divider>
      <div formArrayName="largeursTab">
        <ion-item *ngFor="let control of getWidthArray().controls; let i = index">
          <ion-icon item-end
                    name="remove-circle"
                    color="danger"
                    (click)="onRemoveWidth(i)">
          </ion-icon>
          <ion-input [formControlName]="i" placeholder="Largeur {{i+1}}" type="number" min="0" step="0.1" class="placeholders"></ion-input>
        </ion-item>
      </div>
    </ion-list>

    <button ion-button full (click)="onAddWidth()">Ajouter une largeur</button>
    <button ion-button full color="secondary" (click)="onSubmitForm()">Enregistrer</button>
  </form>
</ion-content>
```

Code B.17 – range-form.html

B.3.7 Formulaire d'ajout d'une largeur à une plage de sections

```
// ANGULAR ELEMENTS
import { Component, OnInit } from '@angular/core';
import { NavController, NavParams } from 'ionic-angular';
import { FormArray, FormBuilder, FormGroup, Validators } from '@angular/forms';
// MODELS & SERVICES
import { Width } from '../../../models/Width';
// PAGES
import { RangesService } from '../../../services/ranges.service';


```

```
@Component({
  selector: 'page-width-form',
  templateUrl: './width-form.html'
})

export class WidthFormPage implements OnInit {

  range_index: number;
  range_name: String;
  widthForm: FormGroup;

  constructor(public navCtrl: NavController,
              public navParams: NavParams,
              private formBuilder: FormBuilder,
              public rangesService: RangesService) {}

  ngOnInit() {
    this.range_index = this.navParams.get('range_index');
    this.range_name = this.navParams.get('range_name');
    this.rangesService.getRangesList().then(result => {
      this.initForm();
    });
  }

  initForm() {
    this.widthForm = this.formBuilder.group({
      new_b: ['', Validators.required],
      hauteursTab: this.formBuilder.array([]),
      prixMetTab: this.formBuilder.array([]),
      prixPiecTab: this.formBuilder.array([]),
      prixFixTab: this.formBuilder.array([])
    });
  }

  getHeightArray() {
    return this.widthForm.get('hauteursTab') as FormArray;
  }
  getMpriceArray() {
    return this.widthForm.get('prixMetTab') as FormArray;
  }
  getPpriceArray() {
    return this.widthForm.get('prixPiecTab') as FormArray;
  }
  getFpriceArray() {
    return this.widthForm.get('prixFixTab') as FormArray;
  }

  onAddHeight() {
    let newControlH = this.formBuilder.control('');
    let newControlM = this.formBuilder.control('');
    let newControlP = this.formBuilder.control('');
    let newControlF = this.formBuilder.control('');
    this.getHeightArray().controls.push(newControlH);
    this.getMpriceArray().controls.push(newControlM);
    this.getPpriceArray().controls.push(newControlP);
    this.getFpriceArray().controls.push(newControlF);
  }

  onRemoveHeight(index: number) {
    this.getHeightArray().removeAt(index);
    this.getMpriceArray().removeAt(index);
    this.getPpriceArray().removeAt(index);
    this.getFpriceArray().removeAt(index);
```

```
  }

  onSubmitForm() {
    let newWidth = new Width(this.widthForm.get('new_b').value);
    for (let control of this.getHeightArray().controls) {
      newWidth.h.push(control.value);
      }
    for (let control of this.getMpriceArray().controls) {
      newWidth.m.push(control.value);
    }
    for (let control of this.getPpriceArray().controls) {
      newWidth.p.push(control.value);
    }
    for (let control of this.getFpriceArray().controls) {
      newWidth.f.push(control.value);
    }
    this.rangesService.addWidth(this.range_index, newWidth).then(result => {
      this.navCtrl.pop();
    });
  }
}
```

Code B.18 – `width-form.ts`

```
<ion-header charset=UTF-8>
  <ion-navbar>
    <ion-title>Nouvelle largeur pour la plage {{range_name}}</ion-title>
  </ion-navbar>
</ion-header>

<ion-content padding>
  <form *ngIf="widthForm" [formGroup]="widthForm">
    <ion-list>
      <ion-item-divider class="dividers">Nouvelle largeur</ion-item-divider>
      <ion-item>
        <ion-input placeholder="Largeur [cm]" formControlName="new_b" type="
    number" class="placeholders"></ion-input>
      </ion-item>

      <ion-item-divider class="dividers">Hauteurs associées</ion-item-divider>
      <div formArrayName="hauteursTab">
        <ion-item *ngFor="let controlh of getHeightArray().controls; let i_h =
    index">
          <ion-icon item-end
                    name="remove-circle"
                    color="danger"
                    (click)="onRemoveHeight(i_h)">
          </ion-icon>
          <ion-input [formControlName]="i_h" placeholder="Hauteur {{i_h+1}}"
    type="number" class="placeholders"></ion-input>
        </ion-item>
      </div>

      <ion-item-divider class="dividers">Prix/mètre associés aux hauteurs</
    ion-item-divider>
      <div formArrayName="prixMetTab">
        <ion-item *ngFor="let controlm of getMpriceArray().controls; let i_m =
    index">
          <ion-input [formControlName]="i_m" placeholder="Prix/mètre pour la
    hauteur {{i_m+1}}" type="number" class="placeholders"></ion-input>
        </ion-item>
      </div>

```

```
34        <ion-item-divider class="dividers">Prix/pièce associés aux hauteurs</
      ion-item-divider>
35        <div formArrayName="prixPiecTab">
36          <ion-item *ngFor="let controlp of getPpriceArray().controls; let i_p =
      index">
37            <ion-input [formControlName]="i_p" placeholder="Prix/pièce pour la
      hauteur {{i_p+1}}" type="number" class="placeholders"></ion-input>
38          </ion-item>
39        </div>
40
41        <ion-item-divider class="dividers">Prix/fixation associés aux hauteurs</
      ion-item-divider>
42        <div formArrayName="prixFixTab">
43          <ion-item *ngFor="let controlf of getFpriceArray().controls; let i_f =
      index">
44            <ion-input [formControlName]="i_f" placeholder="Prix/fixation pour la
      hauteur {{i_f+1}}" type="number" class="placeholders"></ion-input>
45          </ion-item>
46        </div>
47
48      </ion-list>
49
50      <button ion-button full (click)="onAddHeight()">Ajouter une hauteur</button>
51      <button ion-button full color="secondary" (click)="onSubmitForm()">
      Enregistrer</button>
52    </form>
53  </ion-content>
```

Code B.19 – `width-form.html`

B.3.8 Formulaire d'ajout d'une hauteur associée à une largeur d'une plage de sections

```
1  // ANGULAR ELEMENTS
2  import { Component, OnInit } from '@angular/core';
3  import { NavController, NavParams } from 'ionic-angular';
4  import { FormBuilder, FormGroup, Validators } from '@angular/forms';
5  // MODELS & SERVICES
6  import { RangesService } from '../../../services/ranges.service';
7
8
9  @Component({
10   selector: 'page-height-form',
11   templateUrl: './height-form.html'
12 })
13
14 export class HeightFormPage implements OnInit {
15
16   range_index: number;
17   width_index: number;
18   b_value: number;
19   range_name: String;
20   heightForm: FormGroup;
21
22   constructor(public navCtrl: NavController,
23               public navParams: NavParams,
24               private formBuilder: FormBuilder,
25               public rangesService: RangesService) {}
26
27   ngOnInit() {
28     this.width_index = this.navParams.get('width_index');
29     this.range_index = this.navParams.get('range_index');
```

```
30      this.b_value = this.navParams.get('b_value');
31      this.range_name = this.navParams.get('range_name');
32      this.rangesService.getRangesList().then(result => {
33        this.initForm();
34      });
35    }
36
37    initForm() {
38      this.heightForm = this.formBuilder.group({
39        h_new: ['', Validators.required],
40        m_new: ['', Validators.required],
41        p_new: ['', Validators.required],
42        f_new: ['', Validators.required],
43      });
44    }
45
46    onSubmitForm() {
47      let new_hmpf = [this.heightForm.get('h_new').value, this.heightForm.get('
        m_new').value, this.heightForm.get('p_new').value, this.heightForm.get('
        f_new').value] ;
48      this.rangesService.addHeight(this.range_index, this.width_index, new_hmpf).
        then(result => {
49        this.navCtrl.pop();
50      });
51    }
52  }
```

Code B.20 – height-form.ts

```
1   <ion-header charset=UTF-8>
2     <ion-navbar>
3       <ion-title>Nouvelle hauteur pour b={{b_value}}[cm] de la plage {{range_name
        }}</ion-title>
4     </ion-navbar>
5   </ion-header>
6
7   <ion-content padding>
8     <form *ngIf="heightForm" [formGroup]="heightForm">
9       <ion-list>
10        <ion-item-divider class="dividers">Nouvelle hauteur</ion-item-divider>
11        <ion-item>
12          <ion-input placeholder="Hauteur [cm]" formControlName="h_new" type="
        number" class="placeholders"></ion-input>
13        </ion-item>
14
15        <ion-item-divider class="dividers">Prix associés</ion-item-divider>
16        <ion-item>
17          <ion-input placeholder="Prix par mètre [€]" formControlName="m_new" type
        ="number" class="placeholders"></ion-input>
18        </ion-item>
19        <ion-item>
20          <ion-input placeholder="Prix par pièce [€]" formControlName="p_new" type
        ="number" class="placeholders"></ion-input>
21        </ion-item>
22        <ion-item>
23          <ion-input placeholder="Prix par fixation [€]" formControlName="f_new"
        type="number" class="placeholders"></ion-input>
24        </ion-item>
25
26      </ion-list>
27
28      <button ion-button full color="secondary" (click)="onSubmitForm()">
        Enregistrer</button>
29    </form>
```

```
</ion-content>
```

CODE B.21 – height-form.html

B.3.9 Formulaire des paramètres pour un calcul basé sur les Eurocodes

```
// ANGULAR ELEMENTS
import { Component } from '@angular/core';
import { IonicPage, NavController, NavParams } from 'ionic-angular';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
// MODELS & SERVICES
import { ProblemParams } from '../../models/ProblemParams';
import { Range } from '../../models/Range';
import { RangesService } from '../../services/ranges.service';
import { EurocodesService } from '../../services/eurocodes.service'
// PAGES
import { SolutionPage } from '../solution/solution'


@IonicPage()
@Component({
  selector: 'page-calculrap',
  templateUrl: 'calculrap.html',
})
export class CalculrapPage {

  parameters: ProblemParams = new ProblemParams();

  calcRapForm: FormGroup;

  rangesList: Range[];

  if_interBeam: boolean; if_hmax: boolean; if_emin: boolean; if_sens: boolean;

  constructor(public navCtrl: NavController,
              public navParams: NavParams,
              private formBuilder: FormBuilder,
              private rangesService: RangesService,
              private eurocodesService: EurocodesService) {}

  ionViewWillEnter() {
    this.rangesService.getRangesList().then((ranges: Range[]) => {
      if (ranges) {
        this.rangesList = ranges.slice();
        this.initForm();
      }
    });
    this.if_interBeam = false;
    this.if_hmax = false;
    this.if_emin = false;
    this.if_sens = false;
  }

  initForm() {
    this.calcRapForm = this.formBuilder.group({
      dim1: ['', Validators.required],
      dim2: ['', Validators.required],
      classeService: ['', Validators.required],
      categUtil: ['', Validators.required],
      chargePermG: ['', Validators.required],
      entraxeMax: ['', Validators.required],
      ninst: ['', Validators.required],
```

```
57        nfin: ['', Validators.required],
58        ksys: ['', Validators.required],
59        range_index: ['', Validators.required],
60        classeBois: ['', Validators.required],
61        range_index_inter: [''],
62        classeBois_inter: [''],
63        h_max: [''],
64        e_min: [''],
65        sensUnic: ['']
66      });
67    }
68
69    reInitialisation() {
70      this.initForm();
71      this.if_interBeam = false;
72      this.if_hmax = false;
73      this.if_sens = false;
74    }
75
76    onSubmitForm() {
77
78      // PARAMETRES GEOMETRIQUES
79      if (this.calcRapForm.get('dim1').value == this.calcRapForm.get('dim2').value
        ) {
80        this.parameters.dimLP = [this.calcRapForm.get('dim1').value, this.
        calcRapForm.get('dim2').value];
81      }
82      else {
83        this.parameters.dimLP = [Math.min(this.calcRapForm.get('dim1').value, this
        .calcRapForm.get('dim2').value), Math.max(this.calcRapForm.get('dim1').value
        , this.calcRapForm.get('dim2').value)] ;
84      }
85      this.parameters.entraxeMax = Number(this.calcRapForm.get('entraxeMax').value
        );
86      // PROPRIETES DU BOIS
87      this.parameters.classeBois = this.calcRapForm.get('classeBois').value;
88      this.parameters.cat = this.eurocodesService.classesBois[this.calcRapForm.get
        ('classeBois').value].cat;
89      this.parameters.fmk = this.eurocodesService.classesBois[this.calcRapForm.get
        ('classeBois').value].fmk;
90      this.parameters.fvk = this.eurocodesService.classesBois[this.calcRapForm.get
        ('classeBois').value].fvk;
91      this.parameters.modulE = this.eurocodesService.classesBois[this.calcRapForm.
        get('classeBois').value].modulE;
92      this.parameters.modulG = this.eurocodesService.classesBois[this.calcRapForm.
        get('classeBois').value].modulG;
93      // COEFFICIENTS CALCULS
94      this.parameters.gammaM = this.eurocodesService.gammaM[this.parameters.cat];
95      this.parameters.kdef = this.eurocodesService.kdef[this.calcRapForm.get('
        classeService').value];
96      this.parameters.kcr = this.eurocodesService.kcr;
97      if (this.calcRapForm.get('categUtil').value.indexOf('e') >= 0){
98        this.parameters.kmod = this.eurocodesService.kmod[this.calcRapForm.get('
        classeService').value].lt;
99      }
100     else {
101       this.parameters.kmod = this.eurocodesService.kmod[this.calcRapForm.get('
        classeService').value].mt;
102     }
103     if (this.calcRapForm.get('ksys').value == 0){
104       this.parameters.ksys = this.eurocodesService.ksys;
105     }
106     else {
```

```
107         this.parameters.ksys = Number(this.calcRapForm.get('ksys').value);
108       }
109       // COEFFICIENTS DE FLECHE
110       this.parameters.ninst = Number(this.calcRapForm.get('ninst').value);
111       this.parameters.nfin = Number(this.calcRapForm.get('nfin').value);
112       // COEFFICIENTS COMBINAISONS
113       this.parameters.gammaG = this.eurocodesService.gammaG;
114       this.parameters.gammaQ = this.eurocodesService.gammaQ;
115       this.parameters.psi0 = this.eurocodesService.categUtil[this.calcRapForm.get(
          'categUtil').value].psi0;
116       this.parameters.psi1 = this.eurocodesService.categUtil[this.calcRapForm.get(
          'categUtil').value].psi1;
117       this.parameters.psi2 = this.eurocodesService.categUtil[this.calcRapForm.get(
          'categUtil').value].psi2;
118       // CHARGES
119       this.parameters.chargePermG = Number(this.calcRapForm.get('chargePermG').
          value);
120       this.parameters.chargeVarQ = this.eurocodesService.categUtil[this.
          calcRapForm.get('categUtil').value].q;
121       // PLAGE DE SECTIONS
122       this.parameters.range_index = this.calcRapForm.get('range_index').value;
123       // POUTRE INTERMEDIAIRE
124       if (this.if_interBeam) {
125         this.parameters.inter_beam = true;
126         this.parameters.range_index_inter = this.calcRapForm.get('
          range_index_inter').value;
127         this.parameters.classeBois_inter = this.calcRapForm.get('classeBois_inter'
          ).value;
128         this.parameters.cat_inter = this.eurocodesService.classesBois[this.
          calcRapForm.get('classeBois_inter').value].cat;
129         this.parameters.fmk_inter = this.eurocodesService.classesBois[this.
          calcRapForm.get('classeBois_inter').value].fmk;
130         this.parameters.fvk_inter = this.eurocodesService.classesBois[this.
          calcRapForm.get('classeBois_inter').value].fvk;
131         this.parameters.modulE_inter = this.eurocodesService.classesBois[this.
          calcRapForm.get('classeBois_inter').value].modulE;
132         this.parameters.modulG_inter = this.eurocodesService.classesBois[this.
          calcRapForm.get('classeBois_inter').value].modulG;
133       }
134       else {
135         this.parameters.inter_beam = false;
136         this.parameters.range_index_inter = -1;
137         this.parameters.classeBois_inter = "no-inter";
138         this.parameters.cat_inter = "no-inter";
139         this.parameters.fmk_inter = 0;
140         this.parameters.fvk_inter = 0;
141         this.parameters.modulE_inter = 0;
142         this.parameters.modulG_inter = 0;
143       }
144       // RESTRICTIONS
145       if (this.if_hmax) {this.parameters.hmax = Number(this.calcRapForm.get('h_max
          ').value);}
146       else {this.parameters.hmax = 1000;}
147       if (this.if_emin) {this.parameters.emin = Number(this.calcRapForm.get('e_min
          ').value);}
148       else {this.parameters.emin = 0;}
149       if (this.if_sens) {this.parameters.sensUnic = this.calcRapForm.get('sensUnic
          ').value;}
150       else {this.parameters.sensUnic = "";}
151
152       // AFFICHER RESULTATS
153       this.navCtrl.push(SolutionPage, {parameters_set: this.parameters});
154     }
```

```
}
```

Code B.22 – calculrap.ts

```
<ion-header charset=UTF-8>

  <ion-navbar>
        <button ion-button menuToggle>
            <ion-icon name="menu"></ion-icon>
        </button>

        <ion-title>CALCUL RAPIDE</ion-title>
  </ion-navbar>

</ion-header>

<ion-content padding>

  <form *ngIf="calcRapForm" [formGroup]="calcRapForm">
    <ion-list>

      <ion-item-divider class="dividers">Dimensions du gîtage en [m]</ion-item-divider>
      <ion-item>
        <ion-label fixed class="labels">Largeur = </ion-label>
        <ion-input placeholder="0.00" formControlName="dim1" type="number" min="0" step="0.01" class="placeholders"></ion-input>
      </ion-item>
      <ion-item>
        <ion-label fixed class="labels">Longueur = </ion-label>
        <ion-input placeholder="0.00" formControlName="dim2" type="number" min="0" step="0.01" class="placeholders"></ion-input>
      </ion-item>

      <ion-item-divider class="dividers">Classe de service</ion-item-divider>
      <ion-item>
        <ion-select interface="popover" formControlName="classeService" placeholder="Sélectionner la classe de service" class="customSelect">
          <ion-option value="cs1" text-wrap>Classe de service 1</ion-option>
          <ion-option value="cs2" text-wrap>Classe de service 2</ion-option>
          <ion-option value="cs3" text-wrap>Classe de service 3</ion-option>
        </ion-select>
      </ion-item>

      <ion-item-divider class="dividers">Catégorie d'utilisation</ion-item-divider>
      <ion-item>
        <ion-select interface="popover" formControlName="categUtil" placeholder="Sélectionner la catégorie d'utilisation" class="customSelect">
          <ion-option value="a">A: Résidentiel</ion-option>
          <ion-option value="b">B: Bureaux</ion-option>
          <ion-option value="c1">C1: Espaces équipés de tables</ion-option>
          <ion-option value="c2">C3: Espaces équipés de sièges fixes</ion-option>
          <ion-option value="c3">C3: Espaces ne présentant pas d'obstacle</ion-option>
          <ion-option value="c4">C4: Espaces permettant des activités physiques</ion-option>
          <ion-option value="c5">C5: Espaces susceptibles d'acceuillir des foules</ion-option>
          <ion-option value="d1">D1: Commerces de détail</ion-option>
          <ion-option value="d2">D2: Grands magasins</ion-option>
          <ion-option value="e1">E1: Espaces de stockage</ion-option>
          <ion-option value="e2">E2: Usage industriel</ion-option>
```

```
51          <ion-option value="h">H: Toitures inaccessibles</ion-option>
52        </ion-select>
53      </ion-item>
54
55      <ion-item-divider class="dividers">Charge permanente en [kN]</
    ion-item-divider>
56      <ion-item>
57        <ion-label fixed class="labels">G = </ion-label>
58        <ion-input placeholder="1.00" formControlName="chargePermG" type="number
    " min="0" step="0.1" class="placeholders"></ion-input> [kN]
59      </ion-item>
60
61      <ion-item-divider class="dividers">Entraxe maximal en [cm]</
    ion-item-divider>
62      <ion-item>
63        <ion-label fixed class="labels">e<sub>max</sub> = </ion-label>
64        <ion-input placeholder="80.0" formControlName="entraxeMax" type="number"
     min="0" step="1" class="placeholders"></ion-input> [cm]
65      </ion-item>
66
67      <ion-item-divider class="dividers">Limite de flèche &eta;</
    ion-item-divider>
68      <ion-item>
69        <ion-label fixed class="labels">&eta;<sub>inst</sub> = </ion-label>
70        <ion-input placeholder="500" formControlName="ninst" type="number" min="
    0" step="10" class="placeholders"></ion-input> [cm]
71      </ion-item>
72      <ion-item>
73        <ion-label fixed class="labels">&eta;<sub>fin</sub> = </ion-label>
74        <ion-input placeholder="300" formControlName="nfin" type="number" min="0
    " step="10" class="placeholders"></ion-input> [cm]
75      </ion-item>
76
77      <ion-item-divider class="dividers">Effet système</ion-item-divider>
78      <ion-item>
79        <ion-select interface="popover" formControlName="ksys" placeholder="Sé
    lectionner la valeur de k_sys" class="customSelect">
80          <ion-option value="0" text-wrap>automatique</ion-option>
81          <ion-option value="1" text-wrap>1.00</ion-option>
82          <ion-option value="1.1" text-wrap>1.10</ion-option>
83        </ion-select>
84      </ion-item>
85
86      <ion-item-divider class="dividers">Plage de sections à utiliser</
    ion-item-divider>
87      <ion-item>
88        <ion-select interface="popover"
89                    formControlName="range_index"
90                    placeholder="Sélectionner la plage"
91                    class="customSelect">
92          <ion-option *ngFor="let range of rangesList; let i = index"
93                      [value]="i">
94                      {{range.name}}
95          </ion-option>
96        </ion-select>
97      </ion-item>
98      <ion-item-divider class="dividers">Classe de résistance du bois</
    ion-item-divider>
99      <ion-item>
100       <ion-select interface="popover" formControlName="classeBois" placeholder
    ="Sélectionner la classe de résistance" class="customSelect">
101         <ion-option disabled class="option-group-title">Bois massif - ré
    sineux</ion-option>
```

```
102             <ion-option value="c14">C14</ion-option>
103             <ion-option value="c16">C16</ion-option>
104             <ion-option value="c18">C18</ion-option>
105             <ion-option value="c20">C20</ion-option>
106             <ion-option value="c22">C22</ion-option>
107             <ion-option value="c24">C24</ion-option>
108             <ion-option value="c27">C27</ion-option>
109             <ion-option value="c30">C30</ion-option>
110             <ion-option value="c35">C35</ion-option>
111             <ion-option value="c40">C40</ion-option>
112             <ion-option value="c45">C45</ion-option>
113             <ion-option value="c50">C50</ion-option>
114
115             <ion-option disabled class="option-group-title">Bois massif -
    feuillus</ion-option>
116             <ion-option value="d18">D18</ion-option>
117             <ion-option value="d24">D24</ion-option>
118             <ion-option value="d27">D27</ion-option>
119             <ion-option value="d30">D30</ion-option>
120             <ion-option value="d35">D35</ion-option>
121             <ion-option value="d40">D40</ion-option>
122             <ion-option value="d50">D50</ion-option>
123             <ion-option value="d60">D60</ion-option>
124             <ion-option value="d70">D70</ion-option>
125
126             <ion-option disabled class="option-group-title">Bois lamellés-collés
    </ion-option>
127             <ion-option value="gl20h">GL20h</ion-option>
128             <ion-option value="gl22h">GL22h</ion-option>
129             <ion-option value="gl24h">GL24h</ion-option>
130             <ion-option value="gl26h">GL26h</ion-option>
131             <ion-option value="gl28h">GL28h</ion-option>
132             <ion-option value="gl30h">GL30h</ion-option>
133             <ion-option value="gl32h">GL32h</ion-option>
134         </ion-select>
135       </ion-item>
136
137
138
139       <ion-item>
140         <ion-label>Autre plage (poutre intermédiaire)</ion-label>
141         <ion-checkbox [(ngModel)]="if_interBeam" [ngModelOptions]="{standalone:
    true}" [checked]="if_interBeam" [disabled]="if_sens"></ion-checkbox>
142       </ion-item>
143       <ion-item *ngIf="if_interBeam">
144         <ion-select interface="popover"
145                     formControlName="range_index_inter"
146                     placeholder="Sélectionner l'autre plage"
147                     class="customSelect">
148           <ion-option *ngFor="let range of rangesList; let i = index"
149                       [value]="i">
150                       {{range.name}}
151           </ion-option>
152         </ion-select>
153       </ion-item>
154       <ion-item *ngIf="if_interBeam">
155         <ion-select interface="popover" formControlName="classeBois_inter"
    placeholder="Sélectionner la classe de résistance" class="customSelect">
156             <ion-option disabled class="option-group-title">Bois massif - ré
    sineux</ion-option>
157             <ion-option value="c14">C14</ion-option>
158             <ion-option value="c16">C16</ion-option>
159             <ion-option value="c18">C18</ion-option>
```

```
160             <ion-option value="c20">C20</ion-option>
161             <ion-option value="c22">C22</ion-option>
162             <ion-option value="c24">C24</ion-option>
163             <ion-option value="c27">C27</ion-option>
164             <ion-option value="c30">C30</ion-option>
165             <ion-option value="c35">C35</ion-option>
166             <ion-option value="c40">C40</ion-option>
167             <ion-option value="c45">C45</ion-option>
168             <ion-option value="c50">C50</ion-option>
169
170             <ion-option disabled class="option-group-title">Bois massif -
    feuillus</ion-option>
171             <ion-option value="d18">D18</ion-option>
172             <ion-option value="d24">D24</ion-option>
173             <ion-option value="d27">D27</ion-option>
174             <ion-option value="d30">D30</ion-option>
175             <ion-option value="d35">D35</ion-option>
176             <ion-option value="d40">D40</ion-option>
177             <ion-option value="d50">D50</ion-option>
178             <ion-option value="d60">D60</ion-option>
179             <ion-option value="d70">D70</ion-option>
180
181             <ion-option disabled class="option-group-title">Bois lamellés-collés
    </ion-option>
182             <ion-option value="gl20h">GL20h</ion-option>
183             <ion-option value="gl22h">GL22h</ion-option>
184             <ion-option value="gl24h">GL24h</ion-option>
185             <ion-option value="gl26h">GL26h</ion-option>
186             <ion-option value="gl28h">GL28h</ion-option>
187             <ion-option value="gl30h">GL30h</ion-option>
188             <ion-option value="gl32h">GL32h</ion-option>
189         </ion-select>
190       </ion-item>
191
192       <ion-item>
193         <ion-label>Limitation de la hauteur des poutres en [cm]</ion-label>
194         <ion-checkbox [(ngModel)]="if_hmax" [ngModelOptions]="{standalone: true}
    " [checked]="if_hmax"></ion-checkbox>
195       </ion-item>
196       <ion-item *ngIf="if_hmax">
197         <ion-label fixed class="labels">h<sub>max</sub> = </ion-label>
198         <ion-input placeholder="0.00" formControlName="h_max" type="number" min=
    "0" step="0.1" class="placeholders"></ion-input>
199       </ion-item>
200
201       <ion-item>
202         <ion-label>Entraxe minimum en [cm]</ion-label>
203         <ion-checkbox [(ngModel)]="if_emin" [ngModelOptions]="{standalone: true}
    " [checked]="if_emin"></ion-checkbox>
204       </ion-item>
205       <ion-item *ngIf="if_emin">
206         <ion-label fixed class="labels">e<sub>min</sub> = </ion-label>
207         <ion-input placeholder="0.00" formControlName="e_min" type="number" min=
    "0" step="0.5" class="placeholders"></ion-input>
208       </ion-item>
209
210       <ion-item>
211         <ion-label>Restriction sur le sens de la portée</ion-label>
212         <ion-checkbox [(ngModel)]="if_sens" [ngModelOptions]="{standalone: true}
    " [checked]="if_sens" [disabled]="if_interBeam"></ion-checkbox>
213       </ion-item>
214       <ion-item *ngIf="if_sens">
215         <ion-select interface="popover" formControlName="sensUnic" placeholder="
```

```
    Sélectionner le sens de la portée" class="customSelect">
          <ion-option value="long" text-wrap>Sens le plus long</ion-option>
          <ion-option value="court" text-wrap>Sens le plus court</ion-option>
        </ion-select>
      </ion-item>


    </ion-list>

    <button ion-button full color="secondary" (click)="onSubmitForm()">Calculer
    </button>
    <button ion-button full color="danger" (click)="reInitialisation()">Ré
    initialiser</button>
  </form>

</ion-content>
```

Code B.23 – calculrap.html

B.3.10 Formulaire des paramètres pour un calcul personnalisé

```
// ANGULAR ELEMENTS
import { Component } from '@angular/core';
import { IonicPage, NavController, NavParams } from 'ionic-angular';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
// MODELS & SERVICES
import { ProblemParams } from '../../models/ProblemParams';
import { Range } from '../../models/Range';
import { RangesService } from '../../services/ranges.service';
// PAGES
import { SolutionPage } from '../solution/solution'


@IonicPage()
@Component({
  selector: 'page-calculpers',
  templateUrl: 'calculpers.html',
})
export class CalculpersPage {

  parameters: ProblemParams = new ProblemParams();

  calcPersForm: FormGroup;

  rangesList: Range[];

  if_interBeam: boolean; if_hmax: boolean; if_emin: boolean; if_sens: boolean;

  constructor(public navCtrl: NavController,
              public navParams: NavParams,
              private formBuilder: FormBuilder,
              private rangesService: RangesService) {}

  ionViewWillEnter() {
    this.rangesService.getRangesList().then((ranges: Range[]) => {
      if (ranges) {
        this.rangesList = ranges.slice();
        this.initForm();
      }
    });
    this.if_interBeam = false;
    this.if_hmax = false;
```

```
42      this.if_emin = false;
43      this.if_sens = false;
44    }
45
46    initForm() {
47      this.calcPersForm = this.formBuilder.group({
48        dim1: ['', Validators.required],
49        dim2: ['', Validators.required],
50        fmk: ['', Validators.required],
51        fvk: ['', Validators.required],
52        modE: ['', Validators.required],
53        modG: ['', Validators.required],
54        cat: ['', Validators.required],
55        chargeVarQ: ['', Validators.required],
56        chargePermG: ['', Validators.required],
57        entraxeMax: ['', Validators.required],
58        gammaG: ['', Validators.required],
59        gammaQ: ['', Validators.required],
60        psi0: ['', Validators.required],
61        psi1: ['', Validators.required],
62        psi2: ['', Validators.required],
63        ninst: ['', Validators.required],
64        nfin: ['', Validators.required],
65        gammaM: ['', Validators.required],
66        kdef: ['', Validators.required],
67        kmod: ['', Validators.required],
68        kcr: ['', Validators.required],
69        ksys: ['', Validators.required],
70        range_index: ['', Validators.required],
71        fmk_inter: [''],
72        fvk_inter: [''],
73        modE_inter: [''],
74        modG_inter: [''],
75        cat_inter: [''],
76        range_index_inter: [''],
77        h_max: [''],
78        e_min: [''],
79        sensUnic: ['']
80      });
81    }
82
83    reInitialisation() {
84      this.initForm();
85      this.if_interBeam = false;
86      this.if_hmax = false;
87      this.if_sens = false;
88    }
89
90    onSubmitForm() {
91
92      // PARAMETRES GEOMETRIQUES
93      if (this.calcPersForm.get('dim1').value == this.calcPersForm.get('dim2').
        value) {
94        this.parameters.dimLP = [this.calcPersForm.get('dim1').value, this.
        calcPersForm.get('dim2').value];
95      }
96      else {
97        this.parameters.dimLP = [Math.min(this.calcPersForm.get('dim1').value,
        this.calcPersForm.get('dim2').value), Math.max(this.calcPersForm.get('dim1')
        .value, this.calcPersForm.get('dim2').value)] ;
98      }
99      this.parameters.entraxeMax = this.calcPersForm.get('entraxeMax').value;
100     // PROPRIETES DU BOIS
```

```
101       this.parameters.classeBois = 'personnalisée';
102       this.parameters.cat = this.calcPersForm.get('cat').value;
103       this.parameters.fmk = this.calcPersForm.get('fmk').value;
104       this.parameters.fvk = this.calcPersForm.get('fvk').value;
105       this.parameters.modulE = this.calcPersForm.get('modE').value;
106       this.parameters.modulG = this.calcPersForm.get('modG').value;
107       // COEFFICIENTS CALCULS
108       this.parameters.gammaM = this.calcPersForm.get('gammaG').value;
109       this.parameters.kdef = this.calcPersForm.get('kdef').value;
110       this.parameters.kmod = this.calcPersForm.get('kmod').value;
111       this.parameters.kcr = this.calcPersForm.get('kcr').value;
112       this.parameters.ksys = this.calcPersForm.get('ksys').value;
113       // COEFFICIENTS COMBINAISONS
114       this.parameters.gammaG = this.calcPersForm.get('gammaG').value;
115       this.parameters.gammaQ = this.calcPersForm.get('gammaQ').value;
116       this.parameters.psi0 = this.calcPersForm.get('psi0').value;
117       this.parameters.psi1 = this.calcPersForm.get('psi1').value;
118       this.parameters.psi2 = this.calcPersForm.get('psi2').value;
119       // COEFFICIENTS DE FLECHE
120       this.parameters.ninst = this.calcPersForm.get('ninst').value;
121       this.parameters.nfin = this.calcPersForm.get('nfin').value;
122       // CHARGES
123       this.parameters.chargePermG = this.calcPersForm.get('chargePermG').value;
124       this.parameters.chargeVarQ = this.calcPersForm.get('chargeVarQ').value;
125       // PLAGE DE SECTIONS
126       this.parameters.range_index = this.calcPersForm.get('range_index').value;
127       // POUTRE INTERMEDIAIRE
128       this.parameters.inter_beam = this.if_interBeam;
129       if (this.if_interBeam) {
130         this.parameters.range_index_inter = this.calcPersForm.get('
      range_index_inter').value;
131         this.parameters.classeBois_inter = 'personnalisée';
132         this.parameters.cat_inter = this.calcPersForm.get('cat_inter').value;
133         this.parameters.fmk_inter = this.calcPersForm.get('fmk_inter').value;
134         this.parameters.fvk_inter = this.calcPersForm.get('fvk_inter').value;
135         this.parameters.modulE_inter = this.calcPersForm.get('modE_inter').value;
136         this.parameters.modulG_inter = this.calcPersForm.get('modG_inter').value;
137       }
138       else {
139         this.parameters.range_index_inter = -1;
140         this.parameters.classeBois_inter = "no-inter";
141         this.parameters.cat_inter = "no-inter";
142         this.parameters.fmk_inter = 0;
143         this.parameters.fvk_inter = 0;
144         this.parameters.modulE_inter = 0;
145         this.parameters.modulG_inter = 0;
146       }
147       // RESTRICTIONS
148       if (this.if_hmax) {this.parameters.hmax = Number(this.calcPersForm.get('
      h_max').value);}
149       else {this.parameters.hmax = 1000;}
150       if (this.if_emin) {this.parameters.emin = Number(this.calcPersForm.get('
      e_min').value);}
151       else {this.parameters.emin = 0;}
152       if (this.if_sens) {this.parameters.sensUnic = this.calcPersForm.get('
      sensUnic').value;}
153       else {this.parameters.sensUnic = "";}
154
155       // AFFICHER RESULTATS
156       this.navCtrl.push(SolutionPage, {parameters_set: this.parameters});
157     }
158 }
```

Code B.24 – `calculpers.ts`

```
<ion-header charset=UTF-8>

  <ion-navbar>
        <button ion-button menuToggle>
            <ion-icon name="menu"></ion-icon>
        </button>

        <ion-title>CALCUL PERSONNALISE</ion-title>
  </ion-navbar>

</ion-header>



<ion-content padding>

  <form *ngIf="calcPersForm" [formGroup]="calcPersForm">
    <ion-list>

      <ion-item-divider class="dividers">Dimensions du gîtage en [m]</
    ion-item-divider>
      <ion-item>
        <ion-label fixed class="labels">Largeur = </ion-label>
        <ion-input placeholder="0.00" formControlName="dim1" type="number" min="
    0" step="0.01" class="placeholders"></ion-input>
      </ion-item>
      <ion-item>
        <ion-label fixed class="labels">Longueur = </ion-label>
        <ion-input placeholder="0.00" formControlName="dim2" type="number" min="
    0" step="0.01" class="placeholders"></ion-input>
      </ion-item>

      <ion-item-divider class="dividers">Charge variable en [kN]</
    ion-item-divider>
      <ion-item>
        <ion-label fixed class="labels">Q = </ion-label>
        <ion-input placeholder="0.00" formControlName="chargeVarQ" type="number"
     min="0" step="0.1" class="placeholders"></ion-input>
      </ion-item>

      <ion-item-divider class="dividers">Charge permanente en [kN]</
    ion-item-divider>
      <ion-item>
        <ion-label fixed class="labels">G = </ion-label>
        <ion-input placeholder="0.00" formControlName="chargePermG" type="number
    " min="0" step="0.1" class="placeholders"></ion-input>
      </ion-item>

      <ion-item-divider class="dividers">Entraxe maximal en [cm]</
    ion-item-divider>
      <ion-item>
        <ion-label fixed class="labels">e<sub>max</sub> = </ion-label>
        <ion-input placeholder="0.00" formControlName="entraxeMax" type="number"
     min="0" step="1" class="placeholders"></ion-input>
      </ion-item>

      <ion-item-divider class="dividers">Coefficients de combinaison de charge</
    ion-item-divider>
      <ion-item>
        <ion-label fixed class="labels">&gamma;<sub>G</sub> = </ion-label>
        <ion-input placeholder="0.00" formControlName="gammaG" type="number" min
    ="0" step="0.05" class="placeholders"></ion-input>
      </ion-item>
```

```
53        <ion-item>
54          <ion-label fixed class="labels">&gamma;<sub>Q</sub> = </ion-label>
55          <ion-input placeholder="0.00" formControlName="gammaQ" type="number" min
        ="0" step="0.05" class="placeholders"></ion-input>
56        </ion-item>
57        <ion-item>
58          <ion-label fixed class="labels">&psi;<sub>0</sub> = </ion-label>
59          <ion-input placeholder="0.00" formControlName="psi0" type="number" min="
        0" step="0.1" class="placeholders"></ion-input>
60        </ion-item>
61        <ion-item>
62          <ion-label fixed class="labels">&psi;<sub>1</sub> = </ion-label>
63          <ion-input placeholder="0.00" formControlName="psi1" type="number" min="
        0" step="0.1" class="placeholders"></ion-input>
64        </ion-item>
65        <ion-item>
66          <ion-label fixed class="labels">&psi;<sub>2</sub> = </ion-label>
67          <ion-input placeholder="0.00" formControlName="psi2" type="number" min="
        0" step="0.1" class="placeholders"></ion-input>
68        </ion-item>
69
70        <ion-item-divider class="dividers">Limite de flèche &eta;</
        ion-item-divider>
71        <ion-item>
72          <ion-label fixed class="labels">&eta;<sub>inst</sub> = </ion-label>
73          <ion-input placeholder="500" formControlName="ninst" type="number" min="
        0" step="10" class="placeholders"></ion-input>
74        </ion-item>
75        <ion-item>
76          <ion-label fixed class="labels">&eta;<sub>fin</sub> = </ion-label>
77          <ion-input placeholder="300" formControlName="nfin" type="number" min="0
        " step="10" class="placeholders"></ion-input>
78        </ion-item>
79
80        <ion-item-divider class="dividers">Autres coefficients de calcul des
        Eurocodes</ion-item-divider>
81        <ion-item>
82          <ion-label fixed class="labels">&gamma;<sub>M</sub> = </ion-label>
83          <ion-input placeholder="0.00" formControlName="gammaM" type="number" min
        ="0" step="0.05" class="placeholders"></ion-input>
84        </ion-item>
85        <ion-item>
86          <ion-label fixed class="labels">k<sub>def</sub> = </ion-label>
87          <ion-input placeholder="0.00" formControlName="kdef" type="number" min="
        0" step="0.1" class="placeholders"></ion-input>
88        </ion-item>
89        <ion-item>
90          <ion-label fixed class="labels">k<sub>mod</sub> = </ion-label>
91          <ion-input placeholder="0.00" formControlName="kmod" type="number" min="
        0" step="0.05" class="placeholders"></ion-input>
92        </ion-item>
93        <ion-item>
94          <ion-label fixed class="labels">k<sub>cr</sub> = </ion-label>
95          <ion-input placeholder="0.00" formControlName="kcr" type="number" min="0
        " step="0.01" class="placeholders"></ion-input>
96        </ion-item>
97        <ion-item>
98          <ion-label fixed class="labels">k<sub>sys</sub> = </ion-label>
99          <ion-input placeholder="0.00" formControlName="ksys" type="number" min="
        0" step="0.1" class="placeholders"></ion-input>
100       </ion-item>
101
102       <ion-item-divider class="dividers">Plage de sections à utiliser</
```

```
    ion-item-divider>
103       <ion-item>
104         <ion-select interface="popover"
105                     formControlName="range_index"
106                     placeholder="Sélectionner la plage">
107           <ion-option *ngFor="let range of rangesList; let i = index"
108                       [value]="i">
109                       {{range.name}}
110           </ion-option>
111         </ion-select>
112       </ion-item>
113       <ion-item-divider class="dividers">Matériau</ion-item-divider>
114       <ion-item>
115         <ion-select interface="popover" formControlName="cat" placeholder="Sé
    lectionner le matériau">
116           <ion-option value="massif">Bois massif</ion-option>
117           <ion-option value="blc">Bois lamellé-collé</ion-option>
118         </ion-select>
119       </ion-item>
120       <ion-item-divider class="dividers">Paramètres de résistance du bois en
    [MPa]</ion-item-divider>
121       <ion-item>
122         <ion-label fixed class="labels">f<sub>m,k</sub> = </ion-label>
123         <ion-input placeholder="0.00" formControlName="fmk" type="number" min="0
    " step="0.1" class="placeholders"></ion-input>
124       </ion-item>
125       <ion-item>
126         <ion-label fixed class="labels">f<sub>v,k</sub> = </ion-label>
127         <ion-input placeholder="0.00" formControlName="fvk" type="number" min="0
    " step="0.1" class="placeholders"></ion-input>
128       </ion-item>
129       <ion-item>
130         <ion-label fixed class="labels">E = </ion-label>
131         <ion-input placeholder="0.00" formControlName="modE" type="number" min="
    0" step="100" class="placeholders"></ion-input>
132       </ion-item>
133       <ion-item>
134         <ion-label fixed class="labels">G = </ion-label>
135         <ion-input placeholder="0.00" formControlName="modG" type="number" min="
    0" step="10" class="placeholders"></ion-input>
136       </ion-item>
137
138
139       <ion-item>
140         <ion-label>Autre plage (poutre intermédiaire)</ion-label>
141         <ion-checkbox [(ngModel)]="if_interBeam" [ngModelOptions]="{standalone:
    true}" [checked]="if_interBeam" [disabled]="if_sens"></ion-checkbox>
142       </ion-item>
143       <ion-item *ngIf="if_interBeam">
144         <ion-select interface="popover"
145                     formControlName="range_index_inter"
146                     placeholder="Sélectionner l'autre plage"
147                     class="customSelect">
148           <ion-option *ngFor="let range of rangesList; let i = index"
149                       [value]="i">
150                       {{range.name}}
151           </ion-option>
152         </ion-select>
153       </ion-item>
154       <ion-item *ngIf="if_interBeam">
155         <ion-select interface="popover" formControlName="cat_inter" placeholder=
    "Sélectionner le matériau">
156           <ion-option value="massif">Bois massif</ion-option>
```

```
157           <ion-option value="blc">Bois lamellé-collé</ion-option>
158         </ion-select>
159       </ion-item>
160       <ion-item *ngIf="if_interBeam">
161         <ion-label fixed class="labels">f<sub>m,k</sub> = </ion-label>
162         <ion-input placeholder="0.00" formControlName="fmk_inter" type="number"
    min="0" step="0.1" class="placeholders"></ion-input>
163       </ion-item>
164       <ion-item *ngIf="if_interBeam">
165         <ion-label fixed class="labels">f<sub>v,k</sub> = </ion-label>
166         <ion-input placeholder="0.00" formControlName="fvk_inter" type="number"
    min="0" step="0.1" class="placeholders"></ion-input>
167       </ion-item>
168       <ion-item *ngIf="if_interBeam">
169         <ion-label fixed class="labels">E = </ion-label>
170         <ion-input placeholder="0.00" formControlName="modE_inter" type="number"
     min="0" step="100" class="placeholders"></ion-input>
171       </ion-item>
172       <ion-item *ngIf="if_interBeam">
173         <ion-label fixed class="labels">G = </ion-label>
174         <ion-input placeholder="0.00" formControlName="modG_inter" type="number"
     min="0" step="10" class="placeholders"></ion-input>
175       </ion-item>
176
177       <ion-item>
178         <ion-label>Limitation de la hauteur des poutres en [cm]</ion-label>
179         <ion-checkbox [(ngModel)]="if_hmax" [ngModelOptions]="{standalone: true}
    " [checked]="if_hmax"></ion-checkbox>
180       </ion-item>
181       <ion-item *ngIf="if_hmax">
182         <ion-label fixed class="labels">h<sub>max</sub> = </ion-label>
183         <ion-input placeholder="0.00" formControlName="h_max" type="number" min=
    "0" step="0.05" class="placeholders"></ion-input>
184       </ion-item>
185
186       <ion-item>
187         <ion-label>Entraxe minimum en [cm]</ion-label>
188         <ion-checkbox [(ngModel)]="if_emin" [ngModelOptions]="{standalone: true}
    " [checked]="if_emin"></ion-checkbox>
189       </ion-item>
190       <ion-item *ngIf="if_emin">
191         <ion-label fixed class="labels">e<sub>min</sub> = </ion-label>
192         <ion-input placeholder="0.00" formControlName="e_min" type="number" min=
    "0" step="0.05" class="placeholders"></ion-input>
193       </ion-item>
194
195       <ion-item>
196         <ion-label>Restriction sur le sens de la portée</ion-label>
197         <ion-checkbox [(ngModel)]="if_sens" [ngModelOptions]="{standalone: true}
    " [checked]="if_sens" [disabled]="if_interBeam"></ion-checkbox>
198       </ion-item>
199       <ion-item *ngIf="if_sens">
200         <ion-select interface="popover" formControlName="sensUnic" placeholder="
    Sélectionner le sens de la portée" class="customSelect">
201           <ion-option value="long" text-wrap>Sens le plus long</ion-option>
202           <ion-option value="court" text-wrap>Sens le plus court</ion-option>
203         </ion-select>
204       </ion-item>
205
206     </ion-list>
207
208     <button ion-button full color="secondary" (click)="onSubmitForm()">Calculer
    </button>
```

```
209     <button ion-button full color="danger" (click)="reInitialisation()">Ré
        initialiser</button>
210   </form>
211
212 </ion-content>
```

Code B.25 – calculpers.html

B.3.11 Page des solutions optimales en termes de volume et de coût

```
1  // ANGULAR ELEMENTS
2  import { Component } from '@angular/core';
3  import { IonicPage, NavController, NavParams, AlertController } from 'ionic-
       angular';
4  // MODELS & SERVICES
5  import { Width } from '../../models/Width';
6  import { Range } from '../../models/Range';
7  import { RangesService } from '../../services/ranges.service';
8  import { EurocodesService } from '../../services/eurocodes.service'
9  import { ProblemParams } from '../../models/ProblemParams';
10
11
12 @IonicPage()
13 @Component({
14   selector: 'page-solution',
15   templateUrl: 'solution.html',
16 })
17 export class SolutionPage {
18
19   parameters: ProblemParams;
20
21   range_index: number;
22   range: Range;
23
24   inter_beam: boolean;
25   range_index_inter: number;
26   range_inter: Range;
27   range_inter_name: string;
28   if_sensUnic: boolean;
29
30   volArrayLong: (string|number)[];
31   volArrayCourt: (string|number)[];
32   volArrayLong_inter: (string|number)[];
33   volArrayCourt_inter: (string|number)[];
34   costArrayLong: (string|number)[];
35   costArrayCourt: (string|number)[];
36   costArrayLong_inter: (string|number)[];
37   costArrayCourt_inter: (string|number)[];
38
39   inter_beam_display_V: boolean;
40   inter_beam_display_C: boolean;
41
42   resultsVolArray: (string|number)[];
43   resultsCostArray: (string|number)[];
44   //[sens, volume, cost, b, h, number, entraxe, b_inter, h_inter]
45
46   volumeSelected: boolean; costSelected: boolean; paramSelected: boolean;
47
48   constructor(public navCtrl: NavController,
49               public navParams: NavParams,
50               private alertCtrl: AlertController,
51               private rangesService: RangesService,
```

```
52                private eurocodesService: EurocodesService) {
53      this.parameters = this.navParams.get('parameters_set');
54      this.range_index = this.navParams.get('parameters_set').range_index;
55      this.inter_beam = this.navParams.get('parameters_set').inter_beam;
56      this.range_index_inter = this.navParams.get('parameters_set').
        range_index_inter;
57      if (this.navParams.get('parameters_set').inter_beam) {this.range_inter_name
        = this.navParams.get('parameters_set').range_index_inter.name}
58      this.if_sensUnic = (this.navParams.get('parameters_set').sensUnic==="court"
        || this.navParams.get('parameters_set').sensUnic==="long");
59      this.range = {name:'', largeurs:[]};
60    }
61
62    ionViewWillEnter() {
63      this.rangesService.getRangesList().then((ranges: Range[]) => {
64        if (ranges) {
65          this.range = ranges[this.range_index];
66          if (this.inter_beam) {
67            this.range_inter = ranges[this.range_index_inter];
68          }
69        }
70      }).then(() => {
71        if (!this.if_sensUnic) {
72          this.volArrayCourt = this.calculOptimalVolume(this.parameters, this.
        range.largeurs, 'court');
73          this.volArrayLong = this.calculOptimalVolume(this.parameters, this.range
        .largeurs, 'long');
74          this.costArrayCourt = this.calculOptimalCost(this.parameters, this.range
        .largeurs, 'court');
75          this.costArrayLong = this.calculOptimalCost(this.parameters, this.range.
        largeurs, 'long');
76          if (this.inter_beam) {
77            this.volArrayLong_inter = this.calculOptimalVolume_Inter(this.
        parameters, this.range.largeurs, this.range_inter.largeurs, 'long');
78            this.volArrayCourt_inter = this.calculOptimalVolume_Inter(this.
        parameters, this.range.largeurs, this.range_inter.largeurs, 'court');
79            this.costArrayLong_inter = this.calculOptimalCost_Inter(this.
        parameters, this.range.largeurs, this.range_inter.largeurs, 'long');
80            this.costArrayCourt_inter = this.calculOptimalCost_Inter(this.
        parameters, this.range.largeurs, this.range_inter.largeurs, 'court');
81          }
82        }
83      }).then(() => {
84        if (this.if_sensUnic) {
85          this.resultsVolArray = this.calculOptimalVolume(this.parameters, this.
        range.largeurs, this.parameters.sensUnic);
86          this.resultsCostArray = this.calculOptimalCost(this.parameters, this.
        range.largeurs, this.parameters.sensUnic);
87        }
88        else {
89          let volArray: number[];
90          let costArray: number[];
91          if (this.inter_beam) {
92            volArray = [Number(this.volArrayCourt[1]), Number(this.volArrayLong
        [1]), Number(this.volArrayCourt_inter[1]), Number(this.volArrayLong_inter
        [1])];
93            costArray = [Number(this.costArrayCourt[2]), Number(this.costArrayLong
        [2]), Number(this.costArrayCourt_inter[2]), Number(this.costArrayLong_inter
        [2])];
94            if (this.indexOfMinimum(volArray)==0 && volArray[0]!=0){this.
        resultsVolArray = this.volArrayCourt; this.inter_beam_display_V = false;}
95            else if (this.indexOfMinimum(volArray)==1 && volArray[1]!=0){this.
        resultsVolArray = this.volArrayLong; this.inter_beam_display_V = false;}
```

```
96            else if (this.indexOfMinimum(volArray)==2 && volArray[2]!=0){this.
      resultsVolArray = this.volArrayCourt_inter; this.inter_beam_display_V = true
      ;}
97            else if (this.indexOfMinimum(volArray)==3 && volArray[3]!=0){this.
      resultsVolArray = this.volArrayLong_inter; this.inter_beam_display_V = true
      ;}
98            else {this.resultsVolArray = [0, 0, 0, 0, 0, 0, 0, 0, 0];}
99            if (this.indexOfMinimum(costArray)==0 && this.costArrayCourt[1]!=0){
      this.resultsCostArray = this.costArrayCourt; this.inter_beam_display_C =
      false;}
100           else if (this.indexOfMinimum(costArray)==1 && this.costArrayLong[1]!=0
      ){this.resultsCostArray = this.costArrayLong; this.inter_beam_display_C =
      false;}
101           else if (this.indexOfMinimum(costArray)==2 && this.
      costArrayCourt_inter[1]!=0){this.resultsCostArray = this.
      costArrayCourt_inter; this.inter_beam_display_C = true;}
102           else if (this.indexOfMinimum(costArray)==3 && this.costArrayLong_inter
      [1]!=0){this.resultsCostArray = this.costArrayLong_inter; this.
      inter_beam_display_C = true;}
103           else {this.resultsCostArray = [0, 0, 0, 0, 0, 0, 0, 0, 0];}
104         }
105         else {
106           volArray = [Number(this.volArrayCourt[1]), Number(this.volArrayLong
      [1])];
107           costArray = [Number(this.costArrayCourt[2]), Number(this.costArrayLong
      [2])];
108           if (this.indexOfMinimum(volArray)==0){this.resultsVolArray = this.
      volArrayCourt; this.inter_beam_display_V = false;}
109           else if (this.indexOfMinimum(volArray)==1){this.resultsVolArray = this
      .volArrayLong; this.inter_beam_display_V = false;}
110           else {this.resultsVolArray = [0, 0, 0, 0, 0, 0, 0, 0, 0];}
111           if (this.indexOfMinimum(costArray)==0 && this.costArrayCourt[1]!==0){
      this.resultsCostArray = this.costArrayCourt; this.inter_beam_display_C =
      false;}
112           else if (this.indexOfMinimum(costArray)==1 && this.costArrayLong[1]!
      ==0){this.resultsCostArray = this.costArrayLong; this.inter_beam_display_C =
       false;}
113           else {this.resultsCostArray = [0, 0, 0, 0, 0, 0, 0, 0, 0];}
114         }
115       }
116     });
117     this.volumeSelected = false;
118     this.costSelected = false;
119     this.paramSelected = false;
120   }
121
122   ionViewDidEnter() {
123     if (this.resultsVolArray && (this.resultsVolArray[1]==0 || this.
      resultsCostArray[1]==0)){
124       let alert = this.alertCtrl.create({
125         title: 'Erreur',
126         subTitle: 'Pas de solution possible: vérifiez les paramètres.',
127         buttons: ['OK']
128       });
129       alert.present();
130     }
131   }
132
133   calculOptimalVolume(paramset: ProblemParams, largeurs: Width[], sens: string){
134
135     let optiWidth: number = 0;
136     let optiHeight: number = 0;
137     let optiNumber: number = 0;
```

```
138     let optiVol: number = 0;
139     let cost: number = 0;
140     let entraxe: number = 0;
141     let dimL: number;
142     let dimP: number;
143 
144     if (sens === 'long'){
145       dimL = paramset.dimLP[1];
146       dimP = paramset.dimLP[0];
147     }
148     else {
149       dimL = paramset.dimLP[0];
150       dimP = paramset.dimLP[1];
151     }
152 
153     let eg = paramset.modulE/paramset.modulG;
154     let kfin = paramset.getCombinaisonELSfin()/paramset.getCombinaisonELSinst();
155     let paramC = (24*eg)/25;
156 
157     for (var i=0; i<largeurs.length; i++){
158 
159       let width = largeurs[i];
160       let b = width.b;
161 
162       for (var j=0; j<width.h.length; j++){
163 
164         let height = width.h[j];
165 
166         let yd = (0.75*paramset.gammaM*paramset.getCombinaisonELU())/(this.
    eurocodesService.getkh(height,paramset.cat)*this.eurocodesService.ksys*
    paramset.kmod*paramset.fmk*1000);
167         let kmv = (this.eurocodesService.getkh(height,paramset.cat)*paramset.fmk
    )/(paramset.kcr*paramset.fvk);
168         let kinst = ((4*this.eurocodesService.getkh(height,paramset.cat)*this.
    eurocodesService.ksys*paramset.getCombinaisonELSinst()*paramset.kmod*
    paramset.fmk)/(3*paramset.getCombinaisonELU()*paramset.gammaM*paramset.
    modulE));
169 
170         let entraxeMaxELUf = (b/yd) * Math.pow(height/(100*dimL),2);
171         let entraxeMaxELUv = (b/yd) * height/(kmv*100*dimL)
172         let entraxeMaxELS = (b/yd) / ( Math.max(paramset.ninst,kfin*paramset.
    nfin) * kinst * (5*Math.pow(100*dimL/height,3)/32 + (3*eg*100*dimL)/(height
    *20)) );
173         let entraxeMaxH = Math.min(entraxeMaxELUf,entraxeMaxELUv,entraxeMaxELS,
    paramset.entraxeMax);
174         let number = Math.ceil(dimP*100/entraxeMaxH)+1;
175         let entraxeMaxReal = dimP*100/(number-1);
176 
177         let paramB = (32*b)/(5*Math.max(paramset.ninst,kfin*paramset.nfin)*kinst
    *yd*entraxeMaxReal);
178 
179         let f_ELU_f = Math.sqrt(yd)*Math.sqrt(entraxeMaxReal/b);
180         let f_ELU_v = (yd*kmv*entraxeMaxReal)/b;
181         let f_ELS = 1/ ( Math.cbrt((paramB - Math.sqrt( Math.pow(paramB,2) + 4*
    Math.pow(paramC,3)/27 ))/2) + Math.cbrt((paramB + Math.sqrt( Math.pow(paramB
    ,2) + 4*Math.pow(paramC,3)/27 ))/2) );
182 
183         let hmin = Math.max(f_ELU_f, f_ELU_v, f_ELS)*dimL*100;
184 
185         if(height >= hmin && paramset.hmax >= height && entraxeMaxReal >= b &&
    entraxeMaxReal >= paramset.emin) {
186 
187           let volume = number*height*b*dimL/10000; //[m3]
```

```
188
189           if(volume < optiVol || optiVol==0) {
190
191             optiVol = volume; //[m3]
192             optiNumber = number;
193             optiWidth = b; //[cm]
194             optiHeight = height; //[cm]
195             cost = number*dimL*width.m[j] + number*width.p[j] + 2*number*width.f
      [j]; //[€]
196             entraxe = entraxeMaxReal;
197           }
198         }
199       }
200     }
201     return [sens, optiVol.toFixed(3), cost.toFixed(2), optiWidth, optiHeight,
      optiNumber, entraxe.toFixed(2), 0, 0];
202   }
203
204   calculOptimalCost(paramset: ProblemParams, largeurs: Width[], sens: string){
205
206     let optiWidth: number = 0;
207     let optiHeight: number = 0;
208     let optiNumber: number = 0;
209     let optiCost: number = 0;
210     let vol: number = 0;
211     let entraxe: number = 0;
212     let dimL: number;
213     let dimP: number;
214
215     if (sens === 'long'){
216       dimL = paramset.dimLP[1];
217       dimP = paramset.dimLP[0];
218     }
219     else {
220       dimL = paramset.dimLP[0];
221       dimP = paramset.dimLP[1];
222     }
223
224     let eg = paramset.modulE/paramset.modulG;
225     let kfin = paramset.getCombinaisonELSfin()/paramset.getCombinaisonELSinst();
226     let paramC = (24*eg)/25;
227
228     for (var i=0; i<largeurs.length; i++){
229
230       let width = largeurs[i];
231       let b = width.b;
232
233       for (var j=0; j<width.h.length; j++){
234
235         let height = width.h[j];
236
237         let yd = (0.75*paramset.gammaM*paramset.getCombinaisonELU())/(this.
      eurocodesService.getkh(height,paramset.cat)*this.eurocodesService.ksys*
      paramset.kmod*paramset.fmk*1000);
238         let kmv = (this.eurocodesService.getkh(height,paramset.cat)*paramset.fmk
      )/(paramset.kcr*paramset.fvk);
239         let kinst = ((4*this.eurocodesService.getkh(height,paramset.cat)*this.
      eurocodesService.ksys*paramset.getCombinaisonELSinst()*paramset.kmod*
      paramset.fmk)/(3*paramset.getCombinaisonELU()*paramset.gammaM*paramset.
      modulE));
240
241         let entraxeMaxELUf = (b/yd) * Math.pow(height/(100*dimL),2);
242         let entraxeMaxELUv = (b/yd) * height/(kmv*100*dimL)
```

```
243         let entraxeMaxELS = (b/yd) / ( Math.max(paramset.ninst,kfin*paramset.
    nfin) * kinst * (5*Math.pow(100*dimL/height,3)/32 + (3*eg*100*dimL)/(height
    *20)) );
244         let entraxeMaxH = Math.min(entraxeMaxELUf,entraxeMaxELUv,entraxeMaxELS,
    paramset.entraxeMax);
245         let number = Math.ceil(dimP*100/entraxeMaxH)+1;
246         let entraxeMaxReal = dimP*100/(number-1);
247
248         let paramB = (32*b)/(5*Math.max(paramset.ninst,kfin*paramset.nfin)*kinst
    *yd*entraxeMaxReal);
249
250         let f_ELU_f = Math.sqrt(yd)*Math.sqrt(entraxeMaxReal/b);
251         let f_ELU_v = (yd*kmv*entraxeMaxReal)/b;
252         let f_ELS = 1/ ( Math.cbrt((paramB - Math.sqrt( Math.pow(paramB,2) + 4*
    Math.pow(paramC,3)/27 ))/2) + Math.cbrt((paramB + Math.sqrt( Math.pow(paramB
    ,2) + 4*Math.pow(paramC,3)/27 ))/2) );
253
254         let hmin = Math.max(f_ELU_f, f_ELU_v, f_ELS)*dimL*100;
255
256         if(height >= hmin && paramset.hmax >= height && entraxeMaxReal >= b &&
    entraxeMaxReal >= paramset.emin) {
257
258           let cost = number*dimL*width.m[j] + number*width.p[j] + 2*number*width
    .f[j]; //[€]
259
260           if(cost < optiCost || optiCost==0) {
261             optiCost = cost; //[€]
262             optiNumber = number;
263             optiWidth = b; //[cm]
264             optiHeight = height; //[cm]
265             vol = optiNumber*optiHeight*optiWidth*dimL/10000; //[m3]
266             entraxe = entraxeMaxReal;
267           }
268         }
269       }
270     }
271
272     return [sens, vol.toFixed(3), optiCost.toFixed(2), optiWidth, optiHeight,
    optiNumber, entraxe.toFixed(2), 0, 0];
273   }
274
275   calculOptimalVolume_Inter(paramset: ProblemParams, largeurs: Width[],
    largeurs_inter: Width[], sens_inter: string){
276
277     // CALCUL DE LA POUTRE INTERMEDIAIRE - LA MOINS VOLUMINEUSE
278
279     let optiWidth_inter: number = 0;
280     let optiHeight_inter: number = 0;
281     let optiVol_inter: number = 0;
282     let cost_inter: number = 0;
283     let entraxe_inter: number;
284     let dimL: number;
285     let dimP: number;
286
287     if (sens_inter === 'long'){
288       dimL = paramset.dimLP[1];
289       dimP = paramset.dimLP[0];
290       entraxe_inter = (dimP*100)/2;
291     }
292     else {
293       dimL = paramset.dimLP[0];
294       dimP = paramset.dimLP[1];
295       entraxe_inter = (dimP*100)/2;
```

```
296     }
297
298     let eg = paramset.modulE_inter/paramset.modulG_inter;
299     let kfin = paramset.getCombinaisonELSfin()/paramset.getCombinaisonELSinst();
300     let paramC = (24*eg)/25;
301
302     for (var i=0; i<largeurs_inter.length; i++){
303
304       let width_inter = largeurs_inter[i];
305       let b_inter = width_inter.b;
306
307       for (var j=0; j<width_inter.h.length; j++){
308
309         let height_inter = width_inter.h[j];
310
311         let yd = (0.75*paramset.gammaM*paramset.getCombinaisonELU())/(this.
    eurocodesService.getkh(height_inter,paramset.cat_inter)*this.
    eurocodesService.ksys*paramset.kmod*paramset.fmk_inter*1000);
312         let kmv = (this.eurocodesService.getkh(height_inter,paramset.cat_inter)*
    paramset.fmk_inter)/(paramset.kcr*paramset.fvk_inter);
313         let kinst = ((4*this.eurocodesService.getkh(height_inter,paramset.
    cat_inter)*this.eurocodesService.ksys*paramset.getCombinaisonELSinst()*
    paramset.kmod*paramset.fmk_inter)/(3*paramset.getCombinaisonELU()*paramset.
    gammaM*paramset.modulE_inter));
314
315         let paramB = (32*b_inter)/(5*Math.max(paramset.ninst,kfin*paramset.nfin)
    *kinst*yd*entraxe_inter);
316
317         let f_ELU_f = Math.sqrt(yd)*Math.sqrt(entraxe_inter/b_inter);
318         let f_ELU_v = (yd*kmv*entraxe_inter)/b_inter;
319         let f_ELS = 1/ ( Math.cbrt((paramB - Math.sqrt( Math.pow(paramB,2) + 4*
    Math.pow(paramC,3)/27 ))/2) + Math.cbrt((paramB + Math.sqrt( Math.pow(paramB
    ,2) + 4*Math.pow(paramC,3)/27 ))/2) );
320
321         let hmin_inter = Math.max(f_ELU_f, f_ELU_v, f_ELS)*dimL*100;
322
323         if((height_inter >= hmin_inter) && (paramset.hmax >= height_inter)) {
324
325           let volume_inter = height_inter*b_inter*dimL/10000; //[m3]
326
327           if(volume_inter < optiVol_inter || optiVol_inter==0) {
328
329             optiVol_inter = volume_inter; //[m3]
330             optiWidth_inter = b_inter; //[cm]
331             optiHeight_inter = height_inter; //[cm]
332             cost_inter = dimL*width_inter.m[j] + width_inter.p[j] + 2*
    width_inter.f[j]; //[€]
333           }
334         }
335       }
336     }
337
338     if (optiVol_inter != 0) {
339       // CALCUL DU DEMI-GITAGE - LE MOINS VOLUMINEUX
340
341       let dimLbis = (dimP-optiWidth_inter/100)/2;
342       let dimPbis = dimL;
343
344       let bis_result: (string|number)[];
345       let paramset_bis = this.clone(paramset);
346       bis_result = this.calculOptimalVolumeBIS(paramset_bis, largeurs, 'court',
    dimLbis, dimPbis);
347
```

```
348       return [sens_inter, (optiVol_inter+2*Number(bis_result[1])).toFixed(3), (
          cost_inter+2*Number(bis_result[2])).toFixed(2), Number(bis_result[3]),
          Number(bis_result[4]), Number(bis_result[5]), Number(bis_result[6]),
          optiWidth_inter, optiHeight_inter];
349     }
350     else {
351       return [0, 0, 0, 0, 0, 0, 0, 0, 0];
352     }
353   }
354
355   calculOptimalVolumeBIS(paramset_bis: ProblemParams, largeurs: Width[], sens:
        string, dimLbis: number, dimPbis: number){
356     paramset_bis.dimLP[0] = dimLbis;
357     paramset_bis.dimLP[1] = dimPbis;
358     return this.calculOptimalVolume(paramset_bis, largeurs, sens);
359   }
360
361   calculOptimalCost_Inter(paramset: ProblemParams, largeurs: Width[],
        largeurs_inter: Width[], sens_inter: string){
362
363     // CALCUL DE LA POUTRE INTERMEDIAIRE - LA MOINS CHERE
364
365     let optiWidth_inter: number = 0;
366     let optiHeight_inter: number = 0;
367     let optiCost_inter: number = 0;
368     let vol_inter: number = 0;
369     let entraxe_inter: number;
370     let dimL: number;
371     let dimP: number;
372
373     if (sens_inter === 'long'){
374       dimL = paramset.dimLP[1];
375       dimP = paramset.dimLP[0];
376       entraxe_inter = (dimP*100)/2;
377     }
378     else {
379       dimL = paramset.dimLP[0];
380       dimP = paramset.dimLP[1];
381       entraxe_inter = (dimP*100)/2;
382     }
383
384     let eg = paramset.modulE_inter/paramset.modulG_inter;
385     let kfin = paramset.getCombinaisonELSfin()/paramset.getCombinaisonELSinst();
386     let paramC = (24*eg)/25;
387
388     for (var i=0; i<largeurs_inter.length; i++){
389
390       let width_inter = largeurs_inter[i];
391       let b_inter = width_inter.b;
392
393       for (var j=0; j<width_inter.h.length; j++){
394
395         let height_inter = width_inter.h[j];
396
397         let yd = (0.75*paramset.gammaM*paramset.getCombinaisonELU())/(this.
          eurocodesService.getkh(height_inter,paramset.cat_inter)*this.
          eurocodesService.ksys*paramset.kmod*paramset.fmk_inter*1000);
398         let kmv = (this.eurocodesService.getkh(height_inter,paramset.cat_inter)*
          paramset.fmk_inter)/(paramset.kcr*paramset.fvk_inter);
399         let kinst = ((4*this.eurocodesService.getkh(height_inter,paramset.
          cat_inter)*this.eurocodesService.ksys*paramset.getCombinaisonELSinst()*
          paramset.kmod*paramset.fmk_inter)/(3*paramset.getCombinaisonELU()*paramset.
          gammaM*paramset.modulE_inter));
```

```
400
401         let paramB = (32*b_inter)/(5*Math.max(paramset.ninst,kfin*paramset.nfin)
      *kinst*yd*entraxe_inter);
402
403         let f_ELU_f = Math.sqrt(yd)*Math.sqrt(entraxe_inter/b_inter);
404         let f_ELU_v = (yd*kmv*entraxe_inter)/b_inter;
405         let f_ELS = 1/ ( Math.cbrt((paramB - Math.sqrt( Math.pow(paramB,2) + 4*
      Math.pow(paramC,3)/27 ))/2) + Math.cbrt((paramB + Math.sqrt( Math.pow(paramB
      ,2) + 4*Math.pow(paramC,3)/27 ))/2) );
406
407         let hmin_inter = Math.max(f_ELU_f, f_ELU_v, f_ELS)*dimL*100;
408
409         if(height_inter >= hmin_inter && paramset.hmax >= height_inter) {
410
411           let cost_inter = dimL*width_inter.m[j] + width_inter.p[j] + 2*
      width_inter.f[j]; //[€]
412
413           if(cost_inter < optiCost_inter || optiCost_inter==0) {
414             optiCost_inter = cost_inter; //[€]
415             optiWidth_inter = b_inter; //[cm]
416             optiHeight_inter = height_inter; //[cm]
417             vol_inter = height_inter*b_inter*dimL/10000; //[m3]
418           }
419         }
420       }
421     }
422
423     if (vol_inter != 0) {
424       // CALCUL DU DEMI-GITAGE - LE MOINS CHER
425
426       let dimLbis = (dimP-optiWidth_inter/100)/2;
427       let dimPbis = dimL;
428
429       let bis_result: (string|number)[];
430       let paramset_bis = this.clone(paramset);
431       bis_result = this.calculOptimalCostBIS(paramset_bis, largeurs, 'court',
      dimLbis, dimPbis);
432
433       return [sens_inter, (vol_inter+2*Number(bis_result[1])).toFixed(3), (
      optiCost_inter+2*Number(bis_result[2])).toFixed(2), Number(bis_result[3]),
      Number(bis_result[4]), Number(bis_result[5]), Number(bis_result[6]),
      optiWidth_inter, optiHeight_inter];
434     }
435     else {
436       return [0, 0, 0, 0, 0, 0, 0, 0, 0];
437     }
438   }
439
440
441   calculOptimalCostBIS(paramset_bis: ProblemParams, largeurs: Width[], sens:
      string, dimLbis: number, dimPbis: number){
442     paramset_bis.dimLP[0] = dimLbis;
443     paramset_bis.dimLP[1] = dimPbis;
444     return this.calculOptimalCost(paramset_bis, largeurs, sens);
445   }
446
447
448   indexOfMinimum(arr: Number[]) {
449     let index_min = 0;
450       for (var i = 1; i < arr.length; i++) {
451       if (arr[i] < arr[index_min] && arr[i] != 0) {index_min = i;}
452     }
453     return index_min;
```

```
  }

  clone(paramset){
      if(paramset == null || typeof(paramset) != 'object') {return paramset;}
      var param_clone = new paramset.constructor();
      for(var key in paramset){param_clone[key] = this.clone(paramset[key]);}
      return param_clone;
  }

  showHideVolume() {
    if (this.volumeSelected === true) {this.volumeSelected = false;}
    else if (this.volumeSelected === false) {this.volumeSelected = true;}
  }

  showHideCost() {
    if (this.costSelected === true) {this.costSelected = false;}
    else if (this.costSelected === false) {this.costSelected = true;}
  }

  showHideParam() {
    if (this.paramSelected === true) {this.paramSelected = false;}
    else if (this.paramSelected === false) {this.paramSelected = true;}
  }

  isSensCourt(resultArray: (string|number)[]) {
    return resultArray[0]==='court';
  }
  isSensLong(resultArray: (string|number)[]) {
    return resultArray[0]==='long';
  }

}
```

Code B.26 – solution.ts

```
<ion-header charset=UTF-8>

  <ion-navbar>
    <ion-title>Solutions</ion-title>
  </ion-navbar>

</ion-header>

<ion-content padding>

  <ion-list text-wrap>


      <button ion-item (click)="showHideVolume()" text-wrap>
        Solution optimale en termes de volume
      </button>

      <ion-item *ngIf="volumeSelected">

        <div text-center *ngIf="!inter_beam_display_V">
          <h3>Disposition:</h3>
          <p *ngIf="resultsVolArray[1]==0 && !inter_beam_display_V">
            Portée des poutres: 0 [m]
          </p>
          <p *ngIf="isSensCourt(resultsVolArray) && resultsVolArray[1]!=0 && !
    inter_beam_display_V">
            Portée des poutres: {{parameters.dimLP[0]}} [m]
          </p>
```

```
28          <p *ngIf="isSensLong(resultsVolArray) && resultsVolArray[1]!=0 && !
    inter_beam_display_V">
29            Portée des poutres: {{parameters.dimLP[1]}} [m]
30          </p>
31        </div>
32        <div text-center *ngIf="inter_beam_display_V">
33          <h3>Disposition: poutre intermédiaire au centre du gîtage </h3>
34          <p *ngIf="isSensCourt(resultsVolArray) && inter_beam_display_V">
35            Portée de la poutre intermédiaire: {{parameters.dimLP[0]}} [m]
36          </p>
37          <p *ngIf="isSensLong(resultsVolArray) && inter_beam_display_V">
38            Portée de la poutre intermédiaire: {{parameters.dimLP[1]}} [m]
39          </p>
40          <p> Autres éléments placés perpendiculairement à la poutre intermé
    diaire </p>
41        </div>
42
43        <div text-center *ngIf="!inter_beam_display_V">
44            <h3>Section des éléments:</h3>
45            <p> b = {{resultsVolArray[3]}} [cm] x h = {{resultsVolArray[4]}}
    [cm] </p>
46        </div>
47        <div text-center *ngIf="inter_beam_display_V">
48            <h3>Section de la poutre intermédiaire:</h3>
49            <p> b = {{resultsVolArray[7]}} [cm] x h = {{resultsVolArray[8]}}
    [cm] </p>
50            <h3>Section des éléments perpendiculaires:</h3>
51            <p> b = {{resultsVolArray[3]}} [cm] x h = {{resultsVolArray[4]}}
    [cm] </p>
52        </div>
53
54        <div text-center *ngIf="!inter_beam_display_V">
55          <h3>Nombre d'éléments (n) et entraxe (e):</h3>
56          <p> n = {{resultsVolArray[5]}} </p>
57          <p> e = {{resultsVolArray[6]}} [cm] </p>
58        </div>
59        <div text-center *ngIf="inter_beam_display_V">
60          <h3>Nombre d'éléments transversaux (n) et entraxe (e):</h3>
61          <p> n = 2 x {{resultsVolArray[5]}} = {{2*resultsVolArray[5]}} </p>
62          <p> e = {{resultsVolArray[6]}} [cm] </p>
63        </div>
64
65        <div text-center>
66          <h3>Volume total de bois:</h3>
67          <p> {{resultsVolArray[1]}} [m<sup>3</sup>] </p>
68        </div>
69        <div text-center padding-bottom>
70          <h3>Coût total:</h3>
71          <p> {{resultsVolArray[2]}} [€] </p>
72        </div>
73      </ion-item>
74
75
76      <button ion-item (click)="showHideCost()" text-wrap>
77        Solution optimale en termes de coût
78      </button>
79
80      <ion-item *ngIf="costSelected">
81
82        <div text-center *ngIf="!inter_beam_display_C">
83          <h3>Disposition:</h3>
84          <p *ngIf="resultsCostArray[1]==0 && !inter_beam_display_C">
85            Portée des poutres: 0 [m]
```

```
86            </p>
87            <p *ngIf="isSensCourt(resultsCostArray) && resultsCostArray[1]!=0 && !
      inter_beam_display_C">
88              Portée des poutres: {{parameters.dimLP[0]}} [m]
89            </p>
90            <p *ngIf="isSensLong(resultsCostArray) && resultsCostArray[1]!=0 && !
      inter_beam_display_C">
91              Portée des poutres: {{parameters.dimLP[1]}} [m]
92            </p>
93          </div>
94          <div text-center *ngIf="inter_beam_display">
95            <h3>Disposition: poutre intermédiaire au centre du gîtage </h3>
96            <p *ngIf="isSensCourt(resultsCostArray) && inter_beam_display_C">
97              Portée de la poutre intermédiaire: {{parameters.dimLP[0]}} [m]
98            </p>
99            <p *ngIf="isSensLong(resultsCostArray) && inter_beam_display_C">
100             Portée de la poutre intermédiaire: {{parameters.dimLP[1]}} [m]
101           </p>
102           <p> Autres éléments placés perpendiculairement à la poutre intermé
      diaire </p>
103         </div>
104
105         <div text-center *ngIf="!inter_beam_display_C">
106             <h3>Section des éléments:</h3>
107             <p> b = {{resultsCostArray[3]}} [cm] x h = {{resultsCostArray[4]}}
      [cm] </p>
108         </div>
109         <div text-center *ngIf="inter_beam_display_C">
110             <h3>Section de la poutre intermédiaire:</h3>
111             <p> b = {{resultsCostArray[7]}} [cm] x h = {{resultsCostArray[8]}}
      [cm] </p>
112             <h3>Section des éléments perpendiculaires:</h3>
113             <p> b = {{resultsCostArray[3]}} [cm] x h = {{resultsCostArray[4]}}
      [cm] </p>
114         </div>
115
116         <div text-center *ngIf="!inter_beam_display_C">
117           <h3>Nombre d'éléments (n) et entraxe (e):</h3>
118           <p> n = {{resultsCostArray[5]}} </p>
119           <p> e = {{resultsCostArray[6]}} [cm] </p>
120         </div>
121         <div text-center *ngIf="inter_beam_display_C">
122           <h3>Nombre d'éléments transversaux (n) et entraxe (e):</h3>
123           <p> n = 2 x {{resultsCostArray[5]}} = {{2*resultsCostArray[5]}} </p>
124           <p> e = {{resultsCostArray[6]}} [cm] </p>
125         </div>
126
127         <div text-center>
128           <h3>Volume total de bois:</h3>
129           <p> {{resultsCostArray[1]}} [m<sup>3</sup>] </p>
130         </div>
131         <div text-center padding-bottom>
132           <h3>Coût total:</h3>
133           <p> {{resultsCostArray[2]}} [€] </p>
134         </div>
135         </ion-item>
136
137
138       <button ion-item (click)="showHideParam()" text-wrap>
139         Paramètres du problème
140       </button>
141       <ion-item *ngIf="paramSelected">
142         <div text-center>
```

```
143             <h3>Dimensions du gîtage:</h3>
144             <p>
145               {{parameters.dimLP[0]}} [m] x {{parameters.dimLP[1]}} [m]
146             </p>
147         </div>
148
149         <div text-center *ngIf="inter_beam">
150             <h3>Propriétés du bois de la poutre intermédiaire <span class="
    uppercase">({{parameters.classeBois_inter}})</span>:</h3>
151             <p>Résistance à la flexion: {{parameters.fmk_inter}} [MPa]</p>
152             <p>Résistance au cisaillement: {{parameters.fvk_inter}} [MPa]</p>
153             <p>Module d'élasticité: {{parameters.modulE_inter}} [MPa]</p>
154             <p>Module de cisaillement: {{parameters.modulG_inter}} [MPa]</p>
155             <h3>Propriétés du bois des éléments transversaux <span class="
    uppercase">({{parameters.classeBois}})</span>:</h3>
156             <p>Résistance à la flexion: {{parameters.fmk}} [MPa]</p>
157             <p>Résistance au cisaillement: {{parameters.fvk}} [MPa]</p>
158             <p>Module d'élasticité: {{parameters.modulE}} [MPa]</p>
159             <p>Module de cisaillement: {{parameters.modulG}} [MPa]</p>
160         </div>
161         <div text-center *ngIf="!inter_beam">
162             <h3>Propriétés du bois <span class="uppercase">({{
    parameters.classeBois}})</span>:</h3>
163             <p>Résistance à la flexion: {{parameters.fmk}} [MPa]</p>
164             <p>Résistance au cisaillement: {{parameters.fvk}} [MPa]</p>
165             <p>Module d'élasticité: {{parameters.modulE}} [MPa]</p>
166             <p>Module de cisaillement: {{parameters.modulG}} [MPa]</p>
167         </div>
168
169         <div text-center *ngIf="inter_beam">
170             <h3>Plage de sections utilisée pour la poutre intermédiaire:</h3>
171             <p> {{range_inter_name}} </p>
172             <h3>Plage de sections utilisée pour les éléments transversaux:</h3>
173             <p> {{range.name}} </p>
174         </div>
175         <div text-center *ngIf="!inter_beam">
176             <h3>Plage de sections utilisée pour les éléments:</h3>
177             <p> {{range.name}} </p>
178         </div>
179
180         <div text-center>
181             <h3>Charges:</h3>
182             <p>Charge permanente: {{parameters.chargePermG}} [kN]</p>
183             <p>Charge variable: {{parameters.chargeVarQ}} [kN]</p>
184         </div>
185         <div text-center>
186             <h3>Coefficients de pondération de charges:</h3>
187             <p>
188               &gamma;<sub>G</sub>={{parameters.gammaG}}, &gamma;<sub>Q</sub>={{
    parameters.gammaQ}}, &psi;<sub>0</sub>={{parameters.psi0}}, &psi;<sub>1</sub
    >={{parameters.psi1}}, &psi;<sub>2</sub>={{parameters.psi2}}
189             </p>
190         </div>
191         <div text-center>
192             <h3>Coefficients de flèche:</h3>
193             <p>
194               &eta;<sub>inst</sub>={{parameters.ninst}}, &eta;<sub>fin</sub>={{
    parameters.nfin}}
195             </p>
196         </div>
197         <div text-center>
198             <h3>Autres coefficients:</h3>
199             <p>
```

```
200                 &gamma;<sub>M</sub>={{parameters.gammaM}}, k<sub>def</sub>={{
    parameters.kdef}}, k<sub>mod</sub>={{parameters.kmod}}, k<sub>cr</sub>={{
    parameters.kcr}}, k<sub>sys</sub>={{parameters.ksys}}
201             </p>
202         </div>
203
204         <div text-center *ngIf="if_h_max">
205             <h3>Hauteur maximale des éléments:</h3>
206             <p> {{paramters.hmax}} </p>
207         </div>
208
209         <div text-center *ngIf="if_sensUnic">
210             <h3>Sens de portée imposé:</h3>
211             <p> {{paramters.sensUnic}} </p>
212         </div>
213
214       </ion-item>
215
216   </ion-list>
217
218 </ion-content>
```

Code B.27 – `solution.html`

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**